1 2 9 0

UNIVERSIDADE D
COIMBRA

Nuno Miguel Gil Fonseca

Contributos para a monitorização do desempenho de estudantes de programação

Tese no âmbito do Programa de Doutoramento em Ciências e Tecnologias da Informação orientada pelo Professor Doutor António José Mendes e pelo Professor Doutor Luís Macedo e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Dezembro de 2021

Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Departamento de Engenharia Informática.

CONTRIBUTOS PARA A MONITORIZAÇÃO DO DESEMPENHO DE ESTUDANTES DE PROGRAMAÇÃO

Nuno Miguel Gil Fonseca

Tese no âmbito do Programa de Doutoramento em Ciências e Tecnologias da Informação orientada pelo Professor Doutor António José Mendes e pelo Professor Doutor Luís Macedo e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Dezembro de 2021

À minha família e amigos.

Agradecimentos

Apresenta-se neste documento o resultado do trabalho desenvolvido ao longo dos últimos anos e que não teria sido possível sem a colaboração de diversas pessoas e entidades a quem gostaria de expressar os meus sinceros agradecimentos.

Gostaria, pois, de começar por agradecer aos Professores António José Mendes e Luís Macedo, pela forma como orientaram o meu trabalho, por todas as vezes que me forçaram a recentrar as atenções no mais importante, pelas palavras e gestos de encorajamento que foram expressando ao longo de todo o processo e acima de tudo por sempre acreditarem em mim e no meu trabalho.

Dada a natureza deste trabalho, nada teria sido possível sem a colaboração dos estudantes inscritos nas várias unidades curriculares em que o sistema *CodeInsights* foi testado. De igual modo, gostaria de agradecer aos vários professores que utilizaram o sistema nas suas unidades curriculares e cujo valioso *feedback* tanto contribuiu para as melhorias que o sistema sofreu. Gostaria de agradecer com particular destaque à Professora Maria José Marcelino e ao Professor Francisco Carlos Afonso.

Gostaria também de agradecer à Escola Superior de Tecnologia e Gestão de Oliveira do Hospital, nomeadamente ao antigo presidente, o Arquiteto Carlos Veiga, e à atual presidente, Doutora Vera Cunha pela disponibilização da infraestrutura informática na qual as várias instalações do sistema *CodeInsights* foram implementadas e ao Eng. António Paulino e ao Luís Esteves pelo suporte técnico fornecido.

Não poderia também de deixar de agradecer aos membros do laboratório de investigação no âmbito do qual este trabalho foi desenvolvido, o *Educational Technology Lab / Cognitive and Media Systems* do Centro de Informática e Sistemas da Universidade de Coimbra.

Pelo facto de uma parte importante do trabalho de investigação consistir na sua disseminação em conferências, gostaria de agradecer às sucessivas direções do Centro de Informática e Sistemas da Universidade de Coimbra, aos líderes do grupo *Cognitive and Media Systems,* assim como ao Dr. Jorge Ávila pelo apoio administrativo prestado.

Poucas coisas na vida se fazem com sucesso sem amigos, como tal, não poderia deixar de agradecer aos colegas e amigos Luís Veloso, Marco Veloso e Ricardo Rodrigues, que estiveram sempre disponíveis e presentes ao longo de todo percurso.

Gostaria ainda de agradecer aos meus pais, irmãos, cunhados e sogros pela preocupação, disponibilidade e por todo o apoio prestado ao longo destes anos.

Para terminar, um agradecimento mais especial a quem acabou por sentir mais diretamente o impacto da minha dedicação a este trabalho, a minha esposa Lara e as minhas filhas Bianca e Dalila.

A todos os mencionados e a quem, por lapso, possa não ter incluído, muito obrigado!

Resumo

Diariamente, muitos estudantes são confrontados com as dificuldades inerentes ao processo de aprendizagem de programação de aplicações informáticas. Estudos demonstram que um acompanhamento permanente do progresso dos estudantes por parte dos professores é determinante para o seu sucesso. Assim, com esta investigação pretendemos contribuir para melhorar as condições de aprendizagem inicial da programação através da conceção, implementação e avaliação de uma aplicação informática (*CodeInsights*), que recolhe *snapshots* do código desenvolvido pelos estudantes aquando da resolução de exercícios propostos pelos professores e, após o seu processamento, disponibiliza aos professores diversas informações relativas ao desempenho e progresso dos estudantes, de modo a facilitar a identificação, em tempo útil, dos pontos-chave a serem alvo de intervenção por parte dos professores.

O trabalho foi desenvolvido utilizando a metodologia *Design Based Research* (DBR), o que em termos gerais significa que o sistema *CodeInsights* foi desenvolvido de forma iterativa ao longo de dez ciclos, tendo sido testado em contexto real em dezoito experiências, nas quais estiveram envolvidos catorze professores e aproximadamente quinhentos estudantes. O *feedback* de professores e estudantes recolhidos em cada experiência foi utilizado para guiar o rumo a adotar no ciclo seguinte.

As experiências realizadas indicam que se conseguiu um importante contributo para melhorar o processo de ensino/aprendizagem de programação, quer do ponto de vista dos professores, quer do ponto de vista dos estudantes. Por um lado, a disponibilização do próprio sistema desenvolvido permite que diversos professores e estudantes possam usufruir dos seus benefícios para a melhoria do nível ensino/aprendizagem de programação. Por outro lado, a própria forma como o sistema foi desenvolvido e as questões que foram sendo levantadas (e respondidas) ao longo do processo constituem também um importante contributo para esta área de investigação.

Palavras-chave

ensino de programação, monitorização do desempenho dos estudantes, ensino diferenciado

Abstract

Every day, several thousands of students are confronted with the difficulties inherent in the process of learning to program software applications. Studies show that a permanent monitoring of students' progress by teachers is crucial to their success. Thus, with this research we intend to contribute to improve the conditions of initial programming learning through the design, implementation and evaluation of a software application (*CodeInsights*), which collects snapshots of the code developed by students when solving exercises proposed by teachers and, after processing them, provides teachers with several information regarding the performance and progress of students in order to facilitate the timely identification of key points to be targeted for intervention by teachers.

The work was developed using the Design Based Research (DBR) methodology, which in general terms means that the *CodeInsights* system was developed iteratively over ten cycles, and tested in real classroom contexts in eighteen experiments, in which fourteen teachers and approximately five hundred students were involved. The feedback from teachers and students collected in each experiment was used to guide the course of action for the next cycle.

We believe we have managed to make an important contribution to improving the teaching/learning process of programming, both from the teachers' and students' point of view. On one hand, the availability of the developed system itself allows several teachers and students to enjoy its benefits for the improvement of the teaching/learning level of programming. On the other hand, it is our understanding that the very way the system was developed and the questions that were raised (and answered) along the process are also an important contribution to this research area.

Keywords

teaching programming, student performance monitoring, differentiated instruction

Índice

RESUMO ..... **I**

ABSTRACT ..... **III**

ÍNDICE ..... **V**

ÍNDICE DE FIGURAS ..... **XI**

1.INTRODUÇÃO **1**

1.1. PROBLEMAS E DIFICULDADES DE APRENDER PROGRAMAÇÃO 6

1.2. A IMPORTÂNCIA DOS PROFESSORES NO PROCESSO DE ENSINO/APRENDIZAGEM DE PROGRAMAÇÃO 11

1.3. PROBLEMAS E DIFICULDADES DE ENSINAR A PROGRAMAR 13

1.4. O SISTEMA *CODEINSIGHTS* 16

1.5. PRINCIPAIS CONTRIBUIÇÕES 18

1.6. ORGANIZAÇÃO DO DOCUMENTO 19

2.PROCESSO DE INVESTIGAÇÃO **21**

2.1 QUESTÕES DE INVESTIGAÇÃO 21

2.2. METODOLOGIA DE INVESTIGAÇÃO 22

3.ESTADO DA ARTE **25**

3.1. TUTORES INTELIGENTES 26

3.2. *INTELIGENT TEACHING ASSISTANTS* 29

3.3. *AUTO-GRADERS* 32

3.4. *LEARNING ANALYTICS* 48

3.5. SUMÁRIO 52

4. DESENVOLVIMENTO **55**

4.1. ITERAÇÃO ZERO 56

4.1.1. Objetivos *56*

4.1.2. Trabalho desenvolvido *56*

4.1.2.1. Arquitetura do sistema ... *56*

4.1.2.2. Base de dados de exercícios ... *57*

4.1.2.3. Processo de envio do código fonte para o servidor ... *58*

4.1.2.4. Análise do código fonte ... *60*

4.1.2.5. Modelo do conhecimento do estudante ... *61*

4.1.2.6. Aplicação de monitorização do desempenho dos estudantes ... *65*

4.1.2.7. Implementação do sistema ... *66*

4.1.3. Sumário da iteração zero ... *69*

4.2. ITERAÇÃO UM ... 70

4.2.1. Objetivos ... *70*

4.2.2. Caracterização do contexto de testes ... *71*

4.2.3. Trabalho desenvolvido ... *71*

4.2.3.1. Escolha do ambiente de desenvolvimento ... *71*

4.2.3.2. Desenvolvimento do plug-in para a plataforma IntelliJ ... *72*

4.2.3.3. Receção e processamento do código fonte ... *75*

4.2.3.4. Construção e atualização do modelo do estudante ... *77*

4.2.3.5. Ferramenta de monitorização ... *78*

4.2.4. Avaliação do sistema ... *83*

4.2.5. Sumário da iteração um ... *93*

4.3. ITERAÇÃO DOIS ... 94

4.3.1. Objetivos ... *94*

4.3.2. Caracterização do contexto ... *94*

4.3.3. Trabalho desenvolvido ... *95*

4.3.3.1. Desenvolvimento dos plug-ins ... *95*

4.3.3.2. Receção e processamento do código fonte ... *96*

4.3.4.3. Arquitetura do sistema ... *96*

4.3.4.4. Ferramenta de monitorização ... *97*

4.3.4.5. Notificações automáticas ... *103*

4.3.5. Avaliação do sistema ... *104*

4.3.5.1. Fundamentos de Programação (LEI - ESTGOH) *104*

4.3.5.2. Programação (LEI - ESTGOH) *105*

4.3.5.3. Introdução à Programação em Java (Acertar o Rumo - DEI - FCTUC) *107*

4.3.6. Sumário da iteração dois *111*

4.4 ITERAÇÃO TRÊS 112

4.4.1. Objetivos *112*

4.4.2. Descrição do contexto *112*

4.4.3. Trabalho desenvolvido *113*

4.4.3.1. Desenvolvimento dos plug-ins *113*

4.4.3.2. Receção e processamento do código fonte *114*

4.4.3.3. Ritmos de trabalho/aprendizagem *115*

4.4.3.4. Ferramenta de monitorização *126*

4.4.4. Avaliação do sistema *129*

4.4.4.1. Programação (LEGI-FCT-UC) *129*

4.4.4.2. Programação para a Internet II (ESTGOH - IPC) *131*

4.4.5. Sumário da iteração três *135*

4.5. ITERAÇÃO QUATRO 136

4.5.1. Objetivos *136*

4.5.2. Descrição do contexto *137*

4.5.3. Trabalho desenvolvido *137*

4.5.3.1. Desenvolvimento do plug-in *137*

4.5.3.2. Receção e processamento do código fonte *139*

4.5.3.3. Base de dados de exercícios *146*

4.5.3.4. Ferramenta de monitorização - professores *147*

4.5.3.5. Ferramenta de monitorização - estudantes *153*

4.5.3.6. Sistema de notificações automáticas *155*

4.5.4. Avaliação do sistema *169*

4.5.4.1. Programa Acertar o Rumo (DEI - FCTUC) *169*

4.5.6.2. Fundamentos de Programação (LEI - ESTGOH) *175*

4.5.6.3. Programação (LEI - ESTGOH) *179*

4.5.6.4. Programação para a Internet I (LEI - ESTGOH) *182*

4.5.7. Sumário da iteração quatro *182*

4.6. ITERAÇÃO CINCO 184

4.6.1. Objetivos *184*

4.6.2. Descrição do contexto *184*

4.6.3. Trabalho desenvolvido *185*

4.6.3.1. Desenvolvimento do plug-in *185*

4.6.3.2. Receção e processamento dos snapshots *186*

4.6.3.3. Deteção de plágio em tempo real *187*

4.6.3.4. Arquitetura geral do sistema *194*

4.6.3.5. Ferramenta de monitorização *196*

4.6.4. Avaliação *198*

4.6.5. Sumário da Iteração cinco *202*

4.7. ITERAÇÃO SEIS 203

4.7.1. Objetivos *203*

4.7.2. Descrição do contexto *203*

4.7.3. Avaliação do sistema *204*

4.7.3.1. Programa Acertar o Rumo (DEI-FCTUC) *205*

4.7.3.2. Fundamentos de Programação (LEI-ESTGOH) *211*

4.7.3.3. Programação para a Internet II (LEI-ESTGOH) *214*

4.7.5. Sumário da iteração seis *222*

4.8. ITERAÇÃO SETE 225

4.8.1. Objetivos *225*

4.8.2. Descrição do contexto *225*

4.8.3. Trabalho desenvolvido *226*

4.8.3.1. Desenvolvimento dos plug-ins *226*

4.8.3.2. Receção e processamento do código fonte *228*

4.8.4. Avaliação do sistema *236*

4.8.4.1. Introdução à Programação em Java (Acertar o Rumo) *236*

4.8.4.2. Fundamentos de Programação (LEI - ESTGOH) *246*

4.8.5. Sumário da iteração sete *255*

4.9. ITERAÇÃO OITO 256

4.9.1. Objetivos *256*

4.9.2. Descrição do contexto *257*

4.9.3. Avaliação do sistema *257*

4.9.4. Sumário da iteração oito *261*

4.10. ITERAÇÃO NOVE 263

4.10.1. Objetivos *263*

4.10.2. Descrição do contexto *263*

4.10.3. Trabalho desenvolvido *264*

4.10.3.1. Desenvolvimento dos plug-ins *264*

4.10.3.2. Ferramenta de processamento do código *265*

4.10.3.3. Ferramenta de monitorização - professores *265*

4.10.4. Avaliação do sistema *269*

4.10.5. Sumário da iteração nove *275*

5.CONCLUSÕES 277

REFERÊNCIAS 285

ANEXOS 301

ANEXO I - GUIA DE CONSULTA RÁPIDA (*CHEAT SHEET*) 303

ANEXO II - EXEMPLO DE FICHA PRÁTICA TAL COMO ERA APRESENTADA AOS ALUNOS ANTES DA UTILIZAÇÃO DO SISTEMA *CODEINSIGHTS* 307

ANEXO III - EXEMPLO DE FICHA PRÁTICA MODIFICADA COM INCLUSÃO DE DETALHES PARA OS CASOS DE TESTE 313

ANEXO IV - ESTRUTURA BASE DA ENTREVISTA REALIZADA AOS PROFESSORES (IPJ - ACERTAR O RUMO - 2018/2019) 321

ANEXO V - MANUAL DE INSTALAÇÃO E UTILIZAÇÃO (IPJ - ACERTAR O RUMO - 2018/2019) 325

ANEXO VI - EXEMPLO CONSENTIMENTO INFORMADO 333

ANEXO VII - ESTRUTURA DO QUESTIONÁRIO FEITO AOS ESTUDANTES (IPJ - ACERTAR O RUMO - 2018/2019) 337

ANEXO VIII - ESTRUTURA DO QUESTIONÁRIO FEITO AOS ESTUDANTES (PI2/DW2 - ESTGOH - 2018/2019) 343

ANEXO IX - MANUAL DE INSTALAÇÃO E UTILIZAÇÃO (IPJ - ACERTAR O RUMO - 2019/2020) 351

ANEXO X - ESTRUTURA BASE DA ENTREVISTA REALIZADA AOS ESTUDANTES (IPJ - ACERTAR O RUMO - 2019/2020) 365

ANEXO XI - ESTRUTURA DO QUESTIONÁRIO FEITO AOS PROFESSORES (IPJ - ACERTAR O RUMO - 2019/2020) 371

ANEXO XII - ESTRUTURA DO QUESTIONÁRIO FEITO AOS ESTUDANTES (PI2/DW2 - ESTGOH - 2019/2020) 375

ANEXO XIII - ESTRUTURA DO QUESTIONÁRIO FEITO AOS ESTUDANTES (FP - ESTGOH 2020/2021) 383

Índice de Figuras

Figura 1 - Previsão da procura de profissionais da área das TIC até 2020 em Portugal (previsão de 2014) [4] 3

Figura 2 - Exemplo de código de solução baseada em tradução literal do enunciado 8

Figura 3 - Exemplo de código de solução obedecendo às regras da linguagem 8

Figura 4 - As várias fases do processo de investigação utilizando a metodologia DBR. ...22

Figura 5 - Arquitetura de tutor inteligente proposta por Dag e Erkan [58] 26

Figura 6 - Arquitetura do sistema CIMEL-ITS [59] 27

Figura 7 - Arquitetura proposta por Joseph Beck et al. [60] 27

Figura 8 - Arquitetura geral do sistema Logic-ITA [72]. 30

Figura 9 - Representação hierárquica do sistema GAME [85]. 35

Figura 10 - Exemplo do relatório de avaliação de um exercício do sistema Web-CAT [95]. 37

Figura 11 - Exemplo de página com detalhes de uma tentativa de resolução no sistema Web-CAT 38

Figura 12 - Visualização do progresso dos estudantes no sistema TestMyCode [97]. 39

Figura 13 - Exemplo de utilização do CodeRunner [https://coderunner.org.nz/]. 40

Figura 14 - Exemplo do modo "browse" do sistema Retina [100].....41

Figura 15 - Apresentação de estatísticas relativas às submissões de um exercício no sistema OK [http://okpy.org].....43

Figura 16 - Exemplo de teste do código desenvolvido no sistema Spinoza [101].....43

Figura 17 - Exemplo da página utilizada pelos estudantes para resolverem os problemas no sistema Edgar [102]. 44

Figura 18 - Exemplo de utilização do sistema *Gradescope* [https://www.gradescope.com/] 47

Figura 19 - Os diversos níveis de granularidade [110]. 49

Figura 20 - Arquitetura geral do sistema - iteração zero.....57

Figura 21 - Estrutura de informação associada a cada exercício - iteração zero. 58

Figura 22 - Ilustração do processo de envio dos *snapshots* para o servidor. 59

Figura 23 - Exemplo de um cabeçalho de uma classe com identificação do estudante e do exercício - iteração zero.....60

Figura 24 - Representação da rede bayesiana utilizada para construir o modelo do estudante.....62

Figura 25 - Representação do processo de comparação entre modelos.....63

Figura 26 - Esquema representativo da aplicação de monitorização utilizada pelos professores.....65

Figura 27 - Ferramenta temporária de envio de código para o servidor - iteração zero. ..68

Figura 28 - Exemplo da utilização dos "sensores".....69

Figura 29 - Exemplo de utilização da primeira versão do *plug-in* para *PyCharm* - iteração um.....72

Figura 30 - Exemplo de mensagem do sistema apresentada no ambiente *PyCharm* - iteração um. 73

Figura 31 - Exemplo de código *Python* com os elementos identificativos - iteração um. ..74

Figura 32 - Exemplo de definição do *template* do código no *PyCharm*. 74

Figura 33 - Fluxo de processamento de um *snapshot* - iteração um.....75

Figura 34 - Exemplo de lista de regras a utilizar para avaliar o código de um exercício - iteração um 77

Figura 35 - Vista parcial do *dashboard* da ferramenta de monitorização - iteração um. .80

Figura 36 - Resumo de todos os exercícios que um estudante tentou resolver - iteração um.....82

Figura 37 - Detalhes de um *snapshot* após ser processado - iteração um. 82

Figura 38 - Representação do novo processo de identificação dos *snapshots* - iteração um. 85

Figura 39 - Distribuição dos *snapshots* em função da hora/dia da semana.....87

Figura 40 - Submissões efetuadas tardiamente - iteração um. 91

Figura 41 - Submissões efetuadas antes do tempo 91

Figura 42 - Elementos identificativos dos *snapshots* para as linguagens C e Java - iteração dois. 95

Figura 43 - Representação da arquitetura do sistema - iteração dois. 97

Figura 44 - Vista parcial do *dashboard* da ferramenta de monitorização - iteração dois. 98

Figura 45 - Exemplo de visualização da distribuição dos *snapshots* ao longo do tempo - iteração dois. 99

Figura 46 - Exemplo de visualização da distribuição dos *snapshots* em função do dia e hora - iteração dois. 99

Figura 47 - Exemplo de visualização da distribuição dos *snapshots* de cada estudante ao longo do tempo - iteração dois. 100

Figura 48 - Vista parcial da matriz de desempenho - iteração dois. 101

Figura 49 - Página com detalhes das tentativas de resolução de um exercício - iteração dois. 102

Figura 50 - Lista de notificações - iteração dois. 104

Figura 51 - Exemplo de visualização do grafo representativo da sequência pela qual foram resolvidos os exercícios - iteração dois. 109

Figura 52 - Mensagens informativas apresentadas no *PyCharm* - iteração três. 113

Figura 53 - Representação do processo de determinação dos estudantes que estão a trabalhar com um ritmo mais lento e mais rápido. 117

Figura 54 - Evolução da performance dos estudantes ao longo do tempo - iteração três. 118

Figura 55 - Exemplo de visualização da distribuição dos estudantes em termos de ritmo de trabalho ao longo do tempo - iteração três. 118

Figura 56 - Exemplos de visualização da comparação da performance de estudantes face aos colegas de turma 119

Figura 57 - Exemplo de visualização da evolução do progresso dos estudantes de uma turma - versão um. 121

Figura 58 - Exemplo de visualização da evolução do progresso dos estudantes de uma turma - versão dois 122

Figura 59 - Exemplos de visualização da comparação do progresso de estudantes face aos colegas de turma 122

Figura 60 - Comparação da evolução do progresso dos estudantes face aos objetivos definidos pelo professor.....123

Figura 61 - Exemplos de visualizações de comparação da evolução do progresso de estudantes em concreto face aos objetivos definidos pelo professor 123

Figura 62 - Exemplo de visualização da evolução da consistência da performance e progresso do trabalho de um estudante 124

Figura 63 - Exemplo de visualização de performance inconstante de um estudante. 125

Figura 64 - Exemplo ilustrativo da funcionalidade de edição e teste de código pelo professor - iteração três. 126

Figura 65 - Listagem dos exercícios realizados por um estudante - iteração três.....127

Figura 66 - Formulário de introdução de feedback a enviar aos estudantes.....128

Figura 67 - Exemplo de email enviado aos estudantes com *feedback* sobre um *snapshot*. 128

Figura 68 - Lista de *snapshots* com informação sobre fornecimento de *feedback* 129

Figura 69 - Distribuição dos *snapshots* ao longo do tempo - Programação - LEGI - iteração três 130

Figura 70 - Distribuição dos *snapshots* em função da hora e dia da semana - iteração três 132

Figura 71 - Distribuição dos *snapshots* ao longo do tempo (PI2 - LEI - ESTGOH - iteração três). 133

Figura 72 - Evolução da performance dos estudantes ao longo do tempo (PI2 - LEI - ESTGOH - iteração três).....134

Figura 73 - Exemplo da apresentação de *feedback* do validador de HTML do W3C.....138

Figura 74 - Exemplos de diferentes formas de definição dos *inputs* 140

Figura 75 - Exemplos de utilização do método de definição de *inputs* 142

Figura 76 - Formulário de definição de casos de teste 143

Figura 77 - Ilustração do processo de substituição do valor dos inputs 144

Figura 78 - Informação que é necessário introduzir para cada exercício - iteração quatro. 147

Figura 79 - Exemplo da nova configuração do *dashboard* - iteração quatro. 148

Figura 80 - Listagem com os exercícios que o estudante tentou resolver - iteração quatro 148

Figura 81 - Informação sobre resultados da execução do código com casos de teste - iteração quatro. 149

Figura 82 - Exemplo da configuração da "matriz resumo" - iteração quatro. 150

Figura 83 - Exemplo da execução de uma *Applet* no sistema *CodeInsights*. 151

Figura 84 - Exemplo da *live feed*. 152

Figura 85 - Exemplo de listagem de estudantes com informação sobre o seu desempenho - iteração quatro. 153

Figura 86 - Detalhes sobre resultados da execução do código com casos de teste - iteração quatro. 154

Figura 87 - Algoritmo geral do modelo de atenção seletiva artificial proposto por Macedo. 160

Figura 88 - Evolução dos valores de S e de IG. 165

Figura 89 - Listagem de “registo” de geração de notificações. 167

Figura 90 - Formulário de "registo" de notificação. 168

Figura 91 - Formulário de especificação de expectativas. 169

Figura 92 - Distribuição dos *snapshots* ao longo do tempo - Acertar o Rumo - iteração quatro. 171

Figura 93 - Distribuição dos *snapshots* em função do dia e hora - FP ESTGOH - iteração quatro 176

Figura 94 - Distribuição dos *snapshots* ao longo do tempo - FP ESTGOH - iteração quatro 176

Figura 95 - Distribuição dos *snapshots* em função da hora e dia da semana - P ESTGOH - iteração quatro 180

Figura 96 - Exemplo de apresentação dos resultados da execução do código com os casos de teste - iteração cinco. 185

Figura 97 - Representação da abordagem de deteção de plágio tradicional (em cima) vs. deteção em tempo real (em baixo). 189

Figura 98 - Algoritmo de comparação de submissões 190

Figura 99 - Listagem de exercícios com potenciais situações de plágio. 191

Figura 100 - Detalhes sobre a comparação entre todas as submissões para um exercício. 191

Figura 101 - Detalhes da comparação entre duas submissões 192

Figura 102 - Arquitetura geral do sistema - iteração cinco. 194

Figura 103 - As várias etapas do processamento de um *snapshot* 195

Figura 104 - Visualização do estado das últimas tentativas de resolução para cada um dos exercícios propostos. 196

Figura 105 - Nova configuração da matriz resumo - iteração cinco. 197

Figura 106 - Distribuição dos *snapshots* ao longo do tempo - iteração cinco 198

Figura 107 - Matriz com as situações em que potencialmente existia plágio. 199

Figura 108 - Frequência de utilização da ferramenta de monitorização destinada aos estudantes - Programa Acertar o Rumo - iteração seis. 209

Figura 109 - Frequência de utilização da ferramenta de monitorização destinada aos estudantes - FP LEI ESTGOH - iteração seis. 213

Figura 110 - Distribuição dos snapshots recebidos ao longo das horas do dia - PI2 ESTGOH - iteração seis. 215

Figura 111 - Distribuição dos snapshots recebidos para os vários dias da semana - PI2 ESTGOH - iteração seis. 215

Figura 112 - Distribuição dos *snapshots* recebidos ao longo do tempo para cada ficha de trabalho - PI2 / ESTGOH - iteração seis. 216

Figura 113 - Frequência de utilização da ferramenta de monitorização destinada aos estudantes - PI2 LEI ESTGOH - iteração seis. 220

Figura 114 - Listagem de exercícios resolvidos por um estudante - iteração seis. 221

Figura 115 - Ilustração do processo de escolha de consola. 226

Figura 116 - Exemplo da consola de apresentação dos resultados dos testes no Eclipse227

Figura 117 - Exemplo de *feedback* personalizado para um exercício. 227

Figura 118 - Legenda da informação contida na matriz resumo - iteração sete. 229

Figura 119 - Vista parcial da nova configuração da matriz resumo - iteração sete. 230

Figura 120 - Lista de tentativas de resolução efetuadas por um estudante para um exercício - iteração sete. 230

Figura 121 - Exemplo de classificação e comentário inserido manualmente pelo professor. 231

Figura 122 - Vista parcial da tabela resumo das classificações dos exercícios. 232

Figura 123 - Representação dos novos botões de navegação entre exercícios e estudantes. 232

Figura 124 - Apresentação das modificações feitas sobre o código fonte de modo a serem feitos os testes unitários, ...233

Figura 125 - Exemplos de apresentação dos resultados dos testes unitários. 234

Figura 126 - Exemplo da forma como a classificação e o *feedback* é apresentado aos estudantes. 234

Figura 127 - Nova configuração da página de detalhes da última tentativa de resolução de um exercício na ferramenta de monitorização destinada aos estudantes - iteração sete. 235

Figura 128 - Exemplo de um exercício em que o professor procedeu à edição da solução desenvolvida pelos estudantes. 236

Figura 129 - Distribuição das respostas dos alunos relativamente à utilização da ferramenta de apresentação de feedback no Eclipse - IPJ AR - iteração sete. 239

Figura 130 - Distribuição dos estudantes em termos da frequência de acesso à ferramenta de monitorização - IPJ AR - iteração sete. 240

Figura 131 - Distribuição dos *snapshots* recebidos ao longo da duração da experiência - IPJ AR - iteração sete. 242

Figura 132 - Distribuição dos *snapshots* de cada aluno ao longo da duração da experiência - FP ESTGOH - iteração sete. 247

Figura 133 - Distribuição dos *snapshots* recebidos ao longo do tempo - FP ESTGOH - iteração sete. 248

Figura 134 - Distribuição dos estudantes em termos da frequência de acesso à ferramenta de monitorização - FP LEI ESTGOH - Iteração sete. 251

Figura 135 - Utilização do plug-in no Visual Studio Code. 264

Figura 136 - Exemplo da nova forma de representar os resultados de testar o código com os casos de teste - iteração nove. 266

Figura 137 - Exemplo da pré-visualização de um ficheiro HTML realizado por um estudante. 266

Figura 138 - Exemplo da listagem de exercícios destinada aos estudantes - iteração nova. 267

Figura 139 - Exemplo de utilização do editor de código disponibilizado no sistema - iteração nove. 268

Figura 140 - Distribuição das tentativas de resolução dos estudantes ao longo do tempo - iteração nove. 269

Figura 141 - Distribuição das tentativas dos estudantes ao longo das horas do dia - iteração nove. 270

1.Introdução

Vivemos numa sociedade cada vez mais dependente de tecnologia para o desempenho de tarefas tão diversas como a reserva de refeições ou a leitura de notícias. Com efeito, inúmeros sistemas essenciais da nossa sociedade, como os sistemas bancários, de saúde, de abastecimento de água ou de distribuição de eletricidade, estão totalmente dependentes da disponibilidade da tecnologia, sendo em muitos casos inclusivamente impossível a realização das tarefas sem o recurso a sistemas informáticos (e.g. entrega de declarações de impostos como IVA ou IRS, matrículas escolares, concursos de professores, entre outros). Mais recentemente, por força do confinamento provocado pela pandemia de CoVid19, assistiu-se a uma (quase) total dependência da utilização de sistemas informáticos para o suporte da maioria dos sectores de atividades, tais como o ensino, compras alimentares e de outros bens essenciais, as farmácias, consultas médicas, e de quase todo o género de trabalho que teve de ser exercido remotamente.

Em 1998 Meserve [1] publicou um artigo sobre os eventuais perigos da crescente dependência de tecnologia. Segundo este autor, um dos principais problemas reside no facto de por vezes andarmos à procura de uma solução que permita resolver um determinado aspeto do nosso dia a dia no imediato, sem acautelar devidamente as eventuais consequências de longo prazo que daí possam advir. Este fenómeno manifesta-se frequentemente no nosso dia a dia sempre que um sistema informático falha ou está indisponível (por motivos de falha de energia, por exemplo). Se, antigamente, numa mercearia, por exemplo, a conta era feita no papel pelo merceeiro, nos dias que correm, é praticamente impossível fazer o mesmo num qualquer supermercado.

No seu livro "*The Glass Cage: Where Automation Is Taking Us*" [2], Nicholas Carr faz uma interessante análise do impacto que o processo de digitalização teve em setores de atividade como a medicina. Um dos grandes argumentos utilizados a favor da digitalização consiste no aumento da eficiência e produtividade, acompanhada por uma drástica redução de custos. No entanto, Carr verificou que contrariamente ao que seria de esperar, em muitos casos, esta digitalização acabou por resultar no incremento dos preços dos serviços prestados, sem que tivesse existido um considerável incremento na qualidade ou rapidez dos mesmos.

Para além disso, no nosso dia a dia, somos frequentemente confrontados com empresas ou instituições totalmente paralisadas devido a problemas informáticos e que acarretam avultados prejuízos. A título de exemplo, em maio de 2017 uma falha no sistema informático da *British Airways*, que permitia fazer o *check-in* e imprimir os cartões de embarque, levou ao cancelamento de centenas de voos, o que representou um prejuízo de mais de £80.000.000[1]. Curiosamente, alguns meses depois, em agosto, a mesma empresa voltaria a ter novamente problemas graves com o sistema informático[2].

Se é certo que esta crescente dependência da tecnologia comporta uma série de desafios, é ponto assente que se trata de um processo imparável e absolutamente irreversível. Aliás, a consciência tem de ser que a tecnologia se desenvolve cada vez mais a cada segundo e pode-se afirmar que é já um "ser" com vida própria. Assim, resta-nos fazer os possíveis para mitigar as consequências eventualmente negativas que daí possam advir.

Um aspeto deveras importante é o facto da maioria dos erros que afetam os sistemas informáticos se devem muitas vezes a fatores humanos, nomeadamente a defeitos de programação[3]. Jiau et al. [3] referem que o processo de programação de sistemas informáticos pode por vezes ser repetitivo e entediante, o que pode contribuir para que muitas vezes sejam introduzidos erros ou omissões não intencionais no código desenvolvido. Curiosamente, no estudo "*e-Skills in Europe - Portugal Country Report*" [4] realizado em 2014 pela empresa ***Empírica***, era apontada a escrita manual de código *Java* como um problema dos desenvolvedores portugueses:

> "*Overall, more emphasis should be put on software development skills, as for instance, java still keeps on being written by hand instead of automatic modelling and code generation.*"

Tal como o estudo refere, de facto, existem ferramentas *low-code* ou *no-code* que possibilitam a geração de código (*e.g.* ***IronSpeed***[4], ***OutSystems***[5]). No entanto, independentemente de os sistemas serem integralmente concebidos e implementados por programadores (humanos) ou envolvendo algum tipo de automatização, a intervenção

[1] https://www.ft.com/content/98367932-51c8-11e7-a1f2-db19572361bb
[2] http://fortune.com/2017/08/02/british-airways-system-failure-heathrow/
[3] https://www.makeuseof.com/tag/worst-programming-mistakes-in-history/
[4] http://www.ironspeed.com/
[5] https://www.outsystems.com/

dos programadores está sempre presente, nem que seja para efeitos de personalização das soluções geradas automaticamente.

Tendo em consideração a crescente importância do uso da tecnologia no nosso dia a dia e a necessidade da intervenção dos seres humanos no processo de criação desta tecnologia, não será de estranhar que a área das Tecnologias de Informação e Comunicação (TIC), nomeadamente no que concerne ao desenvolvimento de *software*, tenha vindo a ganhar uma atratividade cada vez maior.

No já mencionado estudo "*e-Skills in Europe - Portugal Country Report*", é indicada a falta de vários milhares de profissionais da área das TIC para preencher as vagas que potencialmente venham a ser criadas nos próximos anos. No caso concreto de Portugal, estimava-se que até 2020 existiria a necessidade de contratar aproximadamente 15.000 novos profissionais na área das TIC (Figura 1).

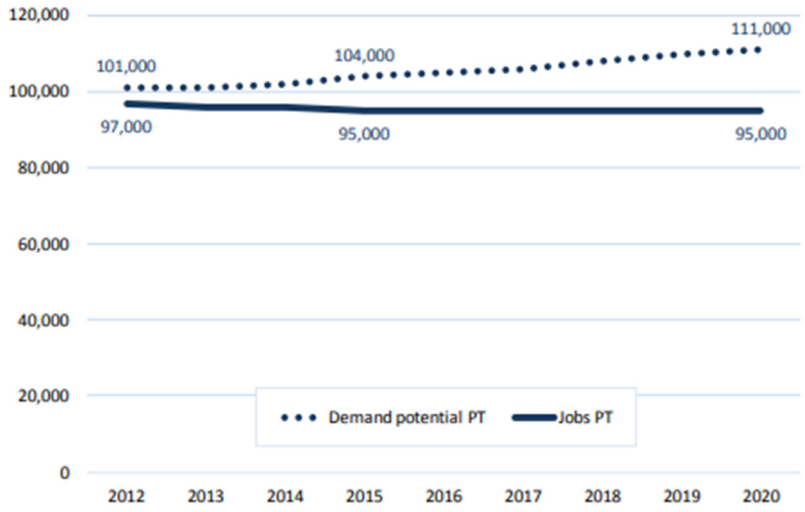

Figura 1 - Previsão da procura de profissionais da área das TIC até 2020 em Portugal (previsão de 2014) [4]

Tendo em conta esta crescente procura a nível mundial por profissionais qualificados, diariamente, vários milhões de pessoas pelo mundo fora estão envolvidos em algum tipo de aprendizagem de programação. Alguns fazem-no no âmbito de cursos de ensino profissional ou secundário, muitos no âmbito de cursos de ensino superior diretamente ligados à área das (TIC) (*e.g.* Engenharia Informática, Engenharia de Computadores, etc.) e outros ainda em cursos superiores não necessariamente ligados às TIC, mas para os quais estas se mostram relevantes (*e.g.* Engenharia Mecânica, Física, etc.).

Dada a enorme atratividade da área e os elevados níveis de desemprego em muitas outras áreas, têm igualmente surgido nos últimos anos diversos cursos intensivos de

programação de curta duração (entre três e doze meses) com uma vertente maioritariamente prática. Na maioria dos casos, estes cursos são dirigidos a detentores de formação superior em outras áreas que não as TIC. Finalmente, existem estudantes autodidatas que decidem aprender a programar, de forma autónoma, recorrendo para tal a cursos ou tutoriais que facilmente podem ser encontrados na Internet.

Nos últimos anos, assistiu-se igualmente a um intensificar da ideia que é importante apresentar a programação aos estudantes tão cedo quando possível [5]. Assim, graças a projetos como o ***Ubbu***[6], ***Scratch***[7], ***Code.org (Hour of code)***[8]**,** entre outros, atualmente milhões de estudantes no mundo inteiro começam a dar os primeiros passos no mundo da programação, enquanto frequentam o ensino básico (ou mesmo antes, em idades tão precoces como os dois anos). Obviamente, o objetivo destas iniciativas não passa por fazer com que todos os estudantes venham a ser programadores ou venham a desempenhar algum tipo de profissão ligada às TIC. Em vez disso, a programação é encarada como um meio interessante para desenvolver a capacidade de resolução de problemas [6].

Perante este aumento na procura por profissionais na área das TIC, não deixa, pois, de ser curioso que, entre os cursos na área das TIC, se encontrem aqueles que apresentam maiores taxas de reprovação e abandono precoce [7]. Com efeito, num estudo da Direção Geral de Estatísticas da Educação e Ciência [8], foi seguido o percurso académico dos estudantes inscritos em ciclos de estudos de licenciatura nas instituições de ensino superior portuguesas entre 2011 e 2015. Neste estudo, os cursos da área disciplinar da "Informática" e os da área de "Engenharia e Técnicas Afins", na qual se enquadram igualmente muitos dos cursos da área das TIC (*e.g.* Engenharia Informática, Engenharia de Computadores, etc.), são os que apresentam simultaneamente maior taxa de abandono e menor taxa de estudantes que tenham concluído o curso no período estudado.

Num estudo semelhante relativamente à Estónia, Kori et al. [9] apresentam valores semelhantes aos obtidos em Portugal. Segundo os autores deste estudo, existem diversos fatores a contribuir para esta elevada taxa de abandono, nomeadamente, fatores demográficos, fatores financeiros, performance que os estudantes tiveram ao longo do ano, e motivação ou fatores ligados à integração social dos estudantes. É importante

[6] https://www.ubbu.io/

[7] https://scratch.mit.edu/

[8] https://code.org/

realçar que grande maioria destes fatores não poderá necessariamente ser relacionado com a especificidade de cursos de TIC e poderá ocorrer em cursos de outras áreas. No estudo levado a cabo por Kori et al., um aspeto relevante ligado à performance dos estudantes, aponta para a maior incidência de desistências entre os estudantes que obtiveram piores resultados na prova de matemática realizada no final do ensino secundário (semelhante aos exames nacionais em Portugal), aspeto também referido no estudo previamente apresentado em relação a Portugal [8].

Um fator também realçado em diversos outros estudos [7] [10] diz respeito à escolha do curso por parte dos estudantes. Com efeito, ao longo dos últimos anos, têm vindo a ser cativados para estes cursos muitos estudantes sem grande motivação para os mesmos e que acabam por escolhê-los simplesmente pelo facto de os níveis de desemprego serem consideravelmente baixos quando comparados com áreas como as Ciências Sociais, por exemplo. Assim, muitos estudantes entram nos cursos com expetativas totalmente erradas em relação ao que os espera e, quando são confrontados com a realidade, têm dificuldade em progredir acabando por desistir ou mudar de curso.

Por forma a tentar reduzir estes índices de abandono, algumas instituições têm levado a cabo ações tais como "dias abertos" ou cursos de iniciação à engenharia informática, como por exemplo o ***DEI Academy***[9]. Deste modo, os estudantes podem ter um contacto próximo com os cursos e instituições de ensino, mesmo antes de ingressarem nelas. Outras instituições de ensino recorrem a processos de admissão de estudantes muito mais elaborados envolvendo testes específicos realizados na instituição de ensino, cartas de motivação e entrevistas. Embora este tipo de abordagens possa ter efeitos positivos no sentido de permitir selecionar à partida os estudantes mais motivados e com mais capacidades, tendo em conta o modo como se processa o acesso ao ensino superior em Portugal, não pode ser utilizado.

Um outro aspeto relevante diz respeito ao facto de uma parte considerável dos estudantes que abandona os estudos ser absorvida pelo mercado de trabalho, ou seja, dada a grande falta de profissionais, mesmo estudantes com dificuldades de aprendizagem acabam por conseguir obter colocação numa empresa, com alguns riscos que tal colocação possa acarretar.

[9] https://academy.dei.uc.pt/

1.1. Problemas e dificuldades de aprender programação

As competências que um profissional da área TIC deve possuir são muito diversas e vão muito além dos aspetos meramente técnicos [11]. Em todo o caso, dada a natureza das tarefas a realizar, e se bem que nem todos os profissionais das TIC sejam programadores, uma boa competência ao nível da programação é sem dúvida importante para o sucesso dos estudantes e posteriormente para o seu sucesso enquanto profissionais. Contudo, o processo de aprendizagem de programação apresenta diversas dificuldades.

Antes de analisar as principais dificuldades sentidas pelos estudantes, será eventualmente conveniente perceber o que é afinal a programação. Segundo Moström [12]:

> *Programming is the act of understanding a problem, formulating a solution, and writing down the solution in such a way that a computer can use the solution to solve the problem.*

Ou seja, programação não é apenas o ato de escrever o código fonte. Em vez disso, programar envolve uma série de outros processos, tais como a compreensão do problema e a conceção das soluções, que deverão ocorrer antes do processo de escrita do código. Para além disso, no processo de codificação das soluções, deverão ser feitos esforços no sentido de encontrar as soluções mais eficazes e eficientes.

De acordo com Rogalsky e Samurçay [13], programar envolve igualmente capacidade de identificar os problemas existentes, verificar como são resolvidos estes problemas no "mundo real", ter conhecimento das ferramentas que determinada linguagem de programação proporciona e, com base nestes três aspetos, implementar uma solução que permita resolver o problema.

Dijkstra [14] defende que para aprender a programar, numa fase inicial, deverão ser utilizadas apenas ferramentas que promovam a capacidade de abstração e de resolução de problemas de modo formal, eventualmente recorrendo à matemática, sem o recurso a qualquer tipo de linguagem de programação. Por outras palavras, segundo Dijkstra, antes de aprenderem a utilizar uma linguagem de programação, os indivíduos devem desenvolver as suas capacidades de identificar e pensar na solução em vez de partir

diretamente para a programação utilizando uma linguagem de alto nível. Esta última abordagem acaba por ser cada vez mais a que é seguida por muitos professores, certamente influenciados e pressionados pela urgência de ensinar os estudantes a fazer coisas mais palpáveis.

Segundo Rhogalsky e Samurçay, um dos principais problemas com a aprendizagem de programação reside no facto de, ao contrário do que acontece com muitos outros conceitos de unidade curriculares como a matemática ou a física, não ser possível, ou pelo menos fácil, encontrar no mundo real representações de conceitos importantes para a programação como variáveis ou recursividade. Este facto obriga a uma elevada capacidade de abstração, que nem todos os estudantes possuem. Também convém sublinhar que, para conseguir programar, é necessário compreender bem o modo como o computador funciona, como os dados são adquiridos do utilizador, como são guardados e como são manipulados internamente, o que para além do mais, pode variar consideravelmente de linguagem para linguagem. Uma vez mais, é necessária uma elevada carga cognitiva e capacidade de abstração.

Além da já mencionada capacidade de abstração e domínio da matemática, segundo Jenkins [15], para conseguir programar, um individuo deverá ter capacidades ao nível da resolução de problemas da forma mais simples e eficiente, perceber a mecânica do processo de programação (criar um ficheiro, editar o ficheiro, compilar o ficheiro, analisar o *output*, etc.), ter capacidade de testar o código, identificar erros de compilação e corrigi-los. Finalmente, os indivíduos deverão ter um conjunto de competências relacionadas com o processo de aprendizagem em si, ou seja, deverão ser, por exemplo, capazes de identificar as suas limitações, não ter medo de perguntar em caso de dúvida ou conseguir identificar quem ou qual o local mais indiciado para obter ajuda.

Uma outra dificuldade geralmente sentida pelos estudantes, principalmente na fase inicial do processo de aprendizagem, passa pelo facto de que, para programar, é necessário utilizar uma linguagem com um sintaxe e semânticas próprias e por vezes bem diversas da linguagem natural [16]. Nesta nova linguagem, há um conjunto de regras que têm de ser obedecidas, tais como a necessidade de terminar as instruções com ";" ou a impossibilidade de utilizar palavras reservados como nomes de variáveis.

Por vezes, este processo de conversão entre o que se pensa fazer e o modo como tal pode ser feito, acaba por constituir uma das principais dificuldades. A título de exemplo, considerando que se pretende desenvolver um programa que imprima "OK", se um

determinado valor estiver entre 10 e 20, e "KO", caso contrário. Fazendo uma tradução direta do enunciado, um estudante poderia ser tentado a resolver o exercício tal como apresentado na Figura 2.

```
if(10 < $valor < 20)
      print ("OK");
else
      print("KO");
```

Figura 2 - Exemplo de código de solução baseada em tradução literal do enunciado

Muito embora do ponto de vista formal esta solução aparentemente esteja correta, será necessário um conhecimento um pouco mais aprofundado da sintaxe da linguagem para perceber que tal solução não funcionará. Em vez disso, seria necessário definir a condição de forma diferente, obedecendo às regras de sintaxe impostas pela linguagem de programação, tal como representado na Figura 3.

```
if($valor > 10 && $valor < 20)
      print ("OK");
else
      print("KO");
```

Figura 3 - Exemplo de código de solução obedecendo às regras da linguagem

Do exemplo anterior, salta à vista outro aspeto importante que está relacionado com o facto de como o próprio enunciado de um exercício poderá condicionar o modo como os estudantes irão tentar resolvê-lo. Neste caso concreto, se no enunciado fosse dito que imprimia "OK", caso o valor fosse maior do que 10 e menor do que 20, e "KO", caso contrário, mesmo um estudante inexperiente muito provavelmente iria tentar resolver este exercício de forma correta.

Um outro aspeto importante que por vezes constitui uma séria dificuldade está relacionado com o facto de, no essencial, um programa acabar por se materializar numa série de operações lógicas. Assim, por vezes, uma falha aparentemente tão insignificante, como a falta de um conjunto de chavetas ou um ponto e vírgula, poderá originar resultados diferentes dos esperados, ainda que o código possa eventualmente ser compilado sem erros. Esta necessidade de prestar atenção aos mais pequenos detalhes é efetivamente um fator considerável na frustração dos estudantes e um importante fator de perda de produtividade.

Hagan et al. [17] e Smith et al. [18] centraram a sua atenção na influência que o conhecimento prévio real (ou não) tem para determinar o sucesso dos estudantes. Tendo em ambos os casos sido verificado que o conhecimento prévio real (ou seja, baseado na demonstração de conhecimentos através da realização de um teste de conhecimentos [19]) constitui de facto um fator determinante para o sucesso dos estudantes nas unidades curriculares de programação.

A este respeito convém mencionar os estudantes repetentes que apesar de supostamente já não estarem a ouvir a matéria pela primeira vez, não são regra geral capazes de obter melhores resultados do que os colegas. Com efeito, estudos levados a cabo por Armstrong et al. [20], permitiram-lhes concluir que muitas das vezes, os estudantes que estão a repetir uma determinada unidade curricular, por terem reprovado, não só não conseguem melhorar os resultados anteriores, como muitas vezes chegam mesmo a obter resultados ainda piores.

Robins [21] refere que o facto de os estudantes já terem tido programação anteriormente não implica necessariamente que estes vão ter melhor desempenho. Pode ter sido utilizado um paradigma de programação diferente ou pode já ter sido há bastante tempo. Com efeito, é frequente encontrar estudantes em unidades curriculares de programação do segundo semestre que pura e simplesmente não possuem competências que supostamente deveriam ter sido adquiridas poucas semanas antes, mesmo entre estudantes que obtiveram bons resultados na unidade curricular do primeiro semestre.

Um outro aspeto importante, e que poderá obviamente ser um corolário de todos os previamente apresentados, é referido por Jenkins [15] e está relacionado com a má reputação que algumas unidades curriculares de programação têm. Deste modo, os estudantes, influenciados por colegas ou antigos estudantes que perpetuaram uma ilusão, poderão encarar logo desde o início a programação como algo difícil e inatingível, sem que tenham eles próprios sentido alguma dificuldade em concreto, acabando por dar origem a um círculo vicioso ao qual pode ser difícil pôr cobro.

Num estudo aprofundado das principais dificuldades sentidas pelos estudantes, Gomes et al. [22] referem que, apesar das dificuldades já apresentadas, é igualmente importante ter em consideração aspetos como as metodologias de ensino utilizadas pelos professores que poderão eventualmente ser desajustadas para promover a aquisição de competências associadas à programação. Referem ainda que muitos estudantes encaram a programação como outra qualquer unidade curricular em que basta assistir às aulas e

ler o material disponibilizado pelos professores para conseguir a aprovação. Ora, como já foi referido, aprender a programar envolve praticar e como tal, não descartando a necessidade de assistir às aulas e ler a matéria, as competências de programação serão efetivamente desenvolvidas resolvendo problemas.

Dado que para programar é necessário adquirir determinadas competências interligadas entre si, Robins [21] defende que em grande medida o insucesso dos estudantes de programação pode ser explicado pelo facto de alguns estudantes não conseguirem adquirir eficazmente algumas competências básicas. Dada a sua precedência em relação às competências seguintes, a não aquisição destas competências iniciais tem como principal consequência a não aquisição eficaz das seguintes. A este fenómeno, Robins chamou "*Learning edge momentum*" e definiu-o do seguinte modo:

> "*Given some target domain of new concepts to be learned, any successful learning makes it somewhat easier to acquire further related concepts from the domain, and unsuccessful learning makes it somewhat harder. Thus when learning a new domain the successful acquisition of concepts becomes self-reinforcing, creating momentum towards a successful outcome for the domain as a whole, and similarly the failure to acquire concepts becomes self-reinforcing, creating momentum towards an unsuccessful outcome.*"

Em termos gerais, Robins defende que, perante a necessidade de aprender determinados conceitos sobre um domínio, a aquisição com sucesso de um determinado conceito dará ao estudante "ânimo" extra para aprender e aplicar devidamente os conceitos seguintes. Já o oposto criará barreiras à progressão da aprendizagem. Neste sentido, a intervenção dos professores para avaliar e ajudar os estudantes a consolidar os conceitos em que eventualmente estão a sentir dificuldades tão rápido quanto possível é fundamental. E quanto mais cedo identificadas as dificuldades, mais eficaz será a intervenção e eficiente a correção do erro.

O conceito de *Learning edge momentum*, acaba de algum modo por estar relacionado com um outro conceito importante, a autoeficácia (*self-efficacy*) [23], que em termos simples consiste na "crença" (*belief*) que um estudante tem relativamente às suas capacidades para executar determinadas tarefas. Para avaliar a autoeficácia recorre-se geralmente à

realização de testes associados a uma escala de autoeficácia [24]. Num processo de aprendizagem é desejável que o valor de autoeficácia de um estudante se mantenha elevado com o decorrer do tempo.

Por todos os motivos apresentados anteriormente, não será de estranhar que as unidades curriculares de programação apresentem consistentemente taxas de reprovação altas (geralmente superiores a 30%) [25]. No entanto, estas taxas de reprovação variam consideravelmente de instituição para instituição e fatores como o tamanho das turmas parecem ser importantes para o sucesso dos estudantes. A título de exemplo, num estudo levado a cabo por Bennedsen e Caspersen, turmas com menos de 30 estudantes apresentaram taxas de reprovação muito inferiores às restantes.

1.2. A importância dos professores no processo de ensino/aprendizagem de programação

O processo de ensino/aprendizagem de programação constitui um importante desafio tanto para os estudantes como para os professores. Para além de fatores como o interesse ou motivação dos estudantes que serão eventualmente transversais a outras áreas do saber, há que ter em consideração a natureza intrínseca da programação.

Nos últimos anos tem-se assistido a uma mudança de paradigma de ensino, no sentido de abandonar uma abordagem mais centrada no professor, a geralmente denominada abordagem tradicional, em favor de uma abordagem mais centrada no estudante.

Numa abordagem mais tradicional, o professor assume uma posição predominante como o transmissor do conhecimento e os estudantes uma posição muito mais passiva, limitando-se a "absorver" o conhecimento transmitido pelos professores sem grande lugar à construção e partilha de conhecimento entre estudantes. Já numa abordagem mais centrada nos estudantes, o professor passa a assumir um papel de mediação, muito menos dominante, sendo dado espaço aos estudantes para construírem o conhecimento ao seu ritmo e com base nos materiais mais adequados ao seu caso e muitas vezes com base em diálogo/interação direta com os seus colegas.

Mesmo no caso das unidades curriculares de programação em que os estudantes têm obrigatoriamente de ser (pró-)ativos e focados intelectualmente para conseguirem resolver os desafios propostos pelos professores na forma de fichas de trabalho e/ou projetos de maior dimensão, poderá continuar a existir algum nível de "professor-centrismo".

Tal como defendido por Eriscsson et al. [26], a prática é um dos principais fatores que contribuem para o sucesso dos estudantes. Convém, no entanto, que seja prática de qualidade e em que haja *feedback* constante sobre como as coisas estão a correr e de que forma podem eventualmente ser melhoradas. É precisamente a este nível que a intervenção dos professores é sem dúvida determinante. Por mais que um estudante tente resolver um determinado problema de uma forma (errada), caso este não tenha *feedback* sobre isso, provavelmente irá continuar a tentar resolver o problema da mesma forma.

Neste sentido, convém referir que os estudantes deverão ser capazes de autonomamente resolver os seus problemas, cabendo ao professor, numa primeira fase, a tarefa de fornecer os elementos essenciais para que estes iniciem a resolução dos problemas e, consequentemente, monitorizar o seu progresso e reparar desvios, intervindo mediante o fornecimento de *feedback* construtivo sempre que tal se revele pertinente. Ericsson et al., defendem o número de 10.000 horas de prática, no entanto, facilmente se compreenderá que 10.000 horas com boa orientação, não equivalem a 10.000 horas com pouca (ou nenhuma) orientação.

Kirschner et al. [27] reforçaram a importância do fornecimento de *feedback* personalizado aos estudantes como um aspeto preponderante para a construção de um processo de aprendizagem eficaz.

De acordo com Bloom [28], o processo de ensino-aprendizagem mais eficaz assenta na tutoria individualizada (um estudante-um professor). No entanto, dada a elevada demanda por profissionais qualificados e consequentemente o elevado número de estudantes inscritos em cursos (e unidades curriculares) de programação, é obviamente muito difícil conseguir este tipo de instrução.

De igual modo, Mendes et al. [29] e Raabe et al. [30] referem que um dos aspetos que mais contribui para melhorar o desempenho dos estudantes é o acompanhamento do seu progresso pelos professores de modo a identificar atempadamente as dificuldades que os estudantes revelem por forma a agir em tempo útil. Também Gomes et al. [31] indicam a adoção de abordagens motivadoras e o fornecimento de suporte personalizado aos estudantes como um fator importante para o sucesso dos mesmos.

Tidos há bem pouco tempo como o futuro da educação, os *Massive Open Online Course* (MOOC) têm como uma das mais importantes premissas o facto de cada estudante poder aprender de acordo com o seu próprio ritmo e de acordo com o tempo que tinha

disponível. No entanto, tal como diversos estudos indicam [32] [33], apesar de existirem vários cursos com centenas e mesmo milhares de alunos inscritos à partida, a verdade é que pouco mais de 10% acabam efetivamente por concluir estes cursos com sucesso.

Principalmente no início, a grande maioria dos cursos fornecidos por estas plataformas não era muito mais do que um conjunto de vídeos disponibilizados online aos quais se associavam algumas ferramentas simples de interação entre os estudantes e os instrutores (p.e. fórum, *chat*, etc.) e ferramentas de avaliação dos conhecimentos adquiridos (p.e. *quizzes*). No entanto, também no caso destes cursos o fornecimento de *feedback* e suporte personalizado aos estudantes (incluindo ao nível da programação), constitui um fator preponderante para o sucesso dos estudantes [34]. No entanto, dado o elevado número de estudantes inscritos a frequentar estes cursos (por vezes na ordem dos milhares), o *feedback* pode pura e simplesmente não existir, ou então, de modo a tal ser possível, é necessário recorrer a sistemas de fornecimento de *feedback* em escala tais como os sistemas ***OverCode*** [35] ou ***AutoStyle*** [36], por exemplo, em que em todo o caso é perdido algum tipo de personalização das mensagens.

Finalmente, poderiam ser contratados dezenas de instrutores. No entanto, atendendo ao facto de muitas vezes estes cursos serem gratuitos ou terem um preço simbólico, facilmente se compreenderá que tal não é financeiramente viável. Assim, quando é fornecido, o *feedback* é usualmente demasiado genérico e os estudantes por vezes não conseguem compreendê-lo adequadamente devido a este não ser focado nos seus problemas em concreto. Como resultado, como já visto anteriormente, muito embora dezenas de milhares de estudantes se inscrevessem nos cursos, apenas uma fração muito pequena consegue a concluí-los com sucesso. Assim, parece uma vez mais evidente que, de facto, o apoio dado pelos professores desempenha um fator chave na performance dos estudantes.

1.3. Problemas e dificuldades de ensinar a programar

Segundo Arah et al. [37], toda a gente é capaz de aprender a desempenhar uma determinada tarefa. A grande diferença reside no ritmo e consequentemente no tempo que pode levar a que aprendam. Um dos grandes problemas é que, na maior parte dos casos, os professores simplesmente não dispõem de tempo para acompanhar o processo de aprendizagem dos estudantes que levam consideravelmente mais tempo do que os colegas para aprender. Do ponto de vista do professor, todos os estudantes devem ser

importantes e nenhum deve ficar para trás, contudo, perante constrangimentos de ordem temporal e não só, os professores poderão simplesmente não poder dispor do tempo necessário para acompanhar devidamente o processo de aprendizagem de todos os estudantes.

Tipicamente, em aulas de caráter mais teórico, o professor apresenta os principais conceitos que posteriormente serão exercitados em aulas de caráter mais prático através da realização de fichas de trabalho com exercícios sobre esse conceito. O grande problema é que por vezes podem ser propostas fichas de trabalho com 10, 20 ou mais exercícios sobre um determinado conceito para resolução num curto espaço de tempo.

Os professores poderão eventualmente acompanhar o estudante nas aulas e até ir vendo o resultado de alguns exercícios, caso os estudantes decidam mostrar, no entanto, dificilmente se conseguirão lembrar de todos os exercícios que um determinado estudante fez individualmente, ou de todas as dificuldades que um estudante específico teve para lá da duração da aula e muito mais dificilmente conseguirão acompanhar de forma eficaz o trabalho realizado fora do contexto das aulas.

Além disso, há que ter em conta as elevadas taxas de absentismo (voluntário ou não) e as diferentes características de personalidade dos estudantes. Por motivos como a timidez ou o medo de serem criticados e/ou exporem as suas fragilidades perante os colegas, muitos estudantes poderão inclusivamente estar na sala de aulas com os professores, sendo, contudo, incapazes de mostrar o trabalho que realizaram ou indicar as dúvidas e/ou dificuldades que os impediram de progredir, preferindo por vezes perguntar aos colegas (que poderão saber ainda menos do que eles) ou então deixar arrastar as dúvidas, com todas as consequências negativas que daí possam advir.

Nas situações em que os estudantes colocam as dúvidas aos professores, poderá ser difícil estes últimos saberem exatamente qual o "estado" atual dos primeiros em termos das competências já adquiridas assim como quais as suas reais dúvidas. Frequentemente os professores caiem na tentação de simplesmente resolver os erros de programação que o estudante está a enfrentar no momento, em vez de tentarem perceber as razões do problema e fornecer os meios necessários para que sejam os próprios estudantes a solucioná-los. Esta situação leva a que as dúvidas dos estudantes persistam e, como tal, poderão voltar a necessitar da intervenção dos professores para resolver problemas semelhantes aos experienciados anteriormente, mas para os quais os estudantes poderão não ser capazes de identificar as causas e muito menos o modo como os resolver.

Frequentemente, os professores são confrontados com situações em que os estudantes, não tendo conseguido adquirir uma ou mais competências, pesquisam na Internet ou pedem aos colegas uma solução para o problema que têm entre mãos, sem que tenham a mínima noção do que está a ser feito e configurando, na grande maior parte das vezes, situações claras de plágio. Para além disso, muitas vezes o código obtido poderá não resolver o problema exatamente do modo como era pedido, ou os estudantes poderão simplesmente não conseguir integrar corretamente esses pedaços de código no código já desenvolvido, o que faz com que percam muito tempo a tentar fazer funcionar código não desenvolvido por eles e que não percebem ao invés de resolverem os problemas por eles próprios. Também nestas situações, a intervenção dos professores poderá ser determinante no sentido de identificar as dificuldades dos estudantes, tentando recolocar o foco dos estudantes no processo de aprendizagem e não na simples produção de um aglomerado de linhas de código que pouco ou nada faz.

Há ainda que ter em conta o facto de nem todos os estudantes conseguirem aprender ao mesmo ritmo. Tal facto é deveras importante, pois se alguns estudantes conseguem adquirir e aplicar corretamente determinada competência facilmente, haverá outros em que será necessário despender muito mais tempo e mesmo assim não há total garantia de que as competências serão de facto adquiridas.

Caso estas diferenças em termos de ritmos de aprendizagem não sejam devidamente tidas em conta por parte dos professores, podem por si só ter consequências devastadoras. Por um lado, para os estudantes que aprendam mais facilmente, que podem ser confrontados com situações em que rapidamente são impedidos de progredir devido aos colegas que aprendem mais devagar, desinteressando-se até mesmo da frequência das aulas. Por outro lado, estes últimos podem ser confrontados com situações em que o ritmo ao qual os conceitos associados a uma determinada competência são apresentados e aplicados é tão acelerado que ainda eles não foram capazes de compreender e aplicar o conceito A e já lhes está a ser apresentado e o conceito D ou E [38]. Para conseguir lidar com todas estas diferentes situações dentro de uma sala de aula, não se pode simplesmente partir para uma segmentação dos estudantes, colocando todos os estudantes "bons" numa turma e os "menos bons" em outra. Em vez disso, há que conseguir encontrar formas de monitorizar as capacidades e limitações de cada estudante e agir em conformidade.

Se a tudo isto juntarmos o facto de na grande maior parte dos casos, as soluções para estes exercícios nunca chegarem a ser submetidas para efeitos de avaliação, facilmente

se conclui que os professores acabam por perder uma importante oportunidade de obter informação sobre o desempenho dos estudantes.

De certa forma confirmando o anteriormente exposto, Carter [39] refere que existem três grandes dificuldades com as quais os professores de unidades curriculares de programação têm de lidar no seu dia: a escolha das estratégias pedagógicas mais adequadas para cada situação; o plágio; e a grande diversidade dos estudantes em termos da sua motivação, conhecimento prévio e também ritmo de trabalho. Em certa medida, todos estas dificuldades poderão estar de certo modo relacionados com a falta de informação que os professores têm sobre as capacidades dos estudantes.

Perante informação muito limitada/incompleta sobre o desempenho dos estudantes, muito dificilmente o professor conseguirá adotar medidas ou tomar as decisões mais acertadas. Quantas vezes no contexto de sala de aulas os professores perguntam se alguém tem dúvidas e ninguém responde? O problema é que perante esta "avaliação", o professor poderá concluir que de facto não há dúvidas e continuar para o próximo tema/conceito, acabando futuramente por constatar que efetivamente havia muitas dúvidas e/ou dificuldades anteriormente.

Em última instância, o não esclarecimento das dúvidas no momento certo pode mesmo conduzir ao abandono/reprovação nas unidades curriculares em causa (caso se trate de estudantes inseridos no contexto de um curso) ou ao simples abandono do processo de aprendizagem (caso se trate de estudantes que aprendam autonomamente).

Esta necessidade de conhecer o "estado atual" de cada estudante em termos de aprendizagem é igualmente importante (ou eventualmente ainda mais!) para conseguir lidar com estudantes com os quais não haja qualquer tipo de contacto presencial habitual, como estudantes de cursos à distância, por exemplo.

1.4. O sistema *CodeInsights*

Pelo anteriormente exposto, torna-se assim evidente a necessidade de construir uma representação conceptual (computacional) do conhecimento dos estudantes - um modelo do conhecimento do estudante - construído e atualizado automaticamente tendo por base as ações do estudante (neste caso concreto da programação, as resoluções dos problemas que lhe são apresentados) de modo a refletir a cada momento o estado em que cada um se encontra.

Necessita-se, pois, dotar os professores de ferramentas que lhes permitam fazer um acompanhamento mais próximo do progresso de cada estudante, possibilitando uma intervenção mais formativa, pró-ativa e acima de tudo, mais informada, de tal modo que, sempre que sejam chamados a intervir, o possam fazer com algum conhecimento prévio sobre os problemas que o estudante possa estar a enfrentar, sem que este facto comprometa a autonomia que os estudantes deverão obrigatoriamente ter.

Com o presente trabalho de investigação pretendeu-se contribuir para melhorar as condições de aprendizagem inicial da programação através da conceção, implementação e avaliação de um sistema de monitorização, o sistema ***CodeInsights***, que recolhe informação dos estudantes na forma de *snapshots* (cópias) do código fonte que estão a escrever enquanto resolvem os exercícios fornecidos pelos professores. Após receção e processamento dos *snapshots* o sistema disponibiliza aos professores um conjunto de informação agregada, visualizações e notificações automáticas sempre que tal se revelar importante.

Após uma análise apurada das soluções já existentes, tornou-se evidente a existência de um conjunto de limitações que estas apresentavam, nomeadamente no que concerne o facto de na generalidade dos casos existir um foco principal na disponibilização de informação aos estudantes e não tanto aos professores. Com efeito, constatou-se que existem défices em relação à quantidade e qualidade da informação disponibilizada aos professores sobre o desempenho dos seus estudantes. Ficou igualmente patente a complexidade do processo de criação e disponibilização de novos exercícios. Fatores estes que poderão inibir muitos professores de recorrer à utilização deste tipo de sistemas. O ultrapassar destas limitações constituiu, pois, um elemento preponderante aquando da conceção de todo o sistema.

Ao longo do processo de conceção e desenvolvimento do sistema, este foi sendo testado em contexto real em várias unidades curriculares de diversos cursos de ensino superior. O *feedback* que foi sendo recolhido ao longo do tempo, foi consecutivamente sendo utilizado no sentido de permitir evoluir o sistema, de modo a melhor conseguir responder às necessidades reais dos professores. No geral, o sistema foi muito bem-recebido, quer pelos professores, quer pelos estudantes, uma vez que o seu uso permitiu que os professores conseguissem acompanhar o progresso dos estudantes de forma muito mais próxima e, consequentemente, permitiu-lhes identificar e atuar sobre as dificuldades dos estudantes de forma mais eficaz.

1.5. Principais contribuições

A principal contribuição deste trabalho é o sistema ***CodeInsights*** cujo desenho, desenvolvimento e avaliação constituiu o ponto central de todo o processo de investigação. Ao longo de todo o processo, várias centenas de estudantes e mais de uma dezena de professores interagiram com ele e além de beneficiarem da sua utilização contribuíram ativamente para que fossem sendo introduzidas melhorias.

Em certa medida, o sistema ***CodeInsights*** já faz parte integrante do processo de aprendizagem de diversas unidades curriculares de programação.

Tendo em conta a sua arquitetura modular, é de esperar que as funcionalidades disponibilizadas pelo sistema ***CodeInsights*** venham a ser estendidas e melhoradas, podendo inclusivamente constituir um ponto de partida para futuros trabalhos de investigação.

Uma outra contribuição importante é o próprio processo de desenvolvimento do sistema, na medida em que exemplifica como uma aplicação com objetivos educativos pode ser desenvolvida de forma incremental, com especial enfoque no envolvimento dos utilizadores finais.

Ao longo das experiências realizadas foram recolhidas algumas centenas de milhar de *snapshots* de código dos estudantes. Assim, já foi iniciado o processo de preparação dos *snapshots* recolhidos com vista à disponibilização dos mesmos a equipas de investigação por forma a serem utilizados nos seus processos de investigação.

Como será apresentado adiante, ao longo deste processo de investigação o sistema ***CodeInsights*** foi sendo desenvolvido de forma iterativa, sendo em cada iteração dada especial atenção a um ou mais questões concretas que iam sendo encontradas. Deste modo, as publicações realizadas ao longo deste processo acabam por refletir de certa forma esta mesma evolução e os principais desafios que foram sendo encontrados.

Assim, além de apresentações em *workshops* de natureza interna ao departamento e grupo de investigação em que o trabalho foi desenvolvido, o trabalho foi disseminado a nível internacional através da sua apresentação em várias conferências internacionais e da posterior publicação dos respetivos artigos e igualmente pela publicação de um capítulo de um livro.

Apresentam-se em seguida as publicações efetuadas no âmbito deste trabalho:

- N. G. Fonseca, L. Macedo, M. J. P. Marcelino e A. J. Mendes, *The importance of using the CodeInsights monitoring tool to support teaching programming in the context of a pandemic*, in 51st Annual Frontiers in Education (FIE'21) Conference, 2021.
- N. G. Fonseca, L. Macedo e A. J. Mendes, *A study on the students' perspective about the usage of a programming monitoring tool*, in Simpósio Internacional de Informática Educativa, 2019.
- N. G. Fonseca, L. Macedo, M. J. P. Marcelino e A. J. Mendes, *Augmenting the teacher's perspective on programming student's performance via permanent monitoring*, in 48th Annual Frontiers in Education (FIE'18) Conference, 2018.
- N. G. Fonseca, L. Macedo e A. J. Mendes, *Using early plagiarism detection in programming classes to address student's difficulties*, in Simpósio Internacional de Informática Educativa, 2018.
- N. G. Fonseca, L. Macedo e A. J. Mendes, *Supporting differentiated instruction in programming classes through permanent progress monitoring*, in SIGCSE 2018, 2018.
- N. G. Fonseca, L. Macedo e A. J. Mendes, *Monitoring the Progress of Programming Students Supported by a Digital Teaching Assistant*, in Portuguese Conference on Artificial Intelligence, 2017.
- N. M. G. Fonseca, L. Macedo e A. J. Mendes, *CodeInsights - Monitorização do desempenho de estudantes de programação*, in Ambientes Virtuais no Ensino Superior, ch. 8, pp. 179-209, 2016.
- N. G. Fonseca, A. J. Mendes e L. Macedo, *CodeInsights - Monitoring programming students' progress*, in International Conference on Computer Systems and Technologies - CompSysTech'16, 2016.

1.6. Organização do documento

Em seguida, no capítulo **Processo de Investigação,** será apresentada a questão de investigação, os principais objetivos a atingir e a metodologia adotada para o conseguir.

No capítulo **Estado da Arte** serão apresentados trabalhos desenvolvidos nos últimos anos e que de algum modo se relacionem com o presente trabalho.

No quarto capítulo, **Desenvolvimento**, serão apresentados os detalhes das várias interações de conceção, desenvolvimento e avaliação do sistema.

Finalmente no capítulo **Conclusões** serão resumidos os objetivos do trabalho, informação sobre o modo como foram atingidos, os principais aspetos que diferenciam o sistema desenvolvido de outras abordagens já existentes e apresentadas algumas possibilidades de melhoria futura.

2. Processo de Investigação

Apresenta-se em seguida informação sobre as questões de investigação, os principais objetivos deste trabalho e a metodologia utilizada para os atingir.

2.1 Questões de investigação

A principal questão de investigação que orientou este trabalho foi motivada pelo que já foi anteriormente referido em relação às dificuldades de aprendizagem de programação, assim como pela necessidade de os professores conseguirem acompanhar de forma mais próxima, veloz e informada o desempenho de cada estudante, saber quais os tópicos em que apresentam mais dificuldades ou quando estão reunidas as condições para prosseguir para o tópico seguinte.

Posto isto, a questão de investigação central formulada foi:

> ***É possível utilizar o trabalho realizado pelos estudantes nas unidades curriculares introdutórias de programação aquando da resolução das fichas de trabalho, de modo a permitir ao professor monitorizar de forma eficaz o seu processo de aprendizagem, com vista a que possa intervir atempadamente sempre que necessário para que os estudantes desenvolvam as competências necessárias?***

Para responder a esta pergunta foi necessário proceder à conceção, desenvolvimento e avaliação de um conjunto de ferramentas, tendo como fim a atingir os objetivos propostos. Importa ainda referir que tendo em conta a abrangência da questão de investigação, para conseguir obter uma resposta satisfatória foi necessário dividi-la num conjunto de questões mais específicas:

> **Q1: Qual a granularidade dos dados que deve ser utilizada para suportar o processo?**
>
> **Q2: Qual a melhor forma para recolher os dados junto dos estudantes?**
>
> **Q3: Qual a informação revelante a apresentar aos professores e qual a melhor forma de o fazer?**

Q4: De que modo é possível integrar um sistema de notificações automáticas com filtragem da informação mais relevante para notificar os professores sobre aspetos que devem merecer a sua atenção?

Ao longo do quarto capítulo serão apresentados detalhes pormenorizados sobre o método utilizado para responder a cada uma destas questões e, consequentemente, como se obteve resposta à questão de investigação principal.

2.2. Metodologia de investigação

Uma vez que os objetivos de investigação passam essencialmente pela conceção, implementação e avaliação de um artefacto - uma ferramenta de monitorização e restantes mecanismos associados - a metodologia de investigação adotada foi a *Design-Based Research (DBR)* [40].

Trata-se de uma metodologia amplamente utilizada em diversas áreas do saber, incluindo a área educativa, com especial relevância para as situações em que deva existir uma forte ligação entre processos de **investigação**, **conceção e desenvolvimento de um artefacto,** seguida da **utilização do artefacto desenvolvido** em situações reais de prática pedagógica e finalmente **avaliação do artefacto utilizado** [41]. Na Figura 4, está representado o ciclo de uma iteração em que é utilizada esta metodologia. Em termos gerais, um processo de investigação em que seja utilizada esta metodologia inclui diversos destes ciclos até que seja obtida uma solução satisfatória.

Figura 4 - As várias fases do processo de investigação utilizando a metodologia DBR.

No essencial, trata-se de uma metodologia que se enquadra no grupo de abordagens de investigação que Juuti et al. [42] designam por ***research for...*** ao invés de ***research about...*** Ou seja, em vez de ser o processo de investigação a focar-se primordialmente na compreensão e conceções teóricas sobre um determinado tema, o principal enfoque reside na procura de soluções "palpáveis" para os problemas que se têm em mãos.

Esta metodologia é, na verdade, baseada em outra designada por *Design Research* [43], que é utilizada há muitos anos em investigação na área da engenharia, indústria e desenvolvimento de sistemas de informação. A principal característica desta abordagem é o facto de assentar no uso de prototipagem evolutiva suportada pelo trabalho de investigação com a finalidade de desenvolver artefactos que permitam resolver problemas concretos.

Com efeito, um trabalho de investigação em que seja utilizada a metodologia DBR consiste num processo iterativo de avanços (e eventuais recuos). Por conseguinte, a solução final é, regra geral, substancialmente diferente da que foi inicialmente conceptualizada e resulta do modo como foram tratados os sucessos e insucesso identificados numa determinada linha temporal.

Esta capacidade de integrar modificações substanciais ao longo do tempo, mesmo em aspetos chave da investigação, é uma das características que faz com que esta metodologia seja a mais adequada para ser utilizada neste trabalho.

Com o processo de avaliação pretende-se, por um lado, verificar se o sistema desenvolvido consegue responder positivamente à questão de investigação formulada e, por outro, suportar o próprio processo de identificação de limitações, procurando novas soluções. Este aspeto é um dos pilares da metodologia de investigação adotada (até porque a DBR é por natureza iterativa), uma vez que a avaliação, muitas vezes de carácter formativo, tem também uma função de procurar identificar formas de melhorar todo o sistema. Para este efeito, ao longo dos dez ciclos que constituíram o processo de investigação, foram realizadas dezoito experiências de avaliação do sistema com utilizadores reais (professores e estudantes).

Nos ciclos iniciais, o principal objetivo passou por avaliar/validar o processo de recolha de dados dos computadores dos estudantes e obter um conjunto de *snapshots* que permitisse testar diversas métricas e formas de apresentação dos dados aos professores. Na prática, os *snapshots* recolhidos permitiram suportar algumas experiências envolvendo

simulações e que foram sendo realizadas nos períodos entre as experiências com utilizadores reais.

Tal como refere Macedo [44], uma das principais vantagens de levar a cabo processos de avaliação simulados prende-se com o facto de não se estar dependente de terceiros (no nosso caso estudantes e professores) e além disso permite também isolar o objeto de estudo de um conjunto de variáveis que não necessitam de ser tidas em consideração. Assim, nos períodos em que não existiam aulas a decorrer ou em que manifestamente não havia disponibilidade de professores para colaborar connosco, o trabalho não parou e foram sempre sendo adicionadas e testadas novas funcionalidades.

Para além do mais, tendo em conta que as experiências reais envolviam estudantes concretos em contextos educativos, era importante que o sistema tivesse sido previamente testado e que não colocasse de modo algum problema/entraves ao progresso académico dos estudantes.

No âmbito dos processos de avaliação/validação com utilizadores reais, foi pedido aos estudantes para instalarem um *plug-in* na ferramenta que utilizavam para programar e que passassem a utilizar uma notação específica para se identificarem a si próprios e aos exercícios que estavam a resolver. O modo como resolviam os exercícios era totalmente da responsabilidade dos estudantes e o sistema não tinha qualquer tipo de intervenção no processo.

3.Estado da Arte

Nas secções seguintes, serão apresentados alguns dos trabalhos desenvolvidos nos últimos anos e que de algum modo se mostraram relevantes para o trabalho realizado, tendo em alguns casos servido de inspiração para as soluções que foram sendo desenvolvidas ao longo deste processo.

Por forma a auxiliar os professores no processo de ensino-aprendizagem de programação, principalmente no primeiro impacto, ao longo dos últimos anos foram apresentados os mais diversos tipos de soluções, que vão desde ferramentas apoiadas no uso de fluxogramas [45] [46], animações [47], simulações [47] até a ferramentas de gravação das aulas para posterior análise [48].

Principalmente devido à introdução de conceitos de programação numa fase muito preliminar da vida dos estudantes, nos dias que correm, estão disponíveis diversas ferramentas de programação gráfica, nomeadamente baseada em blocos [49] [50]. Alguns autores estudaram igualmente a utilização deste tipo de ferramentas em contextos de ensino superior enquanto ferramentas complementares [51].

Por forma a incrementar a motivação dos estudantes, foram igualmente introduzidos conceitos de gamificação tais como *leaderboards*, *badges*, missões ou sistemas de pontuação inclusivamente em ambientes de desenvolvimento mais tradicionais [52].

Muitas destas ferramentas funcionam como um complemento para exercitar e aprofundar conhecimentos relativamente a aspetos específicos do processo de aprendizagem de programação, sem que por vezes exista uma articulação bem definida com o decurso normal das aulas.

Outros autores focaram o seu estudo na avaliação e desenvolvimento de diferentes estratégias e metodologias de ensino para tornar o processo de ensino-aprendizagem de programação mais eficaz, mediante a diversificação das atividades propostas aos estudantes [53], a abolição de aulas teóricas tradicionais [54], a utilização de robôs [55] ou adoção de metodologias extremas baseadas no conceito de "aprender-fazendo" [56].

Todas estas abordagens e sistemas constituem um importante contributo para a melhoria do processo de aprendizagem de programação. No entanto, dado que foco deste trabalho se centra na aquisição de informação sobre o desempenho dos estudantes

enquanto resolvem os exercícios propostos e sua disponibilização aos professores, optou-se nesta secção por prestar especial atenção aos sistemas que de algum modo e em algum momento serviram de suporte e/ou inspiração para o trabalho desenvolvido.

3.1. Tutores inteligentes

Uma abordagem interessante, especialmente para cenários de aprendizagem mais autónoma, consiste na criação de sistemas com capacidade de autonomamente acompanharem o processo de aprendizagem dos estudantes. A estes sistemas dá-se usualmente a designação de **Tutores Inteligentes (*Intelligent Tutoring Systems*).**

No essencial, muitas das arquiteturas apresentadas para a construção de tutores inteligentes acabam de algum modo por ser inspiradas pelo trabalho de Hartley e Sleeman [57]. Segundo estes autores um tutor inteligente é constituído por três componentes distintos: curriculum, comportamento dos estudantes e métodos de ensino.

Resumidamente, o componente de "curriculum" diz respeito aos conteúdos a serem apresentados aos estudantes. O componente "comportamento dos estudantes" está relacionado com percecionar qual o conhecimento que o estudante já possui e o modo como resolve determinados problemas. Finalmente, o componente "métodos de ensino" foca-se em determinar que ações adotar para apoiar o estudante, por exemplo, através da apresentação de conteúdos educativos que o ajudem a perceber melhor alguns conceitos. Um exemplo da aplicação desta arquitetura pode ser encontrado no trabalho de Dağ e Erkan [58] representado na Figura 5)

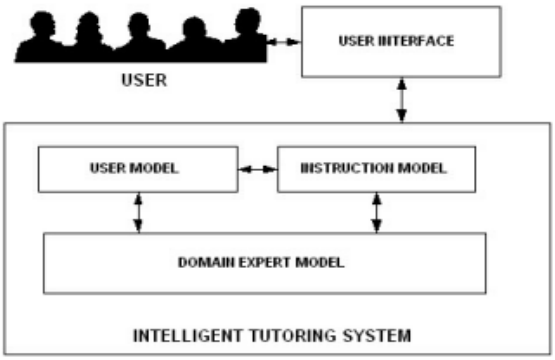

Figura 5 - Arquitetura de tutor inteligente proposta por Dag e Erkan [58]

Como pode ser observado na Figura 5, houve lugar a alterações dos nomes dos componentes propostos por Hartley e Sleeman, no entanto, as funcionalidades

mantiveram-se. A principal alteração consistiu no acrescento de um novo componente - *user interface* - que fica encarregue de todas as comunicações em ambos os sentidos entre o tutor inteligente e os utilizadores.

Outros exemplos de arquiteturas de tutores inteligentes se sucederam, como por exemplo a arquitetura do sistema **CIMEL-ITS** [59], que pode ser observada na Figura 6. Uma vez mais, recorre-se a uma arquitetura assente nos três componentes base, aos quais se juntam outros componentes.

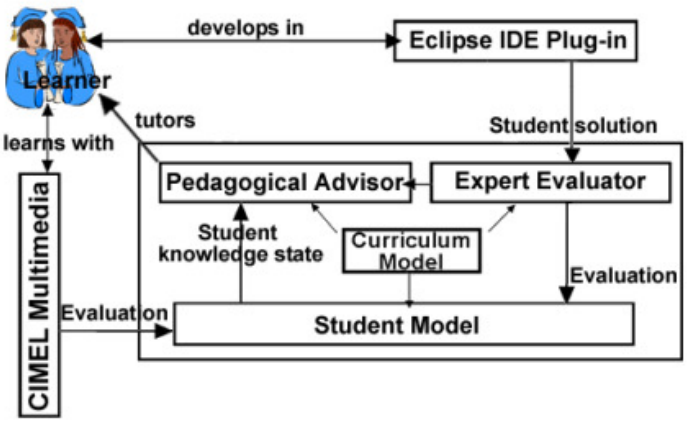

Figura 6 - Arquitetura do sistema CIMEL-ITS [59]

Outra arquitetura usualmente considerada como uma referência, foi proposta em 1996 por Joseph Beck et al. [60]. Segundo os autores, um tutor inteligente deve ser constituído por cinco componentes distintos, tal como ilustrado na Figura 7. Uma vez mais, encontram-se os três componentes essenciais propostos por Hartley e Sleeman, aos quais se juntam mais dois componentes.

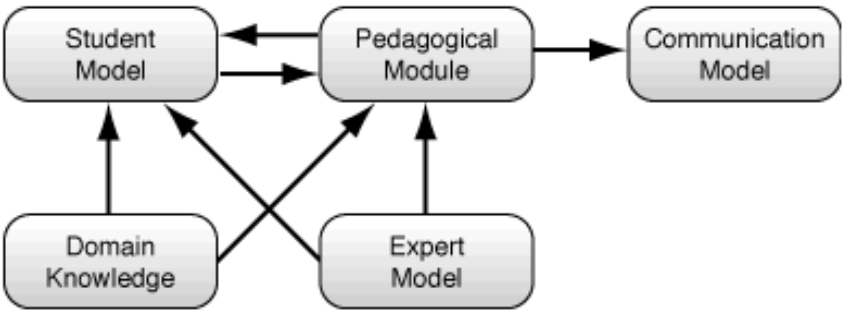

Figura 7 - Arquitetura proposta por Joseph Beck et al. [60]

Segundo van Lehn [61], é possível olhar para os tutores inteligentes segundo duas perspetivas distintas dependendo do momento em que são utilizados: *outer loop* e *inner loop*.

Os tutores inteligentes utilizados no *outer loop*, geralmente designados por *curriculum sequencing* ou *instructional planning* têm como principais funções decidir qual a tarefa que o estudante deve fazer em seguida de modo a completar um determinado problema [62]. Tal decisão poderá ser feita de forma estática, isto é, pode existir à partida uma lista de tarefas com uma ordem pré-definida e o sistema tratará apenas de apresentar ao estudante a tarefa seguinte ou, em alternativa, poderão ser utilizados mecanismos mais avançados que permitam apresentar ao utilizador um conjunto de tarefas que melhor permitam àquele estudante adquirir determinadas competências. Os tutores mais avançados tomam geralmente as decisões sobre quais as tarefas que devem ser apresentadas aos estudantes com base em modelos de estudantes dinâmicos que representam em cada momento o que o estudante sabe e o que já fez para saber o que sabe. Como exemplos concretos deste tipo de abordagens podem ser considerados os sistemas ***Stanford BIP*** [63] e **ANDES** [64].

No segundo grupo de tutores, denominados de tutores de *inner loop*, encontram-se os tutores cuja função consiste em apoiar o desenrolar das tarefas em si. Tutores deste tipo oferecem aos estudantes funcionalidades como fornecimento de *feedback* sobre um determinado aspeto (ou seja, indicam se algo está certo ou errado); apresentação de *feedback* mais informativo sobre um determinado erro que possa estar a ocorrer e informações sobre como ultrapassá-lo; apresentação de pistas sobre o que o estudante deverá fazer em seguida para completar a tarefa atual; avaliação de conhecimentos; revisão das soluções encontradas, etc.

Em muitos dos casos, um tutor inteligente pode apresentar funcionalidades de suporte tanto do ***inner loop*** como do ***outer loop***. O sistema **ANDES** anteriormente referido, por exemplo, apresenta também capacidades de tutoria ao nível da tarefa. O sistema **Cognitive Tutor** [65] é outro exemplo representativo deste tipo de abordagens.

De especial interesse para o presente trabalho, são os tutores inteligentes de programação (***programming intelligent tutoring systems)***, ou seja, tutores inteligentes utilizados no contexto particular do ensino da programação.

Um dos primeiros sistemas deste tipo a surgir foi apresentado por Anderson e Skwarecki em 1986 [66] e cujo objetivo era suportar o processo de aprendizagem da linguagem LISP através da monitorização do desempenho do estudante e apresentação de ajuda sempre que pertinente. Como exemplos adicionais é possível considerar os sistemas: **BITS** [67] - tutor inteligente da linguagem C++; **Play-to-Program** [68] - tutor inteligente para a

linguagem *Phyton*; **JITS (*Java Intelligent Tutoring System*)** [69] - tutor inteligente para a linguagem Java.

3.2. *Inteligent Teaching Assistants*

Estudos experimentais conduzidos por Bloom [28] apontam que, apesar de terem sido feitos consideráveis esforços no desenvolvimento de tutores inteligentes, o processo de aprendizagem com um tutor humano (professor) continua a produzir melhores resultados do que quando são utilizados tutores automáticos.

Posteriormente, van Lehn [70] conduziu uma série de experiências em que foram confrontados os resultados obtidos por estudantes acompanhados por professores humanos e estudantes acompanhados por diversos tipos de tutores inteligentes. Neste estudo, van Lehn apresenta um conjunto de hipóteses para explicar o facto de que, por mais tecnologicamente avançados que sejam, os tutores inteligentes ainda não conseguem permitir atingir os mesmos resultados que os professores humanos. Se, por um lado, os sistemas automáticos têm mais capacidade para aferir/modelar com detalhe os conhecimentos e problemas dos estudantes, por outro lado os professores têm melhores capacidades na parte associada com a apresentação de *feedback* e motivação dos mesmos.

Também Raabe et al. [30] defendem que o bom desempenho dos estudantes está intimamente ligado à capacidade que os professores têm de detetar a tempo as suas dificuldades de modo a conseguirem intervir em tempo útil.

Torna-se, pois, evidente que a melhor solução passará eventualmente por recorrer a abordagens que, de algum modo, consigam conciliar o melhor dos dois mundos. Assim, paralelamente ao desenvolvimento dos tutores inteligentes, foram sendo desenvolvidos outros tipos de sistemas, cujo objetivo principal passa por constituírem um aliado importante para os professores: os designados ***Inteligent Teaching Assistants*** [71].

Em termos gerais, a arquitetura deste tipo de sistemas difere da arquitetura de um tutor inteligente apenas na componente "pedagógica", ou seja, na componente respeitante às ações que devem ser tomadas (recomendação de tarefas a fazer, por exemplo).

Tal como nos tutores inteligentes, devem existir componentes que permitam **aferir/avaliar os conhecimentos** atuais dos estudantes e componentes que permitam **representar o conhecimento dos estudantes (modelos dos estudantes)**. A

principal diferença consiste no facto destes sistemas servirem geralmente para monitorizar o desempenho dos estudantes e eventualmente emitir alertas para os professores. Caberá sempre ao professor decidir no momento quais as ações que devem ser adotadas em cada situação com base na informação de que dispõe, de modo a conseguir melhorar o desempenho dos estudantes.

Sendo os estudantes a principal fonte de informação necessária para suportar um sistema deste tipo, será sempre necessário arranjar estratégias que façam com que estes se interessem por utilizar os sistemas, ou seja, será necessário arranjar uma forma de tornar o sistema útil de imediato para o estudante e atrativo, através da apresentação de *feedback* informativo, por exemplo.

No essencial, é necessário que o estudante sinta que utilizar um determinado sistema é uma mais-valia e que quanto mais o utilizar mais tem a ganhar. Este facto é de extrema importância, pois o estudante não deverá sentir que ao utilizar o sistema está apenas a contribuir para que o professor saiba mais informações sobre ele, sem que o estudante ganhe algo em troca no curto-prazo.

Este sentimento de "vigilância" por parte dos professores é algo que se não for bem acautelado poderá constituir um certo entrave por parte dos estudantes, pelo que desde o início deverá ser claro para ambas as partes qual a finalidade da recolha dos dados.

Como exemplos de ferramentas de apoio ao ensino é possível considerar o sistema **Logic-ITA** [72] cujo objetivo é apoiar o professor no ensino de lógica e que surgiu como evolução natural do tutor inteligente **Logic Tutor** [73]. Na Figura 8, está representada a arquitetura geral deste sistema.

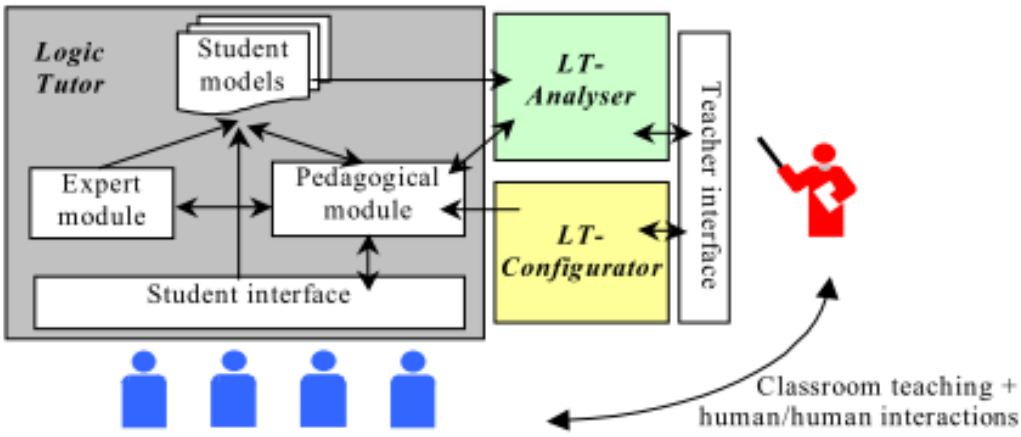

Figura 8 - Arquitetura geral do sistema Logic-ITA [72].

Neste caso concreto é curioso verificar que o sistema de apoio ao professor surge de certo modo acoplado a um tutor inteligente. Na prática, os estudantes obtêm *feedback* automático sobre o seu desempenho pelo tutor, tendo sempre o professor a monitorizar o que está a acontecer, com a possibilidade de em qualquer momento poder intervir caso seja necessário. A intervenção do professor surge basicamente sob a forma da reconfiguração de alguns aspetos do tutor inteligente que se mostrem pertinentes.

Uma arquitetura semelhante foi proposta por Raabe e Giraffa para o sistema **ALICE (*Algorithm Learning Internet based Computer Environment*)** [74] em que o professor é visto como um mediador do processo de aprendizagem levado a cabo por um tutor inteligente. Tal como no caso anterior, a sua intervenção consiste na introdução de informações que contribuam para o melhor funcionamento do sistema. A componente de apoio ao ensino manifesta-se sobretudo na monitorização do desempenho dos estudantes e na identificação de eventuais problemas de aprendizagem que possam existir. Convém realçar que este sistema, incluindo a parte de tutoria inteligente, foi pensado para funcionar como complemento de aulas presenciais.

O sistema **AVA,** proposto por Raabe e Silva [30], também tem como objetivo acompanhar o desempenho dos estudantes de modo a identificar eventuais dificuldades que os estudantes possam sentir. O acompanhamento é feito tendo por base questionários que são frequentemente apresentados aos estudantes.

Chen e Wasson [75] propuseram a utilização de um *instructional assistant agent* no âmbito de um sistema de aprendizagem colaborativa. Tal como nos outros casos, este sistema apoia a atividade do professor através da recolha de informação sobre o modo como o processo de aprendizagem está a decorrer. Esta informação é posteriormente analisada e, caso seja detetado algum tipo de problema, o sistema reporta o sucedido ao professor de modo que este possa agir em conformidade.

A solução proposta por Geoffroy et al. [76] assenta na partilha de tarefas entre sistemas automatizados e o professor num ambiente de ensino de engenharia eletrotécnica. O sistema automatizado tem como principais tarefas fazer a correção automática de exercícios e preparar o feedback a ser apresentado aos estudantes. Quer num caso, quer no outro, a decisão final sobre o *feedback* que será apresentado aos estudantes, assim como as notas atribuídas às resoluções dos exercícios, caberá sempre ao professor.

Uma abordagem totalmente diferente é adotada no sistema **PETCHA** [77]. Neste caso o sistema assiste o professor apenas na fase de autoria dos exercícios a serem propostos

aos estudantes. Com base na informação inserida pelos professores, será posteriormente apresentado automaticamente *feedback* aos estudantes.

Mais recentemente, em certa medida motivados pela utilização generalizada de plataformas de videoconferência como o ***Zoom***, o ***Google Meet*** ou o ***Microsoft Teams*** por causa da pandemia de COVID-19, Zhou [78] propôs a utilização de um sistema inteligente capaz de monitorizar os alunos através da câmara dos seus computadores e alertar o professor quando se verificam situações de distração recorrente por parte dos alunos, sugerindo nestas situações a adoção de medidas específicas para a situação em concreto.

3.3. *Auto-graders*

Uma tarefa muito importante, mas fastidiosa, levada a cabo pelos professores consiste na avaliação dos exercícios de programação que são apresentados ao longo do semestre/ano aos estudantes.

Teoricamente, quanto mais exercícios forem dados aos estudantes, mais facilmente se consegue aferir as suas reais capacidades e torna também possível o fornecimento de *feedback* mais detalhado e assertivo. No entanto, tem como principal inconveniente o facto de quantos mais exercícios se derem aos estudantes para estes fazerem, mais exercícios os professores terão de corrigir, facto este que a partir de dada altura poderá começar a ser incomportável [79].

Para além disso, a própria tarefa de corrigir exercícios de programação pode ser algo subjetiva, uma vez que um determinado professor poderá valorizar mais alguns aspetos do que outros, conduzindo ao facto de diferentes professores poderem avaliar de formas totalmente distintas um mesmo exercício, utilizando diferentes critérios de avaliação. Há ainda que ter atenção que até o mesmo professor poderia avaliar o mesmo exercício de forma diferente, dependendo de diversos fatores, como a tipologia de estudantes ou o momento da entrega dos mesmos.

Assim, com o aumento da disponibilização de cursos de programação em contexto de MOOC, mas também devido ao crescente número de estudantes a frequentar unidades curriculares de programação, que não tem sido acompanhado por um equivalente aumento no número de professores, uma área que tem vindo a registar consideráveis avanços é a dos sistemas de avaliação automática de exercícios de programação,

geralmente designados de "*auto-graders*" ou "*automatic assessment tools*". A principal vantagem da utilização destes sistemas passa pela eliminação/minimização de algumas das limitações anteriormente apresentadas em relação à correção "manual" dos exercícios.

Um dos primeiros sistemas deste tipo a serem apresentados, foi o sistema ***Rensselaer grader*** [80] utilizado em computadores IBM 650[10] e cujo desenvolvimento foi à altura largamente impulsionado pelo aumento súbito do número de estudantes. Citando o autor:

> "*The grader easily justifies itself on economic grounds. It accomplishes more than savings in time and money; it makes possible the teaching of programming to large numbers of students. This spring we had 80 students taking a full semester course in programming; over 120 are expected next spring. We could not accommodate such numbers without the use of the grader.*"

Pelo exposto é, pois, possível observar que o problema de ter de lidar com um número elevado de estudantes não é propriamente novo, no entanto, continua a ser tão válido atualmente, como era há aproximadamente 60 anos.

Em termos gerais, é possível dividir os sistemas de avaliação de exercícios de programação em dois grupos: soluções de *black-box* e soluções de *white-box*.

No primeiro caso encontram-se aqueles sistemas em que *à priori* são indicados conjuntos de *inputs* e respetivos *outputs* esperados. Um sistema deste tipo limita-se a verificar se para um determinado *input* se obteve o *output* esperado, independentemente do modo como a solução foi implementada. Em [81] encontra-se um estudo muito exaustivo sobre este tipo de abordagens realizado por Douce.

Como referido por Tang et al. [82], este tipo de abordagens podem conduzir a situações de confusão e frustração por parte dos estudantes, pois na maior parte dos casos os exercícios são considerados certos se e só se o resultado obtido for exatamente igual ao esperado, bastando muitas das vezes um espaço em branco a mais ou a menos nos resultados para que uma solução não seja considerada correta, muito embora toda a

[10] https://en.wikipedia.org/wiki/IBM_650

lógica de programação esteja certa. Por outro lado, podem existir soluções ineficientes e baseadas em código mal estruturado, mas cujo resultado final esteja igual ao desejado e, como tal, serão consideradas como corretas.

Para um determinado problema poderão existir diversas soluções, com diferentes implicações em termos de consumo de recursos, e também com diferentes níveis de complexidade, pelo que considerar apenas o resultado final poderá ser limitativo.

Tal como no "mundo real", os professores não devem basear as classificações dos trabalhos submetidos pelos estudantes simplesmente no facto de permitirem obter ou não o resultado esperado, também é pertinente basear as classificações dos exercícios em outros aspetos como a qualidade e complexidade do código escrito. Assim surgem as chamadas soluções de "caixa aberta" em que a classificação é obtida em função do que é possível concluir da observação do código fonte submetido para avaliação eventualmente em conjugação com o resultado obtido.

Segundo Stamouli e Huggard [83], um programa é considerado correto quando se verifiquem os seguintes quatro aspetos:

- Sintaxe correta (não existirem erros de compilação);
- Funcionalidades implementadas (os requisitos serem cumpridos);
- Estrutura correta que promova reutilização, extensão e manutenção;
- Desempenho e validação de entradas e saídas.

Todo e qualquer sistema automatizado que consiga aferir a validade destes quatro aspetos conseguirá fornecer mais informações sobre as capacidades de programação dos estudantes do que os sistemas do tipo *black-box* anteriormente apresentados.

Piech et al. [84] recorrem a três técnicas diferentes com vista a avaliar automaticamente os exercícios submetidos pelos estudantes: *bag of words*, *Abstract Syntax Tree (AST) Change Severity* e *Application Program Interface (API) Call Dissimilarity*

A primeira técnica, designada por ***bag of words,*** consiste em fazer uma comparação entre o código fonte da solução submetida pelo estudante e uma solução definida pelo professor, com o objetivo de verificar qual o grau de semelhança entre ambos os ficheiros. Como seria de esperar, esta técnica embora seja a mais fácil de implementar, acaba também por ser a que obtém piores resultados.

A segunda técnica designada por ***Abstract Syntax Tree (AST) Change Severity*** consiste em avaliar o número de alterações que a AST da solução submetida tinha de

sofrer de modo a ficar igual à AST da solução definida pelo professor. Quanto menos alterações forem necessárias, melhor será a solução submetida pelos estudantes.

Finalmente, a terceira e última técnica designada por ***Application Program Interface (API) Call Dissimilarity*** foi a que melhores resultados permitiu obter. Em traços gerais, esta técnica consiste em executar os programas submetidos com um conjunto de *inputs* predefinidos, registando a sequência de *API calls* (chamadas) efetuadas. Esta sequência de chamadas é posteriormente comparada com a sequência de chamadas associada a uma solução definida pelo professor.

Em todas estas três abordagens existe sempre a necessidade de comparar as soluções submetidas pelos estudantes com uma solução definida pelo professor. É, no entanto, sabido tal como já mencionado anteriormente, que para um problema poderão existir diversas soluções possíveis e, em muitos casos, até mesmo soluções que o professor não tenha previsto. Para além disso, este trabalho baseou-se num tipo muito particular de exercícios, não havendo informações sobre a sua aplicabilidade a um leque mais abrangente de exercícios

Para além de verificar se os programas devolvem os *outputs* esperados, o sistema de avaliação de exercícios **GAME** proposto por Blumenstein et al. [85] recorre também à verificação do seguinte conjunto de aspetos: indentação do código fonte; quantidade e qualidade dos comentários existentes no código fonte; estilo de programação (verificação de valores *hard coded*, capacidade de reutilização, extensão e manutenção do código fonte). Na

Figura 9 pode ser observado um esquema representativo de todos os aspetos tidos em conta aquando da avaliação de um exercício.

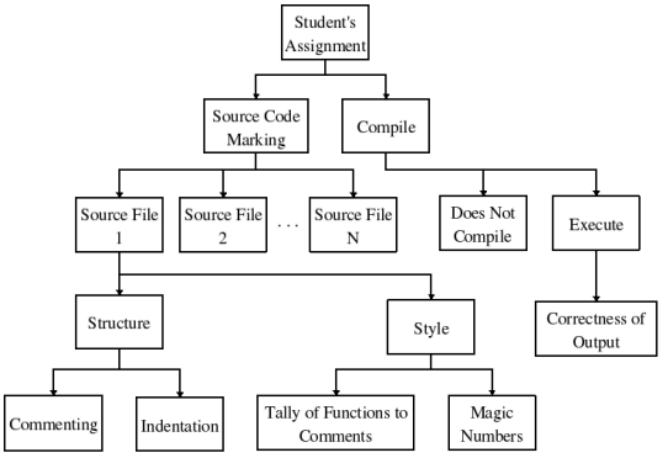

Figura 9 - Representação hierárquica do sistema GAME [85].

Um conjunto de testes levado a cabo pelos autores permitiu concluir que o sistema foi capaz de avaliar os exercícios de uma forma muito satisfatória, produzindo praticamente as mesmas avaliações que um professor humano.

Uma abordagem semelhante foi seguida por Juedes [86] que, para além de um conjunto de casos de teste, recorre também a métricas relacionadas com erros de *design* e quantidade e qualidade da documentação.

Moreira et al. [87] conduziram um estudo onde avaliaram o desempenho de duas abordagens diferentes de avaliação automática de exercícios utilizando regressão linear múltipla: **baseada em métricas de engenharia de *software*** (foco na complexidade dos programas) e baseada no uso de ***n-grams*** (foco na similaridade estrutural). No primeiro caso são utilizadas métricas como **número de funções**, **número de linhas**, **número de palavras reservadas**, **complexidade de McCabe** [88], **volume de Halstead** [89]**,** etc. A abordagem baseada em ***n-grams*** consiste em dividir objetos, textos por exemplo, em pedaços de ***n*** caracteres que posteriormente podem ser comparados utilizando medidas de semelhança vetorial. Esta comparação dá uma ideia clara sobre a semelhança entre dois textos, neste caso concreto, dois ficheiros de código (a solução submetida pelos estudantes e a solução "certa" definida pelo professor).

No caso concreto deste estudo, os resultados mostraram que a técnica baseada na utilização de ***n-grams*** apresentava um desempenho melhor. Os autores afirmaram também que para obter melhores resultados esta abordagem deveria ser aliada à utilização de um conjunto de casos de teste.

Naudé et al. [90] propuseram uma nova medida de semelhança entre grafos denominada **AssignSim** para comparar as semelhanças entre as soluções entregues pelos estudantes e a solução definida como correta pelo professor

Também os sistemas ***Ceilidh*** proposto por Higgins et al. [91] e ***ASSYST*** proposto por Jackson et al. [92] baseiam o seu funcionamento em abordagens do tipo "caixa-fechada". O sistema ***Mooshak*** [93] contribuiu também para a popularização da utilização deste tipo de sistemas, principalmente pelo facto de permitir analisar programas completos (e complexos) e por ter sido utilizado em diversos concursos de programação.

Recorrendo a uma abordagem diferente, o sistema ***Scheme-robo*** [94]**,** desenvolvido para analisar exercícios escritos na linguagem de programação *Scheme*, apenas permite avaliar um procedimento de cada vez, mas, em contrapartida, oferece funcionalidades bastante pertinentes como por exemplo um sistema anti plágio.

O sistema Web-CAT [95], desenvolvido na Virginia Tech, foi originalmente desenvolvido como ferramenta de suporte a uma unidade curricular sobre *Test Driven Development*. Tinha como principal objetivo fornecer aos estudantes uma forma prática de testarem o seu código fonte com um conjunto de testes disponibilizado pelos professores, mas também para testarem os testes que eles próprios tinham escrito. Ainda que originalmente fosse necessário que os estudantes acedessem a uma página web e fizessem manualmente o *upload* de um ficheiro ZIP ou JAR, por exemplo, contendo o código desenvolvido, nas versões mais recentes deste sistema já pode ser utilizado um ambiente integrado de desenvolvimento, neste caso, o Eclipse. Em todo o caso, independentemente do modo como esta é efetuada, após a submissão do código é apresentada ao estudante uma página web com informação detalhada sobre a avaliação automática do código, tal como pode ser observado na Figura 10.

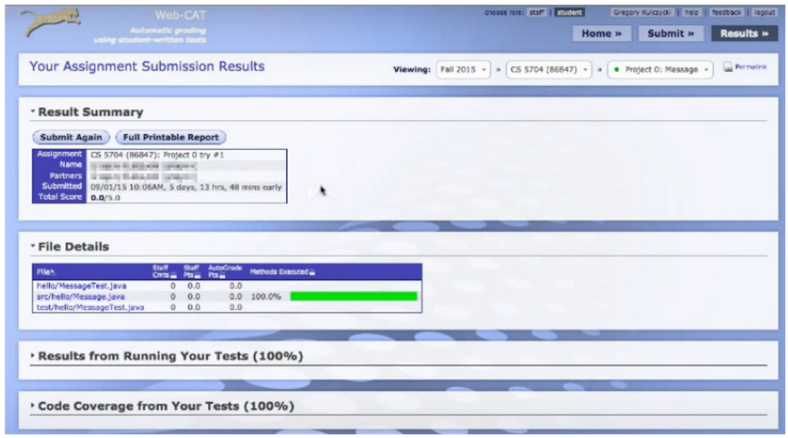

Figura 10 - Exemplo do relatório de avaliação de um exercício do sistema Web-CAT [95].

Entre as múltiplas funcionalidades disponibilizadas, é possível encontrar a configuração das regras de avaliação dos exercícios, em que é possível definir várias componentes, algumas das quais automáticas e outras que dependem da intervenção manual dos professores. Para além disso, permite ainda conjugar funcionalidades de análise do tipo *black-box* com funcionalidades do tipo *white-box*, como por exemplo ferramentas de análise de estilo de escrita de código.

Apesar de se tratar de um sistema muito completo, extensível e com capacidade de lidar com projetos bastante complexos, a sua configuração/utilização quer para os professores, quer para os estudantes é demasiado complexa e morosa, principalmente para situações em que é disponibilizada uma quantidade considerável de exercícios para resolução todas as semanas. Tendo em consideração os próprios motivos pelos quais foi criado, a sua utilização adequa-se sobretudo a projetos/trabalhos finais com alguma dimensão. Esta

situação levaria os autores a criar um sistema alternativo denominado ***CodeWorkout*** [96] destinado a ser utilizado para a resolução de pequenos exercícios de treino.

O sistema disponibiliza exercícios com um conjunto de casos de teste previamente definidos, entre os quais os professores podem escolher e associar a um planeamento definido pelos próprios. Ainda que no *site* seja referido que o próprio professor pode criar os seus exercícios e respetivos casos de teste, aquando do teste efetuado ao sistema esta funcionalidade não estava disponível.

Os estudantes podem utilizar o sistema sem estarem registados, ou então no âmbito de um curso, e para este efeito já será necessário estar registado e associado a esse mesmo curso. Neste último caso, o professor consegue ter acesso à última tentativa de resolução dos exercícios por cada estudante. Aqui terá acesso a informação muito limitada, nomeadamente, o código fonte em si, a data e hora em que a tentativa foi realizada, o conjunto de casos de teste realizados e uma nota obtida em função do número de casos de teste em que passar. Um exemplo desta página pode ser observado na Figura 11.

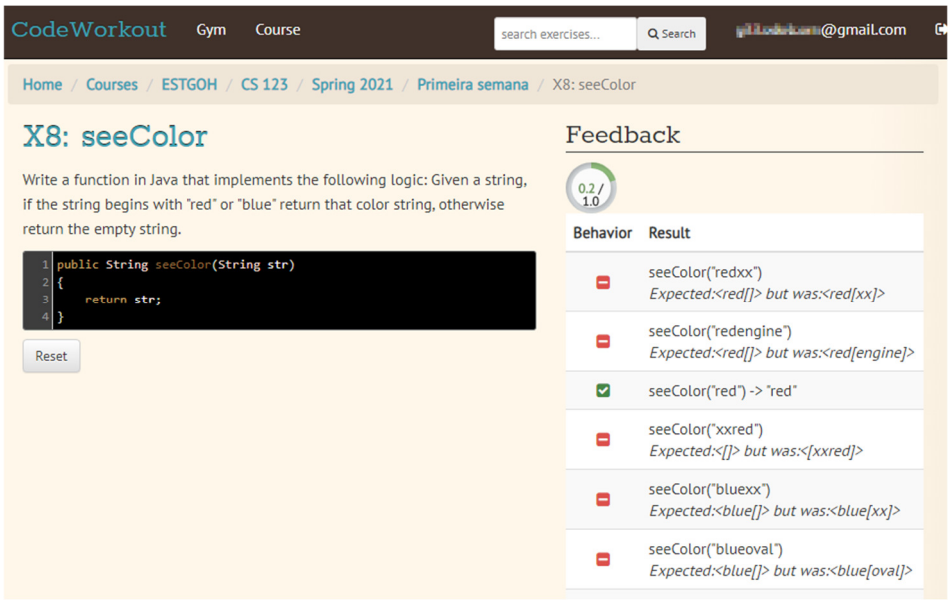

Figura 11 - Exemplo de página com detalhes de uma tentativa de resolução no sistema Web-CAT

Uma página semelhante a esta é utilizada pelos estudantes para fazer a submissão das suas soluções, sendo a única diferença o facto de poderem editar o código e ser disponibilizado um botão "Check my answer!".

Apesar de, no essencial, o sistema estar a funcionar, a informação disponibilizada aos professores é ainda demasiado limitada, apresentando ainda alguns problemas a nível da interação com o utilizador.

No caso do sistema ***TestMyCode*** proposto por Vihavainen et al. [97], juntamente com os enunciados dos exercícios, são fornecidos aos estudantes *templates* dos projetos para o *Integrated Development Environment* (*IDE*) *NetBeans* utilizado para a resolução dos exercícios utilizando a linguagem Java. Estes *templates* incluem o "esqueleto" das classes necessárias para a resolução dos exercícios, colocando em alguns casos o código praticamente todo, deixando apenas algum pormenor em aberto para os estudantes corrigirem/completarem. Para além disso são também disponibilizados os testes unitários que permitirão determinar se o resultado está ou não correto.

Este tipo de abordagem tem como grande vantagem o facto de se conseguir determinar facilmente se o resultado está correto ou não, uma vez que são fornecidos os testes unitários e informação sobre o modo como os *inputs* e *outputs* são definidos. Para além disso, como é possível observar, é fornecido algum *feedback* ao estudante em função dos erros que o ser código possa eventualmente ter. Para além deste feedback imediato, os estudantes podem igualmente solicitar aos professores *feedback* relativo à resolução de um determinado exercício.

Um dos principais aspetos menos positivos deste tipo de abordagem prende-se com o facto de ser necessário muito trabalho extra de preparação dos exercícios por parte dos professores. Outras limitações importantes residem em ser relativamente complexo de utilizar/configurar do ponto de vista dos professores e principalmente, a pouca quantidade e a forma da informação que lhes é disponibilizada em relação ao progresso dos estudantes. Na Figura 12 é apresentado um exemplo da página de apresentação do progresso dos estudantes.

Figura 12 - Visualização do progresso dos estudantes no sistema TestMyCode [97].

Mendes et al. [98] apresentaram o sistema **ELJT (*E-Learning Java Trainer*)** destinado a suportar a aprendizagem de programação da linguagem Java, no qual os professores podem inserir um conjunto de exercícios que os estudantes deverão resolver e acompanhar o progresso dos estudantes. Na prática o professor consegue saber quais os

estudantes que tentaram resolver cada um dos exercícios, se tiveram ou não sucesso e quais os erros mais comuns.

Um outro exemplo interessante é um *plug-in* para o ***Moodle*** chamado ***CodeRunner***[11] e que, em termos práticos, consiste em um novo tipo de pergunta que pode ser utilizada num *quiz* em conjunto com outros tipos de pergunta de resposta mais tradicional como escolha múltipla ou escrita. Este *plug-in* poderá ser utilizado para avaliar código escrito em diversas linguagens de programação e tipicamente espera-se que na pergunta seja fornecido aos estudantes uma descrição do que se pretende, ou seja, a resolução de um programa completo ou apenas uma função. Deverá igualmente ser fornecido um conjunto de exemplos de casos de teste nos quais o programa desenvolvido deverá passar. Na Figura 13 pode ser observado um exemplo da utilização deste sistema.

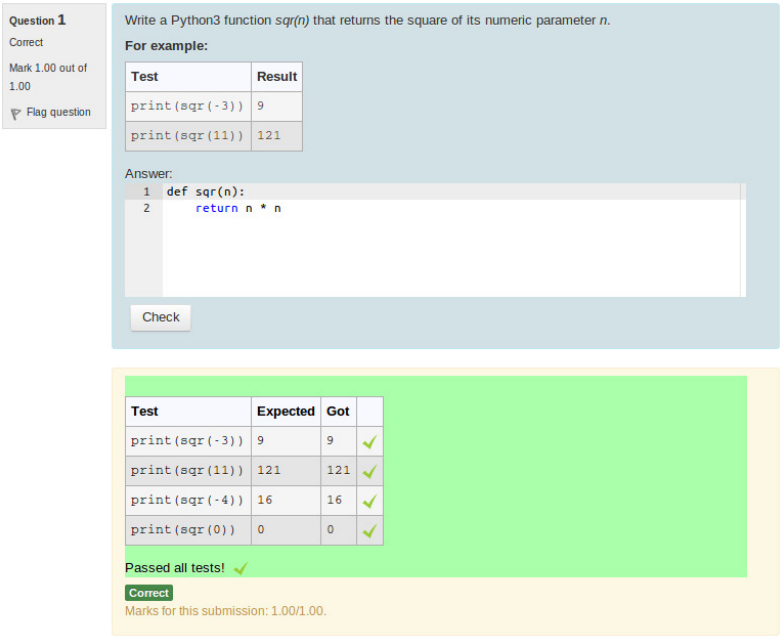

Figura 13 - Exemplo de utilização do CodeRunner [https://coderunner.org.nz/].

Uma característica interessante do sistema é o facto de o professor poder ele próprio definir o modo como o código será avaliado, isto é, poderá fazer um script que analise o código em busca de determinadas instruções potencialmente destrutivas e/ou não enquadradas com o momento em que a unidade curricular está a decorrer e, em função do resultado, apresentar uma mensagem personalizada ao invés de executar o código. Além disso, permite ainda a criação de exercícios utilizando dados gerados aleatoriamente.

[11] https://coderunner.org.nz/

Um outro exemplo de um plug-in para o Moodle muito semelhante ao ***CodeRunner*** é o ***Virtual Programming Lab (VPL)*** [99]. Obviamente, uma das principais vantagens destes sistemas é o facto de serem integrados em um dos mais populares LMS do mundo, facilitando deste modo a tarefa aos estudantes, mas também aos professores, uma vez que não necessitam de utilizar várias ferramentas.

O sistema ***Retina*** [100] proposto por Murphy et al. consiste em uma ferramenta que recolhe *snapshots* do código dos estudantes aquando da resolução dos exercícios propostos pelo professor utilizando Java. Pode ser utilizado via linha de comandos ou integrado nos ambientes de desenvolvimento Eclipse e BlueJ.

Os *snapshots* são enviados para um servidor central para serem processados sempre que os estudantes compilam o código. Se assim o entenderem, os estudantes poderão resolver os exercícios propostos sem utilizar o sistema e/ou utilizar o sistema anonimamente de forma que os *snapshots* sejam recolhidos e possam ser utilizados para a produção de informação agregada, não sendo, contudo, possível identificar de modo algum o autor destes. Em qualquer caso, após o *snapshot* ser processado no servidor, fica disponível para o professor alguma informação sobre todas as tentativas de resolução de um determinado exercício, o modo "*Browse*" representado na Figura 14, ou em alternativa pode aceder a informação agregada sobre uma determinada turma, o modo "*Class*", onde tem acesso aos 10 erros mais frequentes e ao tempo gasto por cada estudante e a média de todos os estudantes na resolução de cada exercício.

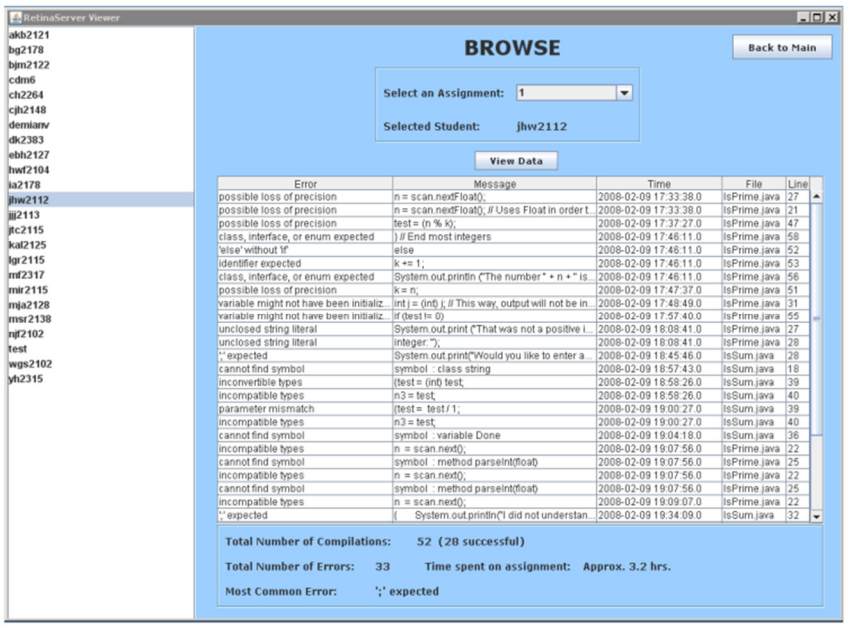

Figura 14 - Exemplo do modo "browse" do sistema Retina [100].

Para os estudantes, é também disponibilizada alguma informação para cada exercício, como por exemplo, o número de compilações efetuadas pelo estudante (e o valor médio da turma) ou o tempo gasto pelo estudante (e o valor médio da turma). É ainda apresentada uma previsão do tempo que irá demorar a resolver o exercício seguinte, tendo por base o tempo que gastou para resolver o atual. Finalmente, é disponibilizado aos estudantes um histórico dos erros de compilação para aquele exercício.

Uma funcionalidade interessante deste sistema é poder disponibilizar um *bot* de conversação que envia mensagens para sistemas de *instant messaging* como o entretanto extinto *Google Talk* sempre que se verificarem algumas situações concretas, nomeadamente: a existência de um número elevado de tentativas de resolução com erros de compilação efetuadas entre as 23h e as 6h; caso o estudante passe demasiado tempo a trabalhar no mesmo exercício; e finalmente, caso o estudante obtenha o mesmo erro de compilação recorrentemente. Para além disso, os estudantes poderão também enviar mensagens para o *bot* com um conjunto de instruções predefinidas e com funcionalidades próprias tais como: "*recommend*", "*explain*", ou "*who*".

O sistema ***OK***[12] desenvolvido e utilizado principalmente na Universidade da Califórnia (Berkeley) oferece também aos estudantes a possibilidade de testarem automaticamente as soluções desenvolvidas para a resolução dos exercícios propostos pelos professores. Embora tenha sido desenvolvido originalmente para a linguagem *Python*, o sistema em si está preparado para funcionar com outras linguagens de programação. Para cada exercício, os professores deverão disponibilizar um conjunto de testes unitários, que serão utilizados no processo de avaliação das soluções dos estudantes.

Os estudantes poderão utilizar um qualquer editor de código para implementar as suas soluções, uma vez que para testar o código o sistema funciona principalmente em ambiente de linha de comandos. É, contudo, fornecida uma aplicação *web*, quer aos estudantes, quer aos professores. Nesta, os professores podem criar/disponibilizar os exercícios, assim como acompanhar o processo de submissão das soluções dos estudantes, nomeadamente, visualizar as tentativas de resolução efetuadas por cada estudante.

É ainda possível aos professores atribuir aos estudantes *feedback* personalizado relativamente a uma determinada tentativa de resolução e aceder a um conjunto de

[12] https://okpy.org/

estatísticas relativas ao modo como as submissões para um determinado exercício estão a decorrer, tal como pode ser observado na Figura 15.

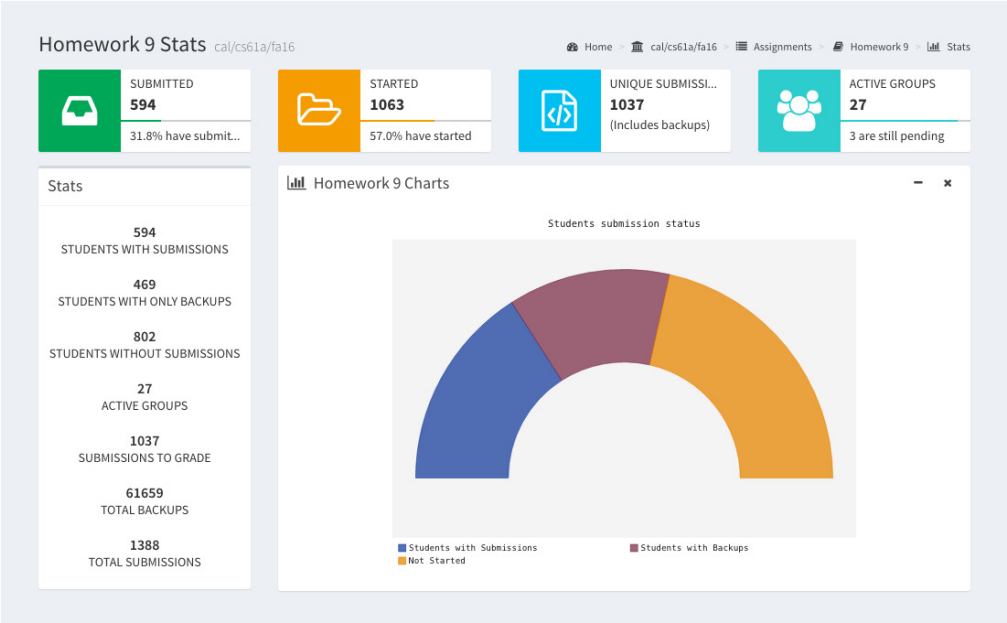

Figura 15 - Apresentação de estatísticas relativas às submissões de um exercício no sistema OK [http://okpy.org].

O sistema ***Spinoza*** proposto por Deeb et al. [101] é uma aplicação *web* na qual os professores fornecem uma série de problemas de programação para os estudantes resolverem. Em termos concretos, aos estudantes é disponibilizado um enunciado e eventualmente uma parte do código já escrito, devendo os estudantes completá-lo por forma a resolver o problema em questão. Ao pressionarem o botão "Run" o código escrito será testado com alguns casos de teste e os resultados são disponibilizados aos estudantes, tal como representado na Figura 16.

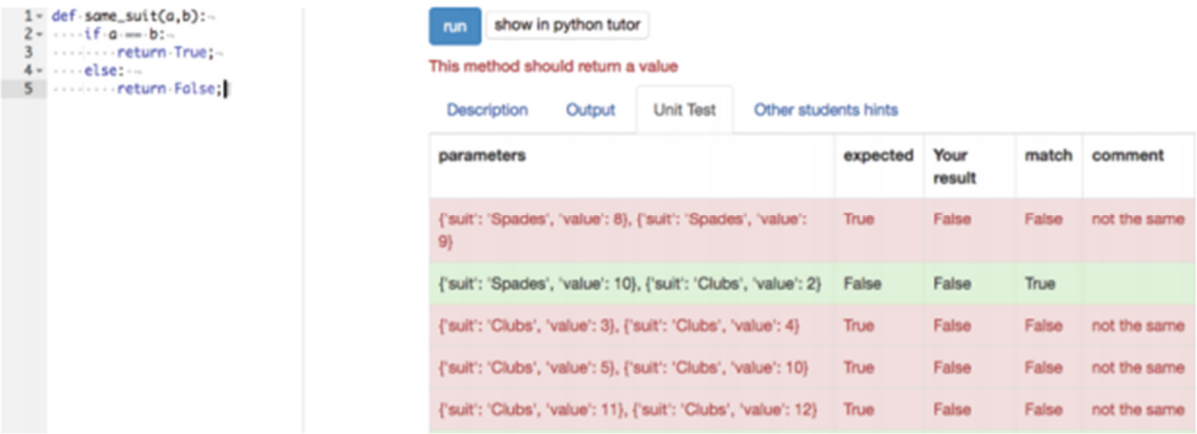

Figura 16 - Exemplo de teste do código desenvolvido no sistema Spinoza [101].

Ao professor são disponibilizadas um conjunto de ferramentas que permitem não só visualizar as últimas tentativas de resolução efetuadas pelos estudantes, mas ferramentas mais avançadas que lhes permitem acompanhar em tempo real o desempenho dos estudantes, como por exemplo o "Spinoza Markov Model" ou o "Engagement graph". São ainda disponibilizadas funcionalidades que permitem que os estudantes comentem as soluções dos seus colegas.

O sistema ***Edgar*** proposto por Mekterovic et al. [102] para a monitorização de processos mais formais de avaliação (*i.e.* testes ou exames) que envolvam a resolução de problemas de programação, proporciona aos professores diversas ferramentas que lhes permitam fazer um acompanhamento em tempo real do progresso dos estudantes, que poderão não se encontrar fisicamente no mesmo local (salas diferentes, por exemplo). Aos estudantes é disponibilizada uma página contendo o enunciado do problema a resolver e uma caixa de texto onde deverão escrever o código necessário para resolver o problema apresentado. Ao pressionarem o botão "Run" o código irá ser testado com um conjunto de casos de teste previamente definidos e o resultado destes testes será apresentado aos estudantes, tal como pode ser visto na Figura 17.

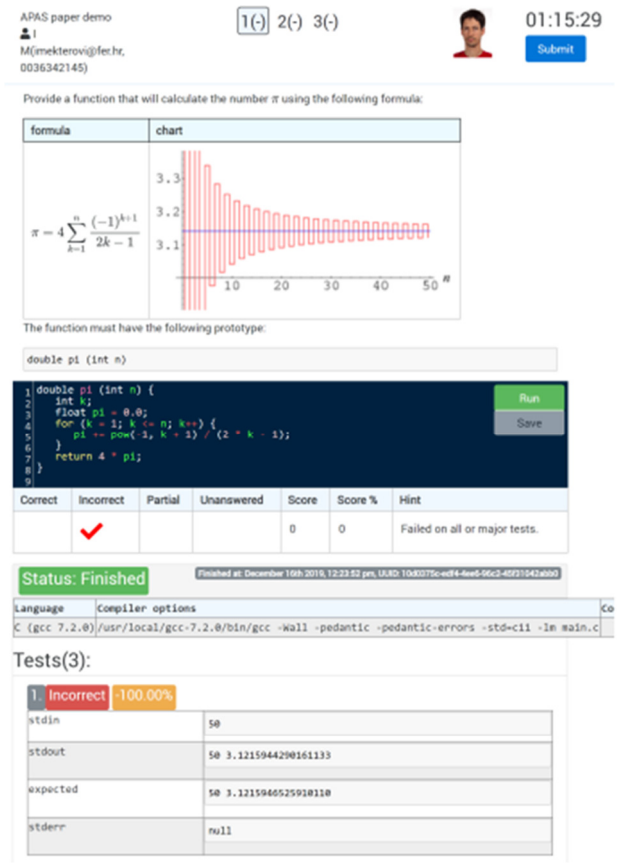

Figura 17 - Exemplo da página utilizada pelos estudantes para resolverem os problemas no sistema Edgar [102].

Como já foi referido anteriormente, a tutoria individualizada é aquela que permite obter melhores resultados em termos de aprendizagem, no entanto, há que ter em conta que existe um número muito maior de estudantes do que professores, pelo que tal processo de aprendizagem se torna muito difícil de pôr em prática quer num contexto de aula presencial, quer num contexto de ensino à distância, MOOCs (*Massive Open Online Course*), por exemplo em que o número de estudantes pode atingir as várias centenas.

Para atacar precisamente este problema, Guo [103] desenvolveu a ferramenta ***Codeopticon***, cujo objetivo principal passa por monitorizar em tempo real o código fonte que os estudantes estão a escrever, enquanto tentam resolver uma série de exercícios disponíveis em http://www.pythontutor.com para uma série de linguagens de programação (*Python, Java, JavaScript,* etc.).

Este sistema possibilita que um tutor possa acompanhar simultaneamente (e em tempo real) vários estudantes, podendo a qualquer momento interagir com eles através de um *chat* existente na aplicação.

Uma limitação deste sistema é o facto de os estudantes terem de programar diretamente no *browser* no *site* em causa, obrigando-os assim a abandonar o *IDE* (*Integrated Development Environment*) que normalmente utilizariam.

Dada a sua natureza, a utilização deste sistema torna-se bastante interessante, principalmente para o ensino à distância em que haja um elevado número de estudantes inscritos. Importa referir que não há lugar a qualquer tipo de acompanhamento do desempenho dos estudantes a longo prazo.

Muito embora gozem de elevadas taxas de aceitação por parte dos estudantes, um problema muitas vezes apontado a este tipo de sistemas é o facto de dificilmente conseguirem ultrapassar as paredes das instituições em que foram desenvolvidos [104]. Segundo Rößling et al., entre os motivos para este facto, encontra-se a questão de os professores terem a sua própria metodologia de trabalho e os seus próprios exercícios, não estando geralmente disponíveis para gastar tempo a adaptar-se a uma nova forma de fazer as coisas. Assim, muitas instituições acabam por desenvolver os seus próprios sistemas, de modo a dar resposta a necessidades próprias dessas instituições. Em todo o caso, quer seja porque os professores aceitem utilizar exercícios de uma base pré-estabelecida, ou se predisponham a gastar o tempo necessário para preparar novos exercícios, este cenário começa a mudar e sistemas como o ***Web-CAT*** ou ***TestMyCode***

são atualmente utilizados em diversas instituições de ensino pelo mundo fora, além daquela em que foram desenvolvidos.

Reconhecidas as vantagens e o aumento da utilização deste tipo de sistemas, nomeadamente por indivíduos autodidatas que não se encontrem inscritos num curso do ensino "tradicional", mas também por instituições de ensino, começaram deste modo a surgir diversos sistemas comerciais, principalmente *online*, com funcionalidades de *auto-grading* semelhantes às anteriormente descritas. O sistema ***Repl.it***[13] assume-se em primeira mão como uma ferramenta de suporte ao processo de programação de forma colaborativa, mas que também possui funcionalidades de *auto-grading*.

Também o ***GitHub*** disponibiliza funcionalidades de *auto-grading* através da sua plataforma ***Classroom***[14]. Neste caso, os professores deverão criar repositórios em que para cada exercício será fornecido o enunciado, o código dos testes unitários e eventualmente o "esqueleto" da solução. Posteriormente, deverão partilhar com os estudantes os endereços desses repositórios. Os estudantes poderão utilizar uma qualquer ferramenta que permita ligar ao servidor do ***GitHub*** para obter os ficheiros fornecidos pelos professores, resolver os exercícios e, de cada vez que fizerem *commit* da sua solução para o servidor, esta será avaliada automaticamente tendo os professores, após alguns minutos, acesso a informação detalhada sobre esse *commit* (o código fonte e o *log* de execução dos testes unitários). O sistema permite ainda que os professores forneçam *feedback* personalizado aos estudantes sobre uma determinada tentativa de resolução.

Esta abordagem, tal como muitas das anteriormente apresentadas, implica a realização de algum trabalho de preparação para cada exercício. Implica também que os estudantes estejam familiarizados com o funcionamento de ferramentas de controlo de versões de código, nomeadamente, **Git**[15], o que não sendo necessariamente um aspeto negativo, eventualmente pode não ser muito comum, principalmente em unidades curriculares introdutórias. Do ponto de vista do professor é fornecida informação limitada, que não vai muito além do histórico e resultado dos *commits* de cada estudante.

Embora tenha sido desenvolvido originalmente como uma ferramenta de suporte ao processo de avaliação de testes de unidades curriculares que não envolvam programação,

---

[13] http://repl.it
[14] https://classroom.github.com/
[15] https://git-scm.com/

o sistema ***Gradescope***[16] disponibiliza igualmente funcionalidades de *auto-grading* de exercícios de programação, que poderão ser conjugadas com a avaliação manual das soluções desenvolvidas pelos estudantes. Na Figura 18 pode ser observado um exemplo da utilização do sistema ***gradescope*** para avaliar a resolução de um exercício.

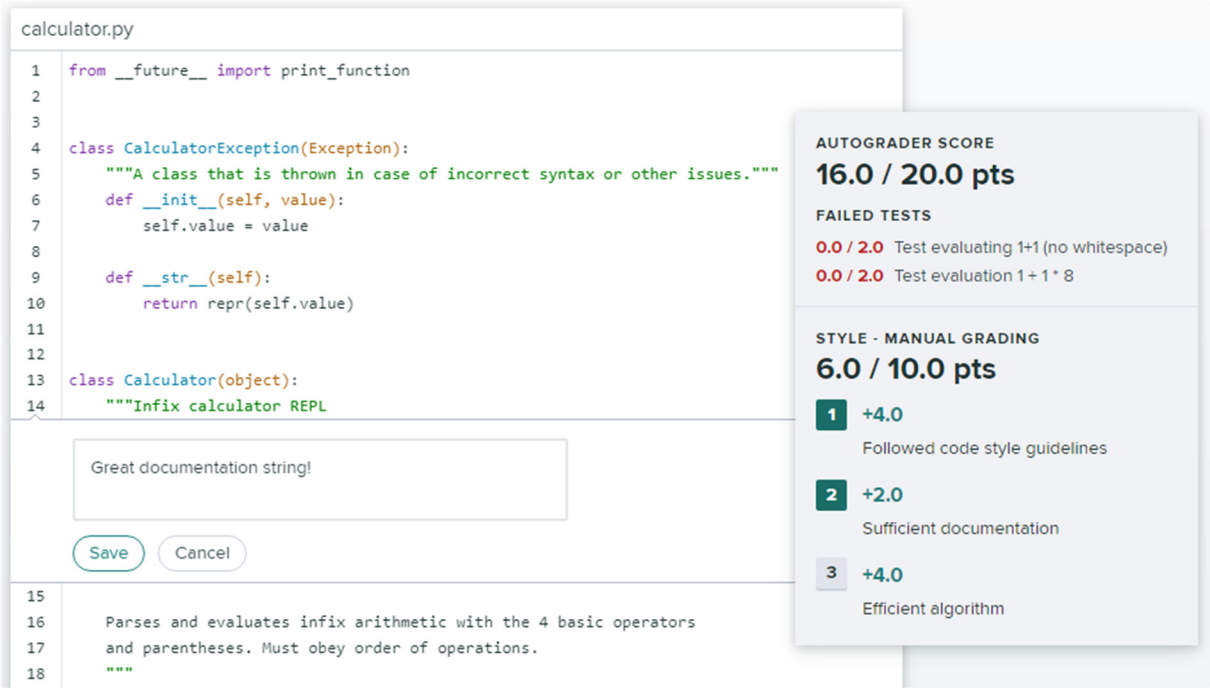

Figura 18 - Exemplo de utilização do sistema *Gradescope* [https://www.gradescope.com/]

Exemplos adicionais deste tipo de sistemas a nível comercial, são as ferramentas: ***Voccareum***[17], ***Codio***[18], ***Mimir***[19] ou ***CodeLab***[20].

Nos últimos anos surgiram algumas soluções interessantes ao nível da inclusão deste tipo de sistemas diretamente em manuais didáticos interativos, nos quais os estudantes podem testar os seus conhecimentos à medida que vão lendo os conceitos. Um dos exemplos mais populares deste tipo de sistemas são os livros disponibilizados pela ***ZyBooks***[21].

Tal como acontece relativamente a alguns dos sistemas provenientes da academia, no caso dos sistemas comerciais, é dada primazia ao processo de *auto-grading* em si e não a providenciar aos professores informação detalhada sobre o desempenho dos estudantes.

---

[16] https://www.gradescope.com/
[17] https://www.vocareum.com/
[18] https://codio.com
[19] https://www.mimirhq.com/
[20] https://www.turingscraft.com/
[21] https://www.zybooks.com/

Independentemente da natureza mais ou menos académica, em termos práticos todos os sistemas apresentam um conjunto de funcionalidades base muito semelhante, ou seja, permitem disponibilizar aos estudantes um conjunto de enunciados de exercícios que estes deverão tentar resolver e que serão avaliados de forma automática utilizando principalmente técnicas do tipo *black-box,* em que são utilizados conjuntos de casos de teste para testar se o *output* produzido coincide ou não com um valor esperado. Esta verificação pode ser feita através da comparação direta de valores ou mediante a utilização de expressões regulares.

3.4. *Learning analytics*

Além dos sistemas apresentados na secção anterior, existem alguns sistemas como o ***SnapViz*** [105] ou ***CodeBrowser*** [106], que disponibilizam funcionalidades de navegação e visualização *off-line* de grandes quantidades de *snapshots* de código desenvolvido pelos estudantes. Também o sistema ***ClockIt*** [107] permite a análise de informação agregada sobre sessões de programação no ***BlueJ IDE***.

Com efeito, nos últimos anos tem havido um crescente interesse na recolha e análise de *snapshots* de código fonte para efeitos de análise *off-line* com vista à construção de modelos que permitam explicar e/ou identificar precocemente determinado comportamento dos estudantes enquanto resolvem problemas de programação. Em [108] Ihantola et al. apresenta-se não só um conjunto de sistemas e abordagens, mas também um exaustivo estudo sobre os principais desafios que quem pretenda desenvolver um sistema deste tipo deve ter em consideração.

Em algumas situações, o predicado é obter informação não sobre o código em si, mas antes sobre o processo de escrita deste. Assim, alguns autores [109] desenvolveram sistemas que registam dados sob a forma de *key-logging,* ou seja, é registada informação sobre o ritmo com que as teclas foram pressionadas, os momentos de pausa que existiram entre os momentos de escrita, etc. Nestes casos, tenta-se avaliar/detetar o comportamento por detrás do processo de escrita do código e não tanto o código que acabou por ser escrito.

Comum a quase todas a abordagens anteriormente descritas é o grande volume de dados produzidos, e que é claramente influenciado pelo nível de granularidade utilizado. Na Figura 19 é possível observar os vários tipos de granularidade existentes.

Dependendo do tipo de trabalho que está a ser levado a cabo, um determinado nível de granularidade pode ser mais interessante do que o outro. Entre outras coisas, este estudo permitiu aos autores concluir que o impacto decorrente da escolha de um determinado nível de granularidade face a outro é igualmente influenciado pelo nível de conhecimentos dos estudantes alvo do estudo.

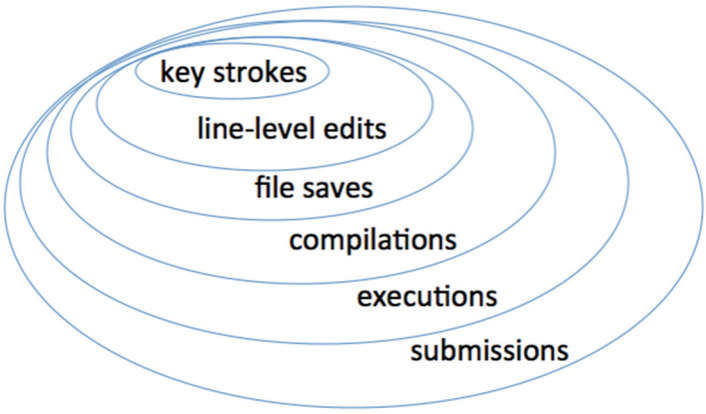

Figura 19 - Os diversos níveis de granularidade [110].

Muito embora não tenham propriamente desenvolvido sistemas de apoio ao professor, foram recentemente conduzidos diversos estudos de *learning analytics*, cujo principal objetivo consiste em analisar informação produzida pelos estudantes aquando do processo de aprendizagem, com vista a detetar padrões, comportamentos, etc.

Exemplos disso são os trabalhos conduzidos por Pirch et al. [84] e Tabanao et al. [111] no âmbito do ensino de programação. Em ambos os casos, recorre-se a *plug-ins* instalados nos ambientes de desenvolvimento utilizados para escrever o código fonte a fim de enviar *snapshots* do código desenvolvido pelos estudantes para posterior análise.

No caso concreto do trabalho de Pirch et al. o objetivo é conseguir determinar qual o caminho que um determinado estudante seguiu para atingir uma determinada solução, ao passo que o trabalho desenvolvido por Tabanao et al. tem como principal objetivo identificar os estudantes que estejam a atravessar dificuldades em resolver os exercícios propostos.

Para além disso, quer num caso, quer no outro, os dados recolhidos foram analisados após o processo de aprendizagem ter terminado, ou seja, caso fossem detetados desvios em relação ao esperado já pouco haveria a fazer em relação aos estudantes envolvidos no estudo, podendo, no entanto, ser tomadas medidas de melhoramento nos anos seguintes.

Não obstante o panorama, os autores afirmam que seria importante conseguir fazer a análise em tempo real, já que poderiam ser reportados aos professores os casos dos estudantes que, de uma forma ou outra, fogem ao que seria esperado, permitindo-lhes deste modo agir em conformidade ainda em tempo útil.

Bumbacher et al. [112] levaram a cabo um estudo com o objetivo de identificar situações em que um determinado estudante poderá estar a necessitar de ajuda através da análise do código fonte que escreveu. Para caracterizar cada *snapshot* de código foi utilizada informação, tal como o número de linhas de código, o número de comentários, o número de blocos de comentário, o número de variáveis, o número de métodos, o número de ciclos e o número de instruções condicionais. Com base nesta informação, são construídos *clusters* que agregam os *snapshots* em que existe um elevado grau de semelhança. O principal objetivo deste trabalho foi identificar comportamentos que possam conduzir à necessidade de pedir ajuda com base em métricas como o número total de *clusters* visitados, número de vezes em que houve mudança de *cluster*, etc.

Uma abordagem semelhante foi seguida por Blikstein et al. [113] em que foram recolhidos alguns milhares de *snapshots* de código fonte e, para cada um deles, foram recolhidos diversos dados comparativos entre uma tentativa de resolução ***n*** e a tentativa anterior ***n-1*** de um exercício em concreto. Objetivamente, foram recolhidos dados relativos aos seguintes elementos: número de linhas adicionadas, número de linhas removidas, número de linhas modificadas, número de carateres adicionados, número de carateres removidos e número de carateres modificados.

Com base nos dados recolhidos tentou obter-se informação sobre o comportamento dos estudantes, como por exemplo:

- saber se os estudantes mais experientes têm uma tendência maior para pensar melhor o código e escrever grandes blocos de código de cada vez;
- saber se os estudantes menos experientes compilam o seu código muito mais vezes, havendo um número muito menor de alterações entre cada compilação;
- saber se a quantidade de alterações entre cada compilação pode de algum modo ser preditor do desempenho dos estudantes.

Os resultados obtidos numa série de experiências levadas a cabo pelos autores permitiram-lhes concluir que, de facto existe, uma relação direta entre o desempenho dos estudantes (nota final) e a quantidade de alterações que fizeram entre cada compilação.

Recorrendo à análise de *snapshots* dos estudantes, Jadud [114] [115] propôs uma métrica com o nome "*Error Quotient*" que, segundo o autor, pode ser utilizada para caracterizar eficazmente a performance dos estudantes. Em termos gerais, esta métrica baseia-se na contagem do número de pares consecutivos de *snapshots* para um mesmo exercício contendo erros de compilação. Segundo o autor, os estudantes que apresentem um valor de *error quotient* muito elevado poderão estar a atravessar dificuldades e, como tal, deverão ser alvo da atenção dos professores.

Watson et al. [116] propuseram um algoritmo chamado **WATWIN**, como uma extensão do trabalho proposto por Jadud. Em termos simples, recorreram ao acrescento de mais informação tal como o número das linhas em que os erros ocorreram (de modo a verificar se os erros são os mesmos ou outros). De acordo com os resultados de alguns testes efetuados pelos autores, este novo algoritmo consegue identificar mais eficazmente situações em que os estudantes possam necessitar de ajuda, do que o anteriormente apresentado *error quotient*.

Vihavainen et al. [117] recorreram à utilização de dados extraídos de *snapshots* previamente recolhidos (*e.g.* hora em que o *snapshot* foi recebido, número de minutos que faltam para o *deadline*, minutos entre *snapshots* consecutivos, números de erros de compilação, percentagem de *snaphots* feitos durante o período da noite) por forma a caracterizar o comportamento dos discentes. Além disso, utilizaram ainda a informação recolhida para tentar prever a performance dos mesmos em outras unidades curriculares (Matemática, neste caso). Atendendo aos resultados apresentados, poder-se-á concluir que a utilização desta informação permite obter resultados satisfatórios em ambos os casos.

Já Ahadi [118] realizou um conjunto de testes utilizando diferentes técnicas de *machine learning* por forma a desenvolver mecanismos de identificação dos estudantes com melhores e piores performances a partir de um *dataset* de *snapshots* anteriormente recolhidos. Os resultados obtidos permitiram-lhe concluir que em alguns casos era possível obter taxas de sucesso superiores a 90%.

Num outro estudo, Ahadi et al. [119] tentaram perceber se o número de tentativas que um determinado estudante faz para a resolução de um determinado exercício pode ser utilizado para prever a performance global deste. O principal objetivo era a utilização das métricas desenvolvidas para identificar estudantes que possam estar numa situação que os possa estar a levar para uma má performance e, deste modo, alertar o professor

para que algo deva ser feito de modo a mitigar a situação. Os autores comparam uma abordagem em que era utilizada apenas informação sobre o resultado final dos exercícios (certo/errado) com a nova abordagem que recorria à utilização do número de tentativas, tendo concluído que a última conseguia obter melhores resultados.

Com base em *datasets* semelhantes aos utilizados anteriormente, Ahadi et al. [120], tentaram ainda verificar até que ponto a existência de flutuações na consistência da performance semanal dos estudantes poderia ser utilizada como preditor do seu desempenho final. A este respeito, chegaram à conclusão de que em termos globais a maior dos estudantes apresenta flutuações semelhantes ao longo do tempo e, como tal, esta informação não é um bom preditor para o desempenho global destes. Contudo, foi observado que a informação sobre a consistência da performance dos estudantes num determinado exercício poderia ser eficazmente utilizada na identificação de estudantes com dificuldades, nesse mesmo exercício.

Utilizando uma abordagem completamente diferente, Annamaa et al. [121] levaram a cabo uma experiência que consistia na resolução de vários exercícios utilizando o IDE ***Thonny*** desenvolvido pelos mesmos autores e que, entre outras funcionalidades interessantes, permite fazer *log* automático de várias ações levadas a cabo pelos discentes (número de carateres escritos num determinado bloco de tempo, número de vezes que o estudante fez *copy/paste* de uma pedaço de código, número de vezes que o código foi executado e o resultado da execução). Uma desvantagem clara deste sistema é o facto de a informação ser recolhida localmente nos computadores dos estudantes, tendo estes que posteriormente enviar a informação manualmente para os professores. Embora estes dados possam vir a permitir obter informações interessantes, acabam por não poder ser utilizadas pelos professores no imediato (mais concretamente em tempo útil).

3.5. Sumário

Como apresentado nas secções anteriores, nos últimos anos, têm sido desenvolvidos e testados diversos sistemas com capacidade para auxiliar os professores na sua tarefa. Na maior parte dos casos, este auxílio surge principalmente na forma de sistemas de *auto-grading* que permitem aos estudantes obter algum tipo de *feedback* imediato relativo à solução que desenvolveram.

Ainda que constituam certamente uma grande mais-valia, estes sistemas implicam geralmente a utilização de um conjunto pré-definido de exercícios. Quando permitem a utilização de exercícios customizados, este processo é geralmente complexo e moroso. Em ambos os casos, este é certamente um fator que pode contribuir para a não utilização destes sistemas por muitos professores.

Foi igualmente observado que a grande maioria dos sistemas tem como principal enfoque a disponibilização de informação aos estudantes e, ainda que alguns disponibilizem alguma informação aos professores de modo a permitir-lhes fazer um eficaz acompanhamento do progresso dos estudantes, esta informação é geralmente escassa e de difícil compreensão, não permitindo aos professores uma rápida perceção do estado global dos estudantes.

Foram também identificados diversos trabalhos na área de *learning analytics*, no entanto, como mencionado anteriormente, acabam por ter como principal desvantagem o facto de consistirem principalmente de construção de métricas ou modelos preditivos resultantes da análise de dados (recorrendo a técnicas de *machine learning*) efetuados *a posteriori* e, como tal, ainda que interessantes, nem sempre são passíveis de serem testados em contextos reais, não sendo muitas vezes a sua aplicabilidade validada.

Estas características e/ou limitações identificadas desempenharam um papel crucial na definição das linhas mestras a seguir aquando do desenvolvimento do trabalho que seguidamente se apresenta.

4. Desenvolvimento

Um dos objetivos deste trabalho era avaliar de que modo as submissões feitas pelos estudantes aquando da resolução dos exercícios podiam ser utilizadas, em tempo real, para fornecer informação aos professores que permitisse que estes identifiquem necessidades de intervenção, imediatas ou numa aula posterior (*e.g.* estudantes com dificuldades em progredir, exercícios que poucos estudantes consigam resolver), e, com base nessa informação, agir em conformidade, em tempo útil.

Com vista a atingir este objetivo, procedeu-se ao desenvolvimento de uma ferramenta denominada ***CodeInsights,*** que, em traços gerais, é responsável por obter e enviar cópias do código desenvolvido pelos estudantes (*snapshots*) para um servidor central mediante a utilização de um *plug-in,* que deverá ser instalado no editor de código ou IDE utilizado. Posteriormente, já no servidor, o código fonte é processado e a informação resultante disponibilizada aos professores.

Uma questão importante e que condicionou em grande medida muitas das decisões tomadas ao longo de todo o processo de conceptualização e implementação deste sistema prende-se com o facto de a utilização do sistema ***CodeInsights*** não dever interferir com a forma como o professor conduz as suas aulas.

Como mencionado anteriormente, foi utilizada a metodologia DBR, o que significa que o artefacto (neste caso, a ferramenta ***CodeInsights***) foi desenvolvida e avaliada em contexto real de sala de aula. Foram efetuadas sete iterações principais, às quais foram acrescentadas uma iteração preliminar (iteração zero), que consistiu maioritariamente na realização de trabalho exploratório e duas iterações posteriores (iterações oito e nove) que, não estando inicialmente previstas, acabaram por se revelar pertinentes, atendendo ao contexto em que ocorreram.

De seguida faz-se uma apresentação detalhada do trabalho desenvolvido ao longo de cada uma das iterações, começando por elencar os objetivos da iteração, apresentando posteriormente o trabalho desenvolvido e a avaliação feita. Finaliza-se com a apresentação de um pequeno sumário da iteração, no qual são apresentados os principais resultados alcançados e elencados igualmente alguns dos aspetos que ficaram pendentes para as seguintes iterações.

4.1. Iteração zero

4.1.1. Objetivos

Esta iteração preliminar teve como principal objetivo a definição de uma arquitetura geral para os sistemas de recolha e de análise de dados. Foi ainda realizado um trabalho de exploração do modo como poderia ser criado e atualizado um modelo do conhecimento dos estudantes com base na informação incluída nos *snapshots* recolhidos, por forma a permitir fornecer a informação aos professores.

Procedeu-se ainda ao estudo dos principais ambientes de desenvolvimento utilizados pelos estudantes que se previa virem a colaborar na avaliação do sistema, com vista a averiguar a melhor forma de operacionalizar o processo de recolha dos *snapshots*.

4.1.2. Trabalho desenvolvido

Tal como indicado por Vihavainen et. al. [110], neste tipo de sistemas, podem ser utilizados diversos níveis de granularidade ao nível dos dados recolhidos, desde granularidades muito finas (*e.g.* registo de teclas pressionadas/ações efetuadas pelos estudantes) até granularidades mais grossas (*e.g.* submissões finais).

Após algum trabalho exploratório decidiu-se que de modo a permitir fornecer informações ao professor em tempo real, por questões de operacionalização, os níveis de granularidade mais finos seriam excluídos, pois o volume de dados que era necessário transmitir e processar era muito elevado e poderia colocar em causa o próprio desempenho do sistema. Assim sendo, entendeu-se que uma granularidade ao nível do *snapshot* de código fonte que é recolhido sempre que o estudante grave o ficheiro ou compile o código (dependendo do ambiente de desenvolvimento utilizado) seria o que melhor se alinhava com os objetivos gerais do trabalho a desenvolver.

4.1.2.1. Arquitetura do sistema

Após decidir qual o nível de granularidade dos dados a recolher junto dos estudantes, procedeu-se à definição da arquitetura geral do sistema a implementar, nomeadamente ao nível da identificação dos principais componentes, assim como a forma como estes componentes deveriam interagir. Na Figura 20 pode ser observada uma primeira abordagem à definição desta arquitetura

Esta arquitetura conceptual baseia-se em certa medida na arquitetura de um sistema de agentes inteligentes de suporte ao ensino/aprendizagem apresentada por Jafari [122].

Em traços gerais, estão envolvidos no processo três entidades distintas: os **estudantes** que submetem os trabalhos, os **professores** que colocam os enunciados no sistema e acedem à informação disponibilizada e o **sistema** que recolhe a informação dos estudantes, processa a informação e disponibiliza o resultado ao professor.

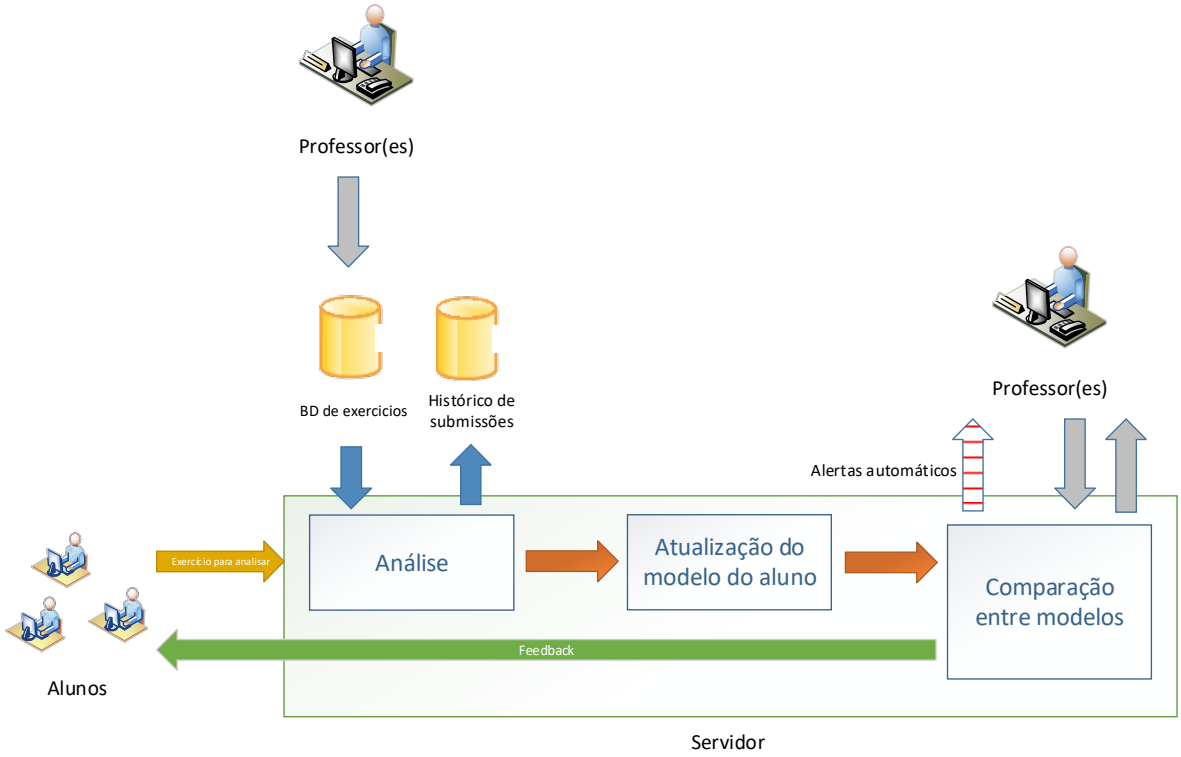

Figura 20 - Arquitetura geral do sistema - iteração zero.

4.1.2.2. Base de dados de exercícios

Por forma a suportar o funcionamento do sistema, foi necessário introduzir os enunciados dos exercícios a disponibilizar aos estudantes numa base de dados.

Cada exercício deve conter o enunciado e um conjunto de informação adicional indispensável para o funcionamento do sistema. Uma das informações mais relevantes diz respeito a um conjunto de casos de teste a utilizar para testar/validar o código dos estudantes (*inputs* utilizados e os respetivos *outputs* esperados).

O professor poderá ainda especificar uma lista de **conceitos** associados a cada **exercício** (*e.g.* declaração e inicialização de variáveis; utilização de instruções condicionais; utilização de ciclos, manipulação de *strings*, etc.).

As **regras** dizem respeito aos aspetos que o professor considere pertinente averiguar para determinar se um determinado exercício pode ser considerado certo ou errado. Na prática, as regras correspondem aos aspetos que um professor iria ter em consideração caso fosse avaliar o exercício "manualmente", para além do resultado produzido.

O sistema deverá disponibilizar um conjunto de **regras** passíveis de serem associadas a qualquer exercício, podendo, no entanto, ser possível adicionar novas regras e, consequentemente, associar essas novas regras ao processo de análise de um determinado exercício.

Finalmente, poderão ainda ser introduzidas **soluções modelo** para cada exercício por forma a apoiar o professor no processo de ajuda aos estudantes.

Na Figura 21, é apresentada uma representação gráfica da informação associada a cada um dos exercícios inseridos no sistema

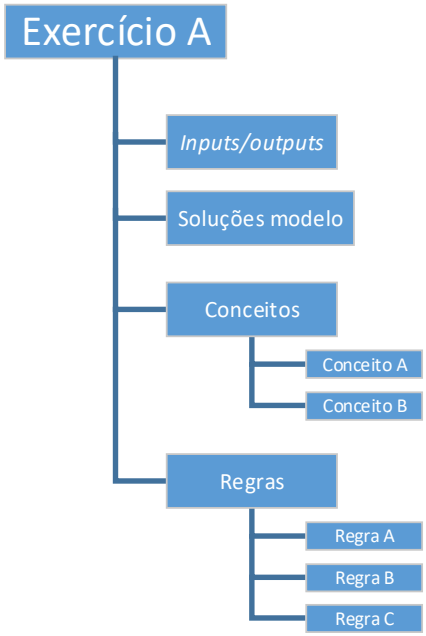

Figura 21 - Estrutura de informação associada a cada exercício - iteração zero.

4.1.2.3. Processo de envio do código fonte para o servidor

Como visto anteriormente, em muitos dos sistemas de *auto-grading* são utilizadas interfaces "*web based*" para os estudantes desenvolverem o código. Numa fase inicial, esta possibilidade foi igualmente equacionada. No entanto, por forma a ser o menos disruptivo possível, acabou por se optar por uma solução que permite que os estudantes continuem a utilizar os ambientes de desenvolvimento tradicionalmente utilizados nas

aulas (que geralmente não são *web based*) e que muito provavelmente irão utilizar nas suas vidas profissionais. O requisito de utilização dos ambientes de desenvolvimento utilizados nas aulas foi igualmente mencionado como sendo "obrigatório" por diversos professores a quem foi sendo apresentada a ideia do sistema a desenvolver.

Um elemento fulcral para o funcionamento do sistema era o componente responsável pelo envio do código dos estudantes para um servidor central. Para tal ser possível, era necessário disponibilizar *plug-ins* para os principais ambientes de desenvolvimento utilizados habitualmente nas aulas. Devido ao elevado número de ambientes de desenvolvimento existentes, o sistema deveria ser pensado de modo a ser independente em relação ao ambiente de desenvolvimento utilizado. Deste modo deveria ser possível que fossem facilmente desenvolvidos e disponibilizados *plug-ins* para outros ambientes de desenvolvimento, sem que fosse necessário fazer modificações à ferramenta de processamento dos *snapshots*. Do ponto de vista do sistema de monitorização, deveria, portanto, ser indiferente qual o editor de código ou IDE utilizado pelos estudantes, podendo inclusivamente estarem a ser utilizados diferentes ambientes de desenvolvimento por estudantes da mesma turma em simultâneo.

Em termos concretos, estes *plug-ins* deveriam ser extensões da operação *"Run…"* (ou eventualmente "*Save*") já presentes nos ambientes de desenvolvimento, de tal modo que para além de ser efetuada a compilação/interpretação e execução código fonte em que o estudante se encontrava a trabalhar localmente no seu computador, era também enviado um *snapshot* do referido código para um servidor central, onde era processado e analisado. Na Figura 22, está ilustrado o processo de envio dos *snapshots* do computador dos estudantes para o servidor.

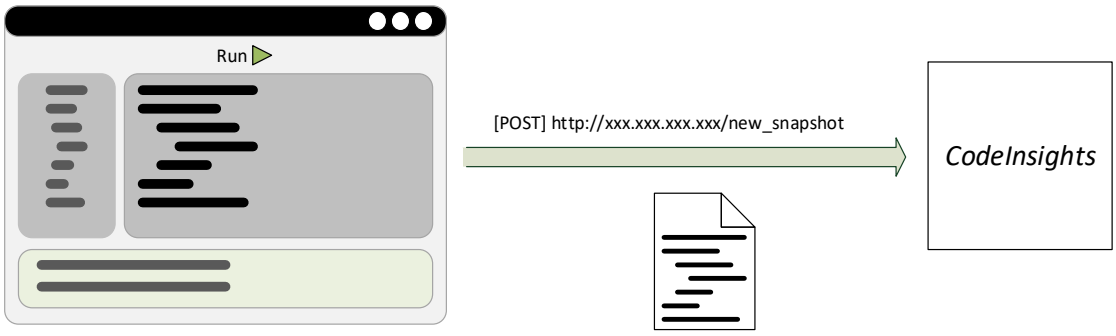

Figura 22 - Ilustração do processo de envio dos *snapshots* para o servidor.

Com base neste modo de funcionamento, os estudantes poderiam continuar a programar sem que aos seus olhos algo de diferente acontecesse, já que o envio dos *snapshots* ocorreria em *background*.

Para o correto funcionamento do sistema era necessário enviar juntamente com o código elementos que permitissem a identificação do estudante e do exercício em que se encontrava a trabalhar. Para este efeito, chegou a ser ponderada a utilização de uma solução baseada em *templates* como a que é utilizada no sistema *TestMyCode*, mas essa solução implicava mais trabalho de preparação dos *templates* por parte dos professores, o que poderia ser visto como um entrave. Assim, optou-se pela utilização de uma abordagem em que são utilizadas *tags* personalizadas inseridas diretamente no código desenvolvido pelos estudantes.

Na Figura 23, podem ser observados os elementos identificativos do estudante e do exercício que são inseridos em cada *snapshot*.

```
/**
 * Codigo fonte do exercício XPTO
 *
 * @authorid 12345
 * @assignmentid 987
*/
class XPTO {
   ...
}
```

Figura 23 - Exemplo de um cabeçalho de uma classe com identificação do estudante e do exercício - iteração zero.

De modo a não obrigar os estudantes a escrever manualmente os elementos identificativos, optou-se por recorrer a uma funcionalidade disponibilizada pela maioria dos ambientes de desenvolvimento que passa pela possibilidade de alterar os *templates* que são utilizados de cada vez que é criado um novo ficheiro. Deste modo, sempre que o estudante criar um novo ficheiro, o conteúdo deste será automaticamente preenchido com um pedaço de código semelhante ao apresentado na Figura 23, necessitando o estudante de alterar apenas o valor associado à identificação do exercício.

Com base no processo de identificação apresentado anteriormente, é, pois, possível receber, processar e guardar no servidor o código fonte de todas as tentativas de resolução de cada exercício feitas por um determinado estudante.

4.1.2.4. Análise do código fonte

Após o código fonte ser recebido no servidor era necessário proceder à sua análise de acordo com o conjunto de regras previamente definidas para o exercício em questão.

Na prática, as regras funcionam como sensores que permitem fazer o diagnóstico do estado do código fonte submetido. Foram utilizadas regras do tipo *black-box* em que apenas era verificado se perante um determinado *input* era obtido o *output* desejado sem ter em conta a qualidade do código fonte. Este tipo de soluções tem a vantagem de ser relativamente fácil de implementar e utilizar. No entanto, estas abordagens não têm em consideração aspetos como a "elegância" ou a complexidade da solução desenvolvida. Por este motivo foram igualmente consideradas abordagens do tipo *white-box* em que de facto se procede a algum tipo de análise mais apurada da solução submetida.

Desde o início, foi estabelecido como requisito essencial que o sistema deveria suportar diferentes linguagens, mantendo o processo de funcionamento inalterado. Em termos práticos, para suportar várias linguagens de programação, apenas o módulo de análise de código fonte teria de ser modificado, já que os restantes módulos deveriam ser independentes da linguagem de programação utilizada.

Deste processo de análise resultaria um relatório em que fossem evidenciados os valores "medidos" para cada uma das regras definidas para o exercício em causa. Estes valores eram posteriormente utilizados para determinar o "estado" em que se encontrava o conhecimento do estudante a dada altura, isto é, para construir um modelo do conhecimento do estudante.

4.1.2.5. Modelo do conhecimento do estudante

O processo de construção de um modelo tem como principal objetivo o desenvolvimento de uma representação conceptual de uma determinada entidade com vista a conseguir percebê-la melhor e manipulá-la sem afetar diretamente a entidade representada pelo modelo. No nosso caso concreto, interessa-nos construir modelos do estudante aplicados à representação do seu conhecimento no domínio da programação básica.

Tal como defendido por Chrysafiadi et al. [123] numa fase inicial, procedeu-se à escolha das características dos estudantes que se pretendia modelar, tendo a escolha recaído sobre a capacidade de os estudantes dominarem ou não determinado conceito de programação com base nas resoluções dos exercícios submetidas.

Em seguida procedeu-se à análise e escolha da configuração de modelo de estudante que melhor se adequava aos objetivos elencados. Assim, considerou-se que a melhor opção era a utilização de uma rede bayesiana. Este modelo era utilizado para representar em cada momento o "estado atual" do seu conhecimento sobre um conjunto de

tópicos/conceitos de programação e a sua aplicação concreta na resolução de problemas específicos.

Como ponto de partida considerou-se utilizar uma rede bayesiana constituída por três tipos distintos de nós, tal como pode ser observado na Figura 24.

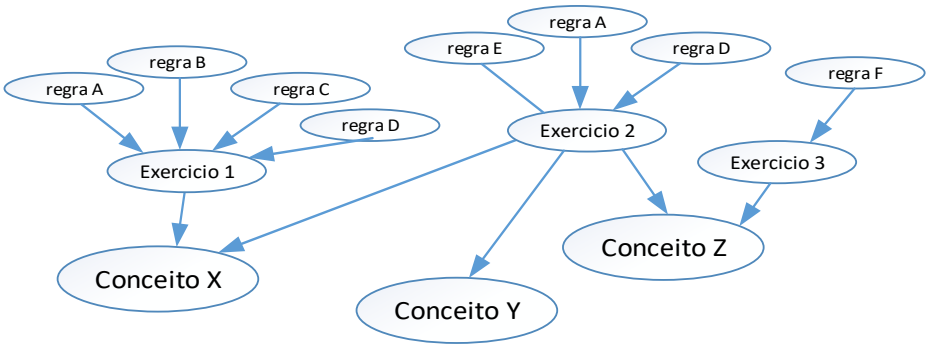

Figura 24 - Representação da rede bayesiana utilizada para construir o modelo do estudante.

O primeiro tipo de nó diz respeito a um conjunto de **regras** consideradas pertinentes para verificar se a solução de um determinado exercício está certa ou errada. A cada nó deste tipo é associado um conjunto de possíveis valores, aos quais posteriormente serão atribuídos valores de probabilidade concretos. A determinação dos valores associados a cada nó em cada momento basear-se-ia na análise do código fonte submetido pelos estudantes. Assumindo, por exemplo, que para a resolução correta de um determinado exercício é necessário recorrer à utilização de um ciclo **`do … while`**, um exemplo de uma regra a utilizar neste caso era a verificação da existência (ou não) desta instrução no código fonte.

No segundo tipo de nó encontram-se os **exercícios** disponibilizados pelos professores. Sendo uma rede bayesiana um modelo probabilístico "de propagação", o valor de probabilidades associado a nodos (ou nós) deste tipo está diretamente ligado aos valores associados aos nós do tipo **regra**, anteriormente apresentados. Ou seja, quando forem atribuídos valores aos nodos do tipo **regra** (colocação das evidências), esses valores irão propagar-se para os nodos do tipo **exercício** ajudando deste modo a determinar o valor que lhes está associado. O significado concreto deste tipo de nodo consiste na probabilidade de o exercício estar certo ou errado com base nos valores de probabilidade dos nodos do tipo regra e ele associados.

Finalmente, o terceiro tipo de nodo corresponde aos conceitos de programação que o estudante deverá ter assimilado. Os valores de probabilidade associados a nodos do tipo **conceito** está diretamente dependente dos valores de probabilidade associados aos

nodos do tipo **exercício**. Concretizando, os valores destes nodos expressam a probabilidade de um determinado conceito ter sido assimilado, com base na probabilidade de os exercícios a ele associados estarem certos ou errados.

Com base no exposto, bastará fornecer como *input* à rede bayesiana (introduzir a evidência) valores de probabilidade dos nodos do tipo **regra** para determinar os valores associados a nodos do tipo **exercício** e, consequentemente, do tipo **conceito**.

Em traços gerais, com base na análise das soluções submetidas pelos estudantes, obter-se-ia informação sobre as probabilidades de um conjunto de regras estarem a ser cumpridas ou não. Pela propagação dos valores de probabilidades das regras obter-se-ia informação sobre as probabilidades de determinados exercícios estarem ou não corretos, o que finalmente conduziria à determinação das probabilidades de um determinado conceito ter sido corretamente apreendido e aplicado.

Para uma apresentação detalhada das várias configurações alternativas de modelos do conhecimento do estudante existentes, consultar o trabalho de Chrysafiadi et al. [123].

Para identificar eventuais dificuldades de aprendizagem de um estudante em relação a determinados conceitos, pensámos em proceder à comparação entre o modelo atual do conhecimento do discente (a rede bayesiana) e um modelo do conhecimento expectável para aquele momento (outra rede bayesiana), como pode ser observado na Figura 25.

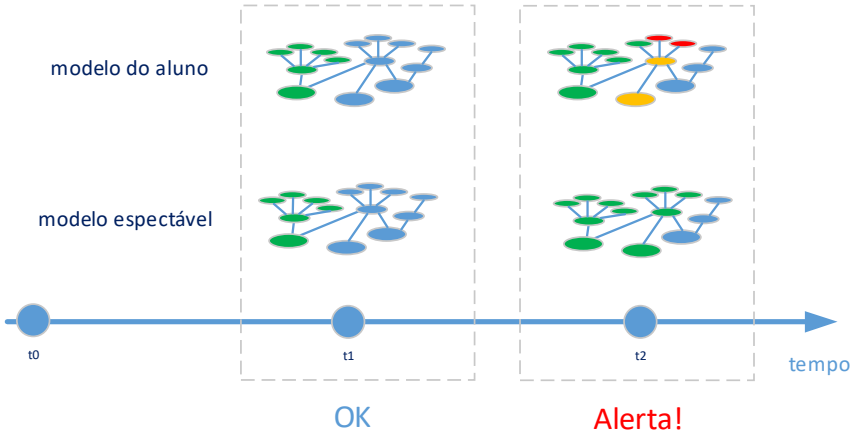

Figura 25 - Representação do processo de comparação entre modelos.

Por omissão, pensou-se em disponibilizar um modelo expectável genérico baseado no conhecimento expectável de um estudante típico para um dado momento. À medida que os professores fossem conhecendo os estudantes poderiam fazer os ajustamentos considerados pertinentes. Pensou-se igualmente que poderia existir mais do que um

modelo expectável para cada momento, de modo a conseguir refletir as reais expectativas que um professor tem em relação ao desempenho de diferentes conjuntos de estudantes.

Em termos concretos, para identificar desvios no desempenho dos estudantes face ao que seria expectável, proceder-se-á ao cálculo do contraste dos valores de probabilidades de cada um dos nós da rede bayesiana representativa de cada um dos modelos. Caso as diferenças sejam consideráveis, será apresentada/enviada uma notificação ao professor.

De modo a não estar constantemente a notificar os professores, era importante definir o significado de "diferenças consideráveis". Poderia, por exemplo, ser adotada uma abordagem baseada na definição empírica de um valor mínimo de diferença a partir do qual os alertas eram apresentados ao professor, ou, em alternativa, considerar a integração de outro tipo de mecanismos, de modo a permitir apresentar ao professor apenas os alertas que constituam efetivamente novidade.

À medida que os professores vão inserindo novos exercícios e indicando as regras e os conceitos, estão automaticamente a contribuir para a definição da estrutura do modelo do estudante. Sempre que os estudantes submetam uma solução para avaliação, esta será analisada e o modelo do estudante será atualizado com base na informação recolhida.

Para além de introduzir os exercícios, os professores deverão definir um conjunto de *milestones* de aprendizagem e, para cada uma delas, configurar o **modelo expectável do estudante**, ou seja, uma rede bayesiana representativa das expectativas do professor em relação às capacidades que os estudantes devem ter num determinado momento.

Na prática, o professor apenas deverá necessitar atribuir valores aos sensores associados aos exercícios que os estudantes deveriam ter realizado até um dado momento com base nas suas expectativas.

De modo a minimizar a interação dos professores com o sistema, é de todo o interesse ser disponibilizado um conjunto de *milestones* e respetivos modelos expectáveis do estudante *baseline*, podendo, no entanto, ser alterados pelos professores sempre que necessário.

Aquando da submissão de uma solução por parte dos estudantes, proceder-se-á à comparação entre o **modelo do estudante** e o **modelo expectável do estudante** para aquela determinada *milestone*.

4.1.2.6. Aplicação de monitorização do desempenho dos estudantes

Como mencionado anteriormente, o acompanhamento eficaz por parte dos professores e a apresentação atempada de *feedback* aos estudantes desempenham um papel importante no processo de aprendizagem. Assim, foi considerado importante disponibilizar ao professor uma aplicação *web-based* onde este possa monitorizar em "tempo real" o desempenho dos estudantes de uma forma visualmente simples e agradável.

Nesta aplicação *web,* são apresentados todos os alertas emitidos pelo módulo de aferição do estado do estudante e, além disso, o professor poderá a qualquer altura pedir para ser apresentada a comparação entre os modelos para um determinado estudante ou turma, tal como representado na Figura 26.

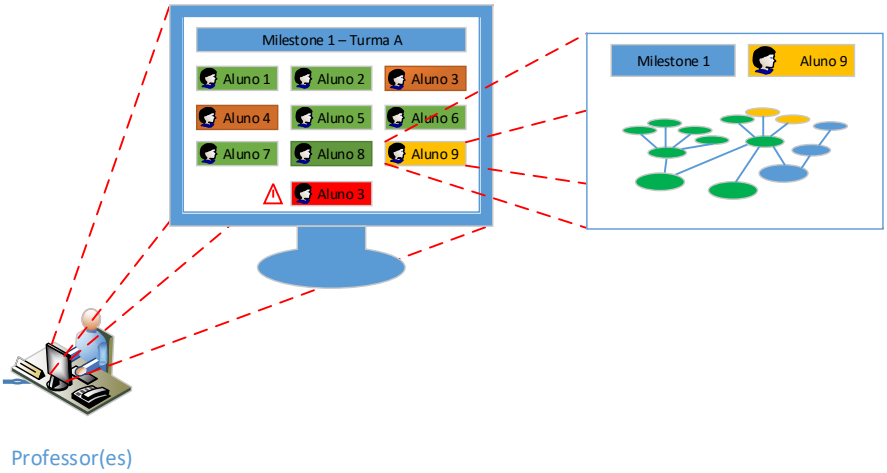

Figura 26 - Esquema representativo da aplicação de monitorização utilizada pelos professores.

O professor poderá assim obter informação sobre o desempenho dos estudantes e, caso sejam detetadas situações indesejadas, agir em conformidade. Estando os estudantes presentes na sala de aulas, o professor poderá sugerir pessoalmente aos estudantes as ações que, no seu entender, deverão ser adotadas para reverter os problemas que os estudantes estejam a enfrentar. De modo a posteriormente poder aferir os efeitos que tais ações efetivamente surtiram, sugere-se que o professor insira anotações no sistema sobre as ações, escolhas e estratégias que adote em relação a um determinado estudante.

O sistema pode igualmente ser utilizado para monitorizar o desempenho de estudantes que possam não estar na sala de aulas por motivos de ordem temporária ou permanente (estudantes trabalhadores-estudantes ou estudantes de cursos de *e-learning*), podendo ser disponibilizada a possibilidade de os professores enviarem *feedback* aos estudantes em causa utilizando a aplicação.

Posteriormente, o professor poderá consultar as mensagens que enviou no passado a cada estudante assim como as anotações que tenha inserido.

4.1.2.7. Implementação do sistema

Após ter sido definida a arquitetura geral do sistema e escolhida a forma como a informação recolhida era representada (o modelo conceptual de representação do conhecimento dos estudantes) deu-se início a algum trabalho exploratório de implementação do sistema. Apresentam-se em seguida detalhes do trabalho realizado neste âmbito.

4.1.2.7.1. Desenvolvimento do *plug-in*

Numa primeira fase, foram concentrados esforços no desenvolvimento do *plug-in* a instalar no computador dos estudantes e que era responsável pela recolha de informação com base no código escrito pelos mesmos.

Nesta fase não estava ainda definido com exatidão quais as linguagens de programação que deveriam ser suportadas e o mesmo acontecia em relação aos ambientes de desenvolvimento para os quais deveriam ser desenvolvidos os *plug-ins*. Em ambos o caso iria depender da disponibilidade dos professores para testar o sistema nas suas unidades curriculares. Perante este cenário foi necessário tomar decisões e, uma vez que havia a possibilidade de o sistema vir a ser testado numa turma em que era usada a linguagem C, optou-se numa primeira fase por suportar esta linguagem de programação.

No que diz respeito ao ambiente de desenvolvimento, nesta fase inicial a escolha incidiu sobre a plataforma *Eclipse* por se tratar de uma plataforma suportada pelos principais sistemas operativos (*Windows*, *Mac OS X*, *Linux*). Para além disso, a plataforma suporta ainda várias linguagens de programação mediante a utilização de um conjunto de ferramentas específico. Por exemplo, no caso da linguagem C é utilizado o CDT (C/C++ *Development Tooling*[22]), no caso da linguagem Java é utilizado o JDT (*Java Development Tools*) e para a linguagem PHP é utilizado o PDT (*PHP Development Tools*).

Muito embora se trate de uma plataforma amplamente difundida e utilizada há já vários anos, e com uma vasta variedade de *plug-ins* disponíveis no *Marketplace*, infelizmente, no que concerne à documentação sobre desenvolvimento de *plug-ins*, verificou-se uma grande falta de informação e, em muitos dos casos, quando esta existia dizia respeito a

[22] https://www.eclipse.org/cdt/

versões anteriores da plataforma *Eclipse* e que não eram necessariamente compatíveis com a mais recente. Ainda assim, depois de um longo processo de pesquisa de informação, foi finalmente atingido o primeiro objetivo que passava por desenvolver o embrião do *plug-in* que viria posteriormente a ser utilizado.

Numa primeira fase, este *plug-in* consistia numa nova opção de menu e botão que, ao serem clicados, obtinham do IDE o código do ficheiro aberto naquele momento, enviando-o seguida via HTTP para um servidor. Caso se pretendesse verificar se o código tinha erros de compilação era necessário adicionalmente clicar no botão ou pressionar a tecla associada a esta funcionalidade (a tecla **F9**, neste caso). Neste ponto dos trabalhos, no servidor, era simplesmente criado um ficheiro de texto contendo o código recebido.

Rapidamente se tornou evidente que não tinha sido uma escolha feliz exigir dos estudantes o acionamento manual de uma opção para enviar o código para o servidor e, como tal, foi necessário encontrar uma alternativa viável. Para tal foram tidas em consideração duas possibilidades:

- "chamar" o compilador quando fosse clicado o botão associado ao *plug-in* desenvolvido;
- executar o *plug-in* desenvolvido quando fosse utilizado o botão (ou tecla de atalho) para proceder à compilação já disponibilizado para o efeito pelo Eclipse;

Tal como já tinha acontecido anteriormente, a documentação era escassa e desatualizada, pelo que durante algum tempo os progressos foram lentos. Após inúmeras tentativas frustradas no sentido de desenvolver a segunda possibilidade, que era obviamente a que faria mais sentido, os esforços concentraram-se em tentar implementar a primeira, o que acabaria efetivamente por acontecer parcialmente. Deste modo, passou a ser possível compilar e, posteriormente, executar o código (caso não tivesse erros de compilação). No entanto, no caso de existirem erros de compilação estes não eram apresentados ao utilizador de forma conveniente.

4.1.2.7.2. Ferramenta de processamento do código

Por forma a dar continuidade ao desenvolvimento do sistema, após algum tempo gasto no desenvolvimento do *plug-in*, foi chegada a altura de concentrar esforços no desenvolvimento da ferramenta de processamento de código no servidor. Atendendo ao facto de o *plug-in* não estar completamente operacional, e uma vez que nesta fase era dada prioridade à componente de servidor, foi desenvolvida uma simples página *web* em

que era introduzido o código que se pretendia verificar e um botão que, quando pressionado, desencadearia o processo de envio do código para o servidor (Figura 27).

Figura 27 - Ferramenta temporária de envio de código para o servidor - iteração zero.

Como referido anteriormente, aquando do processo de avaliação do código deveria ser utilizado um conjunto de regras para cada exercício. Para além de se verificar se o código compilava sem erros, e de testar os programas desenvolvidos com um conjunto de casos de teste, considerou-se igualmente importante analisar o código fonte. Neste sentido, foram estudadas abordagens baseadas na comparação textual entre a solução desenvolvida pelos estudantes e uma ou mais soluções modelo desenvolvidas pelos professores. No entanto, tendo em conta a natureza criativa da programação, assim como uma série de fatores associados ao código fonte (nomes de variáveis, estilo de programação, uso de comentários, etc.) esta solução acabou por ser abandonada. Em alternativa, equacionou-se a utilização de uma solução baseada na comparação das árvores de sintaxe da solução desenvolvida pelos estudantes e a árvore de sintaxe de uma ou mais soluções modelo. Esta solução já tinha sido utilizada com sucesso por Pierch et al. [84]. O principal problema desta abordagem era a sua elevada complexidade. Para além disso, Pierch et al. haviam utilizado esta abordagem para um único tipo de exercício específico. Por contraste, neste caso, a solução a adotar deveria poder ser facilmente utilizada com uma grande variedade de exercícios. Por este conjunto de motivos esta solução acabaria igualmente por ser posta de parte.

Consequentemente, começou a ser concebida e implementada uma solução para a criação de regras que consistia na utilização de "sensores" que iriam verificar se o código submetido pelos estudantes continha determinados elementos. Por exemplo, assumindo

que para resolver um determinado exercício era necessário um ciclo **do … while**, iria ser utilizado um "sensor" para verificar se, de facto, estava ou não a ser utilizado esse ciclo. Se, por exemplo, dentro deste ciclo fosse necessário incrementar uma variável, utilizar-se-ia um outro "sensor" para fazer esta verificação. Como referido anteriormente, estas regras tinham como principal objetivo emular o comportamento do professor aquando da correção de um exercício, ou seja, aquilo que o professor espera que a resolução do exercício tenha. Na Figura 28 pode ser observado o protótipo da aplicação utilizada para testar estes "sensores".

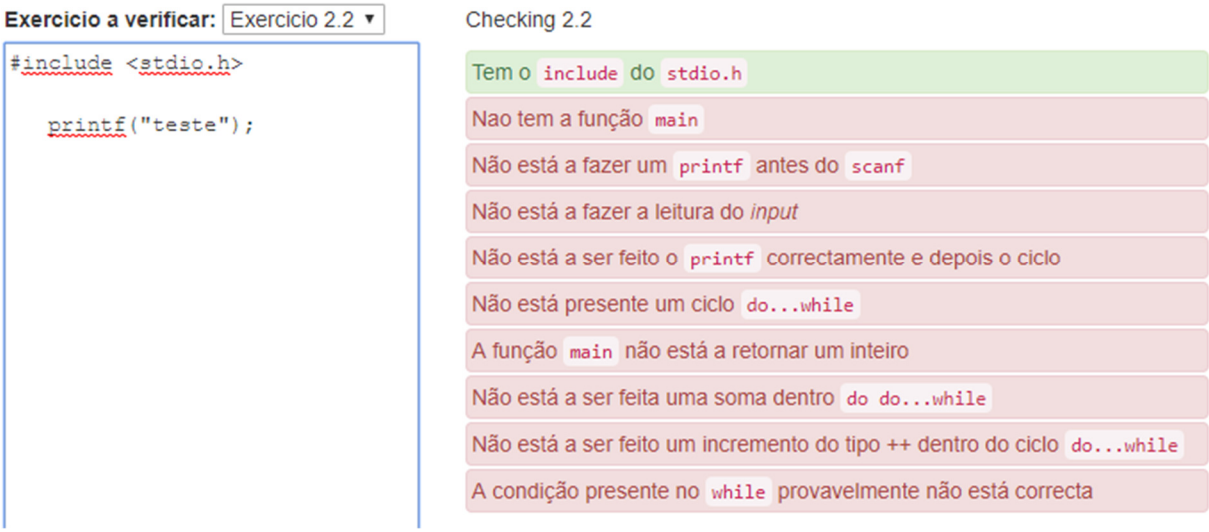

Figura 28 - Exemplo da utilização dos "sensores"

Nesta fase foram implementadas 15 regras distintas apenas para conseguir analisar os dois primeiros exercícios de uma ficha prática de uma unidade curricular introdutória de programação na linguagem C, mostrando que esta opção era difícil de implementar considerando a grande variedade de exercícios habitualmente usados numa unidade curricular introdutória de programação.

4.1.3. Sumário da iteração zero

Nesta iteração, começou-se por proceder à conceptualização geral do sistema e da sua arquitetura. Foi ainda dedicado algum tempo à análise das várias alternativas de configuração de modelo de representação do conhecimento do estudante, tendo acabado por se escolher uma configuração recorrendo a uma rede bayesiana. Para além deste trabalho mais conceptual, procedeu-se igualmente à operacionalização de alguns componentes do sistema, nomeadamente a nível da implementação dos *plug-ins* e

desenvolvimento de uma versão simples de um mecanismo de análise do código fonte com base em "sensores".

Atendendo a que esta iteração inicial consistiu essencialmente em idealizar, projetar e explorar possibilidades, foi encarado com alguma naturalidade o facto de algumas das ideias originais não terem sido completamente integradas, tendo outras ideias sido abandonadas devido à sua complexa operacionalização.

Como seria de esperar, ficaram algumas questões em aberto e que necessitariam maior aprofundamento de futuro. Uma dessas questões prendia-se com o desenvolvimento do *plug-in* que muito embora funcionasse no Eclipse, não o fazia da melhor forma em termos de usabilidade. Além disso, perante a possibilidade de vir a ser testado o sistema em outro ambiente de desenvolvimento, tornou-se necessário desenvolver o *plug-in* para esse ambiente. A ferramenta de processamento do código fonte no servidor ficou igualmente numa fase muito embrionária e como tal era necessário continuar o seu desenvolvimento.

Um outro aspeto que obviamente também necessitaria ser aprofundado dizia respeito à construção, atualização e utilização do modelo do conhecimento dos estudantes por forma a fornecer informação relevante aos professores.

Apesar de esta ser uma iteração preliminar, foram dados passos importantes com vista à preparação das iterações seguintes.

4.2. Iteração um

Tendo como ponto de partida o trabalho realizado na iteração precedente e os novos requisitos específicos do contexto em que o sistema iria ser testado, deu-se início à iteração um, a primeira em que de facto viria a ser implementado um artefacto com vista a ser utilizado em contexto real.

4.2.1. Objetivos

O principal objetivo desta iteração era terminar a implementação do sistema por forma a testá-lo pela primeira vez em contexto real de sala de aula. Pretendia-se com este processo de avaliação averiguar de que modo os estudantes reagiriam à sua utilização e, acima de tudo, verificar até que ponto a utilização do sistema permitiria de facto que os professores conseguissem fazer uma monitorização eficaz do desempenho dos estudantes,

por forma a permitir-lhes identificar situações em que a sua intervenção fosse determinante. Um outro objetivo importante desta iteração passava por verificar até que ponto a configuração do modelo do conhecimento do estudante que havia sido conceptualizada anteriormente poderia efetivamente ser utilizada em contexto real.

4.2.2. Caracterização do contexto de testes

Nesta iteração, o sistema viria a ser utilizado em quatro turmas da unidade curricular de Introdução à Programação e Resolução de Problemas (IPRP) da Licenciatura em Engenharia da Gestão Industrial da Faculdade de Ciências e Tecnologia da Universidade de Coimbra, no 1º semestre do ano letivo 2015/2016. As quatro turmas desta unidade curricular estariam a cargo do mesmo professor e, como já foi referido anteriormente, a linguagem de programação utilizada foi *Python*.

4.2.3. Trabalho desenvolvido

4.2.3.1. Escolha do ambiente de desenvolvimento

Muito embora pudesse ter sido utilizado o Eclipse como ambiente de desenvolvimento, este não tem suporte oficial para a linguagem *Python* tal como acontece em relação às linguagens C, Java e PHP. Para a utilização de *Python* era necessário utilizar um *plug-in* adicional chamado *PyDEV*[23]. Deste modo, após reflexão com o professor responsável pela unidade curricular em questão optou-se por não utilizar o Eclipse.

Um fator importante a ter em consideração aquando da escolha do ambiente de desenvolvimento era a possibilidade de desenvolver *plug-ins* para poder ser implementada a funcionalidade de envio do código dos estudantes para o servidor. A este respeito convirá mencionar que o ambiente de desenvolvimento utilizado em edições anteriores daquela unidade curricular era muito simples e não oferecia qualquer tipo de suporte para a utilização de *plug-ins*, tendo o professor informado que já estava previsto mudar de ambiente de desenvolvimento de qualquer modo.

Assim, após terem sido analisados diversos ambientes de desenvolvimento para a linguagem *Python,* a escolha acabou por recair sobre o *PyCharm*, tendo esta decisão sido acordada com o professor responsável pela unidade curricular em que o sistema iria ser

[23] https://marketplace.eclipse.org/content/pydev-python-ide-eclipse

testado. Um dos fatores que mais pesaram para a escolha deste ambiente de desenvolvimento foi o facto de ser gratuito e baseado na plataforma *IntelliJ* da *JetBrains*. Este último aspeto é muito relevante, uma vez que esta plataforma serve de base para diversos outros ambientes de desenvolvimento, nomeadamente o *IntelliJ* para desenvolvimento em Java, CLion para desenvolvimento em C/C++, o já mencionado *PyCharm* para desenvolvimento em *Python*, o *WebStorm* para desenvolvimento *web* (HTML/CSS/*Javascript*) e o *PHPStorm* para desenvolvimento em PHP. Deste modo, ao ser desenvolvido o *plug-in* para esta plataforma, estaria automaticamente a incluir-se o suporte para diversas linguagens de programação. Para além disso é importante referir que os ambientes de desenvolvimento baseados na plataforma *IntelliJ* são *cross-platform,* o que significa que teriam suporte para os principais sistemas operativos e para além disso são disponibilizadas versões gratuitas para contextos educacionais.

4.2.3.2. Desenvolvimento do *plug-in* para a plataforma *IntelliJ*

Tal como já havia acontecido em relação ao *Eclipse*, também neste caso a documentação sobre desenvolvimento de *plug-ins* para a plataforma *IntelliJ* era consideravelmente simplista e superficial, abordando apenas tópicos muito introdutórios. Ainda assim, ao final de algum tempo estava disponível a primeira versão do *plug-in* do sistema ***CodeInsights*** para a plataforma ***IntelliJ***. Na Figura 29 pode ser observado o aspeto visual do IDE com destaque para os botões e opções de menu do *plug-in*.

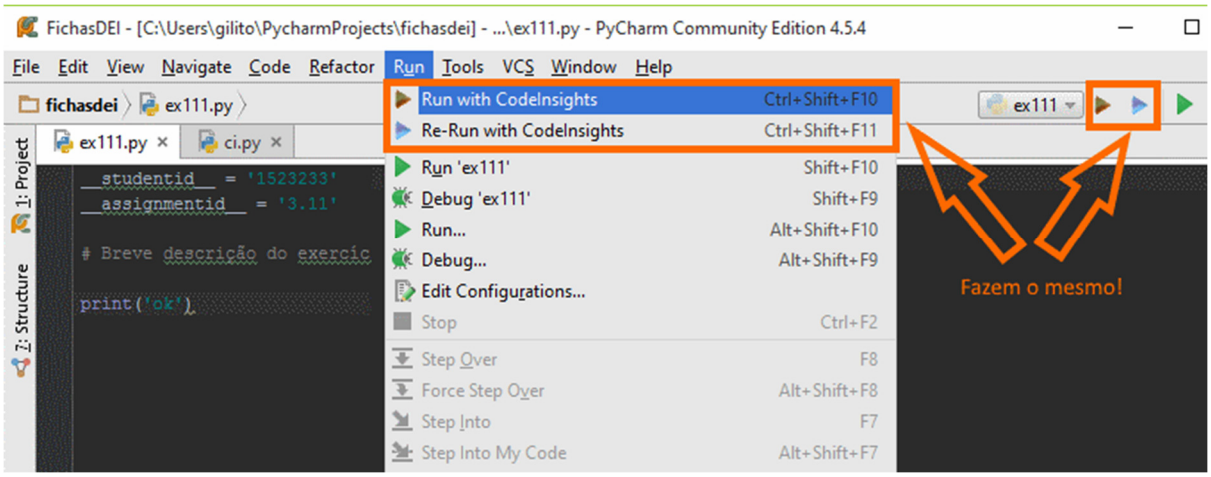

Figura 29 - Exemplo de utilização da primeira versão do *plug-in* para *PyCharm* - iteração um.

O ambiente de desenvolvimento *PyCharm* tem como particularidade o facto de ter duas possibilidades de executar o código desenvolvido. Por este motivo, foi necessário desenvolver e disponibilizar o *plug-in* com duas opções distintas: "*Run with*

CodeInsights" e "*Re-Run with CodeInsights*". A grande diferença entre elas consiste no facto de, na primeira vez que se executa o código, ser possível definir algumas propriedades, ao passo que no segundo caso o código é imediatamente executado com base nas propriedades anteriormente definidas. Para além destas opções no menu, o *plug-in* estava também disponível através de dois botões disponibilizados na barra superior e através de teclas de atalho (`Ctrl + Shift + F10` e `Ctrl + Shift + F11`).

Deste modo, independentemente da opção escolhida, sempre que o código fosse executado, era automaticamente enviada para o servidor um *snapshot* do código fonte aberto no editor. Este processo ocorria completamente em *background* sem que o estudante fosse de modo algum perturbado. Com efeito, este continuaria a obter informação sobre a ocorrência de erros de interpretação provenientes do interpretador local e, caso não existisse nenhum erro de interpretação, poderia executar normalmente o programa desenvolvido na consola do *PyCharm,* como se o *plug-in* nem estivesse a ser utilizado. Nesta iteração, a única informação devolvida pelo sistema de processamento dos *snapshots* dizia respeito ao facto de o *snapshot* ter sido ou não recebido e processado com sucesso. Em todo o caso, de modo a não interferir com a atividade de programação dos estudantes, mesmo estas mensagens eram apresentadas no canto inferior do *PyCharm* de forma não intrusiva. Na Figura 30 é apresentado um exemplo do que era apresentado ao estudante após executar um programa.

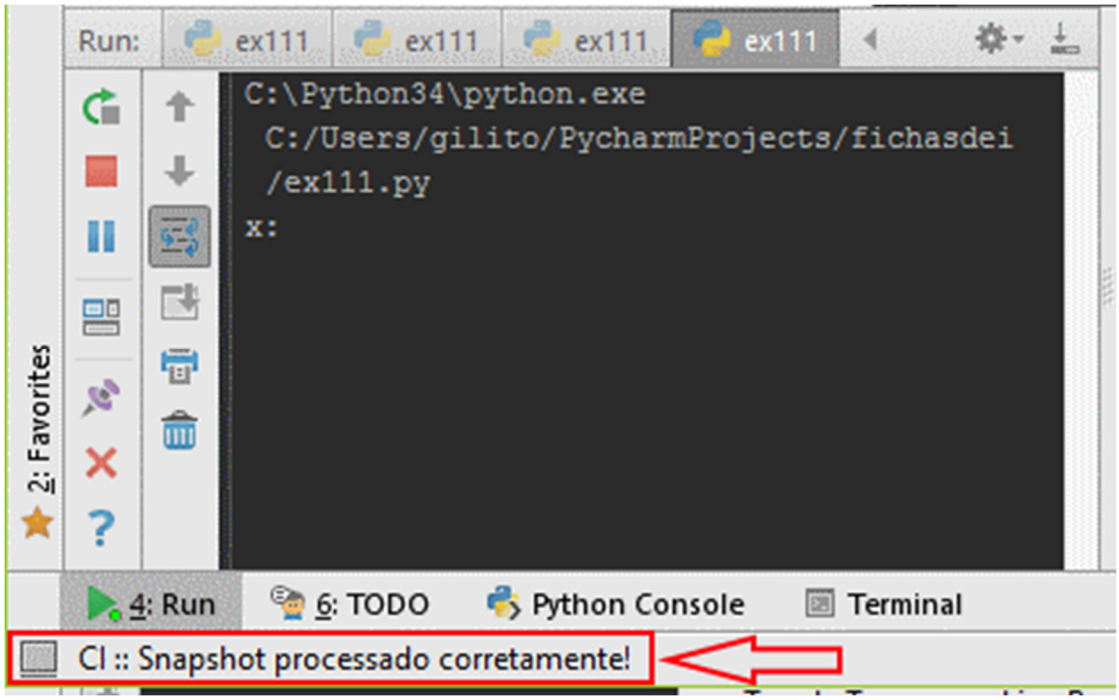

Figura 30 - Exemplo de mensagem do sistema apresentada no ambiente *PyCharm* - iteração um.

De modo a ser possível identificar o estudante que estava a submeter o código, assim como o exercício que estava a ser resolvido, deveriam ser adicionadas duas linhas de código no topo de todos os ficheiros, tal como apresentado na Figura 31:

```
__studentid__ = '1523233'
__assignmentid__ = '3.11'

# Breve descrição do exercício

print('hello!')
```

Figura 31 - Exemplo de código *Python* com os elementos identificativos - iteração um.

De modo a ganhar tempo, e para garantir que os estudantes não se esqueciam de fazer a correta identificação no código, foi sugerido aos estudantes que procedessem à alteração do *template* utilizado pelo *PyCharm* sempre que é criado um novo ficheiro `.py`. Para tal, bastava aceder às propriedades do editor, modificar o referido *template* e, daí em diante, sempre que fosse criado um novo ficheiro, era automaticamente inicializado com o código definido neste *template*. Na Figura 32 pode ser observado o processo de modificação do *template*.

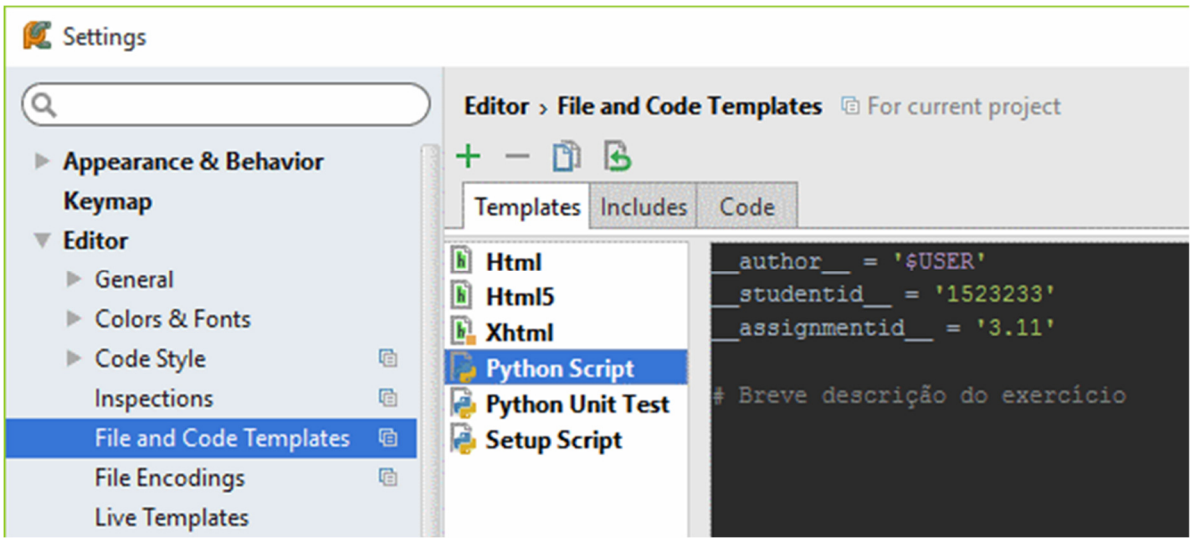

Figura 32 - Exemplo de definição do *template* do código no *PyCharm*.

Após ter sido terminado o desenvolvimento, o *plug-in* foi testado por um grupo restrito de utilizadores (colegas professores) que reportaram alguns pequenos erros que foram prontamente resolvidos, tendo em seguida sido concentrados os esforços no desenvolvimento da ferramenta de receção e processamento do código no servidor.

4.2.3.3. Receção e processamento do código fonte

Após serem recebidos no servidor, os *snapshots* do código fonte provenientes dos estudantes, eram processados de acordo com fluxo de eventos apresentado na Figura 33.

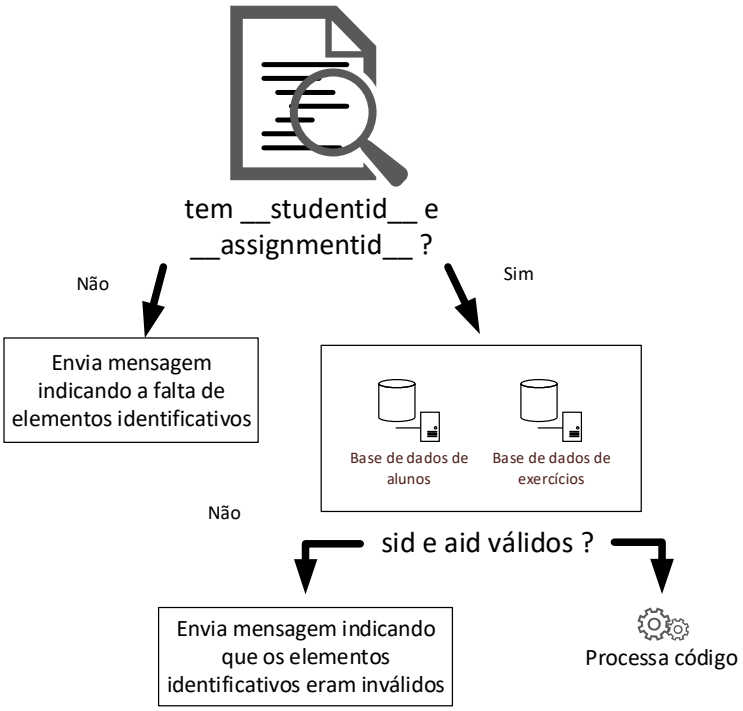

Figura 33 - Fluxo de processamento de um *snapshot* - iteração um.

Em traços gerais, iniciava-se a verificação do código fonte começando por analisar se o mesmo continha os elementos identificativos necessários (__studentid__ e __assignmentid__). Caso estes elementos identificativos não estivessem presentes no código, o processo terminava e era apresentada uma mensagem aos estudantes indicando a falta dos elementos identificativos (contudo, localmente no computador do estudante, o código era interpretado normalmente).

No caso de o código conter os referidos elementos identificativos seguia-se a verificação da validade dos mesmos. Em termos concretos, era verificado se os elementos identificativos de estudantes e exercícios correspondiam de facto a estudantes e exercícios existentes na base de dados.

Caso o identificador do estudante ou do exercício não estivessem corretos, o processo terminaria, sendo apresentada uma mensagem aos estudantes indicando o ocorrido. Em caso contrário, o processamento do snapshot prosseguia com a compilação e execução do código no servidor.

A base de dados de exercícios era constituída pelos exercícios disponibilizados pelos professores aos estudantes. Nesta iteração, para cada exercício era guardada informação sobre:

- o identificador do exercício no formato **`número da ficha.número do exercício`** (*e.g.* 2.1, ou seja, o primeiro exercício da segunda ficha);g
- o enunciado do exercício;
- a identificação dos principais conceitos de programação associados ao exercício (e.g. ciclos, instruções condicionais, etc.);
- informação sobre a dificuldade do exercício (MF - muito fácil, F - fácil, M - médio, D - difícil e MD - muito difícil).

Relativamente aos estudantes inscritos na unidade curricular em causa, era guardada informação sobre:

- o seu elemento identificativo (o número de estudante);
- o nome do estudante;
- a turma do estudante;
- informação sobre o facto de o estudante ser ou não repetente.

Pelo facto de a linguagem *Python* ser interpretada ao invés de compilada, não havia necessidade de duas fases distintas tal como tinha sido previsto para outras linguagens como C ou Java. Assim, bastaria executar o código e analisar o resultado produzido. No caso de existirem erros de interpretação ou de *run time*, era recolhida informação sobre esses erros e guardada no servidor informação sobre:

- a data e hora em que o *snapshot* foi recebido;
- o código fonte;
- os erros de compilação ou *run time* que tenham ocorrido.

No caso de não existirem erros de nenhuma natureza, era guardada informação sobre:

- a data e hora de receção do *snapshot*;
- o código fonte;
- o *output* produzido pelo programa.

Em ambos os casos era igualmente recolhida informação sobre o resultado de aferição das regras definidas para o exercício por forma a atualizar o modelo do conhecimento do estudante.

4.2.3.4. Construção e atualização do modelo do estudante

Como mencionado anteriormente, o processo de construção e atualização do modelo do conhecimento do estudante ficou em aberto na iteração anterior. Assim, e perante um novo contexto em que era utilizada uma linguagem de programação diferente daquela com que se havia trabalhado, houve necessidade de fazer algumas modificações ao nível da definição das regras disponíveis. Para este efeito, em colaboração com o professor da unidade curricular em que o sistema iria ser usado, deu-se início ao processo de definição do conjunto de regras que eram utilizadas para analisar o código produzido pelos estudantes. Na Figura 34 apresenta-se um exemplo da lista de regras utilizadas para analisar um exercício que tinha como objetivo proceder à substituição de todas as vogais de uma *string* por um espaço.

```
def anula_vogais(frase):

    black_list = 'aeiouAEIOU'

    for i in black_list:

        frase = frase.replace(i, ' ')

    return frase

print(anula_vogais_2('nuno miguel'))
```

- a função está bem definida e se recebe 1 parâmetro
- está a ser inicializada uma variável com as vogais
- está a ser feito um ciclo
- está a ser gerado um número aleatoriamente dentro do ciclo
- está a ser feita uma substituição dentro do ciclo
- está a ser feito a impressão ou retorno de algum valor
- está a ser chamada a função para poder saber o resultado

Figura 34 - Exemplo de lista de regras a utilizar para avaliar o código de um exercício - iteração um

Este processo de análise e definição de regras ocorreu com base em propostas de resolução dos exercícios. Tal como já havia sido verificado anteriormente, ficou evidente que era necessário um grande número de regras para conseguir aferir a qualidade das soluções submetidas pelos estudantes. Para além disso, ficou igualmente patente a enorme variedade de soluções que poderiam ser apresentadas para um simples exercício, incluindo algumas que o professor poderia não ter nunca considerado. A título de exemplo, o exercício anterior poderia ter sido igualmente resolvido com um ciclo `while`, ou poderia ter sido resolvido sem o recurso a funções, ou poder-se-ia iterar na *string* e não na variável `black_list`.

Perante esta situação, tornou-se evidente que a aplicação do conceito de regras baseadas em "sensores" não era facilmente operacionalizada e, como tal, acabou por ser abandonada.

O abandono da utilização das regras como havia sido idealizado anteriormente, teve implicações na configuração do modelo do conhecimento do estudante. Em termos concretos, restava apenas a utilização de informação sobre se o exercício estava correto ou não. Em todo o caso, ao longo deste processo de definição das regras foi sendo questionado ao professor que informação concreta gostaria de obter do sistema, tendo esta questão sido igualmente colocada a outros professores. Perante as respostas que foram sendo obtidas, foi-se formando a ideia de que a configuração do modelo do conhecimento do estudante idealizada era demasiado simplista tendo em conta a informação que os professores gostariam de conseguir obter do e no sistema. Tendo em consideração esta confluência de fatores, a utilização de um modelo do conhecimento do estudante baseado na utilização de uma rede bayesiana acabaria igualmente por ser posta de parte.

Contudo, continuava a ser necessário guardar a informação relativa ao conhecimento dos estudantes com base no seu desempenho. Assim sendo, acabaria por se adotar uma solução mais flexível, em que não era utilizado nenhum modelo formal. Em vez disso, optou-se por uma solução baseada na utilização de uma base de dados relacional (MySQL), sendo a informação processada de diversas formas, tendo em conta o que fosse pretendido fazer com ela. Deste modo, tendo por base exatamente o mesmo *input* que havia sido considerado inicialmente (o código fonte), era possível fornecer aos professores informação mais variada e detalhada sobre o desempenho dos estudantes.

4.2.3.5. Ferramenta de monitorização

Com base na nova configuração do sistema, para além da informação previamente apresentada, sempre que um novo *snapshot* era recebido procedia-se à recolha e armazenamento de algumas métricas sobre o código tais como:

- o número de linhas;
- o número de linhas de código (não vazias);
- o número de caracteres;
- a diferença do número de linhas face à tentativa anterior
- a diferença do número de caracteres face à tentativa anterior.

Com base na nova informação recolhida passaria a ser possível apresentar aos professores mais informação e de formas totalmente distintas das que haviam sido inicialmente idealizadas.

Muito embora seja uma presença habitual há já largos anos em áreas como economia, finanças, gestão de *utilities* entre outras, os *dashboards* educacionais só muito recentemente mereceram a atenção dos investigadores [124]. O principal objetivo de um *dashboard* é apresentar de uma forma simples e direta informação agregada sobre o desempenho, performance ou estado atual de uma determinada entidade (ou conjunto de entidades). Neste caso concreto, interessa particularmente a utilização de *dashboards* que possam apresentar informação sobre estudantes, turmas, etc. no que concerne ao seu desempenho ao nível da realização dos exercícios propostos pelos professores.

Verbert et al. [125] [126] levaram a cabo um interessante estudo sobre este tipo de sistemas. Neste trabalho classificaram os *dashboards* educacionais em vários grupos de acordo com o tipo de aulas que se pretende que suportem:

- aulas presenciais tradicionais;
- trabalhos de grupo realizados em aulas presenciais tradicionais;
- aulas de *blended-learning* e *e-learning*.

Segundo Verbert et al., a informação a apresentar no *dashboard* deverá ser relevante para a atividade do professor e sempre que tal seja possível, deverão ser apresentados resultados de agregação/processamento de dados, ao invés de dados em bruto. Desta forma o professor poderá ter uma visão geral sobre os principais aspetos que permitam caracterizar o desempenho do(s) estudante(s), sem necessitar perder tempo a analisar os dados. Em todo o caso, poderá naturalmente ser dado acesso aos professores a dados em bruto e/ou a informações mais detalhadas, caso assim o entendam.

Ainda segundo Verbert et al., sempre que possível, dever-se-á dar preferência à utilização de gráficos ou outras formas de visualização de dados, ao invés de utilizar texto ou dados na forma de tabelas. Ou seja, a informação deve ser fornecida ao professor de modo que este a possa ver e assimilar em pouco tempo e sem que para tal tenha de se esforçar. Uma vez mais, caso pretenda saber mais detalhes, deverá ser-lhe dada essa possibilidade.

Um aspeto importante tem que ver com a organização do espaço. A este respeito dever-se-á garantir que os diferentes elementos do *dashboard* se encontram devidamente separados e ordenados em função da sua relevância (do mais relevante para o menos relevante). Além disso, deverá garantir-se igualmente que os elementos principais do *dashboard* poderão ser visualizados sem necessitar de fazer *scroll* e a informação deverá ser atualizada periodicamente e de forma automática.

Em traços gerais, o *dashboard* deverá conter de forma bem organizada a informação mais relevante para a atividade do professor e esta informação deve poder ser assimilada no menor espaço de tempo possível e com o menor esforço possível.

Com base neste conjunto de considerações, para permitir aos professores monitorizar o desempenho dos seus estudantes, o sistema ***CodeInsights*** disponibiliza um *dashboard* com informação agregada sobre o desempenho dos estudantes. Na Figura 35 é apresentada uma vista parcial do *dashboard* que haveria de ser utilizado pelo professor nesta iteração e no qual é apresentado um resumo do estado atual dos estudantes da(s) sua(s) turma(s).

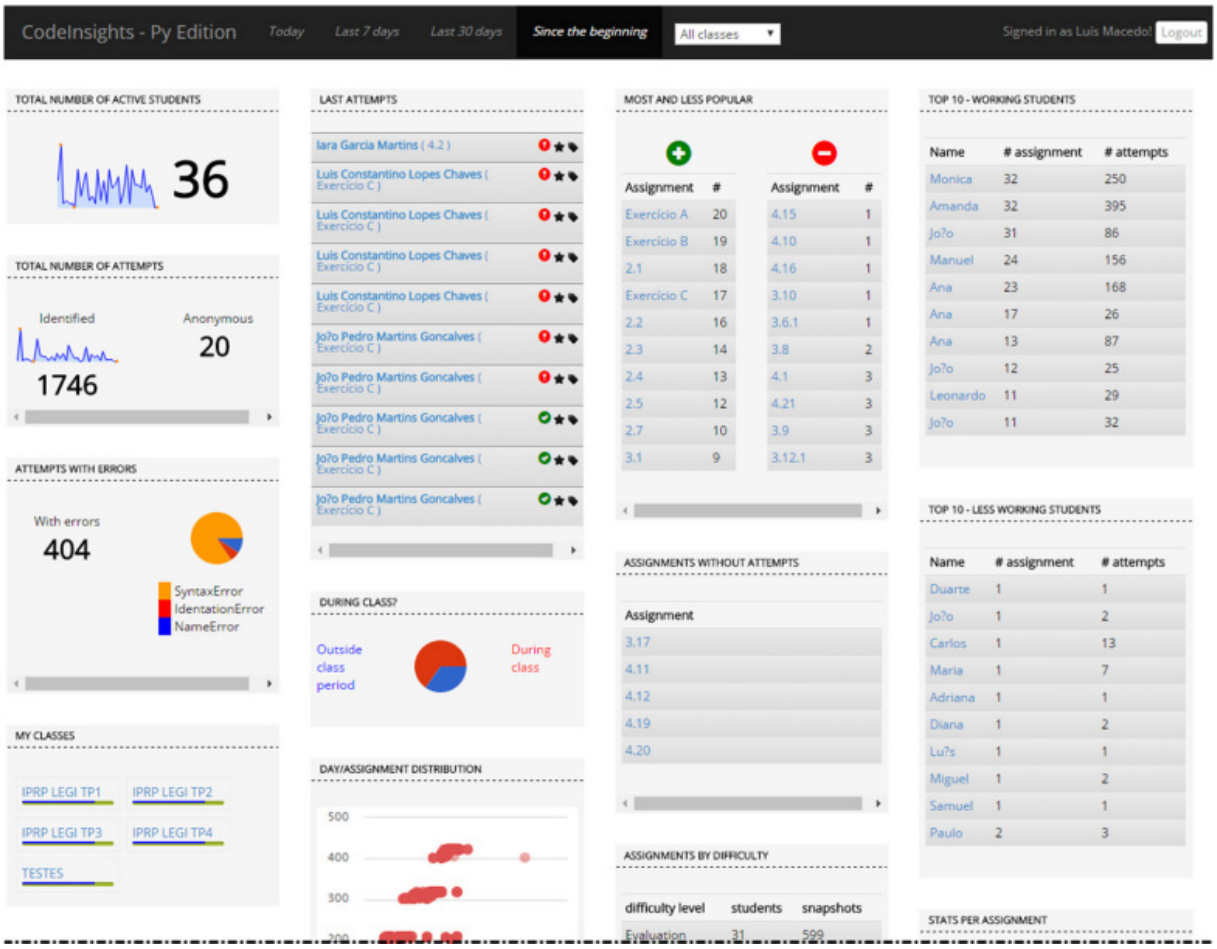

Figura 35 - Vista parcial do *dashboard* da ferramenta de monitorização - iteração um.

Ainda que tivesse sido dada preferência à utilização de elementos gráficos, em algumas situações considerou-se pertinente apresentar ao professor dados em forma tabular. Assim, neste *dashboard* o professor tinha acesso a um conjunto vasto de informações sobre o desempenho dos seus estudantes, tais como:

- Número de estudantes ativos;
- Número de *snapshots* recebidos;

- Número de *snapshots* com erros e distribuição em função dos erros (e.g. SyntaxError, NameError, IdentationError, etc.);
- Lista de exercícios mais populares (top 10) e menos populares (top 10);
- Lista de exercícios sem qualquer tentativa de resolução;
- Lista de estudantes com mais exercícios com pelo menos uma tentativa de resolução;
- Lista de estudantes com menos exercícios com pelo menos uma tentativa de resolução;
- Distribuição do número de estudantes e número total de *snapshots* em função da dificuldade do exercício;
- Distribuição dos *snapshots* em função do âmbito de submissão dos mesmos (em período de aula ou fora do período da aula)
- Últimos 10 *snapshots* recebidos com informação sobre o exercício, nome do estudante e se continha erros de compilação ou erros de *run time*.

O professor tinha, portanto, acesso a um conjunto alargado de informações agregadas sobre o desempenho dos estudantes que podia ser filtrada, por forma a serem apresentados os resultados em relação ao **dia corrente**, para os **últimos sete dias**, para o **último mês**, **desde o início** ou para um **intervalo específico**. Era ainda possível que o professor visse a informação agrupada relativamente a todas as turmas ou apenas em relação a uma turma específica e aceder a informações detalhadas sobre cada tópico, caso assim entendesse.

Esta visão global permitia de uma forma simples, a identificação de situações que mereciam a sua atenção, como por exemplo estudantes que pudessem estar a ter dificuldades e/ou a ficar atrasados face aos colegas, ou diferenças substanciais de desempenho entre turmas.

Ao clicar no nome de um estudante era apresentada uma página com um resumo de todos os exercícios que este estudante tentou resolver, assim como o estado da última tentativa de resolução, tal como pode ser observado na Figura 36.

Ao clicar num exercício em específico tinha-se acesso aos detalhes sobre todas as tentativas de resolução do estudante selecionado para aquele exercício em particular, tal como representado na Figura 37.

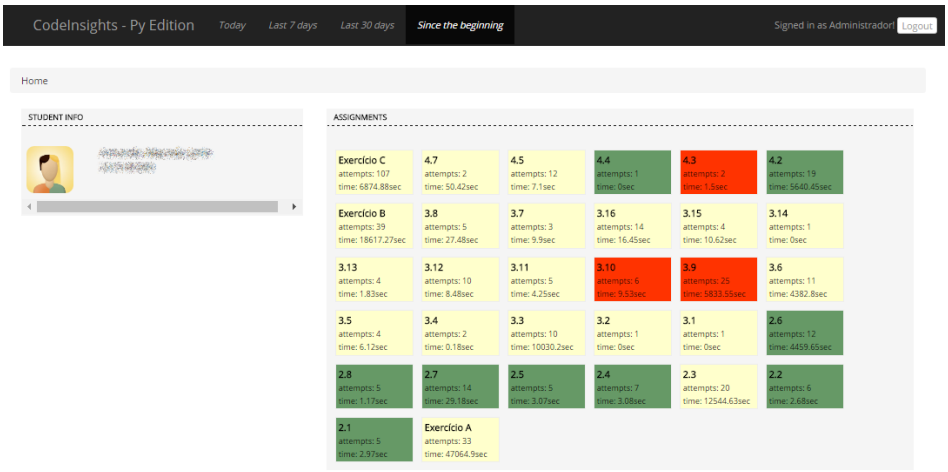

Figura 36 - Resumo de todos os exercícios que um estudante tentou resolver - iteração um.

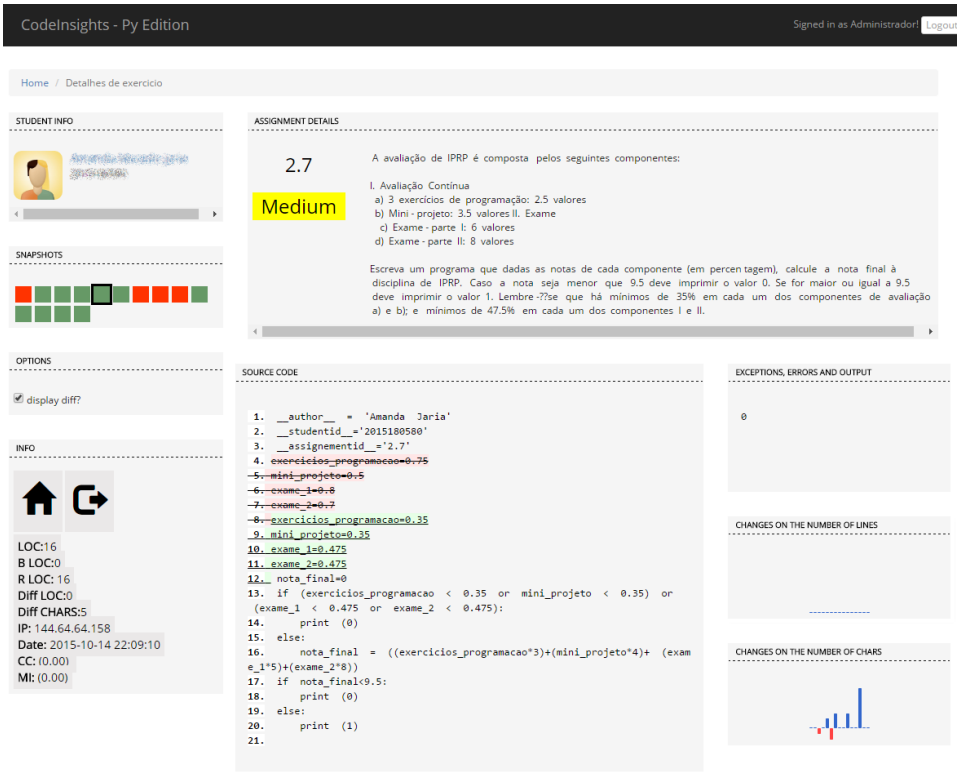

Figura 37 - Detalhes de um *snapshot* após ser processado - iteração um.

Na página ilustrada na Figura 37 é possível visualizar o **código fonte** submetido, os **erros de compilação/interpretação** (caso existam), o ***output*** de resultado (caso seja possível determinar) e um **conjunto de métricas** como o nº de linhas de código, as diferenças em relação à versão anterior, o valor de *ciclomatic complexity*, etc.

Em relação ao código fonte, é possível verificar as diferenças em relação à versão anterior (caso exista) e aceder aos detalhes de uma qualquer versão anterior do código fonte.

Com base na informação agregada e mais detalhada, os professores podem ter uma noção exata sobre o trabalho que cada estudante desenvolveu, permitindo-lhes deste modo identificar situações que possam merecer a sua intervenção imediata junto dos estudantes, ou intervenções de médio prazo tais como adaptar o ritmo das aulas.

O sistema poderá igualmente ser utilizado com um horizonte temporal mais alargado. A título de exemplo, no final do ano, o professor poderia proceder a uma análise dos dados recolhidos, por forma a preparar edições futuras da unidade curricular. Imagine-se o seguinte cenário: o professor constata que para determinados exercícios foram feitas muito poucas submissões (ou eventualmente nenhuma). Perante tal facto talvez seja pertinente verificar se os exercícios em causa não são redundantes ou se os enunciados não estarão eventualmente incompletos ou confusos.

Um outro exemplo da possível utilização do sistema depois de as aulas terem terminado, diz respeito ao estudo do perfil dos estudantes. *Como foi o desempenho dos estudantes que reprovaram? E o desempenho dos estudantes que passaram? Quais os exercícios ou conceitos em que tiveram mais dificuldades? Trabalhavam mais durante as aulas ou fora destas?*

Mesmo que as modificações que o professor faça a este nível possam já não vir a surtir efeito no ano letivo atual, poderão surtir efeito no ano seguinte, podendo deste modo haver ganhos significativos em edições futuras.

4.2.4. Avaliação do sistema

Tal como previsto, o sistema foi avaliado em contexto real com as turmas previamente apresentadas entre os dias 10 de outubro e 17 de novembro de 2015. O principal objetivo desta avaliação foi verificar se a informação recolhida e apresentada aos professores era de facto útil, se o sistema permitia efetivamente ter uma visão mais aprofundada sobre o desempenho dos estudantes e se poderia realmente ser utilizada no dia-a-dia como uma ferramenta de apoio para intervenções junto dos estudantes e para a tomada de decisões de alcance mais abrangente.

Estavam inscritos nesta unidade curricular um total de 101 estudantes (65 do sexo masculino e 36 do sexo feminino). Entre estes estudantes, aproximadamente 50% eram repetentes.

Ao longo desta experiência, foram propostos aos estudantes cerca de 40 exercícios associados a três tópicos distintos (estruturas de seleção, ciclos e manipulação de

strings). Destes exercícios, apenas três (um por tópico) eram tidos em consideração para efeitos de avaliação.

Em edições anteriores desta unidade curricular, não eram feitas quaisquer submissões das soluções dos "exercícios de treino" e apenas a versão final da resolução dos exercícios tidos em consideração para efeitos de avaliação era submetida através do sistema informático em uso na Universidade de Coimbra (Nónio), não tendo o professor informação sobre o processo de resolução dos mesmos, para além do que podia observar presencialmente nas aulas quando estava com os estudantes.

A participação na experiência foi voluntária e não tinha nenhuma influência na definição da nota final da unidade curricular. Com efeito, alguns estudantes optaram por não instalar o *plug-in*, ao passo que alguns instalaram-no inicialmente, tendo procedido à sua desinstalação numa fase posterior.

Convirá referir que durante as aulas muitos estudantes trabalhavam em pares, sendo que este cenário de trabalho em pares/grupos não tinha sido considerado. Por conseguinte, foram acontecendo situações em que apenas eram enviados *snapshots* por parte de um dos elementos do grupo ou, em casos menos frequentes, eram recebidas submissões iguais de ambos os estudantes, sendo que um deles submetia apenas as versões finais.

No primeiro dia da experiência foi detetado um problema que foi necessário corrigir de imediato e que se prendia com o facto de existirem muitos estudantes a cometerem erros de identificação ao nível do ID do estudante e/ou do exercício. Como apresentado anteriormente, existia a possibilidade de ser criado um *template* que ajudaria muito neste processo, mas alguns estudantes tiveram dificuldades em perceber o processo, tendo alguns deles introduzido erros no próprio *template*. Como previsto inicialmente, sempre que um *snapshot* apresentasse problemas de identificação era simplesmente descartado, sem ser sequer processado. Ora, perante os problemas que estavam a existir ao nível da identificação, esta forma de proceder levou a que fosse descartada informação relevante.

Assim, foi necessário proceder ao desenvolvimento de uma nova funcionalidade que consistia em não descartar os *snapshots* com problemas de identificação, mas, como alternativa, adicioná-los a uma base de dados de *snapshots* não identificados. Posteriormente, o professor poderia aceder a esta lista e associar os *snapshots* ao

respetivo estudante e/ou exercício ou descartá-lo efetivamente caso não fosse de todo possível proceder à sua identificação. Este processo pode ser observado na Figura 38.

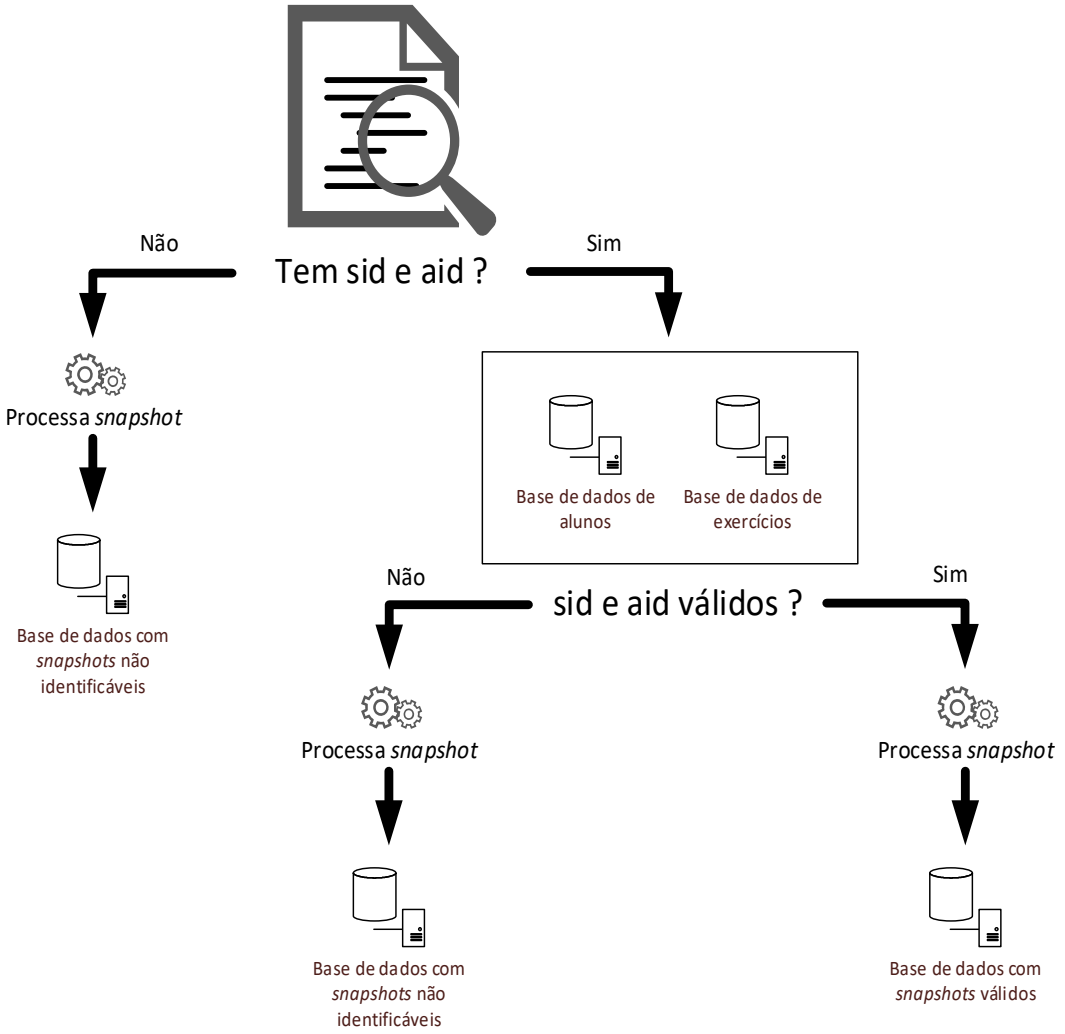

Figura 38 - Representação do novo processo de identificação dos *snapshots* - iteração um.

Convém mencionar que os erros mais frequentes ao nível da identificação estavam associados a erros ortográficos ou falta de caracteres. Por exemplo, em vez de **__assignmentid__**, alguns estudantes escreviam **_assignmentid_** ou **__assignemntid__**, assim como outras variações da mesma palavra. Em outras situações, os estudantes esqueciam-se de alterar o número do exercício que estava definido por omissão no *template* (0.0) pelo número do exercício que estavam a resolver. Também estes *snapshots* eram considerados como não estando identificados corretamente e, por conseguinte, ao invés de serem descartados automaticamente, eram guardados na referida base de dados.

No total da experiência foram recebidos 2967 *snapshots*, sendo que destes apenas 1746 foram considerados válidos. Entre os *snapshots* considerados inválidos encontram-se principalmente *snapshots* com problemas de identificação, mas também um grande número de *snapshots* que, na prática, consistiam em pequenos pedaços de código utilizados pelos estudantes para testar algum pormenor particular do código (por vezes apenas uma linha de código).

Foram recebidos *snapshots* de 36 estudantes distintos, o que apesar de tudo é um número interessante, principalmente se for tido em consideração que, tal como já foi referido, muitos estudantes trabalhavam em par. Contudo, há que mencionar que relativamente a nove estudantes foram apenas recebidos *snaphots* relativos a um exercício. Tal aconteceu porque os estudantes, entretanto começaram a trabalhar em pares ou porque simplesmente procederam à desinstalação do *plug-in*. É igualmente importante salientar que, muito embora estivessem inscritos cerca de 100 estudantes à unidade curricular, mais de 30 estudantes nunca assistiram a nenhuma aula, nem participaram em qualquer momento de avaliação.

Tendo em conta o panorama apresentado, e tendo em consideração que se tratou da primeira vez que o sistema foi utilizado em contexto real, o número de estudantes participantes e o número de *snapshots* recebidos (válidos ou não) ficaram ligeiramente aquém das expectativas. Em todo o caso, foi uma primeira experiência útil e produtiva na medida em que se conseguiu ter uma perceção mais real, prática e eficaz de melhorias a implementar no sistema.

Após o final da experiência, foi possível fazer a confrontação entre os estudantes que assistiam às aulas (com base nas folhas de presenças) e a informação recolhida pelo sistema. Um aspeto interessante que sobressaiu desta análise diz respeito ao facto de diversos estudantes não terem assistido a muitas aulas, mas mesmo assim terem trabalhado na resolução dos exercícios propostos pelos professores. Caso o sistema não estivesse a ser utilizado, o professor não teria forma alguma de acompanhar o desempenho destes estudantes.

Outros estudantes, embora assistissem às aulas, davam continuidade à resolução dos exercícios fora do período destas. Com efeito, aproximadamente 35% dos *snapshots* foram recebidos em períodos fora das aulas desta unidade curricular, sendo este comportamento transversal às quatro turmas. Estes dados são positivos, pois revelam que os estudantes não se limitam a trabalhar no período em que estão nas aulas.

Um outro aspeto que foi interessante verificar diz respeito à distribuição das submissões pelos diferentes dias da semana e horas do dia. Na Figura 39 está representada a relação entre os dias da semana (XX - em que 0 corresponde a domingo e 6 a sábado) e horas do dia (YY - 0h00 a 23h59) para as quatro turmas.

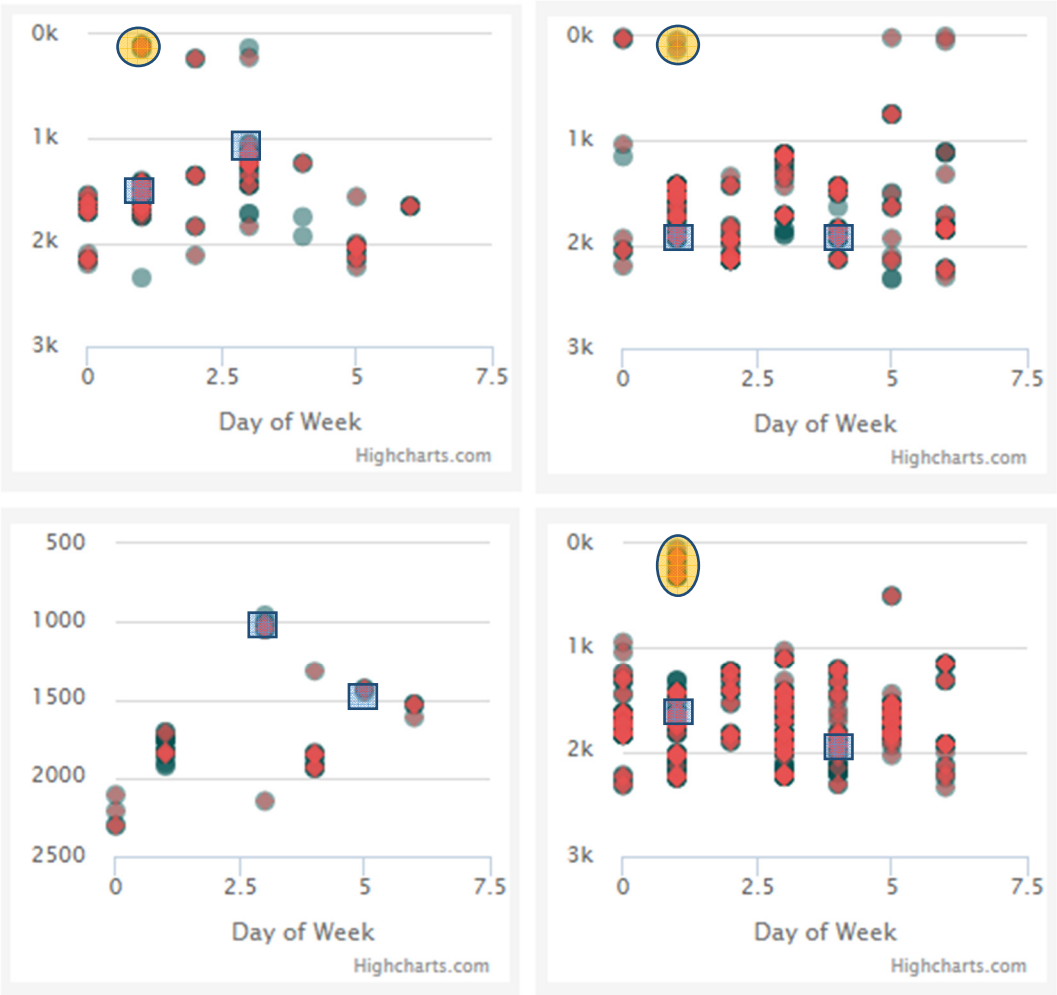

Figura 39 - Distribuição dos *snapshots* em função da hora/dia da semana

Já anteriormente foi indicado que havia um elevado número de submissões efetuadas fora do âmbito das aulas (quadrados na figura). Este facto fica bem patente na Figura 39. É também possível verificar que existe um aumento significativo de submissões efetuadas na madrugada de domingo para segunda-feira (círculos amarelos na figura). Tal facto não é de estranhar, uma vez que o horário fixado para a entrega dos exercícios para avaliação era precisamente às 9h de segunda-feira.

Cerca de 75% dos *snapshots* válidos recebidos foram submetidos por 10 estudantes (ou grupos de estudantes). Na Tabela 1 pode ser observado um resumo do número de exercícios de *snapshots* recebidos e exercícios que estes estudantes tentaram resolver.

Ao analisar estes dados fica claro que foram recebidos consideravelmente mais *snapshots* de alguns estudantes do que de outros. Para além disso, foi possível constatar que a grande maioria dos *snapshots* foram recebidos nas primeiras aulas da experiência, tendo este número diminuído consideravelmente à medida que o tempo foi passando. Contudo, houve igualmente oportunidade de constatar que este comportamento era consistente com a evolução do número de estudantes que assistia às aulas e que, por sua vez, era um fator igualmente consistente com o comportamento observado em anos anteriores. Ou seja, em termos práticos o sistema estava de facto a refletir o que acontecia na realidade.

Tabela 1 - Top 10 dos estudantes com maior número de exercícios com pelo menos um *snapshot*.

Estudante/grupo	**Número de exercícios**	**Número de *snapshots***
A	32	395
B	32	252
C	31	86
D	24	157
E	23	168
F	17	26
G	13	87
H	12	25
I	11	88
J	11	29

Ao analisar estes dados fica claro que foram recebidos consideravelmente mais *snapshots* de alguns estudantes do que de outros. Para além disso, foi possível constatar que a grande maioria dos *snapshots* foram recebidos nas primeiras aulas da experiência, tendo este número diminuído consideravelmente à medida que o tempo foi passando. Contudo, houve igualmente oportunidade de constatar que **este comportamento era consistente com a evolução do número de estudantes que assistia às aulas** e que, por sua vez, era um fator igualmente consistente com o comportamento observado em anos anteriores. Ou seja, em termos práticos o sistema estava de facto a refletir o que acontecia na realidade.

Muito embora os dados da Tabela 1 por si só não permitam obter informação relativamente à qualidade dos *snapshots* submetidos, será interessante observar que, de todos estes estudantes, apenas o estudante J não obteve aprovação no final da unidade curricular. Foi ainda interessante constatar que os estudantes A e B foram os que obtiveram as classificações mais elevadas no final do semestre de entre os estudantes que participaram na experiência.

Um outro aspeto interessante e diz respeito aos exercícios mais populares e menos populares em função do número de estudantes que submeteu pelo menos um *snapshot* relativo aos mesmos. Os resultados podem ser observados na Tabela 2 e Tabela 3.

Tabela 2 - Exercícios mais populares da iteração um

Exercício	Número de estudantes/grupos
A (avaliação)	22
2.1	20
B (avaliação)	20
C (avaliação)	17
2.2	17
2.4	15
2.3	15
2.5	13
3.1	10
2.7	10

Tabela 3 - Exercícios menos populares da iteração um

Exercício	Número de estudantes/grupos
3.6.1	1
4.15	1
4.10	1
4.16	1
3.10	1
3.8	2
3.15	3
4.18	3
4.1	3
4.21	3

Como é possível observar na Tabela 2, entre os exercícios mais populares encontram-se aqueles que contavam para efeitos de avaliação, assim como a maioria dos exercícios da primeira ficha prática envolvendo programação (a número 2). Este facto é interessante e revelador de um fenómeno que viria a ser confirmado pelo professor e que dizia respeito ao facto de muitos estudantes apenas irem às aulas na primeira semana de aulas e nas restantes aulas dedicadas aos exercícios que contavam para efeitos de avaliação. Relativamente aos dados da Tabela 3 é curioso verificar que os exercícios com menor número de estudantes a tentar resolvê-los correspondem aos últimos exercícios das fichas práticas. A maior parte destes exercícios tinha um nível de dificuldade médio a difícil, o que por si só acaba por contribuir para explicar os números apresentados. Ainda a respeito da dificuldade dos exercícios, verificou-se que existiram cinco exercícios para os quais não foi recebido qualquer *snapshot*, sendo que todos estes exercícios eram considerados difíceis ou muito difíceis.

Tendo em conta que os enunciados das fichas práticas eram publicados com todos os exercícios de uma só vez, incluindo os exercícios que contavam para efeitos de avaliação, o comportamento observado anteriormente poderá ser revelador de que os estudantes começavam a resolver os exercícios iniciais das fichas de trabalho e, assim que se sentiam minimamente à vontade, começavam a tentar resolver os exercícios que contavam para efeitos de avaliação. Este comportamento poderá eventualmente ser problemático e vicioso e, como tal, deveria merecer a atenção do professor e eventual mudança de estratégia pedagógica na estruturação das fichas práticas.

Ao considerar os resultados apresentados, um aspeto ao qual o professor poderá considerar diz respeito à indicação da dificuldade do exercício no enunciado. Muito embora seja compreensível que o professor possa querer oferecer estímulo e diversidade através de diferentes níveis de dificuldade explicitando quais são os exercícios mais difíceis, a inclusão desta informação nos enunciados poderá ser de algum modo intimidatória, tal como os resultados acabaram por demonstrar.

Um outro aspeto curioso está relacionado com o desempenho dos estudantes repetentes. Muito embora representassem aproximadamente metade da população discente da amostra, apenas foram recebidos *snapshots* de 13 estudantes (contra 23 não repetentes). Também a nível do número de *snapshots* foi verificada uma diferença considerável. Foram recebidos 596 *snapshots* de estudantes repetentes e 1150 de estudantes não repetentes, sendo que destes 596 a grande maioria era relativa aos três exercícios que contavam para efeitos de avaliação. Este facto poderá eventualmente ser explicado pelo facto de estes estudantes já terem tido outra unidade curricular de programação no segundo semestre do ano anterior na qual também era utilizada a linguagem *Python*. Este cenário poder-lhes-á ter dado alguma confiança extra conduzindo-os a que sentissem que não necessitavam resolver novamente os exercícios mais básicos sem influência direta na nota final.

Na Figura 40 pode ser observada a distribuição das submissões para os diferentes exercícios ao longo do tempo. Cada *drop* (círculo) representa um conjunto de submissões num determinado dia, sendo que quanto maior for o número de submissões daquele exercício naquele dia, maior será o diâmetro do *drop*. Esta figura pode ajudar a identificar situações em que existam estudantes que estejam a resolver os exercícios claramente mais tarde que os restantes colegas. No caso concreto dos *drops* destacados na Figura 40, está-se perante um conjunto de submissões efetuadas com várias semanas

de atraso em relação à maioria dos colegas. Estas situações devem ser identificadas e deve ser dado algum apoio especial a estes estudantes.

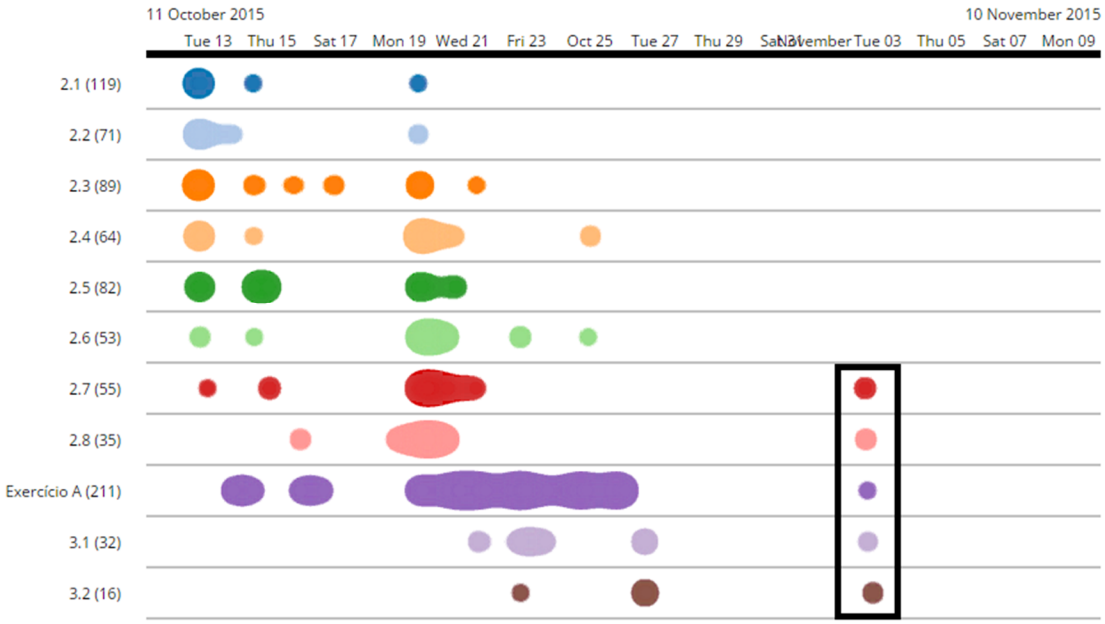

Figura 40 - Submissões efetuadas tardiamente - iteração um.

Este tipo de visualização permite igualmente identificar ocorrências em que existam estudantes a fazer submissões para um determinado exercício antes do tempo considerado normal. Na Figura 41 encontra-se destacada uma situação em que foram feitas submissões para um exercício (a contar para efeitos de avaliação) algum tempo antes da maioria dos colegas.

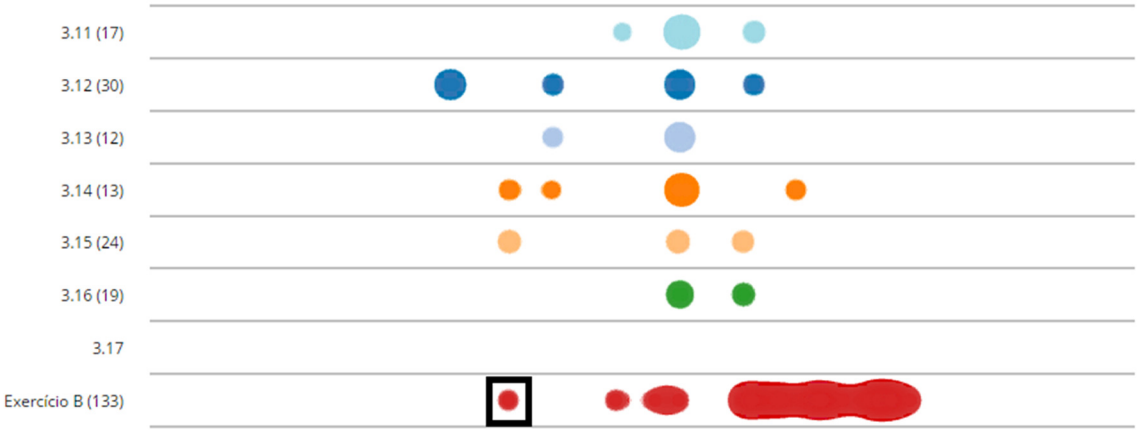

Figura 41 - Submissões efetuadas antes do tempo

Isto pode ser indicativo de que um estudante, ou grupo de estudantes, não estão a seguir os exercícios das aulas, e que se preocupam principalmente em resolver os exercícios que contam para efeitos de avaliação. Este facto pode ser revelador para os professores, pois

poderão ser estudantes com excesso de confiança e que desejam demonstrar conseguir fazer algo que está acima de suas reais possibilidades.

Um outro aspeto que a ferramenta desenvolvida permitiu observar concerne às submissões com erros de compilação (interpretação) e caracterização desses mesmos erros. Assim, do total de 1752 tentativas de resolução recebidas, 413 tinham erros de compilação/interpretação distribuídos como apresentado na Tabela 4.

Tabela 4 - Caracterização dos erros de compilação

Erro	Nº de submissões
SyntaxError	350
NameError	37
IdentationError	26

Ainda que a experiência tenha sido relativamente curta, e com uma participação relativamente limitada, foi possível verificar que, utilizando apenas os *snapshots* do código escrito pelos estudantes, era possível ter uma noção mais clara do desempenho dos estudantes de cada turma e identificar situações para a qual a intervenção do professor poderia ser aconselhável e que eram difíceis de identificar sem o recurso ao sistema. Tornou-se, pois, evidente que ainda que no passado o professor fizesse o possível para identificar os estudantes com mais dificuldades, a utilização do sistema tornava este processo muito mais simples e com o benefício de não expor as fragilidades dos estudantes à frente dos colegas.

Curiosamente, no final do semestre, a taxa de aprovação foi consideravelmente superior à verificada no ano anterior. Contudo, existiram diversos fatores a influenciar este processo. Não se pode afirmar que estes resultados foram obtidos como consequência de ter sido utilizado o sistema. Ainda assim, é possível afirmar que a informação que o sistema foi apresentando ao longo da experiência ao professor foi sendo consistente com a realidade. Ainda que tivesse sido mantido um diálogo permanente com o professor ao longo de toda a experiência, após esta ter terminado teve lugar uma reunião na qual o próprio assegurou que efetivamente a informação fornecida pelo sistema correspondia de verdade à perceção que ele tinha relativamente aos estudantes que utilizaram o sistema e que, como tal, este sistema poderá, de forma muito útil e objetiva, ser utilizado para melhorar a perceção que os professores têm sobre o desempenho dos estudantes.

4.2.5. Sumário da iteração um

Ao longo desta primeira iteração, a utilização do modelo do estudante suportado por uma rede bayesiana, revelou-se demasiado difícil de operacionalizar, tendo a sua utilização acabado por ser abandonada, como já foi explicado anteriormente. Para além da questão da operacionalização, foi crucial para este culminar o interesse por parte do professor em obter outro tipo de informações que o modelo em causa não conseguia suportar.

No geral é possível afirmar que foram feitos progressos importantes nomeadamente a nível do desenvolvimento do *plug-in* e das ferramentas de processamento dos *snapshots* e de apresentação da informação aos professores. Ainda que tenham sido identificados alguns problemas, muitos foram sendo adaptados e corrigidos de imediato, tendo outros acabado por ficar para a iteração seguinte, como por exemplo a simplificação do processo de identificação dos exercícios e estudantes, uma vez que esta questão se revelou problemática ao longo desta iteração.

Nesta iteração acabaram também por ainda não ser utilizados os casos de teste para efeitos de teste do código com um conjunto de *inputs* predefinidos. Assim, não foi possível verificar automaticamente se as soluções estavam efetivamente corretas ou não, apenas era possível saber se o código continha ou não erros de interpretação ou *run time*, facto que constitui obviamente uma importante limitação.

Em todo o caso, é conveniente referir que se tratou da primeira iteração em que o sistema foi utilizado em contexto real (e regular) de sala de aulas. Sendo pertinente destacar algumas das principais observações que foram sendo feitas, muitas das quais passariam despercebidas, se o sistema não estivesse a ser utilizado.

Com efeito, ao longo desta experiência foi possível verificar que muitos estudantes não assistiam às aulas, mas que ainda assim continuavam a trabalhar. Por este motivo, considera-se ser de facto importante dar aos professores a possibilidade de enviar *feedback* diretamente pelo sistema tal como havia sido inicialmente previsto.

O professor responsável pela unidade curricular revelou que durante as aulas identificou alguns estudantes que executavam o seu código utilizando as opções de menu e botões padrão do *PyCharm,* e não as versões ***CodeInsights***, o que se traduziu no não envio de *snapshots* do código para o servidor. Este era um importante aspeto a ser tido em conta futuramente.

4.3. Iteração dois

Tendo como ponto de partida os resultados da avaliação efetuada na iteração um, deu-se início à iteração dois.

4.3.1. Objetivos

Pelo facto de a maior parte das unidades curriculares introdutórias de programação ocorrerem geralmente no primeiro semestre, não foi possível testar o sistema no segundo semestre do ano letivo 2015/2016. No entanto, foram sendo estabelecidos contactos tendo em vista a utilização do sistema no primeiro semestre do ano letivo 2016/2017 em contextos consideravelmente distintos.

Este hiato entre experiências acabou por se revelar importante já que permitiu efetuar modificações significativas em diversos aspetos do sistema. Assim, em grande medida, os principais objetivos desta iteração passaram por efetuar as referidas modificações, muitas delas resultantes do processo de avaliação da iteração anterior, assim como proceder ao desenvolvimento de novas funcionalidades que, entretanto, se revelaram relevantes. Terminado estes desenvolvimentos foi necessário proceder à sua validação em contexto real.

4.3.2. Caracterização do contexto

Nesta iteração, o sistema ***CodeInsights*** foi testado em três unidades curriculares distintas no primeiro semestre do ano letivo 2016/2017: uma unidade curricular de introdução à programação com a linguagem C (Fundamentos de Programação da Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital), uma unidade curricular de introdução à programação orientada a objetos com Java (Programação da Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital) e uma unidade curricular de introdução à programação em *Java* (programa *Acertar o Rumo* da Universidade de Coimbra).

Estes novos contextos acabariam por constituir novos desafios quer ao nível dos ambientes de desenvolvimento, quer ao nível das linguagens de programação utilizadas, mas também ao nível da própria natureza das unidades curriculares e dos estudantes.

4.3.3. Trabalho desenvolvido

4.3.3.1. Desenvolvimento dos *plug-ins*

Tendo em consideração que os professores das unidades curriculares em que foi utilizada a linguagem *Java* sempre haviam utilizado o ambiente de desenvolvimento *Eclipse*, foi necessário recuperar o desenvolvimento do *plug-in* para *Eclipse* que tinha sido iniciado, mas, entretanto, interrompido. Convirá relembrar que, tal como mencionado anteriormente, a principal dificuldade prendia-se com o facto de não ser possível associar o envio dos *snapshots* à ação "*Run*" já disponibilizada pelo *Eclipse*. Tal como a experiência da iteração um demonstrou, a utilização de botões e/ou opções diversas das disponibilizadas por omissão pelos ambientes de desenvolvimento, poderia constituir um problema, devendo por esse motivo ser evitada.

Após muita pesquisa, foi finalmente encontrada uma forma de fazer o pretendido e, consequentemente, o desenvolvimento do *plug-in* para o *Eclipse* (Neon) foi dado como terminado, permitindo nesta versão o envio dos *snapshots* do código para o servidor mediante a utilização das ações "*Run*" disponibilizadas pelo *Eclipse* (menu, botão e teclas de atalho **Ctrl + F11**).

Tal como acontecia no caso do *PyCharm* (e restantes ambientes de desenvolvimento da *JetBrains*), também no *Eclipse* poderia ser definido um *template* base a ser utilizado sempre que o estudante procedia à criação de um novo ficheiro.

Tendo em conta a experiência da iteração anterior, na qual diversos estudantes cometiam erros aquando da criação do *template*, procedeu-se a uma simplificação significativa do processo de identificação tendo como principal objetivo a redução dos erros de identificação. Na Figura 42 pode ser observado um exemplo da nova configuração dos elementos identificativos. Assim, passaria a ser utilizada a *tag* **@sid** para a identificação do estudante (*student id*) e **@aid** para a identificação dos exercícios (*assignment id*).

```
/**
* Breve descrição do código
*
* @sid 123456
* @aid 1.1
*/
```

Figura 42 - Elementos identificativos dos *snapshots* para as linguagens C e Java - iteração dois.

No caso concreto da unidade curricular de Fundamentos de Programação, o professor indicou que regra geral não era utilizado nenhum ambiente de desenvolvimento em concreto, acabando os estudantes por utilizar um editor de código simples, compilando e executando o seu código diretamente via terminal/ linha de comandos. Assim, de modo a não interferir com o habitual funcionamento da unidade curricular, optou-se por desenvolver um plug-in de modo a poder ser utilizado pela linha de comandos. Em termos concretos, foram desenvolvidos dois plug-ins (um para ***Windows*** e outro para ***Linux***) para o envio dos *snapshots* do código para o servidor.

4.3.3.2. Receção e processamento do código fonte

Uma vez que a ferramenta de receção e processamento do código só havia sido testada em contexto real com a linguagem *Python*, houve a necessidade de repensar a estrutura do sistema de modo a permitir suportar várias linguagens de programação utilizando sempre a mesma base de código. Como tal procedeu-se à separação total de todos os componentes dependentes de uma determinada linguagem de programação. Deste modo, no momento da instalação do sistema, introduz-se a definição da linguagem de programação a ser utilizada naquele caso em concreto e, deste modo, serão automaticamente utilizados os componentes associados a essa linguagem.

4.3.4.3. Arquitetura do sistema

Uma vez que iriam ser realizadas modificações profundas relacionadas com o suporte de diversas linguagens de programação, foi igualmente decidido separar totalmente o componente responsável pela compilação/interpretação e execução do código (*sand-box*) de modo a este poder ser instalado numa máquina distinta da que recebe e processa os *snapshots*.

Esta separação foi motivada sobretudo por motivos de desempenho do sistema, mas também por motivos de segurança. Com esta nova configuração, caso ocorresse algum problema grave com a compilação do código fonte, as restantes componentes do sistema continuariam operacionais. A nova arquitetura geral do sistema pode ser observada na Figura 43.

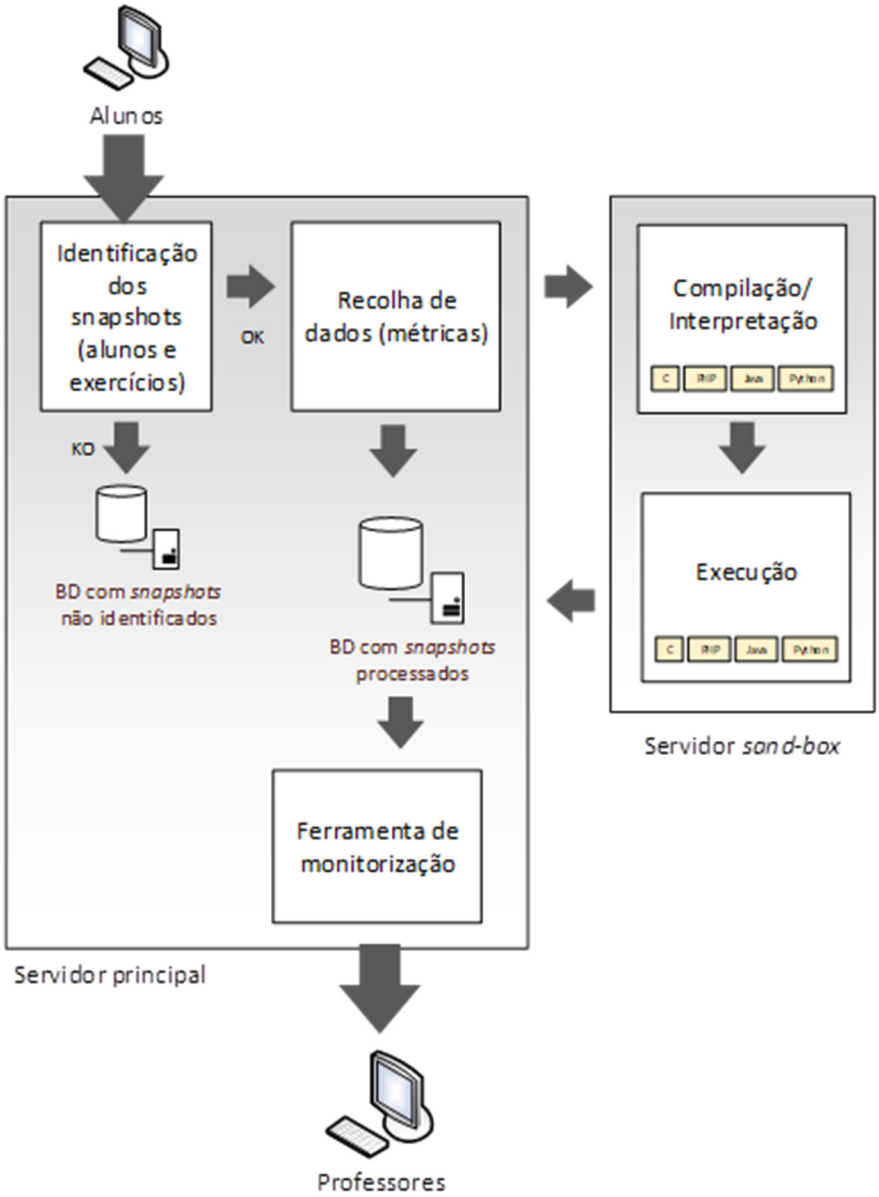

Figura 43 - Representação da arquitetura do sistema - iteração dois.

4.3.4.4. Ferramenta de monitorização

Durante esta iteração procedeu-se igualmente a uma extensa remodelação do aspeto visual e interação com o utilizador da ferramenta de monitorização utilizada pelos professores, de modo a torná-lo mais funcional e simples de utilizar.

Com a introdução de um novo *layout*, foram igualmente introduzidas novas visualizações dos dados recolhidos de modo a conseguir reunir e apresentar mais informação sobre o desempenho dos estudantes.

Como é visível na Figura 44, algumas destas visualizações são apresentadas com um tamanho relativamente pequeno, mas é disponibilizado um botão "view larger" para o caso de o professor pretender analisar a informação com mais detalhe. Entre as novas visualizações introduzidas encontra-se a distribuição dos *snapshots* recebidos ao longo do tempo, organizados em função da ficha prática a que dizem respeito (e.g. WS1, WS2, etc.) tal como pode ser observado na Figura 45 com mais detalhe.

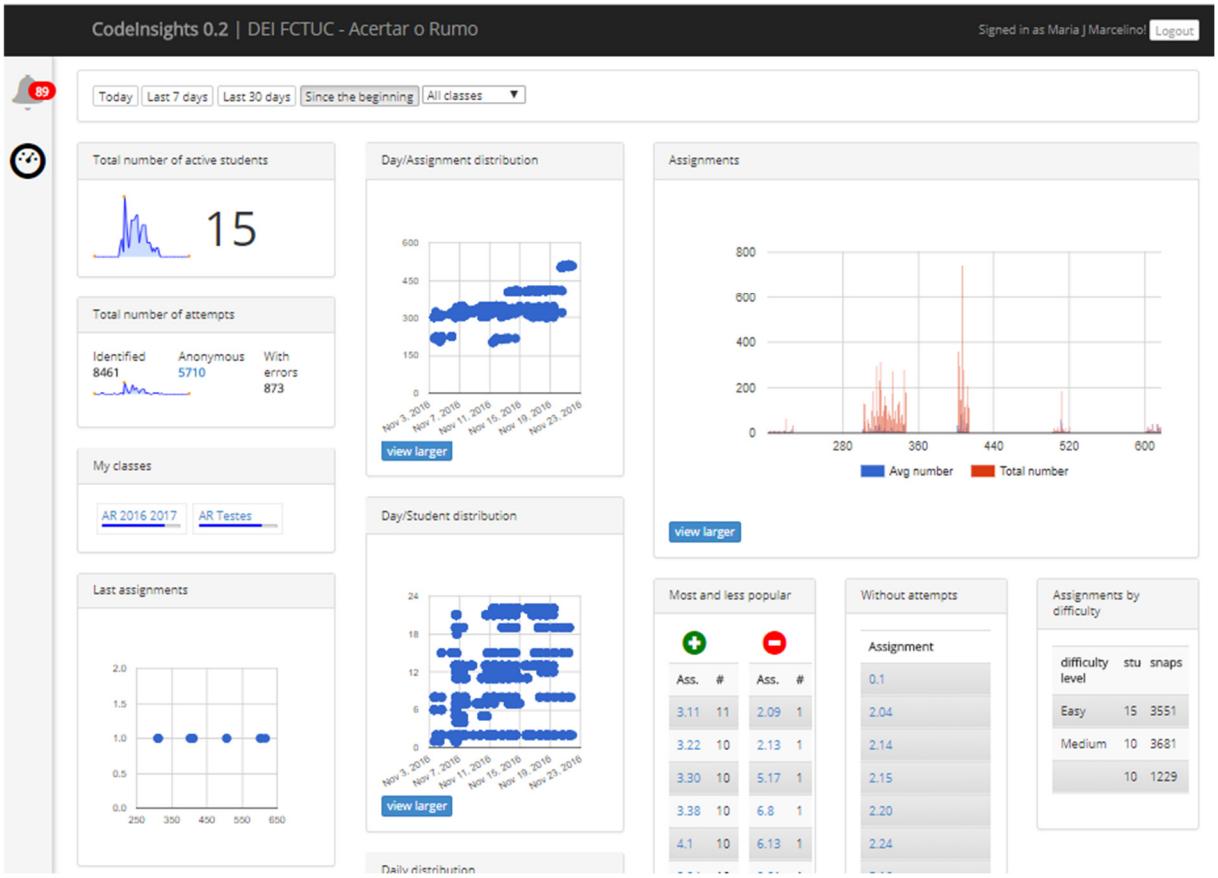

Figura 44 - Vista parcial do *dashboard* da ferramenta de monitorização - iteração dois.

Em termos objetivos, esta visualização permite aos professores ter uma visão clara sobre o ritmo de trabalho dos seus estudantes. A título de exemplo, é visível que na ficha prática nº2 (WS 2) existiram estudantes que começaram a trabalhar significativamente mais tarde do que os colegas. É ainda visível que, apesar de existirem já estudantes a trabalhar na ficha prática nº 4, muitos estudantes continuam a trabalhar na ficha prática nº3. Neste cenário era clara a existência de ritmos de trabalho consideravelmente diferentes, o que poderia levar a uma intervenção do professor.

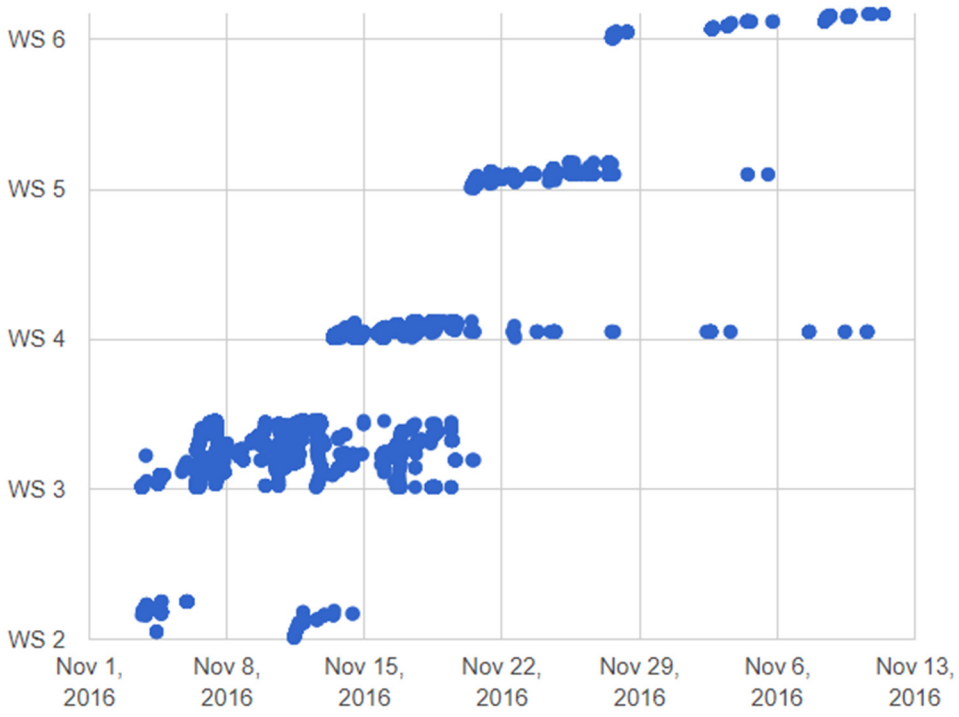

Figura 45 - Exemplo de visualização da distribuição dos *snapshots* ao longo do tempo - iteração dois.

Uma outra visualização introduzida nesta iteração pode ser observada na Figura 46, mostrando a distribuição dos *snapshots* recebidos em função do dia da semana e hora do dia.

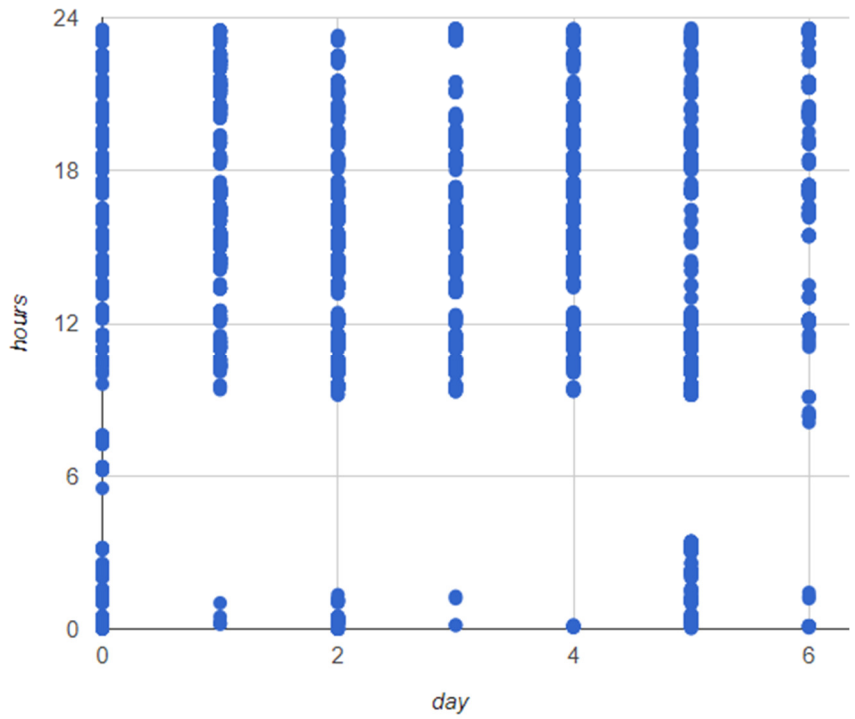

Figura 46 - Exemplo de visualização da distribuição dos *snapshots* em função do dia e hora - iteração dois.

Nesta visualização "0" corresponde a domingo, "1" a segunda-feira e assim sucessivamente. No caso específico desta figura é possível observar que foram sendo recebidos *snapshots* de uma forma consistente em praticamente todos os dias da semana, incluindo fins-de-semana. É ainda possível observar que muitos *snapshots* foram recebidos fora do período normal de aulas (incluindo madrugadas) o que pode ser uma indicação de que alguns estudantes podem estar a ter dificuldades que os obriga a realizar muito trabalho extra fora das aulas para conseguirem acompanhar o ritmo dos colegas. Poderá igualmente ser revelador da existência de estudantes com mais capacidades e/ou conhecimentos que pretendem avançar mais rapidamente ou poderá simplesmente significar que estudantes preferem trabalhar durante a noite. Em qualquer dos casos, os professores podem utilizar esta informação para tomar as medidas que considerem necessárias e pertinentes.

Nesta iteração foi introduzida uma visualização contendo informação sobre a distribuição dos *snapshots* ao longo do tempo para cada estudante. Na Figura 47 é apresentado um exemplo desta distribuição em que cada círculo representa um *snapshot* e cada "linha" um estudante.

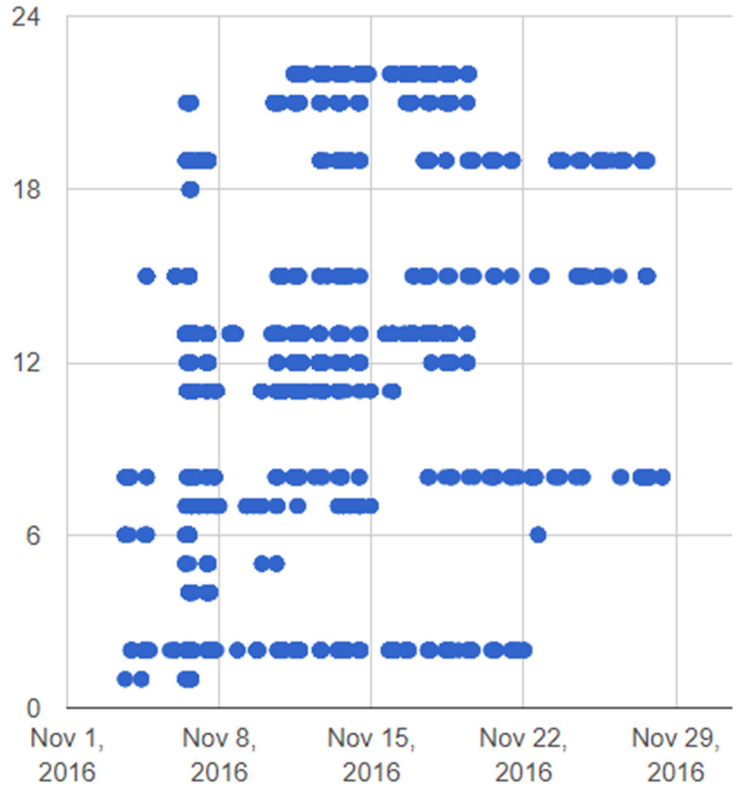

Figura 47 - Exemplo de visualização da distribuição dos *snapshots* de cada estudante ao longo do tempo - iteração dois.

No caso concreto desta visualização, é possível obter informação sobre a regularidade do trabalho de cada estudante. É também possível identificar situações de estudantes que tenham deixado de submeter *snapshots*, podendo o professor abordá-los no sentido de averiguar o que os pode ter levado ao abandono do sistema. A título de exemplo, os estudantes poderão ter desinstalado o *plug-in* devido a questões técnicas ou, no limite, poderão simplesmente ter desistido da unidade curricular. Seja como for, o professor tem acesso a esta informação em tempo real, o que lhe permitirá agir em tempo útil.

Por forma a permitir que os professores conseguissem ter uma visão global do desempenho dos estudantes, foi desenvolvida uma visualização sob a forma de matriz em que era apresentada de forma sintetizada toda a experiência desde o início. Na Figura 48, é apresentada uma vista parcial desta matriz.

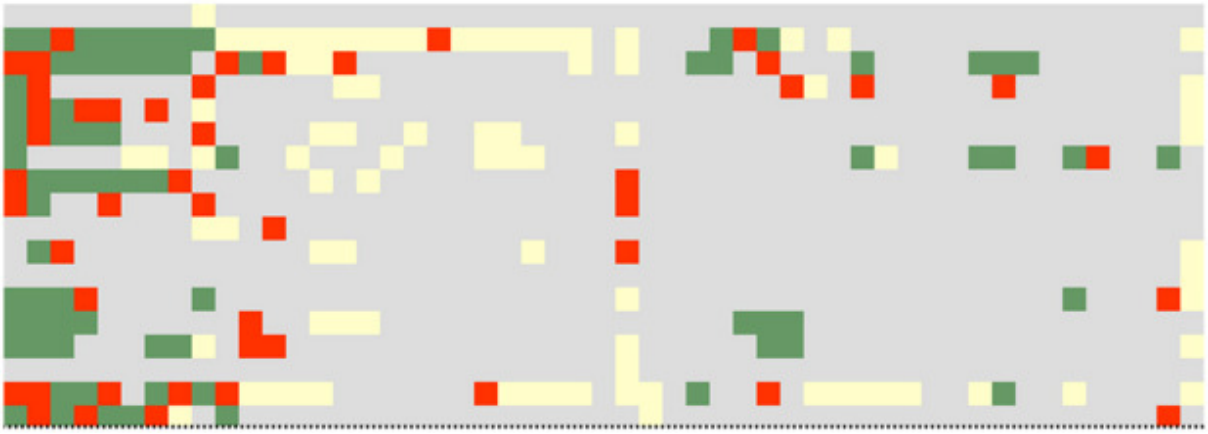

Figura 48 - Vista parcial da matriz de desempenho - iteração dois.

Cada linha da matriz representa um estudante e cada coluna representa um exercício, sendo que cada célula representa a última tentativa feita pelo estudante para um dado exercício. A cor cinzenta indica que não foi feita nenhuma tentativa de resolução, a cor amarela indica que a última tentativa apresentava erros de *run time*, a cor vermelha indica a existência de erros de compilação e, finalmente, a cor verde indica que não existiu qualquer erro, tendo o código sido interpretado e o *output* produzido recolhido com sucesso.

Ao clicar numa determinada célula, o professor tinha acesso a informação detalhada sobre todas as tentativas efetuadas pelo estudante para aquele exercício. Também ao nível desta página foram feitas algumas modificações, podendo o resultado ser observado na Figura 49.

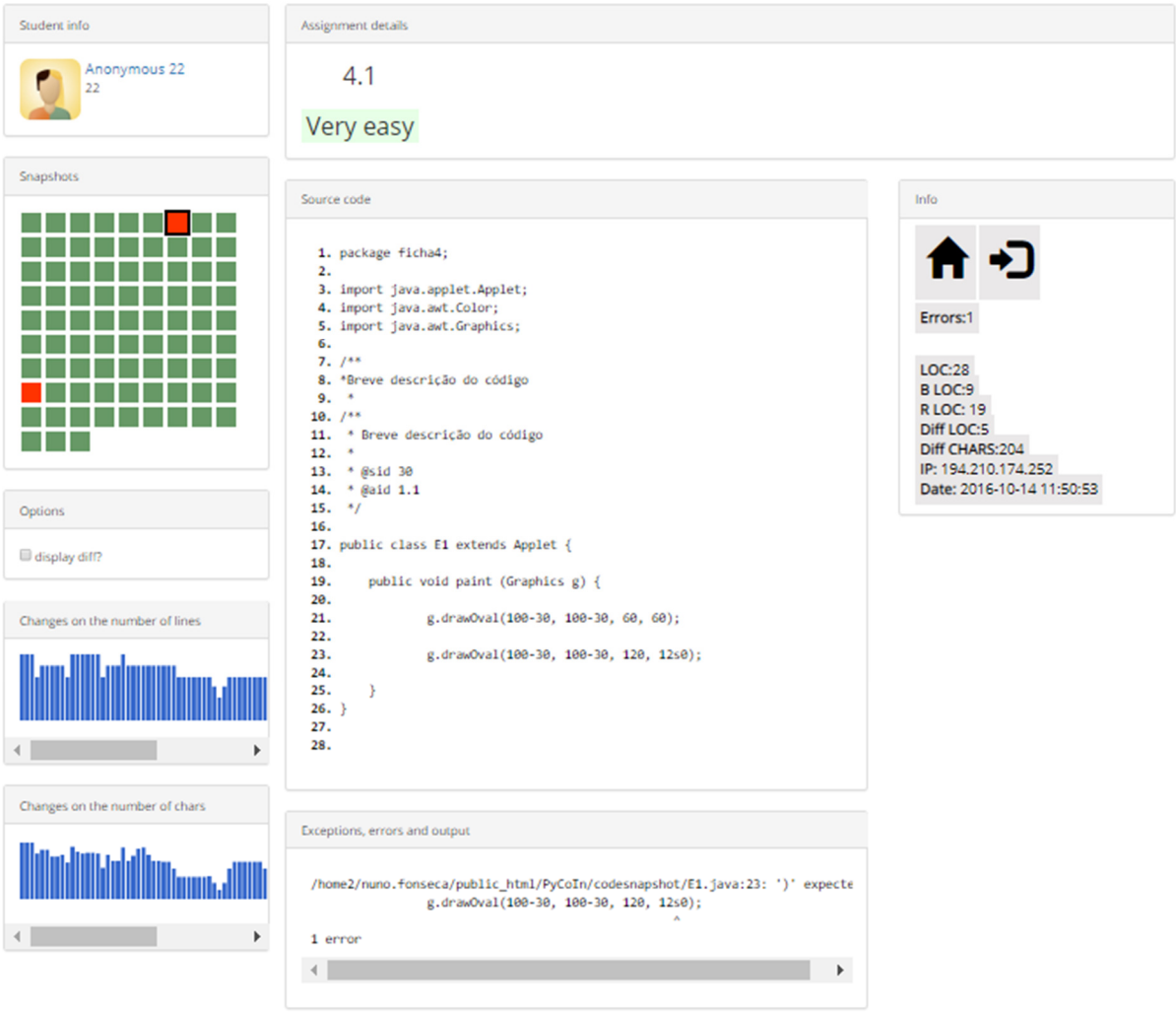

Figura 49 - Página com detalhes das tentativas de resolução de um exercício - iteração dois.

Ao longo desta iteração, foi levado a cabo um intensivo esforço de investigação dos melhores *designs* funcionais na apresentação da informação aos professores, tendo muitos deles indo sendo descartados, outros abandonados e recuperados e alguns outros colocados em *stand-by*. Muitas visualizações já existentes na versão anterior mantiveram-se nesta iteração, ainda que tenham sofrido ligeiras alterações ao nível do aspeto visual.

Convirá realçar que por vezes uma determinada visualização acaba por funcionar como ponto de partida, podendo o professor cruzar a informação com outras visualizações por forma a melhor fundamentar as suas decisões.

4.3.4.5. Notificações automáticas

Apesar de ter existido o cuidado de tornar o sistema de utilização simples, ainda que disponibilizando muita informação para os professores poderem consultar, a verdade é que viria a constatar-se que os professores, dadas as restrições inerentes à sua atividade, não teriam disponibilidade para estar frequentemente a utilizar a ferramenta de monitorização. Mesmo que os professores acedessem com regularidade ao sistema, dada a quantidade de informação que esta disponibiliza, poderiam sempre existir situações relevantes a passar despercebidas, especialmente considerando a existência de estudantes a trabalhar fora das aulas.

Em diversas reuniões com o professor que já havia utilizado o sistema no passado e com os que o iriam utilizar nesta iteração chegou-se à conclusão de que o sistema deveria ter algum mecanismo que permitisse alertar o professor para a existência de informações relevantes. Deste modo, mesmo que o professor não estivesse a aceder ao sistema, era enviada uma notificação via email sobre a existência de situações merecedoras da sua atenção.

Em termos concretos, foi decidido criar dois tipos distintos de notificações:

on-line - notificações geradas em tempo real perante a ocorrência de um determinado evento. Por exemplo, o professor pode querer ser notificado de imediato sempre que um determinado estudante submete *n snapshots* seguidos com erros de compilação ou sempre que um estudante submete mais do que *m snapshots* para o mesmo exercício.

off-line **-** notificações geradas num determinado momento previamente definido (*e.g.* uma vez por hora, uma vez por dia, uma vez por semana) mediante a análise de um conjunto de dados. Por exemplo, o professor poderá querer ser notificado semanalmente sobre a existência de estudantes que na última semana não tenham feito qualquer submissão.

Assim, nesta iteração foi introduzida uma funcionalidade que permitiu gerar notificações para o caso concreto de existência de estudantes com um número excessivo de tentativas de resolução para um determinado exercício (*on-line*), o que poderá ser indicativo de que o estudante está com dificuldades na sua resolução. Para tal, foi definido um limiar a partir do qual era considerado que se tratava de um número excessivo de tentativas. Idealmente, deveriam ser definidos limiares diferentes para cada exercício de modo que, para um exercício simples, o limiar poderia ser mais baixo do que para um exercício com

um nível de dificuldade mais alto. Ainda que esta situação tivesse sido considerada, por restrições temporais, optou-se por utilizar apenas um limiar, adiando a utilização de um mecanismo mais avançado para futuras iterações.

Foi ainda implementada uma notificação a ser gerada no caso de um estudante ter submetido mais do que ***n*** tentativas consecutivas com erros de compilação. O valor de ***n*** poderia ser definido e alterado pelo professor a qualquer altura, mas era sempre igual para todos os estudantes e exercícios.

Quando uma notificação era gerada, era enviado um email para o(s) professor(s) e para além disso, essa notificação era igualmente apresentada na ferramenta de monitorização, tal como apresentado na Figura 50.

Figura 50 - Lista de notificações - iteração dois.

4.3.5. Avaliação do sistema

Como já foi referido anteriormente, o sistema na sua segunda iteração foi utilizado para suportar três unidades curriculares distintas, apresentam-se em seguida os principais resultados destas avaliações.

4.3.5.1. Fundamentos de Programação (LEI - ESTGOH)

O processo de avaliação do sistema ***CodeInsights*** nesta unidade curricular ocorreu entre os dias 19 de outubro e 6 de novembro de 2016. Estavam inscritos nesta unidade curricular 25 alunos. Como já mencionado, foi utilizada a linguagem de programação C e foi decidido não utilizar nenhum ambiente de desenvolvimento em particular, podendo os

estudantes utilizar o editor código que bem entendessem, devendo a compilação e execução ser feita por linha de comandos.

Devido a problemas técnicos não foi possível aos estudantes fazer as submissões enquanto estivessem dentro da escola e ligados à rede informática da mesma. Assim, apenas foram recebidos *snapshots* dos estudantes enquanto trabalhavam em casa, ou caso estivessem na escola, mas ligados a outra rede informática (rede 3G ou 4G pessoal). Perante esta limitação do sistema, diversos estudantes procederam à remoção do *plug-in* impossibilitando o acompanhamento do seu trabalho fora do âmbito das aulas.

Ainda assim, apesar destas dificuldades, foram recebidos aproximadamente 900 *snapshots* válidos (aproximadamente 1200 no total) de oito estudantes. Uma vez mais, os principais motivos pelos quais existiam *snapshots* inválidos prendiam-se com problemas de identificação, ainda que neste caso concreto, na maioria dos casos, não eram inseridos nenhuns elementos identificativos.

Um facto notável que foi possível observar está relacionado com o perfil dos estudantes que não desinstalaram o *plug-in* e continuaram a utilizá-lo até ao final do semestre. Entre estes encontravam-se os estudantes que acabariam por obter melhores classificações finais. Muito embora um dos principais objetivos do sistema passe por fornecer informação aos professores sobre os estudantes que possam estar a ter dificuldades em progredir, é igualmente importante obter informação sobre os estudantes mais avançados, de modo a encontrar formas de lhes dar um tipo de apoio diferenciado relativamente aos colegas com mais dificuldades.

No geral, apesar das limitações encontradas, o sistema permitiu ainda assim ao professor acompanhar o trabalho realizado por alguns estudantes que de outra forma não poderia ter feito.

4.3.5.2. Programação (LEI - ESTGOH)

A avaliação do sistema *n*o contexto desta unidade curricular decorreu entre os dias 28 de setembro e 19 de outubro de 2016. No total estavam inscritos a esta unidade curricular 28 estudantes. No entanto, tal como o professor responsável referiu à partida, seria de esperar que um número considerável de estudantes nunca viesse a assistir às aulas e nem tentar resolver qualquer exercício.

É ainda importante mencionar o facto de esta unidade curricular já pertencer ao segundo ano da licenciatura, não se tratando, portanto, de uma unidade curricular

verdadeiramente introdutória. Com efeito, trata-se de uma unidade curricular focada nos aspetos ligados à programação orientada a objetos utilizando a linguagem Java e na qual são disponibilizados aos estudantes um conjunto de exercícios mais complexos do que os exercícios tradicionalmente disponibilizados em unidades curriculares introdutórias.

Ainda que o sistema ***CodeInsights*** não tivesse sido pensado para o contexto de unidades curriculares mais avançadas, aproveitou-se ainda assim para ver se a informação que era possível recolher e apresentar ao professor era relevante para a atividade deste.

Infelizmente, e uma vez que esta experiência estava a ocorrer nas mesmas instalações físicas da experiência anterior, também aqui existiram dificuldades de ordem técnica numa fase inicial que levaram vários estudantes a desinstalar o *plug-in*.

Como se viria a verificar mais tarde, entre os estudantes que não instalaram o *plug-in,* ou que acabariam por o desinstalar precocemente, encontram-se maioritariamente **estudantes repetentes** (alguns dos quais múltiplas vezes) e estudantes que no final da unidade curricular acabariam por não conseguir obter aprovação.

Em sentido oposto, foi possível constatar que os estudantes que mais *snapshots* submeteram foram os que conseguiram obter melhores classificações finais. Não deixa de ser intrigante que os estudantes que mais poderiam beneficiar da utilização do sistema foram precisamente aqueles que acabaram por evitar a sua utilização.

Apesar da curta duração da experiência, de não ser o contexto ideal para a sua utilização e das dificuldades encontradas, acabariam por ser recebidos cerca de 2000 *snapshots*, dos quais cerca de 1600 foram considerados válidos. Apesar da insistência do professor em que fosse utilizado o *template*, alguns estudantes ignoraram essa indicação e fizeram as suas submissões sem quaisquer elementos que permitissem a sua identificação.

Durante a experiência, foram propostos aos estudantes um total de 16 exercícios organizados em três fichas práticas distintas, cada uma delas associada a um tópico em particular. O professor resolvia em conjunto com os estudantes o primeiro exercício de cada ficha durante a aula, devendo os outros exercícios ser resolvidos pelos estudantes durante o tempo restante das aulas e fora delas.

No total foram recebidos *snapshots* de 14 estudantes distintos devidamente identificados, sendo que sem grandes surpresas os exercícios com um número mais elevado de estudantes a tentar resolver acabariam por ser o primeiro exercício da primeira ficha

prática (1.1), com 11 estudantes, e o primeiro exercício da segunda ficha prática (2.1), também com 11 estudantes, ou seja, os exercícios resolvidos na aula com o professor.

Neste sentido, o sistema revelou-se importante já que permitiu constatar que, salvo raras exceções, entre os estudantes que continuaram a utilizar o sistema, poucos foram os que fizeram alguma submissão para os exercícios não resolvidos na aula com a ajuda do professor. Como eventualmente seria de esperar, os que o fizeram encontram-se no grupo de estudantes que obtiveram as classificações finais mais elevadas. Um outro aspeto intimamente ligado com o anterior, diz respeito à verificação de que a grande maioria das submissões foi feita durante o período das aulas. Esta informação poderá ser utilizada pelo professor para que pondere se será pertinente alterar a estratégia ao nível do trabalho a ser realizado fora do contexto de aula, uma vez que, tal como pode ser observado, a grande maioria dos estudantes não trabalhou fora desse âmbito.

O sistema de notificações foi utilizado apenas na vertente do número excessivo de tentativas de resolução, tendo sido gerado um total de 12 notificações, às quais o professor deu a atenção que considerou oportuna.

Ainda que a experiência não tivesse corrido completamente como esperado, no final desta teve lugar uma reunião com o professor na qual ele referiu que, apesar de tudo o que havia corrido menos bem, ao utilizar o sistema foi-lhe possível obter algumas informações que de outra forma dificilmente teria acesso e que, como tal, gostaria de o poder utilizar novamente em edições futuras da unidade curricular.

4.3.5.3. Introdução à Programação em Java (Acertar o Rumo - DEI - FCTUC)

A versão do sistema desenvolvido nesta iteração viria ainda a ser testada com estudantes da unidade curricular de Introdução à Programação em Java do programa *Acertar o Rumo*. Este programa consiste de um curso intensivo de programação em Java e destina-se essencialmente a indivíduos que já tenham formação superior em áreas diversas da informática e que pretendam deste modo enveredar por uma carreira diferente. Por se tratar de um curso intensivo, os estudantes têm uma carga letiva semanal de 24 horas em sala de aula, nas quais são supervisionados por vários professores.

Por se tratar de uma unidade curricular introdutória, esta constituía o contexto ideal para testar o sistema. Assim, a experiência com esta turma decorreu entre os dias 3 e 28 de outubro de 2016. Durante a experiência foram recebidos aproximadamente 10400 *snapshots*, sendo que destes apenas aproximadamente 8000 foram considerados válidos.

Muito embora estivessem inscritos 22 estudantes, apenas foram recebidos *snapshots* de 15 estudantes, principalmente devido a limitações dos computadores de que dispunham. Alguns estudantes instalaram o *plug-in* e utilizaram-no durante algum tempo, tendo acabado posteriormente por o desinstalar uma vez que lhes estava a causar alguma lentidão no sistema. Infelizmente, estes problemas com a utilização do *Eclipse* acabariam por nos ser comunicados apenas na reta final da experiência.

Este problema da lentidão ocasional do sistema acabou posteriormente por ser investigado, tendo sido identificados como causa principal aspetos ligados à rede informática, mas também aspetos ligados à própria conceção do *plug-in* e do sistema de processamento dos *snapshots*. Em traços gerais, o problema relacionava-se com o envio dos *snapshots* para o servidor pelo *plug-in* ocorrer de forma síncrona, isto é, o *snapshot* era enviado para o servidor e o *plug-in* ficava "bloqueado" à espera de uma resposta do servidor relativa ao sucesso ou insucesso do processamento do *snapshot* de modo a apresentar uma mensagem informativa aos estudantes. Caso existissem flutuações na velocidade da ligação ou caso o processamento dos *snapshots* levasse muito tempo, o *Eclipse* ficaria "bloqueado" alguns segundos. Este era um aspeto que teria obrigatoriamente de ser alvo de revisão em futuras iterações.

É importante mencionar que, por indicação da professora responsável por esta unidade curricular, o sistema foi utilizado com identificadores artificiais que não permitiam identificar de facto os estudantes responsáveis pelas submissões. Assim, a cada identificador artificial, em vez do nome real do estudante, foram associados nomes artificiais tais como "Anonymous 1", "Anonymous 2" e assim sucessivamente. Muito embora este facto implicasse que fosse praticamente impossível fazer intervenções individualizadas junto dos estudantes, continuaria ainda assim a ser possível utilizar o sistema para suportar decisões de natureza mais alargada.

Dada a natureza intensiva deste curso, durante o período da experiência, foram propostos aos estudantes 105 exercícios organizados em cinco fichas práticas, cada uma dedicada a um tópico específico. Convirá a este respeito referir que, em edições anteriores deste curso, não era feita qualquer submissão das soluções realizadas pelos estudantes durante as aulas. A única informação que os professores tinham sobre o desempenho dos estudantes era aquela que iam conseguindo obter mediante a observação direta do trabalho destes. Tendo em conta as diferentes personalidades dos

estudantes, não será difícil admitir que muitos estudantes pudessem ter dúvidas, mas que se sentissem retraídos para as colocar, principalmente em frente aos colegas.

Tendo em consideração o volume de exercícios, alguns com um nível de complexidade relativamente elevada, verificou-se um fenómeno curioso e que consistia no facto de, com demasiada frequência, os estudantes não resolverem os exercícios de forma seguida.

Efetivamente, muitas vezes os estudantes estavam a resolver um exercício e, subitamente, começavam a resolver outro, para posteriormente voltarem à resolução do que estavam inicialmente a resolver. De modo a visualizar este fenómeno, a fim de o compreender na sua complexidade, foi desenvolvida uma ferramenta que permite ao professor verificar para cada estudante se os exercícios estão a ser resolvidos de forma linear, ou se estão a existir muitos avanços e recuos, o que poderá eventualmente ser indicador de que o estudante poderá estar a ter dificuldades, ou revelando outras causas inicialmente não previstas, como tempo reduzido para a resolução de exercícios, matéria lecionada demasiado rápido ou até questões de foro de *deficit* de atenção. Em todo o caso, caberá sempre ao professor confirmar a existência ou não de dificuldades na aquisição de conhecimentos ou outras causas inicialmente não previstas.

Na Figura 51 pode ser observada a referida visualização que assume a forma de um grafo dirigido, sendo que sempre que existem dois arcos entre dois nós, significa que esse estudante voltou atrás. O professor tem a possibilidade de interagir com este gráfico, arrastando nós ou aumentando e diminuindo o tamanho do grafo.

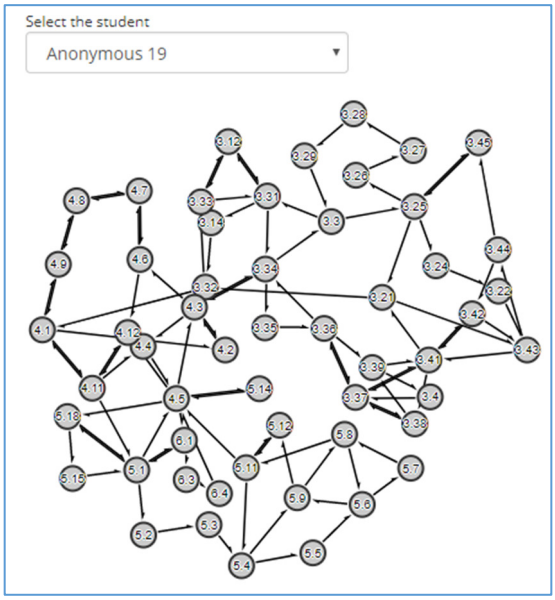

Figura 51 - Exemplo de visualização do grafo representativo da sequência pela qual foram resolvidos os exercícios - iteração dois.

Ainda que em teoria não fosse possível fazer intervenções individualizadas, as várias visualizações e informação agregada disponibilizada pelo sistema foram permitindo à professor responsável fazer uma monitorização detalhada do desempenho dos estudantes.

Também nesta experiência foi utilizado o sistema de notificações automáticas para a existência de estudantes com um número invulgar de tentativas de resolução para um determinado exercício. No total foram geradas mais de 80 notificações relativas a quantidades invulgares de tentativas de resolução. Em alguns casos os estudantes estavam de facto com algumas dificuldades, no entanto em muitos outros casos, principalmente no que diz respeito a um estudante concreto, tratava-se mais de uma questão de comportamento de programação. Em vez de planear devidamente e escrever grande parte do código antes de o testar, este estudante utilizava uma abordagem completamente distinta e que em relação a diversos exercícios passava por compilar o código de cada vez que escrevia qualquer coisa. Por estes motivos, diversos estudantes tiveram exercícios com um número de tentativas bem superior a 150 (189 tentativas num caso concreto).

Muito embora o número de tentativas por si só não constitua propriamente um problema, o professor era notificado, podendo posteriormente obter informações mais detalhadas e verificar se se tratava apenas de uma questão de "estilo" ou se existiam de facto problemas de fundo que justificassem a tomada de medidas adequadas.

Apesar de se tratar de um curso intensivo e no qual os estudantes passavam muitas horas a programar, com base na visualização apresentada anteriormente na Figura 46, que dizia respeito a esta experiência, foi possível comprovar que muitos estudantes continuavam a trabalhar assiduamente em casa, mesmo durante a madrugada. Com efeito, no total 36,4% dos *snapshots* foram submetidos fora do período de aulas. Curiosamente, na maior parte das situações, não eram estudantes com dificuldades, mas sim estudantes que pretendiam simplesmente avançar mais rapidamente. O facto de existirem estudantes com ritmos de trabalho consideravelmente distintos acabava por constituir um problema, pois por vezes os professores preparavam as aulas contando que os estudantes iriam resolver na aula um determinado conjunto de exercícios, no entanto constatavam que muitos destes estudantes já tinham resolvidos esses exercícios e já iam bem mais avançados, ao passo que alguns outros estavam ainda muito atrasados. Esta

situação acaba por obrigar a um esforço extra por parte dos professores, estando bem evidente na Figura 45.

Após terminada a experiência, teve lugar uma reunião com a professor responsável na qual foram debatidos os principais aspetos positivos e negativos da utilização do sistema nesta unidade curricular. Como seria de esperar, um dos aspetos mais relevantes estava relacionado com o anonimato dos estudantes. Ainda que o sistema permitisse obter informação útil para os professores, a não identificação dos estudantes dificultava e limitava o raio de ação dos mesmos. A este respeito a professor acabou por referir que, a partir de dada altura da experiência, já havia identificado a maioria dos estudantes "Anonymous", o que acabou por lhe permitir de facto fazer algumas intervenções diretas com base em informação obtida do sistema.

Finalmente, foi ainda debatido o aspeto já anteriormente apresentado relativo à ocasional lentidão do sistema.

4.3.6. Sumário da iteração dois

Apesar das dificuldades encontradas em cada uma das experiências terem sido muitas e diversas, a sua realização foi importante pois foram identificadas uma série de limitações do sistema que contribuíram para melhorar a experiência de utilização deste e permitir fornecer informação mais útil aos professores o que constitui o principal objetivo do trabalho.

No geral, o *feedback* dos professores foi bastante positivo, uma vez que a utilização do sistema lhes permitiu ter uma visão do desempenho dos estudantes que nunca tinham tido anteriormente. Surpreendentemente acabou por se criar um certo sentimento de "querer saber mais" o que constitui um agradável e estimulante alento extra no trabalho de continuar a melhorar e fazer evoluir o sistema.

Principalmente na experiência do programa *Acertar o Rumo*, ficou evidente que a existência de estudantes a trabalhar com ritmos consideravelmente distintos constitui um problema considerável para os professores. Muito embora já seja disponibilizada informação que permita de algum modo identificar estas situações, conclui-se que deveriam ser estudadas outras formas de conseguir transmitir essa informação aos professores de forma mais simples e concreta.

Relativamente à utilização de casos de teste, nesta iteração foram equacionadas algumas soluções, que, no entanto, por não terem sido devidamente aprofundadas, acabariam por

não ser implementadas. Assim sendo, nesta iteração continuou a não ser possível determinar automaticamente se os programas desenvolvidos pelos estudantes estavam a produzir o resultado esperado ou não. Para este efeito, os professores teriam de verificar eles próprios se o *output* produzido estava ou não correto, tendo em conta o *input* utilizado.

Importará ainda referir que, apesar de todas as limitações, todos os professores reiteraram que a informação recolhida e disponibilizada pelo sistema constituía uma representação fiel (e até aumentada) da sua própria perspetiva dos estudantes e que tinham todo o interesse em voltar a utilizar o sistema em edições futuras das suas unidades curriculares.

4.4 Iteração três

Após analisar cuidadosamente os resultados das experiências anteriores, foi chegado o momento de começar a trabalhar na terceira iteração do sistema.

4.4.1. Objetivos

Ainda que não tivessem corrido exatamente como o desejado, das experiências da iteração anterior surgiram uma série de questões pertinentes e cujas resolução constituiu em grande medida o objetivo desta terceira iteração.

Um aspeto muito importante que havia sido identificado anteriormente, e que tinha que ser aprofundado, estava relacionado com a grande diversidade de ritmos de trabalho dos estudantes existentes numa turma. Assim, entre os objetivos desta iteração foi incluída a conceção, desenvolvimento e teste de mecanismos que permitissem a deteção e visualização de diferenças consideráveis no ritmo de trabalho dos estudantes, por forma a conseguir, entre outras coisas, diferenciar a assistência que lhes é dada.

4.4.2. Descrição do contexto

Nesta iteração, o sistema foi testado em duas experiências distintas.

A primeira experiência realizou-se com estudantes da unidade curricular de Programação da Licenciatura em Engenharia da Gestão Industrial da Universidade de Coimbra, na qual o sistema já havia sido utilizado durante a iteração um e na qual é

utilizada a linguagem *Python* (no âmbito da reformulação do curso, a unidade curricular mudou de nome).

A segunda experiência ocorreu com uma turma da unidade curricular de Programação para a Internet II da Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital, em que foi utilizada a linguagem PHP.

4.4.3. Trabalho desenvolvido

4.4.3.1. Desenvolvimento dos *plug-ins*

Tal como já tinha acontecido, no caso da unidade curricular em que era utilizado *Python,* foi utilizado o ambiente de desenvolvimento *PyCharm*. No entanto, tal como o professor teve a oportunidade de enfatizar, era importante resolver a situação de serem disponibilizadas opções de menu e botões específicos para a utilização do ***CodeInsights*** paralelamente com as opções e menus padrão do *PyCharm*. Assim, após algum tempo de pesquisa, foi finalmente encontrada a forma para conseguir que o *plug-in* fosse executado automaticamente sempre que fossem executadas as instruções padrão do *PyCharm*, não havendo assim necessidade de disponibilizar opções de menu e botões adicionais.

Tendo em consideração que tinham existido problemas com o uso dos elementos identificativos dos *snapshots,* houve também a necessidade de fazer algo a este respeito. Na versão anterior do sistema já era apresentada uma mensagem ao utilizador no caso de existir algum problema com a identificação dos *snapshots*, mas esta mensagem era tão discreta que possivelmente muitos estudantes nem sequer se apercebiam dela. Assim, nesta nova versão, passaram a ser utilizadas mensagens com cores mais visíveis no sentido de chamar a atenção dos estudantes. Na Figura 52 é possível observar alguns exemplos das mensagens que eram apresentadas aos estudantes.

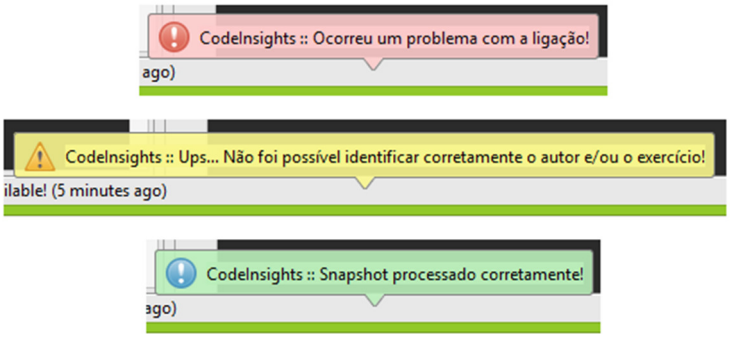

Figura 52 - Mensagens informativas apresentadas no *PyCharm* - iteração três.

Chegou a ser implementada e testada informalmente a utilização de caixas de alerta para este efeito, no entanto, o professor responsável pela unidade curricular manifestou que tais caixas de alerta eram demasiado intrusivas e, como tal, acabariam por não ser utilizadas.

Em Programação para a Internet II foi utilizado como ambiente de desenvolvimento o *Eclipse PDT*, cujo *plug-in* já estava desenvolvido e havia sido utilizado em duas experiências na iteração anterior.

Devido ao sucedido anteriormente com a experiência com o programa *Acertar o Rumo*, em que ocasionalmente poderiam surgir situações em que o *Eclipse* bloqueava, procedeu-se a uma alteração no modo de funcionamento do *plug-in* e da ferramenta de receção e processamento dos *snapshots*. Nesta nova versão, o código era enviado para o servidor, procedendo-se à identificação do estudante e do exercício, sendo enviada imediatamente a resposta para o *plug-in,* continuando o processamento do *snapshot* no servidor, fazendo com que a possibilidade de bloqueio no ambiente de desenvolvimento fosse minimizada e/ou eliminada completamente. Deste modo, mesmo que o código tivesse um ciclo infinito ou outro qualquer elemento que causasse demora no seu processamento no servidor, não haveria impacto no ambiente de desenvolvimento utilizado pelos estudantes. Assim, era possível eliminar a eventual lentidão provocada pelo tempo de processamento dos *snapshots,* no entanto, poderia continuar a subsistir alguma lentidão residual provocada pela própria velocidade da ligação à Internet.

4.4.3.2. Receção e processamento do código fonte

Para além das alterações já abordadas na secção anterior referentes ao modo como se desenrolava a articulação entre *plug-in* e ferramenta de receção e processamento do código fonte, um aspeto identificado na iteração anterior e que viria a merecer alguma atenção nesta estava relacionado com a possibilidade de testar o código dos estudantes no servidor. Tal poderia levar a situações indesejadas. Um exemplo concreto era a execução código em que existam ciclos infinitos (situação comum no contexto de unidades curriculares de introdução à programação). Se os estudantes executarem código contendo um ciclo infinito localmente, este será identificado e o estudante encerrará a consola/terminal em que o código estiver a ser executado. O problema é que o mesmo código será também executado no servidor (*sand-box*), o que deveria ser evitado. Para este efeito, foi criado um *script* que periodicamente iria procurar no sistema processos associados à *sand-box* que estivessem em execução há mais de cinco segundos,

eliminando-os, caso existissem. Embora esta solução possa não ser a mais elegante, tendo em consideração o foco geral do trabalho, acabou por ser a utilizada.

4.4.3.3. Ritmos de trabalho/aprendizagem

O sistema ***CodeInsights*** já disponibiliza aos professores diversas informações que lhes permitem de algum modo constatar a existência de diferenças consideráveis ao nível dos ritmos de trabalho dos estudantes, contudo, os professores necessitavam para tal de analisar "manualmente" a informação fornecida. Desta feita, procedeu-se ao desenvolvimento de um mecanismo automático de identificação de ritmos de trabalho distintos baseado na performance dos estudantes.

4.4.3.3.1. Performance dos estudantes

De modo a conseguir comparar a performance dos estudantes, é necessário definir o modo como esta será quantificada. Em traços gerais, neste contexto, define-se **performance** como uma medida das capacidades que um estudante tem para resolver corretamente as tarefas que lhe são propostas.

Adhadi et. al. [120] utilizam o conceito de *average performance*, que em termos práticos consiste em calcular o valor médio de correção de todos os *snapshots* registados por um estudante para um determinado exercício numa escala de 0 a 100. Neste caso, eram realizados diversos testes unitários, cada um deles contribuindo com uma parte para a definição deste *score*.

Uma vez que não tinha ainda sido desenvolvido o mecanismo de execução automática do código com casos de teste, começou por se equacionar a definição da performance de um estudante com base nos exercícios que este começou a resolver. Ainda que possivelmente na maioria dos casos um estudante que comece a resolver um exercício acabe por conseguir terminar com sucesso, não existe forma de garantir que tal se verifique.

Assim, para as experiências desta iteração foi utilizada uma solução intermédia que passava por indicar nos enunciados quais os valores de *input* que deveriam ser utilizados para efeitos de teste, assim como um exemplo do *output* que os programas deveriam devolver. Infelizmente no caso de alguns exercícios os *inputs* deveriam ser recolhidos do utilizador e não definidos no código e, mesmo para o caso em que eram definidos no código, não era garantido que os estudantes os fossem de facto utilizar. Por este motivo, de modo a garantir um nível mínimo de qualidade das "correções" dos exercícios, todos os dias por volta das 23h foi feita uma verificação manual das "correções" efetuadas e, no

caso de ser necessário, procedeu-se à modificação da avaliação. Convirá realçar que os professores nunca tiveram de o fazer, sendo este processo inteiramente da nossa responsabilidade. Obviamente não era a solução ideal, mas era a que, naquela altura, permitia o mínimo de intrusão no processo de construção do código pelos estudantes.

Com base na avaliação das últimas tentativas de resolução desenvolvidas pelos estudantes, a performance de um estudante num determinado período era definida pela expressão 1, ou seja, como a percentagem dos exercícios propostos naquele período que o estudante conseguiu resolver corretamente (*i.e.* que estão a produzir o *output* esperado).

$$\boldsymbol{P = \frac{nº\ de\ exercícios\ com\ output\ certo}{nº\ de\ exercícios\ propostos\ para\ resolução}} \tag{1}$$

A expressão 1, apresentava, contudo, uma limitação relacionada com o facto de existirem exercícios com graus de dificuldade consideravelmente diferentes. Para colmatar esta limitação, procedeu-se a uma pequena alteração à expressão 1, que passou pela inclusão de pesos, por forma a refletir os diferentes níveis de dificuldade dos exercícios.

$$\boldsymbol{P = \frac{\sum_{i=1}^{n} wi}{\sum_{j=1}^{k} wj}} \tag{2}$$

Na expressão 2, n corresponde ao número de exercícios que o estudante tentou resolver e cuja última tentativa produziu o *output* esperado e k corresponde ao número total de exercícios propostos num determinado período. Naturalmente, os valores de wi e wj correspondem aos pesos associados ao grau de dificuldade de cada exercício. A título de exemplo, para os pesos w, poderia ser considerado o valor 1 para exercícios fáceis, 1,5 para exercícios de dificuldade média e 2 para exercícios de dificuldade elevada e desta forma conseguiu-se obter valores de performance mais adequados.

Tendo por base os resultados individuais da performance de cada estudante, facilmente se conseguiria determinar também o valor global de performance de uma determinada turma com n estudantes, P_{turma}, mediante a utilização da expressão 3.

$$\boldsymbol{Pturma = \frac{\sum_{i=1}^{n} Paluno_i}{n}} \tag{3}$$

Deste modo, um professor que seja responsável por mais do que uma turma poderia comparar a performance da globalidade dos estudantes de cada turma.

Para definir o período para o qual se pretende determinar o valor de performance foi criado o conceito de ***Goals*** (objetivos). Assim, os professores definem um período temporal ao qual associam os exercícios que os estudantes devem conseguir resolver.

Tal como já sublinhado, será mais importante para os professores identificar os estudantes cuja performance seja muito inferior e muito superior à da maioria dos colegas (os *outliers*). Deste modo, após proceder ao cálculo do valor de performance de cada estudante, procede-se à determinação do valor médio de performance da turma para este intervalo (λ), assim com o valor para MAD (*Mean Average Deviation*), que é o valor médio das distâncias entre os vários valores de um conjunto e o valor médio destes valores (λ).

Com base em toda esta informação, era possível identificar os estudantes cujo valor de performance se encontra fora dos limites de MAD. Estudantes cujo valor global de performance fique abaixo do `valor médio + MAD` são estudantes que se encontram a trabalhar com um ritmo manifestamente baixo no período, o que poderá indicar a existência de problemas de aprendizagem. Por outro lado, estudantes cujo valor de performance esteja acima do `valor médio + MAD`, são estudantes com uma performance muito acima da média e, como já foi referido anteriormente, deverão igualmente merecer a atenção dos professores. Na Figura 53 pode ser observada uma representação gráfica da aplicação deste processo.

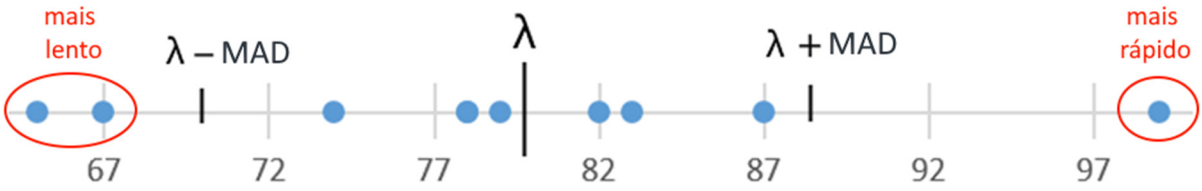

Figura 53 - Representação do processo de determinação dos estudantes que estão a trabalhar com um ritmo mais lento e mais rápido.

A informação sobre a performance de cada estudante, assim como de cada turma, é disponibilizada aos professores na forma de um gráfico que se apresenta na Figura 54. Aqui é possível observar a evolução da performance dos estudantes de uma turma ao longo do período de duração de uma experiência. Para cada dia (coluna) é apresentado um "resumo" da distribuição da performance dos estudantes. No topo de cada coluna é colocado o valor de performance do(s) estudante(s) com maior ritmo de trabalho e no fundo de cada coluna é colocado o valor oposto. A barra mais carregada está centrada no

valor médio da performance para aquele dia, sendo os limites inferiores e superiores definidos pelo valor da `média - MAD` e `média + MAD`, respetivamente.

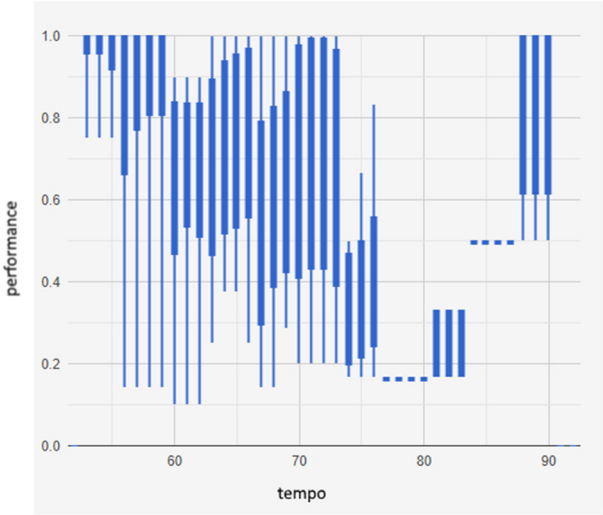

Figura 54 - Evolução da performance dos estudantes ao longo do tempo - iteração três.

Ainda que com base nesta visualização já fosse possível ter uma ideia de como se estavam a comportar os estudantes na globalidade, não era ainda possível saber de uma forma direta quais os estudantes que se encontravam em cada nível de performance. Para este efeito, foi desenvolvida uma outra visualização na qual era apresentada informação sobre o identificador dos estudantes associados a cada nível de performance. Um exemplo desta visualização pode ser observado na Figura 55.

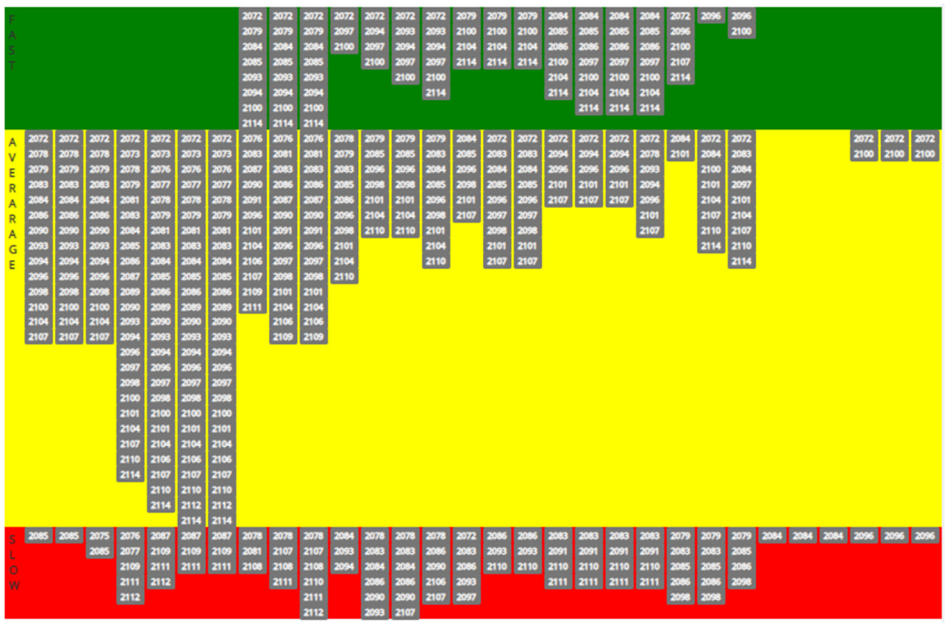

Figura 55 - Exemplo de visualização da distribuição dos estudantes em termos de ritmo de trabalho ao longo do tempo - iteração três.

Convirá referir que os valores de performance para um período só são calculados para os estudantes que tenham feito pelo menos uma submissão nesse período.

Por uma questão de gestão de espaço optou-se por utilizar o identificador do estudante, contudo, ao passar com o cursor do rato por cima de um determinado identificador, é apresentado o nome do estudante. Caso assim entenda, o professor poderá clicar no identificador de um determinado estudante, por forma a aceder a uma página contendo informação pormenorizada sobre esse estudante. Entre as informações disponibilizadas convirá destacar a comparação da performance desse estudante face aos colegas. Apresentam-se na Figura 56 dois exemplos desta visualização na qual é possível observar a evolução da performance de um estudante em concreto (linha azul) face aos restantes colegas de turma (barras vermelhas). Nestes casos é apresentado o paralelismo entre um estudante com uma performance sempre muito elevada (A) com a de um outro estudante com uma performance sempre muito baixa (B).

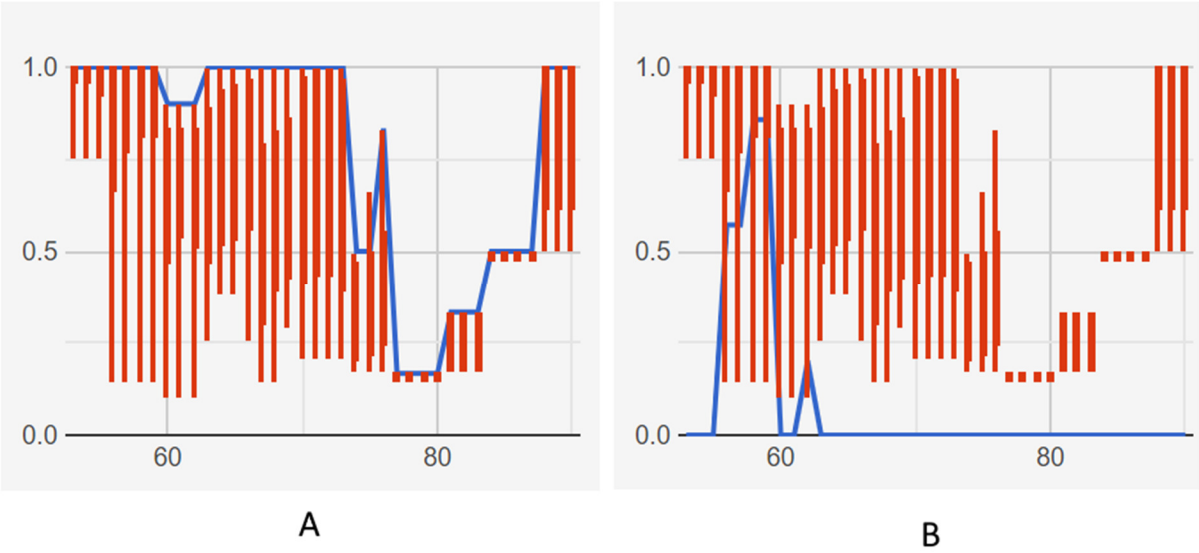

Figura 56 - Exemplos de visualização da comparação da performance de estudantes face aos colegas de turma

A situação representada na Figura 56 B poderá levar a pensar que a partir de dada altura este estudante deixou de trabalhar, uma vez que o seu valor de performance é sempre zero. Muito embora tal possa de facto ser verdade, neste caso concreto não corresponde à realidade, pois há que ter em consideração que o valor da performance num determinado dia é calculado tendo como referência os exercícios associados aos *Goals* para o período em que aquele dia se inclui. Assim, a situação representada nesta figura, corresponde a um estudante com um ritmo de trabalho muito mais lento, e que se encontrava a trabalhar em exercícios de fichas de trabalho anteriores. Esta situação

levou a que para além do conceito de **performance** fosse introduzido também o conceito de **progresso**.

4.4.3.3.2. Progresso dos estudantes

O conceito de progresso é muito semelhante ao conceito de performance previamente apresentado, à exceção do facto de em cada momento ser tido em consideração o trabalho desenvolvido pelo estudante desde o início, independentemente de ter conseguido ou não atingir os *Goals* definidos para o período previsto. Em termos formais define-se o progresso de um estudante *A* num determinado momento *t* (P_{At}) como o somatório dos pesos *wi* de todos os *k* exercícios que o estudante tentou resolver e para os quais a última tentativa estava a produzir o *output* desejado a dividir pelo somatório dos pesos de dificuldade *wj* de todos os exercícios proposto para resolução até ao momento *t* (4).

$$\boldsymbol{PAt} = \frac{\sum_{i=1}^{k} \boldsymbol{w_i}}{\sum_{j=1}^{n} \boldsymbol{w_j}} \tag{4}$$

Tal como no caso da performance, também será possível determinar o progresso médio de uma turma (*PT*) num determinado momento *t*, bastando para tal dividir o somatório dos valores de progresso de cada estudante (PA_{ti}), pelo número de estudantes da turma (5).

$$\boldsymbol{PTt} = \frac{\sum_{i=1}^{n} \boldsymbol{PAti}}{\boldsymbol{n}} \tag{5}$$

Com base nos valores do progresso de cada estudante é possível verificar quais os estudantes cujo progresso está consideravelmente mais acima e abaixo do que a maioria dos colegas. Para tal, procede-se de forma semelhante à que foi anteriormente apresentada em relação à performance, ou seja, a determinação do valor médio do progresso dos estudantes turma para cada dia (λ) assim com o valor para MAD.

Finalmente, é possível identificar os estudantes cujo progresso se encontra fora dos limites de MAD. Estudantes cujo valor de progresso fique abaixo do `valor médio + MAD` são estudantes que estão a progredir a um ritmo consideravelmente baixo, o que poderá indicar a existência de problemas de aprendizagem. Por outro lado, estudantes cujo valor de progresso esteja acima do `valor médio + MAD`, são estudantes a trabalhar com um ritmo muito superior e deverão igualmente merecer a atenção dos professores. Na Figura 57 é possível observar a evolução do progresso dos estudantes de uma turma.

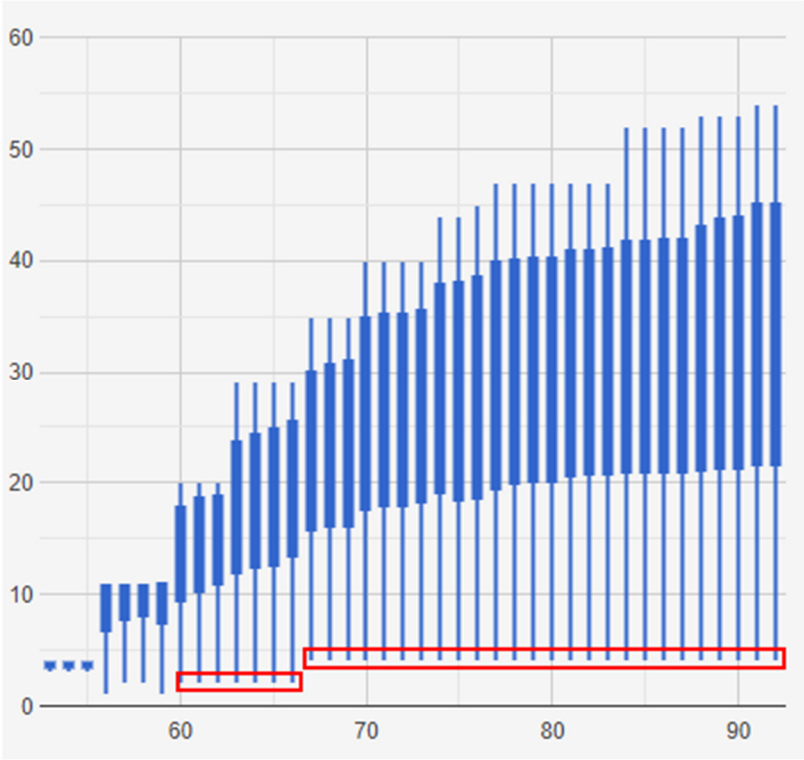

Figura 57 - Exemplo de visualização da evolução do progresso dos estudantes de uma turma - versão um.

Ao observar esta figura, um aspeto que se tornou evidente foi o facto de, ao existirem determinados estudantes que apenas tentaram realizar um ou dois exercícios e posteriormente desistiram (ou deixaram de utilizar o *plug-in*), este fenómeno era refletido nesta visualização, pois iriam existir sempre estudantes cujo valor de progresso era demasiado baixo, o que iria ter influência no cálculo do valor da média e de MAD. Este efeito encontra-se destacado na Figura 57. Para mitigar esta situação, foram adicionados atributos à base de dados que permitiam indicar que um estudante havia desistido e desta forma, em cada dia, eram apenas tidos em consideração os estudantes que ainda não tivessem desistido. O efeito da introdução desta modificação pode ser observado na Figura 58.

Também no caso do progresso é possível aceder a informação concreta sobre cada estudante e a sua posição em vários níveis de progresso (*Fast*, *Average* e *Slow*) através de uma visualização semelhante à representada na Figura 55. A partir desta visualização poderá também ter acesso a informação mais detalhada sobre cada estudante, nomeadamente, a comparação entre a evolução do progresso desse estudante e os restantes colegas da turma.

Na Figura 59 é possível observar essa informação relativa aos estudantes para os quais foi apresentada a evolução de performance anteriormente representada na Figura 56.

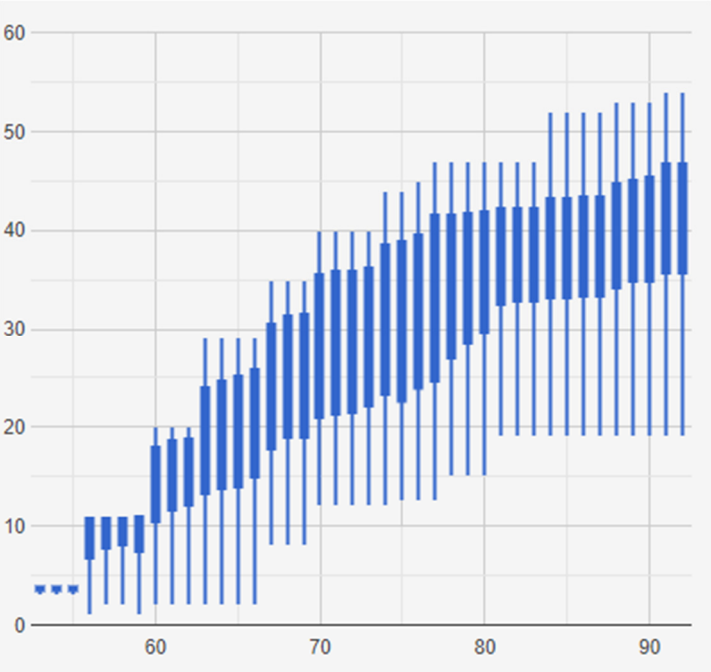

Figura 58 - Exemplo de visualização da evolução do progresso dos estudantes de uma turma - versão dois

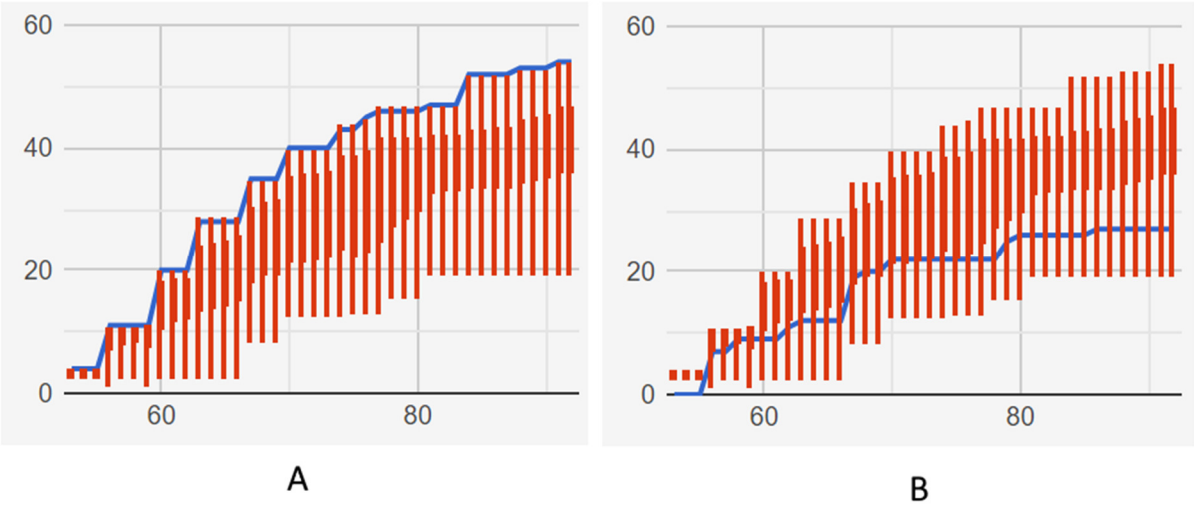

Figura 59 - Exemplos de visualização da comparação do progresso de estudantes face aos colegas de turma

Como anteriormente já tinha sido observado, o estudante A esteve visivelmente a trabalhar com uma performance muito acima dos colegas, na Figura 59 fica bem visível que o mesmo ocorre em relação ao seu progresso. Já em relação ao estudante B, é possível observar que de facto ele não desistiu, como poderia parecer. Em vez disso, continuou a trabalhar, muito embora a um ritmo muito lento e com um progresso global aquém do que seria de esperar.

4.4.3.3.3. Cumprimento de objetivos

Uma vez que os professores procediam à definição de objetivos para determinados períodos temporais, foi ainda possível desenvolver uma visualização na qual era possível observar a evolução do progresso dos estudantes face aos objetivos definidos. Na Figura 60 é possível observar um exemplo desta comparação relativamente aos valores médios de progresso dos estudantes de uma turma (vermelho) e os objetivos definidos pelo professor (azul).

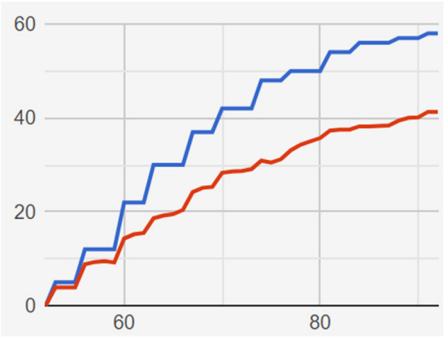

Figura 60 - Comparação da evolução do progresso dos estudantes face aos objetivos definidos pelo professor.

Como é possível observar, a partir de dada altura, começa a existir uma divergência considerável entre os objetivos e o valor real do progresso dos estudantes. Este facto por si só é indicativo de que existem vários estudantes que são incapazes de atingir os objetivos definidos, contudo, nesta figura é possível observar que apesar de tudo, a diferença entre ambas as curvas assume um valor relativamente constante.

Também a este nível é fornecida a possibilidade ao professor de aceder a uma visualização semelhante à apresentada na Figura 60 com informação sobre o progresso de cada estudante. Na Figura 61 é possível observar estas visualizações relativas aos estudantes A e B anteriormente mencionados.

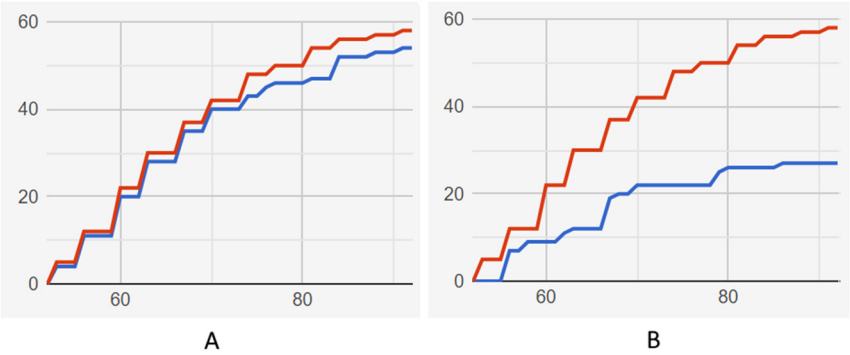

Figura 61 - Exemplos de visualizações de comparação da evolução do progresso de estudantes em concreto face aos objetivos definidos pelo professor

Como os dados anteriormente apresentados poderiam levar a concluir, existe uma aproximação muito grande entre o progresso do estudante A e os objetivos definidos pelo professor. Em sentido oposto, existe uma clara diferença entre o progresso do estudante B e os objetivos definidos pelo professor, diferença esta que como é visível vai aumentando à medida que o tempo passa.

4.4.3.3.4. Consistência

No já mencionado trabalho de Ahadi et al. [120] foi utilizado o conceito de performance média dos estudantes por forma a classificá-la periodicamente em quartis. Com base nesta informação, os autores pretenderam averiguar se a consistência do trabalho dos estudantes (quantidade de mudanças do quartil em que os estudantes se encontravam) poderia ser preditor do seu desempenho final. Tendo por inspiração este trabalho, procedeu-se à implementação de visualizações que permitissem complementar as que foram apresentadas nas secções anteriores. Na prática, estas visualizações tinham como objetivo representar de uma forma simples se ao longo do tempo a performance e o progresso dos estudantes haviam sido consistentes. Na Figura 62 é possível visualizar um exemplo destas visualizações relativas ao estudante B anteriormente referido.

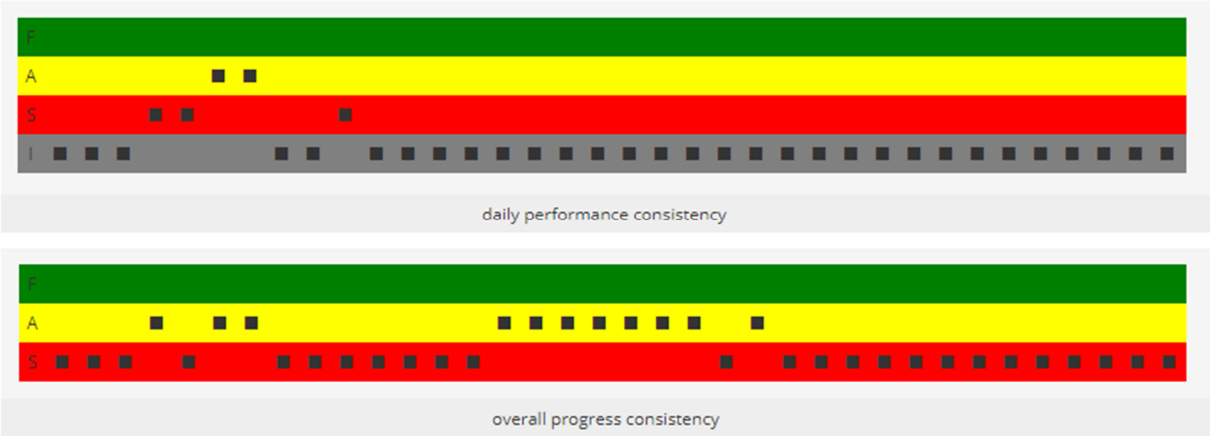

Figura 62 - Exemplo de visualização da evolução da consistência da performance e progresso do trabalho de um estudante

Como já anteriormente havia sido mencionado, quer a performance, quer o progresso dos estudantes poderia ser classificada como *Fast*, *Average* ou *Slow* em função da posição dos respetivos valores face aos valores dos colegas. No caso da performance é possível observar um outro nível I (*Inactive*) que significa que o estudante ou esteve de facto inativo, ou esteve a trabalhar em exercícios não incluídos dos objetivos daquele período. Desta figura é possível obter informação sobre a consistência de trabalho do estudante.

Tal como já havia sido verificado, no caso concreto do estudante B é possível observar que, à exceção de cinco dias, a sua performance foi consistentemente classificada com I. Ou seja, esteve quase sempre a trabalhar em exercícios "atrasados".

No caso concreto da performance, é possível observar na Figura 63, a mesma visualização para um estudante C que ao longo do tempo trabalhou claramente de uma forma inconsistente.

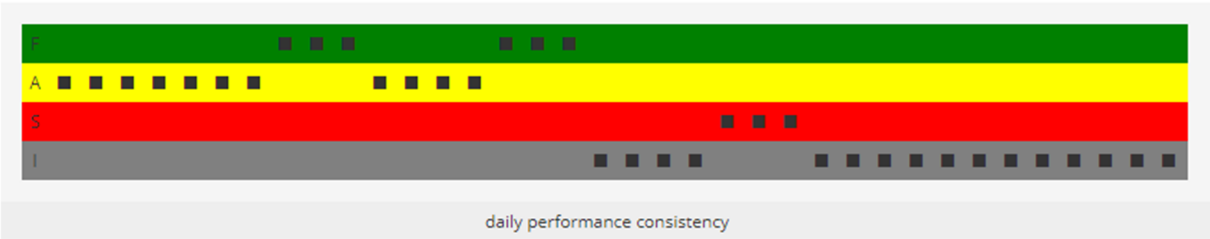

Figura 63 - Exemplo de visualização de performance inconstante de um estudante.

A inconsistência apresentada na Figura 63, poderia eventualmente ser reveladora de dificuldades e poderia levar o professor a averiguar as causas para esta inconsistência, podendo ou não ser necessário adotar algumas medidas adicionais para conseguir mitigar os eventuais problemas.

4.4.3.3.5. Notificações automáticas

Com base na informação disponibilizada, considerámos pertinente a disponibilização de dois novos tipos de notificação *off-line*, ou seja, que podiam ser geradas numa hora pré-definida. Assim, foram desenvolvidas notificações relativas à existência de estudantes com ritmos de trabalho fora do normal (quer em sentido positivo, quer em sentido negativo) e notificações relativas a mudanças de "nível de performance".

No primeiro caso, era definida uma determinada hora na qual diariamente era feita a verificação da existência ou não de estudantes consideravelmente mais avançados ou mais atrasados relativamente aos colegas. Em caso afirmativo, era gerada uma notificação na aplicação e era igualmente enviado um email ao professor com a notificação.

Em relação ao segundo tipo de notificações, procedia-se à comparação entre os "níveis de performance" em que os estudantes estavam no período anterior e atualmente. Caso existissem modificações em pelo menos um estudante, era gerada uma notificação.

Perante a receção destas notificações, os professores podiam verificar quais os estudantes em causa e agir como entendessem adequado.

4.4.3.4. Ferramenta de monitorização

Em relação à ferramenta de monitorização, para além da disponibilização das visualizações apresentadas nas secções anteriores, foram feitos pequenos ajustes em algumas visualizações disponibilizadas no *dashboard* e desenvolvidas algumas novas funcionalidades.

4.4.3.4.1. Edição e teste do código

Uma das novas funcionalidades permite aos professores selecionar um *snapshot*, editar uma versão temporária do código nele contido, compilar e executar essa nova versão. Esta funcionalidade permite, por exemplo, que o professor teste o código com outros valores de *input*, ou altere outro aspeto que considere relevante, sem que para tal tivesse necessidade de criar um novo ficheiro e abri-lo num ambiente de desenvolvimento.

Na Figura 64 é possível observar um exemplo desta funcionalidade de edição do código. Após fazer as modificações necessárias, o professor apenas necessita pressionar o botão "*Run*" e o código é enviado para o servidor de "*sand-box*", será compilado e executado (ou interpretado) e o resultado apresentado numa janela.

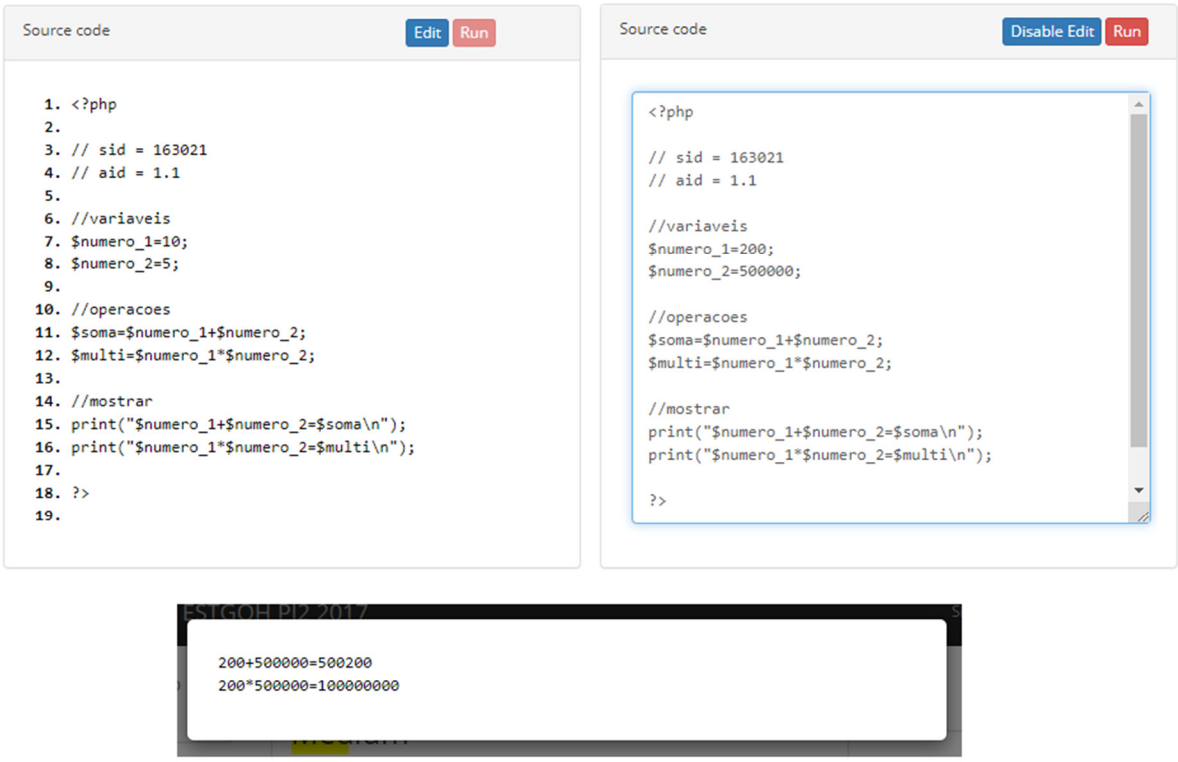

Figura 64 - Exemplo ilustrativo da funcionalidade de edição e teste de código pelo professor - iteração três.

Como foi mencionado anteriormente, o sistema iria tentar verificar automaticamente se o *output* gerado era igual ao esperado ou não. No entanto, devido às limitações do sistema de verificação utilizado, foi desenvolvida uma funcionalidade que permitia ao professor modificar a classificação atribuída, caso considerasse necessário, após ter avaliado e testado o código.

4.4.3.4.2. Listagem dos exercícios que os estudantes tentaram resolver

Com base na informação sobre a correção das soluções, foi igualmente possível atualizar a listagem dos exercícios realizados pelos estudantes, de modo a passar a incluir informação sobre o resultado na forma de um quadrado colorido. A cor destes quadrados variava em função da classificação: verde-escuro (certo); amarelo (parcialmente certo ou indefinido); e vermelho (errado). Na Figura 65 é apresentado um exemplo desta listagem.

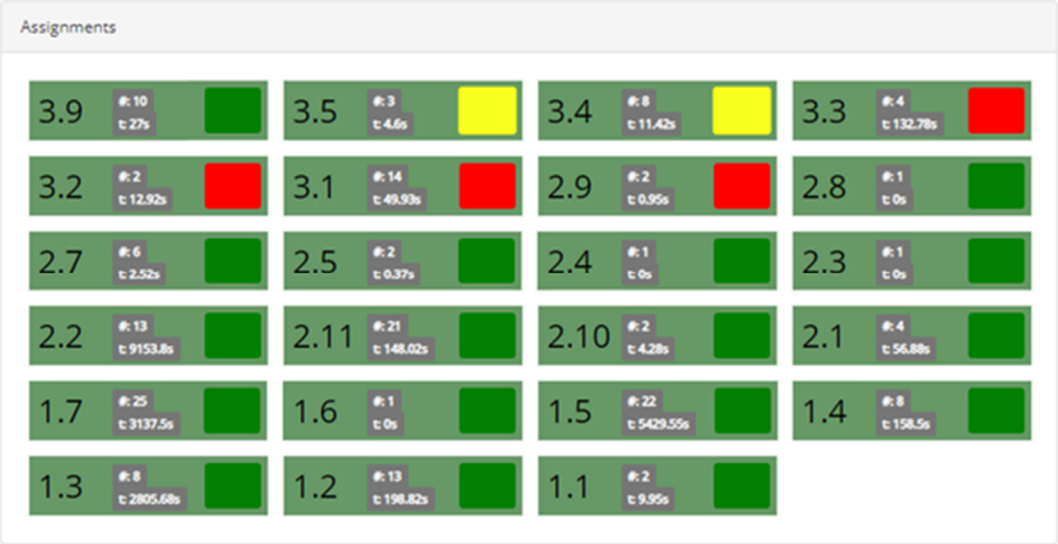

Figura 65 - Listagem dos exercícios realizados por um estudante - iteração três.

4.4.3.4.3. Fornecimento e registo de *feedback*

Nas experiências realizadas anteriormente foi possível observar que existia um elevado número de estudantes a trabalhar fora do período das aulas, muitos dos quais não assistiam às aulas frequentemente. Perante a deteção de algo merecedor de uma intervenção, caso o estudante estivesse na sala de aula, o professor poderia facilmente abordar o estudante e prestar-lhe a devida assistência. Contudo, caso o estudante não estivesse na sala de aulas, o professor necessitaria de abrir o seu cliente de email, fazer uma cópia do código e escrever a mensagem. Assim, foi desenvolvida uma funcionalidade que permitia aos professores fornecerem *feedback* aos estudantes sobre o código contido num *snapshot* diretamente a partir do sistema. Para tal, o professor apenas tem que

escolher o *snapshot* sobre o qual pretende fornecer *feedback* e escrever o texto pretendido. Um exemplo deste formulário pode ser observado na Figura 66.

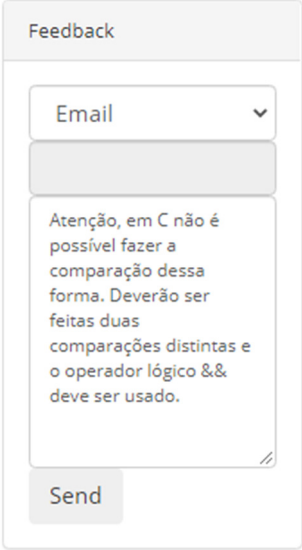

Figura 66 - Formulário de introdução de feedback a enviar aos estudantes

Quando o professor pressiona o botão "Send", é enviado para o email do estudante, uma mensagem com o código fonte associado ao *snapshot* selecionado junto com as anotações escritas pelo professor. Na Figura 67 é possível observar um exemplo de um email enviado.

Figura 67 - Exemplo de email enviado aos estudantes com *feedback* sobre um *snapshot*.

O mesmo formulário foi utilizado para permitir aos professores fazer registos sobre *feedback* fornecido diretamente aos estudantes de forma presencial. Posteriormente, caso necessário, os professores poderiam identificar os *snapshots* sobre os quais havia fornecido *feedback*. Para tal, deveria procurar por *snapshots* com o símbolo "*" para *feedback* enviado por email e com o símbolo "#" para registo de *feedback* presencial. Na Figura 68 é possível observar um exemplo da lista de *snapshots* contendo estes símbolos.

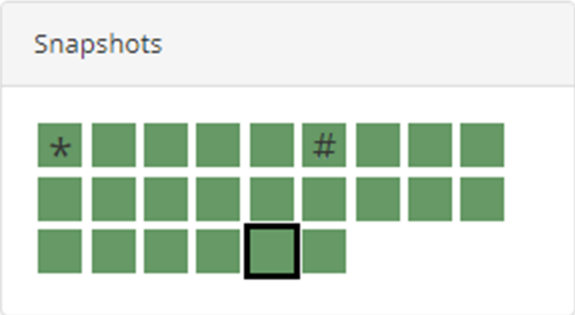

Figura 68 - Lista de *snapshots* com informação sobre fornecimento de *feedback*

4.4.4. Avaliação do sistema

Tal como previsto, o sistema foi avaliado em duas experiências distintas, cujos principais resultados se apresentam de seguida.

4.4.4.1. Programação (LEGI-FCT-UC)

Esta experiência decorreu entre os dias 12 de fevereiro e 28 de março de 2017, em quatro turmas da unidade curricular de Programação. Como foi visto anteriormente, o sistema já tinha sido testado num ano anterior na mesma unidade curricular. No entanto há que sublinhar que, entretanto, tinha havido uma reformulação do curso, tendo esta unidade curricular mudado para o 2º semestre, passando a ser a única unidade curricular de programação, enquanto anteriormente era a primeira de duas unidades curriculares de programação.

Estavam inscritos nesta unidade curricular um total de 87 estudantes (37 do sexo masculino e 50 do sexo feminino). Entre estes estudantes, 43, ou seja, aproximadamente metade, eram repetentes.

Ao longo da duração da experiência foram propostos para resolução cerca de 60 exercícios organizados em seis fichas práticas. Foram recebidos mais de 15.000 *snapshots*, tendo

apenas sido considerados válidos cerca de 9950 provenientes de 45 estudantes. Este elevado número de *snapshots* inválidos resulta principalmente da receção de pequenos pedaços de código utilizados pelos estudantes para efeitos de teste. Para além disso, em número muito menor, foram recebidas algumas centenas de *snapshots* sem todos os elementos identificativos.

Tal como já havia sido observado na anterior utilização do sistema ***CodeInsights*** nesta unidade curricular, existiam alguns estudantes a trabalhar em par, pelo que o número de estudantes de facto envolvidos na experiência foi superior.

Aproximadamente 38% dos *snapshots* foram recebidos durante o período das aulas, tendo os restantes 62% sido recebidos fora deste período. Entre os *snapshots* recebidos, aproximadamente, 3000 originaram erros de interpretação, sendo que 51,9% deles diziam respeito a *SyntaxError*, 12% *TypeError*, 6% *IdentationError*, 10% *NameError* e os restantes 20,1% diziam respeito a erros de *run-time*. Apenas oito estudantes fizeram pelo menos uma tentativa de resolução dos exercícios de dificuldade mais alta, tendo os exercícios com dificuldade mais baixa sido os mais populares entre os estudantes.

Contrariamente ao que acontecia em edições anteriores desta unidade curricular, nenhum dos exercícios das fichas práticas contava para efeitos de avaliação. A principal consequência desta decisão resultou numa grande dispersão do trabalho dos estudantes ao longo do tempo, como é bem visível na Figura 69**Erro! A origem da referência não foi encontrada.**.

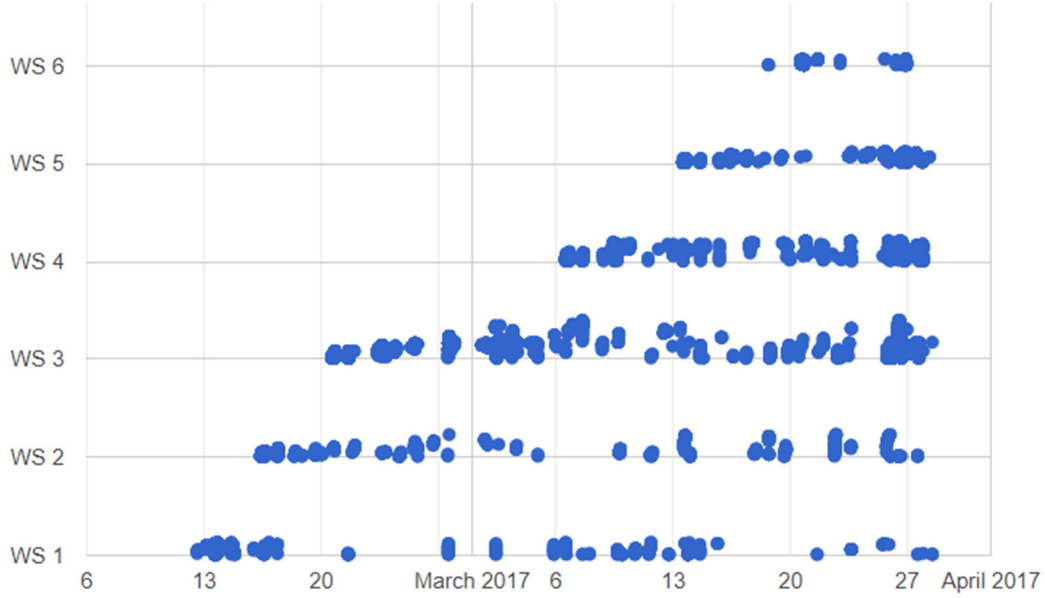

Figura 69 - Distribuição dos *snapshots* ao longo do tempo - Programação - LEGI - iteração três

Na Figura 69 é possível observar que existiram diversos estudantes a trabalhar em exercícios muito tempo depois do período em que era suposto eles tentarem resolvê-los. Esta informação é relevante e poderá certamente ser utilizada pelo professor da unidade curricular na preparação da edição da unidade curricular no ano letivo seguinte.

No final do semestre viria a ser possível confirmar, que na maioria dos casos, os estudantes assinalados no sistema como mais regulares, isto é, que conseguiram resolver mais exercícios com sucesso, acabariam por obter as classificações mais elevadas. Em sentido contrário, os estudantes que com base na informação disponibilizada no sistema evidenciavam mais dificuldades (*i.e.* mais tentativas de resolução com erros, mais exercícios em que última tentativa de resolução apresentava erro) acabariam por apresentar os piores resultados.

Relativamente às notificações, neste caso, o professor optou por utilizar apenas notificações do tipo *online* relativas a quantidades excessivas de tentativas de resolução de um exercício, utilizando o valor 35 como limiar. No total foram geradas 17 notificações deste tipo. Após investigação do professor foi possível comprovar que de facto alguns estudantes estavam a ter dificuldades a utilizar determinadas instruções (*por exemplo,* condições de paragem em ciclos), sendo o número elevado de tentativas resultado de uma abordagem do tipo tentativa/erro.

Finalizada a experiência, realizou-se uma reunião com o professor, na qual foi perguntado sobre até que ponto a imagem que o sistema passava dos estudantes correspondia efetivamente à sua perceção real. A este respeito, o professor afirmou que no geral, o sistema conseguiu de facto espelhar de uma forma bem próxima a realidade. Este professor manifestou uma vez mais o quão interessante e útil era a visão que o sistema lhe permitia ter sobre o desempenho dos estudantes, tanto numa perspetiva global da turma como na especificidade de cada estudante individualmente, principalmente em relação ao trabalho feito fora das aulas e ao facto de a informação ficar realmente disponível de forma imediata.

4.4.4.2. Programação para a Internet II (ESTGOH - IPC)

Esta experiência decorreu entre os dias 24 de fevereiro e 20 de março de 2017 no âmbito de uma unidade curricular de introdução ao desenvolvimento *web server-side* com PHP. A participação na experiência era totalmente voluntária, contudo, era atribuída uma

parte da classificação final para os exercícios propostos para resolução. No caso de um estudante não pretender participar na experiência, não havia qualquer penalização, podendo, tal como acabou por acontecer com dois estudantes, ser feito o envio das resoluções via *Moodle*.

Estavam inscritos a esta unidade curricular 39 estudantes, tendo durante a experiência sido recebidos *snapshots* de apenas 27. No total foram recebidos aproximadamente 16.000 *snapshots*, tendo cerca de 2500 sido considerados inválidos e/ou não passíveis de serem identificados. Aproximadamente 39% dos *snapshots* foram recebidos durante o período das aulas, tendo os restantes 61% sido recebidos fora deste período. A este respeito, há que destacar um aspeto curioso e que se prende com o facto de os estudantes terem duas aulas de três horas por semana desta unidade curricular, a seguir à qual tinham outra aula de duas horas de outra unidade curricular com o mesmo professor. Contudo, como é visível na Figura 70 os *snapshots* foram sendo recebidos de forma bem uniforme ao longo de todos os dias da semana, inclusivamente nos períodos de aulas de outras unidades curriculares.

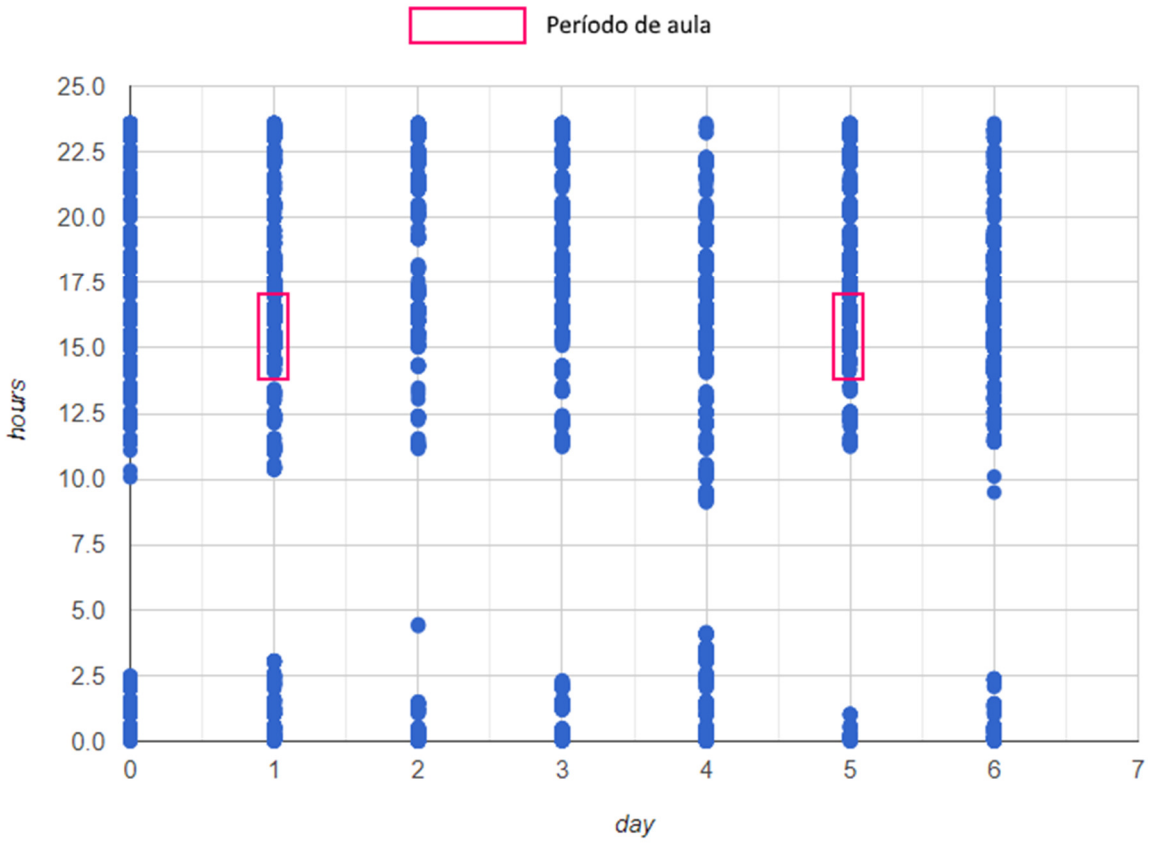

Figura 70 - Distribuição dos *snapshots* em função da hora e dia da semana - iteração três

Ao longo da duração da experiência foram propostos para resolução 51 exercícios distribuídos por cinco fichas de trabalho. Na Figura 71 pode ser observada a distribuição dos *snapshots* ao longo do tempo.

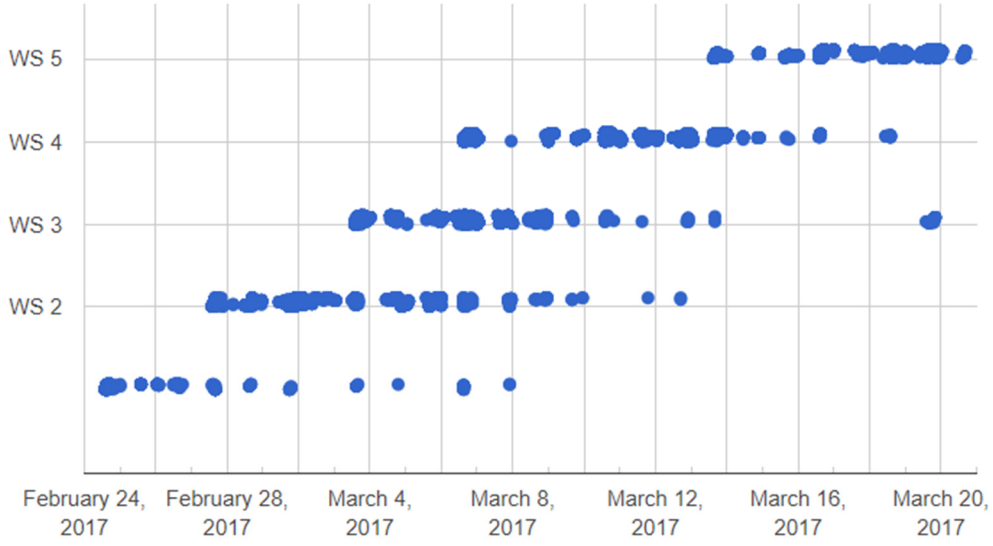

Figura 71 - Distribuição dos *snapshots* ao longo do tempo (PI2 - LEI - ESTGOH - iteração três).

Uma vez mais é possível verificar que existem estudantes a trabalhar com ritmos distintos, existindo a dada altura estudantes a trabalhar na ficha nº4 (WS 4) ao passo que alguns outros ainda resolviam exercícios da ficha nº1 (WS 1). Neste caso concreto, acaba por ser notável o facto de apesar de existir um *deadline* para as submissões de modo a estas serem consideradas para efeitos de avaliação, muitos estudantes continuaram a trabalhar nos exercícios bem para lá desse *deadline*, o que se traduz na revelação de interesse e motivação em aprender por parte dos estudantes, mesmo que não conte para efeitos de avaliação.

Ainda relativamente às diferenças ao nível do ritmo de trabalho, nesta iteração foram disponibilizadas novas visualizações desenvolvidas concretamente para este efeito. Na Figura 72, é possível observar a evolução da performance dos estudantes ao longo da duração da experiência.

É possível observar que ao longo do tempo existiram sempre estudantes com níveis de performance elevados e alguns outros com níveis de performance muito baixos (perto de zero em alguns casos). É ainda possível sublinhar que em termos gerais a ficha de trabalho em que os estudantes conseguiram ter uma performance superior foi a primeira (WS1), sendo a terceira (WS3) aquela na qual globalmente, os níveis de performance foram mais baixos. Em termos médios é curioso verificar que ao longo de toda a

experiência o intervalo definido por `média +/- MAD` se manteve consistente, o que significa que existiu alguma consistência na dispersão da performance dos estudantes.

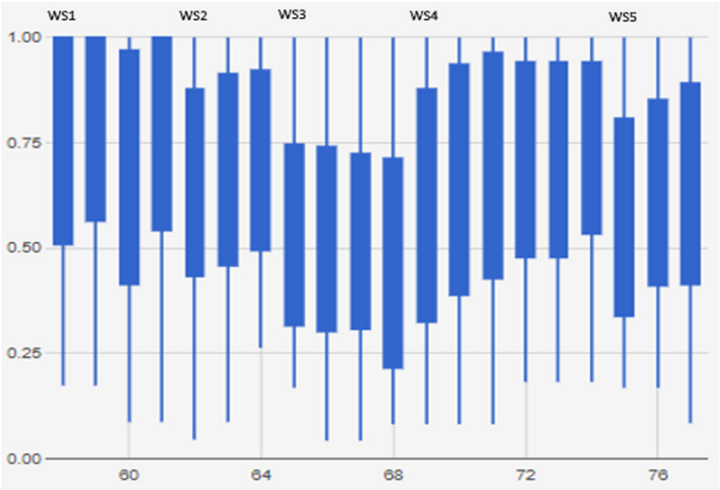

Figura 72 - Evolução da performance dos estudantes ao longo do tempo (PI2 - LEI - ESTGOH - iteração três).

Ainda ao nível da consistência dos estudantes de forma individual, foi possível verificar que existiram diversos estudantes a mudar consistentemente de "nível de performance", principalmente entre os grupos *fast* e *average*. Foi ainda possível constatar que de entre os estudantes mais frequentemente classificados como *slow* se encontravam principalmente estudantes repetentes (em alguns casos por mais de uma vez). Ao analisar o código escrito por estes estudantes era possível concluir que estes apresentavam sérias dificuldades ao nível de noções básicas de programação. Apesar das intervenções individuais e globais levadas a cabo pelo professor, apenas um dos estudantes consistentemente classificado como *slow* conseguiu obter aprovação (com 10 numa escala de 0 a 20). Em sentido oposto, apenas um dos estudantes classificados pelo menos uma vez como *fast* não conseguiu obter aprovação, tendo os restantes obtido excelentes classificações (16 a 19 numa escala de 0 a 20).

Relativamente à geração de notificações, nesta experiência foram utilizadas as notificações *online* para o número excessivo de tentativas de resolução para um exercício por um estudante utilizando 50 como valor para o limiar. No total, foram geradas 64 notificações deste tipo. Tal como no caso anterior, na maior parte das situações o número elevado de tentativas estava associado a abordagens do tipo tentativa/erro, o que poderia uma vez mais ser revelador de dificuldades. Em alguns casos era igualmente notório que os estudantes gravavam e testavam o seu código praticamente de cada vez que escreviam algo de modo a verificar se não existia nenhum erro, ou seja, em diversos casos o número elevado de tentativas não estava associado à existência de dificuldades, mas sim, ao

estilo de trabalho dos estudantes. Alguns acabariam por efetuar mais de 100 tentativas de resolução para o mesmo exercício (mais de 200 em duas situações).

Em relação às notificações sobre existência de estudantes a trabalhar a ritmos consideravelmente distintos e mudanças de "níveis de performance", a verificação foi feita diariamente às 8h00. Como pode ser observado na Figura 72, existiram sempre estudantes a trabalhar a ritmos consideravelmente distintos e existiu igualmente sempre pelo menos um estudante a mudar de "nível de performance". Deste modo, todos os dias foram geradas notificações a este respeito. Ainda que esta informação fosse útil, a partir de dada altura passou a ser previsível, pelo que as notificações acabaram por perder alguma relevância.

4.4.5. Sumário da iteração três

Nesta iteração foram adicionadas diversas novas funcionalidades ao sistema, como por exemplo o fornecimento de *feedback* aos estudantes diretamente a partir do sistema ou a funcionalidade que permite ao professor editar e testar o código dos estudantes. No entanto, o principal foco desta iteração foram as ferramentas de suporte ao fornecimento de assistência diversificada aos estudantes, quer ao nível de visualizações, quer ao nível de notificações. No geral, estas funcionalidades revelaram ser importantes e o professor que as testou mostrou-se agradado e sentiu-se apoiado com a sua utilização. Ainda assim, tornou-se evidente que tal como está pensado o sistema de notificações acaba por não ser tão útil como eventualmente se previu, tendo inclusivamente acabado por ser ligeiramente inconveniente nas situações sublinhadas acima.

Ainda ao nível das notificações, constatou-se que a utilização de um único limiar para todos os exercícios em termos do número invulgar de tentativas de resolução também não era suficiente, uma vez que existem exercícios com níveis de dificuldades bem distintos e que, como tal, poderão levar à necessidade de quantidades distintas de tentativas de resolução. Além disso, foi ainda verificado que existem abordagens distintas à resolução de um mesmo exercício, pelo que para um determinado estudante o facto de ter um número muito elevado de tentativas de resolução não é automaticamente indicativo de que este estudante possa estar com dificuldades.

Posto isto, parece evidente que as notificações são importantes, contudo o seu mecanismo de suporte deverá ser revisto.

Nesta iteração foi ainda visível que muito embora tivesse sido feito um esforço no sentido de ser tão pouco intrusivo quanto possível, o que era obviamente favorável do ponto de vista dos estudantes, em termos práticos o sistema de classificações dos exercícios era demasiado limitado. Como tal, deveria ser revisto por forma a ser efetivamente implementado um mecanismo de testes automatizados do código utilizando um conjunto de casos de teste, devendo, contudo, ser mantida a premissa da mínima intrusão/perturbação.

A dada altura da experiência de Programação para Internet II, alguns estudantes manifestaram que era interessante se estes também tivessem acesso a uma aplicação onde pudessem verificar quais os exercícios que foram recebidos pelo sistema, assim como qual o "estado" em que estavam, isto é, se tinham erros de compilação, por exemplo. Não existindo possibilidade de disponibilizar esta funcionalidade em tempo útil durante esta experiência, foi tida em consideração na iteração seguinte.

Globalmente é seguro dizer que se tratou de uma iteração bastante produtiva em que mais uma vez foi possível desenvolver novas funcionalidades, fazer modificações a algumas já existentes e testá-las em contexto real. Todo o valioso *feedback* fornecido de forma direta ou indireta por professores e estudantes constituiu um importante suporte para a preparação da iteração número quatro.

4.5. Iteração quatro

4.5.1. Objetivos

Na sequência da iteração anterior, um dos aspetos mais importantes a abordar na presente iteração prendia-se com o desenvolvimento de um mecanismo de execução do código dos estudantes com casos de teste. Como houve oportunidade de verificar, o mecanismo utilizado anteriormente era demasiado rudimentar e obrigava a demasiado trabalho adicional de modo a ser mantido o nível de qualidade aceitável.

Outro aspeto da maior importância que deveria ser desenvolvido nesta iteração prendia-se com um sistema de geração de notificações mais abrangente, de modo que pudessem ser disponibilizadas mais notificações, mas acima de tudo, de modo a serem geradas notificações mais relevantes.

Nesta iteração, acabaria ainda por se testar o sistema num contexto novo, mais concretamente, numa unidade curricular de desenvolvimento de páginas *web* com HTML, que muito embora não se trate verdadeiramente de programação, constitui ainda assim um cenário, em que a utilização do sistema ***CodeInsights*** pode contribuir para a melhoria do processo de ensino/aprendizagem.

4.5.2. Descrição do contexto

Para esta iteração foram estabelecidos contactos com vista a testar o sistema ***CodeInsights*** em quatro unidades curriculares distintas no primeiro semestre do ano letivo 2017/2018. No caso concreto de três destas unidades curriculares, o sistema já havia sido utilizado no ano letivo anterior, particularmente numa unidade curricular de introdução à programação em linguagem C (Fundamentos de Programação da Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital), numa unidade curricular de introdução à programação em linguagem Java (Programação da Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital) e numa unidade curricular de introdução à programação em Java (programa Acertar o Rumo da Universidade de Coimbra). Foi ainda possível testar o sistema com os estudantes da unidade curricular de Programação para a Internet I, integrada na Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital.

4.5.3. Trabalho desenvolvido

4.5.3.1. Desenvolvimento do *plug-in*

Tendo em consideração que haviam existido problemas de ordem técnica na anterior utilização do sistema nas unidades curriculares de Programação e Fundamentos de Programação da Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital, houve necessidade de proceder a modificações relativamente à unidade de Fundamentos de Programação. Para esta iteração foi acordado com o professor responsável, a adoção de um ambiente de desenvolvimento muito simples, tendo a escolha recaído sobre o editor *Atom,* principalmente pelo facto de suportar diversas linguagens de programação e por poder ser utilizado nos principais sistemas operativos. Escolhido o editor, procedeu-se ao desenvolvimento do *plug-in* para este ambiente. Por forma a compilar o código

diretamente no *Atom* sem ter a necessidade de recorrer a uma linha de comandos/terminal, procedeu-se à utilização do *plug-in* ***gpp-compiler***[24], que permite compilar o código correspondente ao ficheiro que estiver aberto no editor, mediante a pressão da tecla F5. Caso o código tenha erros, será apresentada informação a esse respeito aos estudantes. Caso contrário, o código será executado no computador dos estudantes. Em qualquer caso, sempre que os estudantes pressionarem a tecla F5, é enviado um *snapshot* do código para o servidor.

O ambiente de desenvolvimento ***Atom*** acabaria igualmente por ser utilizado em Programação para a Internet I. Neste caso, para além de ser enviada uma cópia para o servidor, achou-se relevante o fornecimento de algum *feedback* aos estudantes sobre a qualidade do código fonte produzido. Assim, o *plug-in* a utilizar nesta unidade curricular foi implementado tendo por base o plug-in ***w3c-validation***[25], que utiliza a API do W3C (*World Wide Web Consortium*)[26] para aferir a qualidade do código HTML e apresentar informação sobre os *Errors* e *Warnings* aos estudantes diretamente no *Atom*. Assim, para além das funcionalidades de envio do código fonte para o servidor sempre que o estudante guardava o código (neste caso, não há lugar a compilação), o *plug-in* permitia também fornecer *feedback* aos estudantes por forma a saberem quais os aspetos que deveriam melhorar no seu código. Na Figura 73, apresenta-se um exemplo da informação dada aos estudantes diretamente no ***Atom***.

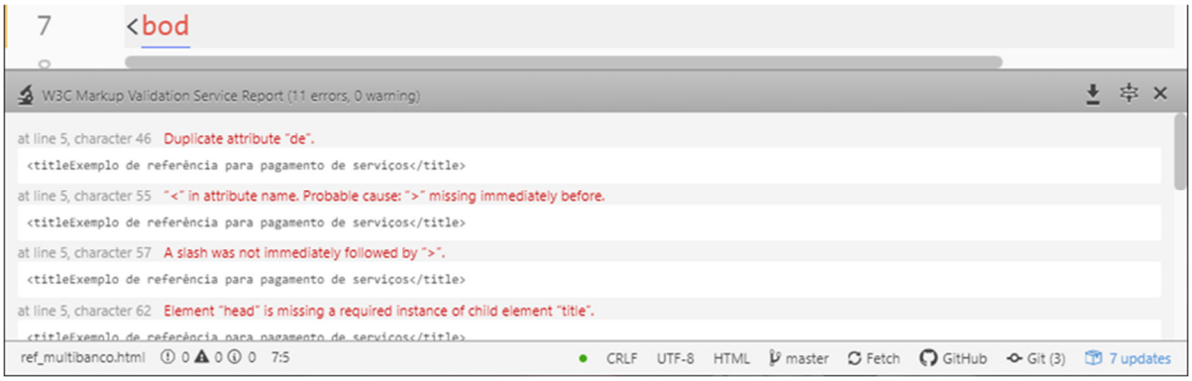

Figura 73 - Exemplo da apresentação de *feedback* do validador de HTML do W3C.

[24] https://atom.io/packages/gpp-compiler

[25] https://atom.io/packages/w3c-validation

[26] https://validator.w3.org/docs/api.html

4.5.3.2. Receção e processamento do código fonte

No caso concreto da linguagem HTML, em termos de execução automática do código do servidor, nada era feito, uma vez que em termos de documentos HTML a sua execução consiste simplesmente na apresentação do resultado da renderização do código num navegador de internet.

Relativamente às restantes linguagens de programação, um dos aspetos que era fundamental abordar e do qual estavam dependentes uma série de funcionalidades estava associado ao processo de execução do código dos estudantes com um conjunto de casos de teste definidos pelos professores por forma a determinar se estava ou não a produzir o resultado esperado. Assim, e de modo a suportar diversas outras funcionalidades do sistema, procedeu-se a algumas modificações no modo como o código fonte era processado.

Até então, em termos do processamento do código fonte, após a identificação do estudante e do exercício, procedia-se à compilação/interpretação do código. Caso não ocorressem erros de compilação/interpretação, procedia-se à execução do código fonte, tal como ele foi desenvolvido pelos estudantes, e procedia-se à recolha do *output* produzido para ser comparado com o valor esperado.

Infelizmente, este não é um processo tão linear como à primeira vista poderá parecer, dada a variedade de exercícios e formas diferentes de os resolver, nomeadamente ao nível da definição/introdução de *inputs*.

Ainda que se pretendesse que o ***CodeInsights*** pudesse ser utilizado sem impor alterações ao funcionamento normal das aulas e à forma como os estudantes resolvem os problemas, para desenvolver esta nova funcionalidade foi necessário definir algumas regras e introduzir pequenos ajustes nos enunciados dos exercícios disponibilizados pelos professores. O grande desafio passou por fazer com que as modificações necessárias fossem tão simples e naturais quanto possível, de modo a não prejudicarem de maneira nenhuma o processo de aprendizagem dos estudantes.

Como já referido, o principal problema prende-se com a definição dos valores de *input* associados às variáveis, de modo que o valor destes possa ser modificado para os valores de teste. Uma vez que é possível fazê-lo de formas completamente diferentes, torna-se, pois, um grande desafio conseguir garantir que os programas sejam sempre devidamente testados. A título de exemplo, assumindo que se pretende desenvolver um programa em

que, fornecidos dois números, se proceda à apresentação da sua soma. Na Figura 74, é possível observar algumas possibilidades de resolução deste exercício utilizando a linguagem *Python*. No primeiro exemplo, recorre-se à definição dos valores dos *inputs* diretamente no código, no segundo exemplo, através da utilização da função `input` que permite recolher valores do teclado e, no último exemplo, através da utilização de argumentos de uma função.

```
# A

# sid 123123
# aid 1.01
a = 4

b = 2

res = a + b

print res
```

```
# B

# sid 123123
# aid 1.01
a = input('valor para a?')

b = input('valor para b?')

res = a + b

print res
```

```
# C

# sid 123123
# aid 1.01
def soma(a,b)

        s = a + b

        print s

soma(2,3)
```

Figura 74 - Exemplos de diferentes formas de definição dos *inputs*

Qualquer uma das soluções apresentadas permite resolver o problema e pode facilmente ser testada localmente no computador dos estudantes e professores. A grande questão era saber como testar devidamente este código no servidor de testes (*sand-box*). Neste contexto de interpretação/compilação e execução automática do código fonte no servidor de testes, o exemplo A não acarreta qualquer problema, pois todos os valores estão definidos à partida. Ainda assim, é necessário determinar como associar os valores dos *inputs* dos casos de teste às variáveis definidas pelos estudantes.

Neste exemplo em concreto, nem é muito preocupante pois `a + b` irá produzir o mesmo valor que `b + a`, pelo que poderia considerar-se simplesmente que se podiam associar os valores dos *inputs* às variáveis pela ordem que estivessem definidas no código. No entanto, a menos que fosse garantido que essa ordem estava pré-estabelecida e que os estudantes forçosamente teriam que a respeitar, tal método não seria viável noutros casos, por exemplo quando as operações envolvidas não forem comutativas.

Já no caso B, a execução automática do código no servidor gerará um erro do tipo `EOFError`, pois não há "ninguém" a introduzir os valores para as funções **`input`**. Este tipo de cenário ocorre de igual modo com outras linguagens de programação. Por exemplo, em C (**`scanf`**) e Java (**`scanner.next()`**). Para estes cenários, poder-se-ia pensar em recorrer a algum processo de instrumentação de código no qual se procedesse à identificação das

funções de recolha de *inputs*, substituindo-as posteriormente pelos valores dos *inputs* de cada caso de teste. No entanto, a não ser que também neste caso seja garantido que os valores dos *inputs* são solicitados ao utilizador por uma determinada ordem previamente estabelecida, é difícil determinar o que corresponde a quê.

Finalmente, também o exemplo C acarreta alguns problemas. Nomeadamente, era necessário garantir que a ordem pela qual os argumentos são definidos está previamente estabelecida.

Era ainda possível recorrer a uma quarta forma de introduzir os *inputs* que passaria pelo envio dos valores na forma de argumentos pela linha de comandos (ex. `subtrai.exe 1 3`) e posterior obtenção através das variáveis `argc` e `argv` ou os seus correspondentes em outras linguagens de programação. No entanto, também em relação a esta possibilidade, era obrigatório definir previamente a ordem pela qual os argumentos eram passados, pois o resultado de `subtrai.exe 1 3` é diferente de `subtrai.exe 3 1`. Para além do mais, pelo que nos foi dado a conhecer, esta forma de obtenção de *inputs* dos utilizadores não é frequentemente utilizada nas unidades curriculares introdutórias e nas situações em que é utilizada, tal ocorre numa fase muito avançada.

Como seria de esperar, procedeu-se a uma análise do modo como outras ferramentas geralmente lidam com esta situação. Assim, ferramentas como ***TestMyCode*** ou ***OK*** recorrem à utilização de testes unitários mediante a utilização de *frameworks* como ***Junit*** ou ***unittest.*** Nestes casos, junto com o enunciado dos exercícios, é igualmente fornecido algum tipo de *scaffolding* na forma de um projeto do *Eclipse* ou na forma do "protótipo" da função ou métodos a desenvolver. Ou seja, os estudantes não têm total liberdade para fornecer nomes às classes, funções ou métodos, uma vez que estes já estão definidos à partida e, por forma que os testes unitários funcionem, é imperativo que não sejam modificados.

Após contactos com professores que já haviam testado o sistema no passado, acabou por se concluir que no caso concreto do sistema ***CodeInsights*** esta opção não era aconselhável, pois obrigaria a uma perda de liberdade muito grande ao nível do modo como os estudantes desenvolvem o seu código. Para além disso, obrigaria a que fossem sempre utilizadas funções, facto que comprometeria a sua utilização na unidade curricular em que é utilizada a linguagem *Python*, por exemplo, uma vez que a utilização de funções apenas era abordada numa fase mais avançada. Outro inconveniente desta abordagem consiste no facto de que passaria a ser necessário que os professores

desenvolvessem o código dos testes unitários associados a cada exercício, o que até então não acontecia e implicava algum trabalho adicional.

Pelos motivos apresentados, tornou-se claro que não era possível proceder à execução do código dos estudantes com casos de teste, sem que fosse necessário impor algum tipo de obrigação/restrição no modo como os estes constroem a sua solução. Perante esta constatação, estabeleceu-se um procedimento genérico de testes que fosse simples e pouco laborioso para os professores, mas também simples de utilizar pelos estudantes, dando-lhes o máximo de liberdade possível para desenvolverem as suas soluções.

Após diversas conversas com os professores optou-se por utilizar uma solução de compromisso entre a simplicidade de criação de novos exercícios pelos professores, a liberdade dada aos estudantes para desenvolverem as suas soluções e a rigidez necessária para conseguir executar corretamente o código com os casos de teste fornecidos. Esta passou pela imposição do nome dado às variáveis de *input* de cada programa. Aquando da criação do enunciado de um exercício, os professores deverão incluir informação sobre o nome das variáveis que devem ser utilizadas para receber os *inputs* da aplicação. Para além disso, é igualmente pedido aos estudantes que coloquem o código referente à definição e inicialização das variáveis de *input* numa zona do código devidamente delimitada. Na Figura 75, é possível observar como ficava o código dos exemplos previamente apresentados após ser reescrito de acordo com as novas recomendações.

```
# A

# sid 123123
# aid 1.01
#begin_inputs
a = 4
b = 2
#end_inputs
res = a + b

print res
```

```
# B

# sid 123123
# aid 1.01
#begin_inputs
a = input('valor para a?')
b = input('valor para b?’')
#end_inputs
res = a + b

print res
```

```
# C

# sid 123123
# aid 1.01
#begin_inputs
a = 4
b = 2
#end_inputs
def soma(v_a,v_b)

        s = v_a + v_b

        print s

soma(a,b)
```

Figura 75 - Exemplos de utilização do método de definição de *inputs*

Deste modo, independentemente da solução utilizada pelos estudantes, e existirão certamente outras soluções para além das apresentadas, desde que seja respeitada a forma como as variáveis de *input* são declaradas e inicializadas, o sistema conseguirá executar o código com os casos de teste.

Como anteriormente mencionado, a grande maioria dos ambientes de desenvolvimento, permite a criação de *templates* a serem utilizados aquando da criação de um novo ficheiro. Assim sendo, de modo a facilitar o trabalho dos estudantes, os *templates* utilizados passariam a incluir a parte relativa à definição dos *inputs*.

Por forma a verificar se um exercício está de facto a produzir ou não o resultado esperado, aquando da criação de um novo exercício, os professores poderão especificar um ou dois casos de teste. Para cada caso de teste, é necessário especificar os *inputs* a serem utilizados e o *output* esperado para esses *inputs*. Na Figura 76, pode ser observada a seção do formulário que os professores deverão utilizar para este efeito para um exercício cujo objetivo passa por substituir numa frase (`frase`) todas as ocorrências de elementos de pontuação por um conjunto de caracteres (`substituto`).

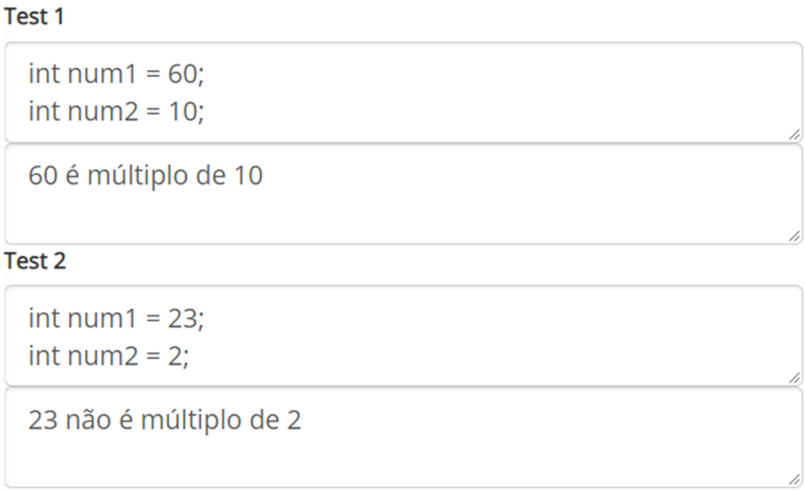

Figura 76 - Formulário de definição de casos de teste

Com base nesta nova configuração, o sistema de processamento dos *snapshots* necessitou de sofrer algumas modificações. Assim, após o código ser compilado tal como foi escrito pelos estudantes, caso não existam erros de compilação, proceder-se-á a criação de uma ou duas novas versões do código nas quais o conteúdo existente entre `#begin_inputs` e `#end_inputs` será substituído pelos valores definidos pelos professores aquando da criação do enunciado. Os novos ficheiros resultantes deste processo serão enviados para serem testados no servidor *sand-box*. Chegados ao servidor, são compilados e no caso de

não existirem erros de compilação, o *output* resultante da execução será recolhido para posteriormente ser comparado com o valor esperado para os *inputs* utilizados. Este processo pode ser observado na Figura 77.

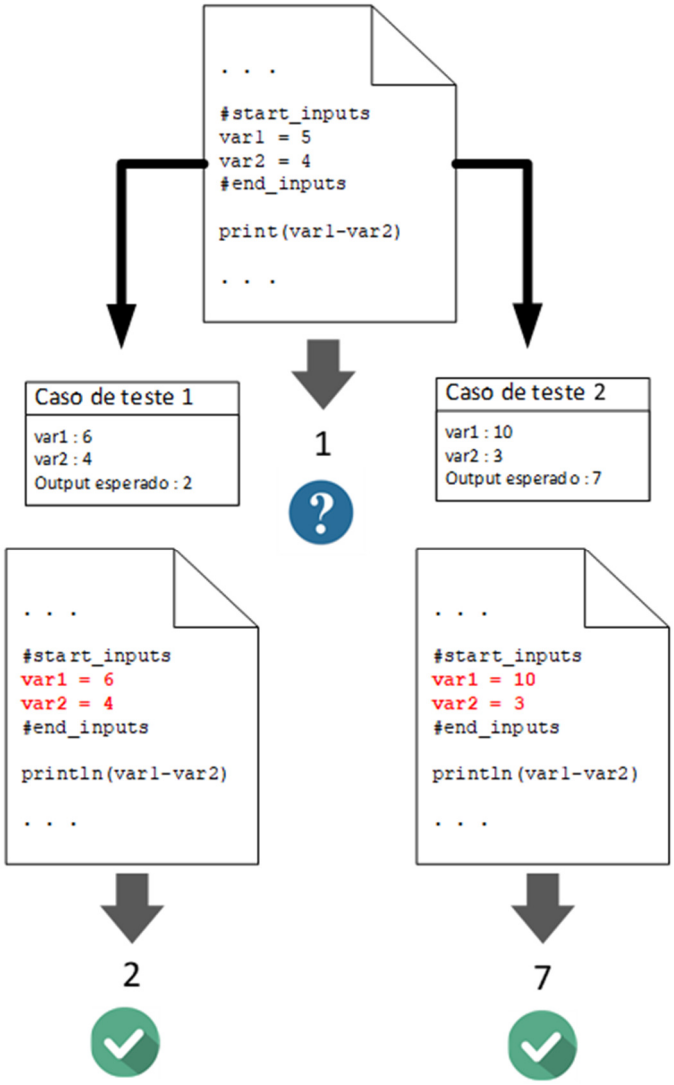

Figura 77 - Ilustração do processo de substituição do valor dos inputs

Uma vez que o conteúdo existente entre `#begin_inputs` e `#end_inputs` será integralmente substituído, é crucial que os estudantes não incluam nesta "zona" outro código que inclua declaração e inicialização de outras variáveis para além das variáveis de *input*. É igualmente necessário que os valores das variáveis de *input* não sejam modificados fora do âmbito predefinido. Muito embora se trate de um processo bastante simples, em todo o caso, para além do manual de instalação e utilização do sistema *CodeInsighs* que já era fornecido aos estudantes, procedeu-se à criação de um guia de consulta rápida (*cheat sheet*) com informações sobre as regras associadas ao processo de

criação de código que os professores deverão fornecer aos estudantes aquando da primeira aula em que o sistema seja utilizado (ver Anexo I).

Para além dos *inputs*, foi igualmente necessário lidar com alguns aspetos relacionados com o *output*. Por forma a aferir se o *output* produzido pelos programas é igual ao expectável, optou-se por fazer uma comparação direta. Deste modo, se os *outputs* produzidos pelos programas utilizando os *inputs* dos casos de teste forem exatamente iguais aos *outputs* esperados, considera-se que o exercício está correto, caso contrário considera-se que o exercício está incorreto. Para tal, os professores deverão incluir nos enunciados dos exercícios um ou mais exemplos de como deverá ser apresentado o *output*. Por exemplo, se o resultado esperado para a um exercício que consista na soma de dois números fosse algo como `2 + 2 = 4`, caso os estudantes não formatem o seu *output* exatamente como esperado (*e.g.* `soma = 4` ou `resultado = 4`), muito embora a solução em si possa estar correta, de um ponto de vista objetivo, trata-se efetivamente de resultados diferentes e, como tal, o exercício era assinalado como não estando a devolver o resultado esperado. No Anexo II é possível observar um exemplo de uma ficha prática tal como era disponibilizada antes da utilização do sistema ***CodeInsights*** e no Anexo III a mesma ficha prática com indicação dos *inputs* e *outputs* esperados.

Esta necessidade de fornecer indicações sobre os nomes das variáveis associadas aos *inputs* e o formato esperado dos *outputs*, embora vá um pouco contra o princípio da não interferência no processo de ensino que sempre guiou este trabalho, acaba por constituir um mal menor face aos benefícios que poderão resultar do facto de se conseguir efetivamente verificar se um programa está ou não a produzir o resultado esperado.

Ainda a respeito da questão dos *inputs*/*outputs*, há que referir que sistemas como o *Spinozza* [101], por exemplo, não utilizam sempre os mesmos dados para *input* e *output*. Em vez disso, aquando da criação dos exercícios, são definidas gamas de valores válidos para *input*. No momento do teste, são obtidos valores para os *inputs* de forma aleatória, necessitando o professor de implementar ele próprio o programa por forma a executar em paralelo a sua versão e a versão desenvolvida pelo estudante na qual serão utilizados os mesmos *inputs* gerados aleatoriamente. Posteriormente, procede-se à comparação de ambos os *outputs*. Caso sejam iguais, o exercício é considerado correto, caso contrário, será considerado incorreto.

É compreensível que esta solução tem a vantagem de efetivamente conseguir testar o código sempre com valores diferentes, o que inviabiliza que os estudantes tenham a

tentação de viciar o sistema através da produção direta de *output* para um determinado conjunto de *inputs*. No entanto, o esforço extra que o professor necessitaria de ter para codificar as suas soluções para todos os exercícios e a carga extra de processamento que era necessário colocar no servidor de testes acabou por determinar que, no nosso caso, não faria sentido utilizar este tipo de abordagens.

Em qualquer dos casos, após finalizado este processo de compilação/interpretação e execução dos *snapshots* com os casos de teste, procede-se ao envio da informação produzida para o servidor principal para se proceder ao armazenamento em base de dados de toda a informação criada. A partir deste momento, os professores poderão ter acesso a todos os dados resultantes do processamento do *snapshot* recebido. Como será visto adiante, nesta iteração foi desenvolvida uma ferramenta de monitorização destinada aos estudantes, pelo que também os estudantes poderiam verificar se o seu código está a produzir o resultado esperado ou não.

4.5.3.3. Base de dados de exercícios

Por forma a suportar as funcionalidades apresentadas na secção anterior, foi necessário fazer algumas modificações ao nível da base de dados. Apresenta-se, em seguida, a nova configuração da base de dados de exercícios.

Para além do **código identificativo** (ex. 1.1, 1.2, …) e do **enunciado**, nos casos em que tal seja possível, deverá ser inserida informação sobre um conjunto de ***inputs*** e respetivos ***outputs* esperados** e uma indicação do **grau de dificuldade** do exercício (e.g. MF - Muito Fácil, F - Fácil, M - Médio, D - Difícil e MD - Muito difícil). Muito embora os *snapshots* sejam sempre aceites, deverá igualmente ser inserida uma data indicativa do prazo de conclusão de um exercício (***deadline***).

Tendo em consideração que já é verificado se o resultado que o programa devolve está certo ou errado, houve a necessidade de incluir uma funcionalidade adicional de modo a garantir que os estudantes não resolvem os exercícios com recurso a determinadas funções proibidas. Para tal, o professor poderá introduzir um conjunto de *keywords* de utilização proibida (***black list***). Sempre que os estudantes utilizem, na resolução de um determinado exercício, uma ou mais *keywords* contidas na ***black list***, a solução em causa será assinalada como contendo algum aspeto indevido. Considere-se, por exemplo, que é pedido aos estudantes para escreverem na linguagem PHP o código necessário para ordenar um *array* de números inteiros. Uma vez que nesta linguagem já existe a função

`sort`, que faz o pretendido, o professor poderia inserir a *keyword* "sort" na *black list* e, deste modo, saberia quais os estudantes que tentaram resolver o exercício sem implementar um algoritmo de ordenação.

Finalmente, o professor deverá indicar qual ou quais os principais **conceitos de programação** que o exercício pretende exercitar (por exemplo: inicialização de variáveis, instruções condicionais, ciclos, ciclos encadeados, …). Esta informação será utilizada posteriormente para dar uma ideia sobre os conceitos em que os estudantes estão a ter mais ou menos dificuldades, o que permitirá ao professor tomar as medidas necessárias para colmatar as falhas que possam estar a existir.

Assim, para cada exercício, o professor terá de introduzir a informação resumida na Figura 78.

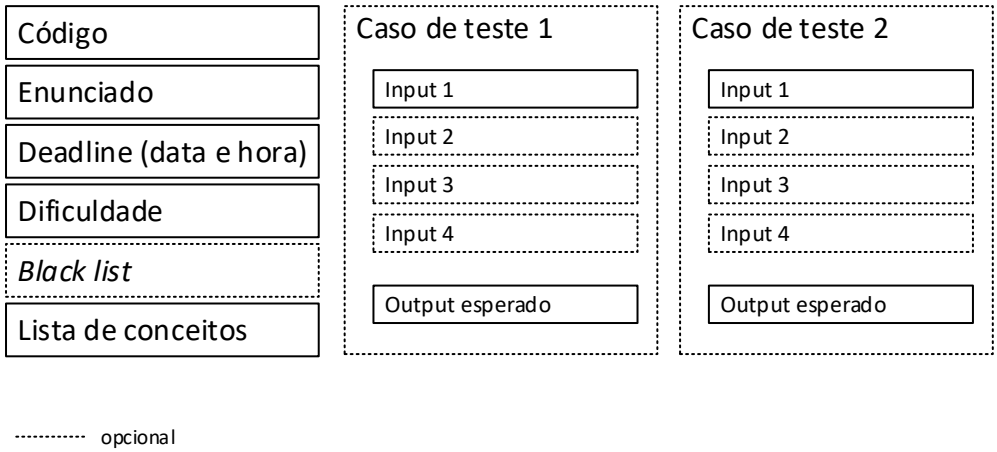

Figura 78 - Informação que é necessário introduzir para cada exercício - iteração quatro.

Na eventualidade de já não ser a primeira utilização do ***CodeInsights*** numa dada unidade curricular, a informação relativa à base de dados de exercícios poderá ser importada de uma edição anterior e posteriormente os exercícios poderão ser editados, removidos e, obviamente, poderão ser adicionados mais exercícios.

4.5.3.4. Ferramenta de monitorização - professores

Para além da criação de um conjunto de novas visualizações que serão apresentadas com detalhe nas secções seguintes, nesta iteração procedeu-se a algumas melhorias na organização da informação do *dashboard* de modo a apresentar a informação que foi considerada mais importante, sem o professor necessitar de fazer *scroll*. Um exemplo da nova configuração do *dashboard* é representado na Figura 79.

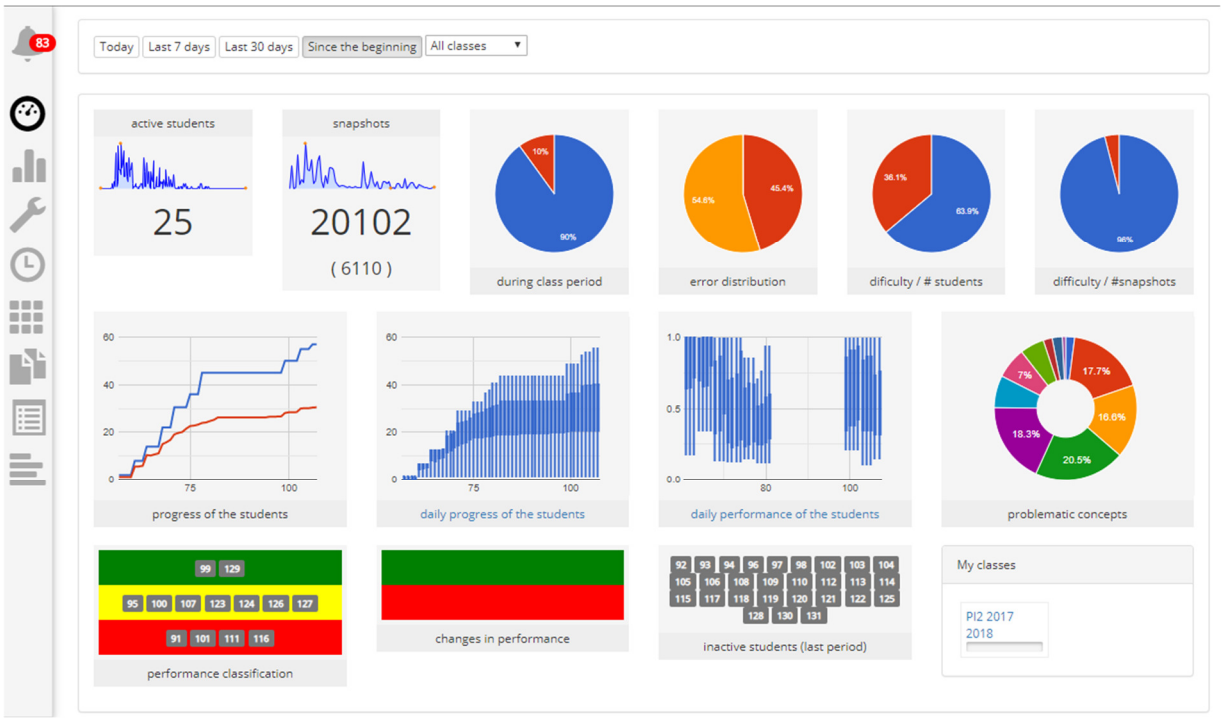

Figura 79 - Exemplo da nova configuração do *dashboard* - iteração quatro.

Para além desta mudança substancial no aspeto visual do *dashboard*, procedeu-se ainda à modificação de algumas funcionalidades já existentes e ao desenvolvimento de algumas funcionalidades novas que se passam a descrever.

4.5.3.4.1. Listagem dos exercícios que um estudante tentou resolver

Uma vez que passou a ser possível verificar se o resultado produzido por um programa era ou não igual ao esperado, foi decidido incluir esta informação diretamente da listagem de exercícios que cada estudante tentou resolver. Um exemplo da nova configuração desta listagem pode ser observado na Figura 80.

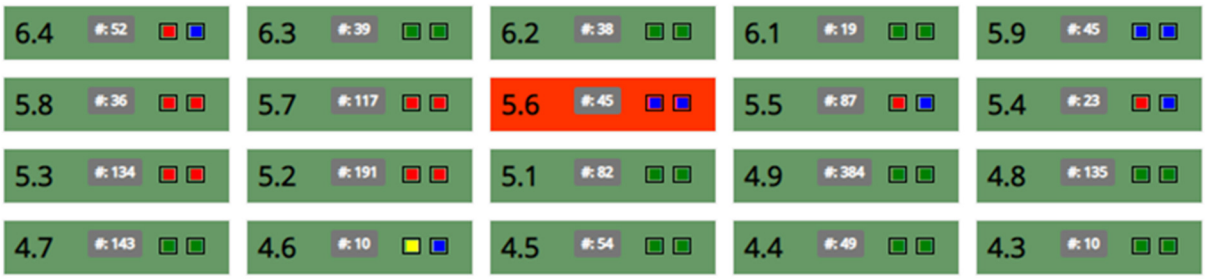

Figura 80 - Listagem com os exercícios que o estudante tentou resolver - iteração quatro

Para cada exercício para além do elemento identificativo é apresentada informação sobre o número de tentativas de resolução recebidas (*snapshots*) e informação sobre os

resultados da execução do código com os casos de teste (dois, no máximo). Para tal, são apresentados dois quadrados coloridos cuja cor apresenta o seguinte significado:

- Verde - o código produziu o resultado esperado para este caso de teste;
- Amarelo - o código produziu um resultado semelhante ao esperado (eventualmente, possuía pequenas diferenças de formatação)
- Vermelho - o código produziu um resultado muito diferente do esperado;
- Azul - este teste não foi efetuado.

Com base nesta nova informação, o professor poderá facilmente identificar os exercícios que não estão a produzir o resultado esperado, investigar as causas para tal e caso entenda necessário adotar alguma medida.

Ao clicar num determinado exercício, para além de toda a informação que já era apresentada, passou igualmente a ser incluída alguma informação sobre os resultados da execução do código com os *inputs* de teste. Na Figura 81, é possível observar um exemplo destes detalhes.

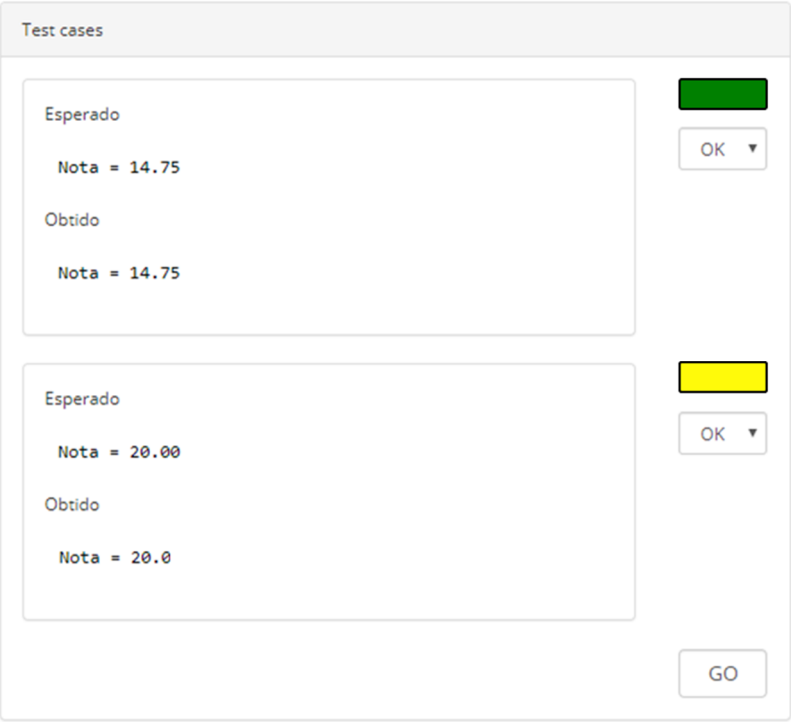

Figura 81 - Informação sobre resultados da execução do código com casos de teste - iteração quatro.

Como foi dito anteriormente, pode-se dar o caso de um exercício ser assinalado como parcialmente correto. Nesta situação o professor poderá analisar o resultado e caso considere a solução correta pode proceder manualmente à alteração do resultado para "OK". Como pode ser observado no exemplo contido na Figura 81, para o segundo caso de

teste, o resultado obtido não é exatamente igual ao que havia sido indicado como desejado. Perante uma situação destas, após análise do código fonte o professor poderá optar por assinalar esta solução como estando totalmente correta ou eventualmente chamar a atenção do estudante para a diferença verificada.

4.5.3.4.2. Matriz resumo

Com a introdução do mecanismo de execução do código com casos de teste, considerou-se que era pertinente introduzir algumas modificações numa das funcionalidades mais utilizadas pelos professores - a "matriz resumo". Assim, deixou de ser uma matriz única e passou a existir uma separação entre os exercícios de cada ficha de modo a conseguir mais facilmente perceber a informação. Contudo, a modificação mais importante consistiu da introdução de informação adicional relativa ao resultado da execução do código com casos de teste diretamente nas células da matriz. Convirá relembrar que, até então, as células poderiam ser de quatro cores: cinzento no caso de o estudante não ter tentado resolver um exercício; verde no caso de ter tentado e a última tentativa não apresentar erros de compilação; vermelho para o caso de a última tentativa ter erros de compilação/interpretação e amarelo no caso de a última tentativa ter erros de *runtime*. Um exemplo na nova configuração da matriz pode ser observado na Figura 82.

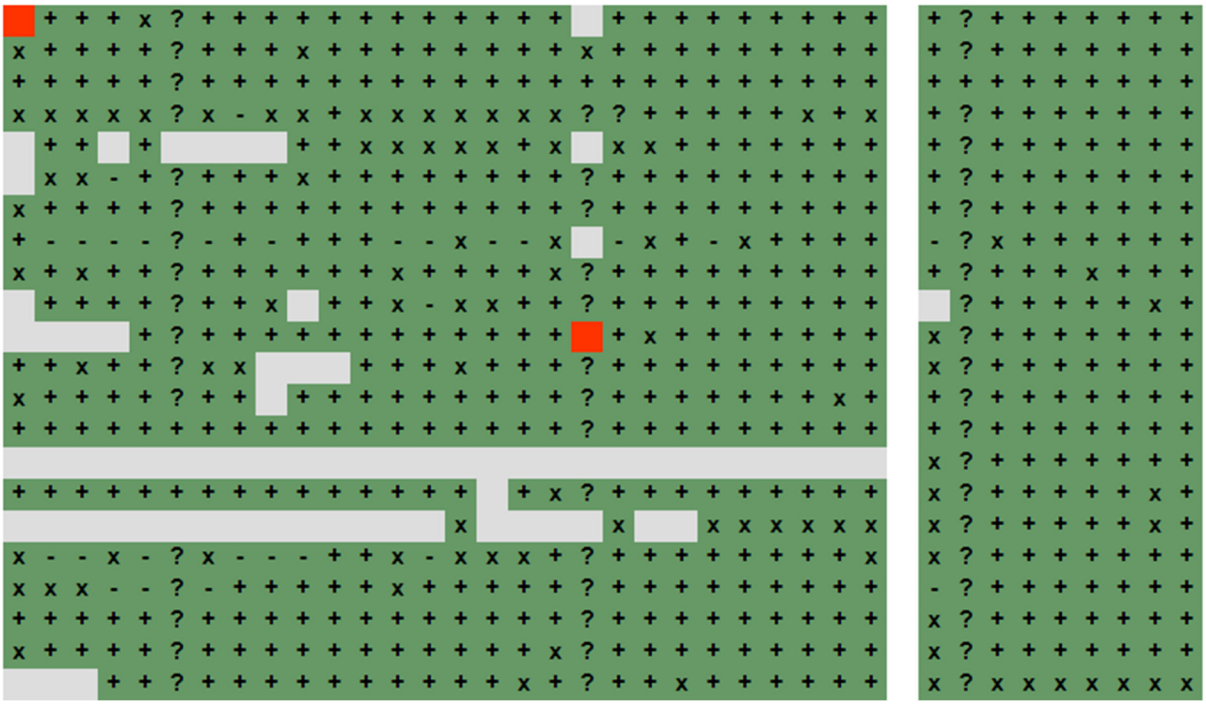

Figura 82 - Exemplo da configuração da "matriz resumo" - iteração quatro.

A modificação introduzida consiste na adição de símbolos no caso das células verdes, ou seja, aquelas em que foi possível executar o código com os casos de teste. Assim, é utilizado o símbolo "+", no caso de o resultado ser igual ao esperado, o símbolo "-", no caso de o resultado estar parcialmente correto, o símbolo "x" no caso de o resultado ser claramente distinto do esperado e, finalmente, o símbolo "?" para o caso de exercícios em que não foi possível determinar a validade do resultado (exercícios não determinísticos, por exemplo). Desta forma, o professor poderia de uma forma simples saber não só quem tentou resolver os exercícios, mas saber também qual o resultado obtido.

4.5.3.4.3. Execução de *Java Applets*

Uma vez que no caso da experiência do programa "Acertar o Rumo" eram utilizadas *Java Applets* para as quais o *output* produzido consiste em elementos visuais, foi desenvolvida uma funcionalidade que permitia aos professores a execução das *Applets* diretamente a partir do sistema, desde que utilizassem um navegador *web* com suporte para esta tecnologia. Na Figura 83, pode ser observado um exemplo da execução de uma *Applet*.

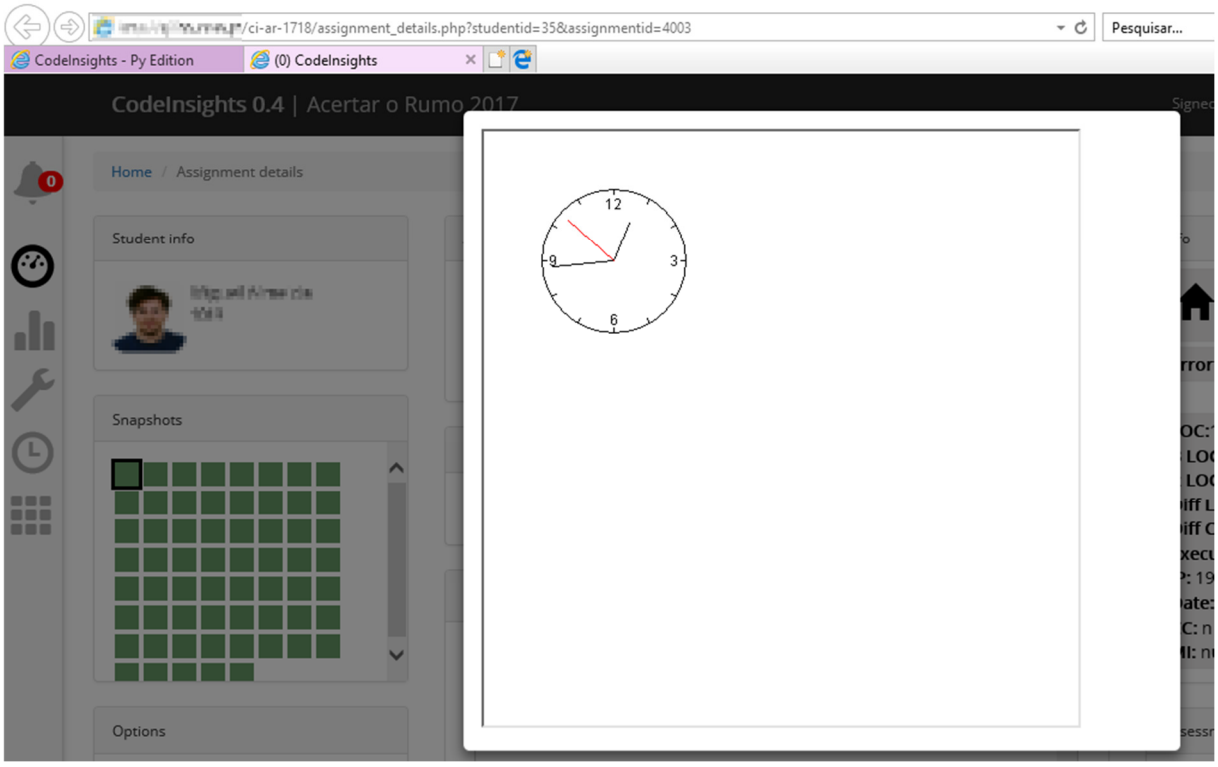

Figura 83 - Exemplo da execução de uma *Applet* no sistema *CodeInsights*.

Uma funcionalidade semelhante foi igualmente desenvolvida para o caso do código HTML, uma vez que também neste caso o resultado da interpretação do código é marcadamente visual.

4.5.3.4.4. *Live-feed*

Uma vez que toda a informação era recebida e processada em tempo real, considerou-se interessante desenvolver uma forma de fornecer ao professor a informação sobre o que estava a acontecer em "tempo real". Para tal, foi desenvolvida uma funcionalidade na qual se resume a atividade dos estudantes em intervalos de 15 segundos. Um exemplo desta *live feed*, pode ser observada na Figura 84.

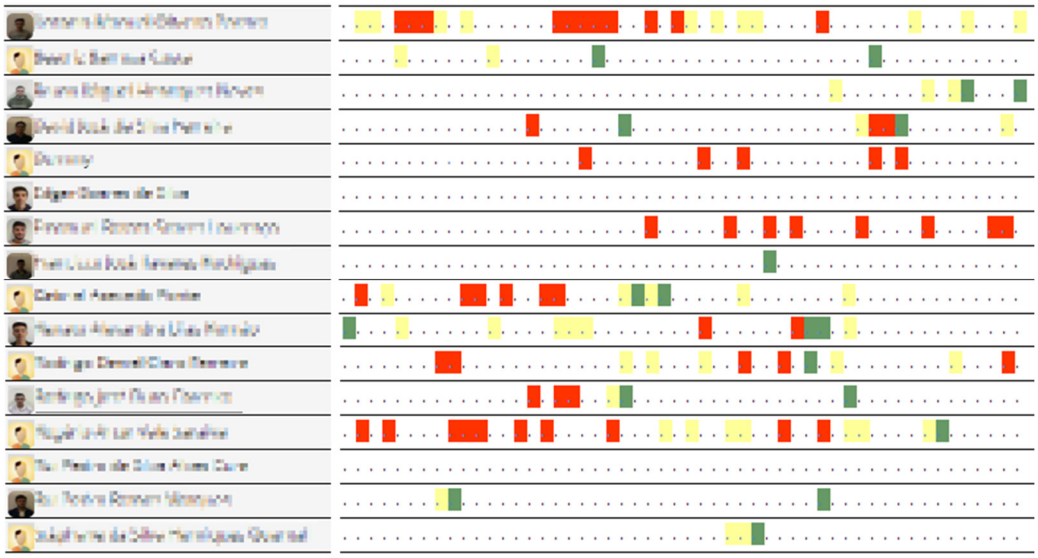

Figura 84 - Exemplo da *live feed*.

Nesta visualização, cada linha corresponde a um estudante e cada "retângulo" diz respeito ao estado do último *snapshot* recebido por cada estudante nos últimos 15 segundos. À medida que o tempo passa, novas colunas vão sendo acrescentadas. A cor dos retângulos tem os seguintes significados:

- Branca - Não foi recebido nenhum *snapshot* deste estudante nos últimos 15 segundos;
- Vermelha - O último *snapshot* recebido apresentava erros de compilação;
- Verde - O último *snapshot* recebido não apresentava erros e o resultado era igual ao esperado;
- Azul - O último *snapshot* recebido não apresentava erros, mas o resultado não era igual ao esperado;
- Amarelo - O último *snapshot* recebido não apresentava erros, mas o resultado não era totalmente igual ao esperado (poderia conter erros de formato, por exemplo);
- Roxo - O último *snapshot* recebido não apresentava erros, mas foi considerado inválido (poderia conter funcionalidades não autorizadas, por exemplo);

Recorrendo a esta funcionalidade, o professor poderá ter uma visão de como estava a correr o trabalho dos seus estudantes em tempo real, podendo a qualquer momento clicar em qualquer um dos retângulos coloridos por forma a obter mais informações e, se considerasse necessário, intervir junto do estudante.

4.5.3.4.5. Informação sobre o desempenho dos estudantes na listagem

A introdução do mecanismo de testes automatizados, acabou por permitir incluir de forma muito simples informação sobre o progresso dos estudantes diretamente na listagem de estudantes. Assim, ao invés de ser apresentada apenas uma listagem de nomes e números de estudante, passou igualmente a ser possível ter uma noção do nível de desempenho em que cada estudante se encontrava, determinado com base no número de exercícios que haviam sido propostos até ao momento e o número de exercícios classificados como corretos. Um exemplo na nova configuração da listagem pode ser observado na Figura 85.

Figura 85 - Exemplo de listagem de estudantes com informação sobre o seu desempenho - iteração quatro.

Para efeitos de construção desta listagem, foram considerados cinco níveis de desempenho, determinados em função da percentagem de exercícios resolvidos com sucesso. Ao nível mais baixo entre 0% e 20% corresponde a cor vermelha, ao nível seguinte entre 21% e 40% corresponde a cor laranja, e assim sucessivamente. Deste modo, é dada mais uma ferramenta aos professores para conseguirem identificar os estudantes com mais dificuldades em resolver os exercícios propostos.

4.5.3.5. Ferramenta de monitorização - estudantes

Como já foi referido, na iteração anterior alguns estudantes manifestaram o interesse em ter acesso a uma ferramenta na qual pudessem ter informação da lista dos exercícios para os quais já realizaram tentativas de resolução. Desde o início, os objetivos deste trabalho passaram pelo desenvolvimento de ferramentas para serem disponibilizadas aos professores. Contudo, foi considerado que de facto esta ferramenta destinada aos

estudantes poderia fazer sentido. Alguns professores disseram que, por vezes, os estudantes lhes pediam para ver se já tinham sido recebidas tentativas de resolução para um determinado exercício. Ao ser-lhes disponibilizada esta ferramenta, poderiam ser eles próprios a fazer essa verificação.

Após algum debate sobre a quantidade de informação a disponibilizar aos estudantes nesta aplicação, acabou por ser decidido incluir a lista dos exercícios para os quais foram recebidos *snapshots*. Esta listagem é em tudo semelhante àquela a que o professor tem acesso e que pode ser observada na Figura 80.

Tal como no caso dos professores, os estudantes também poderiam aceder a informação adicional relativa a cada exercício. Para além do elemento identificativo, do código fonte correspondente à última tentativa recebida, os estudantes tinham acesso ao *output* produzido pelo código tal como foi escrito e informação sobre os resultados da execução do código com os *inputs* de teste. Na Figura 86, é possível observar um exemplo da informação apresentada aos estudantes relativamente aos resultados dos casos de teste.

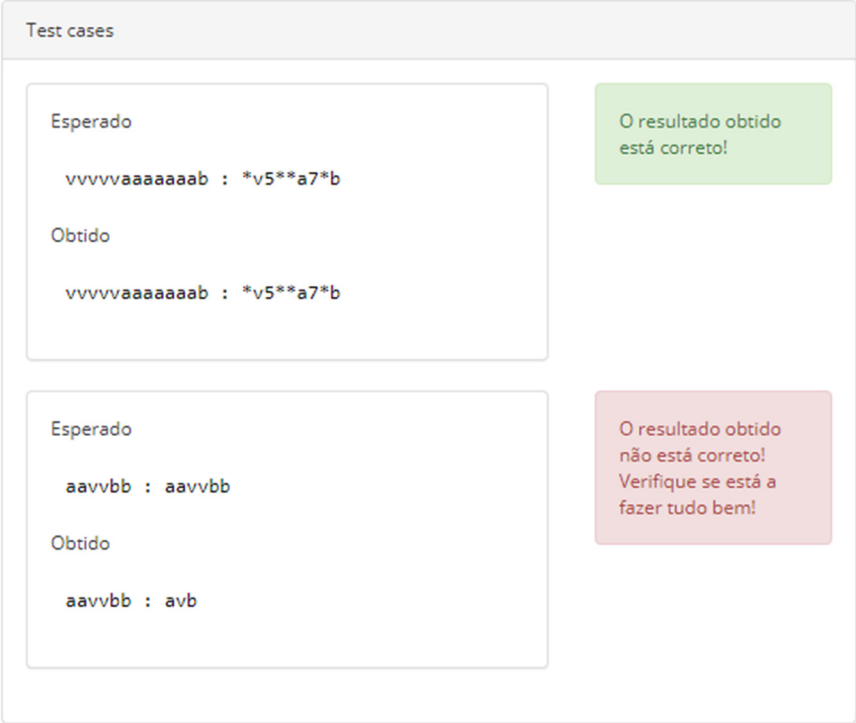

Figura 86 - Detalhes sobre resultados da execução do código com casos de teste - iteração quatro.

Com base na informação apresentada, o estudante podia verificar que o seu código não estava a produzir os resultados esperados em todas as situações, podendo haver a necessidade de proceder a correções.

4.5.3.6. Sistema de notificações automáticas

As várias visualizações disponibilizadas constituem por si só uma importante ferramenta para os professores. Contudo, como já foi mencionado anteriormente, foi considerado importante que o sistema possua capacidades de análise automática da informação recolhida, permitindo deste modo chamar a atenção do professor através de notificações em relação a certos aspetos considerados relevantes e que eventualmente lhe possam ter passado despercebidos.

Para este efeito, quer na iteração anterior, quer nesta iteração, foi utilizado um sistema de notificações automáticas muito simples que embora se tivesse revelado útil, poderia certamente ser melhorado. Este incidia sobre as diferenças de ritmo de trabalho e mudanças de performance dos estudantes. Desde então, foram surgindo ideias e pedidos por parte dos professores para outros tipos de notificações, pelo que, tal como já havia sido mencionado, o desenvolvimento de um sistema mais abrangente de geração de notificações acabaria por ser um dos focos principais desta iteração.

4.5.3.6.1. Tipos de notificações

Como já foi mencionado anteriormente, existem dois tipos principais de notificações: *online* e *off-line*, tendo já sido desenvolvidas duas notificações distintas do tipo *online*. O passo seguinte passou pela identificação de outros aspetos sobre os quais os professores gostariam de ser notificados. Este processo contou com a participação de cinco professores que já haviam utilizado o sistema e de outros professores que nunca haviam utilizado o sistema, mas que gostariam de o vir a fazer.

Assim, após algum tempo de recolha e maturação sobre os diversos aspetos que poderiam levar à geração de notificações, chegou-se a uma lista com cerca de duas dezenas de notificações. Em seguida apresentam-se alguns exemplos de notificações identificadas:

- Número invulgar de tentativas de resolução de um exercício por um estudante;
- Número invulgar de erros de compilação num exercício por um estudante;
- Número invulgar de linhas de código para a resolução de um exercício por um estudante;
- Número invulgar de exercícios resolvidos num determinado intervalo por um estudante;
- Número invulgar de dias sem qualquer tentativa de resolução por um estudante;
- Número invulgar de estudantes a tentar resolver um determinado exercício;

4.5.3.6.2. Granularidade das notificações

Como é possível observar, existem diferentes níveis de granularidade para as notificações. Com efeito, algumas destas notificações dizem respeito ao desempenho de um estudante em específico aquando da resolução de um exercício em concreto, ao passo que outras notificações dizem respeito ao desempenho global de um estudante. Podem ainda considerar-se as mesmas notificações ao nível de uma turma ou para a globalidade dos estudantes.

A título de exemplo, o professor poderá estar interessado em receber notificações caso o número de tentativas de resolução do exercício 1 seja invulgar (superior a um certo limiar) apenas em relação ao estudante A, mas pode estar interessado em receber notificações sobre a existência de estudantes da turma 1 que não tenham feito qualquer tentativa de resolução para o exercício 2. Poderia ainda pretender receber notificações sobre o facto de um estudante B não ter tentado resolver nenhum exercício de uma determinada ficha prática.

Perante esta grande variedade de possíveis notificações, um aspeto que rapidamente foi considerado prendeu-se com o facto de poder não fazer sentido utilizar todos os tipos de notificações em simultâneo. Caso tal acontecesse o sistema poderia facilmente conduzir a situações de *overload* de informação (notificações) que poderiam ser contraproducentes. Assim, achou-se importante que fosse o professor a decidir quais os aspetos sobre os quais gostaria de ser notificado. No sistema idealizado e até certa medida implementado, a geração (ou não) de notificações baseia-se na confrontação de um número de referência (a expectativa do professor) com um número medido (a observação).

Este processo de confrontação entre um valor expectável e um valor observado, encontra-se enquadrado no conceito de "atenção seletiva". Tendo em consideração a importância deste conceito para o sistema de notificações que se pretendia desenvolver, apresentam-se em seguida algumas noções base sobre a atenção seletiva consideradas necessárias para compreender a sua inclusão neste trabalho.

4.5.3.6.3. Algumas noções sobre atenção seletiva

De acordo com Jon Driver [127], o que vemos, ouvimos, sentimos e recordamos depende não só da informação recolhida pelos nossos sentidos, mas também dos aspetos dessa informação a que se decide prestar atenção. De facto, os seres humanos não são capazes de processar toda a informação que é recolhida pelos sentidos. A título de exemplo, é frequente alguém estar num determinado lugar e não reparar em algo ou alguém porque a "mente" estava concentrada em alguma outra coisa.

Nas ciências cognitivas, atenção seletiva é o termo utilizado para definir o mecanismo anteriormente descrito: escolher ou capturar um aspeto específico em detrimento de outro(s). Segundo Kahneman [128] a atenção seletiva é a habilidade que o indivíduo possui para direcionar o foco de atenção a um ponto específico no meio ambiente.

A atenção seletiva tem vindo a ser um tópico central da psicologia cognitiva desde as primeiras décadas do século XX [129]. Posteriormente, acabaria por despertar também o interesse das comunidades científicas ligadas às neurociências cognitivas e biológicas [130] e mais recentemente, à comunidade de Inteligência Artificial e *Machine Learning*.

Por volta da década de 50 do século passado, o principal foco da pesquisa foi o fenómeno conhecido como *cocktail party effect* [131]. Em traços gerais, este fenómeno está relacionado com a capacidade que os seres humanos têm de se concentrar, obter e fornecer informação numa sala cheia de pessoas a falar umas com as outras em simultâneo e em pequenos grupos, produzindo no seu conjunto uma grande quantidade de ruído.

Uma das principais contribuições em relação a este fenómeno é a teoria do **filtro de Broadbent** [132] - um dos primeiros investigadores a relacionar este fenómeno psicológico com conceitos de processamento de informação provenientes das ciências da computação e matemáticas.

Um aspeto importante da atenção seletiva é o facto de ser um processo de natureza diversa em função do mecanismo sensorial ativado (e.g. auditivo, visual, etc.). Deste modo, alguns autores focaram o seu trabalho de investigação em atenção seletiva para informação auditiva audível [133] [134]. Nos anos que se seguiram, a investigação começou igualmente a focar-se em atenção seletiva para informação visual.

A ideia principal por detrás da atenção seletiva para informação visual é basicamente a mesma: estudar como e porque se decide prestar atenção a um determinado item de informação visual e não a outro. Em 2001 Jon Driver [127] apresentou um estudo exaustivo sobre a evolução dos mecanismos de atenção seletiva quer para informação audível, quer para informação visual.

Algum trabalho foi já realizado no sentido de desenvolver mecanismos artificiais de atenção seletiva. Carota et al. [135], por exemplo, trabalharam no desenvolvimento de mecanismos de atenção seletiva utilizando *hardware* "pesado" de modo a conseguirem determinar automaticamente se um determinado item é ou não interessante.

Com uma abordagem consideravelmente diferente, Rucci et al. [136] trabalharam no desenvolvimento de mecanismos de atenção seletiva artificial para informação visual e táctil para robôs.

Mais recentemente, mecanismos de atenção seletiva artificial ganharam grande destaque no seio dos sistemas inteligentes de transportes, nomeadamente no que diz respeito à deteção de objetos/pessoas inesperados [137].

Nas últimas décadas, surgiram alguns modelos computacionais de emoções, resultado da investigação das ciências cognitivas. Uma dessas emoções é a **surpresa**, que segundo vários autores [138] está intimamente ligada a vários processos cognitivos como a aprendizagem e a atenção seletiva. A grande maioria dos modelos computacionais de surpresa desenvolvidos, como por exemplo os modelos de Itti et al. [139] e Peters [140], destinam-se exclusivamente a ser utilizados para itens de informação visual.

De acordo com Itti et al., a surpresa apenas pode existir na presença de incerteza, e pode apenas ser definida de uma forma relativa e subjetiva, sendo fortemente dependente das expectativas do observador. Estes autores desenvolveram um mecanismo que permite calcular a surpresa imediata sentida pelos seres humanos enquanto assistem a *streams* de vídeo *frame* a *frame*. O modelo proposto pode ser utilizado para localizar visualmente em tempo real quais as partes da imagem mais "interessantes".

Segundo Peters, existem ligações muito complexas entre a inteligência e a criatividade, desempenhando a surpresa um papel importante quer na génese quer na avaliação de criatividade. Peters defende ainda que a surpresa está também intimamente relacionada com o processo de perceção e, como tal, todas as máquinas inteligentes devem ser sensíveis à surpresa. O trabalho de investigação de Peters está especialmente centrado na implementação de mecanismos de surpresa em robôs de modo a permitir que estes tenham a capacidade de identificar o que há de novo e interessante ao seu redor.

Meyer et al. [138] defendem que a surpresa tem duas funções principais: informativa e motivacional. Em conjunto, estas funções promovem ações imediatas face ao evento surpreendente e ajudam na predição e controlo de futuras ocorrências desse evento.

Segundo Ortony e Partridge [141], os sistemas inteligentes lidam com grandes quantidades de informação que nem sempre é relevante para os problemas que têm em mão numa determinada altura. De acordo com estes autores, a integração de sistemas artificiais de atenção seletiva é essencial para conseguir lidar com a enorme quantidade de informação existente associada às limitadas capacidades de processamento em tempo

real dessa mesma informação. Sugerem que a atenção seletiva está intimamente ligada com a geração de expectativa e falha, isto é, com surpresa, pelo que os componentes de atenção seletiva necessários deverão assentar em modelos cognitivos de surpresa.

Lorini et al. [142], também apresentarem o seu modelo conceptual e formal da noção de surpresa. Segundo eles, a surpresa pode ser definida através de um modelo formal utilizando uma lógica de crenças probabilisticamente quantificáveis.

Macedo [143] desenvolveu um vasto trabalho à volta do conceito de atenção seletiva artificial baseada em sentimentos cognitivos, como a surpresa ou a incerteza, e sentimentos afetivos. Tendo por base o resultado de diversas experiências psicológicas, o seu trabalho baseia-se na assunção de que a surpresa "sentida" por um agente quando confrontado com um determinado evento/objeto **X** é proporcional ao grau de falta de expectativa desse mesmo objeto/evento ser o escolhido/acontecer. Ou seja, quanto maior é a expectativa de que um evento vai ocorrer, maior será a surpresa se este evento não ocorrer e menor será a surpresa se ele de facto ocorrer. Assim, se um determinado evento não for considerado surpreendente, não vale a pena "prestar-lhe atenção".

4.5.3.6.4. O modelo de atenção seletiva artificial de Macedo

Perante o exposto anteriormente sobre o processo de atenção seletiva, foi com grande naturalidade que foi decido implementar um mecanismo de geração de notificações mais abrangente e suportado por um mecanismo de atenção seletiva. Devido à sua simplicidade quer ao nível da implementação, quer ao nível conceptual, foi decidido implementar um mecanismo de geração de notificações inspirado no mecanismo de atenção seletiva artificial sugerido por Macedo. Na sua forma mais abrangente, este modelo de atenção seletiva artificial é composto por três componentes diferentes: *Surprise Value of Information*, *Uncertainty-based Value of Information* e *Motive Congruence/Incongruence-based Value of Information*.

De acordo com o **Princípio da Atenção Seletiva** proposto pelos autores: "Um agente racional com recursos limitados deve focar sua atenção apenas na informação relevante e interessante". Para tal, são utilizados três números reais $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ e $\boldsymbol{\gamma}$ como limiares acima dos quais os valores absolutos de *Motive Congruence* (MC), *Surprise* (S) ou *Information-Gain* (IG), respetivamente, permitem determinar que um determinado evento ou informação possa ser considerado valioso ou interessante. Na Figura 87, está representado um algoritmo que permite expressar de forma mais concisa o que acabou de ser dito.

```
SE(MC > α E(S > β OU IG > γ):
   Informação relevante
SE NÃO:
   Informação não relevante
```

Figura 87 - Algoritmo geral do modelo de atenção seletiva artificial proposto por Macedo.

Segundo Macedo, não existe propriamente uma forma direta para definir os valores para os limiares $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ e $\boldsymbol{\gamma}$. Com efeito, estes podem ser definidos empiricamente com base em experiências anteriores, por exemplo, ou simplesmente com base no conhecimento/experiência de especialistas do domínio em causa.

Determinação do valor de IG

O componente *Uncertainty-based Value of Information* consiste basicamente na determinação do valor do ganho de informação (*information gain - IG*) provocado pela observação de nova informação numa determinada experiência. Para a determinação deste valor não importa saber qual o valor que efetivamente foi observado, interessa apenas saber que ouve uma observação. Assim o valor de IG pode ser determinado utilizando a expressão (6) e, consequentemente, a expressão (7).

$$IG(E) = \frac{H_{prior}(E) - H_{post}(E)}{\log(m)} \tag{6}$$

$$IG(E) = \frac{-\sum_{i=}^{m} P_{prior}(E_i) \times (\log P_{prior}(E_i)) - (-\sum_{i=}^{m} P_{post}(E_i) \times (\log(P_{post}(E_i)))}{\log(m)g} \tag{7}$$

Nesta expressão, m reporta-se ao número de valores distintos e mutuamente exclusivos que é possível serem observados aquando de uma determinada observação/experiência E. Os valores de H_{prior} e H_{post} correspondem aos valores de entropia verificados antes (*prior*) e depois (*post*) da aquisição de nova informação que pode alterar a distribuição das probabilidades de ocorrência de um determinado evento.

Caso se esteja perante observações finais, isto é, caso se tenha a certeza sobre o resultado de uma experiência, o valor de H_{post} é igual a zero, uma vez que a probabilidade de algo diferente do que foi observado será igual a zero e a probabilidade do que foi observado é 1. Utilizando um exemplo concreto, assumindo que se pretende verificar quantos estudantes entregaram um trabalho prático numa determinada data (*hard deadline*). Não faz sentido prever os que possam ainda vir a entregar depois, uma vez que a

observação era feita única e exclusivamente num determinado momento, não havendo logo à partida possibilidade de existirem eventos/observações a seguir a este momento. Este cenário aplica-se a todas as notificações do tipo *off-line*. Em todo o caso, para estes cenários, o valor de IG pode ser determinado simplesmente pela aplicação da expressão (8).

$$IG(E) = \frac{-\sum_{i=}^{m} P_{prior}(E_i) \times (\log (P_{prior}(E_i))}{\log (m)} \quad (8)$$

No caso de observações de eventos inacabados, isto é, que ainda estejam a decorrer, apenas é possível determinar o valor de *IG* caso exista algum tipo de modelo probabilístico de ocorrência do evento, ou seja, uma forma de determinar a probabilidade dos eventos que podem acontecer após a ocorrência de um determinado evento. Concretizando, no caso da geração de notificações online relativas ao número de tentativas de resolução, por exemplo, não há um modelo probabilístico que nos permita dizer se uma tentativa será a última ou não, assim como quantas ainda se seguirão. Uma vez que não se dispunham efetivamente de modelos probabilísticos que permitissem suportar este processo, nestas situações acabou por se considerar que, de facto, se tratava de eventos acabados e como tal foi considerado H_{post} igual a zero.

Assumindo que um determinado professor conseguiu determinar um conjunto de valores de probabilidades relativas à percentagem de estudantes da turma que vai fazer submissões para dois exercícios E1 e E2 dentro do prazo (eventos finalizados). Estas probabilidades de ocorrência podem ser observadas na Tabela 5 e Tabela 6.

Tabela 5 - Tabela de probabilidades associadas ao exercício E1.

E1		
evento	% de estudantes	probabilidade
A	<= 20 %	0.10
B	]21-79] %	0.75
C	>= 80 %	0.15

Tabela 6 - Tabela de probabilidades associadas ao exercício E2.

E2		
evento	% de estudantes	Probabilidade
a	<= 20 %	0.60
b	]21-79] %	0.25
c	>= 80 %	0.15

Aplicando a expressão 8 ao caso do exercício E1, obtém-se:

$$IG(E_1) = \frac{-(0.10 \times \log(0.10) + 0.75 \times \log(0.75) + 0.15 \times \log(0.15))}{\log(3)} \quad (9)$$

$$IG(E_1) = 0.665$$

Fazendo o mesmo em relação ao exercício E2, obter-se-ia:

$$IG(E_2) = \frac{-(0.60 \times \log(0.60) + 0.15 \times \log(0.15) + 0.25 \times \log(0.25))}{\log(3)} \quad (10)$$

$$IG(E_2) = 0.853$$

Tendo como ponto de partida os exemplos anteriores, no caso do exercício E1, existe um evento com uma probabilidade de ocorrência muito superior à soma da probabilidade de ocorrência dos restantes eventos. Assim, perante a ocorrência de um novo evento, qualquer que este seja, não existirá um substancial ganho de informação (isto é, caso as probabilidades fossem atualizadas, a diferença entre o evento mais provável e a soma das probabilidades dos restantes eventos continuaria a ser elevada), nesta medida, o valor de *IG* é relativamente baixo. Já no caso de E2, a diferença entre a probabilidade de ocorrência do evento mais provável e a soma das probabilidades de ocorrência dos restantes eventos é consideravelmente menor, isto é, existe maior incerteza sobre o possível resultado, deste modo, o valor de *IG* é superior. Por outras palavras, quanto mais a probabilidade estiver distribuída, ou mais se aproximar da igualmente distribuída, maior a incerteza.

Como foi mencionado anteriormente, um fator importante para o eficaz funcionamento deste mecanismo passa pelo processo de definição dos valores dos limiares. No caso concreto do limiar $\boldsymbol{\gamma}$, a utilização de um valor muito elevado, terá o significado que o professor apenas está interessado em receber notificações sobre eventos em que exista um ganho de informação considerável, ou seja, casos em que não exista à partida um evento com probabilidade muito maior de ocorrer do que os restantes - casos em que haja incerteza. Assim, por exemplo, perante a utilização de $\boldsymbol{\gamma} = 0.75$, independentemente do que ocorresse, no caso de E1 ($IG = 0.665$), era considerado que não havia ganho de informação substancial, já relativamente a E2 ($IG = 0.853$), independentemente do que ocorresse, era considerado que essa observação acarretaria um ganho de informação considerável. A utilização de diferentes valores para os limiares iria resultar em conclusões diferentes.

Determinação do valor de S (surpresa)

Foi apresentado anteriormente como proceder para calcular o valor do ganho de informação (*IG*) provocado pela ocorrência de um determinado evento. Irá agora ser apresentado como calcular o valor da surpresa (*S*) sentida pelo professor aquando da ocorrência de um determinado evento. Ao contrário do que acontecia anteriormente, em que era irrelevante saber o que foi de facto observado, no caso do cálculo do valor de surpresa é imperativo saber qual o valor observado, por forma a poder ser confrontado com as expectativas que o professor tinha. Com efeito, segundo Macedo, a intensidade da surpresa sentida por um agente (professor, no neste caso) aquando da ocorrência de um evento $\boldsymbol{E_g}$ (entre um grupo de eventos possíveis e mutuamente exclusivos $\boldsymbol{E_1, E_2, \ldots, E_m}$) pode ser definido em função da diferença (ou contraste) entre a probabilidade de ocorrência desse evento $\boldsymbol{E_g}$ e a probabilidade de ocorrência do evento mais provável $\boldsymbol{E_h}$. Este processo está resumido, na sua forma não linear, na expressão 11.

$$S(E_g) = log(1 + P(E_h) - P(E_g)) \tag{11}$$

Está a ser assumido que existe sempre um evento com mais probabilidade de ocorrência do que os restantes. Se tal não acontecer, isto é, se a probabilidade de ocorrência dos vários eventos for igual, então o valor de surpresa será igual a zero.

Neste contexto, as probabilidades de ocorrência de um determinado valor observado (evento), assumem o nome de expectativas. Assim, com base nas expectativas definidas anteriormente para o exercício E1, **b** é a observação mais provável (P($\boldsymbol{E_h}$)=0.75).

Assumindo que no momento definido como *deadline* foi observado que apenas tinham feito submissões para este exercício 13% dos estudantes (ocorrência de *a*), ou seja, P($\boldsymbol{E_a}$) = 0.1, neste caso, a observação desta ocorrência teria um valor de surpresa de 0.72.

$$S(E_a) = log(1 + P(E_h) - P(E_a)) \tag{12}$$
$$S(E_a) = log(1 + 0.75 - 0.1)$$
$$S(E_a) = 0.72$$

Considerando agora que, no momento definido como deadline, foi observado que tinham feito submissões para este exercício 93% dos estudantes (ocorrência de c), ou seja, *P(Ec) = 0.15*, obter-se-ia um valor de surpresa de 0,67, isto é, inferior ao obtido anteriormente.

$$S(E_c) = log(1 + P(E_h) - P(E_g)) \tag{13}$$
$$S(E_c) = log(1 + 0.75 - 0.15)$$
$$S(E_c) = 0.67$$

Finalmente, perante a observação da realização de submissões de 53% (ocorrência de b, o evento mais expectável), obter-se-ia *P(Eb) = 0.75*, o que resultaria em $S = 0$ (14).

$$S(E_b) = log(1 + P(E_h) - P(E_g)) \quad (14)$$
$$S(E_b) = log(1 + 0.75 - 0.75)$$
$$S(E_b) = 0$$

Procedendo de igual forma relativamente ao exercício E2, obter-se-iam os seguintes valores de surpresa: *S(Ea)=0*, *S(Eb) = 0.43* e *S(Ec) = 0.53*.

De acordo com Macedo, um evento é considerado surpreendente, quando o valor de *S* é superior ao do limiar **β** previamente apresentado. Assumindo, por exemplo, a utilização de **β**=0,65, no caso do exercício E1, tanto a observação da ocorrência de ***a*** como a observação da ocorrência de ***c*** eram consideradas surpreendentes. Já no caso de E2, nenhuma observação era considerada suficientemente surpreendente.

Assim, perante valores pequenos para este limiar, mesmo eventos com um valor de S pequeno poderão ser considerados surpreendentes. Em oposição, utilizando valores muito elevados para o limiar terá como consequência que apenas eventos com um valor de S muito elevado serão considerados surpreendentes. Uma vez mais deverá ser dada a devida atenção ao processo de definição deste limiar.

Determinação do valor de MC (Motive-congruence or happiness)

Finalmente, o componente de ***Motive Congruence/Incongruence-based Value of Information*** é utilizado para associar diversos níveis de "desejo" para receber notificações sobre a ocorrência de um determinado evento (*i.e.,* um professor pode ter mais interesse em receber notificações sobre a ocorrência de um evento do que de outro).
Tomando como exemplo o exercício E1 anteriormente apresentado, o professor pode ter um elevado "desejo" em ser notificado caso a percentagem de estudantes a fazer submissões tenha sido muito baixa (ocorrência de **a**), mas pode não ter tanto "desejo" em ser notificado caso a percentagem tenha sido muito elevada (ocorrência de **c**), ainda que ambas possam ser surpreendentes. Para tal, o professor deverá expressar os seus "desejos" relativos à ocorrência de cada evento numa escala entre -1 e 1, como exemplificado na Tabela 7.

Tabela 7 - Tabela de níveis de "desejo" associados à ocorrência de determinados eventos.

E1		
evento	% de estudantes	Desejo
a	<= 20 %	0.9
b	]21-79] %	0.2
c	>= 80 %	0.3

A definição de valores de desejo negativos é utilizada para, de certo modo, representar um "desejo infeliz", ou seja, que deve ser visto como algo negativo (não desejado). Com base nesta tabela é possível determinar o valor de MC de um determinado evento *Ei*.

$$MC\ (Ei) = |Di| \qquad (15)$$

Também a respeito deste componente, a definição do valor do limiar α é muito importante. Valores baixos poderão levar à geração de notificações que o professor não deseja receber e, por oposição, valores altos poderão levar a que não sejam geradas notificações sobre a ocorrência de eventos que o professor desejava ser notificado.

Até agora foi apresentada a aplicação de cada componente isoladamente, no entanto, segundo este modelo de atenção seletiva artificial, uma informação apenas é considerada relevante/valiosa, caso obedeça aos critérios definidos pelo princípio da atenção seletiva, ou seja, caso $MC > \alpha$ e $(S > \beta$ ou $IG > \gamma)$. Convirá, a este respeito, clarificar o motivo da utilização do "ou" em $S > \beta$ ou $IG > \gamma$. Para tal, será pertinente recorrer à representação da evolução dos valores de S e IG em função da probabilidade de ocorrência de um de dois eventos (*e.g.* obter cara ou coroa no lançamento de uma moeda) que consta da Figura 88.

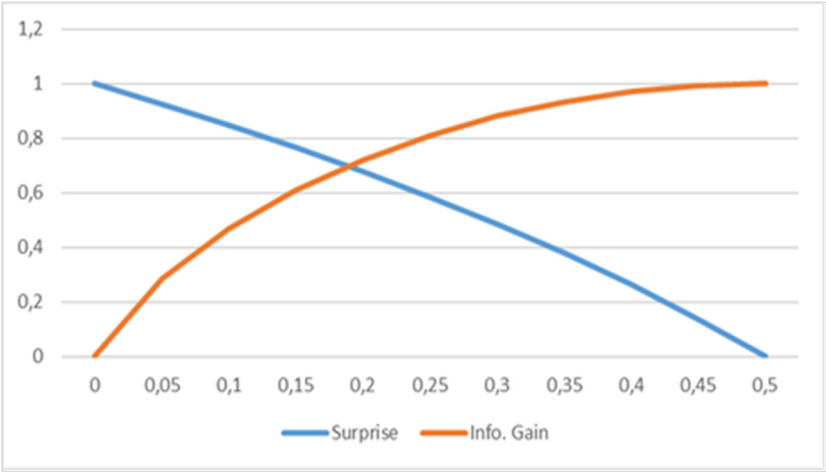

Figura 88 - Evolução dos valores de S e de IG.

Como pode ser observado na Figura 88, o valor de IG aumenta à medida que a probabilidade de ocorrência de um determinado evento também aumenta, o que significa que a ocorrência de um qualquer evento num cenário muito homogéneo irá provavelmente contribuir mais para reduzir a incerteza do que num cenário muito heterogéneo. Em contraste, é possível observar que o valor da surpresa diminui à medida que o valor de probabilidade de ocorrência desse evento aumenta, o que acaba por fazer todo o sentido, pois obviamente, o utilizador ficará menos surpreendidos quando ocorre algo que já se espera que ocorra. Assim sendo, o "ou" é utilizado para acomodar as situações em que a ocorrência de um evento possa não representar um elevado ganho de informação, mas que poderá ser surpreendente e vice-versa.

Apresentam-se em seguida alguns cenários de utilização deste processo com diferentes limiares.

Caso sejam utilizados limiares com valores muito elevados:

Evento que ocorreu	Probabilidade	IG	S	MC	γ	β	α	Gera notificação?
A	0.10	0.665	0.72	0.85	0.85	0.85	0.85	Não
B	0.75	0.665	0	0.40	0.85	0.85	0.85	Não
C	0.15	0.665	0.67	0.75	0.85	0.85	0.85	Não

Caso sejam utilizados limiares com valores muito baixos:

Evento que ocorreu	Probabilidade	IG	S	MC	γ	β	α	Gera notificação?
A	0.10	0.665	0.72	0.85	0.35	0.35	0.35	Sim
B	0.75	0.665	0	0.40	0.35	0.35	0.35	Sim
C	0.15	0.665	0.67	0.75	0.35	0.35	0.35	Sim

Caso sejam utilizados limiares com valores moderados:

Evento que ocorreu	Probabilidade	IG	S	MC	γ	β	α	Gera notificação?
A	0.10	0.665	0.72	0.85	0.70	0.65	0.80	Sim
B	0.75	0.665	0	0.10	0.70	0.65	0.80	Não
C	0.15	0.665	0.67	0.75	0.70	0.65	0.80	Não

Como seria expectável, os valores utilizados para os limiares acabam por ser preponderantes para determinar se uma notificação deve ser gerada ou não.

4.5.3.6.5. Integração do modelo de atenção seletiva no sistema *CodeInsights*.

De modo a operacionalizar o processo anteriormente apresentado, foi desenvolvido e disponibilizado aos professores um conjunto de possíveis notificações. Para poderem usufruir delas, os professores deveriam começar por aceder à ferramenta disponibilizada para o efeito e na qual podem aceder à listagem de registos de notificações, tal como apresentado na Figura 89.

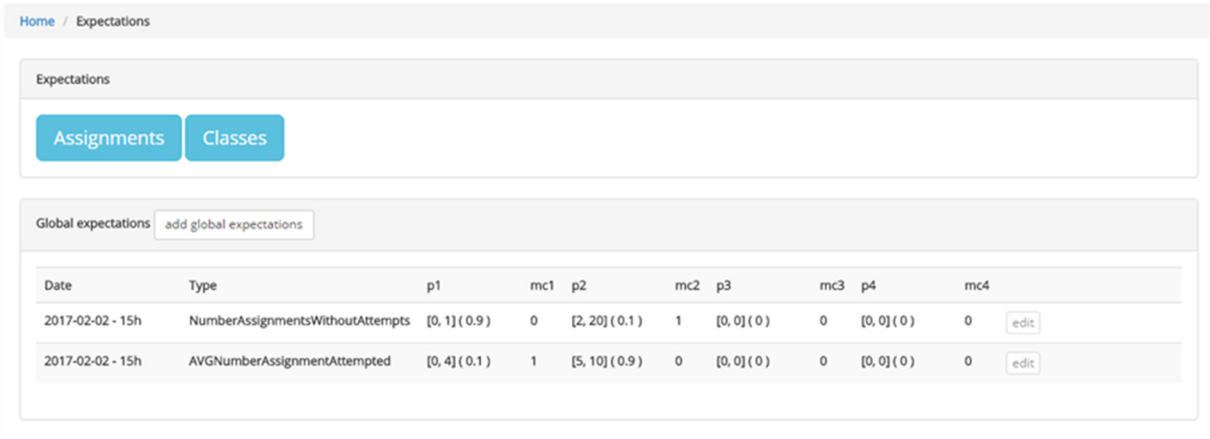

Figura 89 - Listagem de "registo" de geração de notificações.

Convirá relembrar que o sistema disponibiliza notificações do tipo *online* associadas a um determinado exercício e que poderão ser geradas (ou não) em tempo real aquando da receção dos *snapshots* (*e.g.* número invulgar de tentativas de resolução) e notificações do tipo *off-line* que serão (ou não) geradas numa determinada data e hora definida pelo professor. Em relação ao segundo tipo de notificações os professores poderão "registar" notificações associadas a eventos de natureza **global** (*e.g.* número de exercícios sem nenhuma tentativa de resolução), notificações associadas a um **exercício em particular** (*e.g.* número de estudantes que tentaram resolver o exercício *x*), notificações associadas a uma **determinada turma** (*e.g.* número médio de exercícios com pelo menos uma tentativa de resolução para os estudantes da turma) e finalmente notificações associadas a um **estudante em particular** (*e.g.* número final de tentativas de resolução feitas para um exercício).

Dependendo do tipo de notificação que se pretendia registar, ao criar um novo registo era necessário preencher um formulário que poderia solicitar informação distinta (*e.g.* o nome do estudante ou o exercício). No caso de notificações do tipo *off-line,* era necessário proceder à indicação da data e hora em que se pretendia que a verificação fosse feita (e eventualmente a notificação fosse gerada). Em qualquer caso, era sempre necessário

proceder à definição das expectativas e desejos associados à notificação que se estava a registar. Um exemplo deste formulário pode ser observado na Figura 90

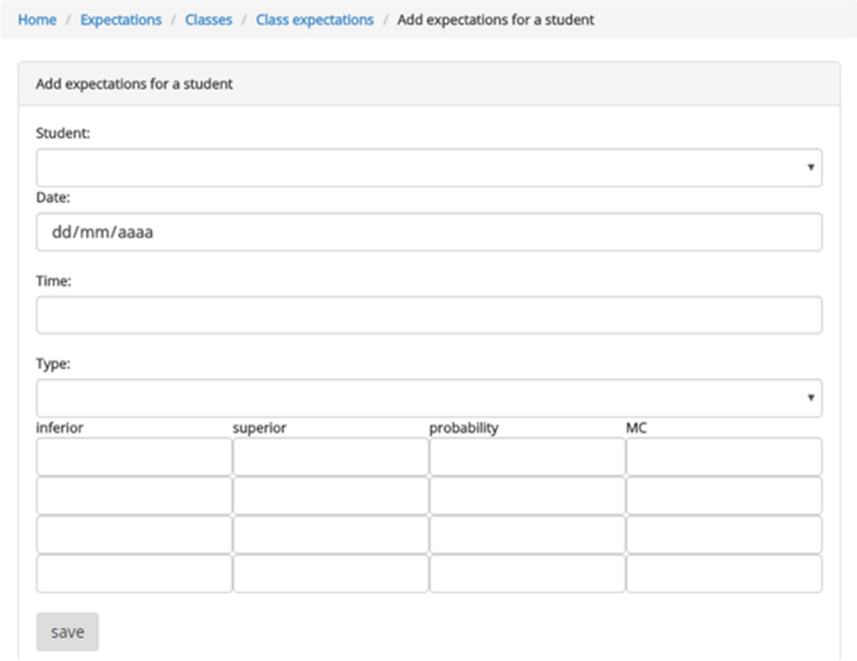

Figura 90 - Formulário de "registo" de notificação.

Como foi visto anteriormente, as expectativas não são mais do que a distribuição de probabilidades de ocorrência de vir a ocorrer um determinado evento. Assumindo que se pretende registar a geração de notificação relativa ao número de estudantes que tinham feito pelo menos uma tentativa de resolução para um determinado exercício. Com base em dados concretos de anos anteriores, ou simplesmente com base na sua experiência/intuição, o professor consegue exprimir as suas expectativas na forma da Tabela 8.

Tabela 8 - Exemplo de tabela de expectativas de um professor sobre a ocorrência de um determinado evento.

Número de estudantes	**probabilidade**	**D**
[0 , 5]	0.1	1
[6, 15]	0.75	0.5
[16, 20]	0.15	0.7

Ou seja, segundo o professor, existe uma probabilidade de 10% de o número de estudantes estar entre 0 e 5, 75% de o número de estudantes estar entre 6 e 10 e 15% de o número de estudantes estar entre 16 e 20. Para especificar estas expectativas o professor deverá preencher o formulário disponibilizado tal como representado na Figura 91.

inferior	superior	probability	MC
0	5	0.1	1
6	15	0.75	0.5
16	20	0.15	0.7

Figura 91 - Formulário de especificação de expectativas.

Por outras palavras, para cada intervalo, será necessário especificar o limite inferior e o limite superior seguido do valor de probabilidade e de desejo associado a esse intervalo.

No caso de uma notificação do tipo *off-line*, no dia e hora indicado é executada a função associada ao tipo de notificação registado, obtido o valor observado (o número de estudantes com submissões feitas, por exemplo) e em seguida levado a cabo o processo de aplicação do modelo de atenção seletiva anteriormente descrito e em função da sua aplicação gerada, ou não, uma notificação.

No caso de se tratar de notificação do tipo *online*, o processo é semelhante e totalmente automático, contudo em vez de ocorrer numa determinada data e hora pré-definida irá ocorrer sempre que é recebido um *snapshot* para o exercício associado à notificação.

4.5.4. Avaliação do sistema

Como mencionado anteriormente, nesta iteração, o sistema foi utilizado em quatro unidades curriculares distintas, apresentando-se em seguida os principais resultados destas avaliações.

4.5.4.1. Programa Acertar o Rumo (DEI - FCTUC)

No caso do Programa Acertar o Rumo, a avaliação do sistema ocorreu entre os dias 28 de setembro de 2017 e 16 de novembro de 2017. Devido à introdução do mecanismo de execução do código com casos de testes, foi necessário proceder a algumas adaptações aos enunciados das fichas práticas, por forma a incluírem informação sobre os nomes das variáveis de *input* assim como exemplos do formato dos *outputs* esperados.

Como seria de esperar, existiram algumas dificuldades iniciais associadas à forma de utilizar o mecanismo de definição das variáveis de *input*, as quais foram resolvidas rapidamente pela maioria dos estudantes. Existiram, no entanto, alguns estudantes que apresentaram efetivamente algumas dificuldades em declarar e inicializar as variáveis

dentro dos "limites" definidos. Na maior parte dos casos, tal ocorreu certamente por esquecimento, uma vez que estes estudantes tinham o cuidado de utilizar os nomes sugeridos para as variáveis. Em todo o caso, é um aspeto que há que ter em consideração em futuras iterações.

Participaram nesta experiência todos os 24 estudantes inscritos no curso, tendo sido recebidos aproximadamente 40.000 *snapshots* válidos referentes a cerca de 150 exercícios, organizados em sete fichas práticas.

O número de *snapshots* efetuados durante a duração da experiência foi certamente superior. No entanto, convirá mencionar que o servidor onde o sistema estava instalado se encontrava em Oliveira do Hospital, tendo existido no período da experiência, mais concretamente a 15 e 16 de outubro de 2017, um grande incêndio que destruiu o sistema de comunicações em toda a área envolvente. Esta ocorrência impossibilitou a receção e processamento dos *snapshots* recebidos durante o período necessário para proceder à instalação do sistema em outro servidor e disponibilização de uma nova versão do *plug-in* associado à nova localização do servidor. Em termos globais, este processo não demorou mais do que 48 horas, no entanto, houve de facto alguma perda de informação.

Ao contrário do que tinha acontecido no ano anterior, nesta edição desta unidade curricular, foi dado acesso ao sistema aos quatro professores associados a ela, permitindo deste modo disponibilizar-lhes informação sobre o trabalho desenvolvido pelos estudantes (e o estado atual de cada um deles), sem necessitar recorrer a outros instrumentos.

Tal como pôde ser observado na experiência do ano anterior, ainda que seja uma unidade curricular muito intensiva durante o período das aulas, a grande maioria dos estudantes continuava a trabalhar em casa. Com efeito, também nesta edição do Programa *Acertar o Rumo* existiu um elevado número de submissões efetuadas fora do período das aulas (aproximadamente 30% do total), contudo, graças à utilização do sistema ***CodeInsights***, os professores puderam de uma forma direta e simples saber exatamente quais os exercícios em que cada estudante se encontrava a trabalhar em cada momento, podendo desta forma preparar a aula seguinte de uma forma muito mais informada.

Pelos motivos expostos, este acabou por constituir o cenário ideal para testar as ferramentas dedicadas ao acompanhamento do progresso e performance dos estudantes, com especial enfoque na identificação dos estudantes com ritmos de trabalho muito

elevados e muito baixos. Na Figura 92, é possível observar a distribuição dos *snapshots* ao longo da duração da experiência.

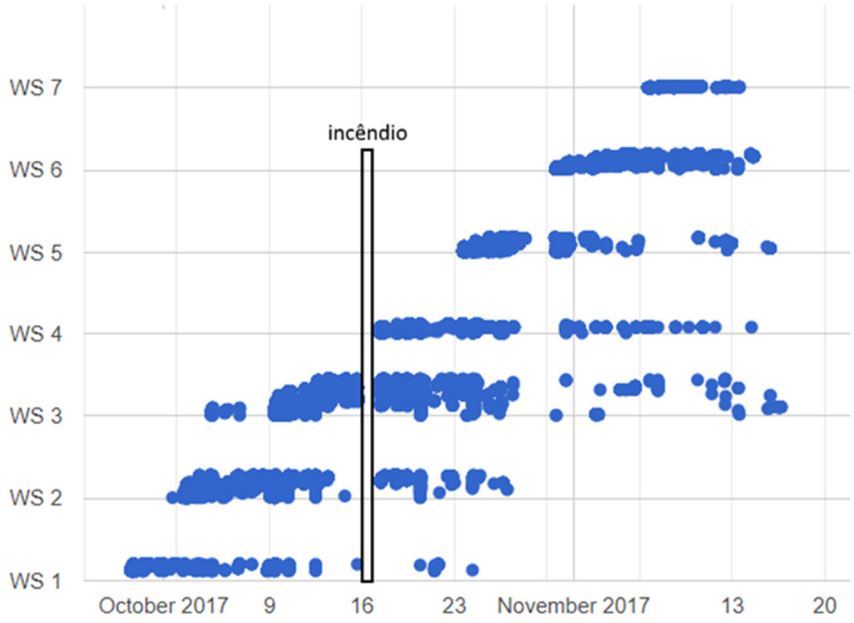

Figura 92 - Distribuição dos *snapshots* ao longo do tempo - Acertar o Rumo - iteração quatro.

Como já seria de esperar, existiram de facto estudantes a trabalhar com ritmos muito distintos. Enquanto alguns já tinham experiência sólida ao nível de programação, muitos outros teriam aqui o primeiro contacto com uma linguagem de programação.

Uma vez que os professores têm uma elevada proximidade com os estudantes, era de esperar que a informação sobre a diferença na sua performance não constituísse uma novidade. Ainda assim, permite-lhes obter essa informação de uma forma objetiva.

O sistema de geração de notificações foi testado nesta experiência relativamente a quatro notificações em concreto: (i) existência de estudantes a trabalhar com diferenças de performance substanciais face aos colegas; (ii) estudantes sem nenhuma submissão nos últimos três dias; (iii) número invulgarmente alto de submissões para o mesmo exercício; (iv) existência de exercícios sem qualquer tentativa de resolução num dado período.

Começando pelo primeiro tipo de notificações, ao longo da experiência foram sendo definidos objetivos para períodos de segunda-feira a quarta-feira e de quinta-feira a domingo, aos quais eram associados um conjunto de exercícios. Tendo em consideração o que foi dito anteriormente sobre o *background* dos estudantes não é de estranhar que em todos os períodos tenham sido geradas notificações deste tipo. Com o passar do tempo, a expectativa de existência de estudantes mais avançados ou atrasados foi crescendo, pois tornou-se óbvio que iriam existir sempre alguns estudantes muito avançados e alguns

muito mais atrasados, o que em concreto por vezes significava um valor de performance quase nula, uma vez que se encontravam a trabalhar em exercícios de fichas anteriores.

Relativamente às notificações sobre a existência de estudantes sem submissões nos últimos três dias, o sistema foi configurado para fazer esta verificação todos os dias às 23h55 com base em expectativas sob a forma de probabilidade de existência de estudantes nestas condições (evento A - não existência (90%); evento B - existência (10%)). Ao longo da experiência, foram geradas quatro notificações deste tipo, tendo sido possível verificar que em duas delas os estudantes estavam doentes, contudo nas restantes duas, veio a confirmar-se que os estudantes tinham desistido do curso.

Em relação às notificações sobre a existência de exercícios sem qualquer tentativa de resolução para o período esperado, a verificação era feita às 23h55 do último dia do período associado aos objetivos e foram sendo definidas expectativas para cada período sob a forma de probabilidade de existência de exercícios associados aos objetivos em vigor sem uma única tentativa de resolução (evento A - não existência (80%); evento B - existência (20%)).

Numa fase inicial, ainda se tentou utilizar valores diferentes de probabilidades dependendo da dificuldade do exercício, contudo rapidamente se chegou à conclusão de que não só esse processo era muito laborioso, como não existia nenhuma base suficientemente sólida para suportar os valores diferenciados. Em todo o caso, não foi gerada nenhuma notificação deste tipo, tendo existido sempre pelo menos um estudante a tentar cada um dos exercícios propostos dentro do prazo definido.

Finalmente, em relação às notificações relativas a estudantes com um número invulgar de tentativas de resolução para um determinado exercício, as expectativas eram definidas automaticamente aquando da criação do exercício com valores por omissão.

Caso assim o entendessem, os professores poderiam alterar as expectativas associadas a um determinado exercício de forma manual, algo que acabou por não acontecer. Em termos concretos, as expectativas utilizadas para este tipo de notificação foram as seguintes: evento A - número de tentativas <100 (95%); evento B - número de tentativas entre 101 e 150 (3%); evento C - número de tentativas entre 151 e 200 (2%) e finalmente, evento D - número de tentativas superior a 200 (1%). Os professores estavam apenas interessados em ser notificados sobre a ocorrência dos eventos B, C e D.

No total, foram geradas 62 notificações deste tipo, sendo que na maioria dos casos foram notificações correspondentes ao evento B (52). Contudo também existiram notificações sobre a ocorrência do evento C (6) e D (4).

Os valores utilizados para as expectativas apresentados foram definidos com base em valores médios de anos anteriores, mas também com base na perceção dos professores, não havendo, portanto, uma base sólida que os sustente e como tal, sem garantias de serem os mais adequados. Ainda que fosse possível definir as expectativas de forma manual e diferenciada para cada exercício (tendo em consideração a sua dificuldade, por exemplo), tal acabou por não acontecer e, deste modo, é de admitir, que em algumas situações possam não ter sido utilizadas as expectativas mais adequadas.

4.5.6.1.1. Entrevista com os professores

Por forma a averiguar de que modo os professores utilizaram o sistema no seu dia-a-dia e até que ponto este foi útil para o desempenho das suas tarefas, no final da experiência foi conduzida uma entrevista com todos os professores. A estrutura base desta entrevista está disponível no Anexo IV. Em seguida apresentam-se as perguntas efetuadas assim como um resumo das respostas dadas pelos professores desta experiência.

Com que regularidade utilizou o sistema *CodeInsights* ao longo da experiência?
Todos os professores responderam que tinham utilizado o sistema diversas vezes por semana, principalmente antes das aulas pelas quais eram responsáveis para obter informações sobre o "ponto da situação". Um dos professores referiu que utilizava igualmente o sistema durante as aulas. Todos os professores mencionaram que regra geral também acediam ao sistema no seguimento da receção de notificações.

Quão útil foi a informação providenciada pelo sistema?
Todos os professores manifestaram que a informação disponibilizada pelo sistema foi bastante útil, principalmente a informação agregada. Para além disso, consideram como sendo muito importante a informação relativa ao trabalho dos estudantes efetuado em casa, que anteriormente não era possível quantificar/avaliar eficazmente.

Qual ou quais as funcionalidades do sistema que utilizou com mais frequência?
Para além da informação disponibilizada no *dashboard* do sistema, todos os professores referiram a matriz como a ferramenta que utilizaram com mais frequência, uma vez que

lhes permitia ter uma visão abrangente sobre a evolução do trabalho dos estudantes. Sempre que pretendiam informação mais detalhada sobre um estudante ou exercício em concreto, recorriam às ferramentas que lhes permitia aceder a essa informação.

Sentiu alguma dificuldade a interpretar a informação fornecida?

Em termos gerais, os professores referiram que não tiveram dificuldades a interpretar a informação providenciada pelo sistema e que consultavam habitualmente. Contudo, manifestaram que era importante que existisse algum tipo de ajuda contextual que permitisse compreender melhor algumas visualizações mais específicas e que não eram utilizadas com tanta frequência.

Adotou medidas diretamente influenciadas pela informação fornecida pelo sistema?

A este respeito, os professores disseram que efetivamente tomaram algumas medidas influenciadas pela informação fornecida pelo sistema, tais como providenciar exercícios para os estudantes que trabalhavam com um ritmo muito mais acelerado que os colegas; explicar melhor alguns exercícios em que visivelmente vários estudantes estavam a sentir dificuldades; abordar pessoalmente alguns estudantes no sentido de tentar ultrapassar alguma dificuldade que estivessem a sentir em determinado momento.

Em futuras unidades curriculares que venha a lecionar, consideraria utilizar o Sistema *CodeInsights*?

Todos os professores manifestaram o interesse em vir a utilizar o sistema em futuras edições desta unidade curricular (caso lhes venha a ser atribuída).

Quais foram as reações dos estudantes relativamente à utilização do sistema?

A este respeito, os professores disseram que numa fase inicial existiu alguma relutância em utilizar o sistema por parte dos estudantes, uma vez que não era claro para eles qual a utilização que seria dada à informação recolhida. Ainda que esta lhes tivesse sido devidamente explicada, os professores disseram que era visível que alguns estudantes sentiam algum desconforto em ter todas as suas submissões enviadas para o servidor. No entanto, a larga maioria dos estudantes reagiu bem à utilização do sistema e a partir de dada altura a sua utilização tornou-se natural.

Tem alguma sugestão para melhorias ao sistema?

Um dos professores sugeriu que seria interessante se os professores pudessem definir qual a informação que era apresentada imediatamente após ser efetuado o login.

Um outro professor sugeriu que deveria ser dado mais alguma atenção a aspetos relacionados com a qualidade do código produzido (e.g. complexidade das soluções).

Foram ainda feitas algumas pequenas sugestões sobre aspetos que poderiam ser melhorados ao nível da interação do utilizador, como a inclusão de legendas mais elucidativas e separar os exercícios por ficha quer na matriz, quer na listagem de exercícios que os estudantes tentaram resolver.

4.5.6.2. Fundamentos de Programação (LEI - ESTGOH)

Na unidade curricular de Fundamentos de Programação, da ESTGOH, o sistema foi utilizado entre os dias 25 de outubro e 31 de dezembro de 2017. Esta experiência iniciou-se alguns dias após o incêndio que assolou a zona de Oliveira do Hospital, pelo que esta não foi afetada por aquele acontecimento.

Embora estivessem matriculados 42 estudantes, apenas foram recebidos *snapshots* de 28 deles. Em todo o caso, a participação na experiência era voluntária e não tinha qualquer tipo de influência na classificação final dos estudantes. No total, foi proposto aos estudantes que resolvessem aproximadamente 90 exercícios agrupados em nove fichas de trabalho, tendo sido recebido um total de aproximadamente 25500 *snapshots* válidos (aproximadamente 30.000 no total). A maioria dos *snapshots* considerados inválidos não apresentava os elementos necessários à correta identificação do estudante e/ou do exercício e, como tal, foram descartados. Na altura, considerou-se que tal acontecia principalmente por esquecimento ou descuido. No entanto, após análise mais cuidada de alguns destes *snapshots* foi possível concluir que alguns haviam sido invalidados propositadamente pelos estudantes, mediante a utilização de identificadores inválidos como por exemplo "`@aid a.a`" ou "`@sid 0`". Este fenómeno da invalidação voluntária das submissões acabaria por vir a merecer mais atenção na iteração número seis.

Apenas cerca de um terço dos *snapshots* recebidos foram realizados durante o período das aulas. Na Figura 93, é possível observar que existe de facto uma distribuição consistente dos *snapshots* ao longo da maioria dos dias da semana e também ao longo das várias horas do dia.

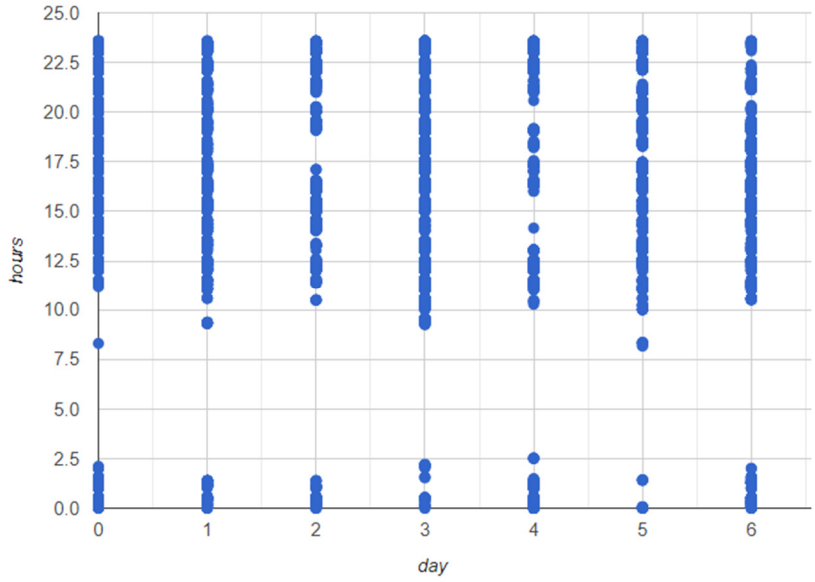

Figura 93 - Distribuição dos *snapshots* em função do dia e hora - FP ESTGOH - iteração quatro

Relativamente à distribuição dos *snapshots* ao longo do tempo, como pode ser observado na Figura 94, relativamente às primeiras quatro fichas de trabalho (WS 4 a WS 7), existiu alguma sobreposição entre fichas. Contudo, a partir da ficha nº 8 é possível verificar que esta sobreposição aumenta consideravelmente e a dada altura existem estudantes a trabalhar em cinco (e mesmo seis) fichas práticas diferentes.

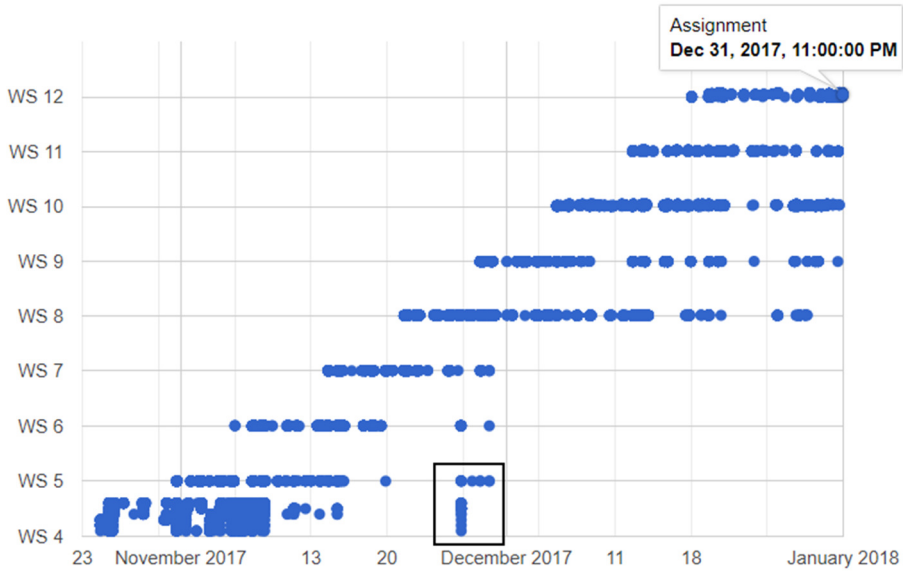

Figura 94 - Distribuição dos *snapshots* ao longo do tempo - FP ESTGOH - iteração quatro

O caso destacado na Figura 94 diz respeito a um estudante que começou a trabalhar mais tarde (eventualmente um estudante colocado na 2ª ou 3ª fase) e, como tal, fez as submissões para todos os exercícios num intervalo de tempo muito reduzido.

Tal como no caso da experiência anterior, houve alguma dificuldade inicial em perceber como funcionava o mecanismo de definição dos *inputs*. Contudo, com o passar do tempo, a generalidade dos estudantes conseguiu utilizar o sistema corretamente. Como exemplos dos erros cometidos destaca-se a utilização de "`//begin_imputs`" em vez de "`//begin_inputs`", a declaração das variáveis fora dos limites e a inicialização dentro dos limites definidos, entre outras, sendo que todas estas ações são reveladoras de alguma falta de atenção.

No caso desta experiência, o professor manifestou a intenção de não utilizar o sistema automático de notificações, optando por utilizar unicamente as diversas visualizações e informação disponibilizada pela ferramenta de monitorização para suportar as suas decisões e intervenções.

4.5.6.2.1. Entrevista ao professor

À semelhança do que aconteceu com a experiência anterior, também neste caso o professor foi entrevistado tendo por base a estrutura de entrevista previamente apresentada. Apresentam-se em seguida as respostas obtidas.

Com que frequência utilizou o sistema *CodeInsights*?

O professor desta unidade curricular indicou que utilizou o sistema principalmente enquanto estava na sala de aula com os estudantes. No entanto, referiu que fora deste âmbito acedia ao sistema algumas vezes por semana para verificar se existiam algumas situações que pudessem merecer a sua atenção.

Quão útil foi a informação providenciada pelo sistema?

O professor manifestou-se bastante agradado com a utilização do sistema uma vez que este lhe permitia aceder, de uma forma centralizada, a muita informação sobre o desempenho dos seus estudantes. Essa informação foi considerada útil, uma vez que em muitos casos os estudantes não assistiam às aulas, ou trabalhavam fora delas, não tendo ele no passado acesso a informação sobre esse trabalho.

Qual ou quais as funcionalidades do sistema que utilizou com mais frequência?

Por utilizar o sistema principalmente enquanto estava na sala com os estudantes, este professor referiu que a funcionalidade que utilizou com mais frequência foi a *live-feed* para ter acesso em tempo-real ao que estava a ser feito pelos estudantes. O professor referiu ainda que a partir da *live-feed*, sempre que considerava pertinente, acedia a

informação detalhada sobre o exercício em que um determinado estudante estava a trabalhar.

Sentiu alguma dificuldade a interpretar a informação fornecida?
Relativamente às funcionalidades que utilizou frequentemente, o professor manifestou não ter tido dificuldades a interpretar a informação apresentada.

Adotou medidas diretamente influenciadas pela informação fornecida pelo sistema?
Uma vez que utilizava principalmente a *live-feed*, o professor referiu que com base em informação obtida no sistema, por diversas vezes abordou pessoalmente os estudantes por forma a ajudá-los a ultrapassar algumas dificuldades encontradas e que não estavam a conseguir ultrapassar.

Em futuras unidades curriculares que venha a lecionar, consideraria utilizar o Sistema *CodeInsights*?
Este professor já havia utilizado o sistema anteriormente e manifestou o interesse em voltar a utilizá-lo futuramente.

Quais foram as reações dos estudantes relativamente à utilização do sistema?
Uma vez que a utilização do plug-in era opcional, o professor mencionou que um número muito reduzido de estudantes indicou que não pretendia instalar o *plug-in*. Entre os estudantes que de facto instalaram e utilizaram o sistema, o professor referiu que sentiu da parte de alguns estudantes um certo desconforto pelo facto de todas as submissões serem enviadas para o servidor.

Tem alguma sugestão para melhorias ao sistema?
Uma vez que o sistema já permite classificar os exercícios resolvidos pelos estudantes, o professor mencionou que poderia ser interessante providenciar uma forma de gerar uma classificação mais formal que pudesse automaticamente ser utilizada na determinação da nota final dos estudantes. O professor referiu ainda que seria interessante se o sistema incluísse algum tipo de mecanismo de deteção de cópia/plágio entre os estudantes.

4.5.6.3. Programação (LEI - ESTGOH)

Tal como previsto, o sistema viria a ser utilizado uma vez mais na unidade curricular de Programação que está inserida no segundo ano da Licenciatura em Engenharia Informática da ESTGOH. No ano anterior, havia sido utilizado o sistema nesta unidade curricular, contudo, devido a problemas de ordem técnica a experiência acabaria por não correr como desejado. Este não era o contexto ideal para a utilização do sistema, uma vez que não se trata de uma unidade curricular introdutória, sendo já a terceira unidade de programação que estes estudantes frequentam. Ainda assim, uma vez que numa fase inicial são abordadas algumas temáticas de caráter mais introdutório sobre a linguagem Java, em concordância com o professor da unidade curricular, foi decidido voltar a tentar.

Tendo em consideração o que foi dito em relação à natureza da unidade curricular, estava logo à partida previsto que o sistema apenas fosse utilizado durante as primeiras semanas de aulas em que são apresentados os conceitos base da linguagem Java e são relembrados alguns conceitos que os estudantes já deveriam conhecer de outras unidades curriculares, como a utilização de instruções condicionais ou ciclos.

A experiência começaria no dia 21 de setembro e acabaria por terminar de forma forçada no dia 15 de outubro, uma vez mais por causa do incêndio. Ao contrário da experiência do *Programa Acertar o Rumo* que decorria em Coimbra e que ainda que tenha sido impactada num momento inicial, voltou ao normal após a instalação do sistema em outro servidor, neste caso, o impacto foi muito maior pois não só não havia comunicações, como também não existiram aulas durante alguns dias após o incêndio.

Ao tratar-se de uma unidade curricular com um número de estudantes repetentes relativamente alto (aproximadamente metade dos estudantes), existiu logo à partida alguma dificuldade em convencer os estudantes a instalar o *plug-in*. Com efeito, foram muito poucos os estudantes repetentes que instalaram e utilizaram o *plug-in*. De realçar que eram estes estudantes que mais poderiam beneficiar da utilização do sistema. Assim, dos 25 inscritos a esta unidade curricular, foram recebidos *snapshots* de apenas 11. Convirá ainda referir que a grande maioria dos estudantes que decidiram não instalar o *plug-in* também não assistiram às aulas regularmente durante o período da experiência.

No total, foram recebidos ao longo da duração da experiência cerca de 1100 *snapshots*, sendo que apenas cerca de 950 foram considerados válidos. Destes, apenas cerca de 33% foram realizados durante as aulas. Ainda que o número de *snapshots* recebidos tenha

sido francamente baixo, foi ainda assim possível constatar que apesar de os estudantes apenas terem duas aulas de três horas com o professor, foram recebidos *snapshots* ao longo dos vários dias da semana, tal como pode ser observado na Figura 95.

Tal como nas restantes experiências, existiram numa fase inicial algumas dificuldades em perceber a forma correta de definir as variáveis de *input*, em todo o caso, com o passar do tempo, acabaria por se resolver.

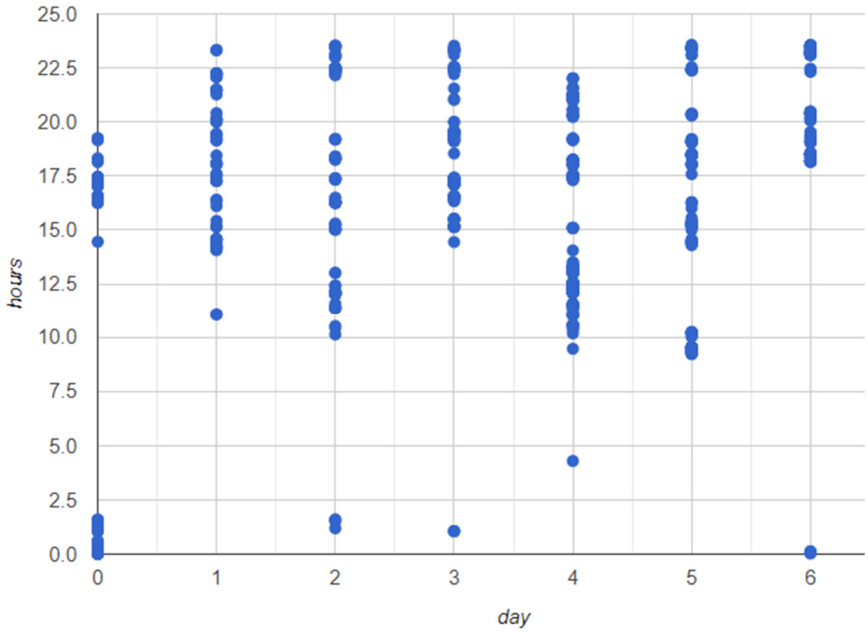

Figura 95 - Distribuição dos *snapshots* em função da hora e dia da semana - P ESTGOH - iteração quatro

Nesta experiência, estava prevista a utilização das notificações referentes à existência de um número invulgar de tentativas de resolução de um exercício, tendo sido definidas as expectativas de forma automática para todos os exercícios, aquando da sua criação, com as seguintes probabilidades: evento A (0-50) - 0,9 (D=0), evento B (51-100) - 0,1 (D=1). Tendo em conta que apenas dois exercícios tiveram mais do que 40 tentativas (43 e 48), acabou por não ser gerada nenhuma notificação deste tipo.

Relativamente às notificações sobre a existência de estudantes sem nenhuma submissão nos últimos três dias, o sistema foi, tal como no caso da experiência do Programa Acertar o Rumo, configurado para fazer esta verificação todos os dias às 23h55 com base em expectativas sob a forma de probabilidade de existência de estudantes nestas condições (evento A - não existência (90%); evento B - existência (10%)). Ao longo da experiência foram geradas sete notificações deste tipo, tendo sido possível verificar que em duas delas os estudantes tinham desinstalado o *plug-in* e nas restantes, perante a não comparências nas aulas e a total falta de participação, o professor acabou por considerar

que estes estudantes haviam desistido. Ainda que tivessem sido assinalados como desistentes, o sistema continuou a receber *snapshots* destes estudantes, curiosamente, na maioria dos casos, alguns dias após serem reestabelecias as comunicações e já depois de a experiência ter sido dada como terminada.

Em relação às notificações sobre a existência de exercícios sem qualquer tentativa de resolução para o período esperado, a verificação era feita às 23h55 do último dia do período associado aos objetivos e foram sendo definidas expectativas para cada período sob a forma de probabilidade de existência de exercícios associados aos objetivos em vigor sem uma única tentativa de resolução (evento A - não existência (80%); evento B - existência (20%)), não tendo sido recebida nenhuma notificação deste tipo.

Por não estar ainda terminado o seu desenvolvimento e teste à data de início da experiência, não foram utilizadas as notificações relativas às diferenças de ritmo de trabalho e mudanças de "nível de performance".

A título de curiosidade, convém dizer apenas que os dois estudantes com mais exercícios corretamente resolvidos e autores de quase metade dos *snapshots* recebidos acabariam, sem grande surpresa, por obter as classificações mais elevadas entre os estudantes da turma.

Infelizmente, uma vez mais a experiência nesta unidade curricular acabaria por não correr como esperado. Se logo à partida existiu um número elevado de estudantes que optou por não instalar o *plug-in*, a ocorrência do incêndio acabaria por levar a que, em concordância com o professor, fosse sugerido aos estudantes para deixarem simplesmente de utilizar o *plug-in*.

Dadas as circunstâncias e tendo em conta que claramente a experiência não havia corrido bem, não foi feita a entrevista ao professor tal como estava planeado. No entanto, houve uma reunião de caráter informal, na qual o professor se manifestou claramente insatisfeito com toda a situação, tendo ainda assim, demonstrado que a pouca informação que o sistema recolheu, serviu-lhe ainda assim para monitorizar o trabalho dos estudantes que decidiram instalar o *plug-in*, com especial incidência no trabalho realizado no período extra-aulas. Em todo o caso, com ambas as partes de acordo, foi decidido que não faria sentido voltar a utilizar o sistema neste contexto.

4.5.6.4. Programação para a Internet I (LEI - ESTGOH)

Ainda que não se trate claramente do contexto para o qual o sistema foi pensado, procedeu-se à sua utilização do ***CodeInsights*** numa unidade curricular de introdução à programação para a internet, com especial enfoque no desenvolvimento do lado do cliente com recurso a HTML. Como mencionado anteriormente, recorreu-se a uma API disponibilizada pelo W3C por forma a conseguir saber de forma automática se o código apresentava ou não erros ou avisos relativos à estrutura dos documentos HTML desenvolvidos pelos estudantes. Assim o sistema foi utilizado neste contexto durante as três semanas nas quais foi abordada esta temática.

Embora estivessem matriculados nesta unidade curricular 30 estudantes, apenas 16 de facto assistiram às aulas e realizaram os exercícios propostos. No final da experiência foram recebidos aproximadamente 5800 *snapshots* válidos referentes a 15 exercícios organizados em três fichas de trabalho. Apenas cerca de 30% dos *snapshots* foram recebidos durante o período das aulas.

Ainda que a duração da experiência não tenha sido muito alongada e o número de estudantes que dela participaram não tenha sido muito grande, foi ainda assim possível demonstrar que o sistema pode de facto ser utilizado em outros contextos diversos daquele para o qual foi idealizado inicialmente.

O facto de HTML não ser considerada uma linguagem de programação faz com que exista por vezes algum desleixo na escrita do código nesta linguagem, uma vez que os navegadores conseguem em grande medida corrigir/ultrapassar os erros existentes.

Graças à utilização dos ***CodeInsights*** neste contexto, o professor conseguiu ter acesso ao trabalho dos estudantes, podendo visualizar de uma forma prática o resultado e para além disso pode ter acesso a informação sobre erros e avisos associados ao código produzido pelos estudantes, o que lhe permitiu identificar diversas situações que considerou merecedoras de intervenção direta. Além disso, permitiu igualmente aos estudantes terem algum *feedback* sobre a qualidade do código por eles produzido de forma imediata, o que foi encarado como sendo uma clara mais-valia.

4.5.7. Sumário da iteração quatro

Do ponto de vista do desenvolvimento, esta acabou por ser uma das iterações mais profícuas. Com a introdução de um mecanismo de utilização de casos de teste foi possível

terminar algumas funcionalidades que haviam sido iniciadas, assim com desenvolver outras. Estas funcionalidades permitem fornecer muita informação importante aos professores sobre o desempenho dos seus estudantes, permitindo-lhes adotar as medidas necessárias para mitigar os problemas encontrados de uma forma mais fundamentada.

Ao nível do sistema de geração de notificações, foram feitos diversos avanços, nomeadamente em termos do desenvolvimento de novos tipos de notificação. Contudo, foram uma vez mais notórias algumas limitações, nomeadamente no que concerne à definição das expectativas dos professores. Não foi efetivamente fácil convencer os professores a definir as expectativas, pelo que acabariam na esmagadora maioria dos casos por ser utilizados os valores de probabilidade definidos por omissão. Como já foi dito anteriormente, para definir automaticamente estas probabilidades de forma confiável, era necessário um grande volume de dados, de que ainda não dispomos.

Para alguns tipos de notificações são utilizados conjuntos de probabilidades pré-definidos e que poderão não ser os mais adequados. Ainda que seja dada a possibilidade de os professores alterarem os valores das probabilidades, o que como pôde ser observado, praticamente aconteceu.

Nesta iteração, foi também disponibilizada uma ferramenta que permitia aos estudantes acompanharem o seu trabalho. Esta acabaria por se revelar uma forma importante para os estudantes verificarem se o seu código está ou não a produzir o resultado esperado. Muito embora esta tenha de facto sido utilizada pela maioria dos estudantes, o facto de ser necessário estar constantemente a aceder à aplicação para saber se o código passou ou não nos testes acaba por ser uma limitação, pelo que deverão ser implementadas formas de apresentação desta informação aos estudantes de uma forma mais direta, à semelhança do que ocorria em relação à apresentação da informação sobre os erros e *warnings* de HTML diretamente no Atom.

Um outro aspeto que deverá merecer alguma atenção nas próximas iterações está associado ao mecanismo de declaração e inicialização das variáveis de *input*. No geral, a sua utilização foi positiva, contudo, tal como já foi mencionado, existiram alguns estudantes que apresentaram algumas dificuldades na sua utilização. Ainda que tenha sido providenciada documentação sobre como o utilizar, eventualmente, esta informação poderá não ter sido suficientemente clara, pelo que algo deverá ser feito a este respeito, uma vez que se trata de um elemento importante do sistema.

As entrevistas realizadas aos professores permitiram constatar que a recetividade do sistema pelos professores foi bastante elevada. Todos os professores referiram que a utilização do sistema lhes permitiu de facto aceder a um conjunto de informações que de outra forma não poderiam ter tido acesso, tendo-lhes permitido realizar um conjunto de intervenções e modificações com vista à melhoria do desempenho dos estudantes. Ainda assim foram feitas algumas sugestões de melhoria, que haveriam de merecer a devida atenção nas iterações seguintes.

Infelizmente, uma das experiências claramente não correu como esperado. Ainda que à partida já se tivesse a noção de que aquele poderia não ser o contexto ideal, a ocorrência do incêndio e a decisão de não instalar o *plug-in* por parte de alguns estudantes acabariam por fazer com que esta experiência fosse de facto pouco produtiva.

4.6. Iteração cinco

4.6.1. Objetivos

Para além de continuar a testar as funcionalidades desenvolvidas na iteração anterior, algumas das quais acabariam por não ser devidamente testadas nessa altura, nesta iteração foram aperfeiçoadas algumas funcionalidades já existentes, com base no *feedback* obtido, e foram desenvolvidas algumas novas que, entretanto, haviam sido solicitadas por alguns professores, como por exemplo ao nível da deteção de plágio.

4.6.2. Descrição do contexto

Ao contrário do que estava previsto, não foi possível utilizar o sistema na unidade curricular de Programação da Licenciatura em Engenharia da Gestão Industrial da Universidade de Coimbra, tal como havia acontecido nas duas últimas edições da mesma. Assim, nesta iteração o sistema viria a ser testado unicamente na unidade curricular de Programação para a Internet II integrada na Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital, na qual é utilizada a linguagem de programação PHP.

4.6.3. Trabalho desenvolvido

4.6.3.1. Desenvolvimento do *plug-in*

Atendendo ao facto de se ter utilizado o ***Atom*** como ambiente de desenvolvimento em algumas experiências da iteração anterior, para esta experiência acabaria igualmente por se desenvolver uma nova versão do *plug-in* especificamente para a linguagem PHP e que era invocado sempre que os estudantes guardavam o código.

Na iteração anterior, acabaria por se constatar que, muito embora a disponibilização da ferramenta de monitorização aos estudantes constituísse uma mais-valia, a necessidade de lhe aceder com regularidade para verificar se os exercícios eram ou não considerados certos acabaria por nos levar a conceber uma solução que permitisse obter essa informação diretamente no ambiente de desenvolvimento. Assim, no caso concreto do ***Atom***, foi desenvolvida e disponibilizada uma funcionalidade que permitia aos estudantes que quando guardassem o código, para além de este ser automaticamente enviado para o servidor, poderiam ainda visualizar diretamente no ***Atom*** o resultado execução do seu código com os casos de teste associados a cada exercício. Na Figura 96, é possível observar um exemplo desta funcionalidade.

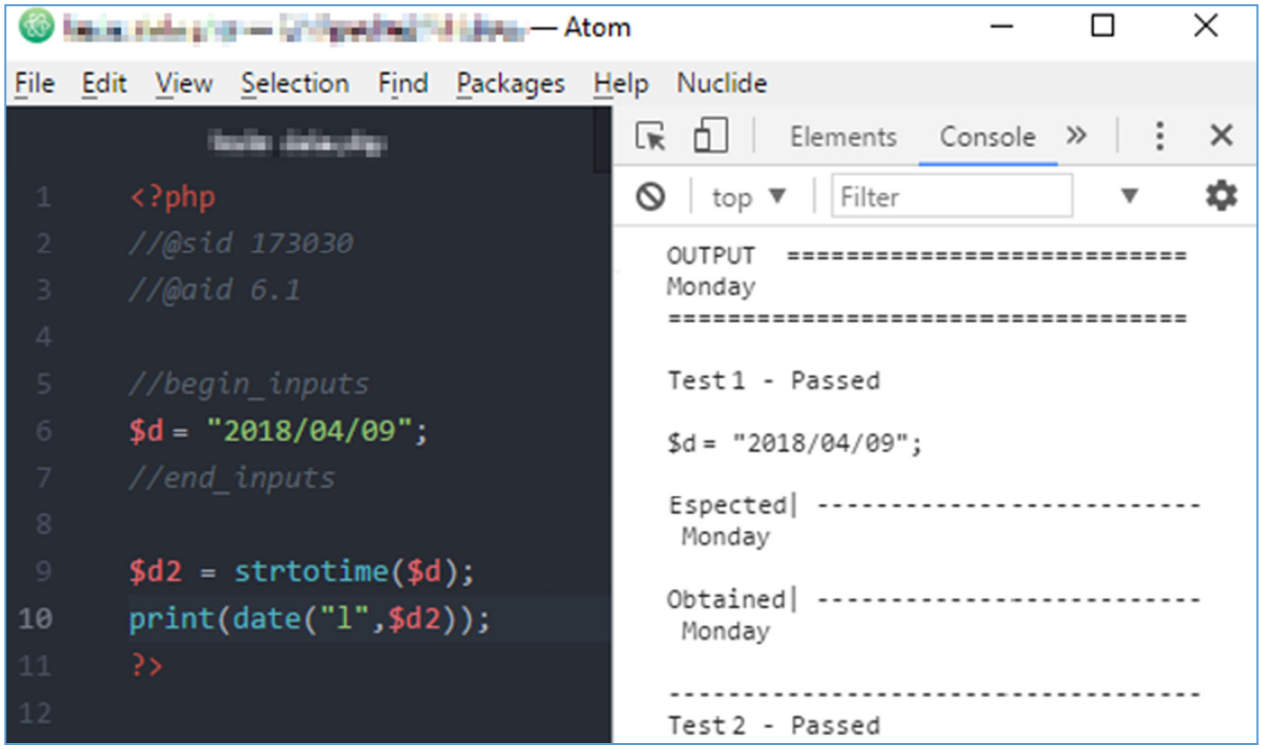

Figura 96 - Exemplo de apresentação dos resultados da execução do código com os casos de teste - iteração cinco.

Assim, sempre que os estudantes guardavam o código, uma cópia deste era enviada para o servidor para ser processada e testada. Caso existissem erros de interpretação/compilação, estes eram apresentados diretamente ao estudante. Caso não existissem quaisquer erros de interpretação/compilação, o código era executado e o output produzido pelo código fonte era apresentado. Para além deste output, para cada teste realizado era apresentada informação sobre: o resultado dos testes (Passed / Passed partially / Failed), os inputs utilizados no teste, e o output esperado e o output obtido.

Se por algum motivo faltassem elementos identificativos, ou caso não fosse possível testar o código, era apresentada aos estudantes uma mensagem com essa informação.

Convirá dizer que, para esta funcionalidade ser possível, era necessário fazer os pedidos ao servidor de forma síncrona, isto é, para conseguir obter os resultados, era necessário que o editor ficasse "bloqueado" durante o tempo necessário para a realização dos testes. Como foi visto anteriormente, esta forma de fazer os pedidos poderia fazer com que o editor dos estudantes ficasse bloqueado por demasiado tempo. Assim, antes de se recorrer a esta funcionalidade, deveriam ser feitos os testes necessários para assegurar que estes períodos de bloqueio não fossem notórios. No caso concreto desta experiência, foram feitos diversos testes que permitiram assegurar que, na esmagadora maioria dos casos, esta funcionalidade iria funcionar de forma quase imediata.

Ainda a respeito desta funcionalidade e da sua utilização no âmbito desta unidade curricular, convirá mencionar que ao tratar-se de uma unidade curricular centrada no desenvolvimento para a *web* do lado do servidor, com PHP, a utilização de algumas funcionalidades deixava de fazer sentido. Com efeito, a execução do código com casos de teste foi utilizada apenas nos casos em que tal fizesse sentido. Por exemplo, a partir do momento em que o resultado da "execução" dos programas fossem pedaços de código HTML com dezenas de linhas, utilizar estas funcionalidades tornava-se demasiado complicado.

4.6.3.2. Receção e processamento dos *snapshots*

Como já foi mencionado, em termos do processamento dos *snapshots*, foi apenas necessário fazer algumas modificações de modo a suportar o envio da informação de *feedback* apresentada na secção anterior.

4.6.3.3. Deteção de plágio em tempo real

Na sequência da solicitação de um professor, mas também porque considerámos verdadeiramente oportuno fazê-lo, nesta iteração foi desenvolvida uma funcionalidade que permitia a deteção de situações de plágio em tempo real.

Segundo Carter [39], a par da diversidade de estudantes, a submissão de trabalhos plagiados constitui um dos principais problemas com que os professores se têm de debater no seu dia a dia. Segundo Faidhi et al. [144] existem diversos tipos diferentes de plágio que vão desde a submissão de uma cópia integral do trabalho dos colegas até métodos bem mais complexos e difíceis de detetar.

Em muitos casos, as formas de plágio mais frequentes e também as mais preocupantes são precisamente as mais fáceis de fazer e também de detetar. Numa parte considerável das situações, os estudantes recorrem ao plágio porque não dedicaram tempo suficiente à elaboração dos trabalhos e como sentem algum tipo de pressão para entregar os trabalhos (*e.g.* contarem para efeitos de avaliação) acabam por pedir as resoluções aos colegas. Assim, numa tentativa de tentar iludir os professores, em muitos casos limitam-se a submeter uma cópia do trabalho dos colegas (eventualmente alterando alguns nomes de variáveis) sem se darem sequer ao trabalho de tentar compreender as resoluções. Muitos outros estudantes, colocam de facto algum empenho na resolução dos exercícios, contudo, a dada altura acabam por encontrar dificuldades que impedem o seu progresso e em vez de tentarem obter ajuda junto dos professores, acabam simplesmente por recorrer ao plágio.

Nos últimos anos, com a generalização do uso de redes sociais e ferramentas de *instant messaging*, tem-se vindo a assistir a um agravar da situação, dada a facilidade com que se consegue obter e partilhar trabalhos entre conjuntos de estudantes [145]. Tal como um estudo recente do *New York Times* revelou [146], trata-se efetivamente de um problema grave e global, ao qual nem mesmo reputadas instituições de ensino, norte-americanas, por exemplo, com severas políticas anti plágio, conseguem escapar.

Além de esta atitude ser eticamente e moralmente questionável e reprovável, acaba igualmente por não contribuir em nada para o processo individual de aprendizagem dos estudantes que praticam tais atos. Ao plagiarem, os estudantes acabam por não aprender, pondo em causa não só a progressão naquela unidade curricular em particular, mas também em termos globais de progressão no curso em que esteja inscrito.

No caso concreto da programação, é possível encontrar diversas soluções de deteção de plágio bastante elaboradas e consequentemente demoradas como os sistemas ***MOSS***[27] ou ***JPlag***[28]. Ainda que possam de facto ser muito eficazes no processo de deteção de plágio, um dos principais problemas da maioria dos sistemas/abordagens existentes consiste no facto de serem geralmente aplicados apenas na última tentativa de resolução submetida pelos estudantes para exercícios de avaliação de carácter sumativo.

Este facto acaba por contribuir para que sejam utilizados principalmente como ferramentas de apoio a processos punitivos, ou seja, o objetivo é apenas detetar quem prevaricou. Tal como defendido, por exemplo, por Leung et al. [147] e por Özbek [148], em vez de simplesmente identificar e punir os estudantes prevaricadores, um sistema de deteção de plágio deveria ser utilizado de modo formativo e como tal ter como principal objetivo a identificação de estudantes que possam estar com problemas de aprendizagem ou falta de motivação e que acabam por recorrer ao plágio. Obviamente, as avaliações sumativas são importantes e detetar plágio nestas situações é igualmente importante, contudo, como já foi mencionado, muitos estudantes acabam por recorrer ao plágio nestes momentos de avaliação, porque as suas dificuldades/problemas não foram devidamente identificadas e ultrapassadas nos momentos de avaliação formativa precedentes.

Tendo em consideração que o sistema ***CodeInsights*** recebe em tempo real as tentativas de resolução dos exercícios por parte dos estudantes, será possível detetar estas situações numa fase precoce, o que permitirá ao professor investigar o que terá levado o estudante a plagiar, e em consequência disso, abordar pessoalmente os estudantes de modo que as eventuais dificuldades sejam mitigadas atempadamente. O objetivo não é de todo ter 100% certeza de que um estudante plagiou, mas sim alertar o professor para uma potencial situação de plágio, permitindo-lhe investigar a situação e, se necessário, tomar as medidas consideradas necessárias. Na Figura 97, pode ser observada uma representação desta mudança de "paradigma".

Esta abordagem formativa contrasta claramente com a abordagem punitiva predominante e, uma vez que permite identificar em tempo real estas situações, poderá em última instância permitir o esclarecimento de dúvidas dos estudantes que de outra forma não seriam identificadas e que poderiam levar a que os estudantes em causa continuassem a plagiar repetidamente, uma vez que as suas dúvidas nunca seriam

[27] https://theory.stanford.edu/~aiken/moss/

[28] https://jplag.ipd.kit.edu/

esclarecidas. Em todo o caso, o que é importante é que as situações de potencial plágio possam ser detetadas tão rapidamente quanto possível, permitindo aos professores intervir como entenderem.

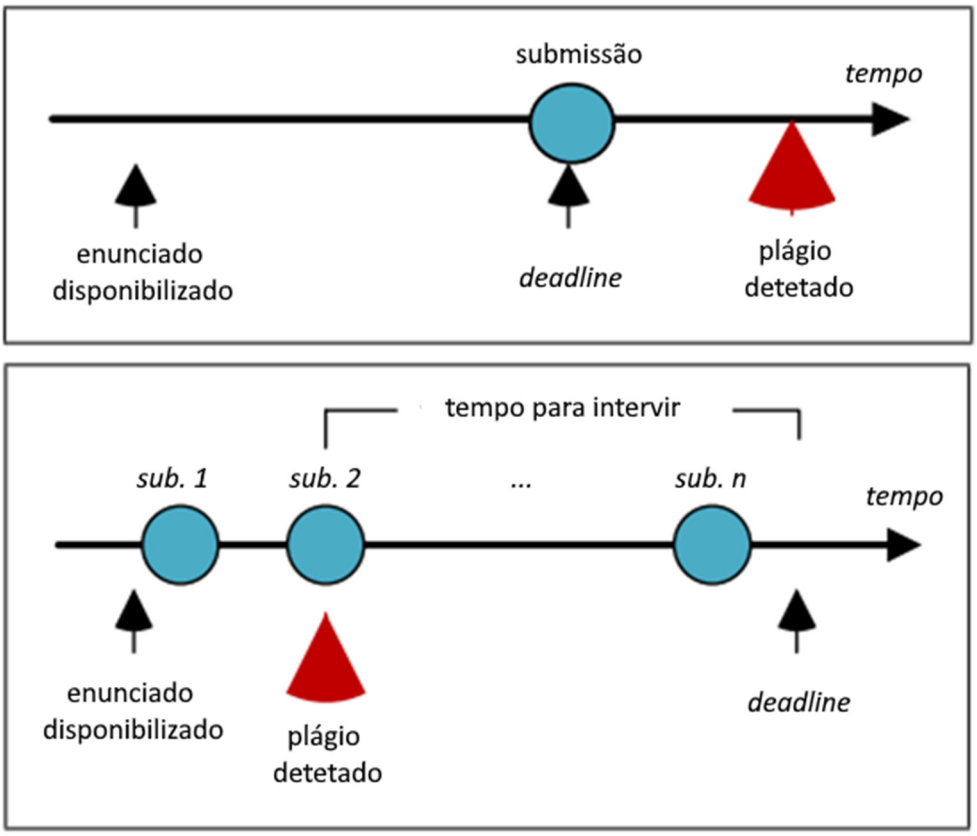

Figura 97 - Representação da abordagem de deteção de plágio tradicional (em cima) vs. deteção em tempo real (em baixo).

Tendo em conta a natureza da exercícios tipicamente resolvidos pelos estudantes em unidades curriculares de introdução à programação (*i.e.* exercícios de um único ficheiro de código, muitas vezes com uma única função), o tipo de plágio predominante (cópia integral) e principalmente o facto de não poder ser um processo demorado (uma vez que a deteção de eventual plágio ocorrerá em tempo real e como mais uma etapa de um já longo processo), optou-se por utilizar uma abordagem muito simples que consiste na comparação direta do código dos *snapshots* recebidos.

Assim, quando um novo *snapshot* (`newSubmission`) é recebido, após ser devidamente identificado o estudante (`Sid`) e o exercício (`Aid`), é comparado com o código de todas as submissões dos restantes estudantes para o mesmo exercício, utilizando um algoritmo tradicional de comparação de *strings*, que fornece como resultado um valor de percentagem de semelhança. Estes valores de semelhança serão armazenados em base

de dados, ficando de imediato acessíveis para o professor. Este processo encontra-se representado no algoritmo representado na Figura 98.

```
Sid = Identify the student
Aid = Identify the assignment
lastSubmissions = getLastSubmissions (Sid, Aid)
For each submission in lastSubmissions
   simm = Compare(newSubmission, submission)
   Save simm
```

Figura 98 - Algoritmo de comparação de submissões

Principalmente ao nível do método de comparação de submissões, esta abordagem poderá ser demasiado simples e que existem, por certo, outras abordagens capazes de obter resultados mais fiáveis. Em todo o caso, há que ter em consideração a quantidade de processamento que já é efetuado sobre os *snapshots*, pelo que a utilização de abordagens mais complexas, iria certamente aumentar consideravelmente o tempo de processamento com todas as desvantagens associadas.

Outro aspeto que convirá mencionar diz respeito ao facto de apenas se estar a comparar o novo *snapshot* com as últimas versões de cada estudante. Pelos mesmos motivos apresentados anteriormente, fazer a comparação com todas as tentativas de todos os estudantes, seria demasiado pesado. A existir de facto plágio, este será provavelmente em relação às versões mais recentes e não em relação às versões iniciais. Em todo o caso, por se tratar de um processo dinâmico, as novas submissões serão sempre comparadas.

Até este momento apenas se procedeu à comparação de *strings*, não tendo ainda sido definido se se trata de plágio ou não. Para este efeito, recorre-se a um limiar de semelhança (`minSimilarity`), a partir do qual se considera que existe demasiada semelhança e como tal poder-se-á estar na presença de plágio. Convirá, uma vez mais, reforçar que com este sistema apenas se pretende chamar a atenção para potenciais situações de plágio, devendo o professor investigar e confirmar (ou não) se se trata efetivamente de plágio.

Para efeitos de apresentação desta informação aos professores, foi desenvolvida uma nova funcionalidade que permite apresentar ao professor uma vista global de todos os exercícios de todos os estudantes em que exista pelo menos uma situação em que a percentagem de semelhança entre o código de dois estudantes seja superior a

`minSimilarity` (retângulos a vermelho). Um exemplo desta visualização pode ser observado na Figura 99.

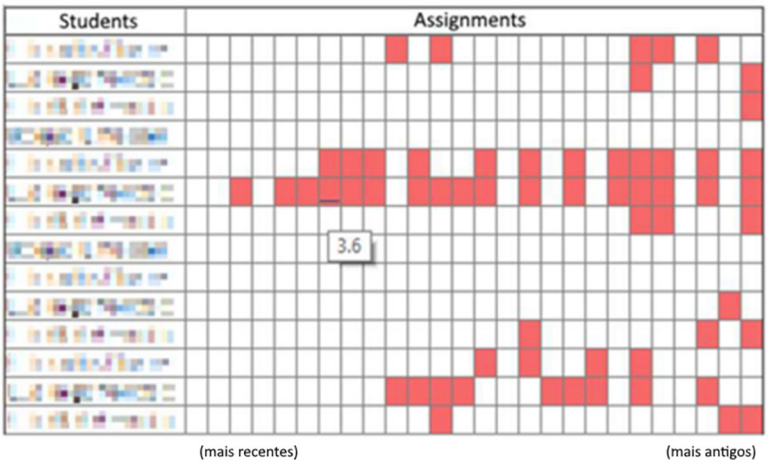

Figura 99 - Listagem de exercícios com potenciais situações de plágio.

Para cada par estudante/exercício assinalado a vermelho, o professor poderá clicar para aceder a mais detalhes. Assim, ao escolher o estudante A1 e o exercício E1, será apresentada uma visualização contendo informação sobre a percentagem de semelhança entre as submissões realizadas por este estudante e as últimas tentativas de resolução (ao momento) de todos os colegas. Por forma a simplificar a visualização, o valor de semelhança está codificado utilizando um esquema de gradação de cores. Estas cores variam do verde-escuro (pouca semelhança) até ao vermelho-escuro (muita semelhança). Ao passar o rato por cima de uma determinada célula, será apresentada uma *tooltip* com informação sobre o valor de semelhança. Um exemplo desta visualização pode ser observado na Figura 100.

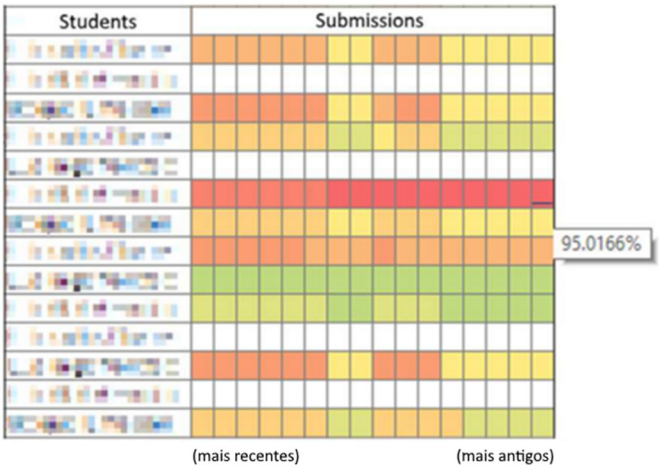

Figura 100 - Detalhes sobre a comparação entre todas as submissões para um exercício.

A partir desta tabela, o professor poderá obter mais informações sobre a comparação entre as várias tentativas, bastando para tal, clicar numa célula. Esta informação será apresentada aos professores na forma de comparação e destaque das semelhanças e diferenças entre ambas as submissões, tal como apresentado na Figura 101.

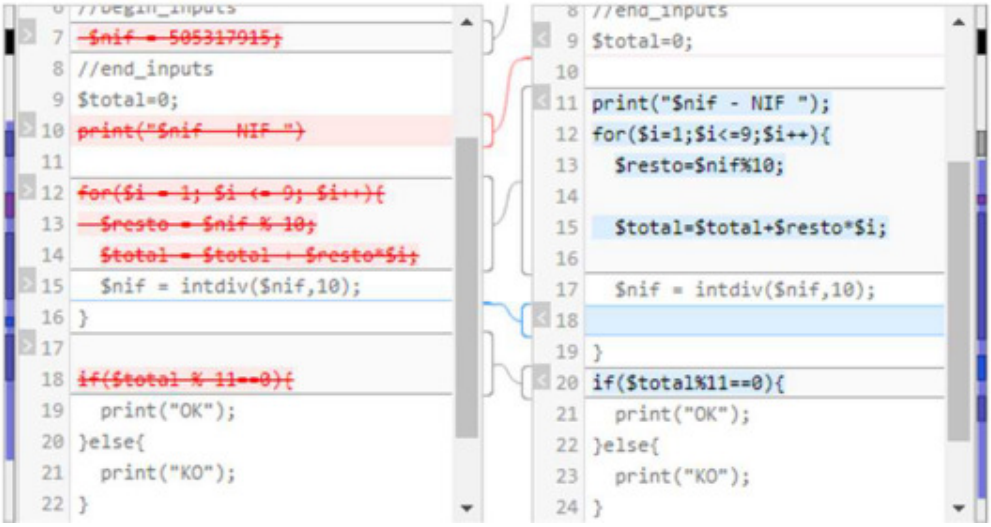

Figura 101 - Detalhes da comparação entre duas submissões

Perante esta informação, o professor poderá comprovar se de facto existiu plágio, podendo em seguida tomar as medidas mais adequadas. Caso chegue à conclusão que de facto não existiu plágio, poderá assinalar esta situação como um "falso positivo".

Muito embora a análise "manual" dos dados e visualizações fornecidas já constitua uma valiosa ferramenta, tendo em consideração que o sistema ***CodeInsights*** já dispõe de um sistema de notificações, foi considerado pertinente utilizar este mecanismo também relativamente às situações de plágio. Assim, foi disponibilizada uma notificação do tipo *online* para este propósito. Em termos gerais o seu funcionamento é muito semelhante ao previamente apresentado relativamente a notificações sobre um número invulgar de tentativas de resolução. Neste caso, as expectativas são expressas em termos da probabilidade de existirem exercícios com um determinado valor de semelhança, tal como o exemplo apresentado na Tabela 9.

Tabela 9 - Tabela de probabilidades e desejos associados ao Exercício 1.

Exercício 1			
	semelhança	**probabilidade**	***D***
A	[0, 95[	0.98	0
B	[96, 100]	0.02	1

Assim, aquando da receção de um novo *snapshot,* após ser realizada a comparação entre *snapshots* para determinar a sua semelhança de acordo com o algoritmo anteriormente apresentado, irá ser verificado se uma nova notificação deverá ser gerada ou não, de acordo com o princípio de atenção seletiva. Quando se verificar que existe um par de *snapshots* relativamente aos quais a notificação deve ser gerada, o processo terminará e a notificação será, eventualmente, gerada.

Por forma a evitar que sejam geradas várias notificações sobre o mesmo assunto, recorre-se à "assinatura" da notificação. Deste modo, a notificação será apenas gerada no caso de se verificar que não existe nenhuma notificação com a mesma assinatura. No caso desta notificação, a assinatura teria a seguinte estrutura: `[tipo de notificação]-[id do estudante]-[id do exercício]-[evento]`.

A título de exemplo, imagine-se que aquando da 20ª tentativa de resolução do estudante A1 para o exercício E1, é verificado que existe pelo menos um grau de semelhança superior a 98% relativamente a pelo menos um *snapshot* já recebido anteriormente de outros estudantes. Por uma questão de eliminação de trabalho eventualmente inútil, a primeira coisa a fazer era verificar se já existia uma notificação com a assinatura `Plag-A1-E1-96100`. Em caso afirmativo o processo terminaria, caso contrário, recorrendo ao princípio da atenção seletiva era determinado se era pertinente ser gerada uma nova notificação.

Ao ver a notificação em causa, o professor poderá aceder a uma página semelhante à que está representada na Figura 100, obter mais detalhes e em função da informação recolhida agir da forma que considere mais adequada à situação. É, contudo, importante reforçar, que o sistema não tomará qualquer tipo de medida adicional para além da geração da notificação, cabendo sempre ao professor determinar as ações que considere pertinentes para ultrapassar esta situação.

Para além deste tipo de notificação *online*, foi ainda idealizado um outro tipo de notificações *off-line*, em que as expectativas eram definidas em termos do número de estudantes para os quais haviam sido detetados *snapshots* em situação de potencial plágio. Contudo, este tipo de notificação acabaria por não ser implementado.

Independentemente de recorrer ao mecanismo de notificações ou não, utilizando estas funcionalidades, o professor tem a possibilidade de estar ao corrente da existência de ocorrência de plágio numa fase precoce e a sua intervenção atempada poderá contribuir para que os estudantes ultrapassem as dificuldades que possam estar a atravessar.

4.6.3.4. Arquitetura geral do sistema

Após terem sido introduzidas algumas novas funcionalidades que não estavam inicialmente previstas (*e.g.* deteção de plágio) e de terem sido finalmente implementadas algumas funcionalidades importantes (*e.g.* execução do código com casos de teste) na iteração anterior, a arquitetura geral do sistema havia sofrido algumas modificações importantes. Na Figura 102, pode ser observada a arquitetura geral do sistema tal como estava no final da iteração cinco.

Em termos gerais, o sistema ***CodeInsights*** é constituído por cinco componentes distintos, contudo interligados e dependentes uns dos outros:

- Ferramenta de processamento dos *snapshots*;
- Ferramenta responsável pela compilação/interpretação e teste dos *snapshots*;
- Base de dados com informação sobre os *snapshots* recebidos;
- Aplicação *web-based* de monitorização do progresso/performance dos estudantes;
- Sistema automático de notificações para os professores.

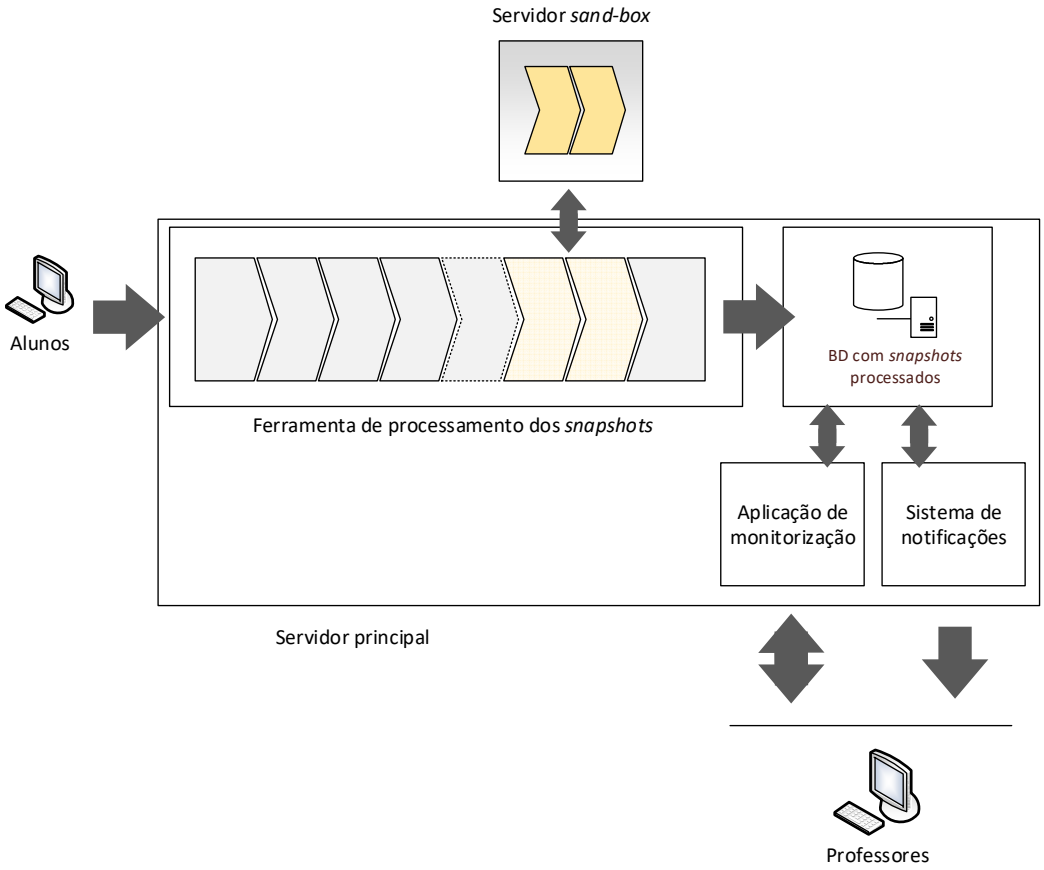

Figura 102 - Arquitetura geral do sistema - iteração cinco.

4.6.3.4.1. Ferramenta de processamento dos *snapshots*

Ainda que todos os componentes do sistema sejam importantes, a ferramenta de processamento dos *snapshots* no servidor acaba sem dúvidas por ser uma das componentes de maior relevância para o funcionamento do sistema.

Este processamento consiste na execução de um conjunto de tarefas sequenciais, previamente apresentadas. Ao longo das várias tarefas vai sendo produzida e recolhida muita informação relevante e que posteriormente virá a ser utilizada para fornecer aos professores a informação que lhes permitirá assistir os seus estudantes da forma mais adequada. Na Figura 103, pode ser observada uma ilustração deste processo.

Tendo em conta o modo como o sistema foi concebido, facilmente podem ser desativadas/ativadas ou mesmo adicionadas outras etapas ao processo. A título de exemplo, poderia pretender-se avaliar o *snapshot* relativamente à complexidade do código desenvolvido (o que acabaria efetivamente por ocorrer no caso da linguagem Python) ou a qualidade da documentação do código. Para tal, bastaria implementar o código necessário para esta verificação ou eventualmente utilizar uma ferramenta externa para esse efeito e adicionar esta nova tarefa à lista de tarefas já existentes.

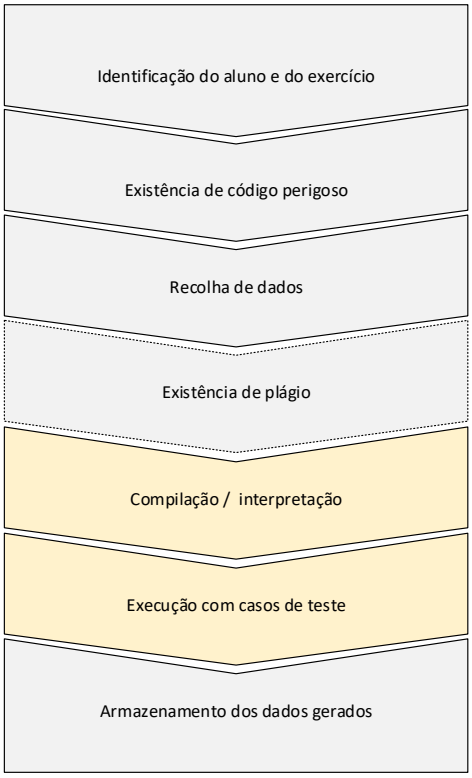

Figura 103 - As várias etapas do processamento de um *snapshot*

4.6.3.5. Ferramenta de monitorização

A nível da ferramenta de monitorização destinada aos professores, foram realizadas algumas pequenas modificações relativamente à interação com o utilizador sugeridas pelos professores que utilizaram o sistema na iteração anterior e foram igualmente desenvolvidas algumas novas visualizações. A título de exemplo, foi desenvolvida uma nova visualização que permite de uma forma muito rápida ter acesso a informação sobre o número de estudantes que conseguiram (ou não) resolver cada um dos exercícios propostos. Um exemplo desta visualização pode ser observado na Figura 104.

Nesta visualização os professores podem rapidamente saber, para cada exercício, quantos estudantes conseguiram resolver os exercícios com sucesso (verde), quantos estudantes tinham erros de compilação na última tentativa submetida (vermelho), quantos estudantes tinham uma solução potencialmente correta (amarelo) e quantos estudantes apresentavam uma solução cujo output não pode ser avaliado (azul). Era ainda possível saber quantos estudantes não haviam sequer tentado resolver o exercício (cinzento).

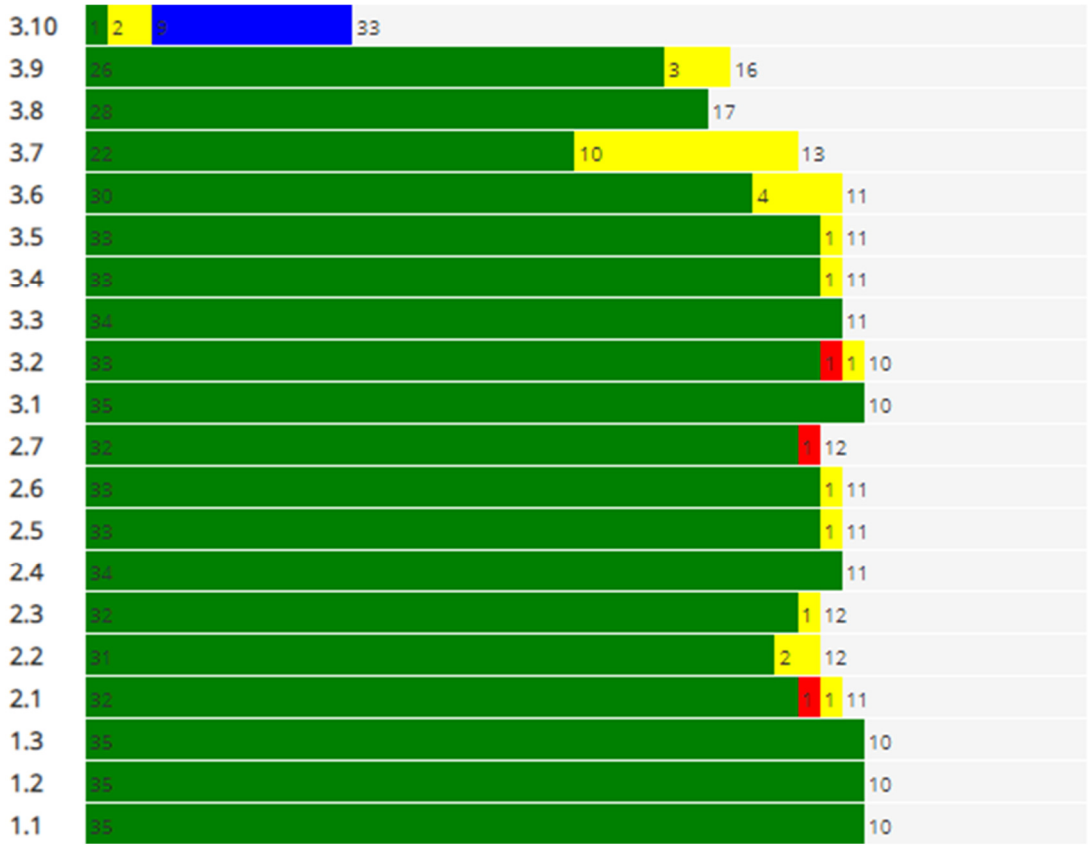

Figura 104 - Visualização do estado das últimas tentativas de resolução para cada um dos exercícios propostos.

Com base nesta informação, o professor consegue rapidamente saber quais os exercícios em que os estudantes estavam a ter mais dificuldades. Poderia, por exemplo, dar-se o caso de faltar algum elemento importante no enunciado, ou o grau de dificuldade ser demasiado elevado.

Uma outra funcionalidade que sofreu algumas modificações com base no *feedback* da iteração anterior, foi a matriz resumo. Como visto, na iteração anterior foi incluída informação sobre os resultados da execução do código com os casos de teste na forma dos símbolos "+", "-", "x" e "?". Contudo, os professores manifestaram que esta forma de apresentar a informação não era suficientemente distintiva. Assim, foi feita uma modificação na qual a informação era representada com recurso à utilização de cores diferentes. Na Figura 105 pode ser observado um exemplo da nova configuração da matriz.

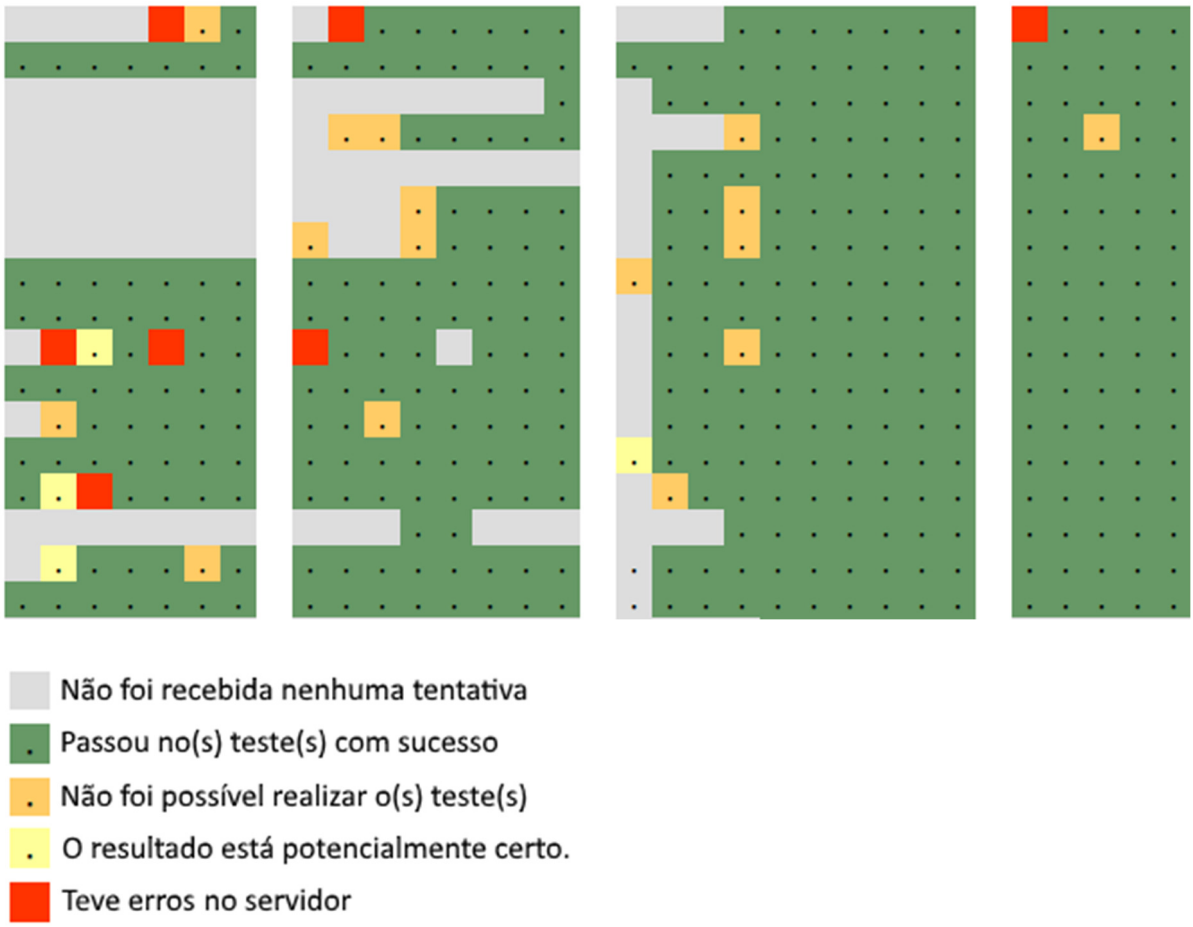

Figura 105 - Nova configuração da matriz resumo - iteração cinco

A pedido dos professores, foi ainda incluída uma legenda conjuntamente com a matriz por forma a facilitar a compreensão da informação nela contida.

4.6.4. Avaliação

Como referido anteriormente, as novas funcionalidades desenvolvidas e as modificações realizadas nesta iteração foram testadas com os estudantes de uma turma da unidade de Programação para a Internet II integrada na Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital. A experiência decorreu, oficialmente, entre os dias 3 de março e 16 de abril de 2018, contudo os *snapshots* foram recolhidos para três fichas de trabalho adicionais até ao dia 25 de maio de 2018. No total foram propostos para resolução ao longo da duração oficial da experiência aproximadamente 50 exercícios organizados em sete fichas de trabalho. Ainda que as fichas de trabalho fossem tidas em conta para efeitos de avaliação, a utilização do *plug-in* não era obrigatória, podendo os estudantes enviar aos ficheiros para o professor via email.

Estando inscritos 40 estudantes a esta unidade curricular, foram recebidos *snapshots* de apenas 27, sendo que 10 eram repetentes. No total, dentro da duração "oficial" da experiência foram recebidos aproximadamente 13.500 *snapshots* válidos (cerca de 15.000 no total). Na Figura 106 é possível observar a distribuição dos *snapshots* ao longo do tempo.

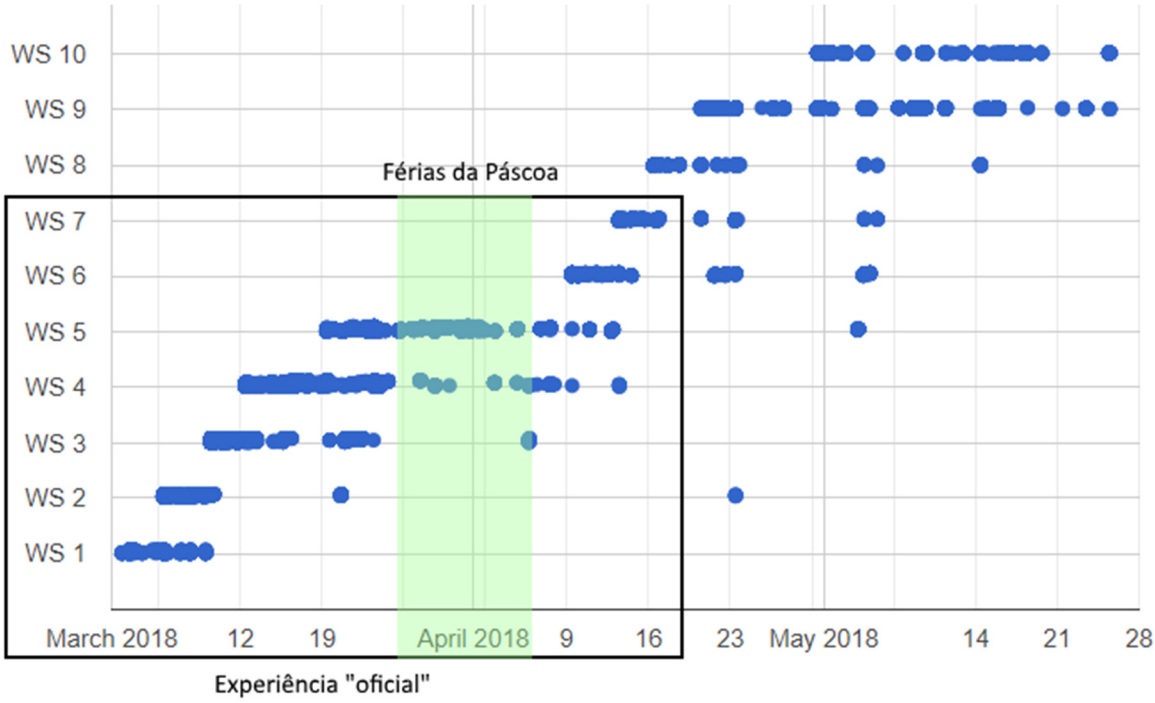

Figura 106 - Distribuição dos *snapshots* ao longo do tempo - iteração cinco

Ainda que as fichas de trabalho fossem tidas em conta para efeitos de avaliação, havendo um *deadline* que deveria ser cumprido, em todas as fichas de trabalho diversos estudantes começaram e/ou terminaram de resolver os exercícios bem para lá do *deadline* definido.

Aproximadamente 60% dos *snapshots* foram recebidos fora do período das aulas. Curiosamente, a maior parte dos estudantes com ritmos de trabalho mais elevados e que no final acabariam por obter os melhores resultados trabalharam maioritariamente durante as aulas e no período imediatamente a seguir (inclusivamente enquanto estavam em aulas de outras unidades curriculares).

Nesta experiência foi testado o sistema de deteção de plágio em tempo real, assim como as notificações a ele associadas. Para efeitos de visualização, foi utilizado um valor para *minSimmilarity = 95%*. Em termos do sistema automático de notificações foi utilizada a tabela de expectativas anteriormente apresentada. Como é possível observar na Figura 107, com o passar do tempo, e à medida em que a dificuldade dos exercícios aumentava e como tal o número de linhas de código escritas aumentava, o número de situações de possível plágio reduziu drasticamente e como tal, também o número de notificações reduziu.

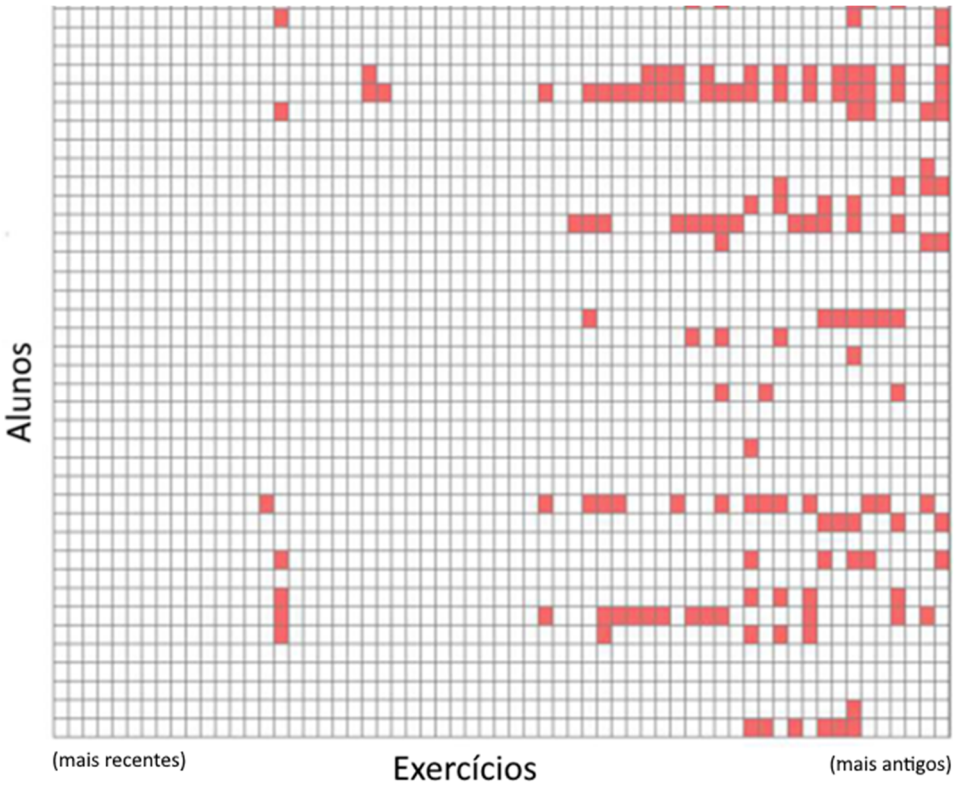

Figura 107 - Matriz com as situações em que potencialmente existia plágio.

Esta situação acaba por ser compreensível, uma vez que num exercício mais simples que possa ser resolvido com apenas cinco ou seis linhas de código, a probabilidade de existirem duas soluções muito parecidas, em que não exista plágio, é muito maior do que num exercício em que sejam necessárias trinta ou quarenta linhas de código. Idealmente, deveria ser definido um valor de *minSimmilarity* distinto para cada exercício, assim como uma tabela de expectativas diferente.

Ainda que fosse de facto possível definir valores distintos para as expectativas de cada exercício, a verdade é que tal como já tinha sido observado anteriormente, trata-se de uma tarefa que muito raramente é feita. Com efeito, para tornar o processo de geração das notificações automáticas mais simples e eficaz, os valores das expectativas deveriam ser definidos de forma automática com base em informação de anos anteriores. Contudo, ainda que esta já não fosse a primeira vez que o sistema estava a ser utilizado nesta unidade curricular, por forma a evitar a utilização de soluções dos exercícios dos anos anteriores, este professor em particular, apresenta todos os anos enunciados diferentes para a grande maioria dos exercícios, o que invalida a utilização de informação de histórico para a definição das expectativas.

Em todo o caso, no total foram geradas aproximadamente 100 notificações associadas a situações de plágio, sendo que em alguns casos, principalmente na fase inicial, se tratou de "falsos positivos".

Em todo o caso, quer as notificações, quer as visualizações constituíram uma valiosa ajuda na identificação de estudantes a plagiar. Relativamente aos resultados que advieram desta identificação, a verdade é que na maior parte dos casos, o plágio era levado a cabo por estudantes repetentes e/ou consistentemente não presentes nas aulas. Como tal, ainda que o professor não os tivesse abordado diretamente por forma a evitar situações de desconforto, tendo em vez disso enviado emails para o fórum da unidade curricular ou reforçado alguns conceitos que visivelmente não estavam totalmente assimilados por alguns estudantes, a verdade é que alguns estudantes continuaram reiteradamente com o mesmo comportamento. Convirá a este respeito, mencionar que muitos destes estudantes possuem sérias dificuldades ao nível de noções básicas de programação, pelo que o problema dificilmente poderia ser ultrapassado sem uma abordagem mais ampla e abrangente. Curiosamente, em muitas das situações, a solução que era submetida não funcionava corretamente e algumas outras continham mesmo

erros grosseiros, o que por si só é revelador das dificuldades e falta de interesse destes estudantes.

Tal como já havia sido verificado em experiências anteriores, também nesta experiência foi possível observar estudantes com reservas em ver todas as suas tentativas de resolução enviadas para o servidor. A título de exemplo, um estudante desenvolvia todo o seu código em outro ambiente de desenvolvimento, e recorria ao *Atom* com o *plug-in* instalado apenas para submeter o código que em seu entender correspondia à solução final.

Apesar da insistência dos professores em que os estudantes procedessem à sua correta identificação e dos exercícios que estavam a resolver, foram recebidos aproximadamente mil *snapshots* indevidamente identificados. Após alguma análise dos *snapshots* inválidos, foi possível concluir que, na sua grande maioria, a incorreta identificação dos *snapshots* aparentava ser intencional e não resultante de esquecimento por parte dos estudantes, uma vez que eram frequentemente utilizados para `@aid` valores introduzidos manualmente como "`x.x`" ou "`a.a`" com o objetivo claro de tornar estes *snapshots* não identificáveis. Assim que os estudantes em causa entendiam que as suas soluções estavam terminadas, procediam à sua correta identificação de modo a serem devidamente processadas pelo sistema.

Ainda que a possibilidade de a utilização do sistema poder de facto causar algum tipo de reservas ou desconforto aos estudantes já tivesse sido identificada, até ao momento esta questão não havia merecido a necessária atenção. Até então a principal preocupação tinha sido em conceber, implementar e avaliar ferramentas que permitissem fornecer aos professores a maior quantidade possível de informação relativa ao desempenho dos estudantes. Contudo, perante estas novas evidências, parecia certo que alguns estudantes não estavam dispostos a fornecer aos professores informação que de algum modo evidenciasse as suas fraquezas ou dificuldades.

Uma vez que, em última instância, os principais beneficiários da utilização do sistema eram os estudantes e que a sua participação no processo era imprescindível, considerou-se pertinente que de futuro se deveria olhar com mais atenção para os aspetos ligados à utilização do sistema pelos estudantes, nomeadamente ao nível dos eventuais receios que possam advir da sua utilização.

Até então, as notificações relativas a quantidades invulgares de tentativas de resolução de um exercício haviam sido utilizadas sempre com o objetivo de detetar quantidades

muito elevadas. Perante este cenário, seria pertinente considerar a sua utilização também para a deteção de quantidades invulgarmente baixas, sendo que, neste caso concreto, não poderia obviamente recorrer a uma notificação do tipo online, mas sim uma notificação do tipo *offline*, que inclusivamente já tinha sido implementada numa iteração anterior.

4.6.5. Sumário da Iteração cinco

Em termos gerais, esta iteração acabaria por ser muito importante pois foram consolidados e reforçados diversos aspetos ligados à utilidade do sistema para os professores, por exemplo, ao nível da introdução do mecanismo de utilização de casos de teste, que foi testado com resultados bastante satisfatórios. Além disso, ainda que não tivesse de todo sido previsto inicialmente, foram desenvolvidas e testadas ferramentas que permitem a deteção atempada (e em tempo-real) de situações de plágio, o que constitui uma fonte adicional de informação sobre eventuais dificuldades que os estudantes possam estar a sentir a dada altura.

Paralelamente a todos estes avanços positivos ao nível da informação e ferramentas disponibilizadas aos professores, tornou-se evidente que nem todos os estudantes estavam dispostos a fornecer aos professores informação, que em boa verdade poderia ser utilizada em seu benefício. Atendendo ao facto de o correto funcionamento do sistema estar totalmente dependente da participação dos estudantes, é importante que se investigue devidamente as causas que levam a que os estudantes adotem este comportamento, de modo a envidar esforços com vista à sua mitigação, sem obviamente colocar em causa o direito dos estudantes a não instalar e utilizar o sistema.

Outro aspeto que nesta iteração voltou a ficar evidente, estava relacionado com a dificuldade em utilizar o mecanismo de geração de notificações baseado em atenção seletiva. Tal como já tinha acontecido anteriormente, o sistema de geração de notificações encontrava-se totalmente operacional e na maioria dos casos, gerava notificações de facto relevantes para os professores. O problema associado à sua utilização prendia-se com o processo de especificação das expectativas. Muito embora fosse possível utilizar valores por omissão, estes eram definidos com valores genéricos e não específicos a cada exercício.

Tal como foi possível observar nas várias experiências já realizadas, ainda que tal fosse possível, os professores quase nunca forneceram valores específicos para os exercícios, o

que acaba por comprometer a assertividade das notificações geradas. Uma vez que muitas das notificações estão intimamente ligadas a exercícios concretos, pensou-se em ir recolhendo dados de cada utilização do sistema com o intuito de ir melhorando os valores *baseline* para as expectativas. Contudo, vários professores que foram utilizando o sistema e outros a quem o sistema foi apresentado, referiram que por forma a evitar situações de plágio, a maioria dos exercícios que propunham aos alunos era diferente de ano para ano.

A conjugação destes fatores levou à decisão de colocar o desenvolvimento e a utilização deste mecanismo em suspenso, e retomá-lo posteriormente caso fossem encontradas formas alternativas de proceder à definição das expectativas.

4.7. Iteração seis

4.7.1. Objetivos

Tendo em consideração o facto de em termos gerais já terem sido alcançados os objetivos que haviam sido definidos inicialmente, nomeadamente ao nível da informação disponibilizada aos professores e o facto de o sistema no seu estado atual ter sido consideravelmente bem aceite pelos professores, foi do nosso entendimento que, além de algumas correções de erros e algumas pequenas modificações pontuais no sistema, nesta iteração não se iria proceder ao desenvolvimento de novas funcionalidades. Em vez disso, no seguimento do corolário da iteração anterior, decidiu-se avaliar a perspetiva dos estudantes face à utilização do sistema. O principal objetivo passou por identificar os principais motivos de desconforto, com vista à adoção de medidas que contribuam para as ultrapassar, para que o sistema consiga de facto apoiar os professores no processo de identificação de dificuldades sentidas pelos estudantes.

4.7.2. Descrição do contexto

Nesta iteração, o sistema foi avaliado em três experiências distintas, todas elas realizadas em unidades curriculares em que o sistema já tinha sido utilizado anteriormente, nomeadamente na unidade curricular de Introdução à Programação em Java do Programa Acertar o Rumo da Universidade de Coimbra e na unidade curricular de Fundamentos de Programação da Licenciatura em Engenharia Informática de ESTGOH, no primeiro semestre do ano letivo 2018/2019. O sistema foi ainda utilizado no

segundo semestre do mesmo ano letivo na unidade curricular de Programação para a Internet II da ESTGOH.

4.7.3. Avaliação do sistema

Como mencionado anteriormente, nestas avaliações interessou principalmente avaliar/analisar o modo como os estudantes utilizavam o sistema desde a utilização (ou não) do *plug-in*, até ao modo como interagiam com a ferramenta de acompanhamento do seu trabalho. Para além disso, pretendeu-se também auscultar os estudantes no sentido de averiguar até que ponto estes se sentiam confortáveis a utilizar o sistema e se consideravam que existiam (ou não) vantagens na sua utilização.

Assim, uma vez que os professores que utilizaram o sistema já o haviam feito no passado, optou-se por não os entrevistar novamente, tendo, no entanto, ocorrido uma conversa informal com todos eles no final das várias experiências. Dado o foco principal da avaliação, no final de cada experiência, procedeu-se à realização de um inquérito aos estudantes por forma a tentar efetivamente compreender melhor a sua relação com o sistema.

À semelhança do que já tinha acontecido anteriormente foi fornecido aos estudantes um manual de instalação/configuração do *plug-in* e de utilização do sistema, com especial enfoque na questão da definição dos *inputs* (Anexo V). Foi também fornecido em versão impressa e em formato digital um guia de consulta rápida da utilização do sistema (*cheat sheet*) semelhante ao disponível no Anexo I.

Uma vez que, entretanto, tinha entrado em vigor o Regulamento de Proteção de Dados (RGPD), antes de iniciar as experiências, eram explicados devidamente aos estudantes os objetivos da utilização do sistema e uma vez que a sua participação nas experiências era voluntária, aos estudantes que pretendessem utilizar o sistema era pedido para assinarem um documento de consentimento de recolha dos seus *snapshots*. No Anexo VI é possível consultar um exemplo do documento assinado pelos estudantes. Em qualquer momento, os estudantes poderiam deixar de utilizar o sistema e caso assim entendessem, poderiam solicitar o apagamento de toda a informação entretanto recolhida a seu respeito.

4.7.3.1. Programa Acertar o Rumo (DEI-FCTUC)

Tal como mencionado anteriormente, esta experiência ocorreu num contexto menos convencional, mas que vem a ser cada vez mais comum: um curso-intensivo com a duração de um ano, destinado a indivíduos que já possuem formação superior em áreas diversas da informática. Tal como nas edições anteriores, a unidade curricular de Introdução à Programação em Java foi ministrada por quatro professores, que estavam com os estudantes em sala de aula durante 24 horas por semana. À semelhança das edições anteriores, o IDE escolhido foi o Eclipse. A experiência em si decorreu entre os dias 25 de setembro e 25 de novembro de 2018. Durante este período, foram propostos aos estudantes aproximadamente 130 exercícios organizados em oito fichas de trabalho. Participaram da experiência todos os 25 estudantes inscritos e foram recebidos mais de 54.000 *snapshots* válidos (aproximadamente 78.000 no total). Dos *snapshots* válidos, aproximadamente 22% foram recebidos fora do período das aulas (imediatamente após, mas também durante a noite e madrugada).

Como foi visto anteriormente, aquando da resolução dos exercícios, os estudantes deverão identificar o exercício em causa mediante a utilização de uma diretiva de comentário (*e.g.*, @aid 3.2). Caso não o façam, os *snapshots* para este exercício serão igualmente enviados para o servidor, embora sejam considerados inválidos (não identificáveis).

Como apresentado anteriormenta, em todas as experiências realizadas existiu sempre um número considerável de *snapshots* indevidamente identificados. Compreensivelmente, devido à novidade da utilização do sistema, no primeiro dia de utilização do sistema iam surgindo algumas dezenas de *snapshots* com problemas de identificação. Graças ao reforço da importância da correta identificação do código que os professores fizeram, o número de *snapshots* mal identificados foi diminuindo drasticamente. Contudo, sensivelmente a meio da segunda semana de aulas, voltou a ser notório o aumento do número de *snapshots* recebidos com problemas de identificação devido à falta de um identificador de exercício válido. Na altura considerou-se uma vez mais, que tal fenómeno se poderia dever ao esquecimento por parte dos estudantes e foi uma vez mais solicitado aos professores para pedirem aos estudantes para não se esquecerem de identificar devidamente o código, no entanto, o número de *snapshots* mal identificados não reduzia com o passar do tempo, o que acabaria por nos levar a analisar com maior detalhe o sucedido.

Uma vez que os *snapshots* mal identificados eram armazenados no sistema, procedeu-se a uma análise detalhada dos mesmos. Como já havia sido observado anteriormente, alguns deles correspondiam a pequenos testes que os estudantes faziam para verificar o funcionamento de uma determinada função ou pedaço de código e como tal, não faria de facto sentido estar a identificar estes ficheiros. Contudo, foi possível identificar centenas de *snaphots* associados a cinco estudantes distintos que consistentemente omitiam ou colocavam um valor inválido para o identificador do exercício (e.g., `@aid x.x, @aid ##.##`) nas suas tentativas iniciais de resolução, acabando posteriormente por inserir o valor correto, quando, no seu entender, a solução estava correta. Curiosamente, estes alunos deixavam ficar o identificador do estudante válido, o que acabaria por ajudar a detetar a situação apresentada.

Em alguns casos, antes de ser submetido um *snapshot* com identificação válida, eram submetidos 100, 200 ou mais *snapshots* com identificação inválida. Por se tratar de uma situação recorrente e consistente para aqueles estudantes, era notório que os estudantes em questão tinham objeções em expor o processo que os levou à solução final aos professores, contudo, não tinham qualquer problema em relação à solução final. Obviamente, teria sido possível abordar diretamente os estudantes em causa e perguntar-lhes quais os motivos que os levaram a adotar aquele comportamento. No entanto, optou-se por não o fazer, tendo os estudantes em causa continuado com o comportamento descrito até ao final da experiência.

Em conversa com os professores, foi possível verificar que o fenómeno descrito coincidiu com algumas intervenções feitas pelos professores na sequência da receção de notificações associadas a números de tentativas de resolução para um exercício claramente acima do esperado. Ou seja, eventualmente, naquela altura os estudantes em questão aperceberam-se que de facto o sistema recolhia a informação e que esta era disponibilizada aos professores. Um dos professores disse a este respeito que tinha sentido na altura algum desconforto por parte de alguns estudantes, ainda que não o tivessem manifestado diretamente ao professor em causa. Não deixa, no entanto, de ser curioso que no início da experiência foi dito aos estudantes que esse era de facto o propósito do sistema, ou seja, recolher informação que permita aos professores intervir junto dos estudantes.

Na experiência anteriormente realizada neste contexto, foram realizadas entrevistas com os professores em que foi possível obter informação sobre o modo como os professores

utilizavam o sistema no seu dia a dia, assim como qual a sua opinião sobre a mesma. Como já foi mencionado, os professores que utilizaram o sistema nesta experiência eram essencialmente os mesmos, pelo que foi decidido não os entrevistar novamente, mas, ainda assim, no final da experiência foi levada a cabo uma conversa informal com todos os professores. Estes indicaram que utilizavam o sistema principalmente antes do início das aulas de modo a terem uma visão global do progresso dos estudantes, mas também no seguimento da receção de notificações.

Perante a forma como os professores utilizavam o sistema, o facto de não terem acesso a todas as tentativas de resolução efetuadas por todos os estudantes não constituía à partida um problema, uma vez que estes estavam principalmente interessados numa visão de "olho de pássaro". No entanto, tal como um professor notou, a dada altura, o número de notificações automáticas geradas pelo sistema diminuiu significativamente. Obviamente, não tendo acesso a todos os dados sobre todos os estudantes, não será possível continuar a fornecer o mesmo tipo de informação aos professores.

Todos os professores indicaram que, embora o tenham feito algumas vezes, não era frequente utilizarem o sistema para obter informação que lhes permitisse suportar intervenções durante o período que estavam com os estudantes nas aulas. Convirá ainda referir que os professores mencionaram que na medida do possível, as intervenções que levaram a cabo, foram feitas de modo mais abrangente e não individualizado, ou seja, tentaram não abordar diretamente um estudante específico, mas sim dando indicações genéricas que eventualmente permitiriam ajudar o estudante em causa ultrapassar as suas dificuldades. Em grande medida, as intervenções ocorriam deste modo por forma a não tornar tão evidente o nível de detalhe da informação à qual os professores tinham acesso e que como foi visto anteriormente levou a que vários estudantes simplesmente optassem por ocultar o processo que os levou a solução final. Como será visto adiante, esta abordagem acabaria por revelar-se importante para explicar a opinião dos estudantes sobre a utilização do sistema.

Como já foi referido, no final da experiência foi solicitado aos estudantes a resposta a um questionário sobre a experiência de utilização do sistema ***CodeInsights*** nas últimas semanas, tento sido respondido por todos os 22 estudantes. As perguntas do questionário podem ser encontradas no Anexo VII. Tal como a participação na experiência, também o preenchimento do questionário era voluntário, sendo as respostas dadas de forma totalmente anónima. Em seguida é apresentado um resumo das respostas obtidas.

Quando questionados sobre se consideravam relevante a utilização deste tipo de ferramentas no apoio ao processo de ensino/aprendizagem, todos os 22 estudantes responderam afirmativamente. Ainda assim, alguns estudantes mencionaram que a utilização deste tipo de ferramentas é relevante, contudo, deverão de facto ser utilizadas para auxiliar os estudantes.

Em seguida, era perguntado aos estudantes se ao longo da duração da experiência tinham sentido que os professores estavam efetivamente a utilizar o sistema na preparação das aulas e/ou para suportar os estudantes nas aulas. A esta questão, responderam afirmativamente sete estudantes (32%), o que significa que a grande maioria dos estudantes não sentiu no seu dia-a-dia reais benefícios da utilização do sistema. Todos os sete estudantes que responderam consideraram as intervenções feitas pelos professores importantes.

Foi ainda pedido aos estudantes para apresentarem alguns exemplos de utilizações que notaram no seu dia-a-dia. Vários estudantes mencionaram que por vezes se conseguiam aperceber que os professores estavam a utilizar (ou tinham utilizado) o sistema, mas não sentiam no seu dia-a-dia muitos benefícios concretos da sua utilização. Alguns estudantes indicaram que algumas vezes sentiram que existiram ajustes no ritmo de apresentação da informação por parte dos professores. Dois estudantes mencionaram o acompanhamento mais próximo de estudantes que estivessem mais atrasados do que os colegas. Muitos estudantes indicaram ainda que por vezes os professores identificavam erros comuns a vários estudantes e procediam ao seu esclarecimento em grupo.

Uma vez que já havia sido detetado o fenómeno da invalidação voluntária das submissões, perguntámos aos estudantes se ao longo da experiência sentiam algum tipo de desconforto com o facto de todas as submissões estarem a ser enviadas para o servidor, tendo seis estudantes (27%) respondido afirmativamente.

Perguntámos em seguida aos estudantes se, por forma a ultrapassar o desconforto sentido, tinham adotado algum tipo de medidas. Como já seria de prever, cinco estudantes responderam afirmativamente, indicando que procediam à utilização de identificadores de exercício inválidos até terem uma solução estável. Curiosamente, quatro destes estudantes responderam anteriormente que não tinham sentido que o sistema era de facto utilizado pelos professores. Certamente existirá uma relação entre estes dois factos. Se por um lado é possível argumentar que estes estudantes passaram a invalidar as suas submissões devido a não receberem *feedback* dos professores, será

também possível argumentar que estes estudantes não receberam *feedback* devido a estarem a invalidar as suas submissões. Em todo o caso, ficou confirmado pelos estudantes que existia de facto uma situação de desconforto associada ao envio de todas as submissões. Quanto questionados sobre os motivos deste desconforto, nenhum estudante respondeu.

Tendo em consideração que este desconforto existe de facto para alguns estudantes, questionámos em seguida se o facto de os estudantes poderem escolher quando as submissões eram enviadas para o servidor, ajudaria a mitigar o desconforto sentido, tendo 12 estudantes (54%) respondido afirmativamente. Como seria de esperar, entre estes 12 estudantes, encontravam-se todos os estudantes que haviam respondido anteriormente que sentiam desconforto e adotaram medidas.

Em seguida questionámos os estudantes relativamente à regularidade com que utilizavam a ferramenta de monitorização do seu trabalho. Os resultados estão representados na Figura 108.

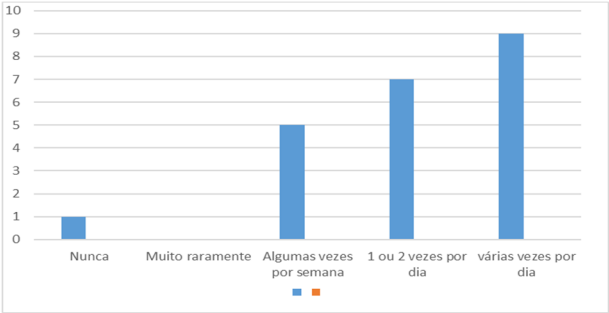

Figura 108 - Frequência de utilização da ferramenta de monitorização destinada aos estudantes - Programa Acertar o Rumo - iteração seis.

Ao tratar-se de um curso intensivo, não é de estranhar que a maior parte dos estudantes tivessem acompanhado o seu percurso na ferramenta disponibilizada para o efeito. Três dos estudantes que faziam submissões inválidas responderam que acediam à ferramenta mais de duas vezes por dia, um deles respondeu uma ou duas vezes por dia e, finalmente, o último disse que acedia à ferramenta várias vezes por semana. Este aspeto é curioso e importante, uma vez que parece indicar que estes estudantes se preocupavam de facto em garantir que todas as suas submissões eram registadas, ou seja, queriam apresentar trabalho feito aos professores, mas não pretendiam de modo algum expor o processo que os levou às soluções finais. Em seguida, foi perguntado aos estudantes se consideraram importante a utilização de uma ferramenta que permita o acompanhamento do trabalho por si desenvolvido. Todos os estudantes responderam afirmativamente.

Na questão seguinte era solicitado aos estudantes que indicassem sugestões para a melhoria da ferramenta. A grande maioria dos estudantes indicou que gostaria de receber *feedback* mais detalhado sobre as soluções desenvolvidas, ao invés de indicar apenas se a solução estava certa ou errada, indicando, por exemplo, formas de melhorar a solução encontrada.

Finalmente, foi pedido aos estudantes para fornecerem críticas e sugestões gerais que possam contribuir para a melhoria de todo o sistema. A grande maioria das respostas foi no sentido de que o sistema deveria ser mais utilizado pelos professores. Transcrevem-se em seguida algumas respostas nesse sentido: "Muito importante na fase inicial. Os profs usavam inicialmente", "Os professores poderiam dar de facto uso à aplicação!", "Fica a ideia de que os profs de facto não usam a ferramenta", "Esperava maior feedback por parte dos profs. […]", "Gostaria de receber feedback dos professores", "Gostaria de receber feedback dos professores!". Um dos estudantes foi um pouco mais longe e indicou que seria interessante se os estudantes conseguissem saber se os professores acediam ao sistema e quando o faziam.

Um aspeto muito importante a reter das respostas dos estudantes prende-se com o facto de a grande maioria não percecionar que o sistema estava de facto a ser utilizado pelos professores e, além disso, muitos estudantes terem mencionado que estavam à espera de receber mais *feedback* dos professores.

Em conversa com a professora responsável por esta unidade curricular sobre estes resultados, foi mencionado que tal como já havia acontecido nas edições anteriores, os professores utilizavam maioritariamente o sistema de forma a terem uma visão "macro" da performance dos estudantes. Ainda que por vezes o tenham feito, na maior parte das vezes o sistema não era utilizado enquanto os professores estavam na sala com os estudantes, uma vez que no seu entendimento, tendo em consideração que os estudantes têm tantas horas de contacto com os professores, teriam sempre inúmeras oportunidades para expor as suas dúvidas e que todos os professores estavam constantemente em contacto com os estudantes.

A utilização de uma ferramenta de monitorização do desempenho dos estudantes poderá compreensivelmente ter levado alguns estudantes a criar expectativas em relação ao tipo de assistência que iriam obter dos professores e que em certa medida acabaram por ser frustradas. Assim, se por um lado alguns estudantes não quiseram expor as suas fraquezas perante os professores, de certo modo, inviabilizando certas medidas de apoio

individualizado, muitos outros estudantes forneceram todos os dados necessários, mas não obtiveram toda a assistência que idealizaram.

Para uma utilização plena do sistema, será, pois, crucial definir à partida junto dos estudantes qual a utilização concreta que irá ser dada ao sistema por forma a não criar falsas expectativas aos estudantes, ressalvando que ainda que toda a informação esteja a ser recolhida em tempo real durante 24 horas por dia, não será que esperar que seja fornecida aos estudantes assistência também em tempo real.

4.7.3.2. Fundamentos de Programação (LEI-ESTGOH)

Como mencionado anteriormente, esta avaliação decorreu no 1º semestre do ano letivo 2018/2019 entre o dia 15 de outubro e o dia 30 de novembro. Tal como havia acontecido na edição anterior, a escolha do ambiente de desenvolvimento manteve-se, sendo, pois, utilizado o Atom para a resolução dos exercícios propostos. No total, estavam inscritos 47 estudantes. Contudo, ao longo da duração da experiência, foram apenas recebidos *snapshots* de 35. Durante a duração da experiência foram recebidos aproximadamente 40.000 *snapshots* válidos (aproximadamente 50.000 no total) respeitantes a 83 exercícios organizados em sete fichas de trabalho. Aproximadamente 11% do total de *snapshots* foram recebidos fora do período das aulas.

Tal como aconteceu em relação à experiência anteriormente descrita, no final, foi tida uma conversa informal com o professor. Este disse que estava uma vez mais satisfeito com a utilização do sistema, uma vez que lhe dava a possibilidade ter uma visão única da performance e progresso dos estudantes, o que lhe permitiu fazer diversas intervenções individualizadas junto deles. Contrariamente aos professores da outra experiência, este professor mencionou que utilizava o sistema maioritariamente durante as aulas, em especial a ferramenta de *live-feed* que lhe permitia ter uma visão global da performance de todos os estudantes em tempo real, permitindo deste modo identificar situações para as quais a sua intervenção fosse pertinente.

Este professor referiu que a dada altura, na sequência de algumas intervenções, alguns estudantes perguntaram se ele tinha mesmo acesso à informação em tempo real. Perante a resposta afirmativa do professor, sentiu algum desconforto por parte de alguns estudantes. Com efeito, nesta experiência foram identificados sete estudantes que manifestaram o mesmo comportamento de evitar voluntariamente a correta receção de todos os *snapshots* no servidor.

Após finalizada a experiência, procedeu-se à realização de um questionário sobre a utilização do sistema ***CodeInsights*** nas aulas, ao qual responderam 22 estudantes. No essencial, as perguntas do questionário realizado são semelhantes às realizadas no âmbito do Programa Acertar o Rumo (Anexo VII).

Em relação à primeira questão sobre a utilidade de sistemas de monitorização por parte dos professores, todos os estudantes responderam que consideravam estes sistemas úteis.

No que diz respeito à segunda questão sobre a visibilidade da utilização do sistema pelo professor para suportar as intervenções e/ou preparar as aulas, aproximadamente 41% dos estudantes (nove) responderam afirmativamente. Convirá eventualmente mencionar o facto de esta unidade curricular ter níveis consideráveis de absentismo às aulas, pelo que é possível que tenham respondido a este inquérito estudantes que não tenham assistido frequentemente às aulas. Como exemplos concretos de intervenções, os estudantes indicaram principalmente a deteção e esclarecimento de erros frequentes.

Quando questionados sobre a pertinência das intervenções feitas pelo professor, todos os nove estudantes responderam que consideraram as intervenções importantes.

Seguidamente, os estudantes eram questionados sobre o facto de sentirem algum desconforto por terem todas as suas tentativas de resolução enviadas para um servidor e disponibilizadas ao professor. A esta questão responderam afirmativamente nove estudantes. Quando questionados sobre as causas deste desconforto, alguns estudantes mencionaram o facto de serem enviadas para o servidor muitas soluções completamente erradas ou incompletas o que poderá dar ao professor uma falsa noção sobre a prestação do estudante.

Na questão seguinte era perguntado aos estudantes quais as medidas adotadas para ultrapassar o desconforto sentido. Todos os nove estudantes mencionaram a submissão com *id* de exercício inválido até ter uma solução estável. Um estudante mencionou também que utilizava outro ambiente de desenvolvimento até ter a solução estável.

Quando questionados sobre se gostariam de poder decidir quais as submissões que devem ser enviadas para o servidor, 72% dos estudantes (16) responderam afirmativamente.

Em seguida, foi perguntado aos estudantes com que frequência acediam à ferramenta que lhes permitia monitorizar o seu trabalho. Os resultados podem ser vistos na Figura 109.

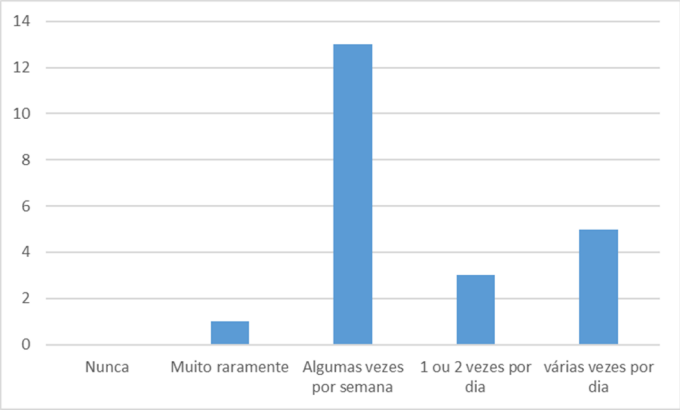

Figura 109 - Frequência de utilização da ferramenta de monitorização destinada aos estudantes - FP LEI ESTGOH - iteração seis.

Pelo facto de se tratar de uma unidade curricular com duas aulas por semana, em que o volume de trabalho realizado fora da sala de aulas não era muito significativo, não será de estranhar que a grande maioria dos estudantes tivesse respondido que acedia à ferramenta "algumas vezes por semana". Eventualmente, alguns dos estudantes que escolheram as outras opções estariam a reportar-se aos dias em que tinham aulas. Em todo o caso, é notório que os estudantes acediam ao sistema com alguma frequência.

Todos os estudantes responderam que consideravam importante a disponibilização deste tipo de ferramentas aos estudantes. Relativamente a sugestões de melhorias da ferramenta destinada aos estudantes, alguns estudantes mencionaram que a ferramenta poderia indicar formas de corrigir os erros. Um estudante fez referência a deixar de ser necessário definir os *inputs* entre as diretivas `//begin_inputs` e `//end_inputs`.

Finalmente, relativamente a críticas e/ou sugestões sobre a globalidade do sistema, três estudantes mencionaram que principalmente no início, por vezes alguns exercícios eram assinalados como estando errados, mesmo que localmente o resultado estivesse certo, tendo, contudo, admitido que tal se devia a não terem definido corretamente os *inputs*. Esta questão é obviamente muito importante e deverá merecer a atenção dos professores, pois apesar de todos os exemplos e documentação que foi dada aos estudantes, aparentemente alguns ainda sentiram dificuldade com este processo.

4.7.3.3. Programação para a Internet II (LEI-ESTGOH)

Esta experiência decorreu entre os dias 21 de fevereiro e 11 de maio de 2019, no âmbito de uma unidade curricular de introdução ao desenvolvimento para a internet *server-side*, na qual é utilizada a linguagem PHP. Ainda que a resolução dos exercícios fosse contabilizada para efeitos de avaliação, a utilização do sistema ***CodeInsights*** era voluntária e os estudantes poderiam optar por enviar as resoluções para o professor por email. No final, todos os estudantes optaram por utilizar o sistema ***CodeInsights***.

À semelhança das outras experiências, no início foi devidamente explicado aos estudantes qual o objetivo do sistema assim como qual a informação que era recolhida e quais as utilizações que seriam dadas a essa informação.

Estavam inscritos nesta unidade curricular 45 estudantes, tendo sido recebida pelo menos uma submissão de um total de 37 deles. Durante toda a experiência, foram fornecidas aos estudantes dez fichas de trabalho com 47 exercícios.

Durante a experiência foram recebidos aproximadamente 38800 *snapshots* válidos (aproximadamente 40000 no total). Aproximadamente 10450 *snapshots* apresentavam erros, dos quais 42,2% correspondiam a erros de sintaxe e os restantes a erros de *runtime* diversos como a falta do símbolo "$" antes do nome das variáveis ou acesso a posições inexistentes em *arrays*.

O tópico em que os estudantes tiveram mais dificuldades foi a manipulação de *strings*, seguida da manipulação de *arrays*.

Os estudantes tinham seis horas de contacto semanal com o professor em contexto de aula. Ainda assim, 57.5% dos *snapshots* recebidos foram realizados fora do período das aulas. Tal aconteceu devido a diversos fatores: por um lado, tratava-se de uma unidade curricular com um número elevado de estudantes repetentes e que não assistiam às aulas, mas que ainda assim não deixavam de trabalhar. Para além disso, era assumido que muitos dos exercícios propostos deveriam ser encarados como trabalho de casa, não sendo, pois, de esperar que fossem todos realizados durante as aulas. Na Figura 110 é possível observar a distribuição dos *snapshots* de cada estudante (eixo dos X) ao longo das 24 horas do dia (eixo dos Y).

Nesta figura é visível que muito embora os estudantes apenas tivessem aulas das 8h30 às 11h30 num dia e das 10h30 às 13h30 em outro dia, foram recebidos *snapshots* ao

longo de praticamente todas as horas do dia, incluindo um número considerável de *snapshots* recebidos durante a madrugada.

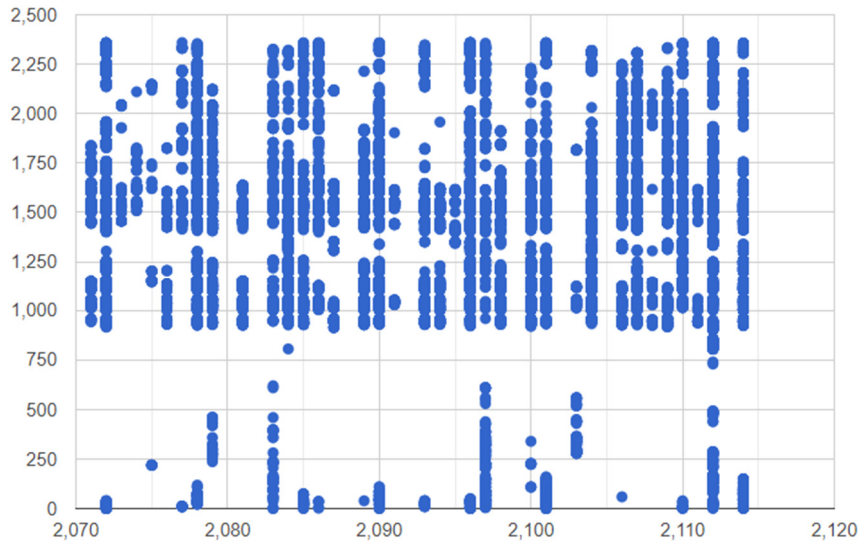

Figura 110 - Distribuição dos snapshots recebidos ao longo das horas do dia - PI2 ESTGOH - iteração seis.

Com a informação disponibilizada é ainda possível constatar que muito embora os estudantes apenas tivessem aulas dois dias por semana, foram recebidos *snapshots* em todos os dias da semana, tal como representado na Figura 111.

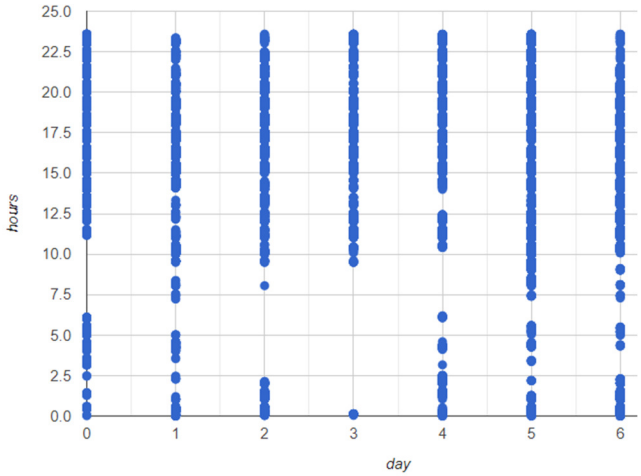

Figura 111 - Distribuição dos snapshots recebidos para os vários dias da semana - PI2 ESTGOH - iteração seis.

Pelo exposto, a utilização do sistema neste contexto acabaria por revelar-se importante, pois, mesmo que os estudantes não assistissem às aulas, o professor poderia ainda assim acompanhar o seu trabalho e intervir sempre que considerasse necessário mediante a utilização da funcionalidade de envio de *feedback* diretamente no sistema.

O professor, utilizou as várias ferramentas disponibilizadas pelo sistema enquanto estava com os estudantes na sala de aulas (*dashboard*, matriz, *live-feed*, etc.), mas também fora deste período. Quer no âmbito das aulas, quer fora deste, o professor fez diversas intervenções de caráter geral para a turma (e.g. dar mais exemplos; assinalar erros frequentes; dilatar o prazo para os estudantes terminarem as fichas de trabalho) e diversas intervenções individuais, quer diretamente junto dos estudantes, quer via email, utilizando o sistema (e.g. assinalar erros recorrentes de estudantes; ajudar a desbloquear erros que impediam a progressão dos estudantes).

Curiosamente, o fenómeno de ocultação do trabalho realizado previamente observado nas outras duas experiências não teve uma incidência tão relevante. Com efeito, após a análise dos *snapshots* inválidos, apenas foi possível identificar *snapshots* deliberadamente invalidados provenientes de dois estudantes.

Apesar da assistência prestada pelo professor, existiram sempre estudantes a trabalhar com ritmos bem distintos, como é bem visível na Figura 112. Aqui, cada linha (eixo dos Y) corresponde a uma ficha de trabalho (*Work Sheet*) e, no eixo dos X, está representada a passagem do tempo, sendo que cada ponto corresponde a uma *snapshot* recebido.

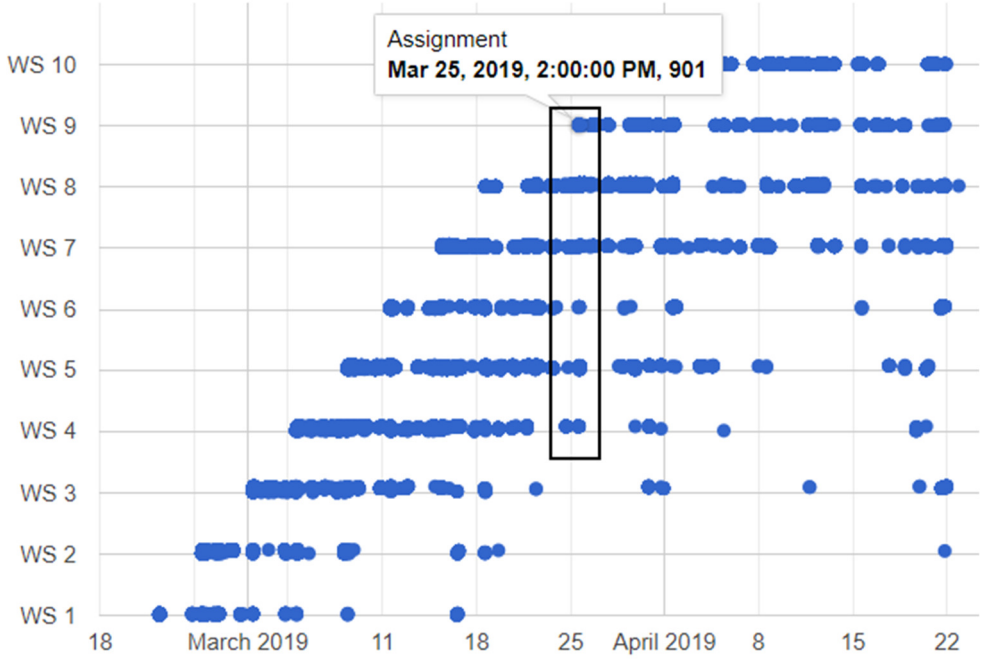

Figura 112 - Distribuição dos *snapshots* recebidos ao longo do tempo para cada ficha de trabalho - PI2 / ESTGOH - iteração seis.

A título de exemplo, na situação destacada na Figura 112, enquanto alguns estudantes começaram a resolver exercícios da ficha nº 9, muitos outros encontram-se ainda a

resolver exercícios das fichas nº 8, 7,6, 5 e 4. Por um lado, tais diferenças poderiam ser explicadas pelo facto de existirem diversos estudantes (principalmente repetentes) que não assistiam às aulas e que, não querendo desistir, procuravam resolver os exercícios quando tinham oportunidade. Em muitos outros casos, embora os estudantes pudessem não ser repetentes, não haviam feito algumas unidades curriculares de base, o que constituía um desafio extra para tentarem acompanhar o ritmo dos colegas. Em qualquer dos casos, foi considerada relevante a disponibilização desta informação ao professor, por forma a que este possa ajustar o ritmo de ensino e/ou adotar outras medidas que considere importantes, tendo sempre em consideração que muito dificilmente será possível harmonizar totalmente os ritmos de trabalho dos estudantes.

Finda a experiência, procedeu-se à realização do questionário aos estudantes. A estrutura do questionário era muito semelhante à das anteriores experiências, tendo sido mantidas todas as questões previamente realizadas e acrescentadas mais algumas. Para além disso, ao invés de ter sido distribuído aos estudantes em formato físico de papel, neste caso, o formulário foi disponibilizado em formato digital na plataforma *Google Forms*. À semelhança das restantes experiências, as respostas eram dadas de forma anónima. As perguntas do questionário fornecido aos estudantes nesta experiência podem ser consultadas no Anexo VIII. O questionário foi enviado por email para os 45 estudantes inscritos, tendo sido preenchido por 18. Como já foi dito anteriormente, existiam muitos estudantes com dificuldades de base ao nível de competências de programação, pelo que este número de estudantes acabou por corresponder aproximadamente ao número de estudantes que não haviam desistido, entretanto.

Relativamente à primeira questão sobre a importância deste tipo de sistemas para os professores monitorizarem o desempenho dos estudantes, todos responderam afirmativamente. Em seguida foi perguntado aos estudantes se já tinham utilizado o sistema ***CodeInsights*** anteriormente. Apenas um estudante respondeu negativamente.

Na questão seguinte foi perguntado aos estudantes sobre a regularidade com que assistiam às aulas. Responderam a esta questão "sempre" sete estudantes (38,9%), "quase sempre" dez estudantes (55,6%) e "raramente" um estudante (5,6%).

Em seguida foi perguntado aos estudantes se ao longo das aulas verificaram que o professor utilizava realmente o sistema para preparar as aulas e para dar assistência aos estudantes. Responderam afirmativamente 15 estudantes (83,3%) e negativamente três estudantes (16,7%). Um dos estudantes que respondeu negativamente havia

respondido que assistia às aulas "raramente", os restantes dois haviam respondido "quase sempre". Ainda assim, é importante verificar que a grande maioria sentiu de facto que o professor utilizava o sistema para prestar assistência aos estudantes.

Em relação a exemplos concretos de utilizações que os estudantes percecionaram que o professor dava ao sistema, os estudantes indicaram: "O professor dirigia-se a nós para resolver questões que não lhe tínhamos colocado", "Quando fazia exercícios das fichas durante a aula e o professor advertia para eventuais erros no código", "Ao ver que os estudantes têm certas dificuldades em certas matérias lecionadas, o professor tenta direcionar a aula para esse tipo de matéria, de forma a esclarecer a maioria dos estudantes.", "Erros comuns por parte dos estudantes" ou "Se o professor verificasse que o estudante tinha algo errado no código, ia ter ao lugar do estudante para o auxiliar.".

Na continuação das respostas anteriores, foi perguntado aos estudantes se tinham considerado as intervenções do professor relevantes. Quinze estudantes (83,3%) responderam "sim", dois estudantes (11,1%) responderam "por vezes" e um estudante (5,6%) respondeu "não". Curiosamente, o estudante que respondeu "não" foi o estudante que anteriormente tinha respondido que tinha assistido às aulas "raramente". Um dos estudantes que respondeu "por vezes" justificou mais adiante o motivo da sua resposta: "às vezes o professor intervém enquanto o estudante procura uma solução". O professor confirmou que de facto esta situação aconteceu algumas vezes no caso de estudantes que estavam muito atrasados face aos colegas, tendo por vezes algumas intervenções perante estes estudantes sido feitas no sentido de permitir acelerar um o seu ritmo, por forma a uniformizar o ritmo de trabalho dos estudantes. Em todo o caso, é uma observação pertinente do estudante e que deverá ser acautelada em utilizações futuras do sistema.

Seguidamente era perguntado aos estudantes se se tinham sentido desconfortáveis com o facto de o sistema fazer a recolha automática do seu código e o professor ter acesso imediato a esse código. Responderam afirmativamente seis estudantes (33,33%) e negativamente os restantes 12 estudantes (66,7%). Tal como nas experiências anteriores, questionámos em seguida os estudantes sobre os motivos para este desconforto. Foram obtidas duas respostas: "Baseia-se no facto de por vezes o código não estar acabado e a submissão já foi realizada" e "Sentia-se alguma pressão em realizar o exercício corretamente, apesar de saber que é para o nosso bem". Ou seja, uma vez mais alguns estudantes demonstraram algumas reservas em que todas as suas tentativas fossem

disponibilizadas ao professor, eventualmente por poderem transmitir uma ideia errada das reais capacidades do estudante.

Na sequência das perguntas anteriores, foi perguntado aos estudantes se de modo a contornar o desconforto sentido, adotaram alguma medida. Dois estudantes responderam afirmativamente. Um deles indicou que fazia as submissões de forma "anónima" até ter uma solução estável e o outro indicou que utilizava outro IDE até ter uma solução estável. Não sendo a utilização do *plug-in* obrigatória, estes resultados acabam por não ser muito significativos já que mesmo que sentindo algum desconforto, a esmagadora maioria dos estudantes continuou a utilizar o sistema normalmente.

Em seguida foi perguntado aos estudantes se se sentiriam mais confortáveis se pudessem decidir quando o código deve ou não ser enviado para o servidor. Para esta questão foram recebidas apenas 17 respostas, sendo quatro delas negativas (23,5%) e as restantes 13 (76,5%) positivas. A respeito desta pergunta convirá notar que todos os estudantes que anteriormente responderam que haviam sentido algum desconforto, responderam que se sentiriam mais confortáveis se pudessem decidir quando os *snapshots* seriam enviados para o servidor. Os estudantes que responderam negativamente, eventualmente fizeram-no com o intuito de demonstrar que não se sentiam desconfortáveis e como tal, o facto de poderem decidir não iria alterar nada. A formulação desta pergunta poderia ter sido melhorada uma vez que pode induzir os estudantes em erro. Em todo o caso, o essencial que há a reter é que de facto ao poderem decidir quando os *snapshots* serão enviados para o servidor, a situação de desconforto seria mitigada.

Em seguida foram feitas duas questões sobre o ambiente de desenvolvimento utilizado (*Atom*). Na primeira questão foi perguntado aos estudantes se consideravam que o facto de os exercícios serem automaticamente corrigidos no *Atom* ajudou no seu processo de aprendizagem. Responderam afirmativamente 17 estudantes (94,4%) e negativamente um estudante (5,6%). Foi ainda perguntado aos estudantes que consideravam importante que fossem apresentados os *inputs* e *outputs* utilizados nos testes automáticos, ao invés de dizer apenas se estava certo ou errado. Todos os 18 estudantes responderam afirmativamente. Foi ainda perguntado se tinham alguma sugestão de melhoria para esta componente do sistema, não tendo sido registada nenhuma resposta.

Na sequência, foram feitas algumas questões aos estudantes sobre a ferramenta que lhes permitia acompanhar o seu trabalho. Relativamente à regularidade com que os estudantes acederam à ferramenta, as respostas podem ser observadas na Figura 113.

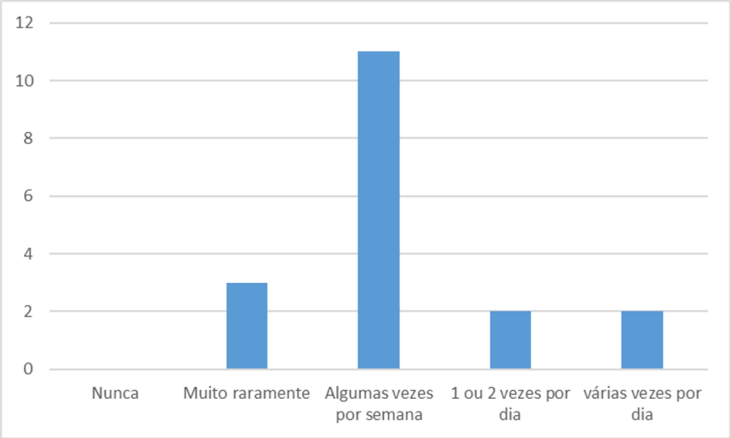

Figura 113 - Frequência de utilização da ferramenta de monitorização destinada aos estudantes - PI2 LEI ESTGOH - iteração seis.

Também nesta unidade curricular, os estudantes só tinham dois momentos de contacto com o professor (duas aulas de três horas cada), pelo que será de esperar que os resultados digam respeito principalmente aos dias em que tinham aulas.

Seguidamente, perguntámos aos estudantes se consideram importante a disponibilização deste tipo de ferramenta que permita aos estudantes ter uma ideia do trabalho efetuado. Todos os 18 estudantes responderam afirmativamente.

Relativamente à ferramenta de acompanhamento, foi finalmente perguntado aos estudantes se tinham alguma crítica ou sugestão a fazer. Foram feitas duas sugestões interessantes: "dar outra cor ao azul na solução" e "a ferramenta poderia mostrar uma espécie de *placeholder* para os exercícios a completar. Um pouco como os espaços em branco de uma "caderneta de cromos". Assim, poderíamos facilmente ter a certeza de que completamos todos os exercícios que deveríamos ter feito.". Por se tratar de sugestões pertinentes e de fácil implementação, foram de imediato implementadas e disponibilizadas aos estudantes. Ambas as sugestões são relativas à apresentação da informação sobre os exercícios realizados, assim como sobre o seu resultado. A solução modificada pode ser observada na Figura 114.

Ambas as situações já haviam sido identificadas anteriormente, contudo, acabaram sempre por ter prioridade baixa. No caso da primeira sugestão, manteve-se a cor azul, no entanto, ao invés de serem de simplesmente utilizados quadrados coloridos, passaram a

ser utilizados ícones, permitindo deste modo melhorar a usabilidade e acessibilidade desta funcionalidade. A segunda sugestão diz respeito à inclusão de informação sobre os exercícios para os quais não foi recebida nenhuma tentativa de resolução (no exemplo, 1.0 e 3.10). Até então, estes exercícios não apareciam na listagem, tendo os estudantes e professores (já que uma listagem semelhante a esta também está disponível para eles) que identificar os exercícios em falta.

Figura 114 - Listagem de exercícios resolvidos por um estudante - iteração seis.

Em seguida, foi pedido aos estudantes para avaliarem em termos gerais a utilização do sistema ***CodeInsights*** naquela unidade curricular numa escala de 1 a 5. Seis estudantes classificaram a utilização do sistema com a classificação de 5 e os restantes 12 com a classificação de 4, o que em termos globais dá um valor médio de 4,5.

Finalmente, para terminar o questionário, foi pedido aos estudantes para fazerem alguma crítica ou sugestão mais genérica sobre o sistema ou sobre a utilização do mesmo. Foram apenas registadas duas respostas: "Muito bom" e "Utilizar o mesmo método de ensino para PI1 de forma que o HTML, CSS e especialmente *JavaScript*, de certa forma sejam igualmente praticados como em PI2". Ainda que não seja propriamente uma crítica ou sugestão concreta sobre o sistema, acaba por resumir um pouco a ideia de que no geral os estudantes gostaram de utilizar o sistema e tal como já haviam manifestado oralmente aos professores, gostariam de o ver a ser utilizado em outras unidades curriculares. No caso concreto da unidade curricular indicada, PI1 (Programação para a Internet I), a verdade é que o sistema já foi de facto utilizado no passado, contudo, devido a uma série de fatores acabou por não ser utilizado no presente ano letivo. Esta observação do/a estudante/a poderá eventualmente convencer o professor a ponderar utilizar o sistema novamente, caso estejam reunidas as condições necessárias.

4.7.5. Sumário da iteração seis

Esta iteração teve a particularidade de ter sido colocado o foco na auscultação da opinião dos estudantes, sem descurar os professores. Foram, pois, levadas a cabo três experiências distintas em três contextos igualmente diversos nas quais participaram professores com abordagens diferentes em relação à utilização do sistema. A diversidade ao nível de contexto e da abordagem seguida pelos professores acabaria por se revelar interessante posteriormente ao nível da reação dos estudantes.

Em todas as experiências, existiu um número considerável de estudantes a demonstrar algum desconforto por terem todas as submissões enviadas para um servidor e disponibilizadas aos professores, tendo inclusivamente adotado medidas para contornar a situação. Foi, no entanto, curioso verificar que em nenhuma das experiências os estudantes procederam à remoção do *plug-in*, ainda que a sua utilização fosse voluntária. Esta situação prende-se certamente com o facto de os estudantes pretenderem mostrar trabalho feito, existindo, no entanto, alguns estudantes com reservas em partilhar o processo seguido para atingir a solução final.

Como revelado nos questionários, assim como em algumas conversas informais com os professores, tal desconforto prende-se principalmente pelo facto de se estar a partilhar uma solução inacabada e que como tal poderá eventualmente conter erros grosseiros. Igualmente importante é o facto de por vezes poderem ser feitas algumas centenas de tentativas até obter uma solução estável. Em ambos os casos, do ponto de vista dos estudantes, estas situações poderão levar os professores a criar uma imagem errada do trabalho dos estudantes.

A totalidade dos estudantes que manifestaram algum desconforto, revelaram que gostariam de poder decidir quais as submissões a enviar para o servidor. Eventualmente devido à natureza do curso e/ou da maturidade dos estudantes, este número foi significativamente inferior no caso da primeira experiência. Ainda que esta atitude de alguns estudantes (não a maioria) seja compreensível, será igualmente de esperar que os professores tenham o discernimento para diferenciar uma solução inacabada cuja resolução esteja ainda a decorrer de uma outra inacabada ou com problemas, mas em relação à qual o estudante já não vá fazer mais nada. Além de que não será de esperar que o professor vá verificar todos os *snapshots* recebidos. Este é claramente um aspeto que, sendo pertinente, sai um pouco fora do âmbito deste trabalho. Poderão eventualmente ser dadas algumas indicações aos professores sobre o modo como devem

lidar com a informação disponibilizada pelo sistema, mas em última instância, a decisão sobre quais as medidas a adotar caberão sempre aos professores.

Um outro aspeto importante prende-se com o facto de os estudantes da primeira experiência terem manifestado que esperavam ter recebido muito mais *feedback* por parte dos professores. Como foi referido anteriormente, esta situação decorreu da abordagem de utilização do sistema seguida pelos professores. Ao utilizarem o sistema maioritariamente fora das aulas para terem uma visão global do progresso dos estudantes, os professores acabaram por transmitir a ideia de que de facto não o utilizavam. A professora responsável argumentou que ao estarem tantas horas com os professores, os estudantes tinham tido inúmeras oportunidades para exporem as suas dúvidas. No entanto, o grande problema é que a grande maioria dos estudantes tem de facto dificuldade/reservas em colocar questões aos professores devido a fatores como a inibição ou o medo de colocar questões demasiado elementares ou eventualmente ridículas. Com a utilização do sistema, alguns estudantes desta experiência criaram a expectativa de que esse receio de expor as suas fraquezas seria mitigado e que seria o próprio professor a identificar todas as suas dificuldades sem terem a necessidade de se expor perante os colegas. Como visto, este processo de acompanhamento mais próximo com fornecimento de assistência mais personalizada acabou por não ocorrer sempre de forma sistemática, o que acabou por frustrar as expectativas de alguns estudantes.

Convirá a este respeito referir que com a utilização do sistema não se pretende de modo algum anular a capacidade que os estudantes deverão ter em procurar soluções para os seus problemas, isto é, não será de esperar que pelo simples facto de um estudante ter um erro de compilação o professor vá de imediato verificar junto do estudante como tal erro poderá ser ultrapassado. Uma vez mais, caberá ao professor saber quando a sua intervenção é pertinente (por exemplo pelo facto de ser notório que o estudante já tentou *várias* alternativas e existe algum pormenor que lhe possa estar a escapar) ou quando deve ser dado espaço ao estudante para tentar ele próprio encontrar a situação (ainda que mais tarde possa vir a ser necessária a intervenção do professor). De igual modo, também não se pretende com este sistema que os estudantes percam (ou deixem de ganhar) a capacidade para colocar questões aos professores na sala de aula ou fora desta. Ao invés, o sistema deverá ser encarado como uma ferramenta que permita ao professor dar uma resposta (ou fazer uma intervenção proactiva) mais fundamentada.

No lado oposto, a abordagem do professor da segunda experiência que utilizou o sistema principalmente nas aulas de modo a acompanhar em tempo real o trabalho dos

estudantes. Em termos globais, esta abordagem acabou por não se traduzir num aumento muito significativo da visibilidade das ações levadas a cabo pelo professor. Ao utilizar principalmente a *live-feed*, o professor tinha uma perspetiva muito imediatista e local do trabalho dos estudantes, o que também acabava por limitar o seu raio de ação. Curiosamente, foi nesta experiência que se verificou o maior número de estudantes a invalidar deliberadamente as submissões. Não tendo dados concretos que permitam justificar esta situação, pode-se apenas especular que se deva eventualmente ao modo como o professor abordava os estudantes com base na informação fornecida pelo sistema.

Finalmente, no caso do professor da terceira experiência, uma utilização mais abrangente e sistemática do sistema fez com que a sua utilização fosse mais percetível por parte dos estudantes. Curiosamente, ainda que o número de estudantes que invalidava deliberadamente as submissões tivesse sido o mais baixo das três experiências, o número de estudantes que se mostrou desconfortável com o envio automático dos *snapshots* para o servidor não foi significativamente inferior em relação às restantes experiências. Eventualmente esta situação ficou a dever-se precisamente ao facto de ser visível que o professor estava de facto a utilizar o sistema constantemente o que como foi visto se traduzia em que alguns estudantes pudessem ter algumas reservas em expor versões inacabadas do código. Relativamente a este tipo de abordagem há que ter em atenção que tal como observado por um estudante, o professor deverá ter o cuidado de dar mais "espaço" aos estudantes para eles encontrarem autonomamente formas de resolver as dificuldades encontradas.

Parece, pois, evidente que uma utilização sistemática do sistema durante as aulas e fora destas permite fornecer uma assistência mais abrangente aos estudantes. No entanto, como já foi referido anteriormente, não se deverá criar a expectativa nos estudantes que os professores irão estar permanentemente a utilizar o sistema. Não só esta situação não é possível, como certamente não é desejável (quer para os professores, quer para o estudante que não deverá estar constantemente dependente da validação do seu trabalho pelos professores). O *feedback* fornecido pelos estudantes servirá certamente para futuramente ajudar os professores que venham a utilizar o sistema a escolher fundamentadamente qual a abordagem de utilização a seguir.

Em todo o caso, é importante verificar que a maioria dos estudantes de ambas as experiências considerou as intervenções feitas pelos professores importantes e a totalidade dos estudantes considerou a utilização deste tipo de sistemas relevante.

Também a ferramenta destinada aos estudantes para procederem ao acompanhamento do seu trabalho mereceu uma avaliação positiva de praticamente todos os estudantes das três experiências. A par do *feedback* positivo fornecido pelos professores, esta avaliação positiva feita pelos estudantes, constitui certamente um fator extra de confirmação do valor do sistema desenvolvido.

4.8. Iteração sete

4.8.1. Objetivos

Os resultados das experiências da iteração anterior indicam que o sucesso e a aceitação do sistema estão intimamente ligados com o tipo de utilização que os professores dão ao sistema e à assistência que dão aos estudantes com base nessa utilização. Assim, o principal objetivo desta iteração passou por verificar até que ponto algumas modificações no sistema no sentido de o aproximar mais dos estudantes assim como algumas mudanças a nível no modo como os professores o utilizam poderão influenciar quer o modo como os estudantes percecionam o sistema, quer o nível de aceitação do mesmo.

Para este efeito, foram feitas algumas modificações ao sistema, que serão em seguida apresentadas, tendo o sistema sido novamente testado em dois contextos em que já havia sido utilizado no passado.

4.8.2. Descrição do contexto

O sistema foi avaliado em unidades curriculares em que já tinha sido utilizado, nomeadamente na unidade curricular de Introdução à Programação em Java (IPJ) do Programa Acertar o Rumo da Universidade de Coimbra e na unidade curricular de Fundamentos de Programação (FP) da Licenciatura em Engenharia Informática da ESTGOH no primeiro semestre do ano letivo 2019/2020. Ainda que o sistema já tivesse sido utilizado nestas unidades curriculares no passado, em ambos os casos existiram mudanças significativas nos professores a elas associados. No caso de IPJ, dos cinco professores, apenas um se mantinha relativamente ao ano anterior (a professora responsável) e no caso de FP o docente nunca a tinha lecionado anteriormente.

4.8.3. Trabalho desenvolvido

4.8.3.1. Desenvolvimento dos *plug-ins*

Um aspeto importante que já havia sido identificado e inclusivamente implementado em algumas versões do *plug-in* prendia-se com a disponibilização de informação sobre os resultados dos testes diretamente no ambiente de desenvolvimento utilizado pelos estudantes. Como referido anteriormente, um dos fatores que impediam a utilização desta funcionalidade em todas as situações, estava associado ao facto de o editor de código ficar bloqueado alguns segundos até que a resposta chegasse.

A este respeito, foram feitas algumas modificações na forma como os testes eram realizados, por forma a permitir agilizar o processo e deste modo diminuir a ocorrência e a duração dos bloqueios. Ainda assim, não existindo a garantia de que tais bloqueios não viessem a ocorrer, foi transmitida informação aos estudantes sobre o motivo da sua eventual ocorrência (*e.g.* instabilidade da rede informática, existência de ciclos infinitos no código escrito pelos estudantes que ocorreriam igualmente no servidor e que levariam alguns segundos até serem detetados e eliminados).

No caso concreto de IPJ, os resultados dos testes efetuados eram disponibilizados automaticamente aos estudantes numa consola secundária. Deste modo, os estudantes poderiam continuar a utilizar o editor de código e a testar os programas na consola principal ou aceder aos resultados dos testes caso pretendessem, na consola "CodeInsights". Para alternarem entre consolas bastava selecionar a consola pretendida no menu disponibilizado para este efeito, tal como representado na Figura 115.

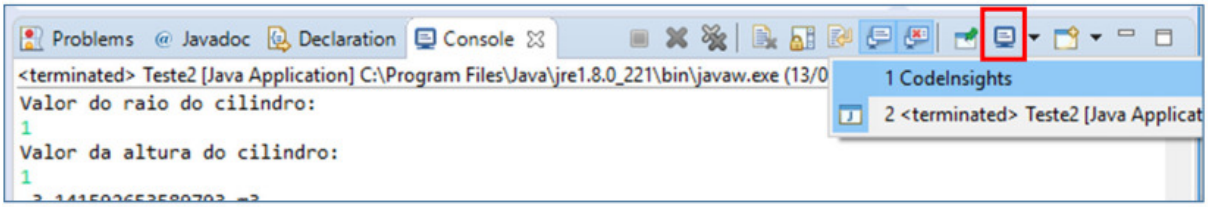

Figura 115 - Ilustração do processo de escolha de consola.

Na Figura 115 é apresentada a utilização da consola principal, com a qual os estudantes podiam interagir mediante a introdução dos *inputs* desejados. Ao mudar de consola, eram apresentados os resultados dos testes automáticos, tal como representado na Figura 116

```
Problems  @ Javadoc  Declaration  Console
CodeInsights
******************* CodeInsights Console *******************

Teste 1 - Falhou

double raio = 2;double altura = 4;

Esperado| ------------------------------------
volume = 50.26548245743669 m3
Obtido  | ------------------------------------
 50.26548245743669 m3
----------------------------------------------

Teste 2 - Falhou

double raio = 2; double altura = 2.4;

Esperado| ------------------------------------
volume = 30.159290672884506 m3
Obtido  | ------------------------------------
 30.159290672884506 m3
----------------------------------------------
```

Figura 116 - Exemplo da consola de apresentação dos resultados dos testes no Eclipse

Relativamente à experiência de Fundamentos de Programação, não foi necessário proceder ao desenvolvimento da funcionalidade de apresentação dos resultados dos testes automáticos, uma vez que era utilizado o Atom como editor de código, tendo esta funcionalidade já sido desenvolvida e testada anteriormente. Foi, contudo, introduzida uma funcionalidade que permitia que os professores inserissem *feedback* personalizado para cada exercício, o qual seria apresentado aos estudantes juntamente com os resultados dos testes, caso o código não passasse em todos eles. Um exemplo da apresentação deste *feedback* pode ser observado na Figura 117.

```
Teste 2 -  Falhou

int A=10, B=7, C=-2;

Esperado| ------------------------------------
A: 10 - B: 7 - C: -2 - menor: -2
Obtido  | ------------------------------------
A: 10 B: 7 C: -2 menor: -2
----------------------------------------------

==============================================
O resultado não foi igual ao esperado! Verifique:
- se o exercicio está devidamente identificado (aid 6.1);
- se está a utilizar bem os nomes das variáveis de input;
- se estas estão a ser declaradas e lidas no local correto;
- se não tem outras variáveis ou código entre //begin_inputs e //e
- e se o formato do output é igual ao esperado.
==============================================
```

Figura 117 - Exemplo de *feedback* personalizado para um exercício.

A pedido dos professores desta unidade curricular, foram ainda feitas mais algumas modificações no *plug-in*, nomeadamente a inclusão de uma "chave de segurança" que permitisse aumentar a fiabilidade da identificação dos estudantes que estão a submeter as soluções e evitar que sejam feitas submissões de forma fraudulenta, mediante a alteração do número do estudante (`@sid`). Assim, aquando da instalação do *plug-in*, os estudantes deviam proceder à configuração do seu editor com uma chave individual de 32 caracteres, que, entretanto, lhes havia sido fornecida pelos professores. Sempre que um *snapshot* era recebido do servidor, para além de se proceder à identificação do identificador do exercício (`@aid`) e do identificador do estudante (`@sid`), passou igualmente a ser feita a validação do valor de `@sid`, confrontando-o com o valor da "chave de segurança" enviada automaticamente pelo *plug-in*. Caso os valores fossem considerados válidos, o processamento do *snapshot* poderia continuar normalmente, caso contrário, este *snapshot* seria considerado inválido e o seu processamento terminaria.

4.8.3.2. Receção e processamento do código fonte

Para além da funcionalidade já referida anteriormente associada à validação da identidade do autor que submeteu os *snapshots*, ao nível do processamento dos *snaphots* não foram feitas muitas outras modificações para além de pequenas correções e ajustes.

4.8.3.3. Ferramenta de monitorização - professores

Nesta iteração, procedeu-se ao desenvolvimento de várias novas funcionalidades na ferramenta de monitorização destinada aos professores. O facto de a maioria dos professores estar a utilizar o sistema pela primeira vez, acabaria por ser importante, uma vez que estes tinham ideias novas sobre o que lhes devia ser disponibilizado, por forma a conseguirem prestar uma melhor assistência aos estudantes.

Uma dessas funcionalidades já tinha sido identificada desde a conceção inicial do sistema, contudo nunca havia sido implementada e passava pela disponibilização de exemplos de resolução para cada exercício. Principalmente, no caso de Introdução à Programação em Java, em que o número de exercícios era elevado e existiam cinco professores a trabalhar em simultâneo, fazia sentido que não fosse necessário que todos os professores procedessem à resolução de todos os exercícios, por forma a estarem mais bem preparados para assistir os estudantes. Assim, bastaria que um deles o fizesse, podendo posteriormente partilhar a(s) resolução(ões) com os colegas diretamente no sistema. Deste modo, caso determinado estudante lhes colocasse uma questão sobre o

modo como um exercício poderia ser resolvido, ou caso estivessem a analisar por iniciativa própria as resoluções feitas pelos estudantes, os professores poderiam rapidamente consultar as resoluções "modelo" já equacionadas. A disponibilização destas soluções "modelo" aos professores não implica, contudo, que os estudantes não possam desenvolver formas alternativas de resolver os problemas. Perante a identificação de uma nova forma de resolver determinado exercício, os professores poderiam adicioná-la à lista para referência futura. A disponibilização desta funcionalidade assumia um carácter de maior relevância tendo em conta a enorme diversidade de ritmos de trabalho observada no passado e que leva a que possam ocorrer situações em que cada estudante da turma possa estar a trabalhar num exercício diferente, inclusive de fichas de trabalho diferentes, o que torna a preparação das aulas mais complexa.

Uma funcionalidade do sistema que nesta iteração acabaria por sofrer diversas modificações foi a "matriz resumo". Com efeito, por forma a melhorar a identificação dos estudantes, para além do nome, passou igualmente a ser disponibilizada uma miniatura da sua fotografia que é aumentada quando o cursor do rato passa por cima desta. Foi ainda incluída no tipo de cada coluna, informação sobre o número do exercício respetivo. Foram ainda feitos alguns ajustes às cores utilizadas. Na Figura 118 é possível observar a legenda disponibilizada aos professores.

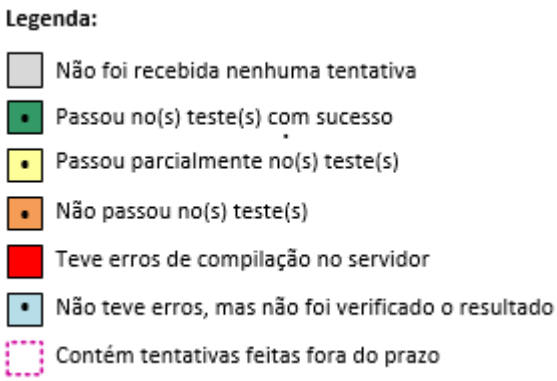

Figura 118 - Legenda da informação contida na matriz resumo - iteração sete.

No caso de Fundamentos de Programação, as resoluções dos exercícios submetidas eram tidas em consideração para efeitos de avaliação. Contudo, embora o professor aceitasse que fossem feitas submissões fora do prazo, apenas a última submissão feita dentro do prazo era tida em consideração. Assim, o professor solicitou a inclusão de elementos visuais que permitisse destacar os casos em que existissem submissões feitas fora do prazo limite. A solução encontrada passou pela inclusão de uma linha púrpura a

tracejado. Um exemplo da nova configuração da matriz pode ser observado na Figura 119.

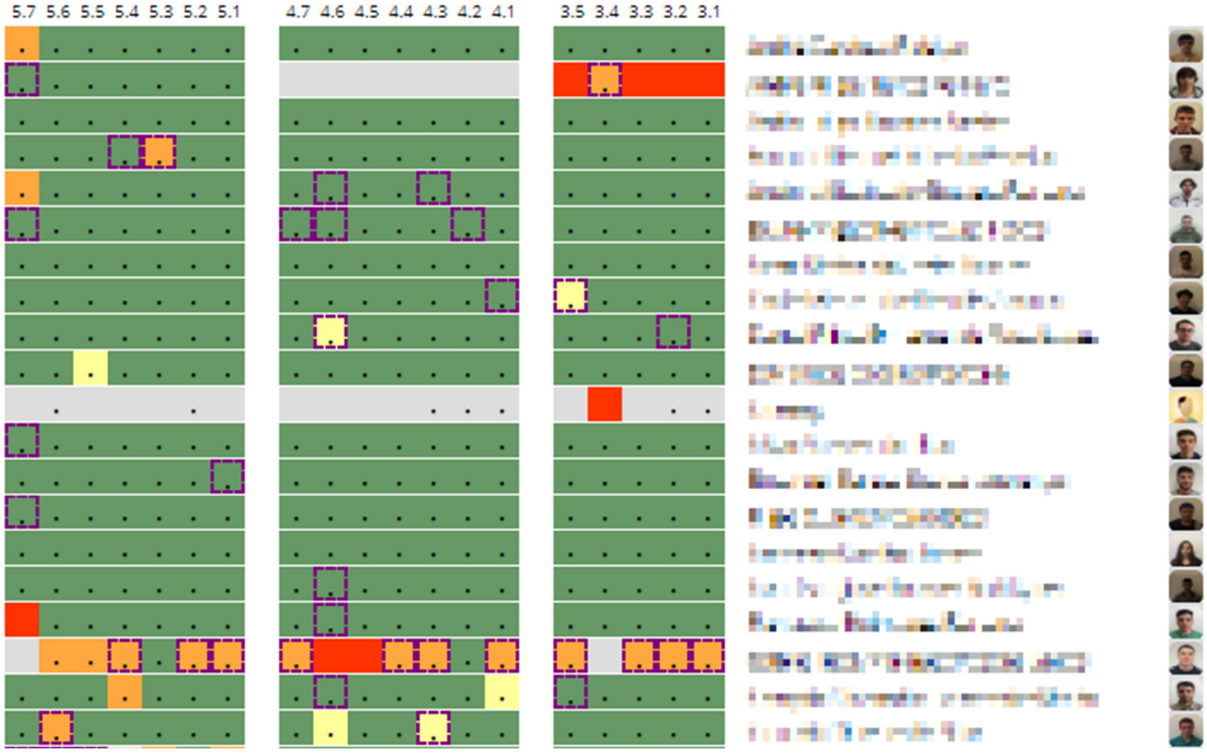

Figura 119 - Vista parcial da nova configuração da matriz resumo - iteração sete.

Esta pista visual foi igualmente utilizada na página com todas as tentativas realizadas por um estudante para um determinado exercício, de modo a destacar as submissões realizadas fora de prazo, tal como ilustrado na Figura 120.

Figura 120 - Lista de tentativas de resolução efetuadas por um estudante para um exercício - iteração sete.

Ainda associado com o facto de os exercícios de Fundamentos de Programação contarem para efeitos de avaliação, o professor decidiu que paralelamente à avaliação automática dos exercícios com base nos casos de teste iria também proceder à avaliação manual dos mesmos. Para este efeito passou a ser possível atribuir a cada exercício uma classificação numérica (0-100) e a introdução de um comentário relativo à solução encontrada pelos estudantes. Esta funcionalidade já tinha sido parcialmente desenvolvida (era apenas possível inserir uma classificação numérica), contudo, nunca havia sido testada em contexto real. Na Figura 121, é possível observar um exemplo do formulário utilizado pelos professores para introduzir a classificação e *feedback* personalizado.

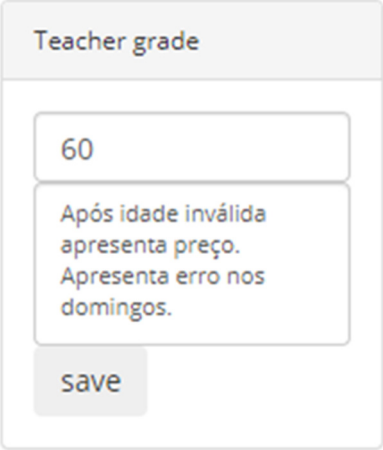

Figura 121 - Exemplo de classificação e comentário inserido manualmente pelo professor.

A introdução do *feedback* personalizado pelo professor, acaba por ter um importante valor formativo, uma vez que ainda que as modificações feitas após a correção dos exercícios já não sejam tidas em consideração para a atribuição da classificação, foi possível constatar que em muitas situações, os estudantes refizeram as suas soluções com base nos comentários do professor.

Uma vez que passou a ser possível a introdução de uma classificação para os exercícios, fazia sentido que fosse desenvolvida uma funcionalidade que permitisse aos professores a visualização das avaliações. Deste modo, procedeu-se ao desenvolvimento de uma funcionalidade que permitia ver as classificações dos exercícios organizados por ficha prática, a classificação parcial de cada ficha prática tendo em consideração os diferentes pesos que é possível atribuir a cada exercício, incluindo exercícios opcionais de bónus e finalmente a classificação global final tendo em conta todas as fichas práticas. Na Figura 122 pode ser observada uma porção desta tabela.

Esta visualização está apenas disponível para os professores, contudo, como será apresentado na secção seguinte, é também fornecido acesso aos estudantes às classificações obtidas em cada exercício, assim como ao *feedback* introduzido pelos professores.

Student Name	8.6	8.5	8.4	8.3	8.2	8.1	Total	7.7	7.6	7.5	4.6	4.5	4.4	4.3	4.2	4.1	Total	3.5	3.4	3.3	3.2	3.1	
[redacted]	-	-	-	-	-	-	0.00	-	-	-	-	-	-	-	-	-	0.00	-	-	-	-	-	0.00
[redacted]	100	90	90	100	80	100	18.67	0	70	100	80	100	100	100	70	90	18.00	100	100	100	100	100	18.10
[redacted]	0	0	0	0	80	100	6.00	-	-	0	-	-	-	-	-	-	0.00	-	-	-	-	-	5.35
[redacted]	0	100	70	100	100	100	15.67	-	80	100	80	100	90	100	100	100	19.00	100	100	100	100	100	18.05
[redacted]	100	90	70	100	80	100	18.00	-	80	90	80	100	100	100	70	90	18.00	100	100	100	100	100	16.55
[redacted]	100	90	80	100	100	100	19.00	-	0	100	90	100	100	100	70	100	18.67	100	100	100	100	100	17.25
[redacted]	50	70	70	100	100	100	16.33	0	90	100	30	80	90	100	100	100	16.67	100	100	100	100	100	16.40
[redacted]	-	-	-	-	-	-	0.00	-	-	-	-	-	-	-	-	-	0.00	-	-	-	-	-	0.00
[redacted]	-	-	-	-	-	-	0.00	-	-	-	-	-	-	-	-	-	0.00	-	-	-	-	-	0.00
[redacted]	70	0	0	0	80	100	8.33	-	-	-	60	80	90	100	100	90	17.33	100	100	100	100	100	13.20
[redacted]	100	0	70	80	100	100	15.00	70	80	100	100	100	100	100	100	90	19.67	100	100	100	100	100	18.10
[redacted]	80	70	70	90	80	100	16.33	0	60	100	70	80	90	100	100	100	18.00	100	100	100	100	100	17.20
[redacted]	-	-	-	-	-	-	0.00	-	-	-	-	-	-	-	-	-	0.00	-	-	-	-	-	0.00
[redacted]	100	90	70	100	100	0	15.33	-	70	100	70	100	100	100	70	80	17.33	100	100	100	100	100	16.50
[redacted]	-	-	-	-	-	-	0.00	-	-	-	-	-	-	-	-	-	0.00	-	-	-	-	-	0.00
[redacted]	-	-	-	-	-	-	0.00	-	-	-	-	-	-	-	-	-	0.00	-	-	-	-	-	0.00
[redacted]	100	100	100	100	80	100	19.33	100	100	100	90	100	0	100	100	90	16.00	100	100	100	100	100	17.53

Figura 122 - Vista parcial da tabela resumo das classificações dos exercícios

A correção manual dos exercícios constitui uma tarefa muito laboriosa, dada a quantidade de estudantes e de exercícios disponibilizados. De modo a melhorar a experiência de utilização do sistema, facilitando deste modo a atividade dos professores, procedeu-se ainda ao desenvolvimento de funcionalidades que permitem aos professores mudar de estudante e/ou de exercício de forma mais rápida sem necessitar de voltar à listagem de estudantes e/ou listagem de exercícios resolvidos por um estudante. Na Figura 123 pode ser observado um exemplo destes novos botões.

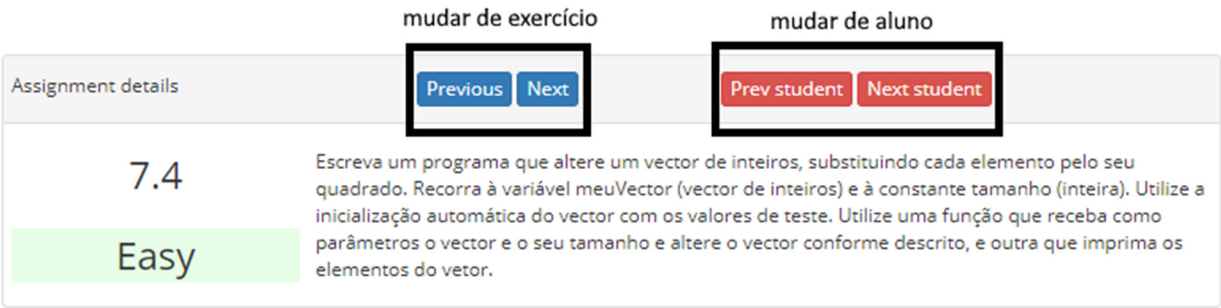

Figura 123 - Representação dos novos botões de navegação entre exercícios e estudantes.

A dada altura da experiência de Fundamentos de Programação, o professor manifestou que para alguns exercícios a existência de apenas dois casos de teste era um pouco limitativo. Já estava previsto aumentar o número de casos de teste de dois para quatro, mantendo o modo de funcionamento, contudo, o professor, mencionou, que ainda assim, apenas quatro poderia ser pouco. O principal problema com um número grande de casos

de teste estava principalmente associado ao tempo que era necessário para proceder à geração de novos ficheiros com os valores dos inputs, compilação, execução e recolha dos *outputs* desses ficheiros. Quantos mais testes fossem realizados, mais tempo demoraria e consequentemente, mais tempo o editor de código dos estudantes poderia eventualmente estar bloqueado.

Assim, foi idealizada outra forma de permitir utilizar um número arbitrário de casos de teste sem que o desempenho fosse afetado. Esta nova forma de fazer os testes só iria, no entanto, funcionar para exercícios em que fosse necessário recorrer obrigatoriamente à utilização de funções. Para além disso, quer o nome da função "principal", quer a ordem pela qual os argumentos eram passados deveria respeitar as indicações fornecidas pelo professor no enunciado, tendo os estudantes liberdade para utilizar as funções auxiliares que entenderem necessárias e com o nome que pretenderem.

Para esta solução recorreu-se à instrumentação do código fonte, na qual se procede à substituição da função `int main()` existente no código por uma outra na qual era feita a chamada de diversos testes unitários (sem limitação de número). Na Figura 124 é possível observar uma representação do processo.

```
#include <stdio.h>
//@cikey 99b4632b6db55da9f1e9219f90fd2aaa
//@sid ********
//@aid 6.5

int bissexto (int n){
  if    ((n%4==0)    &&    (n%100!=0)    ||
(n%400==0))
    return 1;
  else
    return 0;
}

int main()
{
  //begin_inputs
  int ano;
  printf("Introduza um ano:");
  scanf("%d", &ano);
  //end_inputs

  if (ano_bissexto(ano)==1)
    printf("ano: %d - bissexto", ano);
  else
    printf("ano: %d - nao bissexto", ano);
  return 0;
}
```

```
#include <stdio.h>
#include tests.h
//@cikey 99b4632b6db55da9f1e9219f90fd2aaa
//@sid ********
//@aid 6.5

int bissexto (int n){
  if ((n%4==0) && (n%100!=0) || (n%400==0))
    return 1;
  else
    return 0;
}

int main(){
  assert("1","bissexto(2019)", bissexto(2019) == 0);
  assert("2","bissexto(2020)", bissexto(2020) == 1);
  assert("3","bissexto(1900)", bissexto(1900) == 0);
  assert("4","bissexto(2000)", bissexto(2000) == 1);
}

int old_main()
{
  //begin_inputs
  int ano;
  printf("Introduza um ano:");
  scanf("%d", &ano);
  //end_inputs
  if (bissexto(ano)==1)
    printf("ano: %d - bissexto", ano);
  else
    printf("ano: %d - nao bissexto", ano);
  return 0;
}
```

Figura 124 - Apresentação das modificações feitas sobre o código fonte de modo a serem feitos os testes unitários,

Quando o professor executa os testes, os resultados serão apresentados na forma representada na Figura 125.

```
Test: 1 passed
Test: 2 passed
Test: 3 passed
Test: 4 passed
-------------------------
ALL TESTS PASSED
```

Figura 125 - Exemplos de apresentação dos resultados dos testes unitários.

Uma vez que esta solução foi desenvolvida sensivelmente a meio da experiência, convirá mencionar que estava apenas disponível "a pedido" para os professores e não para os estudantes que continuavam apenas ter acesso à informação sobre os dois casos de teste realizados da forma tradicional. A disponibilização desta informação também para os estudantes será certamente um aspeto a equacionar para versões futuras do sistema.

4.8.3.4. Ferramenta de monitorização - estudantes

A principal modificação na ferramenta de monitorização disponibilizada aos estudantes consistiu da inclusão de informação sobre a classificação formal atribuída pelo professor, assim como o *feedback* por ele inserido. Assim, na listagem dos exercícios realizados pelo estudante, para além da informação que já era apresentada, para cada exercício foi acrescentada uma etiqueta onde é apresentada a classificação que o professor lhe atribuiu e ao passar com o cursor por cima será apresentado o *feedback* fornecido pelo professor, tal como apresentado na Figura 126.

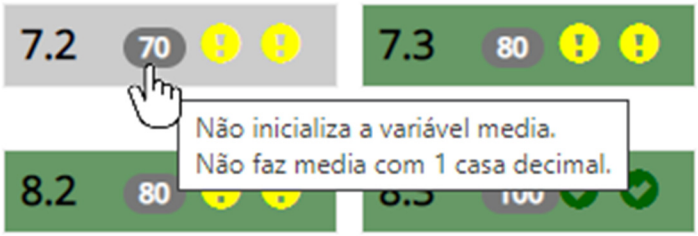

Figura 126 - Exemplo da forma como a classificação e o *feedback* é apresentado aos estudantes.

Foram ainda feitas algumas ligeiras modificações na página em que o estudante pode visualizar os detalhes sobre a última tentativa de resolução que foi submetida. Foi acrescentada informação sobre a classificação e o *feedback* atribuído pelo professor, sobre a data e hora em que aquela tentativa foi submetida e sobre os *inputs* utilizados em cada

caso de teste. Um exemplo do aspeto da nova configuração da página pode ser observado na Figura 127.

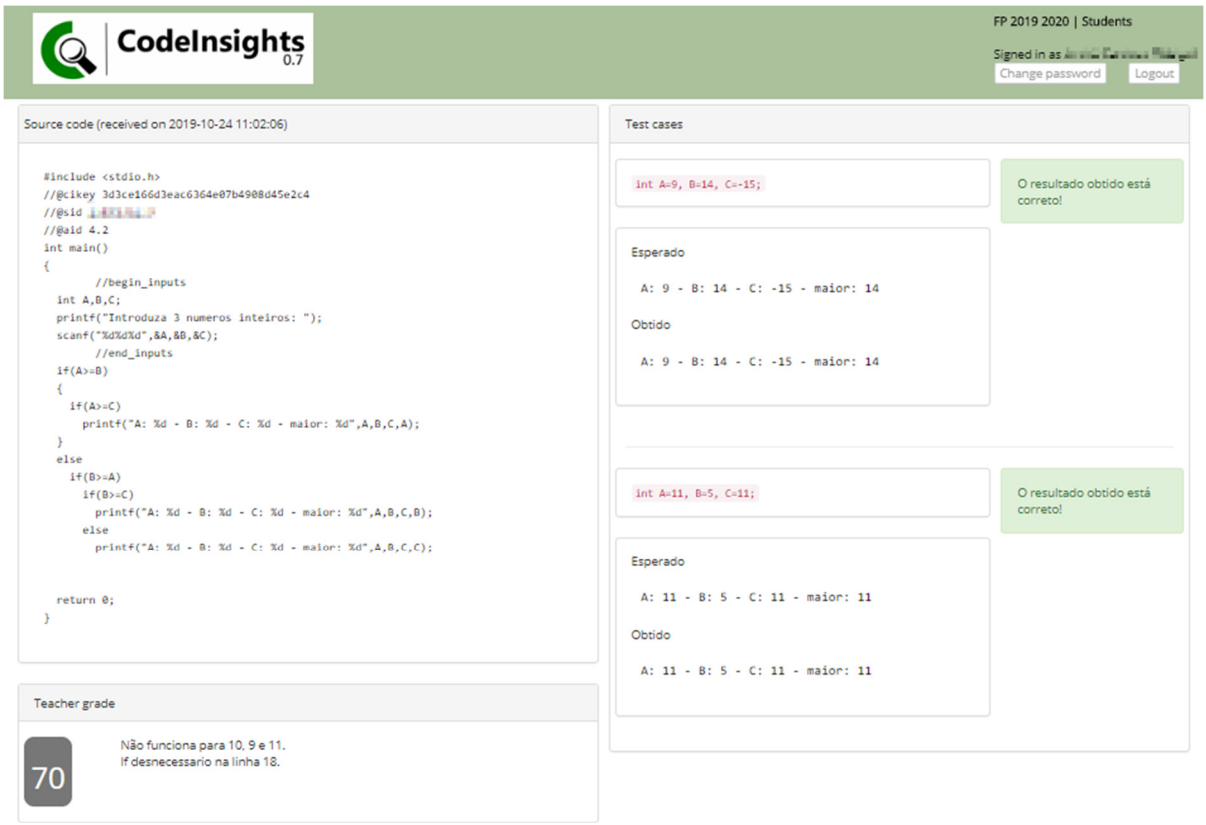

Figura 127 - Nova configuração da página de detalhes da última tentativa de resolução de um exercício na ferramenta de monitorização destinada aos estudantes - iteração sete.

Uma vez que anteriormente já haviam sido desenvolvidas as funcionalidades que permitiam editar uma versão temporária do código dos estudantes, entendeu-se que poderia fazer sentido que o professor pudesse gravar uma cópia desse ficheiro. Para tal, era criada uma nova submissão com as alterações feitas pelos professores. Muito embora já existisse a possibilidade de enviar *feedback* por email aos estudantes, esta funcionalidade tinha a vantagem de o professor não se limitar a descrever o que estava errado e como poderia ser corrigido, mas em alternativa poderia ele próprio exemplificar como o fazer. Obviamente, o professor poderia também editar o código no seu computador e enviar o ficheiro por email, contudo, considerou-se que concentrar a informação toda no mesmo sistema é mais eficiente.

Assim, sempre que um professor procedesse à edição da solução de um exercício desenvolvida por um estudante, na ferramenta de monitorização era apresentado um ícone com um "capelo" indicando que o professor tinha feito uma edição à solução do

exercício em questão, tal como pode ser observado na Figura 128. Posteriormente, o estudante poderia ver as modificações que o professor tinha feito.

Figura 128 - Exemplo de um exercício em que o professor procedeu à edição da solução desenvolvida pelos estudantes

Uma funcionalidade que não chegou a ser implementada, mas que poderá fazer sentido no futuro, consiste da possibilidade de destacar as modificações introduzidas pelo professor, face à solução desenvolvida pelo estudante, por forma a que os estudantes consigam mais facilmente encontrá-las. Em boa verdade, esta funcionalidade até já é disponibilizada para os professores, pelo que será apenas necessário implementá-la também na ferramenta destinada aos estudantes.

4.8.4. Avaliação do sistema

Nesta iteração sete, pretende-se avaliar até que ponto o maior envolvimento dos professores na utilização do ***CodeInsights*** e a disponibilização de mais e melhor *feedback* aos estudantes poderia contribuir para aumentar os níveis de aceitação do sistema por parte dos estudantes. Assim, para esta iteração foram desenvolvidas diversas novas funcionalidades e foi igualmente feito um esforço de melhoria dos manuais de utilizador disponibilizados aos estudantes e aos professores.

À semelhança das experiências anteriores, na primeira aula de cada unidade curricular em que o sistema ia ser utilizado, foram apresentados os objetivos da utilização do sistema, foi distribuída aos estudantes a documentação (manual de utilização e *cheat sheet*) e os estudantes que pretenderam participar nas experiências assinaram um documento dando consentimento para a recolha dos dados. Um exemplo da versão atualizada do manual de instalação e utilização pode ser consultado no Anexo IX.

Apresentam-se em seguida os principais resultados das duas experiências realizadas.

4.8.4.1. Introdução à Programação em Java (Acertar o Rumo)

O sistema ***CodeInsights*** foi utilizado neste contexto pela quarta vez. Como já referido anteriormente, uma particularidade desta edição prendia-se com o facto de quatro dos

cinco professores estarem a lecionar esta unidade curricular pela primeira vez (dois dos professores nunca tinham lecionado anteriormente em qualquer contexto).

A experiência decorreu entre os dias 24 de setembro e 11 de novembro de 2019 e estiveram envolvidos 20 estudantes. Durante a duração da experiência foram disponibilizados 153 exercícios organizados em sete fichas práticas. No total, foram recebidos aproximadamente 63.700 *snapshots* válidos (80.000 no total), tendo cerca de 20% sido recebidos fora do período em que os estudantes estavam em contexto de sala de aula.

Tendo em conta as experiências anteriores, perante este número de *snapshots* inválidos, procedeu-se à sua análise. Constatou-se que na sua maioria diziam respeito a dois estudantes (mais de 5000 associados a um estudante e aproximadamente 2000 associados a outro estudante). Em certa medida, este tipo de comportamento já seria de esperar, mas, tal como aconteceu nas experiências anteriores, foi realizado um inquérito aos estudantes no qual foi obtida mais alguma informação.

Tal como já havia acontecido em todas as experiências anteriores, existia uma grande disparidade a nível dos conhecimentos prévios de programação dos estudantes, o que acabaria uma vez mais por se traduzir em ritmos de trabalho muito diferentes, impedindo em certa medida que os professores conseguissem planear as aulas de uma forma igual para todos. Por este motivo, a utilização do sistema acabaria uma vez mais por constituir uma grande mais-valia.

Tendo em consideração as limitações que haviam sido identificadas anteriormente, nesta experiência optou-se por não utilizar o sistema de notificações automáticas, contudo, convirá destacar que toda a informação que era utilizada pelo sistema para a geração destas notificações estava disponível para visualização na aplicação.

4.8.4.1.1. Inquérito aos estudantes

Uma vez que nesta iteração se procedeu à introdução de diversas novas funcionalidades dirigidas diretamente aos estudantes e outras que ainda que mais dirigidas aos professores, tinham como objetivo final aumentar a qualidade do suporte prestado pelos professores aos estudantes, considerámos que faria todo o sentido ouvir também a opinião dos estudantes. Assim, após ter terminado o período previsto para a experiência, foi solicitado aos estudantes para responderem, de forma anónima, ao inquérito que pode ser consultado no Anexo X. No total, responderam ao inquérito 17 estudantes cujas repostas são apresentadas e analisadas em seguida.

Relativamente à primeira pergunta sobre se consideravam a utilização deste tipo de sistemas importante, apenas um estudante respondeu negativamente, tendo os restantes 16 respondido afirmativamente. O mesmo resultado foi obtido em relação à segunda pergunta sobre se no dia-a-dia tinham sentido que os professores realmente utilizavam o sistema. A este respeito, o estudante que respondeu negativamente fez questão de indicar que "*aprendi a matéria sozinho e descobri os meus erros sozinho sem a ajuda do CodeInsights*".

Em relação à pergunta número três, relativa a exemplos concretos da utilização do sistema nas aulas, foram apresentados diversos exemplos como a identificação de erros comuns e o esclarecimento em grupo dos principais motivos desses erros; o ajuste do ritmo de apresentação de tópicos com base no ritmo de trabalho dos estudantes; a utilização de soluções desenvolvidas pelos estudantes como ponto de partida para o esclarecimento de erros; ou a apresentação de soluções alternativas às desenvolvidas pelos estudantes.

No que diz respeito à pergunta número quatro sobre a relevância das intervenções feitas com base na informação fornecida pelo sistema foram obtidas duas respostas "nem sempre", uma resposta "não" dada pelo mesmo estudante que havia respondido negativamente às primeiras duas questões. Os restantes 14 estudantes responderam "sim".

À seguinte questão sobre o sentimento de desconforto por todas as tentativas de resolução serem enviadas para o servidor, apenas um estudante respondeu que de facto se sentia desconfortável. Na questão seguinte sobre os motivos deste desconforto, indicou o facto de por vezes não conseguir terminar a resolução de determinados exercícios e consequentemente, o que fica visível para os professores é uma solução inacabada. Um outro estudante referiu que não se sentia propriamente desconfortável, mas que por uma questão de perfecionismo não gostava de "sujar" o sistema com tentativas erradas. Muitos estudantes indicaram que de facto não sentiam nenhum desconforto, achando inclusivamente uma verdadeira mais-valia o facto de os professores poderem ter acesso ao processo de conceção das soluções, uma vez que esta funcionalidade permitia aos professores a identificação de problemas e consequentemente permitia-lhes intervir. Uma aluna disse mesmo que esta funcionalidade lhe dava confiança extra.

Na questão seguinte sobre a adoção de medidas por forma a contornar o desconforto sentido, o estudante que respondeu que sentia desconforto na questão anterior, indicou

que utilizava um IDE alternativo até ter uma solução estável. Já o outro estudante indicou que recorria à utilização de um valor inválido para `@aid` até ter uma solução que considerava válida.

Em relação à questão seguinte, seis dos 17 estudantes (35%) responderam que se sentiriam mais confortáveis caso tivessem a possibilidade de decidir quando as tentativas são enviadas para o servidor. Um dos estudantes, indicou que considerava importante decidir quando as submissões eram feitas não por se sentir desconfortável, mas sim devido ao tempo extra que o processamento das tentativas no servidor implicava.

Relativamente à frequência de utilização da ferramenta de apresentação de *feedback* diretamente no Eclipse, os resultados podem ser observados na Figura 129.

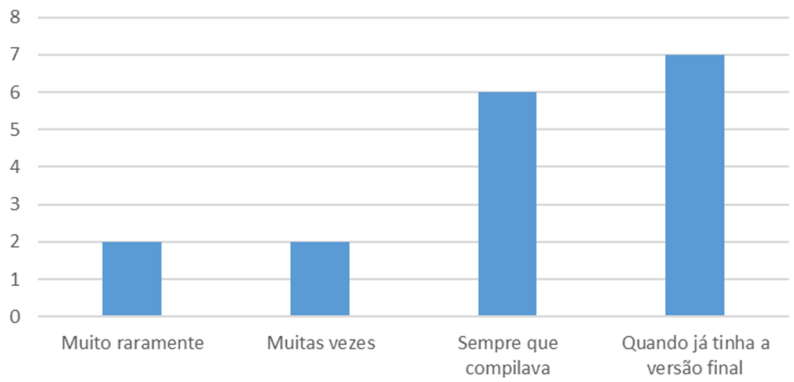

Figura 129 - Distribuição das respostas dos alunos relativamente à utilização da ferramenta de apresentação de feedback no Eclipse - IPJ AR - iteração sete.

Um dos estudantes que respondeu "muito raramente" foi o estudante que anteriormente havia respondido que "*aprendia tudo sozinho*". Já o outro que também respondeu "muito raramente", fez questão de acrescentar que "*preferia utilizar o site de monitorização*".

Em relação à frequência com que os estudantes davam importância à informação apresentada na consola, nove estudantes indicaram que davam sempre importância, já oito indicaram "por vezes", e justificaram a sua resposta o com facto de por vezes o sistema indicar que o código não passava nos nestes, no entanto, como por vezes isto se devia a pequenos problemas de incompatibilidade de caracteres, eles próprios assumiam que a sua solução estava correta, independentemente do que o sistema indicava.

Em seguida, era perguntado aos estudantes se consideravam útil a disponibilização de ferramentas que permitam alguma validação imediata do trabalho desenvolvido.

Responderam afirmativamente 15 estudantes e negativamente dois estudantes (um deles, o que utilizava outro IDE e outro, o estudante que "*aprendia tudo sozinho*").

Na questão seguinte, era pedido aos estudantes para fornecerem críticas/sugestões sobre a ferramenta de disponibilização de resultados dos testes efetuados diretamente no Eclipse. Três estudantes, indicaram que seria interessante a utilização de mais casos de teste, enquanto outros indicaram que deveriam ser utilizados exercícios em que não fosse possível existirem problemas a nível de incompatibilidade de caracteres (*e.g.* "-" e "-"). Este problema de incompatibilidade foi de facto identificado e foram feitas algumas diligências no sentido de o ultrapassar, contudo, uma vez que está intimamente ligado aos enunciados de exercícios disponibilizados aos estudantes, não é algo que esteja completamente dependente do sistema.

Na questão número treze era perguntada a regularidade com que os estudantes utilizavam a ferramenta de monitorização *online*. Os resultados podem ser observados na Figura 130.

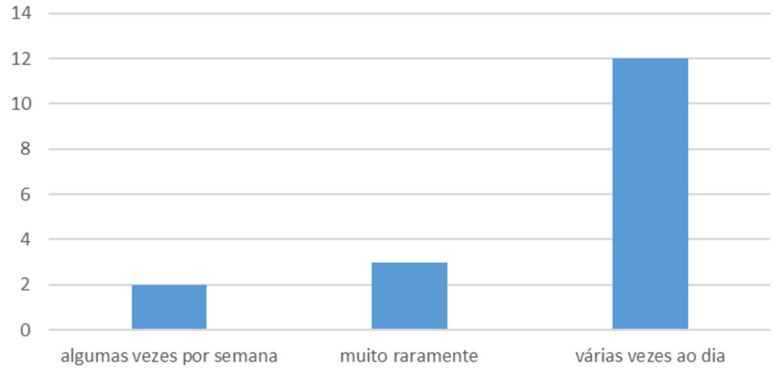

Figura 130 - Distribuição dos estudantes em termos da frequência de acesso à ferramenta de monitorização - IPJ AR - iteração sete.

Como é possível constatar, a grande maioria dos estudantes acedia à ferramenta de monitorização com muita frequência. Um estudante disse mesmo que o facto de conseguir ter uma forma de conseguir "atestar" o seu progresso lhe dava motivação extra para continuar.

Em seguida foi perguntado aos estudantes se consideravam importante que os professores pudessem enviar *feedback* aos estudantes que ficasse visível na ferramenta de monitorização. Todos os estudantes responderam afirmativamente, tendo alguns mencionado que para além de o *feedback* ficar disponível no sistema, deveria sempre ser enviado um email aos estudantes. A possibilidade de envio de email aos estudantes era

de facto possível para o caso de os professores utilizarem a funcionalidade de envio de *feedback* por email, contudo, não foi considerada no caso da edição das soluções dos estudantes pelos professores. Esta funcionalidade irá certamente ser implementada em futuras versões do sistema.

A seguir foram solicitadas críticas/sugestões de melhoria da ferramenta de monitorização, não tendo sido obtida nenhuma resposta.

Para terminar, foram solicitadas críticas ou sugestões de melhoria relativamente a todo o sistema. A este respeito, diversos estudantes indicaram que seria interessante que os professores pudessem introduzir *feedback* personalizado para cada exercício com base nos principais cuidados a ter e nos erros que antecipavam que pudessem ocorrer. Curiosamente, como visto anteriormente, esta funcionalidade acabaria de facto por ser desenvolvida, mas apenas ficou disponível para a outra experiência, cujas aulas começaram ligeiramente mais tarde.

Alguns estudantes mencionaram também que deveria ser criado um mecanismo que permitisse disponibilizar aos estudantes exemplos de resoluções para cada exercício ainda que alguns dias após a disponibilização do enunciado. Esta é uma sugestão interessante e de fácil implementação, uma vez que já existe para os professores. A disponibilização desta funcionalidade, ficará, contudo, dependente da vontade dos professores em partilhar as soluções com os estudantes.

Finalmente, alguns estudantes mencionaram que gostariam de poder utilizar o sistema para todos os exercícios. Curiosamente, no dia antes de ser disponibilizado o inquérito aos estudantes, como já haviam passado alguns dias desde o suposto término da experiência, procedeu-se à desativação do sistema, ou seja, continuava a ser possível consultar toda a informação que já tinha sido recebida, contudo, não era possível submeter novos *snapshots*. Para nossa surpresa, diversos estudantes solicitaram a reativação do mesmo, uma vez que a sua utilização era encarada como importante, e como, entretanto, foram aprendendo coisas novas, gostariam de ter a oportunidade de voltar a refazer alguns exercícios utilizando esses novos conhecimentos e continuar a ter o suporte do sistema. Esta situação pode ser observada na Figura 131.

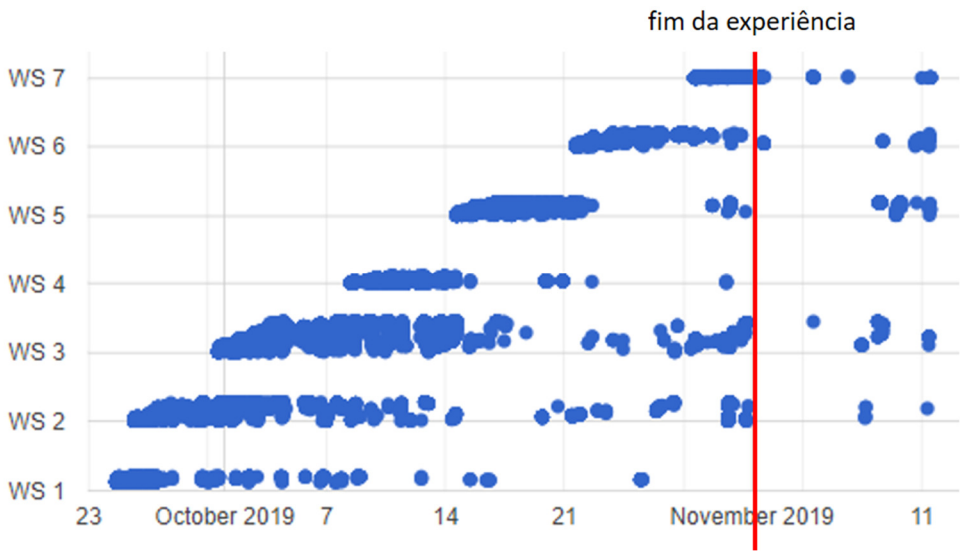

Figura 131 - Distribuição dos *snapshots* recebidos ao longo da duração da experiência - IPJ AR - iteração sete.

Convirá ainda referir que alguns dias após o fim da experiência os estudantes iriam realizar o primeiro momento de avaliação formal que consistia de um teste prático de programação no qual era solicitado que programassem a resolução de dois exercícios, tendo o sistema sido utilizado também como uma ferramenta de apoio ao estudo para esta prova.

Um aspeto que foi muito interessante constatar foi o alinhamento verificado entre as classificações que os estudantes obtiveram neste teste e o desempenho que era possível observar no sistema ao longo das aulas. Assim os estudantes com um ritmo de trabalho mais lento obtiveram classificações mais baixas, ao passo que os estudantes que trabalharam com um ritmo de trabalho consistentemente mais rápido, obtiveram as melhores classificações.

4.8.4.1.2. Entrevista aos professores

Conforme previsto, após terminada a experiência e depois de auscultados os estudantes, procedeu-se à realização de uma entrevista semiestruturada com os cinco professores desta unidade curricular, com vista a compreender a forma como utilizaram o sistema e também recolher críticas e sugestões para futuros melhoramentos deste. A estrutura geral da entrevista pode ser encontrada no Anexo XI. Em termos globais é muito semelhante à que foi utilizada no ano anterior, tendo ainda assim sido reformuladas algumas questões que se revelaram um pouco redundantes.

Relativamente à primeira questão sobre a regularidade com que utilizavam o sistema, todos os professores responderam que utilizavam o sistema frequentemente,

principalmente nos dias anteriores aos dias em que tinham aulas e no próprio dia em que tinham aulas. Vários professores indicaram que utilizavam frequentemente o sistema também durante as aulas.

Sobre o modo como utilizaram o sistema, todos os professores indicaram que o sistema era utilizado principalmente antes do início das aulas por forma a verificar o ponto da situação. Todos os professores indicaram também que por diversas vezes quando verificavam que existiam estudantes mais atrasados face aos colegas ou exercícios com menos resoluções com sucesso, procediam à exploração de mais informação com vista à deteção das eventuais causas para os desvios e dificuldades que os estudantes podiam estar a sentir, permitindo deste modo encontrar a melhor forma de intervir. Um dos professores indicou que frequentemente recorria ao sistema para verificar as diversas formas como os estudantes estavam a abordar os problemas, de modo a conseguir corrigir as eventuais falhas e apresentar formas alternativas e mais otimizadas de o fazer.

Perguntou-se em seguida aos professores quais as funcionalidades do sistema que utilizavam com mais regularidade, tendo todos eles respondido que era a "matriz resumo", tendo em conta a quantidade de informação que é possível obter de uma única visualização. Para além desta visualização, vários professores mencionaram também a página que permite navegar pelas tentativas de resolução dos estudantes.

Na continuação, pretende-se saber se os professores sentiam que com a utilização do ***CodeInsights*** tinham tido acesso a informação sobre o modo de trabalho dos estudantes que não teriam de outro modo, se este facto influenciou o modo como acompanharam os estudantes e em caso afirmativo se poderiam apresentar exemplos. Todos os professores responderam afirmativamente e apresentaram como exemplos o acompanhamento em tempo real do trabalho dos estudantes, mesmo quando não estavam com eles na sala de aulas, incluindo no horário pós-laboral e fins de semana. Referiram ainda o facto de o sistema ter toda a informação agregada e sistematizada. Uma professora mencionou ainda o facto de poder ter uma visão do progresso dos estudantes mais introvertidos. A este respeito, vários professores mencionaram o caso de um estudante que nunca tinha feito qualquer pergunta ou pedido de esclarecimento por iniciativa própria e que através da utilização do ***CodeInsights*** conseguiram verificar que ele estava de facto a progredir de forma consistente, ainda que um pouco mais lentamente. Um outro professor referiu ainda que para além de tudo mais, o sistema é interessante para confirmar (ou não) a perceção que os professores têm sobre alguns estudantes e que por vezes está errada.

Prosseguiu-se, perguntado aos professores se ao longo das aulas tinham tomado decisões influenciadas pela informação que conseguiam obter no sistema e se poderiam apresentar exemplos de tais decisões. Todos os professores responderam afirmativamente e, como exemplo, foi indicada a decisão de reforçar um determinado conceito através da apresentação de novos exemplos ou a apresentação de formas alternativas de resolver os exercícios.

Em seguida perguntámos aos professores se tinham sentido dificuldades a interpretar a informação disponibilizada. No geral, os professores responderam que a informação era de fácil compreensão, contudo, dada a quantidade e variedade de informação disponibilizada, por vezes necessitavam de recorrer às legendas.

Perguntou-se em seguida aos professores se tinham sentido algum tipo de dificuldade ao nível da usabilidade da aplicação e se tinham alguma sugestão de melhoria a fazer. Foram feitas algumas sugestões como a inclusão de ajuda contextual, a possibilidade de redimensionar algumas caixas de código e visualizações. Foi ainda sugerido que no *dashboard* deveria ser possível escolher a informação que era apresentada, assim como a ordem e o tamanho dos elementos a apresentar. Finalmente, um professor sugeriu a inclusão de caixas de confirmação associadas a alguns botões como por exemplo "*Move*" ou "*Trash*" da página com as tentativas de resolução.

Relativamente à perceção que os professores tiveram relativamente à relação sistema/estudante, todos os professores afirmaram que não sentiram qualquer tipo de sentimento de desconforto pelo facto de o sistema estar a ser utilizado e ser sabido que os professores tinham acesso à informação em tempo real. Em boa verdade, a noção que os professores tiveram foi que os estudantes gostariam que os professores utilizassem o sistema mais ativamente e lhes dessem (ainda) mais *feedback*. Os únicos momentos em que era notória alguma insatisfação era quando o sistema indicava que a solução dos estudantes estava errada ou potencialmente certa, quando nos computadores deles estava aparentemente tudo bem. Convirá, contudo, realçar que tinha sido transmitido aos estudantes que na maior parte dos casos, estes problemas estavam relacionados com a utilização incorreta das variáveis de *input* e/ou o formato do *output*. Ainda assim, alguns estudantes, principalmente numa fase inicial, ignoravam as instruções dadas nos enunciados e como tal, acabam por ficar algo frustrados pelo facto de os resultados não serem positivos.

Com a questão seguinte pretendeu-se saber se, na opinião dos professores, a imagem que o sistema lhes dava dos estudantes, correspondia, ou não, à realidade. Todos os professores disseram que globalmente sim, contudo esta imagem não poderia ser baseada unicamente numa visualização agregada como a "matriz resumo", por exemplo, há que explorar um pouco mais para "afinar" esta imagem. Em todo o caso, mesmo relativamente a alguns estudantes que, por exemplo, raramente se preocuparam com o formato dos *inputs* e *outputs*, esta atitude acabaria por se enquadrar com a atitude displicente e em certos casos mesmo provocatória que foram apresentando ao longo das aulas.

Questionámos em seguida os professores sobre se em situações futuras considerariam utilizar o sistema ***CodeInsights*** para suportar unidades curriculares que venham a lecionar no futuro. Todos os professores responderam, sem dúvidas, afirmativamente, tendo ainda referido que tinham pena de não o poderem continuar a utilizar na unidade curricular que estavam a ministrar.

Para terminar, perguntámos aos professores se tinham mais alguma crítica ou sugestão que pretendessem fazer. Todos os professores começaram por dizer que tinham gostado muito de utilizar o sistema, mas ainda assim, havia espaço para melhorias. Um dos professores mencionou que seria interessante se fosse possível utilizar um mecanismo de comparação entre outputs esperados e obtidos um pouco menos sensível a pequenas diferenças como espaços vazios ou diferentes codificações do mesmo símbolo. Um outro professor referiu que seria interessante se o sistema conseguisse automaticamente identificar "padrões" de soluções desenvolvidas pelos vários estudantes. Este mesmo professor referiu que por forma a tentar mitigar alguma lentidão que por vezes ocorria, poderia ser interessante se os estudantes pudessem definir se pretendiam que as suas tentativas fossem processadas de forma síncrona e veriam os resultados diretamente no editor de código (mas que poderia ser mais lento) ou de forma assíncrona, sendo os resultados apenas visíveis na ferramenta de monitorização. Um outro professor sugeriu a possibilidade de permitir fazer o *download* do código fonte referente às últimas tentativas de resolução de vários exercícios do mesmo estudante de uma só vez. Finalmente uma professora disse que seria muito importante que o sistema permitisse ser utilizado em situações em que a resolução de um problema necessite de utilizar mais do que um ficheiro de código, uma vez que este era o motivo pelo qual o sistema não poderia ser utilizado durante toda a duração da unidade curricular em causa.

Uma professora fez ainda questão de mencionar que embora tivesse utilizado bastante o sistema e o tivesse considerado uma preciosa ajuda, sentia que após esta primeira experiência se sentiria muito mais à vontade para o utilizar no futuro e para explorar outras funcionalidades que o sistema oferece, mas que nesta edição não tinha explorado suficientemente. Como exemplos, mencionou as ferramentas de fornecimento de *feedback* aos estudantes via *email* ou a funcionalidade de partilha de possíveis resoluções entre professores.

4.8.4.2. Fundamentos de Programação (LEI - ESTGOH)

Tal como referido anteriormente, o sistema ***CodeInsights*** já tinha também sido utilizado várias vezes no contexto desta unidade curricular, contudo nesta edição, ocorreu a mudança do professor responsável. O anterior professor sempre mostrou interesse e satisfação em utilizar o sistema. De igual modo, o novo professor responsável pela componente prática, mostrou-se também interessado em utilizar o sistema, não só para acompanhar o progresso dos estudantes, mas também como uma ferramenta de apoio à sua atividade, nomeadamente na correção dos exercícios e posterior disponibilização das notas aos estudantes. Com efeito, já foram descritas anteriormente algumas novas funcionalidades introduzidas nesta interação resultantes de sugestões deste professor.

O facto de esta experiência ter começado relativamente mais tarde do que a outra, teve a vantagem adicional de ter existido possibilidade de já ter integradas algumas novas funcionalidades resultantes do *feedback* fornecido pelos outros professores e estudantes que, entretanto, já estavam a utilizar o sistema há algum tempo.

A experiência decorreu entre os dias 17 de outubro e 27 de novembro de 2019. Os estudantes tinham quatro horas de aulas de componente prática com o professor uma vez por semana. Estavam inscritos nesta unidade curricular 47 estudantes, tendo sido recebidos *snapshots* de 35 estudantes. Uma vez que a resolução dos exercícios era tida em consideração para efeitos de avaliação, a utilização do *plug-in* era muito aconselhada, muito embora não fosse obrigatória. Com efeito, alguns estudantes decidiram não instalar o *plug-in*, tendo enviado as resoluções para o professor via email. No total, durante a duração da experiência foram disponibilizados para resolução um total de 39 exercícios organizados em seis fichas de trabalho, tendo sido recebidos aproximadamente 42.100 *snapshots* válidos (mais de 50.000 no total). Entre os *snapshots* inválidos encontravam-se principalmente pequenos pedaços de código, contudo, foi também

possível encontrar um estudante com cerca 700 *snapshots* inválidos, um outro com 405 *snapshots* inválidos e um outro com 302 *snapshots* inválidos. Após alguma análise foi possível verificar que muito possivelmente se tratava de invalidação voluntária das tentativas de resolução. Ainda assim, quer em termos do número de estudantes, quer a nível do número de tentativas inválidas, considerou-se não ter sido uma situação muito preocupante.

Do total de *snapshots* recebidos, apenas aproximadamente 10% foram realizados fora do período de aulas, o que significa que a grande maioria dos estudantes trabalhava nos exercícios apenas durante as horas de aula, não dedicando muito tempo a estudar fora deste período. Com efeito, na Figura 132 é possível observar a distribuição dos *snapshots* de cada estudante (eixo dos YY) ao longo do tempo (eixo dos XX), em que é claramente visível uma concentração da quantidade de *snapshots* realizados no dia em que as fichas de trabalho eram disponibilizadas.

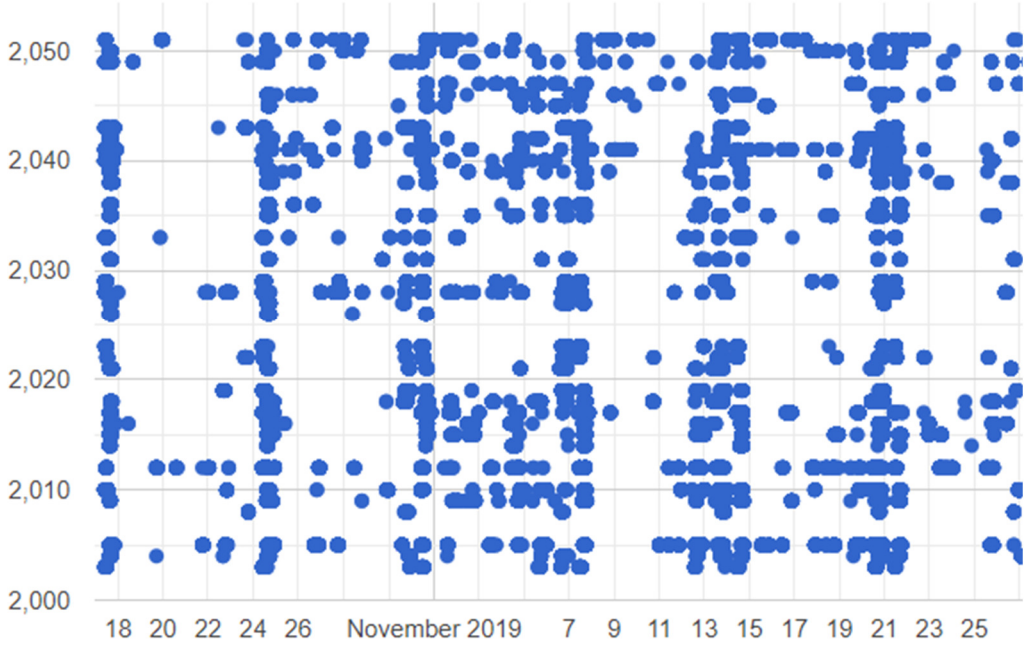

Figura 132 - Distribuição dos snapshots de cada aluno ao longo da duração da experiência - FP ESTGOH - iteração sete.

Ainda que o professor estabelecesse uma data-limite para a entrega das submissões a contar para efeitos de avaliação, como é possível verificar na Figura 133 foram ainda assim recebidas diversas submissões após essa data, o que é indicativo, que mesmo que já não contasse para efeitos de avaliação, alguns estudantes continuavam a resolver os exercícios. Nesta figura, as linhas verticais vermelhas representam as datas-limite definidas pelo professor para a submissão dos exercícios de cada ficha prática.

Uma vez que aquando da correção dos exercícios, o professor geralmente fornecia algum *feedback* aos estudantes sobre formas como poderiam melhorar as suas soluções, verificou-se que grande parte das submissões efetuadas após a data-limite correspondiam a esses estudantes.

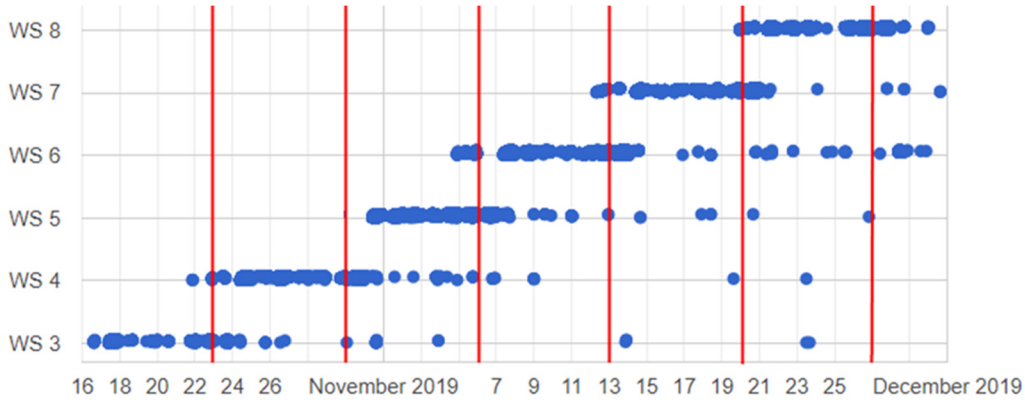

Figura 133 - Distribuição dos snapshots recebidos ao longo do tempo - FP ESTGOH - iteração sete.

Tal como seria de esperar, também aqui foi possível identificar estudantes com ritmos de trabalho consideravelmente distintos. Contudo, estas diferenças eram notórias principalmente no início das aulas em que eram disponibilizados os exercícios, com alguma tendência consistente para estabilizarem mais para o final da aula/dia. Como haveria de ser confirmado múltiplas vezes pelo professor, esta "estabilização" ficava em grande medida a dever-se à partilha das soluções entre os estudantes e não necessariamente ao seu trabalho. A este nível, a utilização da ferramenta acabaria também por desempenhar um importante papel no suporte à identificação de plágio entre estudantes.

Quer pelo facto de o professor ter manifestado a vontade de utilizar o sistema com frequência, quer pelo facto de este ano serem disponibilizados enunciados de exercícios distintos dos últimos anos, o que levaria a dificuldades acrescidas na sua utilização, também neste caso se optou por não utilizar o sistema automático de notificações.

4.8.4.2.1. Inquérito aos estudantes

Alguns dias após o término da experiência foi solicitado aos estudantes para responderem a um inquérito, cuja estrutura pode ser consultada no Anexo X. Responderam a este inquérito 25 dos 35 estudantes de quem foi recebido pelo menos um *snapshot*. São, em seguida, apresentadas as respostas recebidas.

Todos os estudantes responderam afirmativamente à primeira pergunta relativa à importância da utilização de ferramentas de monitorização no processo de ensino/aprendizagem. Em relação à segunda pergunta sobre a noção de que o professor utilizava de facto o sistema, apenas dois estudantes responderam negativamente, contudo, a avaliar pelas respostas que deram em seguida, não ficou claro se estas respostas eram de facto verdadeiras, ou se era apenas algum tipo de reação face a algo que tenha acontecido ao longo das aulas.

Em relação à pergunta número três, relativa a exemplos concretos da utilização do sistema nas aulas, diversos estudantes mencionaram a identificação de erros comuns e o esclarecimento em grupo dos principais motivos desses erros; a apresentação de soluções alternativas às desenvolvidas pelos estudantes ou a abordagem individual de estudantes no intuito de resolver problemas/erros que os estudantes pudessem estar a cometer.

No que diz respeito à pergunta número quatro sobre a relevância das intervenções feitas com base na informação fornecida pelo sistema foram obtidas quatro respostas "nem sempre", tendo os restantes 21 estudantes respondido "sim". A julgar pela justificação apresentada pelos estudantes, é de acreditar que em alguns casos, a causa da não pertinência das intervenções do professor ficou-se a dever ao facto de este ter detetado situações de plágio e outras irregularidades, tendo agido em conformidade, o que não agradou aos estudantes em causa.

À seguinte questão sobre o sentimento de desconforto por todas as tentativas de resolução serem enviadas para o servidor, apenas dois estudantes responderam que de facto se sentiam desconfortáveis. Diversos estudantes mencionaram que não sentiam qualquer tipo de desconforto e até consideravam importante que o professor tivesse acesso a todas as tentativas, uma vez que podia identificar problemas antes da submissão final e assim ajudá-los a corrigir esses problemas.

Em relação aos estudantes que responderam afirmativamente esta questão, na questão seguinte pretendeu-se saber quais os principais motivos para este desconforto. Ambos os estudantes deram respostas semelhantes indicando que por vezes eram recolhidos *snapshots* de código incompleto e/ou experimental e que o professor podia ficar com uma imagem errada deles.

Na questão seguinte sobre a adoção de medidas por forma a contornar o desconforto sentido, ambos os estudantes indicaram que recorriam à utilização de um valor inválido para `@aid` até terem uma solução que consideravam válida. Assim, tal como as suas

respostas indicaram, o principal motivo para "esconderem" o trabalho que efetuavam consistia na preocupação em manter uma "boa imagem" perante o professor.

Em relação à questão seguinte, 13 dos 25 estudantes (52%) responderam que se sentiriam mais confortáveis caso tivessem a possibilidade de decidir quando as tentativas são enviadas para o servidor. Um dos estudantes referiu que considerava boa ideia poder escolher, mas não por uma questão de desconforto, aliás, na questão anterior este estudante referiu que não havia sentido qualquer desconforto.

Sobre a frequência de utilização da ferramenta de apresentação de *feedback* diretamente no Atom, a maioria dos estudantes (15 em 25 - 60%) indicou que utilizava esta ferramenta sempre que compilava o código, já os restantes 10 estudantes (40%) responderam "muitas vezes". Uma vez que o próprio *plug-in* apresentava informação sobre os erros de compilação, caso estes existissem, esta poderá ser a explicação para este comportamento dos estudantes.

A grande maioria dos estudantes (21 em 24 - 84%) indicou que dava sempre importância à informação apresentada, já quatro (16%) indicaram "por vezes", e justificaram a sua resposta o com facto de por vezes o sistema indicar que o código não passava nos testes, no entanto, como por vezes isto se devia a pequenos problemas de incompatibilidade de caracteres, eles próprios assumiam que a sua solução estava correta, independentemente do que o sistema indicava.

Em seguida, era perguntado aos estudantes se consideravam útil a disponibilização de ferramentas que permitam alguma validação imediata do trabalho desenvolvido. Responderam afirmativamente 23 estudantes (92%) e negativamente um estudante (4%). Um outro estudante respondeu "sim" e "não" e justificou a sua resposta. Por um lado, considerava importante, pois era possível ter logo uma ideia sobre o correto funcionamento do código, mas por outro lado, por vezes a apresentação desta informação deixava de ter um caráter informativo e passar em todos os testes passava a ser um objetivo, o que por vezes poderia levar a alguma perda de tempo por causa da necessidade de garantir que o resultado tinha de ser exatamente igual.

Na questão seguinte, era pedido aos estudantes para fornecer críticas/sugestões sobre a ferramenta de disponibilização de resultados dos testes efetuados diretamente no Atom. Curiosamente, as respostas foram de encontro às que já haviam sido dadas na outra experiência. Assim, alguns estudantes indicaram a disponibilização de mais casos de teste e que deveriam ser utilizados exercícios em que não fosse possível existirem

problemas a nível de incompatibilidade de caracteres (*e.g.* "-" e "-"). Uma vez que esta questão está intimamente ligada aos enunciados de exercícios disponibilizados aos estudantes, não é algo que esteja completamente dependente do sistema, ainda assim é um aspeto que não poderá ser descurado no futuro.

Na questão seguinte era perguntada a regularidade com que os estudantes utilizavam a ferramenta de monitorização *online*, estando os resultados apresentados na Figura 134.

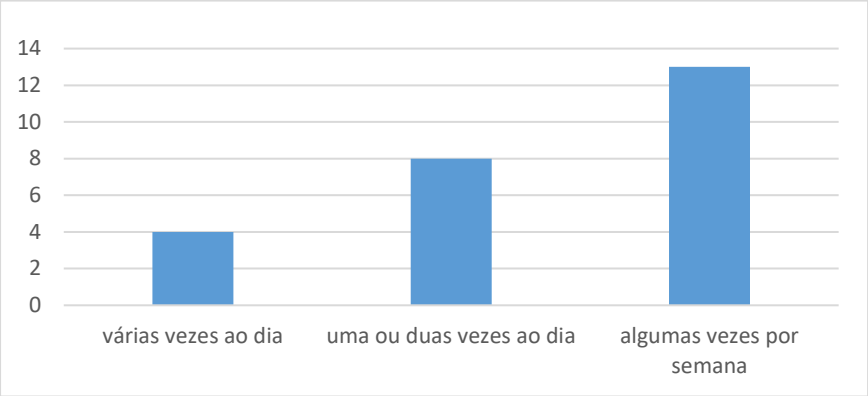

Figura 134 - Distribuição dos estudantes em termos da frequência de acesso à ferramenta de monitorização - FP LEI ESTGOH - Iteração sete.

Uma vez que os estudantes apenas tinham aulas uma vez por semana e que como foi possível constatar anteriormente, estes trabalhavam maioritariamente nesse dia, estes resultados parecem ser consistentes com o comportamento percebido pela análise da informação disponibilizada pelo sistema.

Em seguida foi perguntado aos estudantes se consideravam importante que os professores pudessem enviar *feedback* aos estudantes que ficasse visível na ferramenta de monitorização, tendo todos os estudantes respondido afirmativamente. Convirá a este respeito lembrar que no caso desta experiência, para além de poder criar um clone do código do estudante no qual podia fazer modificações e inserir comentários, tal como na experiência descrita anteriormente, o professor poderia também fornecer *feedback* aquando da correção manual dos exercícios. O professor acabaria por não utilizar muito frequentemente a primeira funcionalidade, tendo em oposição utilizado muito frequentemente a segunda.

A seguir foram solicitadas críticas/sugestões de melhoria da ferramenta de monitorização, tendo um estudante sugerido que em caso de os exercícios não passarem nos testes devia ser assinalado no código o sítio do erro.

Para terminar, foram solicitadas críticas ou sugestões de melhoria relativamente a todo o sistema. Um estudante referiu que não deveria ser obrigatória a utilização do *Atom* como ambiente de desenvolvimento. Este mesmo estudante indicou que deveria existir um mecanismo que permitisse comentar as soluções dos exercícios propostas pelo professor.

Vários estudantes mencionaram a necessidade de serem disponibilizados mais casos de teste e o facto já mencionado anteriormente de por vezes os exercícios serem classificados como "quase certos" por motivos de pequenas incompatibilidades de caracteres.

Alguns estudantes mencionaram que o sistema deveria permitir a utilização de mais do que um ficheiro de código para cada exercício, uma vez que deste modo seria possível continuar a utilizar o sistema até ao final. Esta questão também já havia sido referida na experiência anterior e mesmo em outras iterações, pelo que irá certamente merecer a nossa atenção, contudo, é um problema um pouco mais complexo do que poderá parecer à partida e implicará a imposição de determinadas regras, como a utilização de projetos diferentes para a resolução de cada exercício, por exemplo.

Finalmente, alguns estudantes mencionaram que tinham gostado muito de utilizar o sistema, pois para além de toda a informação que esta lhes fornecia, a sua utilização fazia também que nunca se esquecessem de submeter nenhuma solução, uma vez que estas eram submetidas de forma automática.

4.8.4.2.2. Entrevista com o professor

Ainda que tivesse existido uma forte proximidade com o professor ao longo de toda a experiência, no final desta, achou-se ainda assim pertinente fazer uma breve entrevista com ele. Uma vez mais, optou-se pela realização de uma entrevista em que tinha como ponto de partida uma estrutura previamente definida (Anexo XI), mas que em todo o caso permitia acomodar outras questões que se revelassem pertinentes.

Assim, relativamente à primeira questão, sobre a regularidade com que utilizava o sistema, o professor referiu que utilizava o sistema várias vezes ao dia, durante todos os dias da semana, incluindo fins de semana.

Sobre o modo como utilizava o sistema, o professor indicou que utilizava o sistema de duas formas distintas. Por um lado, utilizava o sistema durante as aulas para seguir em tempo real a performance dos estudantes, o que lhe permitia identificar situações em que a sua intervenção atempada poderia ser determinante. Para além disso, uma vez que tinha optado por proceder à correção "manual" de todos os exercícios, utilizava muito a

ferramenta para suportar o processo de correção dos exercícios, lançamento das classificações e principalmente, fornecimento do *feedback* aos estudantes.

Atendendo à resposta que o professor deu à pergunta anterior, não será de admirar que o professor tenha respondido que as funcionalidades mais utilizadas foram aquelas mais intimamente ligadas ao processo de correção dos exercícios e teste do código escrito pelos estudantes. Contudo, para além destas funcionalidades, mencionou também a matriz resumo e a *live feed*, que mencionou utilizar frequentemente durante as aulas.

Relativamente à questão seguinte, o professor, indicou não ter qualquer dúvida que graças à utilização do sistema conseguiu conhecer os estudantes e o seu desempenho ao nível da programação, de uma forma muito interessante. O simples facto de a informação ser disponibilizada de forma agregada e sempre disponível, constituía por si só uma enorme mais-valia, na opinião deste professor. Um aspeto do qual o professor gostou muito foi o facto de poder acompanhar o trabalho dos estudantes, mesmo fora do âmbito das aulas.

O professor indicou que ao longo de toda a experiência utilizou informação obtida do sistema para dar *feedback* personalizado aos estudantes, mas também para a generalidade da turma. Mencionou ainda que muitas vezes utilizava as soluções que ia vendo no sistema, como ponto de partida da explicação de algum conceito que aparentemente estivesse a ser mal compreendido/utilizado pelos estudantes.

Em relação a dificuldades na interpretação da informação apresentada e ao nível da usabilidade do sistema, o professor foi fazendo uma série de pedidos de melhoria ao longo da experiência, tais como alterações de cores ou redimensionamento de elementos visuais, pelo que algumas dificuldades que havia sentido tinham sido prontamente resolvidas na altura.

Em seguida, o professor foi questionado, relativamente à necessidade de notificações automáticas, tendo mencionado que atendendo ao facto de ele utilizar o sistema com muita frequência e toda a informação ser disponibilizada de outras formas, não sentiu verdadeiramente a necessidade de ser notificado pelo sistema, contudo, não colocou de parte a utilização desta funcionalidade, principalmente em relação a aspetos menos óbvios como a existência de plágio, por exemplo.

Na continuação, sobre a relação entre o sistema e os estudantes, o professor indicou que no geral, ficou com a sensação de que a maioria dos estudantes gostava de utilizar o sistema, principalmente, por este testar automaticamente o código. Sentiu, ainda assim,

relativamente a alguns estudantes alguma frustração, principalmente após terem sido abordados relativamente a situações relacionadas com plágio recorrente. Em todo o caso, no geral, sentiu uma grande aceitação por parte dos estudantes e alguma pena quando disse aos estudantes que a partir de dada altura o sistema deixaria de ser utilizado.

Sobre a validade da imagem dos estudantes que o sistema oferece aos professores, o professor indicou que na generalidade, esta é válida, contudo, principalmente devido ao plágio, a imagem de alguns estudantes está de facto um pouco distorcida, pois na verdade, estes estudantes não são "tão bons" quanto o sistema pode fazer crer. Em todo o caso, uma vez que o sistema também oferece funcionalidades ligadas à identificação de plágio, logo que sejam acauteladas estas situações, a generalidade da informação disponibilizada, é, na opinião do professor, uma boa representação da realidade.

Relativamente à utilização do sistema em futuras edições desta ou de outra unidade curricular de programação, o professor, afirmou que tendo em conta a informação útil que conseguia obter da utilização do sistema, iria sem dúvida utilizar o sistema futuramente.

Para terminar, foi solicitado ao professor que indicasse sugestões de melhoria que em seu entender fizessem sentido. A este respeito, o professor foi perentório em afirmar que o sistema deveria disponibilizar a possibilidade de realizar mais casos de teste, quer na forma mais "tradicional", principalmente relevante nas primeiras fichas de trabalho, quer na sua forma mais "avançada", semelhante a testes unitários, que haviam sido introduzidos no sistema, mas apenas realizados a pedido do professor. Assim, no entender do professor, para além de disponibilizar mais casos de teste, deveria ser possível configurar para cada exercício qual o tipo de testes a realizar. Em ambos os casos, os testes deveriam ser realizados aquando da receção dos *snapshots* e os resultados destes deveriam ser disponibilizados quer aos professores, quer aos estudantes, ainda que fosse necessário proceder a algumas modificações na forma como os resultados eram apresentados. Por exemplo, em vez de utilizar dois quadrados coloridos na listagem dos exercícios, poderia passar a utilizar-se uma representação textual, como por exemplo "3 / 4" com o significado de que foram realizados quatro testes, tendo o código passado em três deles.

4.8.5. Sumário da iteração sete

Como apresentado anteriormente, foram feitas diversas modificações ao sistema no sentido de o tornar ainda mais útil para os professores, idealmente contribuindo para que estes o considerassem mais relevante e, como tal, o usassem mais. Como foi visto em experiências anteriores, alguns estudantes mostraram-se algo renitentes em utilizar o sistema pois não sentiam no seu dia-a-dia que os professores estavam de facto a utilizá-lo, criando deste modo alguma desconfiança. Pretendeu-se, pois, verificar se o aumento da utilização do sistema pelos professores e o consequente aumento da visibilidade do seu uso, contribuíram para o aumento da aceitação do sistema pelos estudantes. No geral, verificou-se que os estudantes sentiram de facto que o sistema estava a ser utilizado pelos professores e não se sentiram incomodados por isso. Pelo contrário, diversos estudantes mencionaram que se sentiam mais seguros ao saberem que os professores estavam a monitorizar o seu trabalho, tendo alguns mencionado que gostariam apenas de ter recebido mais *feedback*. Este sentimento foi, contudo, mais evidenciado na experiência do programa Acertar o Rumo, no qual participam estudantes com uma maturidade maior.

Ainda que o sistema tivesse desde o início sido desenvolvido tendo como destinatários principais os professores, já na iteração zero, havia sido considerado importante a disponibilização de *feedback* aos estudantes. As últimas experiências demonstraram que a disponibilização de *feedback* automático, mas também *feedback* individualizado por parte dos professores constituem um aspeto determinante para a aceitação do sistema pelos estudantes. Também a este nível, foi considerado que algumas das modificações que foram implementadas ao nível da disponibilização de *feedback* automático e "manual", contribuíram para o fortalecimento da aceitação da utilização do sistema pelos estudantes. Um sinal claro deste aumento da aceitação foi o facto de mesmo depois de a experiência ter terminado, os estudantes terem pedido para continuar a utilizar o sistema e em ambas as experiências terem mencionado como uma evolução desejada, uma funcionalidade que lhes permitisse utilizar o sistema durante mais tempo.

Apesar do aumento da visibilidade da utilização do sistema pelos professores, ou eventualmente, por causa desse aumento, continuaram a existir alguns estudantes que manifestaram reservas em partilhar todo o seu trabalho com os professores. É uma situação compreensível, que não é causada por questões de privacidade, mas sim por motivos de alguma proteção da imagem dos estudantes perante os professores. Com

efeito, a partir do momento que estes estudantes tinham uma solução, em seu entender minimamente aceitável, não tinham grandes reservas em fazer as submissões de forma identificada. O grande problema era o caminho desde o início até chegar a esse ponto e o receio de exporem algumas fraquezas, e que obviamente deverá ser respeitado, até porque muito provavelmente estes estudantes teriam, eventualmente, um comportamento ainda mais introvertido no "mundo real".

O facto de o sistema ter sido utilizado nas experiências desta iteração maioritariamente por professores que nunca o tinham feito no passado, revelou-se também muito interessante, uma vez que fizeram diversas solicitações de funcionalidades que ou não existiam de todo ou não tinham sido consideradas relevantes pelos professores que o haviam utilizado no passado.

Ainda que, como seria de esperar, existam alguns pormenores para acertar, no geral, constatou-se que quer os estudantes, quer os professores encararam o sistema como uma ferramenta muito útil para o desempenho das suas tarefas.

4.9. Iteração oito

4.9.1. Objetivos

Após a iteração anterior, as principais questões que haviam sido colocadas originalmente para este trabalho já haviam sido respondidas. Na realidade, a iteração oito não estava prevista, contudo, por ocasião do surgimento da pandemia de COVID-19 e consequente confinamento e passagem para um regime do que diversos autores chamaram de *emergency remote education* [149], a utilização do sistema ***CodeInsights*** acabaria por tornar-se importante para o acompanhamento do trabalho dos estudantes.

Neste contexto, para esta iteração não foram implementadas novas funcionalidades além de algumas correções pontuais. O sistema acabaria por ser utilizado como ferramenta efetiva de apoio à monitorização das aulas remotas. Era igualmente a oportunidade ideal para testar o sistema num contexto em que não existia contacto direto presencial entre professores e estudantes. Para além disso, aproveitou-se a oportunidade para experimentar o sistema de uma forma ligeiramente diferente face às experiências anteriores, permitindo que os estudantes pudessem decidir quando e quais *snapshots* eram enviados para o servidor, ao invés do envio automático de todos os *snapshots*.

4.9.2. Descrição do contexto

Nesta iteração o sistema foi utilizado para suportar a monitorização do trabalho dos estudantes das unidades curriculares Programação para a Internet II e Desenvolvimento Web II das licenciaturas em Engenharia Informática e Sistemas e Tecnologias da Informação respetivamente, ambas da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital. Neste ano letivo, estas unidades curriculares foram lecionadas conjuntamente.

4.9.3. Avaliação do sistema

O sistema foi utilizado no segundo semestre do ano letivo 2019/2020 entre os dias 12 de março e 22 de maio de 2020. Ambas as unidades curriculares estavam a decorrer em simultâneo e no total estavam inscritos um total de 45 estudantes. Durante a experiência foram propostos um total de 36 exercícios de treino organizados em 10 fichas de trabalho, aos quais acresciam mais dois exercícios de complexidade um pouco maior e cuja resolução era tida em conta para a definição da classificação final da unidade curricular.

Uma vez que nas experiências anteriores diversos estudantes haviam revelado algum tipo de constrangimento em ter *snapshots* de todas as suas tentativas de resolução enviadas para o servidor, tendo nos inquéritos indicado que gostariam de poder escolher quando e quais os *snapshots* que eram enviados, aproveitou-se esta oportunidade para testar o funcionamento do sistema com base neste contexto.

Como era de esperar, com este novo modo de funcionamento o número de *snapshots* recebidos foi muito menor do que aquele registado em outras experiências, inclusive da mesma unidade curricular. Com efeito, no somatório de todos os exercícios foram recebidos apenas aproximadamente 2400 *snapshots* de um total de 33 estudantes. Uma parte considerável destes *snapshots* diziam respeito aos dois exercícios que eram tidos em conta para efeitos de avaliação (30%), contudo, mesmo em relação a estes exercícios os estudantes submetiam principalmente as últimas tentativas de resolução. Relativamente aos restantes exercícios, os *snapshots* que foram recebidos tiveram principalmente como origem os estudantes mais participativos e que acabavam por fazer as submissões para mostrar trabalho feito, mas também para suportar a colocação de dúvidas ao professor que estava a acompanhar a aula via *Zoom*. Por vezes, fora do período das aulas, os estudantes pediam ao professor, via email e/ou *Discord*, se este

podia ver a última submissão para um determinado exercício e comentar ou ajudar a ultrapassar alguma dificuldade que estivessem a ter no momento.

Com efeito, entre os 10 estudantes que fizeram submissões para mais exercícios, apenas três não obtiveram aprovação. Um deles, que até dado momento tinha sido um dos mais participativos, eventualmente por motivos pessoais, acabaria por não entregar o último exercício avaliado nem o trabalho prático final e dois outros estudantes repetentes e que tinham muitas dificuldades e que foram utilizando o sistema para expor as suas dúvidas.

Tal como nas experiências anteriores, no final foi disponibilizado um inquérito aos estudantes por forma a saber a sua opinião e idealmente obter informações relevantes para fazer evoluir o sistema. No Anexo XII poderá ser encontrada a lista das perguntas do inquérito que foi realizado via *Google Forms*.

No total foram obtidas 12 respostas, o que apesar de tudo não é um número muito negativo, tendo em conta que dos 33 que fizeram alguma submissão, apenas catorze acabaram por se submeter a qualquer avaliação, principalmente devido à falta de bases de programação e resolução de problemas.

Relativamente à primeira pergunta "*Com que regularidade assistiu às aulas?*", 75% responderam "*sempre*", 16,7% responderam "*muito regularmente*" e 8,3% responderam "*por vezes*".

Na questão seguinte, era perguntado aos estudantes "*Com que regularidade resolvia os exercícios propostos pelo professor?*". A esta pergunta, 50% dos estudantes responderam "*por vezes*", 33% reponderam "*muito regularmente*" e apenas 16,7% responderam "*sempre*". Estes resultados são interessantes pois acabam por ser reveladores do cenário que muito possivelmente se verificava anteriormente à utilização do sistema, ou seja, uma vez que eram os estudantes que decidiam quando e o que submeter não sentiam a pressão de mostrar trabalho feito e como tal, acabavam por simplesmente não fazer. Se num cenário de aulas presenciais ainda era possível ir obtendo alguma informação através da observação direta, num cenário de ensino remoto, esta tarefa era impossível sem a participação dos estudantes na utilização do sistema. Esta forma de proceder, dificulta a tarefa do professor em avaliar devidamente o progresso dos estudantes, assim como a identificação das dificuldades que possam estar a sentir a dada altura.

Em relação à questão seguinte "*Com que regularidade fazia a submissão das soluções por si desenvolvidas no sistema CodeInsights?*", os resultados das respostas acabam por reforçar o que foi dito anteriormente. Assim, apenas um estudante (8,3%) responder que

submetia sempre as suas soluções no ***CodeInsights***, 41,7% responderam "*muito frequentemente*", 25% responderam "*muitas vezes*", 16,7% responderam "*poucas vezes*" e um estudante (8,3%) respondeu "*nunca*". Na continuação desta pergunta, solicitava-se aos estudantes que dessem uma justificação para a frequência de submissão das suas soluções. Entre as respostas obtidas, convém destacar as seguintes; "*Nem sempre realizava os exercícios durante a aula, e muitos eram só para praticar, logo nem sempre submetia*", "*Não eram para avaliação, não submeti*", "*Falta de acompanhamento semanal para estar preparado para as aulas seguintes. Esta falta de acompanhamento deveu-se ao tempo despendido noutras cadeiras*".

Em seguida foi perguntado aos estudantes "*Ao longo deste período verificou que, no seu dia a dia, o professor utilizava realmente o sistema para preparar as aulas e para dar assistência aos estudantes?*", tendo 83,3 % respondido que sim e os restantes 16,7% que não. Na continuação desta pergunta, era solicitado aos estudantes para fornecerem alguns exemplos concretos de utilização do sistema pelo professor. Apresentam-se em seguida algumas das respostas dadas pelos estudantes: "*O prof providenciava feedback às soluções submetidas.*", "*Apesar de eu não entregar no codeinsights, tenho conhecimento que os meus colegas verificavam na plataforma que o professor já tinha visto ou não as coisas submetidas.*", "*Era sempre apresentado feedback, no entanto quando o código era re-submetido nem sempre*", "*Durante a resolução de exercícios que eram propostos pelo professor*". Como é possível ver por estas respostas, a grande maioria dos estudantes que responderam ao inquérito puderam de facto verificar, que apesar de serem feitas poucas submissões, o professor conseguia ainda assim intervir no sentido de ajudar os estudantes através do fornecimento de *feedback*.

Ainda relativamente às intervenções do professor suportadas pela informação obtida no sistema, perguntou-se em seguida aos estudantes se tinham achado estas intervenções relevantes. A esta questão, 83,3% dos estudantes responderam afirmativamente, tendo os restantes 16,7% respondido "nem sempre". Curiosamente, os estudantes que responderam "nem sempre" foram os mesmos que responderam que não tinham tido perceção das intervenções feitas pelo professor e também estavam entre os que assistiam às aulas e submetiam as soluções no ***CodeInsights*** com menor regularidade.

Em seguida foi perguntado aos estudantes "*Ao contrário do que havia acontecido no passado, desta vez, as submissões não eram enviadas de forma automática, podendo os estudantes escolher o que submeter e quando submeter. Prefere desta forma?"*. A esta pergunta 75% dos estudantes responderam que preferiam desta forma, um estudante

(8,3%) respondeu que preferia da forma antiga e dois (16,7%) responderam que "*era indiferente*". Por forma a tentar dar mais significado a estas respostas, pedia-se em seguida para os estudantes justificarem a sua resposta à pergunta anterior. Foram recebidas as seguintes respostas: "*Muitas vezes o primeiro código realizado não está bem e nós temos noção disso, porém só voltamos a pegar nos exercícios algum tempo depois, logo é muito melhor temos o controle de quando enviar os exercícios.*", "*Dá mais liberdade de escolha de participação por parte do estudante.*", "*o feedback do professor é importante*" e "*O Estudante é que decide onde é que sente dificuldade e onde gostava de obter feedback do professor, diminuindo a carga do professor e tornando mais eficiente a aprendizagem do estudante*".

Tal como já havia sido verificado em outras experiências, os estudantes gostam de obter *feedback* dos professores, contudo preferem ser eles a decidir quando pretendem que esse *feedback* possa chegar. Ainda que esta linha de pensamento seja compreensível, a verdade é que o objetivo do sistema não é fazer com que os professores não deem espaço e autonomia aos estudantes para resolverem eles próprios os seus problemas. Pelo contrário, o sistema deverá permitir aos professores identificarem se os estudantes estão ou não a sentir dificuldades, em caso afirmativo, verificar o que já foi feito para as tentar ultrapassar e só em último caso proceder às intervenções consideradas necessárias. Se não existir informação nenhuma sobre o desempenho dos estudantes, então o professor pura e simplesmente não pode intervir proativamente, levando a que muitos estudantes pura e simplesmente não vejam as suas dúvidas esclarecidas e/ou tentem esclarecer as suas dúvidas de forma incorreta ou pura e simplesmente façam plágio.

Seguiu-se uma questão sobre a regularidade com que acederam à ferramenta de monitorização destinada aos estudantes, tendo 75% respondido "Algumas vezes por semana", 16,7% responderam "*Muito raramente*" e um estudante (8,3%) respondeu "*Várias vezes ao dia*". Estas respostas acabam por ser interessantes e consistentes com as restantes. Tendo em consideração que alguns estudantes indicaram que não faziam submissões no sistema regularmente, seria de esperar que também não acedessem no sentido de obter *feedback* e/ou avaliar o seu progresso.

Todos os estudantes responderam afirmativamente em relação a "*Considera importante a disponibilização deste tipo de ferramenta que permita aos estudantes ter uma ideia do trabalho efetuado?*", o que não deixa de ser curioso tendo em conta que alguns deles não faziam submissões regularmente.

Em relação a sugestões para a melhoria da ferramenta destinada aos estudantes, foram obtidas as seguintes respostas: "*Uma forma de comunicar com o professor, e avisar que tem duvidas naquele exercício sem ter que o fazer por comentários no código.*" e "*Sim, gostava de ver adicionada uma ferramenta que permita enviar mensagens ao professor juntamente com o código, sem ser necessariamente um comentário perdido no meio do código*". Ambas as sugestões apontam para a necessidade de criação de um mecanismo que permita aos estudantes expor as suas dúvidas aos professores diretamente no sistema. São sem dúvida sugestões pertinentes relativas a funcionalidades que já haviam sido equacionadas, mas que não foram ainda completamente implementadas.

Seguiu-se uma questão em que era solicitado aos estudantes para avaliarem a utilização do sistema ***CodeInsights*** naquela unidade curricular numa escala de 1 (muito má) a 5 (muito boa), tendo dois estudantes (16,7%) indicado 3, ou seja, uma opinião neutra, seis estudantes (50%) indicaram 4 ("boa") e os restantes quatro estudantes (33,3%) responderam 5 ("muito boa"). Uma vez mais, os estudantes que responderam de forma neutra foram aqueles que revelaram anteriormente não ter participado muito ativamente nas atividades desta unidade curricular. Ainda que os resultados pudessem ter sido melhores, a verdade é que esta experiência acabaria por ocorrer em pleno confinamento, que por si só já constituiu um grande desafio para o desempenho de estudantes e professores.

Finalmente, para terminar o inquérito era solicitado aos estudantes para fazerem alguma crítica e/ou sugestão final, tendo sido apenas recebida a seguinte resposta: "*Outros professores também deveriam utilizar esta ferramenta de forma a certificarem-se que os estudantes estão num bom caminho, especialmente nas UCs em que os estudantes estão a aprender linguagens novas*". Ainda que com base no *feedback* obtido anteriormente por parte dos estudantes e professores que já utilizaram o sistema não restassem dúvidas em relação à pertinência da utilização deste sistema, esta resposta acaba por constituir um reforço e fonte extra de motivação para continuar a melhorar o sistema para que mais estudantes e professores possam beneficiar da sua utilização nos seus processos de ensino/aprendizagem de programação.

4.9.4. Sumário da iteração oito

Após todas as experiências anteriormente realizadas em contexto de sala de aula, existia já uma noção clara de que o sistema permitia de facto aumentar a perceção que os

professores conseguiam ter em relação aos seus estudantes. Ainda assim existiu sempre a vontade de testar a utilização do sistema num contexto de ensino não presencial, o que infelizmente não tinha sido possível até então.

Obviamente, não era de todo previsível que viesse a ocorrer a pandemia provocada pelo vírus SARS-cov-2. Contudo, após o primeiro impacto, acabou por se revelar uma oportunidade interessante para, não só testar o sistema num novo contexto, como também testar algumas modificações no modo como o sistema funcionava tendo em consideração algum do *feedback* obtido nas experiências anteriores.

Uma das principais modificações introduzidas nesta iteração consistia em permitir que os estudantes escolhessem quando e quais as submissões que eram enviadas para o servidor. Como visto em experiências anteriores, muitos estudantes demonstraram algumas reservas em ver todas as suas tentativas de resolução enviadas automaticamente para o servidor, pelo que decidiu-se aproveitar esta oportunidade para testar o funcionamento de forma alternativa e que consistia em deixar os estudantes decidir quais as submissões que pretendiam enviar para o sistema. Esta decisão acabaria por se revelar inadequada pois perante a possibilidade de escolher quando eram feitas as submissões, a maior parte dos estudantes acabou por fazer apenas a última (quando o código já estava completo), o que na prática acaba por invalidar muito do funcionamento do sistema, já que era perdida toda a informação relativa ao processo de construção da solução assim como as dificuldades sentidas. No entanto, continuou a ser possível acompanhar o progresso dos estudantes de uma forma global e o sistema continuou a ser relevante para o esclarecimento de dúvidas e fornecimento de *feedback* sobre as submissões finais (ou quase).

Muito embora tal como visto anteriormente, alguns estudantes procedam de forma a anonimizar algumas submissões, tendo por base o modo como acabaria por decorrer esta experiência, foi considerado preferível voltar ao modo original de funcionamento do sistema. Deverá ainda assim incentivar-se os professores a reforçar o motivo pelo qual se utiliza o sistema e os benefícios decorrentes do seu uso e aceitar à partida que uma pequena percentagem dos estudantes continuará a não se sentir à vontade em partilhar com o professor o processo de construção das suas soluções.

Não se tratou de modo algum de uma oportunidade perdida, até porque serviu para testar a utilização do sistema num novo contexto e com uma nova configuração.

4.10. Iteração nove

4.10.1. Objetivos

Infelizmente a pandemia acabou por ganhar proporções maiores do que eventualmente se tinha antecipado, de tal forma que a "segunda vaga" acabaria por conduzir o país (e o mundo) uma vez mais a uma situação de confinamento parcial ou total, que na prática fez com que na maior parte das instituições de ensino superior as aulas decorressem em regime misto (presencial/*online*) ou exclusivamente *online* [150].

Perante este contexto, um professor que já havia utilizado o sistema anteriormente, manifestou a intenção de utilizar o sistema para o suporte das suas unidades curriculares de introdução à programação com C.

Uma vez mais, foram devidamente explicados os objetivos da utilização do sistema, foi fornecida toda a documentação necessária. Dadas as circunstâncias, o documento de consentimento para a recolha de dados não foi fornecido, mas foi explicado aos estudantes que poderiam optar por resolver os exercícios sem recorrer à utilização do sistema ***CodeInsights*** sem que houvesse qualquer consequência. Em todo o caso, se optassem por utilizar o sistema, estariam automaticamente a dar consentimento para a recolha de informação.

Tendo em conta o modo como tinha corrido a experiência na iteração anterior, voltou-se a optar por serem feitas as submissões de forma automática sempre que os estudantes compilassem o seu código.

4.10.2. Descrição do contexto

O sistema foi, pois, utilizado para suportar a atividade de professor e estudantes de duas unidades curriculares de introdução à programação com a linguagem C das licenciaturas em Engenharia Informática e Sistemas e Tecnologias da Informação ambas da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital, mais concretamente Fundamentos de Programação e Programação I.

4.10.3. Trabalho desenvolvido

4.10.3.1. Desenvolvimento dos *plug-ins*

Em termos gerais, foram feitas algumas modificações no *plug-in* destinado aos IDEs da *JetBrains* (IntelliJ, PyCharm, CLion, etc.), uma vez que a passagem a *deprecated* de um conjunto de classes que eram utilizadas na construção do *plug-in* inviabilizavam a sua utilização de futuro. Uma vez que desde a criação deste *plug-in* o sistema havia passado a suportar a execução de casos de teste, foi aproveitada a ocasião para desenvolver uma funcionalidade que permitisse disponibilizar a informação sobre a execução do código com os casos de teste, assim como a apresentação de *feedback* personalizado para aquele exercício, no caso de o professor o ter definido.

Foi ainda desenvolvido um novo *plug-in* para o editor ***Visual Studio Code.*** Uma das principais vantagens deste *plug-in* era o facto de poder ser utilizado com qualquer linguagem de programação suportada por este ambiente de desenvolvimento. À semelhança dos restantes *plug-ins*, também era fornecida a possibilidade de serem apresentados os resultados da execução do código com os casos de teste (Figura 135).

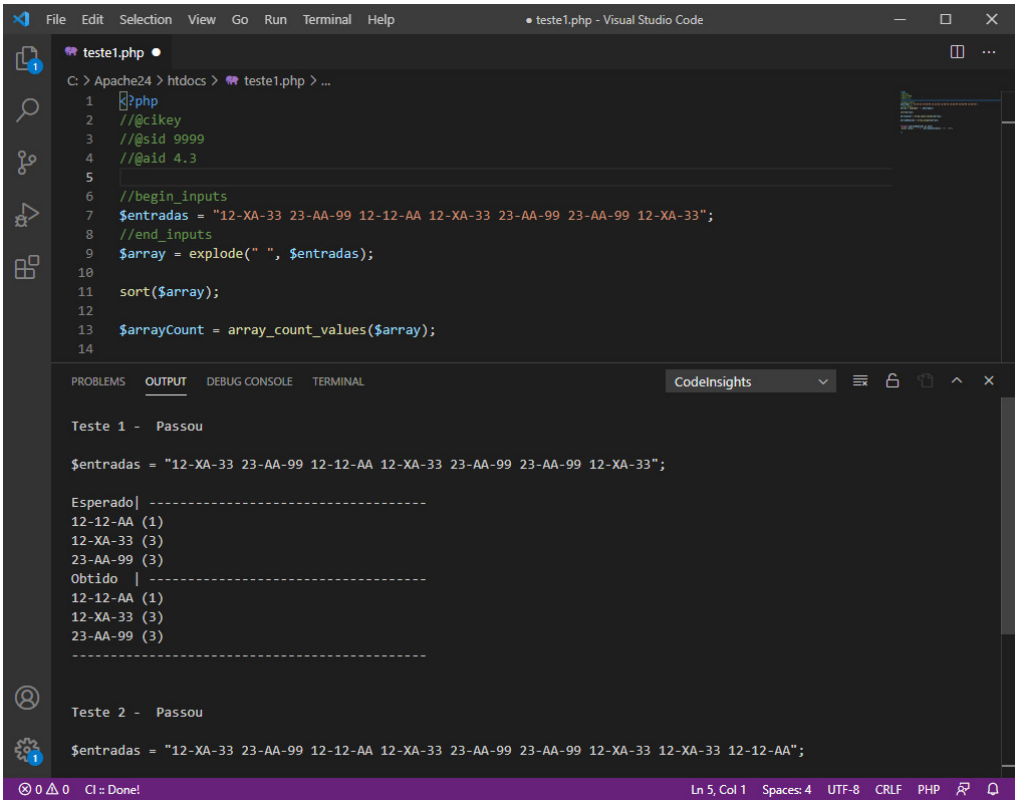

Figura 135 - Utilização do plug-in no Visual Studio Code.

4.10.3.2. Ferramenta de processamento do *código*

Numa utilização anterior do sistema uma professora fez uma sugestão interessante relacionada com o facto de o sistema apenas poder ser utilizado para uma linguagem de programação de cada vez, ou seja, se por algum motivo os professores pretendessem utilizar mais do que uma linguagem de programação em diferentes momentos, era necessário instalar duas instâncias do sistema. Por forma a mitigar esta limitação, a informação relativa à linguagem de programação foi desvinculada da "instalação do sistema" e passaria a estar associada a cada exercício. Deste modo, um exercício poderia ser proposto para ser resolvido em HTML e outro em JavaScript (node.JS), por exemplo. Por forma a permitir esta modificação, foi necessário fazer algumas modificações profundas neste componente do sistema. Caso algum dia venha a ser necessário, poder-se-á facilmente modificar o sistema por forma a permitir que um mesmo exercício possa ser resolvido em várias linguagens de programação diferentes.

4.10.3.3. Ferramenta de monitorização - professores

No que diz respeito à ferramenta de monitorização destinada aos professores foram feitas alterações ao nível da funcionalidade de criação/edição dos exercícios, nomeadamente de forma a permitir disponibilizar aos estudantes um código de *scaffolding*, ou seja, código parcialmente escrito que os estudantes deveriam concluir. Foram ainda adicionados alguns campos que permitem aos professores especificar a linguagem de programação em que o exercício deveria ser resolvido, se a classificação manual atribuída a um exercício deverá ou não ser apresentada aos estudantes ou se o um exemplo de resolução dos exercícios deverá ou não ser disponibilizado aos estudantes.

Tendo em conta que previamente alguns professores tinham dito que a disponibilização de apenas dois casos de teste para cada exercício poderia ser algo limitativa, este número foi aumentado para quatro. Além disso, foram feitas alterações em várias páginas da aplicação por forma a refletir esta modificação. Assim, ao contrário do que ocorria anteriormente, em que eram apresentados quadrados coloridos para representar o resultado obtido em cada um dos testes, passou a ser disponibilizado apenas um retângulo com informação relativa ao número de testes em que o código passou face ao número de testes realizados. A cor de fundo deste retângulo permite reforçar o resultado obtido, assim, verde indica que passou em todos os testes; amarelo indica que o código passou em alguns dos casos de teste realizados; finalmente, cor de laranja indica que o

código não passou em nenhum caso de teste. Na Figura 136 pode ser observado um exemplo da aplicação desta nova forma de apresentar a informação.

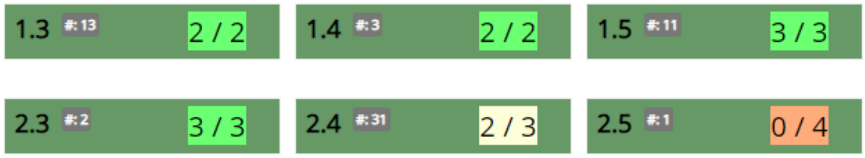

Figura 136 - Exemplo da nova forma de representar os resultados de testar o código com os casos de teste - iteração nove.

Na página destinada à apresentação dos detalhes de cada tentativa de resolução, no caso de a linguagem de programação ser HTML ou JavaScript, passaram a ser disponibilizados dois botões adicionais "*Preview*" e "*Preview in new windows*", que permitem aos professores ver a página HTML "renderizada" e testar as suas funcionalidades, tal como pode ser visto no exemplo apresentado na Figura 137.

Figura 137 - Exemplo da pré-visualização de um ficheiro HTML realizado por um estudante.

4.10.3.4. Ferramenta de monitorização - estudantes

Motivado principalmente pela modificação do número de casos de testes que era possível utilizar e o modo como a informação dos resultados era apresentada, foram igualmente efetuadas algumas ligeiras modificações na lista de exercícios apresentada aos

estudantes. Ao contrário do que acontecia anteriormente, passaram a ser apresentados também os exercícios já disponibilizados pelos professores, mas para os quais o estudante ainda não fez nenhuma tentativa de resolução. Deste modo o estudante poderia ter uma melhor ideia dos exercícios que já tinha resolvido e os que ainda lhe faltavam resolver. Na Figura 138 é apresentado um exemplo da nova configuração da listagem.

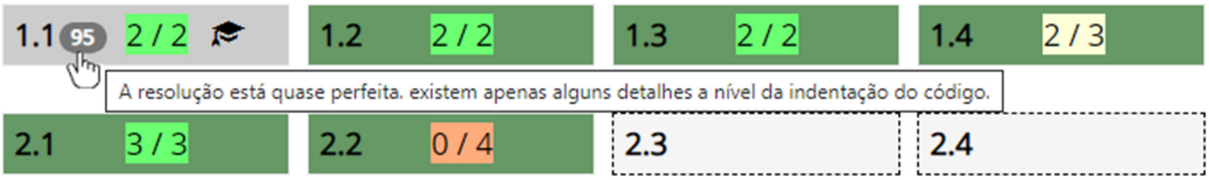

Figura 138 - Exemplo da listagem de exercícios destinada aos estudantes - iteração nova.

Foi ainda modificado o significado que anteriormente havia sido dado ao ícone do capelo. Nesta nova versão do sistema, a existência deste ícone indica que o professor disponibilizou uma proposta de resolução para este exercício. Os estudantes e o professor envolvidos nesta iteração nunca haviam utilizado a versão anterior em que o ícone tinha outro significado, pelo que a sua modificação não trará problemas ao nível de consistência. Ao clicar neste ícone, o estudante terá acesso a uma página com exemplos de resolução do exercício em causa.

Como mencionado anteriormente, passou a ser possível aos professores disponibilizar código de *scaffolding* que os estudantes deveriam utilizar como ponto de partida para a resolução de um determinado exercício. Assim, na ferramenta destinada aos estudantes passou a ser possível aceder aos detalhes de um determinado exercício e aceder a esse código, caso tenha sido disponibilizado.

Além disso, passou igualmente a ser possível resolver o exercício diretamente no ***CodeInsights***, usando um editor de código disponibilizado para o efeito. Este editor permite igualmente testar o código e ver os resultados dos testes, tal como se o estivessem a fazer num dos editores para os quais era disponibilizado um *plug-in*. A qualquer momento, os estudantes poderiam também aceder à última tentativa de resolução de um exercício e continuar a resolução do mesmo. Na Figura 139 pode ser observado um exemplo de utilização deste editor de código.

No caso concreto da linguagem HTML e JavaScript, ao fazer "Run" é apresentada a renderização do código escrito e os estudantes poderão interagir com ela.

Show assignment details

Run

```
1  #include <stdio.h>
2  //@cikey [illegible]
3  //@sid [illegible]
4  //@aid 6.3
5
6  int letra(char c){
7    return (c>=65 && c<=90) || (c>=97 && c<=122);
8  }
9
10 char minuscula(char c){
11   char result;
12   if(c>=65 && c<=90)
13       result = c + 32;
14   else
15       result = c;
16   return result;
17 }
18
19 int main()
20 {
21     //begin_inputs
22   char caractere;
23   printf("Entre com o caracter: ");
24   scanf("%c", &caractere);
25     //end_inputs
26   printf("caractere %c : ", caractere);
27   if(letra(caractere))
28     printf("letra - minuscula: %c", minuscula(caractere));
29   else
30     printf("nao letra");
31
```

```
Snapshot processado corretamente!

Teste 1 -  Passou

char caractere = 'A';

Esperado| ----------------------------
caractere A : letra - minuscula: a
Obtido  | ----------------------------
caractere A : letra - minuscula: a
-------------------------------------


Teste 2 -  Passou

char caractere = '9';

Esperado| ----------------------------
caractere 9 : nao letra
Obtido  | ----------------------------
caractere 9 : nao letra
-------------------------------------


Teste 3 -  Passou

char caractere = 'b';

Esperado| ----------------------------
caractere b : letra - minuscula: b
Obtido  | ----------------------------
caractere b : letra - minuscula: b
```

Figura 139 - Exemplo de utilização do editor de código disponibilizado no sistema - iteração nove.

Como referido anteriormente, consideramos, que na medida do possível, os estudantes deverão utilizar as ferramentas que muito provavelmente irão utilizar num contexto real. Assim, a disponibilização de um editor de código *web-based* integrado diretamente no sistema, poderá parecer contraditória. No entanto, a sua criação permite que os estudantes que não pretendam instalar os plug-ins disponibilizados ou que queiram utilizar um outro editor, possam fazer as suas submissões, ainda que sejam apenas as versões finais e no caso de ser necessário fazer algumas modificações, o possam fazer diretamente no sistema. Esta necessidade acaba por ser ainda mais relevante para situações em que as submissões são tidas em consideração para efeitos de avaliação e por uma questão de conveniência, o sistema ***CodeInsights*** seja a única forma de submissão considerada válida.

Em todo o caso, é possível que um estudante consiga resolver todos os exercícios utilizando diretamente o editor de código disponibilizado, sem nunca necessitar de instalar qualquer *software* no seu computador, podendo deste modo resolver os exercícios em qualquer lado e em qualquer dispositivo, incluído no seu *smartphone* ou *tablet*.

4.10.4. Avaliação do sistema

O sistema foi avaliado no contexto de uma unidade curricular introdutória de programação em que a linguagem C é utilizada. O professor responsável já tinha utilizado o sistema em anos anteriores, num contexto presencial e deste modo já tinha uma noção de como a utilização da ferramenta de monitorização poderia ser útil no contexto do confinamento.

A avaliação ocorreu entre os dias 14 de outubro e 6 de dezembro de 2020. Um total de 73 estudantes estavam inscritos nesta unidade curricular, tendo sido recebida pelo menos uma tentativa de resolução de 61 estudantes. Alguns dos restantes estudantes, embora inscritos, não frequentaram efetivamente a unidade curricular.

Durante este período, o professor forneceu seis fichas de trabalho, cada uma delas relacionada com um tema principal, como o uso básico de variáveis, condições, ciclos, manipulação de *arrays*, ou manipulação de ficheiros. No total, estas fichas de trabalho continham 35 exercícios que os estudantes deveriam resolver durante as aulas, bem como fora deste período. Para todas estas tarefas, o sistema recebeu um total de quase 40.000 submissões válidas. A distribuição destas submissões no tempo é visível na Figura 140, na qual o eixo dos XX representa o tempo e o eixo dos YY representa cada um dos estudantes. Cada ponto azul corresponde a uma tentativa de resolução de um exercício.

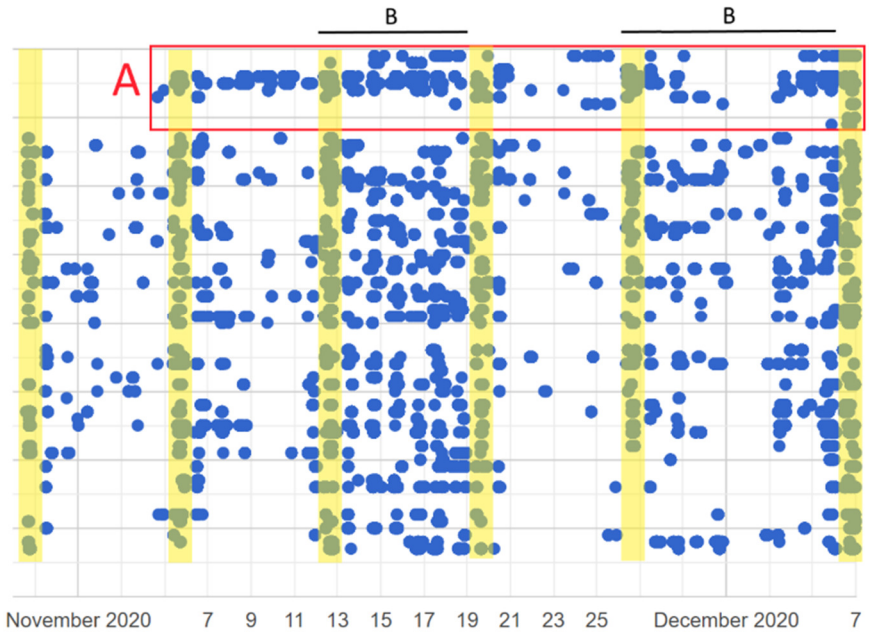

Figura 140 - Distribuição das tentativas de resolução dos estudantes ao longo do tempo - iteração nove.

Nesta figura é possível observar vários aspetos interessantes. Por exemplo, o bloco assinalado com "A" corresponde a um grupo de estudantes que foram colocados no curso numa fase posterior, o que significa que quando começaram a trabalhar na primeira ficha, a maioria dos seus colegas já estavam a trabalhar noutra. Assim, estes estudantes necessitavam fazer trabalho extra fora das aulas para recuperar o tempo perdido. É também possível observar que, embora haja alguma concentração de tentativas durante o período de aulas, há igualmente várias tentativas feitas fora deste período, especialmente para duas fichas de trabalho com alguns exercícios tomados em consideração para a nota final (B).

Outro aspeto interessante diz respeito à distribuição das tentativas recebidas ao longo das horas do dia. Semanalmente, estes estudantes tinham duas aulas de duas horas em que o professor estava com eles remotamente via *Zoom*. Embora cerca de 60% das tentativas tenham sido recebidas durante esse período, uma quantidade considerável não o foi, tal como se apresenta na Figura 141.

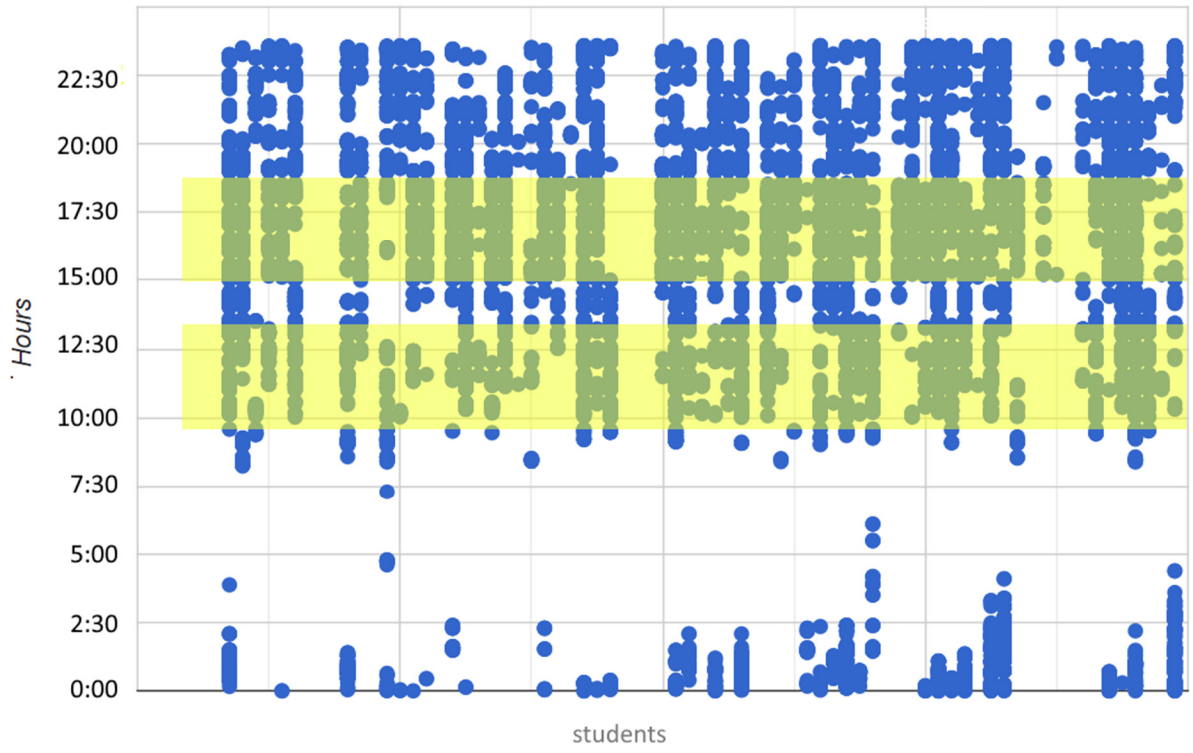

Figura 141 - Distribuição das tentativas dos estudantes ao longo das horas do dia - iteração nove.

Da análise da Figura 141 fica claro que os estudantes trabalharam não só fora das aulas, mas também em períodos que, compreensivelmente, o professor não estava disponível para os assistir de imediato. No entanto, sempre que o professor acedeu ao sistema, pôde

verificar o que os estudantes fizeram nestes períodos e tomar algumas medidas, se necessário.

Embora tivesse existido uma comunicação permanente com o professor durante o período da experiência, quando esta terminou, teve lugar uma reunião com ele através do *Zoom*. Nesta reunião, o professor teve a oportunidade de agradecer a possibilidade de usar o sistema neste contexto, particularmente, nas primeiras semanas, onde os estudantes têm mais dificuldades em compreender as noções básicas e a sintaxe. Segundo o professor, utilizando o sistema, foi capaz de identificar erros e conceitos errados comuns, o que lhe permitiu intervir ou dar *feedback* aos estudantes. Todas estas tarefas foram feitas dentro do sistema, mesmo a classificação manual de alguns exercícios que contavam para avaliação.

Enquanto estava "em aula" com os estudantes, estava ligado via *Zoom* e tinha um separador do navegador permanentemente aberto com o "*live feed*". Sempre que identificava alguma situação que, na sua opinião, deveria ser abordada para todos os estudantes, utilizava o *Zoom* para, por exemplo, apresentar exemplos extra, ou utilizar o código escrito por um determinado estudante para explicar o que deveria (ou não) ser feito numa determinada situação.

Quando os estudantes tinham dúvidas e precisavam da ajuda do professor, podiam solicitá-la verbalmente ou por mensagem privada. Na maioria dos casos, o professor disse que respondeu utilizando a sala principal do *Zoom*, mas sempre que os estudantes solicitavam assistência mais “privada”, o professor recorria a uma sala privada do *Zoom*.

O professor também mencionou que fora do período de aulas era comum receber *e-mails* dos estudantes a perguntar se podia verificar as tentativas para um determinado exercício. Nestes casos, o professor utilizou a ferramenta de *feedback* fornecida pelo sistema para assistir o estudante. Referiu ainda que, fora do período de aulas, verificava por vezes o desempenho dos estudantes, e se encontrasse alguma questão que considerasse importante, tomava as medidas mais adequadas.

Uma vez que já tinha utilizado o sistema anteriormente, não sentiu qualquer dificuldade em utilizá-lo. No entanto, fez algumas sugestões de melhoria, tendo a maioria delas sido imediatamente implementadas. Assim, do ponto de vista do professor, o sistema desempenhou um papel importante para o ajudar a monitorizar e assistir os estudantes, particularmente, devido à distância.

Após o fim da experiência, também considerámos importante conhecer a opinião dos estudantes sobre a utilização do sistema, pelo que, uma vez que a realização de entrevistas individuais com eles consumia muito tempo, construímos um questionário no *Google Forms* (Anexo XIII) que foi enviado a todos eles, como já havia sido feito anteriormente. O questionário esteve disponível durante uma semana, e durante esse período foi recebido um total de 25 respostas.

Começámos por perguntar aos estudantes com que frequência frequentavam as aulas. Dez estudantes responderam "sempre"; dez estudantes responderam, "frequentemente"; um estudante respondeu, "às vezes" e finalmente quatro deles responderam "raramente".

Prossegui-se perguntando com que frequência tinham resolvido os exercícios propostos pelo professor. Dez estudantes responderam "sempre"; treze responderam "frequentemente"; e dois estudantes responderam "raramente".

Na pergunta seguinte, foi solicitado aos estudantes que para indicarem a importância de utilizar uma ferramenta de monitorização no contexto atual. Dezassete estudantes (68%) responderam "muito importante"; três estudantes (12%) responderam "importante"; quatro estudantes responderam "moderadamente importante" (16%); e finalmente, um estudante (4%) respondeu "nada importante".

Para dar mais sentido a estas respostas, pediu-se que fundamentassem a sua resposta. Foram recebidas um total de 17 respostas. Entre as mais positivas, podem encontrar-se respostas tais como: "É bom que o professor possa ter uma noção da situação de cada estudante e prestar assistência personalizada assim que necessário"; "Porque desta forma o professor pode acompanhar o nosso trabalho e ajudar-nos à distância"; "Dada a situação atual em que nos encontramos, penso que este apoio informático é importante"; "Dá uma certa obrigação ao estudante de praticar e ao professor uma forma de controlar o que o estudante faz, dada a ausência de aulas presenciais. "; "Este tipo de sistema tornou-se fundamental para a eficácia do esclarecimento de dúvidas por parte do professor" ou "O professor está mais atento ao progresso do estudante e pode monitorizar o que o estudante está a fazer".

Como esperado, houve também alguns estudantes que não tiveram uma experiência inteiramente positiva: "Muito sensível aos erros! Por vezes o programa estava correto, e parecia que não estava" ou "Por vezes tudo o que era preciso era uma formatação um pouco diferente do esperado, e o exercício estava marcado como errado".

Como as aulas ocorreram remotamente e, como demonstrámos anteriormente, uma parte importante do trabalho do estudante foi feita autonomamente fora das aulas, perguntámos aos estudantes se ter o seu código testado automaticamente contribuiu positivamente para o seu processo de aprendizagem. Dos 25 estudantes que responderam ao questionário, 23 (92%) responderam positivamente e apenas dois estudantes (8%) responderam o oposto.

Com a utilização do sistema não se pretende de modo algum reduzir a autonomia do estudante e aumentar a sua dependência dos professores. Assim, ter o sistema para testar o código dos estudantes automaticamente e fornecer-lhes algum *feedback* quando não está a comportar-se como esperado é certamente uma medida que permite aos estudantes aferir autonomamente se o seu código está a produzir os resultados esperados, ou não. Em qualquer caso, como a maioria dos estudantes reconheceu, é também muito tranquilizador saber que o professor estava vigilante e podia intervir mesmo em situações em que o código tinha passado nos testes, mas a solução não era, por exemplo, a mais eficiente.

Continuámos perguntando sobre a perceção que os estudantes tinham sobre a utilização do sistema pelo professor. A maioria dos estudantes, 88%, respondeu positivamente, enquanto os outros 12% responderam que não sentiam que o professor estava a utilizar o sistema. A este respeito é importante mencionar que existe uma forte correlação entre as respostas às últimas perguntas. De facto, os estudantes que não se aperceberam da utilização do sistema pelo professor são também os que raramente assistiam às aulas ou tentavam resolver os exercícios propostos. Assim, é justo acreditar que as respostas menos positivas a esta última pergunta devem ser vistas à luz desta correlação.

A seguir, foi solicitado aos estudantes que fornecessem exemplos específicos da utilização do sistema pelo professor. Foram recebidas 13 respostas com conteúdo semelhante, pelo que se destacam algumas delas: "Durante as aulas, o professor via o progresso dos estudantes e ajudava-os"; "Sempre que começávamos a aula, ele dizia sempre os principais erros cometidos porque os tinha visto antes"; "Quando reparava que um estudante não estava a resolver os exercícios corretamente, o professor prestava ajuda"; ou "Alguns estudantes estavam a debater-se, mas o professor, através do sistema de monitorização, podia aperceber-se disto e ajudar os estudantes ou explicar melhor o assunto/exercício". Estas respostas acabam por corresponder às expectativas sobre a forma como o professor utilizaria o sistema.

Como visto anteriormente, em usos precedentes do sistema, notou-se que alguns estudantes tinham algumas reservas relativamente a terem todas as suas tentativas enviadas para o sistema e disponibilizadas aos professores, enquanto ainda não estavam terminadas. Assim, perguntámos a estes estudantes se se sentiam desconfortáveis com este facto. A maioria dos estudantes, 72%, responderam que não se sentiam desconfortáveis, enquanto 28% responderam afirmativamente. Posteriormente, solicitou-se aos estudantes que fundamentassem as suas respostas. Entre as respostas recebidas, destacam-se as seguintes: "Em parte é um pouco desconfortável, mas por outro lado podemos ter ajuda imediata"; "Penso que foi demasiada pressão ter o professor a verificar em tempo real o que estávamos a fazer, mas sei que o objetivo era ajudar", "Respondi afirmativamente, porque me senti pressionado a fazer o melhor que podia" e "O objetivo é fazer o melhor que podemos, e o facto de o professor poder aceder ao código só deu mais possibilidades de melhores resultados". Estas respostas são consistentes com as de experiências anteriores, e aparentemente as reservas dos estudantes não estão relacionadas com questões de privacidade, mas sim com alguma pressão por se sentirem observados, ou pela imagem que poderão passar ao professor por este ter acesso a código inacabado que, como tal, poderá ainda conter erros.

Ainda relacionado com o mesmo tópico, perguntámos aos estudantes se tomaram alguma medida para ultrapassar o desconforto que sentiram, e 72% dos estudantes responderam negativamente. A maioria dos restantes estudantes responderam que utilizavam um IDE diferente até terem uma solução final, e só então submetiam a solução no sistema. Mais uma vez, esta resposta é também coerente com experiências anteriores. Obviamente, esta é uma situação que devia ser evitada, reforçando aos estudantes os objetivos do sistema.

Finalmente, foi solicitado aos estudantes que classificassem de 1 a 5 o quanto achavam que a utilização do sistema ajudou no seu processo de aprendizagem. Onze estudantes (44%) atribuíram uma nota de 4 e outros onze estudantes (44%) atribuíram uma classificação de 5, o que significa que 88% dos estudantes sentiram que a utilização do sistema contribuiu para o seu sucesso. Por outro lado, dois estudantes classificaram a utilização do sistema com uma classificação neutra de três e um estudante atribuiu uma classificação de dois. Mais uma vez, as classificações "menos positivas" foram atribuídas pelos estudantes que não compareceram às aulas ou tentaram resolver as tarefas propostas.

4.10.5. Sumário da iteração nove

Uma vez mais, com base no *feedback* dos professores que utilizaram o sistema anteriormente, numa perspetiva de evolução contínua, foram implementadas algumas modificações e desenvolvidas novas funcionalidades. Entre estas funcionalidades convirá destacar a disponibilização de novos *plug-ins* e o suporte de mais linguagens de programação, a possibilidade de utilizar várias linguagens de programação na mesma instalação do sistema, ou a possibilidade de utilizar um número maior de casos de teste. Igualmente importante foi a disponibilização de um editor de código diretamente no sistema que os estudantes podem utilizar para resolver os exercícios.

Segundo o professor, a utilização do sistema foi crucial para conseguir acompanhar o progresso dos seus estudantes e fazer as intervenções que considerou necessárias. Foram, ainda assim, identificados alguns aspetos que podem ser melhorados. Um deles já foi mencionado anteriormente e envolve melhorar a eficácia da comparação dos resultados obtidos e esperados para os casos de teste. Outro exemplo diz respeito à necessidade de depender de aplicações de terceiros, como o *Zoom* ou o correio eletrónico, sempre que os estudantes desejem solicitar a assistência do professor. Para este fim, foi, entretanto, desenvolvida uma funcionalidade que permite aos estudantes solicitarem ajuda diretamente no sistema, utilizando uma abordagem inspirada num *chat*. Foi igualmente desenvolvida uma funcionalidade que permite ao professor realizar com um estudante uma sessão de edição em tempo-real do código fonte que o estudante realizou para um determinado exercício.

Espera-se, obviamente, que a pandemia termine e que regressem as aulas presenciais. Em qualquer caso, tendo em consideração os resultados desta experiência, assim como das realizadas anteriormente, é de acreditar que independentemente de ser utilizado no contexto do ensino à distância (voluntário ou não) ou em aulas mais tradicionais, o sistema será uma ajuda fundamental para professores e estudantes

5. Conclusões

Como já foi referido, o processo de ensino-aprendizagem de programação não é de todo trivial. Muitas das dificuldades existentes acabam por ser intrínsecas ao próprio processo de aprendizagem, independentemente da temática. Contudo, existem algumas características específicas da aprendizagem de programação.

O ponto de partida deste trabalho foi a constatação que muitas vezes os professores não têm a verdadeira noção de quão eficaz está a ser o processo de ensino/aprendizagem. Muitas vezes, essa noção acaba por acontecer apenas nos momentos de avaliação, em que frequentemente já pouco há a fazer, de modo a corrigir eventuais problemas que possam estar a acontecer. Obviamente, em caso de dúvidas, os estudantes podem colocá-las aos professores. No entanto, muitas vezes por inibição de se exporem em frente aos colegas, acabam por não o fazer, deixando arrastar e acumular as dificuldades até um ponto em que acabam por desistir e/ou não conseguir obter aprovação às unidades curriculares.

Com base neste pressuposto, com o presente trabalho pretendeu-se averiguar até que ponto era possível obter informação sobre o desempenho dos estudantes de programação, da forma menos intrusiva e disruptiva possível em relação ao habitual funcionamento das unidades curriculares. A ideia passou por recorrer aos ambientes de desenvolvimento já utilizados no trabalho regular dos estudantes, assim como aos exercícios habitualmente propostos aos estudantes e recolher automaticamente *snapshots* do código que produzem enquanto resolvem os exercícios propostos. Estes *snapshots* são enviados para um servidor central onde são processados. A informação resultante deste processamento fica imediatamente disponível para os professores numa aplicação web desenvolvida para o efeito e poderá ser utilizada para monitorizar em "tempo real" o desempenho dos estudantes, com vista à identificação de problemas/dificuldades, mesmo em relação a estudantes que por algum motivo não frequentem as aulas presenciais.

A metodologia utilizada neste processo de investigação, *Design Based Research* (DBR), levou a que o trabalho tenha sido desenvolvido ao longo de vários ciclos (iterações). Ainda que existisse uma linha condutora que era o fornecimento de informação útil sobre o desempenho dos alunos ao professor, ao longos das dez iterações realizadas foram feitos diversos ajustes, mudanças de foco, avanços e recuos, que foram sempre encarados com naturalidade.

Esta vertente evolutiva e "ágil" da metodologia DBR revelou-se determinante, uma vez que, à medida que o sistema foi sendo desenvolvido e avaliado, houve a necessidade de, por mais de uma vez, recentrar os esforços em aspetos inesperados, mas determinantes para o sucesso do sistema. De modo a avaliar o sistema desenvolvido, foram levadas a cabo dezoito experiências em contexto real, em que o sistema foi integrado com naturalidade no processo educativo em curso e utilizado por catorze professores diferentes e aproximadamente quinhentos estudantes. O *feedback* fornecido, quer pelos professores, quer pelos estudantes, foi determinante para guiar todo o processo de investigação e de desenvolvimento realizado.

Ainda que já tivesse sido apresentada informação ao longo deste documento para responder às questões de investigação mais específicas, apresenta-se em seguida um resumo.

Q1: Qual a granularidade dos dados que deve ser utilizada para suportar o processo?

Por motivos principalmente relacionados com o facto de ser importante que os dados fossem enviados para um servidor, processados e a informação disponibilizada aos professores em tempo real, a granularidade escolhida foi ao nível do *snapshot* do código recolhido sempre que os estudantes compilavam/executavam os seus programas. Deste modo, era possível fornecer informação aos professores em tempo útil, sem impactar o desempenho do computador dos estudantes e sobrecarregar o servidor.

Q2: Qual a melhor forma para recolher os dados junto dos estudantes?

Entendeu-se que a forma ideal para recolher os dados dos alunos deveria basear-se na utilização dos ambientes de desenvolvimento já conhecidos pelos estudantes nas aulas. Além disso, após algumas experiências, tornou-se evidente que uma vez que os dados eram recolhidos aquando da compilação/execução do código, esta recolha deveria ocorrer de forma não intrusiva e utilizando os botões, menus ou atalhos de teclado já disponibilizados para este fim. Esta decisão implicou que fosse necessário desenvolver diversos plug-ins para os ambientes de desenvolvimento utilizados nas diversas unidades curriculares em que o sistema foi utilizado. Com base no *feedback* recolhido de alguns estudantes e professores, acabaria também por ser possível que os alunos fizessem submissões diretamente a partir da ferramenta de monitorização criada para os estudantes sem ser necessário proceder à instalação de nenhum software além do browser, podendo deste modo ser feitas submissões inclusivamente a partir de um *tablet*

ou *smartphone*. Em todo o caso, não se trata de modo algum de alternativas concorrentes, mas sim complementares.

Q3: Qual a informação relevante a apresentar aos professores e qual a melhor forma de o fazer?

Uma parte muito considerável do trabalho desenvolvido focou-se exatamente nesta questão. Assim, ao longo das várias iterações, foram sendo introduzidas diversas visualizações, listagens e páginas contendo informação que os professores tiveram oportunidade de testar, tendo o seu feedback sido fundamental para a evolução registada.

Numa fase muito preliminar, entendeu-se que a simples representação de uma rede bayesiana seria suficiente. Contudo, à medida que foram sendo estabelecidos contactos com professores que iriam utilizar o sistema, cedo se constatou que esta representação era demasiado limitativa e que os professores necessitavam informação mais palpável, rápida de consumir e fácil de entender. De uma forma geral, os professores mostraram interesse em visualizações, tais como a "matriz resumo" ou a *live-feed*, que lhes permitam rapidamente ter um vislumbre do "estado" das turmas e/ou alunos individuais, utilizando elementos visuais simples como a cor de fundo para transmitir informação. Os professores valorizaram igualmente a possibilidade de poder explorar com mais detalhe o trabalho desenvolvido por um estudante concreto, com a possibilidade de ver o código fonte, avaliar as mudanças efetuadas ao longo do tempo ou ver o resultado dos casos de teste.

Como não há duas pessoas iguais, existiram algumas visualizações ou funcionalidades disponibilizadas pelo sistema ***CodeInsights*** que não tiveram um nível de aceitação igual por parte de todos os professores. Em todo o caso, uma característica interessante do sistema, é o facto de serem disponibilizadas diversas funcionalidades e visualizações complementares que os professores podem utilizar de formas distintas ao longo do tempo consoante os objetivos concretos daquele momento.

Q4: De que modo é possível integrar um sistema de notificações automáticas com filtragem da informação mais relevante para notificar os professores sobre aspetos que devem merecer a sua atenção?

Foi sempre ponto assente que o sistema ***CodeInsights*** deveria ser encarado pelos professores como uma ferramenta complementar que estava à sua disposição sempre que considerassem importante a sua utilização. Assim, inicialmente considerou-se

importante que o sistema disponibilizasse mecanismos que permitissem notificar os professores de situações que pudessem merecer a sua atenção, mesmo (ou principalmente) quando não estivessem a utilizar o sistema.

Numa fase inicial, foi desenvolvido um sistema de notificações bastante simples, que acabava por gerar demasiadas notificações, o que se tornava contraproducente. Concebeu-se então um sistema mais avançado, baseado no conceito de atenção seletiva, com vista a ser menos intrusivo e com capacidade de gerar apenas as notificações efetivamente relevantes. Objetivamente, este mecanismo foi implementado e utilizado com sucesso por mais do que uma vez. Contudo, o processo de definição das expectativas, apesar de não ser muito complicado, era trabalhoso e a obtenção dos seus valores numéricos acabava por ser algo subjetiva e não muito fácil de definir. Por este motivo, os professores que testaram o sistema ***CodeInsights*** quase nunca os definiram e as notificações foram sendo geradas com base em valores definidos por omissão, ou por valores definidos por nós. Como consequência, o mecanismo nem sempre funcionou exatamente como devia.

Além do mais, alguns professores manifestaram não pretender receber notificações e serem eles próprios a analisar a informação disponibilizada e a tirar as suas conclusões. Este é claramente o principal aspeto que fica em aberto e que deverá merecer alguma atenção no futuro.

Posto isto, olhando novamente para principal questão de investigação "**É possível utilizar o trabalho realizado pelos estudantes nas unidades curriculares introdutórias de programação aquando da resolução das fichas de trabalho, de modo a permitir ao professor monitorizar de forma eficaz o seu processo de aprendizagem, com vista a que possa intervir atempadamente sempre que necessário para que os estudantes desenvolvam as competências necessárias?**", já foram apresentadas ao longo desta tese evidências de que é de facto possível oferecer informação relevante sobre o desempenho dos estudantes utilizando apenas *snapshots* do seu código sempre que compilam e/ou executam os programas. Estes *snapshots* servem posteriormente para suportar uma grande variedade de ferramentas que permitem aos docentes ter uma visão aumentada do desempenho dos seus alunos, permitindo-lhes identificar as situações que devam merecer a sua intervenção. O próprio ***CodeInsights*** poderá ser utilizado para suportar a assistência dada aos estudantes, por exemplo, ao nível do fornecimento de feedback diretamente no sistema.

Desde o início, era ponto assente que o sistema desenvolvido deveria ser utilizado em contexto real de sala de aulas de modo a apoiar efetivamente a atividade dos professores e não ser apenas uma ferramenta em avaliação para fins meramente académicos. O querer que a utilização do sistema fosse o menos intrusiva possível rapidamente se revelou um dos objetivos mais complicados de atingir. Por um lado, não deveria haver mudanças no modo normal de funcionamento das unidades curriculares, mas por outro lado, pretendia-se fornecer informação útil sobre o desempenho dos estudantes com vista a potenciar as intervenções dos professores. Esta confrontação de opostos, levou a que tivessem efetivamente de ser feitas algumas intrusões não no modo como as unidades curriculares eram conduzidas, nem mesmo nos exercícios que eram propostos aos alunos, mas sim ao nível de como os enunciados eram apresentados aos estudantes e de algum modo na forma como estes tinham de escrever algumas partes do código fonte. Poderá não ter sido a solução ideal, mas acabou por ser aquela que melhor conjugou ambos os requisitos.

Um outro aspeto que se revelou determinante para o sucesso da utilização do sistema, está relacionado com a possibilidade que foi dada aos próprios estudantes de monitorizar o seu desempenho, tendo para o efeito sido desenvolvida uma ferramenta de monitorização destinada aos estudantes. Esta possibilidade não estava inicialmente prevista, mas a sua utilidade e necessidade resultou evidente durante o processo de investigação.

Igualmente relevante foi o facto de se ter percebido que a aceitação da utilização do sistema pelos estudantes aumenta com a visibilidade da sua utilização pelos professores. Assim, a partir de dada altura, o sistema passou a ser encarado não apenas como uma ferramenta de monitorização, mas também como uma ferramenta de apoio às unidades curriculares. Para tal, muito contribuiu o desenvolvimento de uma série de novas funcionalidades, como por exemplo, a divulgação de enunciados através do sistema, suporte à correção manual de exercícios e lançamento das notas ou o repositório das soluções dos exercícios propostos.

Apesar de tudo, continuaram a existir alguns estudantes que manifestaram algum tipo de reservas relativamente à disponibilização de informação tão rica e em tempo real aos professores. Não sendo claramente a maioria, há que respeitar as suas reservas, que são legítimas e que merecem ser estudadas com mais detalhe futuramente, eventualmente por uma equipa multidisciplinar.

Ainda que o sistema tivesse sido idealizado para ser utilizado em contexto presencial, em que o professor pode monitorar o desempenho dos seus alunos enquanto trabalham na sala de aulas, mas também fora desta, as circunstâncias criadas pela pandemia de COVID-19 levaram a que o sistema mostrasse a sua (eventualmente maior) utilidade num contexto em que não existe contacto presencial. Será de esperar que as limitações impostas pela pandemia sejam levantadas a curto-prazo, no entanto, não será de estranhar que gradualmente passem a existir mais ofertas formativas de nível superior em regime de ensino inteiramente à distância e/ou em regime de *blended-learning* pelo que a utilização do sistema nestes contextos será claramente uma mais-valia, quer para os professores, quer para os estudantes.

Desde a realização da última experiência descrita neste documento, o sistema já foi utilizado para suportar o processo de ensino/aprendizagem de mais três unidades curriculares distintas, uma delas pela primeira vez.

Além disso, foi desde então iniciado o desenvolvimento de diversas novas funcionalidades com vista a tornar o sistema mais versátil. A título de exemplo, a possibilidade de estabelecer um chat entre o professor e os estudantes relativos a uma determinada tentativa de resolução de um exercício.

Foi igualmente iniciado o processo de suporte duas novas linguagens de programação/*frameworks*: *Processing* (com conversão para `p5.js`) e `p5.js` nativo. Por se tratar de linguagens de programação em que os resultados são apresentados na forma de visualizações gráficas, será disponibilizada a possibilidade de professores e estudantes utilizarem o sistema ***CodeInsights*** de forma semelhante à anteriormente apresentada para o caso de HTML.

Estão igualmente a ser feitas alterações ao nível da quantidade de casos de teste utilizados para cada exercício. Ao invés de serem possíveis apenas quatro casos de teste, deixará de haver limitação neste número.

Foram ainda iniciadas modificações a nível estrutural de todo o sistema, por forma a permitir que na mesma instalação possam coexistir várias edições da mesma unidade curricular, desde que não em simultâneo. Além de permitir poupar tempo com a instalação aquando do início de uma nova edição, esta modificação permite, por exemplo, que os enunciados de exercícios sejam importados de um ano para o outro, de forma bastante simples. Futuramente, esta modificação poderá igualmente potenciar, por exemplo, a comparação da performance dos estudantes de diferentes anos letivos.

Apesar de toda a informação recolhida e de todas as evoluções efetuadas no sistema desenvolvido, em nosso entender, este é um trabalho que nunca estará terminado, na medida em que de cada vez que se começa a explorar um determinado aspeto abre-se um sem número de novas possibilidades a explorar. Assim, além de continuar a fazer melhoramentos às funcionalidades já disponibilizadas, irão certamente ser concebidas e implementadas novas funcionalidades. Pretende-se, pois, continuar a disponibilizar o sistema ***CodeInsights*** aos professores que o utilizaram e contribuíram ativamente para o seu desenvolvimento, assim como disponibilizá-lo a outros professores que manifestem o interesse em o utilizar.

Um aspeto que por diversas vezes foi abordado pelos professores que utilizaram o sistema ao longo das experiências realizadas e que deverá merecer a nossa atenção futura diz respeito ao alargamento do âmbito em que o sistema é utilizado. Assim, pretende-se que o sistema possa ser utilizado pelos professores e estudantes também para a monitorização da realização de projetos mais complexos e não apenas nos exercícios de "treino".

Ainda que a utilização de soluções de avaliação de exercícios do tipo *black-box* permita obter muita informação útil, é importante dedicar alguma atenção ao código em si, nomeadamente ao nível da identificação automática de erros frequentes ou utilização de soluções pouco eficientes ainda que conduzam a um resultado certo.

Tendo em consideração que em alguns casos o sistema ***CodeInsights*** já é utilizado em várias unidades curriculares distintas do mesmo curso em semestres e anos curriculares diferentes, poderá fazer sentido que haja algum fluxo de informação relativa ao desempenho e às dificuldades dos estudantes de uma unidade curricular para a outra, permitindo deste modo que os professores das unidades curriculares seguintes consigam logo à partida ter uma noção do grupo de alunos, permitindo-lhes fazer os ajustes necessários numa fase preliminar.

Tendo em consideração que a disponibilidade de *datasets* de *snapshots* relativos à resolução de exercícios de programação contendo metadados é escassa, pretende-se num curto espaço de tempo proceder à disponibilização de *datasets* devidamente anonimizados, à comunidade científica que deles possa usufruir para suportar o seu trabalho de investigação. Sempre que tal se propicie, os resultados de novas investigações poderão ser integrados no sistema ***CodeInsights*** e além disso, o próprio sistema poderá servir de plataforma de suporte para estes trabalhos.

Tal como o próprio título da tese indica, com este trabalho pretendeu-se dar um contributo para a melhoria das ferramentas disponíveis para os professores por forma a conseguirem desempenhar melhor a sua atividade. Em certa medida, acaba por constituir uma plataforma para o suporte de investigação sobre a melhoria do processo de ensino/aprendizagem de programação.

Referências

[1] L. A. Meserve, "The Problem with Relying on Technology," *The Ohio Journal of Science,* vol. 98, nº 3, pp. 34-38, June 1998.

[2] N. Carr, The Glass Cage: How Our Computers Are Changing Us, W. W. Norton & Company, 2015.

[3] H. C. Jiau, J. C. Chen e K.-F. Ssu, "Enhancing Self-Motivation in Learning Programming Using Game-Based Simulation and Metrics," *IEEE Transactions on Education,* vol. 52, nº 4, pp. 555 - 562, Nov 2009.

[4] Empirica, "e-Skills in Europe - Portugal Country Report," 2014.

[5] M. U. Bers, Coding as a Playground - Programming and Computational Thinking in the Early Childhood Classroom, New York: Routledge, 2020.

[6] L. Ah-Fur e S.-M. Yang, "The learning effect of visualized programming learning on 6th graders' problem solving and logical reasoning abilities," em *2011 International Conference on Electrical and Control Engineering*, 2011.

[7] M. R. H. Roberts, T. McGill e T. Koppi, "What Students are Telling us about Why They Left Their ICT Course," *Innovation in Teaching and Learning in Information and Computer Sciences,* vol. 10, nº 3, pp. 63-83, 2011.

[8] P. Engrácia e J. O. Baptista, "Percursos no Superior: situação após 4 anos dos alunos inscritos em licenciaturas de 3 anos," Direção-Geral de Estatísticas da Educação e Ciências, Portugal, 2018.

[9] K. Kori, M. Pedaste, L. Äli e E. Tõnisson, "The Role of Programming Experience in ICT Students' Learning Motivation and Academic Achievement," *International Journal of Information and Education Technology,* vol. 6, nº 5, pp. 331-337, May 2016.

[10] N. Moore-Cherry, S. Quin e E. Burroughs, "Why Students Leave: Findings from Qualitative Research into Student Non-Completion in Higher Education in Ireland," University College Dublin / National Forum for the Enhancement of Teaching and Learning, Dublin, Ireland, 2015.

[11] C. Kolle, J. Magenheim, W. Nelles, T. Rhode, N. Schaper, S. Schubert e P. Stechert, "Computer Science Education and Key Competencies," em *9th IFIP World Conference on Computers in Education - WCCE 2009*, Brasil, 2009.

[12] J. E. Moström, "A study of student problems in learning to program," Thesis, Department of Computing Science, Umeå University, Computing Science, Umeå University, Umeå, Sweden, 2011.

[13] J. Rogalski e R. Samurçay, "Acquisition of Programming Knowledge and Skills Psychology of Programming," em *Psychology of Programming*, Academic Press Limited, 1990.

[14] E. W. Dijkstra, "A short introduction to the art of programming," Eindhoven, The Netherlands, Technical University Eindhoven, 1971.

[15] T. Jenkins, "On the difficulty of learning to program," em *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, 2002.

[16] P. Denny, A. Luxton-Reilly, E. D. Tempero e J. Hendrickx, "Understanding the syntax barrier for novices," em *Proceedings of the 16th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2011*, Darmstadt, Germany, 2011.

[17] D. Hagan e S. Markham, "Does it help to have some programming," *SIGCSE Bulletin,* vol. 32, nº 3, pp. 25-28, julho de 200.

[18] D. H. Smith, Q. Hao, F. Jagodzinski, Y. Liu e V. Gupta, "Quantifying the Effects of Prior Knowledge in Entry-Level Programming Courses," em *Proceedings of the ACM Conference on Global Computing Education*, Chengdu, Sichuan, China, 2019.

[19] R. S. Duran, J.-M. Rybicki, A. Hellas e S. Suoranta, "Towards a Common Instrument for Measuring Prior Programming Knowledge," em *ITiCSE '19: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, 2019.

[20] M. J. Armstrong e E. N. Biktimirov , "To Repeat or Not to Repeat a Course," *Journal of Education for Business,* vol. 88, nº 6, pp. 339-344, 2013.

[21] A. Robins, "Learning edge momentum: a new account of outcomes in CS1," *Computer Science Education,* vol. 20, nº 1, pp. 37-71, 2010.

[22] A. Gomes e A. J. Mendes, "Learning to program-difficulties and solutions," em *International Conference on Engineering Education-ICEE*, 2007.

[23] A. Bandura, "Self-Efficacy: Toward a Unifying Theory of Behavioral Change," *Psychological Review,* vol. 84, nº 2, pp. 191-215, 1977.

[24] P. Steinhorst, A. Petersen e J. Vahrenhold, "Revisiting Self-Efficacy in Introductory Programming," em *ICER '20: Proceedings of the 2020 ACM Conference on International Computing Education Research*, Virtual Event, New Zealand, 2020.

[25] J. Bennedsen e M. E. Caspersen, "Failure rates in introductory programming," *ACM SIGCSE Bulletin,* vol. 2, nº 39, pp. 32-36, June 2007.

[26] K. A. Ericsson, R. T. Krampe e C. Tesch-Römer, "The Role of Deliberate Practice in the Acquisition of Expert Performance," *Psychological Review,* vol. 100, nº 3, pp. 363-406, 1993.

[27] P. Kirschner, J. Sweller e R. E. Clark, "Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching," *Educational Psychologist,* vol. 41, nº 2, p. 75-86, 2006.

[28] B. S. Bloom, "The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring," *Educational Researcher,* vol. 13, pp. 4-16,

1994.

[29] A. J. Mendes, L. Paquete, F. A. Cardoso e A. J. Gomes, "Increasing student commitment in introductory programming learning," em *Frontiers in Education*, Seattle, 2012.

[30] A. Raabe e J. Silva, "Um ambiente para atendimento as dificuldades de aprendizagem de algoritmos," em *Anais do XXV Congresso da Sociedade Brasileira de Computação*, 2004.

[31] A. Gomes e M. J. António, "A teacher's view about introductory programming teaching and learning difficulties, strategies and motivations," em *44th ASEE/IEEE Frontiers in Education Conference - FIE'14*, 2014.

[32] D. F. O. Onah, J. Sinclair e R. Boyatt, "Dropout rates of massive open online courses : behavioural patterns," em *Proceedings of EDULEARN14* , 2014.

[33] C. Chen, G. Sonner, P. M. Sadler, D. D. Sasselov, C. Fredericks e D. J. Malan, "Going over the cliff: MOOC dropout behavior at chapter transition," *Distance Education,* vol. 41, nº 1, pp. 6-25, 2020.

[34] E. B. Gregori, J. Zhang, C. Galván-Fernández e F. Fernández-Navarro, "Learner support in MOOCs: Identifying variables linked to completion," *Computers & Education,* vol. 122, pp. 153-168, 2018.

[35] E. L. Glassman, J. Scott, R. Singh, P. J. Guo e R. C. Miller, "OverCode: Visualizing Variation in Student Solutions to Programming Problems at Scale," *ACM Transactions on Computer-Human Interaction,* vol. 22, nº 2, pp. 1-35, 2015.

[36] R. R. Choudhury, H. Yin, J. Moghadam e A. Fox, "AutoStyle: Toward Coding Style Feedback At Scale," em *CSCW '16 Companion: Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, 2016.

[37] B. T. Arah e D. K. Apple, "Everyone Can Learn to Learn," em *NAFEO Conference*, 1993.

[38] S. Donig, "Multiple intelligences and learning styles: Two complementary dimensions," *Teachers College Record,* vol. 106, pp. 96-111, 2004.

[39] J. Carter, "The problems of teaching programming: Do they change with time?," em *In 11th Annual HEA-ICS conference*, Durham, 2010.

[40] A. L. Brown, "Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings," *The Journal of the Learning Sciences,* vol. 2, pp. 141-179, 1992.

[41] T. Anderson e J. Shattuck, "Design-based research: A decade of progress in education research?," *Educational Researcher,* vol. 41, pp. 16-25, 2012.

[42] K. Juuti e J. Lavonen, "Design-Based Research in Science Education: One Step Towards Methodology," *Nordic Studies in Science Education (NorDiNa),* vol. 4, pp. 54-68, 2006.

[43] L. T. Blessing, A. Chakrabarti e K. M. Wallace, "An overview of descriptive studies in relation to a general design research methodology," *Designers: The Key to Successful Product Development,* p. 42-56, 1998.

[44] L. Macedo, The Exploration of Unknown Environments by Affective Agents, 2007.

[45] M. Hu, "Teaching Novices Programming with Core Language and Dynamic Visualisation," em *17th Conference of the National Advisory Committee on Computing Qualifications*, New Zealand, 2004.

[46] M. C. Carlisle, T. A. Wilson, J. W. Humphries e S. M. Hadfield, "RAPTOR: introducing programming to non-majors with flowcharts," *Journal of Computing Sciences in Colleges,* vol. 19, pp. 52-60, 2004.

[47] M. Marcelino, T. Mihaylov e A. Mendes, "H-SICAS, a handheld algorithm animation and simulation tool to support initial programming learning," em *Frontiers in Education Conference*, 2008.

[48] A. Gomes, "A class record and reviewing system designed to promote programming

learning," em *41th ASEE/IEEE Frontiers in Education Conference*, Rapid City, USA, 2011.

[49] R. Jones, T. Boyle e P. Pickard, "Object World: Helping Novice Programmers to Succeed Through a Graphical Object-First Approach," em *4th Annual Conference of the LTSN*, 2004.

[50] M. Hu, M. Winikof e S. Cranefield, "Teaching Novice Programming Using Goals and Plans in a Visual Notation," em *Proceedings of the Fourteenth Australasian Computing Education Conference*, Melbourne, Australia, 2012.

[51] A. Vahldick, P. R. Farah, M. J. Marcelino e A. J. Mendes, "NoBug's Snack Bar: A Computational Thinking Serious Game as an Educational Platform," *IEEE Latin America Transactions,* vol. 17, nº 3, pp. 473-484, 2019.

[52] M. Shahid, A. Wajid, K. U. Haq, I. Saleem e A. H. Shujja, "A Review of Gamification for Learning Programming Fundamental," em *International Conference on Innovative Computing (ICIC)*, 2019.

[53] S. W. Martins, A. J. Mendes e A. D. Figueiredo, "Diversifying activities to improve student performance in programming courses," em *Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*, Sofia, Bulgária, 2010.

[54] A. Matthíasdóttir, "How to teach programming languages to novice students? Lecturing or not?," em *International Conference on Computer Systems and Technologies - CompSysTech'06*, 2006.

[55] L. Major, T. Kyriacou e O. P. Brereton, "Systematic Literature Review: Teaching Novices Programming Using Robots," em *15th International Conference on Evaluation and Assessment in Software Engineering (EASE 2011)*, Durham University, UK, 2011.

[56] A. Vihavainen, M. Paksula e M. Luukkainen, "Extreme Apprenticeship Method in Teaching Programming for Beginners," em *SIGCSE'11*, Dallas, Texas, EUA., 2011.

[57] D. H. Sleeman e J. R. Hartley, "Towards more intelligent teaching systems," *International Journal of Man-Machine Studies,* vol. Vol. 5, 1973.

[58] F. Dağ e K. Erkan, "Realizing of optimal curriculum sequences for a web based general purpose intelligent tutoring system," em *Proceedings Turkish Symposium of Artificial Intelligence and Neural Networks*, 2003.

[59] S. H. Moritz, F. Wei, S. M. Parvez e G. D. Blank, "From objects-first to design-first with multimedia and intelligent tutoring," em *10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, Monte de Caparica, 2005.

[60] J. Beck, M. Stern e E. Haugsjaa, "Applications of AI in Education," *Crossroads,* vol. 3, pp. 11-15, 1996.

[61] K. van Lehn, "The Behavior of Tutoring Systems," em *Int. J. Artif. Intell. Ed.*, 2006.

[62] P. Brusilovsky, "Adaptive and intelligent technologies for Web-based education," *Special Issue on Intelligent Systems and Teleteaching,* vol. 4, pp. 19-25, 1999.

[63] A. Barr, M. Beard e R. C. Atkinson, "The computer as tutorial laboratory: the Stanford BIP project," *International Journal on the Man-Machine Studies,* vol. 8, pp. 567-596, 1976.

[64] K. G. Schulze, R. N. Shelby, D. J. Treacy e M. C. Wintersgill, "Andes: A coached learning environment for classical Newtonian physics," em *Selected Papers from the 11th International Conference on College Teaching and Learning*, 2000.

[65] S. Ritter, J. R. Anderson, K. R. Koedinger e A. Corbett, "Cognitive Tutor: Applied research in mathematics education," *Psychonomic Bulletin & Review,* vol. 14, p. 249-255, 2007.

[66] J. R. Anderson e E. Skwarecki, "The automated tutoring of introductory computer programming," *Communications of the ACM,* vol. 29, pp. 842-849, 1986.

[67] C. J. Butz, S. Hua e R. B. Maguire, "A Web-based Bayesian Intelligent Tutoring System for Computer Programming," *Web Intelligence and Agent Systems,* vol. 4, pp. 77-97, 2006.

[68] M. desJardins, A. Ciavolino, R. Deloatch e E. Feasley, "Playing to Program: Towards an Intelligent Programming Tutor for RUR-PLE," *Association for the Advancement of Artificial,* 2011.

[69] E. R. Sykes e F. Franek, "A Prototype for an Intelligent Tutoring System for Students Learning to Program in Java," em *Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education*, Rhodes, Greece, 2003.

[70] K. van Lehn, "The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems," *Educational Psychologist,* vol. 46, 2011.

[71] K. Yacef, "Intelligent Teaching Assistant Systems," em *ICCE '02: Proceedings of the International Conference on Computers in Education*, 2002.

[72] L. Lesta e K. Yacef, "An Intelligent Teaching Assistant System for Logic," *Intelligent Tutoring Systems,* 2004.

[73] D. Abraham, L. Crawford, L. Lesta, A. Merceron e K. Yacef, "The Logic Tutor: A multimedia presentation," *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning,* 2001.

[74] L. R. A. Giraffa, "Uma Arquitetura de Tutor para Promover Experiências de Aprendizagem Mediadas," em *Simpósio Brasileiro de Informática na Educação - SBIE*, 2006.

[75] W. Chen e B. Wasson, "An Instructional Assistant Agent for Distributed Collaborative Learning," *Lecture Notes in Computer Science,* vol. 2363, pp. 609-618, 2002.

[76] F. Geoffroy, E. Aimeur e D. Gillet, "A virtual assistant for web-based training in

engineering education," *Intelligent Tutoring Systems,* 2002.

[77] R. Queirós e J. P. Leal, "PETCHA - A Programming Exercises Teaching Assistant," em *ACM SIGCSE 17th Anual Conference on Innovation and Technology in Computer Science Education*, 2012.

[78] Z. Zhou, "An intelligent teaching assistant system using deep learning technologies," em *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*, 2020.

[79] A. D. Lino, M. P. Harb, S. R. Brito, A. S. Silva e E. Favero, "Avaliação automática de consultas SQL em ambiente virtual de ensino-aprendizagem," em *2ª Conferência*, Porto, 2007.

[80] J. Hollingsworth, "Automatic graders for programming classes," *Communications of the ACM,* vol. 3, nº 10, p. 528-529, 1960.

[81] C. Douce , D. Livingstone e J. Orwell, "Automatic Test-Based Assessment of Programming: A Review," *Journal on Educational Resources in Computing (JERIC),* vol. 5, 2005.

[82] C. M. Tang, Y. T. Yu e C. K. Poon, "Automated Systems for Testing Student Programs: Practical Issues and Requirements," em *Workshop Proceedings of the 17th International Conference on Computers in Education*, 2009.

[83] I. Stamouli e M. Huggard, "Object oriented programming and program correctness: the students' perspective," em *Proceedings of the second international workshop on Computing education research*, 2006.

[84] C. Piech, M. Sahami, D. Koller, S. Cooper e P. Bilkstein, "Modeling How Students Learn to Program," em *43rd ACM technical symposium on Computer Science Education*, New York, 2012.

[85] M. Blumenstein, S. Green, S. Fogelman, A. Nguyen e V. Muthukkumarasamy, "Performance analysis of GAME: A generic automated marking environment," *Computers & Education,* vol. 50, p. 1203-1216, 2008.

[86] D. W. Juedes, "Experiences in Web-Based Grading," em *33rd ASEE/IEEE Frontiers in Education* , 2003.

[87] M. P. Moreira e E. Favero, "Um Ambiente Para Ensino de Programação comFeedback Automático de Exercícios," em *Workshop Sobre Educação em Computação - Anais do XVIII Congresso da Sociedade Brasileira de Computação*, 2008.

[88] T. McCabe, "A Complexity Measure," *IEEE Transactions on Software Engineering,* Vols. %1 de %2SE-2, nº 4, 1976.

[89] M. H. Halstead, Elements of Software Science (Operating and programming systems series), New York, NYUnited States: Elsevier Science Inc., 1977.

[90] K. . A. Naudé, J. H. Greyling e D. Vogts, "Marking student programs using graph similarity," *Computers & Education ,* vol. 54, nº 2, pp. 545-561, 2010.

[91] C. A. Higgins, "Automated Assessment and Experiences of Teaching Programming," *Journal on Educational Resources in Computing,* vol. 5, 2005.

[92] D. Jackson e M. Usher, "A semi-automated approach to on-line assessment," em *Proceedings of the 5th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 2000.

[93] J. P. Leal e F. Silva, "Mooshak: a Web-based multisite programming contest system," *Software Practice & Experience,* vol. 33, pp. 567 - 581, 2003.

[94] R. Saikkonen, L. Malmi e A. Korhonen, "Fully automatic assessment of programming exercises," em *Proceedings of The 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education*, 2001.

[95] S. H. Edwards, "Using test-driven development in the classroom: Providing students with concrete feedback on performance.," em *In Proceedings of the International Conference on Education and Information Systems: Technologies and Applications (EISTA'03)*, 2003.

[96] S. H. Edwards e K. P. Murali, "CodeWorkout: Short Programming Exercises with Built-in Data Collection," em *June 2017*, ITiCSE '17: Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education.

[97] A. Vihavainen, M. Luukkainen e M. Pärtel, "Test My Code: An Automatic Assessment Service for the Extreme Apprenticeship Method," em *2nd International Workshop on Evidence-based Technology Enhanced Learning. Advances in Intelligent Systems and Computing*, 2013.

[98] A. Mendes, V. Ivanov e M. Marcelino, "A web-based system to support Java programming learning," em *Proceedings of the International Conference on Computer Systems and Technologies*, 2005.

[99] J. C. Rodríguez-del-Pino, E. Rubio-Royo e Z. J. Hernández-Figueroa , "A Virtual Programming Lab for Moodle with automatic assessment and anti-plagiarism features," em *Proceedings of The 2012 Internacional Conference on e-Learning, e-Business, Entreprise Information Systems, & e-Government.*, 2012.

[100] C. Murphy, G. Kaiser, K. Loveland e S. Hasan, "Retina: Helping Students and Instructors Based on Observed Programming Activities," *ACM SIGCSE Bulletin,* vol. 41, nº 1, p. 178-182, 2009.

[101] F. A. Deeb, A. DiLillo e T. Hickey, "Using Spinoza Log Data to Enhance CS1 Pedagogy," em *CSEDU 2018 - International Conference on Computer Supported Education*, 2018.

[102] I. Mekterovic, L. Brkic, B. Milasinovic e M. Baranovic, "Building a Comprehensive Automated Programming Assessment System," *IEEE Access,* pp. 81154 - 81172, 2020.

[103] P. Guo, "Codeopticon: Real-Time, One-To-Many Human Tutoring for Computer Programming," em *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, New York, NY, USA, 2015.

[104] G. Rößling, M. Joy, A. Moreno, A. Radenski, L. Malmi, A. Kerren, T. Naps e R. J. Ross, "Enhancing learning management systems to better support computer

science education," *ACM SIGCSE Bulletin,* vol. 40, nº 4, p. 142-166, 2008.

[105] E. Balzuweit e J. Spacco, "SnapViz: visualizing programming assignment snapshots," em *ITiCSE '13: Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, 2013.

[106] K. Heinonen, K. Hirvikoski, M. Luukkainen e A. Vihavainen, "Using CodeBrowser to seek differences between novice programmers," em *SIGCSE '14: Proceedings of the 45th ACM technical symposium on Computer science education*, 2014.

[107] C. Norris, F. Barry, J. B. Fenwick Jr., K. Reid e J. Rountree, "ClockIt: collecting quantitative data on how beginning software developers really work," *ACM SIGCSE Bulletin,* vol. 40, nº 3, p. 37-41, 2008.

[108] P. Ihantola, T. Ahoniemi, V. Karavirta e O. Seppälä, "Review of Recent Systems for Automatic Assessment of Programming Assignments," em *Koli Calling '10*, Kili, Finland, 2010.

[109] A. H. J. I. P. Vihavainen., A. Vihavainen, J. Helminen e P. Ihantola, "How Novices Tackle Their First Lines of Code in an IDE: Analysis of Programming Session Traces," em *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, Koli, Finland, 2014.

[110] A. Vihavainen, M. Luukkainen e P. Ihantola, "Analysis of source code snapshot granularity levels," em *Proceedings of the 15th Annual Conference on Information Technology Education, SIGITE '14*, New York, NY, USA, 2014.

[111] E. S. Tabanao, M. Rodrigo e M. Jadud, "Predicting at-risk novice Java programmers through the analysis of online protocols," em *seventh international workshop on Computing education research*, 2011.

[112] E. Bumbacher, A. Sandes, A. Deutsch e P. Blikstein, "Student coding styles as predictors of help-seeking behavior," *Artificial intelligence in education,* pp. 856-859, 2013.

[113] P. Blikstein, M. Worsley, C. Piech e M. Sahami, "Programming Pluralism: Using

Learning Analytics to Detect Patterns in the Learning of Computer Programming," *Journal of the Learning Sciences,* 2014.

[114] M. C. Jadud, "A first look at novice compilationbehaviour using bluej," *Computer Science Education,* vol. 15, nº 1, pp. 25-40, 2005.

[115] M. Jadud, "Methods and tools for exploring novice compilation behaviour," em *Proceedings of the second international workshop on Computing education research*, 2006.

[116] C. Watson, F. Li e J. Godwin, "Predicting Performance in an Introductory Programming Course by Logging and Analyzing Student Programming Behavior," em *ICALT '13 Proceedings of the 2013 IEEE 13th International Conference on Advanced Learning Technologies*, 2013.

[117] A. Vihavainen, M. Luukkainen e J. Kurhila, "Using Students' Programming Behavior to Predict Success in an Introductory Mathematics Course," em *Proceedings of the 6th International Conference on Educational Data Mining*, Memphis, TN, USA, 2013.

[118] A. Ahadi, "Early Identification of Novice Programmers' Challenges in Coding Using Machine Learning Techniques," em *Proceedings of the 2016 ACM Conference on International Computing Education Research*, Melbourne, 2016.

[119] A. Ahadi, R. Lister e A. Vihavainen, "On the Number of Attempts Students Made on Some Online Programming Exercises During Semester and their Subsequent Performance on Final Exam Questions," em *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, Arequipa, Peru, 2016.

[120] A. Ahadi, R. Lister, S. Lal, J. Leinonen e A. Hellas, "Performance and Consistency in Learning to Program," em *the Nineteenth Australasian Computing Education Conference*, 2017.

[121] A. Annamaa, A. Hansalu e E. Tonisson, "Automatic analysis of students' solving process in programming exercises," em *IFIP TC3 Working Conference "A New*

Culture of Learning: Computing and next Generations, 2015.

[122] A. Jafari, "Conceptualizing Intelligent Agents for Teaching and Learning," *Educause Quarterly,* vol. 25, nº 3, pp. 28-34, 202.

[123] K. V. M. Chrysafiadi, "Student Modeling Approaches: A Literature Review for the last decade," *Expert Systems with Applications,* 2013.

[124] B. A. Schwendimann, M. J. Rodrıguez-Triana, A. Vozniuk, L. P. Prieto, M. S. Boroujeni, A. Holzer, D. Gillet e P. Dillenbourg, "Perceiving Learning at a Glance: A Systematic Literature Review of Learning Dashboard Research," *IEEE Transactions on Learning Technologies,* vol. 10, nº 1, pp. 30-41, 2017.

[125] K. Verbert, S. Govaerts, E. Duval, J. L. Santos, F. J. M. V. Assche, G. Parra e J. Klerkx, "Learning dashboards: An overview and future research opportunities," *Personal and Ubiquitous Computing ,* vol. 18, nº 6, 2013.

[126] K. Verbert, E. Duval, J. Klerkx, S. Govaerts e J. L. Santos, "Learning Analytics Dashboard Applications," *American Behavioral Scientist,* vol. 57, nº 10, 2013.

[127] J. Driver, "A selective review of selective attention research from the past century," *British Journal of Psychology,* 2001.

[128] D. Kahneman, "Attention and effort," Prentice-Hall, 1973, pp. 120-130.

[129] B. Edgell, "The British psychological society," *British Journal of Psycology,* vol. 27, pp. 113-131, 1947.

[130] M. Posner e J. Driver, "The neurobiology of selective attention," *Current Opinion on Neurobiology,* pp. 165-169, 1992.

[131] S. Haykin e Z. Chen, "The Cocktail Party Problem," *Neural Computation,* 2005.

[132] D. Broadbent, Perception and communication, Oxford: Oxford University Press, 1958.

[133] A. Treisman, "Contextual cues in selective listening," *Quarterly Journal of*

Experimental Psychology., vol. 12, pp. 242-248, 1960.

[134] G. Underwood, "Contextual facilitation from attended and unattended messages," *Journal of Verbal Learning and Verbal Behavior,* vol. 16, pp. 99-106, 1977.

[135] L. Carota, G. Indiveri e V. Dante, "A software-hardware selective attention system," *Neurocomputing,* pp. 58-60, 2004.

[136] M. Rucci e R. and Bajcsy, "Implementing Selective Attention in Machines: The Case of Touch-Driven Saccades," University of Penn, 1994.

[137] J. Moon, J. Yeo, S. Jeong, P. Yoon e L. Minho, "An Automotive Detector Using Biologically Motivated Selective Attention Model for a Blind Spot Monitor," *Neural Information Processing,* 2006.

[138] W. Meyer, R. Reisenzein e A. Schutzwohl, "Towards a process analysis of emotions: The case of surprise," *Motivation and Emotion,* vol. 21, pp. 251-274, 1997.

[139] L. Itti e P. Baldi, "Bayesian surprise attracts human attention," *Advances in Neural Information Processing Systems (NIPS 2005),* vol. 19, pp. 1-8, 2006.

[140] M. Peters, "Towards artificial forms of intelligence, creativity, and surprise," em *Twentieth Annual Conference of the Cognitive Science Society*, Madison, Wisconsin, USA, 1998.

[141] A. Ortony e D. Partridge, "Surprisingness and Expectation Failure: What's the Difference?," em *International Joint Conference on Artificial Intelligence*, Los Altos, CA, 1987.

[142] E. Lorini e C. Castelfranchi, "The cognitive structure of Surprise: looking for basic principles," *Topoi: An International Review of Philosophy,* vol. 26, pp. 133-149, 2007.

[143] L. Macedo e F. A. Cardoso, "Modeling Forms of Surprise in an Artificial Agent," em *23rd Annual Conference of the Cognitive Science Society*, Modeling Forms of Surprise in an Artificial Agent., 2001.

[144] J. A. W. Faidhi e S. K. Robinson, "An empirical approach for detecting program similarity and plagiarism within a university programming environment," *Computers & Education,* vol. 11, nº 1, p. 11-19, 1987.

[145] A. Zrnec e D. Lavbič, "Social network aided plagiarism detection: Social network aided plagiarism detection," *British Journal of Educational Technology ,* vol. 48, pp. 113-128, 2017.

[146] J. e. M. J. B. Bidgood, "As Computer Coding Classes Swell, So Does Cheating," *The New York Times ,* 2017.

[147] C. H. Leung e S. C. L. Cheng, "An Instructional Approach to Practical Solutions for Plagiarism," *Universal Journal of Educational Research,* vol. 5, nº 9, pp. 1646-1652, 2017.

[148] E. A. Özbek, "Plagiarism Detection Services for Formative Feedback and Assessment: Example of Turnitin," *Journal of Educational and Instructional ,* vol. 6, nº 3, pp. 64-72, 2016.

[149] A. Bozkurt, I. Jung, J. Xiao, V. Vladimirschi, R. Schuwer, G. Egorov, S. Lambert e M. ... Paskevicius, "A global outlook to the interruption of education due to COVID-19 Pandemic: Navigating in a time of uncertainty and crisis," *Asian Journal of Distance Education ,* vol. 15, nº 1, pp. 1-126., 2020.

[150] UNESCO , "Education: From disruption to recovery," [Online]. Available: https://en.unesco.org/covid19/educationresponse. [Acedido em 2021 08 19].

Anexos

Apresentam-se em seguida os seguintes anexos:

- Anexo I - Guia de consulta rápida (*cheat sheet*)
- Anexo II - Exemplo de ficha prática tal como era apresentada aos alunos antes da utilização do sistema *CodeInsights*
- Anexo III - Exemplo de ficha prática modificada com inclusão de detalhes para os casos de teste
- Anexo IV - Estrutura base da entrevista realizada aos professores (IPJ - Acertar o Rumo - 2018/2019)
- Anexo V - Manual de instalação e utilização (IPJ - Acertar o Rumo - 2018/2019)
- Anexo VI - Exemplo Consentimento Informado
- Anexo VII - Estrutura do questionário feito aos estudantes (IPJ - Acertar o Rumo - 2018/2019)
- Anexo VIII - Estrutura do questionário feito aos estudantes (PI2/DW2 - ESTGOH - 2018/2019)
- Anexo IX - Manual de instalação e utilização (IPJ - Acertar o Rumo - 2019/2020)
- Anexo X - Estrutura base da entrevista realizada aos estudantes (IPJ - Acertar o Rumo - 2019/2020)
- Anexo XI - Estrutura do questionário feito aos professores (IPJ - Acertar o Rumo - 2019/2020)
- Anexo XII - Estrutura do questionário feito aos estudantes (PI2/DW2 - ESTGOH - 2019/2020)
- Anexo XIII - Estrutura do questionário feito aos estudantes (FP - ESTGOH 2020/2021)

Anexo I - Guia de consulta rápida (*cheat sheet*)

Lembre-se de atualizar o valor de `@aid`

Sempre que começar a resolução de um novo exercício, lembre-se de atualizar o valor de `@aid`. Se não o fizer, este exercício não será corretamente recebido no servidor.

- Utilize valores como **`1.12`**, **`2.1`**, etc.
- **Não** utilize valores como **`ficha1.exercicio1`**

Utilize os nomes indicados para os *inputs*

Em cada enunciado, os professores irão indicar quais os nomes que deverá utilizar para as variáveis de *input*. Respeite esta indicação! Para as restantes variáveis, utilize o nome que considerar mais adequado.

Como definir os *inputs*

Vejamos um exemplo completo de como se deve proceder para escrever o código.

1.2. Desenvolva um programa que calcule o resultado da soma de dois números inteiros `num1` e `num2`.

```
1    ...
2    //begin_inputs
3    int num1 = 2;
4    int num2 = 2;
5    //end_inputs
6
7    int soma = num1 + num2;
```

E se o programa não tiver inputs?

Caso o programa não tenha *inputs*, não se coloca nada entre `//begin_inputs` e `//end_inputs`

```
1    ...
2    //begin_inputs
3    //end_inputs
4    ...
```

Não alterar os valores dos *inputs* fora do local

É igualmente importante que o valor das variáveis de *input* não seja alterado fora destes limites e que toda a lógica do exercício seja realizada fora destes limites.

```
1    ...
2    //begin_inputs
3    int num1 = 2;
4    int num2 = 2;
5    //end_inputs
6
7    num1 = 4;
8    int soma = num1 + num2;
```

Não fazer

Utilize o local dos *inputs* apenas para *inputs*

Na zona destinada aos *inputs* deverá colocar apenas as variáveis de *input*. Não deverá declarar e inicializar outras variáveis, ou colocar outro código neste espaço.

```
1    ...
2    //begin_inputs
3    int num1 = 2;
4    int num2 = 2;
5
6    int soma = num1 + num2;
7    //end_inputs
8    . . .
```

Não fazer

```
1    ...
2    //begin_inputs
3    int num1 = 2;
4    int num2 = 2;
5
6    int soma = 0;
7    //end_inputs
8
9    soma = num1 + num2;
```

Não fazer

E se os inputs forem pedidos ao utilizador?

As regras manter-se-ão inalteradas caso os inputs sejam obtidos do utilizador

```
1    ...
2    //begin_inputs
3     System.out.println("Valor num 1: ");
4     int num1 = sc.nextInt();
5     System.out.println("Valor num 2: ");
6     int num2 = sc.nextInt();
7    //end_inputs
8    ...
9
```

Produza o *output* tal como indicado

Os professores irão igualmente dar exemplos de como é esperado que o *output* seja produzido. Respeite o formato indicado pelo professor!

Verifique o resultado dos testes

Veja os resultados dos testes diretamente no Eclipse ou na ferramenta de monitorização. Se no seu entender o resultado obtido estiver igual ao esperado, mas o sistema classificar como errado, não perca muito tempo a tentar fazer com que passe nos testes.

Em caso de alguma questão ou problema associado ao *plug-in* ou à ferramenta de monitorização: nunogil@dei.uc.pt.

Anexo II - Exemplo de ficha prática tal como era apresentada aos alunos antes da utilização do sistema *CodeInsights*

FICHA 2

1. Escreva um programa que calcule o maior de 3 valores inteiros introduzidos pelo utilizador.

2. Escreva um programa que calcule o máximo, o mínimo, a soma e a média de um conjunto de 3 valores inteiros introduzidos pelo utilizador.

3. Elabore um programa que determine se um número é múltiplo de outro ou não.

4. Construa um programa para calcular as raízes de uma equação do 2º grau (reais ou imaginárias).

5. Faça um programa que peça um valor ao utilizador, correspondente ao raio de um círculo, e, de seguida, lhe apresente um menu com 3 opções: calcular o diâmetro, calcular o perímetro e calcular a área. O programa deve fazer o cálculo e apresentar o resultado correspondente à opção selecionada pelo utilizador.

6. Implemente um programa que leia uma nota de um aluno na escala de 0 a 20 e a converta numa classificação mais qualitativa segundo a tabela:

Nota	**Classificação final**
0-9	Insuficiente
10-13	Suficiente
14-16	Bom
17-20	Excelente

7. Elabore um programa que o auxilie a calcular o seu salário. O programa deve pedir ao utilizador o número de horas que trabalhou nesse mês e quanto ganha por hora. Assuma que desconta para a segurança social 11% e tem uma taxa de retenção de IRS na fonte de 23.5%. O programa deve apresentar os seguintes valores:

   salário bruto
   desconto para a segurança social
   valor retido para efeitos de IRS
   salário líquido.

8. Elabore um programa que o auxilie na elaboração dos orçamentos de pintura da sua empresa. O programa deve pedir o número de metros quadrados a serem pintados. Considere que 1 litro de tinta lhe permite pintar 6 metros quadrados e que a tinta é vendida em latas de 18 litros que custam 80 € ou em latas de 3,6 litros que custam 25 €. O programa deve calcular o número de latas a serem orçamentadas e o respetivo preço para as seguintes situações à sua escolha:

   - orçamentar apenas latas de 18 litros
   - orçamentar apenas latas de 3,6 litros
   - orçamentar latas de 18 litros e latas de 3,6 litros por forma a que o preço seja mínimo.

9. Crie um programa que verifique a partir do valor dos lados, fornecidos pelo utilizador, se podem representar ou não um triângulo.

10. Crie um programa que verifique a partir das coordenadas de 4 pontos, fornecidas pelo utilizador, se representam ou não uma figura geométrica com a forma de um quadrado.

11. Elabore um programa que dada uma hora de partida e a duração de uma viagem, ambas em horas e minutos, calcule a hora de chegada, assumindo que ambas se verificam no mesmo dia.

12. Altere o programa anterior para que não tenha essa restrição, ou seja partida e chegada podem ocorrer em dias diferentes.

13. Escreva um programa que determine se um ponto de coordenadas (x1, y1) está dentro (ou não) de uma circunferência de raio r e centro em (x, y).

14. Escreva um programa que determine se duas circunferência de raios r1 e r2 e centros em (x1, y1) e (x2, y2) respetivamente são tangentes ou não.

15. Escreva um programa que determine se duas circunferências de raios r1 e r2 e centros em (x1, y1) e (x2, y2) respetivamente são concêntricas ou não e, em caso afirmativo, indique qual é a maior e a menor das duas (ou se são iguais).

16. Um método de criação de números para cartões consiste em acrescentar um dígito ao número original de modo a que a soma dos dígitos do novo número seja um número par. Para isso acrescenta um '0' ou um '1' à direita do número original, conforme necessário. Admitindo que os números originais têm todos 4 dígitos, especifique os passos que permitem aplicar este método.

17. Escreva um programa que receba um número em binário com 5 dígitos determine o seu valor equivalente decimal.

18. Escreva um programa que receba um número decimal com 5 dígitos e determine o seu valor equivalente em binário.

19. Escreva um programa que leia um número de quatro dígitos e que determine o número de ocorrências de cada dígito no número.

20. Escreva um programa para calcular o vencimento de um funcionário no final de um mês. Ao vencimento base do funcionário juntam-se as horas extra que fizer. Estas horas são pagas conforme o número de horas extra que fizer por mês segundo a tabela:

Nº de Horas	**Preço da hora**
1-10	1/50 do vencimento
11-20	1/45 do vencimento
21-30	1/40 do vencimento
31-40	1/35 do vencimento
41 ou mais	1/30 do vencimento

21. Construa um programa que dado o número de um mês escreva o nome correspondente.

22. Reescreva o problema para calcular o vencimento de um funcionário no final de um mês, supondo agora que as horas extras são pagas conforme a tabela e usando uma instrução switch:

Nº de horas	**Preço da hora**
1-3	a 1/50 do vencimento
4-6	a 1/45 do vencimento
7-9	a 1/40 do vencimento
10 ou mais	a 1/35 do vencimento

23. Uma companhia que vende flores a floristas e ao público, oferece não só isenção de impostos aos primeiros como um desconto que depende da importância a pagar segundo a tabela:

Importância	**Desconto**
Inferior a 50€	2%
De 50 a 200€	5%
Superior a 200€	10%

Os outros compradores estão sujeitos a uma taxa de 6% de imposto.
Após a realização de uma compra, determine, a partir da importância calculada, quanto o cliente vai efetivamente pagar.

24. Faça um programa que determine se um determinado ano introduzido pelo utilizador é bissexto. São bissextos os anos múltiplos de 4 que não são múltiplos de 100, e ainda os anos múltiplos de 400. Por exemplo, os anos 1996 e 2000 são bissextos, enquanto os anos 1998 e 1900 são comuns.

25. Elabore um programa que leia dois ângulos em graus, minutos e segundos e calcule a sua soma apresentando o resultado de forma reduzida (um ângulo de 0º a 360º).

26. Construa um programa que leia duas datas e determine qual é a maior.

27. Elabore um programa que leia uma data (dia, mês, ano) entre 1 de Janeiro de 1990 e 31 de Dezembro de 2016 e determine a data do dia seguinte.

Anexo III - Exemplo de ficha prática modificada com inclusão de detalhes para os casos de teste

FICHA 2

É importante que identifique corretamente o exercício que está a resolver (exemplo: `@aid 2.1`) e que utilize as palavras assinaladas a verde como nomes para as variáveis. É igualmente importante que entre `//begin_inputs` e `//end_inputs` sejam apenas declaradas e inicializadas as variáveis que correspondam a *inputs* do programa. Todas as restantes variáveis, constantes e a lógica do programa deverá ser realizada fora deste bloco.

1. Escreva um programa que calcule o maior de 3 valores inteiros introduzidos pelo utilizador (`num1`, `num2` e `num3`). Exemplo de *output* para `num1` = 45, `num2` = 10 e `num3` = 26:

```
Maior = 45
```

2. Escreva um programa que calcule o máximo, o mínimo, a soma e a média de um conjunto de 3 valores inteiros introduzidos pelo utilizador (`num1`, `num2` e `num3`). Exemplo de *output* para `num1` = 45, `num2` = 10 e `num3` = 26:

```
Max (45) Min (10) Soma (81) Média (27.0)
```

3. Elabore um programa que determine se um número (`num1`) é múltiplo de outro (`num2`) ou não. Exemplo de *output* para `num1` = 45 e `num2` = 10:

```
45 não é múltiplo de 10
```

   Exemplo de *output* para `num1` = 40 e `num2` = 10:

```
40 é múltiplo de 10
```

4. Construa um programa para calcular as raízes de uma equação do 2º grau (reais ou imaginárias). Recorra às variáveis `coeficienteA`, `coeficienteB` e `coeficienteC` para obter os valores dos coeficientes. Exemplo de output para `coeficienteA` = 1, `coeficienteB` = -5 e `coeficienteC` = 6:

```
R = 3.0 e R = 2.0
```

5. Faça um programa que peça um valor ao utilizador, correspondente ao raio (`raio`) de um círculo, e, de seguida, lhe apresente um menu com 3 opções: 1 - calcular o diâmetro, 2 - calcular o perímetro e 3 - calcular a área. O programa deve fazer o cálculo e apresentar o resultado correspondente à opção selecionada pelo utilizador (utilize a variável `opcao`). Exemplo de *output* para `raio` = 10 e `opcao` = 1:

```
Diâmetro = 20
```

6. Escreva um programa que leia um carácter do teclado (`caracter`) e escreva no ecrã se se trata de uma letra maiúscula, minúscula ou se não é uma letra. Exemplo de *output* para `caracter` = a

```
a - letra minuscula
```

Exemplo de *output* para `caracter` = 9

```
9 - não é uma letra
```

7. Implemente um programa que leia uma nota (`nota`) de um aluno na escala de 0 a 20 e a converta numa classificação mais qualitativa segundo a tabela:

Nota	Classificação final
0-9	Insuficiente
10-13	Suficiente
14-16	Bom
17-20	Excelente

Exemplo de *output* para `nota` = 10.5

```
10.5 - Suficiente
```

8. Elabore um programa que o auxilie a calcular o seu salário. O programa deve pedir ao utilizador o número de horas que trabalhou nesse mês (`numeroHoras`) e quanto ganha por hora (`custoHora`). Assuma que desconta para a segurança social 11% e tem uma taxa de retenção de IRS na fonte de 23.5%. O programa deve apresentar os seguintes valores:
   salário bruto
   desconto para a segurança social
   valor retido para efeitos de IRS
   salário líquido.

Exemplo de *output* para `numeroHoras` = 100 e `custoHora` = 10.5

```
Salário bruto  1050.0
Desconto SS     115.5
Retenção IRS    246.75
Salário líquido 687.75
```

9. Elabore um programa que o auxilie na elaboração dos orçamentos de pintura da sua empresa. O programa deve pedir o número de metros quadrados a serem pintados (`numeroMetros`). Considere que 1 litro de tinta lhe permite pintar 6 metros quadrados e que a tinta é vendida em latas de 18 litros que custam 80 € ou em latas de 3,6 litros que custam 25 €. O programa deve calcular o número de latas a serem orçamentadas e o respetivo preço para as seguintes situações à sua escolha (`opcao`):
   1 - orçamentar apenas latas de 18 litros
   2 - orçamentar apenas latas de 3,6 litros

3 - orçamentar latas de 18 litros e latas de 3,6 litros por forma a que o preço seja mínimo.

Exemplo de *output* para `numeroMetros` = 100 e `opcao` = 1

```
Necessita 1 lata de 18 litros - 80€
```

Exemplo de *output* para `numeroMetros` = 100 e `opcao` = 2

```
Necessita 5 latas de 3.6 litros - 125€
```

10. Crie um programa que verifique a partir do valor dos lados (`lado1`, `lado2`, `lado3`), fornecidos pelo utilizador, se podem representar ou não um triângulo.

11. Crie um programa que verifique a partir das coordenadas de 4 pontos (`x1`, `y1`, `x2`, `y2`, `x3`, `y3`, `x4`, `y4`), fornecidas pelo utilizador, se representam ou não uma figura geométrica com a forma de um quadrado. Exemplo de *output* para `x1`=1, `y1`=1, `x2`=1, `y2`=3, `x3`=3 `y3`=1, `x4`=3, `y4`=3.

```
É um quadrado!
```

12. Elabore um programa que dada uma hora de partida (`horasPartida`, `minutosPartida`) e a duração de uma viagem (`horasDuracao`, `minutosDuracao`), ambas em horas e minutos, calcule a hora de chegada, assumindo que ambas se verificam no mesmo dia. Exemplo de *output* para `horasPartida` = 17, `minutosPartida` = 10, `horasDuracao` = 1, `minutosDuracao` = 30:

```
Hora de chegada = 18h 40m
```

13. Altere o programa anterior para que não tenha essa restrição, ou seja partida e chegada podem ocorrer em dias diferentes. Exemplo de *output* para `horasPartida` = 23, `minutosPartida` = 10, `horasDuracao` = 2, `minutosDuracao` = 30:

```
Hora de chegada = 1h 40m
```

14. Escreva um programa que determine se um ponto de coordenadas (`pontoX`, `pontoY`) está dentro (ou não) de uma circunferência de raio r (`raio`) e centro em (`centroX`, `centroY`). Exemplo de *output* para `pontoX` = 10, `pontoY` = 10, `raio` = 10, `centroX` = 15, `centroY` = 15:

```
Está dentro!
```

15. Escreva um programa que determine se duas circunferência de raios `raio1` e `raio2` e centros em (`centroX1`, `centroY1`) e (`centroX2`, `centroY2`) respetivamente são tangentes ou não. Exemplo de *output* para `centroX1` = 15, `centroY1` = 15, `raio1` = 10, `centroX2` = 35, `centroY2` = 15, `raio2` = 10.

```
As circunferências são tangentes!
```

16. Escreva um programa que determine se duas circunferência de raios `raio1` e `raio2` e centros em (`centroX1`, `centroY1`) e (`centroX2`, `centroY2`) respetivamente são concêntricas ou não e, em caso afirmativo, indique qual é a maior e a menor das duas (ou se são iguais). Exemplo de *output* para `centroX1 = 15`, `centroY1 = 15`, `raio1` = 10, `centroX2 = 15`, `centroY2 = 15`, `raio2` = 20.

```
As circunferências são concêntricas e 2ª é a maior.
```

17. Um método de criação de números para cartões consiste em acrescentar um dígito ao número original (`numeroCartao`) de modo a que a soma dos dígitos do novo número seja um número par. Para isso acrescenta um '0' ou um '1' à direita do número original, conforme necessário. Admitindo que os números originais têm todos 4 dígitos, especifique os passos que permitem aplicar este método. Exemplo de *output* para `numeroCartao = 1231`:

```
Numero final = 12311
```

Exemplo de *output* para `numeroCartao = 1232`:

```
Numero final = 12320
```

18. Escreva um programa que receba um número em binário com 5 dígitos (`numeroBinario`) determine o seu valor equivalente decimal. Exemplo de *output* para `numeroBinario = 10011`:

```
10011 = 19
```

19. Escreva um programa que receba um número decimal com 5 dígitos (`numeroDigital`) e determine o seu valor equivalente em binário. Exemplo de *output* para `numeroDigital = 12345`:

```
12345 = 11000000111001
```

20. Escreva um programa que leia um número de quatro dígitos (`numero`) e que determine o número de ocorrências de cada dígito no número. Exemplo de *output* para `numero = 1222`:

```
1 - 1 vez
2 – 3 vezes
```

21. Construa um programa que dado o número de um mês (`numeroMes`) escreva o nome correspondente. Exemplo de *output* para `numeroMes = 9`:

```
9 - setembro
```

22. Escreva um programa para calcular o vencimento de um funcionário no final de um mês. Ao vencimento base do funcionário (`vencimentoBase`) juntam-se as horas extra que fizer (`horasExtra`). Estas horas são pagas conforme o número de horas extra que fizer por mês segundo a tabela:

Nº de Horas	Preço da hora
1-10	1/50 do vencimento
11-20	1/45 do vencimento
21-30	1/40 do vencimento
31-40	1/35 do vencimento
41 ou mais	1/30 do vencimento

Exemplo de *output* para `vencimentoBase = 1000` e `horasExtra` = 17:

```
Vencimento final = 1377.77 €
```

23. Reescreva o problema anterior supondo agora que as horas extras são pagas conforme a tabela e usando uma instrução `switch`:

Nº de horas	Preço da hora
1-3	a 1/50 do vencimento
4-6	a 1/45 do vencimento
7-9	a 1/40 do vencimento
10 ou mais	a 1/35 do vencimento

24. Uma companhia que vende flores a floristas (F) e ao público (P) (`tipo`), oferece não só isenção de impostos aos primeiros como um desconto que depende da importância a pagar segundo a tabela:

Importância	Desconto
Inferior a 50€	2%
De 50 a 200€	5%
Superior a 200€	10%

Os outros compradores estão sujeitos a uma taxa de 6% de imposto.

Após a realização de uma compra, determine, a partir da importância calculada (`subTotal`), quanto o cliente vai efetivamente pagar. Exemplo de *output* para `tipo = P` e `subTotal` = 35:

```
Valor a pagar = 36.358 €
```

Exemplo de *output* para `tipo = F` e `subTotal` = 235:

```
Valor a pagar = 211.5 €
```

25. Faça um programa que determine se um determinado ano introduzido pelo utilizador (`ano`) é bissexto. São bissextos os anos múltiplos de 4 que não são múltiplos de 100, e ainda os anos múltiplos de 400. Por exemplo, os anos 1996 e 2000 são bissextos, enquanto os anos 1998 e 1900 são comuns. Exemplo de *output* para `ano = 2020`

```
2020 é bissexto
```

Exemplo de *output* para `ano = 2017`

```
2017 é comum
```

26. Elabore um programa que leia dois ângulos em graus (`graus1`, `graus2`), minutos (`minutos1`, `minutos2`) e segundos (`segundos1`, `segundos2`) e calcule a sua soma apresentando o resultado de forma reduzida (um ângulo de 0º a 360º). Exemplo de *output* para os ângulos `45º 10’ 10’’` e `23º 34’ 21’’`:

```
Soma = 68.74º
```

27. Construa um programa que leia duas datas (`dia1`, `mes1`, `ano1` e `dia2`, `mes2`, `ano2`) e determine qual é a maior. Exemplo de *output* para `dia1 = 31`, `mes1 = 1`, `ano1 = 2012` e `dia2 = 11`, `mes2 = 2`, `ano2 = 2022`.

```
11 / 2 / 2022 é maior!
```

28. Elabore um programa que leia uma data (`dia`, `mes`, `ano`) entre 1 de Janeiro de 1990 e 31 de Dezembro de 2013 e determine a data do dia seguinte. Exemplo de *output* para `dia = 31`, `mes = 1` e `ano = 2012`:

```
O dia a seguir a 31/1/2012 é 1/2/2012
```

Anexo IV - Estrutura base da entrevista realizada aos professores (IPJ - Acertar o Rumo - 2018/2019)

CodeInsights

Entrevista de avaliação/validação

Experiência: Acertar o Rumo 2017/2018 (Java) (CodeInsights v4)

Professor:

1. **Com que regularidade utilizou o sistema *CodeInsights* ao longo da experiência?**

2. **Quão útil foi a informação providenciada pelo sistema?**

3. **Qual ou quais as funcionalidades do sistema que utilizou com mais frequência?**

4. **Sentiu alguma dificuldade a interpretar a informação fornecida?**

5. **Adotou medidas diretamente influenciadas pela informação fornecida pelo sistema?**

6. **Em futuras unidades curriculares que venha a lecionar, consideraria utilizar o Sistema CodeInsights?**

7. **Quais foram as reações dos estudantes relativamente à utilização do sistema?**

8. **Tem alguma sugestão para melhorias ao sistema?**

Anexo V - Manual de instalação e utilização (IPJ - Acertar o Rumo - 2018/2019)

CodeInsights v0.4

for Eclipse Oxygen

Apresentação

CodeInsights é uma ferramenta de monitorização do progresso de alunos de programação com base em *snapshots* de código fonte.

Para o seu correto funcionamento é absolutamente necessário que os alunos se identifiquem e que identifiquem o exercício que estão a resolver. No caso de haver essa indicação no enunciado, é importante que sejam utilizados os nomes das variáveis sugeridos e que o também o *output* respeite o formato sugerido.

Instalação

Para instalar o *plug-in* do CodeInsights para Eclipse, basta copiar o ficheiro fornecido (`org.codeinsights4_0.4.0.jar`) para a pasta `dropins` que se encontra localizada na mesma pasta em que o Eclipse foi instalado.

Exemplo: `C:\Users\gilito\eclipse\java-oxygen\eclipse\dropins`

Esta operação deverá ser efetuada antes de executar o Eclipse. Caso o Eclipse já esteja a correr, deverá ser reiniciado.

Após ter procedido como apresentado anteriormente, poderá confirmar se o *plug-in* ficou bem instalado acedendo à opção **Help > Instal New Software…** tal como apresentado na Figura 1.

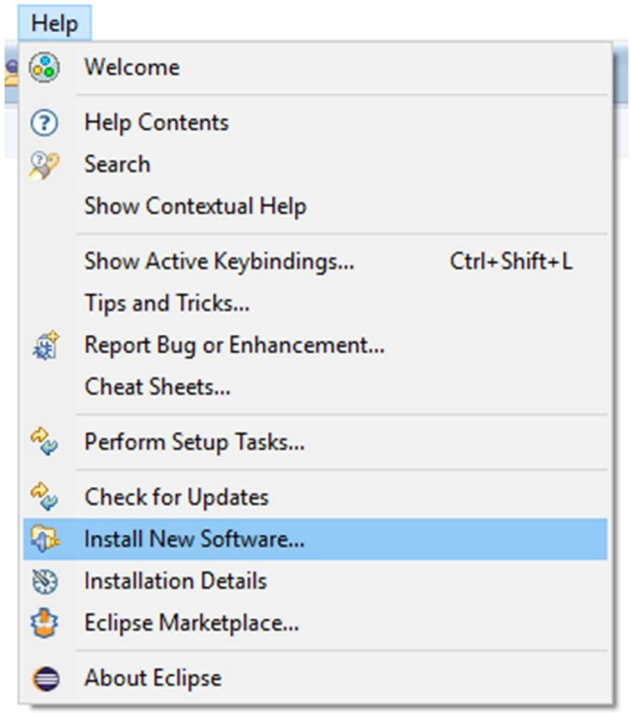

Figura 1 – Menu “Help”

Deverá surgir um diálogo semelhante ao que em seguida se apresenta na Figura 2.

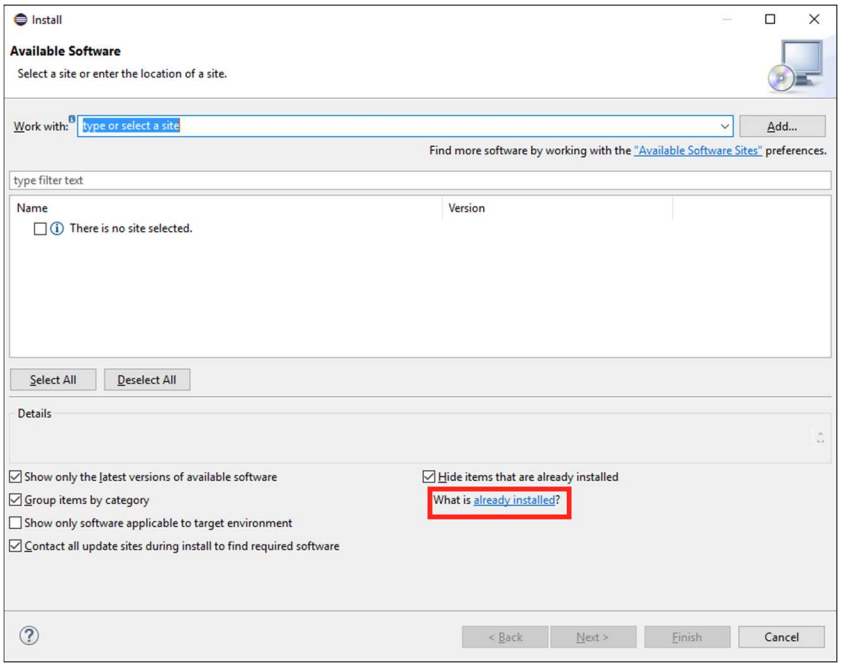

Figura 2 – Diálogo de pesquisa de *plug-ins* para instalar

Aqui chegado deverá clicar na opção "**already installed**" que se encontra destacada na Figura 2. No diálogo que será apresentado em seguida escolha a aba "**Plug-ins**" e em seguida escreva "codeinsights" no campo de pesquisa. Se tudo tiver corrido como esperado, deverá surgir um item semelhante ao que é apresentado na Figura 3.

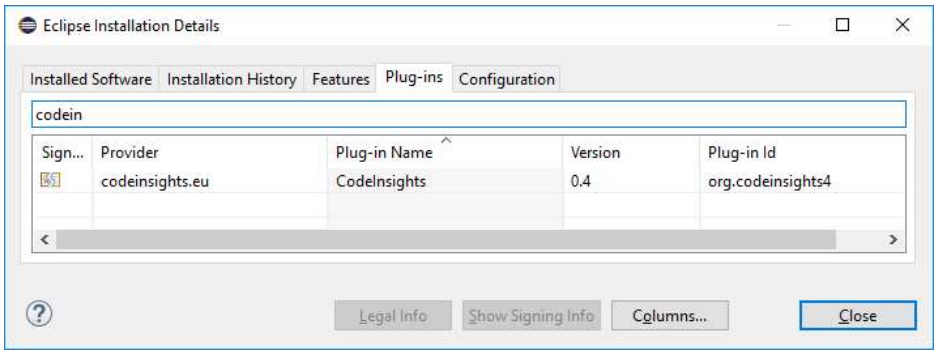

Figura 3 – Diálogo de confirmação da correta instalação do *plug-in*.

Caso, este item não apareça, verifique se escreveu bem o texto a procurar. Se o item não aparecer de qualquer forma, é porque possivelmente ocorreu algum problema com a instalação do *plug-in*. Peça ajuda ao professor!

Alteração do *template* de criação de nova classe

Para o correto funcionamento do *plug-in* é necessário que os alunos forneçam alguma informação sobre o seu número de aluno e sobre o exercício que estão a tentar resolver.

De modo a agilizar este processo, deverá proceder à alteração do *template* de criação de uma nova classe Java. Para tal deverá aceder a `Window > Preferences > Java > Code Style > Code Templates > Code` escolher a opção "**New Java file**" e clicar em `Edit`.

Aqui deverá fazer *copy/paste* do seguinte pedaço de código:

```
${filecomment}
${package_declaration}

/**
 * Breve descrição do código
 *
 * @sid 123456
 * @aid 1.1
 */

${typecomment}
${type_declaration}
```

Deverá ter o cuidado de substituir o valor que se encontra à frente de `@sid` (*student id*) pelo seu número de aluno. Após terminar a edição, basta clicar em "OK".

Em seguida, deverá escolher a opção "**Method body**" e clicar em `Edit`. Aqui chegado, deverá fazer copy/paste do seguinte pedaço de código:

```
//begin_inputs

//end_inputs

${body_statement}
```

Após terminar a edição, basta clicar em "OK".

De agora em diante, sempre que iniciar a resolução de um exercício, deverá começar por criar um novo ficheiro (`File > New > Class`). Deverá indicar o nome que pretende utilizar para a classe (`Name:`) e indicar que pretende que o método `main` seja criado de forma automática, tal como destacado na Figura 4.

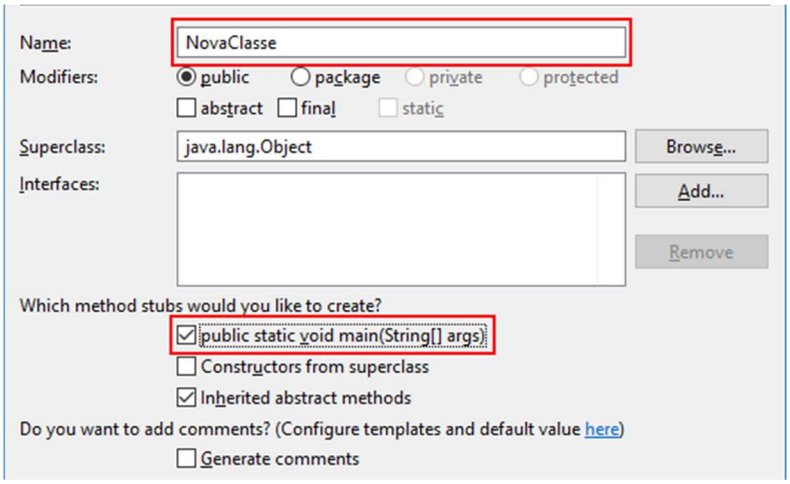

Figura 4 – Diálogo de criação de nova classe

Para terminar o processo, deverá clicar em "Finish". Nesse momento deverá ser criado no Eclipse um novo ficheiro "NovaClasse.java" que deverá estar preparado para ser identificado de forma muito mais rápida, sendo apenas ter o cuidado de alterar o valor de `@aid` sempre que

começar a resolver um novo exercício. (`@aid 1.1` / `@aid 1.2` / `@aid 1.3` / `...`). Em seguida apresenta-se um exemplo de um novo ficheiro acabado de criar.

```
package teste;

/**
 * Breve descrição do código
 *
 * @sid 123456
 * @aid 1.1
 */

public class NovaClasse {

	public static void main(String[] args) {
		//begin_inputs

		//end_inputs
	}
}
```

Como referido anteriormente, é muito importante que na resolução dos exercícios sejam utilizados os nomes das variáveis de *input* indicadas, mas para além disso é igualmente vital que as variáveis sejam declaradas e inicializadas dentro dos limites delimitados por `//begin_inputs` e `//end_inputs`. Vejamos um exemplo:

1.2. Desenvolva um programa que calcule o resultado da soma de dois números inteiros `num1` e `num2`.

Exemplo de *output*:

```
2 + 2 = 4
```

Este exercício poderia ser resolvido pelo aluno com número 123456 do seguinte modo:

```
package teste;

/**
 * Soma dois numeros
 *
 * @sid 123456
 * @aid 1.2
 */

public class SomaDoisNumeros {

	public static void main(String[] args) {

		//begin_inputs
		int num1 = 2;
		int num2 = 2;
		//end_inputs

		int soma = 0;
		soma = num1 + num2;

		System.out.println(num1 + "+" + num2 + "=" + soma);

	}

}
```

É importante que entre `//begin_inputs` e `//end_inputs` se encontre apenas o código que diga respeito à declaração e inicialização das variáveis. Mesmo que os valores das variáveis de input sejam introduzidos pelo utilizador, é importante que a regra se mantenha, tal como apresentado em seguida.

```
package teste;

/**
 * Soma dois numeros
 *
 * @sid 123456
 * @aid 1.2
 */

public class SomaDoisNumeros {

        public static void main(String[] args) {

                Scanner sc = new Scanner(System.in);

                //begin_inputs
                System.out.println("Valor do primeiro numero: ");
                int num1 = sc.nextInt();
                System.out.println("Valor do segundo numero: ");
                int num2 = sc.nextInt();
                //end_inputs

                sc.close();

                int soma = 0;
                soma = num1 + num2;

                System.out.println(num1 + "+" + num2 + "=" + soma);

        }
}
```

É igualmente importante que o valor das variáveis de *input* não seja alterado fora destes limites e que toda a lógica do exercício seja realizada fora destes limites.

Anexo VI - Exemplo Consentimento Informado

Consentimento para recolha e tratamento de dados

CodeInsights

Introdução à Programação em Java

Acertar o Rumo – 2017/2018

Por forma a melhor assistir os alunos no processo de ensino/aprendizagem, nesta unidade curricular irá ser utilizada a plataforma ***CodeInsights***.

Esta plataforma recolhe informação relativa ao código fonte escrito pelos alunos mediante a instalação de um *plug-in* no ambiente de desenvolvimento utilizado nas aulas.

Importa destacar que a informação é recolhida de modo totalmente automático e de forma confidencial.

Eu, _____, aluno nº _____, declaro ter lido e compreendido este documento, bem como as informações verbais que me foram fornecidas pelas pessoas responsáveis. Desta forma, aceito a utilização dos dados que de forma voluntária forneço, confiando em que apenas serão utilizados para os fins indicados e nas garantias de confidencialidade que me são dadas pelos responsáveis.

Data: ___/ ____ / _____

Assinatura: _____

Anexo VII - Estrutura do questionário feito aos estudantes (IPJ - Acertar o Rumo - 2018/2019)

CodeInsights - Inquérito

Acertar o Rumo 2018/2019

Ao longo das últimas semanas, tem vindo a utilizar o sistema *CodeInsights* no processo de resolução dos exercícios propostos nas aulas. Agradecíamos que nos fornecesse alguns detalhes sobre esta experiência. As perguntas seguintes têm como único objetivo tentar perceber o que pode ter corrido menos bem, com vista a melhorar de futuro e, em simultâneo, identificar os aspetos que podem e devem ser mantidos.

Para todas as perguntas será disponibilizado um espaço no qual poderá sustentar a sua resposta.

Desde já, muito obrigado pela sua participação.

1. Considera importante a **utilização deste tipo de sistemas de monitorização** no processo de ensino aprendizagem de programação?

☐ Sim

☐ Não

2. Ao longo deste período sentiu, no seu dia a dia, que o(s) professor(es) **utilizavam realmente o sistema** para suportar a preparação das aulas e/ou a assistência dada aos alunos?

☐ Sim

☐ Não

3. Em caso de resposta afirmativa à pergunta nº2, pode apresentar exemplos concretos?

4. Com base nas suas respostas às perguntas nº2 e nº3, considerou **relevantes as intervenções** feitas pelo(s) professor(es), com base na **informação fornecida pelo sistema**?

☐ Sim

☐ Não

☐ Nem sempre

5. Alguma vez se sentiu **desconfortável** com o facto de saber que o sistema faz a **recolha do seu código** de cada vez que é feita uma compilação e que o professor tem acesso imediato a esse código?

☐ Sim

☐ Não

6. De modo a contornar este desconforto, adotou algum tipo de medida?

☐ Não.

☐ Sim, fazia as submissões de forma "anónima" até ter uma solução estável.

☐ Sim, utilizava outro IDE até ter uma solução estável.

☐ Sim, removi o *plug-in*.

☐ Sim, outra(s) medida(s) (por favor, concretize)

7. Sentir-se-ia mais confortável se pudesse decidir **quando** o código deve ou não ser enviado para o servidor?

☐ Sim

☐ Não

8. Com que regularidade acedeu à ferramenta de acompanhamento destinada aos estudantes?

☐ Várias vezes ao dia

☐ Uma ou duas vezes por dia

☐ Algumas vezes por semana

☐ Muito raramente

☐ Nunca.

9. Considera importante a disponibilização desta ferramenta?

☐ Sim

☐ Não

10. Sobre esta ferramenta em concreto tem alguma crítica/sugestão?

11. Para concluir, tem alguma crítica ou sugestão a fazer sobre o sistema e/ou a sua utilização?

Muito obrigado pela sua participação!

Anexo VIII - Estrutura do questionário feito aos estudantes (PI2/DW2 - ESTGOH - 2018/2019)

Avaliação do sistema CodeInsights

Ao longo das últimas semanas, tem vindo a utilizar o sistema CodeInsights no processo de resolução dos exercícios propostos nas aulas. Agradecíamos que nos fornecesse alguns detalhes sobre esta experiência. As perguntas seguintes têm como únicos objetivos apurar o que pode ter corrido menos bem com vista a melhorar de futuro e, em simultâneo, identificar os aspetos que podem e devem ser mantidos.

Desde já, muito obrigado pela sua participação.

Nuno Fonseca

1. Com que regularidade assistiu às aulas?

Marcar apenas uma oval.

	1	2	3	4	5	
Nunca	◯	◯	◯	◯	◯	Sempre

2. Já tinha utilizado o sistema CodeInsights anteriormente?

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

3. Considera importante a utilização deste tipo de sistemas de monitorização no processo de ensino/aprendizagem de programação?

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

4. Ao longo deste período verificou que, no seu dia a dia, o professor utilizava realmente o sistema para preparar as aulas e para dar assistência aos alunos?

Marcar apenas uma oval.

- Sim
- Não

5. Em caso de resposta afirmativa à pergunta anterior, pode apresentar alguns exemplos concretos?

_____

_____

_____

_____

_____

6. Com base nas suas respostas às perguntas anteriores, considerou relevantes as intervenções feitas pelo professor?

Marcar apenas uma oval.

- Sim
- Não
- Por vezes

7. Sentiu-se desconfortável com o facto de o sistema fazer a recolha automática do seu código e o professor ter acesso imediato a esse código?

Marcar apenas uma oval.

- Sim
- Não

8. Caso tenha respondido "sim" à pergunta anterior, pode dizer quais os motivos deste desconforto?

_____

_____

_____

_____

_____

9. De modo a contornar este desconforto, adotou algum tipo de medida?

Marcar apenas uma oval.

- Não
- Sim, fazia as submissões de forma “anónima” até ter uma solução estável.
- Sim, utilizava outro IDE até ter uma solução estável.
- Sim, removi o plug-in.
- Outra: _____

10. Sentir-se-ia mais confortável se pudesse decidir quando o código deve ou não ser enviado para o servidor?

Marcar apenas uma oval.

- Sim
- Não

Atom

As seguintes perguntas dizem respeito à utilização do Atom.

11. Acha que o facto de os exercícios serem automaticamente corrigidos no Atom ajudou no seu processo de aprendizagem?

Marcar apenas uma oval.

- Sim
- Não

12. Considera importante que sejam apresentados os inputs e outputs utilizados nos testes automáticos?

Marcar apenas uma oval.

- Sim
- Não

13. Tem alguma crítica, sugestão ou existe alguma informação que gostaria de ver adicionada a esta ferramenta no Atom?

Ferramenta de acompanhamento

As seguintes perguntas dizem respeito à utilização da ferramenta de acompanhamento das submissões dos alunos. (http://193.137.79.53/ci-pi2-1819/students/)

14. Com que regularidade acedeu a esta ferramenta?

Marcar apenas uma oval.

- Várias vezes ao dia
- Uma ou duas vezes por dia
- Algumas vezes por semana
- Muito raramente
- Nunca

15. Considera importante a disponibilização deste tipo de ferramenta que permita aos alunos ter uma ideia do trabalho efetuado?

Marcar apenas uma oval.

- Sim
- Não

16. Existe alguma informação ou funcionalidade que gostaria de ver adicionada a esta ferramenta?

Considerações finais

17. Em termos gerais, como avalia a utilização do sistema CodeInsights nesta disciplina?

Marcar apenas uma oval.

	1	2	3	4	5	
Muito má						Muito boa

18. Para concluir, tem alguma crítica ou sugestão a fazer?

Este conteúdo não foi criado nem aprovado pela Google.

Google Formulários

Anexo IX - Manual de instalação e utilização (IPJ - Acertar o Rumo - 2019/2020)

Manual de instalação e utilização - Alunos

Nuno Miguel Gil Fonseca

nunogil@dei.uc.pt

Conteúdo

Apresentação 1
Instalação 2
Alteração do *template* de criação de nova classe 4
Utilização 5
Criação de ficheiros e declaração e inicialização de variáveis 5
Executar o código desenvolvido 8

Apresentação

CodeInsights é uma ferramenta de monitorização do progresso dos alunos de programação com base em *snapshots* de código fonte.

Para o seu correto funcionamento é crucial que os alunos se identifiquem e que identifiquem o exercício que estão a resolver.

No caso de haver essa indicação no enunciado, é importante que sejam utilizados os nomes sugeridos para as variáveis de *input* e que o *output* respeite o formato sugerido.

Instalação

Para instalar o *plug-in* do CodeInsights para Eclipse no seu computador deverá começar por proceder à instalação do Eclipse e em seguida copiar o ficheiro fornecido (**`org.codeinsights8_0.8.0.jar`**) para a pasta **`dropins`** que se encontra localizada na mesma pasta em que o Eclipse foi instalado (*e.g.* `C:\Users\gil\eclipse\java-xxxxxx\eclipse\dropins`).

Se está a utilizar OS X, para saber onde o se encontra instalado o Eclipse, localize o ícone do Eclipse, faça "*right-click*" sobre o ícone e escolha a opção "Show Package Contents". Em seguida, aceda à pasta "Contents" e posteriormente à pasta "Eclipse". Finalmente deverá poder aceder à pasta **`dropins`** para a qual deverá copiar o ficheiro.

Nota: Esta operação deverá ser efetuada antes de executar o Eclipse. Caso o Eclipse já esteja a correr, deverá ser reiniciado.

Após ter procedido como apresentado anteriormente, poderá confirmar se o *plug-in* ficou bem instalado acedendo à opção **`Help`** > **`Instal New Software…`** tal como apresentado na Figura 1.

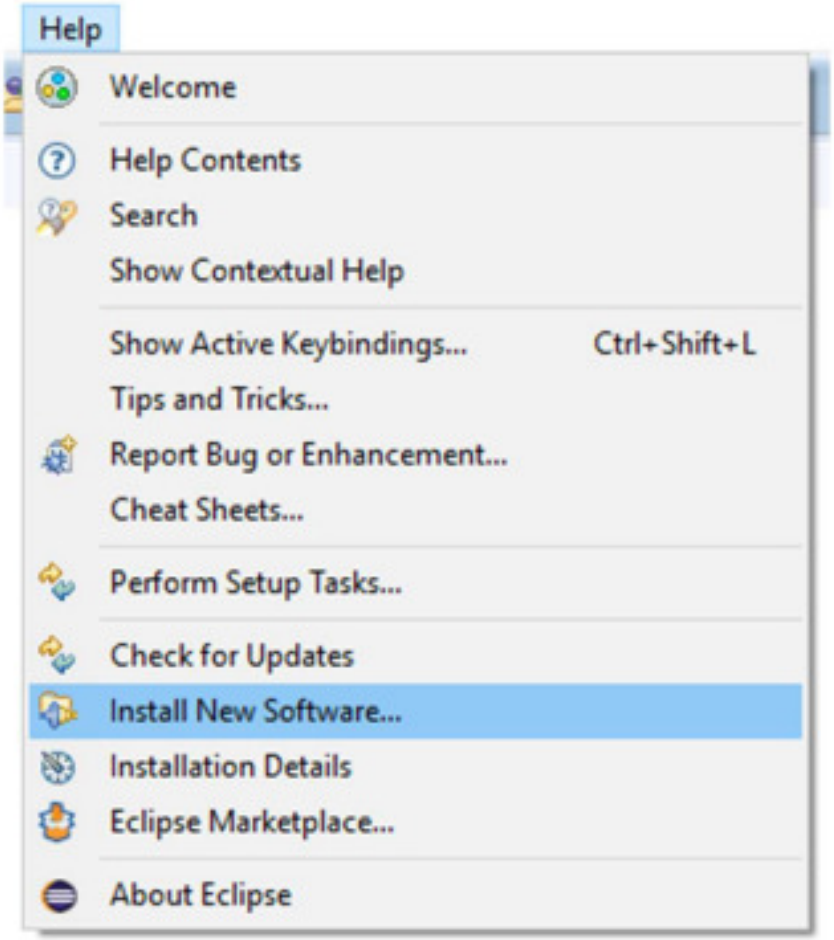

Figura 1 – Menu "Help"

Deverá surgir um diálogo semelhante ao que em seguida se apresenta na Figura 2.

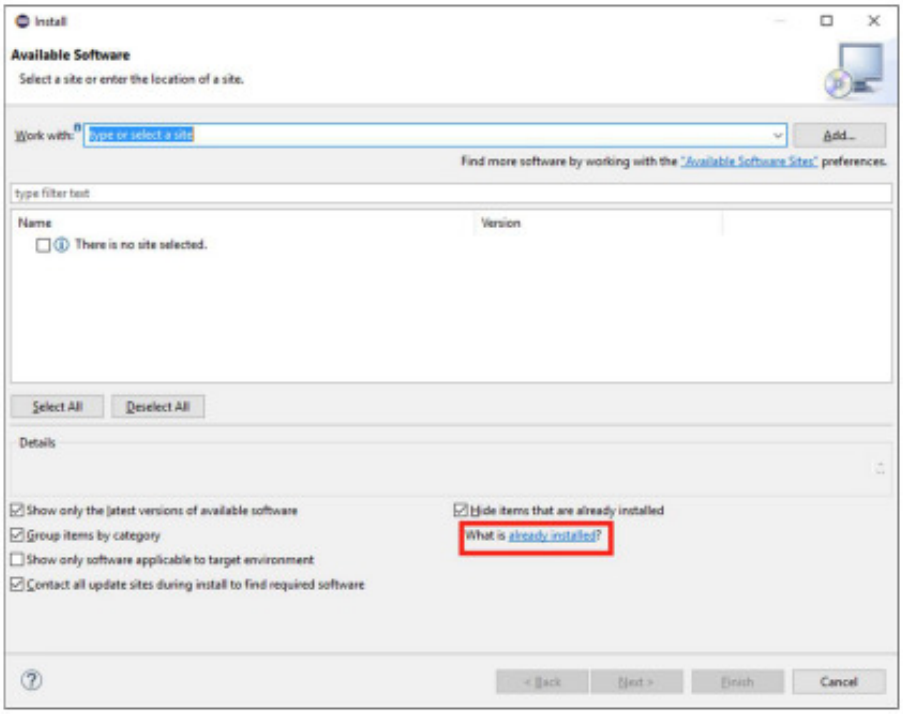

Figura 2 – Diálogo de pesquisa de plug-ins para instalar

Aqui chegado deverá clicar na opção "already installed" que se encontra destacada na Figura2. No diálogo que será apresentado em seguida escolha a aba "Plug-ins" e em seguida escreva "codeinsights" no campo de pesquisa. Se tudo tiver corrido como esperado, deverá surgir um item semelhante ao que é apresentado na Figura 3.

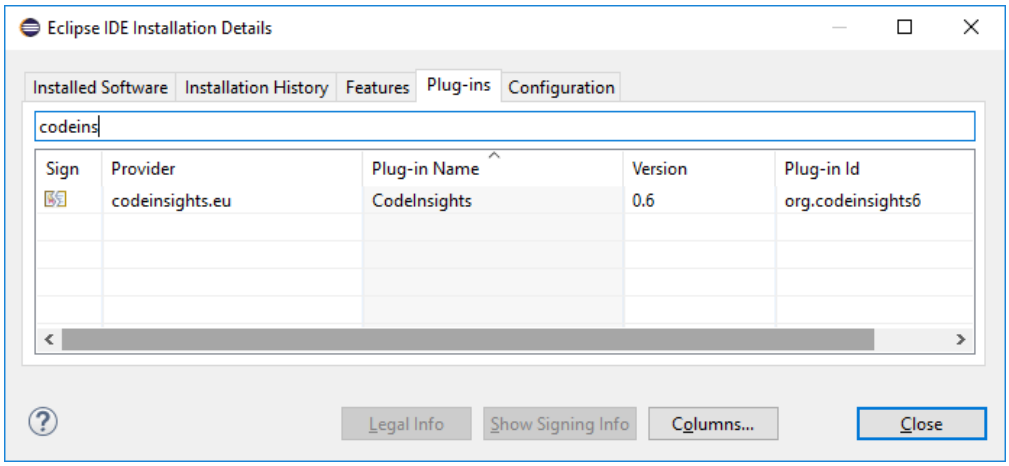

Figura 3 – Diálogo de confirmação da correta instalação do *plug-in*.

Caso este item não apareça, verifique se escreveu bem o texto a procurar. Se o item não aparecer de qualquer forma, é porque possivelmente ocorreu algum problema com a instalação do *plug-in*. Peça ajuda ao professor!

Alteração do *template* de criação de nova classe

Para o correto funcionamento do *plug-in* é necessário que os alunos forneçam alguma informação sobre o seu **número de aluno** (ou outro elemento identificativo) e sobre o exercício que estão a tentar resolver. De modo a agilizar este processo, deverá proceder à alteração do *template* de criação de uma nova classe Java. Para tal deverá aceder a `Window` > `Preferences` > `Java` > `Code Style` > `Code Templates` > `Code` escolher a opção "New Java file" e clicar em "Edit".

Aqui deverá fazer *copy/paste* do seguinte pedaço de código:

```
${filecomment}
${package_declaration}

/**
 * Breve descrição do código
 *
 * @sid 123456
 * @aid 1.1
 */

${typecomment}
${type_declaration}
```

Deverá ter o cuidado de substituir o valor que se encontra à frente de `@sid (student id)` pelo seu número de aluno. Após terminar a edição, basta clicar em "OK".

Em seguida, deverá escolher a opção "Method body" e clicar em Edit. Aqui chegado, deverá fazer *copy/paste* do seguinte pedaço de código:

```
//begin_inputs

//end_inputs

${body_statement}
```

Após terminar a edição, basta clicar em "OK".

Utilização

Criação de ficheiros e declaração e inicialização de variáveis

Deverá começar por criar um novo projeto. Em seguida, sempre que iniciar a resolução de um exercício, deverá começar por criar um novo ficheiro (`File > New > Class`). Deverá indicar o nome que pretende utilizar para a classe (Name:) e indicar que pretende que o método `main` seja criado de forma automática, tal como destacado na Figura 4.

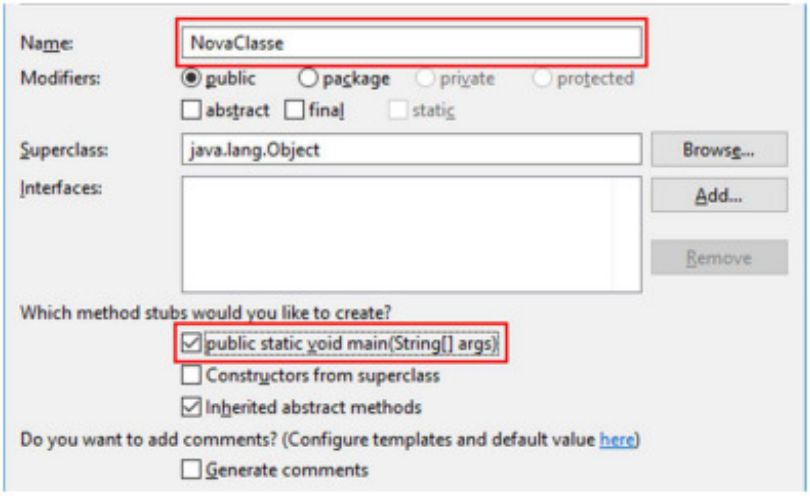

Figura 4 – Diálogo de criação de nova classe

Para terminar o processo, deverá clicar em "Finish". Nesse momento deverá ser criado no Eclipse um novo ficheiro "NovaClasse.java" que deverá estar preparado para ser identificado de forma muito mais rápida, sendo apenas **ter o cuidado de alterar o valor de `@aid` sempre que começar a resolver um novo exercício**. (`@aid 1.1` / `@aid 1.2` / `@aid 1.3` / ...). Em seguida apresenta-se um exemplo de um novo ficheiro acabado de criar.

```
/**
 * Breve descrição do código
 *
 * @sid 123456
 * @aid 1.1
 */

public class NovaClasse {
  public static void main(String[] args) {
    //begin_inputs

    //end_inputs
  }
}
```

Como referido anteriormente, é muito importante que na resolução dos exercícios sejam utilizados os nomes das variáveis de *input* indicadas, mas para além disso é igualmente vital que as variáveis sejam declaradas e inicializadas dentro dos limites delimitados por `//begin_inputs` e `//end_inputs`.

Vejamos um exemplo e um enunciado:

1.2. Desenvolva um programa que calcule o resultado da soma de dois números inteiros num1 e num2.

Exemplo de *output*:

2 + 2 = 4

Este exercício poderia ser resolvido pelo aluno com número **123456** do seguinte modo:

```
/**
 * Soma dois numeros
 *
 * @sid 123456
 * @aid 1.2
 */

public class SomaDoisNumeros {

  public static void main(String[] args) {

    //begin_inputs
    int num1 = 2;
    int num2 = 2;
    //end_inputs

    int soma = 0;
    soma = num1 + num2;

    System.out.println(num1 + "+" + num2 + "=" + soma);
  }
}
```

É importante garantir que entre **`//begin_inputs`** e **`//end_inputs`** se encontre apenas o código que diga respeito à declaração e inicialização das variáveis.

```
    //begin_inputs
    int num1 = 1;
    int num2 = 2;
    int soma = 0;
    //end_inputs
```

É igualmente importante que o valor das variáveis de *input* não seja alterado fora destes limites e que toda a lógica do exercício seja realizada fora destes limites.

```
//begin_inputs
int num1 = 1;
int num2 = 2;
//end_inputs

num2 = 7;

...
```

```
//begin_inputs
int num1 = 1;
int num2 = 2;
int soma = 0;
soma = num1 + num2;

//end_inputs

...
```

Mesmo que os valores das variáveis de *input* sejam introduzidos pelo utilizador, é importante que a regra se mantenha, tal como apresentado em seguida.

```
//begin_inputs
System.out.println("Valor do primeiro numero: ");
int num1 = sc.nextInt();
System.out.println("Valor do segundo numero: ");
int num2 = sc.nextInt();
//end_inputs

...
```

Executar o código desenvolvido

Para executar e testar o seu código basta clicar no botão associado à funcionalidade "Run", tal como pode ser observado na Figura 5. Em alternativa, poderá utilizar as teclas `Ctrl` + `F11`.

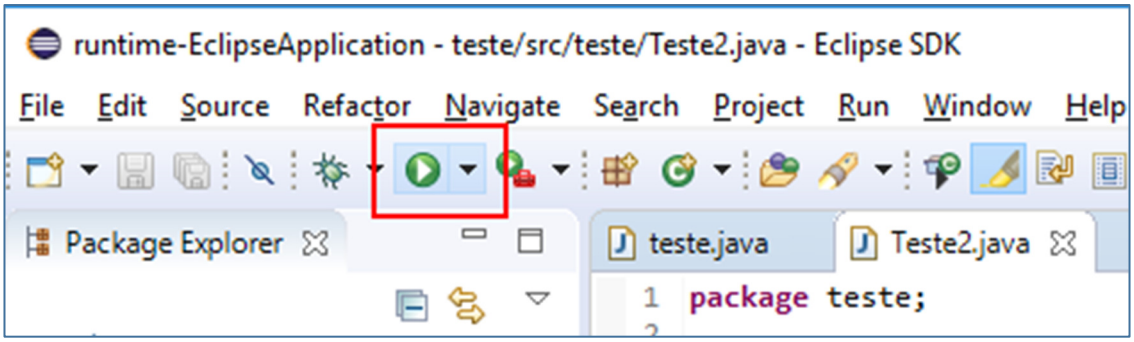

Figura 5 – O botão "Run"

Caso existam erros de compilação do seu código, estes surgirão na aba "Problems", tal como apresentado na Figura 6.

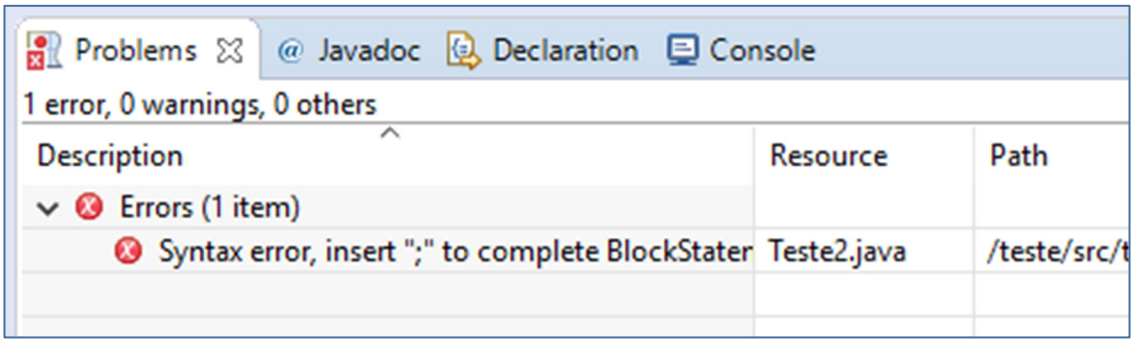

Figura 6 – Lista de problemas existentes com o código

Caso não exista qualquer problema com o seu código, este será executado numa "consola". Caso o seu programa envolva pedir dados ao utilizador, é aqui que os deverá introduzir. Na Figura 7 é apresentado um exemplo da execução de um programa em que é solicitado ao utilizador para inserir valores.

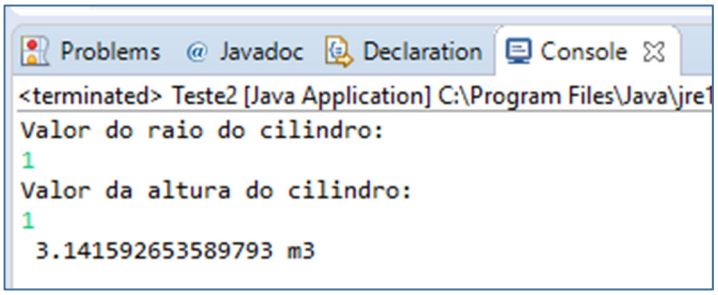

Figura 7 – Exemplo de execução de programa

Nesta consola poderá experimentar os seus programas, introduzindo os *inputs* necessários e observar o *output* produzido.

Por forma a permitir ter uma noção sobre a correção do seu código, sempre que possível, este é executado automaticamente no servidor com um conjunto de *inputs* de teste. Para que tal seja possível, é uma vez mais importante frisar que deverá respeitar as regras anteriormente apresentadas em relação à definição de *inputs* e formatação dos *outputs*.

De modo a saber se o código que está a desenvolver está a produzir (ou não) o *output* desejado poderá fazê-lo diretamente no Eclipse. Para tal, deverá aceder à consola "CodeInsights" clicando na opção de seleção de consolas, tal como apresentado na Figura 8.

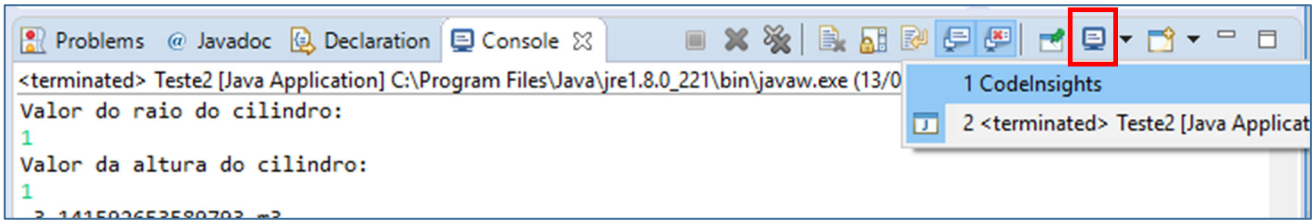

Figura 8 – Seleção da consola "CodeInsights"

A informação que é possível obter nesta consola é a mesma que é possível obter na informação de monitorização, ou seja: resultado do teste (Falhou/Passou/Quase passou); quais os *inputs* utilizados nos testes; *output* esperado e *output* obtido. Na Figura 9 é possível observar um exemplo do relatório de testes automáticos.

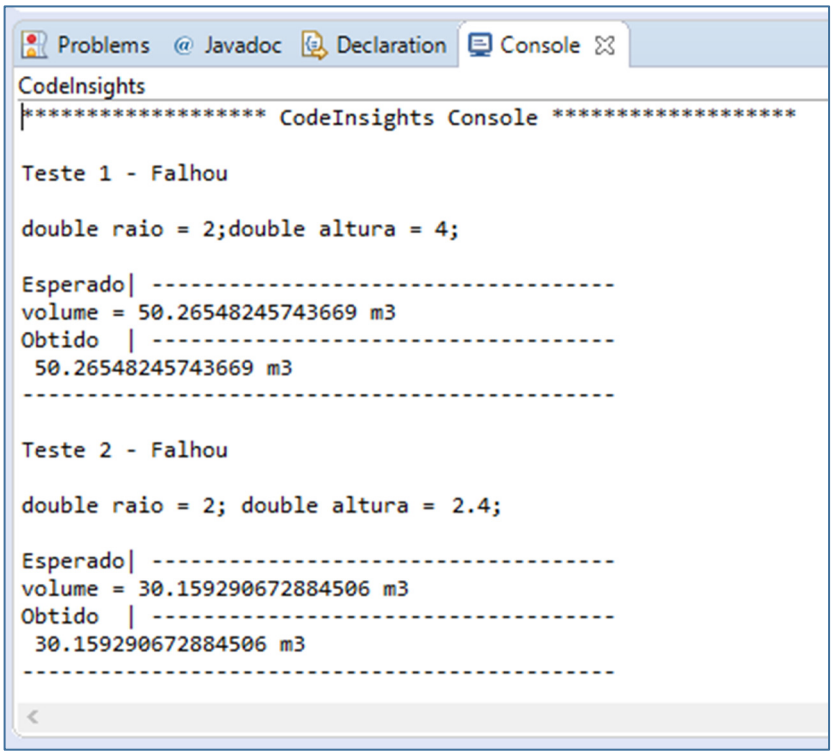

Figura 9 – Exemplo de relatório de testes automáticos

Neste exemplo é possível verificar que o código falhou em ambos os testes. Convirá a este respeito dizer para verificar se está certo ou errado, é feita uma comparação do *output* esperado e do *output* obtido. Assim, tal

como podemos observar no exemplo anterior, ainda que o resultado do cálculo em si esteja certo, a forma como o resultado é apresentado ao utilizador é diferente do que seria de esperar e como tal, não passou nos testes.

Regra geral, quando o resultado é "quase passou", significa que o resultado obtido está mesmo muito semelhante ao esperado, poderá ser apenas uma questão de um carater a mais ou a menos.

Nota importante: Dependendo do sistema operativo, o mesmo carater pode ser representado computacionalmente de forma diferente, assim, se no seu entender o resultado que o seu programa está a produzir está igual ao esperado, mas mesmo assim, não passa no(s) teste(s), não perca tempo e continue a trabalhar em outros exercícios.

Importante relembrar:

- De modo a passar automaticamente nos testes, o resultado deverá ser apresentado exatamente com o mesmo formato que é indicado nos exemplos.
- Os testes automáticos só são realizados com sucesso se utilizar como nomes das variáveis de *input* os nomes sugeridos no enunciado e se forem definidos entre `//begin_inputs` e `//end_inputs`
- É igualmente importante que não declare e inicialize entre `//begin_inputs` e `//end_inputs` nenhuma "variável de trabalho", ou seja, que não seja um input.

Caso tenha algum problema com a utilização ou acesso ao sistema poderá enviar um email para nunogil@dei.uc.pt e tentarei resolver o problema com a maior brevidade possível.

Anexo X - Estrutura base da entrevista realizada aos estudantes (IPJ - Acertar o Rumo - 2019/2020)

CodeInsights - Inquérito

Acertar o Rumo 2019/2020

Ao longo das últimas semanas, tem vindo a utilizar o sistema *CodeInsights* no processo de resolução dos exercícios propostos nas aulas. Agradecíamos que nos fornecesse alguns detalhes sobre esta experiência. As perguntas seguintes têm como único objetivo tentar perceber o que pode ter corrido menos bem, com vista a melhorar de futuro e, em simultâneo, identificar os aspetos que podem e devem ser mantidos.

Para todas as perguntas será disponibilizado um espaço no qual poderá sustentar a sua resposta.

Desde já, muito obrigado pela sua participação.

1. Considera importante a **utilização deste tipo de sistemas de monitorização** no processo de ensino aprendizagem de programação?

☐ Sim

☐ Não

2. Ao longo deste período sentiu, no seu dia a dia, que o(s) professor(es) **utilizavam realmente o sistema** para suportar a preparação das aulas e/ou a assistência dada aos alunos?

☐ Sim

☐ Não

3. Em caso de resposta afirmativa à pergunta nº2, pode apresentar exemplos concretos?

4. Com base nas suas respostas às perguntas nº2 e nº3, considerou **relevantes as intervenções** feitas pelo(s) professor(es), com base na **informação fornecida pelo sistema**?

☐ Sim

☐ Não

☐ Nem sempre

5. Alguma vez se sentiu **desconfortável** com o facto de saber que o sistema faz a **recolha do seu código** de cada vez que é feita uma compilação e que o professor tem acesso imediato a esse código?

☐ Sim

☐ Não

6. Em caso afirmativo à questão anterior, pode indicar os **motivos** para tal desconforto?

7. De modo a contornar este desconforto, adotou algum tipo de medida?

☐ Não.

☐ Sim, fazia as submissões de forma "anónima" até ter uma solução estável.

☐ Sim, utilizava outro IDE até ter uma solução estável.

☐ Sim, removi o *plug-in*.

☐ Sim, outra(s) medida(s) (por favor, concretize)

8. Sentir-se-ia mais confortável se pudesse decidir **quando** o código deve ou não ser enviado para o servidor?

☐ Sim

☐ Não

Relativamente à apresentação dos resultados dos testes na consola do Eclipse.

9. Com que regularidade acedia a esta ferramenta?

- ☐ Sempre que compilava
- ☐ Muitas vezes
- ☐ Quando já tinha a versão final
- ☐ Muito raramente
- ☐ Nunca.

10. Dava importância aos resultados apresentados?

- ☐ Sim, sempre.
- ☐ Por vezes
- ☐ Raramente

11. Considera importante a **disponibilização deste tipo de ferramenta** que permita aos alunos ter algum tipo de validação imediata do trabalho efetuado?

- ☐ Sim
- ☐ Não

12. Sobre esta ferramenta em concreto tem alguma crítica/sugestão?

Relativamente à ferramenta de acompanhamento (http://gilito.com.pt/ci-ar-1920/students/**)**

12. Com que regularidade acedeu a esta ferramenta?

- ☐ Várias vezes ao dia
- ☐ Uma ou duas vezes por dia
- ☐ Algumas vezes por semana
- ☐ Muito raramente
- ☐ Nunca.

13. Considera importante que os professores **possam enviar *feedback* aos alunos** utilizando esta ferramenta?

☐ Sim

☐ Não

14. Sobre esta ferramenta em concreto tem alguma crítica/sugestão?

15. Para concluir, tem alguma crítica ou sugestão a fazer sobre o sistema e/ou a sua utilização?

Muito obrigado pela sua participação!

Anexo XI - Estrutura do questionário feito aos professores (IPJ - Acertar o Rumo - 2019/2020)

CodeInsights

Entrevista de avaliação/validação

Experiência: Acertar o Rumo 2019/2020 (Java) (CodeInsights v6)

Professor:

1. **Com que regularidade utilizou o sistema *CodeInsights* ao longo da experiência?**

2. **De que modo utilizou o *CodeInsights*?**

3. **Qual a funcionalidade do *CodeInsights* que utilizou com mais regularidade?**

4. **Sentiu que com a utilização do *CodeInsights* teve acesso a informação sobre o modo de trabalho dos alunos que não teria de outro modo? Em caso afirmativo, pode apresentar exemplos concretos?**

5. **No decorrer das aulas, tomou decisões influenciadas por informação obtida no *CodeInsights*? Em caso afirmativo, pode apresentar exemplos concretos?**

6. **Sentiu alguma dificuldade a interpretar a informação disponibilizada? Se sim pode dar exemplos?**

7. **Do ponto de vista da usabilidade/facilidade de utilização, teve alguma dificuldade? Existe algum aspeto que considere que deva ser melhorado?**

8. **Qual a sua sensibilidade face à relação entre os alunos e o sistema?**

9. **Considera que a imagem que o sistema lhe dá sobre os alunos corresponde à realidade?**

10. **Em situações futuras consideraria utilizar o *CodeInsights* para suportar as disciplinas de programação que venha a lecionar?**

11. **Tem mais alguma crítica ou sugestão que gostaria de fazer?**

Anexo XII - Estrutura do questionário feito aos estudantes (PI2/DW2 - ESTGOH - 2019/2020)

Avaliação do sistema CodeInsights - PI2/DW2"

Durante este semestre recorreu-se ao sistema CodeInsights para suportar o processo de resolução dos exercícios propostos nas aulas e permitir que o docentende fizesse um acompanhamento mais próximo do trabalho desenvolvido pelos alunos. Agradecíamos que nos fornecesse alguns detalhes sobre esta experiência. As perguntas seguintes têm como únicos objetivos apurar o que pode ter corrido menos bem com vista a melhorar de futuro e, em simultâneo, identificar os aspetos que podem e devem ser mantidos.

Desde já, muito obrigado pela sua participação.

Nuno Fonseca

*Obrigatório

1. Com que regularidade assistiu às aulas? *

Marcar apenas uma oval.

	1	2	3	4	5	
Nunca	◯	◯	◯	◯	◯	Sempre

2. Com que regularidade resolvia os exercícios propostos pelo professor? *

Marcar apenas uma oval.

	1	2	3	4	5	
Nunca	◯	◯	◯	◯	◯	Sempre

3. Com que regularidade fazia a submissão das soluções por si desenvolvidas no sistema CodeInsights? *

Marcar apenas uma oval.

	1	2	3	4	5	
Nunca	◯	◯	◯	◯	◯	Sempre

4. Pode justificar a sua resposta à pergunta anterior?

5. Ao longo deste período verificou que, no seu dia a dia, o professor utilizava realmente o sistema para preparar as aulas e para dar assistência aos alunos? *

Marcar apenas uma oval.

- Sim
- Não

6. Em caso de resposta afirmativa à pergunta anterior, pode apresentar alguns exemplos concretos?

7. Com base nas suas respostas às perguntas anteriores, considerou relevantes as intervenções/feedback feitas pelo professor? *

Marcar apenas uma oval.

- Sim
- Não
- Por vezes

8. Ao contrário do que havia acontecido no passado, desta vez, as submissões não eram enviadas de forma automática, podendo os alunos escolher o que submeter e quando submeter. Prefere desta forma?

Marcar apenas uma oval.

- Sim
- Não
- É indiferente

9. Pode justificar a sua resposta à pergunta anterior?

Ferramenta de acompanhamento

As seguintes perguntas dizem respeito à utilização da ferramenta de acompanhamento das submissões dos alunos. (http://193.137.79.53/ci-pi2-1920/students/)

10. Com que regularidade acedeu a esta ferramenta?

Marcar apenas uma oval.

- Várias vezes ao dia
- Uma ou duas vezes por dia
- Algumas vezes por semana
- Muito raramente
- Nunca

11. Considera importante a disponibilização deste tipo de ferramenta que permita aos alunos ter uma ideia do trabalho efetuado?

Marcar apenas uma oval.

- Sim
- Não

12. Existe alguma informação ou funcionalidade que gostaria de ver adicionada a esta ferramenta?

Considerações finais

13. Em termos gerais, como avalia a utilização do sistema CodeInsights nesta disciplina?

Marcar apenas uma oval.

	1	2	3	4	5	
Muito má						Muito boa

14. Para concluir, tem alguma crítica ou sugestão a fazer?

Este conteúdo não foi criado nem aprovado pela Google.

Google Formulários

Anexo XIII - Estrutura do questionário feito aos estudantes (FP - ESTGOH 2020/2021)

Avaliação do sistema CodeInsights - FP 20 21

Ao longo das últimas semanas, utilizou o sistema CodeInsights no processo de resolução dos exercícios propostos nas aulas. Agradecíamos que nos fornecesse alguns detalhes sobre esta experiência. As perguntas seguintes têm como únicos objetivos apurar o que pode ter corrido menos bem com vista a melhorar de futuro e, em simultâneo, identificar os aspetos que podem e devem ser mantidos.

Desde já, muito obrigado pela sua participação.

Nuno Fonseca

1. Com que regularidade assistiu às aulas (presencialmente ou online)?

Marcar apenas uma oval.

	1	2	3	4	5	
Nunca	◯	◯	◯	◯	◯	Sempre

2. Com que regularidade resolveu os exercícios propostos pelo professor?

Marcar apenas uma oval.

	1	2	3	4	5	
Nunca	◯	◯	◯	◯	◯	Sempre

3. No atual contexto de pandemia considera importante a utilização deste tipo de sistemas de monitorização no processo de ensino/aprendizagem de programação?

Marcar apenas uma oval.

	1	2	3	4	5	
Nada importante	◯	◯	◯	◯	◯	Muito importante

4. Relativamente à pergunta anterior, poderia justificar a sua resposta?

5. Ao longo deste período verificou que, no seu dia a dia, o professor utilizava realmente o sistema para preparar as aulas e para dar assistência aos alunos?

Marcar apenas uma oval.

- Sim
- Não

6. Em caso de resposta afirmativa à pergunta anterior, pode apresentar alguns exemplos concretos?

7. Com base nas suas respostas às perguntas anteriores, considerou relevantes as intervenções feitas pelo professor?

Marcar apenas uma oval.

- Sim
- Não
- Por vezes

8. Sentiu-se desconfortável com o facto de o sistema fazer a recolha automática do seu código e o professor ter acesso imediato a esse código?

Marcar apenas uma oval.

- Sim
- Não

9. Relativamente à pergunta anterior, pode justificar a sua resposta?

_____

_____

_____

_____

_____

10. De modo a contornar este desconforto, adotou algum tipo de medida?

Marcar apenas uma oval.

- Não
- Sim, fazia as submissões de forma "anónima" até ter uma solução estável.
- Sim, utilizava outro IDE até ter uma solução estável.
- Sim, removi o plug-in.
- Outra: _____

11. Sentir-se-ia mais confortável se pudesse decidir quando o código deve ou não ser enviado para o servidor?

Marcar apenas uma oval.

- Sim
- Não

Editor de código

As seguintes perguntas dizem respeito ao editor de código utilizado

12. Qual o editor de código que utilizou para resolver os exercícios?

Marcar apenas uma oval.

- Atom
- O editor do CodeInsights
- Outra: _____

13. Acha que o facto de os exercícios serem automaticamente corrigidos ajudou no seu processo de aprendizagem?

Marcar apenas uma oval.

- Sim
- Não

14. Considera importante que sejam apresentados os inputs e outputs utilizados nos testes automáticos?

Marcar apenas uma oval.

- Sim
- Não

15. Tem alguma crítica, sugestão ou existe alguma informação que gostaria de ver adicionada a esta ferramenta no Atom ou no editor de código do CodeInsights?

_____
_____
_____
_____
_____

Ferramenta de acompanhamento

As seguintes perguntas dizem respeito à utilização da ferramenta de acompanhamento das submissões dos alunos. (http://193.137.79.53/ci-pi2-2021/students/)

16. Com que regularidade acedeu a esta ferramenta?

Marcar apenas uma oval.

- Várias vezes, mesmo nos dias em que não tinha aula
- Várias vezes nos dias em que tinha aula
- Algumas vezes por semana
- Muito raramente
- Nunca

17. Quando utilizava a ferramenta, como valorizava a informação apresentada (por exemplo, se tinha ou não passado em todos os testes)?

Marcar apenas uma oval.

	1	2	3	4	5	
Nada						Muito

18. Considera importante a disponibilização deste tipo de ferramenta que permita aos alunos ter uma ideia do trabalho efetuado e ver o feedback dado pelo professor?

Marcar apenas uma oval.

- Sim
- Não

19. Existe alguma informação ou funcionalidade que gostaria de ver adicionada a esta ferramenta?

_____

_____

_____

_____

_____

Considerações finais

20. Em termos gerais, acha que a utilização do sistema CodeInsights nesta disciplina ajudou na sua aprendizagem?

Marcar apenas uma oval.

	1	2	3	4	5	
Nada	◯	◯	◯	◯	◯	Muito

21. Para concluir, tem alguma crítica ou sugestão a fazer?

Este conteúdo não foi criado nem aprovado pela Google.

Google Formulários