Alexandre A. M. Ferreira

PLUGIN INTERATIVO DE SUPORTE À APRENDIZAGEM DE PROGRAMAÇÃO

1 2 9 0

UNIVERSIDADE D COIMBRA

Alexandre Américo Monteiro Ferreira

PLUGIN INTERATIVO PARA SUPORTE À APRENDIZAGEM DE PROGRAMAÇÃO

VOLUME 1

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software orientada pelo Professor Doutor António José Mendes e pela Professora Doutora Anabela Gomes e apresentada à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Outubro de 2021

Faculdade de Ciências e Tecnologia
Departamento de Engenharia Informática

PLUGIN INTERATIVO PARA SUPORTE À Aprendizagem de programação

Alexandre Américo Monteiro Ferreira

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software orientada pelo Professor Doutor António José Mendes e pela Professora Doutora Anabela Gomes e apresentada à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática

Outubro de 2021

UNIVERSIDADE D
COIMBRA

Resumo

Nas últimas décadas, a área da informática tem tido um crescimento exponencial, que tem vindo sempre a acelerar ao longo dos anos. Atualmente, esta área está completamente integrada na sociedade e representa partes indispensáveis do quotidiano das pessoas. Desta forma, o desenvolvimento desta indústria tem-se intensificado e há, cada vez, mais empresas com um papel ativo na evolução deste setor já que existe uma necessidade crescente de desenvolvimento. De igual forma, a produção destas empresas é maior, o que resulta num crescimento da procura de profissionais qualificados para integrar as suas equipas. O aumento da oferta e a boa remuneração deste tipo de cargos é a conjugação perfeita de fatores que têm atraído cada vez mais jovens à procura de uma carreira profissional de sucesso, a candidatarem-se a cursos superiores na área de informática. Contudo, nestes cursos, é frequente que uma grande parte dos estudantes se deparem com dificuldades iniciais por não conseguirem entender e aplicar os conceitos ensinados em unidades curriculares de Introdução à Programação, acabando por desistir. Tendo em vista esta realidade, surge uma proposta descrita neste documento com o intuito de contribuir para a solução destes problemas que muitos estudantes sentem quando ingressam em cursos de programação. O projeto que integra a proposta mencionada refere-se a um *plugin* de construção e simulação de algoritmos para o Moodle. Este *plugin* foca-se na construção de uma plataforma que permita aos estudantes construírem programas através de fluxogramas e observarem a animação da execução dos programas construídos. O objetivo principal é o de facilitar a aprendizagem inicial de programação por parte dos estudantes, reforçando aspetos que facilitem a apreensão de conceitos lecionados em sala de aula e a sua aplicação, de forma a que estes consigam melhor visualizar e concretizar conceitos mais abstratos de programação.

Palavras-Chave

Introdução à Programação, Construção e simulação de algoritmos, Fluxogramas, Moodle

[Página deixada em branco intencionalmente]

Abstract

In the last decades, the IT area has grown exponentially, which has always been accelerating over the years. Currently, this area is completely integrated into society and represents indispensable parts of people's daily lives. Thus, this industry's development has intensified, and there are, increasingly, more companies with an active role in the evolution of this sector as there is a growing need for development. Likewise, these companies' production is higher, which increases the demand for qualified professionals to integrate their teams. The increase in the offer and the good remuneration of these types of positions is the perfect combination of factors that have attracted more and more young people looking for a successful professional career to apply for higher education courses in information technology. However, in these courses, it is frequent that many students face initial difficulties because they are unable to understand and apply the concepts taught in Introduction to Programming curricular units, eventually giving up. Because of this reality, there is a proposal described in this document to contribute to the solution of these problems that many students experience when entering programming courses. The project that integrates the mentioned proposal refers to an algorithm building and simulation *plugin* for Moodle. This *plugin* focuses on building a platform that allows students to build programs using flowcharts and observe the animation of the built programs' execution. The main objective is to facilitate students' initial learning of programming, reinforcing aspects that facilitate the apprehension of concepts taught in the classroom and their application so that they can better visualize and concretize more abstract concepts of programming.

Keywords

Programming Introduction, Construction and simulation of algorithms, Fluxograms, Moodle

[Página deixada em branco intencionalmente]

Agradecimentos

Em primeiro lugar, gostaria de agradecer em particular ao professor António José Mendes e à professora Anabela José Gomes pela sua total disponibilidade e apoio na realização de todas as atividades que envolveram este projeto. Por fim, gostaria de deixar uma mensagem de agradecimento a toda a família e amigos por todo o apoio oferecido ao longo dos anos, nomeadamente, durante o meu percurso académico.

Este trabalho teve o apoio financeiro da Fundação para a Ciência e a Tecnologia, I.P. (FCT), através de fundos nacionais no âmbito do projeto CISUC com a referência UIDB/00326/2020.

[Página deixada em branco intencionalmente]

Índice

[Página deixada em branco intencionalmente]

Acrónimos

API Application Programming Interface
Moodle Modular Object-Oriented Dynamic Learning Environment
JSON Java*Script* Object Notation
HTML HyperText Markup Language
AMD Asynchronous module definition
PHP Hypertext Preprocessor
VPL Virtual Programming Lab
URL Uniform Resource Locator
SVG Scalable Vector Graphics
LMS Learning Managment System

[Página deixada em branco intencionalmente]

Lista de Figuras

[Página deixada em branco intencionalmente]

Lista de Tabelas

Capítulo 1
Introdução

As dificuldades sentidas pelos estudantes na aprendizagem dos conceitos de programação tem sido um problema que as instituições de ensino têm tentado solucionar [1]. Neste capítulo, procurar-se-á, numa primeira fase, contextualizar o problema e expor como é que este motivou o desenvolvimento deste projeto. De seguida, apresentar-se-ão os objetivos que se consideraram fundamentais para alcançar as metas a que este projeto se propõe. Por fim, será feita uma pequena abordagem em relação à estruturação do documento.

1.1 Contexto

Atualmente, a informática tem um papel claramente influente na sociedade. Tendo tido um papel fundamental na melhoria da qualidade de vida das pessoas ao longo dos anos, é cada vez mais impensável imaginar um futuro sem tecnologia. Esta preponderância faz com que cada vez mais empresas invistam e apostem no desenvolvimento de produtos neste ramo. Por consequência, tem-se verificado um elevado crescimento na procura de trabalhadores qualificados para este tipo de ofícios. Graças a isto, e aliado ao facto destas empresas proporcionarem salários aliciantes aos seus trabalhadores, cada vez mais jovens têm começado a ganhar interesse em desempenhar funções na área. Devido a estes fatores, tem-se verificado um grande aumento na afluência a cursos de informática por parte dos estudantes. Contudo, alguns deles são muitas vezes pressionados pelos pais a optarem por esta que é vista como uma área de sucesso, mesmo não tendo vocação ou particular interesse por este tipo de cursos. Assim sendo, quando, numa fase inicial, são confrontados com problemas na aprendizagem, é expectável que um elevado número deles acabe por se desmotivar e, consequentemente, desistir. As elevadas taxas de reprovação em unidades curriculares introdutórias de programação é um exemplo deste problema e o foco do estudo apresentado neste documento. Este cenário preocupante incentiva as universidades e institutos a investir na procura de métodos que auxiliem os seus alunos a colmatar estas dificuldades. Deste modo, melhora-se o seu desempenho e, por conseguinte, evita-se a sua evasão dos cursos. Neste contexto, surge este projeto que visa a criação de uma ferramenta que facilite a compreensão e aplicação dos vários conceitos que envolvem a aprendizagem inicial de programação.

1.2 Motivação

Como o papel das instituições de ensino é fornecer as melhores condições de aprendizagem possíveis aos seus estudantes, considerou-se que desenvolver uma

proposta de contribuição para a resolução deste tipo de problemas seria de primordial importância. Assim sendo, acreditou-se que deveriam ser criadas ferramentas para facilitar a sua aprendizagem, de modo que os estudantes se sintam mais integrados e interessados no curso. Visto que o seu sucesso não é independente do trabalho desenvolvido pelos seus docentes, seria também importante que se criassem meios que os ajudassem a explicar melhor os aspetos da programação, e ainda a clarificar as dúvidas dos seus aprendizes num contexto de sala de aula. Ao melhorar o aproveitamento de ambos desde cedo, achou-se que existiria a capacidade de promover uma experiência mais frutífera para os estudantes numa fase inicial da sua jornada académica. Foi com este intuito que se decidiu desenvolver esta proposta. Sumariamente, com este projeto tencionou-se desenvolver um meio que auxilie os estudantes a atingir melhor os requisitos propostos nas unidades curriculares introdutórias de programação, e que permita ainda aos docentes ajudá-los a alcançar estes objetivos mais facilmente.

1.3 Objetivos

O surgimento deste projeto teve como intuito desenvolver um ambiente que permitisse alcançar dois grandes objetivos:

O primeiro foi o de desenvolver uma ferramenta de construção de algoritmos através de fluxogramas, que permita, posteriormente, observar uma simulação das operações executadas pelo computador do algoritmo criado. A utilização desta ferramenta destina-se, principalmente, aos estudantes. Ainda assim, os docentes, caso pretendam fazer alguma demonstração, também poderão ter acesso a esta. A sua criação teve como intuito promover uma melhor compreensão e aplicação dos conceitos básicos de programação por parte dos estudantes.

O segundo objetivo considerou importante foi criar esta ferramenta num ambiente de aprendizagem onde, no futuro, possam ser implementadas funcionalidades que permitam aos professores fazer um acompanhamento mais individualizado do progresso dos seus estudantes. Desta forma, estes teriam maior facilidade em identificar os problemas específicos que cada um dos estudantes pudesse apresentar no processo de aprendizagem.

Dentro do grupo de estudantes que ingressam em cursos de informática, incluem-se estudantes que nunca tiveram qualquer experiência anterior com programação. Por vezes, este é um fator que influência negativamente os resultados ao nível da programação, numa fase inicial do curso[2]. A necessidade de os estudantes aprenderem o raciocínio lógico subjacente à programação e estarem em simultâneo, pela primeira vez, estarem em contacto com os detalhes sintáticos de uma linguagem de programação revela-se ser uma tarefa demasiado exigente para alguns[3]. Este fator é agravado pelo curto espaço de tempo que estes jovens têm para realizar essa adaptação[3]. Para além disso, o foco, que deveria existir em termos algorítmicos na aprendizagem inicial de programação, é desviado [4]. Assim, decidiu-se desenvolver esta ferramenta de modo a que os estudantes construam os seus algoritmos sem necessitarem de recorrer a todos os detalhes sintáticos de uma linguagem de programação. Como tal, existiam dois caminhos para atingir este fim:

- Uma mais verbal, com o uso de pseudocódigo;
- Uma mais gráfica, através da utilização de fluxogramas.

Uma vez que a maioria dos estudantes a quem se destina a ferramenta em causa é de uma licenciatura em engenharia e, como tal, com perfil de aprendizagem marcadamente visual [1] juntamente com o facto de os fluxogramas serem visualmente mais apelativos e elucidativos em relação às várias estruturas que o compõe, fez com que a sua opção prevalecesse em relação à do pseudocódigo. A gestão do espaço, normalmente exigida por esta abordagem, não será problemática dada a reduzida complexidade dos algoritmos. Assim, através da redução da sintaxe de uma linguagem de programação e da carga cognitiva associada à sua aprendizagem, é permitido aos estudantes focarem-se essencialmente na compreensão dos conceitos básicos da programação. Deste modo, poderão centrar-se na construção lógica de estratégias para resolverem problemas utilizando esses mesmos conceitos. Na verdade, esta é a parte fundamental para desenvolver competências de programação [4]. Esta aposta deverá tornar o processo de adaptação dos estudantes mais gradual, o que poderá traduzir-se numa aprendizagem mais facilitada.

Na perspetiva dos estudantes iniciantes é bastante complicado entender a forma como um computador opera, ou seja, como é que este interpreta as informações que lhe são passadas num algoritmo e como este se comporta após analisar as mesmas. Estas adversidades são perfeitamente compreensíveis já que este funcionamento é uma matéria abstrata, o que dificulta a sua compreensão[5][6]. Assim sendo, através da simulação de algoritmos, pensou-se que seria possível eliminar esta abstração, ajudando os estudantes a entender mais facilmente os procedimentos que ocorrem num programa. Para que este objetivo fosse alcançado, pensou-se que se poderia fornecer uma animação especifica a cada instrução do algoritmo que o estudante criou. Desta forma, incluiu-se a possibilidade de simular e animar os algoritmos construídos de modo a permitir visualizar os seguintes aspetos:

- Representação dos dados;
- Representação do fluxo de um algoritmo;
- Modo como são feitas as operações.

Através da inclusão destes aspetos no processo de simulação, foi possível reduzir consideravelmente o esforço cognitivo necessário para a compreensão conceptual da programação, já que os estudantes poderão acompanhar visualmente a execução do programa passo a passo[3],[6]. Assim, conseguirão formular mais facilmente os seus modelos mentais para cada um dos conceitos apresentados nas animações. Para tal, achou-se importante destacar todos os elementos que são utilizados durante a execução de cada instrução. Através deste tipo de ferramenta, acreditou-se que seria possível fornecer ao estudante um método de aprendizagem que o focasse na compreensão dos conceitos. Para além disto, ao permitir que eles controlem o desenrolar da animação, torna-se possível fornecer-lhes meios para que estes possam fazer *debug* do seu programa mais facilmente. Desta maneira, terão capacidade de identificar e reparar os erros lógicos que cometeram no seu algoritmo, auxiliando-os também na construção de estratégias para abordar determinado tipo de problema [4]. De modo a tornar isto possível, na fase de construção do algoritmo, incluíram-se estratégias que visaram a redução de erros sintáticos durante este processo. Por

exemplo, impedir que o estudante implemente expressões sintaticamente mal formuladas ou que utilizem valores de tipos distintos numa determinada operação. Assim, para além de se permitir que os estudantes observem as animações sem que ocorra um grande número de erros, acreditou-se que também se estaria a incutir os princípios corretos ao estudante nesta fase. Este fator torna-se importante pois permite que a animação foque na correção dos erros lógicos cometidos pelos estudantes e não nos erros sintáticos que estes poderiam ocorrer no processo de construção do fluxograma.

De modo a alcançar estas duas metas, decidiu-se construir um *plugin* para o Moodle (Modular Object Oriented Distance Learning). O Moodle é um dos sistemas de gestão de aprendizagem mais populares no mundo[8]. Esta é uma plataforma que está em constante evolução e que possui uma grande comunidade de suporte e desenvolvimento repleta de desenvolvedores que também pretendem melhorar o processo de aprendizagem nas mais variadas áreas. Esta plataforma dá aos programadores a possibilidade de personalizarem os seus ambientes de acordo com as suas preferências, por exemplo através da disponibilização de vários tipo de *layout* para as páginas do Moodle, e expor os conteúdos que desejar. Como se trata de uma plataforma de ensino, já possui várias características que podem ter relevância para o projeto aqui apresentado, nomeadamente a inclusão de serviços e aspetos estatísticos com relevo pedagógico. Desta forma, poder-se-á tirar partido destas funcionalidades já existentes e integrá-las com o *plugin* que desenvolvido. Ao mesmo tempo, esta plataforma também oferece um sistema robusto, seguro e integrado graças ao seu núcleo de funcionalidades. Esta integração de sistemas é facilmente feita através da junção de vários tipos de *plugins* que estendem as funcionalidades desejadas. Dado o conteúdo da proposta do documento, seria benéfica, no futuro, a integração do *plugin* do projeto com um *plugin* como o Code Runner que avaliasse o código resultante do fluxograma criada pelo utilizador. Quando analisadas como um todo, estas características tornam evidente o potencial da plataforma para criar um ambiente ideal tanto para os estudantes como para os professores.

1.4 Estrutura do documento

O documento encontra-se dividido nos seguintes capítulos:

- **Cap. 2 – Estado da Arte –** capítulo que se inicia com a identificação dos fatores que conduzem ao insucesso dos estudantes em unidades curriculares de programação. De seguida encontra-se a análise a várias ferramentas semelhantes à que os autores deste documento projetaram. Por fim, segue-se uma análise a algumas funcionalidades do Moodle que se consideraram relevantes para a integração com projeto.
- **Cap.3 – Requisitos –** apresentação dos requisitos funcionais do projeto juntamente com os atributos de qualidade a atingir.
- **Cap.4 – Ferramenta de Construção e Simulação de algoritmos** – este capítulo contém a descrição da ferramenta desenvolvida. Nesta descrição são abordados todos os aspetos relativos às suas funcionalidades e os motivos pedagógicos que levaram à sua utilização.
- **Cap.5 – Arquitetura –** neste capítulo realiza-se primeiro uma contextualização em relação à composição da arquitetura de um sistema

Moodle. Após essa fase, é apresentada a arquitetura do plugin do projeto do documento e o modo como vão ser alcançados os atributos de qualidade definidos no capítulo anterior. Por fim, este capítulo termina com a apresentação do planeamento definido para o projeto.

- **Cap.6 – Implementação –** capítulo que descreve o modo como foi realizada a implementação da proposta do projeto. Esta descrição inicia-se com uma visão geral de como se encontra estruturado o projeto e as respetivas funcionalidades de cada componente que o compõe. Por fim, apresenta-se a uma visão mais aprofundada de como estas componentes implementam as suas funcionalidades.
- **Cap. 7 – Conclusão –** por fim, este capítulo inicia com uma descrição do estado atual do projeto. De seguida realiza uma análise comparativa entre o trabalho realizado e o planeado, apresenta propostas futuras para o desenvolvimento do projeto e termina com umas considerações finais do autor em relação ao projeto.

[Página deixada em branco intencionalmente]

Capítulo 2
Estado da Arte

Neste capítulo, estará disponível primeiramente uma pequena abordagem a várias ferramentas com os mesmos fins pedagógicos da proposta deste documento. De seguida, será exposta uma análise detalhada a ferramentas de construção e simulação de algoritmos. Nesta análise, descrever-se-ão primeiro os pontos fortes e fracos de cada uma das ferramentas. Após essa análise, será exibida uma comparação entre as valências destas. Por fim, serão apresentados alguns *plugins* do Moodle que possuem funcionalidades interessantes para aproveitamento do desenvolvimento deste projeto.

2.1 Identificação dos problemas

De forma a perceber melhor os fatores que influenciam a dificuldade de compreensão dos conceitos básicos lecionados em unidades curriculares de programação, foi-se investigar a literatura. Primeiramente, procurou-se compreender quais os aspetos relevantes para a obtenção de conhecimentos básicos ao nível de programação. Após a reflexão do que foi estudado, conclui-se que a aprendizagem destes conceitos envolve a aquisição de três tipos de conhecimento: sintático, conceptual e estratégico [14]. O conhecimento sintático corresponde ao conhecimento das caraterísticas e regras de sintaxe das linguagens de programação [4]. O conhecimento conceptual remete à compreensão de como os programas são construídos e como os computadores interpretam as instruções dos referidos programas [4]. Este conhecimento é critico para a aprendizagem de programação pois, a sua má compreensão pode conduzir a ideias erradas sobre os conceitos abordados nestas unidades curriculares [4]. Por último, o conhecimento estratégico refere-se à utilização de ambas as faculdades anteriores, em conjunto, para formular uma estratégia para a resolução de um determinado problema [4]. Posteriormente, foi analisada a literatura para recolha de informação em vários estudos, onde os autores procuraram identificar os motivos que dificultam a compreensão dos estudantes relativa aos conceitos básicos de programação, tal como a sua aplicação. Feita a análise desta investigação, foi possível encontrar os seguintes aspetos que contribuem para a realidade referida inicialmente:

- Dificuldades sentidas na aprendizagem dos conhecimentos concetuais – o conhecimento concetual aborda temas abstratos, relativos ao funcionamento do computador. Muito do que acontece num programa não é visível ao programador, o que acaba por causar grandes dificuldades aos estudantes aquando da construção dos programas [6]. Isto acontece, pois, alguns deles acabam por desenvolver interpretações erróneas em relação aos conceitos de programação [4]. Por exemplo, alguns estudantes consideram que numa estrutura condicional ambos os blocos *if* e *else* são executados ou que a ordem na qual se faz a inicialização de uma variável é indiferente [4]. Normalmente, também pensam que um computador funciona como um ser humano e

que consegue executar várias instruções ao mesmo tempo[4]. Esta é outra ideia errada visto que as instruções num programa são executadas sequencialmente pelo computador. Dentro dos conceitos lecionados nestas unidades curriculares, os estudantes revelaram maiores dificuldades na utilização de instruções condicionais e, principalmente, em estruturas de repetição [9]. Em estudantes iniciantes pode-se observar vários tipos de erros relativos na aplicação destes conceitos. Dentro destes erros destacam-se os seguintes: erros relativos à definição das instruções de condição do algoritmo, ao controlo de estruturas de repetição e a utilização de estruturas de dados [9].

- Obstáculos à resolução de problemas – muitos estudantes que ingressam em cursos de programação revelam dificuldades em resolver os problemas de programação que lhes são colocados em contexto de sala de aula [4], [5]. Independentemente de eles possuírem conhecimentos concetuais e sintáticos, estes revelam dificuldades em utilizá-los para a construção de estratégias lógicas de maneira a resolverem problemas de programação [4]. Isto acontece, pois, habitualmente, a aprendizagem dos conteúdos concetuais por parte dos estudantes é feita de forma fragmentada, ou seja, os estudantes são incapazes de interligar os conceitos e aplicá-los na prática [4]. Assim sendo, acabam por não possuir padrões estratégicos para resolverem problemas de programação. A dificuldade na aquisição destes padrões também resulta dos problemas que eles sentem ao fazer *debug* do algoritmo [4].

- Elevada carga de trabalho e esforço cognitivo – a aquisição dos conhecimentos de programação é uma tarefa complexa para muitos dos estudantes. As tarefas habitualmente propostas nas unidades curriculares introdutórias de programação, são caraterizadas por uma carga cognitiva elevada, principalmente, quando são introduzidos diferentes conceitos no mesmo problema [4]. Desta forma, muitos estudantes sentem que para serem bem-sucedidos numa unidade curricular de programação têm de colocar mais horas de trabalho em relação a outras unidades curriculares, o que acaba por prejudicar o seu desempenho nas restantes [10]. Ainda assim, este esforço acrescido, muitas vezes não resulta num bom desempenho o que acaba por desmotivá-los e, em casos extremos, fazer com que não queiram continuar com este percurso académico, levando-os a desistir [10], [11].

- Métodos de estudo adotados – muitos estudantes acabam por se apoiarem no mecanismo da memorização, que se revelou eficaz anteriormente no seu percurso académico, como tentativa de aprender a programar [4], [5]. Utilizando este método, os estudantes desvalorizam a compreensão do que estão a aprender, achando, erroneamente, que lhes basta decorar os procedimentos em questão [5]. A tarefa de assimilação dos conteúdos concetuais estudados é fundamental para que, depois, consigam desenvolver modelos mentais corretos do que acontece num programa e aplicá-los na resolução de problemas [4]. Isto só é possível através de uma abordagem prática, onde estes sejam capazes de criar os seus modelos mentais relativos aos conceitos de programação, através do treino e da sua exposição a vários tipos de cenários [4]–[6]. Só através de uma abordagem prática eles conseguirão aplicar devidamente os conceitos que aprenderam às situações que lhes são colocadas.

- Características dos ambientes de aprendizagem de programação utilizados - normalmente, as unidades curriculares de programação utilizam ambientes de suporte à linguagem de programação demasiado complexos para os conceitos introdutórios que se pretendem explorar[4]. Estes ambientes, por norma, focam-se na melhoria da

experiência profissional do utilizador experiente não sendo orientados para os utilizadores inexperientes [4]. Algumas funcionalidades e alterações ao nível da sintaxe da linguagem podem ser bastantes benéficas para otimizar a experiência de programadores experientes, contudo, podem representar algumas barreiras para estudantes iniciantes [4]. Um exemplo disto é a utilização do operador de adição no Java. Esta linguagem inclui alguns aspetos que poderão ser interpretados com alguma ambiguidade no que respeita a este operador, já que permite a sua utilização para vários tipos de objetos (em estruturas de dados numéricas soma e em *Strings* concatena, por exemplo). Apesar deste tipo de funcionalidades serem benéficas para utilizadores mais experientes, pode ser um quanto confuso para os mais inexperientes [4].

- Baixos níveis de motivação e fraca perceção das capacidades dos estudantes – para que os estudantes alcancem um bom desempenho em qualquer curso superior, é necessário manter os seus níveis de motivação elevados [10]. A primeira impressão de uma aula de programação pode ser bastante negativa, sobretudo quando o estudante se apercebe que há outros estudantes que resolvem os problemas propostos sem qualquer dificuldade [10]. Isto acaba por ser frustrante podendo os estudantes perder a autoconfiança nas suas capacidades. Outro fator que pode influenciar negativamente a sua confiança em ser bem-sucedido nestas unidades curriculares é a perceção sobre a dificuldade em alcançar melhorias nos seus conhecimentos. Este aspeto pode afetar os seus níveis de incentivo para continuar a frequentar estas aulas e trabalhar nos novos exercícios propostos [5], [10].

- Impossibilidade de os professores individualizarem os seus métodos de ensino – devido ao grande número de estudantes presentes nas turmas dos cursos de informática, os professores têm dificuldades em fornecer um acompanhamento individualizado a todos e adaptar as suas explicações às necessidades de aprendizagem de cada um [6]. Visto que as formas de reter e compreender a informação diferem entre cada um deles, é importante fornecer-lhes ferramentas pedagógicas que permitam a construção dos seus modelos mentais e aplicá-los para desenvolver o seu conhecimento [4]. Esta aprendizagem construtiva deve ser de algum modo acompanhada para evitar que desenvolvam ideias incorretas e fiquem com lacunas ao longo da aprendizagem de programação.

2.2 Tecnologias de aprendizagem

Ao longo do tempo, foram-se desenvolvendo várias tecnologias com o intuito de melhorar o ensino e a aprendizagem de programação. Apesar de todas terem este mesmo objetivo, muitas delas utilizam meios distintos para demonstrarem os conceitos de programação a esses estudantes. Dentro destas tecnologias, pode-se caraterizá-las em alguns tipos: micromundos, sistemas de construção e simulação de algoritmos, ambientes integrados, ambientes Web e de colaboração, tutores inteligentes e sistemas de avaliação automática.

Os micromundos são realidades virtuais programáveis que utilizam os movimentos de uma personagem para introduzir os conceitos básicos da programação. Nestes mundos, os utilizadores são introduzidos a ambientes familiares e atrativos, onde podem observar os acontecimentos provocados na personagem através da leitura do programa por eles desenvolvido. Os utilizadores constroem estes programas baseando-se em conceitos de programação simples, utilizando uma mini linguagem que apenas

suporta os conceitos básicos da programação. Dentro destes mundos, destaca-se o Karel the Robot (Pattis, 1981), e o Alice2 (Kelleher et al., 2002).

Os sistemas de construção e simulação de algoritmos são ferramentas que têm ganho popularidade no meio informático. Tal como o nome indica, estes pretendem animar e simular o funcionamento de um algoritmo de programação dando a conhecer aos utilizadores vários conceitos de programação. Estas animações podem ser mais específicas, apenas ao nível da visualização da evolução das estruturas de dados num programa, ou mais abrangentes, através da representação do sequenciamento do programa. Alguns destes ambientes utilizam algoritmos pré-definidos para a animação, outros permitem a construção dos algoritmos, de uma forma mais verbal (por exemplo, utilizando pseudocódigo) ou gráfica (por exemplo, com fluxogramas). Como os ambientes que permitem a construção de algoritmos por parte dos utilizadores são ferramentas do mesmo tipo da proposta deste documento, serão abordados mais à frente neste capítulo.

Os ambientes integrados são outro tipo de ferramenta que tem como objetivo facilitar a aprendizagem dos conceitos de programação e são desenvolvidos para uma determinada linguagem. São ambientes mais simples que os profissionais, graças ao seu caráter pedagógico, e possuem funcionalidades importantes para a compreensão dos conceitos iniciais de programação. Dentro destes, destacam-se o BlueJ (Kölling et al., 2003), que combina edição de código com capacidades de visualização, compilação, debugging e execução, e o JGrasp (Cross e Hendrix, 2006), que consiste num jogo onde dois utilizadores têm como objetivo programar os seus robots para posteriormente se enfrentarem numa batalha. Ambas as ferramentas destinam-se à aprendizagem de Java.

Os ambientes Web e de colaboração consistem em ambientes de aprendizagem desenvolvidos para a world wide web. Estas ferramentas têm como função fornecer ambientes online onde os utilizadores possam desenvolver as suas competências ao nível da programação. Isto pode ser fornecido apresentando o conteúdo de programação de uma forma estruturada, ou através de tutoriais onde os utilizadores podem responder às questões que lhes são colocadas, ou desenvolver código. A avaliação das respostas é feita automaticamente. Dentro deste tipo de ferramentas, destaca-se o AnnAnn (Hooper et al., 2007) e a AulaWeb (García-Beltrán et al., 2005).

Os tutores inteligentes são ambientes que utilizam Inteligência Artificial para promover um ensino personalizado aos utilizadores. Estas ferramentas têm como objetivo identificar problemas nos algoritmos dos programadores através da análise das suas submissões. Dentro deste tipo de ferramentas, destaca-se o PROUST (Johnson e Soloway, 1985) e o LISP Tutor (Anderson e Reiser, 1985).

Os sistemas de avaliação automática são sistemas semelhantes ao mencionado anteriormente. Estes são utilizados para avaliar os conhecimentos que os utilizadores possuem acerca dos conceitos de programação. A capacidade de avaliarem as respostas dos utilizadores utilizando um conjunto de entradas, permite testar e avaliar a qualidade do programa desenvolvido. Dentro deste tipo de ferramentas, destacam-se as opções como o Assyst (Jackson e Usher, 1997), BOSS (Luck e Joy, 1999), Ceilidh (Benford et al., 1993; Foxley, 1999), RoboProf (Daly, 1999), Ceilidh CourseMarker (Higgins et al., 2003), Online Judge (Cheang et al., 2003), HoGG (Morris, 2003) ou o Mooshak.

2.3 Ferramentas de construção e simulação de algoritmos

Tal como foi referido no capítulo introdutório, a proposta deste projeto baseia-se no desenvolvimento de uma ferramenta de construção e simulação de algoritmos similar à pioneira SICAS [1]. Desde então, verificou-se o aparecimento de várias ferramentas semelhantes. De seguida, estão elencadas algumas delas juntamente com a sua respetiva análise, onde são identificados os pontos fortes e fracos. A categorização destas ferramentas feitas nesta análise, advém da utilização dos autores de cada uma. Deste modo, os aspetos referidos a seguir representam opiniões resultantes da experiência dessa utilização, onde se procurou verificar se e como efetuavam a representação dos aspetos mencionados na secção dos objetivos para a construção e simulação dos fluxogramas. Esta análise tem em consideração quatro aspetos relativos à utilização da ferramenta:

- **Construção do fluxograma** – é analisado o modo como é realizada a construção do fluxograma e todas as ações que o utilizador necessita efetuar para concluir o processo.
- **Controlo da animação** – operações que permitem controlar o desenrolar da animação.
- **Representação do fluxo de controlo** - modo como é representado o fluxo de controlo do algoritmo e os dados do algoritmo.
- **Representação dos dados** – modo como são apresentados os dados do algoritmo.

Para além disso, é realizada uma pequena apreciação relativa à *interface* de cada aplicação. Através desta recolha de informação, pretendeu-se retirar aspetos positivos a serem aproveitados neste projeto.

2.3.1 Larp

É uma ferramenta que foi criada em 2004 por Marco Gauvire. Possui um *layout* bem organizado porém não muito apelativo aos olhos do utilizador. A seguir encontram-se apresentadas duas imagens que explicitam as vantagens (fig. 2.1) e desvantagens (fig. 2.2) em relação aos parâmetros de estudo definidos previamente.

Figura 2.1 - Pontos fortes do Larp relativamente aos parâmetros de análise.

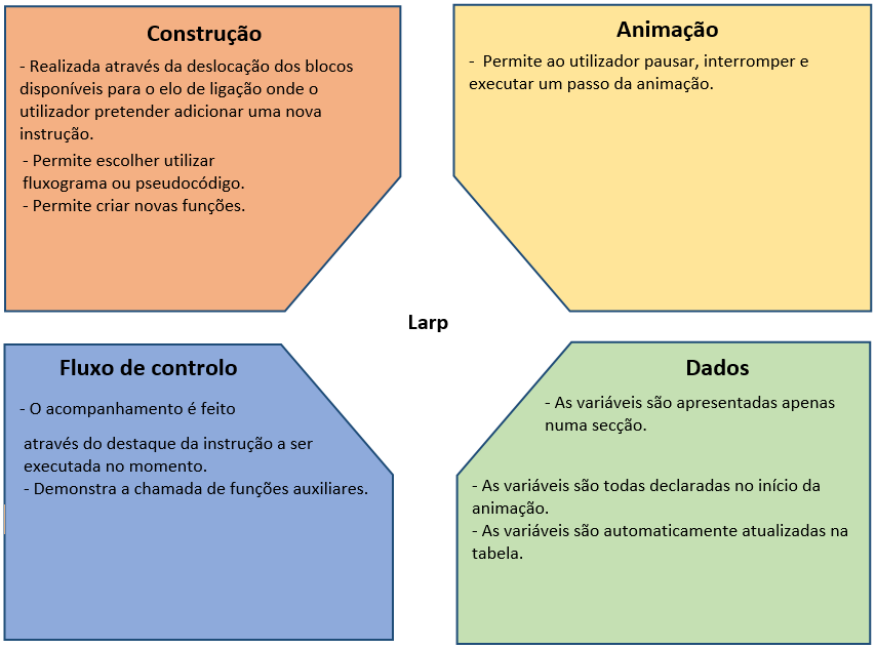
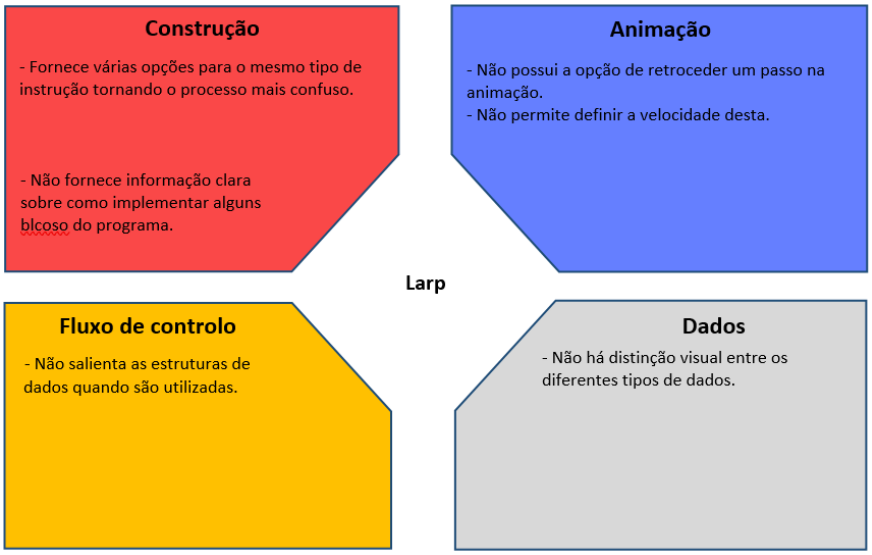


Figura 2.2 - Pontos fracos do Larp relativamente aos parâmetros de análise.



2.3.2 Raptor

Esta ferramenta foi criada por Martin C. Carlise em conjunto com mais três autores, Terry Wilson, Jeff Humphries e Jason Moore, em 2006. Em termos de *interface*, a sua apresentação não é cativante ao nível da conjugação das cores e apresentação dos conteúdos. De seguida, apresentam-se duas imagens que indicam os pontos fortes (fig. 2.3) e fracos (fig. 2.4) desta ferramenta relativamente aos quatro aspetos referidos anteriormente.

Figura 2.3 - Pontos fortes do Raptor relativamente aos parâmetros de análise.

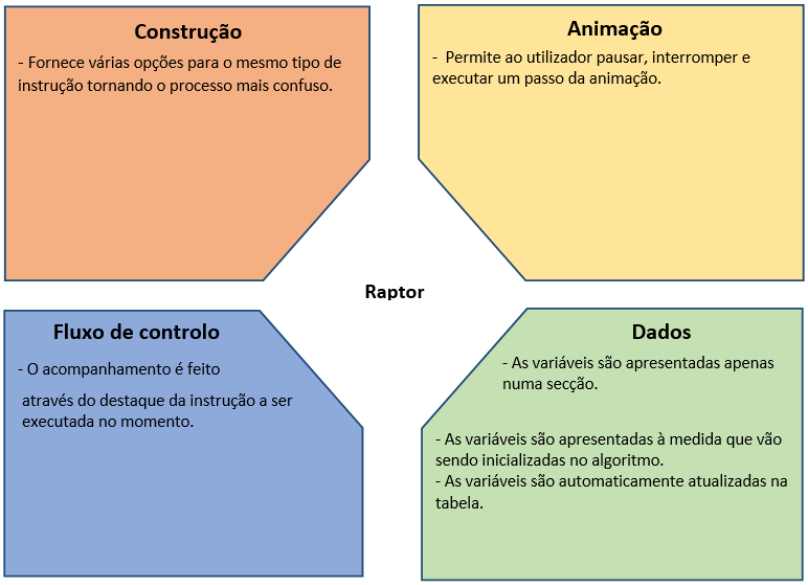


Figura 2.4 - Pontos fracos do Raptor relativamente aos parâmetros de análise.

2.3.3 Visual Logic

Esta ferramenta foi desenvolvida pela PGS Systems em 2010. Possui uma *interface* visualmente pouco apelativa, porém bem estruturada. A seguir encontram-se duas imagens que explicitam as vantagens (fig. 2.5) e desvantagens (fig. 2.6) em relação aos quatro parâmetros de estudo definidos previamente.

Figura 2.5 - Pontos fortes do Visual Logic relativamente aos parâmetros de análise.

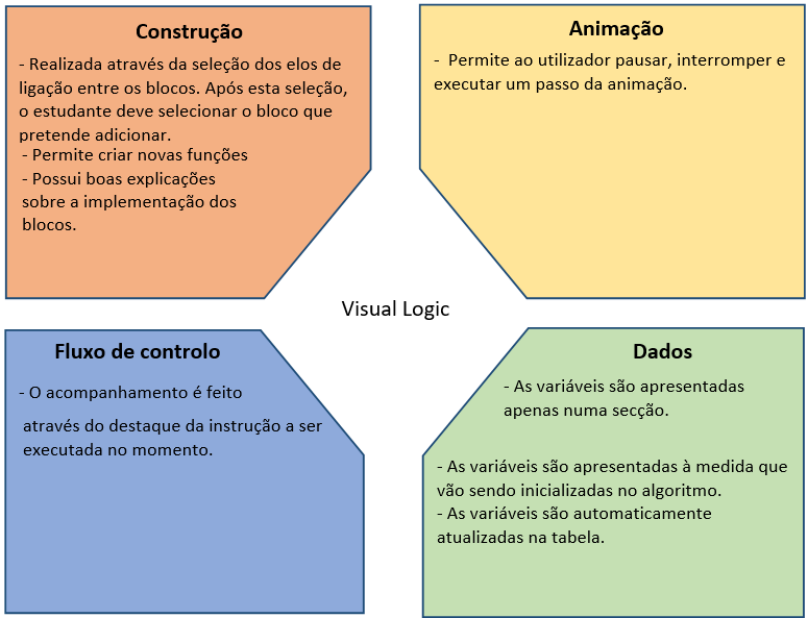
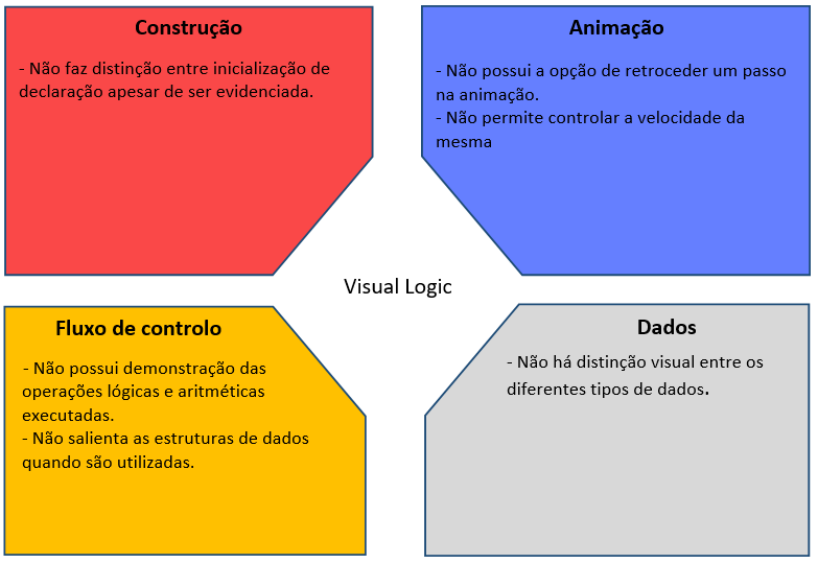


Figura 2.6 - Pontos fracos do Visual Logic relativamente aos parâmetros de análise.



2.3.4 Flowgorithm

O Flowgorithm é uma ferramenta criada em 2014 pela Universidade de Sacramento, nos EUA. Esta possui uma *interface* bem organizada e apelativa graças a uma boa disposição das componentes que apresenta ao utilizador e à boa conjugação de cores que possui nos seus menus. Para além disso, essa conjugação é realizada de forma a associar alguns conceitos ao nível da programação. Por exemplo, agrupamento de tipos de instrução. De seguida, encontram-se duas imagens que apresentam os pontos fortes (fig. 2.7) e fracos (fig. 2.8) desta ferramenta relativamente aos quatro aspetos referidos anteriormente.

Figura 2.7 - Pontos fortes do Flowgorithm relativamente aos parâmetros de análise.

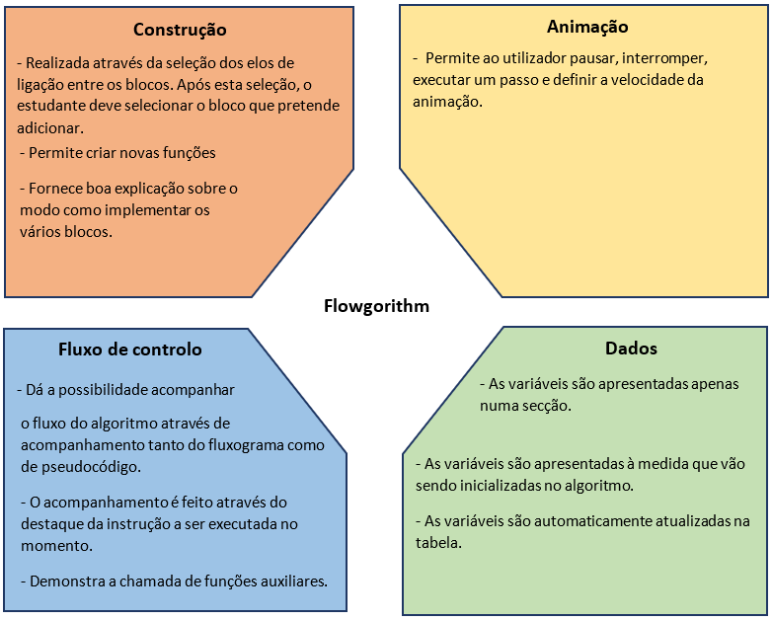
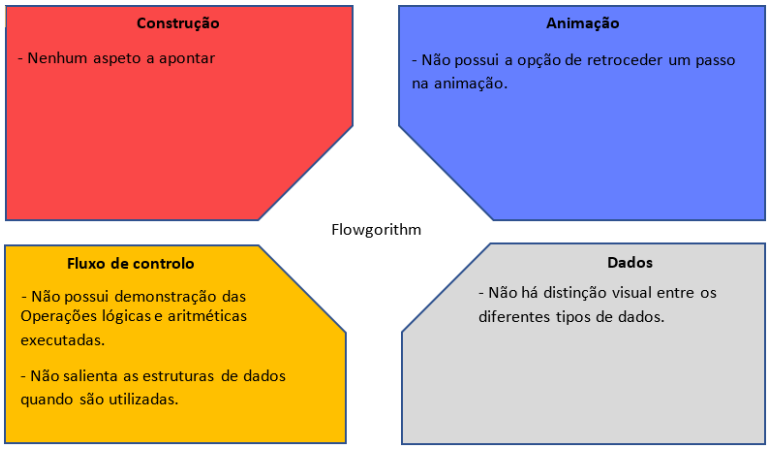


Figura 2.8 - Pontos fracos do Flowgorithm relativamente aos parâmetros de análise.



2.3.4 Jelliot 3.0

Esta ferramenta foi criada em 2014 por Mordechai (Moti) Ben-Ari e Erkki Sutinen, jutamente com um grupo de colaboradores. Possui uma *interface* bem estruturada e apelativa ao utilizador. As seções disponíveis no ecrã estão bem divididas bem como o seu propósito. De seguida, apresentam-se duas imagens que indiquem os pontos fortes (fig. 2.9) e fracos (fig. 2.10) desta ferramenta relativamente aos quatro aspetos referidos anteriormente. Como a construção dos algoritmos para a animação não se enquadra nas pretensões para o projeto, essa vertente da ferramenta não será abordada na análise.

Figura 2.9 - Pontos fortes do Jelliot 3.0 relativamente aos parâmetros de análise.

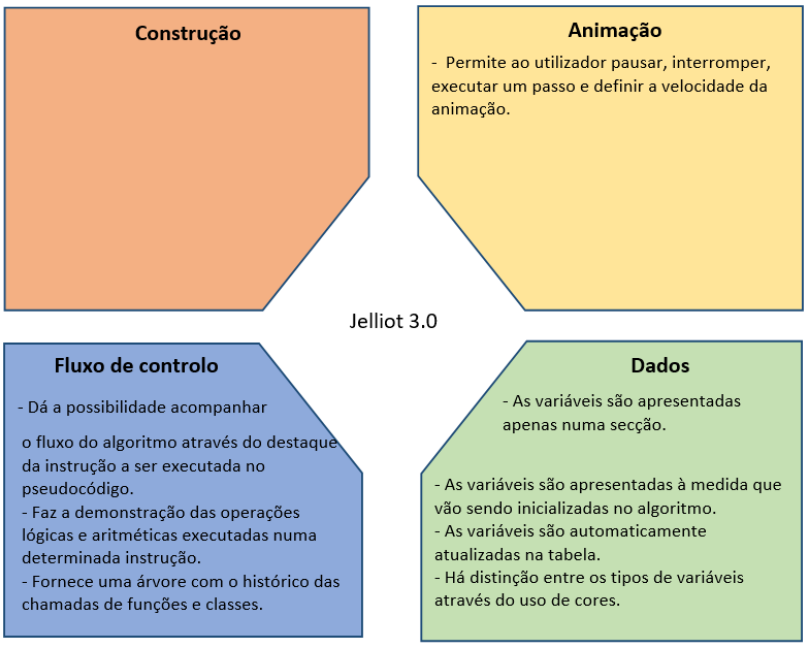
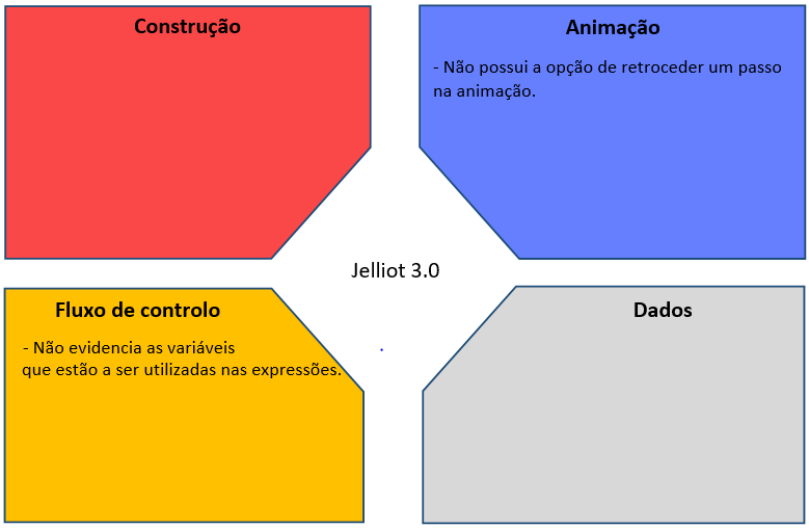


Figura 2.10 - Pontos fracos do Jelliot 3.0 relativamente aos parâmetros de análise.



2.3.5 Comparação das ferramentas

A partir da análise apresentada anteriormente, verificou-se que, num âmbito geral, todas as ferramentas conseguem representar bem cada um destes quatro aspetos considerados na análise. Contudo, nenhuma delas consegue abranger todas (Tabela 2.1). No que toca à *interface* e ao modo de construção do fluxograma, o Flowgorithm destacou-se em relação às demais. Esta ferramenta consegue promover uma metodologia intuitiva na construção do fluxograma. É possível distinguir os tipos de instruções que existem dado que estes são apresentados ao utilizador organizados em grupos, onde cada grupo engloba os blocos que representam o conceito do grupo onde

estão inseridos. Por exemplo, o grupo "Ciclos de Repetição" possui a representação dos blocos para as instruções *for*, *while e do while*. A categorização destes grupos é feita, tanto ao nível do agrupamento dos tipos de instrução como através da utilização de cores distintas para cada um deles. Contudo, uma utilização excessiva de cores pode causar uma sobrecarga visual ao utilizador indesejada. A implementação das operações também é bastante intuitiva e fornece todas as ajudas necessárias para o utilizador saber como definir determinada operação, p.e., como criar um *array* de inteiros ou definir uma estrutura de repetição. O controlo da animação e o resto do *layout* também estava bem acessível para o utilizador, já que lhe permitia controlar a velocidade de animação, interrompê-la sempre que quisesse e recomeçar no estado em que estava e executar a instrução passo-a-passo. O único aspeto negativo que se pode apontar ao Flowgorithm é o facto de não permitir que o utilizador retroceda na animação.

Na representação do fluxo do programa, o Larp e o Jelliot3.0 foram os que apresentaram melhor esta funcionalidade já que, nesta parte, também permitem visualizar todas as operações que ocorrem no programa. Por fim, na representação dos dados, o Jelliot3.0 foi o melhor. Esta possibilitava distinguir visualmente os tipos de objetos que estão a ser utilizados no ecrã. Isto permite ao estudante observar as diferenças entre eles e associá-los a outros conceitos como, por exemplo, o modo como elas são geridos no sistema.

Concluindo, considera-se que, o desenvolvimento de uma ferramenta diferenciadora das acima referidas passa por reunir os aspetos mais positivos que cada uma tem na representação destas quatro funcionalidades e utilizá-las como exemplo a seguir para a construção da ferramenta proposta neste documento.

Tabela 2.1 - Tabela de comparação das características procuradas entre as ferramentas analisadas.

		Larp	Raptor	Visual Logic	Flowgorithm	Jelliot 3.0
Construção do fluxograma	Drag and drop	✓	✓			--
	Selecionar o elemento de inserção			✓	✓	--
Controlo da animação	Definir velocidade				✓	✓
	Recuar um passo					
	Avançar um passo	✓	✓	✓	✓	✓
	Pausar		✓		✓	✓
	Recomeçar	✓	✓	✓	✓	✓
Fluxo de controlo	Execução do fluxograma	✓	✓	✓	✓	✓
	Operações lógicas e aritméticas	✓				✓
	Destacar variáveis na sua utilização					✓
Dados	Atualização automática	✓	✓	✓	✓	✓
	Diferenciação dos tipos de variável					✓

2.4 *Plugins* do Moodle

Tal como foi referido anteriormente, o Moodle é um sistema de gestão de aprendizagem permitindo aos seus utilizadores personalizar os seus ambientes de ensino/aprendizagem de acordo com as suas preferências. Integra, adicionalmente, vários *plugins*, elaborados pelo Moodle e a sua comunidade de programadores, que potencializam as funcionalidades dos diversos ambientes. De momento, importa referir alguns *plugins* que possuem funcionalidades interessantes e que poderiam ser aproveitadas para integração neste projeto.

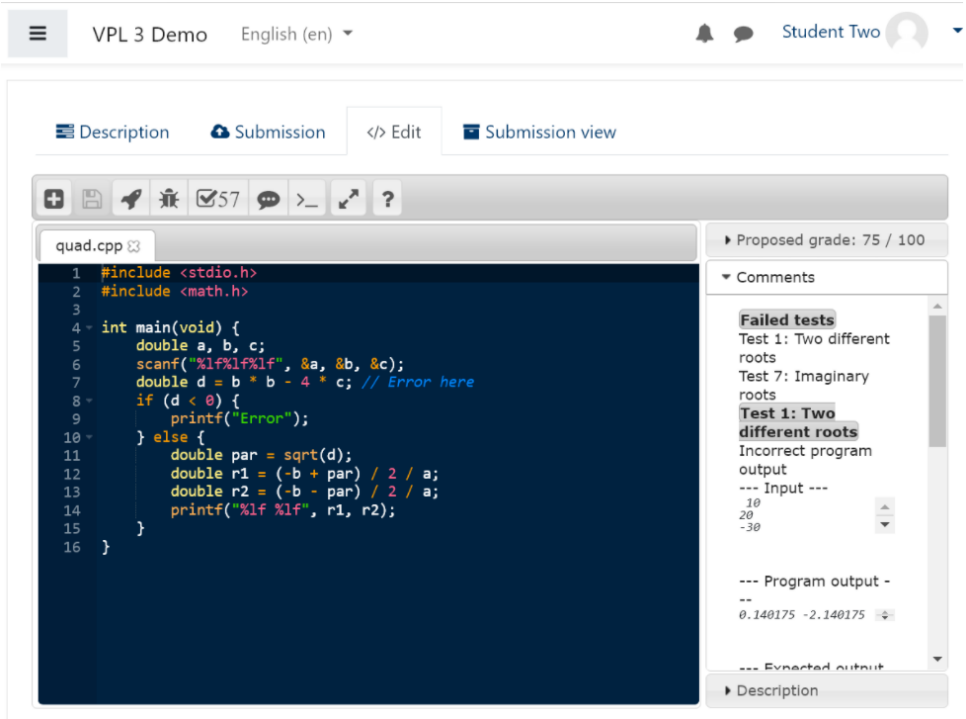
2.4.1 VPL

O VPL é um *plugin* de atividade que possui algumas semelhanças com os objetivos que se pretendem atingir com este projeto, possibilitando aos docentes criar exercícios para os estudantes resolverem. Esta definição é feita segundo três parâmetros (fig. 2.11):

- **Descrição** – descrição do exercício a ser criado;
- **Valores de entrada** – valores que o a solução do exercício irá receber;
- **Valores de saída** – valores que devem ser retornados pela solução face aos valores de entrada recebidos.

Após a criação dos exercícios, os estudantes poderão submeter as suas soluções. Estas podem ser posteriormente visualizadas tanto pelo estudante que as submeteu como pelo professor respetivo.

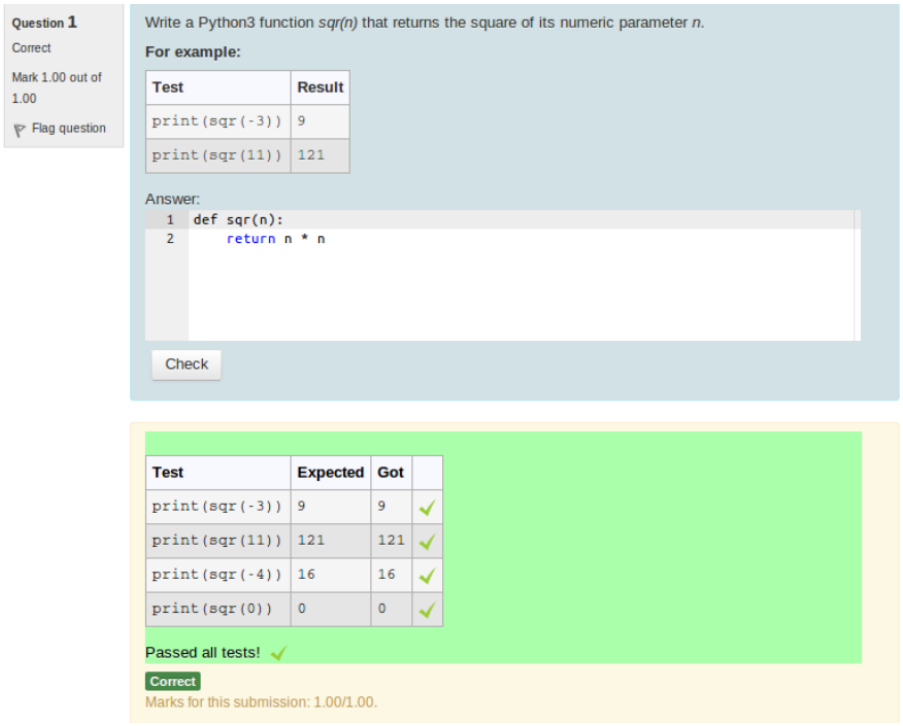
Figura 2.11 - Exemplo de uma atividade do VPL.



2.4.2 Code Runner

É um *plugin* que avalia as resoluções dos estudantes. Esta avaliação é realizada a partir de uma série de testes à mesma resolução. Graças a esta caraterística, é possível avaliar de forma mais precisa, a correção de um programa. Assim, é possível dar a conhecer aos docentes as falhas concretas nas soluções dos seus estudantes e auxiliá-los melhor na correção dessas dificuldades (Fig. 2.12).

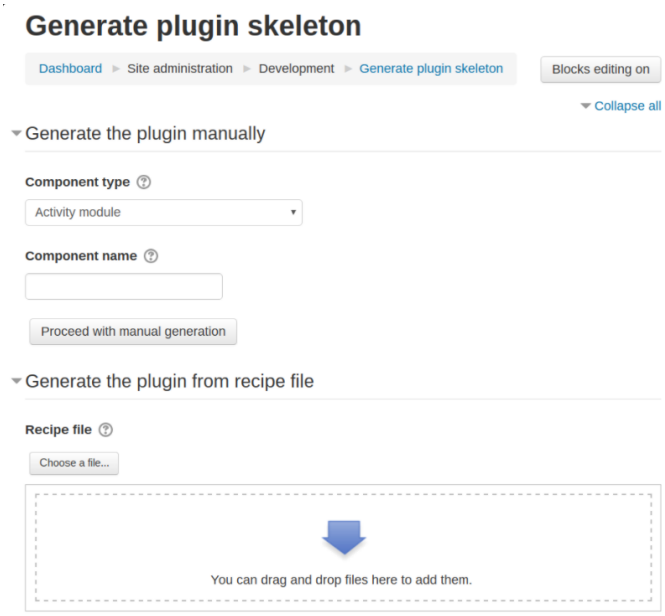
Figura 2.12 - Exemplo de como opera o Code Runner.



2.4.3 *Plugin* Skeleton Generator

Este *plugin* fornece aos desenvolvedores a possibilidade de gerar a estrutura base do código de um determinado *plugin* (Fig. 2.13). Desta forma, a sua utilização pode ser benéfica numa fase inicial da implementação do *plugin.*

Figura 2.13 - Opções existentes para gerar a estrutura de um *plugin* através *Plugin* Skeleton Generator.



2.5 Considerações finais

A partir da análise das ferramentas presentes neste capítulo definiram-se os seguintes aspetos relativamente à ferramenta deste projeto:

- O método de construção do fluxograma irá ser através de eventos *drag and drop;*
- A organização e o aspeto visual da *interface* deverão ser atrativas e intuitivas para o estudante;
- Deverá reportar erros ao estudante durante o processo de construção e animação do fluxograma;
- Deverá ser capaz de representar quatro aspetos durante a animação:
    - Representação do fluxo de controlo – destaque dos elementos a serem executados durante a animação do algoritmo;
    - Representação das operações aritméticas – destaque sequencial das operações executadas numa determinada expressão;
    - Representação dos dados utilizados no algoritmo – destaque na atualização e utilização dos dados presentes no algoritmo;

No que se refere ao Moodle, considerou-se que a integração desta ferramenta na plataforma poderia trazer um conjunto de mais valias importantes no alcance do objetivo do projeto. Por exemplo, o plugin Code Runner poderia ser uma complementação bastante interessante para a ferramenta já que permitiria fornecer uma avaliação segundo vários parâmetros do algoritmo gerado pelo fluxograma construído através da ferramenta. Assim, poderia ser possível determinar o nível de correção do fluxograma gerado pelos estudantes. Para além disso, o VPL fornece uma boa visão de como se poderia estruturar a atividade do *plugin* que irá conter a ferramenta de modo a integrar o professor no ambiente para que este possa identificar as dificuldades sentidas pelos estudantes na criação dos algoritmos e auxiliá-los no

processo de obtenção dos conceitos de programação. A possibilidade de introduzir dados estatísticos nestas atividades é mais uma forma de ajudar a esse processo.

[Página deixada em branco intencionalmente]

Capítulo 3
Requisitos

Após a obtenção dos conhecimentos referidos nos capítulos um e dois relativos à ferramenta de construção e simulação de algoritmos, seguir-se-á a apresentação das características consideradas fundamentais relativamente à ferramenta proposta. Neste capítulo, serão apontados vários aspetos relativos ao procedimento da definição dos requisitos a considerar. Primeiramente, efetuar-se-á a identificação dos atores do sistema. Após esta identificação, seguir-se-á a identificação dos requisitos funcionais e não funcionais, terminando com a descrição e apresentação da *interface* da ferramenta proposta.

3.1 Entidades

Através da leitura do capítulo introdutório deste documento, conseguem-se identificar três entidades intervenientes no projeto deste documento:

- Estudantes – entidade que vai usufruir da ferramenta para aprender e aplicar os conhecimentos básicos ao nível da programação.
- Professores – entidade que vai usufruir da ferramenta para criar exercícios a serem resolvidos pelos estudantes.
- Moodle – entidade que irá fornecer o software para que as duas primeiras executem as suas atividades.

3.2 Requisitos funcionais

Após a definição dos objetivos do projeto, foi necessário delinear os requisitos funcionais necessários para alcançar os propósitos determinados. Desta forma, encontram-se de seguida expostos os casos de uso para as funcionalidades definidas. Através destes diagramas, procurou-se não só clarificar todas as capacidades do sistema de uma forma mais simplificada, como também o modo como as entidades interagem no mesmo (fig. 3.1). A apresentação detalhada de todos os requisitos funcionais encontra-se em apêndice do documento.

Recapitulando os conteúdos abordados na seção “Objetivos” do capítulo introdutório deste documento, foi possível identificar três espetros de funcionalidades:

- **Construção do fluxograma;**
- **Animação do fluxograma;**
- **Gestão da atividade do *plugin*.**

Figura 3.1 - Componentes que constituem um diagrama de caso de uso.

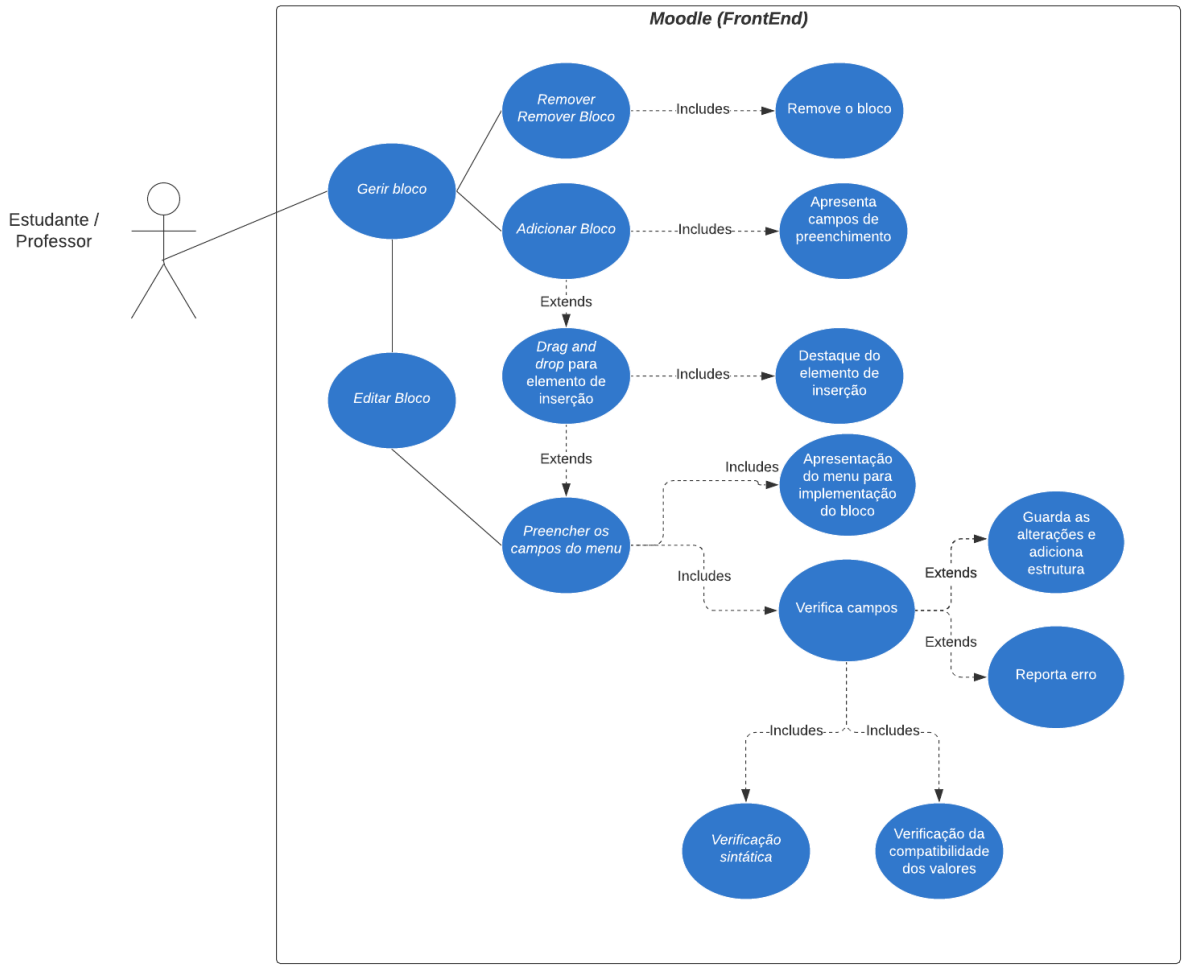
3.2.1 Construção do fluxograma

A construção do fluxograma é uma das partes essenciais do projeto. Como tal, foram necessárias definir várias funcionalidades que permitissem ao estudante construir um fluxograma de acordo com os aspetos pedagógicos que se pretenderam alcançar. Desta forma, esta atividade engloba uma série de operações disponíveis nas imagens abaixo apresentadas (fig. 3.2). Na imagem que se segue, a palavra "Gerir" presente em alguns casos de usos, refere-se às ações criar, remover e editar.

Figura 3.2 - Diagrama de caso de uso para as funcionalidades do estudante na construção do fluxograma.

Para a construção de um bloco ao fluxograma foi importante idealizar este processo de modo a ser intuitivo para os estudantes, possibilitando que estes os construíssem sem possuírem erros sintáticos. Desta forma, foi necessário definir métodos que tratassem de realizar uma verificação sintática das expressões definidas nos blocos e uma verificação relativamente à compatibilidade dos valores utilizados nas mesmas. Na figura 3.3, encontra-se apresentado um esquema que ilustra o processo de gestão dos blocos e as atividades necessárias de executar de modo a atingir o ponto de minimização dos erros sintáticos.

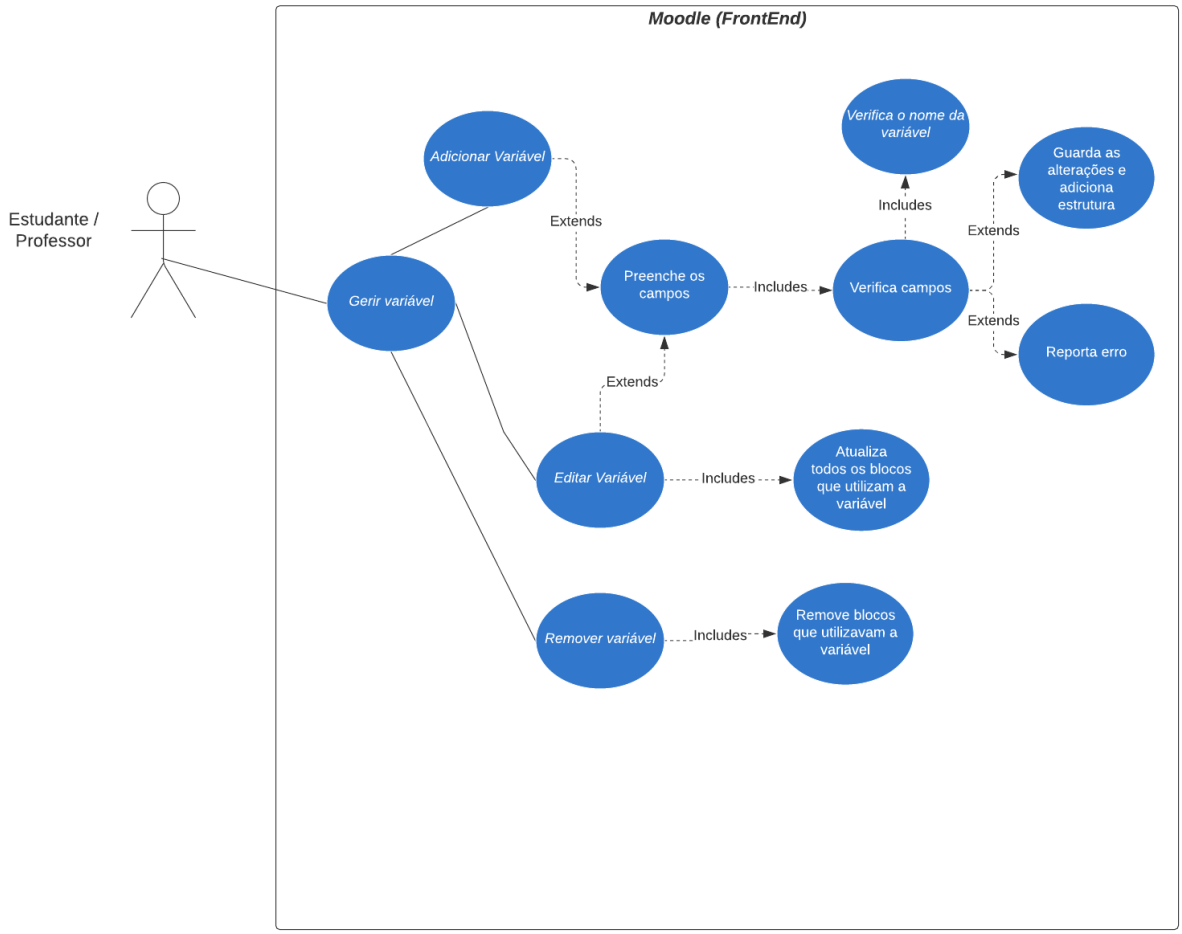
Figura 3.3 - Diagrama de uso para a gestão dos blocos do fluxograma.



Desta forma, considerou-se que estes erros deixariam de ser um fator de desmotivação para o estudante na aprendizagem e aplicação dos conceitos.

Para garantir que os erros sintáticos não se alastrem pelo fluxograma, também foi necessário realizar algumas verificações no processo de gestão das variáveis. Estas verificações ocorrem no quando o estudante pretende editar ou remover uma variável. Isto torna-se necessário pois existe a possibilidade de este querer realizar estas operações numa variável que esteja a ser utilizada num dos blocos que já adicionou ao fluxograma. Na figura 3.5, encontra-se disponível o digrama de caso de uso que ilustra a gestão de variáveis e as verificações realizadas nas várias etapas do processo.

Figura 3.4 - Diagrama de uso para a gestão das variáveis presentes na ferramenta.

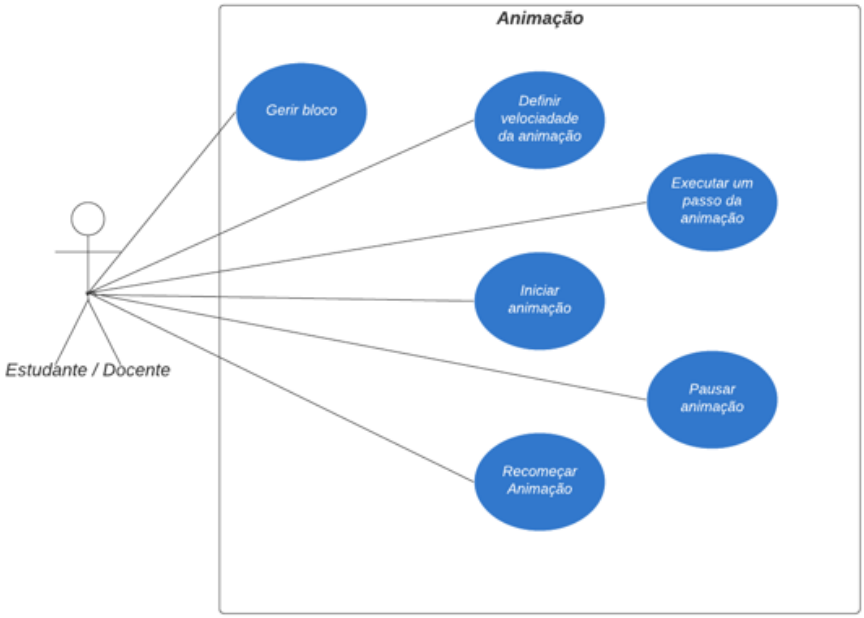


Tal como os estudantes, os docentes também devem usufruir do acesso à ferramenta caso pretendam realizar alguma demonstração num contexto de sala de aula.

3.2.2 Animação do fluxograma

Após a construção do fluxograma os atores têm a possibilidade de simular o algoritmo que criaram. Depois de serem encaminhados para o menu destinado a este fim, devem ter acesso a uma série de operações de forma a controlar o desenrolar da animação (fig. 3.5).

Figura 3.5 - Diagrama de caso de uso para as funcionalidades de ambos os atores na animação do fluxograma.

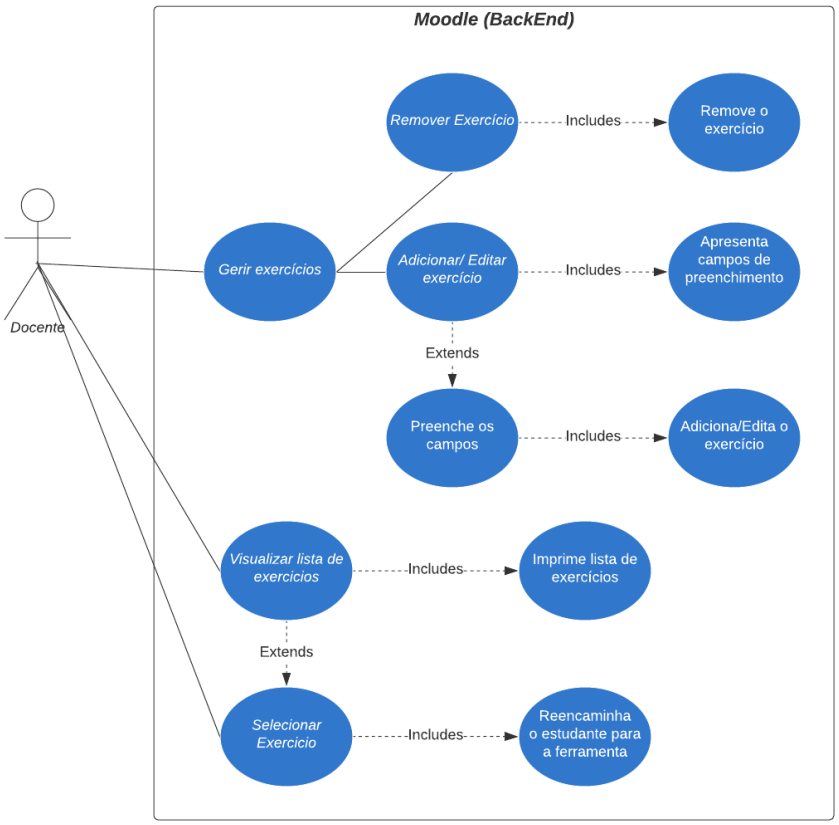


Para além de se fornecer aos estudantes a capacidade de visualizarem os vários passos envolvidos na execução dos seus programas, considerou-se importante dar-lhes a oportunidade de controlarem essa animação. Através das funcionalidades apresentadas anteriormente, acredita-se que estes as possam utilizar para benefício próprio. Graças a elas, os estudantes poderão adaptar este processo de acordo com as suas necessidades como, por exemplo, diminuir a velocidade de execução numa instrução que não compreenda muito bem.

3.3.3 Gestão da atividade do *plugin*

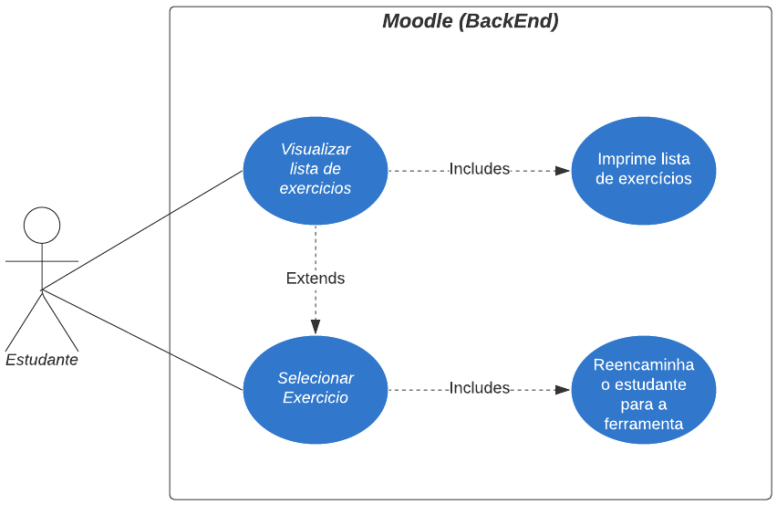
Este grupo de funcionalidades destina-se à realização da gestão da informação do sistema, permitindo que os atores acedam às atividades acima apresentadas. Dadas as caraterísticas das funcionalidades, este grupo foca-se mais na experiência do docente na atividade, ao contrário das anteriores cujo foco era o estudante. Nesta secção encontram-se as funcionalidades que permitem ao docente gerir as atividades criando os exercícios e disponibilizando-os ao estudante(fig. 3.6).

Figura 3.6 - Diagrama de casos de uso para as funcionalidades permitidas ao docente na parte da gestão da atividade.



Neste tipo de funcionalidades, o estudante apenas deve ter acesso à lista de exercícios criados pelo professor e permitir que este selecione um dos exercícios da lista (fig. 3.7).

Figura 19 - Diagrama de casos de uso para as funcionalidades permitidas ao estudante na parte da gestão da atividade.

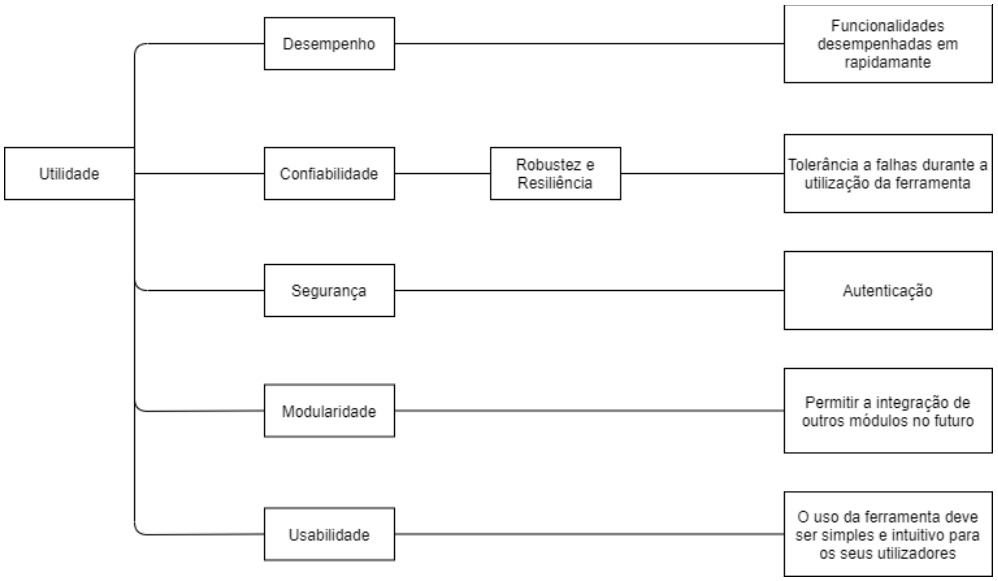


3.3 Requisitos não funcionais

Ao contrário dos requisitos abordados anteriormente, estes não fornecem funcionalidades ao sistema, mas sim o modo como estas funcionalidades são executadas. Assim sendo, os requisitos não funcionais são bastantes importantes na avaliação de um *software* e na estruturação da arquitetura do sistema. Dada a exigência do público-alvo desta ferramenta, é importante que se cumpram vários requisitos deste tipo com o objetivo de garantir que os estudantes se envolvam com a ferramenta e se sintam motivados a utilizá-la. De seguida encontram-se enumerados os requisitos que se consideram importantes agregar à ferramenta proposta deste documento (fig. 3.8):

- **Desempenho** – é importante que todas as atividades envolvidas nesta ferramenta aconteçam de uma forma fluída de modo a garantir que o estudante não perca o interesse na utilização da ferramenta. Assim, todas as funcionalidades associadas à construção do fluxograma devem ser executadas rapidamente;
- **Usabilidade** – visto que o principal propósito desta ferramenta é construir e simular fluxogramas, este requisito é de extrema importância. Tal como o anterior, é importante certificar que o estudante se sente entusiasmado para usar a ferramenta. Desta forma, esta deve ser capaz de possibilitar uma navegação e execução intuitiva de todas as funcionalidades do sistema, principalmente as de construção e simulação de algoritmos. Uma forma de melhorar a experiência de utilização é a inclusão de atalhos para certas funcionalidades do sistema, por exemplo, adicionar/remover um bloco, ou para as opções de controlo da animação;
- **Confiabilidade** – para que os estudantes queiram usufruir desta ferramenta, é necessário garantir que têm confiança que esta, em qualquer situação, vai funcionar devidamente. Este requisito é atingido a partir do preenchimento de outros, nomeadamente disponibilidade, tolerância e recuperação de falhas;
- **Robustez e Resiliência** – a ferramenta deve ser capaz de lidar e recuperar o sistema de qualquer erro que possa pôr em causa o seu funcionamento normal;
- **Segurança** – é essencial garantir que nenhuma entidade externa aceda às páginas geradas pelo *plugin*. Desta forma, deve ser implementado um mecanismo de autenticação quando se acede a uma página do *plugin* para realizar essa verificação;
- **Modularidade** – esta ferramenta deve ser construída de forma a ser possível adicionar novas funcionalidades no futuro. Como tal, as funcionalidades deste sistema devem estar agrupadas em módulos. Este tipo de estruturação coincide com a arquitetura do Moodle que vai ser explicada no capítulo 4.

Figura 3.8 - Árvore de utilidade



3.4 Restrições técnicas

Devido ao facto de a ferramenta ter de ser desenvolvida de modo a ser compatível com o Moodle, o leque de opções técnicas que se podem tomar está condicionado pela arquitetura da plataforma, que exige aos seus desenvolvedores uma série de requisitos que irão ser abordados no capítulo seguinte.

Capítulo 4
Ferramenta de construção e simulação de algoritmos

Neste capítulo será feita a apresentação da ferramenta desenvolvida. Neste processo, serão apresentadas todas as componentes que a constituem juntamente com os motivos pedagógicos que levaram à sua implementação.

4.1 Descrição geral da ferramenta

Dado o cariz do projeto em causa, a parte mais relevante do mesmo remeteu à criação da ferramenta de simulação e construção de algoritmos. O desenvolvimento desta ferramenta, apesar de poder ser utilizada pelo docente para realizar uma demonstração, foca-se essencialmente na sua utilização pelo estudante. Desta forma, esta deve ser estruturada de modo a abranger uma série de aspetos pedagógicos relevantes para melhorar a sua experiência de utilização, não apenas no âmbito genérico da sua utilização, mas essencialmente na compreensão e aplicação dos conceitos básicos de programação.

Tal como o nome do capítulo explicita, esta ferramenta possui dois grandes propósitos:

- Permitir a construção de fluxogramas que representem algoritmos de programação;
- Permitir visualizar uma animação da computação das instruções que compõem o fluxograma criado pelo utilizador.

De forma a alcançar os objetivos referidos anteriormente foi necessário conjugar uma série de aspetos relativos à ferramenta.

O primeiro aspeto remete para a apresentação da *interface*, tanto a nível visual como a nível organizacional. Relativamente ao caráter visual da *interface*, procurou-se utilizar cores suaves de modo a evitar ruído visual e a causar uma boa primeira impressão ao utilizador e, consequentemente, promover uma melhor concentração e foco na sua utilização. Também se procurou utilizar uma série de ícones visualmente representativos da ação que executam. Ao nível organizacional, ambas as *interface*s de construção e simulação foram estruturadas de forma considerada intuitiva para o estudante ser capaz de identificar o propósito de cada secção e idealizar o tipo de operações que podem ser efetuadas em cada uma delas numa primeira interação. Desta forma, considerou-se que a apresentação de menus, durante a fase de implementação, devidamente estruturados e legendados seria uma boa estratégia para atingir esse propósito. Para além disso, implementaram-se estes menus de forma a possibilitar ao estudante posicioná-los no ecrã conforme desejar.

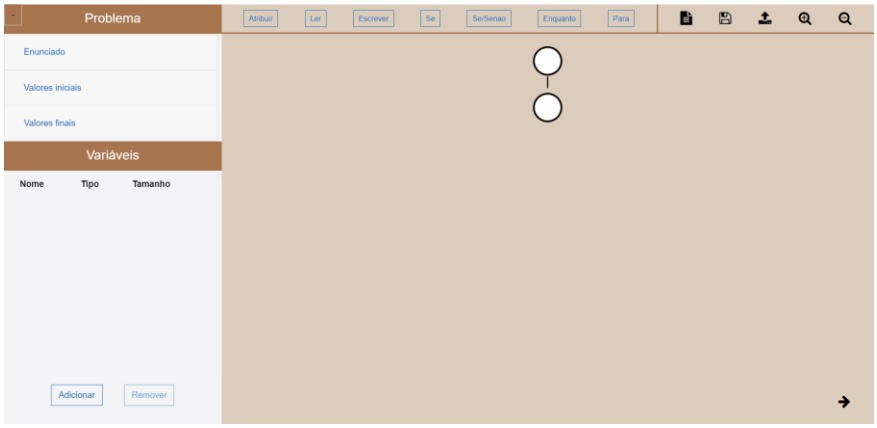
O segundo aspeto está associado aos procedimentos que este terá que executar de forma a construir um algoritmo para, posteriormente, animá-lo. Para promover uma experiência de aprendizagem profícua para o estudante foi necessário implementar o sistema de forma a assisti-lo ao longo do seu percurso pela ferramenta e guiá-lo nesta utilização. Para além disso, foi também importante fornecer-lhes menus de ajuda para as diversas atividades e assim o auxiliar na compreensão da operação que pretende executar tal como os requisitos a serem cumpridos para a completar corretamente. A validação desses requisitos é feita pelo sistema e caso esta não se verifique, o sistema está encarregue de lhe emitir uma mensagem de erro clara relativamente ao requisito que não foi cumprido. Desta forma, foi possível minimizar o número de erros sintáticos e a acumulação dos mesmos que pudessem afetar a animação do fluxograma. Assim, retira-se a necessidade ao estudante de estar a fazer *debug* dos seus erros sintáticos durante o processo de animação e permitir que este se foque no processo de obtenção e construção de modelos lógicos para a resolução que é colocada. Para além disso, considerou-se importante, durante esta fase, reduzir o número de passos necessários para a realização de cada tarefa. Desta forma, o estudante necessita de realizar menos operações e o sistema acaba por o guiar no procedimento de implementação.

4.2 Menu de construção e suas atividades

Na figura 4.1, é possível observa o menu de construção da ferramenta. Como se pode verificar, de modo a atingir os objetivos acima referidos, procurou-se dividir esta *interface* em três partes:

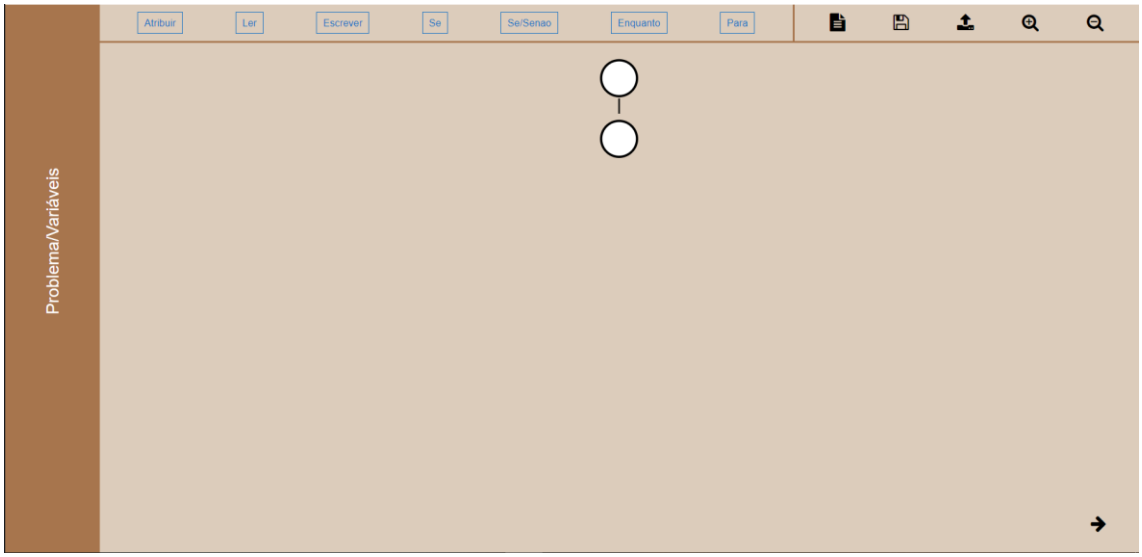
- Uma parte, apresenta as componentes que formam o problema apresentado ao estudante. Nesta secção, este poderá conhecer a descrição do problema que precisa de resolver juntamente com alguns cenários iniciais de implementação do fluxograma (por exemplo: criar variável a e atribuir-lhe o valor 10 inicialmente) e os respetivos valores finais que devem ser produzidos pelo algoritmo a partir desse cenário. Desta forma, procurou-se fornecer um guia adicional para além do enunciado na tentativa de dar a entender ao estudante o propósito do enunciado que lhe foi proposto de uma forma mais clara e dar-lhe uma visão do algoritmo que este tem de criar de forma a obter os mesmos resultados que lhe são apresentados. Para além disso, acreditou-se que exibindo vários casos de teste seria uma forma do sistema apresentar formas para se adaptar aos diferentes estilos de raciocínio dos estudantes.
- Outra parte, apresenta a zona onde será realizada a gestão das variáveis que irão existir no seu algoritmo. Nesta secção o estudante poderá proceder à criação de variáveis a serem utilizadas na construção do fluxograma, editá-las ou removê-las conforme as suas pretensões;
- Uma zona onde o estudante poderá construir o seu fluxograma.

Figura 4.1 - *Interface* do menu de construção da ferramenta.



Analisando a estrutura do menu de construção, é visível a diferença de espaço entre as várias secções. A atribuição de uma maior área à secção de construção do fluxograma, para além de ser necessário graças ao teor da operação e ao espaço necessário para a sua apresentação, permitiu ilustrar que o propósito principal desta parte se destina à construção do fluxograma e que as restantes secções servem de auxílio para a realização desta tarefa. Para acentuar este aspeto e para aumentar o espaço de criação do fluxograma, permitiu-se ocultar a secção lateral do menu de construção (fig. 4.2).

Figura 4.2 - Visão da *interface* com a barra lateral minimizada.



Para realizar o processo de construção do fluxograma, são disponibilizados botões na barra de controlos situada no topo da figura 4.2. Estes botões encontram-se devidamente legendados relativamente ao tipo de instrução que irão representar no fluxograma. Após a identificação do tipo de instrução que se pretende inserir, o estudante deve arrastar o bloco para o elo de ligação entre os blocos onde pretende inserir o novo bloco. Optou-se por esta abordagem por ser uma forma de construção intuitiva e generalizada em várias aplicações disponíveis na *web*. Contudo, para garantir que o processo não fosse disruptivo e se adaptasse às condições físicas de todos os

estudantes, foi necessário disponibilizar uma área de inserção aumentada que não implicasse o direcionamento muito preciso para o elo de ligação. Efetuado o direcionamento de um bloco a uma região de aceitação, o estudante apenas necessita de largar o rato. O utilizador saberá que se encontra numa região de aceitação para a inserção do bloco quando o elo de ligação for destacado. Depois desta ação o sistema apresenta automaticamente o menu de implementação do bloco que inseriu. Desta forma, o sistema conduz o estudante nos processos necessários para implementar e reduz o número de passos necessários para esse processo. Para manter a coerência no fluxograma, caso o estudante decida cancelar esta implementação o sistema remove automaticamente o bloco previamente adicionado.

Por fim, o estudante terá acesso a alguns atalhos relativamente à construção do fluxograma. Este poderá copiar, colar e eliminar blocos do fluxograma utilizando teclas de atalho do seu computador. Para copiar um bloco, o estudante deverá selecionar o bloco que pretende copiar. Após essa seleção, o sistema destaca o bloco selecionado pelo estudante. Depois, para proceder à cópia do bloco, o estudante deverá pressionar as habituais combinações de teclas “Ctrl + C” e o sistema procede à criação de uma cópia do bloco selecionado. Para colar a cópia gerada, o estudante deve selecionar o bloco onde quer que a cópia seja inserida. Após essa seleção, o estudante deve pressionar as teclas “Ctrl + V” para o sistema adicionar o bloco a seguir ao bloco selecionado. Para remover um bloco, o estudante deverá selecionar um bloco do seu fluxograma e depois pressionar as teclas “Delete” ou “BackSpace” presentes no teclado. Desta operação, resulta a remoção do bloco selecionado.

Para a representação das instruções no fluxograma, optou-se por utilizar as estruturas padrão definidas para este tipo de representação.

4.2.1 Blocos da ferramenta

Tendo em vista os objetivos da ferramenta, implementaram-se vários tipos de componentes que representassem os tipos de instrução existentes num algoritmo básico de programação. Como tal, definiram-se blocos para os seguintes conceitos:

- Variáveis
- Entrada e saída de dados
- Estruturas de seleção
- Estruturas de repetição

Estas duas últimas conferem ao estudante a possibilidade de controlar o fluxo dos seus algoritmos.

De seguida encontram-se apresentadas os blocos que representam cada um dos conceitos referidos.

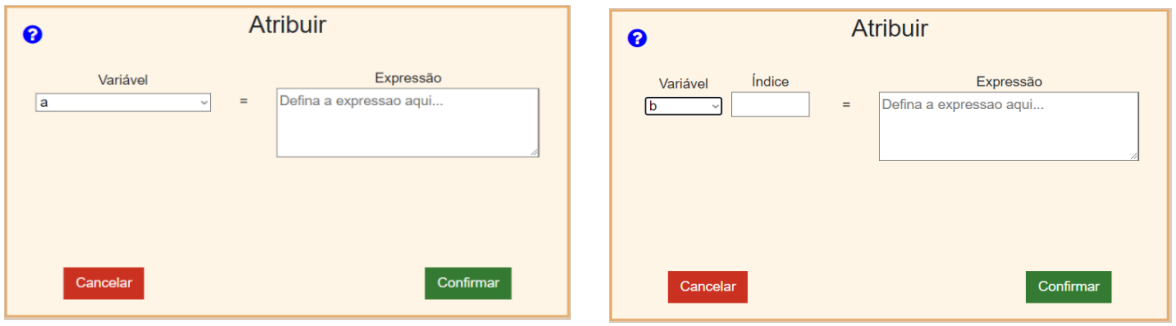
4.2.1.1 Introdução de variáveis

4.2.1.1.1 Atribuir

Esta instrução permite fazer uma atribuição a uma variável do programa. De forma a implementá-la, o estudante deve selecionar uma variável da lista que lhe é

disponibilizada e atribuir-lhe um valor. Se a variável seleciona for um array, o sistema apresenta um novo campo com o índice para o estudante preencher. Esta atribuição deve ser feita através de uma expressão. Depois do estudante preencher todos os campos o sistema deve averiguar se os tipos dos valores utilizados na expressão são compatíveis com o tipo de operação que executam e se o tipo do valor resultante dessa expressão corresponde ao tipo da variável ao qual este está a fazer a atribuição (fig. 4.3).

Figura 4.3 - Ilustração das duas vertentes do menu "Atribuir".

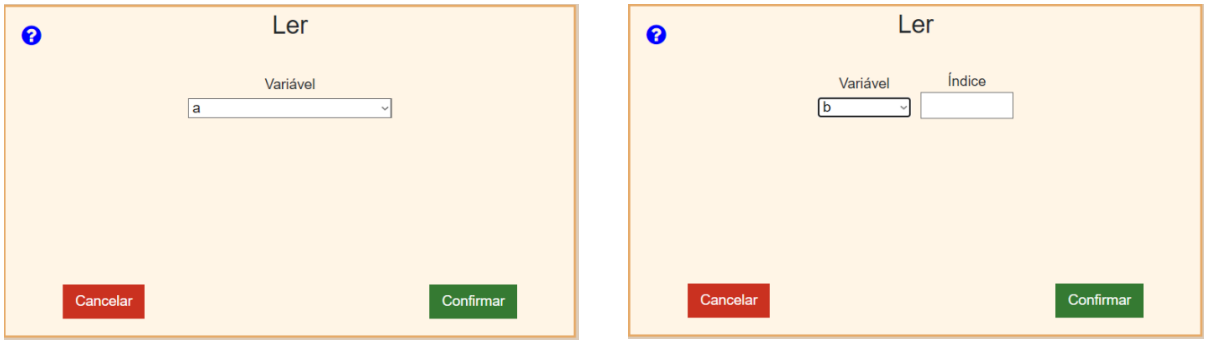


4.2.1.2 Entrada e saída de dados

4.2.1.2.1 Ler

Este bloco permite ao estudante implementar uma instrução que atribua um valor a uma variável durante a animação. Desta forma, o menu apresenta a lista das variáveis existentes no programa para o estudante selecionar a que pretende para a instrução. Tal como acontece no bloco anterior, o sistema apresenta um campo para o índice se a variável for um *array.* Por fim, antes de confirmar a implementação, o sistema verifica se a expressão definida no índice corresponde a uma expressão que devolva um valor numérico (fig. 4.4).

Figura 4.4 - Ilustração das duas vertentes do menu "Ler".



4.2.1.2.2 Escrever

Este bloco permite ao estudante imprimir uma mensagem na consola durante a fase de animação (fig. 4.5). Para tal, o estudante tem duas possibilidades:

- Uma mensagem simples onde apenas insere o nome da variável ou uma expressão. Por exemplo, se o estudante pretender imprimir o valor da variável “a” terá que introduzir o seguinte:

  a

- Concatenar texto a uma ou várias expressões ou variáveis. Esta concatenação é realizada através do operador “+”. De forma a distinguir estas partes utilizam-se aspas que envolvem apenas o texto que o estudante pretende inserir na sua mensagem.

  “Vou imprimir o valor de a:” + a

Neste tipo de instrução, caso o estudante pretenda imprimir uma expressão, deverá sempre rodeá-la por parênteses.

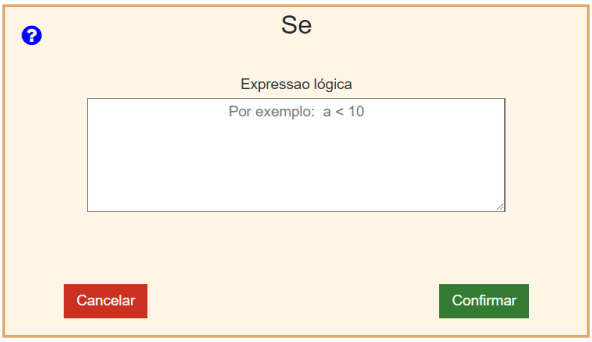
Figura 4.5 - Menu de implementação da instrução "Escrever".

4.2.1.3 Estruturas condicionais

4.2.1.3.1 Se

Este bloco permite que o estudante implemente um bloco que execute uma parte do algoritmo mediante a verificação de uma condição (fig. 4.6). Caso a condição não se verifique, o algoritmo segue automaticamente para a instrução seguinte relativamente à instrução “Se”. Para tal, o estudante necessita de definir uma expressão lógica que traduza a verificação que este pretenda realizar. Tal como acontece na grande maioria dos blocos referidos até então, antes é feita a verificação sobre se a expressão lógica definida pelo mesmo respeita as regras sintáticas e se os valores utilizados na mesma são compatíveis.

Figura 4.6 - Menu de implementação da instrução "Se".



4.2.1.3.2 Se/Senão

Este bloco opera da mesma forma que o anterior, logo, a sua implementação deve seguir as mesmas regras (fig.4.7). O que distingue estes dois blocos encontra-se no modo de operar após a verificação da expressão lógica. Ao contrário da instrução anterior, esta permite ao estudante definir uma série de instruções para o caso de a instrução ser verdadeira e outra para o caso da instrução ser falsa.

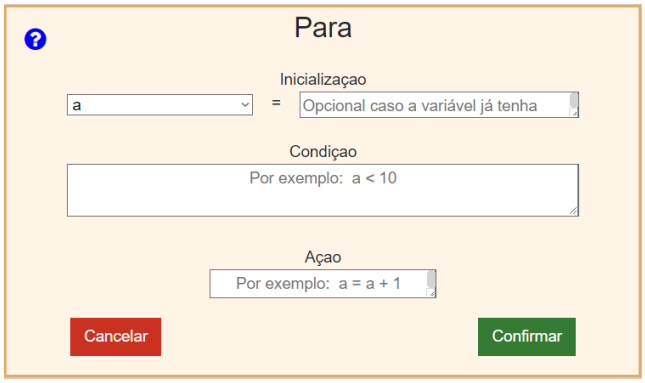
Figura 4.7 - Menu de implementação da instrução "Se/Senão".

4.2.1.4 Estruturas de repetição

4.2.1.4.1 Para

Este bloco permite implementar uma instrução que representa o ciclo *for* em programação (fig. 4.8). Para tal o estudante deve inicializar uma variável de forma semelhante ao que faz no bloco “Atribuir”. Este processo é obrigatório apenas se a variável não tiver sido inicializada anteriormente no fluxograma. Posteriormente, o estudante deve definir a expressão de controlo e a ação a realizar após a execução de todas as instruções inseridas nesta estrutura. Na definição da expressão de controlo o estudante é forçado a definir uma expressão lógica utilizando a variável escolhida anteriormente.

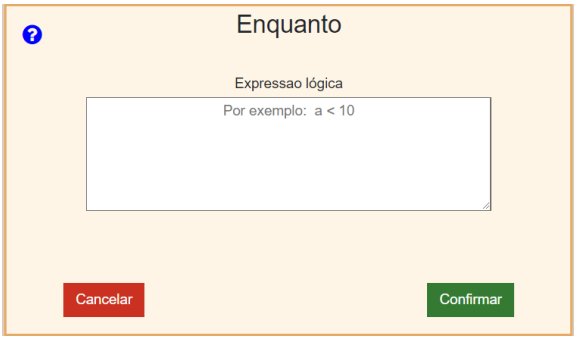
Figura 4.8 - Menu de implementação da instrução "Para".



4.2.1.4.2 Enquanto

Este bloco permite implementar uma instrução que representa o ciclo *while* em programação (fig. 4.9). Para tal, o estudante executa os mesmos passos que efetua na implementação de uma estrutura condicional. Optou-se por esta abordagem de modo a que o processo de implementação deste bloco seja o mais semelhante possível à implementação deste tipo de instrução através de código e os passos necessários para que esta funcione de acordo com o seu propósito. Assim, o estudante deverá iniciar a variável utilizada na condição antes da implementação deste bloco e deverá proceder à redefinição da mesma na última instrução presente nele. Desta forma, acredita-se que este será capaz de distinguir melhor as diferenças entre as duas estruturas cíclicas fornecidas pela ferramenta e produzir uma melhor transição para o desenvolvimento de código. Por esta razão, neste bloco não se realiza a inicialização de variáveis nem a ação para o ciclo evoluir. Caso isso acontecesse, teríamos um modo de implementação idêntico ao *for.*

Figura 4.9 - Menu de implementação da instrução "Enquanto".



4.2.3 Variáveis

De forma a iniciar o processo de construção do fluxograma, o estudante deve primeiro começar por criar um conjunto de variáveis à sua escolha. A ferramenta suporta três tipos de variáveis: numéricas, *Strings* e tabelas numéricas. Optou-se por simplificar ao máximo a tipologia das variáveis de maneira a permitir que o estudante se focasse no principal que é a obtenção e construção de estratégicas lógicas para

abordar os problemas que lhe são colocados. Pela mesma razão, achou-se desnecessário abordar o tipo *String* como um conjunto de carateres, julgando-se que a disponibilização apenas de *arrays* de valores numéricos fossem suficientes para a aprendizagem dos conceitos relativos à gestão destas estruturas. Posto isto, para a criação de uma variável o estudante necessita de preencher primeiramente dois campos:

- Nome da variável
- Tipo

Consoante o tipo selecionado, o sistema fornece a possibilidade ao estudante de criar uma variável do tipo *array* e definir o seu tamanho de acordo com as especificações referidas anteriormente (fig. 4.10 e fig. 4.11).

Figura 4.10 - Disposição do menu para a criação da variável caso o tipo selecionado seja "numérico".

Figura 4.11 – Disposição do menu para a criação da variável caso o tipo selecionado seja "*String*".

Após o preenchimento dos campos disponibilizados, o estudante pode proceder à criação da variável. Contudo, esse processo necessita de ser validado pelo sistema de forma a verificar se todas as normas necessárias foram cumpridas. Caso não se verifique, o sistema informa o estudante sobre o erro que cometeu. Assim, o nome introduzido pelo utilizador deve ser definido segundo as seguintes especificações:

- O nome não deve iniciar com uma letra maiúscula;
- O nome não deve iniciar por um número;
- O nome não deve conter carateres especiais ou qualquer outro que não seja letras do alfabeto, números naturais ou o carater "_";
- O nome da variável não pode condizer com um nome de uma função existente no programa;
- As variáveis do tipo *array,* devem ter um tamanho máximo de 12 unidades.

Por fim, resta apenas referir o tipo de valores que este tipo de variáveis podem assumir. As variáveis numéricas abrangem todo o conjunto dos valores inteiros e decimais. A representação de números decimais deve ser feita utilizando o carater "." para a separação da parte inteira e a parte decimal. Para atribuir um valor a uma *String* o estudante deve envolver o conteúdo definido por aspas. Este conteúdo pode ser formado por todos os carateres:

[a-zA-Z0-9,;.:-_[\]ºª!|@#&/()=?']*

No processo de edição e remoção de variáveis, foi necessário definir o sistema de modo que estas alterações não comprometam a estrutura do fluxograma e prevenir erros que possam advir dessas alterações. Assim, quando o estudante edita uma variável, o sistema percorre todos os blocos do fluxograma e procura pelas ocorrências da variável editada nas instruções. Caso encontre alguma referência à mesma, procede à alteração automática das expressões presentes em todos os blocos e atualiza a variável para o novo nome definido pelo estudante. Caso se trate do processo de remoção de uma variável, realiza o mesmo processo de deteção das ocorrências das variáveis nas instruções em todos os blocos. Se encontrar alguma ocorrência, o sistema remove automaticamente o bloco onde a utilização da variável removida ocorre. Para realizar estes dois processos, o estudante deve primeiro selecionar uma variável apresentada na tabela da secção que será destacada após a seleção. Depois, se quiser remover uma variável, deve pressionar a tecla "Remover" que o sistema disponibiliza na secção e na qual a sua ação foi permitida após a seleção de uma variável da tabela. Caso este pretenda editar uma variável, o estudante deve selecionar duplamente uma variável da tabela (fig. 4.12). Após esta ação, o sistema apresenta um menu de edição onde são expostas as características atuais das variáveis juntamente com os campos de edição. Neste processo, para variáveis simples o sistema apenas permite a redefinição do nome da variável não permitindo modificar os tipos das variáveis. Caso a variável seja um *array*, o sistema também fornece a possibilidade de alterar o tamanho do mesmo. Ao contrário do que acontece no processo da adição de uma variável, o sistema não requere o preenchimento de todos os campos do menu de implementação. Assim sendo, caso o estudante não preencha um determinado campo o sistema assume que este não pretende modificar esse campo.

Figura 4.12 - Menu apresentado ao utilizador para a edição de uma variável.

4.2.4 Funções auxiliares

A ferramenta possui um conjunto de funções auxiliares pré-definidas pelo sistema. Estas funções têm como função permitir ao utilizador executar operações comuns em linguagens de programação sem a necessidade de este compreender os passos computacionais da sua execução. Este conjunto permite ao estudante manipular os vários tipos de variáveis integradas na ferramenta. De seguida, encontram-se listadas todas as funções presentes na ferramenta juntamente com tipo de variável que cada uma maneja. A lista de funções auxiliares existentes na ferramenta encontra-se em anexo.

4.2.5 Expressões

Como foi possível identificar anteriormente, a definição de expressões são uma parte fundamental para a implementação dos blocos do fluxograma. Dada a sua importância, não se definiram estas expressões de acordo com as regras sintáticas de uma linguagem de programação específica. Apesar desta se assemelhar à linguagem utilizada pelo Java, procurou-se alterar a representação de símbolos menos familiarizados pelo estudante. Assim, procurou-se substituir a representação destes símbolos por via textual de forma a permitir que o estudante identifique visualmente o significado e a funcionalidade de determinado símbolo. Relativamente aos valores possíveis de integrar nas expressões, o estudante poderá utilizar os seguintes:

- Valores numéricos
- Funções
- Variáveis
- *Strings*

De forma a implementar uma expressão corretamente, o estudante deve cumprir com as seguintes regras:

- Haver concordância entre o número de parênteses;

- Haver concordância entre os tipos de valores utilizados numa determinada operação;
- Na chamada de funções, não poder haver espaço entre o nome da função e os parenteses que iniciam a apresentação dos parâmetros da mesma;
- Na chamada de funções, os parâmetros das mesmas devem ser *Strings,* variáveis, valores numéricos, expressões aritméticas ou chamada de uma nova função;
- Na chamada de funções, haver concordância entre o tipo de valor passado e o tipo de parâmetro da função onde está localizado;
- Na chamada de funções, os argumentos devem ser separados por uma vírgula sucedida de pelo menos um espaço.
- Na utilização de variáveis, não poderem ser utilizadas variáveis que ainda não tenham sido inicializadas no fluxograma;
- No acesso a valores de *arrays,* assegurar que todos os valores ou expressões inseridas no índice são numéricos;
- Para realizar uma operação lógica entre duas expressões relacionais, devem-se envolver as expressões entre parênteses.
- Não podem ocorrer dois operadores seguidos;
- Não podem existir identificadores não reconhecidos pelo sistema;
- Deve definir uma expressão de acordo com o tipo de bloco que está a implementar.

4.2.5.1 Expressões aritméticas

Devido à familiaridade dos estudantes com os operadores utilizados neste tipo de expressões não se sentiu a necessidade de alterar a sua representação. Assim sendo, de seguida encontram-se disponibilizados os operadores aceites pela linguagem juntamente com uma descrição da operação que efetua.

Tabela 4.1 - Operadores aritméticos presentes na ferramenta.

Operador	Significado	Exemplo
+	Adição	x + y
-	Subtração	x – y
/	Divisão	x / y
*	Multiplicação	x * y
%	Módulo	x % y
^	Exponencial	x ^ y

4.2.5.2 Expressões lógicas

Relativamente à definição dos operadores das expressões lógicas, estes podem-se dividir em dois grupos: relacionais e lógicos. Tal como os operadores mencionados na secção anterior, os operadores relacionais são igualmente do conhecimento dos estudantes devido aos conhecimentos adquiridos ao longo do seu percurso académico na área da matemática. Desta forma, não se sentiu a necessidade de alterar a sua representação. Contudo, o mesmo não acontece com os operadores lógicos. Como tal,

decidiu-se representar estes operadores através de uma representação textual de modo a que os estudantes percebam claramente a operação que estão a realizar quando utilizam estes operadores.

Tabela 4.2 - Operadores relacionais presentes na ferramenta.

Operador	Significado	Exemplo
<	Menor	x < y
>	Maior	x > y
<=	Menor ou Igual	x <= y
>=	Maior ou Igual	x >= y
==	Igual	x == y
!=	Diferente	x != y

Tabela 4.3 - Operadores lógicos presentes na ferramenta.

Operador	Significado	Exemplo
AND	E	x AND y
OR	OU	x OR y

4.2.5.3 Regras de precedência dos operadores

Em expressões complexas, é necessário compreender a ordem como estas são executadas. Para tal, é necessário ilustrar a prioridade dos vários operadores presentes numa expressão e as implicações que as suas utilizações têm no cálculo da expressão. De seguida, apresentam-se as tabelas que indicam os níveis de precedência entre os vários operadores lógicos e aritméticos utilizados na ferramenta. As tabelas encontram-se estruturadas de forma decrescente em relação ao nível de prioridade do operador.

Tabela 4.4 - Tabela de precedência dos operadores utilizados na ferramenta.

Precedência
^
*, /, %
+, -
<=, >, >=, !=, = =
AND
OR

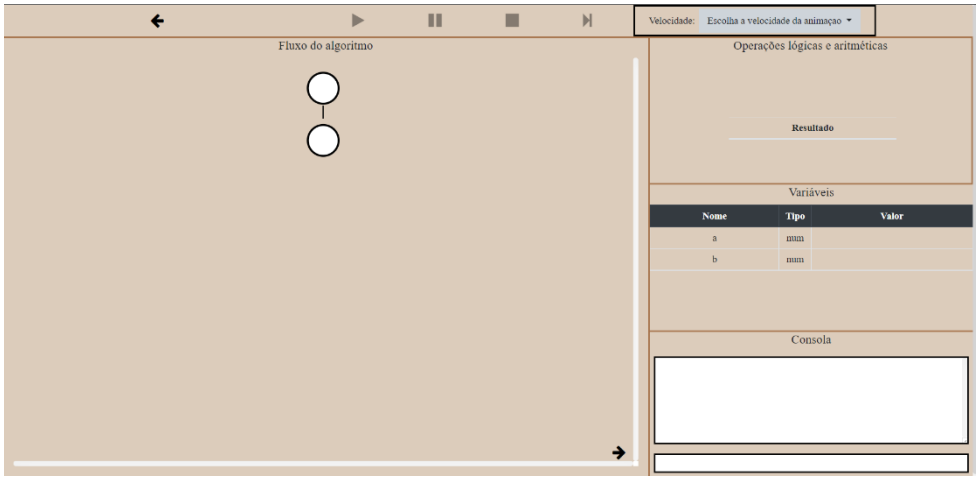
4.3 Menu de animação

Após a construção do fluxograma, o estudante poderá proceder à animação e simulação do fluxograma que construiu. Contudo, antes de o estudante ser encaminhado para o *layout* de animação, o sistema verifica, caso o fluxograma possua blocos representativos de estruturas seleção ou de repetição, se cada um deles possui

pelo menos um bloco em cada um dos lados. A partir da figura 4.13, pode-se observar um *layout* distribuído em quatro seções:

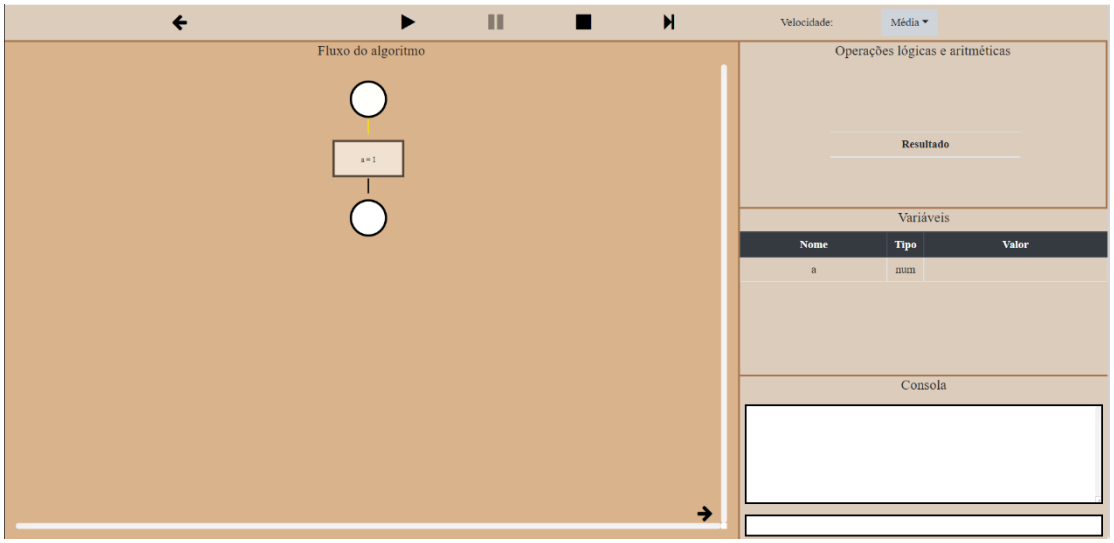
- Uma secção para a representação do fluxo de controlo do fluxograma;
- Uma secção para a execução das operações lógicas e aritméticas;
- Uma secção para a representação dos dados do fluxograma;
- Uma secção para a exibição dos dados na consola e entrada de dados pelo teclado.

Figura 4.13 - *Interface* do menu de animação da ferramenta.



Ao longo da animação, o estudante poderá observar os aspetos computacionais relativos a cada uma das secções referidas anteriormente. Dada a quantidade de informação exposta no ecrã, procurou-se construir a animação de modo a que esta seja capaz de conduzir o estudante a observar os aspetos computacionais de interesse de forma clara (fig. 4.14). Assim, a cada passo da execução, o sistema destaca a operação que está a ser executada, na secção respetiva onde essa operação está a ocorrer. De seguida encontram-se expostas os propósitos de cada secção definida na *interface*.

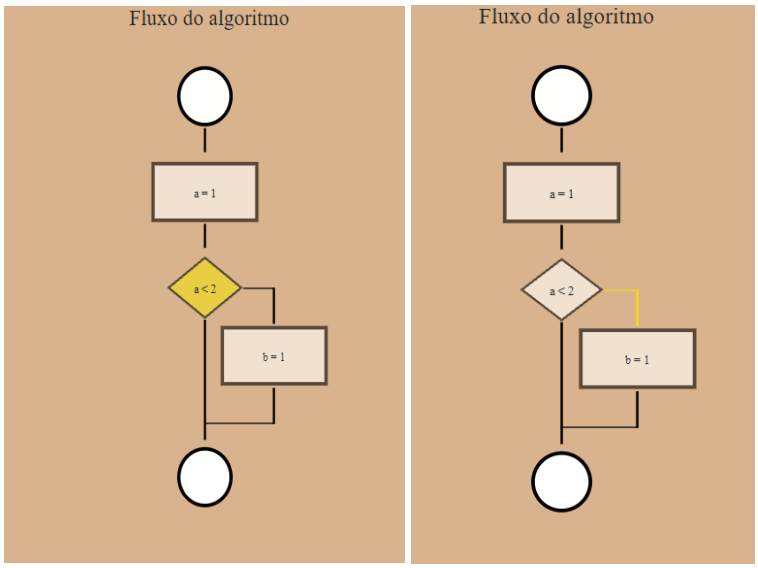
Figura 4.14 - Imagem elucidativa do destaque da secção do Fluxo de controlo durante a sua animação.



4.3.1 Fluxo de controlo

Esta secção tem como objetivo representar a navegação do fluxograma ao longo da execução da animação. Desta forma, o estudante poderá observar as implicações da sua implementação na navegação do fluxograma. Este aspeto torna-se relevante quando este implementa instruções de controlo como ciclos e condições já que o resultado obtido na interpretação das mesmas influencia o percurso do algoritmo. Assim, nesta secção o estudante poderá visualizar o sequenciamento dos blocos percorridos no seu algoritmo através do destaque das estruturas que compõem cada bloco e dos elos de ligação que ligam os blocos do fluxograma (fig. 4.15).

Figura 4.15 - Imagem representativa do processo de animação do fluxo de controlo do bloco "Se".



4.3.2 Operações lógicas e aritméticas

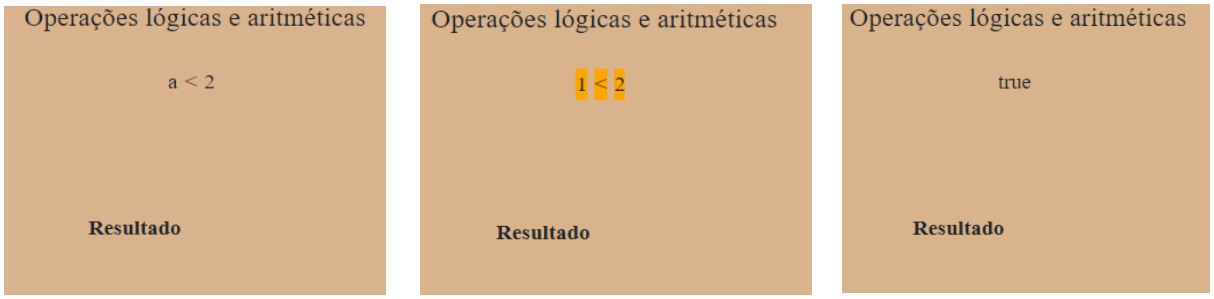
Esta secção tem como objetivo dar a possibilidade ao estudante de compreender os procedimentos, passo a passo, efetuados na interpretação de operações aritméticas e lógicas. Assim, o estudante poderá observar os seguintes passos executados pelo computador:

- Destaque da operação que vai ser efetuada;
- Caso a operação utilize um valor de uma variável, o sistema fornece uma animação que representa o acesso a esse valor por parte do computador;
- Apresentação do resultado obtido;
- Substituição dos valores utilizados na operação pelo resultado obtido.

Considerou-se este aspeto relevante não tanto na parte de demonstrar as operações aritméticas, visto que o estudante já deve conhecer o modo como estes se executam, mas sim para mostrar a interpretação das operações lógicas. Para além destas não serem tão familiares à maior parte dos estudantes, principalmente se não tiverem

qualquer experiência anterior a nível de programação, o seu contributo é essencial para ditar o fluxo do algoritmo visto que estas são utilizadas em estruturas de controlo. Desta forma, o estudante poderá observar claramente o resultado da operação e o efeito que teve na navegação do fluxograma a partir da animação do fluxo de controlo (fig. 4.16).

Figura 4.16 - Imagem representativa do processo de algumas das animações existentes na secção "Operações lógicas e aritméticas".



4.3.3 Dados do fluxograma

Nesta secção, o estudante poderá observar animações relativas à atualização e acesso a valores das variáveis. Quando um valor é atribuído a uma variável o sistema procede ao encaminhamento do resultado obtido para o local da tabela onde se situa a variável em causa (fig. 4.17).

Figura 4.17 – Secção "Variáveis" no final de ser realizada uma atribuição.

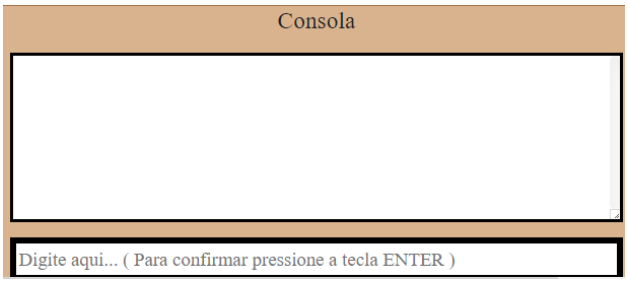
Variáveis

Nome	Tipo	Valor
a	num	1

4.3.4 Consola

Esta secção serve para os alunos poderem realizar a instrução relativa à leitura de uma variável e da visualização de uma mensagem do algoritmo. Na animação da leitura de um variável, para além do destaque realizado à secção, como acontece com todas as outras, o sistema apresenta uma informação adicional a indicar onde o estudante deve introduzir o valor a atribuir à variável e o modo como deve submeter o valor que introduziu (fig. 4.18).

Figura 4.18 - Visão da secção "Consola" quando está a ser realizado o pedido de uma variável ao utilizador.



4.3.5 Ações de controlo

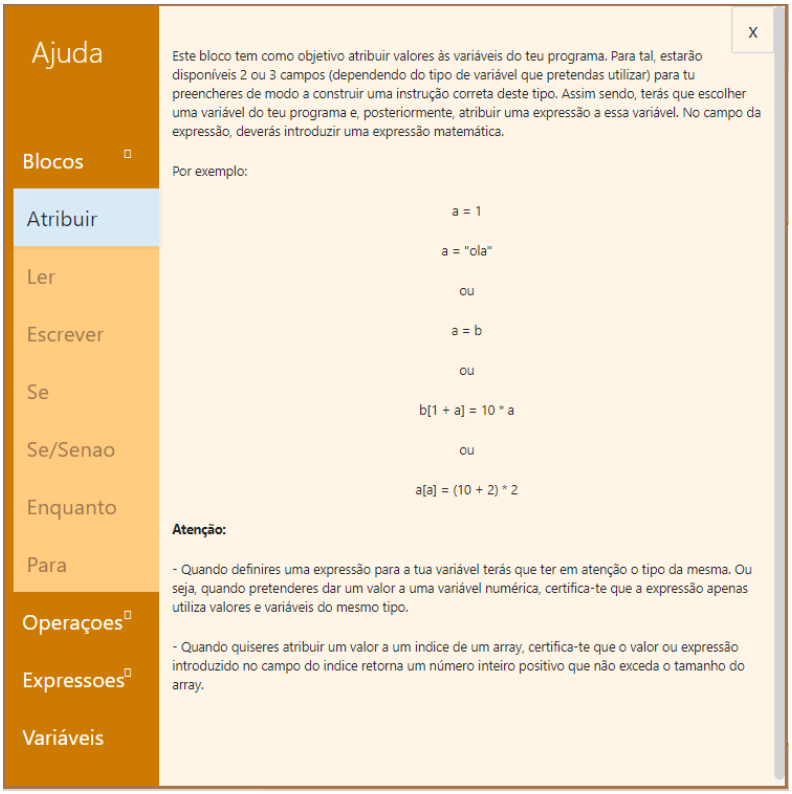
Durante o processo de animação, são disponibilizadas uma série de ações que permitem ao estudante controlar o desenrolar da animação. Para tal, o estudante possui um leque de opções para esta tarefa presentes na parte superior da *interface* da animação. Considerou-se importante fornecer estas funcionalidades de forma a permitir que este seja capaz de adaptar este processo às suas necessidades. Como tal, foram disponibilizadas as seguintes opções ao estudante:

- **Controlar a velocidade de animação;**
- **Executar a animação passo a passo;**
- **Iniciar uma animação contínua da animação;**
- **Interromper a animação;**
- **Recomeçar a animação.**

4.4 Menus de Ajuda

A partir das figuras apresentadas dos menus dos blocos existentes no sistema, cada um deles possui um ícone que ilustra um ponto de interrogação. Este ícone tem como funcionalidade iniciar o menu de ajuda. Este menu é constituído por um tabulação relativa a cada conceito envolvido no processo de construção do fluxograma. Considerou-se importante fornecer um meio complementar para explicar os propósitos de cada ação durante a fase de implementação tal como o modo como devem ser implementados caso a informação exposta nos menus não seja suficientemente explicita para eles. Assim, após uma seleção deste ícone, o sistema apresenta o menu de ajuda na secção relativa ao componente onde o menu de ajuda foi requerido.

Figura 4.19 - Menu ajuda apresentado caso a sua requisição tenha ocorrido na construç do bloco "Atribuir".



Capítulo 5
Arquitetura

Após a definição dos requisitos para o projeto, foi necessário estruturar o sistema de modo a englobar todas as funcionalidades descritas nesse capítulo e definir como os requisitos não funcionais irão ser alcançados. Como tal, neste capítulo segue-se a apresentação da arquitetura para o projeto. Contudo, antes que se aborde esse tópico, será feita uma apresentação da arquitetura de um sistema para o Moodle de modo a se perceber o enquadramento do sistema do projeto nesta plataforma.

Uma vez que a ferramenta desenvolvida vai ser integrada na plataforma Moodle, considera-se de interesse descrever as arquiteturas subjacentes à integração e comunicação entre todos os componentes do sistema. De modo a ser possível operar no Moodle, é necessária a integração de várias componentes com o código fonte do sistema. Estas componentes são o Servidor *web* e a Base de dados, que serão em seguida descritos de forma a melhor compreender a arquitetura da plataforma.

5.1 Servidor

O servidor web normalmente utilizado por sistemas no Moodle é o Apache[12]. Este servidor deve ser capaz de suportar a linguagem Hypertext Preprocessor (PHP) pois é a linguagem utilizada na implementação do sistema do Moodle. PHP é uma linguagem que permite a criação de *scripts* que operam do lado do servidor e é utilizada essencialmente para o desenvolvimento de aplicações *web* [13]. Esta linguagem possui a capacidade de conseguir misturar a sintaxe HTML no seu código para realizar alguma funcionalidade numa página *web*. Todo o código PHP é executado ao nível do servidor ao invés de ser no lado do cliente como acontece com o Java*script*. As funcionalidades num sistema destes são desenvolvidas através da criação de ficheiros que utilizem esta linguagem. Caso necessário, algumas funcionalidades podem utilizar ficheiros Java*script* e CSS. A comunicação entre as funcionalidades existentes no sistema, o acesso à base dados e à Moodle Data é feita através da chamada de métodos do servidor [14]. Visto que o Moodle consiste numa aplicação *web*, resta agora saber como é que é feita a comunicação entre o sistema e as páginas *web* acessíveis ao utilizador. Esta é realizada através do protocolo HTTP onde os URLs correspondem à localização do *script*, que está a ser exibido na página, na pasta do projeto [14], [15]. Por exemplo, se a página que se quer mostrar ao utilizador corresponde a uma atividade, o URL irá ser [15]:

(ip do servidor/domain) /mod/nome do plugin/index.php

Mais à frente neste documento, irá ser explicada a estrutura do código de um sistema deste tipo. Em conjunto com este URL vai um parâmetro ID relativo ao curso no qual a atividade está inserida. No que diz respeito à passagem de informação através dos URLs dos pedidos dos utilizadores para o sistema, são utilizados os típicos métodos GET ou

POST. A diferença entre os dois é que os parâmetros passados para o sistema vão discriminados no URL no método GET, enquanto que no método POST, já que integram o corpo do pedido enviado para o sistema, esses parâmetros estão ocultados.

5.2 Base de dados

Relativamente à escolha da base de dados a utilizar no sistema, o Moodle oferece uma grande variedade de opções aos programadores. Dentro deste leque de escolhas estão [14], [16]:

- MySQL;
- PostgreSQL;
- Microsoft SQL;
- Oracle.

Contudo, as bases de dados completamente suportadas pelo Moodle são o MySQL e o PostgreSQL[17]. Como a atividade da ferramenta não requer uma grande utilização da base de dados e o projeto se trata do desenvolvimento de um componente a ser integrado num sistema Moodle, a escolha pelo PostgreSQL adveio de uma maior familiaridade do autor com a sua utilização.

5.3 Módulos Java*script*

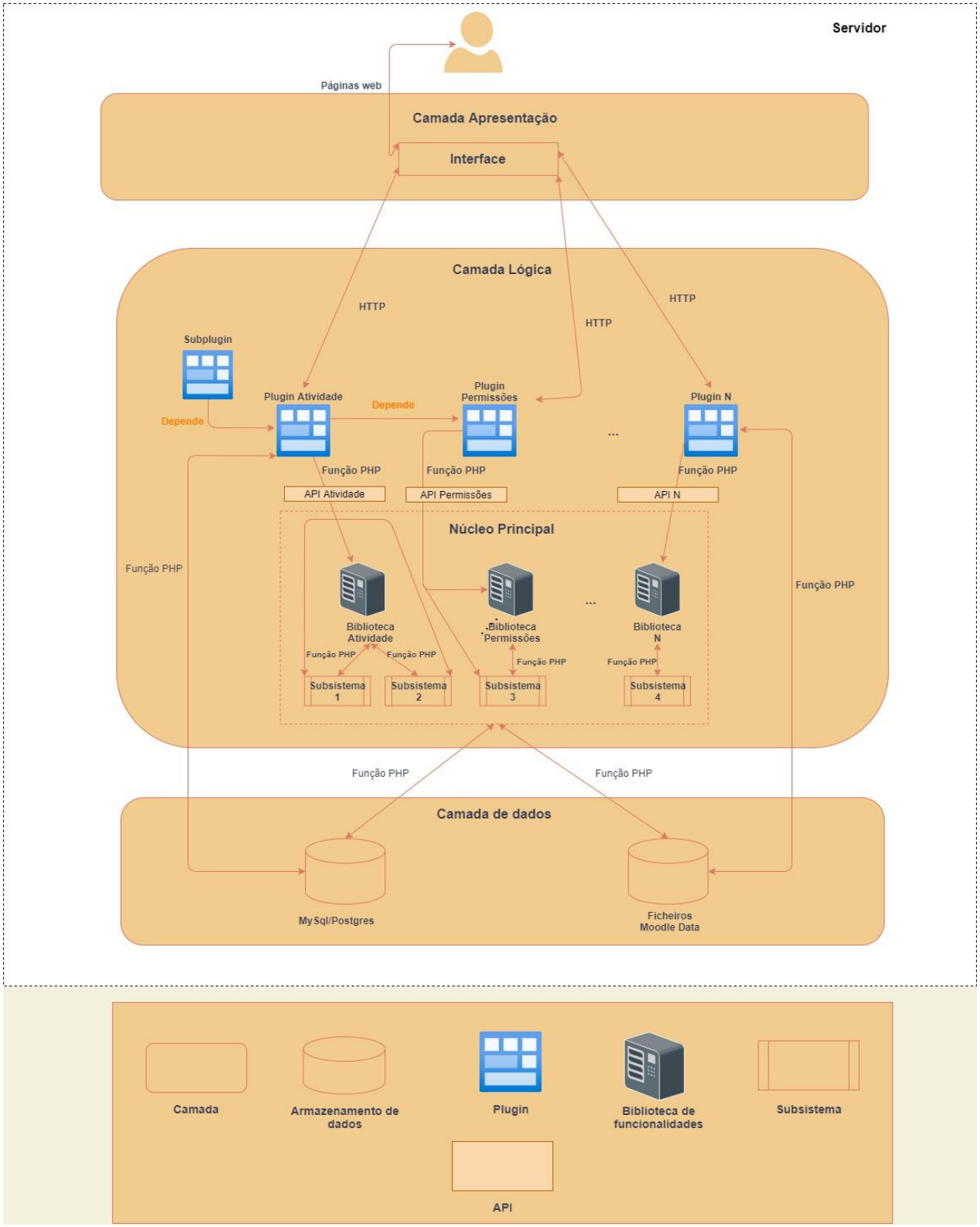
De modo a integrar interações nas páginas *web*, o Moodle permite introduzir funcionalidades Java*script* nas mesmas. Esta introduçao é feita através da importação de módulos Java*script* para a página através do requireJS. O requireJS é uma tecnologia utilizada pelo Moodle e permite que este carregue módulos Javascipt para as suas páginas *web* [18]. Esses módulos são módulos AMD[18]. O AMD (Asynchronous Module Definition) permite encapsular código Java*script* que opera diretamente no *browser.* De modo a integrar estes módulos no Moodle, é necessário compilá-los utilizando o Grunt, que realiza a preparação dos módulos para poderem ser utilizá-los pelo sistema[20].

5.4 Arquitetura de um sistema Moodle

Tal como foi referido, esta ferramenta foi desenvolvida para o Moodle. Assim sendo, segue-se uma análise aos vários aspetos que compõem uma arquitetura de um sistema para esta plataforma e o modo como estes estão organizados. Assim, recorreu-se à análise da documentação fornecida no site oficial do Moodle sobre esta matéria [14], a partir da qual se concluiu que a plataforma está estruturada segundo uma arquitetura de três camadas (fig. 5.1):

- **A camada de apresentação –** representa a *interface*, neste caso páginas *web*, onde o utilizador vai requisitar os pedidos ao sistema para executar as funcionalidades que lhe são fornecidas.
- **A camada lógica -** é composta por todos os módulos que fornecem as funcionalidades ao sistema.
- **A camada de dados -** é onde são guardadas todas as informações que circulam no sistema.

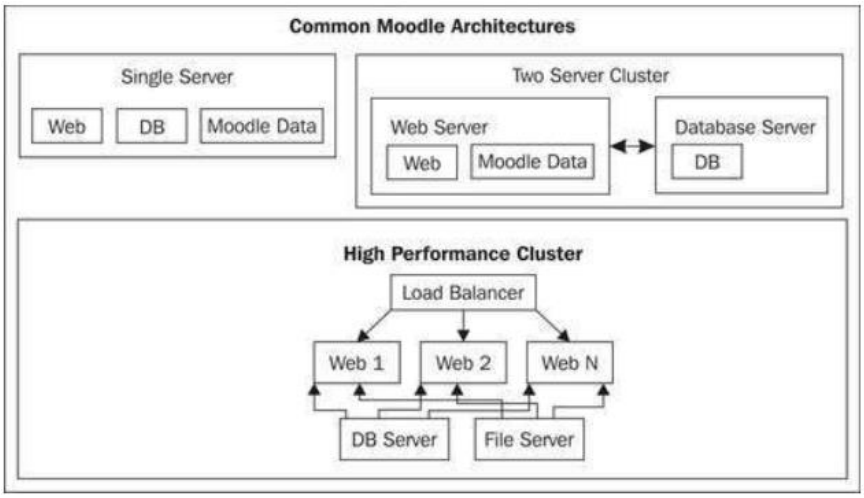
Figura 5.1 - Visão geral de uma arquitetura local de um sistema Moodle.



Relativamente à organização destas camadas, elas podem ser colocadas de diferentes maneiras. A disposição mais trivial seria integrar todas estas componentes no mesmo servidor. Obviamente, esta alternativa não é a ideal para garantir uma performance

desejável para os sistemas que irão ser usados simultaneamente por vários utilizadores, visto que irá aplicar uma grande sobrecarga no servidor. De forma a tornar o sistema escalável e garantir essa performance, é necessário agrupá-las num *cluster*. Este conceito define uma arquitetura de um sistema cujas componentes estão divididas em vários servidores. Assim, no caso dos sistemas abordados nesta secção, uma primeira fase para escalar o sistema consiste em dividir o sistema inicial em dois servidores: um servidor *web* para o tratamento das funcionalidades do Moodle e dos ficheiros nele inseridos, ou seja, a Moodle Data, e outro para a base de dados. Para garantir uma performance de alta qualidade, existe a possibilidade de utilizar um *Load Balancer*. Através deste método, é possível dividir igualmente o trabalho por todos os servidores *web*, evitando a sua sobrecarga [16] (Fig. 5.2).

Figura 5.2 - Diferentes organizações para a arquitetura de um sistema para o Moodle[15].



A camada lógica é constituída por um núcleo principal que é a parte central do seu sistema dado que sustenta todas as suas componentes. Sem ele nada funciona. Este núcleo é composto por um conjunto de bibliotecas que fornecem várias funcionalidades, permitindo a criação de um LMS (Learning Managment System). Juntamente com estas bibliotecas, este núcleo também é composto por vários subsistemas que estão associados a uma biblioteca específica. As funcionalidades inicialmente introduzidas por este núcleo são:

- Criação de cursos construídos a partir de um conjunto de atividades e outros recursos como, por exemplo, blocos dispostos de forma hierárquica;
- Diferenciação dos utilizadores entre estudante e professor;
- Gestão da inscrição dos estudantes nos cursos disponíveis na plataforma;
- Fornecimento de mecanismos que definem as permissões de acesso a certas funcionalidades mediante o tipo de utilizador;
- Configuração do perfil do utilizador. Sempre que um utilizador cria uma conta o perfil é automaticamente criado;
- Tratamento estatístico e mensagens *log*.

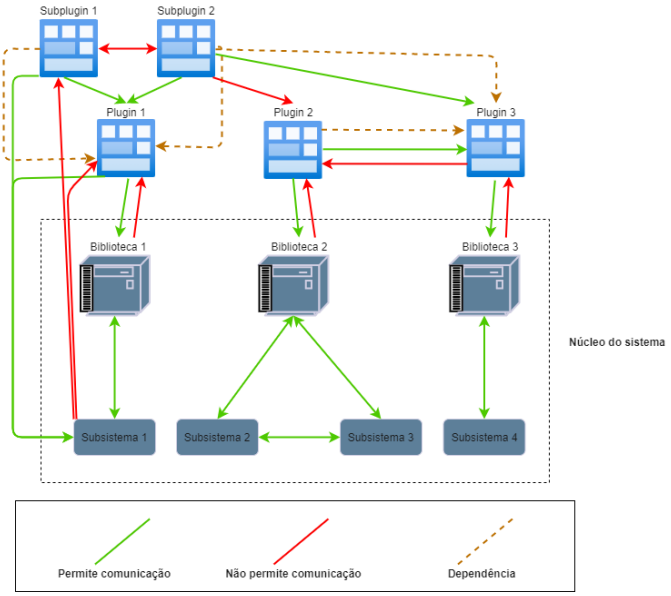
Como se sabe, o Moodle é um sistema modular, ou seja, permite a integração de módulos que adicionem funcionalidades ao seu sistema. Estes módulos, normalmente independentes, são designados de *plugins*. Os *plugins* permitem aos programadores

estender as funcionalidades iniciais do Moodle já que estão ligados diretamente a uma das funcionalidades fornecidas pelo núcleo. Estes *plugins* podem possuir *subplugins* que, seguindo a mesma lógica, estendem a sua funcionalidade. Assim, o Moodle proporciona aos programadores uma grande flexibilidade para desenvolver sistemas de acordo com as características que pretendem, mesmo estando assente num núcleo fixo. Existem vários tipos de *plugins* e cada um estende uma funcionalidade especifica do núcleo. Dentro destes tipos, destacam-se os seguintes, em termos de importância no desenvolvimento do sistema:

- **Atividade** – é o recurso principal para a criação dos cursos e integração de ferramentas no sistema;
- **Blocos** – são *interface*s adicionadas à página web do sistema que servem para dispor algum tipo de informação aos utilizadores;
- **Temas** – encarregam-se do aspeto visual das páginas web do sistema;
- **Formatos de curso** – estruturam a ordem pela qual o curso e as suas atividades são apresentados ao utilizador
- **Linguagem** – o Moodle permite definir várias linguagens nas suas páginas;
- **Autenticação** – controla o modo como é feita a autenticação no sistema;
- **Permissões** – define quais utilizadores são elegíveis aos cursos disponíveis no sistema;
- **Repositório** - permite fazer *uploads* de ficheiros para o sistema.

Examinando a informação anterior, consegue-se verificar a existência de dois tipos de componentes que constituem a camada lógica destes sistemas. Um tipo compõe o núcleo do sistema e o outro os *plugins*. Importa agora perceber como é que todos os componentes comunicam. A comunicação entre os componentes que envolvem um sistema no Moodle é feita em função de algumas regras [19]. Todos os componentes podem comunicar entre si caso seja necessário executar determinada funcionalidade. No entanto a comunicação entre o núcleo e os *plugins* é feita de forma unidirecional, ou seja, apenas o *plugin* tem a capacidade de comunicar com uma biblioteca do componente principal. Esta comunicação é feita através de uma API referente à biblioteca do núcleo [15]. Isto é, o *plugin* irá comunicar com a biblioteca da respetiva funcionalidade que pretende estender através da API associada a essa biblioteca. A comunicação entre *plugins* é, igualmente, unidirecional e apenas possível se forem definidas essas dependências. No caso dos *subplugins* é possível permitir que este comunique com outros *plugins*, para além do seu "pai", também através da definição das dependências (fig. 5.3).

Figura 5.3 - Diagrama de comunicação entre os componentes que compõe a camada lógica de um sistema.



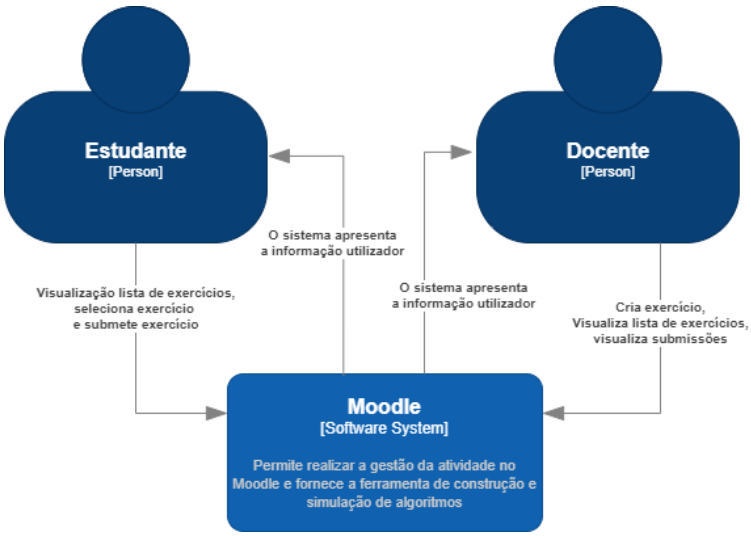
5.5 Arquitetura do *plugin*

Após a obtenção de conhecimentos relativamente ao modo como se encontra estruturado um sistema Moodle, foi possível desenhar a arquitetura para o *plugin* do projeto e definir o modo como este seria integrado no Moodle. Para a sua definição, recorreu-se à metodologia C4 [22] que se encontra disponibilizada de seguida.

5.5.1 Diagrama de contexto

A partir deste modelo, pretendeu-se estabelecer uma visão geral do sistema. Assim, através da figura 5.4 é possível observar uma replicação de baixo nível do sistema. Nesta figura, encontram-se ilustradas as entidades que compõe o sistema e uma generalização das atividades entre elas.

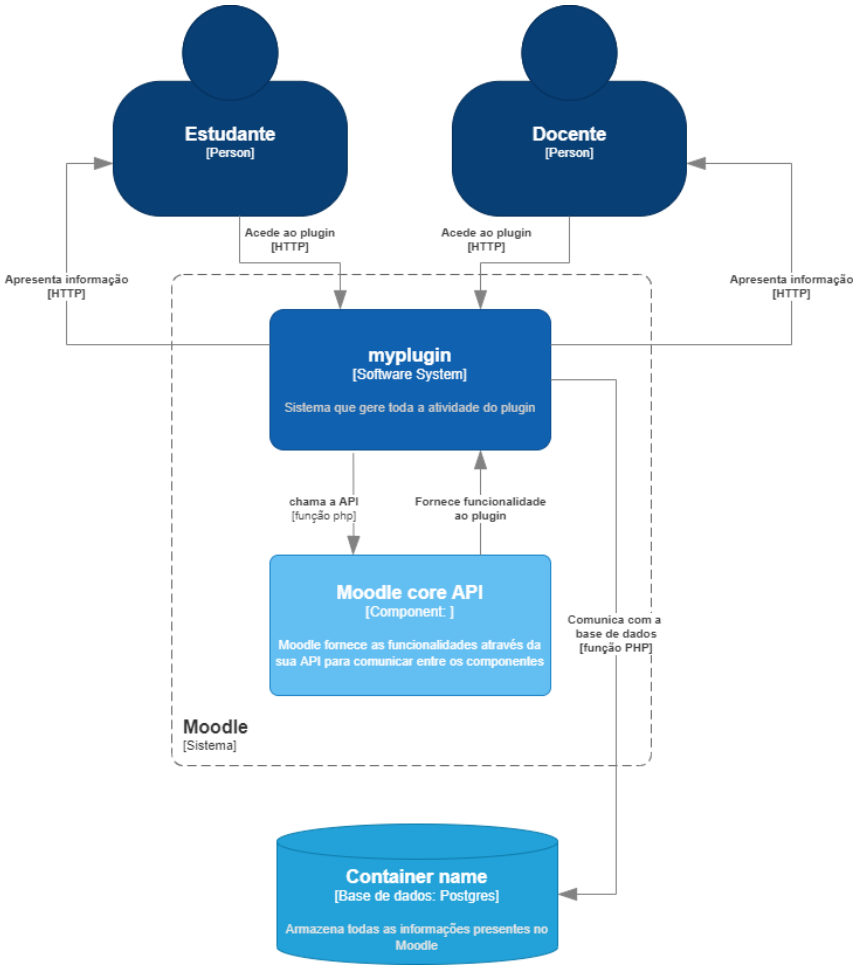
Figura 5.4 - Diagrama de contexto do *plugin*.



5.5.2 Diagrama de Contentores

Após uma introdução superficial das funcionalidades do sistema, segue-se a apresentação mais aprofundada em relação à constituição do sistema. Através do diagrama de contentores, permitiu-se demonstrar os elementos principais que constituem o sistema e o modo como estes comunicam. A partir da figura 5.5, é possível observar a presença de um contentor designado "my*plugin*". Este contentor representa o *plugin* desenvolvido no projeto, logo, este engloba todas as funcionalidades definidas nos objetivos. Para que tal seja possível, este *plugin* necessita de aceder às funcionalidades fornecidas pelo núcleo do sistema Moodle. Este acesso é realizado graças às APIs disponibilizadas pelo Moodle.

Figura 5.5 - Diagrama de contentores do *plugin*.

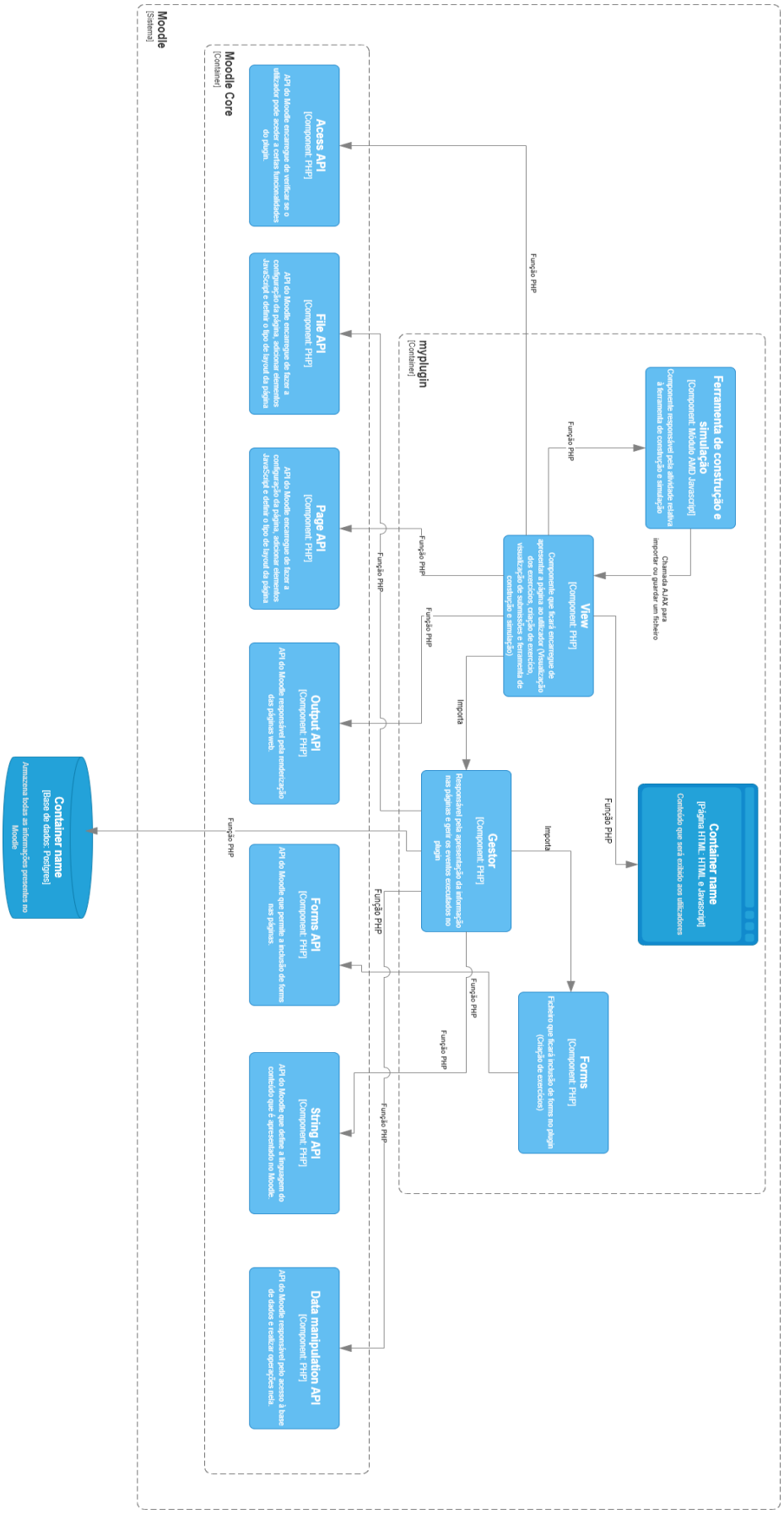


5.5.3 Diagrama de componentes

Depois da apresentação das entidades que constituem o sistema, é necessário compreender que componentes as constituem. Desta forma o diagrama de componentes apresentado na figura 5.6 apresenta as várias componentes que formam o "my*plugin*" bem como as APIs necessárias para alcançar os objetivos do projeto. A

utilização das APIs presentes nesta figura resultou de um estudo realizado relativamente ao conjunto das que são fornecidas pelo Moodle e das funcionalidades a que cada uma delas permite aceder[20].

Figura 5.6 - Diagrama de Componentes do *plugin*.

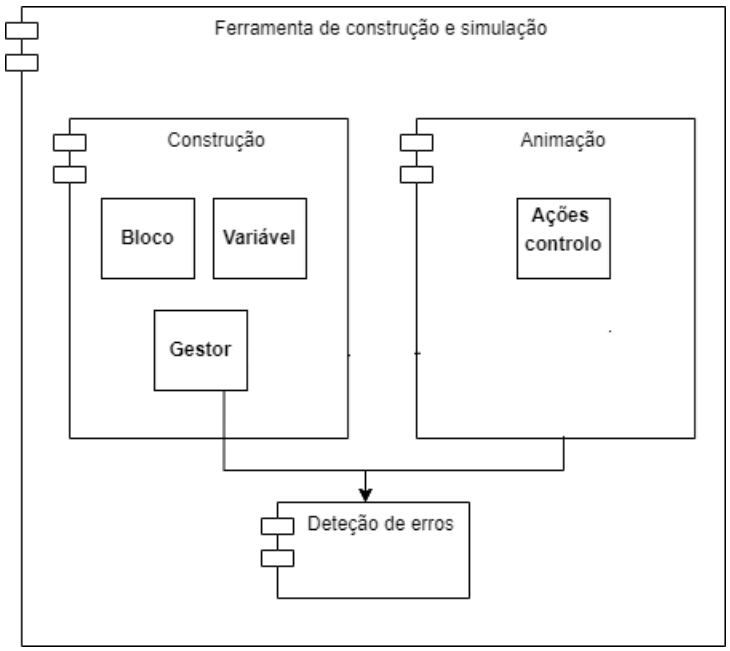


5.5.4 Módulos da ferramenta de construção e simulação

Após a exposição dos componentes internos que constituem a estrutura do *plugin*, é importante apresentar uma visão mais aprofundada da estrutura do módulo responsável pela atividade da ferramenta de construção e simulação de algoritmos. A partir da figura 5.7, consegue-se observar a presença de três módulos:

- **Construção:** atividade relativa à construção do fluxograma;
- **Animação:** atividade relativa à animação do fluxograma;
- **Deteção erros:** módulo responsável por verificar a correção de todas as ações do utilizador no uso da ferramenta e das ações resultantes da animação do fluxograma criado por este.

Figura 5.7 - Visão das componentes que compõe o módulo de construção e simulação de algoritmos.



O módulo de construção foi idealizado de modo a possuir três componentes. Duas delas (Bloco e Variável) tinham como objetivo realizar a representação dos objetos utilizados pelo utilizador no processo de criação do fluxograma. Por último, a componente de gestão representa a entidade responsável pela gestão de toda a atividade executada pelo utilizador (adição/remoção de blocos, implementação de blocos, atalhos de utilização, criação de variáveis, ...). Assim, esta foi responsável por fornecer todas as ações ao utilizador de modo a este conseguir completar o processo de criação de fluxogramas de forma simples e clara.

O módulo de animação encontra-se estruturado de forma semelhante em relação ao módulo anterior, adaptado às suas especificidades e às ações que compõem esta fase de utilização da ferramenta. Possui uma componente (Ações de controlo) que fornece ao utilizador a possibilidade de realizar as operações de controlo durante a fase de animação.

O módulo de deteção de erros teve como intuito fornecer ao sistema a capacidade de rastrear erros no processo do uso da ferramenta e reportá-los ao utilizador. Assim, este pode atuar em duas fases:

- Construção – realiza a análise sintática das expressões durante a fase de implementação do fluxograma. Para além disso, verifica a correspondência entre o que foi introduzido pelo utilizador e o que é suposto preencher em cada campo nos vários componentes necessários à implementação do fluxograma. Por fim, este módulo também analisa as componentes das expressões de forma a verificar se existe congruência entre essas e o tipo de operação que estão a representar. Por exemplo, impedir realizar operações aritméticas entre variáveis de diferentes tipos.
- Animação – durante o processo de animação, este módulo está encarregue de detetar possíveis erros que possam ocorrer nesta fase. Esses erros podem ser de dois tipos:
    - Tentativa de aceder a um valor for dos limites de um *array;*
    - Atribuição de um valor diferente da variável pedida numa instrução do tipo "Ler".

5.5.5 Requisitos não funcionais

Tal como expectado na definição de uma arquitetura, a arquitetura do projeto encontra-se estruturada de modo a atingir os requisitos não funcionais definidos no capítulo anterior. Assim sendo, estes requisitos foram atingidos da seguinte forma:

- **Desempenho:** garantir que a utilização da ferramenta de construção e simulação ocorre de uma forma rápida e fluída é um requisito fundamental para o projeto. Para tal, este requisito foi garantido graças à definição desta ferramenta em módulos Java*script*. Assim, todas as interações resultantes da atividade serão mais rápidas visto que não é necessário consultar o servidor para as desempenhar. Assim, este requisito depende apenas da otimização do código desenvolvido para a mesma.
- **Segurança:** de modo a garantir este requisito recorreu-se às funcionalidades disponibilizadas pelo Moodle. A partir da análise da documentação disponível, conclui-se que este requisito poderia ser alcançado seguindo as recomendações fornecidas pela mesma[21]. Por exemplo, através do método "require_login()" é possível realizar o processo de autenticação do utilizador no sistema e garantir que o acesso a uma página do sistema não está a ser efetuado por uma entidade externa ao Moodle.
- **Usabilidade:** de forma a garantir este requisito seguiram-se as normas de Nielson de forma a conceber os protótipos da ferramenta seguindo os seguintes aspetos:
    - **Visibilidade do sistema –** procurou-se definir estes protótipos de modo que o estudante esteja consciente do que está a acontecer no sistema através da utilização mensagens especificas para cada momento da utilização;
    - **Correspondência entre o sistema e o "mundo real" –** um fator que se teve em consideração foi desenhar a ferramenta de modo a que o estudante se sentisse o mais familiarizado possível com os conceitos que lhe são apresentados ao longo do percurso de utilização. Exemplo disso, é a representação dos operadores lógicos via textual;

- **Consistência e normas –** a necessidade da ferramenta apresentar os seus conteúdos de forma consistente foi outro aspeto tido em conta na sua definição. Assim, procurou-se manter essa consistência na apresentação dos conteúdos (não ter designações diferentes para o mesmo conceito) e nas operações a realizar para a construção do fluxograma;
- **Prevenção de erros –** de forma a atingir os objetivos definidos e proporcionar uma boa experiência de utilização da ferramenta aos estudantes, foi importante minimizar os possíveis erros que poderiam advir da sua utilização. Um exemplo disso, foi a apresentação de mensagens de erro na fase de construção dos blocos de modo a prevenir a maior parte dos erros na fase de animação.
- **Reconhecer em vez de memorizar -** procurou-se estruturar a ferramenta de modo a que o estudante pudesse observar todas as informações necessárias para a construção do algoritmo a qualquer momento;
- **Flexibilidade e eficácia na utilização –** idealizou-se a ferramenta tanto para utilizadores leigos como para utilizadores mais experientes. Assim, a integração de atalhos como "Ctrl + C" e "Ctrl + V" possibilitaram acelerar o processo de construção do fluxograma proporcionando uma melhor experiência a utilizadores com maior experiência.
- ***Design* estético e minimalista** – de forma a não causar um grande ruído visual ao estudante, procurou-se definir a ferramenta o mais simplista possível. Como tal, procurou-se não utilizar uma grande conjugação de cores de modo a não promover a tentativa de associação das mesmas a conceitos. Também certificou-se que toda a informação exposta no ecrã era simples, direta e continha apenas o essencial.
- **Ajudar os utilizadores a reconhecer, diagnosticar e recuperar erros –** considerou-se relevante apresentar ao estudante mensagens de erro de modo a ele detetar mais facilmente os erros que cometeu.
- **Documentação e Ajuda –** nunca é demais fornecer meios alternativos para o estudantes, caso pretendam, possam obter mais informação relativamente as fases associadas à utilização da ferramenta. Para tal, é necessário que o acesso a esta informação seja fácil, direto e que esteja disponível a qualquer altura ao estudante. Para tal, introduziu-se um botão de ajuda em cada menu de construção que reencaminha automaticamente o estudante para o menu de ajuda referente à instrução onde acionou o evento.

- **Confiabilidade:** através do módulo "deteção de erros" conseguiu-se garantir toda a veracidade das respostas dadas pelo sistema no processo de construção e simulação dos algoritmos.

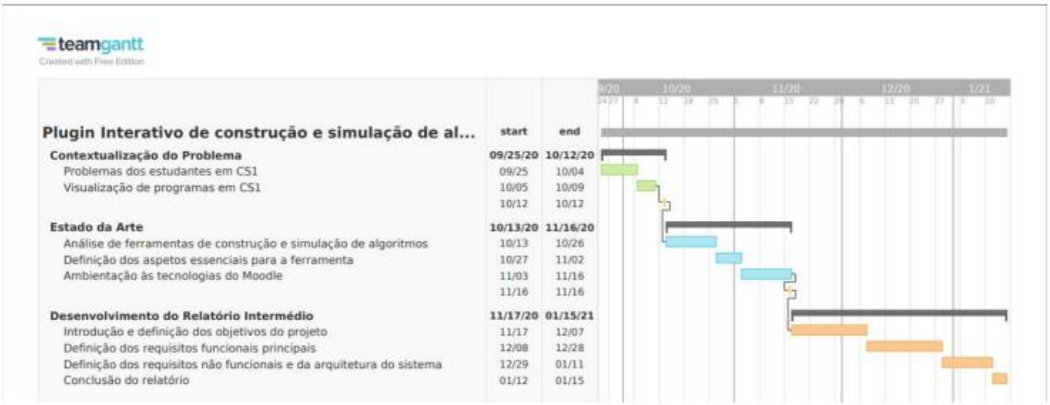
5.6 Planeamento

5.6.1 1º Semestre

As tarefas desempenhadas no primeiro semestre tiveram um carater mais teórico. Estas tarefas podem ser divididas em três temáticas: contextualização do problema, estado da arte, desenvolvimento do relatório intermédio. As metas realizadas neste primeiro semestre foram definidas em reuniões quinzenais entre os membros da equipa. As reuniões tinham a finalidade de avaliar o trabalho desenvolvido durante este período, onde eram sugeridas alterações aos aspetos incorretos do trabalho realizado, e, posteriormente, eram definidas as metas para a quinzena seguinte.

Inicialmente, começou-se por realizar um processo de aquisição de conhecimentos sobre as temáticas que envolvem a proposta deste documento, nomeadamente, no desempenho dos estudantes em cursos de programação e na influência da visualização de programas na compreensão dos conceitos de programação por parte dos mesmos. Após a obtenção destes conhecimentos procedeu-se à utilização e análise de várias ferramentas de construção e simulação de algoritmos que, consequentemente, permitiu a identificação de alguns aspetos essenciais para integrar na ferramenta deste documento. De seguida, foi feita uma ambientação às tecnologias utilizadas no Moodle, onde se procurou perceber como é que um *plugin* é desenvolvido nesta plataforma. Por fim, procedeu-se à escrita do relatório intermédio. Nesta última temática, iniciou-se pela escrita da introdução e definição dos objetivos do projeto. Depois, definiu-se os requisitos funcionais principais da proposta. Em último lugar, estabeleceram-se os requisitos não funcionais e a arquitetura do sistema. A seguir, encontra-se apresentado um diagrama de Gant que espelha a atividade desenvolvida neste primeiro semestre (fig. 5.8). As várias etapas realizadas neste semestre foram definidas em reuniões quinzenais entre os membros do grupo de trabalho, onde eram discutidos os aspetos referentes às tarefas realizadas nessa quinzena e o planeamento dos tópicos a serem abordados na quinzena seguinte.

Figura 5.8 - Diagrama de Gantt que ilustra o planeamento do 1º Semestre.



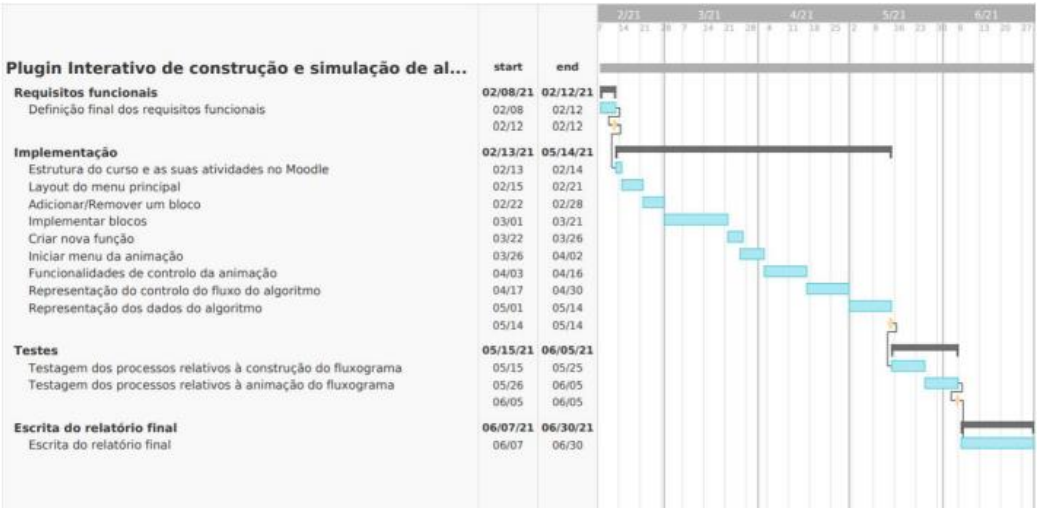
5.6.2 2º Semestre

No segundo semestre, o trabalho desenvolvido foi de cariz mais prático. A partir do lavor realizado no primeiro semestre, seguiu-se com a implementação do software e a sua respetiva testagem. Para tal foi necessário definir um modelo de planeamento para esta fase. O modelo de planeamento definido foi o Feature-Driven development (FDD). Este modelo representa uma estratégia ágil e iterativa de desenvolvimento de *Software* que foca na apresentação incremental das funcionalidades ao cliente. Desta forma, à medida que as funcionalidades eram expostas permitia obter feedback em relação às mesmas dando a conhecer aspetos a melhorar e a corrigir ao longo do processo (fig. 5.9). Através deste modelo, foi possível desenhar o planeamento segundo a prioridade das funcionalidades necessárias para o projeto. Assim o processo de desenvolvimento pôde ser adaptado às circunstâncias resultantes do processo e permitiu definir metas consoante o desenvolvimento das funcionalidades ia ocorrendo.

Figura 5.9 - Ilustração das fases de processo da metodologia FDD.

A seguir, encontra-se apresentado um diagrama de Gant que espelha a atividade desenvolvida neste primeiro semestre (fig. 5.10). As várias etapas realizadas neste semestre foram definidas em reuniões quinzenais entre os membros do grupo de trabalho, onde eram discutidos os aspetos referentes às tarefas realizadas nessa quinzena e o planeamento dos tópicos a serem abordados na quinzena seguinte.

Figura 5.10 - Diagrama de Gantt que ilustra o planeamento do 2º Semestre.



Capítulo 6
Implementação

Neste capítulo, apresentar-se-á o modo como o projeto foi implementado pelo autor. Desta forma, inicia-se este capítulo com a configuração do sistema Moodle que permitiu o desenvolvimento posterior do *plugin* proposto no projeto. Antes de se iniciar essa apresentação dos aspetos específicos do projeto, encontra-se uma pequena introdução a aspetos relativos à estruturação do código no Moodle e regras necessárias para a implementação de um plugin neste sistema. Refere-se também condições impostas pelo Moodle na criação de um *plugin* atividade que é o tipo de *plugin* utilizado para a integração da ferramenta no sistema Moodle.

6.1 Configuração do sistema Moodle

Antes de iniciar a implementação do *plugin*, foi necessário realizar a configuração do ambiente. Tal como foi definido no capítulo 4, o Moodle e a base de dados deveriam estar alocados em endereços diferentes.

Na máquina com o endereço 10.3.3.107 encontra-se a base de dados, necessitou-se de instalar um servidor postgres. A versão do postgres escolhida foi a 12. Após esta instalação, necessitou-se de criar um utilizador e replicar a estrutura da base de dados do Moodle para o servidor. Para tal, seguiram-se os passos presentes na documentação do Moodle relativo a este tópico[22].

A instalação do Moodle foi efetuada na máquina alocada no endereço 10.3.3.151. Para a sua instalação seguiram-se os passos indicados na documentação do Moodle relativos este processo[12].

Por fim, foi necessário estabelece a ligação entre as duas máquinas de modo a possibilitar realizar alterações na base de dados. Para tal, foram necessários os seguintes passos:

- Permitir o tráfego no porto 5432. Este porto está designado para realizar comunicações entre a base de dados.
- Na máquina onde se encontra a base de dados, permitir o tráfego a partir do endereço 10.3.3.151 no porto 5432. Para tal utilizou-se o seguinte comando:

```
ufw allow from 10.3.3.107 proto tcp to 10.3.3.151 port 5432
```

- Na máquina onde se encontra a base de dados, permitir o tráfego a partir do endereço 10.3.3.107 no porto 5432. Para tal utilizou-se o seguinte comando:

```
ufw allow from 10.3.3.151 proto tcp to 10.3.3.107 port 5432
```

Para ativar as regras da firewall foi necessário executar o comando

ufw enable

De seguida, encontra-se disponibilizada uma imagem que indica os endereços onde cada umas das componentes acima referidas estão alocadas (fig. 6.1).

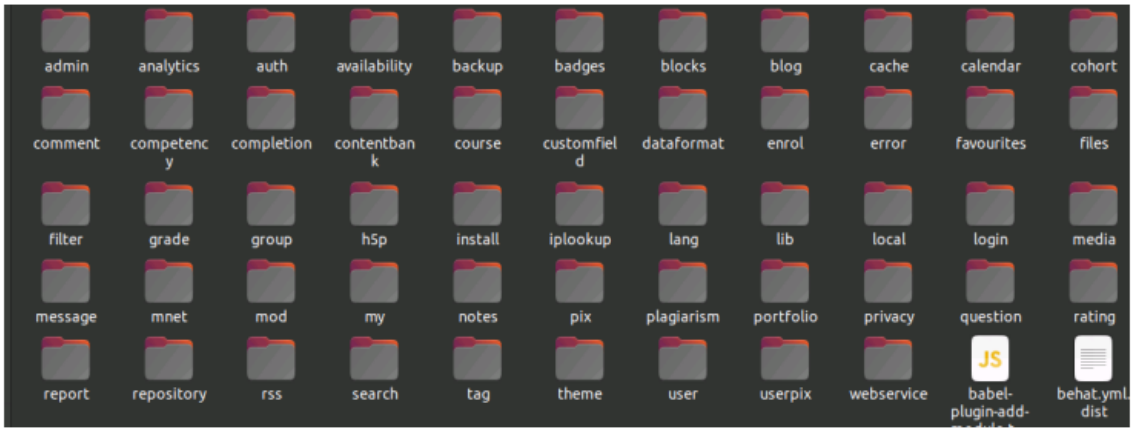
Figura 6.1 - Comunicação entre as redes que englobam o sistema Moodle do projeto.

De modo a que o processo de implementação respeitasse as regras sintáticas impostas pelo Moodle, utilizou-se o ambiente phpStorm que possui uma configuração que já impõe essas regras ao desenvolvedor e o notifica caso não as esteja a seguir [23].

6.2 Estrutura do código do Moodle

O código de um projeto Moodle encontra-se estruturado de uma forma bastante simples e fácil de compreender para os programadores que desejam iniciar a desenvolver funcionalidades para a plataforma. Como se pode observar a partir da figura 6.2, o projeto Moodle está dividido em várias pastas, sendo que cada uma delas representa um tipo de funcionalidade que o sistema fornece [4]. Por exemplo, a pasta com o nome "mod" integra todas as funcionalidades relativas às atividades do Moodle e a "lib" todas as bibliotecas existentes no núcleo do sistema.

Figura 6.2 - Estrutura do código do Moodle.



Posto isto, sempre que quisermos adicionar um *plugin* ao sistema, é apenas necessário criar uma pasta com o nome do *plugin* que se pretende desenvolver e introduzi-la numa destas pastas, mediante o tipo de funcionalidade que este *plugin* vai estender. O nome do *plugin* que se pretende adicionar terá de ser acompanhado com um prefixo relativo ao tipo do *plugin*. Por exemplo, um *plugin* que vá estender uma funcionalidade do tipo bloco, e que se chamará "exemplo" terá o nome de "bloco_exemplo". Este nome

compõe o nome Frankenstyle do *plugin* que consiste num identificador específico e convencional para o *plugin* (tabela 6.1).

Tabela 6.1 - Exemplo da relação entre o tipo de *plugin* e o nome que lhe deve ser atribuído.

Tipo do Plugin	Nome do Plugin	Frankenstyle	Caminho
Bloco	exemplo	bloco_exemplo	blocks/exemplo

6.3 Visão geral do *plugin*

Tal como foi referido no capítulo da definição dos requisitos, para alcançar os objetivos deste projeto foram necessárias definir um conjunto de funcionalidades. Tendo em conta essas funcionalidades e as caraterísticas do projeto, definiu-se que o tipo de *plugin* do Moodle onde estes propósitos seriam alcançados era através de um *plugin* atividade. Desta forma houve a necessidade de compreender os requisitos impostos para a criação de um *plugin* deste tipo recorreu-se à do Moodle relativamente a este tópico [24].

Nesta secção irá ser feita a apresentação do código desenvolvido na fase de implementação. Esta apresentação consistirá na demonstração do modo como o código está organizado juntamente com uma descrição das funcionalidades de cada uma das suas componentes.

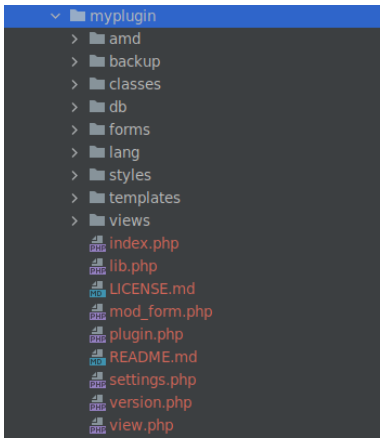
6.3.1 Estrutura do *plugin*

Na seguinte figura 6.4 encontra-se a organização do código do *plugin*. Como é observado, é possível identificar os seguintes componentes utilizados na implementação do *plugin*:

- **amd –** pasta que contém os módulos criados para a utilização da ferramenta de construção e simulação de fluxogramas.
- **db –** pasta que engloba uma série de aspetos relativos à base de dados do *plugin*. Esta possui os ficheiros que realizam o tratamento de instalação da base de dados do plugin na base de dados do Moodle (install.xml), atualização (upgrade.php), desinstalação (unnistall.php) e a definição das permissões dos utilizadores ao uso das funcionalidades do *plugin* (access.php);
- **forms –** pasta que possui os *forms* utilizados para o processo de criação, edição e remoção de um exercício. Estes *forms* são representados através de classes que estendem da Form API fornecida pelo Moodle;
- **lang –** pasta que contem as *Strings* responsáveis pela linguagem utilizada no *plugin.*
- **styles –** pasta que armazena os estilos definidos para os componentes do *layout* da ferramenta de construção e simulação de algoritmos;

- **templates –** pasta que contém o *template* utilizado pelo Moodle para a renderização do layout da ferramenta de construção e simulação de algoritmos;
- **views –** possui os ficheiros responsáveis pela apresentação das páginas do *plugin* aos seus utilizadores;
- **index.php –** ficheiro responsável por listar todas as instâncias existentes no *plugin* e que estão associadas a um determinado identificador de curso onde a atividade está inserida.
- **mod_form.php –** permite importar elementos a serem adicionados às instâncias do *plugin* na execução de um *form.*
- **plugin.php –** representa uma classe que gere a atividade do *plugin*, nomeadamente as informações que são apresentadas ao longo da utilização do mesmo;
- **view.php –** ficheiro responsável pela definição do *layout* da página e definição dos URLs das páginas do *plugin.* Esta importa as páginas presentes na pasta "views" de acordo com as funcionalidades permitidas pelo utilizador em questão.

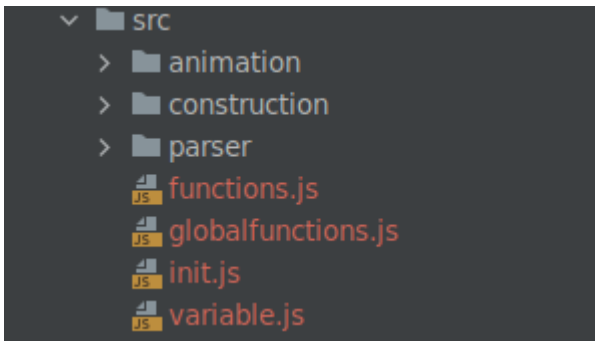
Figura 6.3 - Estrutura do código do *plugin* do projeto.



6.3.2 Ferramenta de construção e simulação de algoritmos

De seguida encontra-se apresentada uma visão geral da estrutura dos módulos responsáveis pela gestão da ferramenta de construção e simulação de algoritmos.

Figura 6.4 - Estrutura do código da ferramenta de construção e simulação de algoritmos.



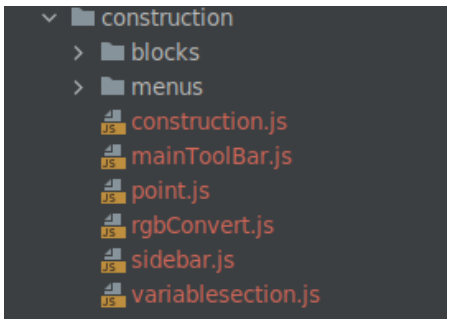
Analisando a figura 6.6, é possível observar o modo como o código se encontra organizado no projeto. A partir da sua visualização é possível identificar as seguintes componentes:

- **animation** – pasta que possui as funcionalidades relativas à gestão do menu de animação.
- **construction** – pasta que possui as funcionalidades relativas à gestão do menu de construção.
- ***parser*** – armazena todos os *parsers* utilizados na ferramenta.
- **functions.js** – ficheiro responsável pela definição de todas as funções auxiliares aceites no programa e utilização das mesmas no programa. Esta permite fornece uma série de funcionalidades ao sistema relativas a este tópico. Esta fornece a capacidade de verificar se uma função existe no programa, verifica se os parâmetros são do mesmo tipo que os parâmetros que a função recebe e, por fim, realiza a operação de cada função. A lista de funções que se podem utilizar na ferramenta encontra-se disponível em anexo.
- **globalfunctions.js** – armazena um conjunto de funcionalidades que vários ficheiros utilizam no projeto e que não podem ser atribuídas a um determinado objeto do programa. Entre elas encontram-se funcionalidades como verificar os tipos de input, tratamento da exibição dos menus de implementação da ferramenta e a função que permite ao utilizador posicionar esses menus no ecrã;
- **init.js** – ficheiro responsável pela inicialização da ferramenta e que é utilizado na chamada do módulo na parte do servidor.
- **variables.js** - contém os objetos que representam uma variável no sistema e armazena todas as variáveis criadas no processo de construção, tal como os procedimentos relativos à sua gestão. Assim sendo, neste ficheiro estão disponibilizadas as seguintes funcionalidades: procura de uma variável, adição de uma variável, remoção de uma variável, edição de uma variável, a leitura de variáveis a partir de um ficheiro JSON e a conversão das variáveis do programa para o formate JSON.

6.3.2.1 Construção do fluxograma

A partir dos requisitos que se definiram no capítulo 3, dividir este processo em duas partes: construção do fluxograma e a implementação dos blocos do mesmo.

Figura 6.5 - Estrutura do código responsável pela construção do fluxograma.

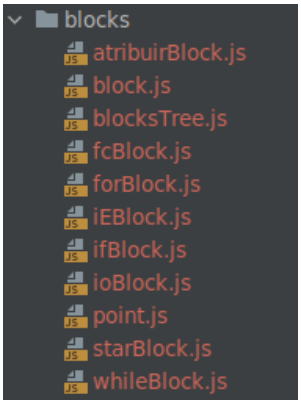


A partir da figura 6.7, podem-se observar os seguintes componentes:

- **blocks –** pasta que contém os objetos responsáveis pela construção da imagem dos blocos e todas as funcionalidades incorporadas por cada um deles no momento da animação;
- **menus –** pasta que contém os objetos responsáveis pela construção dos menus de implementação dos blocos e do menu ajuda, tal como as funcionalidades inerentes a cada um;
- **construction.js –** ficheiro que engloba todas as funcionalidades responsáveis gestão do fluxograma. Nele estão inseridas as funções responsáveis pela definição dos processos de adição, remoção e edição dos blocos do fluxograma. Para além disso, este controla toda a representação gráfica do fluxograma durante a construção do mesmo. Por fim, este também ficou encarregue de realizar o processo de cópia de blocos do fluxograma e a colagem dos mesmos.
- **mainToolBar.js** – ficheiro que contém todas as funcionalidades existentes na barra de controlo presente no menu de construção. Desta forma, neste ficheiro encontram-se as funcionalidades que permitem ao sistema importar informação de uma atividade a partir de um ficheiro JSON, guardar informação de uma atividade num ficheiro JSON e obter o pseudocódigo do fluxograma da atividade.
- **sideBar.js** – ficheiro que contém todas as funcionalidades existentes na barra lateral presente no menu de construção.
- **variableSection.js** – ficheiro que gere todas as funcionalidades existentes na secção das variáveis do menu de construção no que se refere à sua apresentação gráfica destas componentes. Assim, este é responsável pela gestão dos processos de adição, remoção e edição das variáveis. Para além disso, fornece a possibilidade de o utilizador ordenar as variáveis presentes na secção em função do seu tipo.

Após a introdução às funcionalidades que compõe a atividade de construção do fluxograma, necessita-se abordar os elementos que este processo utiliza. Tal como foi referido anteriormente, a pasta “blocks” possui os ficheiros responsáveis pelo desenho dos blocos e dos seus elementos para posteriormente poderem ser apresentados ao utilizador.

Figura 6.6 - Organização da pasta "blocks" presente no código da ferramenta.



Através da observação da figura 6.8, são possíveis observar vários ficheiros, muitos deles com semelhanças na sua terminação. Para a compreensão da estruturação do código relativo à construção dos blocos e gestão de informação do fluxograma, considera-se importante destacar os seguintes ficheiros:

- **block.js** – ficheiro que possui todas as funcionalidades comuns a todos os blocos, tanto a nível de apresentação como na configuração da sua animação. Os restantes ficheiros que possuem a terminação "Block" descendem deste ficheiro. Nesses ficheiros são realizadas as adaptações na apresentação e na animação mediante o bloco que representam. Ao nível da apresentação, este bloco possui as funcionalidades de criar o elemento e de dispor da informação que ele possui. Ao nível da animação este possui todas as funcionalidades existentes no processo da animação dos blocos. Estas são as seguintes:
  - Iniciar a animação;
  - Configurar animação;
  - Destaque do bloco;
  - Destaque da secção;
  - Destaque dos elementos de ligação do bloco;
  - Apresentação da expressão na secção da representação do fluxo de controlo;
  - Configuração da animação das operações existentes na expressão;
  - Formatação das componentes a serem apresentadas na expressão (variáveis, funções, …);
  - Atualização do valor de uma variável;
  - Utilização de um variável numa operação;
  - Limpeza das secções depois das operações;
  - Aceder e criar a *fila de espera* da animação;
  - Representar o bloco em pseudocódigo.
- **blockTree.js** – ficheiro que engloba todas as funcionalidades referentes à gestão da informação interna do fluxograma. Estas funções estão encarregues de proporcionar as seguintes valências ao programa:
  - Inserir um bloco na estrutura que armazena o a informação do fluxograma;
  - Remover um bloco na estrutura que armazena o a informação do fluxograma;
  - Pesquisar por um bloco no fluxograma e verificar se ele existe ou não;

- Verificar se uma variável já foi inicializada no momento em que um determinado bloco foi definido;
- Traduzir o fluxograma para pseudocódigo;
- Traduzem a estrutura de um ficheiro JSON para um fluxograma;
- Traduzem a estrutura de um fluxograma para um ficheiro JSON;
- Verificam se o fluxograma cumpre as condições para ser animado.

De forma a concluir o processo de construção do fluxograma, necessita-se de ser expor os menus de implementação dos blocos e de auxílio a esse processo. A implementação desses menus encontra-se situada na pasta “menus” assim como foi mencionado anteriormente.

Figura 6.7 - Organização da pasta "menus" presente no código da ferramenta.

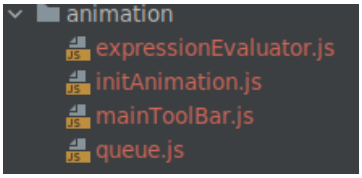
Com base na figura 6.9, podemos observar uma série de ficheiros presentes nesta pasta. A lógica de associação destes ficheiros é semelhante à dos blocos. Dentro destes ficheiros considera-se relevante destacar um que serve de base para a implementação de todos os outros. Esse ficheiro é responsável por fornecer os métodos para a criação da estruturação principal do menu e para a verificação dos valores introduzidos pelo utilizador durante a implementação do bloco. Esta verificação é realizada apenas se o utilizador confirmar o processo. Relativamente a esta última vertente destaca-se as seguintes funcionalidades:

- **Verificação dos valores** – faz a identificação dos *tokens* permitindo ao sistema perceber se o valor em causa se trata de uma função, de uma variável, de um número ou de uma *String;*
- **Verificação de uma variável –** verifica se um dado valor passado como variável existe no programa;
- **Verificação dos parâmetros de uma função –** verifica todos os aspetos relativos à chamada de uma determinada função, ou seja, se o número de parâmetros passados é o requerido pela mesma e se os tipos dos mesmos correspondem aos que são requeridos pela função.

6.3.2.2 Animação do fluxograma

Quando o utilizador decidir que pretende visualizar este necessita de inicializar o menu de simulação. Nesta parte da ferramenta, necessitou-se implementar todas as operações que lhe permitissem controlar a informação de acordo com a sua vontade.

Figura 6.8 - Estrutura do código responsável pela animação do fluxograma.



A partir da observação figura 6.10 é possível constatar a existência dos seguintes ficheiros:

- **expressionEvaluator.js** – ficheiro responsável pela avaliação das expressões lógicas e aritméticas executadas no algoritmo tal como a configuração da animação do respetivo processo;
- **initAnimation.js** - ficheiro responsável pela inicialização do menu de animação;
- **mainToolBar.js –** ficheiro que possui todas as atividades relativas à gestão da animação por parte do utilizador. Desta forma este possui as funcionalidades de iniciar, pausar, parar e executar um passo da animação. Por fim, também ao utilizador definir a velocidade à qual a animação irá decorrer;
- **fila de espera.js –** ficheiro que cria o objeto "fila de espera" que representa uma fila de espera para ser utilizada na configuração e no desenrolar da animação.

6.4 Visão de alto nível

6.4.1 Base de dados

Segundo os objetivos definidos no primeiro capítulo, o *plugin* devia permitir a disponibilizar um conjunto de exercícios para o estudante resolver e uma lista de submissões das resoluções efetuadas pelos mesmos. Assim sendo, de forma a conseguir armazenar informação que permitisse gerir estas duas funcionalidades e todas as outras relacionadas com estas definiram-se as seguintes tabelas para a base dados do *plugin*:

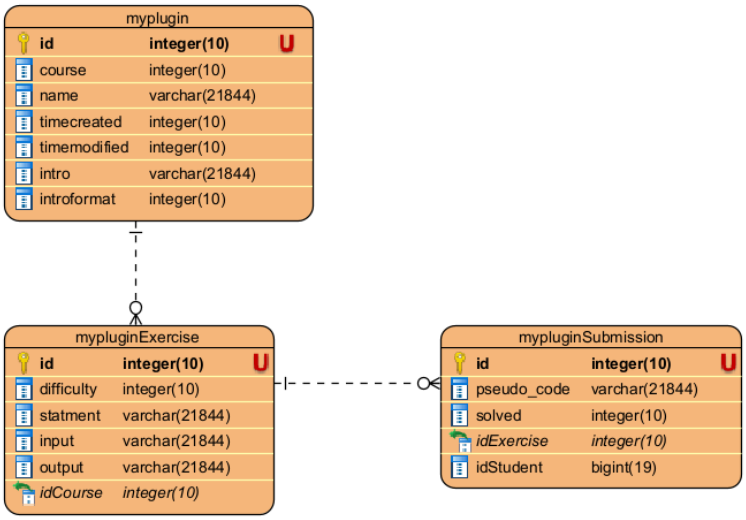
- **my*plugin* –** tabela que armazena as atividades criadas pelo *plugin* no Moodle.
  - **id –** id associado à atividade;
  - **course –** id do curso onde a atividade foi criada;
  - **name –** nome da atividade;
  - **timecreated –** data de quando à instância foi adicionada ao curso;
  - **Timemodified –** data da última modificação à instância;
  - **intro –** descrição da atividade
  - **introformat –** formato do campo de introdução
- **my*plugin*Exercise** – tabela responsável por armazenar os exercícios disponíveis na atividade.
  - **id –** identifica o exercício;
  - **difficulty –** nível de dificuldade do exercício;
  - **enunciado –** enunciado do exercício;
  - **input –** exemplos de valores iniciais para o exercício;

- **output –** exemplos de valores de retorno para o exercício;
- **idCourse –** id do curso no qual o exercício está inserido.

- **my*plugins*submission** – tabela que contém todas as resoluções submetidas pelos estudantes na atividade.
  - **id –** identifica a submissão;
  - **pseudo_code –** pseudocódigo resultante do fluxograma submetido pelo estudante;
  - **solved –** *status* que verifica se o exercício foi resolvido com sucesso ou não.
  - **idExercise –** id que associa a submissão ao respetivo exercício;
  - **idStudent** – id que associa o estudante à submissão.

De seguida, encontra-se o esquema da base dados que foi definida para o *plugin* onde estão ilustradas as relações entre as várias entidades que a constituem (fig. 6.11).

Figura 6.9 - Diagrama ER da base dados do *plugin* do projeto.



6.4.2 Gestão da informação utilizada na ferramenta

A partir da análise da introdução geral do *plugin* à ferramenta de visualização, foi possível identificar quatros tipos de estruturas fundamentais para o funcionamento da mesma:

- **Variáveis;**
- **Funções;**
- **Menus;**
- **Blocos;**
- **Queue**

De forma a controlar a informação que circula no programa, recorreu-se à criação de classes para representarem estes objetos. A opção por esta definição das variáveis deveu-se principalmente ao facto de possibilitar a utilização de herança entre os objetos, nomeadamente entre os menus e os blocos. De seguida, será feita uma

descrição dos atributos que cada uma destas classes possui e que informação cada um deles representa no objeto.

Para realizar a gestão das variáveis considerou-se necessário definir um conjunto que propriedades de modo a permitir ao sistema categorizá-las em relação a essas propriedades para posteriormente interpretar se uma ação pode ser executada. Desta forma, definiram-se os seguintes atributos para esta classe (tabela 6.2):

- **name –** armazena o nome da variável;
- **type –** indica o tipo da variável que está a ser utilizada, ou seja, verifica se esta é do tipo numérico ou do tipo *String;*
- **isArray –** indica se a variável é um array ou não. De acordo com os requisitos definidos, esta propriedade apenas pode ser verdadeira caso o tipo da variável for numérico.
- **length –** propriedade que indica o tamanho de uma variável. Tal como a anterior, para a atribuição desta propriedade é necessário preencher um requisito. Ou seja, para esta ser atribuída a variável em causa necessita de ser um *array*.
- **value –** propriedade que armazena o valor que é atribuído à variável durante a animação para posteriormente poder ser utilizada pelo sistema caso seja requerida numa operação durante a animação.

Tabela 6.2 - Atributos da classe que representa um variável no programa.

Atributo	Tipo
name	*String*
type	"num" ou "String"
isArray	true ou false
length	Número inteiro
value	Número, *String* ou array de numéricos

Relativamente à classe responsável pela criação de um objeto que representa uma função auxiliar no programa, a definição dos seus atributos teve um intuito semelhante à gestão das variáveis. Este possui os seguintes atributos (tabela 6.3):

- **name –** armazena o nome da função auxiliar;
- **type –** tipo de valor que a função retorna;
- **parameters –** array que possui os tipos de parâmetros ordenado. É fundamental para realizar a verificação se a utilização de uma função no processo da implementação dos blocos respeita as características da função.
- **pValues –** semelhante ao atributo, porém, em vez de armazenar o tipo das variáveis, guarda o valor de cada parâmetro. Este é importante na execução da animação.

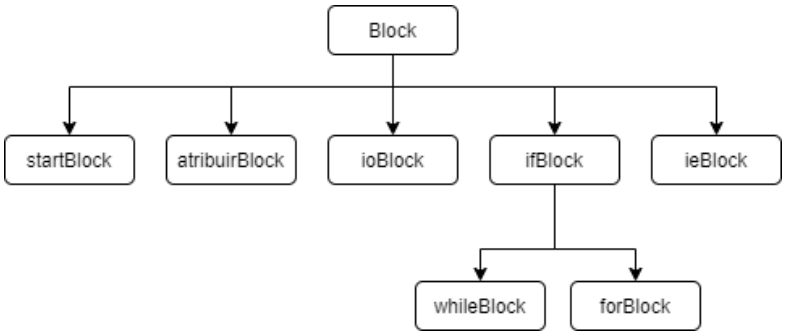
Tabela 6.3 - Atributos da classe que representa uma função no programa.

Atributo	Tipo
name	*String*
type	"num" ou "String"
parameters	array com valores composto por três tipos de valor: 'num', 'String', 'array'

pValues	Número inteiro

No que se refere à criação do objeto responsável por representar um bloco do fluxograma, a aplicação dos conceitos de herança teve um papel importante na fase de implementação. A sua aplicação foi fundamental de modo a permitir a reutilização de funcionalidades comuns aos vários blocos. De seguida, encontra-se uma figura 6.12 que demonstra a estrutura hierárquica na qual os blocos se encontram organizados.

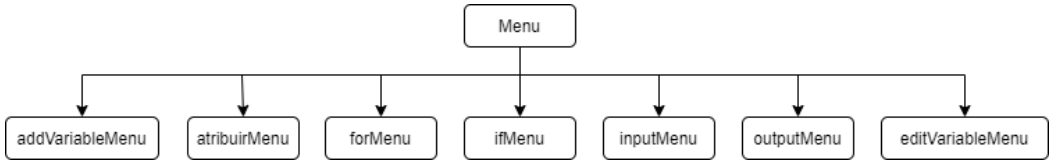
Figura 6.10 - Representação da estrutura hierárquica entre os blocos existentes no programa.



A partir desta figura, consegue-se observar que grande parte das classes estendem da classe “Block”. As classes “whileBlock” e “forBlock” têm como pai a classe “ifBlock” devido ao facto de estas possuírem a mesma representação dos seus elos de ligação.

Por fim, resta-se abordar as classes responsáveis pela criação dos menus de implementação e de suporte presentes na ferramenta. Cada classe possui um *layout* específico que é definido no processo de criação do mesmo em cada uma. Este processo de criação aproveita a estrutura base fornecido pelo pai. Para além disso, todas as funcionalidades de gestão de atividade dos blocos encontram-se disponíveis na classe “Menu”.

Figura 6.11 - Representação da estrutura hierárquica entre os menus existentes no programa.



6.4.3 Deteção de erros

A deteção de erros foi uma parte fundamental no funcionamento da ferramenta, principalmente na fase de construção do algoritmo. Como um dos objetivos a alcançar era implementar fase de construção de modo que o estudante produza o menor número de erros, foi necessário construir uma série de funcionalidades que analisasse essa fase e reporta-se ao mesmo qualquer erro que tenha cometido de forma clara e precisa. Deste modo, permitiu-se que este corrija o seu erro de forma imediata visto que este não terá que rastrear essa incorreção já que o programa realiza esse processo por ele. Assim sendo, foi necessário realizar o tratamento de erros relativamente a três aspetos:

- **Definição de variáveis**;
- **Implementação dos blocos**;
- **Verificações na fase de animação**.

6.4.3.1 Variáveis

No processo de criação de variáveis, estes devem respeitar um número de condições de modo a ser permitido pelo sistema referidas no capítulo de descrição da ferramenta. Desta forma, a partir das condições acima referidas, o nome de uma variável deve respeitar a seguinte expressão regular:

*[a-z] [a-zA-z0-9_] **

De forma a executar a validação dos nomes introduzidos pelo utilizador recorreu-se à avaliação da expressão regular acima indicada através criação de um objeto "RegExp". Este objeto é fornecido pelo Java*script* que permite aos programadores verificar se uma String respeita as regras de uma determinada expressão regular [25]. Após ser realizada a validação do nome é feita a verificação para se ver se o nome introduzido não está associado ao nome de nenhuma função do sistema. Relativamente ao impedimento da criação de *arrays* de *Strings,* foi controlado através da apresentação do menu ao utilizador. Ou seja, mediante o tipo de variável selecionado pelo utilizador é-lhe exibida a opção para definir a variável como *array* e indicar o seu tamanho. Por fim, o sistema verifica se o utilizador preencheu todos os campos necessários para a definição da variável. No que se refere ao processo de edição e eliminação de variáveis, foi necessário realizar mais algumas verificações de modo a impedir que erros indesejados passassem para o lado da animação. Assim sendo, nestes dois processos foi implementada uma pesquisa que deteta a utilização das variáveis no fluxograma. No caso da edição o valor é alterado nos blocos onde a variável é utilizada enquanto se se tratar de uma remoção, o programa elimina todos os blocos que utilizassem a variável seleciona para este processo.

6.4.3.2 Implementação dos blocos

A deteção de erros na fase de implementação revelou ser um processo mais complexo em relação ao das variáveis. Esse facto foi inevitável dado o conjunto de nuances existentes neste processo. Isto acontece graças ao facto de o procedimento de implementação ser diferente para cada instrução. Contudo, apesar das distinções existentes entre os vários blocos, todos, à exceção do bloco "Ler", executam os mesmos passos para validar a sua implementação. Em primeiro lugar, foi necessário executar a validação das expressões aceites em cada um dos blocos. Para tal, foi necessário arranjar uma solução que permitisse avaliar cada conteúdo da expressão individualmente. Tendo em consideração as especificações de cada um desses procedimentos, sentiu-se a necessidade de desenvolver três *parsers*:

- **assign*Parser*** **-** utilizado para avaliar a sintaxe de expressões que correspondam a atribuições;
- **output*Parser*** **-** utilizado para avaliar a sintaxe de expressões do bloco "Escrever" do programa;

- **logic*Parser* –** utilizado para avaliar a sintaxe de expressões lógicas.

Para a criação destes *parsers*, importou-se uma biblioteca externa designada por PeggyJS [26]. Esta permite definir uma gramática e avaliar se uma expressão respeita essa gramática ou não. Após essa validação, esta é capaz de retornar uma *String* com informação em relação à expressão analisada. Através desta funcionalidade, foi possível construir uma *String* que incluísse todos os *tokens* utilizados numa determinada expressão. A utilização desta tecnologia deveu-se ao facto de se ter considerado ser uma alternativa menos dispendiosa em relação à criação de um compilador próprio para este processo de validação das expressões. Assim, apenas foi necessário definir as gramáticas para cada um destes *parsers* e formatar o retorno de cada um deles de modo a facilitar o processo posterior de validação de cada um desses *tokens* (as gramáticas definidas encontram-se em anexo). Após a validação das expressões, o sistema necessita de verificar os *tokens* resultantes da análise da expressão por parte do *parser.* A partir desta análise, pretendeu-se verificar o seguinte:

- Caso se trate de uma atribuição, verifica se os tipos dos *tokens* utilizados na expressão coincidem com o tipo da variável à qual está a ser feita a atribuição;
- verificar se as operações são efetuadas com valores do mesmo tipo.

Para verificar estes dois pontos, é necessário detetar o tipo de cada *token* foi necessário identificar o que é este representava no sistema. Assim, cada *token* é analisado para determinar se se trata de uma String, de uma variável, de um número ou de uma função. Se for variável ou função, o sistema efetua verifica se esta existe e retorna o seu tipo de modo a realizar a verificação. Avaliando os *tokens* individualmente, foi possível dar mensagens de erro especificas para o estudante. Em anexo encontram-se disponíveis as mensagens de erro que podem ser produzidas durante esta fase.

6.4.3.3 Animação

Apesar de o objetivo era reduzir ao máximo o número de erros possíveis durante a fase da animação, alguns dos cenários eram impossíveis de prever na fase de construção. Assim, durante a animação o sistema é capaz de informar o utilizador relativamente a quatros:

- introdução de um valor não condizente com o tipo da variável que a variável pedida numa instrução do tipo “Ler”;
- Tentativa de aceder a um índice que não existe num *array;*
- Tentativa de aceder a um valor num *array* que ainda não foi inicializado.

A incapacidade de rastrear estes erros durante o processo de construção visto que o primeiro ponto resulta da atividade do utilizador com a ferramenta e os restantes pontos possuem a impossibilidade de prever todos os cenários possíveis no cálculo das expressões que irão resultar no valor para aceder ao valor do *array.* Por exemplo, tentar aceder a uma posição do *array* utilizando uma variável que foi inicializada a partir da instrução “Ler”.

6.4.4 Representação dos blocos

Na representação dos blocos, foram necessárias implementar uma série de funcionalidades que tratassem do seu aspeto visual. Para realizar essa representação, recorreu-se à tecnologia SVG. A opção da utilização desta tecnologia recaiu sobre o facto das imagens geradas por estes vetores serem capazes de manter a qualidade de apresentação quando estas sofrem alterações na sua dimensão. Para além disso, esta permite ao programador inserir elementos HTML dentro do elemento SVG. Desta forma, foi possível transportar a implementação dos blocos mais simples e aproveitá-la para a construção de blocos mais complexos com os de estruturas condicionais e cíclicas. Os blocos encontram-se estruturados da seguinte forma:

- **Uma divisão principal** – elemento html denominado por 'div' onde o desenho do bloco irá estar inserido. Desta forma, conseguiu-se facilitar a questões da centralização dos elementos no *layout* de construção.
- **Um SVG principal** – elemento SVG que engloba todos os elementos que constituem o desenho do bloco e que define a *viewBox* para os restantes elementos.
- **O bloco** – elemento SVG que possui o polígono que representa a imagem do bloco. Este polígono é construído a partir de um conjunto de pontos que possuem as coordenadas da *viewBox* para desenhar a figura.
- **O elo de ligação** – elemento SVG constituído pela linha que representa a ligação entre o próximo bloco.

Caso os blocos representem uma estrutura de condição ou cíclica, são adicionadas duas secções que representam a parte verdadeira e falsa da instrução. Primeiramente, esta parte apenas possui os elos de ligação que foram essa parte. Contudo, quando um bloco é adicionado a uma das partes, essa parte divide-se nas seguintes secções:

- **Lado superior** – elo de ligação entre o bloco principal e o bloco filho da parte superior;
- **Objeto estrangeiro** – representa o elemento que possui todos os blocos filhos e que foram inseridos neste bloco. Dentro deste elemento encontra-se uma "div" que contém os elementos dispostos de forma semelhante com o *layout* onde se encontra o fluxograma.
- **Lado inferior** - elo de ligação entre o bloco principal e o bloco filho da parte inferior;

Por último, todas estas secções que constituem a figura do bloco estão identificadas com o id do bloco de modo a permitir realizar as operações respetivas a cada uma na fase de construção.

6.4.5 Construção do fluxograma

A construção do fluxograma envolve uma série de tarefas. Para que estas funcionalidades estivessem disponíveis ao utilizador, foi necessário desenvolver vários aspetos que o permitissem. Quando o menu de construção é inicializado, é apresentado

um fluxograma inicial constituído por dois blocos, o inicial e o final. De forma a permitir a adição de um bloco foi necessário implementar:

- O evento *drag and drop* dos botões que representam as instruções presentes na barra de controlo principal do menu de construção;
- O evento para reconhecer que o elemento anterior se encontra num elo de ligação de um dos blocos do fluxograma. Este evento permite o destaque do elo de ligação caso o rato esteja a sobrepor a sua área de inserção e acione a função de adição do bloco do fluxograma.

Cada vez que um bloco é adicionado ao fluxograma, o sistema está encarregue de definir os seguintes eventos:

- **Destaque das áreas de inserção no processo de animação**
- **Adição de um novo bloco**
- **Destaque do bloco**
- **Iniciação do menu de implementação**

De seguida encontra-se a tabela 6.4 que apresenta os tipos de eventos que acionam as funcionalidades acima referida.

Tabela 6.4 - Eventos inseridos nos blocos do fluxograma.

Funcionalidade	Evento	Id do elemento
Destaque do bloco	click	(id do bloco)-block-div
Iniciação do menu de implementação	double click	(id do bloco)-block-div
Destaque das áreas de inserção no processo de animação	drag	(id do bloco)-addArrow-ani
Adição de um novo bloco	drop	(id do bloco)-addArrow-ani

Caso bloco adicionado represente uma estrutura de condição ou cíclica, a funcionalidade de destaque da área de inserção também necessita de ser adicionada aos restantes elementos onde esta pode ocorrer. Esses elementos dependem da fase de construção em que se encontra o bloco. Numa fase inicial esta funcionalidade é destacada aos seguintes elementos:

- **(id)-(lado)**

Após a inserção do primeiro bloco a este tipo de blocos, é efetuada a redefinição da estrutura do lado onde este foi adicionado, logo, foi necessário adicionar esta funcionalidade aos seguintes elementos:

- **(id)-(lado)-upperSide**
- **(id)-(lado)-lowerSide**

A palavra “lado” acima exposta pode assumir os valores “rightSide” ou “leftSide” mediante o tipo de bloco em causa. Se esse bloco for do tipo “Se/senão”, contém os dois valores, caso contrário apenas possui o lado “rightSide”.

O controlo do aspeto do fluxograma no processo de adição foi implementado mediante o elemento onde ocorreu a ação. Para tal foi necessário identificar o tipo de adição a partir do id desse elemento. Assim sendo, assume o seguinte comportamento para cada tipo de elemento:

- **(id)-addArrow** – o sistema identifica que se trata de uma adição adjacente a um bloco, logo, adiciona a "div" do bloco à "div" pai onde ocorreu a inserção.
- **(id)-(lado)** – o sistema verifica que se trata da primeira inserção no lado do bloco. Desta forma faz a redefinição da estrutura do bloco pai e insere o filho.
- **(id)-(lado)-upperSide** - o sistema verifica que se trata de uma inserção do lado superior. Desta forma, o sistema define que se deve criar o elemento com elo de ligação.
- **(id)-(lado)-lowerSide** - o sistema verifica que se trata de uma inserção do lado superior. Desta forma, o sistema define que se deve criar o elemento sem elo de ligação. Juntamente com esta inserção, o sistema também redefine o bloco anterior ao agora selecionado de modo a este apresentar o elo de ligação.

Em cada uma das quatro opções, o sistema realiza a verificação final de se o bloco onde o evento se realizou se encontra dentro de uma instrução condicional ou cíclica. Caso isto se verifique, é efetuada a redefinição da estrutura do bloco recursivamente.

6.4.6 Configuração da animação

Antes de se iniciar a animação de um bloco foi necessário proceder à sua configuração. A configuração da animação foi implementada de modo a ser gerida pelas classes que representam os blocos. Para realizar essa configuração, foi necessário utilizar uma metodologia que permitisse agendar os vários tipos de animação que constituem a animação completa de um bloco. De forma a agendar todas as animações sequencialmente, decidiu-se optar pela utilização de filas de espera fornecidas pela biblioteca jQuery[27]. Esta funcionalidade permite executar uma lista de funções dispostas numa lista de espera associada a um elemento HTML. Como as animações dos blocos são compostas por vários tipos de animação em vários elementos do menu da animação, foram necessárias utilizar várias filas de espera. Para representar uma fila de espera, definiu-se uma classe com os seguintes atributos:

- **queuename** – indica o nome associado à fila de espera;
- **queueObject –** indica o nome do objeto associado à fila de espera;
- **index –** valor que reserva o número de funções da fila de espera que foram executadas;
- **queueFunctions –** variável que armazena a lista de funções adicionadas;

Através da utilização desta classe foi possível configurar a animação dos blocos. Para tal, para cada grupo de animações existentes na animação dos blocos foi criada uma fila de espera especifica para cada um delas. De seguida encontram-se explicados os vários tipos de animação possíveis no sistema juntamente com as suas propriedades e em que tipo de blocos se inserem.

6.4.7 Controlo da animação

O controlo da animação foi possível graças a uma série de funcionalidades. Tal como se conseguiu perceber através da leitura do documento, essas funcionalidades são:

- **Definir velocidade de animação –** permite definir a velocidade à qual a animação vai ocorrer;
- **Iniciar a animação** – inicia o processo de animação a partir do bloco atual. O processo iniciado através do processo de *dequeue* da fila de espera principal da animação do bloco.
- **Executar um passo da animação –** opera de forma semelhante ao anterior. Contudo, o processo de animação é interrompido após a animação do bloco;
- **Pausar a animação –** o sistema interrompe a animação e limpas todas as filas de espera que o bloco possuía.
- **Parar a animação –** executa o processo realizado no ponto anterior a todos os blocos do fluxograma que foram executados até ao momento e coloca o primeiro bloco do fluxograma como o próximo bloco a ser animado.
- **Voltar ao menu de construção**.

Para a implementação destas funcionalidades foram essenciais as seguintes propriedades:

- **index** do objeto "Queue";
- **blockQueues** do objeto "Block".

Tal como foi referido anteriormente, o atributo "index" do objeto "Queue" é o valor que indica o número de funções da fila de espera que foram executadas. A sua importância deve-se ao facto de, graças a ele, o sistema sabe em que animação recomeçar a animação depois de uma pausa solicitada por parte do utilizador. Ou seja, quando uma animação é recomeçada, o sistema apenas irá proceder à configuração das animações cuja ordem pertence à fase posterior ao índice corrente. Contudo, para que isto aconteça foi necessário, primeiramente, certificar que as filas de espera contidas na animação não eram criadas novamente. Assim, esta verificação foi possível através do atributo "blockQueues" que armazena todas as filas de espera criadas na animação de um bloco. Assim, após o reiniciar a animação o sistema verifica se a fila de espera existe ou não neste atributo. Se existir inicia o processo de animação a partir do momento onde tinha sido interrompido. Se não existir cria a fila de espera, executa a sua configuração a adiciona-a ao atributo "blockQueues" do objeto "Block".

De forma a garantir que as interações do utilizador durante esta fase não comprometiam a funcionalidade do sistema, foi necessário garantir que o sistema gerisse as operações disponíveis ao utilizador opções cada ação que este executasse. Assim, após cada interação do utilizador com o sistema eram bloqueadas as operações que ele poderia fazer de seguida após essa ação. De seguida encontra-se uma tabela que demonstra todas as interações possíveis pelo utilizador e a respetiva resposta do sistema às mesmas.

Tabela 6.5 - Interações possíveis no controlo da animação e respetiva resposta do sistema.

Ação do utilizador	Ações permitida pelo sistema	Ações impedida pelo sistema
Iniciar	• Parar • Pausar • Alterar velocidade de animação	• Iniciar • Executar um passo • Voltar ao menu de construção
Pausar	• Iniciar • Parar • Executar um passo da animação • Alterar velocidade de animação • Voltar ao menu de construção	• Pausar
Parar	• Iniciar • Executar um passo da animação • Alterar velocidade de animação • Voltar ao menu de construção	• Pausar • Parar
Executar um passo da animação	• Alterar velocidade de animação	• Iniciar • Pausar • Parar • Executar um passo da animação • Voltar ao menu de construção

6.5 Utilização da ferramenta

Para aceder à ferramenta é necessário executar os seguintes passos:

- Colocar o código do projeto na pasta "mod" do código do Moodle;
- Atualizar a base de dados no site do administrador para o sistema reconhecer o novo plugin;
- Criar uma atividade num curso;
- Preencher os campos relativos à criação da atividade;
- Iniciar a atividade clicando no *link* com o nome associado à atividade criada;
- Depois de entrar na atividade, deve criar um exercício preenchendo os campos pedidos nessa fase;
- Após a criação do exercício, deve selecionar a *aba* com o nome "Visualizar lista de exercícios" presente na barra de navegação da atividade;

- Quando se encontrar na página de visualização dos exercícios, para iniciar a ferramenta de construção e simulação de algoritmos, deve selecionar a hiperligação presente no nome do exercício;
- Após selecionar um exercício, será reencaminhado para a ferramenta e proceder à sua utilização.

Capítulo 7
Conclusão

7.1 Estado atual do projeto

Até à data, o propósito principal do trabalho encontra-se desenvolvido. A partir do trabalho realizado, foi possível desenvolver uma ferramenta que permita aos estudantes construir e simular algoritmos de uma forma interativa e apelativa. Esta encontra-se desenvolvida tendo em conta os vários pontos de vista pedagógicos que se consideraram importantes alcançar para garantir que a sua utilização seja proveitosa para os estudantes na aprendizagem dos conceitos básicos de programação. Acredita-se que este objetivo foi alcançado graças à implementação dos seguintes fatores:

- Definição dos menus de implementação simples e claros relativamente aos campos necessários a preencher;
- Redução dos erros de implementação graças à verificação da sintaxe das expressões definidas nesta fase e da concordância dos valores nela utilizados. Desta forma garantiu-se que o estudante realize uma simulação do seu algoritmo com o menor número de erros possíveis.
- Apresentação de erros precisos relativamente à implementação dos blocos e criação de variáveis.
- Implementação de atalhos como copiar, colar e remover blocos para melhorar a experiência de utilização da ferramenta;
- Fornecimento de menus de ajuda ao longo dos vários passos que envolvem a fase de construção do fluxograma;
- Observação detalhada das execuções das instruções que constituem um algoritmo de programação. Este foi possível graças a uma apresentação sequencial de todos os conceitos computacionais abordados em cada uma das instruções ao nível dos seguintes aspetos:
    - Representar o fluxo de controlo;
    - Representar os dados;
    - Representar as operações lógicas e aritméticas;
- Controlo da animação por parte do estudante permitindo que este defina a velocidade de atuação da animação e efetue as seguintes operações:
    - Iniciar a animação
    - Executar um passo da animação
    - Parar a animação
    - Interromper a animação

Contudo, houve várias metas importantes que não foram alcançadas durante a fase de implementação. Dentro destas metas, a mais relevante foi não completar a integração do plugin com Moodle de modo a torná-lo num plugin oficial da plataforma. Apesar de esta se encontrar desenvolvida de modo operacional com o Moodle, não se

conseguiu desenvolver o plugin de modo a torná-lo oficial e a poder ser divulgado. Apesar de esta fase ter sido iniciada encontra-se numa fase muito embrionária possibilitando apenas a navegação pelas páginas do plugin, a criação de um exercício, apresentação da lista de exercícios existentes na atividade e a seleção de um dos exercícios criados para a ferramenta ser iniciada. Desta forma, não foram implementadas funcionalidades que permitissem ao professor obter informação relativamente às dificuldades que os seus estudantes sentiriam na resolução dos exercícios criados por ele. Relativamente ao desenvolvimento da ferramenta, falhou a implementação do uso de funções auxiliares. Também os aspetos típicos de *debugging* em compiladores ficaram um pouco aquém do pretendido. Assim, a possibilidade de executar apenas partes do fluxograma, definidas pelo utilizador, através da introdução de *breakpoints* ou o recuo na simulação são aspetos por concluir. Para além disso, o processo de carregar e guardar ficheiros e de obter o pseudocódigo correspondente ao fluxograma não se encontra concluído devido a falhas de integração com o Moodle apesar da leitura e estruturação desses ficheiros estar desenvolvida. Por fim, um objetivo que também não foi alcançado foi a testagem da ferramenta com estudantes de modo a obter o seu feedback durante o processo de utilização e, assim, traçar melhorias necessárias para a mesma.

7.2 Trabalho idealizado vs trabalho realizado

Tal como foi referido na secção anterior, o trabalho idealizado para o projeto não corresponde na totalidade ao trabalho realizado pelo autor.

Figura 7.1 - Diagrama de Gantt com o trabalho idealizado para o 2º Semestre.

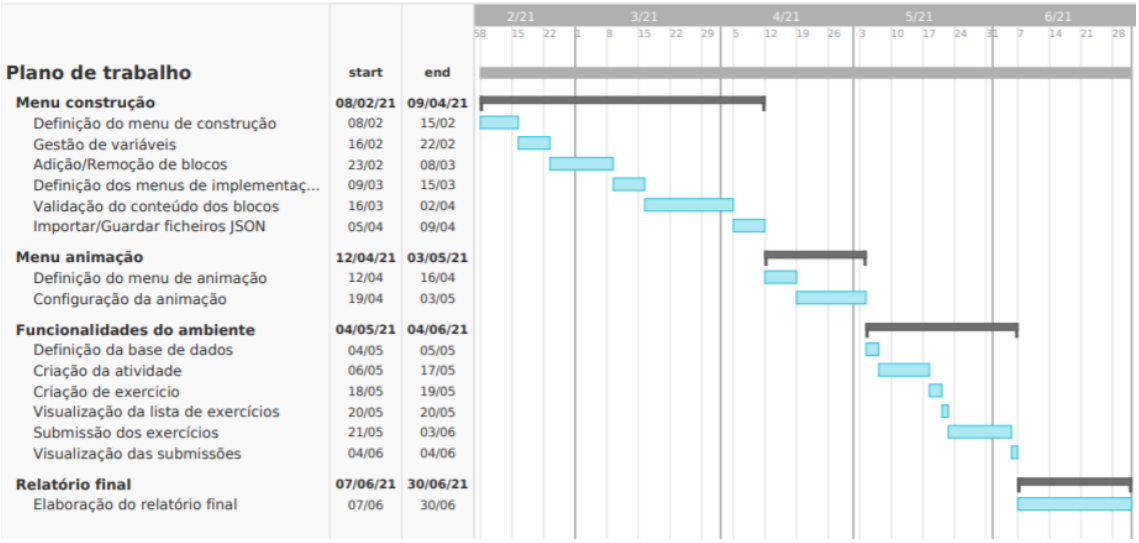
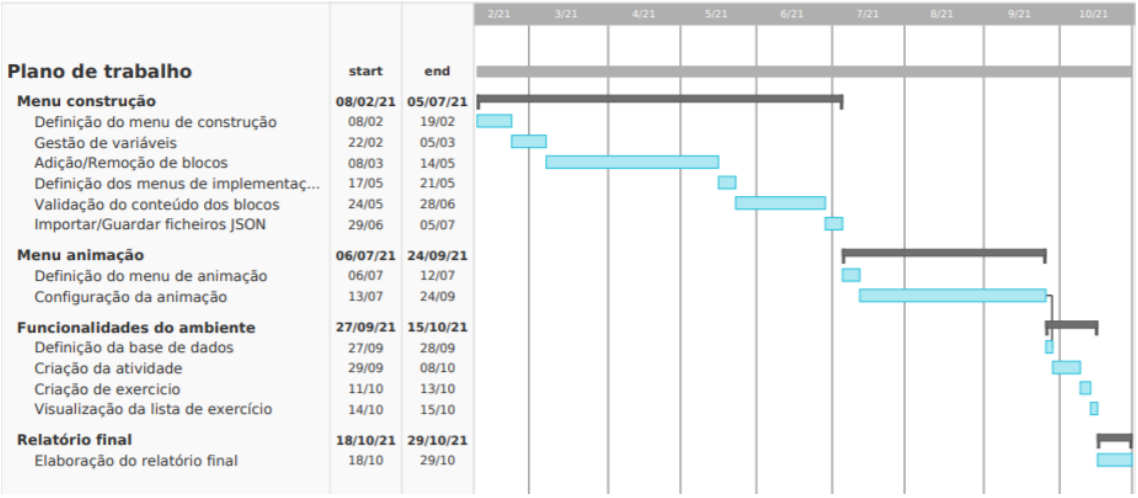


Figura 7.2 - Diagrama de Gantt com o trabalho real efetuado no 2º Semestre.



Plano de trabalho	start	end
Menu construção	**08/02/21**	**05/07/21**
Definição do menu de construção	08/02	19/02
Gestão de variáveis	22/02	05/03
Adição/Remoção de blocos	08/03	14/05
Definição dos menus de implementaç...	17/05	21/05
Validação do conteúdo dos blocos	24/05	28/06
Importar/Guardar ficheiros JSON	29/06	05/07
Menu animação	**06/07/21**	**24/09/21**
Definição do menu de animação	06/07	12/07
Configuração da animação	13/07	24/09
Funcionalidades do ambiente	**27/09/21**	**15/10/21**
Definição da base de dados	27/09	28/09
Criação da atividade	29/09	08/10
Criação de exercício	11/10	13/10
Visualização da lista de exercício	14/10	15/10
Relatório final	**18/10/21**	**29/10/21**
Elaboração do relatório final	18/10	29/10

A partir da análise das figuras 7.1 e 7.2, pode-se verificar a discrepância entre o tempo planeado e trabalho real. Na figura 7.2, observa-se que o desenvolvimento da ferramenta de construção e simulação de algoritmos ocupou quase todo o período, do 2º semestre, de implementação do projeto. Os processos que se estenderam mais foram os de adição e remoção de blocos do fluxograma, o da configuração da animação e o da validação das expressões dos blocos. A demora na implementação destas duas funcionalidades acabou por comprometer o cumprimento das metas estabelecidas para o segundo semestre e alcançar todos os objetivos definidos para o projeto. Na secção seguinte, encontram-se apresentadas as dificuldades sentidas ao longo do processo de desenvolvimento que justificam a demora no desenvolvimento destas funcionalidades.

7.3 Dificuldades

Durante o desenvolvimento do projeto, foram encontradas algumas dificuldades que acabaram por atrasar o processo. As tarefas deste processo mais desafiantes para o autor foram as seguintes:

- **Gestão da apresentação dos blocos de seleção e repetição** – a implementação da funcionalidade que realizasse a gestão visual destas estruturas durante o processo de construção revelou ser mais desafiante do que esperado. Devido ao desconhecimento por parte do autor da tecnologia utilizada para a representação dos blocos, o SVG, esse processo apresentou algumas dificuldades ao autor. Estas dificuldades foram principalmente sentidas na gestão da apresentação dos blocos condicionais ou cíclicos durante os processos de adição e remoção dos blocos.
- **Configuração da animação** – a implementação da representação da animação do fluxograma juntamente com as suas funcionalidades de controlo foi um processo que também colocou dificuldades ao autor. Dada a inexperiência do mesmo em desenvolvimento de funcionalidades deste tipo, tanto a idealização do processo como a posterior implementação foram difíceis. Isto aconteceu devido ao desconhecimento do autor de metodologias para implementar este tipo de funcionalidade. Assim, foi necessário realizar uma pesquisa relativamente às possibilidades existentes para efetuar essa

implementação e escolher uma dessas possibilidades. Devido a estes fatores, e à complexidade associada à implementação dos tipos de animação a desenvolver, este foi um processo que demorou mais tempo que o esperado.

- **Integração de bibliotecas externas no Moodle** – tal como foi referido, para a validação das expressões foi utilizada uma biblioteca externa para gerar um *parser* para as mesmas. A implementação deste processo também se estendeu por mais tempo do que o esperado devido às restrições do Moodle relativamente ao acesso a bibliotecas externas.
- **Validação das expressões** – visto que o objetivo era o de reduzir o número de erros sintáticos cometidos pelos alunos durante a construção do fluxograma, foi necessário avaliar cada *token* individualmente e garantir que cada um deles respeitava as regras necessárias para que a expressão fosse simulada sem que ocorresse qualquer tipo de erro. A dificuldade exponenciou-se devido às possíveis combinações existentes na definição das expressões. Por exemplo, argumentos de funções serem funções ou índices de um determinado *array*.

7.4 Trabalho futuro

Dado o teor do projeto apresentado no documento, o seu contributo pedagógico pode ser bastante relevante no futuro. Para tal, considera-se que futuramente seria desejável a integração de um conjunto de funcionalidades que melhorariam a eficácia desta ferramenta não apenas quando utilizada por estudantes como também pelos professores no acompanhamento aos seus estudantes, nomeadamente:

Conclusão do desenvolvimento do plugin para o Moodle

Este fator é um dos fatores fundamentais alcançar visto que era um dos principais objetivos do projeto. Assim, este fator iria contribuir para a disseminação do projeto. Dentro deste processo, é importante também desenvolver as funcionalidades que forneçam uma melhor experiência ao docente relativamente à deteção das dificuldades dos seus estudantes.

Inclusão da chamada de funções

Visto que a existência de funções definidas pelo utilizador corresponde a um conceito importante na aprendizagem dos conceitos básicos de programação, a sua adição à ferramenta seria uma mais-valia já que era um dos objetivos a alcançar.

Integração com o CodeInsights

Visto que um dos objetivos do projeto era fornecer dados estatísticos aos docentes relativamente ao desempenho dos seus estudantes nos exercícios, a integração do plugin com esta ferramenta poderia ser bastante benéfica para alcançar este objetivo. O CodeInsights é uma ferramenta que recebe uma codificação num vasto conjunto de linguagens de programação e em pseudocódigo e permite visualizar uma série de aspetos estatísticos relativos à sua resolução pelos estudantes. Desta forma, a integração entre as duas ferramentas traria mais um conjunto de benefícios adicionais ao ensino e aprendizagem de programação introdutória.

Tutorial interativo

De forma a melhorar a experiência da utilização da ferramenta por parte dos estudantes, considera-se relevante desenvolver um tutorial para a primeira utilização da ferramenta. Este tutorial deveria consistir na execução de uma série de passos que o estudante executaria após pedido do sistema de forma a criar um fluxograma simples e animá-lo. Ao longo deste tutorial, os vários conceitos apresentados ao estudante no procedimento da implementação devem ser explicados por parte do sistema. Deste modo, dar-se-ia uma alternativa mais precisa e que requeria menor esforço inicial por parte do estudante na aprendizagem da utilização da ferramenta.

Introdução de análises estatísticas na atividade

Pensa-se que a visualização de dados estatísticos também pode ser benéfica para o estudante. Podendo observar dados relativos à sua atividade na resolução dos exercícios propostos na ferramenta, fornece-se a possibilidade de o próprio identificar as lacunas nos seus conhecimentos. Desta forma, considera-se que esta possibilidade é importante para promover um comportamento auto didata por parte do mesmo na obtenção de conhecimentos ao nível da programação.

Sistema de sugestão de resolução

Numa fase mais adiantada do plugin, considera-se benéfica a implementação de um sistema que trate da gestão da atividade autonomamente. Desta forma, mediante os dados armazenados relativamente à resolução de cada estudante, considera-se que seria benéfico se a ferramenta fosse capaz de propor a resolução de um exercicio consoante a análise desses dados. Desta forma, seria possível auxiliar não apenas o estudante na identificação dos aspetos a melhorar na sua aprendizagem , como também o professor que estaria liberto da carga de efetuar as propostas de soluções individuais a cada estudante.

7.5 Considerações finais

Observando o trabalho desenvolvido neste estágio, pensa-se que este poderia ter sido melhor. Apesar do desenvolvimento da ferramenta de construção e simulação de algoritmos ter sido concluído, este possui algumas faltas relativamente aos objetivos traçados inicialmente. Para além disso, a falha na concretização do plugin do projeto num plugin oficial para o Moodle.

A nível técnico, considera-se que a elaboração deste projeto foi bastante proveitosa para o autor. Este processo permitiu ao autor adquirir conhecimentos mais profundos em relação ao desenvolvimento de tecnologias em Java*script*. Dentro disto, destacou-se a aprendizagem do uso da biblioteca JQuery que foi fundamental durante a fase de implementação do projeto, principalmente, no processo de configuração da animação. Para além disso, ainda que de uma forma não muito profunda, este projeto permitiu obter alguns conhecimentos no desenvolvimento de aplicações que utilizem PHP no *backend* da aplicação. Por fim, também permitiu adquirir conhecimentos relativos ao Moodle que poderão vir a ser importantes caso o autor se depare com um projeto que envolva um outro plugin para este sistema.

A nível pessoal, dadas as circunstâncias vividas no passado ano, permitiu ter uma noção de trabalho remoto e o modo como este pode ser conduzido. Esta experiência foi benéfica visto que é um cenário cada vez mais presente em empresas informáticas, especialmente depois da pandemia vivida recentemente. Para além disso, a implementação deste projeto demonstrou a vertente volátil do desenvolvimento de um projeto e a importância de se conseguir adaptar as dificuldades encontradas nesse processo.

Concluindo, considera-se que este projeto, para além de ter servido para desenvolver competências técnicas, serviu para dar a conhecer ao autor o longo caminho que este necessita de percorrer para vingar futuramente no ramo. Este caminho envolve a aquisição de vários conhecimentos na área e desenvolvimento de uma melhor capacidade de organização das tarefas e de gestão do tempo.

Referências

[1] A. J. Mendes, A. Gomes, M. Esteves, M. J. Marcelino, C. Bravo, and M. A. Redondo, "Using simulation and collaboration in CS1 and CS2," *ACM SIGCSE Bulletin*, vol. 37, no. 3, pp. 193–197, 2005, doi: 10.1145/1151954.1067499.

[2] D. H. Smith, Q. Hao, F. Jagodzinski, Y. Liu, and V. Gupta, "Quantifying the Effects of Prior Knowledge in Entry-Level Programming Courses," *CompEd 2019 - Proceedings of the ACM Conference on Global Computing Education*, pp. 30–36, 2019, doi: 10.1145/3300115.3309503.

[3] H. N. Liang, C. Fleming, J. Morey, K. Sedig, and K. L. Man, "Students' perception on the use of visual tilings to support their learning of programming concepts," in *Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering, TALE 2013*, 2013, pp. 121–126. doi: 10.1109/TALE.2013.6654412.

[4] Y. Qian and J. Lehman, "Students' misconceptions and other difficulties in introductory programming: A literature review," *ACM Transactions on Computing Education*, vol. 18, no. 1, pp. 1–24, 2017, doi: 10.1145/3077618.

[5] A. J. Gomes, Á. N. Santos, and A. J. Mendes, "A study on students' behaviours and attitudes towards learning to program," *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, pp. 132–137, 2012, doi: 10.1145/2325296.2325331.

[6] A. Gomes and A. J. Mendes, "An environment to improve programming education," *ACM International Conference Proceeding Series*, vol. 285, pp. 1–6, 2007, doi: 10.1145/1330598.1330691.

[7] E. Lahtinen, T. Ahoniemi, and A. Salo, "Effectiveness of integrating program visualization to a programming course," *Proceedings of The Seventh Koli Calling Conference on Computer Science Education*, pp. 195–198, 2007, [Online]. Available: http://dl.acm.org/citation.cfm?id=2449350%0Ahttp://crpit.com/confpapers/CRPITV88Lahtinen.pdf

[8] "Moodle," *https://moodle.org/?lang=pt#slide1*.

[9] Y. Cherenkova, D. Zingaro, and A. Petersen, "Identifying challenging CS1 concepts in a large problem dataset," *SIGCSE 2014 - Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pp. 695–700, 2014, doi: 10.1145/2538862.2538966.

[10] A. Petersen, M. Craig, J. Campbell, and A. Tafliovich, "Revisiting why students drop CS1," *ACM International Conference Proceeding Series*, pp. 71–80, 2016, doi: 10.1145/2999541.2999552.

[11] P. Kinnunen and L. Malmi, "CS minors in a CS1 course," *ICER'08 - Proceedings of the ACM Workshop on International Computing Education Research*, pp. 79–90, 2008, doi: 10.1145/1404520.1404529.

[12] Docs.moodle.org, "Step-by-step Installation Guide for Ubuntu," *https://docs.moodle.org/311/en/Step-by-step_Installation_Guide_for_Ubuntu*.

[13] "PHP: Começando - Manual." https://www.php.net/manual/pt_BR/getting-started.php (accessed Jan. 18, 2021).

[14] Docs.moodle.org, “Moodle architecture - MoodleDocs.” https://docs.moodle.org/dev/Moodle_architecture#Moodle_as_a_modular_system (accessed Jan. 18, 2021).
[15] E. P. Allman *et al.*, “The Architecture of Open Source Applications (Volume 2): Moodle.” http://www.aosabook.org/en/moodle.html (accessed Jan. 18, 2021).
[16] “Directory and system structure - Moodle 1.9 Extension Development.” https://subscription.packtpub.com/book/hardware_and_creative/9781847194244/1/ch01lvl1sec02/directory-and-system-structure (accessed Jan. 18, 2021).
[17] “The right database to use,” *https://moodle.org/mod/forum/discuss.php?d=383021*.
[18] A. Chung, “RequireJS,” *https://requirejs.org*.
[19] A. Chun, “Why AMD?,” *https://requirejs.org/docs/whyamd.html*.
[20] Docs.moodle.org, “Grunt,” *https://docs.moodle.org/dev/Grunt*.
[21] Docs.moodle.org, “Communication Between Components - MoodleDocs.” https://docs.moodle.org/dev/Communication_Between_Components (accessed Jan. 18, 2021).
[22] S. Brown, “C4 Model,” *https://c4model.com*.
[23] Docs.moodle.org, “Core APIs,” *https://docs.moodle.org/dev/Core_APIs*.
[24] Docs.moodle.org, “Security overview report,” *https://docs.moodle.org/311/en/Security_overview_report*.
[25] Docs.moodle.org, “PostgreSQL,” *https://docs.moodle.org/311/en/PostgreSQL*.
[26] Docs.moodle.org, “Setting up PhpStorm,” *https://docs.moodle.org/dev/Setting_up_PhpStorm*.
[27] Docs.moodle.org, “Activity Modules,” *https://docs.moodle.org/dev/Activity_modules*.
[28] Mozilla and individual contributors (2005-2021), “Expressões Regulares,” *https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Regular_Expressions*.
[29] D. Majda, “Peggy,” *https://peggyjs.org/documentation.html*.
[30] OpenJS Foundation, “jQuery Queues,” *https://api.jquery.com/jquery.queue/*.
[31] “Notação polonesa inversa,” *https://ms.livingeconomyadvisors.com/2513-what-is-reverse-polish-notation-rpn*.

Apêndices

Apêndice A
Requisitos funcionais

R01 – Criar Exercício

	Professor	Moodle
Nome do caso de uso	Criar exercício	
Âmbito	Criar um exercício para ser resolvido pelo estudante	
Nível	Alta	
Partes interessadas e seus interesses	O professor pretende criar um exercício para o aluno resolver	O sistema apresenta ao utilizador três parâmetros para este preencher: - Nome do exercício - Descrição do problema - Exemplos de valores de entrada - Exemplos de retorno
Pré-condições	O professor tem permissões para aceder ao sistema	
Pós-condições	O exercício é criado e adicionado à base de dados.	
Desencadeadores	O professor pretende criar um exercício para o estudante resolver	
Cenário principal de sucesso	1 – O professor seleciona a opção de "Criar exercício" 3 – O professor submeter o exercício	2 – O sistema apresenta uma página ao utilizador para preencher três parâmetros para este preencher: - Descrição do problema - Cenários inicias - Valor final consoante o cenário inicial 4 – O sistema cria o exercício e reencaminha o utilizador para a página anterior.
Caminhos alternativos	3a) O nome do exercício não foi preenchido 3b) O professor pressiona o botão "Cancelar	3a.1) O sistema informa o professor que a lista necessita de preencher o campo com o nome do exercício 3b.1) O Sistema cancela a operação e reencaminha o utilizador para a página anterior

R02 – Editar Exercício

	Professor	Moodle
Nome do caso de uso	Editar exercício	
Âmbito	Editar um exercício	
Nível	Média	
Partes interessadas e seus interesses	O professor pretende corrigir um campo do exercício que definiu anteriormente.	O sistema apresenta ao utilizador três parâmetros para este preencher: - Nome do exercício - Descrição do problema - Exemplos de valores de entrada - Exemplos de retorno
Pré-condições	- O professor tem permissões para aceder ao Sistema - O professor encontra-se na página da lista de exercícios	
Pós-condições	O exercício é alterado.	
Desencadeadores	O professor pretende alterar um exercício	
Cenário principal de sucesso	1 – O professor seleciona a opção de editar exercício 3 – O professor submeter o exercício	2 – O sistema apresenta uma página ao utilizador para preencher três parâmetros para este preencher: - Descrição do problema - Cenários inicias - Valor final consoante o cenário inicial 4 – O sistema altera as componentes do exercício e reencaminha para a página anterior.
Caminhos alternativos	3a) O professor pressiona o botão "Cancelar"	3a.1) O Sistema cancela a operação e reencaminha o utilizador para a página anterior

R03 – Remover Exercício

	Professor	Moodle
Nome do caso de uso	Remover exercício	
Âmbito	Remover um exercício	
Nível	Baixa	
Partes interessadas e seus interesses	O professor pretende remover um exercício	
Pré-condições	- O professor tem permissões para aceder ao Sistema - O professor encontra-se na página da lista de exercícios	
Pós-condições	O exercício é removido	
Desencadeadores	O professor pretende criar um exercício para o estudante resolver	
Cenário principal de sucesso	1 – O professor seleciona a opção de "Remover"	2 – O sistema remover o exercício.
Caminhos alternativos		

R04 – Listar exercícios

	Professor / Estudante	Moodle
Nome do caso de uso	Listagem dos exercícios	
Âmbito	O Sistema pretende apresentar ao utilizador a lista de exercícios existentes na atividade	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende visualizar a lista de exercícios.	O Sistema pretende disponibilizar os exercícios ao utilizador
Pré-condições	O utilizador possui permissões para entrar na atividade	
Pós-condições	O utilizador visualiza a lista de exercícios	
Desencadeadores	O utilizador pretende demonstrar / resolver um exercício.	
Cenário principal de sucesso	1 – O utilizador seleciona a opção "Lista de exercícios" na barra de navegação da atividade do *plugin*.	2 – O sistema reencaminha o utilizador para a página referente à listagem dos exercícios
Caminhos alternativos		

R05 – Iniciar a ferramenta de construção e simulação de algoritmos

	Utilizador	Moodle
Nome do caso de uso	Iniciar a ferramenta de construção e simulação de algoritmos	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador aceda à plataforma de construção e simulação de algoritmos	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende desenvolver habilidades ao nível da programação através de uma abordagem prática onde este aprenda e aplique os conceitos básicos desta matéria	O sistema pretende permitir ao utilizador que aceda à ferramenta
Pré-condições	O utilizador está na página onde está disponibilizada a lista de exercícios existentes no curso	
Pós-condições	O utilizador acede ao menu principal da plataforma de construção e simulação de algoritmos e pode ver dois blocos, um inicial e um final, para começar a construção do seu fluxograma	
Desencadeadores	O utilizador pretende resolver os exercícios propostos no curso de forma a obter habilitações ao nível da programação inicial	

Cenário principal de sucesso	1 – O utilizador seleciona um exercício da lista de exercícios do curso	2 – O sistema reencaminha o utilizador para a página da construção do fluxograma.
Caminhos alternativos		

R06 – Criar variável

	Utilizador	**Moodle**
Nome do caso de uso	Criar variável	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador adicione uma variável	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende introduzir uma variável no seu sistema	O sistema pretende permitir ao utilizador que aceda à ferramenta
Pré-condições	O utilizador está no menu principal da ferramenta	
Pós-condições	O utilizador observa que a variável foi adiciona à seção referente às variáveis existentes no algoritmo	
Desencadeadores	O utilizador pretende resolver os exercícios propostos no curso de forma a obter habilitações ao nível da programação inicial	
Cenário principal de sucesso	1 – O utilizador seleciona a opção de adicionar uma nova variável à função 3 – O utilizador introduz o nome da variável 4 – O utilizador seleciona a opção "Confirmar"	2 – O sistema mostra um menu que apresente ao utilizador: - Uma informação a dizer que irá adicionar uma variável à sua função - Uma caixa de texto para o utilizador inserir o nome que pretende dar a sua variável - Uma lista de carateres que não se podem introduzir no 5 – O sistema adiciona a variável à seção referente às variáveis no menu principal
Caminhos alternativos	3a) O utilizador insere carateres não permitidos ou o nome introduzido começa com um valor numérico	3a.1) O sistema informa o utilizador do erro 3a.2) O sistema reencaminha o utilizador novamente para o menu da adição de variáveis

R07 – Editar variável

	Utilizador	Moodle
Nome do caso de uso	Editar variável	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador editar uma variável	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende introduzir uma variável na sua função	O sistema pretende permitir ao utilizador que aceda à ferramenta
Pré-condições	O utilizador está no menu principal da ferramenta	
Pós-condições	O utilizador observa que a variável foi alterada e que todos os blocos onde era utilizada foram atualizados	
Desencadeadores	O utilizador pretende alterar as propriedades de uma variável	
Cenário principal de sucesso	1 – O utilizador seleciona a opção de adicionar uma nova variável à função 3 – O utilizador introduz o nome da variável 4 – O utilizador seleciona a opção "Confirmar"	2 – O sistema mostra um menu que apresente ao utilizador: - Uma informação a dizer que irá adicionar uma variável à sua função - Uma caixa de texto para o utilizador inserir o nome que pretende dar a sua variável 5 – O sistema adiciona a variável à seção referente às variáveis no menu principal
Caminhos alternativos	3a) O utilizador insere carateres não permitidos ou o nome introduzido começa com um valor numérico	3a.1) O sistema informa o utilizador do erro 3a.2) O sistema reencaminha o utilizador novamente para o menu da adição de variáveis

R08 – Remover variável

	Utilizador	Moodle
Nome do caso de uso	Remover variável	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador remova uma variável	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende remover uma variável	O sistema pretende permitir ao utilizador que aceda à ferramenta
Pré-condições	O utilizador está no menu principal da ferramenta	
Pós-condições	O utilizador observa que a variável foi removida e todos os blocos que a utilizavam também foram removidos do fluxograma	
Desencadeadores	O utilizador pretende resolver os exercícios propostos no curso de forma a obter habilitações ao nível da programação inicial	
Cenário principal de sucesso	1 – O utilizador seleciona a seleciona uma variável do seu sistema 3 – O utilizador pressiona o botão "Remover"	2 – O sistema disponibiliza o botão para remover a variável 5 – O Sistema remove a variável e todos os blocos que a utilizavam
Caminhos alternativos	3a) O utilizador não pressiona o botão "remover"	3a.1) O Sistema retira o destaque à variável seleciona e omite o botão "Remover"

R09 – Adicionar bloco ao fluxograma

	Utilizador	Moodle
Nome do caso de uso	Adicionar bloco ao fluxograma	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que, através da adição de um bloco, o utilizador construa o fluxograma	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende adicionar um bloco ao fluxograma que está a construir	O sistema pretende mostrar ao utilizador as opções disponíveis para a seleção do tipo de bloco que este quer adicionar ao seu fluxograma
Pré-condições	O utilizador encontra-se no menu principal da ferramenta de construção e simulação de algoritmos	
Pós-condições	O utilizador observa que o bloco escolhido foi adicionado ao seu fluxograma	
Desencadeadores	O utilizador pretende construir um algoritmo, através dosfluxogramas, para resolver o problema que lhe foi colocado	
Cenário principal de sucesso	1 - O utilizador seleciona, com o botão esquerdo do rato, um ramo que liga dois blocos do seu fluxograma 3 - O utilizador seleciona o tipo de bloco que pretende adicionar	2 – O sistema apresenta um menu ao utilizador com as opções disponíveis para os blocos 4 – O sistema adiciona o tipo de bloco selecionado pelo utilizador ao fluxograma e reencaminha o utilizador para o menu principal
Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de “Ajuda” no menu principal e procura a aba referente à funcionalidade que pretende executar	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade

R10 – Remover bloco do fluxograma

	Utilizador	Moodle
Nome do caso de uso	Remover bloco do fluxograma	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador remova um bloco, à exceção do primeiro e último, do fluxograma que está a construir	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende remover um bloco ao fluxograma que está a construir	O sistema pretende fornecer a possibilidade ao utilizador de utilizar esta funcionalidade
Pré-condições	1 - O utilizador encontra-se no menu principal da ferramenta de construção e simulação de algoritmos 2 - O utilizador possui um fluxograma com, pelo menos, três blocos	
Pós-condições	O utilizador observa que o bloco escolhido foi removido do seu fluxograma	
Desencadeadores	O utilizador pretende remover um bloco do seu fluxograma pois acredita que não é necessário para atingir o objetivo do problema proposta	
Cenário principal de sucesso	1 - O utilizador seleciona um dos blocos (exceto os extremos) do seu fluxograma 3 - O utilizador pressiona a tecla apagar do teclado	2 - O sistema destaca o bloco 4 – O sistema remove o bloco selecionado pelo utilizador e reencaminha o utilizador para o menu principal
Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de “Ajuda” no menu principal e procura a aba referente à funcionalidade que pretende executar	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade

R11 – Editar bloco do fluxograma

	Utilizador	Moodle
Nome do caso de uso	Remover editar do fluxograma	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador edite um bloco, à exceção do primeiro e último, do fluxograma que está a construir	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende editar um bloco ao fluxograma que está a construir	O sistema pretende fornecer a possibilidade ao utilizador de utilizar esta funcionalidade
Pré-condições	3 - O utilizador encontra-se no menu principal da ferramenta de construção e simulação de algoritmos 4 - O utilizador possui um fluxograma com, pelo menos, três blocos	
Pós-condições	O utilizador observa que o bloco escolhido foi alterado	
Desencadeadores	O utilizador pretende corrigir que não ficou implementado como pretendia	
Cenário principal de sucesso	1 - O utilizador seleciona duplamente um dos blocos (exceto os extremos) do seu fluxograma 3 – O utilizador realiza as alterações que pretende e confirma o evento.	2 – O Sistema apresenta o menu de implementação do bloco que selecionou 4 – O Sistema atualiza a informação do bloco e apresenta-a ao utilizador.
Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de “Ajuda” no menu principal e procura a aba referente à funcionalidade que pretende executar	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade

R12 – Criar uma nova função

	Utilizador	Moodle
Nome do caso de uso	Criar uma nova função	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador crie uma nova função	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende adicionar uma função auxiliar para complementar a sua solução ao problema	O sistema retende fornecer meios ao aluno para realizar este tipo de operação
Pré-condições	O utilizador encontra-se no menu principal da ferramenta de construção e simulação de algoritmos	
Pós-condições	O utilizador é reencaminhado para o *layout* da função que adicionou para construção do seu fluxograma	
Desencadeadores	O utilizador pretende criar uma função auxiliar que auxilie na resolução do problema principal	
Cenário principal de sucesso	1 – O utilizador seleciona a opção para criar uma nova função na barra de controlo do menu principal 3 – O utilizador preenche oscampos requisitados no menu	2 - O sistema apresenta um menu ao utilizador onde este tem de definir o nome da função, e o número de parâmetros de entrada que vai receber. Deve também definir qual a variável da função que irá retornar. 4 – O sistema guarda as informações, cria a nova função e reencaminha o utilizador para o *layout* de construção desta função
Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de “Ajuda” no menu principal e procura a aba referente à funcionalidade que pretende executar	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade

R13 – Implementar bloco do fluxograma

R13.1 – Implementar bloco “Atribuir”

	Utilizador	Moodle
Nome do caso de uso	Implementar bloco do tipo “Atribuir”	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador implemente um bloco no fluxograma que atribua um valor a uma variável do seu algoritmo	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende implementar os blocos do seu fluxograma para atingir a solução ao problema colocado	1 – O sistema pretende possibilitar o utilizador a realizar esta operação 2 – O sistema pretende informar o utilizador sobre a funcionalidade que aquele bloco introduz no programa e mostrar casos de exemplo que auxiliem o utilizador, de acordo com o seu objetivo, a preencher todas as componentes do bloco
Pré-condições	1 – O utilizador encontra-se no menu principal da animação 2 - O seu fluxograma possui, pelo menos, um bloco deste tipo	
Pós-condições	O utilizador observa o bloco que selecionou nome da variável e o valor que lhe atribuiu	
Desencadeadores	O utilizador pretende introduzir uma instrução no seu algoritmo que atribuir um valor a uma variável.	

Cenário principal de sucesso	1 – O utilizador seleciona duplamente um bloco deste tipo existente no seu fluxograma com o lado esquerdo do rato 3 - O utilizador observa as informações mostradas no menu apresentado pelo sistema 4 - O utilizador escolhe a variável selecionado um item da *listbox* 5 – O utilizador seleciona o tipo de dado que quer atribuir à suavariável através da *listbox* 6 – O utilizador seleciona a opção "Confirmar"	2 – O sistema apresenta um menu para a implementação deste bloco a indicar: - Uma informação a referir que este bloco serve para atribuir um valor a uma variável - A importância da concordância entre o valor atribuído à variável e o tipo da mesma - Casos de exemplo de valores a atribuir para cada um dos tipos de dados - Apresenta uma *listbox* com as variáveis declaradas no algoritmo até ao momento - Uma caixa de texto para ser introduzido o valor para a variável 7 – O sistema guarda as informações e reencaminha o utilizador para o menu principal e coloca no bloco do fluxograma em causa, o nome da variável e o seu valor.
Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de "Ajuda" no menu principal e procura a aba referente à funcionalidade que pretende executar 3a) O utilizador atribuiu um valor errado para a variável. Por exemplo, carateres numa variável numérica	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade 3a.1) O sistema informa o utilizador sobre o erro 3a.2) O sistema reencaminha o utilizador para o menu de implementação do bloco novamente

R13.2 – Implementar bloco "Ler"

	Utilizador	Moodle
Nome do caso de uso	Implementar bloco do tipo "Ler"	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador implemente um bloco no fluxograma que requisite um valor ao utilizador, através da consola, para atribuir a uma variável	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende implementar os blocos do seu fluxograma para atingir a solução ao problema colocado	1 – O sistema pretende possibilitar o utilizador a realizar esta operação 2 – O sistema pretende informar o utilizador sobre a funcionalidade que aquele bloco introduz no programa e mostrar casos de exemplo que auxiliem o utilizador, de acordo com o seu objetivo, a preencher todas as componentes do bloco.
Pré-condições	1 – O utilizador encontra-se no menu principal da animação 2 – O seu fluxograma possui, pelo menos, um bloco deste tipo	
Pós-condições	O utilizador observa o bloco que selecionou com o nome da variável à qual vai ser atribuído o valor pela consola	
Desencadeadores	O utilizador pretende introduzir uma instrução no seu algoritmo que requisite um valor, através da consola, para atribuir a uma variável do seu programa.	
Cenário principal de sucesso	1 – O utilizador seleciona duplamente um bloco deste tipo existente no seu fluxograma 3 - O utilizador observa as informações mostradas no menu apresentado pelo sistema 4 - O utilizador seleciona uma variável, caso exista, presente na *listbox* disponibilizada 5 – O utilizador, caso pretenda, introduz uma mensagem na caixa de texto	2 - O sistema apresenta um menu para a implementação deste bloco a indicar: - Uma informação a referir que este bloco serve para permitir que seja atribuído um valor a uma variável através da introdução de um valor na consola - A importância da compatibilidade do tipo da variável à qual se pretende atribuir um valor e o valor que será introduzido na consola - A importância da variável à qual o utilizador pretende atribuir o

	6 – O utilizador seleciona a opção "Confirmar"	valor, estar declarada no algoritmo. - Apresenta uma *listbox* com as variáveis declaradas no algoritmo até ao momento - Uma caixa de texto onde o utilizador poderá introduzir uma mensagem que será exibida na consola quando esta instrução for executada 7 – O sistema guarda as informações e reencaminha o utilizador para o menu principal onde este pode observar o bloco com o nome da variável à qual vai ser atribuído o valor pela consola
Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de "Ajuda" no menu principal e procura a aba referente à funcionalidade que pretende executar	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade

R13.3 – Implementar bloco “Escrever”

	Utilizador	Moodle
Nome do caso de uso	Implementar bloco do tipo “Escrever”	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador implemente um bloco no fluxograma que mostre, na consola, o valor de uma variável	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende implementar os blocos do seu fluxograma para atingir a solução ao problema colocado	1 – O sistema pretende possibilitar o utilizador a realizar esta operação 2 – O sistema pretende informar o utilizador sobre a funcionalidade que aquele bloco introduz no programa e mostrar casos de exemplo que auxiliem o utilizador, de acordo com o seu objetivo, a preencher todas as componentes do bloco
Pré-condições	1 – O utilizador encontra-se no menu principal da animação 2 - O seu fluxograma possui, pelo menos, um bloco deste tipo	
Pós-condições	O utilizador observa a alteração no seu fluxograma	
Desencadeadores	O utilizador pretende introduzir uma instrução no seu algoritmo que mostre valor de uma variável do seu algoritmo na consola	
Cenário principal de sucesso	1 – O utilizador seleciona duplamente um bloco deste tipo existente no seu fluxograma 3 - O utilizador observa as informações mostradas no menu apresentado pelo sistema 4 - O utilizador seleciona uma variável, caso exista, presente na *listbox* disponibilizada 5 – O utilizador, caso pretenda, introduz uma mensagem na caixa de texto 6 – O utilizador seleciona a opção “Confirmar”	2 – O sistema apresenta um menu para a implementação deste bloco a indicar: - Uma informação a referir que este bloco serve para mostrar o valor de uma variável na consola - A importância da variável, à qual o utilizador pretende atribuir o valor, estar declarada no algoritmo. - Apresenta uma *listbox* com as variáveis declaradas no algoritmo até ao momento - Uma caixa de texto onde o utilizador poderá introduzir uma mensagem que será exibida na consola quando esta instrução for executada 7 – O sistema guarda as informações e reencaminha o utilizador para o menu

		principal e coloca no bloco do fluxograma em causa, a variável que irá ser mostrada na consola
Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de "Ajuda" no menu principal e procura a aba referente à funcionalidade que pretende executar	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade

R13.4 – Implementar bloco "Se"

	Utilizador	**Moodle**
Nome do caso de uso	Implementar bloco do tipo "Se"	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o estudante implemente um bloco no fluxograma que introduza uma estrutura condicional, *If,* no seu algoritmo	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende implementar os blocos do seu fluxograma para atingir a solução ao problema colocado	1 – O sistema pretende possibilitar o utilizador a realizar esta operação 2 – O sistema pretende informar o utilizador sobre a funcionalidade que aquele bloco introduz no programa e mostrar casos de exemplo que auxiliem o utilizador, de acordo com o seu objetivo, a preencher todas as componentes do bloco
Pré-condições	1 – O utilizador encontra-se no menu principal da animação 2 - O seu fluxograma possui, pelo menos, um bloco deste tipo	
Pós-condições	O utilizador observa no bloco que escolheu a expressão condicional que definiu	
Desencadeadores	O utilizador pretende introduzir uma instrução condicional no seu algoritmo	
Cenário principal de sucesso	1 – O utilizador seleciona duplamente um bloco deste tipo existente no seu fluxograma 3 - O utilizador observa as informações mostradas na caixa de imagem 4 – O utilizador define a expressão condicional	2 – O sistema apresenta um menu para a implementação deste bloco a indicar: - Uma informação a referir que este bloco serve para definir uma instrução condicional no algoritmo - O facto deste tipo de instrução possuir dois caminhos onde apenas um deles será executado mediante a expressão condicional

	5 – O utilizador seleciona a opção "Confirmar"	- Casos de exemplo sobre a definição destas expressões condicionais - Uma caixa de texto relativa à expressão condicional 6 – O sistema guarda as informações e reencaminha o utilizador para o menu principal e coloca no bloco do fluxograma em causa, a expressão condicional.
Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de "Ajuda" no menu principal e procura a aba referente à funcionalidade que pretende executar 4a) O utilizador define uma expressão inválida	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade 4a.1) O sistema informa o utilizador sobre o erro que cometeu 4a.2) O sistema reencaminha o utilizador para o menu de implementação do bloco novamente

R13.5 – Implementar bloco "Se/Senão"

	Utilizador	**Moodle**
Nome do caso de uso	Implementar bloco do tipo "Se/Senão"	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o estudante implemente um bloco no fluxograma que introduza uma estrutura condicional, *If else,* no seu algoritmo	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende implementar os blocos do seu fluxograma para atingir a solução ao problema colocado	3 – O sistema pretende possibilitar o utilizador a realizar esta operação 4 – O sistema pretende informar o utilizador sobre a funcionalidade que aquele bloco introduz no programa e mostrar casos de exemplo que auxiliem o utilizador, de acordo com o seu objetivo, a preencher todas as componentes do bloco
Pré-condições	3 – O utilizador encontra-se no menu principal da animação 4 - O seu fluxograma possui, pelo menos, um bloco deste tipo	
Pós-condições	O utilizador observa no bloco que escolheu a expressão condicional que definiu	

Desencadeadores	O utilizador pretende introduzir uma instrução condicional no seu algoritmo	
Cenário principal de sucesso	1 – O utilizador seleciona duplamente um bloco deste tipo existente no seu fluxograma 5 - O utilizador observa as informações mostradas na caixa de imagem 6 – O utilizador define a expressão condicional	2 – O sistema apresenta um menu para a implementação deste bloco a indicar: - Uma informação a referir que este bloco serve para definir uma instrução condicional no algoritmo - O facto deste tipo de instrução possuir dois caminhos onde apenas um deles será executado mediante a expressão condicional

R13.6 - Implementar bloco “Para”

	Utilizador	Moodle
Nome do caso de uso	Implementar bloco do tipo “Para”	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador introduza um bloco no fluxograma, relativo à estrutura de repetição *For,* no seu fluxograma	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende implementar os blocos do seu fluxograma para atingir a solução ao problema colocado	O sistema pretende informar o utilizador sobre a funcionalidade que aquele bloco introduz no programa e mostrar casos de exemplo que auxiliem o estudante, de acordo com o seu objetivo, a preencher todas as componentes do bloco
Pré-condições	O utilizador ter selecionado um bloco do tipo “Para” na funcionalidade “Adicionar bloco” ou ter clicado duplamente sobre um bloco deste tipo existente no seu fluxograma	
Pós-condições	O utilizador observa a alteração no seu fluxograma	
Desencadeadores	O utilizador pretende introduzir uma instrução cíclica no seu algoritmo que execute as instruções dentro desta estrutura durante um determinado período	
Cenário principal de sucesso	1 – O utilizador seleciona duplamente um bloco deste tipo existente no seu fluxograma 3 - O utilizador observa as informações mostradas no menu apresentado pelo sistema 4 – O utilizador preenche as indicações disponibilizadas pelo sistema de acordo com as suas pretensões 5 – O utilizador seleciona a opção “Confirmar”	2 – O sistema apresenta um menu para a implementação deste bloco a indicar: - Uma informação a referir que este bloco serve para definir uma estrutura cíclica do tipo *For.* Esta estrutura repete todas as instruções dentro deste bloco durante um período definido pelo utilizador - Casos de exemplo com a definição de estruturas deste tipo e que características a diferenciam das restantes do mesmo tipo - Uma *listbox* com as variáveis numéricas existentes no sistema para ser utilizada para percorrer este ciclo - Uma caixa de texto para o valor inicial e outra para o valor final do ciclo

		- Uma *listbox* para definir o sentido do ciclo (crescente ou decrescente) seguido do valor que vai ser incrementado/decrementado a cada iteração. Este valor vai ser incrementada à variável referida anteriormente 6 – O sistema guarda as informações e reencaminha o utilizador para o menu principal e coloca no bloco do fluxograma em causa, o nome do bloco juntamente com a expressão que define a condição de execução do ciclo
Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de "Ajuda" no menu principal e procura a aba referente à funcionalidade que pretende executar	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade

R13.7 - Implementar bloco “Enquanto”

	Utilizador	Moodle
Nome do caso de uso	Implementar bloco do tipo “Enquanto”	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador introduza um bloco no fluxograma, relativo à instrução cíclica *While,* no seu fluxograma	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende implementar os blocos do seu fluxograma para atingir a solução ao problema colocado	O sistema pretende informar o utilizador sobre a funcionalidade que aquele bloco introduz no programa e mostrar casos de exemplo que auxiliem o utilizador, de acordo com o seu objetivo, a preencher todas as componentes do bloco
Pré-condições	O utilizador ter selecionado um bloco do tipo “Enquanto” na funcionalidade “Adicionar bloco” ou ter clicado duplamente sobre um bloco deste tipo existente no seu fluxograma	
Pós-condições	O utilizador observa a alteração no seu fluxograma	
Desencadeadores	O utilizador pretende introduzir, no seu algoritmo, uma instrução cíclica no seu algoritmo que verifique uma condição para averiguar a possibilidade de executar as instruções dentro desta estrutura	
Cenário principal de sucesso	1 – O utilizador seleciona duplamente um bloco deste tipo existente no seu fluxograma 3 - O utilizador observa as informações mostradas no menu apresentado pelo sistema 4 – O utilizador preenche as indicações disponibilizadas pelo sistema de acordo com as suas pretensões 5 – O utilizador seleciona a opção “Confirmar”	2 – O sistema apresenta um menu para a implementação deste bloco a indicar: - Uma informação a referir que este bloco serve para definir uma estrutura cíclicas do tipo *While.* As execuções das instruções desta estrutura são executadas após a verificação da expressão condicional que a integra - Casos de exemplo com a definição de estruturas deste tipo e que características a diferenciam das restantes do mesmo tipo - Uma caixa de texto onde será inserida a expressão de condição para a execução desta estrutura 6 – O sistema guarda as informações e reencaminha o utilizador para o menu principal e coloca no bloco do fluxograma em causa, o nome do bloco juntamente com a expressão que define a condição de execução do ciclo

Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de “Ajuda” no menu principal e procura a aba referente à funcionalidade que pretende executar 4a) O utilizador define uma expressão inválida	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade 4a.1) O sistema informa o estudante sobre o erro que cometeu 4a.2) O sistema reencaminha o estudante para o menu de implementação do bloco novamente

R14 – Controlar e observar a animação

R14.1 – Iniciar menu da animação

	Utilizador	Moodle
Nome do caso de uso	Iniciar menu da animação	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador aceda ao menu da animação do algoritmo que construiu	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende observar a animação do algoritmo que construiu	O sistema pretende encaminhar o utilizador para um menu onde pode executar várias operações relativas ao controlo da animação do algoritmo que construiu
Pré-condições	O utilizador deve construir um fluxograma para executar esta funcionalidade	
Pós-condições	O utilizador é encaminhado para o menu da animação do seu algoritmo	
Desencadeadores	O utilizador pretende visualizar a animação do seu algoritmo	
Cenário principal de sucesso	1 - O utilizador seleciona a opção no menu principal correspondente ao início da animação. 4 – O utilizador observa o menu da animação	2 – O sistema converte o fluxograma construído pelo utilizador em código de modo a este poder ser compilado 3 – O sistema apresenta o menu da animação ao utilizador. Este menu está dividido em 3 secções: - Uma secção que apresenta o fluxograma que o utilizador construiu - Uma secção para apresentar as variáveis introduzidas no algoritmo - Uma secção para apresentar as operações lógicas e aritméticas realizadas nas instruções doalgoritmo - Uma barra de controlo que permita ao utilizador opções que permitam ao utilizador definir a velocidade da animação, iniciar a animação, pausar a animação, executar um passo da animação, retroceder um passo na animação e parar a animação.
Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de "Ajuda" no menu principal e procura a aba referente à funcionalidade que pretende executar	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade

R14.2 – Iniciar a animação

	Utilizador	Moodle
Nome do caso de uso	Iniciar a animação	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador inicie a animação do fluxograma que construiu	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende observar a animação do algoritmo que construiu	O sistema pretende demonstrar ao utilizador as animações relativas ao funcionamento do computador quando interpreta o algoritmo submetido do utilizador
Pré-condições	O utilizador deve construir um fluxograma para executar esta funcionalidade	
Pós-condições	O utilizador observa as animações do seu fluxograma	
Desencadeadores	O utilizador pretende visualizar como é que o computador interpreta e age quando lhe é submetido o algoritmo que o utilizador construiu	
Cenário principal de sucesso	1 - O utilizador seleciona a opção no menu principal correspondente ao início da animação. 4 – O utilizador observa as animações colocadas no menu da animação	3 – O sistema compila o código e apresenta neste menu todas as animações relativas aos aspetos abrangidos no fluxograma construído pelo utilizador de forma sequencial 5 – Quando a animação terminar, o sistema permanece no menu da animação com a informação relativa à animação toda exposta
Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de “Ajuda” no menu principal e procura a aba referente à funcionalidade que pretende executar 4a) Ocorre um erro de compilação	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade 4a.1) O sistema informa o utilizador da ocorrência do erro e que tipo de erro foi 4a.2) O sistema reinicia o menu da animação

R14.3 – Parar a animação

	Utilizador	Moodle
Nome do caso de uso	Parar a animação	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador termine a animação do fluxograma que construiu	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende concluir a animação	O sistema pretende fornecer meios que permitam ao utilizador realizar esta operação
Pré-condições	O utilizador deve ter iniciado a animação	
Pós-condições	O utilizador é reencaminhado para o menu principal	
Desencadeadores	O utilizador encontrou um problema no seu fluxograma e quer parar a animação para corrigir o problema	
Cenário principal de sucesso	1 - O utilizador seleciona a opção para terminar a animação	2 – O sistema termina a animação e reinicia o menu da animação
Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de "Ajuda" no menu principal e procura a aba referente à funcionalidade que pretende executar	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade

R14.4 – Pausar a animação

	Utilizador	Moodle
Nome do caso de uso	Pausar a animação	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador interromper a animação do fluxograma que construiu	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende interromper a animação	O sistema pretende fornecer meios que permitam ao estudante realizar esta operação
Pré-condições	O estudante deve ter iniciado a animação	
Pós-condições	O estudante fica no menu da animação onde terá que decidir que próxima ação tomar	
Desencadeadores	O estudante decide interromper a animação para executar uma ação disponível no painel de controlo da animação numa determinada altura da animação	
Cenário principal de sucesso	1 - O utilizador seleciona a opção para terminar a animação	2 – O sistema termina a animação e reencaminha o utilizador para o menu principal
Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de "Ajuda" no menu principal e procura a aba referente à funcionalidade que pretende executar	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade

R14.5 – Executar um passo na animação

	Utilizador	Moodle
Nome do caso de uso	Executar um passo na animação	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador avance um passo na animação do fluxograma que construiu	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador pretende recuar na animação para visualizar novamente alguma animação	O sistema pretende fornecer meios que permitam ao utilizador realizar esta operação
Pré-condições	1- O utilizador deve encontrar-se no menu da animação 2 – O utilizador deve ter pausado a animação	
Pós-condições	O utilizador observa a animação referente à instrução executada nesse passo e fica no menu da animação onde terá de decidir que próxima ação tomar	
Desencadeadores	O utilizador pretende repetir um passo da animação para compreender melhor um conceito abordado num determinado instante da animação	
Cenário principal de sucesso	1 - O utilizador seleciona a opção para recuar um passo na animação presente na barra de controlo do menu da animação	2 – O sistema avança a animação para a instrução seguinte ao momento em que o utilizador requisitou esta funcionalidade
Caminhos alternativos	1a) O processo não é intuitivo para o utilizador 1b) O utilizador seleciona a opção de “Ajuda” no menu principal e procura a aba referente à funcionalidade que pretende executar	1b.1) O sistema apresenta uma informação sobre o modo como se deve executar esta funcionalidade

R14.6 – Observar as animações relativas ao controlo de fluxo do algoritmo

	Utilizador	Moodle
Nome do caso de uso	Observar as animações relativas ao controlo de fluxo do algoritmo	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador observe várias animações relativas ao controlo de fluxo do algoritmo que construiu	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador visualizar os aspetos demonstrados pela animação	Pretende demonstrar ao utilizador as animações referentes ao fluxo de controlo do algoritmo que construiu.
Pré-condições	1- O utilizador deve encontrar-se no menu da animação 2 - O utilizador seleciona uma opção na barra de controlo da animação que execute a animação do algoritmo	
Pós-condições	O utilizador observa as animações relativas ao controlo do fluxo	
Desencadeadores	O utilizador inicia a animação do algoritmo que construiu	
Cenário principal de sucesso	1 – O utilizador observa a animação	2 – O sistema é capaz de representar as seguintes animações em relação a esta métrica: - Sombrear o bloco que está a ser executado na animação - Observar como o computador interpreta e executa operações lógicas e aritméticas na seção “” do menu da animação
Caminhos alternativos	4a) Ocorre um erro de compilação	1a.1) O sistema informa o utilizador da ocorrência do erro e que tipo de erro foi 1a.2) O sistema reinicia o menu da animação

R14.7 – Observar as animações relativas ao controlo dos dados existentes no algoritmo

	Utilizador	Moodle
Nome do caso de uso	Observar as animações relativas à representação de dados do algoritmo	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador observe várias animações relativas à representação dos dados do algoritmo que construiu	
Prioridade	Alta	
Partes interessadas e seus interesses	O utilizador visualizar os aspetos demonstrados pela animação	O sistema pretende demonstrar ao utilizador as animações referentes à representação dos dados do algoritmo que construiu
Pré-condições	1- O utilizador deve encontrar-se no menu da animação 2 - O utilizador seleciona uma opção na barra de controlo da animação que execute a animação do algoritmo	
Pós-condições	O utilizador observa as animações relativas à representação dos dados	
Desencadeadores	O utilizador inicia a animação do algoritmo que construiu	
Cenário principal de sucesso	1 – O utilizador observa a animação	2 – O sistema é capaz de representar as seguintes animações em relação a esta métrica: - Sempre que uma variável é declarada no algoritmo, essa variável é adicionada à secção das variáveis do algoritmo no menu da animação - Destacar as variáveis a ser usadas na instrução que está a ser executada na animação - Destacar as variáveis sempre que são inicializadas/modificadas
Caminhos alternativos	1a) Ocorre um erro de compilação	1a.1) O sistema informa o utilizador da ocorrência do erro e que tipo de erro foi 1a.2) O sistema reinicia o menu da animação

R14.8 – Definir a velocidade da animação

	Utilizador	Moodle
Nome do caso de uso	Definir a velocidade da animação	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador observe várias animações relativas à representação dos dados do algoritmo que construiu	
Prioridade	Média	
Partes interessadas e seus interesses	O utilizador pretende definir a velocidade da animação mediante as suas necessidades	O sistema pretende demonstrar ao utilizador as animações referentes à representação dos dados do algoritmo que construiu
Pré-condições	1- O utilizador deve encontrar-se no menu da animação	
Pós-condições	O utilizador observa as animações relativas à representação dos dados	
Desencadeadores	O utilizador pretende observar com maior detalhe uma animação do algoritmo	
Cenário principal de sucesso	1 – O utilizador seleciona a opção de definir a velocidade da animação no menu da animação 2 - O utilizador seleciona uma opção	2 – O sistema apresenta as várias escolhas 3 – O sistema guarda a informação
Caminhos alternativos	1a) Ocorre um erro de compilação	1a.1) O sistema informa o utilizador da ocorrência do erro e que tipo de erro foi 1a.2) O sistema reinicia o menu da animação

R15 – Converter fluxograma em código

	Utilizador	Moodle
Nome do caso de uso	Converter fluxograma em código	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador converta o fluxograma que construiu em código	
Prioridade	Média	
Partes interessadas e seus interesses	O utilizador pretende como é que é representado o fluxograma que construiu em código	O sistema pretende converter o fluxograma do utilizador em código
Pré-condições	1- O utilizador deve encontrar-se no menu da animação	
Pós-condições	O utilizador observa o código do fluxograma que construiu	
Desencadeadores	O utilizador pretende ser exposto à sintaxe das linguagens de programação	
Cenário principal de sucesso	1 – O utilizador seleciona uma opção na barra de controlo para converter o fluxograma em código 3 – O utilizador seleciona a linguagem para a qual pretende traduzir o seu fluxograma 6 – O utilizador observa o código	2 – O sistema apresenta uma janela com uma lista de linguagens de pseudocódigo 4 – O sistema traduz o fluxograma e em código compatível com a linguagem escolhida pelo utilizador 5 – O sistema apresenta numa janela o código
Caminhos alternativos		

R16 - Controlar a ampliação do *layout* de construção do fluxograma

	Utilizador	Moodle
Nome do caso de uso	Controlar a ampliação do *layout* de construção do fluxograma	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador amplie ou afaste o *layout* do fluxograma	
Prioridade	Baixa	
Partes interessadas e seus interesses	O utilizador pretende controlar a ampliação do *layout* de construção do fluxograma	O sistema pretende permitir que o utilizador execute esta funcionalidade
Pré-condições	O utilizador deve encontrar-se no menu da animação	
Pós-condições	O utilizador observa que o *layout* de construção foi ampliado/afastado	
Desencadeadores	O utilizador pretende obter uma visão geral do fluxograma que construiu	
Cenário principal de sucesso	1 – O utilizador seleciona uma opção na barra de controlo para ampliar/afastar o *layout* de construção do fluxograma	2 – O sistema executa a função requisitada pelo utilizador
Caminhos alternativos		

R17 – Controlar a ampliação do *layout* de construção do fluxograma

	Utilizador	Moodle
Nome do caso de uso	Controlar a ampliação do *layout* de construção do fluxograma	
Âmbito	Interação entre o utilizador e o sistema de forma a permitir que o utilizador amplie ou afaste o *layout* de construção do fluxograma	
Prioridade	Baixa	
Partes interessadas e seus interesses	O utilizador pretende modificar a ampliação do *layout* para se adequar as suas necessidades visuais	O sistema pretende permitir que utilizador execute essa opção
Pré-condições	O utilizador encontra-se no menu principal	
Pós-condições	O utilizador observa a ampliação do *layout* a ser aumentada/reduzida	
Desencadeadores	O estudante pretende construir o fluxograma da função que selecionou	

Cenário principal de sucesso	1 – O utilizador seleciona a uma das opções disponíveis para o controlo da ampliação	2 – O sistema efetua a operação escolhida pelo utilizador
Caminhos alternativos		

R18 – Modo livre na utilização da ferramenta

	Professor	**Moodle**
Nome do caso de uso	Modo livre na utilização da ferramenta	
Âmbito	Interação entre o estudante e o sistema de forma a permitir que o professor utilize a ferramenta sem que esta tenha um exercício associado	
Prioridade	Média	
Partes interessadas e seus interesses	O professor pretende utilizar a ferramenta para construi um exemplo de um algoritmo	O sistema pretende dar acesso à ferramenta ao professor
Pré-condições	O professor tem permissões para aceder à ferramenta	
Pós-condições	O professor é encaminhado para o menu principal da ferramenta	
Desencadeadores	O professor pretende explicar um conceito de programação através da ferramenta num contexto de sala de aula	
Cenário principal de sucesso	1 – O professor seleciona a opção para iniciar a ferramenta	2 – O sistema reencaminha o professor para o menu principal da ferramenta
Caminhos alternativos		

Apêndice B
Funções axiliares

Funções que operam sobre valores numéricos

- **sqrt -** função que permite executar a raiz quadrada de um valor;
  - Parâmetros:
    1. Valor numérico positivo
  - Tipo de retorno
    Valor numérico
- **abs -** função que retorna o valor absoluto de um número;
  - Parâmetros:
    1. Valor numérico
  - Tipo de retorno
    Valor numérico
- **round -** função que permite arredondar um número;
  - Parâmetros:
    1. Valor numérico
  - Tipo de retorno
    Valor numérico
- **floor -** função que permite arrendar um número para baixo;
  - Parâmetros:
    1. Valor numérico
  - Tipo de retorno
    Valor numérico
- **ceil -** função que permite arrendar um número para cima;
  - Parâmetros:
    1. Valor numérico
  - Tipo de retorno
    Valor numérico
- **log -** função que retorna o logaritmo de um número;
  - Parâmetros:
    1. Valor numérico
  - Tipo de retorno
    Valor numérico
- **Log2 -** função que retorna o logaritmo de 2 de um número;
  - Parâmetros:
    1. Valor numérico
  - Tipo de retorno
    Valor numérico
- **log10 -** função que retorna o logaritmo de 10 de um número;
  - Parâmetros:

        1. Valor numérico
    - Tipo de retorno
        Valor numérico
- **rand –** função que retorna um valor aleatório de 0 até ao valor definido pelo parâmetro;
    - Parâmetros:
        1. Valor numérico inteiro positivo
    - Tipo de retorno
        Valor numérico inteiro positivo
- **cos -** função que retorna o cosseno de um número;
    - Parâmetros:
        1. Valor numérico
    - Tipo de retorno
        Valor numérico
- **sin -** função que retorna o seno de um número;
    - Parâmetros:
        1. Valor numérico
    - Tipo de retorno
        Valor numérico
- **tan -** função que retorna o tangente de um número;
    - Parâmetros:
        1. Valor numérico
    - Tipo de retorno
        Valor numérico

Funções que operam sobre *Strings*

- **length -** função que retorna o tamanho de uma *String*;
    - Parâmetros:
        1. *String*
    - Tipo de retorno
        Valor numérico
- **substring -** função que retorna o uma partição de uma *String* a partir de um intervalo;
    - Parâmetros:
        1. *String*
        2. Valor numérico inteiro positivo
        3. Valor numérico inteiro positivo
    - Tipo de retorno
        String
- **replace -** função que substitui uma parte de uma *String* por outro caso a segunda parte ocorra na primeira;
    - Parâmetros:
        1. *String*
        2. *String*

- Tipo de retorno
  String

- **toLowerCase –** converte todos os carateres de uma *String* para letras minúsculas;
  - Parâmetros:
    1. *String*
  - Tipo de retorno
    String
- **toUpperCase -** converte todos os carateres de uma *String* para letras maiúsculas;
  - Parâmetros:
    1. *String*
  - Tipo de retorno
    String
- **concat -** função que permite concatenar duas *Strings*;
  - Parâmetros:
    1. *String*
    2. *String*
  - Tipo de retorno
    String
- **charAt -** função que retorna o carater numa dada posição da String;
  - Parâmetros:
    1. *String*
    2. Valor numérico inteiro positivo
  - Tipo de retorno
    String
- **indexOf -** função que retorna o índice de ocorrência de um determinado carater;
  - Parâmetros:
    1. *String*
    2. *String*
  - Tipo de retorno
    Valor numérico inteiro positivo

- **reverse -** função que reverte a ordem dos carateres de uma *String*;
  - Parâmetros:
    1. *String*
  - Tipo de retorno
    String

Apêndice C
Gramáticas

Gramática para atribuições

```
start
   = Expression
   / _

Expression
         = Assignment

Assignment
   = Variable _ "=" _ LogicalExpression
   / Variable _ "=" _ Expr

Expr
   = MethodCall
   / additive
   / StringsOp _ ('+' _ Expr)?

MethodCall
   = FunctionName "(" _ FunctionArguments _ ")" _ (MathOp _ Expr)?
   / FunctionName "(" _ ")" _ (MathOp _ Expr)?


FunctionArguments
   = Expr _ "," _ FunctionArguments
   / Expr

LogicalExpression
   = "NOT(" LogicalExpressionAux ")" _ LogicOperator _ LogicalExpression
   / "(" LogicalExpressionAux ")" _ LogicOperator _ LogicalExpression
   / LogicalExpressionAux _ LogicOperator _ LogicalExpression
   / "NOT(" LogicalExpressionAux ")"

   / "(" LogicalExpressionAux ")"

   / LogicalExpressionAux
```

```
LogicalExpressionAux
   = LogicalTerms _ LogicOperator _ LogicalExpressionAux
   / LogicalTerms _ LogicOperator _ LogicalExpressionAux
   / LogicalTerms _ LogicOperator _ LogicalExpressionAux
   / LogicalTerms


LogicalTerms
   = "NOT(" _ LogicalTermsAux _ LogicOperator _ LogicalTerms _ ")"
   / "(" _ LogicalTermsAux _ LogicOperator _ LogicalTerms _ ")"
   / LogicalTermsAux _ LogicOperator _ LogicalTerms
   / LogicalTermsAux


LogicalTermsAux
    = Expr _ LogicComparator _ Expr

additive
   = multiplicative _ "+" _ additive
   / multiplicative _ "-" _ additive
   / multiplicative

multiplicative
   = primary _ "*" _ multiplicative
   / primary _ "/" _ multiplicative
   / primary _ "%" _ multiplicative
   / primary _ "^" _ multiplicative
   / primary

primary
   = MethodCall
   / Variable
   / Num
   / "(" _ additive:additive _")"


Output
   = _ TextAux? _ "%" OutputAux Output
   / _ TextAux?

OutputAux
   = "("_ Expr _ ")" Output
   / Variable Output
   / Variable "["
   / "(" _ MethodCall _ ")" Output
```

```
   / "(" LogicalExpression _ ")" Output

FunctionName "function"
        = [a-z][a-zA-Z0-9_]*

StringsOp
   = Strings _ "+" _ StringsOp
   / Strings

Strings
        = Aspas _ TextAux _ Aspas

Characters
        = [a-z][a-zA-Z0-9_]*

Variable "variable"
   = Characters "["id2:Expr "]"
   / Characters

Num "num"
        = "-"? [0-9]+(.[0-9]+)?

LogicOperator "lo"
   = "AND"
   / "OR"

LogicComparator "lc"
   = ">="
   / "<="
   / ">"
   / "<"
   / "=="
   / "!="

Text
        = [a-zA-Z0-9,;.:-_*]+

TextAux
   = Text _ TextAux
   / Text

_ "whitespace"
   = [ \\\n\\r]*
```

```
MathOp =
   "+"
   /"-"
   /"*"
   /"/"
   /"%"
   /"^"

Aspas
   = "'"
   / '"'
```

Gramática para expressões lógicas

```
start
   = Expression
   / _

Expression
         = LogicExpression

Assignment
   = Variable _ "=" _ LogicalExpression
   / Variable _ "=" _ Expr

Expr
   = MethodCall
   / additive
   / StringsOp _ ('+' _ Expr)?

MethodCall
   = FunctionName "(" _ FunctionArguments _ ")" _ (MathOp _ Expr)?
   / FunctionName "(" _ ")" _ (MathOp _ Expr)?


FunctionArguments
   = Expr _ "," _ FunctionArguments
   / Expr

LogicalExpression
   = "NOT(" LogicalExpressionAux ")" _ LogicOperator _ LogicalExpression
   / "(" LogicalExpressionAux ")" _ LogicOperator _ LogicalExpression
   / LogicalExpressionAux _ LogicOperator _ LogicalExpression
```

```
   / "NOT(" LogicalExpressionAux ")"

   / "(" LogicalExpressionAux ")"

   / LogicalExpressionAux


LogicalExpressionAux
   = LogicalTerms _ LogicOperator _ LogicalExpressionAux
   / LogicalTerms _ LogicOperator _ LogicalExpressionAux
   / LogicalTerms _ LogicOperator _ LogicalExpressionAux
   / LogicalTerms


LogicalTerms
   = "NOT(" _ LogicalTermsAux _ LogicOperator _ LogicalTerms _ ")"
   / "(" _ LogicalTermsAux _ LogicOperator _ LogicalTerms _ ")"
   / LogicalTermsAux _ LogicOperator _ LogicalTerms
   / LogicalTermsAux


LogicalTermsAux
    = Expr _ LogicComparator _ Expr

additive
   = multiplicative _ "+" _ additive
   / multiplicative _ "-" _ additive
   / multiplicative

multiplicative
   = primary _ "*" _ multiplicative
   / primary _ "/" _ multiplicative
   / primary _ "%" _ multiplicative
   / primary _ "^" _ multiplicative
   / primary

primary
   = MethodCall
   / Variable
   / Num
   / "(" _ additive:additive _")"


Output
```

```
   = _ TextAux? _ "%" OutputAux Output
   / _ TextAux?

OutputAux
   = "("_ Expr _ ")" Output
   / Variable Output
   / Variable "["
   / "(" _ MethodCall _ ")" Output
   / "(" LogicalExpression _ ")" Output

FunctionName "function"
         = [a-z][a-zA-Z0-9_]*

StringsOp
   = Strings _ "+" _ StringsOp
   / Strings

Strings
         = Aspas _ TextAux _ Aspas

Characters
         = [a-z][a-zA-Z0-9_]*

Variable "variable"
   = Characters "["id2:Expr "]"
   / Characters

Num "num"
         = "-"? [0-9]+(.[0-9]+)?

LogicOperator "lo"
   = "AND"
   / "OR"

LogicComparator "lc"
   = ">="
   / "<="
   / ">"
   / "<"
   / "=="
   / "!="

Text
         = [a-zA-Z0-9,;.:-_*]+
```

```
TextAux
   = Text _ TextAux
   / Text

_ "whitespace"
   = [ \\\n\\r]*

MathOp =
   "+"
   /"-"
   /"*"
   /"/"
   /"%"
   /"^"

Aspas
   = "'"
   / '"'
```

Gramática para expressões de *output*

```
start
   = Expression
   / _

Expression
         = Output

Assignment
   = Variable _ "=" _ LogicalExpression
   / Variable _ "=" _ Expr

Expr
   = MethodCall
   / additive
   / StringsOp _ ('+' _ Expr)?

MethodCall
   = FunctionName "(" _ FunctionArguments _ ")" _ (MathOp _ Expr)?
   / FunctionName "(" _ ")" _ (MathOp _ Expr)?


FunctionArguments
```

```
   = Expr _ "," _ FunctionArguments
   / Expr

LogicalExpression
   = "NOT(" LogicalExpressionAux ")" _ LogicOperator _ LogicalExpression
   / "(" LogicalExpressionAux ")" _ LogicOperator _ LogicalExpression
   / LogicalExpressionAux _ LogicOperator _ LogicalExpression
   / "NOT(" LogicalExpressionAux ")"

   / "(" LogicalExpressionAux ")"

   / LogicalExpressionAux


LogicalExpressionAux
   = LogicalTerms _ LogicOperator _ LogicalExpressionAux
   / LogicalTerms _ LogicOperator _ LogicalExpressionAux
   / LogicalTerms _ LogicOperator _ LogicalExpressionAux
   / LogicalTerms


LogicalTerms
   = "NOT(" _ LogicalTermsAux _ LogicOperator _ LogicalTerms _ ")"
   / "(" _ LogicalTermsAux _ LogicOperator _ LogicalTerms _ ")"
   / LogicalTermsAux _ LogicOperator _ LogicalTerms
   / LogicalTermsAux


LogicalTermsAux
    = Expr _ LogicComparator _ Expr

additive
   = multiplicative _ "+" _ additive
   / multiplicative _ "-" _ additive
   / multiplicative

multiplicative
   = primary _ "*" _ multiplicative
   / primary _ "/" _ multiplicative
   / primary _ "%" _ multiplicative
   / primary _ "^" _ multiplicative
   / primary

primary
   = MethodCall
```

```
  / Variable
  / Num
  / "(" _ additive:additive _")"


Output
  = _ TextAux? _ "%" OutputAux Output
  / _ TextAux?

OutputAux
  = "("_ Expr _ ")" Output
  / Variable Output
  / Variable "["
  / "(" _ MethodCall _ ")" Output
  / "(" LogicalExpression _ ")" Output

FunctionName "function"
        = [a-z][a-zA-Z0-9_]*

StringsOp
  = Strings _ "+" _ StringsOp
  / Strings

Strings
        = Aspas _ TextAux _ Aspas

Characters
        = [a-z][a-zA-Z0-9_]*

Variable "variable"
  = Characters "["id2:Expr "]"
  / Characters

Num "num"
        = "-"? [0-9]+(.[0-9]+)?

LogicOperator "lo"
  = "AND"
  / "OR"

LogicComparator "lc"
  = ">="
  / "<="
  / ">"
```

```
    / "<"
    / "=="
    / "!="

Text
          = [a-zA-Z0-9,;.:-_*]+

TextAux
    = Text _ TextAux
    / Text

_ "whitespace"
    = [ \\\n\\r]*

MathOp =
    "+"
    /"-"
    /"*"
    /"/"
    /"%"
    /"^"

Aspas
    = "'"
    / '"'
```

Apêncide D
Funções de animação

HighlightBlock

Tal como o nome indica, esta animação é responsável por realizar o destaque do bloco que vai ser animado na secção do fluxo de controlo. Desta forma, esta é constituída pelos seguintes passos:

- Destaque da seção “fluxo de controlo”;
- Destaque da estrutura do bloco que vai sofrer a animação;

Tabela D.1 - Tabela que indica os valores associados à criação da fila de espera para a animação "HighlightBlock".

Nome da fila de espera	Objeto associado à fila de espera
flowAni	m-flow-section

Tabela D.2 - Animações que fazem parte da animação "HighlightBlock" e os elementos que sofrem essas animações.

Tipo de animação	Elemento animado
Destaque da secção “fluxo de controlo”	m-flow-section
Destaque do bloco	(id)-block. firstChild

Tabela D.3 - Instruções que utilizam esta animação.

Tipo de bloco
Bloco inicial e final
Atribuir
Ler
Escrever
Se
Se/Senão
Enquanto
Para

AddExpression

Esta animação é responsável por introduzir a expressão dos blocos na secção “Operações aritméticas e lógicas”. Esta possui grande importância pois permite a posterior

demonstração da computação destas expressões ao estudante. Assim, esta animação é constituída pelos seguintes passos:

- Retorno da secção “Fluxo de controlo” à cor inicial;
- Destaque da secção “Operações aritméticas e lógicas”;
- Adiciona a expressão à secção anterior;

Tabela D.4 - Tabela que indica os valores associados à criação da fila de espera para a animação "AddExpression"

Nome da fila de espera	Objeto associado à fila de espera
exprAni	div-controlo

Tabela D.5 - Animações que fazem parte da animação "AddExpression" e os elementos que sofrem essas animações.

Tipo de animação	Elemento animado
Retorna à cor inicial da secção “fluxo de controlo”	m-flow-section

Tabela D.6 - Instruções que utilizam esta animação.

Tipo de bloco
Atribuir
Escrever
Se
Se/Senão
Enquanto
Para

Tal como foi referido anteriormente, o processo de adição da expressão à secção é de extrema importância para a depois se realizar o processo de execução da mesma. Assim, antes de se realizar a adição dos vários *tokens* que compõe a expressão foi necessário formatá-los de forma que estes pudessem ser interpretados corretamente quando se processasse o cálculo da mesma. Uma expressão aritmética e lógica pode conter os seguintes *tokens*:

- **Operadores;**
- **Variáveis** (simples e *arrays)***;**
- **Funções;**
- ***Strings*;**
- **Valores numéricos**.

Tendo em consideração os tipos de elementos que podem constituir uma expressão, é possível observar-se diferenças entre eles ao nível da sua complexidade. Enquanto os operadores, variáveis simples, valores numéricos e *Strings* são estruturas singulares, o acesso a variáveis que são *array* e chamada de funções utilizam vários *tokens*. Desta forma, realizou-se o seguinte para estruturas as estruturas:

- Para os *tokens* singulares, criou-se um elemento “div” cujo id e o conteúdo correspondem ao valor do *token* que o elemento vai representar;

- Para a chamada de funções e variáveis de *array,* criou-se um elemento “div” cujo id corresponde ao nome da função ou variável que representa. Posteriormente, são adicionados todos os *tokens* singulares que constituem esta estrutura à div que pai que a representa.

Desta forma, o sistema consegue identificar o tipo de elemento verificando se a “div” que o representa possui filhos ou não. Após a criação e formatação destes elementos eles são inseridos numa nova fila de espera integrada na principal que ficou responsável por conter a animação de todos os elementos adicionados à secção onde a expressão vai ser exposta.

Tabela D.7 - Tabela que indica os valores associados à animação da adição da expressão.

Nome da fila de espera	Objeto associado à fila de espera
showAni	div-controlo

Tabela D.8 - Animações que fazem parte do processo de adição da expressão.

Tipo de animação	Elemento animado
Aparecimento dos *tokens* da expressão	div criada para o *token*

PerformOperation

Após a adição de uma expressão ao *layout* da animação, é necessário calcular o valor da mesma. Desta forma, esta animação é constituída pelos seguintes passos:

- Cálculo da expressão;
- Retorno à cor inicial da secção “Operações lógicas e aritméticas”.

Tabela D.9 - Tabela que indica os valores associados à criação da fila de espera para a animação "performOperation".

Nome da fila de espera	Objeto associado à fila de espera
opAni	Primeiro elemento da expressão

Tabela D.10 - Animações que fazem parte da animação "performOperation" e os elementos que sofrem essas animações.

Tipo de animação	Elemento animado
Retorna à cor inicial da secção “Operações lógicas e aritméticas”	m-flow-section

Tabela D.11 - Instruções que utilizam esta animação.

Tipo de bloco
Atribuir
Escrever
Se
Se/Senão
Enquanto
Para

De forma a ilustrar o processo de computação das expressões, foi necessário implementar uma série de procedimentos que permitissem identificar o tipo de objeto que cada *token* dessa expressão representava e realizar as operações existentes na expressão por ordem correta. Para tal, primeiro foi necessário os vários *tokens* presentes na expressão de modo que o seu processamento ocorre de acordo com as regras das expressões. Para alcançar este propósito, recorreu-se à conversão da expressão para a notação polonesa inversa[28]. Esta notação permite organizar a expressão segundo o fluxo lógico das operações nela executadas sem a utilização de parênteses (tabela D.12).

Tabela D.12 - Exemplo de uma conversão para notação polonesa inversa.

Expressão original	Notação polonesa inversa
3 * (1 + 2)	3 1 2 + *

Após esta conversão, foi possível percorrer a expressão e apresentar a animação sequencial da execução da expressão. Para tal, criou-se duas filas de espera:

- Uma para o cálculo das operações e respetiva animação;

Tabela D.13 - Tabela que indica os valores associados à fila de espera do cálculo da expressão.

Nome da fila de espera	Objeto associado à fila de espera
calculateAni	div-controlo

Tabela D.14 - Animações que fazem parte da animação do cálculo da expressão.

Tipo de animação	Elemento animado
Substituição dos *tokens* utilizados na operação pelo resultado	m-result-value.firstChild

- Uma para demonstrar o acesso aos valores utilizados na operação.

Tabela D.15 - Tabela que indica os valores associados à fila de espera associada à demonstração do acesso aos valores das variáveis do sistema.

Nome da fila de espera	Objeto associado à fila de espera
showValue	Primeiro *token* da operação

Tabela D.16 - Animações que fazem parte da animação do acesso a valores das variáveis do sistema.

Tipo de animação	Elemento animado
Acesso aos valores das variáveis armazenadas do sistema	Elemento presente na tabela dos valores das variáveis

AddVariable

Esta animação corresponde ao processo de atribuição de um valor a uma variável. Este processo é constituído pelos passos seguintes:

- Destaque da secção "Variáveis" do *layout* de animação;
- Adição do valor na tabela das variáveis;
- Retorno da secção "Variáveis" à cor inicial.

Tabela D.17 - Tabela que indica os valores associados à criação da fila de espera para a animação "AddVariable".

Nome da fila de espera	Objeto associado à fila de espera
addVar	m-r-var

Tabela D.18 - Animações que fazem parte da animação "addVariable" e os elementos que sofrem essas animações.

Tipo de animação	Elemento animado
Deslocação do resultado para o local onde o valor da variável está representado na tabela	m-result-value.firstChild

Tabela 0.19 - Instruções que utilizam esta animação.

Tipo de bloco
Atribuir
Para

ClearSections

Esta animação corresponde ao processo de limpeza da expressão presente na secção "Operações lógicas e aritméticas". Esta consiste na execução das seguintes operações:

- Remoção da expressão;
- Retorno da secção "Operações lógicas e aritméticas" à cor inicial.

Tabela D.20 - Tabela que indica os valores associados à criação da fila de espera para a animação "clearSections".

Nome da fila de espera	Objeto associado à fila de espera
clearSec	m-expr-divs[0]

Tabela D.21 - Animações que fazem parte da animação "clearSections" e os elementos que sofrem essas animações.

Tipo de animação	Elemento animado
Desaparecimento da expressão	*tokens* presentes na secção da expressão

Tabela 0.22 - Instruções que utilizam esta animação.

Tipo de bloco
Atribuir
Escrever
Se
Se/Senão
Enquanto
Para

AddArrowAni

Esta animação é responsável por ilustrar o seguimento do próximo bloco que vai ser executado. Esta é composta pelos seguintes passos:

- Destaque da seção "fluxo de controlo";
- Destaque do elemento de ligação entre o bloco que foi animado e o bloco que vai ser animado de seguida;

Tabela D.23 - Tabela que indica os valores associados à criação da fila de espera para a animação "HighlightBlock".

Nome da fila de espera	Objeto associado à fila de espera
addArrowAni	m-flow-section

Tabela D.24 - Animações que fazem parte da animação "HighlightBlock" e os elementos que sofrem essas animações.

Tipo de animação	Elemento animado
Destaque da secção “fluxo de controlo”	m-flow-section
Destaque do elo de ligação	(id)-addArrowBlock

Tabela 0.25 - Instruções que utilizam esta animação.

Tipo de bloco
Bloco inicial e final
Atribuir
Ler
Escrever
Se
Se/Senão
Enquanto
Para

HighlighSide

Esta animação ocorre de forma semelhante que a anterior. Contudo, em vez de animar os elementos presentes no “(id)-addArrowBlock” executa essa animação nos elementos que constituem um dos lados do bloco de uma instrução condicional ou cíclica. Os elementos a serem animados dependem do resultado obtido pela expressão lógica associada ao bloco. Caso seja verdadeira, animam os elementos que fazem parte da secção da direita do bloco “rightSide”, caso contrário animam o lado oposto. Nas instruções que apenas possuem um lado, são animados os elementos que fazem a ligação direta ao próximo bloco a seguir ao bloco animado segundo a estrutura do fluxograma.

Tabela D.26 - Tabela que indica os valores associados à criação da fila de espera para a animação "HighlightSide".

Nome da fila de espera	Objeto associado à fila de espera
sideAni	m-flow-section

Tabela D.27 - Animações que fazem parte da animação "HighlightSide" e os elementos que sofrem essas animações.

Tipo de animação	Elemento animado
Destaque da secção “fluxo de controlo”	m-flow-section
Destaque do elo de ligação	(id)-(lado)-upperSide / (id)-connect + (id)-addArrowBlock

Tabela D.28 - Instruções que utilizam esta animação

Tipo de bloco
Se
Se/Senão
Enquanto
Para

GetInputValue

Esta animação permitiu ao sistema implementar o processo de pedido de uma variável ao utilizador. Para tal foram necessários os seguintes passos:

- Destaque da secção “Consola”
- Pedir valor ao utilizador
- Valor é adicionado à tabela

Tabela D.29 - Tabela que indica os valores associados à criação da fila de espera para a animação "GetInputValue".

Nome da fila de espera	Objeto associado à fila de espera
inputAni	m-console

Tabela D.30 - Animações que fazem parte da animação " GetInputValue" e os elementos que sofrem essas animações.

Tipo de animação	Elemento animado
Destaque da secção “Consola”	m-console

Tabela 0.31 - Instruções que utilizam esta animação.

Tipo de bloco
Ler

De forma a realizar esse pedido, implementou-se uma *promise*. Através da sua utilização, foi permitido que o sistema avançasse na animação apenas após o processo de input estivesse completo. Para avançar, esta *promise* fica à espera de um evento por parte do utilizador. Este evento é acionado quando o utilizador pressiona a tecla “Enter” no teclado. Após essa ação, o sistema verifica se o valor obtido corresponde ao tipo de variável pedida pelo fluxograma e caso se verifique efetua a animação de adição de uma variável à tabela. Este último processo é semelhante ao “AddVariable” referido anteriormente neste capítulo. Contudo em vez deste utilizar o resultado de uma expressão, utiliza o valor inserido pelo estudante na secção de input disponível na consola.

showOuputValue

Esta animação permitiu ao sistema implementar o processo de impressão de uma mensagem ao utilizador. Para tal foram necessários os seguintes passos:

- Destaque da secção “Consola”;
- Formatação da mensagem;
- Mensagem é impressa na consola com os valores pedidos pelo utilizador.

Tabela D.32 - Tabela que indica os valores associados à criação da fila de espera para a animação "showInputValue".

Nome da fila de espera	Objeto associado à fila de espera
outputAni	m-console

Tabela D.33 - Animações que fazem parte da animação " showOuputValue" e os elementos que sofrem essas animações.

Tipo de animação	Elemento animado
Destaque da secção “Consola”	m-console
Aparecimento da mensagem na consola	m-console-area

Tabela D.34 - Instruções que utilizam esta animação

Tipo de bloco
Escrever

De forma a construir a expressão para ser exposta ao utilizador, foi necessário fazer a repartição da mesma de modo que o sistema conseguisse identificar quais elementos da expressão eram necessários calcular para imprimir a mensagem corretamente. Desta forma necessitou-se de fazer a separação entre as mensagens escritas pelo utilizador e os valores que deveriam ser impressos. Após a identificação das partes da expressão que correspondiam aos valores a serem impressos, necessitou-se de realizar a demonstração da computação destes valores ao utilizador. Para tal, cada um dos *tokens* resultantes deste processo, sejam eles apenas uma variável ou uma expressão, definiu-se que deveriam ser expostos na seção “Operações lógicas e aritméticas”. Graças a este aspeto, este procedimento utiliza os mesmos procedimentos da funcionalidade “AddExpression” e “PerformOperation”. Após este processo estar concluído, a mensagem é construída e exibida na consola ao utilizador.