Mestrado em Engenharia Informática

Relatório Final de Estágio

Management Information System (Django)

Francisco Monsanto
monsanto@student.dei.uc.pt

Orientadores:
Professor Hugo Oliveira
Mestre Carlos Oliveira
Data: 30 de Junho de 2014

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Resumo

Neste relatório, é apresentado um Sistema de Informação (SI) que, agregando as várias ferramentas já existentes numa empresa e introduzindo novas funcionalidades, permite ter um maior controlo ao nível do portfólio da empresa, da execução dos projectos e da gestão de recursos humanos e financeiros. Este sistema consiste numa Application Programming Interface (API) que utiliza informação existente noutras plataformas utilizadas pela empresa, mas que não é devidamente analisada. Isto permite introduzir melhorias significativas na gestão organizacional da empresa a todos os níveis (de projecto, equipa, portfólio e negócio).

No primeiro capítulo, é feita a contextualização do estágio e são explicados os seus objectivos. De seguida, no segundo capítulo, é feito um estudo do estado da arte, sendo analisadas as soluções existentes no mercado e as suas principais características. No capítulo Metodologia, é descrito o processo de desenvolvimento do trabalho durante o estágio e o seu planeamento. De seguida, no capítulo Especificação técnica, são mostrados os requisitos elicitados e a arquitectura delineada para o sistema. São também apresentados os módulos desenvolvidos de uma perspectiva de alto nível, bem como as estruturas de dados que lhes dão suporte. Finalmente, ainda neste capítulo, é feita uma análise das ferramentas escolhidas e das razões que levaram à sua escolha. No capítulo Implementação são discutidos em maior detalhe alguns pormenores relativos à implementação do sistema. O capítulo *Framework* de métricas apresenta uma reflexão sobre o possível uso do trabalho efectuado como base de uma possível *framework* para recolha de dados e apresentação de métricas relevantes para a gestão de projectos e de empresas. Finalmente, no último capítulo são apresentadas as considerações finais e uma reflexão sobre o possível trabalho futuro.

Palavras chave: Sistema de Informação (SI), Project and Portfolio Management (PPM), Gestão de Projecto, Capability Maturity Model Integration (CMMI), Metodologias ágeis, *dashboards*

Agradecimentos

Ao Professor Hugo Oliveira, pela disponibilidade, paciência e boa-disposição mostradas para responder a qualquer dúvida e rever o relatório.

À Ubiwhere, por me ter dado a oportunidade de fazer um estágio tão interessante como este, num ambiente incomparável. Em especial, ao Nuno e ao Carlos, pela paciência para me aturar, pela ajuda inestimável que deram e pelo espírito de camaradagem e boa-disposição durante todo o estágio.

Aos meus amigos, que me fazem manter o bom humor mesmo nos momentos mais difíceis e que fazem com que tudo valha a pena. Em especial, ao Ricardo, Zé e Filipe, os melhores amigos que alguém pode ter.

Aos avós, pelos ensinamentos e pela inspiração que são para mim.

Ao meu Padrinho, por ser, para mim, alguém verdadeiramente especial.

À Maggie, a melhor irmã do Mundo, com quem sei que posso sempre contar, nos melhores e nos piores momentos.

Aos meus pais, a quem devo tudo, que são os melhores exemplos que podia ter e que, mais longe ou mais perto, sempre me acompanharam e ajudaram em todo o percurso até à escrita deste trabalho.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

ALM Application Lifecycle Management

API Application Programming Interface

CMMI Capability Maturity Model Integration

CSS Cascading Style Sheets

ER Entidade-Relação

ERP Enterprise Resource Planning

GP Generic Practices

HTML HyperText Markup Language

HTTPS HyperText Transfer Protocol (Secure)

IM Instant Messaging

JSON JavaScript Oriented Notation

LDAP Lightweight Directory Access Protocol

MVC Model-View-Controller

ORM Object-Relational Mapping

PA Process Area

PM Project Manager

PO Product Owner

PPM Project and Portfolio Management

QA Quality Assurance

QC Quality Control

REST REpresentational State Transfer

RH Recursos Humanos

SaaS Software as a Service

SEI Software Engineering Institute

SGBD Sistema de Gestão de Base de Dados

SI Sistema de Informação

TI Tecnologias da Informação

TM Team Manager

UIS Ubiwhere Information System

URM Ubiwhere Relationship Manager

XML Extensible Markup Language

YAML Yet Another Markup Language

Capítulo 1

Introdução

Numa empresa de desenvolvimento de *software*, a gestão dos projectos assume uma importância extrema. Para que esta seja o mais eficaz e eficiente possível, é indispensável ter um SI que dê o suporte adequado às metodologias usadas no desenvolvimento de *software*.

O estágio insere-se no âmbito de um projecto em desenvolvimento na empresa Ubiwhere[1], uma empresa de desenvolvimento de *software* sedeada em Aveiro, e também com escritório no Instituto Pedro Nunes, em Coimbra.

O objectivo do projecto é dotar a empresa de um sistema com as capacidades necessárias para a implementação de melhorias processuais e com impacto no funcionamento da organização a todos os níveis. Este sistema, o Ubiwhere Information System (UIS), aglomera os dados das outras plataformas em uso na empresa, o que permite obter informação valiosa sobre o funcionamento e performance da mesma. Este sistema visa facilitar a tarefa dos gestores, através da disponibilização da informação, e dos restantes colaboradores, através da diminuição do tempo despendido em tarefas não produtivas.

1.1 Contexto

Tal como referido anteriormente, a gestão dos projectos é uma parte fundamental para uma empresa de desenvolvimento de *software*. O mau planeamento e gestão inadequada destes são os principais factores responsáveis por taxas de insucesso de 20% em "pequenos" projectos (orçamentos de menos de 350,000 USD), segundo um estudo da Gartner[2] para os projectos de Tecnologias da Informação (TI) em 2012 [1]. Esta taxa é ainda superior para

[1] `www.ubiwhere.com`

[2] Empresa de investigação sobre tecnologia, `www.gartner.com`

orçamentos mais elevados.

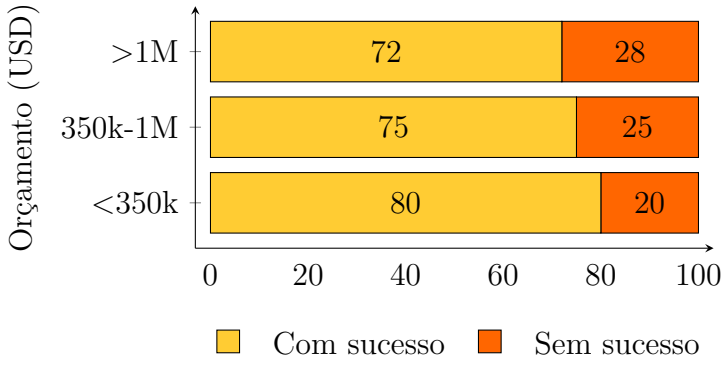

Figura 1.1: Taxa de sucesso e insucesso dos projectos de TI

Hoje em dia, os SIs são parte fulcral de uma empresa e podem englobar os mais diversos aspectos, desde a relação com os clientes até à gestão de Recursos Humanos (RH) da própria empresa. Ter um SI adequado acaba por se tornar numa grande vantagem competitiva, permitindo estes sistemas, quando bem implementados e aproveitados, atenuar os problemas acima descritos. Nessa medida, o SI deve dar suporte às práticas, políticas e processos da empresa. No caso da Ubiwhere, a base processual assenta sobre as *frameworks* ISO-9001[3], NP-4457[4] e, sobretudo, CMMI[5]. A aplicação destas *frameworks* nas empresas visa a estruturação dos processos para evitar as falhas anteriormente referidas.

A Ubiwhere também adopta metodologias ágeis. Estas são úteis na medida em que permitem a uma equipa de desenvolvimento lançar produtos mais rapidamente, através de práticas que obrigam a uma maior comunicação entre os *stakeholders*, a detectar problemas numa fase mais inicial do projecto, entre outras vantagens face a metodologias de estrutura mais rígida [2][3][4].

Embora a utilização de ferramentas adequadas não seja especificada ou sequer obrigatória segundo estas metodologias, a verdade é que se torna praticamente impossível de alcançar as melhorias que estas proporcionam sem o suporte tecnológico adequado. Por exemplo, a comunicação constante requerida pelo Scrum é muito difícil de atingir, especialmente em equipas cujos membros não estejam todos no mesmo espaço físico; também para

[3] *Standard* de *Quality Management*, `http://www.iso.org/iso/iso_9000`

[4] Certificação de Sistemas de Gestão de Investigação, Desenvolvimento e Inovação, `http://www.ipq.pt/custompage.aspx?pagid=4050`

[5] `http://cmmiinstitute.com/`

manter a documentação necessária é indispensável ter as ferramentas certas [5]. Quanto ao CMMI, é importante ter ferramentas que eliminem algumas tarefas repetitivas ou mais susceptíveis a erros [6].

Nas secções seguintes, descrevemos as metodologias CMMI e Scrum. Estas são as metodologias às quais o sistema a desenvolver deve dar suporte.

CMMI

Antes de mais, interessa descrever o CMMI: o que é e para que serve. O CMMI é um modelo desenvolvido pelo Software Engineering Institute (SEI)[6] e pela universidade de Carnegie Mellon[7]. Está dividido em três "constelações" diferentes: DEV (desenvolvimento de produtos), SVC (prestação de serviços) e ACQ (aquisições de bens ou serviços). A constelação que é referida implicitamente ao longo do trabalho é a DEV, dado que a Ubiwhere é uma empresa de desenvolvimento de *software*.

O CMMI é uma *framework* de melhores práticas. Em particular, o CMMI-DEV refere-se às melhores práticas relativas aos processos de desenvolvimento de *software* [7]. Este modelo pretende então descrever o que caracteriza os bons processos (e não descrever os processos em si), oferecendo um conjunto de guias para que as empresas os possam melhorar [8].

Um processo consiste na definição de pessoas e responsabilidades para a execução de uma tarefa, bem como dos procedimentos e métodos que descrevem como a realizar. Um processo define ainda quais as ferramentas e equipamento de suporte a serem usados [8].

As empresas que adoptem o CMMI podem ser certificadas pelo SEI nos níveis 2 a 5. Esta certificação é relevante na medida em que existem clientes que exigem às empresas dados níveis de certificação. Algumas empresas portuguesas com a certificação de CMMI[8] incluem a ISA[9] (nível 2), Novabase[10] (nível 3) e Critical Software[11] (nível 5), o que mostra a sua relevância.

A implementação das metodologias CMMI pode levar a ganhos consideráveis, por exemplo, em termos de qualidade dos produtos, orçamentação mais precisa e planeamento mais previsível [9]. No entanto, há que referir também que nem sempre a implementação destas decorre da melhor maneira e que há várias críticas ao modelo. Estas prendem-se sobretudo com os custos de implementação e com o grande foco que existe na produção de

[6] SEI: `http://www.sei.cmu.edu/`

[7] Universidade Norte-Americana, `http://www.cmu.edu/index.shtml`

[8] Lista de certificações em `https://sas.cmmiinstitute.com/pars/pars.aspx`

[9] Intelligent Sensing Anywhere, `http://www.isasensing.com/pt/`

[10] `http://www.novabase.pt/`

[11] `http://www.criticalsoftware.com/`

documentação e na melhoria dos processos em lugar dos próprios produtos. Isto é geralmente visto como sendo pouco produtivo [10].

Scrum

O Scrum é uma metodologia ágil de desenvolvimento de projectos de *software* em que é dada especial atenção ao controlo contínuo do trabalho realizado. Este controlo permite que a tomada de decisões (planeamento de *releases*) seja apoiada em dados reais e seja, consequentemente, mais eficaz.

Os projectos são divididos em *sprints*, que são períodos de trabalho relativamente curtos (normalmente entre 1 e 3 semanas). No fim de cada *sprint*, todos os *stakeholders* são informados sobre os resultados do trabalho, por forma a planear os próximos *sprints* e/ou reajustar os planos de trabalho.

Hierarquicamente, o Scrum é constituído por pequenas equipas (que devem ter entre 3 a 7 membros), sendo os *team members* os responsáveis pelo desenvolvimento do trabalho. O Product Owner (PO) é responsável por transmitir a visão do projecto à equipa, e deve funcionar como representante dos interesses do cliente. O *Scrum Master* tem como função fazer a interface entre a equipa e o PO, por forma a melhorar o desempenho da equipa [11].

1.2 Motivações

No estudo da Gartner referenciado anteriormente [1], são descritas as principais causas para o insucesso de projectos na área das TI, apresentadas na figura 1.2.

Qualquer projecto corre estes riscos; no entanto, podem ser evitados ou, pelo menos, mitigados, através da aplicação dos princípios do CMMI e do Scrum. Relativamente aos riscos destacados nesse estudo, podem ser tomadas as seguintes medidas:

- Para evitar a fraca qualidade dos projectos, deve haver uma monitorização de resultados e métricas provenientes das actividades de Quality Assurance (QA) e Quality Control (QC) descritas no CMMI.
- A grande variação de custos pode ser controlada através das práticas de planeamento e monitorização de custos, também descritas no modelo.
- A entrega tardia de projectos pode ser controlada através da monitorização regular de métricas dos *sprints* do Scrum, e comparando estas frente ao que foi planeado no início do projecto. Isto permite que haja um melhoramento contínuo no próprio planeamento, o que, por sua vez, faz com que seja mais fácil prever as datas de entrega.

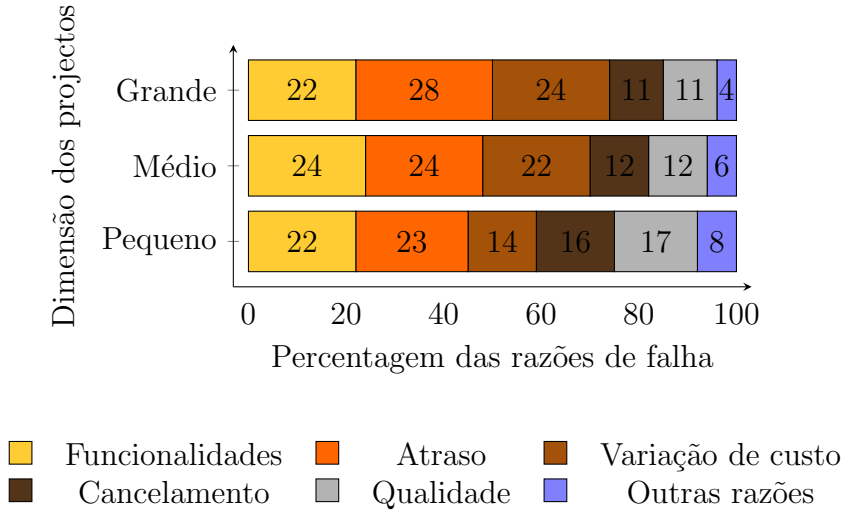

Figura 1.2: Percentagem de problemas que ocorrem nos projectos

- Quanto aos problemas com funcionalidades, podem ser evitados produzindo e analisando cuidadosamente os requisitos do sistema (Requirements Management e Development CMMI) e, em certa medida, através da abordagem ágil inerente ao Scrum, que permite ir adaptando o produto às necessidades do cliente.

Estas actividades podem ser desenvolvidas de forma mais fácil e automatizada com o uso de ferramentas adequadas. Para as soluções apresentadas nos parágrafos anteriores, é importante ter um sistema que seja capaz, por exemplo, de recolher automaticamente e apresentar métricas sobre os Recursos Humanos, *sprints*, projectos e contexto organizacional em tempo real. Isto permite aos colaboradores focarem-se nos projectos, sem terem que se preocupar com *overheads* relativos à recolha de dados. Permite também aos gestores analisar o trabalho desenvolvido, de forma a terem os dados necessários para os ajudar na tomada de decisões. Desta forma, os colaboradores têm uma melhor noção daquilo que deles é esperado, enquanto que os gestores podem tomar decisões mais conscientes, objectivas e, logo, tendencialmente mais produtivas.

1.2.1 Capability Maturity Model Integration (CMMI) Nível 3

Uma das motivações para a realização deste projecto é a passagem do nível 2 para o nível 3 do CMMI-DEV. O nível já obtido pela empresa assegura que

já existe uma gestão bastante completa a nível de projecto. A obtenção da certificação de nível 3 implica uma estandardização dos processos da empresa, que podem ser depois adaptados a cada projecto específico.

No nível 2 do CMMI, é esperado que a empresa consiga melhorar as práticas de gestão de projecto. No entanto, este nível apenas lança as fundações para que isto aconteça, sendo que, muitas vezes, não se notam melhorias de desempenho significativas, apesar da simplificação e uniformização dos processos. No nível 3, estas práticas devem ser alargadas a todos os níveis organizacionais, permitindo uma maior estandardização dos processos. Espera-se também que haja um foco no melhoramento contínuo destas práticas, por forma a acompanhar o crescimento e evolução da própria empresa.

No anexo A.2, são mostradas quais as funcionalidades do UIS que darão suporte às práticas do nível 3 do CMMI.

1.2.2 Plataformas existentes na Ubiwhere

As ferramentas usadas actualmente na Ubiwhere para a gestão interna não têm o grau de eficácia e eficiência desejados, sendo esta uma das motivações do projecto. Assim, torna-se importante descrever estas ferramentas e analisar as suas limitações.

Estas ferramentas são o Redmine[12], para a gestão de projectos, folhas de cálculo Google Sheets[13] para a análise de métricas relativas a *sprints*, projectos e à própria organização, a Intranet para a gestão de presenças dos colaboradores e, finalmente, o Ubiwhere Relationship Manager (URM) para a gestão de relações com clientes. As duas últimas ferramentas foram desenvolvidas internamente na Ubiwhere.

O facto de serem usados sistemas diferentes para estas funções causa algumas dificuldades, sendo o maior destes problemas o isolamento dos dados que cada aplicação gera. Por exemplo, se um Project Manager (PM) quiser obter informação sobre a performance de um dado colaborador, vai ter que aceder às várias plataformas da empresa: as suas alocações a projectos e equipas estão representadas numa folha de cálculo, o número de horas de esforço em cada tarefa encontra-se no Redmine, enquanto que a informação sobre as suas férias, presenças e faltas são guardados na Intranet. Esta separação dos dados em diferentes "silos" faz com que não haja uma forma simples de lhes aceder ou de os analisar, sendo esta análise demorada, penosa e, sobretudo, muito propensa a erros.

[12] `http://www.redmine.org/`

[13] `https://support.google.com/drive/answer/49008?hl=en`

No entanto, este não é o único problema da separação das ferramentas. Outro exemplo está relacionado com a gestão das autenticações e autorizações dos colaboradores nas diferentes plataformas. Assim, por exemplo, se um colaborador deixar de trabalhar para a empresa, o seu registo tem que ser alterado ou apagado da plataforma interna, e é necessário também remover quaisquer documentos partilhados via Google Docs individualmente.

Há ainda alguns problemas inerentes a cada plataforma em si. Estes problemas prendem-se principalmente com o facto de serem pouco adequadas às tarefas a que tentam dar resposta, quer seja devido às interfaces datadas, à falta de funcionalidades essenciais, ou simplesmente ao facto de estarem inacabadas (no caso das ferramentas desenvolvidas internamente).

Na tabela 1.1 é apresentado um resumo das limitações das diferentes plataformas discutidas anteriormente. Tendo em conta todos estes aspectos, é fácil perceber que será essencial para a empresa ter um sistema de informação que agregue as funcionalidades das plataformas mencionadas e acrescente novas possibilidades. Isto permitirá não só facilitar algumas operações mais rotineiras, como também tirar mais partido da informação gerada, resultando numa gestão mais fácil de pessoas, projectos e, consequentemente, do próprio negócio.

Ferramentas	Limitações
Todas	Pouca automação Dificuldade de gestão de autorizações Dados distribuídos dificultam análises
Redmine	Foco quase exclusivo na gestão de projecto Uso de *plugins* pouco suportados para suprir necessidades Poucas ferramentas de análise Interface inadequada
Intranet	Arquitectura pouco flexível Módulo de marcação de férias não operacional *Time tracking* pouco funcional *Bugs* Ferramentas administrativas insuficientes Interface datada
Google Sheets	Introdução de dados manual, morosa Dados pouco legíveis para análise
URM	Ferramenta inacabada

Tabela 1.1: Limitações das plataformas usadas na Ubiwhere

1.3 Objectivos

Nesta fase, a Ubiwhere pretende automatizar alguns dos processos definidos para o âmbito da certificação no nível 2 do CMMI. Para além disso, necessita de desenvolver as infraestruturas que permitam atingir o nível de maturidade 3, dando também suporte à aplicação do Scrum.

O objectivo deste projecto é dotar a Ubiwhere de um novo SI. Este sistema irá substituir algumas das plataformas mencionadas acima, e irá usar os dados presentes em outras para gerar *dashboards* de métricas que ajudem à gestão de projectos e da própria empresa. O sistema deverá:

- integrar com o sistema de gestão de projectos actualmente em uso na Ubiwhere, o Redmine. Este sistema continuará em uso, ficando o UIS encarregue de obter os dados relevantes;
- substituir os antigos sistemas da Intranet, URM e Google Spreadsheets para a gestão de presenças, portfólio e métricas financeiras/alocações, respectivamente, centralizando os dados por elas anteriormente gerados;
- possibilitar a recolha de dados financeiros e de gestão de projecto para a geração de *dashboards* de métricas, utilizando os dados gerados pela implementação dos dois pontos acima;
- ter uma arquitectura que que seja suficientemente flexível para poder recolher e fornecer dados de diferentes plataformas, não ficando dependente nem dos sistemas que alimentam esta plataforma, nem do *frontend* que seja especificado.

Assim, decidiu-se que deveria ser criada uma API *RESTful* [14] que agregue todas estas funcionalidades e serviços, disponibilizando a informação em formatos facilmente interpretáveis por outras plataformas. Esta será a parte nuclear do SI da empresa. O desenvolvimento desta API é o âmbito do estágio.

Ao criar um sistema com estas características, a empresa passará a conseguir recolher informações, praticamente em tempo real, sobre as várias áreas de gestão, sejam de projecto, financeiras ou de recursos humanos. Assim, o estagiário acredita que os responsáveis pelas tomadas de decisão nestas áreas da empresa passarão a ter à sua disposição mais dados (e mais relevantes) para executar o seu trabalho.

Fora do âmbito do estágio ficam alguns aspectos como:

- O *design* da interface da aplicação e a implementação da mesma. Estas tarefas ficarão a cargo dos *designers* da empresa.
- A construção de testes. Os testes serão implementados pelo departamento de QA da empresa.

1.4 Estrutura

No capítulo 2, é feito o levantamento e descrição das principais ferramentas disponíveis neste momento que têm funcionalidades semelhantes às que se desejam implementar neste produto. Esta secção está dividida em diferentes categorias de produtos, em que se comparam as diferentes funcionalidades de cada um.

No capítulo 3, são apresentados os métodos de trabalho usados durante o estágio, bem como o planeamento do trabalho e sua execução.

No capítulo 4, é feita a especificação técnica do sistema. Aqui, são apresentados os requisitos funcionais do sistema, sob a forma de *user stories*, e os requisitos não-funcionais. É apresentada também a arquitectura do UIS, bem como os modelos de dados subjancentes. É ainda feita a descrição dos módulos e do seu funcionamento numa perspectiva de alto nível. Finalmente, são especificadas as ferramentas usadas para o desenvolvimento do sistema, sendo explicadas as razões que levaram à sua escolha.

Depois disto, no capítulo 5, é feita uma análise a alguns detalhes mais específicos sobre a implementação do sistema, com uma explicação de mais baixo nível dos métodos mais importantes. É também apresentado o ambiente de testes usado durante a implementação e os resultados finais desses mesmos testes.

O capítulo 6 consiste numa reflexão sobre como o trabalho desenvolvido pode servir de base a uma *framework* de geração de métricas, que poderá por sua vez ser aproveitada por outros sistemas e empresas.

Finalmente, no capítulo 7 são apresentadas as conclusões sobre o trabalho, bem como o trabalho futuro a desenvolver.

Capítulo 2

Estado da Arte

Nesta secção, são apresentados os resultados da pesquisa efectuada sobre o estado da arte das ferramentas com funcionalidades que deverão estar presentes no sistema desenvolvido. Dado o âmbito muito alargado deste projecto, existem vários tipos de ferramentas com estas funcionalidades. Assim sendo, o estudo está dividido em vários grupos, que são os seguintes:

- PPM: Este tipo de ferramentas é o mais relevante, sendo que o seu objectivo é semelhante ao deste projecto: melhorar a visibilidade dos dados dos projectos da empresa, permitindo a recolha de métricas globais sobre estes, alimentadas pelas métricas de granularidade mais baixa (*sprint* e cada projecto). Algumas destas ferramentas permitem também, em maior ou menor grau, controlar a própria execução de cada projecto.

- ALM: Este tipo de ferramentas tem algumas características em comum com as de PPM, tendo muitas vezes algumas capacidades a nível de análise e manutenção do portfólio de projectos. No entanto, estas são mais dedicadas ao desenvolvimento das aplicações, desde o planeamento à entrega final.

- ERP: Estas ferramentas são as mais abrangentes deste grupo, tratando de quase todos os aspectos relacionados com a gestão de uma empresa. Aqui, a sua inclusão deve-se ao facto de terem algumas funcionalidades que não são abrangidas pelas anteriores, e que também são necessárias aqui (nomeadamente no que diz respeito à gestão de RH).

- Construção de *dashboards*: Estas ferramentas permitem fazer a construção e visualização de *dashboards* com a informação a apresentar. A análise a ferramentas exclusivamente dedicadas a este aspecto justifica-se com a importância atribuída a esta parte do projecto.

No final de cada uma das secções seguintes, é apresentada uma conclusão sobre o conjunto de ferramentas analisado. Visto que as ferramentas de PPM são, como já referido, as mais semelhantes ao produto que queremos, é feita uma comparação entre as características de cada uma delas. Esta comparação não é feita nas conclusões referentes aos outros grupos de ferramentas, dado que foram incluídas no estudo numa perspectiva de tentar encontrar funcionalidades interessantes para o sistema. Desta forma, não seria relevante apresentar uma comparação directa entre cada uma, até porque foram analisadas ferramentas bastante distintas que seriam difíceis de analisar em conjunto. Em vez disso, no final das secções respectivas, são destacadas as funcionalidades adicionais encontradas.

2.1 Project and Portfolio Management (PPM)

O principal objectivo das ferramentas de PPM é centralizar a informação sobre todos os projectos do portfólio da empresa. Assim, os gestores de topo passam a ter acesso a análises detalhadas sobre o conjunto dos processos, e os PMs passam a ter informação detalhada sobre os seus projectos.

Estas são ferramentas bastante completas, que devem cobrir, com maior ou menor detalhe, todos os aspectos de planeamento de recursos (financeiros ou não), gestão de projecto e geração e visualização de métricas. Visto que todas as ferramentas analisadas apresentam características semelhantes, são mais focados os aspectos que as diferenciam ou em que se destacam.

De entre estas ferramentas, algumas são *cloud-based*, enquanto que outras podem ser alojadas internamente. As primeiras têm algumas vantagens como, por exemplo, serem geralmente mais baratas (e permitirem subscrições por um dado período de tempo) e não terem que ser configuradas, facilitando a sua adopção inicial. No entanto, isto traz algumas desvantagens, como o facto de poderem não ser tão adaptáveis à empresa e o facto de os dados serem guardados na *cloud*, o que levanta algumas questões a nível de segurança e confidencialidade. O mesmo se aplica aos restantes grupos de ferramentas.

No final da secção, as ferramentas de PPM são comparadas entre si, tendo em conta as funcionalidades que o UIS deverá ter. Para além destas, são ainda acrescentadas outras que podem ser consideradas como interessantes para este sistema, ainda que apenas numa perspectiva de trabalho futuro.

CA Clarity PPM

O Clarity PPM é uma solução da empresa CA[1]. Tem uma arquitectura de *web services*, contendo uma série de módulos que permitem adequar as funcionalidades obtidas às necessidades da empresa.

Pode ser adquirido como Software as a Service (SaaS), sendo esta uma versão *on-demand*, ou seja, em que podem ser adquiridos apenas os módulos que forem necessários, em adição aos básicos. Pode também ser alojado na empresa.

Uma das características diferenciadoras que esta ferramenta tem é a integração com outras ferramentas da mesma empresa, que resultam numa grande diversidade de funcionalidades. Entre estas, inclui-se um portal no qual os colaboradores e clientes podem partilhar ideias entre si, podendo estas ser votadas pelos restantes membros, o que pode facilitar a inovação dentro da empresa.

Compuware Changepoint

O Changepoint[2] apresenta uma solução robusta e completa. Esta ferramenta pode ser adquirida como SaaS, ou ser administrada localmente.

Um dos principais factores de destaque é a ferramenta de *time-tracking*, que é muito completa, permitindo fazer o *tracking* em modo *offline*, reportar tempo mesmo que não esteja directamente relacionado com o projecto (férias, ausências, etc.), entre outros aspectos. Isto faz com que os PMs possam facilmente gerar relatórios detalhados sobre a performance dos colaboradores, usando métricas como horas despendidas por *task*, projecto, *request* ou mesmo cliente.

Oracle Primavera EPPM

O Oracle Primavera EPPM[3] é uma solução muito robusta e complexa, que é mais indicada para a gestão de grandes projectos, inerentemente complexos.

Uma das funcionalidades a destacar nesta solução é o módulo de gestão de riscos. Este módulo oferece uma grande variedade de tipos de riscos definidos como na matriz SWOT (matriz usada na definição estratégica dos pontos fortes e fracos de uma empresa em oportunidades ou ameaças[4]), podendo o

[1] `http://www.ca.com/us/products/detail/ca-clarity-ppm.aspx`

[2] `http://www.compuware.com/en_us/changepoint/solutions/project-portfolio-management.html`

[3] `http://www.oracle.com/us/products/applications/primavera/overview/index.html`

[4] `http://www.swotmatrix.com/`

utilizador associar a cada risco uma probabilidade de acontecer, um prazo no qual pode suceder, e o custo estimado que terá. Para além disso, podem ainda ser associados aos riscos acções de prevenção ou mitigação, por forma a poder responder mais rapidamente.

HP PPM

O *software* de PPM da HP[5] distingue-se por ser muito completo, através da integração com vários produtos da mesma empresa. Por exemplo, ao integrar com o HP Agile Manager, esta ferramenta oferece uma solução para o planeamento e execução de projectos segundo metodologias ágeis de maneira mais completa que as restantes.

Outras características importantes desta ferramenta são o versátil motor de criação de *workflows* e a interface intuitiva, que foi redesenhada na última versão, tendo um aspecto actual e *user-friendly*.

Esta é uma solução que é obtida como SaaS.

Innotas

O Innotas[6] é uma solução *cloud-based* disponibilizada como SaaS.

Um dos seus pontos fortes é a alocação de recursos para os projectos. Através da sua interface gráfica, o Innotas permite perceber rapidamente quais os recursos que ainda podem ser alocados a um projecto, bem como perceber quais os recursos já alocados. Também é possível filtrar esta informação por diversos campos (tipo de recurso, datas, projecto, etc.). Esta informação é depois usada para criar métricas sobre o desempenho dos colaboradores pelos vários projectos da empresa.

Daptiv PPM

O Daptiv PPM[7] é uma solução SaaS disponibilizada na *cloud*. Este é apresentado como um factor benéfico para as empresas, pois reduz o custo de adopção, enquanto promove alguma flexibilidade devido ao facto de ter vários módulos.

Outro ponto importante desta ferramenta é a capacidade de integração com um grande número de ferramentas de Project Management como o Jira, Rally, Salesforce, etc.

[5] `http://www8.hp.com/us/en/software-solutions/software.html?compURI=1171920#.UuFcGxDFK00`

[6] `http://www.innotas.com/`

[7] `http://www.daptiv.com/`

Finalmente, a empresa também disponibiliza algumas das funcionalidades deste produto em plataformas *mobile*, desenvolvidas em HyperText Markup Language (HTML) 5.

Upland PowerSteering

O PowerSteering[8] é outra solução SaaS.

Esta ferramenta apresenta como principal ponto forte as capacidades de *reporting*. O "Visual Portal" permite a visualização de qualquer *dashboard* criado de uma maneira flexível, enquanto que a criação dos *dashboards* em si é muito adaptável, permitindo ao utilizador criar as suas próprias métricas para cada nível do projecto ou portfólio. Os relatórios podem também ser exportados para PDF, Excel, etc.

AtTask

O AtTask[9] é mais uma solução do tipo SaaS. O principal ponto que tem a favor é o forte suporte dado à gestão do trabalho propriamente dito, com suporte directo às metodologias ágeis, algo que outras ferramentas nesta secção não possuem. Isto faz com que os PMs possam planear as iterações e ver o estado do trabalho num só lugar, característica que é essencial para o UIS.

Outras características importantes que diferenciam esta ferramenta são o facto de ter uma interface bastante *user-friendly*, nomeadamente no que toca à execução do trabalho e tempo gasto, sendo que o utilizador facilmente consegue registar o tempo, organizar as tarefas e comentá-las. É ainda disponibilizada uma aplicação *mobile* para iPad, na qual é possível ver alguns relatórios com algum grau de personalização.

Planview Enterprise

O Planview Enterprise[10] pode ser adquirido como SaaS ou instalado localmente.

O suporte directo dado às metodologias ágeis é uma das grandes vantagens desta ferramenta. Esta permite criar *roles* baseados nos do Scrum (POs, Scrum Masters, etc.), gerindo automaticamente as permissões associadas. Permite também fazer o planeamento dos *sprints*, bem como a sua gestão, de forma simples. Esta gestão é feita através da disponibilização de métricas de *sprint* e de taxa de implementação das *user stories* definidas.

[8] `http://www.powersteeringsoftware.com/`

[9] `http://www.attask.com/`

[10] `http://www2.planview.com/`

Miscrosoft Project Server

A solução apresentada pela Microsoft[11] pode ser adquirida como SaaS, como parte da solução Office365, ou pode ser alojada internamente.

Esta é uma plataforma que apresenta várias características que se destacam. De entre estas, uma das mais importantes para o projecto é a capacidade de associar materiais a tarefas. Isto é importante visto que a empresa necessita de saber a melhor maneira de alocar os recursos de *hardware* disponíveis pelos seus colaboradores.

Outra capacidade interessante é a de representar visualmente as tarefas sob a forma de uma linha temporal. Isto permite que os colaboradores tenham imediatamente ao seu dispor uma maneira de verificar em que estado está o trabalho que têm para desenvolver, em termos de metas temporais.

Clarizen

O Clarizen[12] é uma ferramenta que se destaca pela introdução de algumas funcionalidades inovadoras. Destas, destaca-se a forte integração com o email. Visto que esta é uma ferramenta que mesmo os colaboradores sem grande conhecimento a nível de tecnologias costumam dominar, esta é uma funcionalidade muito interessante: podem criar-se tarefas, podem ser enviados *bug reports* e até mesmo projectos, através do email.

Outra característica interessante e que se destaca é relativa à colaboração entre as pessoas alocadas a um projecto, existindo uma espécie de fórum em que os utilizadores podem mesmo referenciar através de ligações qualquer item, seja um projecto, tarefa ou documento.

Conclusões

Nesta secção, são comparadas as ferramentas apresentadas anteriormente. Na tabela 2.1, é feita uma comparação entre cada ferramenta, sendo indicado se esta possui ou não as funcionalidades consideradas importantes para o UIS. Para além destas características, são ainda estudadas outras que, não sendo consideradas importantes para o sistema, para já, podem vir a ser interessantes numa perspectiva de trabalho futuro.

As funcionalidades aqui enunciadas são baseadas nas necessidades que a empresa tem actualmente, e que deverão ser incorporadas no UIS, e as quais estiveram na base da especificação das *user stories*.

[11] `http://office.microsoft.com/en-us/project/project-management-and-ppm-showcase-microsoft-project-FX103802304.aspx`

[12] `http://www.clarizen.com/`

	Clarity	Changepoint	Primavera	HP PPM	Innotas	Daptiv	PowerSteering	AtTask	Planview	Project Server	Clarizen
Alocação de equipas	❍	●	❍	❍	●	❍	●	●	❍	❍	●
Alocação de utilizadores	●	●	●	●	●	●	●	●	●	●	●
Ver esforço despendido	●	●	●	●	●	●	●	●	●	●	●
Métricas por *sprint*	◗	❍	❍	●	❍	❍	❍	●	●	❍	●
Métricas por projecto	●	●	●	●	●	●	●	●	●	●	●
Reporting em *dashboards*	●	●	●	●	●	●	●	●	●	●	●
Gestão de *sprints*	❍	❍	❍	●	❍	❍	❍	●	●	❍	●
Gestão de equipas	◗	●	●	❍	●	●	●	●	●	●	●
Gestão de férias	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	●
Planeamento de custos	●	●	●	●	●	●	●	❍	●	●	❍
Métricas sobre custos	●	●	●	●	●	●	●	❍	●	●	●
Planeamento de riscos	●	●	●	●	●	❍	❍	❍	●	●	●
Criação de novas métricas/KPI	●	❍	❍	❍	❍	❍	●	❍	●	●	❍
Deployment in-house	●	●	●	❍	❍	❍	❍	❍	●	●	❍
Gestão de processos	●	●	●	●	❍	❍	●	●	●	●	●
Gestão de inventário	❍	❍	❍	❍	❍	❍	❍	❍	❍	●	❍
Gestão de necessidades de formação	❍	❍	❍	❍	❍	❍	●	❍	❍	❍	❍
Skills de colaboradores	●	●	●	❍	❍	●	●	❍	●	●	●
Motor de *Workflows*	●	●	●	●	❍	❍	❍	●	●	❍	❍

(a) Característica:
●– Incluída
❍– Não incluída
◗– Parcialmente suportada

Tabela 2.1: Comparação de aplicações de PPM

Analisando a tabela 2.1, podemos ver que, embora todas as ferramentas possuam muitas das características que o UIS deverá ter, há algumas que nenhuma delas tem, tal como a alocação provisória de recursos ou a gestão de férias dos colaboradores. Para além disso, podemos ver também que as metodologias ágeis não são suportadas por muitos destes produtos, pelo menos directamente (sem recurso à integração com outras plataformas).

Concluindo, a nível de funcionalidades, e embora nenhuma ferramenta tenha todas as funcionalidades desejadas, algumas delas poderiam ser utili-

zadas como alternativa ao sistema que estamos a desenvolver. No entanto, o UIS terá vantagens para a empresa visto que está a ser desenvolvido para a sua situação específica. Assim, o leque de funcionalidades desenvolvido é adaptado às necessidades da empresa no momento, enquanto que a arquitectura do sistema permite uma expansão rápida no futuro, caso seja necessário. Para além do mais, por se tratar de uma API, existe a possibilidade de suportar múltiplas plataformas, enquanto que algumas destas soluções são, por exemplo, exclusivamente *web-based*. Isto facilita ainda a integração com os serviços que mais interessem à empresa, e o rápido desenvolvimento de módulos que possibilitem a integração de novos serviços, à medida que seja necessário. Os sistemas apresentados acima não permitem isto, ou permitem através de maiores custos e complexidade.

Finalmente, e como é explicado nas secções abaixo, há várias funcionalidades de outros tipos de sistema que seria muito importante estarem presentes na solução final. Isto seria mais difícil de alcançar usando apenas um tipo de ferramenta. No caso do produto desenvolvido neste projecto, e tendo em conta a arquitectura flexível mencionada acima, será bastante mais fácil de integrar estas novas funcionalidades do que em qualquer uma destas ferramentas.

2.2 Application Lifecycle Management (ALM)

As ferramentas de ALM têm como principal propósito a gestão do "ciclo de vida" das aplicações, ou seja, tratam da gestão de todos os aspectos envolvidos desde a sua criação até ao lançamento (e também da manutenção). Como tal, estas ferramentas possuem mecanismos para lidar com a definição e gestão de requisitos, integração com ferramentas de desenvolvimento e de repositórios, gestão de projecto, metodologias ágeis e qualidade.

Team Foundation Server

O Visual Studio Team Foundation Server é a solução da Microsoft em termos de sistemas de ALM[13]. Sendo da Microsoft, esta ferramenta permite a integração com várias outras da mesma empresa, tanto dedicadas à gestão de projecto (MS Project), como de âmbitos diferentes (Excel, Word, entre outros, são usados, por exemplo, na definição de requisitos). Esta ferramenta também se destaca pelo forte suporte dado às metodologias ágeis e, nomeadamente, ao Scrum. Assim, é possível fazer a gestão de *backlog*, dos *sprints* e das tarefas a realizar em cada um.

[13] `http://msdn.microsoft.com/library/vstudio/fda2bad5`

Uma das características mais interessantes encontradas aquando da análise destes produtos foi a capacidade de gerar diagramas de sequência directamente a partir do código fonte. Sendo um dos objectivos deste projecto ter um maior controlo sobre os projectos, enquanto se elimina trabalho repetitivo e que não é, directamente, produtivo, esta é uma capacidade relevante. Desta forma, é fácil gerar documentação que facilita bastante a compreensão do código, sem que isso implique trabalho adicional para os programadores.

Jira

O Jira é uma ferramenta popular, criada pela Atlassian, para a gestão de projecto e ALM[14]. Uma das grandes vantagens que tem é ser muito flexível, uma vez que tem uma grande quantidade de *add-ons* disponíveis através da sua loja *online*.

Uma das possibilidades interessantes com esta ferramenta é a importação de *workflows* desta loja *online*. Esta funcionalidade dá aos utilizadores a possibilidade de importar uma grande quantidade de *workflows* que definem as tarefas mais comuns na gestão e desenvolvimento dos projectos, o que pode ajudar a simplificar estes processos. Para além de os poder importar, o utilizador tem ainda a possibilidade de os personalizar, caso estes não correspondam exactamente aos processos da sua empresa. Outra das vantagens desta plataforma é a integração com o Hipchat. O Hipchat[15] é uma aplicação de Instant Messaging (IM) desenvolvida especialmente para a colaboração em projectos. Assim, para além de ter características desenvolvidas para facilitar a colaboração entre todos os membros envolvidos no projecto, a integração com o Jira permite receber notificações sobre *commits*, alterações do estado do projecto, ou transições nos *workflows*, directamente no *chat*.

Rally Platform

O Rally é uma plataforma cujo foco está na gestão de projectos ágeis[16]. Esta plataforma é disponibilizada como SaaS, embora possa, segundo a empresa, ser disponibilizada *on premises* em algumas condições.

A característica que mais se destaca nesta ferramenta é o ênfase que é dado à visão geral de todo o projecto. Assim, é dada ao utilizador a possibilidade de verificar facilmente, e num só ecrã, todo o tipo de informações sobre o projecto. Esta ferramenta possui *dashboards* personalizáveis que a tornam bastante intuitiva.

[14] `https://www.atlassian.com/software/jira`
[15] https://www.atlassian.com/software/hipchat/overview
[16] `http://www.rallydev.com/about/what-is-rally`

Collab TeamForge

O Collab TeamForge[17] é mais uma ferramenta ALM que é fornecida na *cloud*. O TeamForge também é dedicado às práticas ágeis.

Algumas das características mais interessantes desta plataforma são a capacidade de gerar *templates* para projectos, permitindo assim que as práticas e processos da empresa sejam mais facilmente adoptados por todos os projectos do portfólio desta. Outra funcionalidade bastante útil é a de gerar facilmente *micro-sites* para cada projecto. Estes *sites* podem também ser adaptados de um *template* e adaptados consoante os projectos. Esta funcionalidade permite uma maior integração de todos os *stakeholders* no processo de desenvolvimento do projecto, através de um local onde se pode guardar informação sobre este, podendo ficar acessível a pessoas que não pertençam à empresa.

Tuleap

O Tuleap[18] é uma ferramenta que se destaca das restantes aqui referidas, em primeiro lugar, por ser *open-source*.

No entanto, a característica que mais se destaca nesta ferramenta é a sua usabilidade. Isto é particularmente evidente na gestão das tarefas ágeis. Aqui, o utilizador tem ao seu dispor uma interface *drag and drop*, que permite pegar em cada tarefa, representada por um bloco, e largá-la na área correspondente ao seu estado actual (por exemplo, *ongoing*, *review*, *done*, etc.).

Conclusões

Embora as ferramentas de ALM estejam mais viradas para o nível de execução do projecto do que o trabalho deste relatório deverá estar, a sua análise é relevante na medida em que foi possível extrair algumas ideias sobre capacidades que o produto poderá vir a ter no futuro. Ao estudar estas ferramentas, foi possível verificar que uma boa parte delas tentam reunir as funcionalidades de PPM e ALM (e o contrário também se verifica). Este facto também atesta a utilidade de ter um produto que junte estas funcionalidades.

Assim, as características mais relevantes encontradas foram:

- Geração de diagramas de sequência a partir do código fonte
- Utilização de *workflows* pré-definidos, com possibilidade de os adaptar

[17] `http://www.collab.net/products/teamforge`
[18] `https://tuleap.net/`

- Integração com plataformas de mensagens instantâneas como o HipChat
- Adaptabilidade e usabilidade dos ecrãs de gestão e informação dos projectos
- Criação de *micro-sites* para cada projecto
- Templates para a criação de projectos
- Interface *drag-and-drop* para gestão de tarefas

2.3 Enterprise Resource Planning (ERP)

Nesta secção, apresentamos as soluções de ERP encontradas que se destacam por serem bastante completas ou por serem muito competentes em alguns aspectos em específico.

Destas ferramentas, o OpenERP é *open-source*, destacando-se pelo grande suporte da comunidade e quantidade de módulos existentes.

Relativamente às restantes, o SAP ERP é a solução da maior empresa da área (e que detém maior quota de mercado [15]) e uma das mais completas que existe. Finalmente, a NetSuite foi uma das empresas da área com maior crescimento em 2012 [15], sendo que o seu produto se destaca essencialmente pela capacidade de *reporting* e *Business Intelligence*.

OpenERP

O OpenERP[19] é uma das ferramentas *open-source* mais populares de ERP. A interface da plataforma Web é bastante *user-friendly* e tem muitas opções no que toca à visualização de informação.

Uma das principais características prende-se com o facto de ter disponíveis muitos módulos que lhe conferem todo o tipo de funcionalidades. Um destes módulos, cujas funcionalidades seriam muito úteis para o UIS, é o módulo de gestão de férias e ausências. Este módulo permite definir o tipo de ausência do funcionário (férias, doença, licenças com vencimento, etc.) e atribuir-lhe essa ausência. Também pode ser o funcionário a colocar um pedido de ausência, ou um administrador a atribuir um período de ausência a um dado grupo de utilizadores, por exemplo. Para além disto, este módulo permite ainda aceder a relatórios e análises sobre as ausências (por departamento, por tipo, etc.).

[19] `https://www.openerp.com/`

A gestão de ausências era um módulo já planeado para o UIS, mas algumas funcionalidades, como os diferentes tipos de ausências ou a criação de relatórios, podem ser adicionadas para aumentar as capacidades da nova plataforma.

SAP ERP: SuccessFactor

A SAP é a líder mundial na produção de software de ERP. É uma das soluções de ERP mais completas, oferecendo módulos que respondem a quase todas as necessidades da gestão de uma empresa de qualquer tipo e dimensão.

O módulo que destaco aqui é o de SuccessFactor[20]. Neste módulo, que pode ser integrado com o módulo de Human Capital Management do SAP ERP, é possível fazer a gestão dos RH da empresa de acordo com as suas valências. Assim, um administrador pode manter um perfil detalhado de cada colaborador, com aspectos como a sua formação, principais *skills*, certificações, etc., podendo depois procurar os colaboradores que mais se adequém a uma dada tarefa ou projecto. Este módulo tem outras funcionalidades importantes, como o estabelecimento e análise de objectivos por colaborador, e a possibilidade de atribuição de *badges* ou comentários positivos a outros colaboradores, aquando da realização de uma tarefa.

NetSuite

O NetSuite[21] é uma ferramenta baseada na *cloud* que se destaca face às restantes pelas funcionalidades em termos de *Business Intelligence*. De facto, esta plataforma oferece uma grande versatilidade em termos de análise e *report* de dados, disponibilizando as métricas definidas em tempo real.

No entanto, algumas das *features* que se destacam nesta ferramenta são:

- Ferramentas que ajudam na colaboração entre os trabalhadores. Aqui, destaca-se um calendário que permite ver as tarefas alocadas a cada colaborador numa só vista, o que facilita a tarefa dos gestores de projecto e dos outros colaboradores, permitindo ter rapidamente uma noção de quem está a fazer que tarefas.

- Introdução de elementos de *gamification*, como *leaderboards* referentes ao desempenho dos funcionários e atribuição de métricas específicas a atingir. Este é um aspecto interessante e que poderia ser utilizado como motivação para manter os colaboradores produtivos.

[20] `http://www.successfactors.com/en_us.html`
[21] http://www.netsuite.com/portal/home.shtml

Conclusões

No final desta secção, interessa então destacar as funcionalidades mais interessantes retiradas de cada um destes produtos, que poderão ser incorporadas no conjunto de funcionalidades do UIS. Estas são:

- Funcionalidades extra relativas ao módulo de gestão de férias, como atribuição por departamento, vários tipos de ausências, etc.;
- Funcionalidades de gestão de RH relativas aos *skills* dos funcionários: manutenção de uma "rede" de funcionários e das suas capacidades, que permite saber que recursos alocar a que projectos;
- Calendário com todos os colaboradores e tarefas que lhes estão atribuídas;
- Introdução de conceitos de *gamification*, como a criação de *leaderboards* relativas ao desempenho, para aumentar a produtividade dos colaboradores.

Estas funcionalidades presentes nos sistemas de ERP analisados, a serem introduzidas no UIS, aumentariam, sem dúvida, a sua utilidade e o seu valor para a empresa.

2.4 Ferramentas para *Dashboards*

Dada a importância que foi atribuída à visualização de dados e geração de *dahsboards*, são apresentadas algumas ferramentas que tratam da apresentação de dados.

A visualização da informação que é gerada pelo sistema é, de facto, um dos pontos fulcrais deste projecto. Para que haja uma análise correcta e eficaz das métricas e dados gerados, há que conseguir apresentá-los de forma facilmente compreensível ao utilizador final.

Assim, interessa analisar algumas ferramentas para geração de *dashboards*, com vista a idealizar a maneira como o sistema deve apresentar a informação.

Tableau

O Tableau[22] é uma ferramenta que oferece grande flexibilidade e controlo sobre os *dashboards* criados.

De facto, esta ferramenta permite obter dados de várias fontes (vários tipos de bases de dados, ficheiros Excel, etc.), e manipulá-los de uma forma

[22] `http://www.tableausoftware.com/`

interactiva (*drag and drop*) para criar *dashboards*. Pode definir-se um *dashboard* com o número e distribuição que quisermos de gráficos e painéis.

Para além disto, também oferece muitas opções em termos de visualização de dados, com diversos tipos de gráficos à disposição, e várias maneiras de manipulá-los (*slidebars*, *zoom*, etc.).

Uma das características interessantes deste produto é a opção *forecasting*. Esta opção permite, para um gráfico gerado anteriormente, fazer uma previsão para um dado período de tempo de como esses dados vão evoluir. Isto poderia ser útil no UIS em vários níveis de análise: poderia ser usado, por exemplo, para prever a evolução de *burndown charts*, o que permitiria ter uma melhor ideia do tempo restante para terminar as tarefas associadas a um projecto. Poderia ainda ser usado ao nível financeiro, para prever a evolução dos custos de um projecto e dos desvios em relação aos orçamentos inicialmente previstos.

Qlikview

O Qlikview[23] é outra ferramenta muito completa. Nesta, destaca-se principalmente o seu sistema visual de encontrar relações entre objectos numa página, através de um código de cores (quando um objecto é seleccionado, é destacado a verde, e todos os que estão relacionados com ele a branco).

No entanto, uma das *features* que poderia ser mais útil no âmbito deste projecto é a possibilidade de partilhar *dashboards*. Assim, um utilizador pode criar um *dashboard* personalizado, com várias análises relevantes na mesma página, tendo depois a possibilidade de o partilhar com outros colaboradores. Os utilizadores têm ainda a possibilidade de associar comentários aos *dashboards* a que têm acesso. Isto faz com que se facilite o acesso à informação mais relevante por parte das pessoas que lhe devem aceder.

Dundas

O Dundas[24] é mais uma ferramenta que permite gerir, de forma integrada, todo o processo de Business Intelligence, desde a recolha de dados até à sua visualização,

Uma característica útil deste produto foi introduzida na sua última versão: a geração de relatórios periódicos. Com esta opção, o utilizador pode seleccionar o *dashboard* que entender, com filtros à sua escolha, e seleccionar um intervalo temporal para gerar automaticamente os relatórios (diário, semanal, mensal ou definido pelo utilizador). Para além disto, o utilizador pode

[23] `http://www.qlikview.com/`
[24] `http://www.dundas.com/`

ainda escolher qual o formato do relatório (imagem, ficheiro Excel ou simplesmente um *link* para o relatório). Finalmente, há ainda a opção de enviar automaticamente os resultados via email para recipientes seleccionados, ou guardar estes resultados em disco.

Spotfire

O Sporfire[25] é outra ferramenta de visualização e geração de *dashboards* que é facultada através da *cloud*.

O que distingue esta ferramenta das restantes é o facto de oferecer uma personalização mais alargada. Isto deve-se ao facto de o utilizador poder usar código HTML para gerar páginas com *dashboards* personalizados, de acordo com as necessidades. Desta maneira, enquanto que, nas outras ferramentas, a personalização está limitada ao que a ferramenta permite fazer, com a utilização do HTML as possibilidades são muito mais alargadas.

Conclusões

Após a análise das ferramentas para a construção de *dashboards* e visualização de dados, foram descobertas algumas funcionalidades interessantes, que podem ajudar a melhorar a solução final. De entre estas funcionalidades, discutidas anteriormente, destacam-se:

- A capacidade de fazer previsões, usando os dados actuais como *input* para algoritmos como os de *time-series*;
- Gerar relatórios automaticamente em certos períodos de tempo pré-determinados;
- Permitir alguma flexibilidade a nível da disposição dos componentes na página;
- Partilha de relatórios e *dashboards* gerados
- Associação de notas e comentários a relatórios e *dashboards*

2.5 Conclusões gerais

O estudo das ferramentas que compõem o Estado da Arte actual permitiu ter uma melhor compreensão sobre quais as funcionalidades que um bom SI deve ter. A inclusão de vários tipos de ferramentas deveu-se ao facto de

[25] `http://spotfire.tibco.com/`

ser necessário desenvolver um sistema que dê resposta a vários problemas diferentes. O posicionamento do UIS como uma camada de serviços permite também que, no futuro, caso se determine que alguma destas ferramentas pode ser usada na empresa para tratar de um problema em específico, os dados por elas gerados possam ser aproveitados e centralizados. O facto de compreender um conjunto tão alargado de ferramentas possibilitou também a elicitação de algumas funcionalidades e características que poderão ser acrescentadas ao produto final do estágio. Isto serviu de base às *user stories* sugeridas pelo aluno, apresentadas mais à frente na tabela 4.11.

Capítulo 3

Metodologia

Nesta secção, é explicado o processo de desenvolvimento de *software* utilizado ao longo do estágio, bem como o planeamento efectuado e a sua comparação com o trabalho realizado.

3.1 Desenvolvimento de *software*

Como já foi referido atrás, a Ubiwhere usa metodologias ágeis no desenvolvimento de *software*. Como tal, foi usado o Scrum no desenvolvimento deste projecto.

Os *sprints* tiveram, maioritariamente, uma duração de 15 dias. Houve algumas excepções, que se deveram a factores como a elaboração deste relatório, variação do número e complexidade de funcionalidades a implementar. No início dos *sprints*, houve lugar a uma reunião com o PM para discutir estas funcionalidades e a sua implementação, havendo também lugar a uma reunião no fim dos *sprints* para fazer o ponto da situação e verificar o trabalho realizado. A execução de tarefas foi planeada a partir das *user stories* apresentadas na secção 4.1, que foram divididas em tarefas mais pequenas.

O Redmine foi usado para consultar as *user stories* e tarefas atribuídas. Nesta ferramenta são ainda registadas informações sobre qual o *sprint* a que cada tarefa corresponde, qual o tempo de início e o tempo previsto para terminar, entre outras. Esta ferramenta também serviu para registar o tempo despendido na realização de cada tarefa, bem como a percentagem de trabalho já realizado para cada tarefa.

Em termos de código, foi usado o Eclipse, juntamente com o *plugin* Aptana[1], dedicado ao desenvolvimento de aplicações *web*. O Aptana permite fazer o *upload* do código para o servidor de testes de maneira simples.

[1] `http://www.aptana.com/`

Para além disto, é usado ainda um repositório Git, alojado numa instância interna da Ubiwhere suportada pelo Gitolite[2], com todo o código fonte. O cliente usado para aceder ao Git foi o Source Tree[3], que permite, através de uma interface gráfica simples, verificar as alterações do projecto e criar *branches*. Foram utilizadas três *branches*: uma de desenvolvimento, usada pelo estagiário, outra de QA, para a especificação de testes e sua realização, e outra de produção. Foi sendo feito um *merge* periódico da branch de desenvolvimento para a de QA para o estabelecimento de *builds* estáveis, sendo também feito o *merge* desta para a de produção quando já possuía um conjunto de funcionalidades suficiente e aprovado pela QA.

O estagiário foi incluído numa equipa com outros 3 elementos, com as funções de *designer*, *tester* e PM. As responsabilidades do estagiário passam por desenvolver a API do sistema, garantindo que esta se comporta da maneira expectável, passando os testes definidos pelo *tester* da equipa. Em termos da tecnologia a usar, houve total liberdade para escolher todas as bibliotecas necessárias para o desenvolvimento da plataforma. O estagiário esteve também encarregue de produzir toda a documentação que permita a compreensão e fácil desenvolvimento futuros. Finalmente, também ficou a seu cargo a tarefa dede definir as estruturas de dados necessárias para dar resposta aos requisitos apresentados.

Em termos do processo de desenvolvimento de *software* propriamente dito, para cada módulo, foi feita a modelação da base de dados antes do início do desenvolvimento. Todos estes modelos foram submetidos a uma validação externa. Foram elaborados os diagramas necessários para cada módulo (ER e, em alguns casos, diagramas de actividade para clarificar as funcionalidades), sendo os módulos posteriormente implementados e submetidos à equipa de QC/QA para a criação de testes. Aquando da aprovação nos testes, o trabalho desenvolvido era submetido para os repositórios de código da empresa. Para além disto, como referido, foi gerada toda a documentação necessária para cada módulo.

Há ainda que referir a criação de migrações usando o *south*[4]. Estas migrações especificam o estado das tabelas de cada módulo num dado momento. Isto permite fazer a criação das tabelas respectivas na base de dados. Para além disso, as migrações permitem manter uma série de transformações para que, caso haja alguma modificação nas tabelas (por exemplo, se for introduzido ou modificado um atributo ou se existir uma nova relação), a base de dados possa ser migrada para reflectir a versão mais recente. Assim, garante-

[2] `http://gitolite.com/gitolite/`

[3] `http://www.sourcetreeapp.com/`

[4] Ferramenta para a criação de migrações, `http://south.readthedocs.org/en/latest/index.html`

se que, caso seja introduzida alguma modificação errada, pode reverter-se as tabelas de um módulo para o estado anterior. Outra vantagem da utilização de migrações é o facto de permitirem que os dados sejam mantidos aquando da modificação de tabelas. Estas tabelas são criadas através do mecanismo de ORM da *framework* usada para o desenvolvimento da aplicação, o Django, como será explicado mais à frente.

3.2 Planeamento

Nesta secção, são apresentados o planeamento inicial, o trabalho realizado e respectivos desvios relativamente ao planeamento.

3.2.1 Planeamento inicial

Numa fase inicial, estavam previstas cinco grandes *milestones* para o projecto, definidas da seguinte forma:

- T1 – Análise dos sistemas existentes na empresa;
- T2 – Análise e especificação dos requisitos do sistema;
- T3 – Definição da arquitectura para as várias funcionalidades pretendidas.
- T4 – Desenvolvimento das funcionalidades especificadas;
- T5 – Ensaio e Testes das funcionalidades especificadas;

Os três primeiros pontos estavam previstos para o primeiro semestre, ficando os dois últimos para o segundo. No entanto, estas *milestones* foram cumpridas mais rapidamente do que o inicialmente previsto, dado que o estagiário pôde dedicar 32 horas por semana ao estágio no primeiro semestre, o dobro das 16 que estão previstas para a realização do estágio. Como tal, o desenvolvimento das funcionalidades foi bastante antecipado. É de notar também que foi feito algum desenvolvimento no primeiro semestre, ainda antes da especificação de requisitos estarem completamente definidas. Este desenvolvimento serviu para o estagiário se familiarizar com as *frameworks* de desenvolvimento, o que facilitou o trabalho do 2º semestre. No entanto, com a especificação mais detalhada dos requisitos, foi necessário reestruturar ou refazer algum do trabalho desenvolvido nesta fase. O maior detalhe na especificação dos requisitos prendeu-se sobretudo com a introdução de critérios de aceitação mais finos.

Assim, o trabalho do 2º semestre centrou-se no desenvolvimento de uma segunda versão da API, que consiste no melhoramento da implementação da versão 1, bem como na correcção de problemas encontrados durante a fase de testes. Esta segunda versão tem também novas funcionalidades, tendo sido implementados alguns módulos não terminados na versão anterior.

Posto isto, também a fase de testes foi antecipada, tendo sido introduzida uma nova *milestone*. Esta *milestone* é a T6 - Entrada da API em ambiente de produção, que diz respeito ao *deployment* de algumas funcionalidades e início da sua utilização na empresa. Esta *milestone* estava prevista para o início de Abril, tendo sido cumprida no dia 9 desse mês.

Há que referir que estas *milestones* não eram estáticas, tendo sido revisitadas ao longo dos *sprints*. Isto deve-se à utilização do Scrum que foi referida anteriormente. Assim, as tarefas definidas nas *milestones* T3, T4 e T5 entraram em praticamente todos os *sprints*.

Para além destas *milestones*, foram ainda traçadas algumas *milestones* funcionais. Estas referem-se ao lançamento de versões em que dadas funcionalidades, neste caso, correspondentes a módulos, estejam terminadas. Assim, as *milestones* funcionais foram as seguintes:

- F1 - Módulos *Auth*, *Users*, *Sprints* e *Projects*;
- F2 - Módulos *Effort*, *Metrics*, *Budgets* (posteriormente integrado em *Costs*), *Costs* e *ActivityInfo* (anteriormente *Execution*);
- F3 - Introdução de CVs, criação do *endpoint business_days*, módulos *Checkin* e *Absences*, melhoramentos nos módulos *Projects*, *Roles* e *Metrics*;
- F4 - Geração de zips, melhoramentos nos módulos *Vacations* e *Metrics*, completar os módulos *Costs* e *ActivityInfo*.

  A *milestone* F1 foi atingida no fim do *sprint* 4, como previsto. F2, prevista para o final do primeiro semestre, foi atingida parcialmente, uma vez que os módulos *Costs* e *ActivityInfo* não ficaram completos. Estes módulos foram depois redefinidos, tendo sido adiados para a *milestone* F4. No entanto, devido à falta de tempo, acabaram por não ficar concluídos, embora já possuam algumas funcionalidades aquando do final do estágio. A *milestone* F3 foi, no entanto, a mais significativa, dado que dizia respeito à *milestone* T6, ou seja, a entrada da API em ambiente de produção e, consequentemente, o início da sua utilização pelos colaboradores da empresa. Este objectivo foi atingido, como referido anteriormente, na data prevista.

3.2.2 1º Semestre

Do trabalho efectuado durante o primeiro semestre, destaca-se o desenvolvimento de módulos para a API do *back-end* necessários para dar suporte ao sistema, com a respectiva documentação e ainda a elaboração de alguns testes. Nas secções seguintes, é feita uma descrição mais detalhada de cada um destes aspectos. O *Backlog* de tarefas realizadas no primeiro semestre pode ser consultado no anexo B.

Para além disto, foi ainda despendido algum tempo na elaboração deste relatório, a maior parte do qual no estudo do estado da arte. Esta parte do relatório acabou por ser alvo de algumas restruturações, o que levou a que o tempo gasto nesta tarefa tenha sido significativamente maior do que o esperado. Para além disso, o estudo sobre o CMMI, bem como o detalhe dos requisitos e da arquitectura, foram outras tarefas que ocuparam um tempo significativo.

É importante referir que o projecto do UIS foi começado um pouco antes do início do trabalho do estagiário, pelo que alguns módulos já tinham começado a ser desenvolvidos. No entanto, quase todos tiveram que ser, em maior ou menor medida, refeitos ou adaptados para responderem aos requisitos. Assim, de seguida, são enumerados estes módulos, sendo explicada qual a contribuição do estagiário no seu desenvolvimento:

- *Auth*: a autenticação através do Lightweight Directory Access Protocol (LDAP) já se encontrava completa, pelo que não tive participação no seu desenvolvimento, tendo apenas feito ligeiras correcções à informação passada aquando do *login* de utilizadores;

- *Users*: o módulo que guarda a informação dos utilizadores já tinha a maior parte das funcionalidades necessárias. O estagiário acrescentou apenas alguns campos necessários (tipo de contrato e a data do seu início e fim para informação sobre as férias) e uma função para extrair a informação sobre os utilizadores presente no Redmine.

  Também os módulos *Projects* e *Redmine* já tinham começado a ser desenvolvidos. No entanto, possuíam apenas algumas funcionalidades de base, tendo sido o estagiário a desenvolver quase todas as funcionalidades. Relativamente aos projectos, foram feitas as alterações necessárias para que toda a informação fosse guardada do lado do UIS (estava inicialmente planeado deixar a maior parte da informação do lado do Redmine, mas verificou-se que isto seria pouco prático), incluindo os métodos necessários para fazer a sincronização da informação. Relativamente ao trabalho no módulo *Redmine*, estavam desenvolvidos apenas algumas funções para obter informação relativa às tarefas das

sprints. O estagiário desenvolveu as funções para obter informação das tarefas restantes, e ainda informação sobre as equipas, projectos e os utilizadores envolvidos nestas equipas e projectos.

- *Projects*: o módulo dos projectos já tinha começado a ser desenvolvido. No entanto, houve lugar a várias modificações feitas por mim, desde a mudança para um modelo Object-Relational Mapping (ORM), à introdução de novos campos e novas funções (como, por exemplo, a actualização da base de dados do UIS com os dados do Redmine);

- *Redmine*: o módulo de interacção com o Redmine já tinha algumas funções relativas aos projectos e métricas. Todas as funções para os módulos restantes foram desenvolvidas por mim.

  Os restantes módulos foram, na sua totalidade, desenvolvidos por mim, no âmbito do presente projecto.

Documentação

Para cada método desenvolvido para cada *endpoint*, foi elaborada uma página Wiki no Redmine com a sua documentação, para facilitar o desenvolvimento futuro por parte de outros colaboradores da empresas. Em cada página são descritos os *endpoints*, o objectivo do método e outras informações sobre como o usar (parâmetros, *request body*, etc.).

Testes

Com o intuito de começar a ambientação com as ferramentas de teste e validar já alguns dos módulos desenvolvidos, foi desenvolvido um conjunto de testes para o módulo *vacations*. Estes visam verificar se os métodos funcionam conforme o esperado. Isto inclui a verificação dos resultados devolvidos, bem como da resposta do sistema para casos em que seja introduzida informação incorrecta, etc.. De seguida, são descritos alguns exemplos dos testes desenvolvidos e realizados:

- Marcação de férias: Verifica se o pedido é feito com sucesso e se os dados são escritos na base de dados;

- Consulta de férias: Verifica se as diferentes consultas (com filtros como por utilizador, por período, etc.) devolvem os dados correctos.

- Atribuição de dias de férias: Verifica se o administrador consegue atribuir dias de férias aos colaboradores.

- Consulta de dias de férias disponíveis: Testa se os dias disponíveis são contabilizados correctamente (tendo em conta os dias disponíveis inicialmente e as férias já marcadas).

- Marcação de férias em anos diferentes: Testa a situação particular da marcação de férias que abrangem anos diferentes, ou seja, que comecem no fim de um ano e terminem no início do seguinte.

- Marcação de férias sem dias disponíveis: Verifica se o utilizador é devidamente impedido de marcar férias, caso já não tenha dias restantes.

- Marcação de férias para dias já decorridos: Verifica se o utilizador é devidamente impedido de marcar férias para dias que ja tenham decorrido.

- Marcação de férias obrigatórias: Testa se um administrador consegue marcar férias obrigatórias para um utilizador, e se este é devidamente notificado através de um email.

A metodologia de testes de integração definida e adoptada pelo estagiário foi validada pela equipa de QA da empresa e constituiu o ponto de partida para a implementação de testes dos restantes módulos.

3.2.3 2º Semestre

Como já referido, o 2º semestre serviu, sobretudo, para a continuação do desenvolvimento da API e, nomeadamente, da segunda versão mencionada anteriormente. Foram também continuadas as restantes tarefas, como a produção da documentação necessária.

Alguns dos módulos terminados ou implementados foram o *Costs*, *Rubrics* e *Moderation*.

Também houve novos testes desenvolvidos a par com o código produzido, e passaram a ser sujeitos a Integração Contínua através da ferramenta Jenkins[5], assegurando assim que a implementação de novas *releases* não comprometia desenvolvimentos anteriores (testes de regressão).

Utilizando o Jenkins, o código é automaticamente avaliado, usando os testes especificados, de cada vez que é feito um novo *push* para o Git. Os testes são realizados sobre as *branches* de desenvolvimento e de QA, referidas em 3.1.

[5]Servidor *open-source* para fazer testes sobre o código num ambiente de integração contínua, `https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins`

No caso de uma *build* do Jenkins falhar, o estagiário recebia uma mensagem por email, onde era mostrado o resultado dos testes, juntamente com as mensagens de erro, caso existissem. Isto permite garantir que qualquer erro que haja nas novas *builds* seja detectado tão cedo quanto possível, por forma a evitar que se lancem novas versões com erros, e garantindo que as versões que entram em produção são estáveis.

Para além disto, foi implementada uma nova versão do módulo de métricas, mais flexível e dinâmica do que a que tinha sido implementada durante o primeiro semestre. Isto tornou possível a ideia da criação da *framework* de métricas referida no capítulo 6. Há que salientar a importância dada ao desenvolvimento deste módulo, uma vez que é o módulo central do UIS, tendo sido as outras funcionalidades horizontais implementadas principalmente para lhe dar suporte.

Foram ainda implementadas novas funcionalidades, de entre as quais se destacam os avisos assíncronos por email, a gravação e consulta de pdfs, a geração de ficheiros zip e xlsx e o *upload* de imagens para os "perfis" dos utilizadores.

Ainda durante o segundo semestre, houve lugar a duas *code reviews* utilizando o Gerrit[6], o que permitiu ao estagiário fazer algumas melhorias ao seu trabalho, tanto a nível da estrutura do código, como da própria arquitectura do sistema.

Finalmente, começaram a ser desenvolvidas algumas das funcionalidades no *front-end*, tendo também começado a ser usadas pelos restantes colaboradores da empresa.

[6]Ferramenta de revisão de código *web-based* que permite a revisão de código submetido no Git por parte de outros colaboradores, `https://code.google.com/p/gerrit/`

Capítulo 4

Especificação técnica

Neste capítulo, são apresentados os requisitos e a arquitectura do sistema. Na secção 4.1, são detalhados os requisitos definidos para o sistema. Os requisitos visam traduzir os objectivos do sistema em pequenas frases que detalhem as suas funcionalidades. De seguida, em 4.2, é apresentada a arquitectura do sistema, definida de forma a dar resposta aos requisitos definidos na secção anterior. Na secção 4.3, são detalhados os módulos desenvolvidos e o seu funcionamento. Depois, em 4.4 são apresentadas as estruturas de dados que dão suporte a estes módulos. Finalmente, em 4.5, são explicadas as razões que levaram à escolha das ferramentas usadas na construção da plataforma.

4.1 Requisitos

Os requisitos funcionais do UIS foram especificados sob a forma de *user stories* – uma maneira não-formal de expressar os requisitos do sistema, em que é usada uma linguagem facilmente compreensível por todos. Assim, com algumas frases, consegue detalhar-se qual deverá ser o comportamento do sistema e a interacção entre este e o utilizador. As *user stories* aqui apresentadas seguem um modelo que pode ser descrito como:

> "*Como <função do utilizador>, quero <funcionalidade>, para <objectivo>*"

As funções dos utilizadores do sistema são os seguintes:

- *Manager*: utilizador que pertence à gestão de topo da empresa;
- PM: o gestor de projecto;

- Team Manager (TM): o gestor de equipa, sendo uma equipa o equivalente a um departamento da empresa (por exemplo, *development*, *design*, QC, etc.);
- *User*: um utilizador sem privilégios especiais.

As *user stories* foram divididas consoante os diferentes módulos a que dizem respeito, sendo apresentadas nas tabelas 4.1 a 4.10, seguindo o formato descrito anteriormente. Cada *user story* tem ainda um identificador único. Para além disto, a cada *user story* corresponde uma prioridade, definida numa escala de 1 (extremamente importante) a 5 (acessório), e as respectivas dependências. As prioridades foram definidas pelo PM, de acordo com a maior ou menor urgência por parte da empresa em ter estas funcionalidades disponíveis. É de notar que não existem muitas dependências entre os requisitos, uma vez que estes foram desenvolvidos de forma a serem o mais modulares possível, como será explicado em maior detalhe mais à frente. No entanto, visto que a maior parte dos módulos utilizam informação sobre utilizadores e/ou projectos, quase todos eles estavam dependentes destes. Isto também traduz a maior prioridade atribuída às *user stories* referentes a estes módulos.

A tabela 4.11 refere-se a *user stories* que foram definidas pelo estagiário, com base no estudo feito na secção do Estado da Arte, estando ainda sujeitas a aprovação. Finalmente, na tabela 4.12 são apresentados os requisitos não-funcionais do sistema.

ID	Como	quero	para	Prioridade
A1	PM	especificar o esforço necessário (horas) por um utilizador ou equipa	simplificar o processo de alocação de RH mensal	3
A2	Manager	alocar equipa/RH a um projecto num dado intervalo	distribuir os RH da Ubiwhere pelos projectos	2

Tabela 4.1: *User stories* para o módulo Allocations

ID	Como	quero	para	Prioridade
E1	Manager	ver o esforço despendido usando diferentes critérios	saber o custo real do trabalho	2
E2		ver estatísticas e métricas não financeiras usando diferentes critérios	saber qual a produtividade dos RH	1

Tabela 4.2: *User stories* para o módulo Effort

ID	Como	quero	para	Prioridade
M1	PM	ver as métricas de um dado *sprint*	perceber a *performance* da minha equipa	1
M2	Manager	ver o *dashboard* de métricas CMMI usando filtros específicos	saber o estado de cada projecto	2
M3	User	ver métricas de alto nível de um projecto	saber mais sobre o portfólio da Ubiwhere	1
M4		ver as métricas de um utilizador	saber se ele está a trabalhar eficientemente	2

Tabela 4.3: *User stories* para o módulo Metrics Dashboard

ID	Como	quero	para	Prioridade
AU1	Manager	dar acesso a informação às pessoas responsáveis por ela	evitar que haja acessos indevidos à informação	1
AU2	User	fazer *login* no UIS usando as credenciais da Ubwihere	aceder à plataforma	1

Tabela 4.4: *User stories* para o módulo de Authentication e Authorization

ID	Como	quero	para	Prioridade
P1	PM	inserir ou editar a informação sobre os meus projectos	ter a informação sobre os projectos actualizada	1
P2	User	ver a informação sobre um projecto	saber mais sobre o portfólio da Ubwihere	1
P3		listar os projectos usando critérios específicos	descobrir projectos que obedecem a dadas condições	1

Tabela 4.5: *User stories* para o módulo Projects

ID	Como	quero	para	Prioridade
S1	Manager	exportar para PDF/XLSX as alocações de equipas ou RH	guardar provas do planeamento de RH	4
S2		exportar um *snapshot* para PDF/XLSX	aceder a informação sem ter que aceder ao UIS	5
S3		carregar um *snapshot* para o UIS	aceder a informação guardada previamente	5
S4		associar comentários a um *snapshot*	ter informação sobre o *snapshot*	5

Tabela 4.6: *User stories* para o módulo Snapshots

ID	Como	quero	para	Prioridade
SP1	User	ver as *sprints* de um projecto e a sua informação	saber mais detalhes sobre um dado projecto	2
SP2	PM	adicionar, editar e remover um *sprint* de um projecto	gerir os *sprints* de um projecto	2

Tabela 4.7: *User stories* para o módulo Sprints

ID	Como	quero	para	Prioridade
T1	User	ver e aceder à informação básica da equipa	saber mais sobre as equipas	3
T2		ver em que equipas está um utilizador a trabalhar		3
T3	TM	adicionar e remover RHs à minha equipa	construir a minha equipa	3
T4		saber quanto do esforço da equipa é necessário no projecto	ajudar-me a alocar os recursos	3

Tabela 4.8: *User stories* para o módulo Teams

ID	Como	quero	para	Prioridade
U1	Manager	editar informação básica, contratual e pessoal de um utilizador	guardar informação detalhada sobre um utilizador	1
U2	User	consultar informação dos outros utilizadores	conhecer melhor os meus colegas	3

Tabela 4.9: *User stories* para o módulo Users

ID	Como	quero	para	Prioridade
V1	Manager	aprovar ou rejeitar pedidos de férias	informar alguém se pode ir de férias	1
V2		atribuir um número de dias disponíveis por ano a um dado RH	gerir o tempo de férias de cada pessoa	2
V3	User	alocar equipa/RH a um projecto num dado intervalo	distribuir os RH da UW pelos projectos	2
V4		marcar férias se tiver dias disponíveis	informar os gestores que quero ir de férias	2
V5		ver se tenho dias de férias disponíveis num dado ano	saber se posso marcar férias	2
V6		ver um calendário com as férias	saber quando as outras pessoas vão estar de férias	1
V7		editar ou remover um pedido de marcação de férias	evitar que haja pedidos errados	1

Tabela 4.10: *User stories* para o módulo Vacations

ID	Como	quero	para
S1	Manager	aplicar *workflows* pré-definidos	facilitar a implementação de processos
S2		aplicar *templates* de projectos pré-definidos	facilitar o processo de criação de projectos
S3		saber qual o tipo de férias que um colaborador tem	facilitar a gestão de RH
S4		previsões (algoritmos *time-series*) para previsões financeiras	prever a performance financeira da empresa
S5		geração periódica automática de relatórios	ter informação disponível, sem interagir com o UIS
S6		partilhar relatórios ou *dashboards* com outros colaboradores	disponibilizar informação pertinente a quem interessar
S7		associar notas/comentários a relatórios e *dashboards*	permitir dar e ver opiniões sobre as métricas usadas
S8		poder personalizar as páginas de *dashboards*	poder ver a informação que é relevante para mim
S9	PM	ter informação sobre as *skills* dos colaboradores	fazer a alocação de RH mais eficazmente
S10		calendário com as tarefas atribuídas a cada colaborador	saber a carga de trabalho e disponibilidade dos RH
S11	User	integração com plataformas de IM (como o Hipchat)	facilitar a colaboração com os colegas de equipa
S12		introdução de conceitos de *gamification* (*leaderboards...*)	estimular o aumento da produtividade

Tabela 4.11: *User stories* Sugeridas pelo aluno

ID	Requisito
N1	A API deve fornecer os dados em formatos interoperáveis (pelo menos JSON e XML)
N2	Os *endpoints* da devem funcionar sobre HyperText Transfer Protocol (Secure) (HTTPS)
N3	A integração com sistemas externos deve ser feita através de uma camada de abstracção para facilitar a eventual mudança de ferramentas externas, evitando mudanças na camada superior do código
N4	As chaves de acesso à API devem ser revogadas semanalmente
N5	A sincronização de dados com sistemas externos deve ser feita *on-demand*
N6	A plataforma deve ser desenvolvida em Python, dada a maior especialização da empresa nessa linguagem
N7	O Sistema de Gestão da Base de Dados (SGBD) pode ser um dos seguintes: PostgreSQL, PostGIS, MongoDB ou Cassandra
N8	As bibliotecas externas devem ser *open-source*
N9	A gestão de bibliotecas deve ser feita usando o (Footnote não está a aparecer)! *pip*[1]
N10	Cada *request* à API deve ter um teste de integração correspondente

Tabela 4.12: Requisitos Não-Funcionais

4.2 Arquitectura

A arquitectura do sistema foi concebida de forma a ser o mais modular e *loosely coupled* possível. Neste contexto, isto significa que se tenta alcançar uma maior flexibilidade na integração de diferentes sistemas, ou seja, a arquitectura deve ser suficientemente flexível para suportar a adição ou substituição de diferentes componentes [16]. Isto deve-se ao facto de um sistema deste tipo ter que ser adaptável às necessidades da empresa, devendo tornar possível a adição de novas funcionalidades ou de fontes de informação diferentes ao longo do tempo. Para além disto, o sistema tem também que ser integrado com algumas ferramentas já em uso.

Fontes de dados

Em primeiro lugar, há que separar as fontes de dados. Neste momento, a informação sobre projectos está guardada na plataforma Redmine, estando a informação sobre os Recursos Humanos guardada na Intranet. No que diz respeito à gestão de projectos, foi decidido pela empresa, numa fase inicial, continuar a usar o Redmine. Assim, o UIS permitirá ver informação sobre os projectos, bem como as respectivas métricas, mas a gestão de *sprints* e tarefas continuará a ser feita no Redmine. Dado que, para o cálculo correcto das métricas, é essencial que os dados estejam actualizados, os acessos aos dados do Redmine são feitos em *runtime* para o módulo *Metrics*. Para a visualização de outros dados de projecto menos importantes (a sua descrição e características), em que não é essencial garantir que os dados estão de acordo com o Redmine, é feita uma sincronização periódica ou manual. Esta sincronização é feita quando desejado, através de chamadas à API, ou utilizando comandos criados em Python, que podem ser corridos periodicamente, através de *cronjobs*[2].

Para extrair a informação presente no Redmine, foi criada uma camada que permite obter todos os dados sobre projectos anteriores (como horas gastas, *sprints*, *bugs*, etc.), que permite fazer o cálculo das métricas alimentadas por esses dados (*bugs* por *sprint*, média de *bugs* por projecto, etc.). Os dados são obtidos através da API do Redmine.

No que diz respeito aos dados da Intranet, e visto que as funcionalidades desta plataforma serão substituídas pelo UIS, estes serão extraídos apenas uma vez para o ambiente de produção. Para este efeito, foram desenvolvidos os comandos necessários para proceder a esta extracção, transformação e

[2]Mecanismo do Linux que permite correr aplicações ou comandos em períodos especificados, `https://service.futurequest.net/index.php?/Knowledgebase/Article/View/23`

carregamento, como será explicado mais à frente. Estes dados são relativos a presenças, férias e faltas de colaboradores.

Outra fonte de dados usada, como referido, são ficheiros Excel. Estes contêm informação sobre as presenças do pessoal que pertence aos quadros da Ubiwhere mas trabalha deslocado nas instalações da PT Inovação. O sistema de marcação de tempos desta empresa gera um ficheiro Excel que pode ser importado para o UIS, como será explicado posteriormente.

Finalmente, existe também uma integração com o LDAP da empresa, para fazer a autenticação dos utilizadores.

Os módulos para integrar outros sistemas ainda não foram desenvolvidos e farão parte do trabalho futuro. Alguns dos módulos que já estão previstos incluem a integração de dados de sistemas como o Jenkins, o Sentry[3], Google Drive e outros sistemas de gestão documental.

API

Para garantir as características acima relativas ao *loose coupling* da plataforma, a comunicação com os clientes é feita através de uma API RESTful. Esta API é disponibilizada via HTTP(S), utilizando os formatos JavaScript Oriented Notation (JSON), Extensible Markup Language (XML) e Yet Another Markup Language (YAML), que são incluídos por defeito na *framework* Tastypie, como explicado em 4.5. Isto faz com que esta camada seja totalmente agnóstica em relação às tecnologias dos clientes *front-end*, o que é importante para um sistema que deverá ter clientes *web* e, futuramente, *mobile*, sem ficar dependente destes.

Quanto à estrutura do código, é necessário começar por referir que foi utilizada a *framework* Django. Esta é uma *framework* de desenvolvimento *web* escrita em Python. O Django é considerado como sendo uma *framework* Model-View-Controller (MVC) [17], embora tenha algumas diferenças em relação a este modelo. No modelo MVC, a parte *Model* refere-se à camada de acesso a dados, *View* à parte do sistema responsável pela escolha e apresentação da informação, e o *Controller* à parte responsável por decidir que *view* escolher, dependendo da situação, e aceder ao *Model* para a recolher. Os responsáveis pelo desenvolvimento do Django consideram a sua ferramenta como sendo *Model-Template-View*:

- *Model* trata de tudo o que é relativo aos dados: como acedê-los, validá-los, e quais as relações entre eles;
- *Template* trata da apresentação dos dados. No caso do UIS, visto tratar-se de uma API, a "apresentação" de dados é a própria interface

[3] `https://getsentry.com/welcome/`

com a API através de JSON/XML/YAML, ficando a parte de visualização propriamente dita a cargo dos clientes que a consomem;

- *View* é a camada de lógica de negócio, contendo a lógica que acede ao modelo e encaminha para o *Template* desejado, e funcionando como "ponte" entre as outras duas partes. Neste caso, esta camada é desenvolvida usando o Tastypie visto que tem que funcionar como API RESTful. O Tastypie permite então construir esta API, dando a possibilidade de fazer o *override* das funções quando necessário. Este comportamento é explicado em detalhe mais à frente.

Assim, os módulos em que o sistema se divide possuem, cada um, estas três partes.

A plataforma a criar tinha, portanto, que ter uma arquitectura com estas características. Coube então ao estagiário especificar, documentar e implementar esta arquitectura. Na figura 4.1, é apresentado um esquema da arquitectura do sistema. Os módulos referidos na figura serão explicados na secção seguinte.

4.3 Módulos

Nesta secção, são descritos os módulos da API do UIS, os respectivos *endpoints* e as suas funções. Depois, numa perspectiva de alto nível, é explicado o funcionamento geral dos mesmos.

Dado que o principal foco do sistema recaiu na recolha de métricas, este módulo, bem como todos aqueles sem os quais este sistema e este módulo não poderiam funcionar, são considerados como os módulos *core*:

- *Metrics*: Módulo responsável por todos os cálculos das várias métricas de *sprint*, de projecto, de portfólio e organizacionais da empresa. São as funcionalidades implementadas neste módulo que possibilitam a gestão dinâmica da organização, como mostrado mais à frente no capítulo 6.

- *Auth*: Utiliza a biblioteca *python-ldap* para obter o *single sign-on* (mecanismo que permite a um utilizador autenticar-se uma única vez, ganhando acesso às várias plataformas disponíveis[4]) do utilizador no LDAP da empresa e fazer a autenticação no UIS.

- *Authorization*: Restringe o acesso do utilizador a determinados recursos, consoante a sua posição na empresa.

[4] `http://searchsecurity.techtarget.com/definition/single-sign-on`

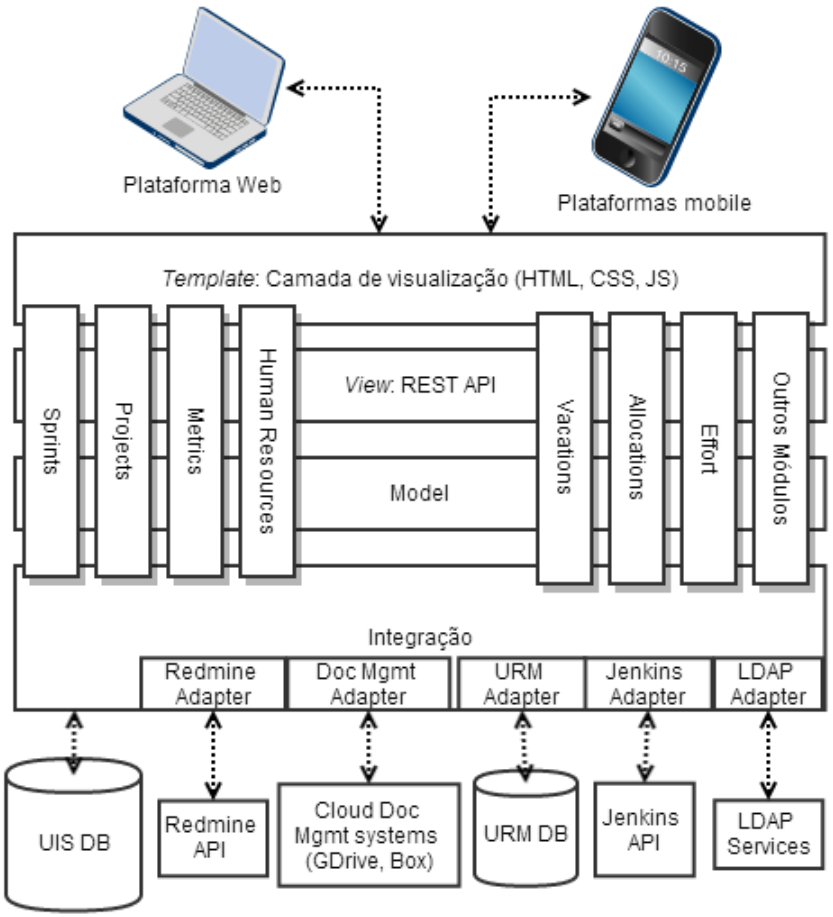

Figura 4.1: Arquitectura do Ubiwhere Information System

- *Redmine*: Interface com a API do Redmine, que permite obter informação guardada neste.

Os restantes módulos construídos servem para dar apoio aos módulos descritos acima. Embora o funcionamento do sistema fosse possível sem eles, a sua implementação ajuda a estruturar os dados obtidos de outras fontes, tornando mais fácil o processamento de dados no módulo das métricas. Para além disto, também acrescentam outras funcionalidades ao sistema, nomeadamente a nível da gestão de Recursos Humanos. Estes módulos são:

- *Allocations*: Trata das alocações de equipas ou utilizadores a equipas ou projectos.

- *Activities*: Guarda informação relativa a cada uma das actividades do projecto. Estas actividades são descritas no anexo C.
- *Checkin*: Regista as horas de entrada e saída dos colaboradores.
- *Costs*: Informação sobre despesas da empresa em cada projecto.
- *Effort*: Lida com o registo e consulta do tempo efectivo despendido em cada tarefa por um dado utilizador.
- *Roles*: Guarda informação sobre o papel de cada utilizador na organização e em cada projecto.
- *Moderation*: Faz a moderação de acções sobre pedidos já aprovados ou rejeitados pela administração.
- *Projects*: Obtém e guarda informação relativa a projectos.
- *Rubrics*: Informação sobre as rúbricas de um projecto.
- *Sprints*: Informação sobre os *sprints* de um projecto.
- *Teams*: Obtém a informação sobre as equipas da Ubiwhere.
- *Users*: Módulo que acrescenta mais informação sobre os utilizadores ao módulo *User* do Django.
- *Vacations*: Módulo para a gestão das férias dos colaboradores.
- *Work Locations*: Informação sobre os locais de trabalho de cada colaborador, visto que a empresa tem mais do que um escritório e tem alguns colaboradores a trabalhar deslocados em outras empresas.

As funcionalidades de cada módulo são disponibilizadas através de *endpoints*, para os quais são encaminhados os pedidos REST. Nas figuras 4.2 e 4.3, podemos ver uma representação dos diferentes *endpoints* que correspondem a cada módulo, bem como as acções que cada um permite realizar. Cada acção representada corresponde a um verbo REST: "obter" corresponde ao verbo GET, "criar" ao verbo POST, "actualizar" ou "editar" ao verbo PATCH e "apagar" ao verbo DELETE.

Apenas os módulos *Authorization* e *Redmine* não possuem *endpoints*, dado que consistem apenas em grupos de funções que são usadas internamente para os mecanismos de autorização e de obtenção de informação do Redmine.

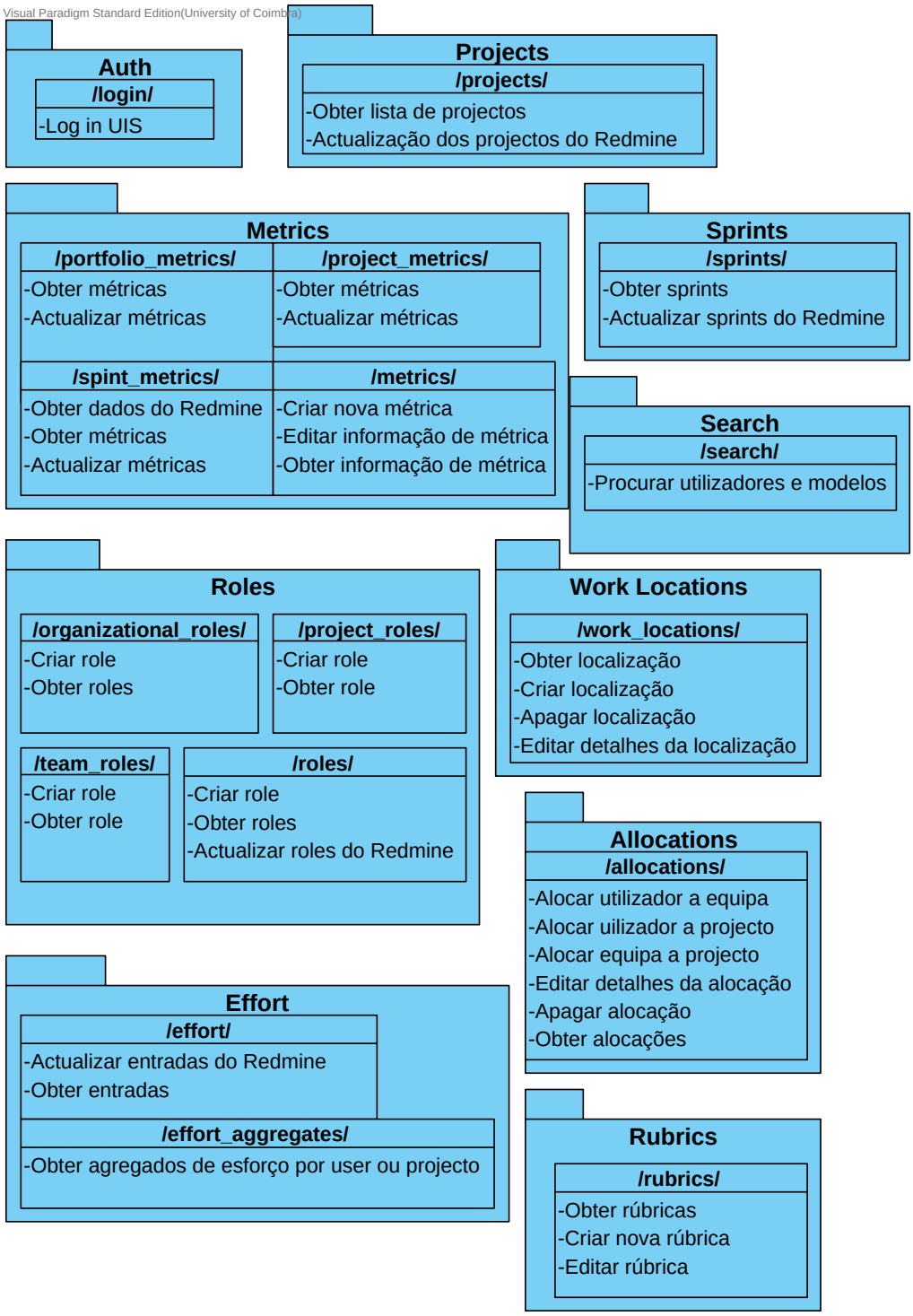

Figura 4.2: Diagrama de *packages*

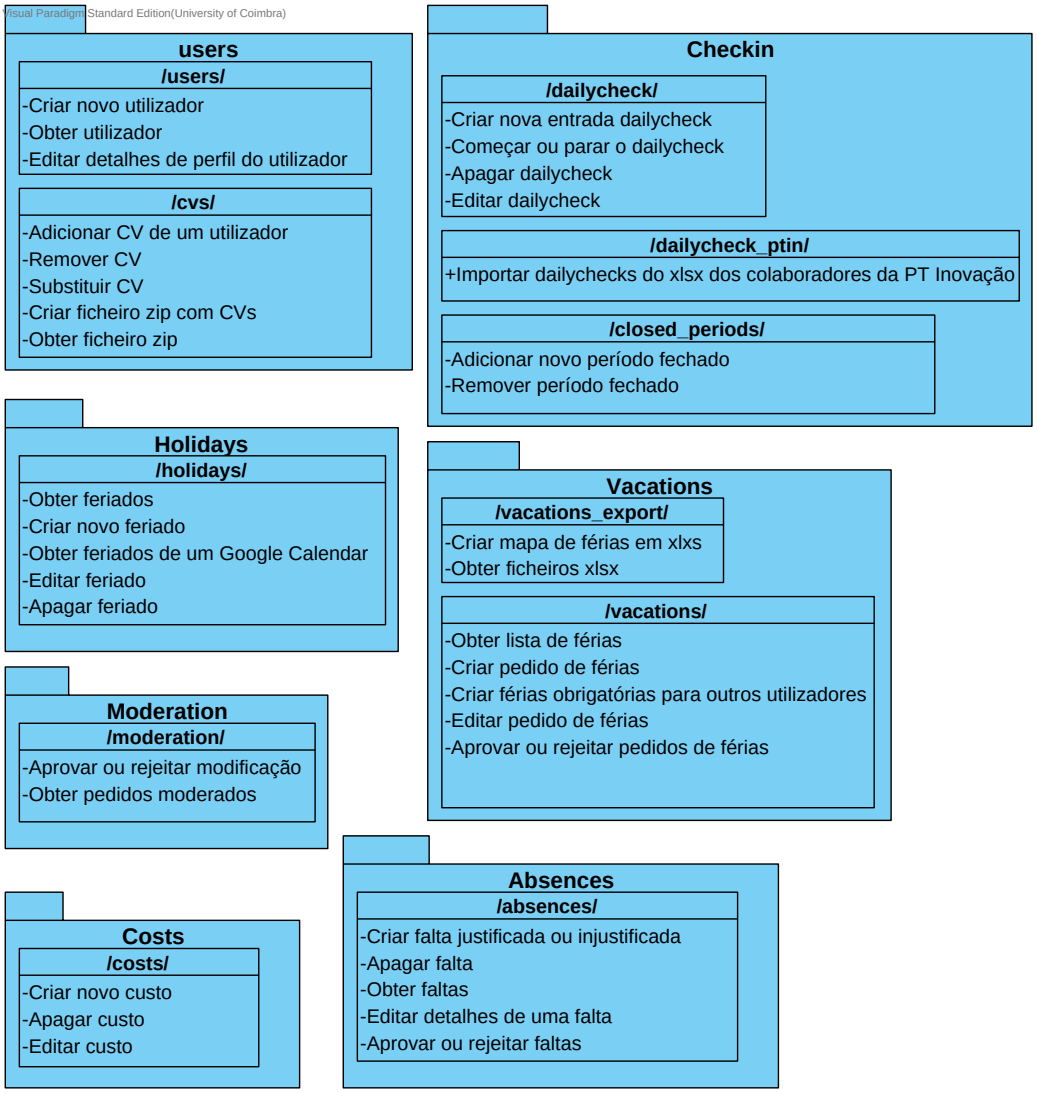

Figura 4.3: Diagrama de *packages* (cont.)

Em termos de funcionamento, e de uma perspectiva de mais alto nível, de forma geral, é feito um pedido HTTP a um dos *endpoints* referidos atrás. Estes pedidos são recebidos pela API, que se encarrega de processá-lo e devolver a informação desejada e/ou fazer as alterações correspondentes aos dados.

Para explicar este funcionamento geral, podemos recorrer a um exemplo, neste caso, do módulo *Checkin*. Na figura 4.4, temos um exemplo da interface *web* criada para este módulo. Aqui, podemos ver que o utilizador tem um calendário no qual estão assinalados os seus tempos de entrada e saída da

empresa, bem como o tempo gasto em cada projecto em cada dia. Na figura 4.5, podemos ver o diagrama de sequência em que são apresentadas as acções realizadas para introduzir informação relativa às horas de entrada e saída de um determinado dia. Em *Generate personal calendar*, são apresentadas, de forma mais resumida, as acções realizadas para apresentar a informação disponível no ecrã: informação sobre faltas, feriados e *dailychecks*. De seguida, o utilizador pode introduzir nos campos respectivos a informação relativa ao dia e, ao pressionar *submit*, vai efectuar um pedido POST à API. A API é encarregue de verificar se a informação para esse utilizador nesse dia já existe na base de dados e, conforme o resultado desta verificação, gravar ou não a informação introduzida e gerar uma resposta condizente.

De forma geral, é esta a maneira como os pedidos funcionam: o *frontend* faz o pedido à API, que fica encarregue de processar o pedido, realizar as operações necessárias (seja retornar, acrescentar, alterar ou apagar informação) e retornar os resultados para que o *frontend* os apresente ao utilizador.

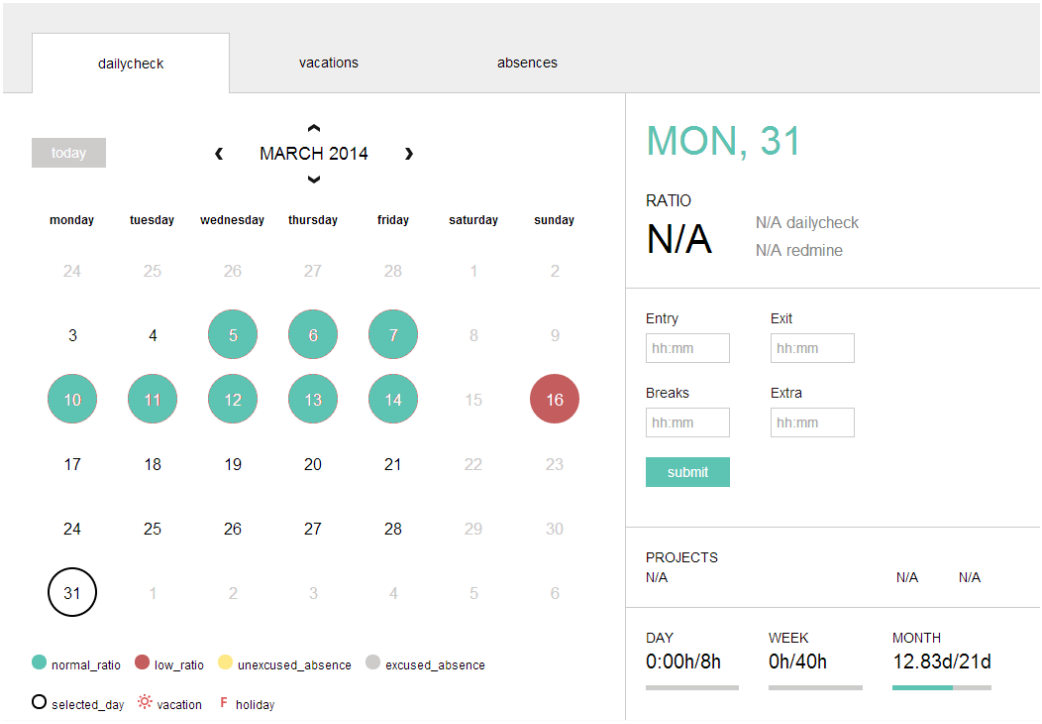

Figura 4.4: Interface para a gestão de presenças e horas de trabalho

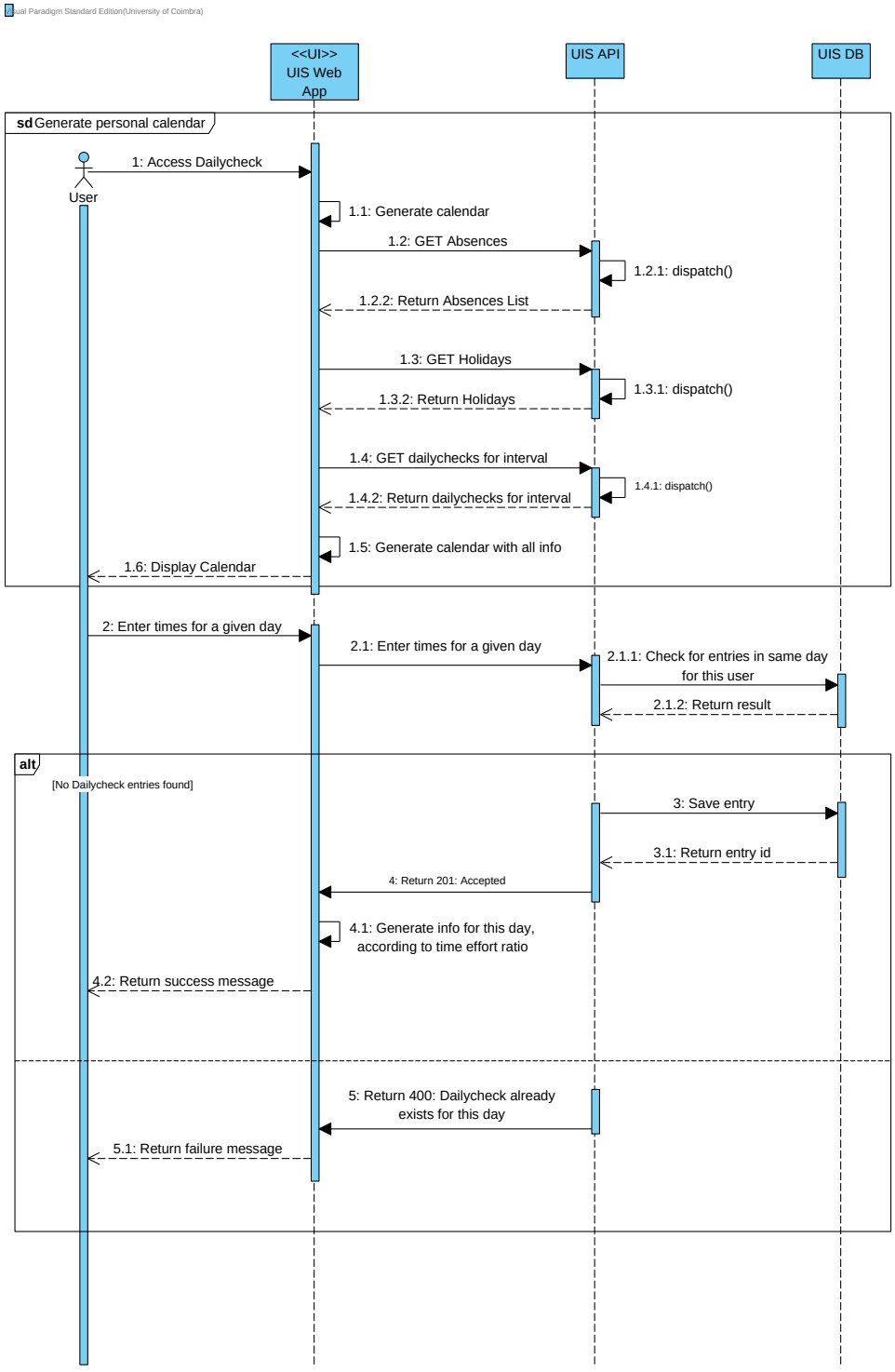

Figura 4.5: Diagrama de sequência para a acção de entrada de tempos no *UIS*

4.4 Estruturas de dados

É importante analisar mais detalhadamente a estrutura de dados utilizada, por forma a conseguimos ter um maior entendimento do funcionamento dos módulos e das ligações entre eles. Como tal, são apresentados os diagramas de Entidade-Relação (ER) do sistema de seguida. Para evitar alguma repetição, quando são referidas entidades já descritas anteriormente, não são apresentados todos os seus atributos.

O Django possui um sistema de ORM. Isto quer dizer que é feito um mapeamento entre os objectos Python e a base de dados utilizada. Dentro de cada módulo, existem um ou mais modelos, classes criadas que são subclasses de *Model* do Django. Na maioria das vezes, cada um destes modelos vai corresponder a uma tabela da base de dados, sendo cada um dos atributos do modelo um campo dessa tabela. Estes atributos têm um tipo específico, que determina o tipo que vai ter o campo respectivo na tabela. Para além disso, também é possível definir as relações com os outros modelos.

Na figura 4.6, podemos ver os diagramas de Entidade-Relação (ER) dos módulos *user*, *auth* e *tastypie*. Aqui, é importante fazer a distinção entre a entidade *user* e a entidade *userprofile*. A primeira reflecte a classe *user* que o Django contém por defeito, enquanto que a segunda foi implementada pelo estagiário. Isto deveu-se à necessidade de ter mais informação sobre os utilizadores no UIS. Assim, a classe *userprofile* foi desenvolvida para juntar esta informação extra ao modelo que o Django já oferece, e que já trata de algumas questões como autenticação e acesso à página de *admin*. O *userprofile* contém então uma chave estrangeira para o modelo *user*, servindo também esta chave como chave primária. Os modelos do *tastypie* tratam do acesso à API, sendo que o modelo ApiKey contém as chaves de acesso de cada utilizador, e o modelo ApiAccess é populado quando é definida uma política de *throttling* para restringir o número de acessos de cada utilizador à API.

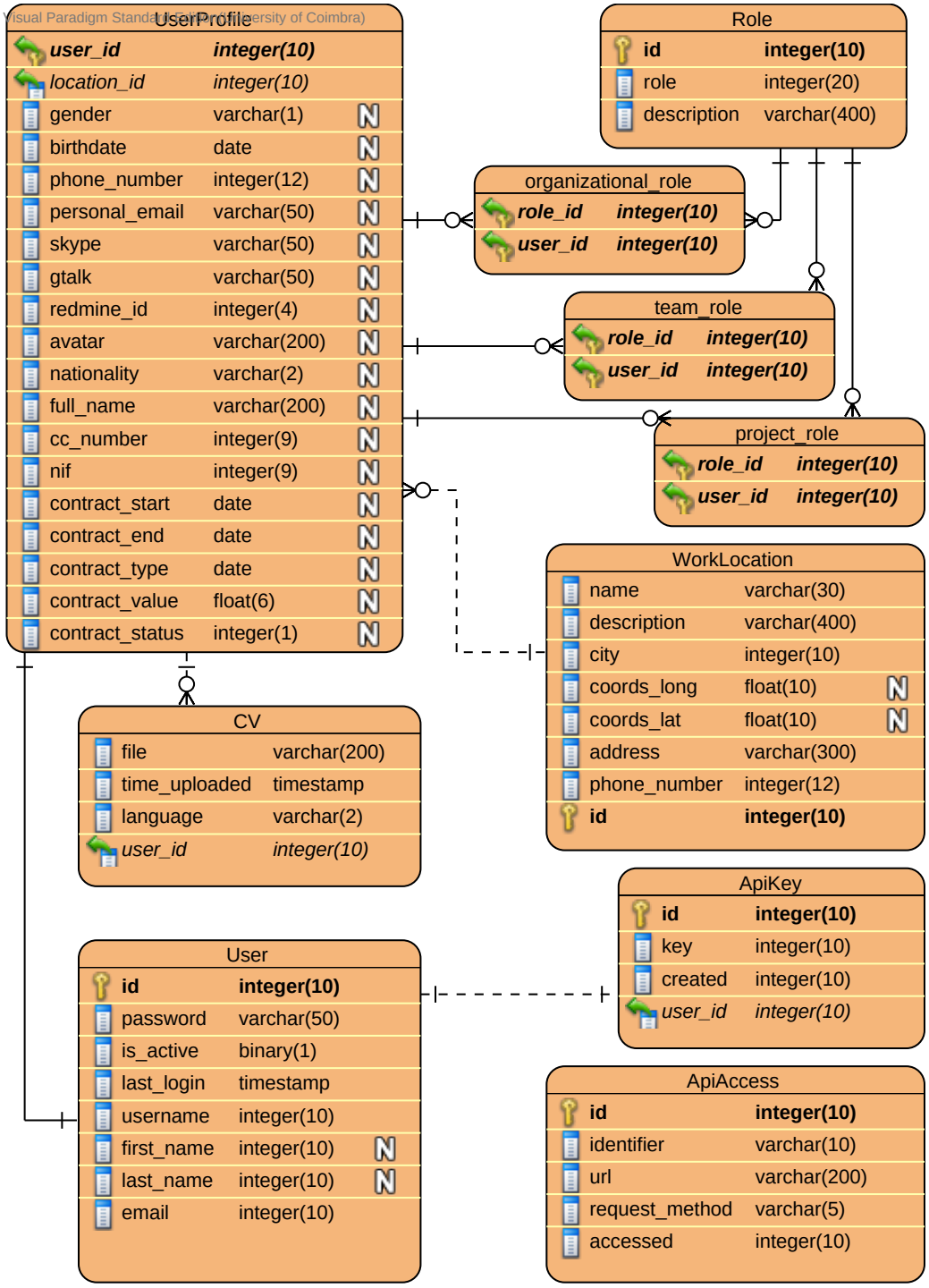

Figura 4.6: Diagrama ER: módulos *users*, *auth* e *tastypie*

Na figura 4.7, estão representados os modelos para os módulos *project*, *sprints* e *metrics*. O modelo dos projectos guarda toda a informação relevante sobre estes. Os campos deste modelo replicam a informação que o Redmine providencia, sendo ainda acrescentados os campos *PM_cost* e *dev_cost*, que dizem respeito aos encargos com RH para cada projecto, e servem para calcular algumas métricas financeiras.

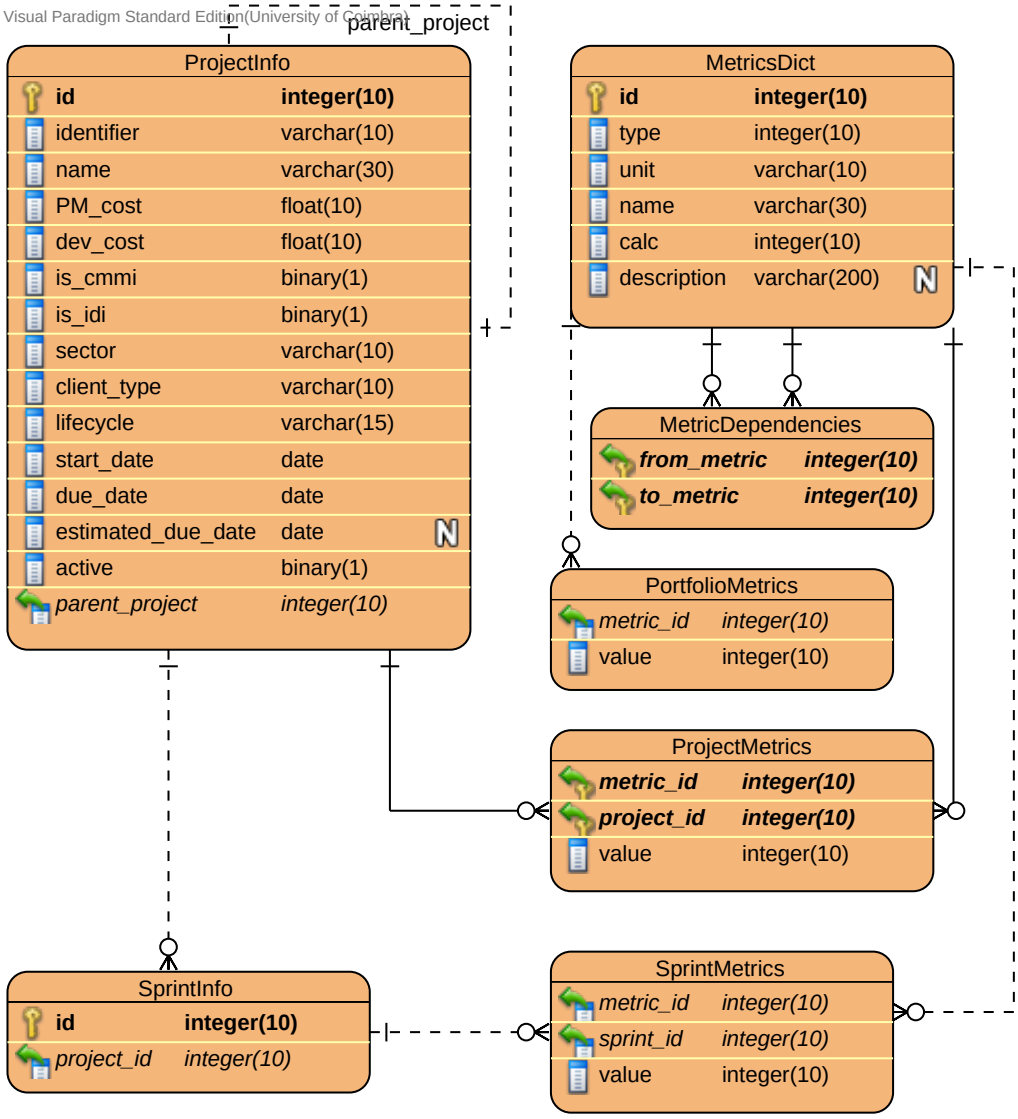

Figura 4.7: Diagrama ER: módulos *projects*, *sprints* e *metrics*

Em relação às métricas, existem três tabelas, cada uma relativa a um dos níveis das métricas identificados na figura, como será explicado no capítulo 6: *sprint*, projecto e portfólio. Cada métrica é representada por um objecto *metricsdict*, que contém informação relativa à métrica: nome, tipo, unidade e descrição. Depois, cada um dos tipos de métricas descritos anteriormente herda o modelo *metrics*, que consiste num objecto *metricsdict* e o valor numérico da métrica. Estes tipos de métricas diferentes têm ainda outro campo, uma chave estrangeira para o *sprint* ou projecto, nos dois primeiros casos, ou a descrição dos filtros usados no cálculo da métrica de portfólio. Resta referir a relação *dependencies*: como é mostrado na figura E.1, as métricas de um "nível" são alimentadas pelas do nível anterior, podendo ter uma ou mais dependências. A relação *dependencies* traduz isto, sendo uma relação *many-to-many*.

De seguida, temos, representado na figura 4.8, a estrutura de dados dos módulos *allocations* e *effort*. Estes módulos dizem respeito à alocação de Recursos Humanos para projectos ou equipas, e o esforço efectivo destes. Na tabela *effort*, o valor *hours* representa o número de horas gasto numa dada *activity* do Redmine. O campo *last_update* serve para registar quando foi a última vez que um campo foi alterado no UIS, para que, ao verificar os dados provenientes do Redmine, possamos determinar se a entrada precisa ou não de ser actualizada. Existe ainda uma chave estrangeira, *dailycheck_entry*, que associa cada entrada a uma da tabela *dailycheck* (ou seja, as entradas na tabela *effort* de uma dada data, para um dado utilizador, terão todas ligação à entrada da tabela *dailycheck* para o mesmo utilizador e data). Para as alocações, existe uma tabela em que é registado, para cada utilizador, qual a equipa ou projecto a que vai ser alocado, em que período de tempo, e com que percentagem desse período. A alocação pode ser também de uma equipa a um projecto. Estas possibilidades justificam a existência das chaves estrangeiras *from_user* e *to_user*, e from_team, *to_team* e *to_project*. O facto de ter estas chaves estrangeiras implica que algumas delas fiquem guardadas com valores *null*. No entanto, foi preferido este comportamento, mais simples, ao invés de uma maior normalização das tabelas, que traria uma maior complexidade ao modelo.

Na figura 4.9, podemos ver o diagrama correspondente aos módulos *vacations*, *absences* e *checkin*. Estes são os três módulos relacionados com a gestão de presenças do pessoal da Ubiwhere. Em relação ao módulo do *checkin*, a tabela *DailyCheck* mantém toda a informação relativa às horas de trabalho dos funcionários. Cada entrada nesta tabela contém a informação relativa a um dia de trabalho de um funcionário. A tabela *ClosedPeriod* guarda informação relativa a períodos de tempo que a Administração queira "fechar", ou seja, para os quais não quer permitir que seja introduzida nova

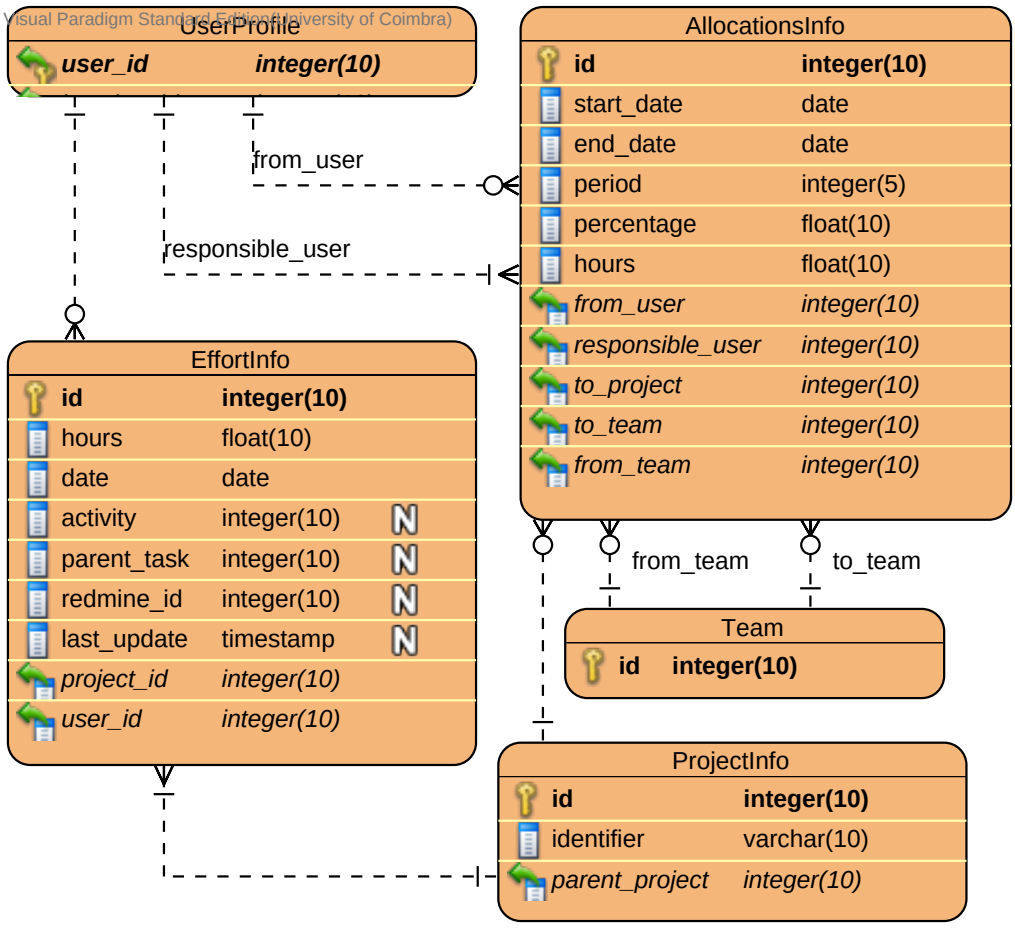

Figura 4.8: Diagrama ER: módulos *allocations* e *effort*

informação, para efeitos de contabilidade. No módulo *Vacations*, a tabela *VacationDays* contém entradas que registam o número total de dias de férias disponíveis para um dado utilizador, num dado ano. A tabela *VacationsInfo* guarda a informação sobre os períodos de férias de cada utilizador (data de início e fim, dias de trabalho totais ocupados pelas férias e o estado do pedido, em relação à aprovação ou não por parte da administração). Finalmente, para o módulo *absences* é guardada a data de uma falta para um trabalhador, podendo ter uma justificação ou não, e um estado, semelhante ao existente no módulo *Vacations*.

Finalmente, na figura 4.10, temos o diagrama correspondente aos módulos *costs*, *rubrics* e *activities*. Estes módulos servem para fazer a gestão dos orçamentos e gastos dos projectos para as diferentes rúbricas. Estas rúbricas

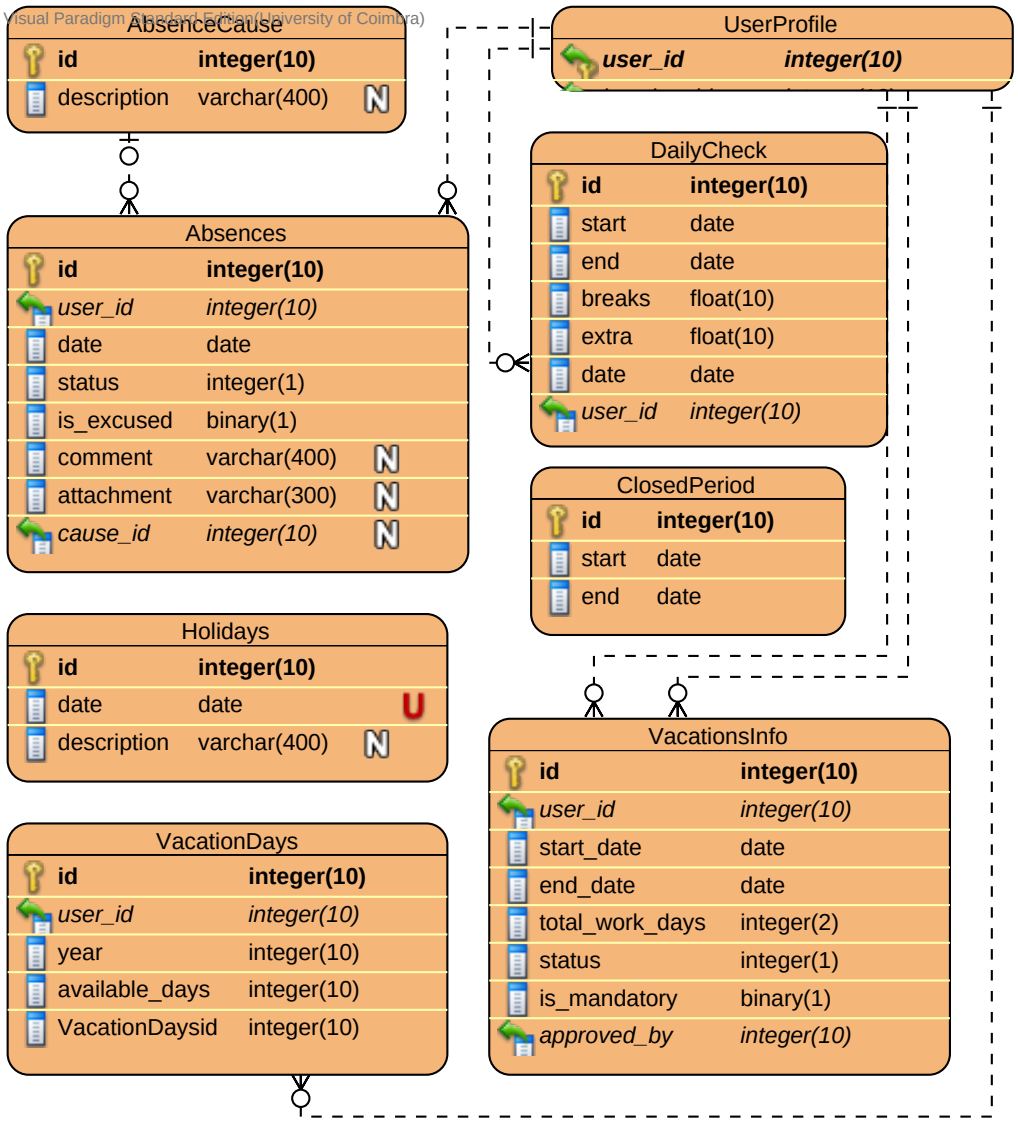

Figura 4.9: Diagrama ER: módulos *dailycheck*, *vacations* e *absences*

consistem em facetas do projecto com as quais a empresa tem encargos, tais como Recursos Humanos, *hardware* e outros gastos. Como podemos ver, um projecto tem uma relação *many-to-many* com uma rúbrica. Assim, a tabela *ProjectRubric* vai conter informação de cada rúbrica para cada projecto, tendo também especificado o valor orçamentado no campo *budgeted*.

Depois, as entradas da tabela *Cost* vão ter, cada uma, um custo imputado a cada uma destas rúbricas para cada projecto, incluindo o seu valor e descrição. Por outro lado, a tabela *Activity* diz respeito ao módulo *ActivityInfo* e, consequentemente, às actividades de projecto descritas em C. Assim, para cada tarefa, vamos ter o seu peso no projecto em percentagem, a sua taxa de concretização. Estes valores ajudam a determinar qual o estado do projecto em termos do trabalho já realizado. Os custos com Recursos Humanos durante o projecto são calculados utilizando os valores do esforço, e são comparados com a taxa de concretização do projecto para obter uma estimativa de quanto será gasto ao longo do projecto.

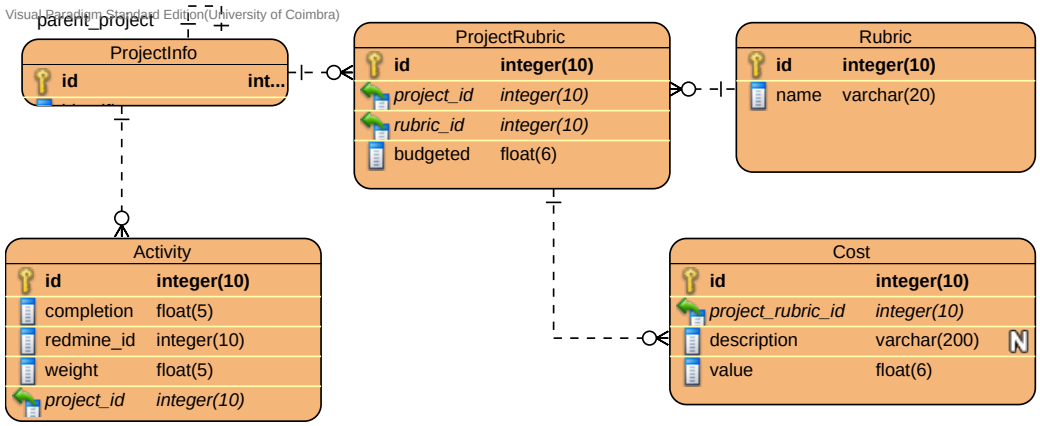

Figura 4.10: Diagrama ER: módulos *costs*, *rubrics* e *activities*

4.5 Escolha de ferramentas

A nível de tecnologias, a plataforma está a ser desenvolvida usando Django[5], uma *framework* de desenvolvimento *web* de alto nível em Python.

As outras tecnologias que estão a ser utilizadas são:

- PostgreSQL[6] como Sistema de Gestão de Base de Dados (SGBD);
- HTML, CSS e Javascript para gerar a interface *web*.

[5] `https://www.djangoproject.com/`
[6] `http://www.postgresql.org/`

Django

Inicialmente, estava prevista a utilização da *framework* Ruby on Rails, mas esta acabou por não ser utilizada. Isto deve-se ao facto de os responsáveis pelo projecto na empresa terem pensado implementar controladores directamente sobre o Redmine, antes de lançarem o estágio. No entanto, uma análise mais profunda acabou por lhes indicar que seria mais vantajoso ter uma API que interagisse com o Redmine, para tornar a plataforma mais flexível. Desta forma, torna-se mais fácil adaptar o sistema a novas fontes de dados, caso seja necessário ou interessante para a empresa. Posto isto, para este caso, o Django enquadrava-se melhor no *know-how* da empresa, o que facilitaria o desenvolvimento e o próprio acompanhamento do estágio.

Assim, e de acordo com as explicações dos autores da ferramenta [18], as principais vantagens tiradas da utilização da *framework* Django são:

- Boa documentação;
- Facilidade e rapidez de desenvolvimento e *deployment*;
- Estabilidade;
- Escalabilidade.

Outra grande vantagem do Django é ser bastante fácil de dominar e permitir criar aplicações Web com menos esforço. Comparando, por exemplo, com o Ruby on Rails, as opiniões dividem-se, havendo quem prefira uma sobre a outra. Uma das vantagens do Ruby on Rails é ter uma maior brevidade no código [19]. No entanto, isto pode levar a que este seja mais difícil de entender. O Django tem por isso a vantagem de ser mais explícito.

API: *Framework* Tastypie

A parte fulcral deste projecto passava pelo desenvolvimento de uma camada de API RESTful. Para o fazer, foi escolhida a *framework* Tastypie. Esta *framework* permite desenvolver APIs por cima dos modelos definidos no Django. O Tastypie foi então utilizado para facilitar o desenvolvimento da API, dado que trata de uma parte do processamento dos pedidos e tem já implementados métodos para todas as acções RESTful. Assim, é possível adicionar funcionalidades mais complexas a estes métodos, sem ter que desenvolver as partes mais básicas da API. De facto, o Tastypie facilita esta adição de funcionalidades, fornecendo vários *hooks*[7], que dão ao programador a pos-

[7]Partes do código que permitem alterar o comportamento de algumas funções em casos específicos, sem ter que alterar o código-fonte da *framework*, como explicado em `http://www.codeproject.com/Articles/2082/API-hooking-revealed`

sibilidade de criar comportamentos diferentes onde necessário, sem ter que alterar o código fonte da *framework*. Isto era essencial neste trabalho, visto que, em muitos módulos, são necessários comportamentos mais complexos do que apenas ler/escrever/editar/eliminar - é necessário aceder a fontes de dados externas, obter e combinar dados de vários modelos, etc.

Haveria outras ferramentas a considerar, sendo que destas se destacam o Django-Piston e o Django-rest-framework. No entanto, a primeira já está algo obsoleta, sendo que a sua última versão não é sequer compatível com as últimas versões do Django. Relativamente à *rest-framework*, oferece um conjunto de funcionalidades semelhante ao Tastypie mas, aqui, o *know-how* já existente na empresa relativamente a esta última acabou por determinar a sua utilização.

As principais vantagens desta ferramenta são, tais como enunciadas em [20]:

- Boa documentação;
- Bem testada;
- Facilidade de utilização;
- Bom leque de recursos incluídos por defeito: (autenticação, *throttling*, *caching*, paginação, suporte para JSON, etc.);

PostgreSQL

A escolha de SGBD recaiu sobre o PostgreSQL. Um factor que pesou muito nesta escolha, inicialmente, foi a mudança da *framework* de desenvolvimento de Ruby para Django. Existia também um conhecimento mais profundo por parte da empresa em relação à integração de Django e PostgreSQL, pelo que o MySQL, escolhido inicialmente, acabou por ser preterido. Os próprios criadores do Django recomendam a utilização do PostgreSQL, indicando que "alcança um bom equilíbrio entre custo, características, rapidez e estabilidade"[21].

No entanto, é pertinente fazer uma comparação entre o PostgreSQL e outras ferramentas *open-source* como o MySQL. Embora as diferenças entre as duas ferramentas não sejam muito grandes, podemos ter também em conta a *performance* de uma e outra. Uma comparação feita usando o *benchmark* TPC-H[8] mostra que a *performance* do PostgreSQL é ligeiramente superior à do MySQL na maioria das *queries* [22].

[8] `http://www.tpc.org/tpch/`

Para além disto, o PostgreSQL tem a vantagem de permitir o uso de soluções como o PostGIS e o Postgres-XC. O PostGIS possibilita a expansão do UIS para a produção de relatórios georreferenciados; por sua vez, o Postgres XC permite tornar esta plataforma escalável em futuros desenvolvimentos e em ambientes baseados na *cloud*, caso esta plataforma venha a assumir um carácter comercial.

Integração com outros sistemas

O sistema prevê também a integração com as plataformas já existentes. Estas plataformas são o Redmine, para a extracção de dados sobre projectos anteriores que permitam calcular as métricas que lhes são referentes, e o LDAP, para a autenticação de utilizadores usando as suas credenciais da empresa.

A interacção com o Redmine é totalmente feita através da API RESTful do mesmo[9], tendo sido construído um módulo que serve de interface entre o UIS e o Redmine.

Para a integração da autenticação com o LDAP, é usada a extensão Python-LDAP, que oferece "uma API orientada a objectos para o acesso aos serviços de directório do LDAP, a partir de programas em Python"[10].

Para os dados extraídos da Intranet, e visto que esta plataforma assentava numa base de dados MySQL, foram criados algumas funções e comandos em Python, usando a biblioteca MySQLdb[11].

Para o futuro, ainda que fora do âmbito do estágio, foi posta a possibilidade de integração com outros sistemas como, por exemplo, o Jenkins, já mencionado anteriormente, ou o Google Docs.

[9] `http://www.redmine.org/projects/redmine/wiki/Rest_api`

[10] `http://www.python-ldap.org/`

[11] Biblioteca para fazer a ligação a bases de dados MySQL, `http://sourceforge.net/projects/mysql-python/`

Capítulo 5

Implementação

Nesta secção, são explicados os detalhes sobre a implementação do projecto. A secção divide-se na implementação da API, onde são apresentados os métodos e objectos mais importantes da mesma. São também apresentados os mecanismos de extracção, transformação e carregamento de dados de plataformas externas, criação e leitura de ficheiros, de moderação, autenticação e autorização e as páginas de *admin*.

5.1 API

Os métodos da API permitem executar as funções REST. Assim, torna-se fundamental perceber estes métodos para ter um melhor conhecimento do funcionamento do sistema. Como tal, de seguida, são descritos os métodos mais importantes que dão suporte a cada uma das funções REST. Para o fazer, recorre-se a um exemplo em que a interface gráfica *web* já foi implementada: a gestão de *dailychecks*. Os *dailychecks* referem-se à informação sobre as horas de entrada e saída de cada funcionário, para cada dia da semana. Para fazer esta gestão, é apresentado ao utilizador um calendário para cada mês, com a informação de faltas, férias e horas de trabalho para cada dia. São ainda mostradas algumas métricas como o número de horas por cumprir para esse dia, para esse mês, entre outras.

Com este exemplo, é possível mostrar o funcionamento das acções GET/ POST/PATCH/DELETE.

GET

Aquando do acesso ao ecrã mostrado na figura 4.4, são feitos alguns pedidos GET à API, de acordo com aquilo que é mostrado na figura.

Ao receber o pedido GET, o Django processa o *url*, criando uma nova instância do recurso (classe do Tastypie) correspondente. Aqui, é chamado o método *dispatch* que vai processar o pedido e chamar o método necessário. Este método, no caso de um pedido GET, pode ser *get_list* ou *get_object*, dependendo de se tratar de um pedido de lista ou de um objecto específico.

Nas figuras 5.2 e 5.1, podemos ver os diagramas de sequência para os métodos *dispatch* e *get_list*, neste caso para o recurso *VacationInfoResource*.

No primeiro diagrama, podemos ver que o método *dispatch* se encarrega de verificar se um método é válido. Esta validação serve para verificar se a acção REST requerida existe para este recurso. De seguida, é feita outra verificação para ver se o utilizador que fez o pedido está autenticado. Finalmente, antes da chamada ao método correspondente, é feita uma verificação de *throttle*, para verificar se o utilizador excedeu o número de pedidos estabelecido. A chamada ao método é feita de acordo com o pedido executado. Assim, neste caso, foi feito um pedido GET ao *endpoint vacations*, pelo que vai ser chamado o método *get_list*. Se o pedido feito fosse relativo a um pedido específico (por exemplo, vacations/2 para aceder ao recurso *vacations* com id 2), seria chamado o método *get_object*.

No diagrama 5.1, é mostrado o método *get_list*. Este método é responsável por obter a lista de recursos pedidos pelo utilizador. Assim, em primeiro lugar, é chamado o método *obj_get_list*. Este método constrói quaisquer filtros que o utilizador tenha especificado (as férias, por exemplo, podem ser filtradas por data de início, fim ou utilizador) e chama o método *get_object_list* do objecto *VacationsInfo*. Esta última chama a função *objects.get* que obtém da base de dados as entradas correspondentes aos filtros passados. De seguida, a função *get_list* chama o método *authorized_read_list* da classe *VacationsAuthorization*, que vai aplicar filtros à lista retornada anteriormente consoante o nível de autorização do utilizador. Já tendo a lista final de objectos a retornar, esta vai ser ordenada e paginada e, de seguida, é aplicado o *full_dehydrate*, sendo a informação contida nos objectos da lista convertida para um objecto *bundle*, com um dicionário com os dados a serem mostrados. Estes dados são depois serializados para o formato desejado e retornados.

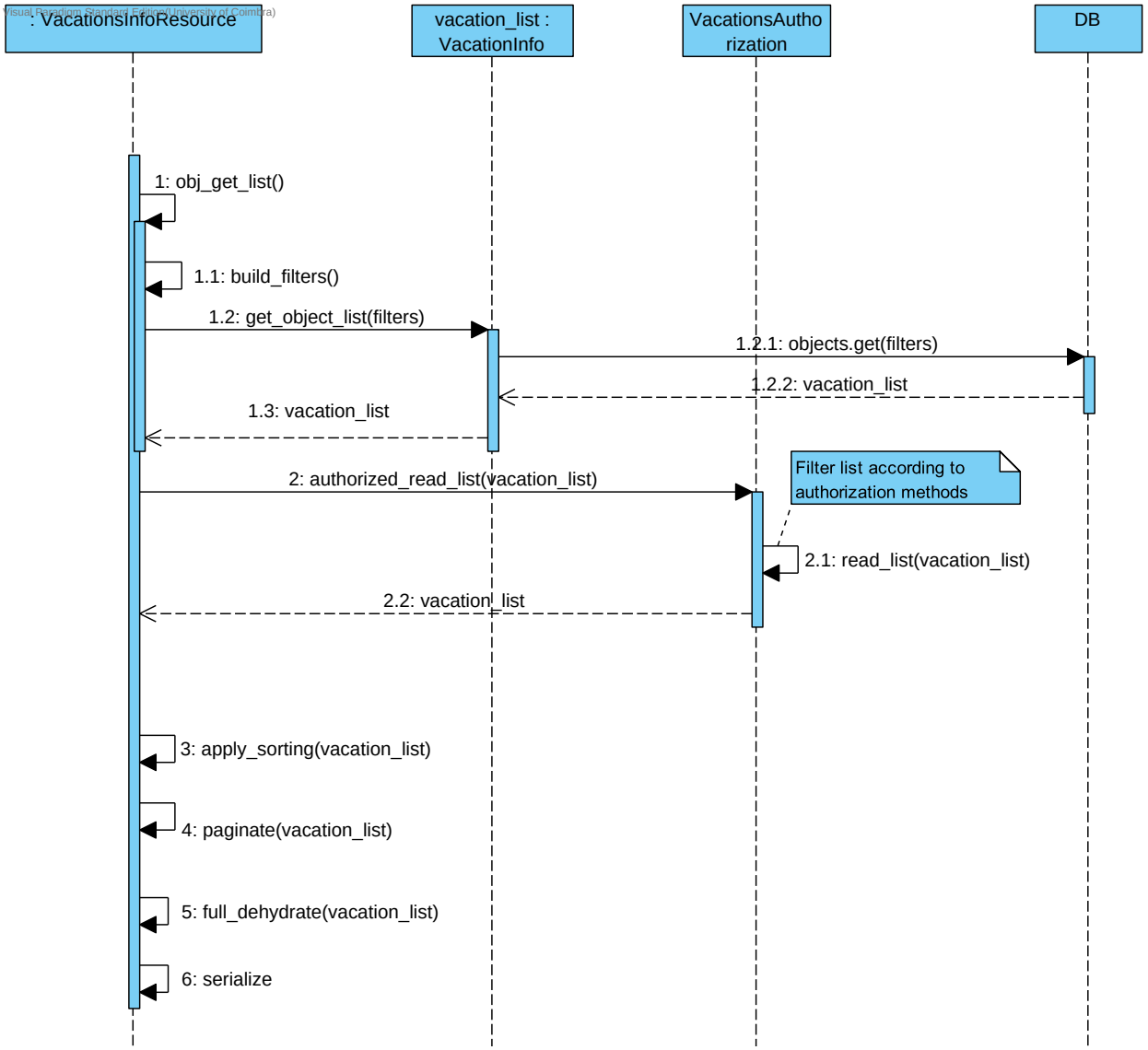

Figura 5.1: Diagrama de sequência para o método *get_list* no módulo *Vacations*

POST

Para o verbo POST, o método usado é o *post_list*. Este método faz primeiro a "desserialização" do conteúdo enviado através do corpo do pedido. De seguida, constrói um objecto *bundle* em que este conteúdo é guardado no atributo *data*. Depois, é chamado o método *obj_create*. Este é responsável pela criação propriamente dita do objecto. Aqui, é criado um

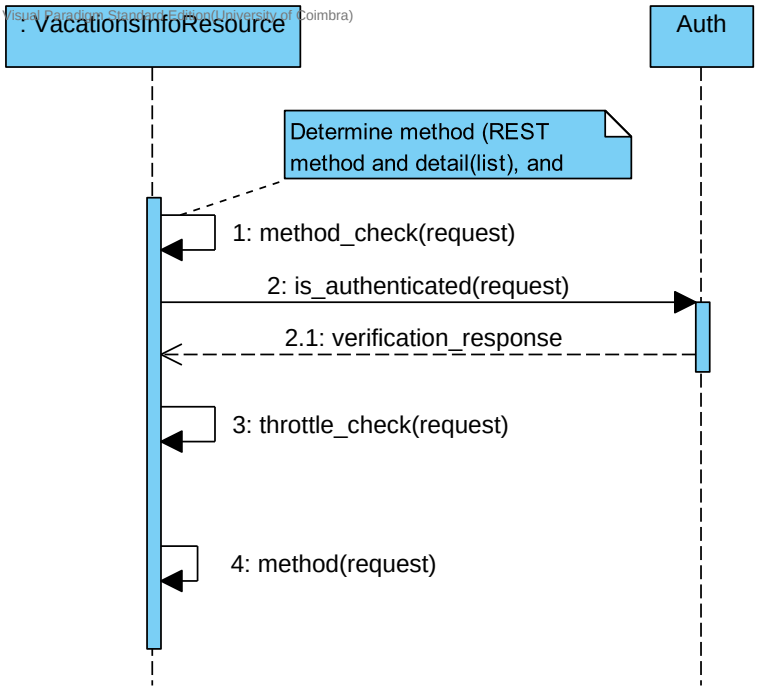

Figura 5.2: Diagrama de sequência para o método *dispatch*

objecto do tipo correspondente ao modelo usado, sendo depois chamado o método *full_hydrate*, explicado mais à frente, que vai atribuir ao objecto criado os valores passados através do corpo do pedido. Para além disso, este método pode ainda fazer o cálculo de outros valores, como explicado mais à frente. Finalmente, o objecto é guardado na base de dados e é retornada a resposta apropriada.

O funcionamento dos métodos PATCH e DELETE é bastante semelhante a este, pelo que o seu funcionamento não será explicado de forma tão detalhada. Quanto ao primeiro, apenas difere do POST na medida em que é obtido o objecto sobre o qual se pretende realizar a acção, são alterados os seus atributos conforme a informação passada no corpo do pedido e, finalmente, o objecto é guardado. Para o DELETE, o objecto é obtido e apagado, sendo removido da base de dados.

Objectos *bundle*

Os objectos do tipo *bundle* são um dos conceitos centrais do Tastypie. Um *bundle* é uma abstracção que permite associar os dados passados à aplicação

a uma instância não guardada de um objecto. Quando é feito um pedido a um *endpoint*, o método *build_bundle* é responsável pela criação deste objecto. Um *bundle* é um objecto associado ao modelo ao qual o *endpoint* diz respeito. Os objectos deste tipo têm os atributos:

- *data*: um dicionário que contém a informação passada através do corpo do pedido efectuado.
- *obj*: um objecto Python do mesmo tipo do modelo correspondente.
- *request*: um objecto do Django que contém informação sobre o pedido HTTP efectuado.

Se tivermos como exemplo um pedido do tipo PATCH, o *bundle* vai conter em *obj* a instância do objecto gravado que queremos editar, enquanto que o atributo *data* vai conter os campos que queremos alterar. Isto permite-nos alterar os atributos do objecto de acordo com os dados recebidos e guardá-lo. No entanto, se fizermos um pedido POST, os dados que forem passados por JSON à aplicação vão ser guardados em *data*, enquanto que *obj* vai ser um objecto vazio. Desta forma, podemos, através do método Hydrate explicado mais à frente, atribuir os valores desejados aos atributos do objecto a criar. No pedido DELETE, *obj* vai conter o objecto a apagar, enquanto que *data* vai estar vazio. Finalmente, no pedido GET, cada *bundle* vai corresponder a uma entrada a apresentar, tendo *obj* a instância do objecto correspondente.

Ciclos Hydrate e Dehydrate

Um dos conceitos centrais da *framework* Tastypie é o ciclo *Hydrate/ Dehydrate*. Estes são os métodos que tratam da conversão de dados para o modelo e vice-versa.

O ciclo *Dehydrate* consiste na serialização da informação presente num modelo um dado tipo e conversão para uma estrutura de dados, mais simples, a ser consumida pelo cliente. O Tastypie permite então fazer o *override* deste método quando necessário. Isto é útil para casos em que a informação a que queremos aceder não é guardada na base de dados, o que acontece quando queremos obter valores que são calculados em *runtime*. Isto acontece, por exemplo, quando acedemos ao número de dias de férias que um utilizador ainda tem num dado ano. Este valor não é guardado para evitar a sua actualização de cada vez que há uma alteração nos períodos de férias de um utilizador. Para o obter, basta então obter da base de dados todos os períodos de férias do utilizador para esse ano e subtrair o total de dias úteis destes períodos aos seus dias totais disponíveis. O método *Dehydrate* verifica que existe um método *dehydrate_available_days* (sendo *available_days*, neste

caso, o nome do campo para o qual queremos alterar o comportamento do método) e chama-o, sendo estes cálculos efectuados neste método.

O ciclo *Hydrate* trata do processo inverso: a conversão de uma estrutura de dados recebida do cliente para o modelo de dados correspondente. Aqui, podemos alterar o comportamento para guardar valores calculados através de outros valores introduzidos pelo utilizador, por exemplo. Um exemplo deste caso, mais uma vez para o módulo *vacations*, é o cálculo do número de dias de trabalho que um dado pedido de férias irá ocupar. Assim, basta criar o método *hydrate_business_days*, que calcula este número, e é chamado pelo método *hydrate* aquando da introdução de dados por parte do utilizador.

Override de métodos

Os comportamentos descritos acima são assegurados pela própria *framework* Tastypie. Assim, a maior parte do trabalho centrou-se em fazer o *override* dos métodos necessários para acrescentar funcionalidades àquelas que esta ferramenta apresenta por defeito. Assim, para além do exemplo dado na subsecção anterior relativo aos ciclos *hydrate* e *dehydrate*, existem vários outros exemplos. Estes passam pelos métodos utilizados pelos mecanismos de autorização, como veremos mais à frente, e, principalmente, pelos métodos usados nos pedidos GET/POST/PATCH/DELETE. Alguns exemplos são o cálculo de campos que dependam de outros campos ou modelos nos métodos GET (número de dias de férias restantes para um utilizador, por exemplo), a verificação da correcção de valores introduzidos usando o método POST (verificar se um utilizador tem dias de férias disponíveis ao introduzir um novo pedido de férias), ou a verificação da validade de valores alterados usando o método PATCH (verificar se o utilizador está a tentar mudar férias já rejeitadas). Outros exemplos prendem-se com os pedidos que requerem que seja obtida informação do Redmine em tempo real. Nestes, incluem-se pedidos GET cuja informação depende do Redmine (a informação sobre orçamentos depende da informação sobre *sprints*, que tem que ser acedida no Redmine por cada vez que é feito um pedido) ou PATCH em que queremos actualizar informação do UIS para a versão mais recente presente no Redmine (informações sobre projectos, por exemplo).

Outros motivos que levam a que tenha que ser feito o *override* de algumas funções prende-se com o facto de haver determinadas funcionalidades que não são suportadas pela *framework* ou pelo próprio Django. Uma das funcionalidades não suportadas pelo Django é a utilização de chaves compostas[1], que são necessárias em muitos dos modelos mostrados acima. Assim,

[1]Chave construída com 2 ou mais campos que identifica univocamente uma entrada da tabela.

nestes modelos, é gerado automaticamente um id unívoco, mas foi necessário desenvolver mecanismos para garantir que, aquando da introdução de registos na base de dados, não são introduzidos valores já existentes.

5.2 Dados de sistemas externos

Como havia sido discutido anteriormente, o UIS necessita de dados de fontes distintas. À data de conclusão do presente estágio, integradas as fontes Redmine, Intranet, LDAP e ficheiros xlsx.

Dado que este sistema já substituiu o uso da Intranet na empresa, foi apenas necessário obter os dados aquando da entrada do sistema em produção. Para este efeito, o estagiário desenvolveu um processo de extracção, transformação e carregamento de dados. Assim, os campos das tabelas da Intranet foram mapeados para os atributos dos modelos correspondentes no UIS. Depois, foram desenvolvidos *scripts* em Python que, usando a biblioteca MySQLdb, obtêm os dados directamente da base de dados MySQL da Intranet. Esta biblioteca permite executar *queries* SQL sobre a base de dados e fazer o *parse* dos dados recebidos. São feitas todas as verificações necessárias para que não sejam introduzidos no UIS dados incorrectos ou incompletos, isto porque existem entradas relativas a alguns módulos (nomeadamente *dailycheck* e de *vacations*) em que há valores incorrectos, como campos a *null* ou entradas repetidas. Quando se verifica que há dados incorrectos, as entradas correspondentes são ignoradas. Caso passem estas verificações, os valores são transformados segundo os mapeamentos referidos anteriormente e guardados na base de dados do UIS.

Relativamente ao Redmine, foram implementadas funções que permitem que obter os dados tanto através de comandos, como nos próprios pedidos à API. Isto é necessário visto que há dados que são obtidos periodicamente, enquanto que outros são acedidos em *runtime*. As decisões tomadas relativamente ao tipo de acesso aos dados prendeu-se com um *tradeoff* entre a frequência de modificação destes na fonte, e a performance, dado que os acessos ao Redmine através da API são bastante lentos. Assim, para dados que são alterados menos frequentemente, é feita uma actualização periódica através de *cronjobs* que correm os comandos necessários. Este é o caso para toda a informação de projectos e equipas. Para a informação de *sprints*, os acessos são feitos em *runtime*, dado que a informação é alterada muito mais frequentemente. Isto acontece porque os utilizadores vão introduzindo a informação relativa às tarefas (que afecta a taxa de completude do *sprint*) ao longo do dia.

Quando é necessário aceder à informação guardada no Redmine, são uti-

lizadas as funções construídas no módulo correspondente. Cada função diz respeito a um tipo de dados (project, *sprints*, tarefas, etc.). De forma geral, estas funções recebem *ids* ou outros parâmetros que permitam filtrar os dados (estado de um projecto, data de início de uma *sprint*, etc.), e constroem o URL a ser usado para aceder ao Redmine. De seguida, a função passa este URL à função *get_from_redmine_api*, que faz o pedido e retorna o JSON de resposta.

5.3 Criação e leitura de ficheiros xlsx

Como havia sido descrito na arquitectura do sistema, foi feita a integração com documentos Excel. Esta integração inclui a leitura e criação de ficheiros deste tipo.

Para o fazer, foi usada a biblioteca *openpyxl*[2], que permite ler e gravar ficheiros xlsx. Usando esta biblioteca, foram criados *endpoints* específicos para dar suporte às funcionalidades que envolvem estes ficheiros. Nomeadamente, o método POST no *endpoint vacations_export* cria um "mapa" com as férias de utilizadores, a partir de um *template*, podendo o utilizador receber o ficheiro através do método GET. Por sua vez, o método POST no *endpoint dailychecks_ptin* serve para importar para o UIS os *dailychecks* relativos aos colaboradores da Ubiwhere que se encontram a trabalhar nas instalações da PT Inovação[3]. Este ficheiro é gerado automaticamente pelo sistema de controlo de entradas desta empresa, ficando preenchidos os tempos de entrada e saída da empresa para cada trabalhador em cada dia. O ficheiro é lido, sendo criadas as entradas respectivas na tabela *dailychecks*.

A leitura dos ficheiros é feita através da função *load_workbook* do *openpyxl*, que recebe como parâmetro o *path* para o ficheiro a ler. De seguida, o *parse* do ficheiro é feito recorrendo ao método *cell*, que permite aceder aos valores de cada célula, recebendo como parâmetro as coordenadas respectivas. O atributo *value* guarda o valor da célula, podendo este ser acedido e alterado. Assim, estando os documentos estruturados de uma maneira específica, é possível fazer a sua leitura e escrita conforme desejado.

[2] `https://pythonhosted.org/openpyxl/`

[3] PT Inovação, empresa de comunicações e TI, `http://www.ptinovacao.pt/en/about.html`

5.4 Moderação

Foi implementado um módulo que permite fazer a moderação de alguns pedidos. A moderação consiste num mecanismo de "supervisão" de pedidos por parte de um utilizador responsável. Um exemplo pode ser dado para o módulo de férias. Se tivermos um pedido de alteração de um período de férias já aprovadas ou rejeitadas pela administração, este pedido fica automaticamente em moderação. Isto significa que o pedido não é directamente efectuado, mas que vai ser guardado e à espera de aprovação. Isto permite que o estado do pedido seja alterado apenas se a administração assim o entender. Isto dá maior controlo à administração sobre pedidos já aprovados, o que evita que estes sejam constantemente mudados.

O exemplo aqui anterior pode ser generalizado, dado que esta funcionalidade foi desenvolvida de maneira a poder ser aplicada a qualquer outro módulo em que se justifique. O diagrama de actividade da figura 5.3 permite perceber melhor o funcionamento deste módulo.

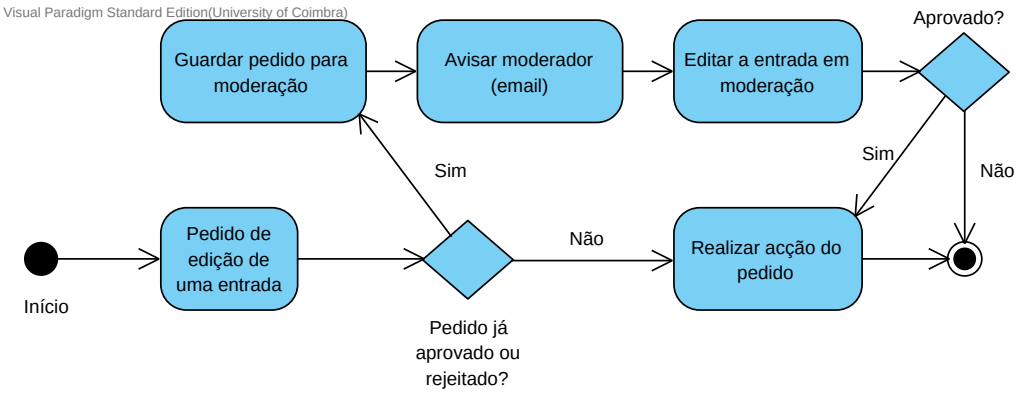

Figura 5.3: Diagrama de actividade do módulo *Moderation*

Em termos de implementação, como já analisado na secção 4.4, o módulo possui um modelo *Moderation*, que contém informação sobre o objecto a alterar (o tipo e a própria instância), o *bundle* correspondente ao pedido enviado, a data, o utilizador, os utilizadores a quem cabe a responsabilidade de moderar o pedido, e o estado de moderação (pendente, aprovado ou rejeitado). Quando queremos utilizar a moderação, basta chamar o método *to_moderation*, que se encarrega de guardar esta informação. O método GET no endpoint *moderation* permite a um utilizador verificar se tem algum pedido para moderar, podendo depois usar o método PATCH para alterar o estado da moderação. Se o estado da moderação for alterado para "aprovado", é utilizado o método *moderate*, que chamará a função necessária para

realizar o pedido que estava pendente; caso contrário, o estado é alterado para rejeitado e o pedido não é realizado.

5.5 Autenticação e autorização

Num sistema deste tipo, os mecanismos de autenticação e autorização são essenciais para assegurar que a informação é disponibilizada exclusivamente a quem a deve poder consultar. Assim, são implementados sistemas de autenticação (responsáveis por verificar se o utilizador existe) e de autorização (responsáveis por verificar se o utilizador registado tem permissões para efectuar a acção requerida).

Autenticação

Para a gestão de autenticações, como já referido anteriormente, foi usado o módulo *python-ldap*. Este módulo permite obter as credenciais guardadas no directório *ldap* da empresa, pelo que os utilizadores não têm que criar novas contas para aceder a este sistema.

Ao fazer *login* através do método POST *login* da API, é atribuída ao utilizador uma *API Key*, caso as credenciais estejam correctas (o utilizador é criado automaticamente no UIS caso ainda não esteja registado). Esta API Key deverá depois ser enviada em cada pedido à API, através do *header* HTTP *Authentication*.

Autorização

Para a gestão de autorizações, foi criado um módulo específico. Este módulo possui uma classe com os métodos de autorização para cada tipo de pedido (GET, POST, PATCH/PUT, DELETE). No entanto, cada módulo implementa uma nova classe de autorização derivada desta, fazendo *override* dos métodos necessários. Isto dá-se devido aos diferentes esquemas de autorização concebidos para cada módulo. Assim, um PM tem, por exemplo, acesso a todos os recursos relacionados com os seus projectos no módulo *Users*, mas tem acesso apenas aos seus recursos no módulo *Absences*. Os papéis de utilizador são os mesmos referenciados na secção 4.1, aos quais é acrescentado o papel de *Middle Management*. Estes utilizadores têm privilégios semelhantes aos do utilizador administrativo nalgumas situações. Na figura 5.4 é esquematizado o sistema de autorizações por acção e por *role* na empresa.

		User Profile	User Vacations	User Skills & CV	User Contract Info	Vacations Request	Available vacation days	Daily Check	Effort	Checkin Checkout	Sprint Metrics	Project Metrics	Portfolio Metrics	Business Metrics	Absences	Allocations	Holidays	Projects Profile	Project Budget	Effort aggregates	Team Info	Close Dailycheck Periods
TLM	GET	y	y	y	y	y	y	y	y	---	y	y	y	y	y	y	y	y	y	y	y	y
	PATCH	y	---	y	y	y	y	---	---	---	y	y	y	y	y	y	y	y	y	---	y	---
	POST	---	---	y	y	y	y	---	---	---	y	y	y	y	y	y	y	y	y	---	y	y
	DELETE	---	---	y*	---	y	y	---	---	---	---	---	---	---	y	y	y	---	y	---	---	y
MM	GET	y	y	y	y*	y	y	y	y	---	y	y	y	y	y	y	---	y	y	y	y	y
	PATCH	y	---	y	---	---	---	---	---	---	---	---	y	---	---	y	---	y	---	---	y	---
	POST	---	---	y	---	---	---	---	---	---	---	---	y	---	---	y	---	y	---	---	y	y
	DELETE	---	---	y*	---	---	---	---	---	---	---	---	---	---	---	y	---	---	---	---	---	y
TM	GET	y	y	y	y*	---	own	---	---	---	own	own	---	---	y	y	---	y	---	own	y	---
	PATCH	---	---	---	---	---	---	---	---	---	own	own	---	---	---	own	---	---	---	---	---	---
	POST	---	---	---	---	---	---	---	---	---	own	own	---	---	---	own	---	---	---	---	---	---
	DELETE	---	---	---	---	---	---	---	---	---	---	---	---	---	---	own	---	---	---	---	---	---
PM	GET	y	y	y	y*	---	own	---	---	---	own	own	---	---	y	y	---	y	own	own	y	---
	PATCH	---	---	---	---	---	---	---	---	---	own	own	---	---	---	own	---	own	own	---	own	---
	POST	---	---	---	---	---	---	---	---	---	own	own	---	---	---	own	---	own	own	---	own	---
	DELETE	---	---	---	---	---	---	---	---	---	---	---	---	---	---	own	---	---	---	---	---	---
User	GET	y	y	y	own	own	own	own	own	own	own	own	---	---	own	own	y	y	---	own	y	---
	PATCH	own	---	own	---	own	---	own	---	own	---	---	---	---	own	---	---	---	---	---	---	---
	POST	---	---	own	---	own	---	own	---	own	---	---	---	---	own	---	---	---	---	---	---	---
	DELETE	---	---	own	---	own	---	own	---	---	---	---	---	---	own	---	---	---	---	---	---	---

Figura 5.4: Tabela de autorizações por acção

As células a verde representam acções que podem aceder a todos os dados disponíveis. As células a amarelo representam acções que podem aceder a apenas alguns dados (por exemplo, relativos ao próprio utilizador) e as células a vermelho acções que o utilizador não tem autorização para efectuar. Para além disto, os utilizadores possuem sempre todas as permissões dos utilizadores do "nível abaixo", à excepção do caso dos PMs e TMs, que são considerados a um nível semelhante.

Assim, por exemplo, no módulo *Allocations*, o utilizador sem permissões especiais pode ver apenas as suas próprias alocações, e não pode efectuar nenhuma outra operação sobre elas. O *Project Manager* podem, por sua vez, ver as informações das alocações de todos os outros utilizadores, mas apenas podem efectuar as restantes operações sobre dados relativos a utilizadores que lhes estejam relacionados (ou seja, utilizadores inseridos nos projectos ou equipas do PM ou TM, respectivamente). Desta forma, os PMs e TMs podem fazer um plano das alocações dos colaboradores para o mês seguinte, podendo estas ser aprovadas posteriormente pelos administradores.

Relativamente à implementação deste sistema, foi criada uma classe que herda da classe de autorização do Tastypie. Esta *framework*, tal como já foi referido, oferece vários *hooks* para modificar os comportamentos de algumas partes e, no caso das autorizações, isso é feito através de funções específicas para cada método REST. Assim, temos métodos read/write/ update/delete para objectos específicos ou listas. No módulo Authorization, foi implementado um esquema geral que segue os casos apresentados na figura 5.4, implementando os casos mais comuns. Depois, pode fazer-se o *override* em alguns módulos, cujo comportamento seja diferente. Por exemplo, no caso mais geral, o PM pode ler informação relativa a todos os funcionários que estejam envolvidos nos seus projectos. No entanto, no módulo *checkin*, por exemplo, o PM não pode aceder a esta informação. Desta forma, basta implementar neste módulo as funções *read_list* e *read_detail* para que reflictam este comportamento.

5.6 Páginas de *admin*

Houve algumas funcionalidades que foram desenvolvidas no que à API diz respeito, que acabaram por não ser implementadas no *frontend* à data de conclusão do estágio. Como tal, o estagiário utilizou uma funcionalidade disponibilizada pelo Django, que diz respeito às páginas de *admin*. Estas são páginas geradas pelo Django e que, baseadas nos modelos respectivos, apresentam uma listagem de objectos desses modelos. Ao clicar nestes objectos, o utilizador pode ver mais detalhes sobre eles, e até editá-los ou apagá-los. Há

ainda a possibilidade de criar novos objectos. O estagiário pôde definir quais os módulos a serem representados nestas páginas, bem como definir quais os campos a apresentar, e quais deles podem ser pesquisáveis ou filtrados. Na imagem 5.5, podemos ver a página de *admin* do módulo *absences*.

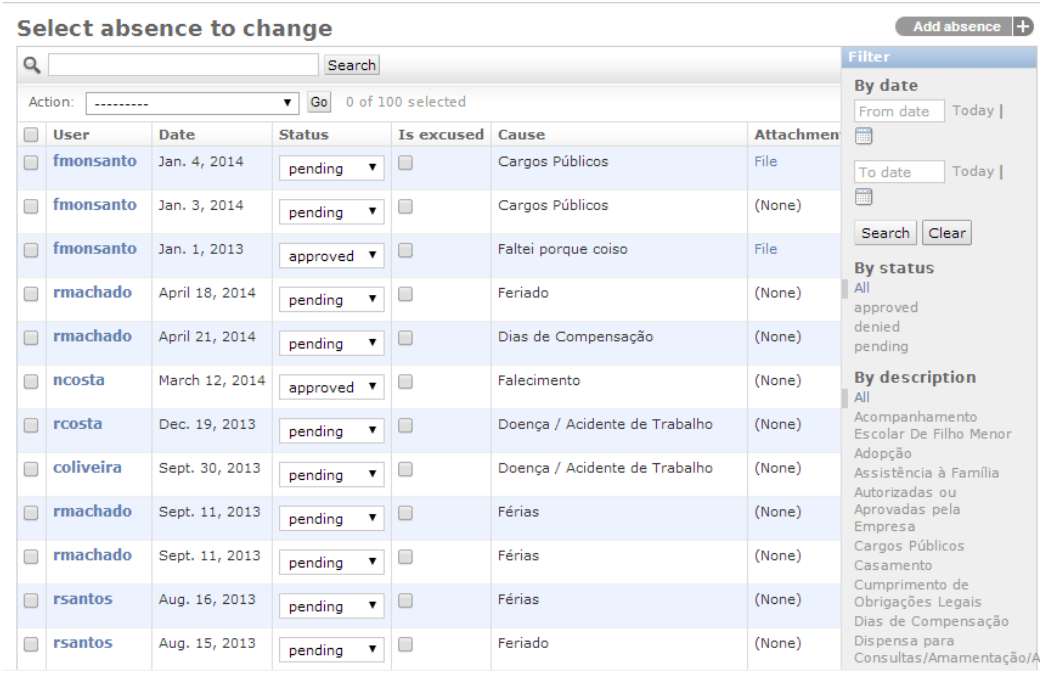

Figura 5.5: Página de administração para o módulo *absences*

A utilização desta característica do Django permitiu que alguns dos módulos pudessem já ser usados pela administração, mesmo sem que a interface *web* estivesse terminada. No entanto, há que ressalvar que esta não seria uma opção viável para fazer a gestão do sistema, visto que as verificações que foram implementadas no Tastypie para certificar que, por exemplo, não são inseridos campos com valores não permitidos, não são aplicadas aqui. Assim, estas páginas funcionam sobretudo como uma maneira simplificada de ver os objectos guardados no UIS e poder editá-los se assim for necessário.

Capítulo 6

Framework de métricas

Nesta secção, são apresentadas as métricas usadas na Ubiwhere e é feita uma reflexão sobre a maneira como o trabalho realizado pode ser aproveitado para construir uma *framework* de métricas de projecto e financeiras. O estagiário achou necessário incluir este capítulo visto que a parte relativa às métricas é a mais importante do projecto. Isto deve-se ao facto de a certificação do nível 3 do CMMI ser uma das grandes motivações para este projecto, sendo a obtenção destas métricas indispensável para que a certificação possa ser obtida. Para além disto, a obtenção destas métricas ajuda a melhorar as práticas do Scrum, como explicado em 1.2.

Em primeiro lugar, é feita a contextualização deste capítulo, explicando as métricas existentes na Ubiwhere através de um exemplo concreto. De seguida, é explicado o funcionamento actual do sistema de métricas. Finalmente, é apresentada a especificação daquilo que poderá ser esta *framework* no futuro, incluindo uma explicação do seu funcionamento.

6.1 Contexto

Antes de mais, é importante mostrar quais as métricas que o que a Ubiwhere utiliza actualmente e a forma como são calculadas pelo sistema, com as respectivas hierarquias e dependências. Nas figuras E.1 e E.2 do anexo E são apresentadas estas métricas.

Como podemos ver, foram definidos 4 níveis de métricas: *sprint*, projecto, portfólio de projectos e negócio. Algumas das métricas de um dado nível são alimentadas pelas do nível anterior. Os primeiros dois níveis referem-se a métricas calculadas no âmbito das *sprints* e dos projectos. O nível de *sprint* é, por sua vez, alimentado pelos dados relativos ao trabalho de cada colaborador, e podem ser recolhidos em todos os níveis organizacionais ou por

equipas. O nível de portfólio contém métricas que são calculadas usando as métricas de um determinado conjunto de projectos, que pode ser seleccionado consoante uma série de características dos projectos (tipo de clientes, datas de início e fim, etc.), ou mesmo de todos os projectos da empresa. Finalmente, as métricas de negócio dizem respeito a uma série de indicadores financeiros e de qualidade, calculados através das métricas do portfólio dos projectos da empresa.

Tomemos como exemplo a métrica "# (número) de bugs introduzidos" ao nível de *sprint*. Esta é relativa ao número de *bugs* que foram introduzidos no projecto durante a *sprint* correspondente. Esta vai alimentar a métrica de projecto "# de bugs total", uma vez que o número de *bugs* total de um projecto é igual à soma de todos os *bugs* introduzidos durante as suas *sprints*. Esta métrica, por sua vez, vai alimentar a de portfólio relativa à média de *bugs* por projecto. Como podemos ver, existe uma espécie de hierarquia nas métricas, bem como uma série de dependências relativas aos níveis abaixo.

6.2 Funcionamento actual

Uma vez que o sistema desenvolvido é bastante flexível, será possível construir uma *framework* reutilizável noutras empresas para a geração de métricas. Para isto, seriam necessários os módulos *Projects* e *Sprints*, e o desenvolvimento de uma interface equivalente à que foi desenvolvida neste trabalho para a integração com o Redmine, para integrar com outra plataforma escolhida.

Os dados relativos às métricas de *sprint* são obtidos através do Redmine. Esta plataforma permite, através da API, obter os dados relativos a todas as tarefas de um *sprint* ou projecto. Assim, é feita uma contagem do item desejado. Se tomarmos como exemplo a métrica referida anteriormente relativa ao número de *bugs*, é efectuado um pedido ao Redmine que devolve todos os *bugs* de uma *sprint*. Assim, o valor da métrica vai corresponder ao número de itens retornados. Se a métrica for relativa ao total de horas, em vez do número total de entradas, é retornado o número de horas gastas nas tarefas respectivas. Estes valores servem então, como referido, de base às métricas do nível de projecto. Assim, podemos calcular o número médio de *bugs* por *sprint* ou o número de horas médio gasto na sua correcção. Este cálculo é efectuado quando é feito um pedido PATCH para o *endpoint project_metrics*.

O sistema foi desenvolvido de maneira a que estes níveis possam ser alterados. Também as próprias métricas podem ser alteradas, podendo ser modificadas a sua descrição e até a fórmula de cálculo. Desta forma, seria possível introduzir mecanismos que possibilitem a uma outra empresa pegar neste sistema e adaptá-lo às suas necessidades específicas. Estas alterações

podem fazer-se de forma semelhante às alterações que foram introduzidas pelo estagiário na *framework* Tastypie: através da utilização de *hooks* que permitem modificar o comportamento de base da *framework*.

Visto que, e à semelhança do resto do sistema, os resultados das métricas são disponibilizados através de uma API RESTful, é possível construir *dashboards* também eles personalizados e adaptados às necessidades da organização em causa. Na figura 6.1, podemos ver o exemplo da interface criada para as métricas de *sprint* da Ubiwhere. Aqui, podemos ver como toda a informação relevante sobre uma *sprint* fica disponível para visualização na mesma página, sob a forma de gráficos ou mesmo numérica, o que facilita muito a sua análise.

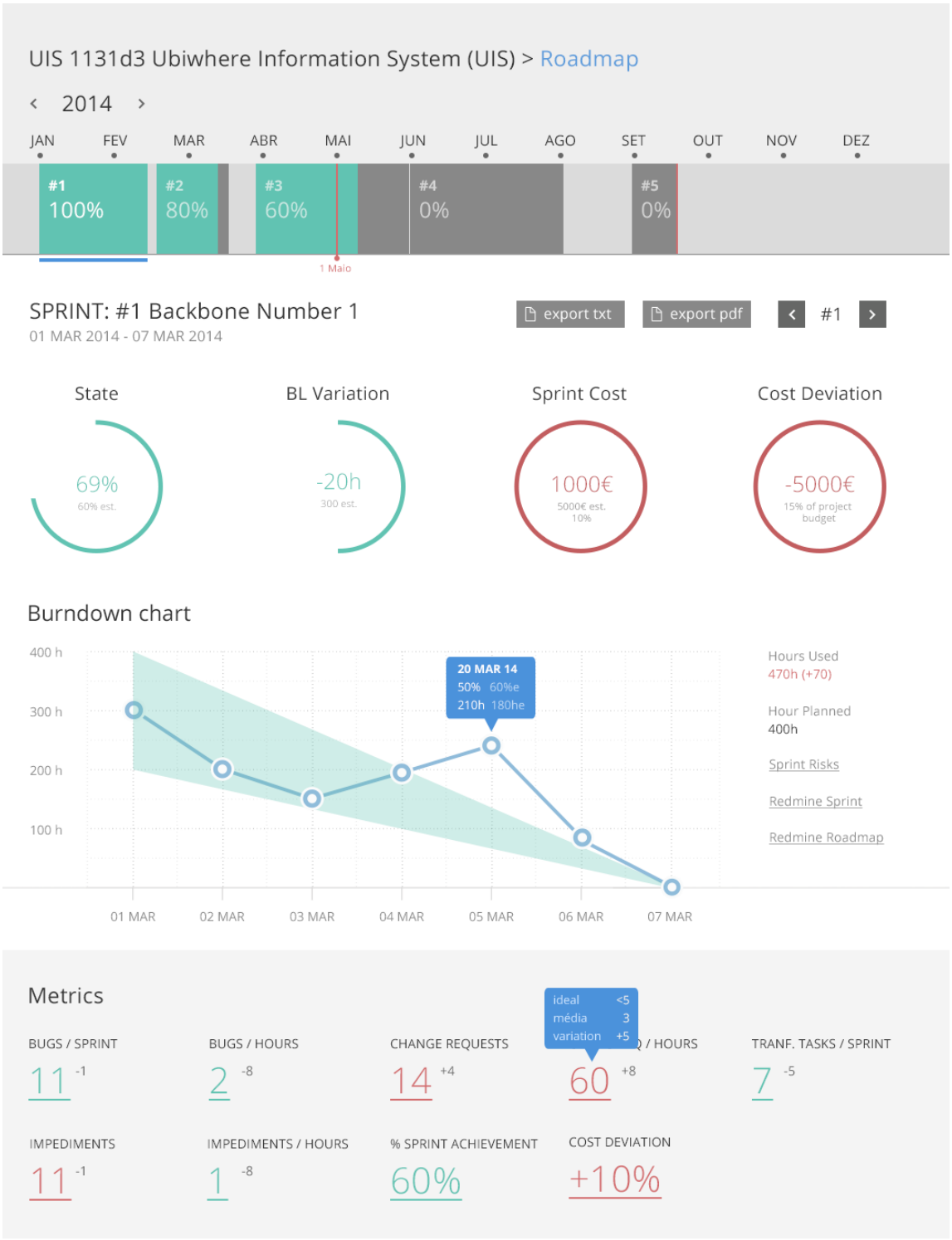

Figura 6.1: *Screenshot* do *dashboard* de métricas de *sprint*

6.3 Especificação

Para melhor especificar esta *framework*, foram criadas *user stories* que detalham as funcionalidades que esta deveria ter numa versão final. Estas *user stories* foram já introduzidas no *backlog* futuro do produto. Aqui, temos um *role*, Responsável, que não tinha sido especificado antes. Neste caso, o Responsável será uma pessoa que esteja acima na hierarquia definida para as autorizações, como mostrado na figura 5.4.

ID	Como	quero	para
MF1	PM	criar métricas de projecto personalizadas	garantir que vão de encontro às necessidades do projecto
MF1.1		combinar duas ou mais métricas para o cálculo de outras métricas	
MF1.2		criar métricas personalizadas usando fontes de dados externas	
MF2		criar métricas individuais personalizadas para os *end-users*	fazer com que estes tenham uma melhor noção da sua performance
MF3		atribuir métricas e objectivos aos membros do meu projecto	melhor aferir a performance deles no projecto
MF4	*End-user*	aceder às minhas próprias métricas e objectivos	saber a minha performance e o que é esperado
MF5	Responsável	moderar as métricas e objectivos criados	garantir que vão de encontro aos objectivos da empresa

Tabela 6.1: *User stories* para a *framework* de métricas

Embora já seja possível satisfazer estes requisitos, isto implica alterar, no próprio código, a maneira como se calculam as métricas e se obtêm os dados respectivos. A ideia passa então por desenvolver o módulo de tal forma que estas funcionalidades estejam disponíveis para o utilizador final, sem que ele tenha que alterar o código da aplicação.

Para além disso, seria necessário, para satisfazer a *user story* MF5, aplicar o módulo de moderação de pedidos a esta *framework*. Isto faria com que os PMs e TMs pudessem criar métricas que estivessem sujeitas a aprovação por parte da gestão de topo, cabendo a ela a verificação de que estas métricas são adequadas e se enquadram com os objectivos da empresa.

Concluindo, com a ferramenta criada, fica possibilitada a criação de uma matriz de métricas, com as respectivas ligações e precedências. Assim, é possível atribuir diferentes métricas e, consequentemente, objectivos, a diferentes colaboradores da organização, consoante as suas funções na mesma, desde o *developer* até à própria gestão de topo. Estas métricas podem ou não ter pontos de contacto e devem estar alinhadas com os objectivos de negócio da empresa.

Capítulo 7

Conclusões e trabalho futuro

O trabalho desenvolvido ao longo do ano lectivo resultou num produto que acrescentará muito valor à empresa. Isto deve-se ao facto de esta plataforma permitir agregar e analisar os dados gerados pelos vários processos da empresa, o que facilitará muito o trabalho dos seus gestores e, consequentemente, dos restantes colaboradores. Para além disso, como foi sugerido durante o relatório, esta plataforma pode ser aproveitada por outras empresas, para que estas possam desenvolver as suas próprias aplicações usando apenas alguns módulos específicos, dada a grande flexibilidade da mesma.

A plataforma criada permite recolher dados de várias fontes diferentes, bem como fazer a gestão dos dados criados pela utilização dos vários módulos, nomeadamente a nível de gestão de Recursos Humanos. Permite também disponibilizar estes dados de forma a que estes possam ser consumidos por outras aplicações de forma fácil e estruturada, não estando, no entanto, dependente destas mesmas aplicações. Estes aspectos, juntamente com o facto de os módulos implementados serem bastante independentes entre si, fazem com que o desenvolvimento futuro fique facilitado.

De facto, a forma como esta plataforma foi estruturada faz com que a empresa possa continuar a acrescentar funcionalidades que melhorem esta plataforma e a tornem ainda mais relevante para a gestão e para os processos de tomada de decisão, sem que isso implique a modificação do que foi feito até aqui.

Para além disto, foi feito um trabalho importante ao nível da especificação da *framework* de métricas. A flexibilidade da solução de métricas implementada permite já que haja um maior controlo a nível da performance de cada colaborador, o que leva a que seja mais fácil implementar objectivos mais concretos e, consequentemente, mais facilmente mensuráveis.

7.1 Contribuições

Como foi referido no capítulo 3, o estagiário foi responsável por todo o desenvolvimento da API. Isto implicou não só a implementação dos módulos, mas também o próprio desenho da sua arquitectura. Assim, coube-lhe a responsabilidade de conceber todos os módulos, a partir dos modelos utilizando o ORM do Django, sem nunca descurar o funcionamento da base de dados que lhe dá suporte. Foram ainda implementadas todas as funcionalidades mostradas no capítulo 5. O estagiário esteve também envolvido em algumas decisões importantes relativas à engenharia do sistema. Um exemplo prende-se com a arquitectura de sincronização entre o UIS e as várias fontes de dados. Finalmente, e embora o desenvolvimento do *frontend* não tenha ficado a seu cargo, o estagiário acabou por assegurar o funcionamento das páginas de *admin* do Django, que permitem ver e editar muita da informação do sistema através de uma interface visual.

Para além disso, houve ainda lugar à produção de toda a documentação necessária, bem como à produção de exemplos de utilização da API.

7.2 Balanço

O estágio foi uma excelente forma de aprendizagem, não só em termos de Engenharia Informática, mas também em termos de trabalho em equipa num projecto prático, utilizando metodologias ágeis, a ser utilizado imediatamente pela empresa "no mundo real".

Houve também alguns aspectos que não correram tão bem como poderiam, o que levou a que tenham havido algumas funcionalidades e módulos que ficaram incompletos, nomeadamente a nível dos custos e execução dos projectos. Isto deveu-se, sobretudo, à primeira versão da funcionalidade ter sido especificada à imagem do que a organização faz actualmente. No entanto, a implementação, experimentação e testes destas funcionalidades, com o contributo directo do estagiário, levou a que se tivesse concluído que a organização beneficiaria, num futuro próximo, de uma metodologia de controlo de custos e execução de projectos mais flexível, no sentido da melhor adaptação a diferentes tipologias de projecto. Também houve um certo atraso no desenvolvimento de algumas funcionalidades devido, em parte, a um atraso no projecto devido à introdução tardia do elemento para desenvolver o *frontend* na equipa, e ao tempo que levou até serem desenvolvidos os primeiros *mockups* do mesmo.

No entanto, no final, os principais objectivos do trabalho foram atingidos, tendo sido desenvolvidas e testadas a esmagadora maioria das funcionalida-

des inicialmente propostas, que fazem com que o sistema tenha o impacto que era esperado na empresa. Também é importante referir novamente que a plataforma já está em utilização na Ubiwhere, ainda que algumas das funcionalidades já presentes na API não tenham sido ainda desenvolvidas no *frontend*. Está também previsto que, em empresas parceiras, já em 2014, seja utilizado o UIS, dado o elevado interesse que a ferramenta suscitou, sobretudo devido à facilidade de integração com as ferramentas de gestão de projecto já existentes.

7.3 Trabalho Futuro

De acordo com o que foi explicado no início do capítulo, a plataforma criada foi desenvolvida de modo a permitir o acrescento de novas funcionalidades. Assim, esta é a parte mais óbvia do trabalho futuro: de facto, para além dos pontos que foram sugeridos pelo estagiário na tabela 4.11, podem ainda ser desenvolvidos módulos para que sirvam para fazer a integração de novos serviços (Jenkins, sistemas de gestão de documentos na *cloud*, etc.), gestão de relacionamentos (com clientes e parceiros), gestão de oportunidades de negócio, gestão de processos ou gestão de portfólio para dar suporte às actividades dos departamentos de suporte da empresa (*marketing*, vendas, *design*, etc.).

Para além disto, e de acordo com o que foi discutido no final do capítulo 6, pode ainda haver lugar a um melhoramento da *framework* de métricas, nomeadamente através do desenvolvimento das funcionalidades que lhe permitam ser ainda mais flexível ao nível da definição das métricas e objectivos. A implementação das funcionalidades que permitam dar resposta às *user stories* definidas na tabela 6.1 será outro ponto importante para o trabalho futuro a desenvolver. Desta forma, este módulo pode tornar-se ainda mais útil dentro da empresa, para além de poder constituir uma solução a ser comercializada pela empresa.

Bibliografia

[1] L. Mieritz, "Survey Shows Why Projects Fail", Gartner, rel. téc. Junho, 2012, pp. 1–9.

[2] M. Paasivaara e S. Durasiewicz, "Using Scrum in a Globally Distributed Project : A Case Study", *Software Process Improvement and Practice*, vol. 13, pp. 527–544, 2008.

[3] V. Mahnic e S. Drnovscek, "Agile Software Project Management with Scrum", em *EUNIS 2005 Conference-Session papers and tutorial abstracts*, 2005, p. 6.

[4] M. Cristal, D. Wildt e R. Prikladnicki, "Usage of SCRUM practices within a global company", em *Proceedings - 2008 3rd IEEE International Conference Global Software Engineering, ICGSE 2008*, 2008, pp. 222–226.

[5] E. Hossain, M. A. Babar e H.-y. Paik, "Using Scrum in Global Software Development: A Systematic Literature Review", *2009 Fourth IEEE International Conference on Global Software Engineering*, pp. 175–184, jul. de 2009.

[6] H.-c. Young, T.-H. Fang e C.-H. Hu, "A successful practice of applying software tools to CMMI process improvement", *Journal of Software Engineering Studies*, vol. 1, nº 2, pp. 78–95, 2006.

[7] M. Chrissis, M. Konrad e S. Shrum, *CMMI for Development*. 2011, p. 687.

[8] M. Bandor, "Process and Procedure", 2007.

[9] M. Chrissis, M. Konrad e S. Shrum, *CMMI Guidlines for Process Integration and Product Improvement, Part One*. 2003.

[10] M. E. Fayad e M. Laitinen, "Process Assessment Considered Wasteful", vol. 40, nº 11, pp. 125–128, 1997.

[11] *Scrum Methodology*, endereço: `www.scrummethodology.com`, 2008.

[12] H. Glazer, D. Anderson, D. J. Anderson, M. Konrad e S. Shrum, "CMMI ® or Agile : Why Not Embrace Both !", nº November, 2008.

[13] J. Sutherland, C. R. Jakobsen e K. Johnson, "Scrum and CMMI Level 5: The Magic Potion for Code Warriors", *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, pp. 1–10, 2008.

[14] A. Rodriguez, "Restful web services: The basics", *Online article in IBM DeveloperWorks Technical Library*, pp. 1–11, 2008.

[15] L. Columbus, *2013 ERP Market Share Update: SAP Solidifies Market Leadership - Forbes*, endereço: `http://www.forbes.com/sites/louiscolumbus/2013/05/12/2013-erp-market-share-update-sap-solidifies-market-leadership/`, 2013.

[16] D. Kaye, *Loosely Coupled: The Missing Pieces of Web Services*. RDS Strategies LLC, 2003, p. 334.

[17] J. Deacon, "Model-view-controller (mvc) architecture", *... de 2006.] http://www. jdl. co. uk/briefings/MVC. pdf*, pp. 1–6, 2009.

[18] *Django Project Documents*, endereço: `https://docs.djangoproject.com/en/1.5/faq/general`, 2013.

[19] B. Askins e A. Gree, "A Rails/Django Comparison", *Open Source Developers' Conference*, vol. 3, 2006.

[20] D. Greenfeld, *Choosing an API framework for Django*, endereço: `http://pydanny.com/choosing-an-api-framework-for-django.html`, 2012.

[21] A. Holovaty e J. Kaplan-Moss, *The Definitive Guide to Django: Web Development Done Right*. 2009, p. 499.

[22] I. V. López e G. B. Gutiérrez, "El Benchmark TPC-H en MySQL y PostgreSQL", *ingenieria.lm.uasnet.mx*, 2009.