MESTRADO EM ENGENHARIA INFORMÁTICA

ESTÁGIO / DISSERTAÇÃO

RELATÓRIO FINAL

DESENVOLVIMENTO DO MÓDULO DE GESTÃO DE PROJETOS NO SISTEMA DE INFORMAÇÃO DA UBIWHERE

ALUNA

Joana Cardoso Belém

ORIENTADORES

Professor Doutor Nuno Laranjeiro, DEI-FCTUC

Mestre Ricardo Machado, Ubiwhere

FACULDADE DE CIÊNCIAS E TECNOLOGIAS

DA UNIVERSIDADE DE COIMBRA

ANO LETIVO 2014/2015

MESTRADO EM ENGENHARIA INFORMÁTICA
ESTÁGIO / DISSERTAÇÃO
RELATÓRIO FINAL

Desenvolvimento do módulo de gestão de projetos no Sistema de Informação da Ubiwhere

ALUNA
Joana Cardoso Belém

ORIENTADORES:
Professor Doutor Nuno Laranjeiro, DEI-FCTUC
Mestre Ricardo Machado, Ubiwhere

JÚRI:
Arguente: Professor Doutor Carlos Bento
Vogal: Professor Doutor António Dourado Pereira Correia

FACULDADE DE CIÊNCIAS E TECNOLOGIA
DA UNIVERSIDADE DE COIMBRA

ANO LETIVO 2014/2015

Resumo

Hoje em dia, os principais fatores de insucesso de um projeto encontram-se associados à má gestão dos mesmos. A Ubiwhere, querendo minimizar a probabilidade de insucesso dos seus projetos optou por seguir as abordagens Capability Maturity Model Integration (CMMI) e Scrum. Acontece que estas abordagens assentam em muitos processos repetitivos que necessitam de mais tempo que os Recursos Humanos (RH) podem disponibilizar. Exemplos desses processos são, numa fase em que é feito o planeamento do projeto, a obtenção de estimativas para o que vai ser desenvolvido, e numa fase onde é feita a revisão do projeto, a criação de Snapshots que contém toda a informação relacionada com o mesmo, desde o estado estimado e real ao desvio de custos do mesmo no momento da sua criação.

Pensando na automação dos processos seguidos assim como na centralização das várias ferramentas utilizadas pela mesma, a Ubiwhere criou um Sistema de Informação de raiz, o Ubiwhere Information System (UIS). Este sistema já se encontra em utilização por todos os RH da empresa e contém, principalmente, funcionalidades relativas à gestão dos mesmos. No que toca à gestão de projetos este ainda não oferece grande suporte, surgindo assim a necessidade da adaptação do mesmo ao departamento de desenvolvimento.

Este estágio visa a implementação de funcionalidades, a integrar no UIS, que permitam a recolha, tratamento e análise de dados relacionados com a gestão de projetos e que ajudem na gestão corrente das equipas, como por exemplo o cálculo das diferentes métricas definidas pela empresa.

No final do estágio pretende-se que o sistema dê suporte não só à gestão de RH mas também aos processos essenciais da gestão de projetos.

Palavras Chave: Sistema de Informação, Gestão de Projetos, CMMI, Scrum

Abstract

Nowadays, the main reasons for failure of a project are associated with bad management of that same project . Ubiwhere opted to follow the CMMI and Scrum approaches, to minimize the probability of insucess of its projects. It just so happens that these approaches are based in many repetitve processes that need more time that the human resources can provide. Some examples of those processes are, on a phase where the project planning is done, the achievement of estimates for what is going to be developed, and on a phase where is done the revision of the project, the creation of Snapshots that have all the information related to the project, from the expected and real state to the cust deviation on the moment of its creation.

Thinking on the automation of the projects followed and on the centralization of the various tools used, Ubiwhere created one SI from the ground up, the UIS. This system is already being used by every RH and contains, mainly, features related to the management of the same. As regards to the projects management, it still doesn't give many support, having the need to adapt to the same development department.

This internship aims the implementation of features, to integrate on UIS, that allow the collection, processing and analysis of data related to the projects management and help in current management of the teams, e.g. the calculation of different metrics defined by the company.

At the end of the internship it is intended that the system supports, not only the RH management but also the essential processess of the projects management.

Keywords:
Information Systems, Project Management, CMMI, Scrum

Agradecimentos

À minha família por me aturar e fazer acreditar nos momentos mais difíceis, não sendo possível atingir esta etapa sem o apoio dos mesmos.

Ao professor Nuno Laranjeiro por toda a disponibilidade e paciência demonstrada, encontrando-se sempre pronto para responder a qualquer dúvida existente e fazer a revisão do relatório.

À Ubiwhere, pela oportunidade dada. Agradeço ao Ricardo Machado pelo acompanhamento demonstrado e à equipa presente em Coimbra pela boa disposição e inter-ajuda sempre demonstrada. Em especial, obrigada ao Francisco Monsanto, ao Nuno Khan, ao Luís Almeida e ao André Duarte por se encontram disponíveis a ajudar sempre que necessário. Não esquecer também, um obrigado ao Nuno Costa pelo acompanhamento do projeto prestado no primeiro semestre.

Por fim, a todos os meus amigos, um muito obrigado pela amizade demonstrada, pois ajudou a tornar esta caminhada mais fácil.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

API Application Programming Interface

BD Base de Dados

CM Configuration Management

CMMI Capability Maturity Model Integration

CMMI-DEV CMMI for Development

CMMI-ACQ CMMI for Acquisition

CMMI-SVC CMMI for Services

CSS Cascading Style Sheets

CSV Comma Separated Value

HTML HyperText Markup Language

HTTPS Hyper Text Transfer Protocol Secure

JSON JavaScript Object Notation

LDAP Lightweight Directory Access Protocol

MA Measurement and Analysis

ORM Object-Relational Mapping

PM Project Manager

PMC Project Monitoring and Control

PP Project Planning

PPQA Process and Product Quality Assurances

RH Recursos Humanos

REST REpresentational State Transfer

REQM Requirements Management

SI Sistema de Informação

SaaS Software as a Service

SAM Supplier Agreement Management

SEI Software Engineering Institute

SGBD Sistema de Gestão de Base de Dados

TC Test Case

TI Tecnologias da Informação

TS Test Suite

TSV Tab Separated Values

UIS Ubiwhere Information System

URM Ubiwhere Relationship Manager

US User Stories

XML eXtensible Markup Language

XPATH XML Path Language

Capítulo 1

Introdução

O presente relatório expõe todo o trabalho desenvolvido ao longo do ano letivo de 2014/2015 no âmbito da cadeira Estágio/Dissertação do Mestrado de Engenharia Informática de Coimbra. O estágio insere-se na área de Sistemas de Informação e teve lugar na Ubiwhere, uma empresa que se foca no desenvolvimento de soluções inteligentes e tecnologias de ponta sendo atualmente uma das mais prestigiadas empresas jovens, não só a nível nacional como internacional, nas áreas Cidades Inteligentes e Internet do Futuro. Sediada em Aveiro e com um escritório no IPN em Coimbra, onde decorreu o estágio.

Por parte da empresa, o estagiário teve como orientador, no 1º Semestre, o Mestre Nuno Costa e no 2º Semestre o Mestre Ricardo Machado e esteve inserido numa equipa com mais 2 elementos, um *designer*, que ficou a cargo de criar os protótipos do trabalho desenvolvido e um Project Manager (PM) que geriu o trabalho desenvolvido pelo estagiário. Por parte da Universidade, o estagiário teve como orientador o professor Doutor Nuno Laranjeiro.

1.1 Contexto e Motivação

O relatório Chaos Manifesto 2013, realizado pelo The Standish Group, mostra que o número de projetos de *software* que são entregues com sucesso tem vindo a aumentar ao longo dos tempos [1]. Esse aumento deve-se a diversos fatores, destacando-se a monitorização dos processos do projeto, *skills*, custos, ferramentas utilizadas, decisões tomadas, entre outros. Na tabela 1.1 encontram-se os resultados do estudo realizado entre os anos 2004 e 2012. Por *Successful* entendem-se os projetos que foram entregues a tempo e que cumprem o orçamento estipulado. Os *Failed* correspondem aos projetos que fracassaram, não tendo sido entregues sequer. Por último, os projetos que se encaixam dentro do campo *Challenged* correspondem aos projetos que foram entregues mas com tempo ou custos diferentes do estimado.

	2004	2006	2008	2010	2012
Successful	29%	35%	32%	37%	39%
Failed	18%	19%	24%	21%	18%
Challenged	53%	46%	44%	42%	43%

Tabela 1.1: Resultado do projeto CHAOS entre os anos 2004 e 2012

Apesar do aumento de projetos terminados com sucesso, ainda existe uma grande percentagem de projetos que falham ou que são entregues mas com custos e tempos fora do estipulado. Segundo o estudo realizado por Lars Mieritz, analista na empresa Gartner [2], as principais causas do fracasso dos projetos são devido a:

- Funcionalidades incompletas ou que não estão de acordo com o estipulado

- Incumprimento dos prazos definidos
- Falta de qualidade dos produtos
- Variação de custos acordados inicialmente
- Cancelamento inesperado do projeto

Estas causas encontram-se normalmente associadas ao mau planeamento e controlo do projeto. De forma a minimizar a má gestão dos projetos, existem diversas práticas e políticas que podem e devem ser seguidas. Exemplos dessas práticas são as definidas pelo CMMI e pela metodologia ágil Scrum, ambas seguidas pela empresa.

O CMMI encontra-se dividido em diferentes nível, sendo a Ubiwhere reconhecida como a empresa mais jovem em Portugal a obter a certificação L2 do mesmo. Neste nível, os processos prometem uma melhoria relativamente a todas as fases da gestão de projetos. Já o Scrum consiste numa metodologia de desenvolvimento ágil que se encontra dividida em diversos momentos cíclicos e repetitivos (*Sprints*) que englobam todas as fases de um projeto (planeamento, revisão e retrospetiva do que foi feito). Em cada um desses momentos existem várias práticas que devem ser seguidas que respeitam diretamente algumas das necessidades do CMMI.

Apesar destas práticas aumentarem a probabilidade de um projeto ser bem sucedido, estas encontram-se associadas a um conjunto de processos repetitivos que necessitam de muita atenção e dedicação por parte dos RH da empresa, nomeadamente dos PM. Assim sendo, a utilização de ferramentas que permitam a automação dos processos da empresa torna-se essencial pois possibilita que a mesma não só consiga cumprir os processos definidos como os possa cumprir num curto espaço de tempo, diminuindo assim os custos associados aos mesmos.

Para a gestão dos seus projetos a Ubiwhere utiliza a ferramenta Redmine, contudo esta ferramenta não oferece suporte para todas as suas necessidades, nomeadamente na obtenção de informação no que toca à análise e à gestão ágil dos projetos. O fato da Ubiwhere seguir processos muito específicos faz com que seja complicado encontrar uma ferramenta que suporte todas as necessidades da mesma.

Assim, com o intuito de automatizar e dar suporte aos processos da empresa, a Ubiwhere optou pela criação de um Sistema de Informação de raiz, o UIS. Este sistema segue uma abordagem "3-Tier", sendo constituído por uma API RESTful que é disponibilizada através de um servidor *web*, cliente *web* e eventualmente aplicações móveis. Com esta ferramenta a Ubiwhere pretende a centralização das várias ferramentas utilizadas pela empresa aglomerando informação relacionada com os RH assim como relativa à camada de negócio da mesma. Esta ferramenta não serve de substituição do Redmine, que continua com informação operacional relativa aos projetos (tarefas, requisitos, bugs, entre outros) mas sim como complemento, ao ser possível a extração de dados presentes no mesmo para que possa ser feita a sua análise e tratamento, por exemplo, na forma de diferentes métricas que a empresa necessita conhecer. A recolha dessas métricas é bastante importante para a empresa de forma a conduzir e informar os gestores sobre as decisões operacionais que se têm de tomar diariamente.

Atualmente, o UIS já se encontra em utilização por todos os utilizadores da empresa, sendo nela que os mesmos fazem o CheckIn, marcam e visualizam as suas faltas e férias. A nível da camada de negócio, é possível verificar a lista de projetos existentes na empresa e alguma informação básica sobre os mesmos, como por exemplo, PM, data de início e fim e recursos associados a cada um (informação obtida diretamente do Redmine). Além disso, já se encontram calculados os custos externos associados a cada projeto e algumas das métricas definidas pela empresa, sendo necessária uma análise das mesmas para fazer a sua validação

e posteriormente correção assim como implementação das ainda não calculadas. Apesar dessas funcionalidades, o sistema ainda não oferece grande suporte relativamente à gestão de projetos. É necessário perceber quais os processos a que a empresa dá resposta, em cada um das fases do projeto, e quais deles podem ser automatizados com foco à integração dos artefatos que são utilizados regularmente pelas equipas geridas através do Scrum. Exemplos de funcionalidades necessárias são:

- Numa fase de planeamento do projeto, a automação do processo de criação de estimativas.
- Numa fase de revisão do projeto, a obtenção de informação relacionada com o mesmo. Por exemplo, a obtenção dos desvios face a estimativa calculada, em diversos campos, tais como estado e custos do mesmo. Além disso, a extração dessa mesma informação de maneira a que possa ser analisada e tratada posteriormente.

1.2 Objetivos

O principal objetivo do estágio consistiu na automação de processos seguidos pela empresa, com especial foco na gestão de projetos. Este trabalho pretendeu facilitar o papel dos PM e dos gestores de topo reunindo a informação exigida pelo modelo de maturação CMMI nos momentos contemplados pela metodologia Scrum.

O trabalho realizado teve como objetivos:

- Integrar o UIS com o Redmine e o Google Spreadsheets, sistemas onde atualmente é armazenada grande parte da informação relacionada com os projetos da empresa.

  Com esta integração pretende-se não só a extração de informação presente nesses sistemas para que possa ser analisada e devidamente tratada, mas também a adição de informação já processada como por exemplo a informação relativa às métricas calculadas pelo UIS, de forma a guardar toda a informação no Redmine.

- Permitir a visualização fácil e percetível do estado de um *Sprint*, através da utilização de ferramentas utilizadas no processo Scrum.

- Automatizar fase de planeamento do projeto,

  - Obter informação necessária para a realização da reunião de planeamento
  - Automatizar o processo de criação de estimativas, ao fazer a análise e extração de informação proveniente de SpreadSheets para o Redmine.
  - Recolher informação que permita o cálculo da disponibilidade dos RH da empresa para que possa ser feita a sua alocação a projetos. Esta disponibilidade é obtida através da informação das alocações existente no UIS.

- Automatizar à fase de revisão do projeto,

  - Recolher informação que permita a obtenção informação necessária para a realização da reunião de revisão, nomeadamente:
    - Cálculo de métricas ainda não implementadas assim como correção das existentes.
    - Cálculo dos custos do projeto, nomeadamente no que toca a custos relacionados com os RH. Através das informações presentes no Redmine, como a percentagem de realização do projeto e através do custo associado a cada RH.
  - Garantir que não há perdas de informação relevante dos projetos da empresa ao criar um mecanismo que mantenha o seu histórico, visualizando-a sempre que necessário.

Coube ao estagiário, a implementação das funcionalidades no servidor e respetiva API e no cliente *web* utilizado, sendo o principal foco relativo a tarefas de *backend*. O design da aplicação *web* ficou do âmbito do estágio.

1.3 Estrutura

O presente trabalho encontra-se dividido em diferentes capítulos, encontrando-se de seguida uma breve descrição de cada um.

Capítulo 2: Estado da Arte Neste capítulo é feito o contexto do projeto e o estudo do que existe no mercado a nível de ferramentas que apoiam as equipas que utilizam a metodologia Scrum.

Capítulo 3: Planeamento e Metodologia Neste capítulo é apresentado o planeamento e a metodologia de todo o trabalho desenvolvido no âmbito do projeto. Por fim é exposto o estudo realizado relativo aos riscos associados ao mesmo.

Capítulo 4: Especificação Técnica Este capítulo encontra-se divido em 2 partes, uma referente aos requisitos do projeto e outra à arquitetura da solução.

Capítulo 5: Implementação Neste capítulo encontra-se uma apresentação do que foi implementado no âmbito do projeto.

Capítulo 6: Produto Final Neste capítulo encontram-se alguns dos *layouts* do cliente *Web* nos quais são apresentadas algumas das informações calculadas apresentadas na secção Implementação.

Capítulo 7: Testes No capítulo Testes são enunciados os diversos testes realizados para garantir a validação do sistema.

Capítulo 8: Conclusões e Trabalho futuro Neste capítulo é feita uma breve conclusão e apresentação do trabalho futuro.

Bibliografia É descrita a bibliografia utilizada para a criação do relatório.

Anexos Secção onde são apresentados todos os anexos do projeto.

Capítulo 2

Estado da Arte

Neste capítulo será feita uma análise do que existe no mercado a nível de ferramentas que apoiam as equipas que utilizam a metodologia Scrum. Visto que a Ubiwhere além da metodologia Scrum também segue as práticas do modelo CMMI, é necessário em primeiro lugar explicar em que consistem estas duas abordagens assim como perceber como se relacionam e o que têm em comum. Assim, inicialmente será descrito o contexto do projeto onde os capítulos principais são relativos ao Scrum, CMMI e ao mapeamento entre eles.

Estando o contexto do projeto descrito, serão apresentadas alguns das ferramentas existentes no mercado para esse efeito assim como será feita uma comparação entre elas tendo em conta as funcionalidades que será necessário desenvolver no âmbito do projeto.

Como explicado anteriormente, para a realização do projeto é necessário extrair informação de outros sistemas, nomeadamente do Redmine. Apesar deste fornecer uma API para o efeito, esta não permite a extração de alguns informações, sendo feito um estudo de possíveis ferramentas que possam ser utilizadas para a extração desse tipo de informação. Após a apresentação dessas ferramentas é feita uma análise das vantagens e desvantagens de cada assim como uma comparação a nível de performance.

2.1 Contexto

2.1.1 Scrum

A palavra Scrum teve "origem no desporto coletivo *rugby*, que representa a colocação da bola em jogo e onde toda a equipa reúne esforços para conquistar o objetivo: a posse de bola" [3]. Atualmente, segundo o relatório "The State of Scrum: Benchrks Guidelines"realizado em 2013 [4], é a metodologia ágil mais utilizada, representando 40% do total enquanto apenas 12% é referente a metodologias tradicionais.

A metodologia Scrum surge como solução para a constante mudança das necessidades dos clientes antecipando possíveis problemas no decorrer do projeto e tem como palavras-chave: **adaptação**, **inspeção** e **transparência**. O *Sprint* é o coração deste método girando todas as etapas do Scrum em torno do mesmo. Por *Sprint* considera-se um determinado período de tempo, no máximo um mês, no qual a equipa de desenvolvimento se compromete a realizar determinadas tarefas obtendo no final uma parte funcional do projeto. Um projeto é constituído por vários *Sprints*.

Na figura C.8 encontra-se representado o processo Scrum.

A apresentação do Scrum será feita dividindo o processo em 3 categorias:

- Papéis - Representam os diferentes cargos existentes no processo Scrum.
- Eventos - Correspondem às reuniões de acompanhamento do *Sprint*.

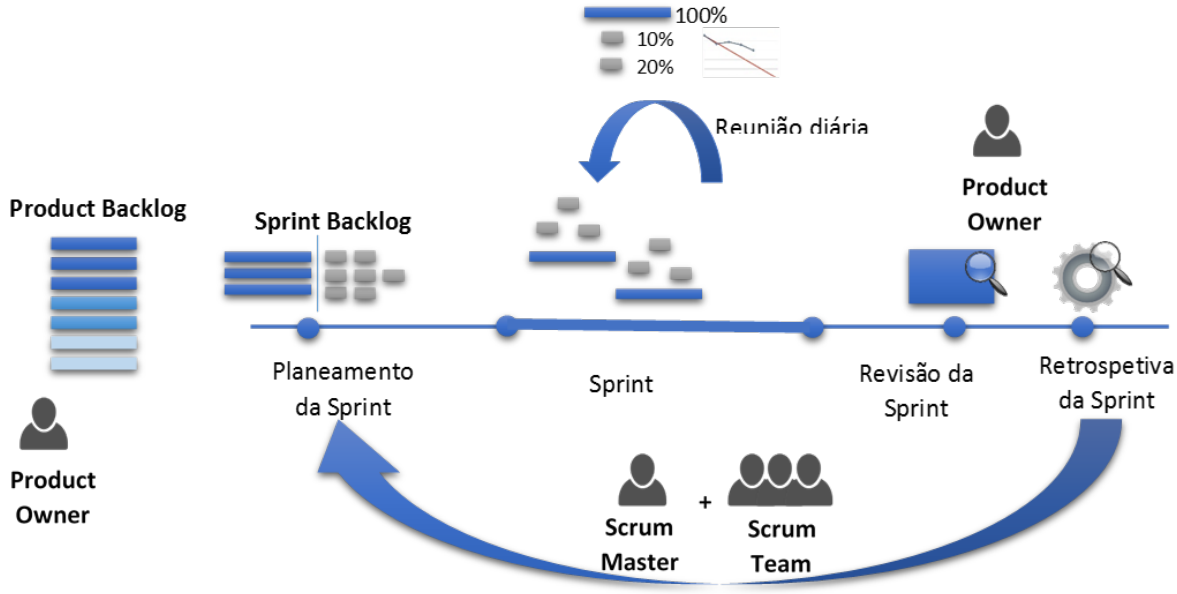


Figura 2.1: Processo Scrum

- Artefatos - Representam os documentos necessários para acompanhar todo o processo Scrum.

2.1.1.1 Papéis

Existem certos cargos, cada um com as suas respetivas funções, que devem ser respeitados para colocar em prática a metodologia Scrum. Na figura C.8 encontram-se apresentados os diferentes cargos existentes e é feito o mapeamento desses cargos com os diferentes momentos existentes no processo Scrum.

- Product Owner – tem obrigação de comunicar a visão do produto para o resto da equipa representando os interesses do cliente através da escolha e priorização dos requisitos que satisfazem as necessidades do cliente. É o responsável pelo produto pois é o único que tem autoridade para agir e tomar decisões relacionadas a este.

- Scrum Master - responsável pelo processo Scrum, ficando com o cargo de ensinar e garantir que toda a equipa segue as regras definidas. Tem a obrigação de proteger a equipa contra metas não exequíveis e de resolver qualquer obstáculo que surja durante o processo. Na Ubiwhere quem fica encarregue por cumprir estas necessidades é o PM de cada Projeto.

- Scrum Team – representa toda a equipa de desenvolvimento e teste do projeto. São responsáveis por gerar incrementos de produto bem estruturados e testados a cada *Sprint*. A equipa deve ser constituída por um número reduzido de elementos, entre 6 a 10, sendo estes auto-organizáveis, auto-gerenciáveis e multi-funcionais.

2.1.1.2 Eventos

Cada *Sprint* deve incluir quatro tipos de eventos: uma reunião de planeamento, uma reunião diária durante a sua realização, uma reunião de revisão e uma reunião de retrospetiva. Após

a realização destas reuniões o ciclo do *Sprint* é novamente iniciado, repetindo-se até que o projeto termine.

- Reunião de Planeamento do *Sprint* - Como se pode verificar na figura C.8, corresponde à reunião realizada antes de começar o *Sprint*. Nesta reunião começa-se por decidir que requisitos serão realizados durante o *Sprint*. Após a seleção dos requisitos é feita a sua divisão em pequenas tarefas sendo atribuída a cada uma delas uma estimativa do tempo que demora a ser realizada.

- Reunião Diária – Pequena reunião diária (máximo 15 minutos), durante o decorrer do *Sprint*, onde o Scrum Team apresenta o que foi e o que será feito ao longo do dia ao Scrum Master. Qualquer impedimento para a realização das tarefas será comunicado durante a reunião, ficando a cargo do Scrum Master a sua resolução o mais rapidamente possível.

- Reunião de Revisão do *Sprint* – Reunião realizada após a data limite do *Sprint* onde é feita a revisão e monitorização do *Sprint*/Projeto. É nesta reunião que é apresentado o que foi desenvolvido ao Product Owner, ficando a cargo deste a validação do que foi e o que não foi concluído.

- Reunião de Retrospetiva do *Sprint* – Reunião realizada após a reunião de revisão e antes de uma nova reunião de planeamento. Nesta reunião é feita uma perspetiva do que correu bem, o que correu menos bem e o que fariam diferente ajudando na eficiência e eficácia dos próximos *Sprints*.

2.1.1.3 Artefatos

São vários os artefatos característicos da metodologia Scrum, englobando todas as fases presentes do desenvolvimento.

- Product Backlog – lista dinâmica criada pelo *Product Owner* que representa o conjunto de todos os requisitos funcionais e não funcionais do projeto descritos como *User Stories*. Por *User Story* considera-se uma simples descrição de uma funcionalidade que é requerida pelo utilizador. Em anexo encontra-se um maior detalhe relativo à definição de *User Stories*. Na Ubiwhere, cada *User Story* tem associado uma medida de complexidade (*Story Point*) que é atribuída utilizando normalmente a Técnica de *Planning Poker*, técnica descrita no respetivo capítulo.

- Sprint Backlog – conjunto de tarefas definidas na reunião de planeamento do *Sprint* e com as quais o *Scrum Team* se compromete a desenvolver durante o *Sprint*.

- Gráficos de Monitorização - gráfico que permite uma análise rápida e eficaz do estado do projeto sendo o mais utilizado o *Burndown Chart*. Neste gráfico é apresentada a correlação entre o trabalho realizado até ao momento e o trabalho que ainda falta realizar até terminar o projeto. Existem diferentes variantes deste gráfico sendo utilizado como controlo durante os *Sprints* assim como durante todo o projeto.

- *Taskboard* - Quadro que contém informação sobre as tarefas a realizar durante o *Sprint*. Encontra-se dividido por diferentes colunas (*New*, *In Progress* e *Done*). A sua utilização permite uma fácil gestão das tarefas de cada *Sprint* oferecendo visibilidade e transparência no processo de desenvolvimento.

2.1.1.4 Técnica Planning Poker

Planning Poker consiste numa técnica de cálculo de estimativas utilizada para estimar a complexidade de *User Stories*. O seu principal objetivo é a obtenção de estimativas mais precisas pois conta com a opinião de toda a equipa de desenvolvimento. Normalmente, ao invés de se atribuírem unidades de tempo são atribuídos valores que representam a complexidade de uma dada *User Story* (*Story Points*). Estes valores permitem uma maior perceção do esforço necessário e por sua vez do tempo necessário para a sua realização. Podem utilizar-se várias representações da complexidade, sendo que uma das mais utilizadas é a sequência de Fibonacci.

Nesta técnica começa-se por escolher uma ou mais *User Stories* de referência que vão servir de comparação para as *User Stories* a estimar. Tendo como base as *User Stories* escolhidas , cada elemento da equipa, que vai estar presente no desenvolvimento do projeto, seleciona o valor de complexidade que considera mais apropriado. Após é feita a análise dos valores apresentados verificando se existem diferenças significativas. Caso existam procede-se à explicação de tais valores de modo a chegar-se a um consenso e é selecionada a estimativa que pensam ser mais correta.

A utilização desta técnica é bastante importante para a empresa, contudo o processo seguido atualmente não é o mais eficaz. Na Ubiwhere existe um *template* em formato Excel onde o PM coloca as *User Stories* referentes a um *Sprint*. Após, cada elemento do projeto coloca, para cada *User Story*, a estimativa que considera mais apropriada, cabendo ao PM fazer a recolha e análise das estimativas escolhidas pelos utilizadores. Após discussão, caso não se chegue a nenhum consenso, fica a seu cargo a decisão da estimativa que considera mais correta, atualizando a informação das *User Stories* no Redmine. No âmbito do projeto pretende-se a automação deste processo de forma a facilitá-lo e diminuir o tempo despendido nele.

2.1.2 CMMI

O CMMI [5] consiste num modelo desenvolvido pelo Software Engineering Institute (SEI) da Universidade Carnegie Mellon que contém um conjunto de práticas pelas quais uma empresa se deve guiar para melhorar os seus processos e por conseguinte melhorar a sua performance e qualidade de produtos. Neste modelo não se encontram regras de como dar resposta a esses processos mas sim indicações do que se deve fazer.

A versão CMMI 1.3 é a mais recente versão do CMMI e encontra-se dividida em três modelos:

- CMMI-DEV, modelo focado para o desenvolvimento de produtos e serviços e que é seguido pela Ubiwhere.
- CMMI-ACQ, modelo focado para a aquisição e terceirização de bens e serviços.
- CMMI-SVC, modelo focado para empresas prestadoras de serviços.

O CMMI pode seguir 2 tipos de representação, contínua em que diferentes processos atingem diferentes níveis, ou por estágios (representação seguida pela Ubiwhere) em que a maturidade é medida por um conjunto de processos. Tanto numa como noutra representação existem 5 níveis de maturidade.

Resumidamente, no nível 1 não existe definição de processos, o principal objetivo consiste na criação do produto final não sendo estruturados processos para o seu desenvolvimento.

No nível 2 pode afirmar-se que já existe a definição/estruturação de alguns processos, sendo o principal foco na área de gestão e alteração do projeto. Neste nível já existe um plano

de trabalho, a monitorização e controlo desse trabalho garantindo que a entrega do produto ocorre como planeado. Como enunciado na secção Introdução, a Ubiwhere encontra-se no nível 2, sendo a passagem para o nível 3 um dos seus objetivos a curto prazo.

No nível 3 o foco é relativo à estandardização dos processos existentes. Com este nível pretende-se a consistência das práticas implementadas, sendo adotados *standards* por toda a organização. Além disso, existe um maior rigor na descrição dos processos relativamente ao nível 2.

Por último, no nível 4 e nível 5 utilizam a estatística para prever e melhorar a performance da organização.

Cada nível do modelo contém diferentes áreas de processo, sendo no total constituído por 22. Pode dizer-se que uma área de processo consiste num conjunto de práticas relacionadas com uma certa área que, se aplicadas conjuntamente, conseguem melhorar essa determinada área. Uma vez que a Ubiwhere se encontra classificada com **CMMI-DEV nível 2**, será dada especial atenção a este nível. Neste nível existem 7 áreas de processo, cada uma delas com objetivos diferentes:

- Gestão Requisitos: Gerir requisitos e compromissos do projeto garantindo o alinhamento entre estes.
- Planeamento do Projeto: Estabelecer e manter planos que definam as atividades do projeto.
- Monitorização e Controlo do Projeto: Monitorizar o progresso do projeto de modo a tomar ações corretivas caso exista um desvio significativo relativo ao esperado.
- Medição e Análise: Medir e analisar dados que auxiliem as necessidades dos gestores de informação.
- Garantia da Qualidade do Processo e do Produto: Garantir a qualidade do produto.
- Gestão de Configurações: Estabelecer e manter a integridade dos produtos usados.
- Gestão de Acordo com o Fornecedor: Gerir a aquisição de produtos / serviços de fornecedores.

Em anexo encontram-se as práticas específicas que se devem dar suporte para atingir o nível 2 do CMMI. De todas estas áreas de processo, a única que não se aplica na Ubiwhere é a área Gestão de Acordo com o Fornecedor. Para cada uma das outras, será apresentada uma breve descrição do que é suposto a Ubiwhere fazer relativamente a cada uma.

A área **Gestão Requisitos** obriga a Ubiwhere a definir todo um conjunto de processos relativos aos requisitos dos seus projetos. Tudo o que envolve os requisitos do projeto é feito nesta área, desde o seu levantamento à sua possível alteração e adição de novos requisitos. Grande parte desta área pode ser feita utilizando o Scrum, encontrando-se na secção "mapeamento entre CMMI e Scrum"o resultado.

A área **Planeamento do Projeto**, como o seu nome indica está relacionada com o planeamento do projeto. Esta área obriga a tratar de todo o trabalho que antecede um projeto. A definição do ciclo de vida do projeto assim como a estimação do calendário e do orçamento do projeto é tido em conta nesta área. Para isso é necessário criar estimativas para todo o trabalho a realizar durante o projeto. Antes de iniciar o projeto é importante também começar por identificar os riscos e todos os recursos do projeto como por exemplo as métricas a que o projeto vai estar sujeito e toda a equipa que vai estar envolvida no projeto. Na secção "mapeamento entre CMMI e Scrum"encontra-se uma breve explicação sobre como o Scrum pode ajudar na resolução de alguns práticas presentes nesta área.

Na área de **Monitorização e Controlo do Projeto**, há a definição de todos os processos referentes ao controlo do projeto. Cabe ao PM verificar se o projeto se encontra de acordo com o planeado monitorizando os diferentes aspetos que envolvem o projeto. Devido ao CMMI, a Ubiwhere é obrigada a definir um conjunto de métricas que avaliam o estado do projeto. A monitorização destas métricas e do progresso do projeto permitem aos PM e aos gestores de topo analisar a *performance* das equipas. Outros aspetos que também precisam de ser monitorizados são os custos e o esforço necessário para a realização do projeto. Sempre que exista algum desvio ao esperado fica a cargo do gestor de projeto a sua correção. No contexto deste projeto, pretende-se a obtenção de todos os dados necessários que permitam a verificação e análise do projeto. Alguns dessas informações podem ser obtidas através da utilização dos artefatos presentes na metodologia Scrum, encontrando-se essa informação na secção referente ao mapeamento entre o CMMI e o Scrum.

A área **Medição e Análise** obriga à definição e criação de diferentes métricas que permitem fazer a análise de diferentes aspetos do projeto, como por exemplo do número de *bugs* do projeto, as horas necessárias para a sua resolução, entre outros. A Ubiwhere é obrigada a definir todo um conjunto de métricas que englobam diferentes níveis, desde o nível mais especifico que engloba apenas um *Sprint* ao nível mais geral que representa toda a cada de negócio. Estas métricas permitem uma análise do trabalho realizado encontrando-se já definidas e especificadas pela própria empresa. Na imagem 2.2 é possível observar as métricas definidas pela empresa. Atualmente já existe suporte automático para algumas destas métricas no UIS, contudo não se encontram devidamente testadas possibilitando a criação de métricas erradas o que é prejudicial para a empresa. Além disso, para calcular as métricas de um nível é necessária a informação das métricas do nível inferior. Mesmo que as métricas calculadas de raiz se encontrem bem calculadas, como dependem de outras métricas, podem originar informações erradas. Assim, no âmbito deste trabalho pretende-se a validação e melhoramento das métricas existentes assim como o cálculo das métricas que ainda não são suportadas pela empresa.

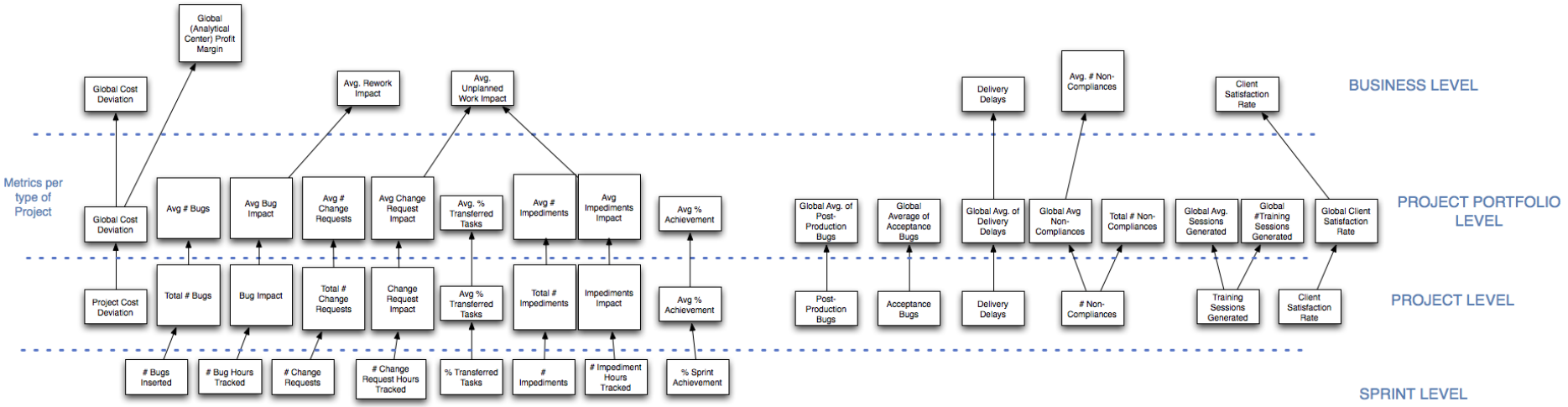


Figura 2.2: Métricas Ubiwhere

Na área **Garantia da Qualidade do Processo e do Produto** são apresentadas várias práticas que permitem uma análise detalhada do projeto garantindo assim a qualidade deste. Esta área envolve a avaliação de processos, produtos e serviços, documentação e apresentação de resultados. A definição deste processo deve ficar logo definida no início do projeto ao ser criado um plano de garantia de qualidade. No âmbito do projeto não serão desenvolvidas funcionalidades relacionadas com esta área.

A área de **Gestão de Configurações** obriga a Ubiwhere não só a definir todos os itens de configuração (componentes que necessitam ser administrados para que um determinado Serviço de Tecnologias da Informação (TI) possa ser entregue) utilizados pela empresa como a criar *Baselines* com toda a informação relativa ao estado do projeto. Estas *Baselines* englobam toda a documentação necessária para o seu planeamento e revisão até ao código gerado no âmbito do processo de desenvolvimento. Basicamente, tudo o que for envolvido na criação do projeto deve ser guardado. Pretende-se no âmbito deste trabalho que, através das várias informações presentes nas ferramentas Redmine, UIS e Google Spreadsheets, se recolham todos os dados necessários para a criação destas *Baselines*.

2.1.3 Mapeamento entre CMMI e Scrum

Apesar do CMMI seguir um conjunto de processos rigorosos, em contraste com a metodologia Scrum em que agilidade é uma das suas palavras chave, a sua junção é possível. Segundo o próprio SEI, o mapeamento entre as duas abordagens pode aumentar significativamente a *performance* da empresa e aponta que o fato da sua coligação não funcionar se deve à utilização do CMMI como *standard* em vez de um modelo. Afirma, também, que estas abordagens se completam sendo mais eficazes quando utilizadas juntamente pois ajudam na mitigação de alguns pontos fracos uma da outra [6]. O CMMI oferece solução à falta de rigor e estandardização do processo de desenvolvimento enquanto que o Scrum oferece agilidade tornando o CMMI mais eficiente na medida em que, além do foco nos processos, também se concentra nas necessidades do cliente aumentando a qualidade do produto e a satisfação do cliente.

Um bom caso de estudo que demonstra uma relação positiva entre ambos foi feito pela empresa Systetic que se encontra classificada com CMMI nível 5 e adotou a metodologia Scrum em 2006. Ao fazer o mapeamento conseguiu reduzir quase na ordem dos 50% diferentes categorias de trabalho, desde trabalho inicial, *rework*, defeitos e até diminuiu o trabalho despendido no foco dos processos [**?**].

Analisando as várias áreas de processo presentes no CMMI, assim como todos os envolventes no processo Scrum, pode afirmar-se que são várias as características que são partilhadas entre ambos. As práticas do Scrum são um bom exemplo para alguns das práticas do CMMI, principalmente no nível 2 onde as principais áreas são relativas à gestão e alteração do projeto [**?**]. Destas áreas as que têm uma grande cobertura direta utilizando a metodologia Scrum são:

- Gestão de Requisitos
- Planeamento do Projeto
- Monitorização e Controlo do Projeto

Em anexo encontra-se o mapeamento entre as práticas específicas (SP) de cada área apresentada, assim como as práticas de Scrum que devem ser seguidas para as meter em prática. Essas práticas encontram-se divididas pelos objetivos específicos de cada área (SG).

De seguida encontra-se um resumo do mapeamento de cada área anteriormente apresentada.

A área "Gestão de Requisitos"pode ser facilmente satisfeita utilizando a metodologia Scrum. Como apresentado no capítulo Scrum, os requisitos no Scrum encontram-se definidos no *Product Backlog*. Para a criação de requisitos que vão de encontro ao que o cliente necessita, é necessário que a equipa de desenvolvimento esteja a par do que o cliente quer. A compreensão dos requisitos pode ser feita durante a reunião realizada antes de todo o processo Scrum, onde o Product Owner apresenta o *Product Backlog* à equipa de desenvolvimento. Se existirem dúvidas, devem ser apresentadas durante essa reunião evitando o desenvolvimento incorreto dos mesmos. Cabe ao Product Owner fazer o acompanhamento bidirecional dos requisitos e manter atualizado o *Product Backlog* garantindo que o que está a ser desenvolvido vai de encontro ao que o cliente quer.

A área de processo "Planeamento do Projeto"encontra-se dividida em três categorias: estabelecimento de estimativas, criação de um plano do projeto e obtenção de compromisso com o plano. Para a criação do plano de um projeto é necessário conhecer o *Scope* do projeto, todo o trabalho que o projeto envolve. Este *Scope* pode ser calculado tendo em conta o *Product Backlog* e todo o trabalho necessário para colocar em prática o processo Scrum. A criação de um plano envolve a que sejam criadas estimativas para todo o esforço necessário para a realização do projeto assim como dos custos totais do projeto. Tendo em conta o *Product Backlog* e o processo utilizado para a obtenção de estivas no Scrum, o processo Planning Poker, é possível a criação de estimativas mais realistas garantindo que o plano do projeto se encontra o mais próximo da realidade possível. Quanto à garantia de que os recursos do projeto se comprometem com o plano traçado pode ser feita durante as reuniões de planeamento do *Sprint*.

Relativamente à última área apresentada "Monitorização e Controlo do Projeto", pode dizer-se que grande parte da monitorização do projeto pode ser feita durante as reuniões diárias e as reuniões de revisão dos *Sprints*. É durante estas reuniões que há a verificação dos compromissos e do envolvimento de cada *Stakeholder* assim como do estado do projeto. A utilização de uma *Taskboard* pode facilitar a verificação do trabalho realizado por cada utilizador de forma simples e intuitiva. Quanto ao estado do projeto, podem ser utilizados os gráficos de monitorização envolvidos no processo Scrum pois permitem perceber o estado real do projeto uma vez que comparam o trabalho estimado com o trabalho realizado.

2.2 Ferramentas Scrum

Nesta secção encontra-se informação relativa a diferentes ferramentas para equipas geridas pela metodologia Scrum. Primeiro, uma vez que a empresa utiliza o Redmine para a gestão dos seus projetos, foram analisados os *plugins* existentes para essa ferramenta. É feita uma comparação entre os vários *plugins* analisados, sendo apresentados os aspetos em comum e os aspetos que os diferenciam. É importante perceber quais as características existentes nos *plugins* pois podem ser uma boa opção para a satisfação de algumas das necessidades da empresa.

Após a apresentação dos *plugins* são apresentadas algumas ferramentas que, apesar de apresentarem aspetos comuns com os *plugins*, são ferramentas mais completas, alguns delas combatendo não só necessidades da metodologia ágil Scrum como algumas do modelo CMMI.

Esta pesquisa foi realizada tendo em conta diferentes objetivos. Primeiro mostrar o porquê da empresa não optar por soluções já existentes no mercado e segundo perceber que funcionalidades existem e como é feito o seu tratamento, como é o caso do mecanismo utilizado para a obtenção de estimativas.

2.2.1 Plugins Redmine

Como já enunciado, foi feito um estudo no que toca a *plugins* para o Redmine. Após pesquisa, verificou-se que são vários os *plugins*que pretendem facilitar a gestão de equipas geridas pelo Scrum. Neste relatório são apresentados os seguintes:

- Redmine Backlogs [7]
- Scrumbler [8]
- Scrum 2B [9]
- Agile Plugin [10]
- Agile Dwarf [11]

Todos os *plugins* apresentados são equivalentes a nível de funcionalidades apresentando-se uns com melhores características a nível de usabilidade que outros. Por usabilidade entende-se a facilidade com que as pessoas podem utilizar uma ferramenta. Cada *plugin* apresentado pode ser utilizado por todos os envolventes no decorrer do projeto pois contém funcionalidades pensadas nas tarefas de cada ator do Scrum, desde a criação do Product Backlog à criação automática do gráfico *Burndown Chart*. Na secção seguinte, encontra-se a comparação entre os vários *plugins* apresentados.

2.2.1.1 Comparação entre *plugins*

Como se verificou que são vários os aspetos em comum, optou-se por se fazer uma divisão entre o que lhes é comum e o que os diferencia. Relativamente aos aspetos em comum, todos:

- Apresentam o Product Backlog
- Permitem a criação de vários *Sprints*
- Dão suporte *drag and drop* para gerir o Product Backlog e os *Sprints*
- Utilizam uma *Taskboard*
- Criam automaticamente o gráfico *Burndown Chart*

Apesar de serem vários os aspetos em comum, também são vários os aspetos que os diferenciam. De todos, o Redmine Backlogs é o único que faz referência aos impedimentos que vão surgindo ao longo do projeto. Mesmo que este aspeto possa ser contornado pelos outros *plugins*, não existe uma área específica para tal, tornando menos intuitivo o seu uso.

Ao nível da informação apresentada na *Taskboard*, o Agile Plugin é o mais completo pois apresenta um maior número de informação em cada *card*, desde informação mais comum entre os *plugins* como por exemplo o nome da tarefa e a percentagem de trabalho realizado até ao momento até à exibição de diferentes gráficos relativos a cada tarefa.

O Agile Dwarf, apesar de fácil de utilizar, a nível de interface e usabilidade é o mais fraco. Tanto a criação do Product Backlog e dos Sprints Backlog como a *Taskboard* de gestão de cada Sprint, apresentam o mesmo design, o que se torna confuso para o utilizador. Além disso, outro aspeto que enfraquece este *plugins* é o fato de na *Taskboard* não ser possível diferenciar as diferentes tarefas como acontece nos outros *plugins*.

Um pormenor interessante no Scrumbler é o fato de apresentar o valor de *Story Points* existentes no Product e Sprint Backlog o que torna mais fácil para o utilizador perceber se o conteúdo do Backlog se encontra adequado ou não.

Um aspeto que diferencia o Scrum 2B dos outros *plugins* é o facto deste *plugin* apenas criar novas vistas tendo em conta o que existe no Redmine enquanto que os outros *plugins* permitem a gestão da informação modificando a base de dados no qual o Redmine está ligado.

De todos os *plugins*, o único que tem uma versão paga é o Agile Plugin. É de destacar a variedade de gráficos gerados além do *Burndown Chart*, que é o único criado através dos outros *plugins* e na sua versão *free*.

Após análise das características dos *plugins*, verificou-se que alguns das necessidades da empresa podem ser facilmente cobertas com a sua utilização. Estas são referentes à utilização de uma *Taskboard* e do gráfico de monitorização *Burndown Chart*. Apesar destas duas funcionalidades se encontrarem em todos os *plugins* apresentados é necessário perceber se algum deles é compatível com a versão do Redmine utilizado pela empresa, a versão 2.3.4. Assim, foi feita uma pesquisa de quais as versões suportadas por cada um, encontrando-se os resultados na tabela 2.1.

Plugin	**Versões Suportadas**
Redmine Backlogs	2.2.4, 2.3.2
Scrumbler	1.2.x, 1.3.x
Scrum 2B	2.6.x, 2.5.x, 2.4.x, 2.3.x, 2.2.x, 2.1.x, 2.0.x
Easy Agile	2.6, 2.5, 2.4, 2.3
Agile Dwarf	Todas as versões

Tabela 2.1: Versões do Redmine suportadas pelos *plugins* estudados

Como se pode verificar, existem alguns *plugins* que são suportados pela versão do Redmine utilizado pela empresa, o Scrum 2B e o Agile Dwarf.

2.2.2 Ferramentas All-in-One para equipas geridas por Scrum

Ao longo dos últimos anos o mercado tem apostado na criação de ferramentas que exploram esta metodologia prometendo ajudar as empresas na gestão dos seus projetos. Atualmente existe um grande leque de ferramentas utilizadas para este fim, sendo impossível a descrição e comparação de todas elas. Foi feita a análise das seguintes ferramentas:

- Version One [12]
- JIRA [13]
- Axosoft Scrum [14]
- Yodiz [15]
- ScrumDo [16]
- ScrumDesk [17]

Considera-se que as ferramentas escolhidas representam u boa amostra do que existe no mercado. Para cada u delas é apresentada u breve descrição, onde se são detalhadas características necessárias para o desenvolvimento do projeto. No fim da apresentação das ferramentas é feita u comparação entre todas tendo em conta as necessidades da empresa.

2.2.2.1 Version One

Ferramenta que abrange o maior número de *features* no que toca à gestão ágil de projetos, estando disponível como Software as a Service (SaaS) ou On-Premise. Segundo o 8th Annual State Of Agile Survey é a ferramenta específica para a gestão de projetos ágeis mais utilizada (41%), ocupando o 3º lugar contra o Excel e o Microsoft Project (Ferramentas não especializadas para o desenvolvimento ágil). Ainda segundo este relatório é a ferramenta que mais é recomendada pelos seus utilizadores (93%). Permite a integração com diversos tipos de ferramentas sendo o Redmine uma delas. Encontra-se dividida pelas diferentes etapas do desenvolvimento permitindo a sua utilização em todas elas.

De destacar a integração da funcionalidade *Estibly* que permite a criação de estimativas em *real time* utilizando a técnica Planning Poker. Ao utilizar esta funcionalidade, cabe ao responsável pelo processo o fornecimento de um *url* aos elementos da equipa. Este url identifica o processo e limita a sua utilização apenas aos utilizadores que o têm. Assim que todos os elementos se encontrem presentes no processo cabe ao responsável escolher, uma a uma, as *User Stories* que pretende estimar. Após a escolha de uma *User Story*, todos os elementos da equipa optam pela estimativa que consideram mais adequada. Assim que todos tenham todo a sua decisão, o responsável recebe a informação e analisa-a. Cabe ao responsável decidir qual a estimativa mais correta tendo em conta as estimativas apresentadas. No caso em que as estimativas são muito diferentes, há a necessidade de esclarecer o porquê dessas diferenças. Após discussão e esclarecimento do porquê das estimativas escolhidas, o responsável pode repetir o processo e obter novas estimativas.

Apesar de ser uma ferramenta bastante completa a todos os níveis de desenvolvimento destaca-se pela quantidade de funcionalidades de *reporting* e análise. Além das funcionalidades base permite a monitorização do progresso utilizando mais de 50 métricas diferentes, o que a torna uma ferramenta bastante poderosa neste aspeto.

2.2.2.2 JIRA

Ferramenta muito utilizada para a gestão de projetos desenvolvida em Java pelo grupo Atlassian. Permite a importação de dados de outras ferramentas, sendo o Redmine uma delas. No 8th Annual State Of Agile Survey encontra-se nas primeiras posições de ferramenta recomendada com 87%, ficando logo abaixo da ferramenta Version One.

É uma ferramenta bastante versátil pois contém um elevado número de *add-ons* que permitem a adição de várias funcionalidades, não só a nível de gestão de projetos como por exemplo a nível da gestão de testes. Também fornece uma API REST caso o utilizador pretenda criar os seus próprios add-ons.

De destacar o *add-on* JIRA Agile desenvolvido para facilitar a gestão de equipas que utilizam a metodologia Scrum. O seu principal foco é na gestão do trabalho de cada Sprint através da utilização de uma *Taskboard* que pode facilmente ser gerida através de *drag and drop* e pode seguir diferentes *workflows*, tanto criados pelo utilizador como um dos muitos existentes na ferramenta.

Por último, a integração de vários sistemas de controlo de versões como Git e Subversion permitindo a sua associação à tarefa correspondente também é um aspeto interessante desta ferramenta.

2.2.2.3 Axosoft Scrum

Ferramenta bastante completa no que toca à gestão de projetos de equipas geridas não só por Scrum como por outras metodologias ágeis, desenvolvida pela equipa Axosoft. O conteúdo da ferramenta pode ser guardado tanto localmente como na *cloud* que a Axosoft possui para o efeito. Assim como a ferramenta JIRA, é constituída por diversos *add-ons*, permitindo a importação e migração de dados por exemplo da ferramenta JIRA e a integração com várias ferramentas desde IDEs a sistemas de controlo de versões como o GitHub e o TortoiseSVN.

Esta ferramenta é constituída por três listas diferentes o que facilita a gestão da informação. Uma para gerir o Product Backlog, outra relativa a *bugs* e outra para gestão dos vários impedimentos que vão surgindo ao longo do projeto. Um aspeto interessante é a visualização de cada lista numa *Taskboard* que pode ser gerida através de *drap and drop* e permite uma melhor transparência relativamente a estes aspetos.

Apesar destas funcionalidades, é de destacar o foco que faz ao nível de informação que poderá ser utilizada para análise do projeto. Um dos pontos fortes desta ferramenta é o número de diferentes gráficos e cálculos que são criados automaticamente sendo possível criar diferentes *Dashboards* com diferentes conteúdos, ficando ao critério do utilizador a sua utilização consoante o que considera mais apropriado.

Além desta informação, outro aspeto interessante é o foco individual em cada um dos elementos do projeto ajudando cada individuo na análise diária do seu trabalho sendo criada informação do trabalho desenvolvido e do trabalho restante assim como da velocidade do trabalho realizado.

2.2.2.4 Yodiz

Ferramenta que pode ser utilizada via *web* ou através de várias Apps para Android e IOS. Permite a integração com o Google Docs, associando a cada projeto documentos guardados neste. Destaca-se das outras ferramentas pois apresenta um grande leque de estatísticas relativas aos *Sprints* e projetos, possibilitando em alguns casos perceber diretamente o seu estado relativamente a u fase anterior. Um exemplo é o cálculo de número de Bugs de um projeto.

2.2.2.5 ScrumDo

Ferramenta pensada para a metodologia Scrum que permite a integração com várias ferramentas, como por exemplo a ferramenta JIRA para a gestão de projetos e a versão de controlo GitHub. O seu design encontra-se bastante simples e intuitivo o que ajuda e facilita a sua utilização. A possibilidade de exportação de diversas informações para PDF, possibilitando a recolha e análise dessas informações sem estar ligado à ferramenta é uma das funcionalidades existentes. Além dessa funcionalidade, é de destacar a quantidade de estatísticas apresentadas, bastante simples e intuitivas. Uma funcionalidade interessante desta ferramenta é o fato de possuir uma API REST que possibilita a extração de informação ou a sua integração com programas criados pelo utilizador.

2.2.2.6 ScrumDesk

Ferramenta *web based* desenvolvida com o intuito de ajudar equipas que seguem os princípios ágeis. Encontra-se dividida em duas versões, Srt e Professional. A versão Professional encontra-se com muito mais funcionalidades sendo uma delas um mecanismo de Planning Poker. Este mecanismo é feito em *runtime*, ficando o Scrum Master com a função de moderar o processo escolhendo, uma a uma, as *User Stories* que serão estimadas e o tempo que cada elemento da equipa tem para selecionar a estimativa adequada. Após obter todas as estimativas, cabe ao Scrum ster a escolha do valor final. Outro fator a destacar é a exportação de relatório para PDF, Word ou Excel com diversas informações tais como as *User Stories* de cada *Sprint*.

2.2.2.7 Comparação entre ferramentas

Na tabela 2.2 encontra-se a comparação entre as diferentes ferramentas apresentadas. Esta encontra-se dividida em várias categorias que correspondem aos momentos existentes no processo Scrum. Em cada uma delas encontram-se as necessidades da empresa com ela relacionadas.

Como se pode ver na tabela 2.2 apresentada, as ferramentas Scrum não cobrem todas as necessidades da empresa. Estas ferramentas foram criadas explicitamente para combater as necessidades das metodologias ágeis, não sendo uma boa solução para o problema apresentado, onde além desta metodologia também se tem de suportar as práticas do modelo de maturação CMMI, nomeadamente o cálculo das várias métricas definidas pela empresa.

Quanto à gestão de todo o processo Scrum pode afirmar-se que são ferramentas completas e podem ser utilizadas na maior parte do processo. Desde a definição do *Product Backlog*, à criação dos *Sprints*, passando pela gestão de tarefas a serem desenvolvidas, até à fase de revisão e retrospetiva do mesmo. Contudo no que toca ao CMMI, mais propriamente ao cálculo das métricas exigidas e à exportação de toda a informação do projeto pode afirmar-se que se encontram bastante incompletas.

A que mais se adequa às necessidades da empresa é a ferramenta Version One pois além de suportar todo o processo Scrum, permite a importação de informação da ferramenta Redmine assim como o cálculo de alguns métricas que a empresa necessita. Contudo, é de notar que as métricas que a Ubiwhere se vê obrigada a calcular são métricas que se encontram definidas pela própria empresa, não sendo possível o cálculo da totalidade por parte de nenhuma ferramenta analisada. De notar também que, a ferramenta Version One tem custos associados de $39 por utilizador por mês, não sendo a sua opção viável para a empresa.

	Version One	JIRA	Axosoft Scrum	Yodiz	ScrumDo	ScrumDesk
Planeamento do Produto						
Gestão Product Backlog	X	X	X	X	X	X
Mecanismo Planning Poker	X	X	X	X	X	X
Reunião de Planeamento						
Gestão Sprint Backlog	X	X	X	X	X	X
Mecanismo que ajude na alocação de recursos	-	-	-	-	-	-
Decorrer da Sprint						
Taskboard	X	X	X	X	X	X
Gestão impedimentos	X	-	X	-	-	X
Reunião de Revisão						
Burndown Chart	X	X	X	X	X	X
Métricas por Sprint	+/-	-	+/-	+/-	+/-	+/-
Métricas por Projeto	+/-	-	-	-	+/-	-
Métricas por Tipos de Projeto	+/-	-	-	-	-	-
Métricas por negócio	+/-	-	-	-	-	-
Métricas de custos	-	-	-	-	-	-
Reunião de Retrospetiva						
Gestão de retrospetivas	X	-	X	-	-	X
Outros						
Exportação informação do projeto	+/-	+/-	+/-	+/-	+/-	+/-
Integração Redmine	X	X	-	X	-	-
Integração Jenkins	X	-	-	X	-	-
Integração Git	X	X	X	X	X	-
Preço	$39/u/m	$1800	$25/u/m	$20/u/m	$200/m	$299/m

2.3 Ferramentas para extração de informação

No âmbito do trabalho é necessário a obtenção de alguns informações que são calculadas em *runtime* pelo Redmine. Um exemplo disso é a percentagem de realização de uma *User Story* tendo em conta as diferentes tarefas descendentes desta.

Apesar de o Redmine disponibilizar uma API RESTful que permite a obtenção de alguns informações relacionadas com os projetos e tarefas existentes, não possibilita a obtenção de informações que são calculadas em *runtime*. Para a sua obtenção é necessário fazer diversos cálculos que podem ser pesados a nível de performance e custos além de que o seu cálculo manual aumenta a probabilidade de possíveis erros.

Após estudo, verificou-se que a única solução para contornar este problema seria a utilização de *Web Crawlers*. Pode afirmar-se que *Web Crawler* é um programa que permite a extração do conteúdo de *web pages* sabendo simplesmente o seu *Url* e definindo o tipo de dados que se quer extrair. O *Web Crawler* começa por extrair o código fonte da página tornando possível fazer *Screen Scrapping* da informação pretendida, transformando informação não estruturada em informação estruturada. A nível de *performance* e custos torna-se mais eficiente pois simplesmente retira a informação da página, anteriormente já calculada pelo Redmine, não sendo necessário efetuar nenhum cálculo.

Analisando a hipótese de se adotar ferramentas que permitam a extração de informação através de *Web Crawlers* verifica-se que existem várias para este efeito. Após alguma informação recebida por parte da empresa e posteriormente pesquisa, percebeu-se que ferramentas possíveis para fazer esta extração são o Scrapy [18], o Import io [19], Kimono [20] e Mozenda [21]. Optou-se pela escolha destas ferramentas por oferecerem soluções completas não sendo necessário a utilização de mais nenhuma ferramenta para completar a obtenção da informação, isto é, além de extraírem informação de páginas *web* também fazer o *Screen Scrapping* dessa informação.

2.3.0.8 Scrapy

Framework do *python* que permite fazer *web crawling* e *screen scraping* de *web sites*. Através da definição de um *Spider*, classe onde se encontra informação de quais os *web sites* e como será feito o *parse* da informação extraída, é possível obter a informação pretendida.

A sua utilização poderá ser simples ou mais complexa dependendo se é necessário fazer a autenticação nos *web sites* assim como do tipo de informação que o utilizador pretende. Para fazer *scrapping* da informação o *Scrapy* utiliza *selectors* que selecionam a informação através de expressões XML Path Language (XPATH) e/ou Cascading Style Sheets (CSS).

De modo a centralizar a informação, é possível guardá-la utilizando um *item*, que consiste num *container* criado para esse efeito. Através deste *container* torna-se mais fácil exportar a informação extraída, sendo neste caso possível fazer a sua extração nos formatos JavaScript Object Notation (JSON), Comma Separated Value (CSV) e eXtensible Markup Language (XML).

2.3.0.9 Import io

Plataforma *web-based* que permite a extração de informação de *web sites* numa tabela e a criação de uma API sem necessidade de *coding skills*. Através da seleção manual de alguns exemplos, a plataforma aprende e gera um algoritmo que permite recolher a informação pretendida, sendo bastante fácil e intuitiva a sua utilização. No caso em que o *web site* necessita de autenticação é pedido ao utilizador que faça o *login* na página de autenticação, dando acesso à plataforma para fazer o *crawl* da informação. Esta informação é guardada na *cloud* sendo essa a principal desvantagem da sua utilização. Relativamente à exportação da

informação, pode ser feita em diversos formatos sendo estes JSON, CSV, HyperText Markup Language (HTML) e XLS.

2.3.0.10 Kimono

Ferramenta *web-based* que segue o mesmo conceito da ferramenta Import io. Através da seleção manual, a ferramenta reconhece a informação semelhante, criando padrões que permitem a obtenção de informação estruturada de forma fácil e rápida. Permite a criação automática de uma API que pode ser utilizada para a obtenção da informação selecionada, nos formatos JSON ou CSV. Assim como as ferramentas apresentadas anteriormente permite a obtenção de dados de páginas que precisam de autenticação. Esta versão ainda se encontra numa versão Beta, estando só disponível numa versão *free*, versão essa que não é apropriada para a empresa pois só permite a criação de API públicas.

2.3.0.11 Mozenda

Assim como o Import io e o Kimono, a ferramenta Mozenda permite a extração da informação sem ser necessário *code skills* o que torna a sua utilização bastante acessível. Através da *interface* gráfica é possível fazer a seleção manual da informação que se pretende extrair assim como agendar datas para fazer essa extração automaticamente, possível nos formatos CSV, Tab Separated Values (TSV) e XML. Além disso, disponibiliza uma API REST que permite, através da criação de código, a obtenção de informação.

2.3.0.12 Comparação Ferramentas de extração de informação

De modo a se poder verificar qual a melhor ferramenta para fazer a extração da informação foi feita uma pesquisa tendo sido criada a tabela 2.3.0.12 como resultado. Nessa tabela, encontra-se a comparação entre as várias ferramentas tendo em conta aspetos importantes para a decisão da ferramenta adequada.

	Scrapy	*Import io*	*Kimono*	*Mozenda*
Crawling de *web sites* autenticados	X	X	X	X
Code skills	X	-	-	-
Local onde é guardada a informação	**Client - Side**	**Server - Side (Cloud)**	**Server - Side**	**Server - Side**
Exportação de informação JSON	X	X	X	-
Preço	Grátis	Grátis	Não definido	$199/mês

Tabela 2.3: Comparação características ferramentas extração de informação

Como se pode ver na tabela , existem alguns semelhanças entre as ferramentas utilizadas, como o facto de ser possível obter dados de *sites* autenticados. Esta característica é muito importante pois a informação que se pretende extrair encontra-se protegida só sendo acedida através da autenticação na página. As ferramentas apresentadas são bastante fáceis de utilizar, sendo necessário apenas *code skills* na ferramenta Scrapy. Uma das características mais importantes e que tem bastante peso na decisão da ferramenta a utilizar é relativa ao local onde são guardados os dados obtidos, do lado do cliente ou do lado do servidor. Ambas as soluções têm vantagens e desvantagens, sendo umas mais apropriadas para uns

casos e outras para outros. Outro fator que também de deve ter em conta é o preço da ferramenta. Verifica-se que quando a isto as melhores soluções são o Scrapy e o Import io pois são ferramentas *free*. Para perceber qual a melhor solução para este caso específico, foi feita uma pesquisa encontrando-se na tabela 2.4 a comparação entre as duas, tendo em conta diversos fatores.

	Client-Side	*Server-Side*
Segurança da informação	+	-
Velocidade na obtenção da informação	+	-
Acessibilidade da informação	-	+
Custos de manutenção	-	+

Tabela 2.4: Comparação entre *Client-Side* e *Server-Side*

Neste caso em específico, o fator mais importante é relativo à segurança dos dados pois, sendo a informação confidencial é muito importante que esta se encontre o mais segura possível evitando ataques de terceiros. Como se pode ver na tabela 2.4 uma solução *Client-Side* oferece maior segurança quando comparado com uma solução *Server-Side*. Além disso, a velocidade com que é possível obter os dados dos diferentes sistemas costuma ser maior numa solução *Client-Side* do que numa solução *Server-Side*. Verifica-se que numa solução *Server-Side* os dados encontram-se mais facilmente acessíveis pois estando a informação na *Cloud* é possível aceder à informação em diferentes sítios, não sendo necessário ficar preso ao local onde se encontram os dados, contudo esta característica não é relevante para a empresa. A nível de custos, uma solução *Server-Side* costuma ser is cara do que u solução *Client-Side*, contudo esse aspeto não e muito relevante pois cada ferramenta tens os seus custos associados. Após análise dos diversos fatores apresentados, verifica-se que para a empresa a melhor solução é o **Scrapy**, pois a informação fica do lado do cliente e a ferramenta não tem custos associados à sua utilização.

Capítulo 3

Planeamento e Metodologia

Neste capítulo é apresentado não só o planeamento do trabalho realizado ao longo do projeto como os desvios relativamente ao mesmo. Além disso, neste capítulo é possível conhecer qual a metodologia utilizada para a realização do mesmo. Por fim, é apresentada uma análise dos riscos associados ao projeto que podem afetar o planeamento inicialmente estipulado.

3.1 Planeamento

Na figura 3.1 encontra-se o diagrama de Gantt respetivo ao planeamento de todo o projeto. Este divide-se em duas áreas, uma correspondente ao 1º Semestre e outra correspondente ao 2º Semestre. Além disso, também é possível verificar os desvios efetuados face ao planeamento definido inicialmente. Nesta secção é apresentada uma breve descrição das categorias em que se inserem as fases apresentadas no diagrama 3.1.

3.1.1 1º Semestre

Relativamente ao 1º Semestre, este dividiu-se em quatro categorias:

- **Contextualização**: corresponde às 3 primeiras fases apresentadas no diagrama de Gantt. Durante este período de tempo foi feito o estudo dos processos existentes na Ubiwhere assim como as metodologias seguidas pela mesma, Scrum e CMMI.
- **Estado da Arte**: foi neste período de tempo que o estagiário andou a analisar as várias ferramentas apresentadas no capítulo 2.
- **Levantamento de Requisitos**: esta fase foi dividida em duas partes. Primeiro, foi feito o levantamento das necessidades da Ubiwhere tendo sido depois validadas por parte da empresa confirmando ou não as necessidades reais da mesma. Após confirmação, foram criadas as *User Stories* que mapeavam as necessidades definidas e houve novamente uma verificação por parte da empresa que as priorizou conforme a sua importância. Por último foram divididas pelos diferentes módulos tendo sido feita a sua submissão para o Redmine. Mais informações relativa aos requisitos do projeto encontram-se na secção User Stories (4.1).
- **Estudo e Definição da Arquitetura**: durante este período foi feito um estudo da arquitetura existente e adaptação da mesma para dar suporte ao projeto. O resultado desse estudo encontra-se na secção Arquitetura (4.2).
- **Escrita Relatório Intermédio**: a última fase foi relativa à criação e revisão do relatório intermédio.

3.1.2 2º Semestre

Quanto ao 2º Semestre, este dividiu-se essencialmente em 3 categorias:

- **Implementação**: fase despendida para a implementação dos requisitos apresentados. Maior detalhe no capítulo Implementação (5). Como se pode observar na figura, a implementação foi dividida, essencialmente, pelos módulos onde os as User Stories (presentes na secção 4.1) pertencem, pois nalgumas situações foi necessário a implementação de algumas funcionalidades inexistentes e que pertenciam a outros módulos. Contudo, de uma maneira geral pode dizer-se que os módulos implementados foram os apresentados no diagrama e maior parte das funcionalidades foram implementadas na altura apresentada. Para todo o código desenvolvido, e aquele que já se encontrava implementado mas não possuía qualquer documentação, foi criada a respetiva documentação e alocada no Redmine na secção já existente para o efeito. Em anexo, na secção é possível observar, de um modo geral, toda a documentação criada e um exemplo de uma página gerada.
- **Testes e correções**: fase bastante importante em que foi feita a validação e respetiva correção de erros do código implementado. Mais detalhes relativamente a esta fase encontram-se no capítulo Testes, capítulo número 7.
- **Escrita Relatório Final**: a par do desenvolvimento foi escrito o relatório final de estágio.

3.2 Desvios ao Planeamento

Como se pode observar na imagem 3.1, o planeamento do projeto sofreu algumas alterações face ao planeamento inicial. No 1º Semestre, verifica-se que o plano foi seguido quase na totalidade, existindo apenas um pequeno atraso na obtenção das User Stories (US), contudo este atraso em nada prejudicou o projeto pois estas foram obtidos em menos tempo que o planeado. Relativamente ao 2º Semestre, o plano já sofreu mais algumas alterações. Considera-se que o fato de existir uma grande diferença nos desvios do planeamento do 1º Semestre face ao 2º Semestre deve-se ao fato de no 1º existir uma maior especificação do trabalho a desenvolver. Neste caso, visto que o projeto não é criado de raiz, existe uma maior probabilidade de errar nas estimativas, como por exemplo, considerar-se que algumas funcionalidades se encontram corretamente desenvolvidas e só na altura da implementação verificar o contrário.

Na opinião do estagiário, as alterações face ao planeamento esperado foram graças a/ao:

- Mudança de prioridade de algumas funcionalidades, como no módulo Allocations.
- Surgimento de novas funcionalidades anteriormente não esperadas, como é o caso das funcionalidades referentes ao Módulo Users.
- O fato de se considerar que algumas funcionalidades, como por exemplo as funcionalidades referentes ao Módulo Efforts, não fossem necessárias para o desenvolvimento de módulo anteriormente agendados, como por exemplo o módulo Costs.
- Funcionalidades incorretas e mais incompletas do que era esperado.
- Dificuldade em definir claramente e objetivamente os requisitos em conjunto com a empresa.
- Mudança de PM do projeto.

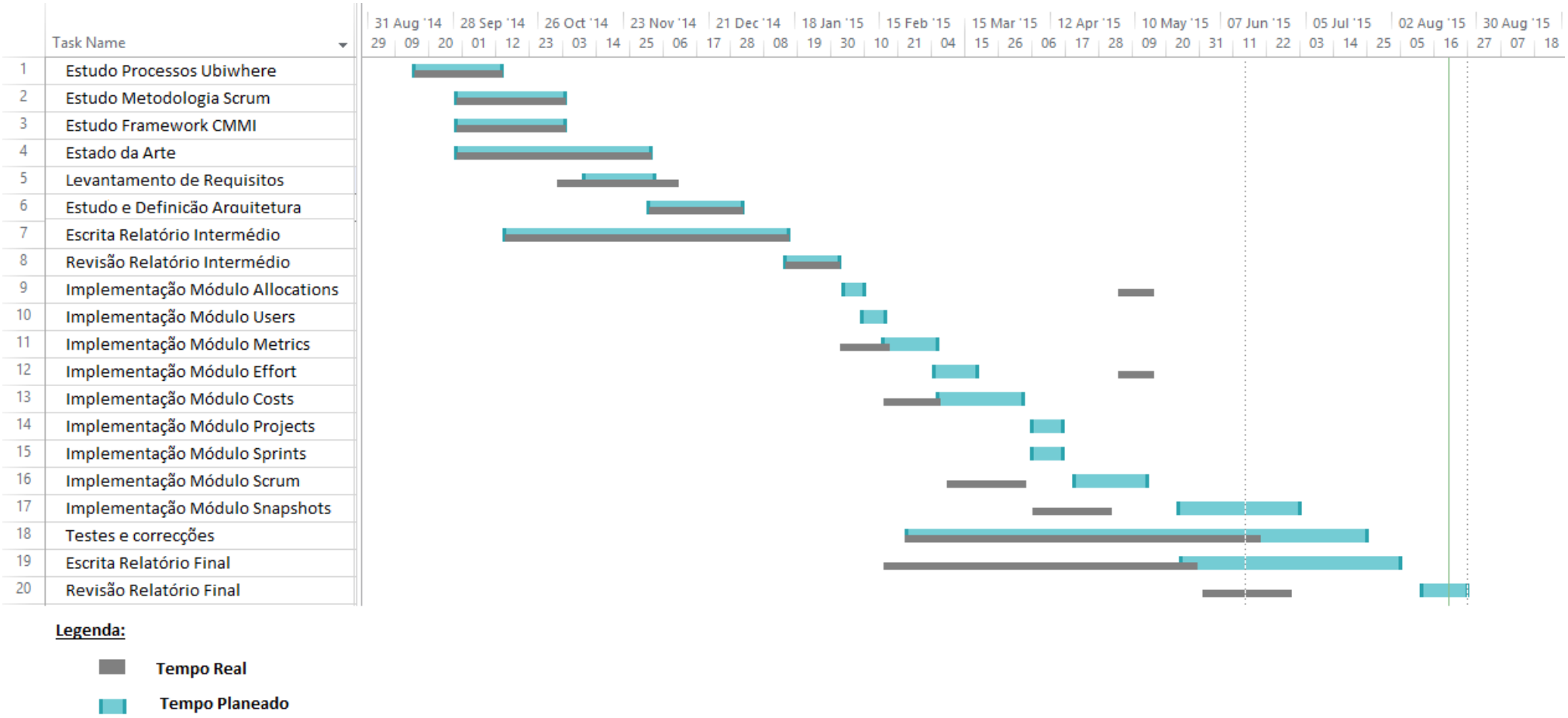


Figura 3.1: Diagrama de Gantt com informação sobre planeamento estimado e real do projeto

3.3 Metodologia e Ferramentas utilizadas

Inicialmente foi definido que a metodologia a utilizar seria baseada em Scrum. As *User Stories* seriam divididas em tarefas mais pequenas e atribuídas a vários *Sprints* de duas semanas. Por cada *Sprint* seria feita uma reunião de planeamento onde era atribuídas as tarefas adjacentes a este e onde o estagiário se comprometia ao seu desenvolvimento. Após cada *Sprint* seria feita uma nova reunião onde o estagiário apresentava o que desenvolveu, e caso não tivesse atingido todos os objetivos apresentaria a razão. Contudo, com a mudança de orientador, e visto que este não se encontrava no mesmo local do estagiário, decidiu-se fazer todas as semanas uma reunião em que era apresentado o trabalho realizado e possível problemas, assim como definido o que fazer na semana seguinte.

Para a gestão de tarefas foi utilizada a ferramenta **Redmine**. Nesta ferramenta encontrou-se toda a informação relativa a cada *Sprint* / tarefa a realizar no âmbito do projeto, tendo sido feito o registo de horas de trabalho no mesmo. Relativamente ao desenvolvimento de código foi utilizada a ferramenta **PyCharm** [22]. Esta ferramenta permitiu uma boa visualização e gestão do código. Para a gestão da BD utilizada, além do PyCharm foi utilizado o **pgAdmin** [23].

Quanto ao código, este esteve alojado numa instância da Ubiwhere utilizando o repositório **Git** [24] e estava dividido em dois projetos diferentes, um referente ao código do sistema e respetiva API e outro ao código do cliente *web*. Para cada um dos projetos foi criada uma *branch* para o estagiário. Além dessa *branch* encontravam-se já definidas a *branch* relativa à qualidade onde foram feitos os testes ao código submetido pelo estagiário e outra *branch* de produção que contém todo o código que se encontra funcional e devidamente testado. Para fazer a gestão do Git foi utilizada a ferramenta **SourceTree** [25].

De forma a interagir e visualizar a resposta da API foi utilizada a ferramenta **Postman** [26]. Já para testar a aplicação foram utilizadas as ferramentas **Selenium** [27] e **SoapUI** [28]No Selenium foram feito alguns testes funcionais relativamente ao cliente Web. Já o SoapUI foi utilizado para testar o sistema e respetiva API.

3.4 Análise de Riscos

Para aumentar a probabilidade de sucesso de um projeto é necessário estudar e compreender possíveis riscos que lhe estão associados, possibilitando a sua resolução e adaptação o quanto antes possível. Assim sendo, foi feita uma análise de possíveis riscos do projeto, encontrando-se essa informação na tabela 3.1. Cada risco apresenta uma probabilidade e grau de impacto no projeto caso tal situação aconteça. Tanto a probabilidade como o grau de impacto encontram-se definidos numa escala de 1 a 5, sendo 1 correspondente a uma probabilidade / impacto para o projeto muito baixa e 5 uma probabilidade muito alta e que têm um impacto muito elevado relativamente ao sucesso do projeto. Também, caso exista, para cada risco encontra-se definido um plano de mitigação, onde é descrito como se deve proceder de forma a mitigar tal risco.

Risco	**Probabilidade**	**Impacto**	**Plano de mitigação**
Dependência de sistemas externos (Redmine, Google Spreadsheets), não podendo garantir que se encontram sempre acessíveis.	1	5	Quanto ao Google Spreedsheets, poder-se-á fazer o tratamento dos ficheiros localmente, fazendo a sua adaptação assim que possível.
Dificuldade na compreensão do código desenvolvido anteriormente e que serve de base para a realização do projeto.	2	2	Dispensar horas extra no seu estudo e pedir ajudar, se possível, a quem desenvolveu o código anteriormente.
Código que serve de base para o trabalho a desenvolver estar mais incompleto do que o esperado.	3	4	Dispensar horas extra na sua resolução de forma a não comprometer o trabalho esperado na realização do projeto.
Dificuldade na adaptação das tecnologias utilizadas.	3	3	Dispensar horas extra no estudo das tecnologias utilizadas de forma a não comprometer o trabalho esperado na realização do projeto.
Indefinição, por parte da empresa, do que realmente necessitam.	3	5	Fazer pressão para uma melhor definição do que é suposto, sugerindo algumas possibilidades de modo a facilitar a decisão.
Mudança de requisitos relativamente às necessidades da empresa.	2	5	Sendo o projeto uma continuação / complemento do sistema de informação da empresa, podem surgir alguns requisitos com maior prioridade que os definidos no início. Se necessário aplicar horas extras para não comprometer o trabalho esperado.

Tabela 3.1: Análise de Riscos do Projeto.

Capítulo 4

Especificação Técnica

Nesta secção começar-se-à por apresentar as User Stories recolhidas no âmbito do projeto. Após é feita a apresentação da arquitetura de alto-nível da solução assim como de vários diagramas de sequência que representam a sequência dos diferentes componentes do sistema e a sua interação. Ainda nesta secção é apresentado o modelo entidade-relacionamento da base de dados do projeto.

4.1 User Stories

Nesta secção são apresentados os requisitos que se teve de dar resposta no âmbito do projeto. Estes encontram-se divididos em requisitos funcionais, que dizem respeito a funcionalidades que o sistema deverá suportar, e requisitos não-funcionais, que não dizem respeito às funcionalidades do sistema mas sim a componentes técnicos do mesmo.

Os primeiros encontram-se definidos na forma de US e seguem a seguinte sintaxe:

Como [utilizador] quero [objetivo] para [motivo].

As *User Stories* apresentadas referem-se a diferentes utilizadores, onde:

- User: simples RH da Ubiwhere, sem privilégios especiais
- PM: Project Manager
- Manager: gestor de topo da empresa

Coube ao estagiário a identificação das necessidades da empresa e posteriormente a transformação dessas necessidades em *User Stories*. Após, foi feita a validação por parte da empresa tendo sido atribuída uma prioridade que vai de 1 a 5 consoante a sua importância, em que 1 corresponde à prioridade mínima e 5 à prioridade máxima. Como **medida de sucesso do projeto**, definiu-se a implementação de todos os requisitos com prioridade superior a 2, isto é, todos os requisitos entre 3 e 5. No decorrer do projeto foi feito o refinamento das US a implementar, originando algumas alterações relativamente às US definidas no início do projeto.

Nas tabelas seguintes são apresentadas as *User Stories* do projeto assim como o ID e prioridade correspondente, encontrando-se estas divididas pelos diferentes módulos onde se inserem. Por fim são apresentados os requisitos não funcionais do sistema. Estes encontram-se na tabela 4.10 e já estavam definidos pela empresa.

ID	Como	quero	para	Prioridade
L.1	PM	usar um processo Planning Poker	-	5
L1.1	PM	decidir a complexidade de uma *User Story* utilizando o processo *Planning Poker*	perceber o esforço necessário para a sua realização	5
L1.2	PM	gerir um processo de *Planning Poker*	obter estimativas de todos os elementos da equipa	5
L1.3	User	quero contribuir com a minha opinião sobre a complexidade de uma *User Story* do projeto em que estou envolvido	criar estimativas mais realistas	5
L2	PM	avaliar qualitativamente um Sprint recorrendo ao estado do Sprint anterior e às métricas de projeto atuais	efetuar uma avaliação mais informada do estado do projeto	5
L3	PM	criar o *Sprint Backlog*	conhecer as User Stories que serão desenvolvidas durante o Sprint	2
L4	User	utilizar uma Taskboard	facilitar a gestão das minhas tarefas	2
L5	PM	visualizar o Sprint Burndown Chart ou o Backlog Burdown Chart	ter uma vista mais alto nível do Sprint ou do Backlog	4
L6	PM	adicionar, editar e eliminar pontos a favor e contra de um Sprint	facilitar próximos Sprints	2
L7	PM	gerar a documentação relativa à *Sprint Review* e exportar diretamente para o Redmine	facilitar o processo de avaliação do Sprint	5
L8	PM	gerir o conjunto de requisitos de um projeto (Product Backlog)	obter um conjunto de requisitos atualizado de acordo com as necessidades do cliente	2
L9	PM	ver a listagem do Product Backlog de um projeto	conhecer todo o conjunto de requisitos de um projeto	1
L10	PM	saber o estado de cada tarefa	verificar se os objetivos estão a ser cumpridos	1
L11	PM	adicionar, editar e eliminar bugs que surjam ao longo do Sprint	fazer a sua gestão e atribuição para resolução	2
L12	PM	adicionar, editar e eliminar impedimentos que surjam ao longo do Sprint	fazer a sua gestão e resolvê-los	2
L13	PM	adicionar uma justificação a cada tarefa "unfinished"	perceber o motivo de não ter sido concluída	1
L14	PM	gerir ações de correção	resolver problemas encontrados	1
L15	PM	atribuir ações de correção à Sprint correspondente	se proceder à sua resolução	1

Tabela 4.1: *User stories* para o módulo Scrum

ID	Como	quero	para	Prioridade
I1	PM	gerar uma baseline/snapshot do projeto	ficar com o histórico do projeto	5
I2	PM	exportar para PDF/XLSX toda a informação relativa ao Sprint	guardar informações relativas a este	5
I3	PM	exportar para PDF/XLSX toda a informação relativa ao projeto	guardar informações relativas a este	5
I4	PM	visualizar uma baseline/snapshot antigo do projeto	visualizar o histórico do projeto	5

Tabela 4.2: *User Stories* para o módulo Snapshots

ID	Como	quero	para	Prioridade
H1	Manager	ver o esforço despendido baseado em critérios específicos	saber o custo real do trabalho	4

Tabela 4.3: *User Stories* para o módulo Effort

ID	Como	quero	para	Prioridade
A5	PM	conhecer a disponibilidade dos membros da equipa	fazer a atribuição de tarefas	4

Tabela 4.4: *User Stories* para o módulo Allocations

ID	Como	quero	para	Prioridade
F1	PM	adicionar ou editar um novo orçamento para o projeto	estipular uma quantidade específica para gastar durante o projeto	4
F2	PM	verificar o orçamento do meu Sprint	verificar o quanto eu gastei em apenas um Sprint	4
F3	PM	verificar o orçamento global do meu projeto	obter uma imagem sobre os custos do projeto	4
F4	Manager	ter acesso aos custos de todos os projetos, podendo filtrar por projeto ou tipo de projeto	conhecer os custos totais dos projetos	4

Tabela 4.5: *User Stories* para o módulo Costs

ID	Como	quero	para	Prioridade
G1	PM	ver métricas específicas de um Sprint	saber a performance real da equipa	4
G3	PM	ver as métricas CMMI de um projeto	saber se o meu projeto se encontra de acordo com os objetivos	4

Tabela 4.6: *User Stories* para o módulo Metrics

ID	Como	quero	para	Prioridade
U1.4	User	ter acesso e visualizar informação relativa às *skills* dos outros utilizadores	conhecer melhor os meus colegas	4
U1.5	User	listar e pesquisar outros utilizadores	obter o utilizador que eu quero	4
U1.7	User	ver informação específica sobre os locais de trabalho da Ubiwhere	conhecer mais sobre cada localização	4

Tabela 4.7: *User Stories* para o módulo Users

ID	Como	quero	para	Prioridade
B2	User	ver e ter acesso a todos os utilizadores que estão ou estiveram envolvidos no projeto	conhecer melhor o projeto	4
B4	User	ver e ter acesso as diferentes tipos de informação relacionadas com o projeto	conhecer melhor o mesmo	4

Tabela 4.8: *User Stories* para o módulo Projects

ID	Como	quero	para	Prioridade
E1	User	ver e ter acesso a informação relativa a um sprint específico	conhecer mais detalhes sobre um projeto	4

Tabela 4.9: *User Stories* para o módulo Sprints

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 abstração para facilitar a eventual mudança de ferramentas externas, evitando mudanças na camada superior do código
N4	As chaves de acesso a 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 especializac ao da empresa nessa linguagem
N7	O Sistema de Sistema de Gestão de Base de Dados (SGBD) pode ser um dos seguintes: PostgreSQL, PostGIS, MongoDB ou Cassandra
N8	As bibliotecas externas devem ser *open-source*

Tabela 4.10: Requisitos Não-Funcionais

4.2 Arquitetura

4.2.1 Estrutura Geral do Sistema

Como já enunciado, o UIS foi criado para dar suporte às necessidades da empresa, não só a curto como a longo prazo. Uma vez que as necessidades atuais da empresa não serão as necessidades do futuro, tornou-se essencial a criação de um sistema o mais flexível possível que permitisse a adição de novas funcionalidades sem que prejudicasse as funcionalidades existentes. Assim sendo, optou-se pela criação de uma arquitetura "o mais modular e *loosely coupled* possível"[29]. Sistemas com este tipo de arquitetura são constituídos por módulos independentes que podem ser ligados entre si, sem que a substituição ou adição de um novo módulo afete o resto do sistema.

Além disso, o sistema deveria também de dar suporte a diferentes tipos de clientes, *web* e *mobile*. De modo a facilitar a criação de diferentes clientes, tornando-os agnóstico ao sistema, decidiu-se que o sistema seria disponibilizado através de uma API RESTful. Como o seu nome indica, uma API consiste numa *interface* que permite a utilização de funcionalidades desenvolvidas, sem que se conheça o código fonte. Para isso são definidos um conjunto de princípios que mostram o que se deve fazer para usar determinada funcionalidade. Neste caso específico a informação é disponibilizada através do protocolo Hyper Text Transfer Protocol Secure (HTTPS) e é apresentada nos formatos JSON, XML e YAML.

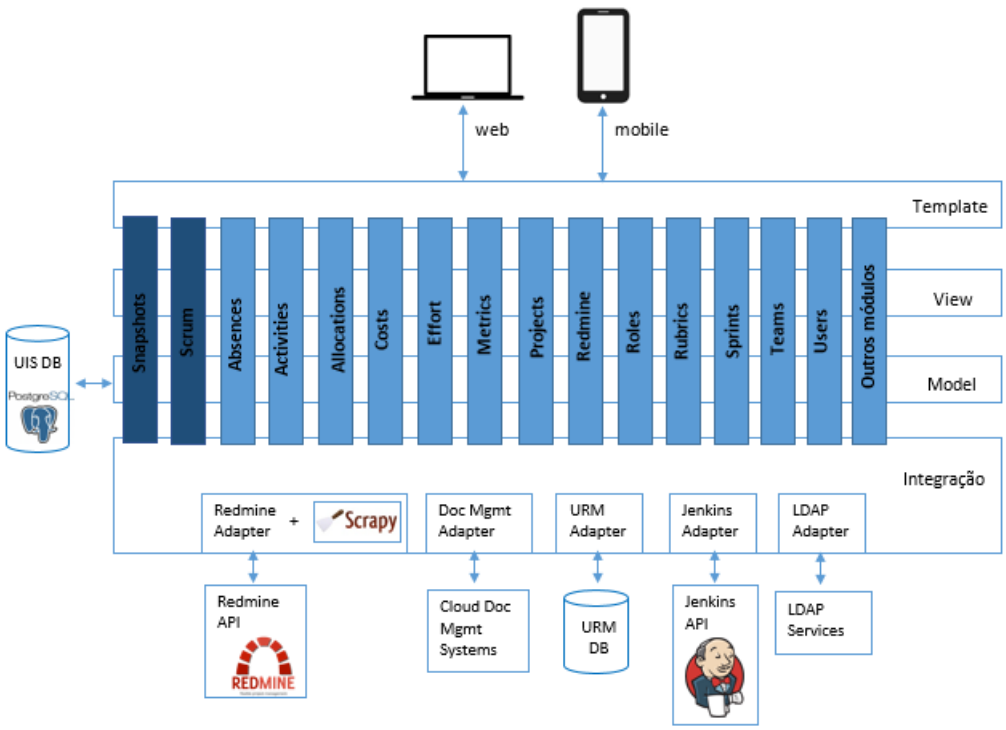


Figura 4.1: Arquitetura do Sistema de Informação da Ubiwhere

Na imagem 4.1 encontra-se a arquitetura de alto-nível do sistema. A imagem apresentada foi baseada na arquitetura original [29]. A azul escuro estão representados os módulos criados de raiz, o módulo Scrum e o módulo Snapshots, e a azul claro todos os outros módulos que já existiam e foram necessários para a realização do projeto. Mais informação sobre as funcionalidades de cada um dos módulos apresentados encontra-se na subsecção Módulos.

Como se pode observar na figura 4.1, o sistema encontra-se dividido em diferentes camadas:

- **Integração**: Na camada Integração é feita a *interface* com os diferentes sistemas utilizados pela Ubiwhere. No âmbito do projeto, destaca-se o Redmine e o Google Sheets. Relativamente à ferramenta Redmine, a integração foi feita utilizando a API fornecida pela mesma e através da ferramenta Scrapy, ferramenta analisada na secção Estado da Arte. Quanto à informação proveniente das folhas de cálculo Google Spreadsheets, foi possível a sua gestão através da biblioteca *gspread*. Além disso, através da configuração de vários *scripts* foi possível fazer a comunicação entre o sistema e o Google Sheets.

- **Model**: Representa a camada de acesso a dados. É nesta camada que é feito o mapeamento com a Base de Dados utilizada. Como se pode ver na imagem 4.1 a base de dados utilizada foi a PostgreSQL .

- **Template**: Camada de apresentação dos dados. No caso do UIS, corresponde à informação apresentada pela API.

- **View**: Camada que contém a lógica de negócio. Faz a ponte entre a camada Model e Template ficando responsável por preparar toda a informação requerida para que posteriormente possa ser apresentada. No caso do UIS, esta camada é desenvolvida utilizando a *framework* Tastypie. No subcapítulo Tecnologias encontra-se uma breve descrição dessa *framework*.

  O sistema encontra-se divido nestas camadas pois para a estruturação do código optou-se pela utilização da *framework* Django, uma *framework* tencionada para o desenvolvimento rápido para *web* que utiliza como base a linguagem Python. Esta *framework* é baseada no padrão MVC contudo os criadores do Django têm um ponto de vista diferente relativamente à nominação das camadas, afirmando que segue um padrão MTV (Model Template View). Estas camadas referem-se às camadas assim denominadas na imagem 4.1.

4.2.2 Estrutura do Sistema Detalhada

Nesta secção é feita a apresentação detalhada do sistema. Esta secção não só apresenta as sub-camadas das camadas apresentadas na imagem 4.1 como apresenta como é feita a comunicação entre as mesmas.

Antes de mais, visto que para a construção da camada View foi utilizada a *framework* Tastypie, é necessário fazer uma breve apresentação da mesma. Tastypie consiste numa *framework* desenvolvida em Python utilizada para a construção de API's RESTFUL. Esta *framework* assenta sobre os modelos definidos no Django permitindo total controlo do que é apresentado. Além disso, por *default* contém um conjunto de métodos implementados para todas as ações RESTful (GET - obter, POST - criar, PATCH - editar, DELETE – eliminar, entre outras) aos quais se pode fazer *override* de modo a fazer o tratamento da informação pretendida.

O coração desta *framework* são os *Resources*, sendo um *Resource* uma representação de dados semelhantes, quer seja correspondente a um modelo do Django ou simplesmente uma forma semelhante de armazenamento dos dados. Cada módulo enunciado é constituído por um ou mais *Resources* sendo que cada um se encontra associado a um nome único, através do qual é feita a comunicação com a API.

Associados aos *Resources* encontram-se dois conceitos fundamentais, o ciclo Hydrate e o ciclo Dehydrate. O Tastypie utiliza o ciclo Hydrate para tornar os dados serializados enviados pelo cliente em informação que possa ser utilizada pelo modelo. Já o ciclo Dehydrate transforma os dados existentes no modelo em dados mais simples para consumo do utilizador. Resumindo, pode dizer-se que os *Resources* que são enviados para o cliente são "dehydrated"e

serializados. Já a informação proveniente do cliente é "hydrated"e transformada em objetos *Resource*. Após breve explicação de como funciona a framework Tastypie, procede-se agora à apresentação e explicação da imagem referente à arquitetura detalhada do sistema que corresponde à imagem 4.2.

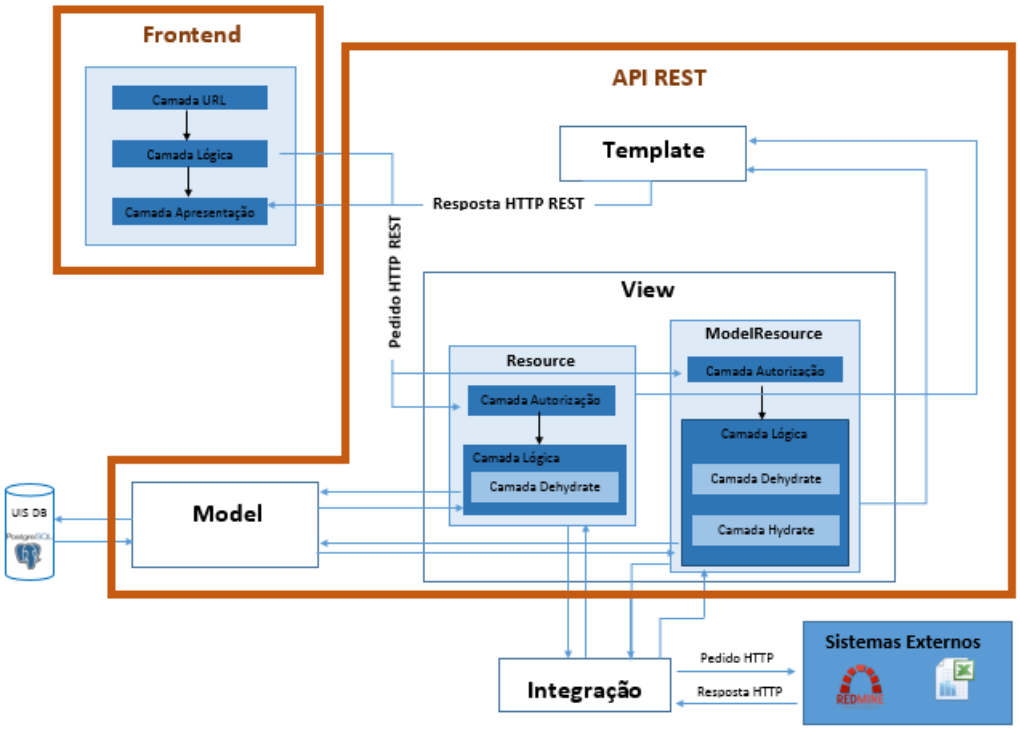


Figura 4.2: Arquitetura do Sistema de Informação da Ubiwhere (detalhada)

Frontend

Começando pela camada Frontend, que corresponde ao cliente Web e onde é feita a interface com o cliente, pode observar-se que esta se divide em 3 sub-camadas, a camada URL, a camada Lógica e a camada Apresentação.

A primeira camada, URL, consiste na camada que verifica se o URL inserido pelo utilizador se encontra definido na aplicação. Em caso negativo, surge um erro no *browser* avisando que este não existe, caso contrário é feito o mapeamento do mesmo para a respetiva secção da camada Lógica. É nesta camada que se encontra toda a lógica associada ao Frontend. Isto é, é nesta camada que é feita a comunicação com a API, através de pedidos HTTP REST aos *Resources* definidos pela mesma, e o tratamento da informação que será apresentada ao utilizador. Após toda a informação recolhida, esta é enviada para a camada Apresentação que mostra o resultado anteriormente obtido. Esta camada, ao contrário de todas as outras que se encontram definidas em Python, encontra-se desenvolvida em HTML, Javascript e CSS.

API RESTful

Passando agora a explicação de como funcionam as camadas relativas ao servidor e respetiva API. Como já enunciado, a comunicação do cliente com o Servidor é feita através de pedidos HTTP REST aos *Resources* definidos pela API. Estes *Resources* encontram-se definidos na **camada View**.

Na imagem encontram-se representados os dois tipos existentes: *ModelResource* e *Resource*. Enquanto que o primeiro se encontra associado a um modelo definido na camada Model, o segundo não. Apesar de serem provenientes de representações diferentes, ambos têm o mesmo objetivo, permitir a comunicação com a API. Independentemente do Resource "chamado" e ação realizada sobre o mesmo, começa-se por verificar se o utilizar que fez o pedido tem permissão para o fazer. A validação relativa à permissão ou não encontra-se na camada Autorização. Só após ser dada permissão pela mesma é que o pedido é processado, sendo esse tratamento realizado na camada Lógica.

Como é visível na imagem, esta camada encontra-se dividida em duas, a camada Hydrate e a camada Dehydrate. Na camada Hydrate é feito o processamento de toda informação proveniente do utilizador para que possa ser guardada no sistema, já na camada Dehydrate é feito o processamento de toda a informação que será apresentada ao utilizador. Como se pode observar, no *Resource* existe apenas uma das camadas enquanto que no *ModelResource* existem as duas. Tal fato deve-se ao fato do *Resource* não se encontrar associado a nenhum modelo, ao contrário do *ModelResource*.

Relativamente à camada Lógica, ainda é possível verificar que é através desta que existe a comunicação com a **camada Model** e com a **camada Integração**, ambas já apresentadas na secção relativa à estrutura geral do sistema. Após ser feito o processamento de toda a informação necessária esta é enviada para a **camada Template** que, envia a resposta retornada pela API para a camada de apresentação.

4.2.3 Diagramas de Sequência

Nesta secção são apresentados alguns diagramas de Sequência que mostram o fluxo entre os diferentes componentes do Sistema, desde o cliente Web, API REST e Sistemas Externos (Redmine, Google Docs). Em qualquer um dos diagramas apresentados, é possível perceber que a interação do utilizador é feita com o cliente Web, sendo este que trata dos pedidos à API através dos *endpoints* definidos na mesma. A API processa os dados, comunica com os Sistemas Externos, e após terminar as suas funções devolve ao cliente uma resposta, que apresenta situações diferentes consoante o resultado retornado pela API.

Na imagem 4.3 encontra-se representada a sequência de passos realizados para a automatização do processo Planning Poker. Como se pode observar, só um utilizador que tenha acesso ao documento pode solicitar a sua automação, caso contrário é devolvida uma mensagem de erro. Quanto à automação propriamente dita, pode verificar-se que esta é feita passo a passo, ou seja, o *update* da informação existente no *template*(exemplo na imagem C.2), no Redmine, é feito tarefa a tarefa.

Os outros diagramas são relativos à criação e exportação de informação através do UIS. Na imagem 4.4 encontra-se representado o processo de exportação de informação relativa à Sprint Review para o Redmine, enquanto que na imagem 4.5 encontra-se informação relativa à criação e exportação de um Snapshot de Sprint para os formatos PDF e XLS.

Como se pode ver na imagem 4.4, antes de se fazer a extração da informação para o Redmine é verificado se a respetiva página já existe. Em caso afirmativo, a página é atualizada com a informação relativa à Sprint, caso contrário, a página é criada sendo feita a adição da informação na mesma. A informação relativa a cada Sprint encontra-se guardada em diferentes locais, BD e Redmine, sendo alguma também calculada em *real-time*.

Relativamente à criação de Snapshots de Sprint, pode verificar-se que, independentemente do formato de extração escolhido, toda a informação relacionada com a Sprint é adicionada na BD assim como é guardado o respetivo ficheiro no servidor HTTP NGIX, servidor onde se encontra alocado o projeto UIS. Contudo, existem diferenças relativamente à extração da informação nos diferentes formatos. Enquanto que para a criação de Snapshots no formato XLS é feita a extração da informação anteriormente guardada na BD, para a criação dos Snapshots em formato PDF é feita a extração dos *screenshots* existentes no cliente Web que contém a informação pretendida.

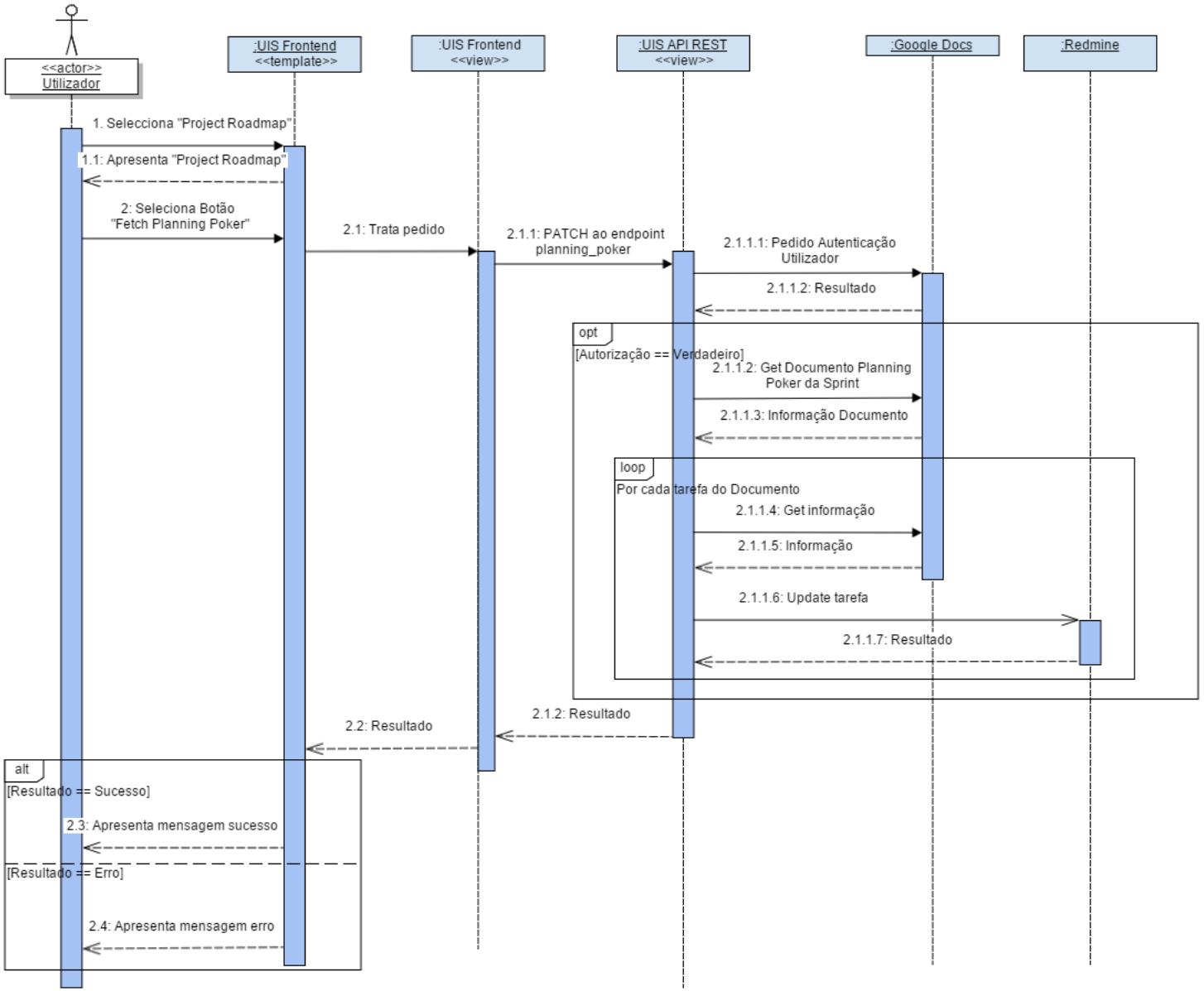


Figura 4.3: Diagrama de Sequência da US L1 - Utilização de um Processo de Planning Poker

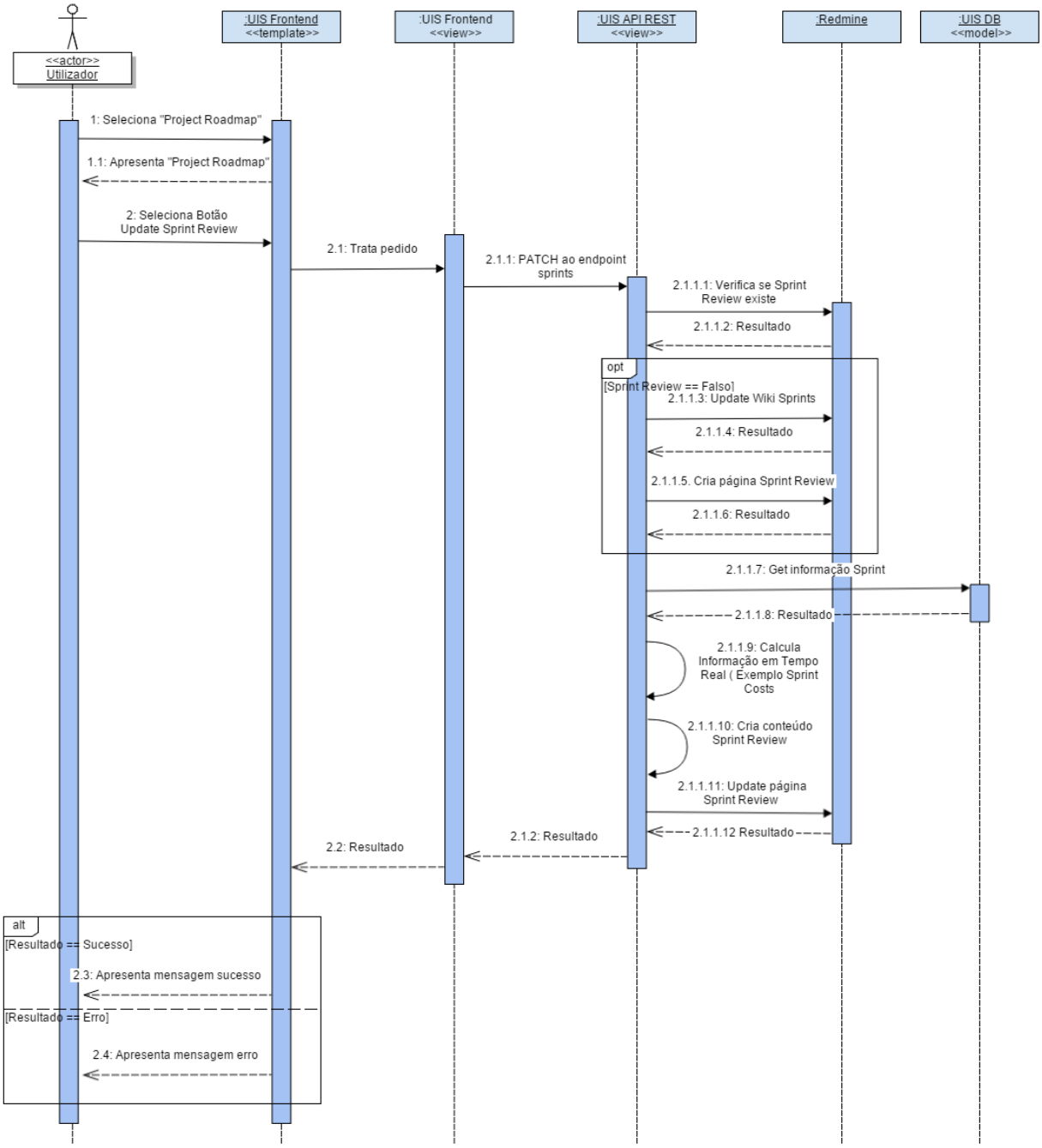


Figura 4.4: Diagrama de Sequência da US L7 - Exportar Wiki Review para o Redmine

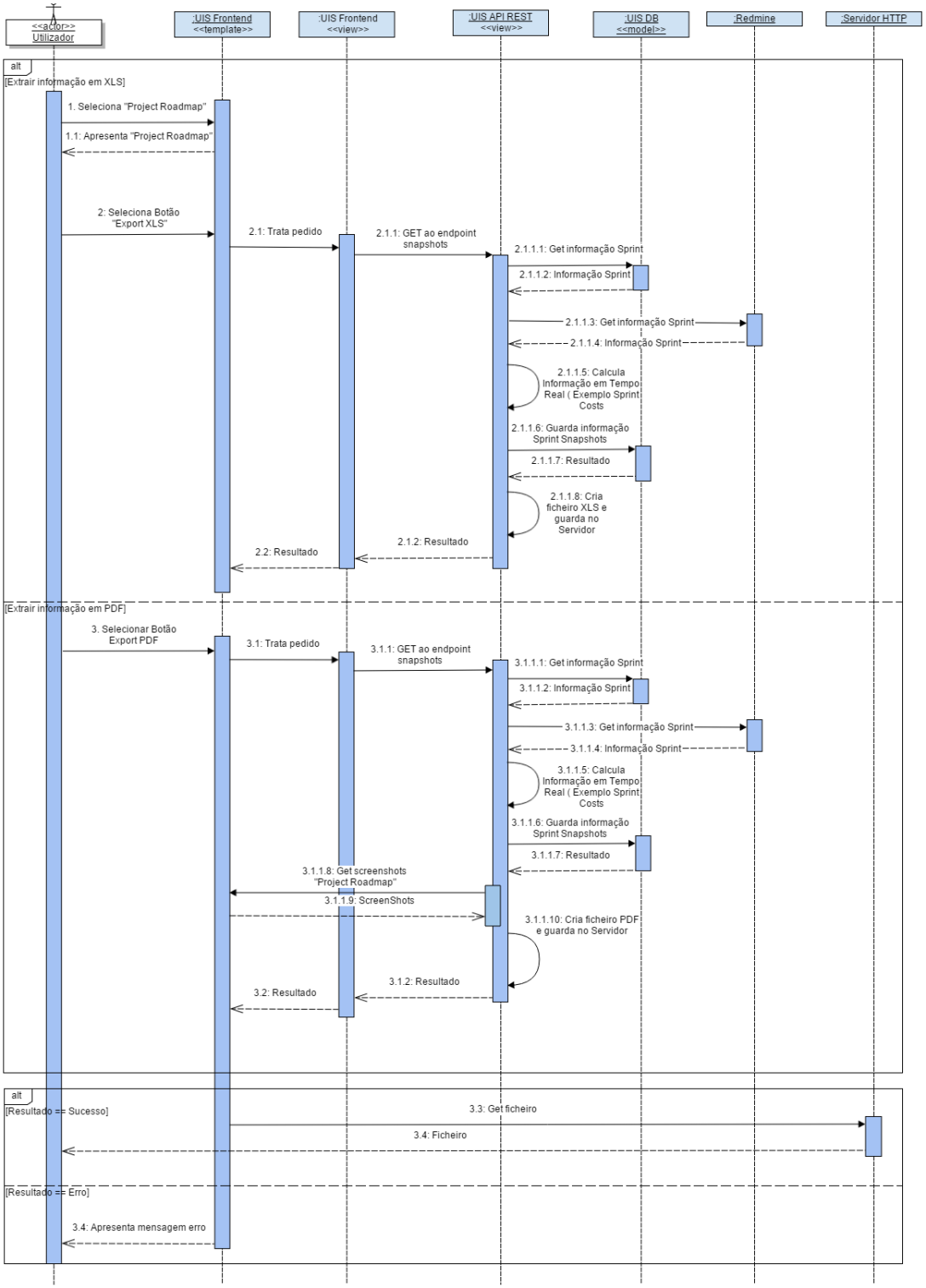


Figura 4.5: Diagrama de Sequência da US I2 - Exportar toda a informação da Sprint

4.2.4 Módulos

De seguida encontra-se uma breve explicação dos módulos que constituem o Sistema de Informação da Ubiwhere e que se encontram relacionados com o projeto, encontrando-se a negrito os módulos que se criaram de raiz no âmbito do mesmo:

- Absences - módulo onde se encontra toda a informação relacionada com as faltas dos utilizadores.
- Activities - módulo que contém informação relativas às atividades dos projetos. Em anexo encontra-se informação relativa às mesmas.
- Allocations - módulo onde é guardada toda a informação relativa às alocações de cada utilizador ou equipa. Com a informação presente neste módulo é possível verificar a disponibilidade dos utilizadores possibilitando a atribuição de tarefas consoante essa disponibilidade.
- Costs – módulo que contém informação relativa a custos associados a cada projeto.
- Effort – módulo onde é guardada informação relativa ao tempo despendido em cada tarefa, por cada utilizador.
- Metrics – módulo muito importante para a gestão de projetos, onde é calculado todo o conjunto de métricas definidas pela empresa para fazer a avaliação dos seus projetos.
- Projects – módulo que recebe e guarda todas as informações relativas a um projeto, informações estas que se encontram na ferramenta Redmine.
- Redmine – módulo que integra o UIS com o Redmine sendo possível a obtenção de informações existentes neste através da API do Redmine.
- Roles - módulo onde é guardada informação relativa ao papel de cada utilizador na organização e em cada projeto.
- Rubrics - módulo que contém informação sobre as rúbricas de um projeto. Por rúbricas entende-se os vários tipos de gastos necessários para a realização do projeto, como por exemplo Hardware, Recursos Humanos.
- **Scrum** - módulo onde se encontram as funcionalidades referentes à metodologia Scrum.
- **Snapshots** - módulo que contém informação relativa à criação de Snapshots.
- Sprints – módulo que contém informação relativa aos Sprints de um projeto.
- Users - módulo que contém a informação sobre os utilizadores da Ubiwhere.
- Teams - módulo que contém a informação sobre as equipas da Ubiwhere.

4.2.5 Modelo de Dados

Como se pode ver na figura 4.1, o SGBD utilizado foi o **PostgreSQL**. A comunicação do sistema com o SGBD escolhido foi feita utilizando o sistema de Object-Relational Mapping (ORM) existente no Django. Este sistema permitiu a utilização de uma simples *interface* para a interação com a Base de Dados (BD) ao invés da criação de comandos na linguagem SQL. Entrando em mais detalhe, cada modelo foi mapeado numa tabela cujas colunas correspondiam a cada um dos atributos do modelo.

Com o intuito de tornar mais percetível a relação entre os diferentes módulos enunciados, optou-se por apresentar o modelo Entidade-Relacionamento do projeto. Este encontra-se dividido em diferentes imagens (4.6, 4.7, 4.8, 4.9, 4.10 e 4.11) e apenas apresenta as entidades significativas para o projeto. Para diferenciar o que já existia e o que foi necessário adicionar / eliminar para ir de encontro com os objetivos do projeto, foram definidas diferentes cores para a sua representação. A azul encontra-se o que foi adicionado e a vermelho o que foi eliminado e atualmente já não é utilizado. De notar que foram alterados não só alguns atributos mas também foram adicionadas novas tabelas na BD (tabelas estas que se encontram com margem azul).

Como se pode observar nas diferentes imagens, um módulo pode ser constituído apenas por uma ou por diferentes tabelas na BD. Na imagem 4.6, encontram-se representados os módulos *Metrics*, *Project*, *Sprint*. Na imagem 4.7 é possível observar como se conectam os módulos *Users*, *Absences* e *Allocations*. Na imagem 4.8 encontra-se a relação entre os módulos *Costs*, *Rubrics*, *Roles*, *Users*, *Projects*, *Teams* e *Efforts*. Por fim, nas imagens 4.9, 4.10 e 4.11 encontra-se informação relacionada com o módulo Snapshots, encontrando-se divididas pelos diferentes tipos de Snapshots, Snapshot de Sprint, Snapshot de Projeto e Snapshot de todos os projetos da Ubiwhere.

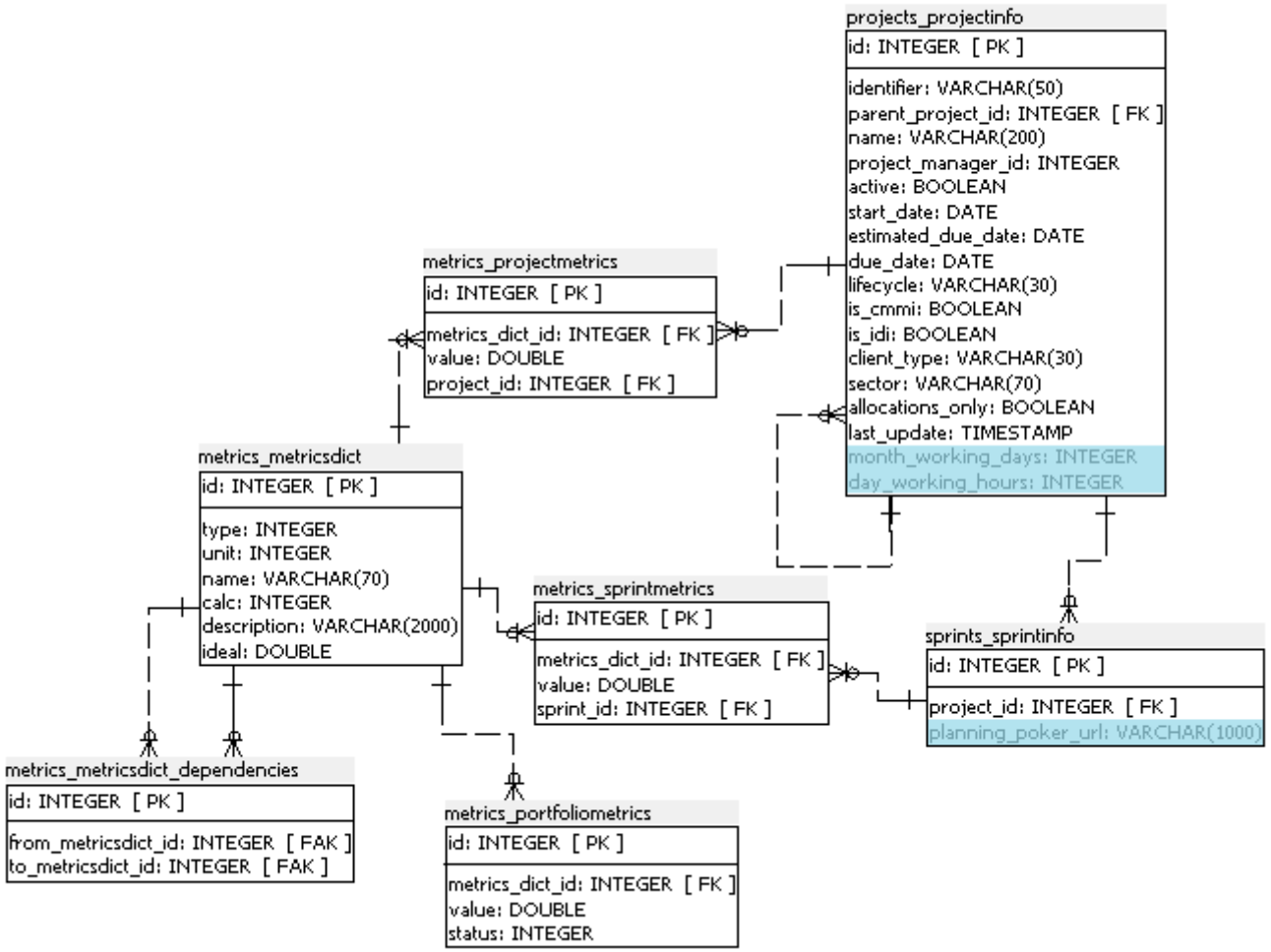


Figura 4.6: Diagrama ER: Módulos *Metrics*, *Project*, *Sprint*

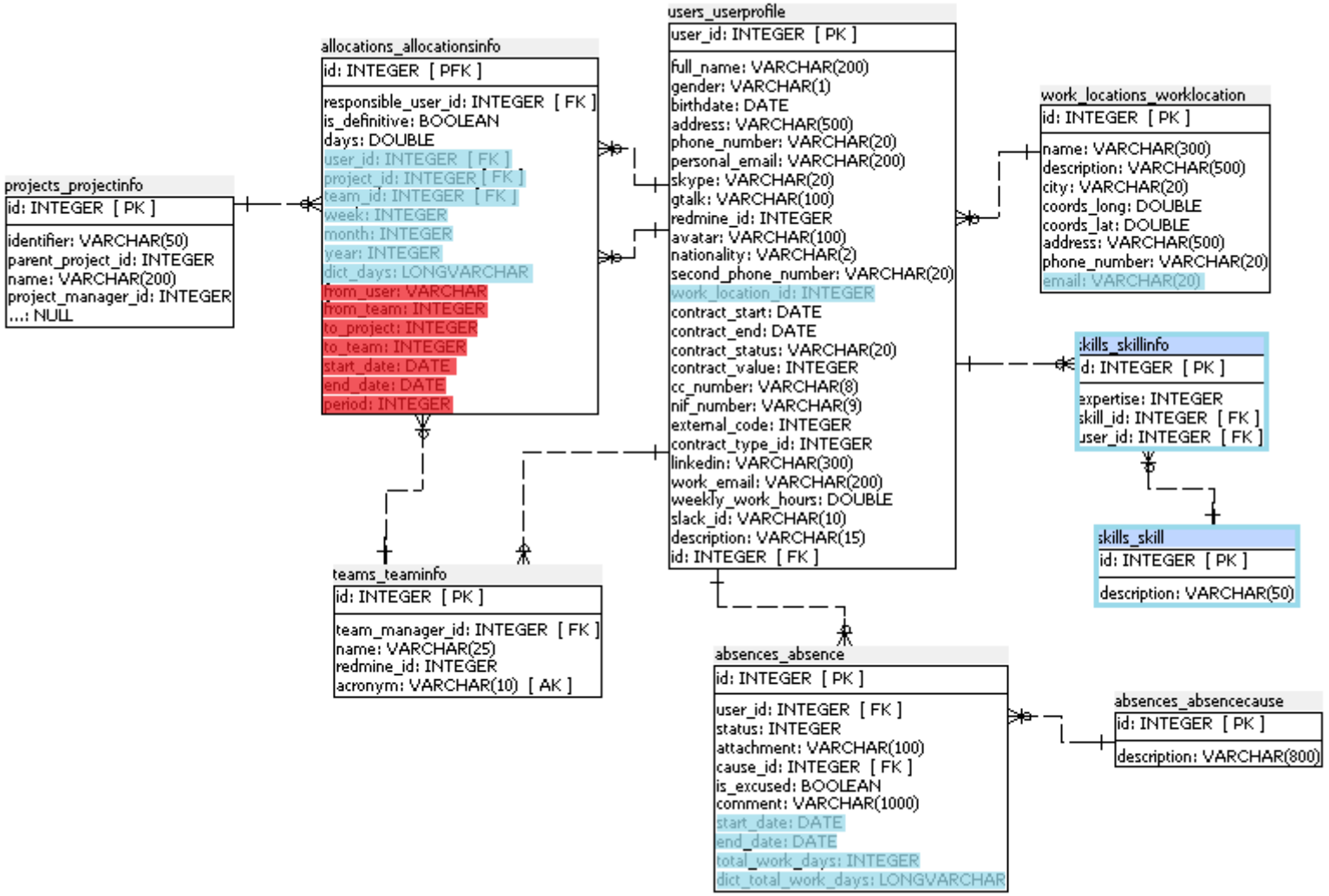


Figura 4.7: Diagrama ER: Módulos *Users*

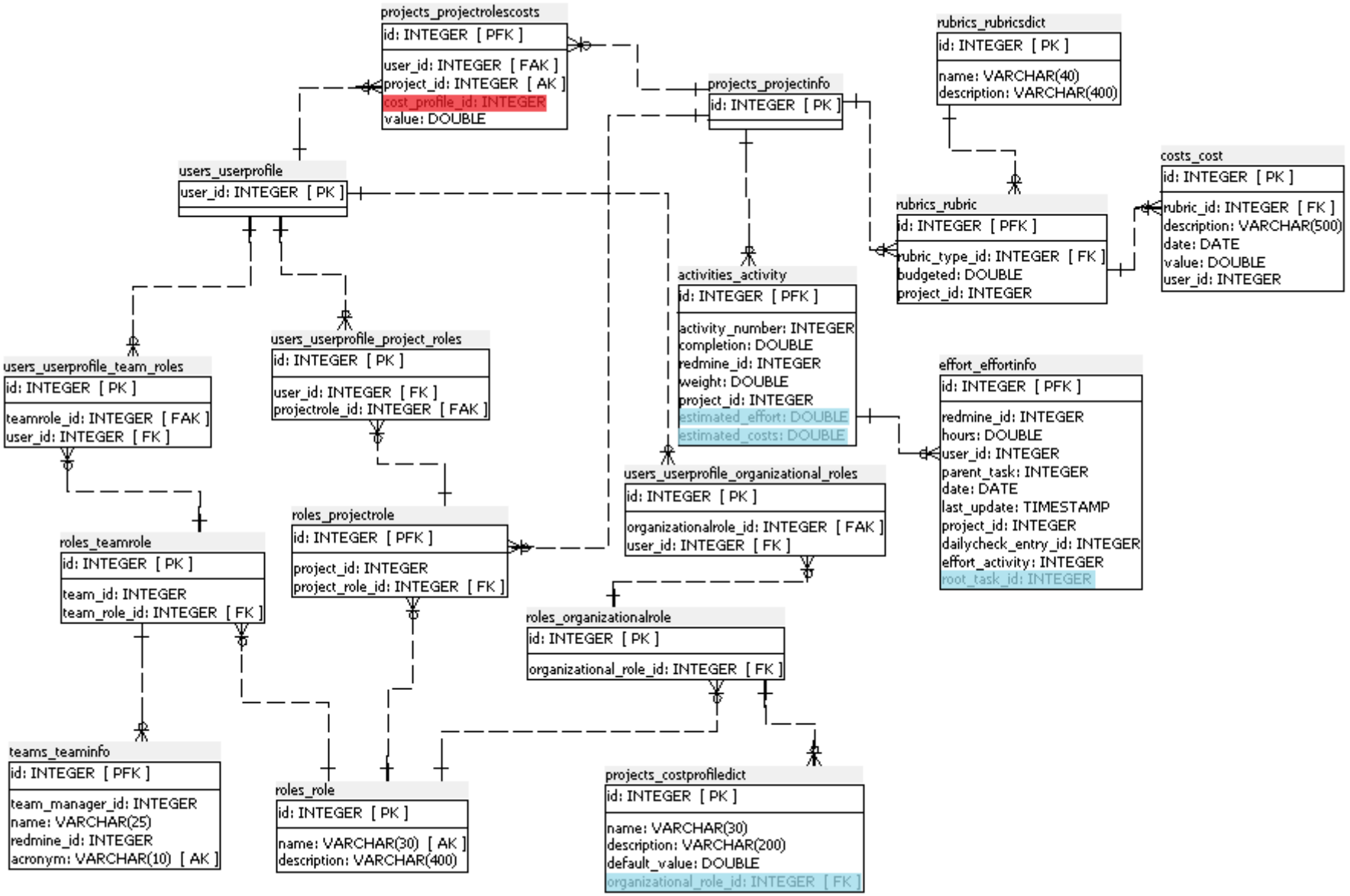


Figura 4.8: Diagrama ER: Módulos *Costs*, *Rubrics*, *Roles*

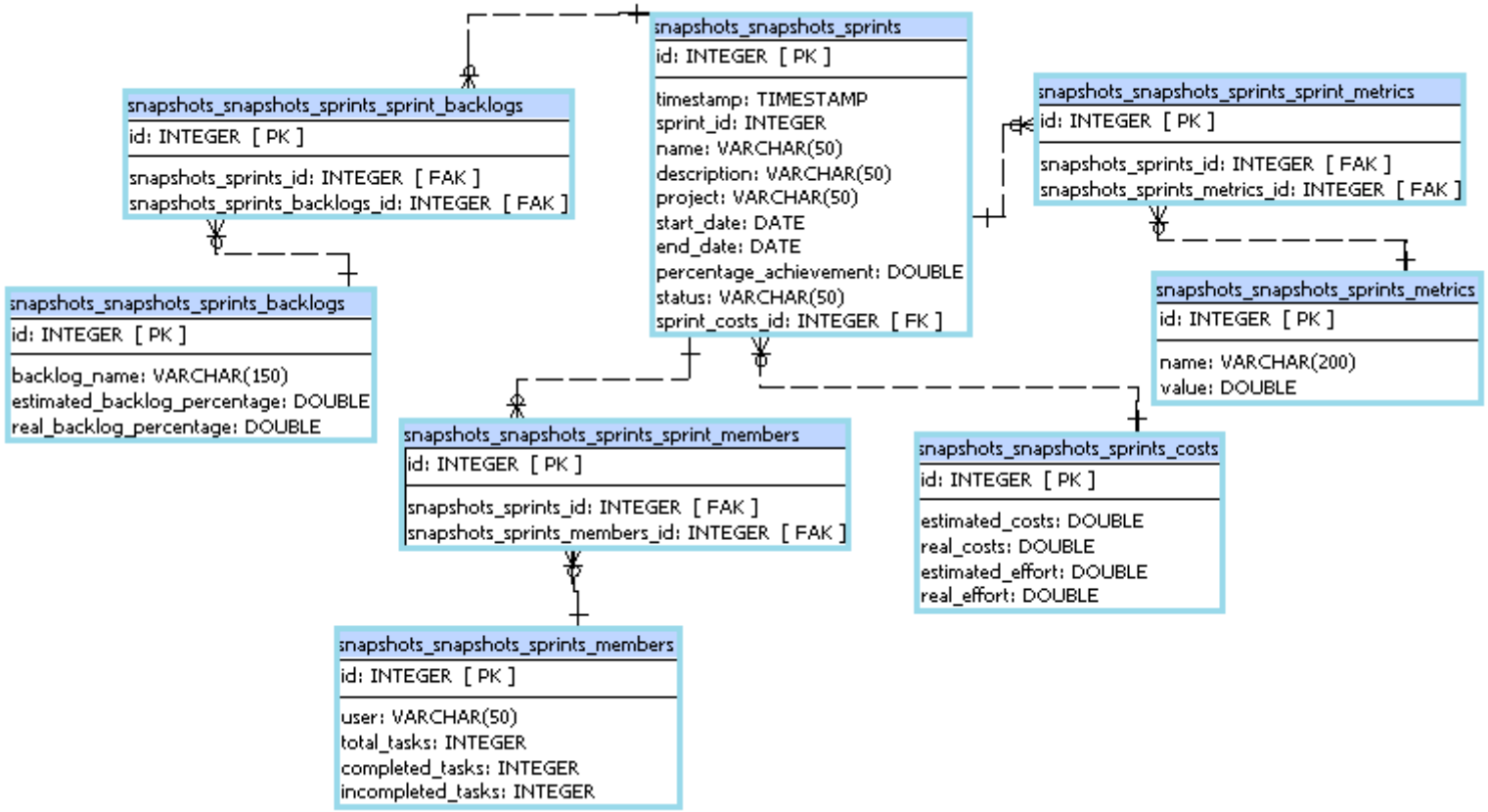


Figura 4.9: Diagrama ER: Módulo *Snapshots* (*Snapshots de Sprint*)

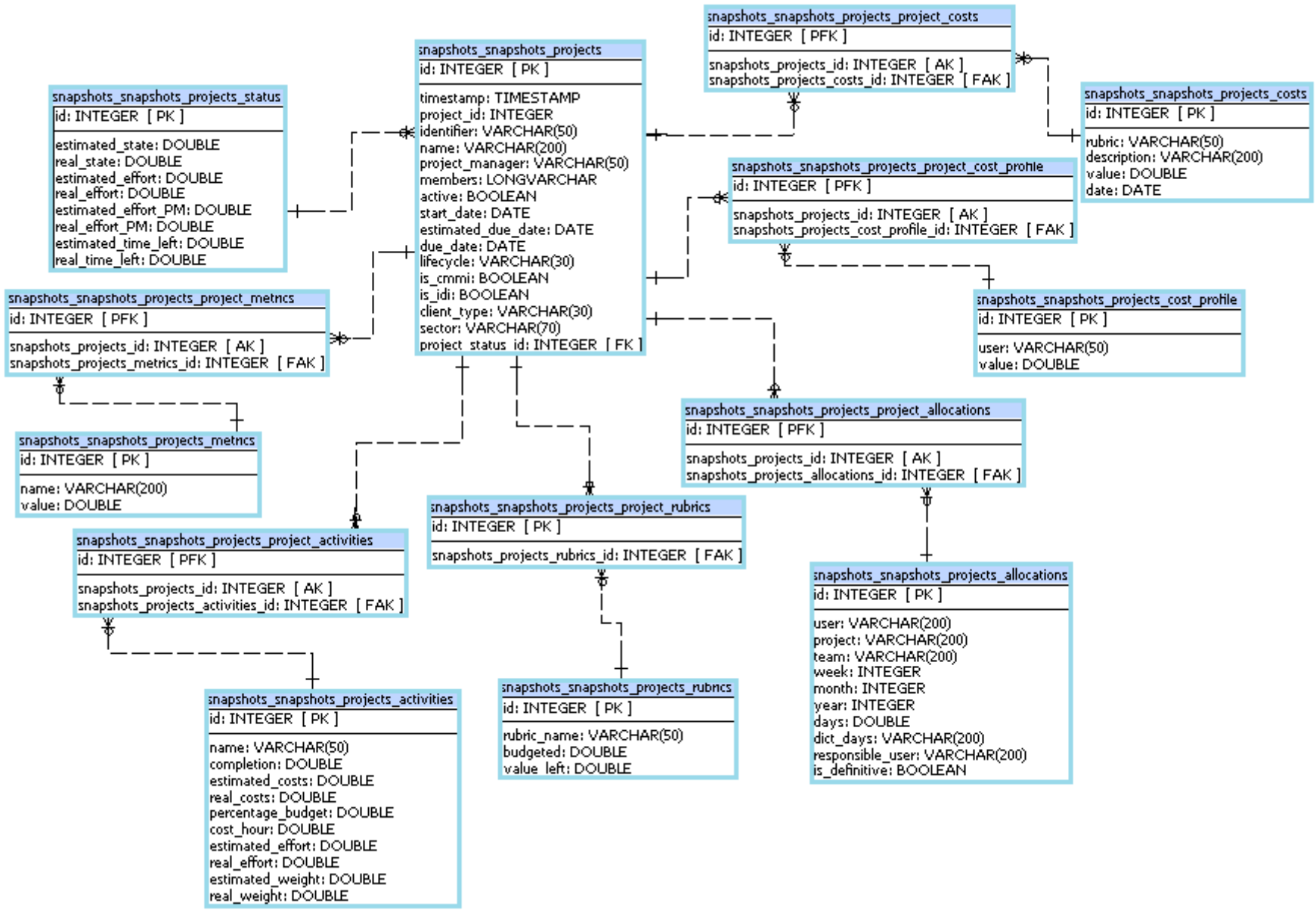


Figura 4.10: Diagrama ER: Módulo *Snapshots* (*Snapshots de Projeto*)

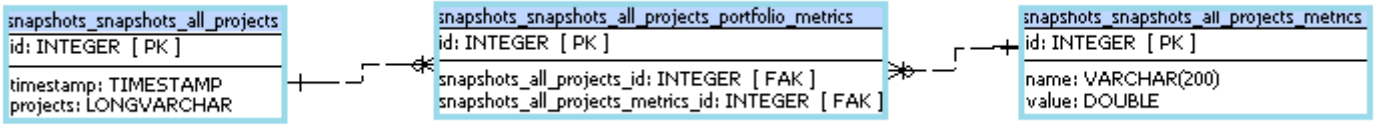


Figura 4.11: Diagrama ER: Módulo *Snapshots* (*Snapshots de todos os Projetos*)

Capítulo 5

Implementação

Neste capítulo encontra-se informação relativa à obtenção de informação e respetiva implementação realizada no âmbito do projeto.

Esta secção pretende mostrar o trabalho implementado, encontrando-se dividida pelos diferentes momentos da metodologia Scrum onde se inserem:

- Planeamento da Sprint
- Durante a Sprint
- Revisão da Sprint

Toda a informação apresentada de seguida foi implementada no servidor do UIS e encontra-se acessível através do cliente Web.

5.1 Planeamento da Sprint

5.1.1 Listagem Users e Skills

Numa fase de planeamento de um projeto ou de uma Sprint é necessário que o PM selecione os recursos mais adequados para a realização do mesmo. Apesar desta informação ser útil e importante para a empresa, esta não se encontrava definida em nenhum local. Assim sendo, o modelo do UIS foi modificado permitindo que cada utilizador tivesse associado um conjunto de Skills e qual o seu grau de *expertise*. As modificações realizadas podem ser observadas na imagem 4.7.

Além da modificação do modelo do UIS, foram adicionadas novas "vistas"no Frontend.

5.1.2 Alocações

Na Ubiwhere, as alocações de recursos a projetos são feitas manualmente pelo PM de cada projeto. Existem diferentes tipos de alocações, semanais e mensais, que provêm de *templates* diferentes, mas cujo objetivo final é o mesmo, indicar o número de dias que cada recurso se encontrará alocado a um projeto. Cada *template* corresponde a um ficheiro xls e encontra-se alojado no Google Docs, podendo ser facilmente acedido.

O trabalho realizado no módulo Alocações pode dividir-se em duas fases:

1. Alteração e adaptação do módulo Alocações existente.

   Apesar do módulo Alocações já existir no iniciou do estágio, este encontrava-se bastante complexo e confuso. Foi pedido então ao estagiário, tendo em conta alguns requisitos,

que alterasse o modelo atual. No modelo de dados apresentado na figura 4.7 encontram-se as alterações feitas ao mesmo. Ao fazer estas alterações, coube ao estagiário a adaptação do código existente, garantindo que este não deixaria de funcionar devido à mudança do modelo das Alocações.

2. Importação automática das alocações existentes nos *templates* para o UIS.

   Para fazer a importação das alocações existentes para o UIS foram criados um conjunto de *Scripts* que permitiram a interação com a API do UIS. Estes *Scripts* já se encontravam inicializados, tendo sido feito primeiro a análise e validação dos mesmo. Após foi feita a correção e adição de novas funcionalidades, como por exemplo obter não só o número de dias mas os dias em que o recurso estará alocado.

Quanto às alocações, por parte do estagiário, apenas houve alterações relativamente ao servidor e respetiva API, tendo as funcionalidades no cliente sido adicionadas por um programador de Frontend da empresa. Tal situação ocorreu devido à urgência de sua implementação no cliente e o estagiário se encontrar alocado a outros requisitos.

5.1.3 Automação processo Planning Poker

Após conversa com o *PM* do projeto, decidiu-se que o *template* Planning Poker iria ser alocado no Google Docs, facilitando assim o seu acesso por parte dos utilizadores. Na imagem 4.3 é possível verificar como é feita a automação do processo Planning Poker e com o intuito de clarificar a mesma, encontra-se em anexo o exemplo de um *template* Planning Poker.

Para a análise e obtenção da informação proveniente do *template* foi utilizada a biblioteca Gspread e para fazer o *update* dessa informação no Redmine foi utilizada a biblioteca python-redmine. Informação relativa a estas bibliotecas encontra-se na secção Bibliotecas externas utilizadas.

5.2 Durante a Sprint

Um dos objetivos do projeto era permitir a visualização fácil e percetível do estado de um *Sprint*. Para isso foi definido que um dos objetivos do projeto seria a utilização de uma Taskboard para a gestão das tarefas dos utilizadores. Para a satisfação deste objetivo poderia utilizar-se um dos *plugins* enunciado na secção Estado da Arte. Contudo, percebeu-se que, devido à estrutura de tarefas utilizada pela Ubiwhere, este objetivo não era exequível, pelo menos na maneira anteriormente pensada pela empresa. Assim, foi feito o refinamento deste objetivo, passando de nível 4 para nível 2 no que toca às prioridades da empresa.

5.3 Revisão da Sprint

O trabalho realizado durante o estágio incidiu maioritariamente na obtenção e exportação de informação útil e necessária para a reunião de revisão da Sprint.

5.3.1 Obtenção informação

No âmbito deste projeto, foi necessário obter diferentes tipos de informação relacionadas com projetos, sprints e mesmo atividades dos projetos. Ao contrário das métricas, que se encontravam documentadas e definidas, este tipo de dados não tinha qualquer documentação que explicasse no que consistiam e como se poderiam obter. Coube ao estagiário perceber como

se obter cada uma, encontrando-se nas secções seguintes uma breve explicação e fórmulas utilizadas para o seu cálculo.

5.3.1.1 Validação e implementação de Métricas

Nesta secção encontra-se uma breve explicação do trabalho desenvolvimento relativamente ao módulo Metrics. Antes de mais, foi feito um estudo de todas as métricas de Sprint para perceber no que consistiam e como se calculavam. Estando o estudo feito, foi possível analisar várias Sprints, algumas existentes, outras criadas para o efeito, de modo a fazer a sua validação e verificar se o código até então implementado se encontrava de acordo com o esperado. Foram escolhidos diferentes Sprints pois representavam diferentes situações o que permitia uma maior cobertura de casos possíveis, permitindo assim uma validação mais eficaz. Na secção Testes encontra-se explicado em maior detalhe os testes realizados.

5.3.1.2 Cálculo de Esforço e Custos

Nesta secção encontra-se informação relativa ao cálculo de Esforço e Custos que a Ubiwhere necessita de conhecer, quer seja relativa a Sprints, Atividades ou Projetos. Pode afirmar-se que estas duas variáveis se encontram diretamente relacionadas, implicando que um aumento do esforço exigido pelo projeto origine um aumento do custo do mesmo. Como representam variáveis que se encontram em constante mudança, optou-se por efetuar o seu cálculo em *Real-Time*.

Relativamente ao Esforço, este segue a mesma lógica quer em Sprint, Projetos e Atividades, mudando apenas o intervalo de valores necessário para o seu cálculo. No caso de Sprint, o seu cálculo é feito tendo em conta as tarefas do próprio Sprint. Quanto ao esforço gasto em cada atividade é tido em conta as tarefas que descendem da mesma. Por último, para conhecer o esforço de cada projeto são selecionadas as tarefas de todas as atividades do projeto que se encontram entre o início e fim do projeto.

O Redmine, tendo em conta o *Effort* registado por cada utilizador nas diferentes tarefas existentes, calcula em *Real-Time* o esforço total do projeto e de cada atividade. Tal informação poderia ser obtida através da ferramenta Scrapy, facilitando a sua aquisição. Contudo, ao analisar melhor esta questão o estagiário percebeu que a Ubiwhere queria apenas o esforço tendo em conta a data de início e fim do mesmo assim como apenas de tarefas que se encontram atribuídas a alguma das atividades existentes. Apesar de não ser suposto existirem projetos com *efforts* registados antes da data inicial e sem atividades atribuídas, a verdade é que isso acontece (teste existente na tabela 7.6). Assim, o estagiário optou por fazer o cálculo do mesmo e caso se verificasse alguma dessas situações apresentar uma mensagem, alertando o PM Manager para que pudesse resolver essa situação.

Tendo em conta o modelo existente, a maneira como se poderia obter o esforço de cada atividade não era a mais eficaz. Seria necessário calcular, sempre que se queria saber o esforço, para cada registo, a atividade a que pertencia. De modo a facilitar essa informação, o estagiário optou por modificar o modelo existente, acrescentando na tabela *Effort* o campo *root_task* (imagem 4.8). Assim, apenas aquando a inserção dos registos dos esforços no UIS era necessário obter tal informação.

Diferente do cálculo do esforço, é o cálculo de custos. Apesar de também utilizarem a mesma relação, existe diferenças entre eles. Através do modelo existente não era possível obter informação relativa ao custo de cada utilizador. Após estudo, e tendo em conta o modelo existente, foi sugerido pelo estagiário uma modificação ao mesmo, que foi validada pelo PM do respetivo projeto. Na imagem 4.8 encontram-se as modificações efetuadas.

A informação relativa às fórmulas utilizadas para o cálculo do esforço e custo das Sprints, Atividade e Projetos encontram-se nas secções seguintes.

5.3.1.3 Cálculo informações Sprints

Nesta secção encontra-se informação relativa ao cálculo de informações relacionadas com os Sprints.

Esforço da Sprint

- Estimado:

$$ES_e = \sum_{t_s=1}^{n} Ee_{ts}$$

, onde Ee_{ts} corresponde ao esforço estimado de uma tarefa na Sprint

- Real:

$$ES_r = \sum_{t_s=1}^{n} \sum_{e_{ts}=1}^{m} Er_{ets}$$

,onde Er_{ets} corresponde ao esforço de uma tarefa na Sprint

Custo da Sprint

- Estimado:

$$CS_e = \sum_{t_s=1}^{n} Ee_{ts} * C_{ts}$$

, onde Ee_{ts} corresponde ao esforço estimado de uma tarefa na Sprint e C_{ts} corresponde ao custo da mesma.

- Real:

$$CS_r = \sum_{t_s=1}^{n} \sum_{e_{ts}=1}^{m} Er_{ets} * C_{ets}$$

, onde Er_{ets} corresponde ao esforço de uma tarefa na Sprint e C_{ets} corresponde ao custo da mesma.

Variação do Sprint Backlog

Durante a realização de cada Sprint, os Backlogs em que este se encontra associado vão modificando a sua percentagem de conclusão, contudo esta informação é relativa a todas as Sprints realizadas até ao momento não sendo possível conhecer qual a variação de cada Sprint em particular. Assim, era necessário o seu cálculo, não só o estimado, que representava o valor total que iria modificar caso as tarefas fossem todas concluídas, mas também a percentagem real.

Primeiro foi necessário fazer alguns testes com o intuito de perceber como esse valor era obtido pelo Redmine e se proceder à sua implementação. Na tabela 7.8 é possível verificar alguns dos alguns testes realizados.

Concluiu-se que para a obtenção do *Sprint Backlog* era necessário, por cada tarefa do respetivo *Sprint* obter as *US* que se encontravam envolvidas e por cada uma calcular o peso na respetiva US.

- Estimado:

$$SBV_e = \sum_{t=1}^{n} \sum_{U_t=1}^{m} \frac{W_{Ut}}{D_{Ut}}$$

, onde W_{Ut} representa o peso da US e D_{Ut} representa o número de descendentes da US.

- Real:

$$SBV_r = \sum_{t=1}^{n} \sum_{U_t=1}^{m} P_t * \frac{W_{Ut}}{D_{Ut}}$$

, onde P_t representa a percentagem de conclusão da tarefa, W_{Ut} representa o peso da US e D_{Ut} representa o número de descendentes da US.

5.3.1.4 Cálculo informações Atividades

Já nesta secção encontra-se informação relativa ao cálculo de informações relacionadas com as diferentes atividades de um Projeto.

- Esforço Estimado

$$C_a = \sum_{t_a=1}^{n} Ee_{ta}$$

, onde Ee_{ta} corresponde ao esforço estimado de uma tarefa na Sprint

- Esforço Real

Para o cálculos do esforço real gasto numa atividade, foi necessário obter todos os *Efforts* registados nas tarefas pertencentes a cada atividade (E_{eta}).

$$H_a = \sum_{t_a=1}^{m} \sum_{e_{ta}=1}^{j} E_{eta}$$

- Custo Estimado

$$C_a = \sum_{t_a=1}^{n} Ee_{ta} * C_{ta}$$

, onde Ee_{ts} corresponde ao esforço estimado de uma tarefa na Sprint e C_{ts} corresponde ao custo da mesma.

- Custo Reais

Tendo em conta as horas despendidas por cada utilizador (E_{eta}) e o custo associado ao mesmo (C_{eta}), foi possível obter o custo total de cada atividade.

$$C_a = \sum_{t_a=1}^{m} \sum_{e_{ta}=1}^{j} E_{eta} * C_{eta}$$

- Percentagem de *Budget*

  A percentagem de *Budget* gasto na atividade pode ser obtido tendo em conta a relação entre o *Budget* do projeto (B_e)e os custos totais da atividade (C_a).

$$B_a = \frac{B_e}{C_a * 100}$$

- Custo /H

  Tendo o custo total da atividade (C_a) e o número de horas despendidas para a sua realização (H_a), é possível obter o Custo /H da mesma.

$$C\_H_a = \frac{C_a}{H_a}$$

- Peso Estimado da Atividade

  O peso previsto para cada atividade pode ser obtido tendo em contas os custos (C_a) e *Budget* estimado (B_e) da mesma.

$$W_{ea} = \frac{C_a * 100}{B_e}$$

- Peso Real da Atividade

  Já o peso real pode ser calculado tendo em conta os verdadeiros custos (C_a) e o Budget da atividade (B_r).

$$W_{ra} = \frac{C_a * 100}{B_r}$$

5.3.1.5 Cálculo informações Projetos

Por fim, nesta secção encontra-se informação relativa ao cálculo de informações relacionadas com os Projetos.

Estado

- Estimado

  O Estado Estimado do Projeto foi obtido tendo em conta a data prevista para a conclusão do mesmo. Assim, considerou-se que o estado estimado podia ser calculado através da relação entre os dias que passaram desde o início do projeto (N) e o número total de dias estimados (T) para a realização do Projeto.

$$S_e = \frac{N}{T} * 100$$

- Real

  O cálculo do Estado Real do Projeto foi feito tendo em conta a percentagem de execução das tarefas dos *Backlogs* do mesmo. Sabendo a percentagem de realização de cada tarefa de cada *backlog* (P_{tb}) foi possível conhecer a percentagem real até ao momento.

$$S_r = \sum_{b=1}^{n} \frac{\sum_{t_b=1}^{m} P_{tb}}{m}$$

Esforço

- Estimado

$$E_e = \sum_{a=1}^{n} \sum_{t_a=0}^{m} E_{ta}$$

, onde E_{ta} corresponde ao esforço estimado de uma dada tarefa

- Real

$$E_r = \sum_{a=1}^{n} \sum_{t_a=1}^{m} \sum_{e_{ta}=1}^{j} E_{eta},$$

, onde E_{eta} corresponde ao esforço gasto numa dada tarefa

Tempo Restante

Outra informação que era importante conhecer era o número de meses, estimados e reais, necessários para a conclusão do projeto. Para obter esta informação foi necessário fazer a relação entre o tempo e a respetiva percentagem de trabalho realizado. Conhecendo também a quantidade de trabalho que uma pessoa consegue realizar durante um mês (MH), foi possível conhecer o tempo suposto e o tempo real para terminar o projeto.

- Estimado:

$$TL_e = \frac{\frac{(100-S_e)*E_e}{S_e}}{MH}$$

, onde S_e representa o estado estimado do projeto e E_e representa o esforço estimado para a realização do projeto.

- Real:

$$TL_r = \frac{\frac{(100-S_r)*E_r}{S_r}}{MH}$$

, onde S_r representa o estado real do projeto e E_r representa o esforço real necessário até ao momento para a construção do projeto.

Budget

- Estimado

  O *Budget* Estimado foi obtido através do somatório de todos os *budgets* estimados para as rúbricas do projeto.

$$B_e = \sum_{ru=1}^{n} B_r u$$

- Real

  O Budget Real foi obtido tendo em conta todos os custos associados às rúbricas do projeto.

$$B_r = \sum_{ru=1}^{n-1} \sum_{c_r u=1} C_{cr} + \sum_{t=1}^{n} E_t * C_t$$

  , onde C_{cr} corresponde aos custos associados às diferentes rúbricas do projeto (excluído a rúbrica RH), E_t corresponde ao esforço de uma tarefa e C_t corresponde ao custo dessa tarefa (custo RH).

5.3.2 *Storage* e Extração informação

A exportação de informação foi feita de diferentes maneiras:

- Sprint Review (Redmine Wiki)
- Snapshots (nos formatos XLS e PDF)

No que toca à implementação, este foi o último módulo a ser construído pois dependia de todos os outros passos realizados anteriormente.

5.3.2.1 Sprint Review - Redmine Wiki

Na imagem 4.4 encontra-se representado o diagrama de sequência que representa o fluxo relativo à criação da página Sprint Review na Wiki do Redmine. Além disso, na respetiva secção é encontra-se uma breve explicação do mesmo. Para a criação dessa página foi seguido o *template* utilizado pela empresa, *template* este encontrado na imagem **??** em anexo.

5.3.2.2 Snapshots

Como já explicado na secção Estado da Arte, a criação de Snapshots com toda a informação relativa aos projetos da empresa era um aspeto fulcral e necessário devido à certificação da Ubiwhere com o nível 2 do CMMI.

Para satisfazer as necessidades da empresa relativa à criação de Snapshots, foi necessário modificar o modelo de dados atual, encontrando-se tais alterações nas imagens 4.9, 4.10 e 4.11. Cada imagem corresponde a um tipo diferente de Snapshot, sendo eles, respetivamente:

- Snapshot de Sprint
- Snapshot de Projeto
- Snapshot com informação de todos os projetos

Em anexo encontra-se representado um exemplo de cada tipo de Snapshot criado.

Através do diagrama de sequência, já apresentado (imagem 4.5), é possível verificar o fluxo relativo à criação de um Sprint Snapshot. Optou-se apenas por apresentar o diagrama relativa a um Sprint Snapshot pois os outros tipos de Snapshots seguem o mesmo modelo.

Como explicado anteriormente, cada Snapshot pode ser exportado em diferentes formatos: PDF, que permite uma melhor visualização e XLS, que permite uma melhor análise e tratamento da informação. Para exportação dos Snapshots nos diferentes formatos foi necessário a utilização de várias bibliotecas externas. Relativamente aos Snapshots no formato XLS foi utilizada a biblioteca **XlsxWriter**. Já para a criação dos Snapshots em formato

PDF foram utilizadas várias bibliotecas, a biblioteca **Selenium** e a biblioteca **Reportlab**. A primeira foi utilizada para a obtenção dos Screenshots que consistiriam no conteúdo do PDF sendo a segunda utilizada para a construção do próprio PDF. Também explicado anteriormente, aquando a criação de um Snaphots foi adicionada informação sobre o mesmo na BD assim como o respetivo ficheiro foi guardado no servidor onde se encontra alocado o projeto UIS. Uma vez que cada Snapshot é criado em *Real-Time* e representa o estado do projeto no exato momento da criação, foi crucial optar por guardar toda a informação no servidor do projeto, garantindo que toda a informação relativa aos Snapshots fica guardada e pode ser acedida posteriormente.

5.4 Outros

5.4.1 Absences

No que toca ao módulo *Absences*, foi pedido uma alteração ao modelo atual. Em vez de se marcar uma *Absence* de apenas um dia foi pedido que fosse possível marcar *absences* introduzindo um intervalo de tempo. Este módulo já se encontrava em produção e em utilização por parte dos utilizadores da empresa, existindo já o respetivo código no Servidor e no Cliente Web. Apesar de parecer bastante simples, com esta alteração foi necessário adaptar todo o código que se encontrava ligado, tanto no Servidor como no Cliente Web.

5.4.2 Work Locations

No UIS já se encontrava definido o modelo relativo aos locais de trabalho dos utilizadores. Contudo, esta informação ainda não podia ser vista pelos utilizadores, pois ainda não se encontrava implementada no Cliente Web. Foi pedido então ao estagiário, tendo em conta um *layout* definido pelo *designer*, que procedesse à sua implementação.

Na imagem 6.5 é possível ver o *layout* implementado.

Capítulo 6

Produto Final

Nesta secção serão apresentados alguns dos *layouts* do cliente *Web* que se encontram relacionados com o que foi desenvolvido no âmbito do projeto. Na imagem 6.1 encontra-se informação detalhada de um Projeto. Como se pode observar, é neste *layout* que se encontra a informação calculada na subsecção 5.3.1.5. Já na imagem 6.2 encontra-se informação detalhada de um Sprint. Dentro de algumas informações, neste *layout* encontra-se a informação calculada na subsecção 5.3.1.3. Ainda relativamente ao cálculo de informações, mais propriamente de informações relativas às atividades do projeto na subsecção 5.3.1.4, pode observar-se que se encontra tal informação no *layout* 6.3. Além dessas informações, é neste *layout* que são apresentadas as rúbricas e custos de um projeto. Por fim, no *layout* 6.4 encontra-se a informação relativa aos utilizadores da Ubiwhere e no *layout* 6.5 encontra-se informação relativa aos *Work Locations* da Ubiwhere.

Enquanto os dois últimos *layouts* não existiam e foram criados de raiz pelo estagiário, os três primeiros já se encontravam desenvolvidos mas com informação estática, tendo sido feita a adaptação dos mesmos para receber a informação calculada no Servidor. Nestes apenas foram acrescentadas informações necessárias para que pode-se ser feita a interação com as funcionalidades desenvolvidas no Servidor, como por exemplo a criação dos Botões para a exportação de Snapshots no *layout* 6.1.

Figura 6.1: *Layout* relativo a um Projeto

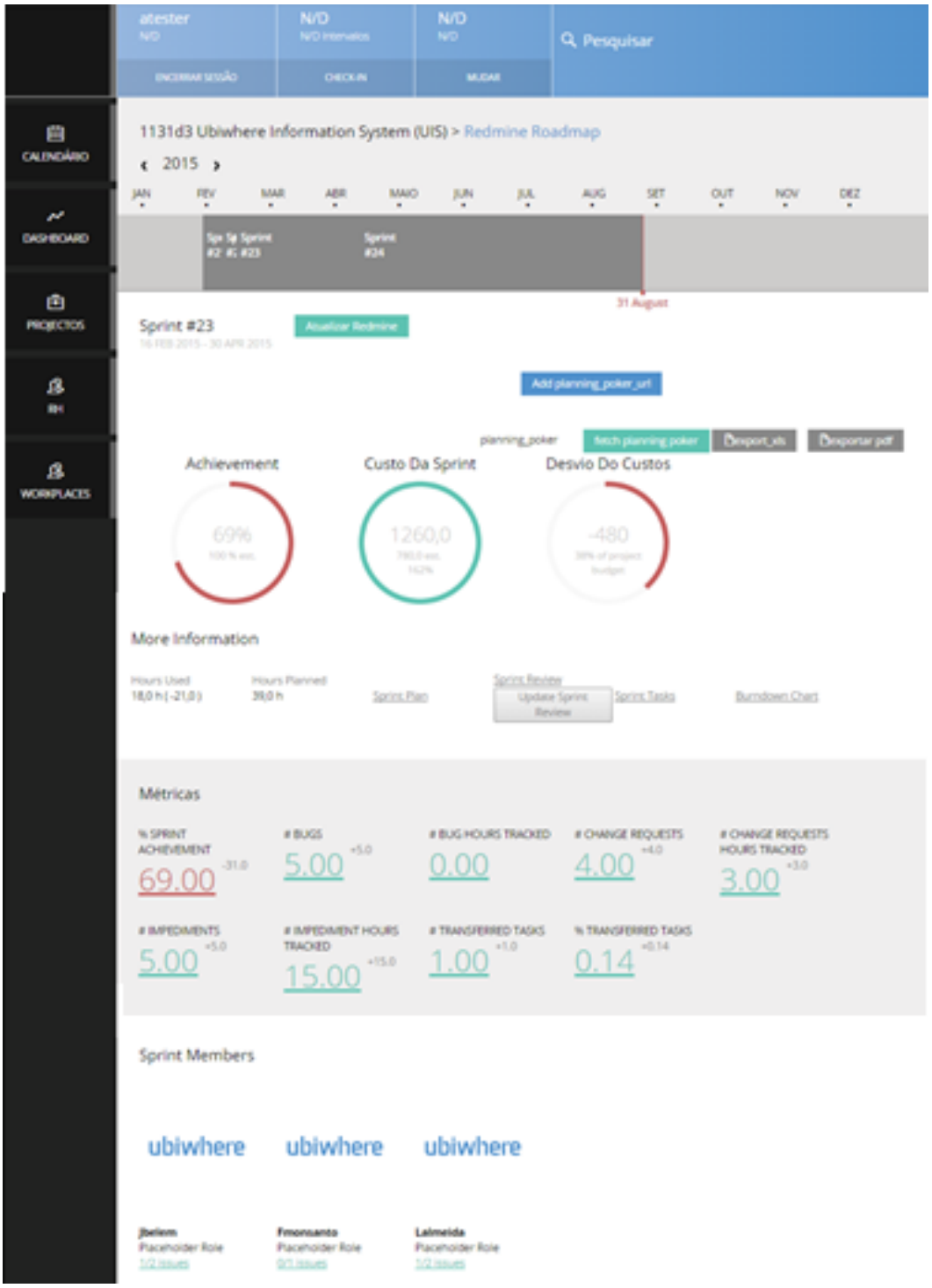


Figura 6.2: *Layout* relativo a um Sprint

Figura 6.3: *Layout* detalhado do Budget de um Projeto

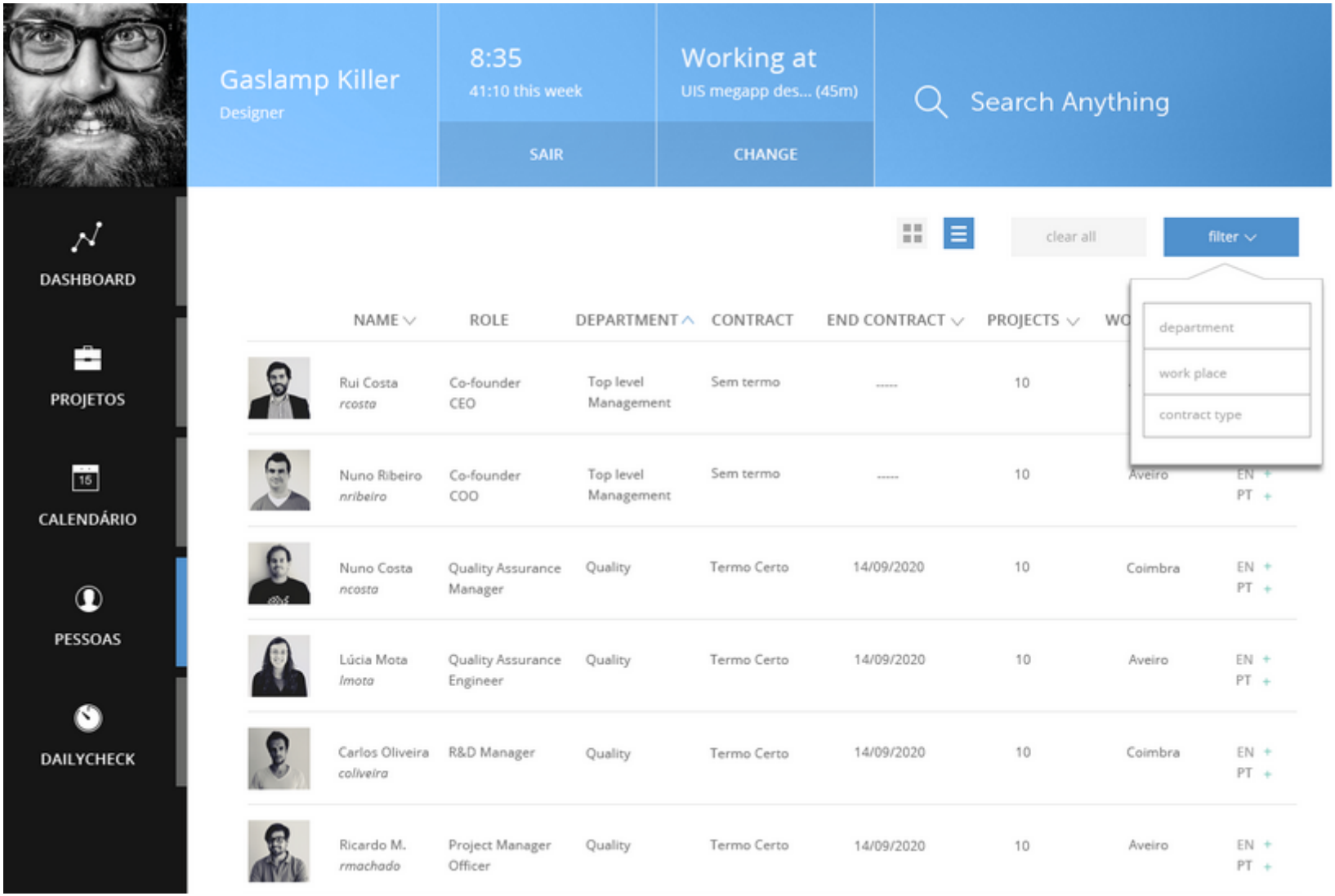


Figura 6.4: *Layout* relativo aos utilizadores da empresa

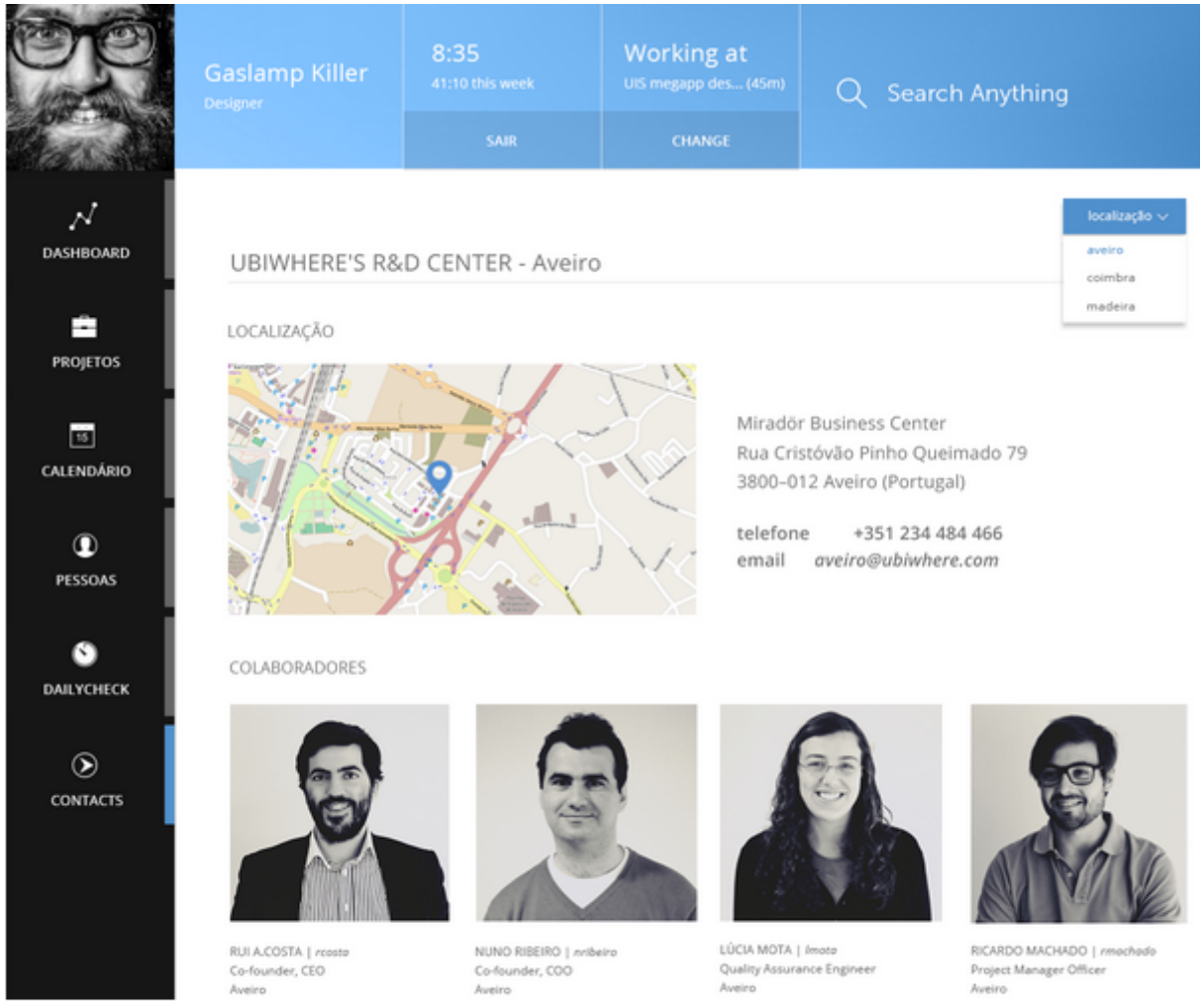


Figura 6.5: *Layout* relativo aos *Work Locations* da empresa

Capítulo 7

Testes

Nesta secção encontra-se informação sobre os testes realizados para a validação do que foi desenvolvido. Primeiro é feita uma apresentação geral dos testes desenvolvidos, sendo posteriormente feita a apresentação de cada tipo de teste com mais detalhe.

7.1 Plano de Testes

A fase reservada para a execução de testes ao sistema desenvolvido é uma das fases mais importantes no desenvolvimento de software pois só assim é possível verificar e validar tudo o que foi feito garantindo que este vai de encontro ao que é suposto. De forma a validar o sistema desenvolvido e verificar se o que foi feito se encontrava conforme o que era esperado pela empresa, foram definidos diferentes tipos de testes:

- Testes Unitários e de Integração
- Testes à API
- Testes de Sistema

Apenas foram definidos testes funcionais, testes que validam a parte funcional do projeto ao fazerem confronto entre o resultado esperado e o resultado obtido pelo sistema. Na opinião do estagiário não se justificava a realização de testes não funcionais, testes que validam os componentes que não se relacionam com a funcionalidade como a *performance* e usabilidade do sistema, por parte do mesmo pois:

- Testes de Usabilidade já foram feitos pelo *designer* aquando a construção dos protótipos do projeto.
- Testes de Performance não são relevantes pois o sistema será utilizado apenas pelos recursos da empresa (menos de 50 utilizadores) e nalguns casos apenas por um grupo específico como PM ou Managers.

Nas próximas secções encontra-se informação mais detalhada relativamente a cada tipo de testes realizado.

7.2 Testes Unitários e de Integração

A validação das unidades (métodos e funcionalidades) foi feita à medida que estas iam sendo desenvolvidas. Ao serem realizados este tipo de testes foi possível encontrar, logo no início, pequenos *bugs* e proceder à sua correção evitando a criação de problemas com maior impacto no produto final. Para cada teste foi feito primeiro o cálculo manual do valor esperado tendo sido feito depois a comparação do mesmo com o valor obtido. Nos casos em que os resultados não eram iguais foi feita a sua correção até os dados corresponderem.

Numa fase em que já existiam diversos unidades implementadas e testadas procedeu-se à realização de alguns testes de Integração. Estes testes permitiram verificar como é que as várias funcionalidades interagiam e se essa interação ocorria sem falhas. Mais uma vez foram calculados os resultados esperados tendo sido feita a comparação com o valor obtido. Assim como nos testes unitários, foram feitas as devidas correções até os testes passarem. Para a criação dos testes optou-se por utilizar uma abordagem incremental ao invés de uma abordagem não incremental. Na abordagem seguida, a integração das unidades foi feita à medida que estas iam sendo desenvolvidas em vez de se esperar pelo desenvolvimento de todas para a sua integração, o que tornou mais fácil perceber onde se encontravam alguns erros.

Nas seguintes subsecções encontra-se informação relativamente ao cálculo manual de várias informações necessárias, como as métricas e custos das Sprints.

7.2.1 Testes às Métricas

Foram testados 6 Sprints, 2 de um Projeto denominado por Projeto A e 4 do Projeto B. A informação relativamente a esta validação encontra-se, no caso do Projeto A, na tabela 7.1 e no caso do Projeto B na 7.2. Como se pode verificar na tabela, cada Sprint contém um resultado "Esperado", um resultado "UIS - Antes"e um resultado "UIS - Depois". Por resultado "Esperado"compreende-se o valor calculado manualmente tendo em conta o significado de cada métrica. Já os outros dois valores correspondem ao valor obtido pelo UIS, sendo o "UIS - Antes"correspondente ao estado com que o estagiário começou o seu estágio. Já o "UIS - Depois"corresponde ao resultado retornado pelo UIS após respetiva implementação do estagiário, nos casos em que o resultado "UIS - Antes"não se encontrava conforme o que era suposto. A vermelho encontram-se as métricas que falharam no teste. Apesar de quase todas estarem implementadas, pode verificar que os resultados obtidos não se encontravam conforme o esperado, procedendo-se então à sua implementação.

Foi feito o mesmo procedimento com as métricas de Projeto. Foram testados os Projetos referidos anteriormente, Projeto A e B, encontrando-se a informação relativamente à validação das métricas de ambos na tabela 7.3. Como observado, eram várias as métricas que não se encontravam bem implementadas, ou de todo implementadas. Procedeu-se à sua implementação.

Por fim, após a correção de todas as métricas apresentadas anteriormente, foram testadas as métricas de Portfólio, encontrando-se os resultados na tabela 7.4. Para facilitar a execução do teste, assumiu-se que o Portfólio da empresa era constituído apenas pelos Projetos A e B.

-	**Esperado**	**UIS - Antes**	**UIS - Depois**	**Esperado**	**UIS - Antes**	**UIS - Depois**
Bugs Inserted	2	2	2	9	9	9
Bug Hours Tracked	4 + 0.25	0	4.25	12.92	0	12.92
Change Request	4	4	4	3	3	3
Change Request Hours Tracked	2 + 2.5	0	4.5	3 + 1 + 2	0	6
Impediments	0 + 2	2	2	0 + 5	5	5
Impediments Hours Tracked	0 + 0.25 + 1	null	1.25	3 + 4 + 0.08 + 1.52	null	8.6
Transfered Tasks	0	0	0	0	0	0
% Transfered Tasks	0	0	0	0	0	0
% Sprint Achievement	100	1	100	100	1	100

Tabela 7.1: Validação Métricas de Sprint - Projeto A

-	**Esperado**	**UIS Antes**	**UIS Depois**	**Esperado**	**UIS Antes**	**UIS Depois**	**Esperado**	**UIS Antes**	**UIS Depois**	**Esperado**	**UIS Antes**	**UIS Depois**
Bugs Inserted	2	2	2	0	0	0	2	5	2	2	2	2
Bug Hours Tracked	3	0	3	2 + 5	0	7	0	0	0	0	0	0
Change Request	0	0	0	0	0	0	4	4	4	0	0	0
Change Request Hours Tracked	0	0	0	3	0	3	3	0	3	0	0	0
Impediments	0 + 0	0	0	0 + 0	0	0	4 + 0	5	4	0 + 0	0	0
Impediments Hours Tracked	0 + 0	null	0	1	null	1	5	null	5	0	null	0
Transfered Tasks	0	0	0	5	5	5	1	1	1	0	0	0
% Transfered Tasks	0	0	0	0.56	0.56	0.56	0.14	0.14	0.14	0	0	0
% Sprint Achievement	56	0.52	56	29	0	29	0	0.1	0	100	0	100

Tabela 7.2: Validação Métricas de Sprint - Projeto B

Métrica	Esperado	UIS Antes	UIS Depois	Esperado	UIS Antes	UIS Depois
Delivery Delay	0	null	0	(699 - 698) / 698	null	0.0014
Total Change Request	3 + 4	7	7	4	4	4
Change Requests Impact	(6 + 4.5) / 3998.14	0	0.0026	6 / 2618.17	0	0.0023
Total Bugs	9 + 2	11	11	6	6	6
Project Bugs Impact	(12,92 + 4,25) / 3998.14	0	0.0043	(3 + 7) / 2618.17	0	0.0038
Acceptance Bugs detected	1	null	1	2	null	2
Post-Production Bugs detected	2	null	2	1	null	1
Training Sessions Generated	Inserido à mão	null	Inserido à mão	Inserido à mão	null	Inserido à mão
Non-Compliances	29	29	29	0	0	0
Client Satisfaction Rate	Inserido à mão	null	Inserido à mão	Inserido à mão	null	Inserido à mão
Cost Deviation	0	null	0	((59213 * 100) / 23009) - 100	null	157.35
Total transferred tasks	0	0	0	6	6	6
Avg % Transferred Tasks	0	0	0	(0 + 5/9 + 0 + 1/7) / 4	0.17	0.17
Total Impediments	5 + 2	7	7	4	4	4
Project Impediments Impact	(8.6 + 1.25) / 3998.14	null	0.0025	(1 + 5) / 2618.17	null	0.0023
Avg Achievement	100	1	100	46.25	0.405	46.25

Tabela 7.3: Validação Métricas de Projeto A e B

Métrica	Esperado	UIS Antes	UIS Depois
Global Client Satisfaction Rate	(80 + 70) / 2	75	75
Avg Bug Impact	(0.0043 + 0.0038) / 2	0.00405	0.00405
Avg Change Request Impact	(0.0026 + 0.0023) / 2	0.00245	0.00245
Avg Impediments Impact	(0.0025 + 0.0023) / 2	0.0024	0.0024
Global Cost Deviation	0 + 157.35	157.35	157.35
Total Non Compliances	(29 + 0)	29	29
Total Training Sessions Generated	(4 + 3)	7	7
Avg Total Bugs	(11 + 6)/2	8.5	8.5
Avg Change Requests	(7 + 4) / 2	5.5	5.5
Avg Impediments	(7 + 4) / 2	5.5	5.5
Avg Transferred Tasks	(0 + 6) / 2	3	3
Avg Post Production Bugs	(2 + 1) / 2	1.5	1.5
Avg Acceptance Bugs	(1 + 2) / 2	1.5	1.5
Avg Delivery Delays	(0 + 0.0014) / 2	0.007	0.007
Avg Non Compliances	(29 + 0) / 2	14.5	14.5
Avg Training Sessions Generated	(4 + 3) / 2	3.5	3.5

Tabela 7.4: Validação Métricas de Portfólio

7.2.2 Testes Esforço e Custos

Na tabela 7.5 encontra-se informação relativa ao custo associado a cada utilizador referenciado nos testes apresentados.

Utilizador	Custo *Default*	Custo Projeto
Francisco	0	-
Joana	35	-
Luís	55	20
Nuno	35	-
Ricardo	35	-
Rui	50	-

Tabela 7.5: Tabela Exemplo de custos dos Utilizadores

Teste Esforço do Projeto

Neste tipo de teste pode verificar-se o porquê de não se ter optado pelo esforço calculado em *Real-Time* obtido tendo em conta o cálculo para o mesmo apresentado anteriormente e o esforço obtido tendo em conta o Redmine.

Esforço Total	Valor	Problema
Valor Esperado	2614,52h	-
UIS	2614,52h	-
Redmine	2655,17h	Contém *efforts* registados antes do início do projeto
Soma Atividades Redmine	153,53 + 546,75 + 287,98 + 1256,99 + 171,85 + 32,63 + 1 = 2453,73	Contém *efforts* registados que não estão em nenhuma atividade

Tabela 7.6: Esforço total do projeto UIS

Teste Custo de uma Sprint

De forma a validar se o cálculo relativo aos custos de Sprint e Projeto se encontravam corretos, foram realizados diferentes testes. Nesta subsecção é apresentado um teste feito aos custos de um dado Sprint. Cada linha da tabela corresponde a uma determinada tarefa do Sprint, e contém informação relativa ao seu responsável, ao tempo estimado e ao tempo gasto na realização da mesma. Por fim, encontra-se informação relativa ao tempo despendido (estimado e real) assim como dos custos (estimado e real) do mesmo.

Nota: Os valores utilizados para o cálculo dos custos são os valores enunciados na tabela 7.5.

Issue	**Responsável**	**Tempo Estimado**	**Tempo Gasto (durante o Sprint)**
22259	Francisco	0	2.81h Francisco + 2h Joana
22260	Luís	0	0.5h Luís
22261	Luís	0	1h Luís
22262	Luís	0	1h Luís
22263	Luís	0	1h Luís
22264	Luís	0	0.5h Luís
22265	Luís	0	3h Luís
22267	Luís	12	5.9h Luís
22269	Luís	4	1.36h Luís
22271	Luís	12	0
22272	Luís	10	1h Luís
22277	Francisco	0	0
22311	Francisco	0	0
22312	Joana	0	5h Joana
22313	Luís	0	2.29h Luís
22314	Luís	0	1h Luís
22315	Francisco	0	0
22316	Francisco	0	0
22453	Francisco	0	0
22300	Luís	0	6h Luís
	Total Tempo Esperado	38h Luís	24.55h Luís + 2.81h Francisco + 7h Joana
	Total Tempo UIS	38h	34.36h
	Total Custo Esperado	38h * 20	24.55h * 20 + 2.81h * 0 + 7h * 35
	Total Custo UIS	760	736

Tabela 7.7: Teste Custos de uma Sprint do UIS

7.2.3 Teste Variação Sprint Backlog

Este teste teve como objetivo compreender como é que era feito o cálculo da variação do Sprint Backlog no Redmine. Foram criados e testados vários Sprints para o efeito, sendo na tabela 7.8 apresentado um desses testes.

Como se pode verificar esta tabela encontra-se dividida em duas partes, a primeira referente às tarefas existentes no Sprint. Para cada uma são apresentados os Backlogs em que a mesma se encontra inserida e qual a % do Backlog antes de se iniciar o teste. Já na segunda tabela encontram-se os valores de % dos Backlogs, tendo em conta diferentes % de conclusão de cada tarefa apresentada na primeira coluna.

ID	Issue	US	Backlogs envolvidos na Sprint	%Backlog antes do teste
1	26949	None	-	-
2	26950	US 26553	Product Backlog V1.0	$\simeq 81\%$
3	27220	US 20190, 20187	Product Backlog V3.0	$\simeq 59\%$

ID	% de conclusão 0%	% de conclusão 50%	% de conclusão %100
1	-	-	-
2	$\simeq 81\%$	$\simeq 82\%$	$\simeq 84\%$
3	$\simeq 59\%$	$\simeq 61\%$	$\simeq 62\%$

Tabela 7.8: Exemplo de Cálculo Variação do Sprint Backlog

7.3 Testes à API

Para aumentar a cobertura de testes realizados, e visto que a interação com o UIS foi feita através de uma API RESTful, decidiu-se que a aplicação de testes à mesma era um aspeto bastante importante. Assim, foi possível verificar se a resposta da mesma, em diversas condições, tais como a autenticação e envio de dados inválidos, era a esperada. Apenas foram realizados testes às User Stories que validaram o projeto.

Para a criação destes testes o estagiário seguiu o processo utilizado na empresa. Por cada User Story definida no projeto foi criado um *Test Suite*, isto é, um conjunto de testes que têm como objetivo validar que determinada User Story segue determinados comportamentos. Cada teste realizado no âmbito de cada *Test Suite*, denominado *Test Case*, seguiu o *template* definido pela empresa. Na imagem D.8 é possível observar o conjunto de passos necessário para a criação do mesmo.

Nome *Test Case*

1- Número de revisão
2- Objetivo do teste
3- *User Story / Requirement* a ser validado com o seguinte teste
4- *Pre-conditions* (Condições ambientais e de estado necessárias para a realização do teste)
5- Ambiente esperado
6- *Remarks* (Informações adicionais relativa à execução do teste)
7- Passos para a execução do teste

Passos	Método HTTP	Operação
1	GET	Obter informação
2	POST	Criação de um novo item
3	PUT	*Overwrites* de um item
4	PATCH	*Update* de um item
5	DELETE	Apagar um item

8- *Post-conditions* (Condições ambientais e de estado que deve ser cumpridas após a execução de um teste)
9- Informação relativa a testes automáticos
10- Dependências (Referência a *Test Cases* que seja necessário correr antes do próprio)

Figura 7.1: *Template Test Case* utilizado pela empresa

A informação relativa a cada *Test Suite* e respetivos *Test Cases* foi guardada no Redmine. Na imagem 7.2 é possível verificar os testes criados relativamente à User Story U1.4 - Como User quero ter acesso e visualizar informação relativa às *skills* dos outros utilizadores para conhecer melhor os meus colegas". Já nas imagens 7.3 e 7.4 é possível verificar, com detalhe, um dos *Test Cases* realizados.

Em anexo, na secção D.2, encontra-se informação sobre todos os outros *Test Cases* realizados.

Test Suite #21860: TS.A.1.4 [API] - As a User, I want to access and view use...	New	QC
Test Case #28262: TC A.1.4.2 [API] - Show a specific skill	New	Joana Belém
Test Case #28263: TC A.1.4.3 [API] - Show all skills	New	Joana Belém
Test Case #28264: TC A.1.4.4 [API] - Show a skill / show all skills without...	New	Joana Belém
Test Suite #28265: TS A.1.4.1 - As an User, I want to add and edit my own sk...	New	Joana Belém
Test Case #28266: TC A.1.4.1.1 [API] - Create a skill	New	Joana Belém
Test Case #28267: TC A.1.4.1.2 [API] - Create a skill without authorized user	New	Joana Belém
Test Case #28269: TC A.1.4.1.4 [API] - Delete a skill without authorized user	New	Joana Belém
Test Case #28270: TC A.1.4.1.5 [API] - Update a skill	New	Joana Belém
Test Case #28271: TC A.1.4.1.6 [API] - Update a skill without authorized user	New	Joana Belém
Test Case #28268: TC A.1.4.1.3 [API] - Delete a skill	New	Joana Belém
Test Case #28272: TC A.1.4.1.7 [API] - Create a skill - Invalid information	New	Joana Belém
Test Case #28273: TC A.1.4.1.8 [API] - Delete an inexistent skill	New	Joana Belém
Test Case #28274: TC A.1.4.1.9 [API] - Update an inexistent skill	New	Joana Belém
Test Case #28275: TC A.1.4.1.10 [API] - Update a skill - Invalid information	New	Joana Belém
Test Case #28354: TC A.1.4.5 [API] - Show a specific skill type	New	Joana Belém
Test Case #28355: TC A.1.4.6 [API] - Show all skill types	New	Joana Belém
Test Case #28356: TC A.1.4.7 [API] - Show a skill type / show all skill typ...	New	Joana Belém
Test Case #28358: TC A.1.4.8 [API] - Create a skill type	New	Joana Belém
Test Case #28359: TC A.1.4.9 [API] - Delete a skill type	New	Joana Belém
Test Suite #21861: TS.A.2.2 [FE] - As a Manager, I want to add or edit a res...	New	QC
Test Suite #21862: TS.A.2.2 [API] - As a Manager, I want to add or edit a re...	New	QC
Test Suite #21863: TS.A.2.1 [FE] - As an User, I want to update my own perso...	In Progress	QC

Figura 7.2: *Test Cases* relativos às skills dos utilizadores

Activity #15824: A5. Pre-Production and Tests
Sub-activity #21792: A5 - Test Specification
Sub-activity #15799: Module A (Users) - Test Specification
Test Suite #21860: TS.A.1.4 [API] - As a User, I want to access and view users skills information, in order to know better my colleagues
Test Suite #28265: TS A.1.4.1 - As an User, I want to add and edit my own skills, in order to let others know where I excel

TC A.1.4.1.1 [API] - Create a skill

Added by Joana Belém 6 months ago. Updated 6 months ago.

Status:	New	**Start date:**	2015-03-03
Priority:	Normal	**Due date:**	
Assignee:	Joana Belém	**% Done:**	0%
Category:	-	**Spent time:**	-
Target version:	-		
Deadline notification:	No	**Last Execution Status:**	Passed
TC Type:		**Is Automatic?:**	

Description

Revision number

1

Objectives

The main goal of this TC is to check if user can create a skill

User Story / Requirement

#27751

Pre-conditions[1]

N/A

Expected Environment

.

Remarks

Check documentation: https://redmine.ubiwhere.com/projects/1131d3-uw-is/wiki/Post_Skill

Figura 7.3: Exemplo *Test Case*

Test Steps

user : TOP LEVEL MANAGEMENT or MIDDLE MANAGEMENT

Step Name	Step Description	Expected Result
Step *1*	HTTP POST Request	API received an HTTP POST packet with body data containing json with information of skill
Step *2*	HTTP 201 Response	API answers with a HTTP Response 201 (Created)
Step *3*	HTTP POST Request	Repeat the step 1 with same information
Step *4*	HTTP 400 Response	API answers with a HTTP Response 400 (Bad Request)

other users

Step Name	Step Description	Expected Result
Step *1*	HTTP POST Request	API received an HTTP POST packet with body data containing json with information to a skill to other user
Step *2*	HTTP 400 Response	API answers with a HTTP Response 400 (Bad Request)
Step *3*	HTTP POST Request	API received an HTTP POST packet with body data containing json with information to a skill to own user
Step *4*	HTTP 201 Response	API answers with a HTTP Response 201 (Created)
Step *5*	HTTP POST Request	Repeat the step 3 with same information
Step *6*	HTTP 400 Response	API answers with a HTTP Response 400 (Bad Request)

Post-conditions[2]

> *If successful, the skill is added in UIS DB, else, nothing happen*

Automatic Test Info

File Path:
Environment:
Fixtures:

Dependencies[3]

Notes:

[1] **Pre-conditions** means Environmental and state conditions that must be fulfilled before the component or system can be executed with a particular test or test procedure.

[2] **Post-conditions** means Environmental and state conditions and that must be fulfilled after the execution of a test or test procedure.
[3] **Dependencies** refers to Test Cases that must be executed before this one.

Figura 7.4: Exemplo *Test Case* - Continuação

7.3.1 Automação de *Test Cases*

Pensando em facilitar a obtenção de possíveis erros no trabalho desenvolvido devido a modificações futuras, o estagiário optou por automatizar os testes enunciados. Para a sua automação foi utilizada a ferramenta SoapUI, uma ferramenta utilizada para a criação de testes funcionais. Na imagem 7.5 encontra-se a estrutura definida pela Ubiwhere, estrutura esta utilizada pelo estagiário.

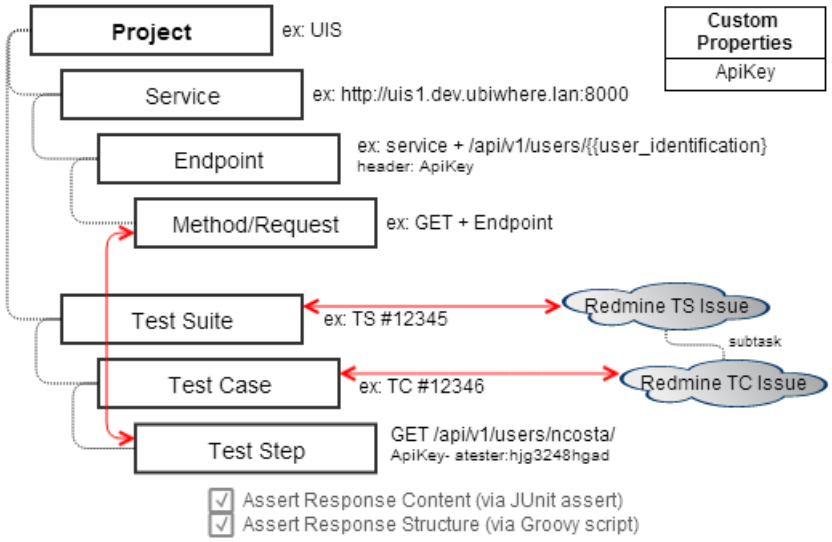


Figura 7.5: Estrutura SoapUI

Como se pode subentender através da imagem apresentada, no SoapUI foram definidos diferentes *endpoints*, cada um correspondente a um dos *Resources* criados, assim como foram definidos diferentes métodos para os mesmos (GET, POST, PATCH e DELETE). Cada *Test Suit* e *Test Case* criado no Redmine foi adicionado no SoapUI, tendo sido, por cada passo definido no *Test Case*, criado um *Test Step* que consistia na evocação de um método.

Na imagem 7.6 é possível observar alguns testes desenvolvidos e corridos no SoapUI. Estes testes foram desenvolvidos pelo estagiário com a finalidade de serem adicionados no Servidor contínuo Jenkins pelo Tester da empresa para que fossem corridos sempre que fosse feito um *push* para o Git. Não tendo ainda sido possível a adição dos testes no Jenkins, a realização destes testes foi importante e pode ser posteriormente adicionado ao mesmo.

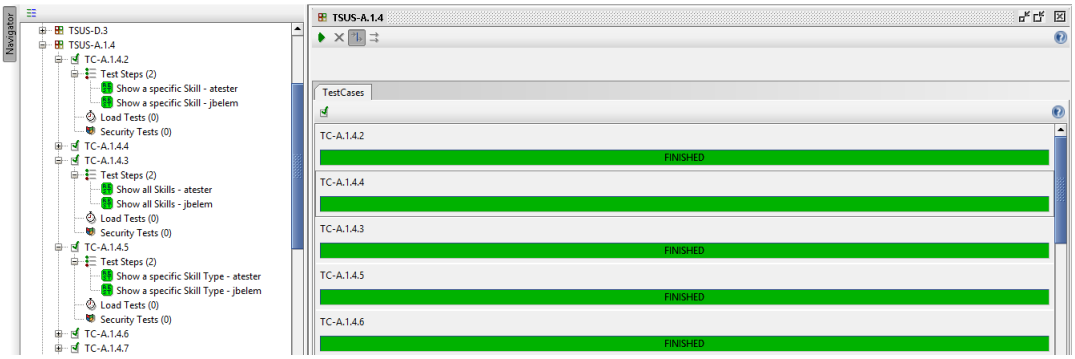


Figura 7.6: Exemplo SoapUI

7.4 Testes de Aceitação

Neste tipo de testes verifica-se o sistema como um todo confrontando os requisitos funcionais definidos inicialmente com as funcionalidades do sistema implementado. Este tipo de testes foram realizados para verificar se o que foi desenvolvido se encontrava de acordo com os requisitos que foram definidos. Estes testes apenas foram feitos após o completo desenvolvimento de cada requisito.

No anexo D.3 encontra-se o plano de testes de aceitação criado.

Capítulo 8

Conclusões

8.1 Trabalho realizado

No primeiro semestre o trabalho realizado consistiu essencialmente no estudo das metodologias seguidas pela empresa, CMMI e Scrum, assim como dos processos seguidos pela mesma. Através do estudo de vários tipos de ferramentas utilizadas por quem usa a metodologia Scrum, foi possível perceber o que existe no mercado e quais as funcionalidades das mesmas, ajudando na construção de alguns requisitos do projeto. Além disso, também foi possível perceber o porquê da empresa não ter optado por uma ferramenta já existente e desenvolver a sua própria. Ainda no primeiro semestre foi feito um estudo mais geral da arquitetura existente, de modo a perceber os seus constituintes e como estes se ligavam.

No segundo semestre procedeu-se à reformulação e desenvolvimento dos requisitos apresentados neste relatório. Além disso, foram feitas diversas validações de modo a garantir a qualidade do que foi implementado.

8.2 Contributo

Ao longo do ano letivo, foram vários os contributos que o estágio teve para com a empresa, onde se destacam:

- Implementação de novas funcionalidades importantes para a empresa
- Deteção de erros e criação da documentação inexistente relativa às funcionalidades já implementadas
- Criação da documentação do trabalho realizado
- Realização, criação e automação de testes, não só ao código implementado como ao código existente, que permitiu a implementação de código com maior qualidade e permitirá a deteção antecipada de futuros erros.
- Garantia de que a informação relativamente aos projetos da Ubiwhere é guardada e pode ser acedida posteriormente.
- Exportação de informações relacionadas com os projetos da Ubiwhere em diferentes formatos, adequando-se ao tipo de análise necessária.

Na opinião do estagiário, o trabalho por ele realizado foi importante para a empresa não só pela implementação de funcionalidades necessárias para a satisfação das necessidades da empresa, mas também pelo aumento da fiabilidade do mesmo devido à verificação e validação,

nomeadamente na obtenção de dados e métricas necessárias e bastante importantes para a empresa. Além disso, a correção e complemento da documentação da aplicação ajudará e facilitará o trabalho a ser desenvolvido, posteriormente, por outros elementos.

8.3 Principais obstáculos

Apesar de se considerar que o estágio foi bem sucedido, existiram diversos obstáculos que dificultaram a sua execução. Esses obstáculos encontram-se na seguinte lista:

- O fato de se tratar de um projeto interno fez com que nem sempre houvesse a disponibilidade desejada por parte da empresa e do estagiário para que o projeto pudesse avançar da melhor forma.
- Mudança de PM a meio do mesmo.
- Impossibilidade de implementar algumas funcionalidades, como se queria inicialmente, devido a fatores externos, desperdiçando tempo a tentar a sua implementação. Por exemplo, durante a criação do Sprint Review no Redmine era suposta adicionar os gráficos Burndown Chart, contudo a API do Redmine ainda não permiti o anexo de imagens às páginas Wiki. Para combater este problema foi pensada a utilização da biblioteca selenium para fazer a simulação do envio de um ficheiro, mas tal solução também não era exequível. Optou-se então, em último recurso, por não se adicionar as imagens e deixar o link para as mesmas.

8.4 Trabalho Futuro

Sendo o projeto relacionado com o Sistema de Informação da empresa, pode afirmar-se que nunca será terminado. Dentro de várias adições possíveis, destaca-se:

- Aplicação de técnicas de Inteligência de Negócio à informação guardada permitindo a obtenção de informações significativas e úteis para a empresa, ou seja, uma melhor análise do seu negócio.
- Adição de novas funcionalidades que vão de encontro às necessidades da empresa, nomeadamente na obtenção de diferentes níveis do CMMI.
- Integração do UIS com outro tipo de ferramentas que a empresa venha a utilizar.

8.5 Lições Aprendidas

Apesar de não ser o primeiro contacto do estagiário com o mundo empresarial, pode afirmar-se que o estágio realizado foi o primeiro contacto em que obteve uma experiência mais realista e prolongada.

Através do estágio realizado, o estagiário não só enriqueceu como adquiriu novas *Skills*, não só a nível técnico, as chamadas *Hard Skills*, como a nível pessoal as *Soft Skills*. Para a realização do estágio o estagiário teve de colocar-se na posição de diferentes cargos, desde Developer e Tester. Com maior detalhe, pode dizer-se que na posição de *Developer* o estagiário adquiriu novas capacidades principalmente no que toca à utilização das tecnologias Django e Tastypie. Numa componente de avaliador/medidor de Qualidade o estagiário também obteve novos conhecimentos relativos às ferramentas SoapUI e Selenium, ferramentas anteriormente nunca utilizadas e que certamente ajudaram na criação e validação de código no futuro.

A nível de *Soft Skills*, foram fomentadas algumas características, nomeadamente autonomia, relacionamento interpessoal e a capacidade de pensamento crítico e resolução de problemas ao tentar encontrar sempre a solução mais eficiente para os problemas apresentados e tendo sempre em conta a opinião da equipa e colegas de trabalho.

Foi no estágio que o estagiário teve a primeira experiência com a metodologia de trabalho Scrum (mesmo que não fosse a 100%), metodologia esta cada vez mais utilizada no mundo empresarial.

Resumidamente, considera-se que o estágio foi uma experiência bastante enriquecedora e que ajudará, não só, o estagiário na sua vida profissional mas também a nível pessoal.

Bibliografia

[1] T. S. Group, "Chaos manifesto 2013 big, think and small, act," 2013.

[2] "Gartner survey shows why projects fail." http://thisiswhatgoodlookslike.com/2012/06/10/gartner-survey-shows-why-projects-fail/. Accessed: 2015-01-13.

[3] N. A. de Albuquerque Costa, "Planeamento e gestão concorrente das equipas scrum: Das arquiteturas lógicas aos métodos Ágeis," 2013.

[4] S. Alliance, "The state of scrum: Benchmarks and guidelines," 2013.

[5] "Trinity management consultants limited." http://www.trinity-cmmi.co.uk/TR/index.htm. Accessed: 2015-01-10.

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

[7] "Redmine backlogs." http://www.redminebacklogs.net/. Accessed: 2015-01-10.

[8] "Scrumbler." http://www.redmine.org/plugins/scrumbler. Accessed: 2015-01-10.

[9] "Scrum2b." http://www.redmine.org/plugins/scrum2b. Accessed: 2015-01-10.

[10] "Agile plugin." http://redminecrm.com/projects/agile/pages/1. Accessed: 2015-01-10.

[11] "Agile dwarf." http://www.agiledwarf.com/. Accessed: 2015-01-10.

[12] "Version one." http://www.versionone.com/. Accessed: 2015-01-10.

[13] "Jira." https://www.atlassian.com/software/jira. Accessed: 2015-01-10.

[14] "Axosoft." http://www.axosoft.com/. Accessed: 2015-01-10.

[15] "Yodiz." http://www.yodiz.com/index.html. Accessed: 2015-01-10.

[16] "Scrumdo." https://www.scrumdo.com/. Accessed: 2015-01-10.

[17] "Scrumdo." http://www.scrumdesk.com/. Accessed: 2015-01-10.

[18] "Scrapy." http://doc.scrapy.org/en/latest/index.html. Accessed: 2015-01-10.

[19] "import io." https://import.io/. Accessed: 2015-01-10.

[20] "Kimono." https://www.kimonolabs.com/. Accessed: 2015-01-20.

[21] "Mozenda." http://www.mozenda.com/. Accessed: 2015-01-20.

[22] "Pycharm." https://www.jetbrains.com/pycharm/. Accessed: 2015-01-10.

[23] "pdadmin." http://www.pgadmin.org/. Accessed:.

[24] "Git." https://git-scm.com/. Accessed:.

[25] "Sourcetree." https://www.sourcetreeapp.com/. Accessed:.

[26] "Postman." https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop. Accessed:.

[27] "Selenium." http://www.seleniumhq.org/. Accessed:.

[28] "Soapui." http://soapui.org/. Accessed: 2015-07-14.

[29] F. Monsanto, "Management information system (django)," 2013.

Appendices

Apêndice A

Estado da Arte

A.1 Atividades - Projeto Ubiwhere

Como já enunciado, na Ubiwhere, os Projetos encontram-se divididos em diferentes Atividades. São elas:

- **A0 - Project Management** : corresponde a todas as tarefas que a Gestão de Projetos tem de efetuar para garantir que este seja executado de acordo com as normas e processos estabelecidos.
- **A1 - Concept/Feasibility** : é definido o conceito do projeto e são efetuados estudos preliminares com vista a determinar a viabilidade, detalhar o trabalho e medir o esforço que será necessário nas atividades seguintes.
- **A2 - Technical Specifications** : é feita toda a especificação técnica do Projeto, desde Levantamento de Requisitos a Especificação da Arquitetura e Testes necessários.
- **A3. Knowledge and Skills Acquisition** : corresponde a todas as tarefas que a equipa precisa de cumprir para garantir a aquisição de conhecimentos e competências necessários à execução do projeto.
- **A4. Development** : atividade de desenvolvimento de código e/ou demais tarefas associadas à execução dos objetivos do projeto.
- **A5. PreProduction and Test** : onde se enquadra a execução de testes e todos os pré-lançamentos do projeto. Está intimamente relacionada com a atividade A4 e prolonga-se até imediatamente antes da fase de produção, podendo prolongar-se, caso se continue o desenvolvimento para lá da colocação do produto do mercado.
- **A6. Production** : colocação do produto em ambiente de produção.
- **A7. Promotion and divulgation of Results (opcional)** : Promoção e divulgação dos resultados do Projeto, caso esta tarefa esteja entregue à Ubiwhere.
- **A8. Support (opcional)** : Atividades de manutenção correctiva e evolutiva, caso estas atividades estejam entregues à Ubiwhere.

Apêndice B

Arquitetura

B.1 Diagramas de Sequência para as ações RESTful

Nesta secção encontram-se os diagramas de sequência referentes às ações RESTful utilizando a ferramenta Tastypie. O estagiário optou por mostrar como se comportam as principais ações existentes no Tastypie pois para fazer o tratamento dos dados o estagiário teve de fazer *override* de algumas das funções existentes em cada uma delas.

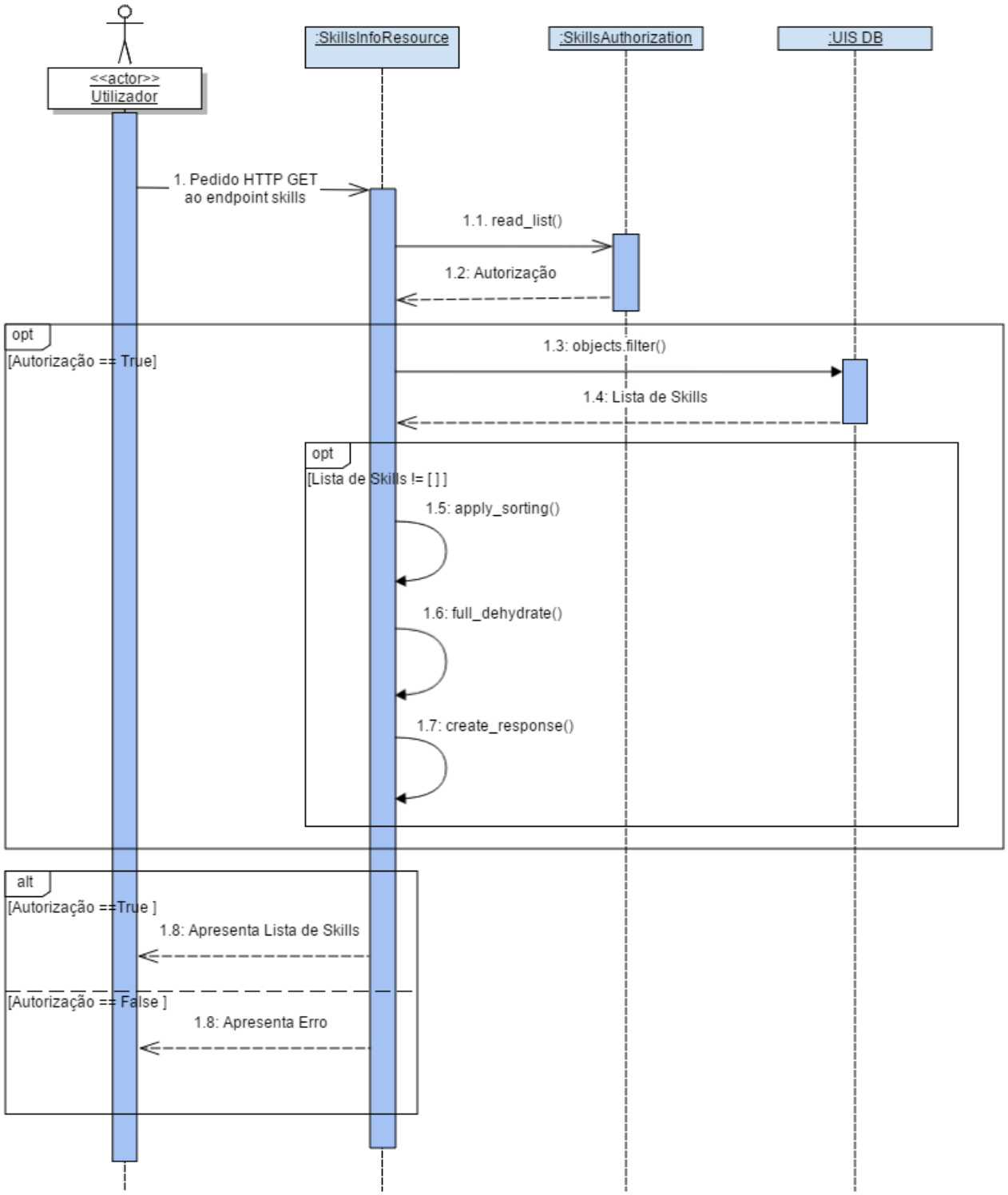


Figura B.1: Diagrama de Sequência da ação GET

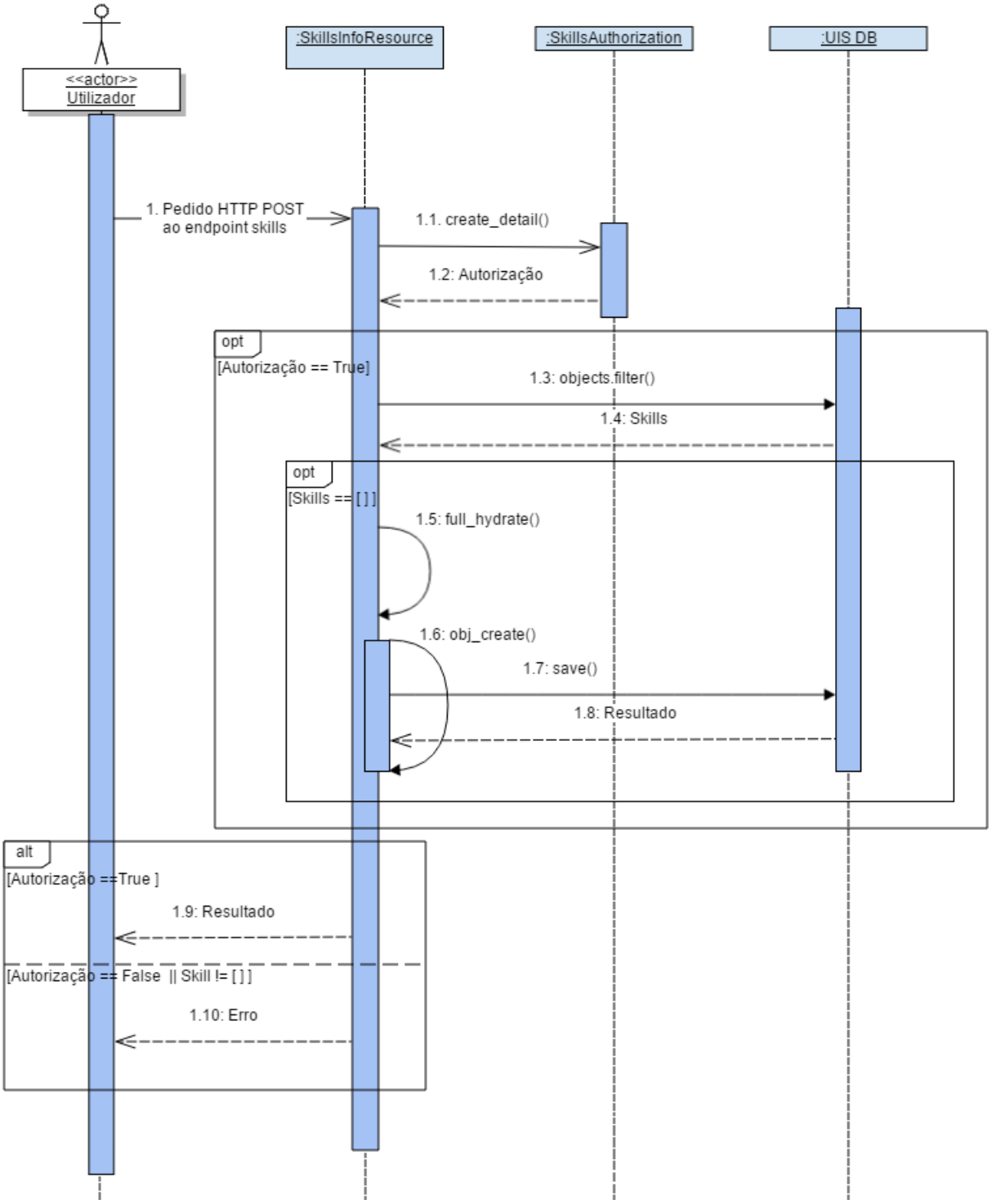


Figura B.2: Diagrama de Sequência da ação POST

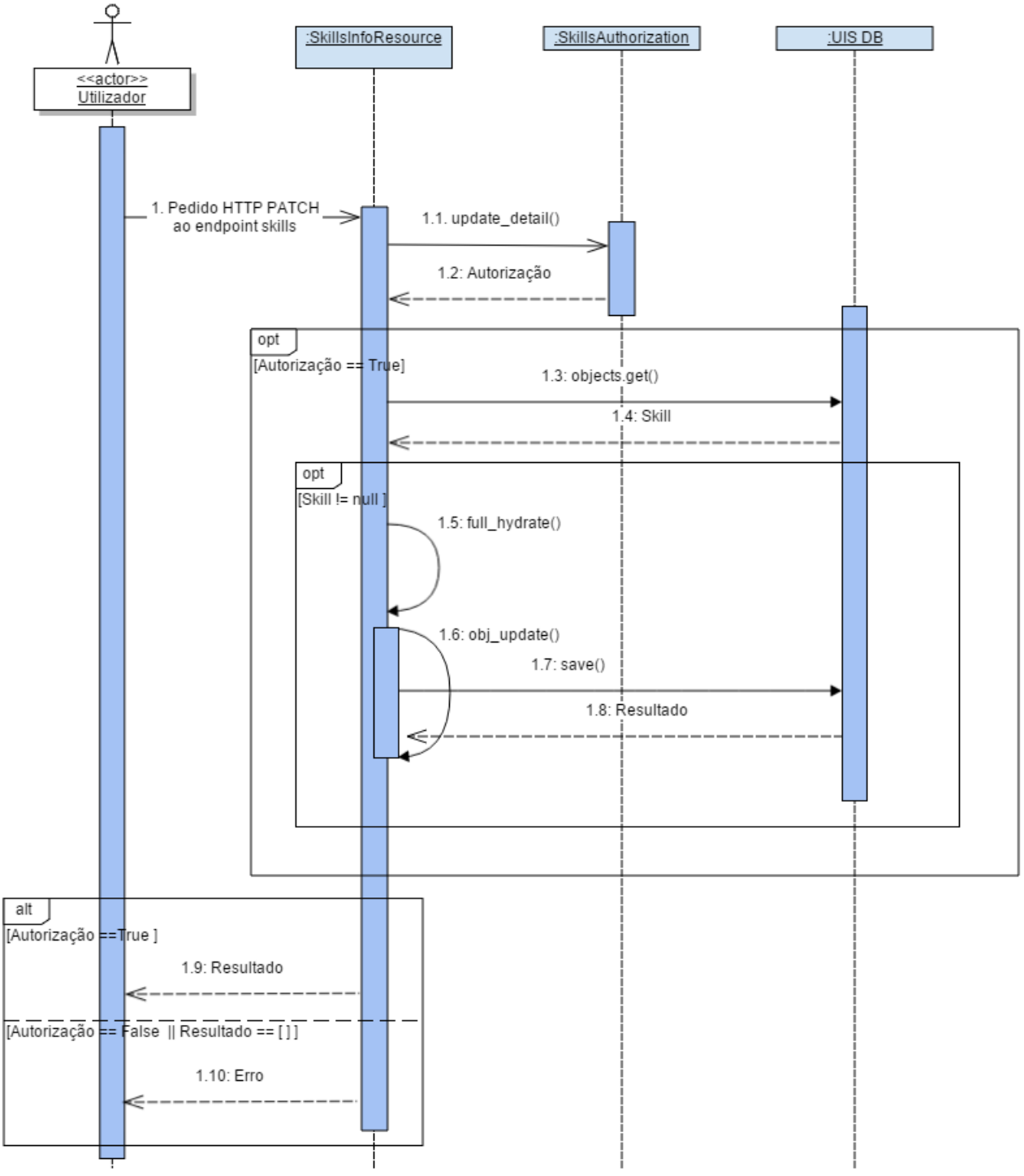


Figura B.3: Diagrama de Sequência da ação PATCH

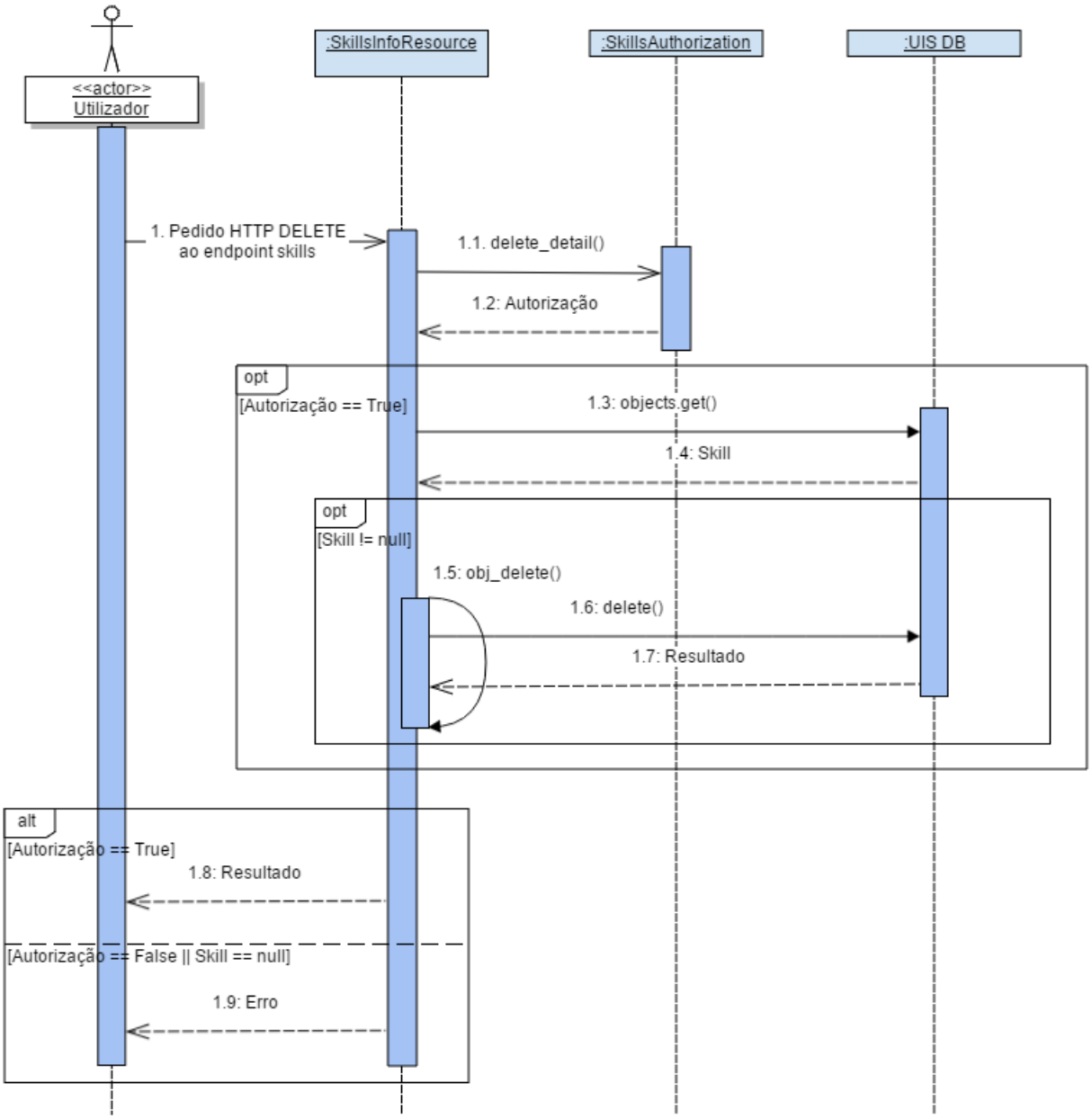


Figura B.4: Diagrama de Sequência da ação DELETE

Apêndice C

Implementação

C.1 Automação processo Planning Poker

Nesta secção é feita a apresentação do *template* Planning Poker utilizado pela empresa. O estagiário achou que seria útil a sua apresentação tornando mais intuitivo o diagrama de sequência do mesmo.

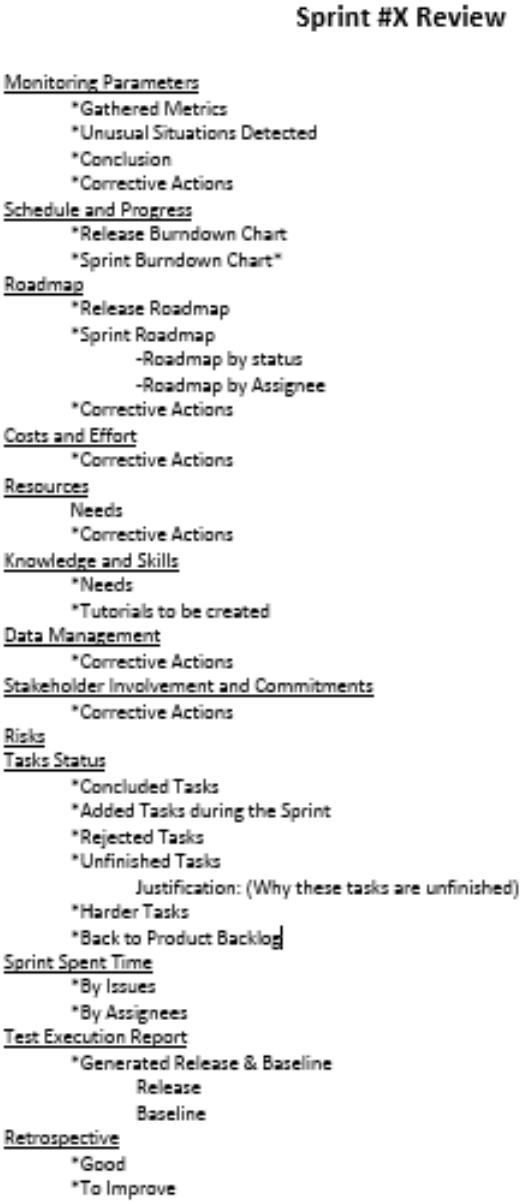
s22_planning_poker.xls [Modo de Compatibilidade]

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Sprint Backlog (US - Tasks)	P	Responsib	WC	AC	BC	Expected	Observations							
2	Task #28099: Review code and feature		Joana	32	16	8	18			h/dia	7		Total Expected		
3	Task #28095: Skills endpoint - CRUD actions		Joana	16	12	8	12			dias/sprint	10				
4	Task #28093: Skills endpoint - search feature		Joana	14	10	6	10					Disp	Aloc	Desv	
5	Task #28092: Create dictionary with skills set		Joana	8	5	3	6			Joana	100%	70		-70	
6	Task #28091: Sync only "UIS Active" projects between Redmine and UIS		Joana	12	8	6	9					0	0	0	
7	Bug #27458: Cron Job update projs with update_sprints flag enabled is crashing		Joana	12	8	6	9					0	0	0	
8	Bug #26341: Vacation days command is including inactive users		Joana	6	3	2	4					0	0	0	
9	Bug #22319: Business Days should only include approved vacations		Joana	4	2	0	2					0	0	0	
10				0		0	0					0	0	0	
11				0		0	0					**70**	**0**	**-70**	
12				0		0	0								
13				0		0	0								
14				0		0	0								

Figura C.1: Exemplo Template Planning Poker

C.2 Sprint Review - Redmine Wiki

Nesta secção encontra-se representado o *Template* que existente para a Sprint Review.



Sprint #X Review

Monitoring Parameters
- *Gathered Metrics
- *Unusual Situations Detected
- *Conclusion
- *Corrective Actions

Schedule and Progress
- *Release Burndown Chart
- *Sprint Burndown Chart*

Roadmap
- *Release Roadmap
- *Sprint Roadmap
  - -Roadmap by status
  - -Roadmap by Assignee
- *Corrective Actions

Costs and Effort
- *Corrective Actions

Resources
- Needs
- *Corrective Actions

Knowledge and Skills
- *Needs
- *Tutorials to be created

Data Management
- *Corrective Actions

Stakeholder Involvement and Commitments
- *Corrective Actions

Risks

Tasks Status
- *Concluded Tasks
- *Added Tasks during the Sprint
- *Rejected Tasks
- *Unfinished Tasks
  - Justification: (Why these tasks are unfinished)
- *Harder Tasks
- *Back to Product Backlog

Sprint Spent Time
- *By Issues
- *By Assignees

Test Execution Report
- *Generated Release & Baseline
  - Release
  - Baseline

Retrospective
- *Good
- *To Improve

Figura C.2: Exemplo *Template* Sprint Review

C.3 Snapshots

Nesta secção encontra-se um exemplo de todos os Snapshots produzidos.

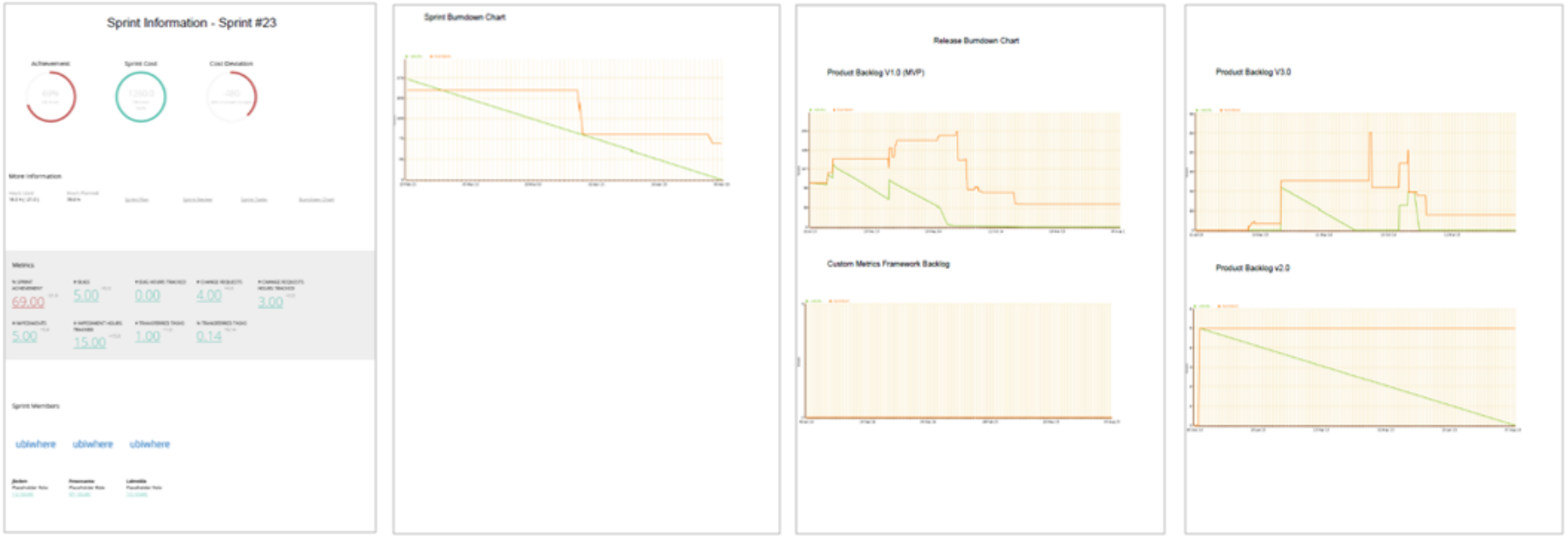


Figura C.3: Exemplo Snapshot de Sprint - PDF

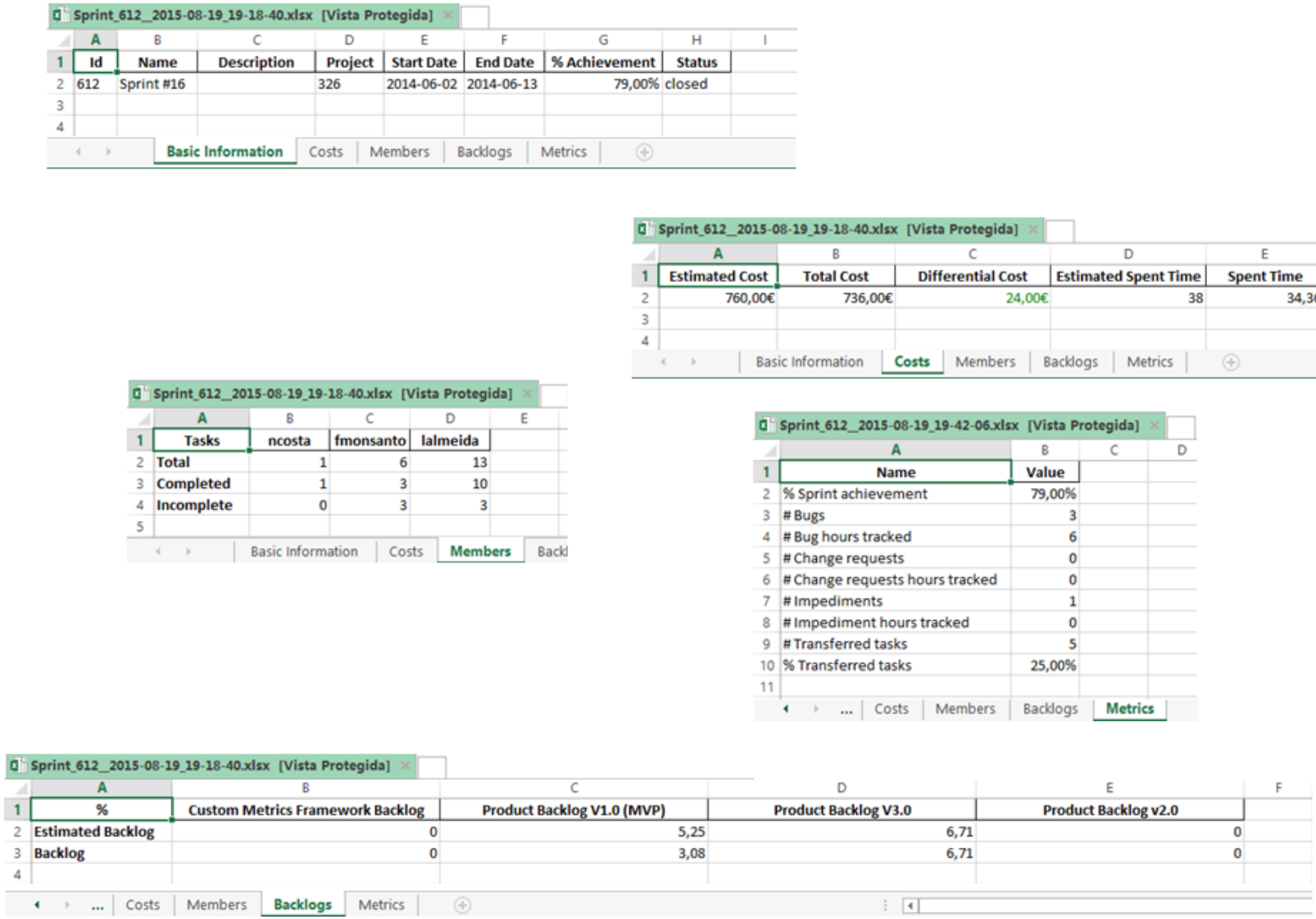


Figura C.4: Exemplo Snapshot de Sprint - XLS

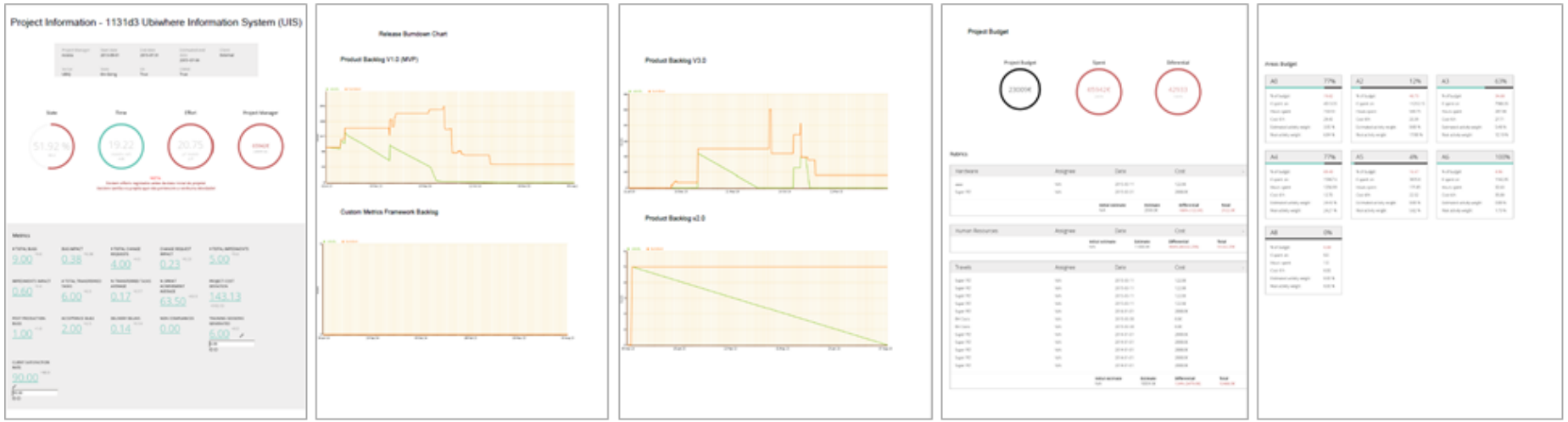
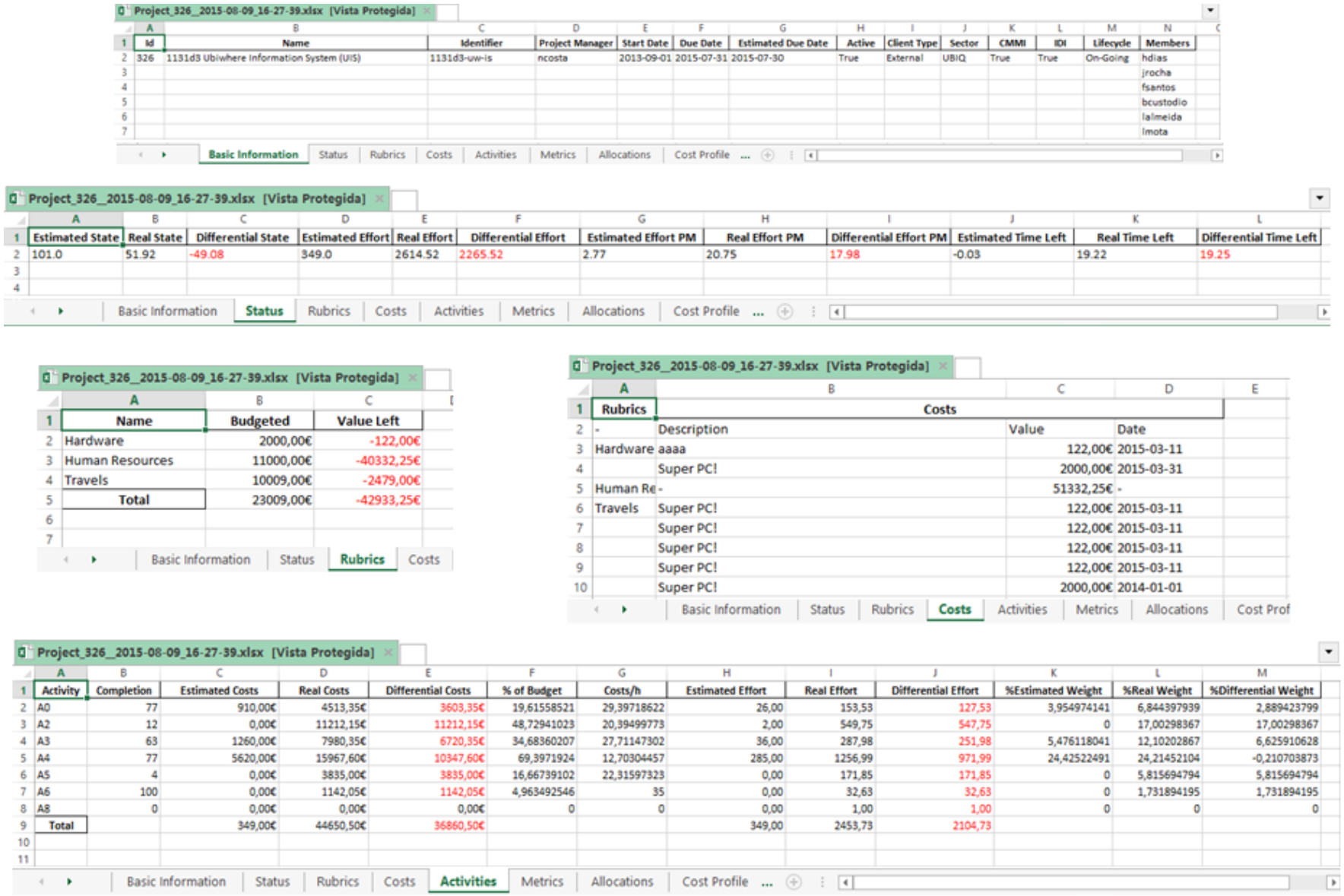


Figura C.5: Exemplo Snapshot de Projeto - PDF



Project_326_2015-08-09_16-27-39.xlsx [Vista Protegida]

Id	Name	Identifier	Project Manager	Start Date	Due Date	Estimated Due Date	Active	Client Type	Sector	CMMI	IDI	Lifecycle	Members
326	1131d3 Ubiwhere Information System (UIS)	1131d3-uw-is	ncosta	2013-09-01	2015-07-31	2015-07-30	True	External	UBIQ	True	True	On-Going	hdias
													jrocha
													fsantos
													bcustodio
													lalmeida
													lmota

Basic Information | Status | Rubrics | Costs | Activities | Metrics | Allocations | Cost Profile ...

Project_326_2015-08-09_16-27-39.xlsx [Vista Protegida]

Estimated State	Real State	Differential State	Estimated Effort	Real Effort	Differential Effort	Estimated Effort PM	Real Effort PM	Differential Effort PM	Estimated Time Left	Real Time Left	Differential Time Left
101.0	51.92	-49.08	349.0	2614.52	2265.52	2.77	20.75	17.98	-0.03	19.22	19.25

Basic Information | Status | Rubrics | Costs | Activities | Metrics | Allocations | Cost Profile ...

Project_326_2015-08-09_16-27-39.xlsx [Vista Protegida]

Name	Budgeted	Value Left
Hardware	2000,00€	-122,00€
Human Resources	11000,00€	-40332,25€
Travels	10009,00€	-2479,00€
Total	23009,00€	-42933,25€

Basic Information | Status | Rubrics | Costs

Project_326_2015-08-09_16-27-39.xlsx [Vista Protegida]

Rubrics	Costs		
-	Description	Value	Date
Hardware	aaaa	122,00€	2015-03-11
	Super PC!	2000,00€	2015-03-31
Human Re	-	51332,25€	-
Travels	Super PC!	122,00€	2015-03-11
	Super PC!	122,00€	2015-03-11
	Super PC!	122,00€	2015-03-11
	Super PC!	122,00€	2015-03-11
	Super PC!	2000,00€	2014-01-01

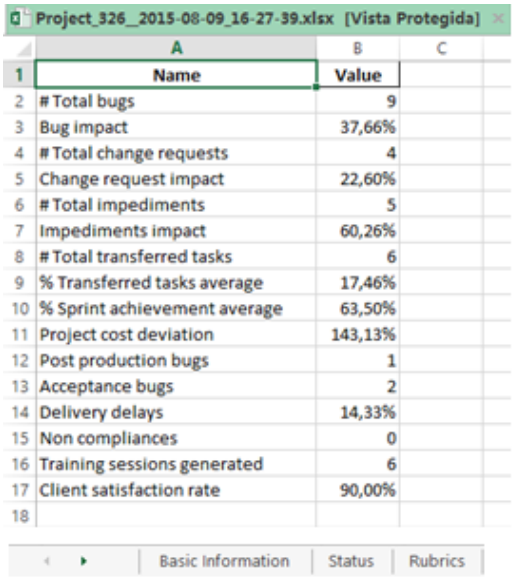
Basic Information | Status | Rubrics | Costs | Activities | Metrics | Allocations | Cost Prof

Project_326_2015-08-09_16-27-39.xlsx [Vista Protegida]

Activity	Completion	Estimated Costs	Real Costs	Differential Costs	% of Budget	Costs/h	Estimated Effort	Real Effort	Differential Effort	%Estimated Weight	%Real Weight	%Differential Weight
A0	77	910,00€	4513,35€	3603,35€	19,61558521	29,39718622	26,00	153,53	127,53	3,954974141	6,844397939	2,889423799
A2	12	0,00€	11212,15€	11212,15€	48,72941023	20,39499773	2,00	549,75	547,75	0	17,00298367	17,00298367
A3	63	1260,00€	7980,35€	6720,35€	34,68360207	27,71147302	36,00	287,98	251,98	5,476118041	12,10202867	6,625910628
A4	77	5620,00€	15967,60€	10347,60€	69,3971924	12,70304457	285,00	1256,99	971,99	24,42522491	24,21452104	-0,210703873
A5	4	0,00€	3835,00€	3835,00€	16,66739102	22,31597323	0,00	171,85	171,85	0	5,815694794	5,815694794
A6	100	0,00€	1142,05€	1142,05€	4,963492546	35	0,00	32,63	32,63	0	1,731894195	1,731894195
A8	0	0,00€	0,00€	0,00€	0	0	0,00	1,00	1,00	0	0	0
Total		349,00€	44650,50€	36860,50€			349,00	2453,73	2104,73			

Basic Information | Status | Rubrics | Costs | Activities | Metrics | Allocations | Cost Profile ...

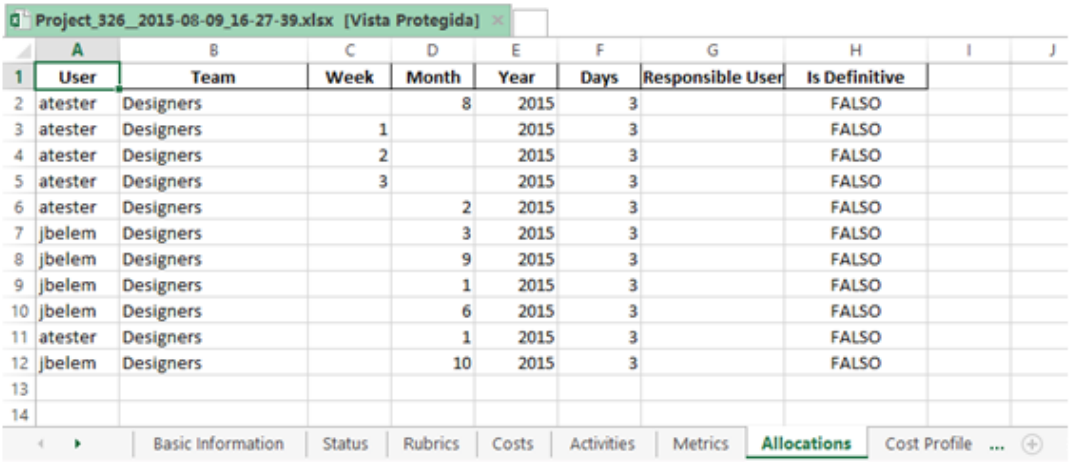
Figura C.6: Exemplo Snapshot de Projeto - XLS (1)



Project_326_2015-08-09_16-27-39.xlsx [Vista Protegida]

	A	B	C
1	Name	Value	
2	# Total bugs	9	
3	Bug impact	37,66%	
4	# Total change requests	4	
5	Change request impact	22,60%	
6	# Total impediments	5	
7	Impediments impact	60,26%	
8	# Total transferred tasks	6	
9	% Transferred tasks average	17,46%	
10	% Sprint achievement average	63,50%	
11	Project cost deviation	143,13%	
12	Post production bugs	1	
13	Acceptance bugs	2	
14	Delivery delays	14,33%	
15	Non compliances	0	
16	Training sessions generated	6	
17	Client satisfaction rate	90,00%	
18			

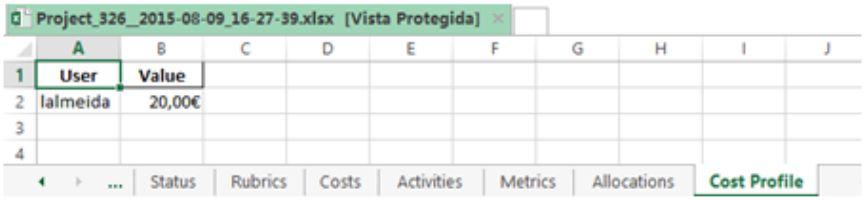
Basic Information | Status | Rubrics



Project_326_2015-08-09_16-27-39.xlsx [Vista Protegida]

	A	B	C	D	E	F	G	H	I	J
1	User	Team	Week	Month	Year	Days	Responsible User	Is Definitive		
2	atester	Designers		8	2015	3		FALSO		
3	atester	Designers	1		2015	3		FALSO		
4	atester	Designers	2		2015	3		FALSO		
5	atester	Designers	3		2015	3		FALSO		
6	atester	Designers		2	2015	3		FALSO		
7	jbelem	Designers		3	2015	3		FALSO		
8	jbelem	Designers		9	2015	3		FALSO		
9	jbelem	Designers		1	2015	3		FALSO		
10	jbelem	Designers		6	2015	3		FALSO		
11	atester	Designers		1	2015	3		FALSO		
12	jbelem	Designers		10	2015	3		FALSO		
13										
14										

Basic Information | Status | Rubrics | Costs | Activities | Metrics | Allocations | Cost Profile ...

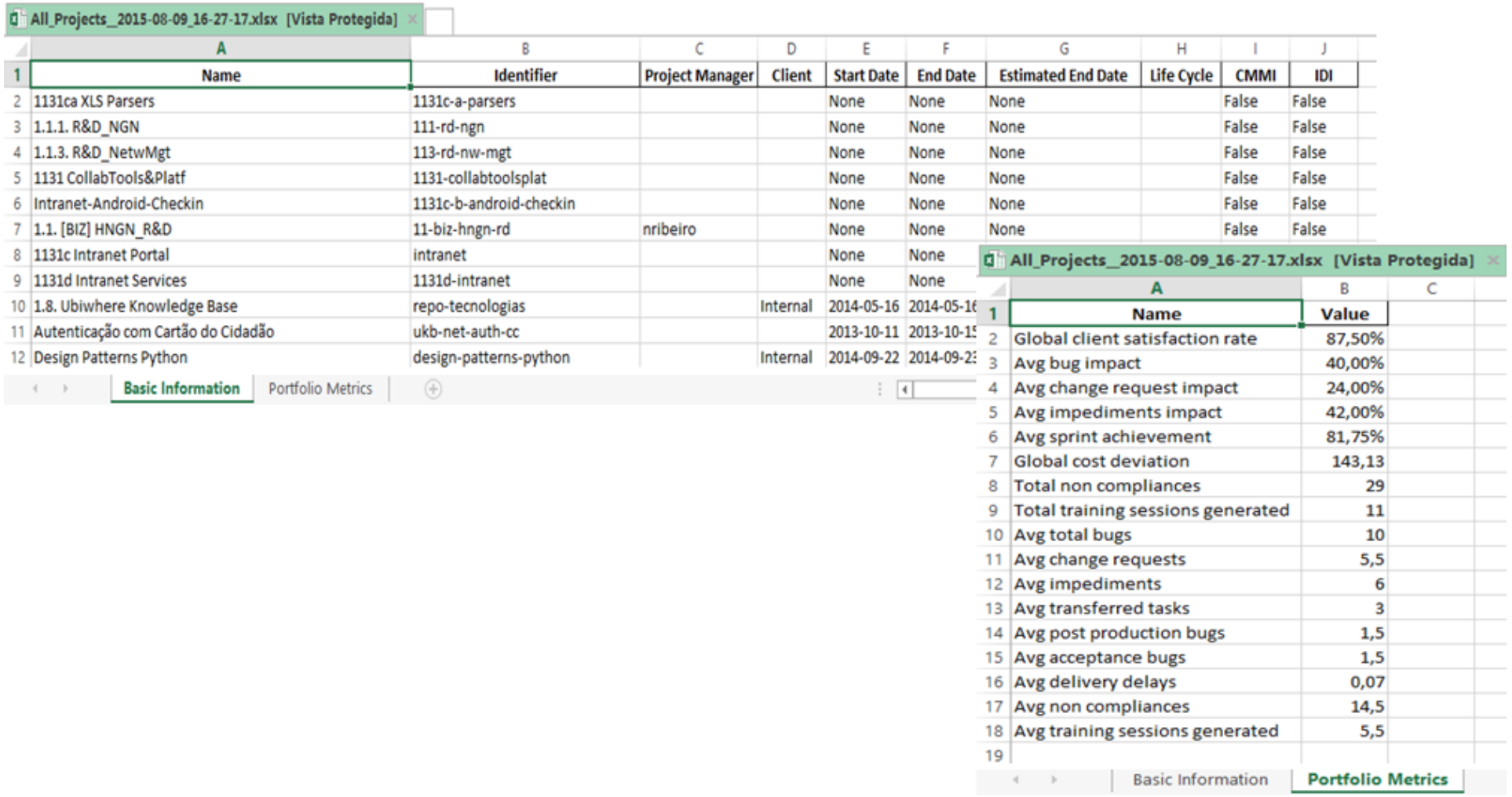


Project_326_2015-08-09_16-27-39.xlsx [Vista Protegida]

	A	B	C	D	E	F	G	H	I	J
1	User	Value								
2	lalmeida	20,00€								
3										
4										

... | Status | Rubrics | Costs | Activities | Metrics | Allocations | Cost Profile

Figura C.7: Exemplo Snapshot de Projeto - XLS (2)



All_Projects_2015-08-09_16-27-17.xlsx [Vista Protegida]

	A	B	C	D	E	F	G	H	I	J
1	Name	Identifier	Project Manager	Client	Start Date	End Date	Estimated End Date	Life Cycle	CMMI	IDI
2	1131ca XLS Parsers	1131c-a-parsers			None	None	None		False	False
3	1.1.1. R&D_NGN	111-rd-ngn			None	None	None		False	False
4	1.1.3. R&D_NetwMgt	113-rd-nw-mgt			None	None	None		False	False
5	1131 CollabTools&Platf	1131-collabtoolsplat			None	None	None		False	False
6	Intranet-Android-Checkin	1131c-b-android-checkin			None	None	None		False	False
7	1.1. [BIZ] HNGN_R&D	11-biz-hngn-rd	nribeiro		None	None	None		False	False
8	1131c Intranet Portal	intranet			None	None				
9	1131d Intranet Services	1131d-intranet			None	None				
10	1.8. Ubiwhere Knowledge Base	repo-tecnologias		Internal	2014-05-16	2014-05-16				
11	Autenticação com Cartão do Cidadão	ukb-net-auth-cc			2013-10-11	2013-10-15				
12	Design Patterns Python	design-patterns-python		Internal	2014-09-22	2014-09-23				

Basic Information | Portfolio Metrics

All_Projects_2015-08-09_16-27-17.xlsx [Vista Protegida]

	A	B	C
1	Name	Value	
2	Global client satisfaction rate	87,50%	
3	Avg bug impact	40,00%	
4	Avg change request impact	24,00%	
5	Avg impediments impact	42,00%	
6	Avg sprint achievement	81,75%	
7	Global cost deviation	143,13	
8	Total non compliances	29	
9	Total training sessions generated	11	
10	Avg total bugs	10	
11	Avg change requests	5,5	
12	Avg impediments	6	
13	Avg transferred tasks	3	
14	Avg post production bugs	1,5	
15	Avg acceptance bugs	1,5	
16	Avg delivery delays	0,07	
17	Avg non compliances	14,5	
18	Avg training sessions generated	5,5	
19			

Basic Information | Portfolio Metrics

Figura C.8: Exemplo Snapshot de Todos os Projetos - PDF

C.3.1 Bibliotecas externas utilizadas

Para a realização do projeto foi necessário a utilização de algumas bibliotecas externas. Apesar destas já se encontrarem enunciadas no relatório, o estagiário optou por fazer a centralização das mesmas facilitando a obtenção de informação por parte de quem lê o presente relatório. As bibliotecas utilizadas foram:

- **gspread** - Biblioteca que permite a gestão de Google Spreadsheets. Utilizada para fazer a análise do *template* utilizado para o Planning Poker, que se encontra alocado no Google Docs.
- **python-redmine** - Biblioteca utilizada para comunicar com o Redmine. Esta biblioteca encontra-se bem documentada e oferece suporte para 100% das *features* da API do Redmine. Optou-se pela sua escolha pois, ao contrário da API do Redmine, encontra-se bem documentada e com informação válida.
- **reportlab** - Biblioteca utilizada para a construção dos *Snapshots* em formato PDF.
- **XlsxWriter** - Biblioteca utilizada para a criação dos *Snapshots* em formato XLS.
- **selenium** - *Software testing framework* para aplicações Web. Utilizada para a criação de alguns testes e também para a obtenção de *Screenshots*, como por exemplo relativos aos Burndown Charts do projeto.
- **Scrapy** - *Web crawling* e *scraping Framework* utilizada para a obtenção de informação proveniente do Redmine.

C.4 Documentação

Nesta secção encontra-se representada informação sobre a documentação criada no âmbito do projeto. Na imagem C.9 é possível é possível observar toda a documentação criada, encontrando-se a azul o que foi criado pelo estagiário. Já na imagem C.10 é possível observar com detalhe uma página da documentação, neste caso relativamente ao Get de uma Skill. Como se pode observar encontra-se definido diferentes tipos de informação desde o *endpoint* utilizado aos campos retornados.

Figura C.9: Documentação criada

Get Skill

Lets a user get a specific skill.

Details

VERSION	1
METHOD	GET
HEADERS	Accept: application/json
ENDPOINT	@/api/v1/skills/{PARAMETER}/
RETURN	Status code
ERRORS	**401** Whenever user is not authenticated. **404** If skill not exist
AUTHENTICATION	Required
STATUS	⚠ Needs feedback

Parameters

Parameter	Required	Type	Description
{skill_id}	Yes	Int	The skill id to retrieve

Body Data Fields

None

Return Value Fields

Field	Type	Description
expertise	Int	Skill Expertise - Levels {1- Beginner, 2- Middle , 3- Advanced}
id	Int	This skill's id
skill	String	Skill type of skill
user	String	User of skill

Usage Example

URL

http://django2.dev.ubiwhere.lan:8080/api/v1/skills/80

Body data

None

Response


```
{
    "expertise": 3,
    "id": 80,
    "requested_time": 1425305628.252,
    "resource_uri": "/api/v1/skills/80/",
    "skill": "/api/v1/skills_types/1/",
    "user": "/api/v1/users/2/"
}
```


Figura C.10: Exemplo Documentação: Get Skill

Apêndice D

Testes

D.1 Testes Unitários e de Integração

Nesta secção encontra-se informação relativa ao cálculo manual de outras informações necessárias que foram calculadas e não se encontram na relatório principal.

D.1.1 Teste Custos de diferentes atividades

Nesta subsecção encontra-se os testes feitos aos custos das atividades de um projeto, encontrando-se o resultado nas tabelas seguintes. Como estes testes, além de permitir a validação dos custos da de cada atividade, foi possível confirmar se o cálculo relativo aos custos dos RH do projeto se encontrava bem feito.

Activity 0		
Issue	**Responsável**	**Tempo Estimado**
15766	Nuno	4h
19501	Nuno	2h
25070	Nuno	20h
	Tempo Esperado	26h Nuno
	Tempo Esperado UIS	26h
	Total Custo Esperado	26h * 35
	Total Custo Esperado UIS	910
Activity 2		
Issue	**Responsável**	**Tempo Estimado**
22780	Francisco	2h
	Tempo Esperado	2h Francisco
	Tempo Esperado UIS	2h
	Total Custo Esperado	2h * 0
	Total Custo Esperado UIS	0
Activity 3		
Issue	**Responsável**	**Tempo Estimado**
16591	Ricardo	36h
	Tempo Esperado	36h Ricardo
	Tempo Esperado UIS	36h
	Total Custo Esperado	36h * 35
	Total Custo Esperado UIS	1260

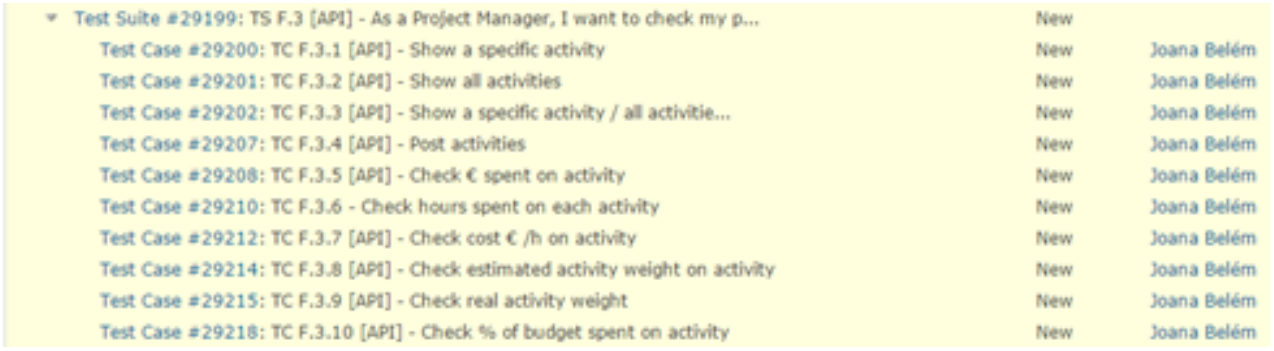
Tabela D.1: Teste Custos de várias atividades de um projeto, neste caso, Activity 0 , Activity 2 e Activity 3

Activity 4		
Issue	**Responsável**	**Tempo Estimado**
19334	Francisco	2
23249	Luís	0.5
23250	Luís	0.5
26527	Joana	5
26528	Joana	5
26530	Joana	6
26531	Joana	4
26532	Joana	3
26533	Joana	5
26536	Francisco	3
26537	Joana	4
26538	Joana	3
19327	Luís	12
23247	Luís	0.5
23248	Luís	0.5
16123	Francisco	1
19431	Francisco	4
16594	Francisco	6
19329	Luís	20
22267	Luís	12
22269	Luís	4
22270	Luís	12
22271	Luís	12
22272	Luís	10
22273	Luís	14
22274	Luís	8
23251	Francisco	1
23252	Francisco	4
16596	Francisco	4
16597	Francisco	1
16598	Francisco	7
23725	Rui	12
22266	Rui	8
19664	Rui	10
19332	Francisco	6
19333	Francisco	1
19335	Francisco	35
19338	Francisco	4
19325	Luís	15
19326	Luís	15
26475	Joana	5
	Tempo Estimado	136h Luís + 79h Francisco + 40h Joana + 30h Rui
	Tempo Estimado UIS	285h
	Total Custo Estimado	136h * 20 + 79h * 0 + 40h * 35 + 30h * 50
	Total Custo Estimado UIS	5620

Tabela D.2: Teste Custos de várias atividades de um projeto, neste caso, Activity 4

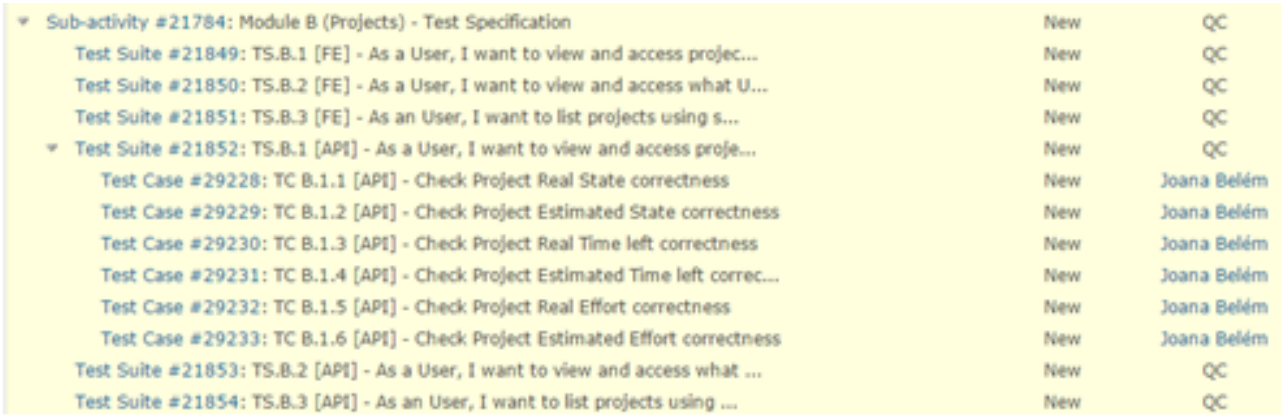
D.2 *Test Cases* criados

Nesta secção encontra-se informação relativa a todos os *Test Cases* criados. Essa informação pode ser observada nas imagens que se seguem.



Test Suite #29199: TS F.3 [API] - As a Project Manager, I want to check my p...	New	
Test Case #29200: TC F.3.1 [API] - Show a specific activity	New	Joana Belém
Test Case #29201: TC F.3.2 [API] - Show all activities	New	Joana Belém
Test Case #29202: TC F.3.3 [API] - Show a specific activity / all activitie...	New	Joana Belém
Test Case #29207: TC F.3.4 [API] - Post activities	New	Joana Belém
Test Case #29208: TC F.3.5 [API] - Check € spent on activity	New	Joana Belém
Test Case #29210: TC F.3.6 - Check hours spent on each activity	New	Joana Belém
Test Case #29212: TC F.3.7 [API] - Check cost € /h on activity	New	Joana Belém
Test Case #29214: TC F.3.8 [API] - Check estimated activity weight on activity	New	Joana Belém
Test Case #29215: TC F.3.9 [API] - Check real activity weight	New	Joana Belém
Test Case #29218: TC F.3.10 [API] - Check % of budget spent on activity	New	Joana Belém

Figura D.1: *Test Cases* relativos às atividades dos projetos



Sub-activity #21784: Module B (Projects) - Test Specification	New	QC
Test Suite #21849: TS.B.1 [FE] - As a User, I want to view and access projec...	New	QC
Test Suite #21850: TS.B.2 [FE] - As a User, I want to view and access what U...	New	QC
Test Suite #21851: TS.B.3 [FE] - As an User, I want to list projects using s...	New	QC
Test Suite #21852: TS.B.1 [API] - As a User, I want to view and access proje...	New	QC
Test Case #29228: TC B.1.1 [API] - Check Project Real State correctness	New	Joana Belém
Test Case #29229: TC B.1.2 [API] - Check Project Estimated State correctness	New	Joana Belém
Test Case #29230: TC B.1.3 [API] - Check Project Real Time left correctness	New	Joana Belém
Test Case #29231: TC B.1.4 [API] - Check Project Estimated Time left correc...	New	Joana Belém
Test Case #29232: TC B.1.5 [API] - Check Project Real Effort correctness	New	Joana Belém
Test Case #29233: TC B.1.6 [API] - Check Project Estimated Effort correctness	New	Joana Belém
Test Suite #21853: TS.B.2 [API] - As a User, I want to view and access what ...	New	QC
Test Suite #21854: TS.B.3 [API] - As an User, I want to list projects using ...	New	QC

Figura D.2: *Test Cases* relativos aos projetos

Figura D.3: *Test Cases* relativos às métricas de projeto

Figura D.4: *Test Cases* relativos às alocações dos utilizadores

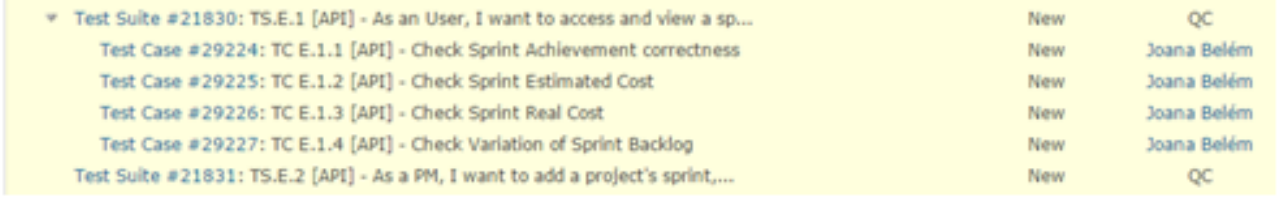


Figura D.5: *Test Cases* relativos às sprints

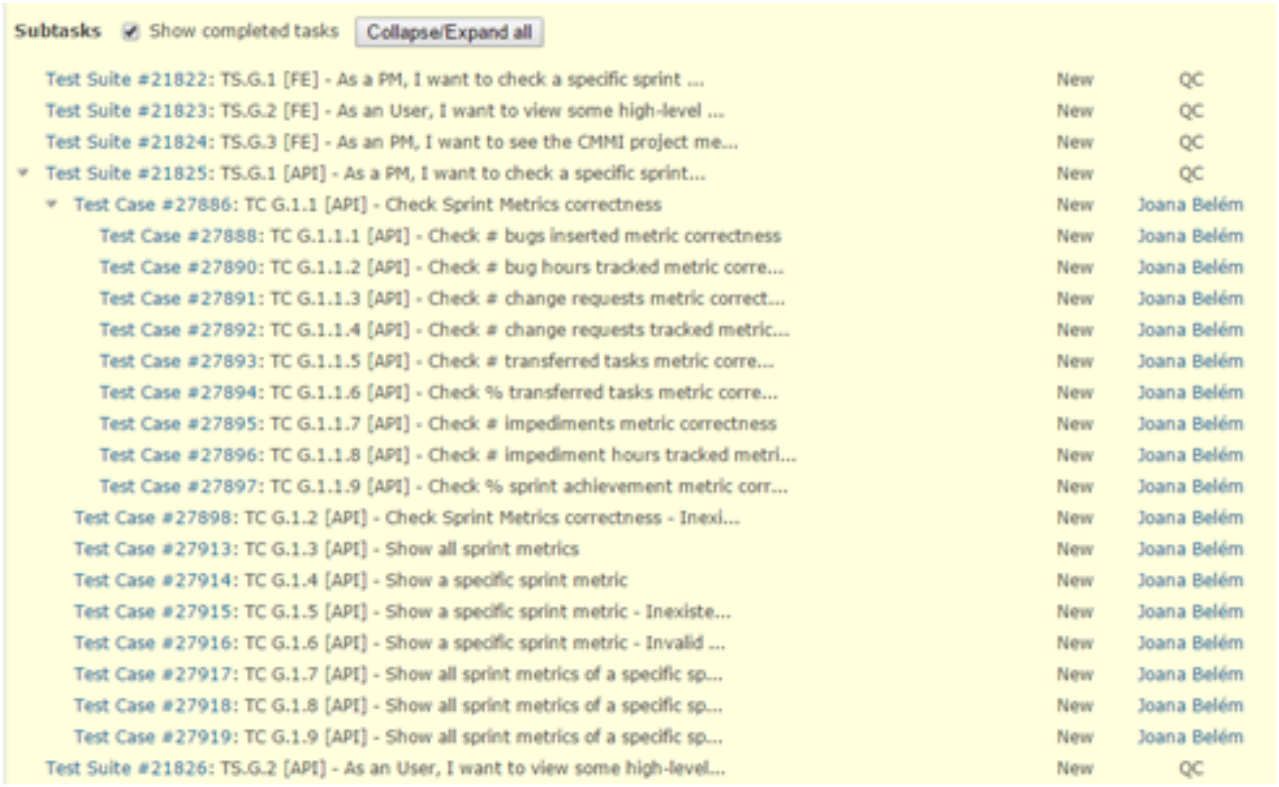


Figura D.6: *Test Cases* relativos às métricas de sprint

Figura D.7: *Test Cases* relativos às skills dos utilizadores

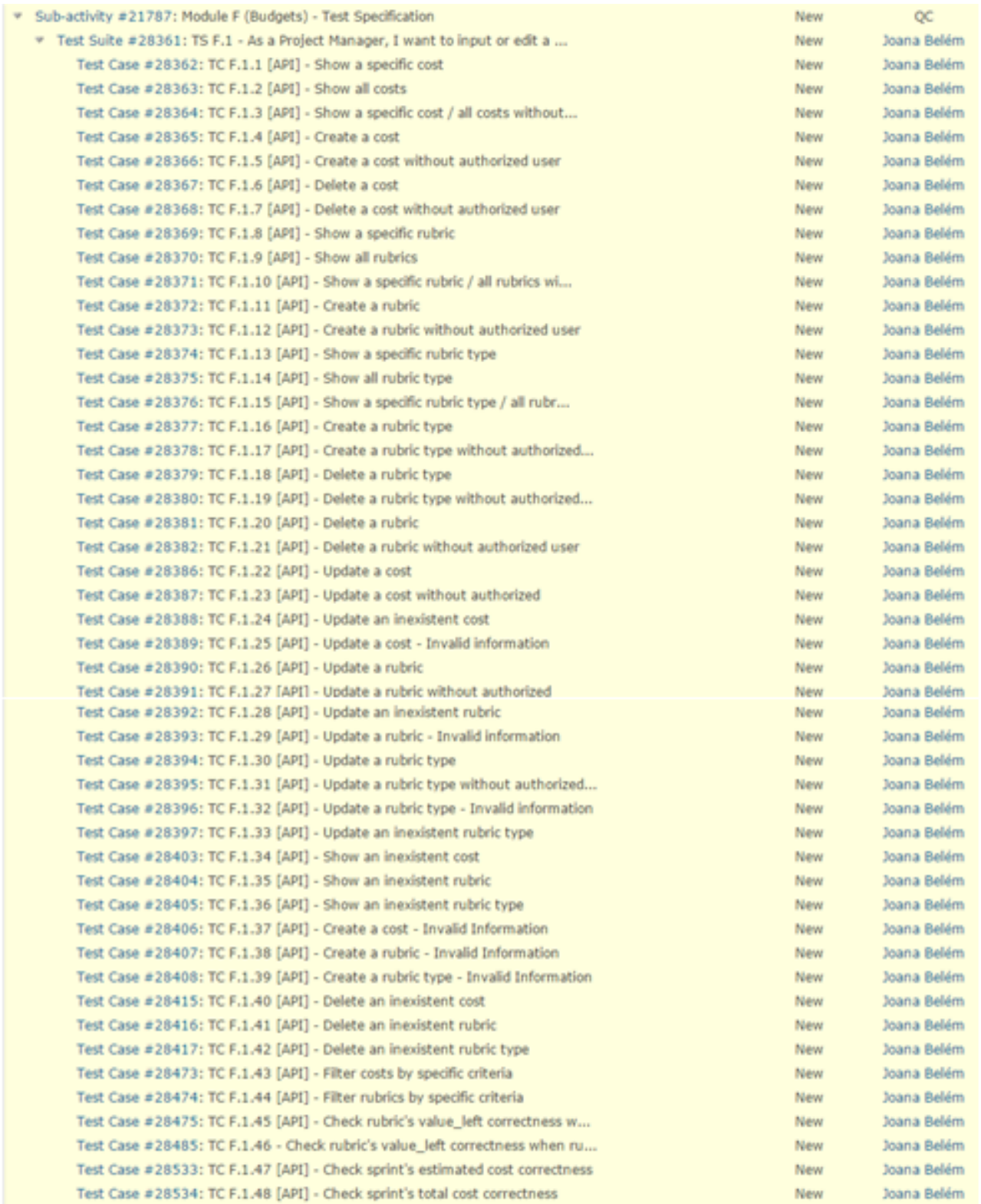


Figura D.8: *Test Cases* relativos às rubricas e custos

D.3 Plano de Testes de Aceitação

Nesta secção encontra-se o plano de testes de aceitação realizado. Estes testes encontram-se divididos pelos diferentes módulos onde se inserem, existindo a verificação se as User Stories definidas foram ou não implementadas assim como observações do porquê os testes não terem passado.

ID	User Story	Prioridade	Resultado	Observações
L1	Como PM quero usar um processo Planning Poker	5	Passou	-
L2	Como PM quero avaliar qualitativamente um Sprint recorrendo ao estado do Sprint anterior e às métricas de projeto atuais para efetuar uma avaliação mais informada do estado do projeto	5		
L3	Como PM quero criar o *Sprint Backlog* para conhecer as User Stories que serão desenvolvidas durante o Sprint	2	Não Passou	Não implementado
L4	Como User quero utilizar uma Taskboard para facilitar a gestão das minhas tarefas	2	Não Passou	Não implementado
L5	Como PM quero visualizar o Sprint Burndown Chart ou o Backlog Burdown Chart para ter uma vista mais alto nível do Sprint ou do Backlog	4	Passou	-
L6	Como PM quero adicionar, editar e eliminar pontos a favor e contra de um Sprint para facilitar próximos Sprints	2	Passou	Fazer a sua adição e gestão na Sprint Review
L7	Como PM quero gerar a documentação relativa à *Sprint Review* e exportar diretamente para o Redmine para facilitar o processo de avaliação do Sprint	5	Passou	-

Tabela D.3: Testes de Aceitação para o módulo Scrum

ID	User Story	Prioridade	Resultado	Observações
I1	Como PM quero gerar uma baseline/snapshot do projeto para ficar com o histórico do projeto	5	Passou	
I2	Como PM quero exportar para PDF/XLSX toda a informação relativa ao Sprint para guardar informações relativas a este	5	Passou	
I3	Como PM quero exportar para PDF/XLSX toda a informação relativa ao projeto para guardar informações relativas a este	5	Passou	
I4	Como PM quero visualizar uma baseline/snapshot antigo do projeto para visualizar o histórico do projeto	5	Passou	

Tabela D.4: Testes de Aceitação para o módulo Snapshots

ID	User Story	Prioridade	Resultado	Observações
H1	Como Manager quero ver o esforço despendido baseado em critérios específicos para saber o custo real do trabalho	4	Passou	

Tabela D.5: Testes de Aceitação para o módulo Efforts

ID	User Story	Prioridade	Resultado	Observações
A3	Como PM quero alocar equipas/HRs a um projeto específico, num determinado período de tempo para que os Manager conheçam as necessidades do meu projeto	4	Passou	-
A5	Como PM quero conhecer a disponibilidade dos membros da equipa para fazer a atribuição de tarefas	4	Passou	-

Tabela D.6: Testes de Aceitação para o módulo Allocations

ID	User Story	Prioridade	Resultado	Observações
F1	Como PM quero adicionar ou editar um novo orçamento para o projeto para estipular uma quantidade específica para gastar durante o projeto	4	Passou	-
F2	Como PM quero verificar o orçamento do meu Sprint para verificar o quanto eu gastei em apenas um Sprint	4	Passou	-
F3	Como PM quero verificar o orçamento global do meu projeto para obter uma imagem sobre os custos do projeto	4	Passou	-

Tabela D.7: Testes de Aceitação para o módulo Costs

ID	User Story	Prioridade	Resultado	Observações
G1	Como PM quero ver métricas específicas de um Sprint para saber a performance real da equipa	4	Passou	-
G3	Como PM quero ver as métricas CMMI de um projeto para saber se o meu projeto se encontra de acordo com os objetivos	4	Passou	-

Tabela D.8: Testes de Aceitação para o módulo Metrics

ID	User Story	Prioridade	Resultado	Observações
U1.4	Como User quero ter acesso e visualizar informação relativa às *skills* dos outros utilizadores para conhecer melhor os meus colegas	4	Passou	-
U1.5	Como User quero listar e pesquisar outros utilizadores para obter o utilizador que eu quero	4	Passou	-
U1.7	como User quero ver informação específica sobre os locais de trabalho da Ubiwhere para conhecer mais sobre cada localização	4	Passou	-

Tabela D.9: Testes de Aceitação para o módulo Users

ID	User Story	Prioridade	Resultado	Observações
B2	Como User quero ver e ter acesso a todos os utilizadores que estão ou estiveram envolvidos no projeto para conhecer melhor o projeto	4	Passou	-
B4	Como User quero ver e ter acesso as diferentes tipos de informação relacionadas com o projeto para conhecer melhor o mesmo	4	Passou	-

Tabela D.10: Testes de Aceitação para o módulo Projects

ID	User Story	Prioridade	Resultado	Observações
E1	Como User quero ver e ter acesso a informação relativa a um sprint específico para conhecer mais detalhes sobre um projeto	4	Passou	-

Tabela D.11: Testes de Aceitação para o módulo Sprints