UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

DESENVOLVIMENTO DE UMA PLATAFORMA DE GESTÃO E MONITORIZAÇÃO DE PROJECTOS DESENVOLVIDOS SEGUINDO A METODOLOGIA *SCRUM*

Sara Catalão Pimentel

Mestrado em Engenharia Informática

Sistemas de Informação

2010

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

DESENVOLVIMENTO DE UMA PLATAFORMA DE GESTÃO E MONITORIZAÇÃO DE PROJECTOS DESENVOLVIDOS SEGUINDO A METODOLOGIA *SCRUM*

Sara Catalão Pimentel

PROJECTO

Trabalho orientado pelo Prof. Doutor Thibault Nicolas Langlois
e co-orientado por Helder Nuno Rodrigues Faria

Mestrado em Engenharia Informática
Sistemas de Informação

2010

Agradecimentos

À Truewind pela forma como receberam, pela oportunidade, liberdade e todo o apoio que me deram.

Quero agradecer em especial a toda a equipa que me acolheu no meu primeiro projecto, Ricardo Gonçalves, André Santos e Raquel Figueiredo, pela paciência, dedicação e ensinamentos que me deram. Agradecer ao Helder Faria meu orientador, pelo tempo pessoal, o esforço, a ajuda preciosa que me deu no meu trabalho, a revisão desta tese que acredito que por vezes não tenha sido fácil.

No percurso da vida académica agradeço a todos os meus amigos que me acompanharam em inúmeras noitadas, que me apoiaram, que não me deixaram nunca desistir, que me animaram em momentos difíceis, pelas discussões de projectos, pelo estudo em conjunto para os exames, entre muitos outros momentos que me proporcionaram. Agradeço em especial, à minha companheira, de guerra, de casa, de projectos, à minha querida mais que amiga, Ana Leal a sua presença nestes últimos anos foi fulcral tornando tudo mais fácil, um grande obrigado.

Por último mas não menos importante à minha família, em especial ao meu pai, por tudo aquilo que me conseguiu proporcionar sem ele nada disto seria possível e à minha querida mãe por todo o apoio que me deu, por toda a força que me deu em momentos difíceis, por partilhar os momentos de felicidade, por tentar compreender, quão difícil é por vezes o mundo da faculdade, por tudo. Um muito obrigado.

À minha família e aos meus verdadeiros amigos

Resumo

Existem diversas metodologias de construção de software que partilham o principal propósito de satisfazer o cliente o melhor possível. Um dos maiores riscos no desenvolvimento de software é o do produto final ficar distante das necessidades inicialmente antevistas. Com isto surgem as *metodologias ágeis* que tentam mitigar os riscos das metodologias tradicionais, utilizando ciclos iterativos onde no final de cada ciclo são demonstradas aos utilizadores finais as funcionalidades desenvolvidas.

Na instituição de acolhimento já era utilizada uma metodologia ágil designada por *Scrum*. Havendo diversos pontos onde a aplicação desta metodologia era carenciada, como por exemplo a utilização de folhas de cálculo dispersas para o registo de tarefas e horas gastas em cada tarefa e para a monitorização do processo de desenvolvimento, surgiu como objectivo deste projecto a criação de uma plataforma para gestão e monitorização de projectos desenvolvidos na empresa seguindo a metodologia *scrum*.

Apesar de já existir inúmeras ferramentas com este propósito nenhuma delas ia de encontro às necessidades concretas da empresa, surgindo assim uma oportunidade de criar uma solução à medida da empresa.

A solução foi desenhada e realizada na plataforma *Java Enterprise Edition (JEE)*, tendo sido aplicada no decorrer do seu processo de desenvolvimento os princípios da metodologia *scrum*.

Este relatório descreve a experiencia adquirida durante o estágio, a metodologia *scrum*, os principais objectivos da solução desenvolvida, bem como as melhorias que representa face às ferramentas que se encontravam em uso na empresa. É relatado todo o processo envolvido para a realização desta, assim como a tecnologia utilizada na sua construção.

Palavra-chave: Metodologias Ágeis, *Scrum*, JEE

Abstract

There are several methodologies for software development focused on customer satisfaction. One of the biggest risks in software development is the delivery on an end product that does not satisfy the initial needs. Agile methodologies attempt to mitigate the risks of traditional methodologies, by using iterative cycles where at the end of each cycle are demonstrated to end users the features developed.

At the hosting institution was already used an agile methodology called Scrum. There were several aspects where the usage of this methodology presents flaws, such as the use of scattered spreadsheets for the tracking of tasks and effort spent on each task and for monitoring the development process; emerged as a goal of this project the creation of a platform for management and monitoring of projects undertaken in the company using scrum methodology.

Although several tools already exist for this purpose, none of them satisfies de concrete needs of the company, thus resulting in an opportunity to create a tailored solution for the company.

The solution was designed and realized using Java's Platform, Enterprise Edition (JEE); during de development process of the application, the principles of scrum methodology where applied.

This report describes the experience acquired during the internship, scrum methodology, the main goals of the developed solution and the improvements obtained against the tools that were in use in the company. It is described the whole process involved to achieve this, as well as the technology used in its construction.

Keywords: Agile Methodologies, Scrum, JEE

Conteúdo

Lista de Ilustrações

Lista de Tabelas

Lista de Exemplos

Capítulo 1

Introdução

Este documento apresenta e descreve o projecto realizado para a disciplina Projecto de Engenharia Informática, no âmbito do Mestrado de Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa, na empresa Truewind, Sistemas de Informação S.A., durante o ano lectivo 2009/2010. O projecto desenvolvido intitula-se **Desenvolvimento de Uma Plataforma de Gestão e Monitorização de Projectos Desenvolvidos Seguindo a Metodologia *Scrum***.

O trabalho foi realizado em duas grandes fases:

- Participação num projecto que decorria na instituição de acolhimento, desenvolvido com utilizando a linguagem Java, no qual foram utilizados alguns dos princípios da metodologia; a integração do aluno na equipa responsável pela execução deste projecto, para permitir por um lado familiarizar-se com a utilização destas metodologias num contexto real de um projecto, e por outro adquirir um nível de competência mínimo no desenvolvimento de aplicações na linguagem Java.
- Desenvolvimento na linguagem Java de uma aplicação de suporte à gestão de processo de desenvolvimento em projectos para os quais seja utilizada a metodologia *scrum*.

1.1 Motivação

No mundo do desenvolvimento de software acontece frequentemente o produto final ficar distante das necessidades inicialmente antevistas, em especial em projectos longos ou com reduzida interacção com o utilizador final. Cada vez mais, uma das formas utilizadas na abordagem a esta questão passa pela utilização de metodologias

ágeis no processo de desenvolvimento de software. As metodologias ágeis valorizam e propiciam:

- Indivíduos e interacções mais do que processos e ferramentas;
- Software funcional mais do que documentação extensa;
- Colaboração com o utilizador final mais do que a negociação do contrato;
- Responder à mudança mais do que seguir um plano rígido;
- Garantir a satisfação do utilizador final através da entrega antecipada e contínua de software funcional;
- O acolher de mudanças tardias, sendo uma mais-valia para o cliente;
- Cooperação entre pessoas do negócio e programadores;
- Construir projectos com indivíduos motivados, criando um ambiente de confiança;
- Simplicidade;
- O diálogo regular para transmissão de informação;
- A atenção contínua à excelência técnica e ao desenho e arquitectura.

No âmbito dos seus projectos de desenvolvimento de software, a Truewind utiliza sempre que possível a metodologia *scrum* na gestão do processo de desenvolvimento. Nesta lógica, surge a necessidade de as suas equipas utilizarem ferramentas que auxiliem e suportem esta forma de funcionamento. Apesar de aplicações desta natureza estarem disponíveis no mercado, acabam por nunca ser inteiramente adequadas à realidade concreta dos projectos da Truewind por via dos seguintes factores:

- Serem genéricas, não respondendo a algumas especificidades do universo de clientes da Truewind;
- Apresentarem limitações no concerne à sua interligação com as ferramentas de integração contínua e testes unitários;
- Carecerem de uma integração com a ferramenta utilizada internamente pela Truewind, para registo de tempos gastos em projectos.

1.2 Objectivos

Através da integração do aluno na equipa responsável pelo desenvolvimento de um projecto que decorreu na Truewind, pretendeu-se atingir os seguintes objectivos:

- Consolidação das suas competências na área do desenvolvimento de software na linguagem Java, nomeadamente através da utilização da plataforma J2EE.
- Aprendizagem das técnicas de testes unitários e de integração contínua actualmente já consolidadas na equipa na qual o aluno se integrou.
- Familiarização com as metodologias de desenvolvimento ágil, em especial a metodologia *scrum*, utilizada para a gestão projecto no qual foi integrado.

Como consequência da aquisição de novos conhecimentos e novas experiências no decorrer do referido projecto, apresentou-se como segundo grande objectivo para o presente projecto, o desenvolvimento na linguagem Java de uma aplicação para a gestão e monitorização de projectos desenvolvidos utilizando a metodologia *scrum*. As tarefas realizadas incluem a análise, o desenho e a codificação desta aplicação que foi constituída pelos três seguintes módulos:

- Gestão e planeamento – registo e manutenção do *product backlog*; gestão do ciclo de desenvolvimento associado a cada *sprint*, incluindo o registo de toda a informação a ele associada: *sprint backlog*, alocação de tarefas, de capacidade da equipa e ainda a recolha de estado e esforço de execução das tarefas.
- Monitorização – quadro de bordo de monitorização de cada ciclo de desenvolvimento, incluindo *burndown chart* e *cumulative flow chart*;
- Registo de tempos – integração com a ferramenta de registo de horas e actividades actualmente em utilização na Truewind, de forma evitar a dupla inserção de dados por parte dos recursos envolvidos em projectos onde seja aplicada a metodologia *scrum*.

1.3 Integração na instituição de acolhimento

O estágio começou em Novembro de 2009 com uma fase de integração na empresa, iniciada nas instalações da Truewind e que decorreu no seu essencial, através

da integração do aluno numa equipa da unidade Java, responsável pelo desenvolvimento de um sistema de elevada complexidade para um organismo na esfera da administração pública. Nesta fase os objectivos definidos foram conhecer a estrutura e organização da empresa, as diferentes unidades de negócio e os seus métodos de trabalho.

Ao mesmo tempo foi iniciada uma formação específica em programação na linguagem Java, nomeadamente no desenvolvimento de soluções Web baseadas na plataforma J2EE, com vista à familiarização com esta tecnologia. Sendo uma linguagem da qual o aluno já detinha alguns conhecimentos, verificou-se ainda assim, uma forte necessidade de consolidar e solidificar a competência nesta área, onde por um lado, houve um grande apoio da equipa na qual o aluno se integrou, e por outro, se efectuou durante um período de algumas semanas autoformação, com leitura e realização de tutoriais.

1.4 Estrutura do documento

Este documento apresenta e descreve o projecto realizado para a disciplina Projecto de Engenharia Informática, encontrando-se organizado de acordo com o enumerado na Tabela 1:

Nº Capítulo	**Título**	**Descrição**
1	Introdução	Descrição sumária do trabalho desenvolvido da sua motivação e objectivos.
2	Planeamento e Metodologia	Planeamento inicial e a calendarização da execução real do projecto, assim como, refere a metodologia utilizada para a realização do projecto.
3	Contexto Actual	Descreve pormenorizadamente a metodologia *scrum*. Análise de ferramentas já existentes.
4	Tecnologias	Neste capítulo enumeram-se as tecnologias utilizadas para realização do projecto.
5	Arquitectura e Descrição	Arquitectura do sistema e da aplicação. Análise

Nº Capítulo	Título	Descrição
	Funcional	da aplicação. Descrição geral das suas funcionalidades.
6	Trabalho Realizado	Trabalho realizado para a concepção da aplicação proposta.
7	Conclusão e Trabalho Futuro	Conclusão do trabalho realizado identificação de trabalho futuro.

Tabela 1 – Estrutura do documento

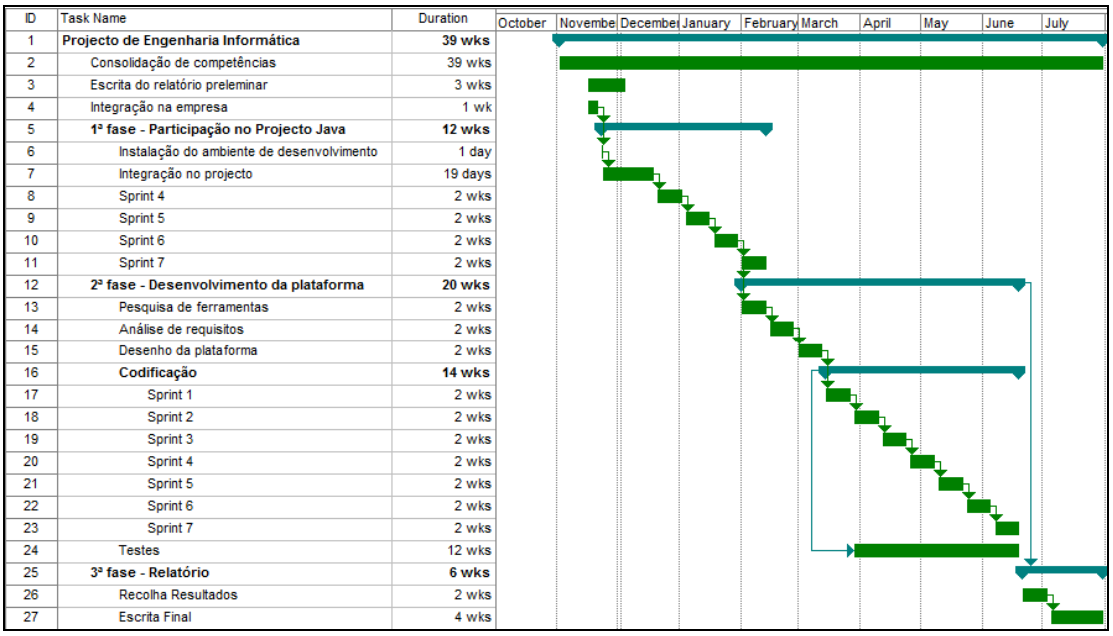
Capítulo 2

Planeamento e Metodologia

Neste capítulo começa-se por apresentar o planeamento inicial do trabalho e a comparação com a respectiva execução. Na segunda secção descreve-se a metodologia utilizada para a realização do projecto desenvolvido na segunda fase do estágio, para a qual foram seguidos os princípios descritos pela metodologia *scrum*, desde logo por ser este o tipo de projectos cuja execução se pretende suportar através da utilização da ferramenta desenvolvida.

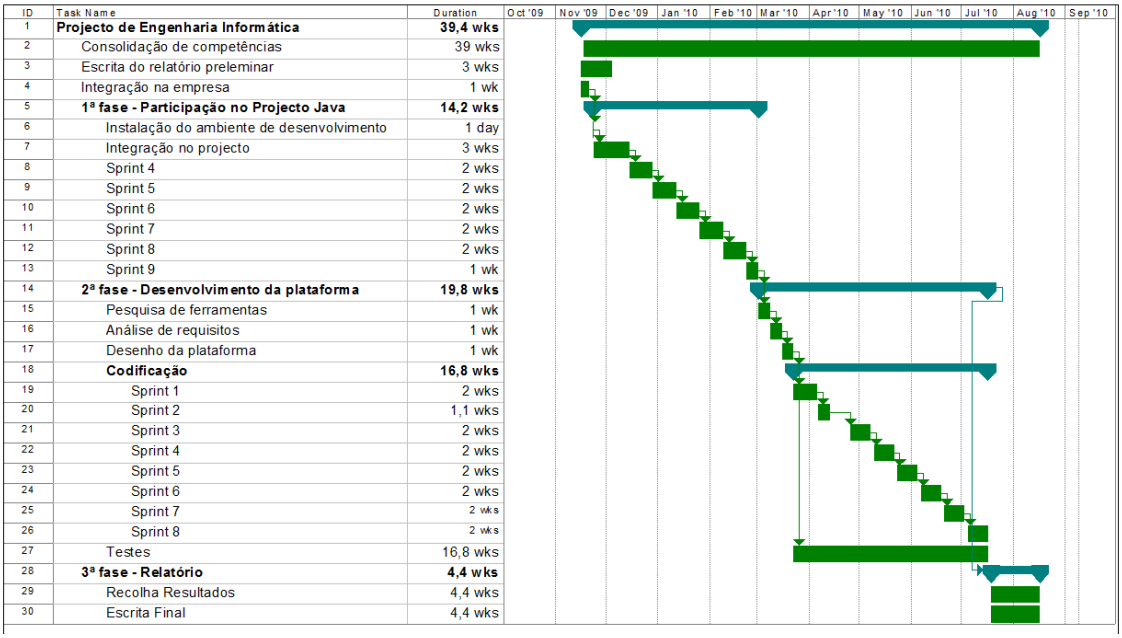
2.1 Planeamento

Na Ilustração 1 e na Ilustração 2 são respectivamente apresentados os diagramas de Gantt com o planeamento inicial do estágio e com o trabalho efectivamente realizado.



ID	Task Name	Duration
1	**Projecto de Engenharia Informática**	**39 wks**
2	Consolidação de competências	39 wks
3	Escrita do relatório preleminar	3 wks
4	Integração na empresa	1 wk
5	**1ª fase - Participação no Projecto Java**	**12 wks**
6	Instalação do ambiente de desenvolvimento	1 day
7	Integração no projecto	19 days
8	Sprint 4	2 wks
9	Sprint 5	2 wks
10	Sprint 6	2 wks
11	Sprint 7	2 wks
12	**2ª fase - Desenvolvimento da plataforma**	**20 wks**
13	Pesquisa de ferramentas	2 wks
14	Análise de requisitos	2 wks
15	Desenho da plataforma	2 wks
16	**Codificação**	**14 wks**
17	Sprint 1	2 wks
18	Sprint 2	2 wks
19	Sprint 3	2 wks
20	Sprint 4	2 wks
21	Sprint 5	2 wks
22	Sprint 6	2 wks
23	Sprint 7	2 wks
24	Testes	12 wks
25	**3ª fase - Relatório**	**6 wks**
26	Recolha Resultados	2 wks
27	Escrita Final	4 wks

Ilustração 1 – Mapa de Gantt: Planeamento inicial



ID	Task Name	Duration
1	**Projecto de Engenharia Informática**	**39,4 wks**
2	Consolidação de competências	39 wks
3	Escrita do relatório preleminar	3 wks
4	Integração na empresa	1 wk
5	**1ª fase - Participação no Projecto Java**	**14,2 wks**
6	Instalação do ambiente de desenvolvimento	1 day
7	Integração no projecto	3 wks
8	Sprint 4	2 wks
9	Sprint 5	2 wks
10	Sprint 6	2 wks
11	Sprint 7	2 wks
12	Sprint 8	2 wks
13	Sprint 9	1 wk
14	**2ª fase - Desenvolvimento da plataforma**	**19,8 wks**
15	Pesquisa de ferramentas	1 wk
16	Análise de requisitos	1 wk
17	Desenho da plataforma	1 wk
18	**Codificação**	**16,8 wks**
19	Sprint 1	2 wks
20	Sprint 2	1,1 wks
21	Sprint 3	2 wks
22	Sprint 4	2 wks
23	Sprint 5	2 wks
24	Sprint 6	2 wks
25	Sprint 7	2 wks
26	Sprint 8	2 wks
27	Testes	16,8 wks
28	**3ª fase - Relatório**	**4,4 wks**
29	Recolha Resultados	4,4 wks
30	Escrita Final	4,4 wks

Ilustração 2 – Mapa de Gantt: Execução do projecto

O trabalho realizado dividiu-se nas seguintes tarefas:

Consolidação de competências – tarefa contínua, com uma carga maior numa fase inicial, correspondente à aprendizagem e consolidação dos conhecimentos do aluno quer na linguagem de programação Java, quer na utilização de metodologia *scrum*.

Participação no projecto Java – integração num projecto em curso num cliente da Truewind, com o objectivo de consolidar as competências técnicas na linguagem

Java e na plataforma J2EE, e ainda de participar numa equipa em que foi utilizada a metodologia *scrum.*

A participação do aluno decorreu entre o início de Novembro de 2009 o início de Março de 2010.

No anexo **Outros Projectos**, são descritos de forma abreviada o âmbito do projecto, e a natureza das tarefas nas quais o aluno participou.

Desenvolvimento da plataforma – segunda fase do estágio correspondente ao desenvolvimento de uma plataforma de gestão e monitorização de projectos em que foi utilizada a metodologia *scrum*. Esta fase para além da codificação da ferramenta, incluiu ainda a pesquisa e documentação de ferramentas análogas já existentes no mercado, o levantamento de requisitos e o desenho da solução. Foi igualmente efectuada a análise da interface com a ferramenta de registo de tempos em utilização na empresa.

Iniciando-se em meados de Março de 2010 até Julho de 2010.

Relatório – Nesta fase final teve como objectivo a escrita do presente relatório descrevendo o trabalho realizado.

Iniciando-se em meados Julho de 2010 até Agosto de 2010.

A execução do projecto decorreu no essencial de acordo com o planeamento inicialmente proposto. Ainda assim, ocorreram alguns desvios relevantes:

- A participação do aluno na primeira fase prolongou-se por mais três semanas do que o inicialmente previsto, por contingências inerentes ao projecto no qual estava envolvido;
- Este atraso foi ainda assim compensado pelo tempo efectivamente gasto nas tarefas de pesquisa de ferramentas, análise de requisitos e desenho da plataforma, que se revelou inferior ao inicialmente proposto;
- Na fase de codificação o aluno demorou mais duas semanas do que inicialmente previsto, na medida em que foi necessário mais um *sprint* de forma resolver alguns obstáculos com que o aluno se deparou no decorrer da realização da aplicação.

2.2 Metodologia

Com base na percepção adquirida pelo aluno no decorrer do projecto no qual esteve envolvido em relação à aplicação em concreto da metodologia *scrum*, optou-se

pela adopção desta metodologia, com as necessárias adaptações, à gestão do processo de desenvolvimento da ferramenta construída na segunda fase deste estágio.

Apesar de esta metodologia estar orientada para a gestão de equipas, utilizaram-se ainda assim alguns dos princípios e conceitos mais importantes que são prescritos no *scrum*. Foram utilizados os conceitos de *product backlog*, de desenvolvimento iterativo organizados em *sprints* curtos, consubstanciados em *sprint backlogs* focados na entrega de funcionalidades potencialmente utilizáveis. Foram também realizadas *demo sessions* perante colaboradores da empresa envolvidos em projectos concretos, de forma a recolher o seu *feedback* no intuito detectar oportunidades de melhoria e de acolher as mudanças necessárias a uma melhor adequação da solução desenvolvida aos objectivos da empresa.

Na Tabela 2 é apresentado o *product backlog* criado inicialmente para a segunda fase do estágio.

ID	**Tipo de trabalho**	**Área de trabalho**	**Descrição**	**Pri.**[1]	**Est.**[2]
1	Tarefas de administração/ gestão	Documentação e relato	Escrita relatório	1	12
2	Funcionalidade	Interface com utilizador	Criação da CSS	1	3
3	Funcionalidade	Interface com utilizador	Criação do menu principal	1	3
4	Funcionalidade	Autenticação	O Utilizador consegue fazer *login*	1	5
5	Funcionalidade	Autenticação	O Utilizador consegue fazer *logout*	1	5
6	Funcionalidade	Gestão de Utilizadores	É possível Criar/Editar Utilizador	7	3

[1] Prioridade – de 1 a 10 (de prioridade muito baixa a prioridade muito alta)

[2] Estimativa de esforço – de 1 a 15 (de esforço reduzido a esforço muito elevado)

ID	Tipo de trabalho	Área de trabalho	Descrição	Pri.[1]	Est.[2]
7	Funcionalidade	Gestão de Projectos	É possível Criar/Editar Projecto	8	5
8	Funcionalidade	Gestão de Projectos	É possível Criar/Editar Equipa do projecto	7	3
9	Funcionalidade	Gestão de Projectos	Criar Equipa e associar ao projecto	7	5
10	Funcionalidade	Gestão de Projectos	Associar perfis aos elementos da equipa	5	3
11	Funcionalidade	Gestão de *Product Backlog*	É possível Criar/Editar Áreas de Trabalho	5	3
12	Funcionalidade	Gestão de *Product Backlog*	É possível Criar/Editar histórias	8	8
13	Funcionalidade	Gestão de Product Backlog	É possível Criar/Editar um Product Backlog	8	8
14	Funcionalidade	Gestão de *Product Backlog*	Exportar história para *SprintBacklog*	6	8
15	Funcionalidade	Gestão de *Product Backlog*	É possível Criar/Editar um *Sprint Backlog* utilizando as histórias e subdividindo estas em tarefas	8	8
16	Funcionalidade	Gestão de *Sprint Backlog*	É possível Criar/Editar Tarefas	8	8
17	Funcionalidade	Gestão de *Sprint Backlog*	É possível preencher horas diárias gastas numa tarefa	8	8
18	Funcionalidade	Gestão de *Sprint Backlog*	É possível exportar tarefas inacabadas para outro *sprint*	4	8

ID	Tipo de trabalho	Área de trabalho	Descrição	Pri.[1]	Est.[2]
19	Funcionalidade	Gestão de *Sprint Backlog*	É possível Criar/Editar capacidade diária da equipa durante o *sprint*	7	8
20	Funcionalidade	Gestão de *Sprint Backlog*	Visualizar uma análise de cada *sprint* (*burndown chart* e *cumulative flow chart*)	3	12
21	Funcionalidade	Gestão de *Sprint Backlog*	Visualização de um quadro geral do estado das tarefas de cada *sprint*	2	12
22	Funcionalidade	Gestão de Projectos	Exportação de dados	1	1
23	Funcionalidade	Gestão de *Product Backlog*	Exportação de dados	1	1
24	Funcionalidade	Gestão de *Sprint Backlog*	Exportação de dados	1	1
25	Funcionalidade	Módulo de Auditoria	Para cada registo, sempre que há uma alteração do mesmo, deve ser guardado quem fez a alteração, a data da alteração e, se possível a alteração realizada. Sendo possível visualizar estes registos	1	3
26	Funcionalidade	Gestão de *Sprint Backlog*	É possível saber o estado do *Sprint Backlog*	2	5
27	Funcionalidade	Gestão de *Sprint Backlog*	Saber horas gastas em cada tarefa, realização de	2	5

ID	Tipo de trabalho	Área de trabalho	Descrição	Pri.[1]	Est.[2]
			teste e histórias		
28	Funcionalidade	Gestão de *Sprint Backlog*	Criação de tarefas automáticas	4	5
29	Funcionalidade	Gestão de *Sprint Backlog*	É possível estimar a velocidade da equipa com base nos *sprints* executados	2	5
30	Funcionalidade	Módulo de Integração	Integração com o WTS de Projectos e Utilizadores	1	12

Tabela 2 – Product Backlog do projecto

Capítulo 3

Contexto Actual

3.1 A metodologia *Scrum*

O *scrum* é uma metodologia ágil usada principalmente em projectos com um grau de incerteza alto e com requisitos pouco definidos. Apesar da metodologia *scrum* ser utilizada principalmente neste tipo de projectos não quer dizer que não se possa utilizar *scrum* noutro tipo de projectos, um pouco menos rígidos mas com base nas principais características do *scrum*, designado como *"slow scrum"*.

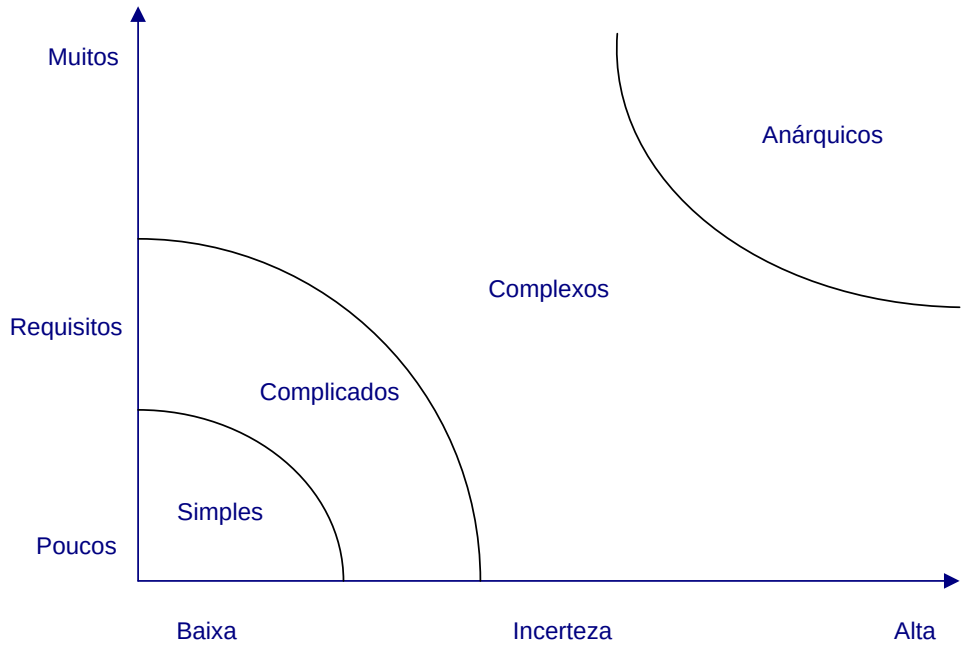


Ilustração 3 – Complexidade projectos

Devido ao alto nível de incerteza os riscos do projecto são aumentados, levando a situações em que, para se mitigarem estes riscos, são adoptadas metodologias que atrasam significativamente o *breakeven* do projecto. Seja por via da elaboração inicial de documentos de especificação muito detalhados, seja por via de atrasos no processo de desenvolvimento provocados por erros ou omissões nos documentos de especificação, seja por erros de estimação detectados em fase tardias do projecto. O *scrum* tenta mitigar os principais riscos do projecto:

- Risco de falhar as expectativas do cliente
    - Mitigado através de entregas frequentes e da recolha de *feedback* acerca das mesmas
- Risco de desfasamento das estimativas e planeamento
    - Mitigado através de constante prioritização e estimação
- Risco de não conseguir resolver os impedimentos prontamente
    - Mitigado por curtas reuniões de progresso diárias
- Risco de incapacidade de entrega do produto
    - Mitigado através de entregas regulares
- Risco de aceitação de compromissos quando não existe capacidade da parte da equipa
    - Mitigado através do comprometimento da equipa para a duração do *sprint*

Na Ilustração 4 é apresentado o ciclo de vida de um projecto seguindo a metodologia *scrum* incluindo os artefactos e intervenientes envolvidos durante a realização do projecto.

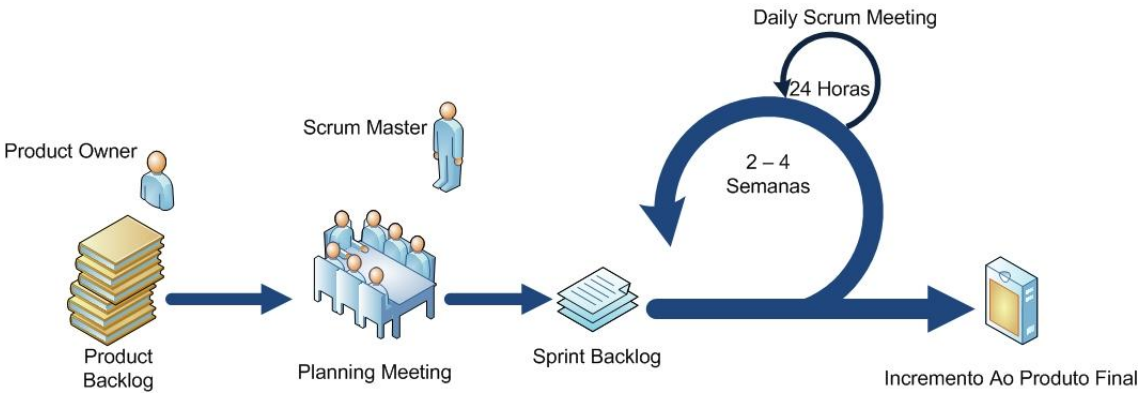


Ilustração 4 – Ciclo do *Scrum*

O *product backlog* consiste na lista de tarefas a efectuar no âmbito do desenvolvimento de uma aplicação ou sistema, descritas sob a forma de funcionalidades a serem suportadas. As funcionalidades são definidas como histórias (*user stories*); cada história deve ser escrita com uma linguagem simples descrevendo o comportamento que o sistema deve ter perante uma determinada acção do utilizador.

O *product owner* é o responsável por alimentar esta lista de tarefas para o desenvolvimento do produto, sendo também responsável por prioritizar e estimar cada tarefa bem como, monitorizar o desenvolvimento do projecto face aos objectivos e visão do produto.

O *scrum master* não deve ser considerado como um líder mas sim um representante da equipa perante o *product owner*. O *scrum master* tem como principais funções eliminar todos os obstáculos e dificuldades que a equipa tenha, coordenar os pedidos de alocações de tempo dos membros da equipa para as tarefas, monitorizar o decorrer das tarefas relativamente ao tempo estimado, reportar ao *product owner* a produtividade da equipa.

A execução de um projecto seguindo o *scrum* é baseado na organização das suas tarefas num conjunto de *sprints*, em que cada *sprint* é uma fase iterativa do projecto, tipicamente com uma duração de duas a quatro semanas, na qual a equipa se compromete a entregar um conjunto de funcionalidades a executar no decorrer de cada *sprint*. Cada *sprint* é iniciado com uma reunião de planeamento em que a equipa decide as funcionalidades a entregar no final do mesmo. De seguida existe outra reunião que define como cada tarefa será executada, sendo construído o *sprint backlog* onde as tarefas são detalhadas e distribuídas pelos elementos da equipa.

Diariamente durante cada *sprint* realiza-se uma reunião, *daily meeting*, entre os membros da equipa; esta não tem mais que quinze minutos e é moderada pelo *scrum master*. Cada membro da equipa reporta o que fez no último dia e o que se encontra a fazer nesse momento, quais os obstáculos e dificuldades com que se deparou; *o scrum master* está presente e tenta resolver ou auxiliar aliviando as dificuldades sentidas pelos membros da equipa. Estas reuniões servem também para que cada elemento da equipa tenha uma ideia geral do ponto de situação do projecto e de que tarefa cada elemento da equipa anda a realizar.

No último dia do *sprint* é demonstrado ao *product owner* todas as novas funcionalidades que foram concluídas durante o *sprint*, na *demo session*. Esta tem uma

importância fulcral, o *product owner* e restantes utilizadores finais envolvidos, estão em condições de desde logo dar o seu *feedback* relativamente às funcionalidades demonstradas, identificando nomeadamente necessidades e oportunidades de melhoria, numa fase muito precoce do projecto.

Por fim de cada *sprint* também é efectuada uma análise retrospectiva do *sprint* tentando detectar os erros e as falhas cometidas de forma a corrigir e melhorar o desenrolar do *sprint* seguinte.

Para auxiliar a equipa ao longo do *sprint* para que este seja concluído com sucesso, permitindo aferir do seu desempenho e cumprimento do plano estabelecido, existem dois gráficos que são construídos e alimentados ao longo do decorrer do *sprint*. *Burndown Chart* (Ilustração 5) e *Cumulative Flow* (Ilustração 6).

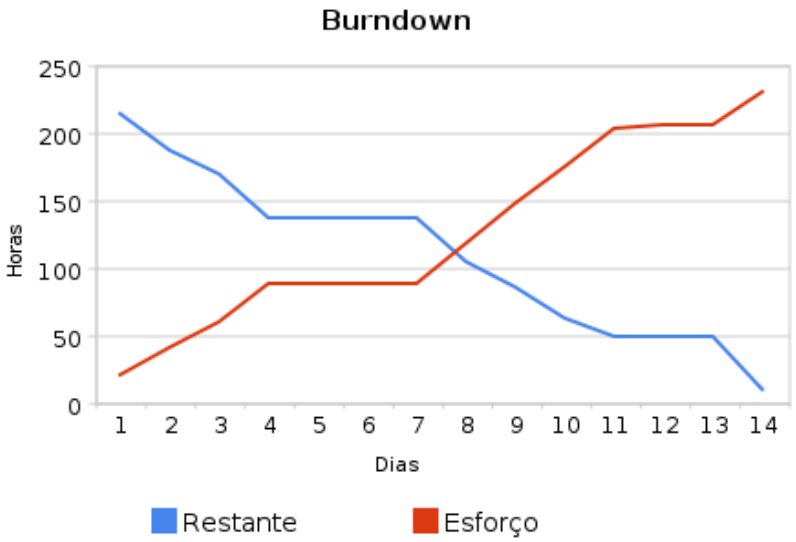


Ilustração 5 – Gráfico *Burndown*

Este gráfico é inicialmente calculado com base no esforço estimado para todas as tarefas incluídas no *sprint backlog*, sendo actualizado no decorrer do *sprint* com o esforço efectivamente registado pelos elementos da equipa e pelas tarefas concluídas. Desta forma permite aferir o desempenho da equipa face à estimativa inicial comparando o esforço efectivamente aplicado com o estimado. A situação ideal seria que no último dia do *sprint* as horas restantes fosse igual a zero, isto é, todas as tarefas concluídas com sucesso, e o esforço igual ao somatório inicialmente estimado, o que reflectiria numa boa estimativa da parte da equipa.

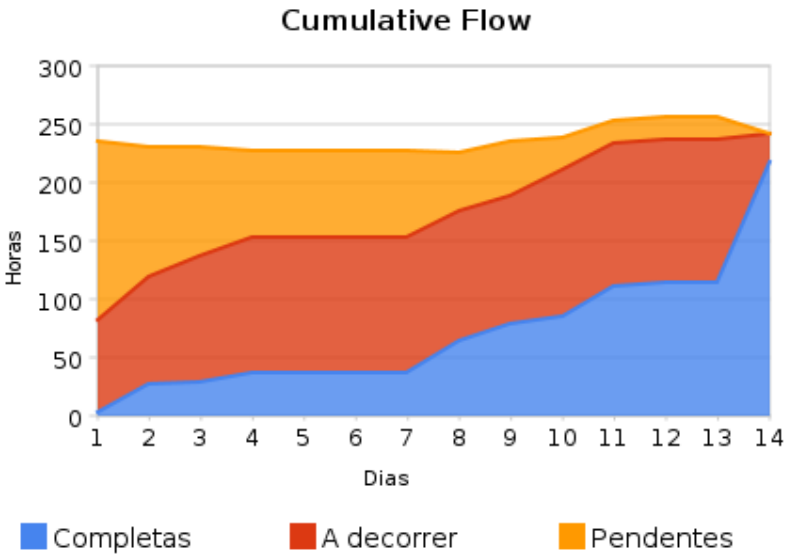


Ilustração 6 – Gráfico *Cumulative Flow*

Este gráfico representa o esforço total para as tarefas que se encontram completas, a decorrer ou que ainda não foram iniciadas no decorrer do *sprint*. Como podemos ver na imagem Ilustração 6, no fim deste *sprint* não estavam todas as tarefas completas ficando algumas tarefas por terminar, o que é de imediato visível neste gráfico, por não haver no último dia o ponto de intercepção das três rectas.

3.2 A sua aplicação prática

O *scrum* não acrescenta no essencial novidades significativas relativamente às metodologias tradicionais, baseando-se acima de tudo num conjunto de boas práticas sobre os aspectos cruciais para o sucesso da implementação de um projecto de software.

Analisando a forma de aplicação desta metodologia no projecto no qual o aluno esteve envolvido, é de relevar a existência de algumas carências na adopção e utilização da metodologia, muitas vezes associada às ferramentas utilizadas para a gestão do processo de desenvolvimento. A Truewind utiliza um conjunto de folhas de cálculo que consolidando a informação relativa quer ao *product backlog*, quer aos *sprint backlogs*, são igualmente utilizadas para preenchimento por parte de todos os elementos envolvidos na equipa, dos dados referentes à execução do projecto. Detectaram-se desde logo as seguintes fragilidades:

- A utilização de uma folha de cálculo numa lógica partilhada acarreta um conjunto de problemas, por um lado, ao nível do seu acesso concorrente e, por outro, no garante da consistência intrínseca da folha de cálculo e da informação registada. Por exemplo, um dos problemas recorrentes prende-se com limitações do *template* utilizado face à necessidade de inserção de tarefas ao longo de um *sprint*;
- Inexistência de um rastreio eficaz de uma tarefa, isto é, caso uma tarefa não seja concluída durante o *sprint*, sendo a sua execução continuada no *sprint* seguinte, não existe forma de saber o esforço e duração efectiva da tarefa;
- A inexistência de uma forma de ligação coerente entre *sprints*;
- Erros na estimação da duração das tarefas realizadas pelos membros da equipa, são deficientemente evidenciados podendo prejudicar a execução do projecto;
- Existência de inconsistências entre os *sprint backlogs* e o *product backlog*, originados por via da manipulação das folhas de cálculo;
- Dificuldade em saber a duração real do tempo consumido na realização de testes;
- Inexistência de interligação com as ferramentas de testes unitários e integração contínua em utilização no projecto, dificultando a existência de uma visão única e expedita do estado do processo de desenvolvimento, quer numa perspectiva quantitativa, quer numa perspectiva qualitativa;
- Obrigatoriedade de um duplo registo, por parte dos membros da equipa, do esforço aplicado na execução das tarefas do projecto, com o intuito de garantir o cumprimento dos procedimentos internos da Truewind que exigem o registo das horas gastas na ferramenta utilizada internamente para efeitos de registos de tempo.

3.3 Estado da Arte

Foi inicialmente elaborado um levantamento e análise de algumas das aplicações já existentes para gestão e planeamento de projectos utilizando a metodologia *scrum*.
O objectivo deste levantamento foi o de analisar a maturidade das ferramentas disponíveis, e o de identificar funcionalidades a incluir na aplicação desenvolvida.

Na Tabela 3 é apresentado um quadro síntese, onde se identificam as funcionalidades e características consideradas como mais relevantes das aplicações analisadas; o quadro apresentado não é exaustivo, na medida em que apresenta apenas as funcionalidades consideradas como fundamentais para o âmbito deste projecto.

Não sendo possível retirar uma conclusão definitiva com base neste quadro, na medida em que não são, por exemplo, aferidas questões como a complexidade de utilização de cada uma das suas ferramentas ou o seu custo de licenciamento e manutenção, permite ainda assim, identificar um conjunto de funcionalidades que sendo suportadas por todas ou por algumas das soluções analisadas, foram consideradas como relevantes para inclusão na solução desenvolvida.

	Sprintometer	***Scrumninja***	***Firescrum***	***Scrumdesk***	***Agileexpress***	***Acunote***	***Agilebuddy***
Open source	✗	✗	✓	✗	✓	✗	✗
Autenticação de utilizadores	✗	✗	✓	✓	✓	✓	✓
Criar e editar projectos	✓	✓	✓	✓	✓	✓	✓
Criar e editar *user stories*	✓	✓	✓	✓	✓	✓	✓
Criar e editar tarefas	✓	✓	✓	✓	✓	✓	✓
Diferenciação de tarefas de codificação e tarefas de teste	✓	✗	✗	✗	✗	✗	✗
Criar e editar *product backlog*	✓	✓	✓	✓	✓	✓	✓
Criar e editar *sprint*	✓	✓	✓	✓	✓	✓	✓
Registo diário de horas	✓	✓	✗	✗	✗	✓	✗
Modo *Post-it*	✗	✓	✓	✓	✓	✗	✗

	Sprintometer	***Scrumninja***	***Firescrum***	***Scrumdesk***	***Agileexpress***	***Acunote***	***Agilebuddy***
Planning Poker	✗	✗	✓	✓	✗	✗	✗
Integração com outras aplicações	✗	✗	✗	✓	✗	✓	✗
Criação de gráficos de análise	✓	✓	✗	✓	✗	✓	✗
Exportação para *Microsoft Excel*	✓	✗	✗	✓	✗	✗	✗

Tabela 3 – Diferenças entre as aplicações existentes

Para além das funcionalidades fundamentais associadas à gestão de utilizadores, de projectos, do *product backlog* e dos *sprint backlogs*, destacamos as abaixo enumeradas, por terem sido consideradas relevantes para o âmbito do projecto:

- **Diferenciação de tarefas de codificação e tarefas teste** – facilita a monitorização e análise do estado de evolução de um projecto, através do registo do detalhe do tempo gasto exclusivamente em tarefas de teste, mitigando desta forma uma das limitações inicialmente identificadas, associada ao controlo do esforço aplicado neste tipo de tarefas; presente apenas numa das aplicações analisadas; foi considerada ainda assim como relevante para a solução desenvolvida;
- **Modo *post-it*** – Outra da funcionalidade identificada é o suporte a uma prática comum na utilização de *scrum*, baseada na utilização para cada s*print* de um quadro com *post-it*, em que estes normalmente contêm as histórias ou tarefas a executar durante o *sprint*.

  Cada *post-it* tem a descrição da história ou tarefa e os pontos ou as horas estimadas, podendo ainda estar divididos por cores, representando cada módulo (área de trabalho) do projecto, ou o elemento da equipa que vai realizar a tarefa ou a história. Normalmente este quadro está dividido em três colunas: Por fazer, a decorrer e concluído. Os *post-its* são movidos para as

colunas correspondentes ao longo do *sprint* conforme a evolução do seu estado.

A fácil utilização dos *post-it*, e o seu baixo custo, dá uma percepção rápida e prática de como decorre o *sprint*, podendo ser uma mais-valia a reprodução na aplicação.

- ***Poker planning*** – Outra da funcionalidade identificada foi o uso de *poker planning*, à imagem do apresentado na Ilustração 7. Trata-se de uma forma muito utilizada para estimar (normalmente em pontos) o esforço associado a uma *user story*. As cartas do *poker planning* são baseadas na sequência *Fibonacci*. A carta da chávena de café representa que o elemento da equipa precisa de uma pausa e o ponto de interrogação representa que o elemento da equipa não tem a certeza da estimativa. Cada elemento da equipa tem um baralho deste. É seguido o procedimento descrito de seguida.

  Neste processo o *scrum master* detém normalmente o papel de moderador. Após todos os elementos da equipa tomarem conhecimento da *user story* que estão a estimar escolhem uma carta sem a mostrar ao resto da equipa. Após o tempo ter terminado mostram todos a sua carta.

  Se houver consenso entre todas as estimativas para essa história então a história fica estimada. Se os valores forem diferentes então cada elemento justifica o porque da sua decisão e volta-se a repetir o processo, em que cada elemento da equipa escolhe uma carta, tendo agora em conta o que os elementos da equipa disseram anteriormente para justificar os valores que escolheram, até que haja consenso, ou o moderador decida qual a estimativa adequada.

  Uma das vantagens de usar o *planning poker* é a inexistência de todos os valores nas cartas (por exemplo o 4) os elementos da equipa ao quererem escolher uma carta que não existe tendem a escolher a carta mais pessimista, o que por vezes é conveniente por se ajustar mais com a realidade dos projectos. Outra vantagem é que os elementos da equipa não são influenciados inicialmente na escolha da estimativa, por escolherem uma carta sem saber o que os outros estão a escolher, o que pode não acontecer com o método tradicional (neste, todos os elementos da equipa emitem

sucessivamente a sua opinião acerca da estimativa, sendo sempre influenciados pelas estimativas já conhecidas).

O *planning poker* poder ser realizado fisicamente com cartas ou realizado via aplicações Web ou ferramentas desenvolvidas com este propósito.

Ilustração 7 – *Planning Poker*

Na Tabela 4 são apresentadas, para as aplicações analisadas que foram desenvolvidas no regime de *open source*, as linguagens de programação / plataformas utilizadas na sua criação. Na medida em que um dos objectivos estratégicos para a instituição de acolhimento, passava pela formação do aluno na linguagem de programação Java, foi desde logo colocada de lado a hipótese de o desenvolvimento da solução apresentada neste relatório ter por base alguma das ferramentas abaixo listadas. Desta forma, optou-se pela construção de uma solução de raiz, orientada para as necessidades concertas da instituição de acolhimento.

	Linguagem / Plataforma
Firescrum	ActionScript, C#, Java
Agileexpress	Ruby on Rails

Tabela 4 – Linguagens de programação das soluções *open source*

Capítulo 4

Tecnologias

Neste capítulo são enumeradas e descritas as principais componentes e ferramentas utilizadas para o desenvolvimento da aplicação, sendo igualmente identificado o seu papel no seu processo de construção.

A plataforma Java Enterprise Edition (JEE) desempenhou um papel central na construção da aplicação, ao fornecer a infra-estrutura base de desenvolvimento assentando sobre o servidor aplicacional. A escolha da utilização desta plataforma prendeu-se por um lado com a sua adequação para a construção da solução pretendida, e por outro, como forma de reutilizar e aprofundar os conhecimentos adquiridos durante a participação no projecto na primeira fase do estágio.

A plataforma JEE

A plataforma JEE é considerada um padrão desenvolvimento que fornece funcionalidades para a construção em Java de aplicações distribuídas, tolerantes a falhas e multi-camada. Esta plataforma está orientada para a criação de aplicações multi-camada baseadas em componentes que são executados num servidor aplicacional.

Alguns benefícios:

- Suporte completo a *web services*.
- Utiliza "*containers*" como forma de simplificação do desenvolvimento. Java EE Containers providenciam a separação entre lógica de negócio, recursos e gestão do ciclo de vida, permitindo ao programador focar-se apenas na parte lógica de negócio.
- Simplifica a conectividade. A plataforma simplifica a conectividade entre a aplicação e os outros sistemas facilitando a disponibilização desses recursos para ambientes *web*, dispositivos móveis, etc.

Enumeram-se de seguida alguns componentes disponibilizados pela plataforma JEE utilizados no âmbito deste projecto.

Java Message System API (JMS)

O Serviço JavaMessage (JMS) API é um padrão de mensagens que permite que aplicações Java EE possam criar, enviar, receber e ler mensagens. Permite a comunicação distribuída, de forma flexível, confiável e assíncrona.

JMS API foi criada pela *Sun* e diversas empresas parceiras, definindo um conjunto comum de interfaces e respectiva semântica permitindo que as aplicações desenvolvidas em Java consigam comunicar e interagir com outras implementações de serviços de mensagens.

JMS API minimiza o conjunto de conceitos que um programador deve dominar, oferecendo ainda assim recursos suficientemente sofisticados para suportar aplicações baseadas em mensagens. Esforça-se igualmente por maximizar a portabilidade de aplicações através de fornecedores JMS.

A JMS API permite a comunicação de uma forma não apenas flexível, mas também:

- Assíncrona: Um fornecedor JMS pode entregar mensagens que recebe para um cliente, um cliente pode não ter de as solicitar a fim de recebê-las.
- Confiável: A JMS API pode garantir que uma mensagem é entregue apenas uma única vez, baixa a complexidade de construção para aplicações que se podem dar ao luxo de perder as mensagens ou de receber mensagens duplicadas.

Enterprise Java Beans 3.0 (EJB)

Enterprise Java Beans (EJB) é um dos principais componentes suportado pela plataforma JEE. É um componente do tipo servidor que executa no contentor (*container*) do servidor aplicacional. Os principais objectivos da tecnologia EJB são fornecer uma forma rápida e simplificada de desenvolvimento de aplicações Java baseada em componentes distribuídas, transaccionais, seguras e portáveis.

A plataforma J2EE providencia componentes dedicadas à camada de lógica de negócio e à de acesso a repositórios de dados. Os EJBs são classes Java que, respeitando determinados interface encapsulam a lógica de negócio de uma aplicação. Através do EJB o programador foca-se na funcionalidade que deseja implementar e libertando-se de questões que pode encaminhar para o contentor de EJBs, como por exemplo a gestão transaccional. Estas funcionalidades oferecidas pelo contentor aplicacional representam uma mais-valia significativa no processo de desenvolvimento.

Existem dois tipos fundamentais de EJB:

Session Beans

Executa uma tarefa para o cliente. Pode manter ou não o estado durante uma sessão com o cliente. *Session bean* representa um único cliente no servidor aplicacional. Para aceder às aplicações o cliente invoca o *session bean*, que será responsável pela realização do trabalho para o seu cliente, diminuindo a sua complexidade e executando tarefas de negócio dentro do servidor.

Características *session bean*:

- Não é partilhado;
- Só pode ter um cliente;
- Não é persistente;
- Implementam funcionalidades síncronas.

Existem dois tipos de *session bean*:

- Sem persistência de estado (*Stateless*) – não mantém nem conserva o estado do cliente. Quando o cliente invoca um método, instância as variáveis que especificam o estado do cliente, quando o método acaba o estado do cliente não é mantido. Contudo o cliente pode alterar o seu estado e associá-lo ao *bean*, tratando-se neste caso de uma responsabilidade do cliente. Devido a sessão suportar múltiplos clientes sem preservar o estado este tipo garante uma maior escalabilidade para as aplicações.
- Com persistência de estado (*Statefull*) – o estado do cliente é mantido ao longo da sessão. O cliente interage com o *bean* de uma forma

transparente, sendo mantido o estado de conversação até que o cliente remova o *bean*.

Message Driven Beans (MDB)

Processa mensagens de modo assíncrono entre os EJBs e a API de mensagens (*Java Message Service* – JMS).

Normalmente actua como *JMS Message Listener*, semelhante a um *Event Listener* mas em vez de receber eventos, recebe mensagens JMS. As mensagens podem ser enviadas por qualquer componente Java EE, por exemplo, um cliente de uma aplicação, outro *bean* ou como componente Web, por uma aplicação JMS ou sistema um que não utilize tecnologia Java EE.

Este tipo de *bean* assemelha-se ao *Session Bean* sem persistência estado (*Stateless*):

- MDB não retêm qualquer estado ou especificação do cliente;
- Todas as instâncias do MDB são equivalentes. Permitindo atribuir uma mensagem a qualquer instância;
- Um MDB consegue processar várias mensagens de vários clientes.

Características MDB:

- Executam após a recepção da mensagem de um único cliente;
- São chamados de forma assíncrona;
- Curta duração;
- Não representam os dados directamente na base de dados, mas podem aceder e actualizar esses dados;
- Não matem estado.

Java Persistence API (JPA)

A JPA define um mapeamento objecto-relacional para objectos Java simples e comuns (*Plain Old Java Objects* - POJPs).

Normalmente uma entidade representa uma tabela da base de dados, cada instância da entidade corresponde a uma linha da tabela. *Entity Class* é uma classe Java que relaciona a classe com a tabela da base de dados e os atributos da classe aos campos da tabela.

A utilização do JPA é bastante vantajosa por não só fazer o mapeamento entre as tabelas, das suas relações e heranças.

A realização de interrogações a base de dados pode ser feita de duas maneiras:

- *Java Persistence Query Language* (JPQL) – permite a realização de interrogações estáticas e dinâmicas à base de dados por meio de outras classes já criadas;
- *SQL nativo* – linguagem de interrogação implementada pelo sistema relacional de suporte.

```
@Entity
@Table(name="TAREFA",schema=ScrumConstants.SCRUM_DEFAULT_SCHEMA)
@SequenceGenerator(name = "SequenceIdGenerator", sequenceName =
ScrumConstants.SCRUM_DEFAULT_SCHEMA_WITH_POINT + "TAREFA_SEQ")
@PersistenceUnit(name=PUList.SCCT)
@NamedQueries({@NamedQuery(name = Tarefa.FIND_LAST_ID, query = "SELECT
MAX(t.idLinha) FROM Tarefa t WHERE t.sprintBacklog.idSprintBacklog =
:idSprintBacklog"),
…})
public class Tarefa implements Serializable{
	…
	public static final String FIND_LAST_ID = "Tarefa.findLastId";

	@Id
	@GeneratedValue(generator = "SequenceIdGenerator")
	@Column(name="ID_TAREFA")
	private Long idTarefa;

	@Column(name="ID_LINHA")
	private int idLinha;

	@ManyToOne
	@JoinColumn(name="ID_SPRINT_BACKLOG")
	private SprintBacklog sprintBacklog;

	…
	public Tarefa() {
		super();
	}
…
```

```
}
```

Exemplo 1 – Excerto de um exemplo da utilização JPA

Java Server Faces (JSF)

Para poder caracterizar esta tecnologia é necessário compreender outras. Na Ilustração 8 são esquematizadas estas tecnologias e as suas relações.

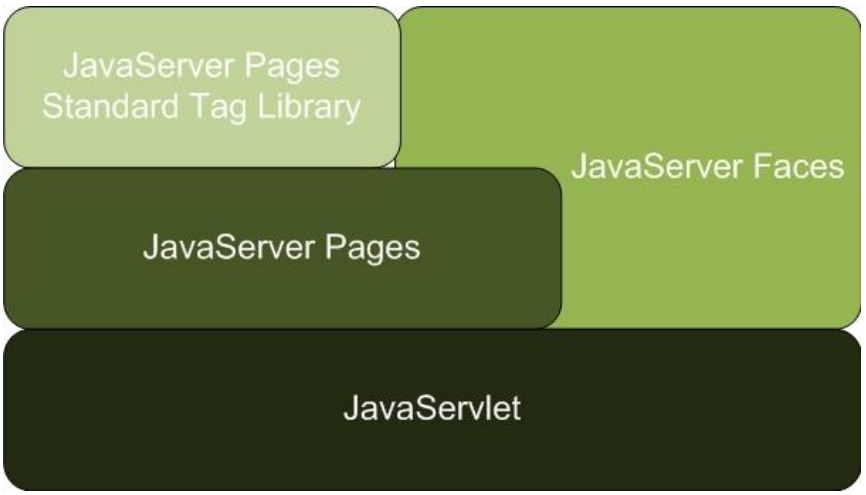


Ilustração 8 – Tecnologias de Aplicações Web

Servlet

Uma *servlet* é uma classe Java utilizada para estender as capacidades disponibilizadas pelo servidor aplicacional, a informação dinâmica é processada por meio de um modelo pedido-resposta.

Embora as *servlets* possam responder a qualquer tipo de solicitação, são normalmente usadas para aplicações Web.

Para essas aplicações, a tecnologia Java Servlet HTTP define classes *servlet* específicas. Todas as *servlets* devem implementar a interface *Servlet*, que define os métodos de ciclo de vida.

JavaServer Pages (JSP)

Tecnologia utilizada no desenvolvimento de aplicações para Web. Uma das principais vantagens é a portabilidade de plataforma, que permite a sua execução em diversos sistemas operativos. Esta tecnologia permite ao programador produzir aplicações que acedam a repositórios de dados, manipulem arquivos no formato texto,

capturem informações a partir de formulários e captem informações sobre o visitante e sobre o servidor.

Uma página criada com a tecnologia JSP, após a sua instalação num servidor aplicacional compatível com a tecnologia Java EE, é transformada numa *Servlet*.

JavaServer Faces (JSF)

Mais que uma ferramenta para desenvolver aplicações Web, trata-se de uma infra-estrutura sobre a se podem criar aplicações. As principais componentes de JSF incluem:

- API para representar os componentes da interface e gerir o seu estado; tratamento de eventos, a validação do lado do servidor e conversão de dados, definição de navegação da página, suporte à internacionalização e a mecanismos de acessibilidade; fornece extensibilidade para todas estas características;
- JavaServer Pages (JSP) bibliotecas de etiquetas personalizadas para expressar os componentes da interface dentro de uma página JSP e de componentes para a associar os objectos no lado do servidor.

Dadas estas duas características, é obrigatório existir uma separação entre apresentação, o código de tratamento de dados, e o controlo da navegação. JSF é baseado no padrão de arquitectura de software *Model-view-controller* (MVC):

- *Vista* – Renderiza as componentes do modelo. Recebe os dados do modelo e especifica como os dados devem ser apresentados. Actualiza a apresentação dos dados quando o modelo é alterado. É materializada pelo conjunto de páginas JSP.
- *Modelo* – Representa a lógica de negócio. Trata da informação em função dos eventos enviados pelo utilizador através da vista.
- *Controlador* – Define o comportamento da aplicação. Despacha os pedidos do utilizador e selecciona a vista que o vai apresentar. Interpreta a entrada dos dados do utilizador mapeia para eventos a serem realizados pelo modelo

JSF inclui:

- Um conjunto de páginas JSP (de utilização opcional na camada de apresentação);
- Um conjunto de "*backing beans*", que são *Java Beans* que definem as propriedades dos componentes e funções em cada uma das páginas;
- Um ficheiro de configuração da aplicação que define regras de navegação, validação e os "*backing beans*" que suportam a aplicação;
- Um descritor de implementação;
- Suporte para conjuntos de objectos personalizados criados pelo programador. Estes objectos podem incluir componentes personalizados, validadores e conversores;
- Um conjunto de etiquetas personalizadas que representam os objectos criados pelo utilizador.

Benefícios da componente JSF:

Uma das maiores vantagens da tecnologia JSF é que oferece uma clara separação entre o comportamento e apresentação. As aplicações Web que utilizam a tecnologia JSP conseguem essa separação em parte. No entanto, uma aplicação JSP não consegue mapear os pedidos HTTP para o componente de manipulação de evento específico, nem gerir elementos da interface como objectos mantendo o seu estado no servidor, características suportadas por uma aplicação JavaServer Faces.

A separação da lógica de apresentação também permite que cada membro da equipa de desenvolvimento de aplicações Web apenas realize a sua parte de desenvolvimento, fornecendo um modelo de programação simples, que permitir agregar posteriormente o código pelos diversos membros da equipa.

Mais importante ainda, a tecnologia JSF oferece uma forte arquitectura para gerir o estado dos componentes, componente de processamento de dados, validação de entrada do utilizador e eventos de manipulação.

ICEfaces

ICEfaces é uma *framework* Ajax de código aberto que permite que programadores de aplicações Java EE criem e implantem aplicações ricas para Internet utilizando a linguagem Java.

O *ICEfaces* aproveita todos os padrões de ferramentas e ambientes de execução baseados em Java EE. Aplicações em *ICEfaces* são aplicações *JavaServer Faces* (JSF);

desta forma as características do desenvolvimento de aplicações Java EE aplicam-se directamente e os programadores Java ficam livres de fazer qualquer desenvolvimento relacionado com *JavaScript*.

Spring Security

O *Spring* security é uma plataforma desenvolvida para JEE que fornece recursos avançados de autenticação e autorização em aplicações JEE para Web. *Spring Security* sendo fácil de aprender e gerir, fornece directrizes para a maioria das operações comuns, permitindo aplicações completamente seguras em apenas algumas linhas de XML. Sendo uma plataforma bastante eficiente é hoje utilizada em vários ambientes críticos como agências do governo, aplicações militares ou bancos centrais.

Lightweight Directory Access Protocol (LDAP)

Lightweight Directory Access Protocol, ou LDAP, é um protocolo para actualizar e pesquisar directórios sobre o protocolo TCP/IP. Um directório LDAP geralmente segue o modelo X.500, que é uma árvore de nós, cada um consiste num conjunto de atributos com seus respectivos valores. O LDAP foi criado como uma alternativa ao muito mais incómodo *Directory Access Protocol* (DAP).

JUnit

O JUnit é uma plataforma *open-source* com o objectivo de dar suporte à criação de testes automatizados na linguagem de programação Java. A utilização desta plataforma facilita a criação de código para a automação de testes. É possível validar os métodos para verificar se funcionam da forma esperada, exibindo possíveis erros ou falhas.

Para cada método o programador submete-o a um conjunto de estímulos referindo os dados de entrada a passar por cada estímulo e o comportamento esperado como dados de saída. As entradas são parâmetros e as saídas são o valor de retorno, excepções ou o estado do objecto. Tipicamente um teste unitário executa um método individualmente e compara uma saída conhecida após o processamento da mesma.

JMock

JMock é uma biblioteca que suporta *Test-Driven Development* de código Java com objectos *mock*, ou seja permite criação de objectos que simulando o comportamento de objectos reais de forma controlada, permitem testar o comportamento do objecto real, de uma forma rápida e simples.

Em testes unitários, *mock objects* são muito úteis quando os objectos reais são difíceis de criar ou impossíveis de serem incorporados em testes unitários.

O JMock gera automaticamente estes objectos partindo de uma especificação do seu interface dada pelo programador, sendo também possível especificar o comportamento que se pretende que os objectos simulados tenham, indicando as mensagens que este interface deve fornecer quando vier a ser invocado pelo objecto que se pretende testar.

A biblioteca JMock:

- Torna mais rápida e fácil de definição de objectos fictícios, para que não quebre o ritmo da programação;
- Permite especificar com precisão as interacções entre os objectos, reduzindo a fragilidade dos testes;
- É fácil de estender.

TruewindEjb3Unit

Esta é uma biblioteca desenvolvida pela Truewind que visa auxiliar a realização de testes para EJB. Os EJBs utilizam recursos que são injectados pelo servidor aplicacional. Realizar os testes fora desse contexto requer que a plataforma de testes se substitua ao servidor aplicacional, inicializando estes recursos e injectando-os nos EJBs. Esta biblioteca analisa o EJB que se pretende testar, inicializa os recursos requeridos e injecta-os no EJB, simplificando a realização dos testes.

Esta biblioteca substitui-se também à geração transaccional oferecida pelo servidor aplicacional tratando as transacções de teste. Sempre que é iniciado um teste é criada uma nova transacção, o teste é executado e aferem-se as expectativas contra os resultados apurados, caso o teste tenha alterado dados (por exemplo alterando ou inserindo valores numa base de dados ou registando uma mensagem numa fila) reverte todas as alterações realizadas, repondo o estado original do ambiente antes da execução do teste, preservando a informação armazenada.

São também disponibilizados nesta biblioteca utilitários para inspeccionar e manipular os objectos que se pretendem testar (por exemplo inspeccionando os valores de atributos privados ou invocando métodos privados), permitindo validar o funcionamento dos objectos e reduzir a granularidade das unidades testadas sem alterar o interface dos objectos.

Capítulo 5

Arquitectura e Descrição Funcional

Neste capitulo descreve-se a aplicação desenvolvida, quais os seus objectivos e as funcionalidades executadas.

5.1 Descrição Geral

A aplicação desenvolvida tem como propósito servir como ferramenta de gestão e monitorização dos projectos da empresa que utilizam a metodologia *scrum*. Para tal a aplicação baseia-se nos artefactos descritos na metodologia *scrum.* Deve permitir que os seus utilizadores registem projectos, que cada projecto associe um *product backlog*, onde são descritas todas as funcionalidades que serão implementadas. Após associado o *product backlog* é necessário criar os *sprint backlogs*, que contêm as histórias descritas no *product backlog*, decompostas em tarefas a realizar em cada ciclo. São criados tantos *sprints* quantos necessários até o projecto terminar.

Um dos grandes objectivos da aplicação é o de os utilizadores registarem o esforço consumido na execução de cada tarefa, permitindo com este registo acompanhar e analisar a execução de cada *sprint*, nomeadamente através dos gráficos *Burndown Chart* e *Cumulative Flow*.

5.2 Arquitectura da aplicação

A arquitectura da aplicação definida com base nos conhecimentos adquiridos no decorrer do projecto no qual o aluno esteve integrado na primeira fase do estágio.

A arquitectura do sistema encontra-se dividida em três módulos:

- Negócio
- Apresentação

- Modelo/Comuns

Na Ilustração 9 é apresentada a arquitectura da aplicação, cujos módulos são descritos ao longo das próximas secções.

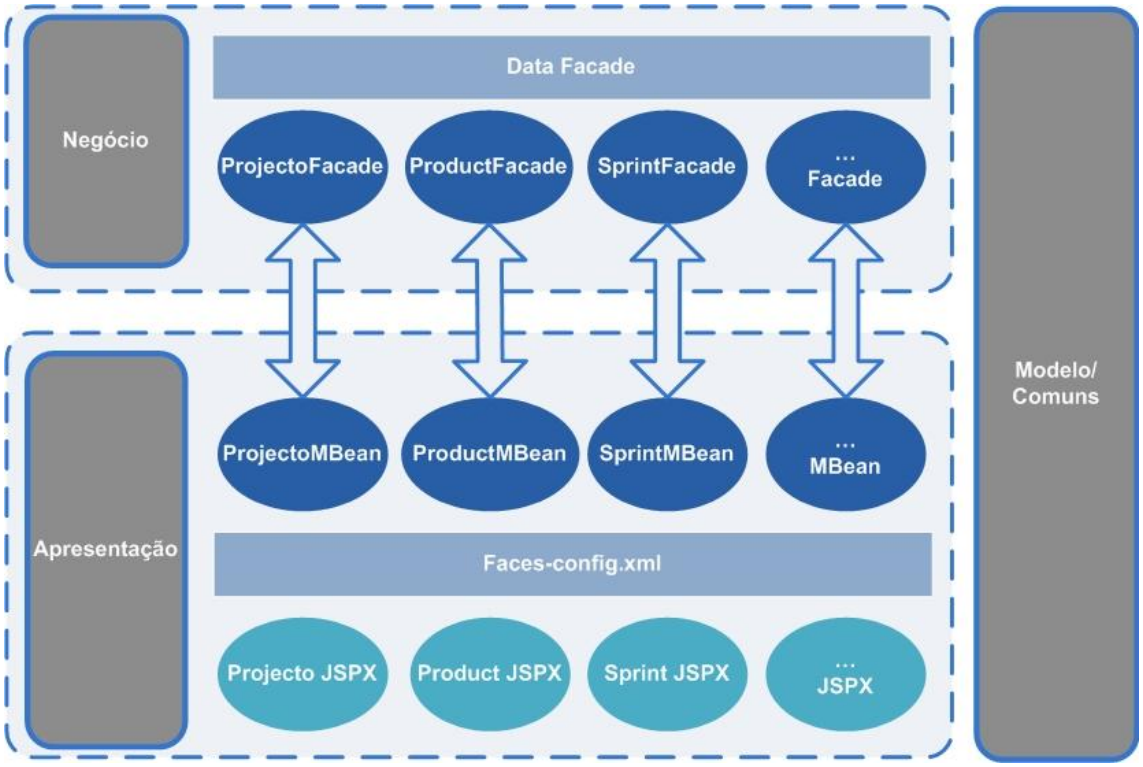


Ilustração 9 – Arquitectura da Aplicação

5.2.1 Negócio

Este módulo tem como principal objectivo implementar as regras de negócio usando EJBs responsáveis pela manipulação necessária de dados.

Este módulo foi criado a partir do padrão de desenho *Facade*. Este padrão oferece uma interface unificada para um conjunto de interfaces de um subsistema, ou seja, é definida uma interface de nível mais alto que torna o subsistema mais fácil de ser utilizado pelos seus clientes, protegendo-os da complexidade dos componentes do subsistema.

Para atingir este objectivo criou-se, conforme é ilustrado na imagem, uma *Facade* para cada módulo da aplicação, nomeadamente para o módulo de gestão de projectos, gestão do *product backlog*, etc. Cada uma destas classes é disponibilizada em forma de EJB e utilizada pela classe correspondente na camada de apresentação.

Foi criada uma *Facade* específica que é unicamente usada de forma interna; denominada como *DataAccessFacade*, distingue-se por ser a única que interage com o repositório de dados. Todas as outras *Facades* utilizam esta para quando necessitam de realizar operações sobre os dados, encapsulando os detalhes do acesso à base de dados de suporte ao negócio.

5.2.2 Apresentação

Esta camada foi desenvolvida segundo o padrão *Model View Controller* (MVC), representado na imagem abaixo disponibilizado pelo JSF.

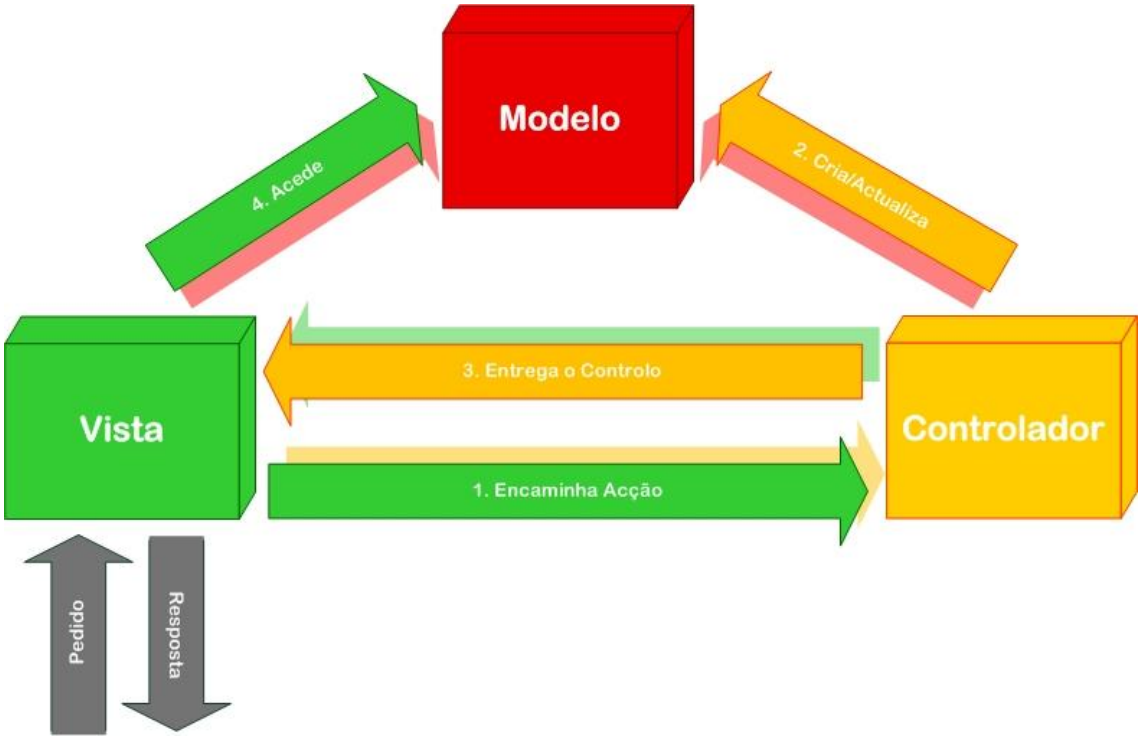


Ilustração 10 – *Model View Controller* (MVC)

O MVC distribui funcionalidades entre os objectos da aplicação, de modo a minimizar o grau de complexidade entre os objectos. Para isso este modelo encontra-se dividido em três camadas: o Modelo (*Model*), a Vista (*View*) e o Controlador (*Controller*).

- **Modelo** – Representa os dados de negócio, juntamente com a lógica de negócio ou operações que governam o acesso e modificação desses dados. O *modelo* notifica a *vista* quando este se altera e permite a consulta da

vista sobre o *modelo* de seu estado. O *modelo* é implementado através de um conjunto de POJOs (objectos Java simples, não estendem qualquer tipo de modelos ou convenções existentes) que contêm as variáveis e métodos que podem ser associados aos componentes. Estes métodos e variáveis podem respeitar apenas a interface ou podem comandar a interacção com a camada de negócio.

- **Vista –** retorna o conteúdo de um *modelo*. Ela recebe os dados do *modelo* e especifica como esses dados devem ser apresentados. Actualiza a apresentação dos dados quando há alterações do *modelo*. A *vista* é exposta através de ficheiros JSPX, criando um novo ficheiro para cada página da aplicação. Caso existam ficheiros com secções comuns é possível criar um ficheiro com esta secção, que é referenciado pelas páginas que utilizam esta secção. Esta técnica foi utilizada por exemplo no menu, elemento comum à generalidade dos formulários; esta secção está guardada num ficheiro reutilizado por todas as páginas que dele necessitam.
- **Controlador** - define o comportamento da aplicação. Ele interpreta entradas do utilizador e mapeia-as em acções a serem executadas pelo modelo. Numa aplicação web, as entradas do usuário são HTTP GET e POST. Um *controlador* selecciona o próximo modo de exibição a utilizador, com base nas interacções do utilizador e os resultados das operações do *modelo*. O *controlador* é implementado pela própria plataforma a partir de um ficheiro de configuração especificado em XML denominado de *faces-config.xml*.

5.2.3 Modelo/Comuns

Este módulo corresponde à junção de dois módulos distintos, o módulo *modelo* e o módulo contém as classes partilhadas entre a camada de negócio e de apresentação. Como ambos os módulos eram visíveis em ambas as camadas foi decidido manter apenas um módulo. Foi criado um pacote (*package*) *modelo* onde se encontram todas as classes do *modelo*, que representam os objectos do repositório de dados; foram ainda incluídas todas as outras classes partilhadas, que servem tipicamente, para transporte de dados entre as camadas de apresentação e de negócio.

Quando um utilizador interage a com aplicação realizando uma acção sobre a página HTML gerada, essa acção é encaminhada para o *controlador* (JSF), este sabe qual é o *bean* que está associado ao componente sobre o qual foi realizada a acção e verifica se o *bean* existe. Esta verificação é realizada na medida em que *beans* podem ser de vários tipos, podendo manter-se activos durante a vida da aplicação, da sessão do utilizador ou até mesmo apenas durante o carregamento da página. Se o *bean* não existir, ele cria-o e de seguida invoca o método adequado ou actualiza as variáveis necessárias. Depois de realizadas as actualizações ao nível do *modelo*, este comunica à *vista* que terminou a sua execução. Quando a *vista* recebe a notificação consulta o *modelo* e actualiza os valores dos seus componentes com os agora presentes no *modelo*.

5.3 Descrição funcional

Nesta secção são descritas as funcionalidades mais relevantes suportadas pela aplicação.

Na Ilustração 11 são apresentadas esquematicamente as principais funcionalidades da aplicação. A utilização da aplicação não tem obrigatoriamente de seguir o fluxo aqui apresentado, mas espelha ainda assim aquilo que será o ciclo normal de gestão de um projecto, utilizando a metodologia *scrum*.

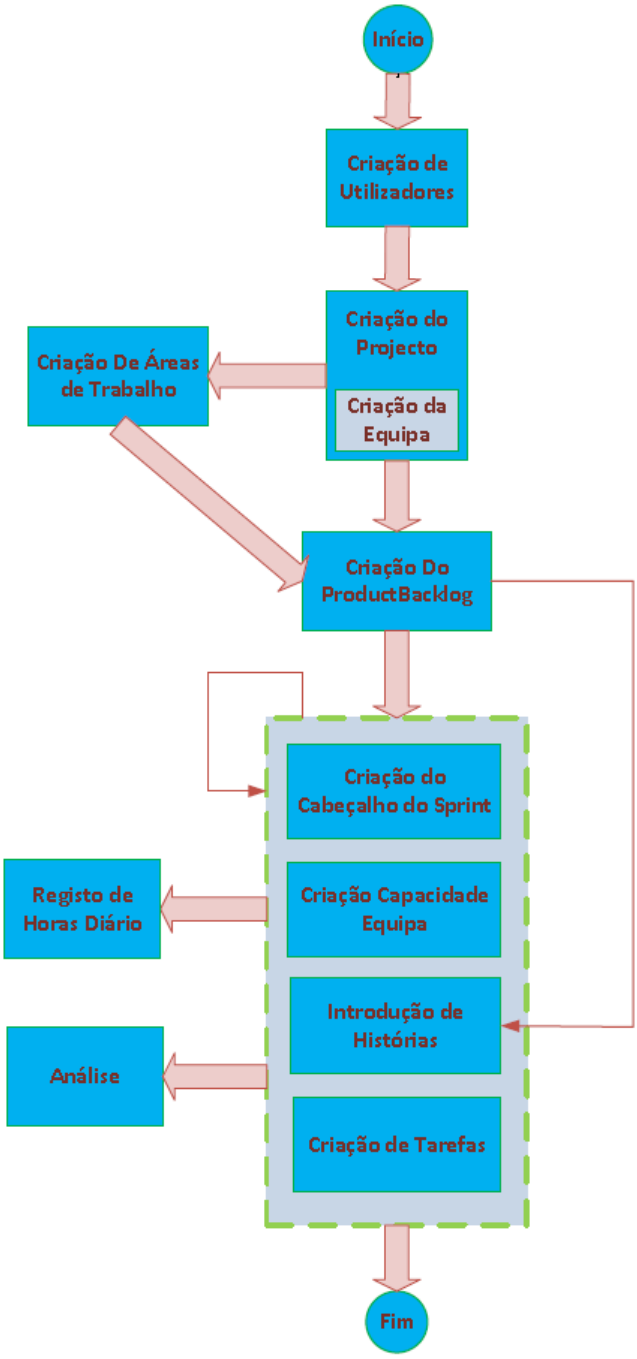


Ilustração 11 – Funcionalidades da aplicação.

5.3.1 Utilizadores

São suportadas duas formas de autenticação de utilizadores, de forma a suportar quer utilizadores internos da empresa, autenticados utilizando o protocolo LDAP disponibilizado na infra-estrutura da Truewind, numa lógica de *single sign on*, quer ainda utilizadores externos para os quais foi criado um mecanismo de autenticação autónomo, utilizando o componente *Sprint Security*.

O processo de criação de utilizadores permite a recolha dos dados pessoais considerados relevantes para a gestão dos projectos em que se encontram envolvidos. Na criação destes utilizadores é possível obter os dados pessoais previamente existentes, no sistema de registo de tempos utilizado pela Truewind, para utilizadores da Truewind, sendo automaticamente carregados os seus dados pessoais para a aplicação, que são persistidos na base de dados local da aplicação.

É suportada a pesquisa e listagem de utilizadores assistida através dos seguintes critérios de pesquisa:

- Nome completo;
- Nome próprio;
- Apelido;
- Contacto telefónico.

Ao obter a lista dos utilizadores é possível seleccionar e editar um utilizador.

5.3.2 Projecto

Nesta secção é possível definir e criar os projectos a monitorizar. Ao criar o projecto definem-se os dados base do projecto, associando-se ainda uma equipa e identificando-se os respectivos membros com base na lista de utilizadores registados. A todos os membros da equipa têm de ser associado um perfil. Os perfis disponíveis são os de *Administrador*, *Scrum Master*, *Product Owner* e *Programador*. Um elemento da equipa pode ser associado mais que um perfil no contexto de cada projecto.

Na criação de projectos é dada a possibilidade de efectuar a pesquisa de entre os projectos já registados no sistema de registo de tempos utilizado pela Truewind, e desta forma preencher automaticamente os dados do projecto, para de seguida serem guardados na base de dados local.

É suportada a pesquisa e listagem de projectos, assistida através dos seguintes critérios de pesquisa:

- Nome;
- Cliente;
- Data.

5.3.3 Áreas de trabalho

As áreas de trabalho descrevem tipicamente os módulos da aplicação a desenvolver. São distintas em projecto, sendo por isso criadas áreas de trabalho que permitem agregar as *user stories* numa forma lógica. Se o utilizador não criar nenhuma área de trabalho aquando da criação do projecto é criada por omissão, uma área de trabalho designada como *"Geral"*.

É suportada a pesquisa e listagem de áreas de trabalho, assistida através do seguinte critério de pesquisa:

- Projecto.

Ao obter a lista de áreas de trabalho é possível seleccionar e editar uma área de trabalho.

5.3.4 Product Backlog

O *product backlog* representa uma lista de histórias das funcionalidades e tarefas a executar. Esta lista é por omissão apresentada filtrada por todos os projectos registados, pelo utilizador que está autenticado e pelas *user stories* que estejam no estado A Decorrer ou Por Fazer. Para editar um *product backlog* de um determinado projecto então é necessário filtrar esta lista pelo projecto pretendido.

No contexto da gestão do *product backlog* é necessário a partir deste criar *sprints* através da exportação das *user stories* para os *sprints*.

São suportadas duas formas de exportação das *user stories*:

1. Exportar história para *sprint* activo – caso não exista nenhum *sprint* activo no projecto então é criado automaticamente um novo *sprint*, sendo copiadas as *user stories* seleccionadas e criadas duas tarefas para cada uma delas, uma tarefa associada à execução da *user story* e a tarefa para a execução dos testes dessa mesma tarefa.

   Ao nível da gestão do *sprint* pode o utilizador editar e remover as tarefas que criadas.
2. Exportar *user stories* do *sprint* planeado – na construção da *user story* pode-se definir o número do *sprint* para o qual está planeada a sua execução. Seleccionado este número de *sprint* é possível copiar todas as *user stories* a ele associadas constituindo o respectivo *sprint backlog*.

5.3.5 Sprint Backlog

Na definição de um *sprint backlog* são indicados entre outros elementos o número do *sprint*, a data de início e a sua data de fim. Com esta definição é necessário definir a capacidade dos membros da equipa neste *sprint*, que por omissão se assumem como estando alocadas a cem por cento nos dias úteis ao longo de todo o período do *sprint*. Para cada membro da equipa é possível introduzir ajustes à capacidade alocada de forma a reflectir a sua disponibilidade.

Com isto o *sprint* fica criado pronto a ser introduzidas as histórias que compõem este *sprint* e introduzindo as tarefas.

A partir do formulário de gestão do *sprint backlog*, é igualmente possível adicionar as *users stories* que estiverem planeadas no *product backlog* para o *sprint* com o respectivo número.

É suportada a pesquisa e listagem de *sprints*, assistida através do seguinte critério de pesquisa:

- Projecto.

Ao obter a lista de *sprints* é possível seleccionar e editar um *sprint*.

5.3.6 Registo

O registo deve ser utilizado diariamente pelos utilizadores para indicarem o esforço consumido na execução das tarefas em que estiveram envolvidos; para tal devem indicar o esforço já aplicado, bem como, o esforço que estimam como necessário para terminar uma determinada tarefa.

É a partir deste registo que se obtêm dados como: a data que uma tarefa foi iniciada, a data que foi finalizada ou o esforço aplicado em cada tarefa.

A partir da página de registo é possível visualizar a informação de detalhe acerca da tarefa através de uma janela de *popup*, contendo esta informação.

5.3.7 Análise e Monitorização

Através deste módulo pretende-se de forma expedita apresentar o estado de execução de cada *sprint*.

Para tal, com base em métricas como: esforço remanescente de uma tarefa; esforço médio registado por dia; esforço total estimado de tarefas por fazer; número de

tarefas por fazer. São construídos os gráficos: *Burndown Chart* e o *Cumulative Flow Chart*.

5.3.8 Listagem

Nos formulários de listagem da aplicação deve ser possível exportar os dados obtidos para os formatos PDF e Excel.

5.4 Repositório e Modelo de Dados

A base de dados é uma componente fulcral no sistema que armazena todos os dados introduzidos na aplicação. No contexto do desenvolvimento deste projecto, foi utilizado o sistema de gestão de base de dados *Oracle DatabaseOracle Database Express Edition 10g*. Dada a arquitectura seleccionada para a aplicação, poderia ser utilizado de forma transparente qualquer base de dados corrente (*Microsoft SQL Server, MySQL, PostGres, etc.*) como repositório de dados.

Na Ilustração 12 está representado o modelo de dados final resultante das diversas iterações associadas à conclusão de cada um dos *sprints* em que o projecto foi organizado.

Em todas as tabelas são utilizados quatro campos de auditoria, representando o utilizador que criou o registo, a data de criação, o último utilizador a modificar o registo, e a data de modificação. Estes campos servem essencialmente propósitos de auditoria e diagnóstico.

Na Tabela 5 são enumeradas as tabelas que integram o repositório de dados da aplicação.

Tabela	Descrição
TIPO_VALOR	Tabela com valores estáticos; Guarda todos os valores que cada TIPO_VALOR pode tomar
VALORES	Tipos de listas de valores. Os tipos possíveis são: Prioridade, Estimativa, Estado, Tipo Trabalho
PERFIL	Lista de perfis de utilizadores existentes
UTILIZADOR	Lista de utilizadores da aplicação

Tabela	Descrição
PROJECTO	Lista de projectos
EQUIPA_UTILIZADOR	Equipa e perfil de um utilizador de um dado projecto
AREA_DESENVOLVIMENTO	Lista de áreas de desenvolvimento de cada projecto
PRODUCT_BACKLOG	Lista de *product backlogs*
USER_STORY	Lista de *user stories* pertencentes a um *product backlog*
SPRINT_BACKLOG	Lista com todos os *sprint backlogs*
SPRINT	Dados de cabeçalho de cada *sprint backlog*
TAREFA	Lista de tarefas associadas a cada *sprint*.
CAPACIDADE	Alocações da equipa do projecto em cada *sprint backlog*
REGISTO	Registo de esforço dos utilizadores para uma tarefa num determinado dia

Tabela 5 – Lista de tabelas do repositório de dados

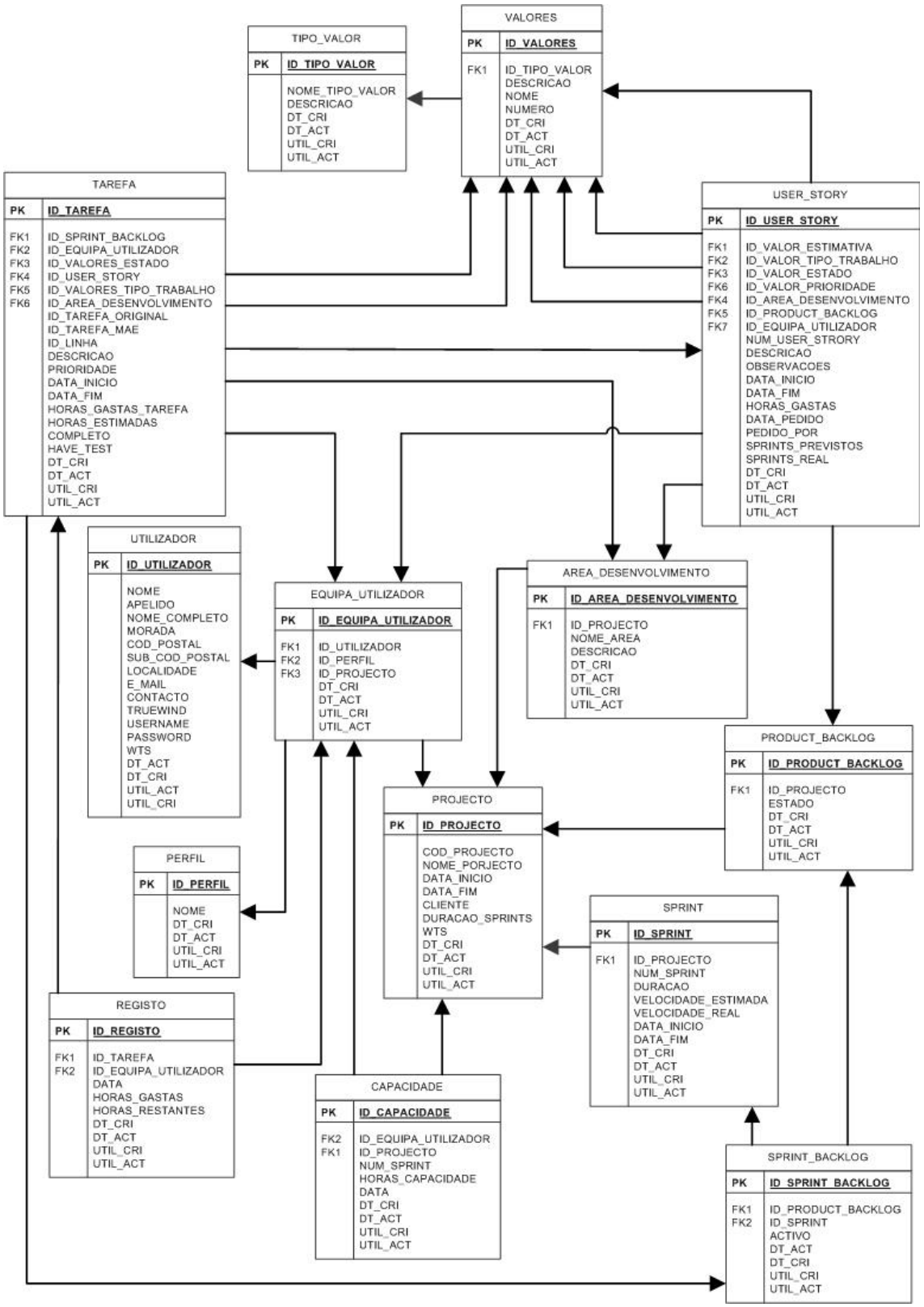


Ilustração 12 – Modelo de dados

Capítulo 6

Trabalho Realizado

Neste capítulo são descritos e apresentados ao longo das próximas secções os aspectos mais relevantes da aplicação desenvolvida. Na secção 6.1 é apresentado, com fins ilustrativos, o caso de uso de uma das funcionalidades desenvolvidas; na secção 6.2 são descritos alguns componentes base criados, reutilizados nas diversas áreas funcionais da aplicação que são apresentadas na secção 6.3 . Nas secções seguintes são abordados os testes unitários e efectuada uma análise estatística relativa ao volume de código desenvolvido.

6.1 Caso de uso exemplo

Esta secção foi incluída como auxílio à leitura das seguintes secções através da inclusão, a título de exemplo, do caso de uso, Criar Utilizador, correspondente à funcionalidade criada a partir da *user story* definida inicialmente no *product backlog.* No decorrer do projecto não foram criados casos de uso, tendo antes sido decompostas as funcionalidades descritas no *product backlog* sob a forma de *user stories*, ao serem integradas no âmbito de um *sprint backlog.*

Caso de Uso: Criar novo utilizador
Actor Principal: Utilizador
Pré-Condições: Existe um utilizador autenticado
Pós-Condições: O novo utilizador fica criado sendo possível autenticar-se
Cenário Principal:

1. O utilizador selecciona no menu “Utilizador”, submenu “Criar”
2. O utilizador preenche os campos pessoais do novo utilizador
3. O utilizador define se o novo utilizador é da Truewind

4. O utilizador preenche o *username* do novo utilizador
5. O utilizador guarda o novo utilizador

Cenários Alternativos:

2a. Preenchimento dos campos pessoais automático

1. O utilizador selecciona o botão WTS
2. Selecciona o novo utilizador a criar
3. Selecciona o botão continuar
4. O utilizador guarda o novo utilizador
5. Termina o caso de uso

3a. O utilizador não é membro da Truewind

1. O utilizador retira a opção de Utilizador Truewind
2. O utilizador preenche o *username* do novo utilizador
3. O utilizador preenche uma palavra-chave para o novo utilizador
4. O utilizador preenche a confirmação da palavra-chave
5. O utilizador guarda o novo utilizador
6. Termina o caso de uso

As imagens seguintes demonstram a utilização da aplicação no âmbito deste caso concreto.

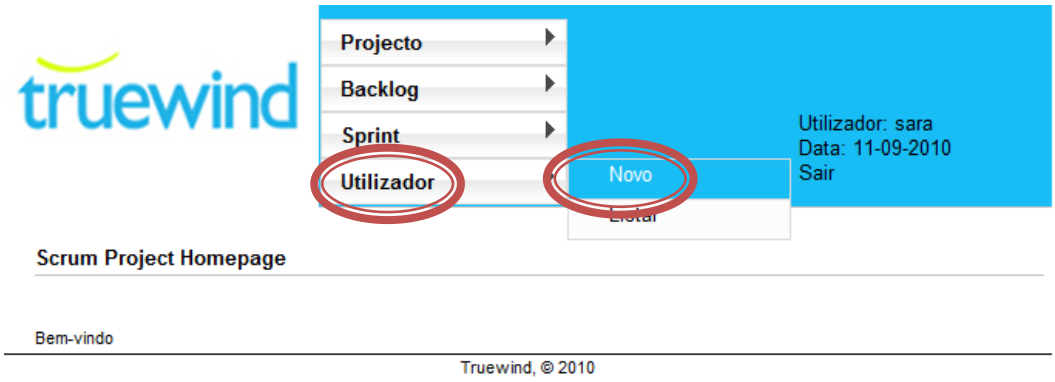


Ilustração 13 – Passo 1 do caso de uso

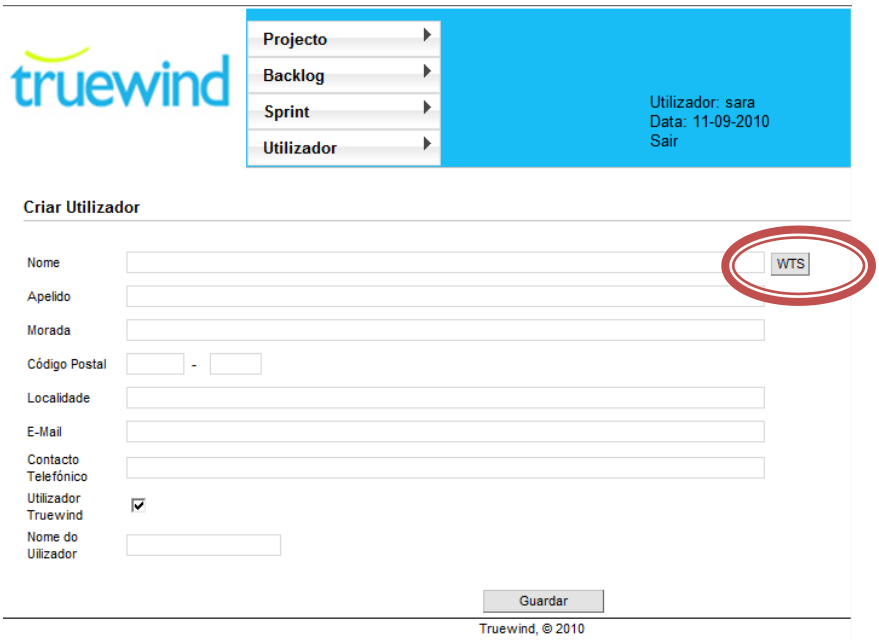


Ilustração 14 – Cenário 2 a) passo 1 do caso de uso

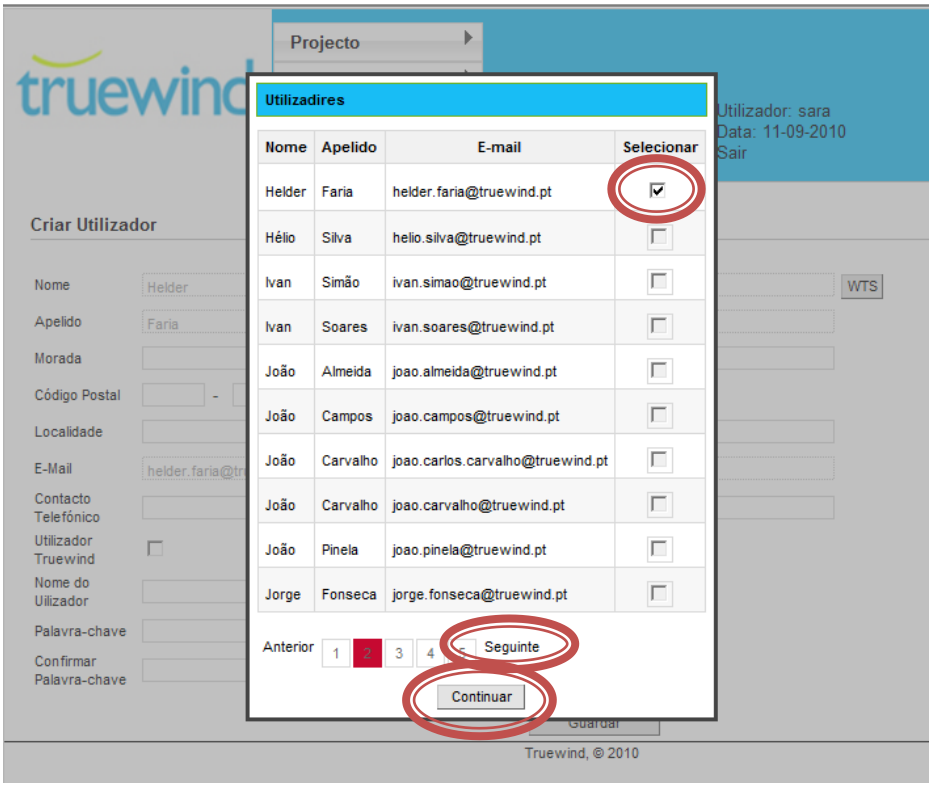


Ilustração 15 - Cenário 2 a) passo 3 do caso de uso

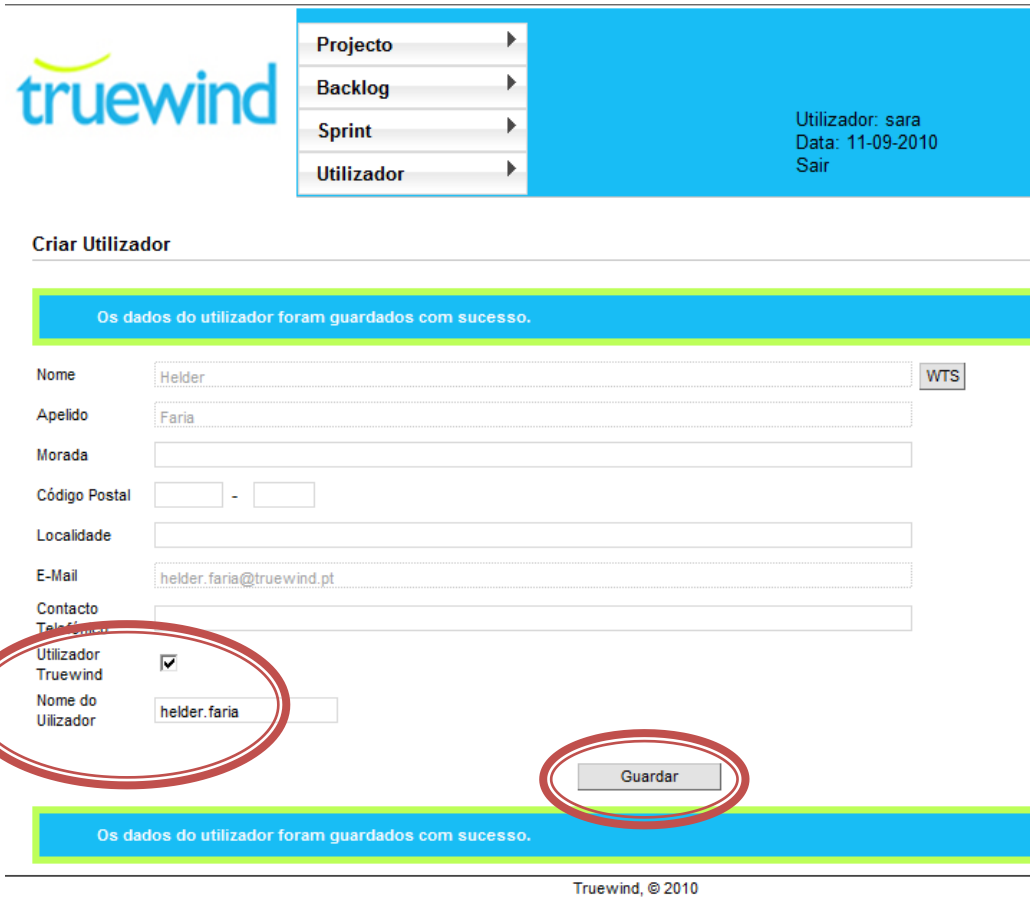


Ilustração 16 - Cenário 2 a) passo 4 do caso de uso

6.2 Componentes Base

6.2.1 Páginas Filtráveis

Havendo formulários responsáveis pela apresentação de listagens potencialmente muito extensas, passíveis de levar a um tempo carregamento muito lento e de representar uma carga excessiva ao nível do servidor aplicacional, foi implementada uma solução, que permite o carregamento parcial das listagens com base em paginadores e na aplicação de filtros; esta funcionalidade permite que os utilizadores filtrem os resultados por um conjunto de parâmetros para restringir o conteúdo da listagem. Os filtros a aplicar variam conforme a informação disponível em cada listagem. Na Ilustração 17 é apresentado um exemplo da aplicação deste conceito.

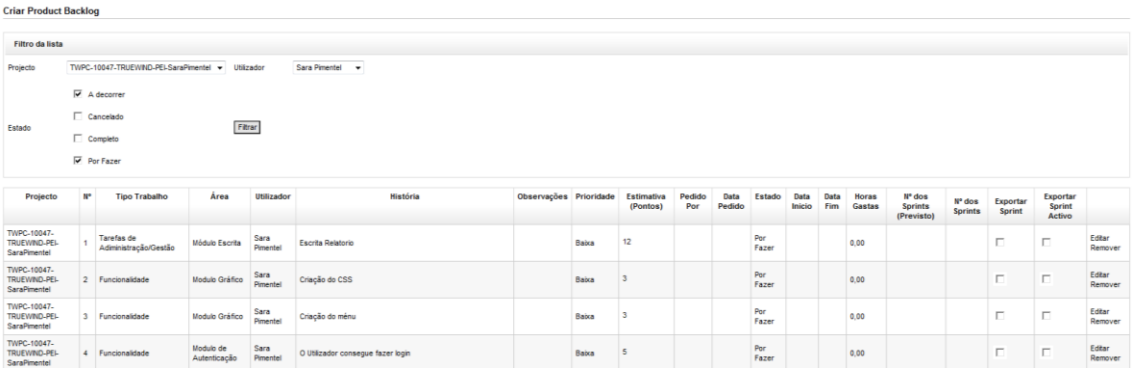


Ilustração 17 – Parte do ecrã *Product Backlog*

A componente do *IceFaces* utilizada para a apresentação das listagens é o *dataTable*, ao qual foi associado, um *dataPaginator*, fornecem as funcionalidades base de paginação. Um dos atributos do *dataTable*, de preenchimento obrigatório, e que deve ser mapeado numa variável do *managed bean* correspondente é do tipo *DataModel*, permitindo redefinir o conjunto de métodos e acções que são chamados ou despoletados com o processo de carregamento e navegação.

Foi desenvolvido um modelo base para a criação dos m*anaged beans* que suportam este tipo de páginas a partir de uma análise de quais os atributos e funcionalidades comuns a todas as páginas deste tipo, tendo-se incluído neste modelo os seguintes componentes:

- Um conjunto de variáveis específicas para cada um dos filtros;
- Um conjunto de variáveis para gestão da ordenação das listagens;
- Um método abstracto para o carregamento das listagens;
- Um método abstracto para a aplicação de filtros;
- Um método abstracto para cálculo do número total de entradas.

Para além destes componentes, esta classe requer ainda que no momento da sua extensão seja indicado qual o modelo de dados (classe utilizada para mapear os dados a carregar do repositório de dados) utilizado para a página em questão. Este modelo define qual o tipo de dados da lista que será retornada no carregamento da listagem. Esta classe foi denominada de *PaginaFiltravelMBean*.

Para cada página que utiliza filtros foi criado um *manage bean* do filtro; este tem como objectivo transmitir os filtros entre a camada de apresentação e negócio, e posteriormente traduzir os mesmos para que o JPA realize as interrogações pretendidas. Esta classe contém os valores de todas as variáveis envolvidas no processo de filtragem.

Para associar a cada uma dessas variáveis informação sobre qual o campo a que correspondem foi utilizada a anotação *@ColumnToFilter* destinada a guardar a informação necessária para determinar a forma como este deve ser utilizado pelo JPA. Estas anotações foram colocadas nas variáveis de cada um dos filtros, indicando qual o nome da entidade JPA a que está associado e qual o operador a ser utilizado.

Uma vez que o número de campos sobre o qual vão ser aplicadas restrições e os respectivos operadores podem variar, foi necessário implementar um mecanismo de tradução que permitisse reflectir em JPQL ou SQL os filtros aplicados.

Foi igualmente reutilizado um interface (*IFiltro*) de implementação obrigatória em todos os filtros e a classe *HelperFiltros*, que disponibiliza um método estático que recebendo um *IFiltro* e devolve uma *String* com a parte condicional da interrogação em formato JPQL. Para interrogação em SQL nativo existe um método idêntico que devolve a interrogação em formato SQL.

Na Ilustração 18 é apresentada parte da aplicação da utilização de filtros da página do *Product Backlog* apresentada anteriormente.

Neste exemplo temos na primeira parte (1), um excerto da classe *ProductBacklogMBean* que estende a classe *PaginaFiltravelMBean*, o *managed bean* de filtro que está associada a este *managed bean*. Na segunda parte, (2) é ilustrado o *managed bean* de filtro, que estende uma classe *BasicFilter* (3) que implementa o *IFiltro*, responsável pela definição das colunas a filtrar. No *ProductBacklogMBean* existe um método reescrito (*override*) *listSetup*(), que devolve a lista segundo a interrogação criada pelo método do *HelperFiltros* utilizando a classe *BacklogListFilter*. Esta lista está associada a um atributo do componente *dataTable* existente na página *criarProduct.jspx*.

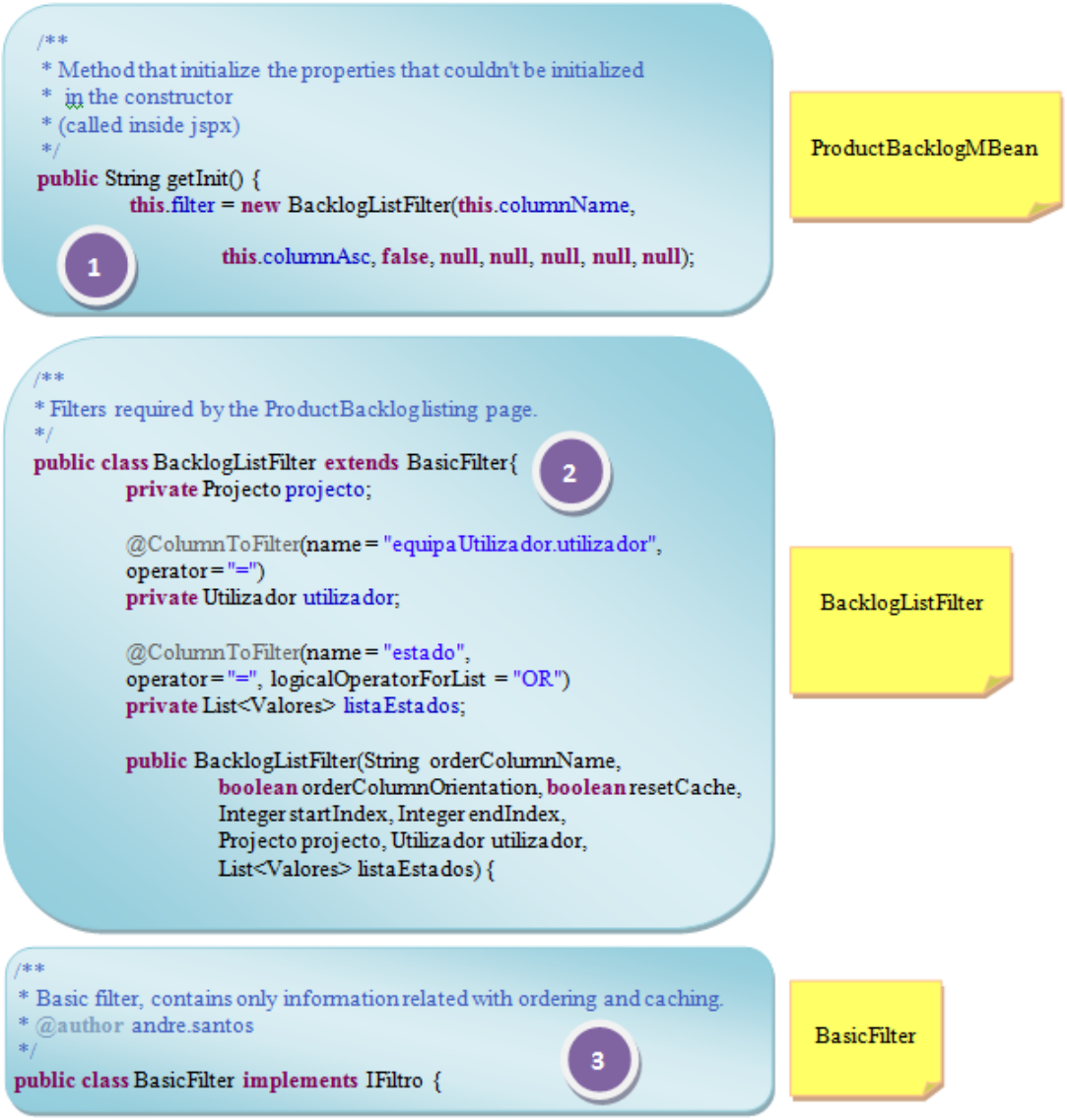


Ilustração 18 – Utilização de filtros

6.2.2 Tabelas Editáveis

Ao longo dos vários ecrãs da aplicação encontramos várias tabelas cujo conteúdo está disponível para edição na própria tabela. Exemplos deste cenário são:

- Tabela do *product backlog*
- Tabela do *sprint backlog*
- Tabela de registo.

Foi desenvolvido este componente na medida em que fazia sentido, nestes três casos, os utilizadores modificarem estas tabelas nelas próprias. Na Ilustração 19, é apresentada a título de exemplo, parte da página dos registos, em que como podemos verificar, cada registo diário é editável na própria linha da tabela.

Ilustração 19 – Parte do ecrã de Registo

Para permitir a fácil manipulação e organização dos dados na edição de tabelas, foi criada a classe *EditableEntity*. Esta classe reúne as características genéricas de um objecto passível de editar numa tabela:

- *editableEntity* – a interface a implementar;
- *editable* – atributo do tipo booleano que indica se os campos da entidade são editáveis;
- *editing* – atributo do tipo booleano que indica se os campos da entidade encontram-se em edição;
- *removable* – atributo do tipo booleano que indica se a entidade pode ser removida.

Para gerir a página de criação de registo, foi criado um *managed bean* denominado *RegistosMBean*. Neste *managed bean* é criada a lista com as entidades editáveis para alimentar a tabela.

Para possibilitar a ordenação da lista, mesmo com dados acabados de introduzir pelo utilizador, foi utilizado um comparador de entidades editáveis, *EditableEntityRegistoComparator*, que implementa um *Comparator<EditableEntity>* e executa a ordenação da lista pelas colunas. Esta classe é utilizada pelos métodos que executam a ordenação no *managed bean* (Exemplo 2).

```
/**
* Get the column name to do the order
*/
public void columnOrder(ActionEvent event) {
     sortColumn=((CommandSortHeader)event.getSource()).getColumnName();
```

```
        orderData();
    }

    /**
     * Order the data
     */
    private void orderData() {
        Collections.sort(data, new EditableEntityRegistoComparator(
            EditableEntityRegistoComparator.ComparatorEnum.fromValue(
                sortColumn), sortAscending));
    }
```

Exemplo 2 – Métodos de ordenação *RegistoMBean*

Estes métodos são chamados no início do carregamento da página, ordenando por omissão por uma coluna apropriada e são depois chamados quando o utilizador selecciona uma das possíveis colunas para ordenação. Esta acção, despoleta um *ActionEvent* sobre um componente, neste caso, um componente de *Icefaces* denominado *CommandSortHeader*, que irá ser recebido no método *columnOrder*. Como na página *jspx* este componente permite definir o nome da coluna a que se refere, utilizando esse nome, é identificada a coluna que o utilizador seleccionou e é possível a aplicação do critério de ordenação (Exemplo 3).

```
<ice:column rowspan="2">
    <ice:commandSortHeader id="sortNumSprintColumn"
        columnName="numSprint" arrow="true" immediate="true"
        actionListener="#{registosMBean.columnOrder}">
        <ice:outputText
            value="#{msgRegistos.columnDescricao}" />
    </ice:commandSortHeader>
</ice:column>
```

Exemplo 3 – Excerto da página *criarRegisto.jspx*, ordenação

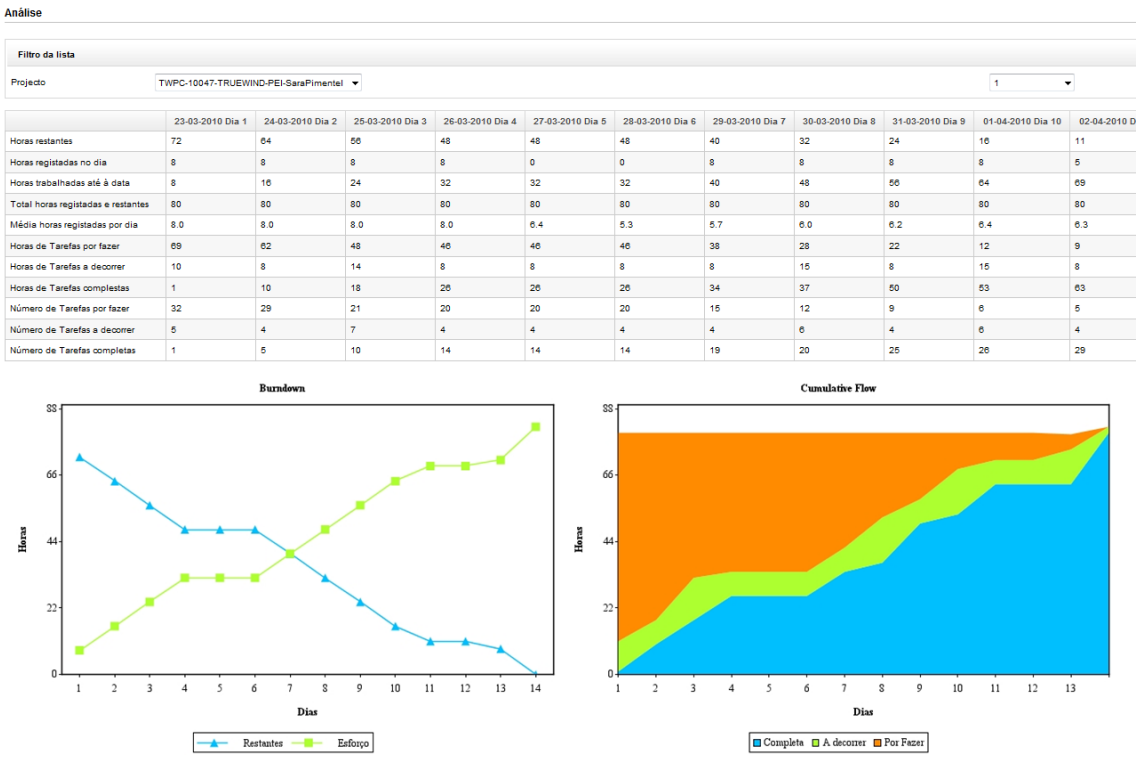
Na classe *EditableEntityRegistoComparator*, é definida uma enumeração, *ComparatorEnum*, que possui todas as colunas ordenáveis e, em função da coluna seleccionada, é executada a comparação dos dados na lista, como é exemplificado abaixo.

```
case NUM_SPRINT:{
return asc
      ? Integer.valueOf(oStory1.getTarefa().getSprintBacklog().
          getSprint().getNumSprint()).compareTo(
              Integer.valueOf(oStory2.getTarefa().getSprintBacklog().
                  getSprint().getNumSprint()))
      : Integer.valueOf(oStory2.getTarefa().getSprintBacklog().
          getSprint().getNumSprint()).compareTo(
             Integer.valueOf(oStory1.getTarefa().
                  getSprintBacklog().getSprint().getNumSprint()));
}
```

Exemplo 4 – Excerto do código da classe *EditableEntityRegistoComparator*

6.2.3 Gráficos

O *scrum*, como já referido anteriormente, preconiza a utilização de dois gráficos que auxiliam a monitorização do estado de um *sprint* ao longo da sua execução, *Burn Down Chart* e o *Cumulative Flow*, que permitem aos utilizadores terem uma visão simples do progresso do *sprint*. Na Ilustração 20 são aprestados exemplos da aplicação deste conceito.



	23-03-2010 Dia 1	24-03-2010 Dia 2	25-03-2010 Dia 3	26-03-2010 Dia 4	27-03-2010 Dia 5	28-03-2010 Dia 6	29-03-2010 Dia 7	30-03-2010 Dia 8	31-03-2010 Dia 9	01-04-2010 Dia 10	02-04-2010 D
Horas restantes	72	64	56	48	48	48	40	32	24	16	11
Horas registadas no dia	8	8	8	8	0	0	8	8	8	8	5
Horas trabalhadas até à data	8	16	24	32	32	32	40	48	56	64	69
Total horas registadas e restantes	80	80	80	80	80	80	80	80	80	80	80
Média horas registadas por dia	8.0	8.0	8.0	8.0	6.4	5.3	5.7	6.0	6.2	6.4	6.3
Horas de Tarefas por fazer	69	62	48	46	46	46	38	28	22	12	9
Horas de Tarefas a decorrer	10	8	14	8	8	8	8	15	8	15	8
Horas de Tarefas complestas	1	10	18	26	26	26	34	37	50	53	63
Número de Tarefas por fazer	32	29	21	20	20	20	15	12	9	6	5
Número de Tarefas a decorrer	5	4	7	4	4	4	4	6	4	6	4
Número de Tarefas completas	1	5	10	14	14	14	19	20	25	26	29

Ilustração 20 – Ecrã de Análise

Para a realização destes gráficos foi utilizada uma componente do *IceFaces* chamada de *outputChart*. Esta componente utiliza um pacote de código aberto denominado *JCharts* para a realização dos gráficos. Esta componente permite a criação de gráficos circulares ou de gráficos de eixo, e para cada tipo de gráfico diversas formas de apresentação, entre outros, gráficos de barras, de pontos ou de linhas. Para a realização da aplicação foi utilizado para o *Burn Down Chart* um gráfico de Linhas e para o *Cumulative Flow* utilizado um gráfico de Áreas Empilhadas. Na componente *outputChart* é possível definir um conjunto de atributos e seus valores que determina a forma de apresentação do gráfico, entre outros, o tipo de gráfico, o nome do gráfico, as legendas, as cores, a localização da legenda.

O componente oferece uma forma flexível de definir os valores dos atributos, quer de uma maneira estática quer de uma maneira dinâmica. Para realizar de forma estática basta na página *jspx*, na componente *outputChart* preencher nos seus atributos os valores pretendidos. Na forma dinâmica, a opção utilizada, passou por colocar nos atributos o método do *managed bean* responsável pela determinação dos valores a associar aos atributos.

Os valores dos atributos associados a legendas, às cores definidas, aos pontos dos gráficos ou aos valores dos eixos são listas de valores calculadas de forma análoga.

Abaixo são apresentados excertos do código, da página *jspx* (listarAnalise.jspx), com as componentes *outputChar* associadas aos dois gráficos (Exemplo 5) que utiliza o *managed bean* (Analise.MBean.java) para obter a lista dos pontos do gráfico *Burn Down* e a lista das cores do gráfico *Cumulative Flow* (Exemplo 6).

```
<ice:outputChart id="axisOutputChart"
      type="#{analiseMBean.type}"
      chartTitle="#{analiseMBean.labeltitle} "
      yaxisTitle="#{analiseMBean.labeltitleY}"
      xaxisTitle="#{analiseMBean.labeltitleX}"
      xaxisLabels="#{analiseMBean.labels}"
      labels="#{analiseMBean.legendLabels}"
      colors="#{analiseMBean.paints}"
      data="#{analiseMBean.data}"
      renderOnSubmit="#{analiseMBean.newChart}"
      horizontal="#{analiseMBean.horizontal}"
      rendered="#{analiseMBean.mostraResultados ||
```

```
      !analiseMBean.initialized}"
       width= "650" height="450"
/>
<ice:outputChart id="axisOutputChart2"
      type="#{analiseMBean.typeCumulativeFlow}"
      chartTitle="#{analiseMBean.labeltitleCumulativeFlow}"
      yaxisTitle="#{analiseMBean.labeltitleY}"
      xaxisTitle="#{analiseMBean.labeltitleX}"
      xaxisLabels="#{analiseMBean.labels}"
      labels="#{analiseMBean.legendLabelsCumulativeFlow}"
      colors="#{analiseMBean.paintsCumulativeFlow}"
      data="#{analiseMBean.dataCumulativeFlow}"
      renderOnSubmit="#{analiseMBean.newChart}"
      horizontal="#{analiseMBean.horizontal}"
      rendered="#{analiseMBean.mostraResultados ||
            !analiseMBean.initialized}"
      width= "650" height="450"
/>
```

Exemplo 5 – Excerto do código *listarAnalise.jspx*

```
/**
 * List of points calculated from the remaining hours and effort
 * @return The list of the data used by the chart
*/
public List getData() {
     List data;
     double[] ponto = null;
     double[][] valores = new double[sprint.getDuracao()][];
     for (int i = 0; i< sprint.getDuracao();i++){
      ponto = new double[2];
      ponto = new double[]{lista.get(i).getHorasRestantesEstimadas().
                  doubleValue(),lista.get(i).
                        getHorasRegistadasAteData().doubleValue()};
      valores [i] = ponto;
     }
     data = new ArrayList(Arrays.asList(valores));
     return data;
}
/**
 * The list of the colors used by the chart CumulativeFlow
```

```
 * Color(0,191,255) Blue
 * Color(173,255,47)Green
 * Color(255,140,0) Orange
  * @return The list of the colors
*/
public List getPaintsCumulativeFlow() {
      List paints = new ArrayList(Arrays.asList(new Color[]{
            new Color(0,191,255),
            new Color(173,255,47),
            new Color(255,140,0)}));
     return paints;
}
```

Exemplo 6 – Excerto do código *AnaliseMBean.java*

6.3 Aplicação

Nesta secção é apresentada a aplicação desenvolvida sendo descritas as suas principais funcionalidades.

Seguindo as boas práticas, no desenho das páginas foram aplicadas as mesmas regras, de forma a garantir uma apresentação consistente de todas elas, quer na disposição dos seus conteúdos e quer na respectiva forma de navegação.

6.3.1 Gestão de utilizadores

Nesta área da aplicação é efectuada a gestão de utilizadores. Na Ilustração 21 e Ilustração 22 é respectivamente apresentada o formulário de pesquisa e listagem e o formulário de edição e visualização de utilizador. Neste último é suportada a integração com o WTS através da pesquisa e cópia dos dados registados por essa aplicação.

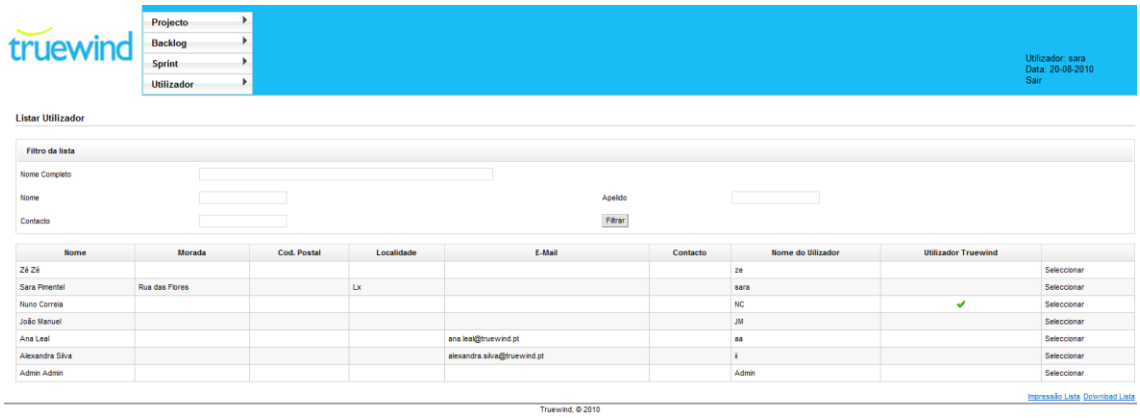


Ilustração 21 – Ecrã de Listagem de Utilizadores

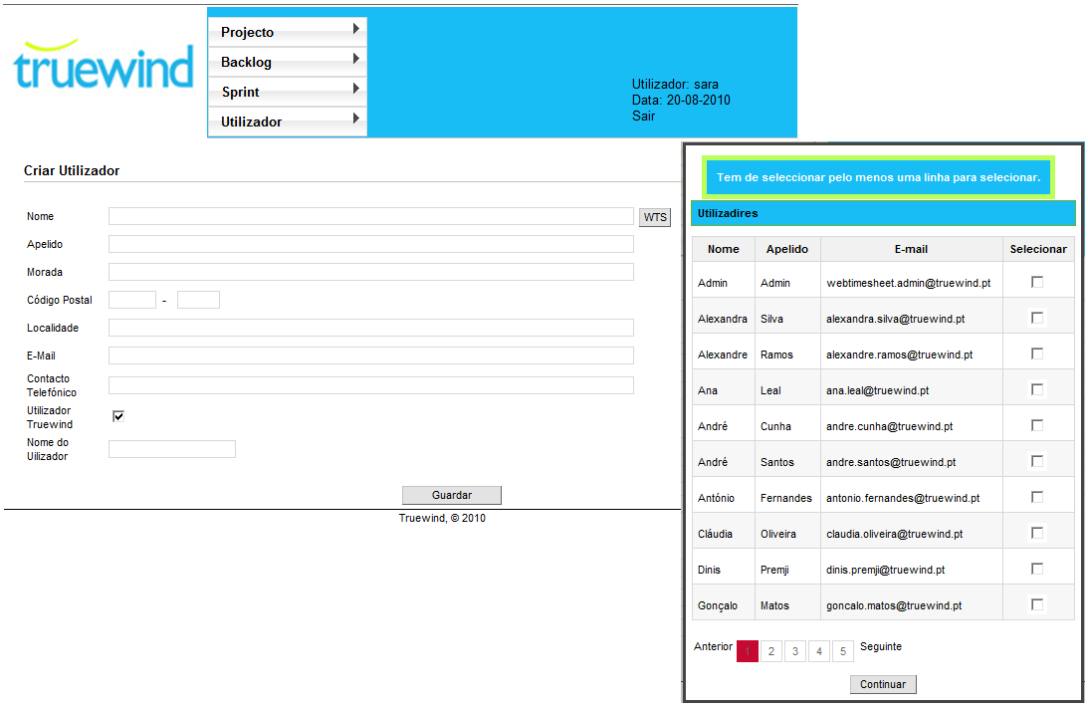


Ilustração 22 – Ecrã de Criação de Utilizadores

6.3.2 Gestão de projectos

Esta área da aplicação permite efectuar a gestão de projectos de acordo com os requisitos identificados na fase de desenho desta funcionalidade. Na Ilustração 23 e Ilustração 24 são respectivamente apresentados o formulário de pesquisa e listagem e o formulário de edição e visualização de projectos. Neste último é suportada a integração com o WTS através da pesquisa e cópia dos dados registados por essa aplicação.

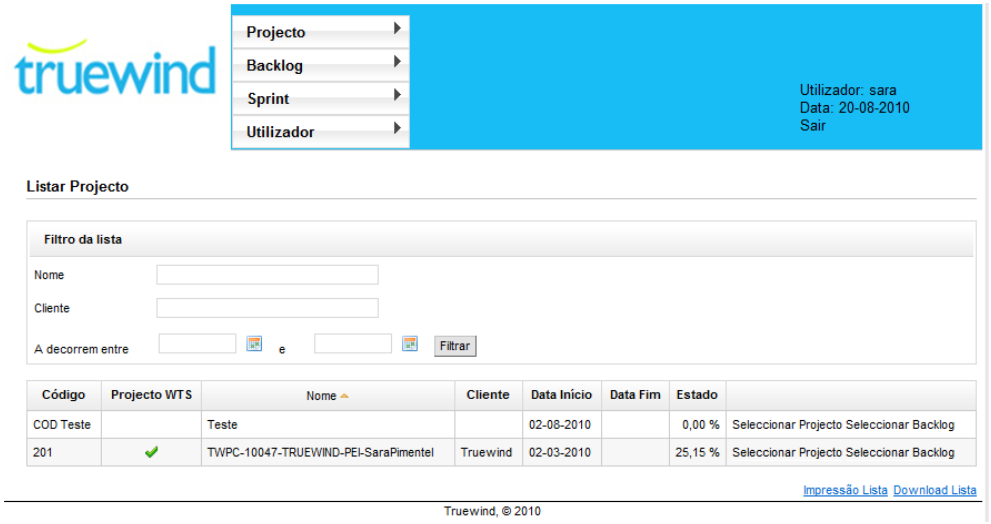


Ilustração 23 – Ecrã de Listagem de Projectos

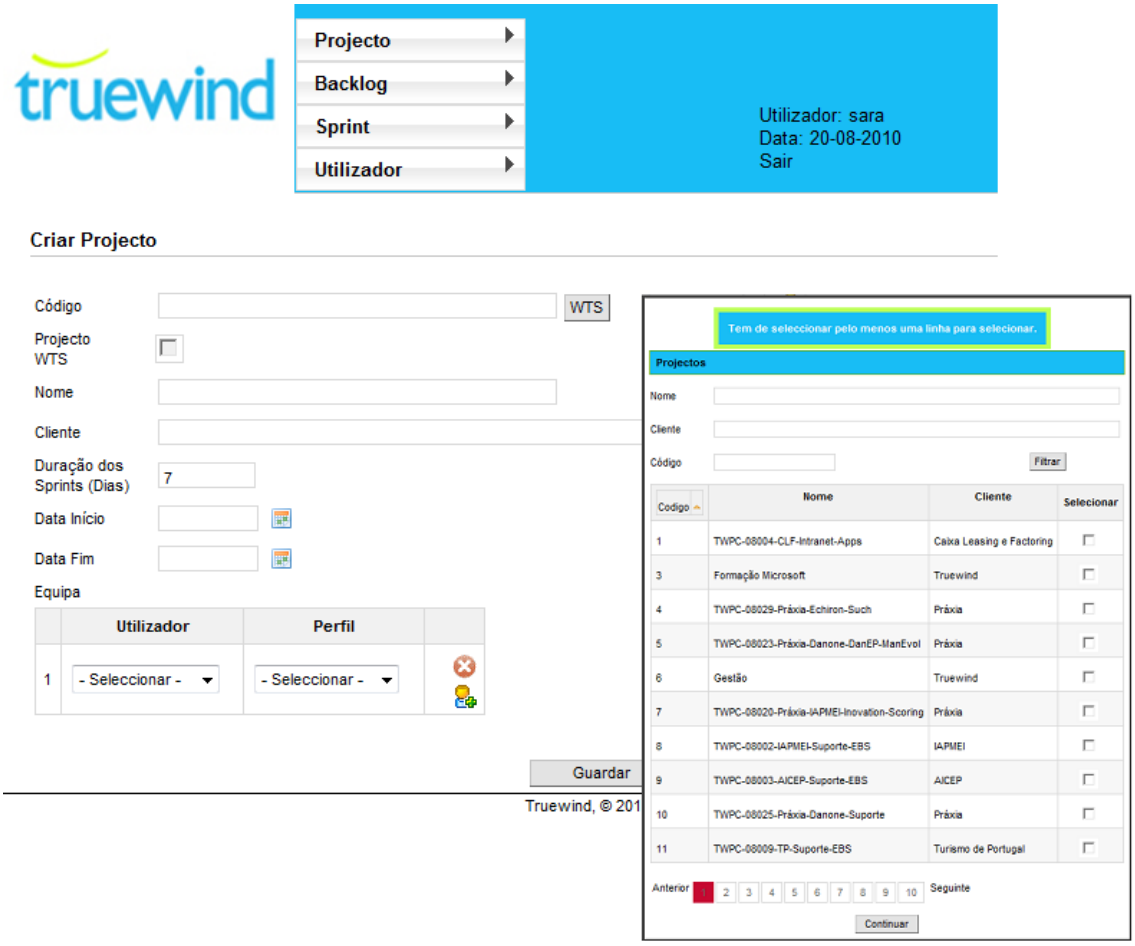
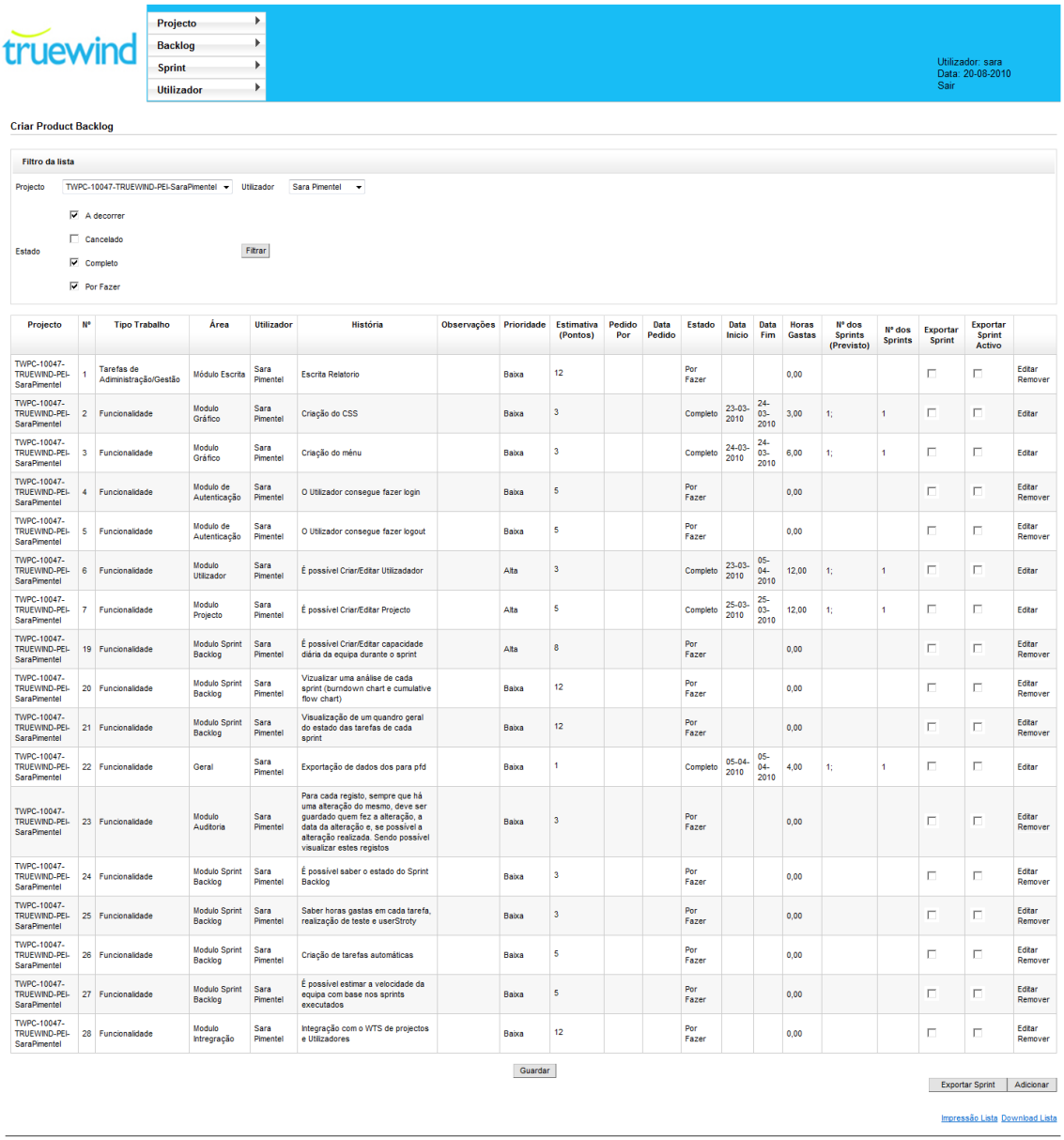


Ilustração 24 – Ecrã de Criação de Projectos

6.3.3 Gestão do *product backlog*

Nesta área da aplicação é efectuada a gestão dos *product backlogs*. Na Ilustração 25 é apresentado um formulário suportando as funcionalidades desenhadas de pesquisa, criação, edição e remoção de *user stories*. Nesta área inclui-se ainda a exportação de *user stories* para os *sprints backlogs*.



Projecto	Nº	Tipo Trabalho	Área	Utilizador	História	Observações	Prioridade	Estimativa (Pontos)	Pedido Por	Data Pedido	Estado	Data Inicio	Data Fim	Horas Gastas	Nº dos Sprints (Previsto)	Nº dos Sprints	Exportar Sprint	Exportar Sprint Activo	
TWPC-10047-TRUEWIND-PEI-SaraPimentel	1	Tarefas de Administração/Gestão	Módulo Escrita	Sara Pimentel	Escrita Relatorio		Baixa	12			Por Fazer			0,00					Editar Remover
TWPC-10047-TRUEWIND-PEI-SaraPimentel	2	Funcionalidade	Modulo Gráfico	Sara Pimentel	Criação do CSS		Baixa	3			Completo	23-03-2010	24-03-2010	3,00	1;	1			Editar
TWPC-10047-TRUEWIND-PEI-SaraPimentel	3	Funcionalidade	Modulo Gráfico	Sara Pimentel	Criação do ménu		Baixa	3			Completo	24-03-2010	24-03-2010	6,00	1;	1			Editar
TWPC-10047-TRUEWIND-PEI-SaraPimentel	4	Funcionalidade	Modulo de Autenticação	Sara Pimentel	O Utilizador consegue fazer login		Baixa	5			Por Fazer			0,00					Editar Remover
TWPC-10047-TRUEWIND-PEI-SaraPimentel	5	Funcionalidade	Modulo de Autenticação	Sara Pimentel	O Utilizador consegue fazer logout		Baixa	5			Por Fazer			0,00					Editar Remover
TWPC-10047-TRUEWIND-PEI-SaraPimentel	6	Funcionalidade	Modulo Utilizador	Sara Pimentel	É possível Criar/Editar Utilizadador		Alta	3			Completo	23-03-2010	05-04-2010	12,00	1;	1			Editar
TWPC-10047-TRUEWIND-PEI-SaraPimentel	7	Funcionalidade	Modulo Projecto	Sara Pimentel	É possível Criar/Editar Projecto		Alta	5			Completo	25-03-2010	25-03-2010	12,00	1;	1			Editar
TWPC-10047-TRUEWIND-PEI-SaraPimentel	19	Funcionalidade	Modulo Sprint Backlog	Sara Pimentel	É possível Criar/Editar capacidade diária da equipa durante o sprint		Alta	8			Por Fazer			0,00					Editar Remover
TWPC-10047-TRUEWIND-PEI-SaraPimentel	20	Funcionalidade	Modulo Sprint Backlog	Sara Pimentel	Vizualizar uma análise de cada sprint (burndown chart e cumulative flow chart)		Baixa	12			Por Fazer			0,00					Editar Remover
TWPC-10047-TRUEWIND-PEI-SaraPimentel	21	Funcionalidade	Modulo Sprint Backlog	Sara Pimentel	Visualização de um quandro geral do estado das tarefas de cada sprint		Baixa	12			Por Fazer			0,00					Editar Remover
TWPC-10047-TRUEWIND-PEI-SaraPimentel	22	Funcionalidade	Geral	Sara Pimentel	Exportação de dados dos para pfd		Baixa	1			Completo	05-04-2010	05-04-2010	4,00	1;	1			Editar
TWPC-10047-TRUEWIND-PEI-SaraPimentel	23	Funcionalidade	Modulo Auditoria	Sara Pimentel	Para cada registo, sempre que há uma alteração do mesmo, deve ser guardado quem fez a alteração, a data da alteração e, se possível a alteração realizada. Sendo possível visualizar estes registos		Baixa	3			Por Fazer			0,00					Editar Remover
TWPC-10047-TRUEWIND-PEI-SaraPimentel	24	Funcionalidade	Modulo Sprint Backlog	Sara Pimentel	É possível saber o estado do Sprint Backlog		Baixa	3			Por Fazer			0,00					Editar Remover
TWPC-10047-TRUEWIND-PEI-SaraPimentel	25	Funcionalidade	Modulo Sprint Backlog	Sara Pimentel	Saber horas gastas em cada tarefa, realização de teste e userStroty		Baixa	3			Por Fazer			0,00					Editar Remover
TWPC-10047-TRUEWIND-PEI-SaraPimentel	26	Funcionalidade	Modulo Sprint Backlog	Sara Pimentel	Criação de tarefas automáticas		Baixa	5			Por Fazer			0,00					Editar Remover
TWPC-10047-TRUEWIND-PEI-SaraPimentel	27	Funcionalidade	Modulo Sprint Backlog	Sara Pimentel	É possível estimar a velocidade da equipa com base nos sprints executados		Baixa	5			Por Fazer			0,00					Editar Remover
TWPC-10047-TRUEWIND-PEI-SaraPimentel	28	Funcionalidade	Modulo Intregração	Sara Pimentel	Integração com o WTS de projectos e Utilizadores		Baixa	12			Por Fazer			0,00					Editar Remover

Ilustração 25 – Ecrã do *Product Backlog*

6.3.4 Gestão dos *sprint backlogs*

Esta área da aplicação permite efectuar a gestão dos *sprint backlogs* de acordo com os requisitos identificados na fase de desenho desta funcionalidade. Na Ilustração 26 e Ilustração 27 são respectivamente apresentados o formulário de pesquisa e listagem e o formulário de edição e visualização do *sprint backlog*.

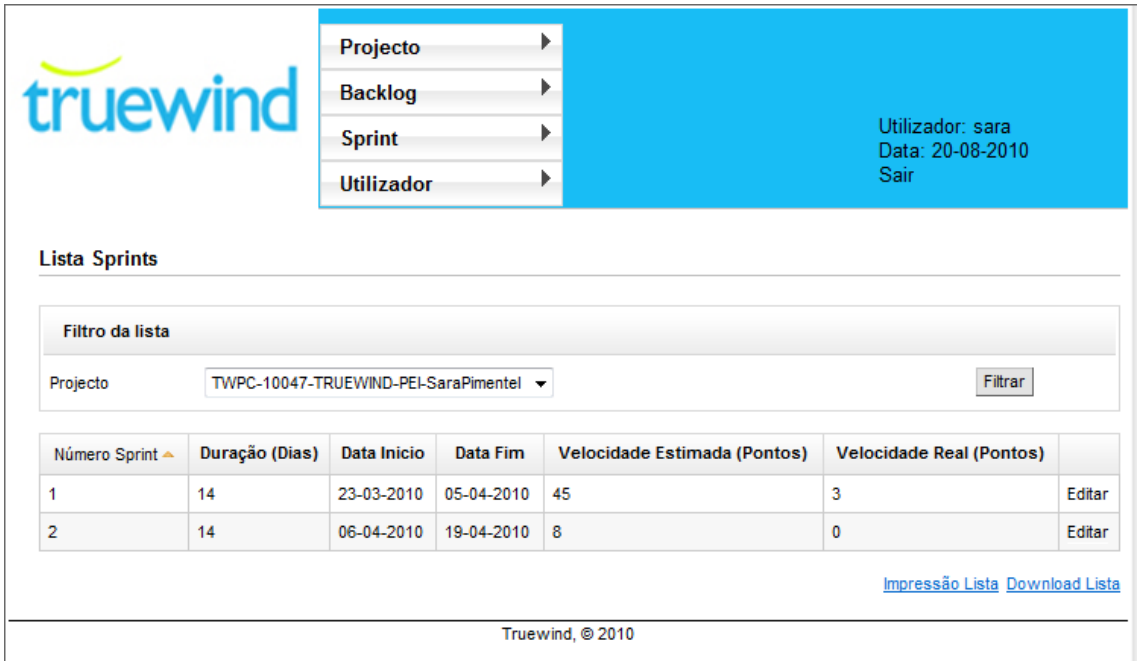


Ilustração 26 – Ecrã de Listagem de *Sprint Backlogs*

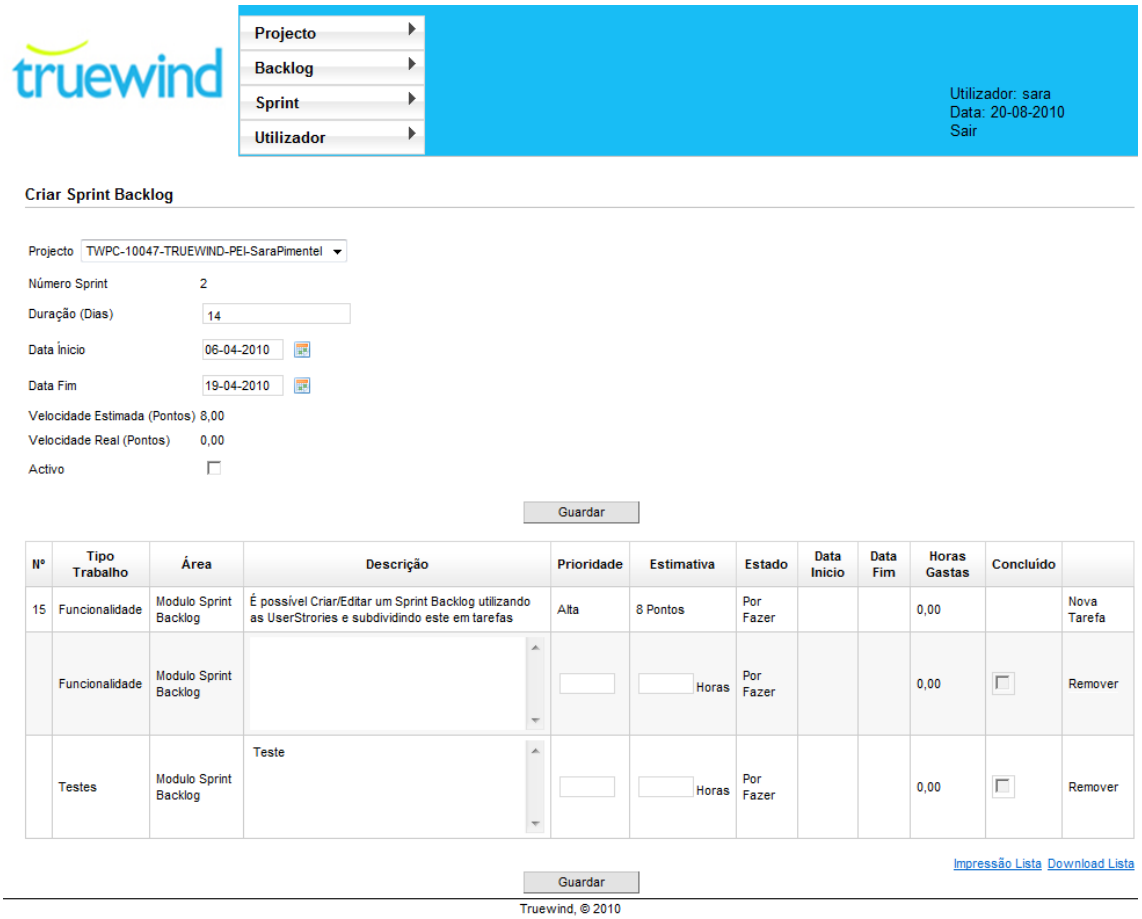


Ilustração 27 – Ecrã de Criação do *Sprint Backlog*

6.3.5 Registo de esforço aplicado

Esta área da aplicação permite registar o esforço consumido na execução das tarefas e da evolução do seu estado incluindo a actualização da estimativa de esforço necessário para a sua conclusão. Na Ilustração 28 e Ilustração 29 são respectivamente apresentados o formulário de pesquisa e listagem e o formulário de edição do esforço consumido pelo utilizador.

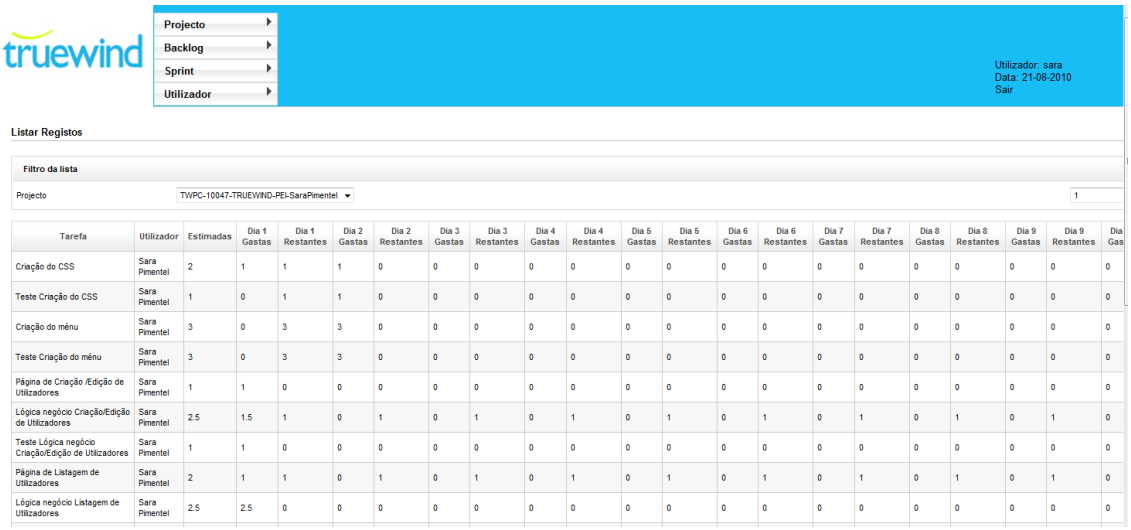


Ilustração 28 – Ecrã de Listagem dos Registo de Esforço por S*print*

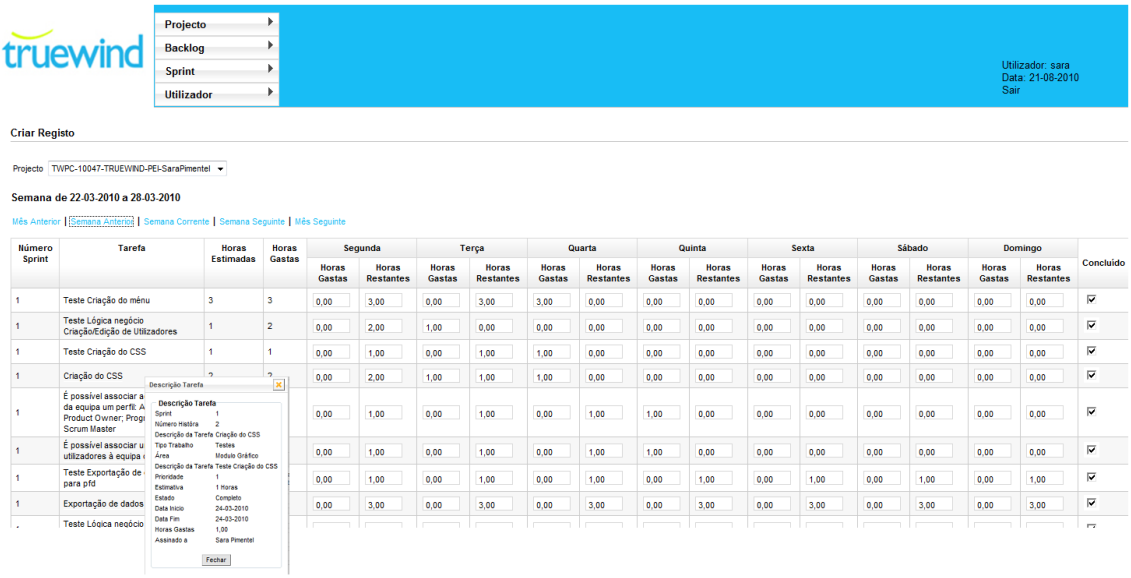


Ilustração 29 – Ecrã de Registo de Esforço do Utilizador por S*print*

6.3.6 Monitorização

Esta área da aplicação permite efectuar o acompanhamento e monitorização da execução dos *sprints*, conforme é apresentado na Ilustração 30. Inclui-se aqui é por um lado uma tabela com um conjunto de indicadores chave e os gráficos *Burndown Chart e Cumulative Flow*.

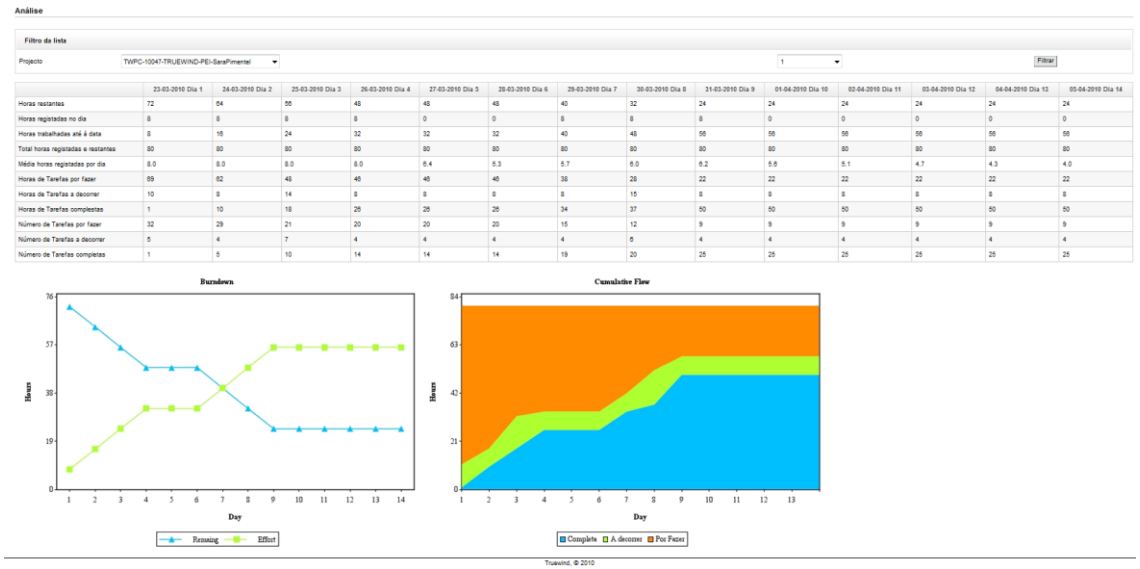


Ilustração 30 – Ecrã de Monitorização dos *Sprints*

6.3.7 Integração

É utilizada na empresa de acolhimento, um produto comercial denominada *Web Time Sheet* (WTS), que tem como principal objectivo permitir aos colaboradores da empresa efectuar o registo do esforço despendido nos projectos em que se encontram envolvidos. Nesta aplicação estão registados todos os colaboradores da empresa, bem como todos os seus projectos.

Nesta lógica, surge necessidade de permitir a integração entre a aplicação criada e a aplicação existente tendo como objectivo obter dados do WTS para a aplicação criada. Na Ilustração 31 é apresentada a topologia seleccionada para permitir o acesso de forma transparente ao repositório de dados associado ao WTS, residente numa base de dados SQL Server.

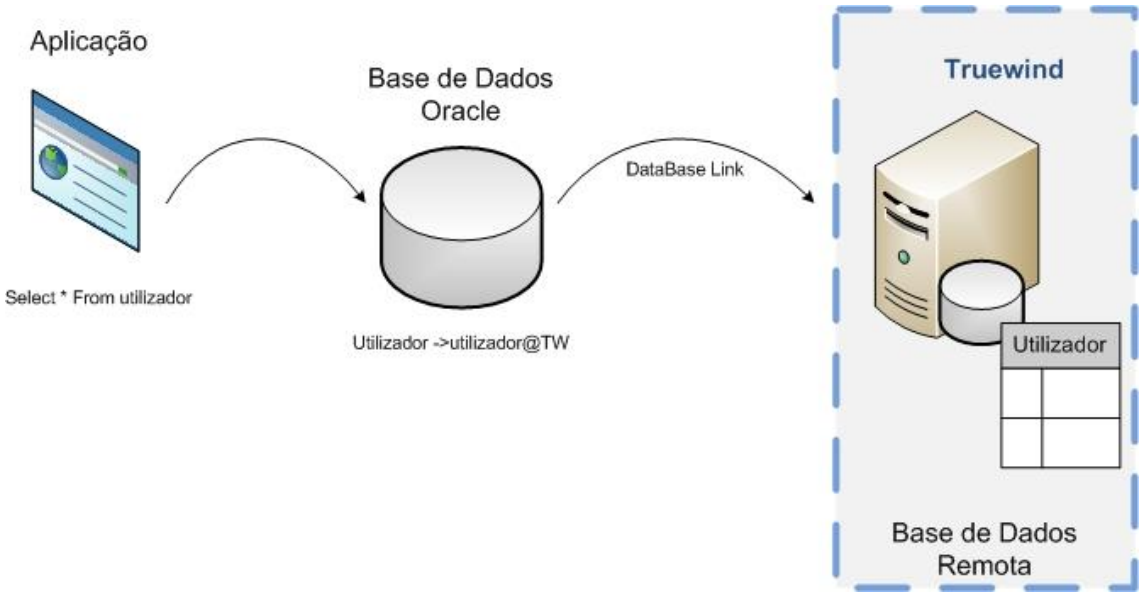


Ilustração 31 – Exemplo da utilização de um *DataBase Link*

A solução escolhida passou pela utilização de um componente disponibilizado pela servidor de base dados Oracle, denominado de *DataBase Link*, que permite o acesso, de forma transparente para os clientes da base de dados, a base de dados remotas como é disso exemplo o cenário apresentado.

Foi criado um *DataBase Link* na base de dados local, responsável pela conexão à base de dados SQL Server onde estão armazenados os dados do WTS. Esta ligação permite que aplicação aceda a base de dados local, onde de uma forma transparente para a aplicação, obtendo e manipulando ainda assim, os a base de dados remota do *WTS*.

Com base nesta infra-estrutura é possível apresentar foi, por exemplo, construído no formulário de criação de projectos, Ilustração 32, uma opção que permite a apresentação uma janela de *popup* com uma lista de Projectos, Ilustração 33, que, neste caso, pode ainda ser filtrada por nome do projecto, cliente e código do projecto.

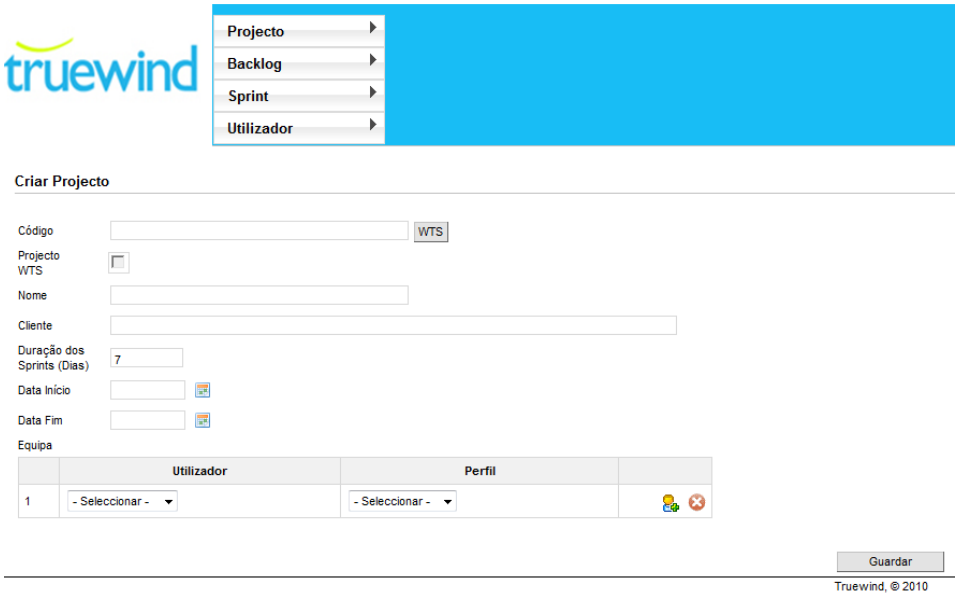


Ilustração 32 – Ecrã de criação de um Projecto

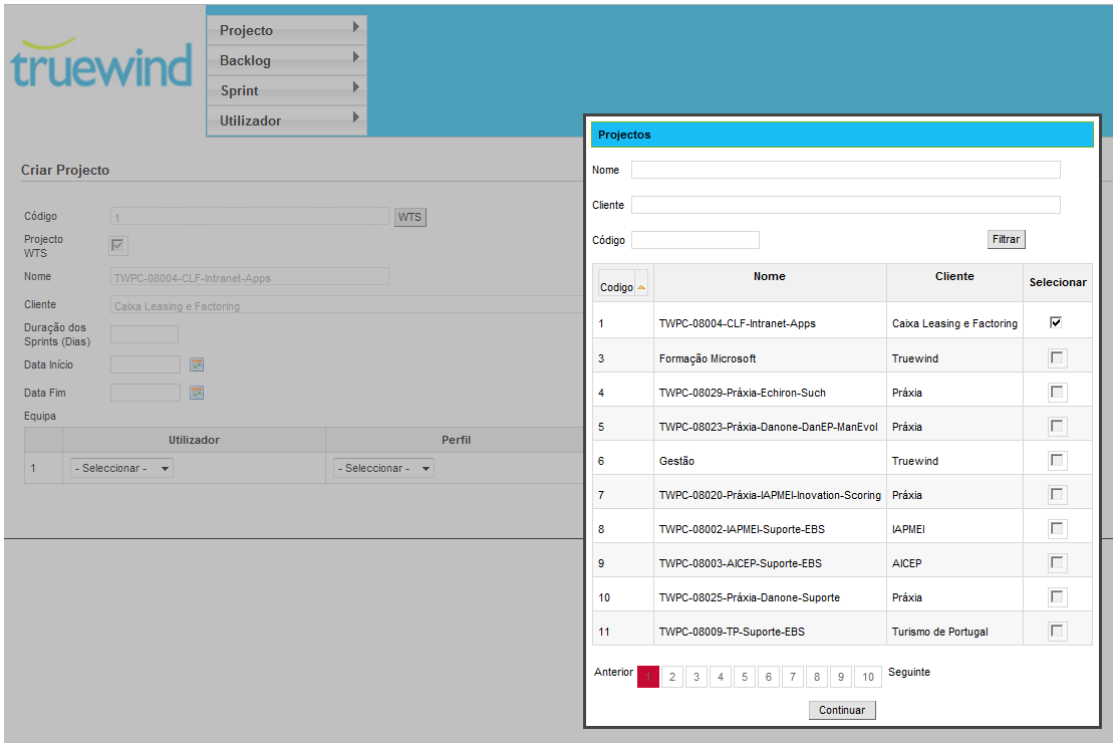


Ilustração 33 – *Pop-up* com a lista de projectos

A lista apresentada é obtida por intermédio *o Database Link*, criado localmente. Na perspectiva da aplicação esta acção é transparente, sendo apenas necessário efectuar uma interrogação à base de dados local para obter os dados expostos por *views* residentes na base de dados WTS.

6.4 Testes unitários

Ao longo do desenvolvimento foi construída uma bateria de testes com o intuito de melhorar a qualidade da solução desenvolvida, identificado possíveis falhas de programação.

Ao longo da codificação do projecto foram realizados dois tipos de testes:

- Testes de caixa-preta;
- Testes unitários.

Os testes de caixa-preta são testes onde o comportamento interno da aplicação não é analisado, mas apenas o seu comportamento externo. Estes testes foram realizados ao longo da execução do projecto inserindo dados na aplicação, e verificando se os resultados obtidos eram os esperados. Estes testes são adequados para detectar alguns tipos de erros, sendo requerido a quem estiver a realizar os testes, conhecimentos sobre o comportamento esperado da aplicação. Não garantem por si só o bom funcionamento da aplicação, por haver cenários não testados ou mesmo resultados não esperados que possam passar despercebidos na realização dos testes. Para enriquecer a bateria de testes foram igualmente realizados testes unitários.

Para a realização destes testes foi utilizada a plataforma de JUnit, conjugada com a biblioteca TruewindEjb3Unit anteriormente descrita.

Os testes unitários foram realizados sobre o módulo de Negócio, ou seja, todas as classes que utilizam o *DataAcessFacadeBean* onde esta interage com a base de dados. Cada método foi testado individualmente submetendo-o a um conjunto de estímulos, indicando quais os dados de entrada a passar a cada método e especificando qual o comportamento esperado do método, indicando quais os valores de saída esperados, quais as excepções, ou mesmo o estado do objecto após a sua execução.

Foram criadas classes e métodos que auxiliam a concretização dos testes, como no Exemplo 7 abaixo apresentado, em que, por exemplo, a classe *ProjectoTest* e o método *getProjectoTest*() testam a criação de um projecto padrão.

```
/**
 * Test for
 * {@link productBacklog#saveArea(AreaDesenvolvimentoarea)}
 * @throws InvocationTargetException
 * @throws IllegalAccessException
 * @throws IllegalArgumentException
```

```
 */
@Test
public void testSavArea() throws ProductBacklogFacadeException {
     Projecto projecto = ProjectoTest.getProjectoTest();
     projecto = dataAccessFacadeBean.makePersist(projecto);
     AreaDesenvolvimento area = AreaTest.getAreaTest();
     area.setProjecto(projecto);

     AreaDesenvolvimento areaDevolvida = instance.saveArea(area);
     assertNotNull(areaDevolvida);
     assertEquals(area.getNomeArea(), areaDevolvida.getNomeArea());
     assertEquals (area.getDescricao(), areaDevolvida.getDescricao());
     assertEquals(projecto.getIdProjecto(),areaDevolvida.getProjecto().
            getIdProjecto());
}
```

Exemplo 7 – Teste unitário

6.5 Análise

Como forma de medir o volume do trabalho efectuado para a realização da aplicação, foi utilizado um *plugin* do *eclipse*, que faz um levantamento de métricas, como por exemplo, o número de linhas de código ou o número de classes. Foram retiradas algumas dessas métricas que são apresentadas na Tabela 6.

	Módulo Commons	**Módulo EjbModule (Negócio)**	**Módulo TestHelper**	**Módulo Web (Apresentação)**	**Total**
Número de classes	51	23	19	89	**182**
Número de interfaces	3	5	0	0	**8**
Número de métodos	636	194	7	932	**1769**
Número de linhas de código	4510	4221	550	9054	**18335**
Total	**5200**	**4443**	**576**	**10075**	

Tabela 6 – Métricas do projecto

Capítulo 7

Conclusão e Trabalho Futuro

Neste capítulo são, por um lado, apresentadas as conclusões do trabalho realizado ao longo do desenrolar Projecto de Engenharia Informática, e por outro, apresentados alguns caminhos possíveis para prossecução enquanto trabalho futuro.

7.1 Conclusões

O Projecto em Engenharia Informática descrito neste documento encerra dois grandes objectivos, por um lado, o desenvolvimento das aptidões e competências do aluno como corolário do ciclo de estudos que ora termina, e por outro, o desenvolvimento de uma aplicação que representa para a instituição de acolhimento uma oportunidade de melhorar a eficiência na gestão e monitorização dos seus projectos.

Nesta lógica representou para o aluno uma experiência extremamente aliciante e recompensadora, quer ao nível pessoal, pelo interesse e pela aprendizagem que requereu, quer ao nível profissional, pela utilidade que apresenta para os colaboradores da Truewind.

Sendo o principal propósito da aplicação desenvolvida a gestão e monitorização de projectos desenvolvidos seguindo a metodologia *scrum*, exigiu do aluno a compreensão da filosofia subjacente às metodologias ágeis e em particular ao *scrum*, sendo para tal decisiva a sua integração na fase inicial deste estágio, num projecto em curso na instituição de acolhimento, no qual foi aplicada esta metodologia. Esta fase representou uma grande mais-valia para o contexto da execução da aplicação, na medida em que permitiu ao aluno entrar em contacto, num contexto real de trabalho, com aplicação prática de conceitos sobre os quais tinha até então apenas teorizado.

De igual forma, e na vertente tecnológica, a experiência de participação no referido projecto, permitiu ao aluno o desenvolvimento e enriquecimento das suas competências na construção de projectos na linguagem Java, em particular através da utilização da plataforma J2EE. Os conhecimentos adquiridos nesta fase do estágio revelaram-se decisivos para a construção da aplicação descrita neste documento, a qual permitiu consolidar e solidificar os conhecimentos adquiridos nesta tecnologia,

Por último, ao longo do estágio, o aluno deparou-se com inúmeros desafios, relacionados, quer com a vertente tecnológica, quer com a vertente funcional, os quais representaram e lhe exigiram uma grande capacidade de autonomia e de tomada de decisões que até aqui não tinha tido oportunidade de aplicar na prática.

De entre os objectivos iniciais do desenvolvimento da aplicação, não foi totalmente atingido o de integração da aplicação desenvolvida com a ferramenta de registo de tempos em utilização na instituição de acolhimento, nomeadamente o de evitar o duplo registo de dados por parte dos utilizadores da aplicação desenvolvida.

Durante a fase de análise deste componente foi verificado pelo aluno, tratando-se da interacção com um produto comercial, que ao contrário das expectativas iniciais a aplicação utilizada não dispunha de raiz de uma interface aberta para a integração de dados. Verificou-se adicionalmente, que tal interface, estando no seu essencial orientada para a extracção e não para actualização de dados, não permitiria ainda assim, o cumprimento integral deste objectivo, ao apenas contemplar informação relativa a projectos, utilizadores e tarefas, representando além do mais um custo adicional em termos de licenciamento.

Foi ainda assim demonstrada a integração com a ferramenta utilizada para registo de tempos, através da possibilidade de integrar na aplicação desenvolvida os utilizadores e projectos aí registados.

7.2 Trabalho futuro

Como perspectivas de trabalho futuro no desenvolvimento da aplicação construída pelo aluno no âmbito deste projecto antecipam-se desde logo duas vertentes:

- Enriquecimento funcional da aplicação desenvolvida através do desenvolvimento e consolidação de novos componentes;
- Melhoria intrínseca da aplicação desenvolvida através da utilização de técnicas de validação e aperfeiçoamento do código construído.

No primeiro plano, destacam-se as seguintes funcionalidades:

- Atingir uma integração plena com ferramenta de registo de tempos utilizada na instituição de acolhimento, nomeadamente de forma a atingir o objectivo inicial de evitar o duplo registo de tempos. Para tal será necessário efectuar uma análise profunda sobre o modelo de dados da referida aplicação de forma a permitir emular através da integração, as operações de actualização e registo do esforço despendido;
- Integrar no contexto das métricas quantitativas já recolhidas pela aplicação (*user stories* concluídas, esforço despendido e esforço remanescente estimado), métricas qualitativas de forma a permitir obter uma aferição mais rica em relação ao estado de execução do projecto. Exemplos das métricas que poderiam ser integradas, são a percentagem de cobertura dos testes atingida ou métricas associadas à análise estática de código.

No segundo plano, destacam-se a aplicação das seguintes técnicas e procedimentos, com as quais seria possível melhorar a qualidade a aplicação desenvolvida:

- Extensão da aplicação de testes unitários a toda aplicação, limitados no contexto deste projecto apenas ao módulo de negócio;
- Utilização de ferramentas de análise estática de código;
- Análise de cobertura dos testes unitários;
- Aplicação de ferramentas para construção e gestão de projectos em Java, facilitando o seu processo de configuração em múltiplos ambientes (por exemplo, desenvolvimento, testes e produção);
- Utilização de ferramentas de integração contínua, de forma a permitir a aplicação automatizada de baterias de testes a cada ciclo de desenvolvimento.

Anexos

Glossário

Acrónimo	Significado
Ajax	Assynchronous Javascript And XML
DAP	Directory Access Protocol
DNS	Data Source Name
EJB	Enterprise Java Beans
HTTP	Hipertext Transfer Protocol
JEE	Java Enterprise Edition
JMS	Java Message System
JPA	Java Persistence API
JPQL	Java Persistence Query Language
JSF	Java Server Faces
JSP	Java Server Pages
LDAP	Lightweight Directory Access Protocol
MDB	Message-Driven Bean
MVC	Model View Controller
OC4J	Oracle Containers for J2EE
PDF	Portable Document Format
PL/SQL	Procedural Language/Structured Query Language
POJP	Plain Old Java Object

Acrónimo	Significado
POM	Project Object Model
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol - Internet Protocol
VPN	Virtual Private Network

Product backlog – consiste na lista de tarefas a efectuar no âmbito do desenvolvimento de uma aplicação ou sistema, descritas sob a forma de funcionalidades a serem suportadas. As funcionalidades são definidas como histórias (*user stories*); cada história descreve o comportamento que o sistema deve ter perante uma determinada acção do utilizador.

Sprint – é uma fase iterativa do projecto, tipicamente com uma duração de duas a quatro semanas, na qual a equipa se compromete a entregar um conjunto de funcionalidades a executar no decorrer de cada *sprint*.

Sprint backlog – lista com as tarefas detalhadas e distribuídas pelos elementos da equipa.

Burndown chart – gráfico inicialmente calculado com base no esforço estimado para todas as tarefas incluídas no *sprint backlog*, sendo actualizado no decorrer do *sprint* com o esforço efectivamente registado pelos elementos da equipa e pelas tarefas concluídas.

Cumulative flow chart – gráfico realizado ao longo do *sprint* representando o esforço total para as tarefas que se encontram completas, a decorrer ou que ainda não foram iniciadas no decorrer do *sprint*.

Eclipse – ferramenta integrada de desenvolvimento de software vulgarmente utilizada para a linguagem Java.

Bibliografia

Acunote – [Online] http://www.acunote.com (Março de 2010)

Agilebuddy – [Online] http://www.agilebuddy.com (Março de 2010)

Agileexpress – [Online] http://agileexpress.sourceforge.net (Março de 2010)

Maven – [Online] http://maven.apache.org/ (Outubro de 2009)

Cobertura – [Online] http://cobertura.sourceforge.net/ (Outubro de 2009)

FindBugs – [Online] http://findbugs.sourceforge.net/ (Outubro de 2009)

Firescrum – [Online] http://www.firescrum.com/ (Março de 2010)

Hudson – [Online] http://hudson-ci.org/ (Outubro de 2009)

IceFaces – [Online] http://www.icefaces.org/main/home/[3]

Inflectra – [Online] http://www.inflectra.com/ (Março de 2010)

The Java EE 5 Tutorial – [Online] http://java.sun.com/javaee/5/docs/tutorial/doc/ (Outobro de 2009)

jMock – [Online] http://www.jmock.org/ (Outubro de 2009)

JUnit – [Online] http://www.junit.org/ (Novembro de 2009)

LDAP – http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol (Março de 2010)

MVC – [Online] http://java.sun.com/blueprints/patterns/MVC-detailed.html (Março de 2010)

Oracle – [Online] http://www.oracle.com/[4]

PMD – [Online] http://pmd.sourceforge.net/ (Outubro de 2009)

Scrum – http://en.wikipedia.org/wiki/Scrum_(development) (Outobro de 2009)

Scrumdesk – [Online] http://www.scrumdesk.com/ (Março de 2010)

Scrumninja – [Online] http://scrumninja.com/ (Março de 2010)

Sierra, Kathy and Bates, Bert – Head First EJB Passing the Sun Certified Business Component Developer Exam. O'Reilly Media, October 2003

Sierra, Kathy and Bates, Bert – Head First Java, Second Edition. O'Reilly Media, February 2005

[3] Bibliografia acedida constantemente

[4] Bibliografia acedida constantemente

Spring – [Online] http://static.springsource.org/spring-security/site/ (Abril de 2010)

Sprintometer – [Online] http://sprintometer.com (Março de 2010)

Outros Projectos

Na primeira fase do estágio o aluno, numa lógica de consolidação das suas competências quer na linguagem Java, e na plataforma J2EE, quer ainda, na aplicação prática da metodologia *scrum*, participou num projecto em curso na instituição de acolhimento destinado a um instituto na esfera da administração pública.

A equipa onde o aluno foi integrado era constituída por quatro elementos, dispondo na sua maioria de uma larga experiência de desenvolvimento de projectos utilizando as tecnologias e metodologias referidas.

Este projecto tinha como objecto a construção de uma aplicação distribuída de elevado nível de complexidade, denominada de Sistema Central de Cobrança de Taxas (SCCT), servindo diversos balcões de atendimento ao público distribuídos por todo o país, abarcando no essencial as seguintes funcionalidades:

- Módulo de Autenticação – Autenticação dos utilizadores do sistema de acordo com o seu perfil de acesso;
- Módulo de Gestão de Taxas;
- Módulo de Gestão de Clientes – permite consultar os dados do Cliente e processos de pagamento pendentes do Cliente; Criação de entidades;
- Módulos de Cobranças Directas e de Negocio;
- Módulo de Pagamento por Multibanco – concentra todas as actividades relacionadas com os pagamentos Multibanco;
- Módulo de Gestão de Caixa – fornece as ferramentas que permitam controlar as cobranças de taxas realizadas em cada combinação delegação/balcão/posto atendimento/cobrador;
- Módulo de Integração Contabilística – integração entre SCCT e outro sistema com o objectivo de enviar os dados contabilísticos referentes a cobranças, dados necessários para a conciliação bancária e registos referentes a reembolsos;

- Módulo de Conciliação Bancária – permite integrar informação dos extractos bancários;
- Módulo de *Reporting* – reprodução vários relatórios;
- Módulo de Gestão de Perfis – permite gerir associações como atribuir tarefas a perfis e perfis a pessoas;
- Módulo de *Workflow* – gerir os fluxos processuais de aprovações e notificações do SCCT;
- Módulo de Auditoria – permite responsabilizar todos os actores pelas suas acções, assegurando a completa auditabilidade do sistema (quem fez o quê, quando, como);
- Módulo Gestão de Parâmetros de Sistema – reúne todos os parâmetros do SCCT, permitindo a sua manutenção regular;

Para além de um período de estudo e autoformação que o aluno teve oportunidade de efectuar no contexto da sua integração neste projecto o aluno participou no desenvolvimento dos seguintes componentes, com um enfoque para a parte de apresentação, ao longo de diversos *sprints* nos quais esteve envolvido:

- Realização do Modulo de Gestão de Parâmetros de Sistema;
- Realização do tratamento dos processos originários do Posto de Atendimento ao Cidadão;
- Realização do módulo de Auditoria;
- Realização do módulo de Integração Contabilística;
- Realização do tratamento de modificações de meios de pagamento e dados pessoais das facturas e impressão de 2ªVia;
- Correcção de diversos *bugs* detectados.