UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

SISTEMA PARA GESTÃO ÁGIL DE PROJETOS DE SOFTWARE

ROBSON RICARDO GIACOMOZZI

BLUMENAU
2011

2011/1-20

ROBSON RICARDO GIACOMOZZI

SISTEMA PARA GESTÃO ÁGIL DE PROJETOS DE SOFTWARE

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Sistemas de Informação— Bacharelado.

Prof. Everaldo Artur Grahl, Mestre - Orientador

BLUMENAU
2011

2011/1-20

SISTEMA PARA GESTÃO ÁGIL DE PROJETOS DE SOFTWARE

Por

ROBSON RICARDO GIACOMOZZI

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Everaldo Artur Grahl, Mestre – Orientador, FURB
Membro:	Prof. Jacques Roberto Heckmann, Mestre – FURB
Membro:	Prof. Wilson Pedro Carli, Mestre – FURB

Blumenau, 28 de junho de 2011.

Dedico este trabalho aos meus pais que no esforço e determinação puderam me apoiar, incentivar e proporcionar a realização desta graduação.

AGRADECIMENTOS

Aos meus pais, Osnir e Juleide, que sempre me apoiaram e estiveram presentes nos meus estudos e que estão muito felizes e orgulhosos com essa conquista.

Ao meu orientador, Everaldo, por ter acreditado na proposta e na minha capacidade de concluir este trabalho.

Aos meus amigos, pela ajuda e o apoio.

A Edusoft Tecnologia Ltda. e seus funcionários que me apoiaram e incentivaram durante a graduação.

Ao colegiado do curso de Sistemas de Informação da FURB, pelo aprendizado adquirido durante a minha graduação.

Não importa o quanto você bate, mas sim o quanto aguenta apanhar e continuar. O quanto pode suportar e seguir em frente. É assim que se ganha!

Rocky Balboa

RESUMO

Este trabalho apresenta um sistema *web* para o gerenciamento ágil de projetos de *software*. O sistema suporta os papéis *Product Owner*, *Scrum Master* e *Scrum Team*, os artefatos *Product Backlog*, *Sprint Backlog* e *Burndown Chart*, além das cerimônias *Sprint Planning*, *Sprint Review*, *Sprint Retrospective* e *Daily Scrum Meeting*, todos elementos do método ágil *Scrum*. Para o desenvolvimento do sistema foram utilizadas a linguagem PHP, com o *framework* CakePHP, e banco de dados MySQL. Suportando a estrutura do *Scrum*, o sistema se torna uma ferramenta de alta relevância para controlar, de forma ágil, projetos de *softwares*.

Palavras-chave: Sistemas de Informação. Gerência de projetos. Métodos ágeis. Scrum. CakePHP.

ABSTRACT

This work presents a web based system to manage agile software projects. The system supports the Product Owner, Scrum Master and Scrum Team roles, the artifacts Product Backlog, Sprint Backlog and Burndown Chart, and the ceremonies Sprint Planning, Sprint Review, Sprint Retrospective and Daily Scrum Meeting, all elements of the Scrum agile method. For the development of the system was used PHP language with the CakePHP framework and MySQL database. Supporting the structure of Scrum, the system turns to be a high relevant tool to control, in an agile way, software projects.

Key-words: Information Systems. Project Management. Agile. Scrum. CakePHP.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE SIGLAS

DER - Diagrama Entidade-Relacionamento

EA - *Enterprise Architect*

ER - Entidade Relacionamento

JSP – *Java Server Pages*

MVC – *Model View Controller*

PHP - *Hypertext Preprocessor*

PO – *Product Owner*

SM – *Scrum Master*

ST – *Scrum Team*

TI - Tecnologia da Informação

URL - *Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

O gerenciamento de projeto fornece um poderoso conjunto de ferramentas para que os profissionais possam aprimorar suas habilidades em planejar, implementar e administrar atividades para atingir objetivos organizacionais específicos (GRAY; LARSON, 2009). Martins (2007) afirma que, na criação de *softwares*, os frequentes fracassos em projetos motivaram a criação de diversas metodologias, técnicas e processos para gerenciamento de projetos, tanto para desenvolvimento como para implantação de sistemas. Atualmente, existem diversas abordagens para gerenciamento de projetos, algumas são classificadas como metodologias clássicas e outras como ágeis.

O processo, na abordagem clássica, que também se pode chamar de metodologias tradicionais ou pesadas, é totalmente planejado antes de ser executado. Durante o desenvolvimento, a equipe desenvolve e executa o projeto conforme o planejamento.

A maioria dos projetos de *softwares*, que utilizam a abordagem tradicional, baseiam-se no modelo *Waterfall*[1], onde existe uma grande fase de planejamento que dá todo o suporte ao desenvolvimento do *software*. Com a utilização deste modelo, é muito difícil para as empresas aceitarem mudanças de requisitos nas fases seguintes, pois exige uma regressão à fase de planejamento, o que pode resultar em uma mudança drástica no desenvolvimento do *software*.

Com foco na eficiência, Beck (2001) propõe que a análise de requisitos seja extremamente mutável, "abraçando" as mudanças como principal área de atuação dos processos ágeis. Sendo assim, os planejamentos são constantes, não existindo uma etapa exclusiva para isso.

O *Scrum* é uma metodologia ágil que vem ganhando grande visibilidade nos últimos cinco anos, na qual as necessidades de negócio são que determinam as prioridades (BROD, 2009). O *Scrum* mantém o foco na entrega do maior valor de negócio, onde são necessárias revisões e adaptações constantes no planejamento do projeto.

A partir disto, este trabalho tem por finalidade a criação de um sistema *web* que suporte o *Scrum* e toda sua estrutura de trabalho, alinhado aos objetivos da abordagem ágil.

---

[1] O modelo *waterfall*, também chamado de cascata, tornou-se conhecido na década de 70. Nele as atividades do processo de desenvolvimento são estruturadas numa cascata onde a saída de uma é a entrada para a próxima (LEITE, 2007).

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é o desenvolvimento de um sistema *web* para gestão de projetos de *softwares*, fornecendo o apoio necessário às empresas que utilizam metodologias ágeis no desenvolvimento de seus produtos.

Os objetivos específicos do trabalho são:

a) suporte à estrutura básica do *framework Scrum*, na gestão ágil de projetos;
b) controle das solicitações de implementações e *bugs* para desenvolvimento;
c) monitoramento do andamento de uma iteração através do gráfico de *burndown*;
d) gestão dos impedimentos gerados durante a execução das *sprints*.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está disposto em quatro capítulos. No primeiro capítulo apresenta-se a introdução, os objetivos e a estrutura do trabalho.

No segundo capítulo tem-se a fundamentação teórica, destacando-se os conceitos de processos de *software*, com ênfase em metodologias ágeis e no *framework Scrum* bem como os trabalhos correlatos.

No terceiro capítulo é apresentado o desenvolvimento da ferramenta, incluindo detalhes sobre a especificação, implementação e tecnologia utilizada.

No quarto capítulo apresenta-se a conclusão e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os aspectos teóricos sobre este trabalho, tais como o processo de produção de um *software*, metodologias ágeis e abordagem ao *framework Scrum*. A última seção deste capítulo descreve alguns trabalhos correlatos.

2.1 PROCESSO DE SOFTWARE

De acordo com Pressman (2006), processo de *software* é um conjunto de tarefas requeridas para a produção de um *software*. O foco principal de um *software* é a qualidade gerada do produto final.

Embora existam vários processos de desenvolvimento de *software*, quatro atividades fundamentais são comuns a todos os processos (SOMMERVILLE, 2007):

a) especificação: definição do que o sistema deve fazer; definição dos requisitos e das restrições do *software*;

b) desenvolvimento: projeto e implementação do *software*; o sistema é desenvolvido conforme sua especificação;

c) validação: é validado para garantir que as funcionalidades implementadas estejam de acordo com o que foi especificado;

d) evolução: evolução do *software* conforme a necessidade do cliente.

Segundo Ludvig e Reinert (2007, p.20), "o desafio mais comum dentro do desenvolvimento de *software* é entregar um produto que atenda às reais necessidades do cliente, dentro do prazo e orçamento previstos".

Ferreira e Lima (2006) comentam que a engenharia de *software*[2] há muito tempo vem enfrentando problemas relativos a atrasos de entrega de projeto, orçamentos extrapolados e insatisfação de clientes e usuários.

[2] A engenharia de *software* tem por objetivos a aplicação de teoria, modelos, formalismos e técnicas e ferramentas da ciência da computação e áreas afins para a produção (ou desenvolvimento) sistemática de *software* (LEITE, 2007).

O CHAOS *Report*[3] (STANDISHGROUP, 2009) ilustra muito bem a necessidade de adaptação dos projetos. Na atualização de 2009, a pesquisa analisou 365 empresas e 3682 projetos foram avaliados. Seguem os principais resultados:

a) 68% dos projetos foram cancelados ou excederam orçamento e prazo estimados;
b) o custo médio excedeu 45% sobre o custo planejado;
c) o atraso médio foi de 63% sobre o prazo planejado;
d) 24% dos projetos fracassam;
e) 44% dos projetos são entregues com sucesso parcial;
f) e apenas 32% dos projetos obtem sucesso.

Segundo a pesquisa, os fatores mais apontados como motivadores do sucesso parcial ou fracasso dos projetos foram a falta de envolvimento do usuário, as expectativas não realistas, os ciclos (tempos) irrealistas e a falta de gestão da Tecnologia da Informação (TI), entre outros.

Visando a qualidade final do produto gerado, são propostos métodos que buscam padronizar as atividades e o ciclo de vida de um processo de *software*. Esta padronização implica em criar documentos, artefatos e marcos capazes de representar o contexto do software (NETO, 2004).

Dentre os processos de *software* existentes e na tentativa de aumentar o número de projetos executados com sucesso, as metodologias ágeis vêm chamando a atenção dos profissionais por serem adaptáveis e realistas com os desafios dos projetos atuais.

2.2 MÉTODOS ÁGEIS

Nós últimos anos a atual sociedade vem tornando-se cada vez mais dependente de *software* e tecnologia. Neste cenário, a demanda por *software* tem crescido exponencialmente, tornando o mercado mais competitivo e exigindo que a engenharia de *software* desenvolva a capacidade de adaptar-se rapidamente a mudanças (BASSI, 2008).

As metodologias ágeis são o resultado do esforço de um grupo de profissionais veteranos na área de *software*, que em 2001, criaram um manifesto que descreve um conjunto

[3] Relatório publicado pelo *The Standish Group* a cada 2 anos que analisa o resultado de projetos de TI.

de valores e práticas comuns para o desenvolvimento de *software*. Esse manifesto identifica valores e princípios que devem prevalecer no desenvolvimento de *software* (BECK, 2001).

O manifesto ágil consiste em uma declaração com quatro máximas que regem o desenvolvimento ágil.

> Estamos descobrindo maneiras melhores de desenvolver *software* fazendo-o nós mesmos e ajudando outros a fazê-lo. Através desse trabalho, passamos a valorizar:
>
> - **Indivíduos e suas interações entre eles** mais que processos e ferramentas;
> - **Software funcionando** mais que documentação abrangente;
> - **Colaboração do cliente** mais que negociação de contratos;
> - **Responder à mudança** mais que seguir um plano.
>
> Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda (AGILE MANIFESTO, 2001).

Além das quatro máximas apresentadas, o manifesto cita estes doze princípios:

a) a maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de *software* de valor;
b) aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas;
c) entregar *software* funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos;
d) pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto;
e) construir projetos ao redor de indivíduos motivados, dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho;
f) o método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara;
g) *software* funcional é a medida primária de progresso;
h) processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes;
i) contínua atenção à excelência técnica e bom *design*, aumenta a agilidade;
j) simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito;
k) as melhores arquiteturas, requisitos e *designs* emergem de times auto-organizáveis;

l) em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

Segundo Orth e Priklandnicki (2009), pode-se identificar quatro aspectos que o manifesto ágil impulsiona: a) a cultura; b) a comunicação; c) as pessoas; e d) o processo. O manifesto prega que os projetos deem o suporte adequado às necessidades da equipe, formando profissionais motivados e confiantes, que reflitam sobre como tornarem-se mais efetivos, empregando seus esforços em entregar *software* funcionando.

Diferentemente da abordagem tradicional, que tem seus requisitos inteiramente descritos e congelados, a abordagem ágil responde melhor às mudanças no decorrer do processo (ABRAHANSSON et al., 2003).

Atualmente, existem várias abordagens e modelos ágeis de processo. Entre as mais relevantes e utilizadas, pode-se citar o *Extreme Programming* (XP) e o *Scrum*.

O *Scrum* foi a base deste trabalho, pela importância que tem adquirido perante a comunidade de desenvolvedores e pelas características que garantem um compromisso com a entrega de valor ao cliente, sem perder o foco nos processos, na cultura e nas pessoas que desenvolvem o *software*.

2.3 SCRUM

O *Scrum* é um processo ágil (alguns autores definem *Scrum* como "*framework"*), iterativo e incremental, que descreve um conjunto de "padrões de processo de software", com práticas objetivas e adaptáveis. Esse processo é adequado para projetos, geralmente de *softwares*, com prazos apertados e requisitos que mudam frequentemente, focando no valor de negócio gerado (TOMÁS, 2009).

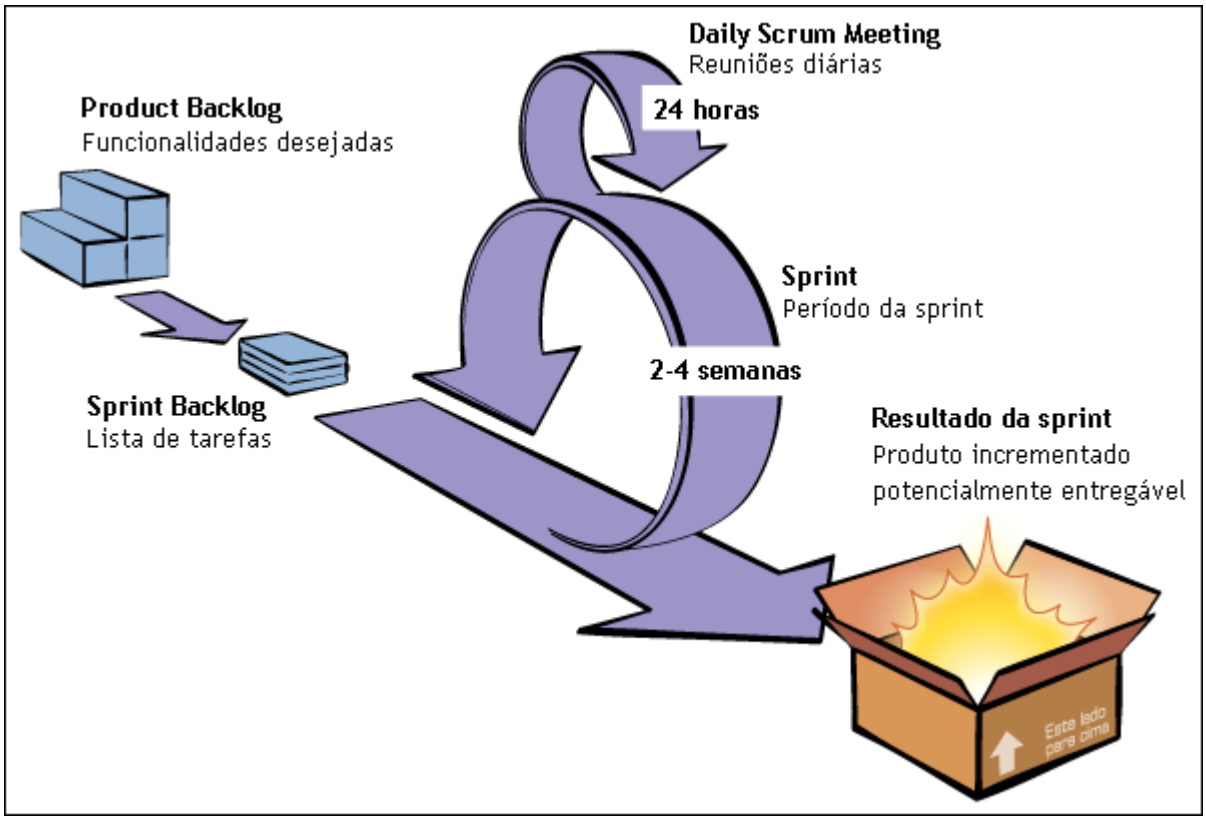
Foi desenvolvido inicialmente por Jeff Sutherland e sua equipe, no início da década de 1990. Seu nome foi inspirado numa jogada de esporte *Rugby*, onde o objetivo é retirar os obstáculos à frente do jogador que correrá com a bola, para que possa avançar o máximo possível no campo e marcar pontos (ORTH; PRIKLADNICKI, 2009). Empresas como Google, Yahoo!, Microsoft, IBM, Cisco, Symantec e Siemens utilizam *Scrum* em alguns de seus projetos (GOMES; FAIAS JUNIOR, 2009).

No *Scrum*, os projetos são divididos em ciclos (tipicamente mensais) chamados de *sprints*. A *sprint* representa um *timebox*[4] dentro do qual um conjunto de atividades deve ser executado.

As funcionalidades a serem implementadas em um projeto são mantidas em uma lista que é chamada de *Product Backlog*. No início de cada *sprint*, faz-se um *Sprint Planning Meeting*, ou seja, uma reunião de planejamento na qual o *Product Owner* prioriza os itens do *Product Backlog* e a equipe seleciona as atividades que ela será capaz de implementar durante a *sprint* que se inicia. As tarefas alocadas em uma *sprint* são transferidas do *Product Backlog* para o *Sprint Backlog* (IMPROVEIT, 2009).

A cada dia de uma *sprint*, a equipe faz uma breve reunião, chamada *Daily Scrum Meeting*. O objetivo é disseminar conhecimento sobre o que foi feito no dia anterior, identificar impedimentos e priorizar o trabalho do dia que se inicia.

Ao final de uma *sprint*, a equipe apresenta as funcionalidades implementadas em uma *Sprint Review Meeting*. Finalmente, faz-se uma *Sprint Retrospective* e a equipe parte para o planejamento da próxima *sprint*. Assim reinicia-se o ciclo. A figura 1 demonstra o ciclo de vida do processo.



Fonte: adaptado de IMPROVEIT (2009).

Figura 1: Ciclo de vida do *framework Scrum*

[4] Período de tempo determinado

2.3.1 Papéis do Scrum

Existem apenas três papéis no *Scrum*, o *Product Owner* (PO), o *Scrum Master* (SM) e o *Scrum Team* (ST).

O *Product Owner* é normalmente o cliente ou alguém interessado no resultado do projeto. Ele tem como principais responsabilidades definir e priorizar as funcionalidades do produto, aceitar ou rejeitar o resultado dos trabalhos, resolver dúvidas da equipe quanto ao entendimento dos requisitos.

O *Scrum Master* é o responsável por garantir que o *Scrum Team* se oriente pelos valores e práticas do *Scrum*. Ele protege a equipe certificando-se de que os membros não se comprometam com compromissos além dos que eles conseguem cumprir dentro de uma *sprint*. O SM facilita o *Daily Scrum Meeting* e se torna o responsável pela remoção de quaisquer obstáculos observados pelo time durante estas reuniões.

Também chamada de equipe, o *Scrum Team* é o conjunto de pessoas que possuem a responsabilidade de desenvolver e entregar as *sprints* realizadas. Deve ter como características ser disciplinada e auto-gerenciada, com atributos multifuncionais e comprometidos com um objetivo comum. Geralmente, uma equipe tem de seis a dez pessoas, embora pode-se ter equipes maiores (IMPROVEIT, 2009).

2.3.2 Cerimônias

No *framework Scrum* tem-se quatro cerimônias (ADAPTWORKS, 2011). As cerimônias são reuniões, formais ou não, que acontecem durante os ciclos das *sprints*.

As reuniões *Sprint Planning*, *Daily Scrum Meeting* e *Sprint Review* caracterizam bem o ciclo de vida de cada *sprint*: início, meio e fim (DÓRIA, 2009).

O *Sprint Planning Meeting* é uma reunião na qual estão presentes o PO, o SM e o ST, bem como qualquer pessoa interessada que esteja representando a gerência ou o cliente. Durante a reunião, o PO descreve as funcionalidades de maior prioridade para a equipe. A equipe faz perguntas durante a reunião de modo que seja capaz de quebrar as funcionalidades em tarefas técnicas, após a reunião. Essas tarefas irão dar origem ao *Sprint Backlog*.

Ao final de cada *sprint*, uma reunião é realizada, chamada *Sprint Review*. Durante esta reunião, o *Scrum Team* apresenta o que foi realizado durante a *sprint*. Tipicamente, esta apresentação é feita na forma de uma demonstração das novas funcionalidades. Participantes da *Sprint Review* incluem o PO, o SM, o ST, a diretoria, clientes e engenheiros de outros projetos e outras pessoas interessadas no resultado da *sprint*.

Logo após a *Sprint Review* ocorre uma reunião entre os membros do time para discutirem o que foi bem na *sprint* e o que precisa ser melhorado para a próxima, denominada *Sprint Retrospective*. Participam todos os membros da equipe e mais o SM – este como facilitador. Esta reunião é de muita importância, pois é através dela que o time consegue aprimorar o processo, para que nas próximas *sprints*, o time evolua e aumente a sua maturidade de desenvolver *software*.

Todos os dias da *sprint*, o ST realiza reuniões diárias. Normalmente, essas reuniões são realizadas no mesmo lugar, na mesma hora do dia. É comum serem realizadas na parte da manhã, para ajudar a estabelecer as prioridades do novo dia de trabalho (IMPROVEIT, 2009).

A *Daily Scrum Meeting* não é uma reunião usada para solucionar problemas ou dúvidas. As dúvidas que surgirem deverão ser postergadas e, de modo geral, serem tratadas pelo grupo envolvido imediatamente após a reunião. O ST responde às seguintes perguntas durante a reunião:

a) o que você fez ontem?;
b) o que você vai fazer hoje?;
c) existe algum impedimento?.

Existem casos em que o ST não consegue remover os impedimentos diretamente (normalmente questões mais técnicas), mas deve assegurar que alguém do time solucione rapidamente o problema.

2.3.3 Artefatos

Os artefatos do *Scrum* são as ferramentas básicas para se trabalhar com este *framework* e servem como guias e indicadores durante o processo. São divididos em três, o *Product Backlog*, o *Sprint Backlog* e o *Burndown Charts*.

O *Product Backlog* é uma lista de requisitos do sistema ou produto em desenvolvimento. Este artefato é usado desde o planejamento do projeto até sua conclusão e evolui à medida que o produto e o ambiente onde é usado evoluem (SCHWABER, 2004).

Os itens do *Product Backlog* devem ser estimados e priorizados. Para estimar, pode ser usada a técnica do *Planning Poker*, que consiste em um jogo de cartas baseada na sequência numérica de *Fibonacci*[5] (1, 2, 3, 5, 8, 13, ...).

Na aplicação do *Planning Poker*, cada participante da *Sprint Planning* (ST, SM e PO) fica com um conjunto de cartas em mãos. A atribuição do grau de dificuldade de cada item do *Product Backlog* é feita com as cartas, chegando a um consenso sobre qual é a pontuação final do item discutido. De acordo com Marçal (2007), é importante ter um moderador para que o processo de atribuição seja produtivo e não perca o foco do trabalho.

A figura 2 ilustra um exemplo de priorização dos itens do *Product Backlog*, baseado na importância. Esse método de priorização é nomeado por alguns autores como método MoSCoW – *Must, Should, Could, Want*, e consiste em numerar os requisitos de acordo com sua importância (ALBUQUERQUE, 2007).

Fonte: adaptado de ALBUQUERQUE (2007).

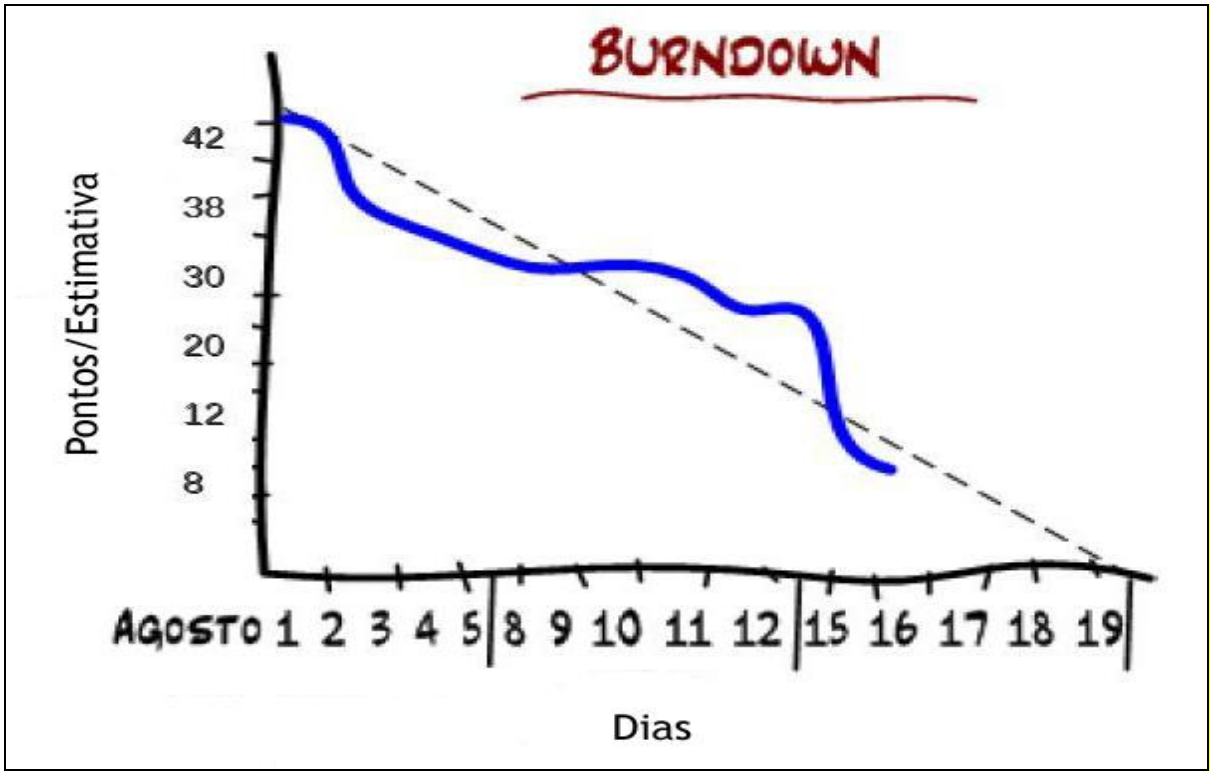
Figura 2: Priorização do *Product Backlog* com o método MoSCoW

[5] É utilizada a sequência numérica de *Fibonacci*, pois a distância entre os valores aumenta no decorrer da sequência, auxiliando na aplicação do grau de dificuldade para cada item do *Product Backlog*.

Chama-se *Sprint Backlog* a lista de tarefas selecionadas pela equipe para serem trabalhadas durante uma iteração. O time monta esta lista atribuindo uma estimativa de esforço para cada uma das tarefas durante a reunião de planejamento (SCHWABER, 2004).

O gráfico de *burndown*, também chamado de *burdown chart*, é um artefato disponível para toda a equipe durante a iteração para que a mesma possa acompanhar a quantidade de trabalho realizado e restante em relação ao número de dias. Ele exibe o conflito entre o que foi planejado e a velocidade real de execução do trabalho (SCHWABER, 2004).

À medida que o progresso do trabalho for registrado, o gráfico mostra o quanto de trabalho precisa ser feito até o final da *sprint*. Desta forma é possível perceber se a *sprint* será concluída dentro do prazo ou não. Na figura 3, pode-se observar que a linha pontilhada demonstra a previsão. Já a linha sólida, representa o trabalho executado.



Fonte: adaptado de Menezes (2011).

Figura 3: Exemplo do gráfico de *burndown* de uma *sprint*

2.4 TRABALHOS CORRELATOS

Devido ao grande número de trabalhos correlatos encontrados sobre o tema ferramentas e sistemas de gerenciamento de projetos de *software*, optou-se por alguns com

melhor suporte ao *Scrum*, tais como o *Scrum Project*, o *DotProject*, o Pronto! e o trabalho desenvolvido por Mello (2010).

2.4.1 Scrum Project

Kluge (2009) apresentou um Trabalho de Conclusão de Curso (TCC) na Universidade do Vale do Itajaí, intitulado de *Scrum Project*, com o objetivo de criar uma ferramenta de apoio ao gerenciamento de projetos de *software* baseada nos princípios do *Scrum*.

Na implementação dessa ferramenta, foram utilizados softwares livres como o PHP, para linguagem de programação e PostgreeSQL para banco de dados. A ferramenta implementa os princípios básicos do *Scrum*, como cadastro de *sprints* e registro de reuniões, além de fornecer alguns relatórios de acompanhamento do projeto.

O *Scrum Project* não disponibiliza a visualização do gráfico de *burndown*, que tornaria o acompanhamento e monitoramento do projeto mais fácil. A figura 4 apresenta a tela de edição de um projeto inserido no sistema.

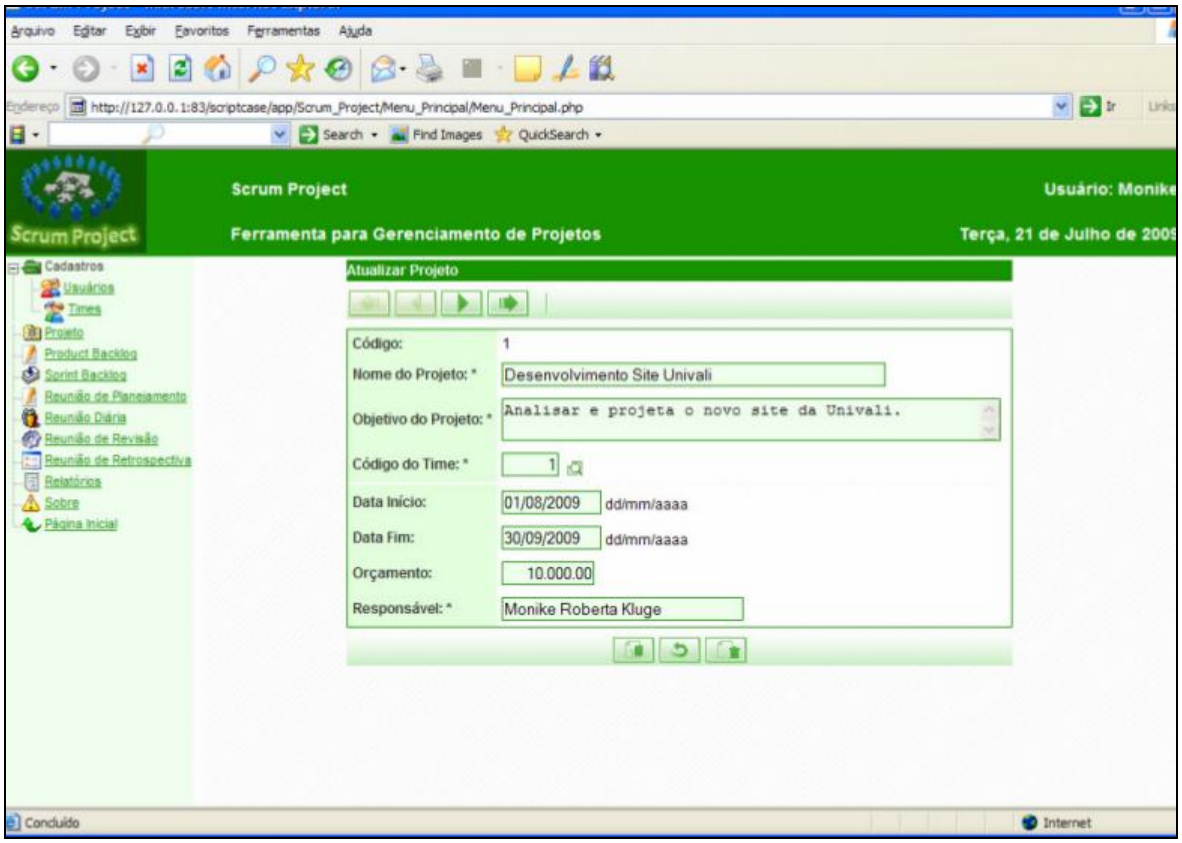


Figura 4: Tela de atualização do cadastro de um projeto

2.4.2 DotProject

Na monografia de TCC, Pereira (2005) apresenta uma ferramenta *web* para estender e adaptar o *software opensource*, para gerenciamento de projetos, dotProject, utilizando os conceitos de metodologias ágeis e do *framework Scrum*.

A ferramenta implementa os conceitos de artefatos, cadastro e priorização de tarefas do modelo *Scrum*, assim como o acompanhamento e monitoramento da evolução e progresso do projeto baseado na própria estrutura da aplicação dotProject. A figura 5 demonstra a listagem de tarefas que compreendem o *Product Backlog*.

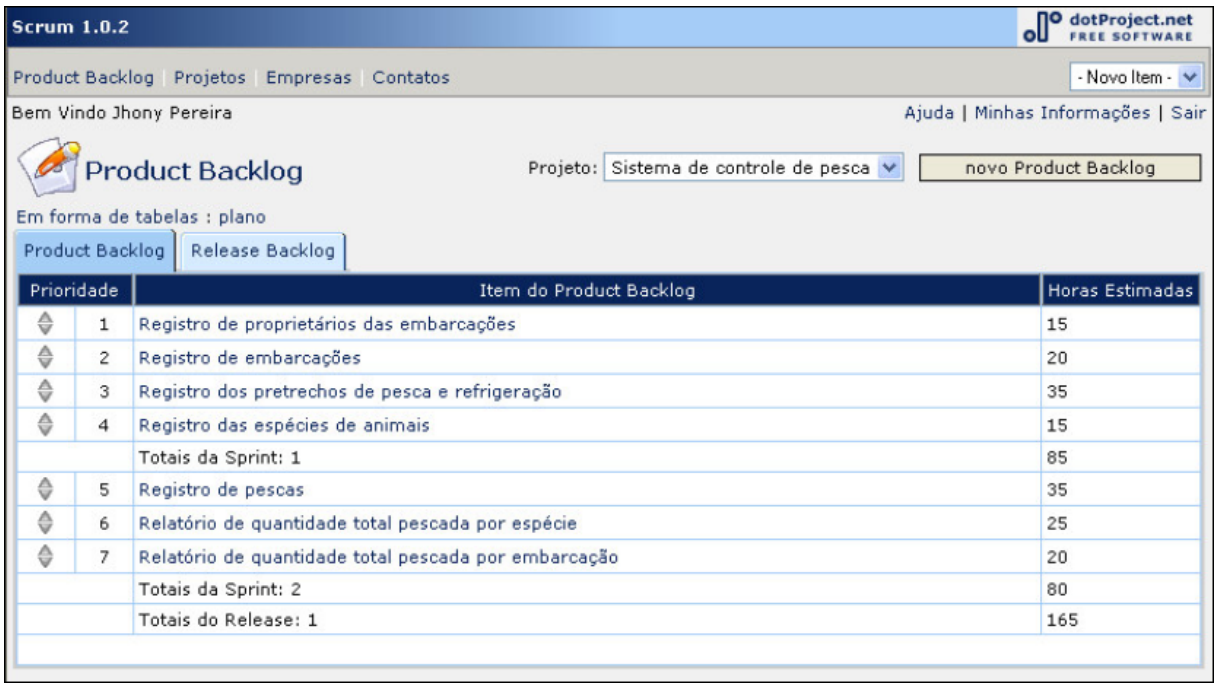


Figura 5: Tela de listagem do *Product Backlog*

2.4.3 Pronto**!**

Gomes e Faias Junior (2009) apresentaram como trabalho de TCC, o "Pronto. *agile software development*", onde desenvolveram um *software* para cadastros, gerenciamento e acompanhamento de solicitações com objetivo de atingir a satisfação dos clientes, demonstrando maior transparência do que será entregue nas versões do produto.

Foi desenvolvido baseado na metodologia ágil *Scrum* e tem um diferencial que é o uso do quadro *Kanban*[6]. O *Kanban* simula um quadro para gestão visual, onde os *tickets* (no caso deste *software*, as solicitações) são exibidos em pequenos quadrados, conforme mostrado na figura 6. Suas cores definem se são defeitos (vermelho), estórias (amarelo) ou tarefas (cinza).

A ferramenta foi desenvolvida utilizando *Java*, JSP, *Tomcat*[7] e o padrão de arquitetura MVC[8]. No endereço eletrônico "http://pronto.bluesoft.com.br/", acessado em 25/09/2010, Gomes e Faias Junior (2009) disponibilizaram uma demonstração da ferramenta bem como a própria monografia.

Figura 6: Tela com as tarefas pendentes no *Kanban*

2.4.4 TCC

No trabalho TCC de MELLO (2010), foi desenvolvida uma ferramenta *web* para gerenciar projetos de *softwares* baseados no *Scrum*, usando as tecnologias da Microsoft como o Visual Studio, Expression Blend e Silverlight. Para banco de dados, foi utilizado o MySQL.

---

[6] *Kanban* é uma palavra de origem japonesa e seu significado é "placar". Na indústria de *software* este termo é aplicado à sistemas de gestão à vista onde o resumo de cada tarefa da iteração é exibido em um cartão e posicionado em um quadro disponível para consulta de qualquer pessoa envolvida no projeto. (GOMES; FAIAS JUNIOR, 2009).

[7] *Tomcat* é um servidor de aplicações *web*.

[8] MVC é um padrão de arquitetura de aplicações que visa separar a lógica da aplicação (*Model*), da interface do usuário (*View*) e do fluxo da aplicação (*Controller*).

A ferramenta implementa os artefatos e cerimônias básicos disponíveis no *Scrum*, além da emissão de relatórios para acompanhamento dos projetos pelo *Product Owner*, como problemas identificados durante as *sprints* e andamento do cronograma. A ferramenta não disponibiliza a visualização do gráfico de *burndown*. A figura 7 apresenta a tela de cadastro de itens ao *Product Backlog*. Na figura 8, demonstra-se o relatório gerado dos problemas encontrados durante um projeto.

Figura 7: Tela de cadastro de itens para o *Product Backlog*

06/07/2010

Listagem de problemas encontrados

Sprint	Nome	Data	Problemas encontrados
8	João da Silva	05/07/2010	Falta de comprometimento com a equipe.
8	João da Silva	06/07/2010	Alguns itens solicitados não foram entregues.
8	João da Silva	07/07/2010	Estávamos com problemas de conexão com o banco.
8	João da Silva	08/07/2010	Vários requisitos não estavam bem definidos.
8	João da Silva	09/07/2010	Cliente não enviou os arquivos solicitados.

Figura 8: Relatório gerado com os problemas encontrados durante um projeto

3 DESENVOLVIMENTO DO SISTEMA

Neste capítulo são descritas as informações levantadas, detalhes da especificação, os diagramas de casos de uso e entidade relacionamento, a operacionalidade do sistema e ao final, os resultados e discussão.

3.1 LEVANTAMENTO DE INFORMAÇÕES

No início do segundo semestre de 2009, teve-se a oportunidade de participar de um projeto de melhoria de processo, denominado PRODUSOFT, na empresa. Esse projeto visava uma nova postura e orientação no desenvolvimento dos *softwares* produzidos pela empresa, buscando qualidade e maior previsibilidade na entrega das versões e releases.

Optou-se então, pelas metodologias ágeis e pelo uso do *Scrum*, para formatar e formar a base do processo que seria desenhado. O *Scrum* foi escolhido para orientar a gestão dos projetos e pelas suas boas características, como a pouca burocracia no processo, as pequenas e rápidas iterações (*sprints*), a necessidade de comprometimento efetivo de toda a equipe de trabalho, pela entrega de valor ao cliente com maior frequência, pela agilidade nas respostas à mudanças (no nosso caso, mudanças de requisitos), entre outros aspectos.

Diante desse cenário, a empresa utilizava ferramentas para gestão de projetos que não estavam alinhadas com a nova proposta de trabalho. Não tinha-se um sistema capaz de suportar a estrutura do *Scrum* e dos próprios métodos ágeis. Era difícil controlar as solicitações (formar e manter priorizado o *Product Backlog*), planejar as *sprints*, registrar os impedimentos, acompanhar a evolução do desenvolvimento do produto sem que se fizesse uso de planilhas para auxiliar e complementar a deficiência da ferramenta utilizada.

No segundo semestre de 2010, iniciou-se a disciplina de TCC-I, onde foi proposto o desenvolvimento de uma ferramenta com suporte à estrutura do *Scrum* e que a adoção dos métodos ágeis fosse mais tranquila e amparada por uma boa ferramenta de gestão.

Os requisitos básicos e necessários para desenvolver esse sistema de gestão ágil de

projetos de *software*, foram extraídos da experiência, vivência e *know-how*[9] na adoção e implantação do *Scrum*, onde participou-se ativamente do projeto de melhoria de processo na empresa.

3.2 ESPECIFICAÇÃO

Esta seção descreve os requisitos funcionais (RF) e não funcionais (RNF), bem como os diagramas de casos de uso e de entidade-relacionamento desenvolvidos para o sistema. A ferramenta Enterprise Architect (EA), em sua versão 7.5.850, foi utilizada na elaboração dos diagramas de casos de uso e a ferramenta *case* DBDesigner4, versão 4.0.5.0.6.

3.2.1 Requisitos funcionais e não funcionais

Nesta sub-seção serão apresentados os principais RFs e RNFs. O quadro 1 apresenta os requisitos funcionais e sua rastreabilidade com seus respectivos casos de uso.

Requisitos funcionais	**Caso de uso**
RF01: O sistema deverá permitir ao administrador do sistema o cadastramento de usuários.	UC001
RF02: O sistema deverá permitir ao administrador do sistema o cadastramento de projetos.	UC002
RF03: O sistema deverá permitir ao administrador do sistema a associação de usuários a um projeto cadastrado.	UC003
RF04: O sistema deverá permitir ao *Product Owner* o cadastramento de estórias de usuários e/ou *bugs* para um *Product Backlog*.	UC004
RF05: O sistema deverá permitir ao *Product Owner* a priorização dos itens do *Product Backlog*.	UC005
RF06: O sistema deverá permitir ao *Product Owner* o cadastramento de *releases* de um projeto.	UC006
RF07: O sistema deverá permitir ao *Product Owner* o planejamento de uma *release*, associando os itens do *Product Backlog*.	UC007
RF08: O sistema deverá permitir ao *Scrum Master* o cadastramento de *sprints* de uma *release*.	UC008
RF09: O sistema deverá permitir ao *Scrum Master* o planejamento de uma	UC009

[9] Termo em inglês, utilizado para designar uma técnica, um conhecimento ou uma capacidade desenvolvida por uma organização ou por uma pessoa.

sprint, associando os itens planejados em uma *release*.	
RF10: O sistema deverá permitir ao *Scrum Master* consultar as cerimônias ocorridas durante uma *sprint*.	UC010
RF11: O sistema deverá permitir ao *Scrum Master* o acompanhamento de um impedimento, gerado pelo *Scrum Team*.	UC011
RF12: O sistema deverá permitir ao *Scrum Master* o fechamento de uma *sprint*.	UC012
RF13: O sistema deverá permitir ao *Scrum Team* consultar as cerimônias ocorridas durante uma *sprint*.	UC010
RF14: O sistema deverá permitir ao *Scrum Team* o cadastramento de tarefas de trabalho para uma estória de usuário.	UC013
RF15: O sistema deverá permitir ao *Scrum Team* o cadastramento de *bugs* para uma estória de usuário.	UC014
RF16: O sistema deverá permitir ao *Scrum Team* o registro de impedimentos para uma tarefa de trabalho.	UC015
RF17: O sistema deverá permitir ao *Scrum Team* o registro de impedimentos para um *bug*.	UC015
RF18: O sistema deverá permitir ao *Scrum Team* a consulta do gráfico de *burndown* de uma *sprint*.	UC016
RF19: O sistema deverá permitir ao *Scrum Team* a atualização do status de uma tarefa de trabalho ou *bug*, do qual é responsável.	UC017
RF20: O sistema deverá permitir ao *Scrum Team* o cadastramento das cerimônias de uma *sprint*.	UC018

Quadro 1: Requisitos funcionais

O quadro 2 apresenta os requisitos não funcionais do sistema.

Requisitos não funcionais
RNF01: O sistema deverá armazenar as senhas dos usuários de forma criptografada.
RNF02: O sistema deverá ser desenvolvido utilizando a linguagem PHP.
RNF03: O sistema deverá utilizar o banco de dados MySQL.
RNF04: O sistema deverá ser desenvolvido para a plataforma *web*.
RNF05: O sistema deverá ser desenvolvido utilizando o *framework* CakePHP.

Quadro 2: Requisitos não funcionais

3.2.2 Diagramas de casos de uso

Os diagramas de casos de uso estão divididos em visões, conforme os tipos de usuários, o administrador do sistema, o *Product Owner*, o *Scrum Master* e o *Scrum Team*. Exceto o administrador do sistema, os demais tipos de usuários fazem parte dos papéis do *Scrum*. Os principais casos de uso estão descritos no Apêndice A.

Na figura 9, tem-se o diagrama de caso de uso na visão do administrador do sistema.

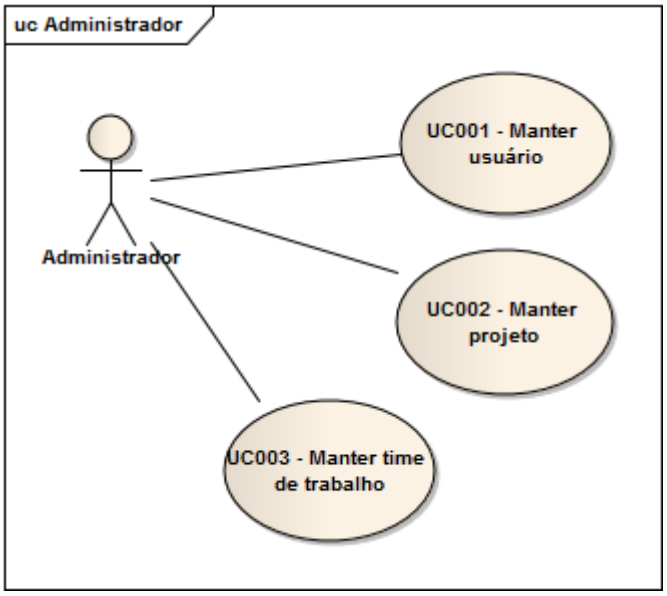


Figura 9: Diagrama de caso de uso na visão do administrador do sistema

Na figura 10, têm-se o diagrama de caso de uso na visão do *Product Owner*.

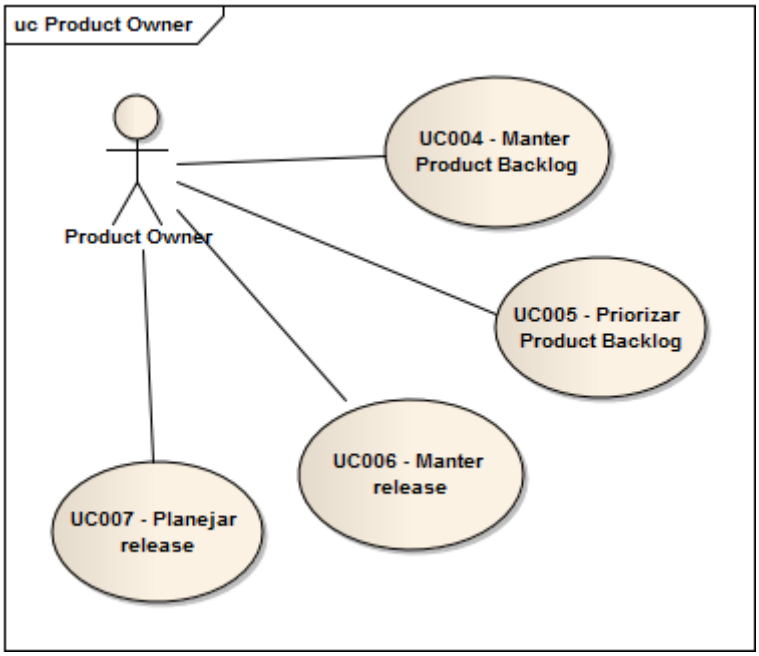


Figura 10: Diagrama de caso de uso na visão do *Product Owner*

Na figura 11, têm-se o diagrama de caso de uso na visão do *Scrum Master*.

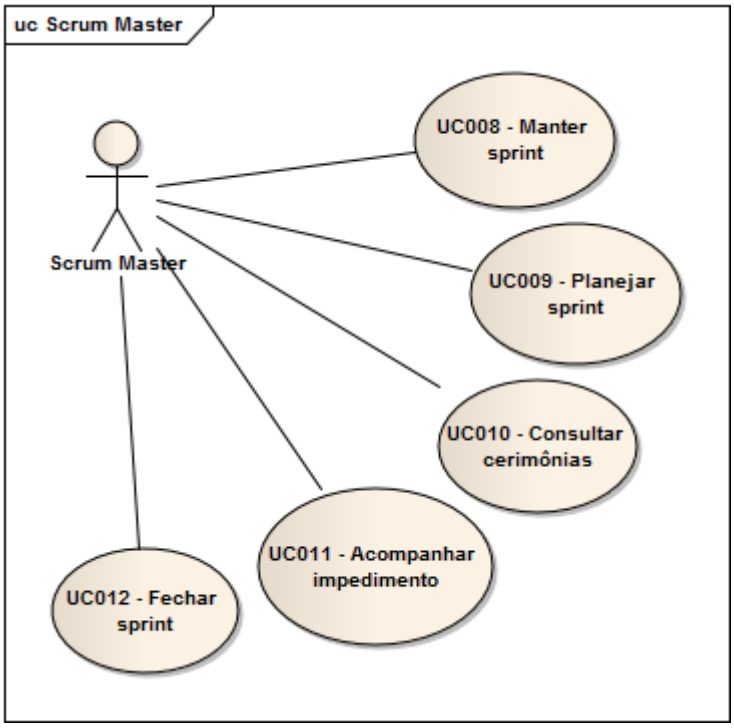


Figura 11: Diagrama de caso de uso na visão do *Scrum Master*

Na figura 12, têm-se o diagrama de caso de uso na visão do *Scrum Team*.

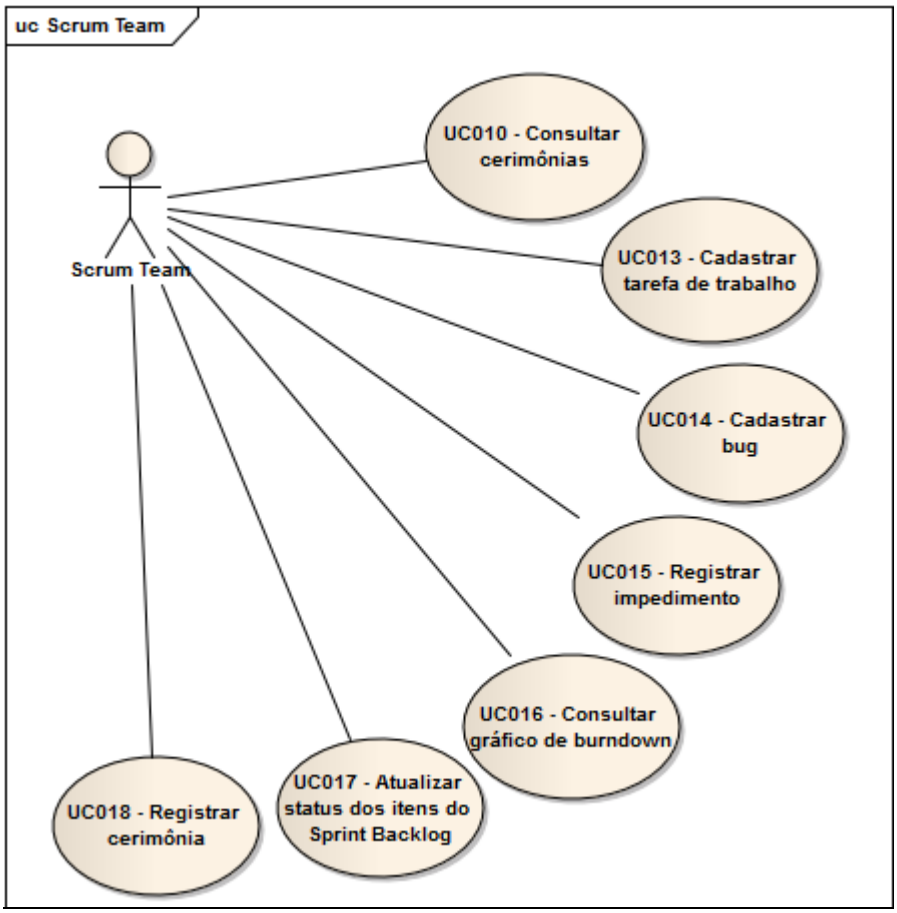


Figura 12: Diagrama de caso de uso na visão do *Scrum Team*

3.2.3 Diagrama entidade-relacionamento

O Diagrama Entidade-Relacionamento (DER) é um diagrama que descreve o modelo de dados de um sistema com alto nível de abstração. A figura 13 apresenta o DER com as entidades que serão persistidas no banco de dados do sistema.

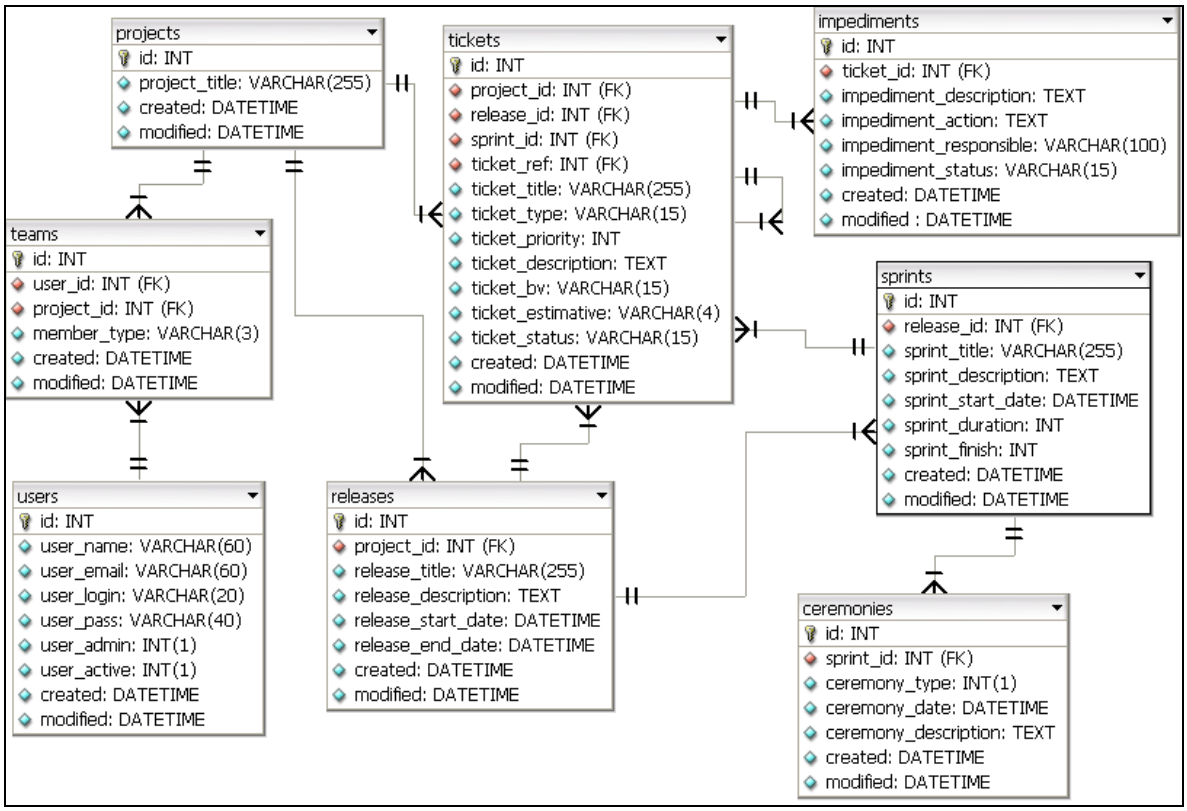


Figura 13: Diagrama entidade-relacionamento

O dicionário de dados está descrito no Apêndice B. Entretanto, a seguir é apresentada uma breve descrição das entidades utilizadas para o desenvolvimento do sistema:

a) *projects*: entidade responsável por armazenar informações referentes aos projetos;
b) *releases*: entidade responsável por armazenar informações referentes aos *releases* de um projeto já cadastrado;
c) *teams*: entidade responsável por armazenar informações referentes à associação do usuários cadastrados com os tipos de papéis do *Scrum*;
d) *users*: entidade responsável por armazenar informações referentes aos usuários que utilizarão o sistema;
e) *tickets*: entidade responsável por armazenar informações referentes às estórias de usuários bem como os *bugs* e tarefas de trabalhos que deverão ser implementadas em um projeto;

f) *sprints*: entidade responsável por armazenar as iterações de uma *release*;
g) *impediments*: entidade responsável por armazenar os impedimentos ocorridos na implementação de uma estória de usuário, tarefa de trabalho ou *bug*;
h) *ceremonies*: entidade responsável por armazenar informações referentes as cerimônias ocorridas durante uma *sprint* (iteração).

3.3 IMPLEMENTAÇÃO

Nesta seção estão apresentadas informações sobre as ferramentas e técnicas utilizadas no desenvolvimento do sistema, juntamente com a operacionalidade da implementação.

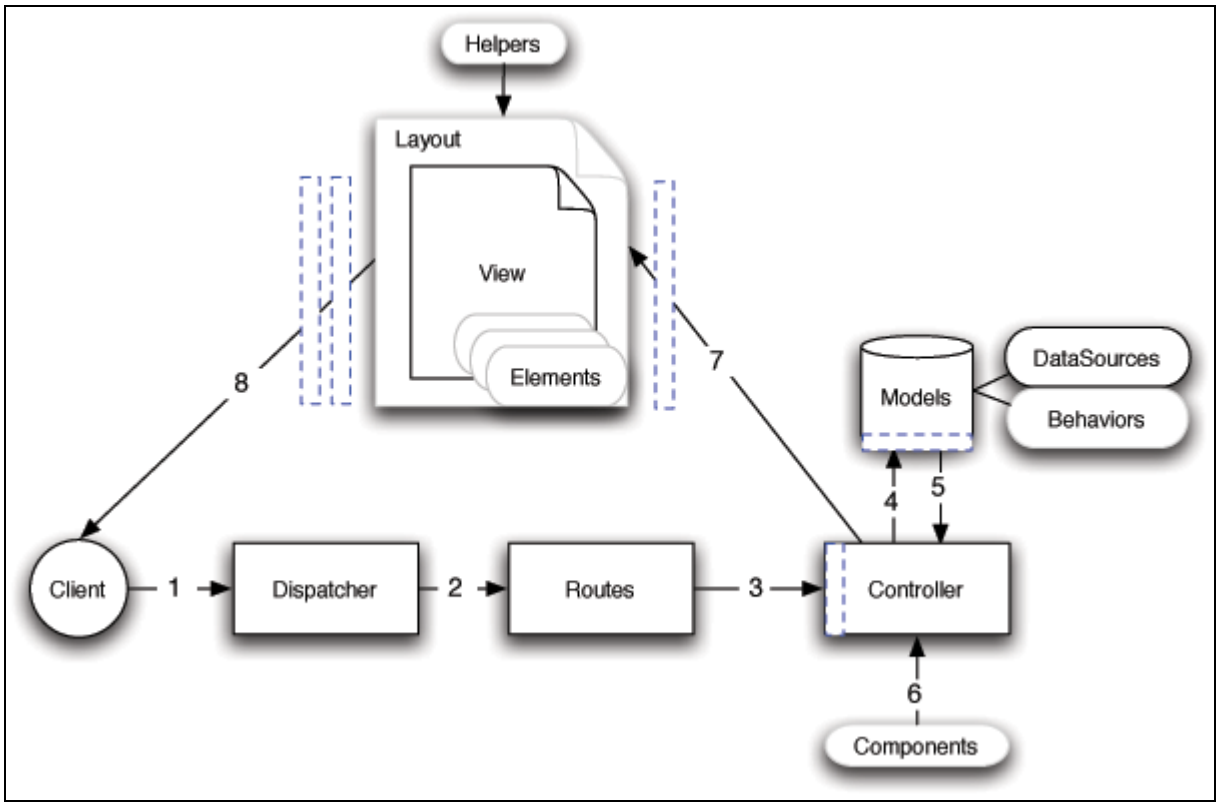
3.3.1 Técnicas e ferramentas utilizadas

No desenvolvimento do sistema, foram utilizados os *softwares*: a) *Apache 2.2.17* – como servidor de aplicação *web*; b) MySQL 5.5.8 – como banco de dados; c) phpMyAdmin[10] 3.3.9 – como gerenciador de banco de dados; d) *Hypertext Preprocessor* (PHP) 5.3.5 – como linguagem de programação; e) CakePHP – como *framework* de desenvolvimento; e f) Eclipse PDT[11] – como ambiente de desenvolvimento.

O CakePHP é um *framework* escrito em PHP, gratuito e de código aberto, para desenvolvimento rápido e ágil em PHP, fornecendo uma arquitetura extensível para desenvolvimento, manutenção e distribuição de aplicações, utilizando padrões de projeto conhecidos como o MVC (CAKEPHP, 2010). O *framework* provê uma base robusta para a aplicação, podendo manipular cada aspecto, desde a requisição inicial do cliente até o ponto final da renderização da página. Na figura 14, é apresentada a estrutura de requisições do CakePHP.

[10] A aplicação phpMyAdmin fornece um conjunto de *scripts* PHP para gerenciar banco de dados MySQL, com interface *web*.

[11] O projeto Eclipse PDT prevê uma ferramenta de desenvolvimento baseada na plataforma Eclipse. Ela abrange todos os componentes de desenvolvimento necessário para desenvolver na linguagem PHP (ECLIPSE, 2011).



Fonte: CAKEPHP (2011).

Figura 14: Estrutura típica de requisições no CakePHP

Na codificação de classes, métodos, atributos, variáveis e comentários do sistema e para o bando de dados, foi utilizado o idioma inglês. O inglês foi escolhido pela facilidade e comodidade no desenvolvimento, além do fato de que tanto os comandos do PHP como o CakePHP foram escritos em inglês.

3.3.2 Operacionalidade da implementação

Esta sub-seção apresenta as principais telas do sistema desenvolvido com uma breve apresentação de suas funcionalidades.

O sistema foi dividido em visões, conforme o papel de cada usuário, sendo administrador do sistema, *Product Owner*, *Scrum Master* e *Scrum Team*. Todos os dados presentes nas telas mostradas a seguir, são fictícios e servem apenas para exemplificar o uso do sistema.

3.3.2.1 Acessando o sistema

Na tela apresentada na figura 15, o usuário deve informar seu nome de usuário e senha para acessar o sistema.

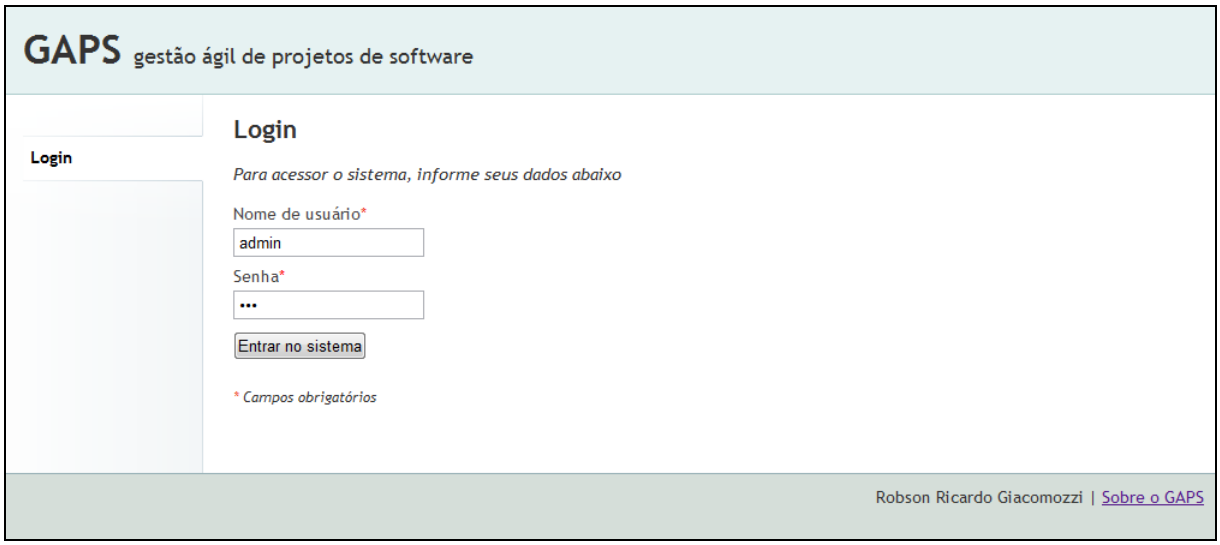


Figura 15: Tela de *login* do sistema

Todos os usuários acessam essa tela, e para os que participam de algum projeto como *Product Owner*, *Scrum Master* ou *Scrum Team*, após o *login* é apresentada a tela de seleção de projeto, conforme mostra a figura 16.

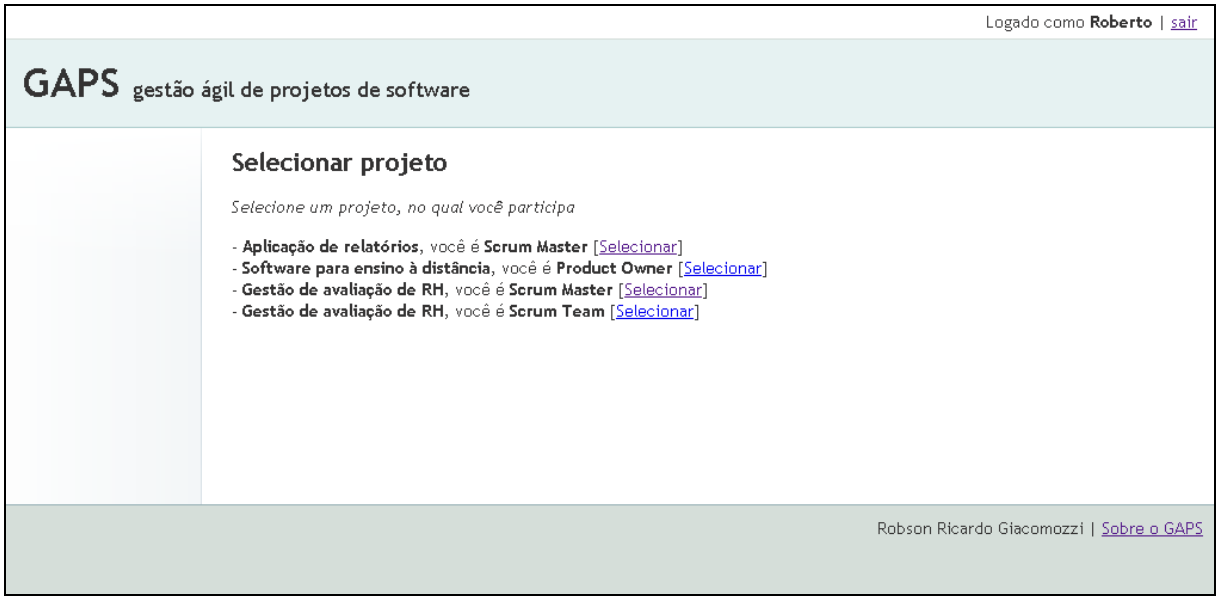


Figura 16: Tela de seleção de projeto e papel

A tela de seleção de projeto lista os times disponíveis, do qual o usuário faz parte (participa). Nesse momento, o usuário já está autenticado no sistema e deve selecionar o projeto e papel para prosseguir com o acesso ao sistema.

3.3.2.2 Visão do administrador do sistema

Na figura 17 é possível visualizar a tela inicial do sistema, onde o administrador do sistema poderá gerenciar os usuários, projetos e times de trabalhos que estarão disponíveis no sistema.

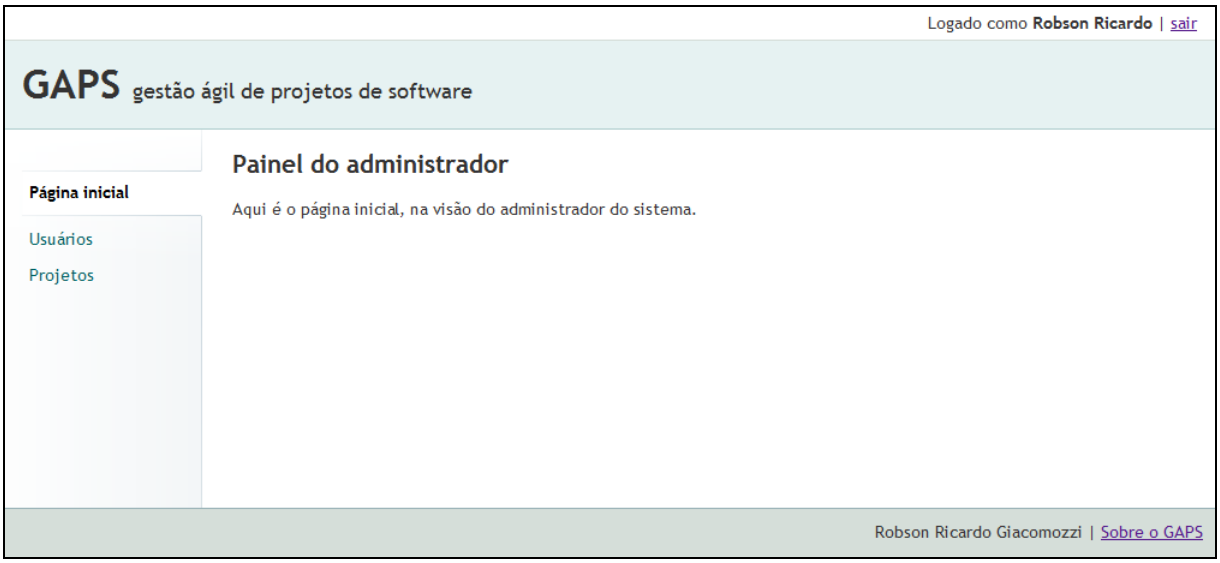


Figura 17: Tela inicial do administrador do sistema

Ao acessar o item de *menu* "Usuário", será apresentado ao administrador a tela de listagem dos usuários cadastrados no sistema. A figura 18 demonstra a tela dos usuários já cadastrados, onde é possível adicionar novos registros, editar e excluir os cadastros já existentes.

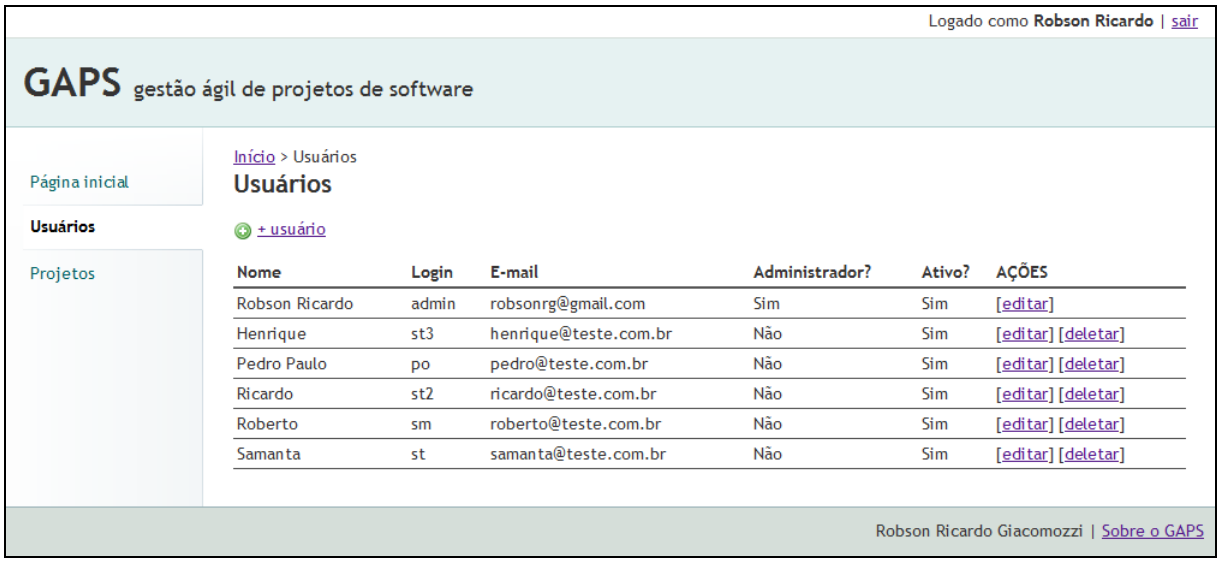


Figura 18: Tela dos usuários cadastrados

O administrador tem a possibilidade de cadastrar novos usuários, conforme figura 19.

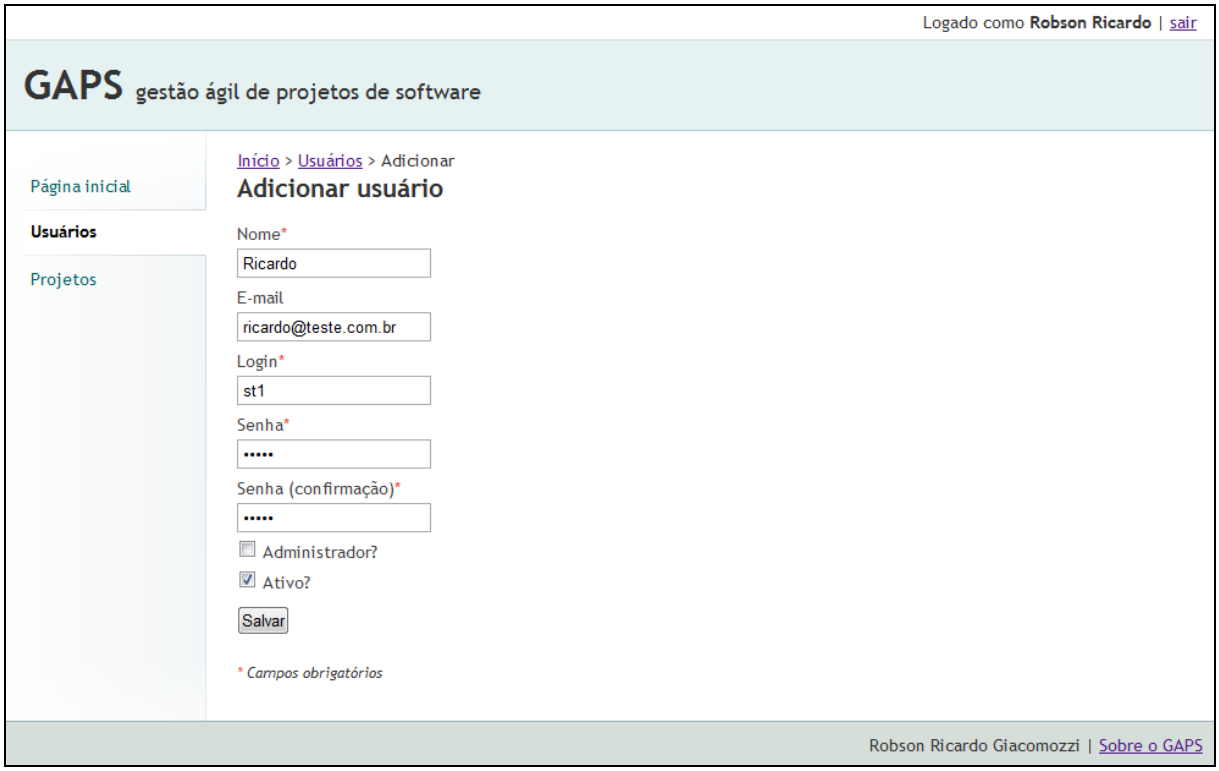


Figura 19: Tela de cadastro de usuários

No item de *menu* "Projetos", será apresentada a tela de listagem dos projetos cadastrados no sistema. A figura 20 demonstra a tela de projetos, onde é possível adicionar novos registros, editar e excluir os já existentes, além da possibilidade de acessar a tela de definição do time de trabalho para o projeto.

Figura 20: Tela dos projetos cadastrados

Para cada projeto cadastrado, é possível definir um time de trabalho. A definição desse time é feita adicionando ou removendo participantes, conforme mostra a figura 21.

Figura 21: Tela de cadastro de um participante

A figura 22 mostra a tela de listagem dos participantes do time de trabalho do projeto.

Figura 22: Tela dos participantes do time de trabalho de um projeto

3.3.2.3 Visão do *Product Owner*

Nesta visão do sistema, o PO poderá gerenciar o cadastro de *releases*, manter e priorizar os itens do *Product Backlog* e planejar as *releases*.

Na figura 23 é possível visualizar a tela inicial do sistema onde, conforme a prioridade definida, apresenta os itens do *Product Backlog*.

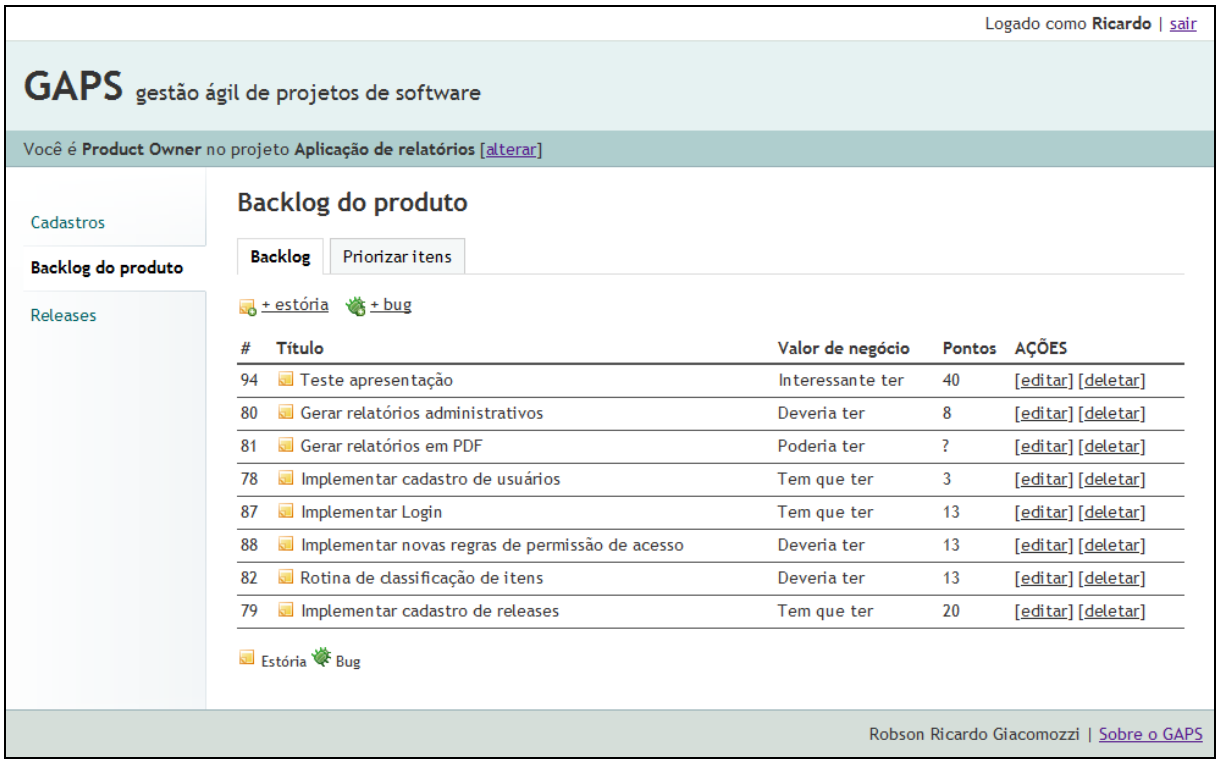


Figura 23: Tela dos itens cadastrados no *Product Backlog*

Ainda na tela do "*Backlog* do produto", o PO tem a possibilidade de cadastrar uma nova estória de usuário, incrementando o *Product Backlog*, conforme mostra a figura 24.

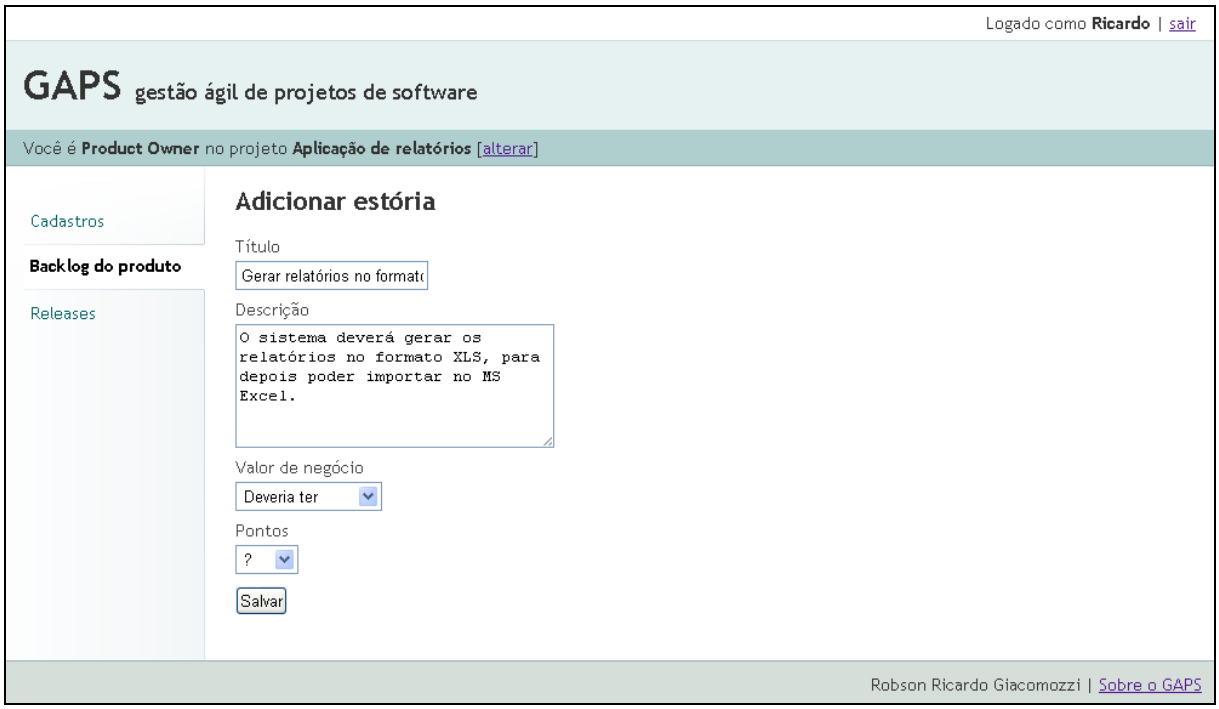


Figura 24: Tela de cadastro de uma estória de usuário

A figura 25 mostra a tela de cadastro de *bugs*, que também são adicionados ao *Product Backlog*.

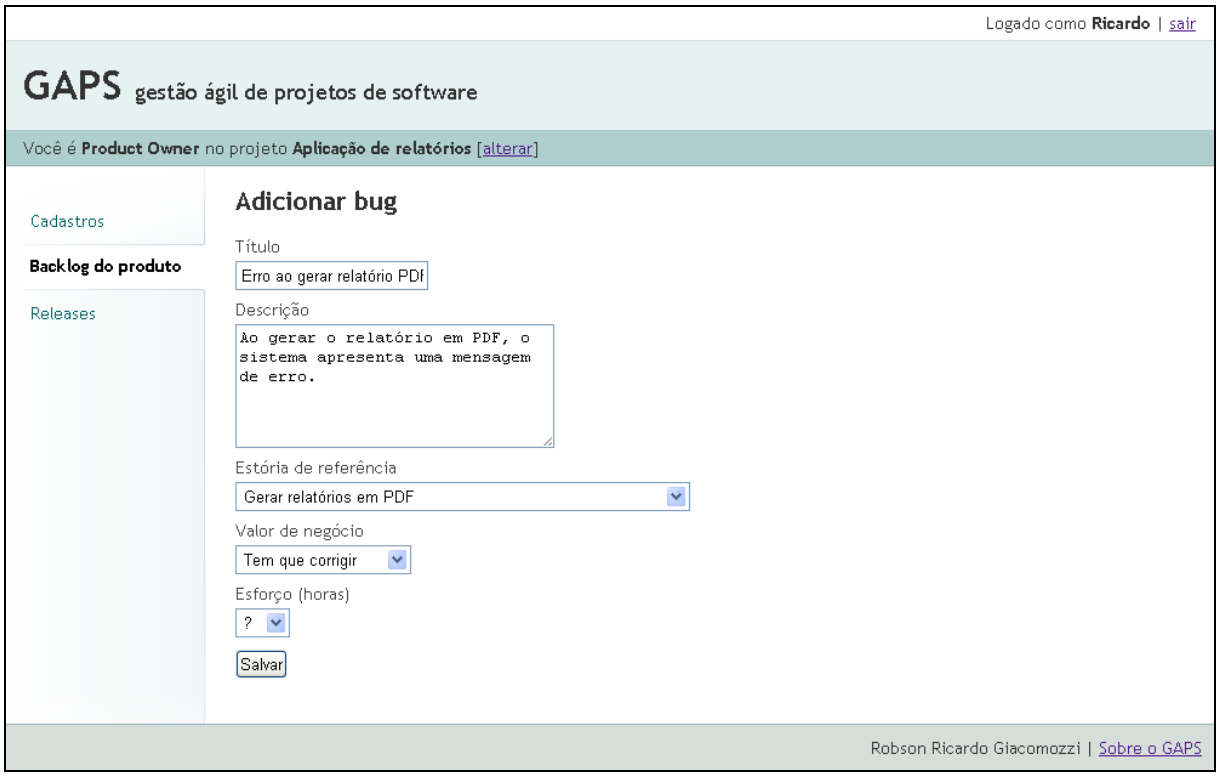


Figura 25: Tela de cadastro de um *bug*

O PO é responsável pelo Retorno sobre Investimento[12] (ROI) do projeto. Ele deve priorizar os itens do *Product Backlog* de forma a obter o maior retorno possível (SILVEIRA, 2008). Na figura 26 é demonstrada a tela onde o PO prioriza o *Product Backlog*, arrastando os itens para cima ou para baixo, reordenando-os conforme a necessidade. Ao acessar o item de *menu* "*Backlog* do produto" e, na sequencia, a aba "Priorizar itens", o sistema carrega a tela de priorização dos itens.

[12] ROI, ou no inglês, *Return On Investment*, é a relação entre o dinheiro ganho ou perdido através de um investimento. É o dinheiro utilizado para investir e que receberá retorno (com lucro) ou com prejuízo (CASTRO, 2010).

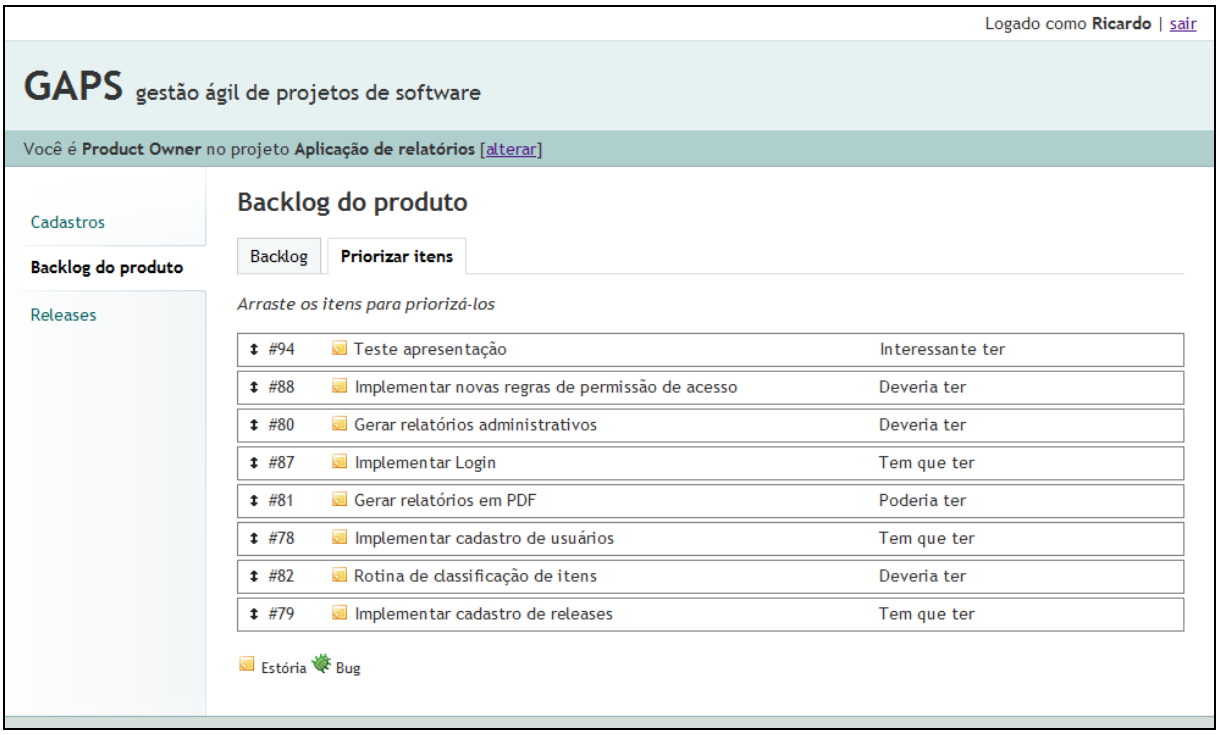


Figura 26: Tela de priorização dos itens do *Product Backlog*

Na figura 27, é apresentado o código-fonte do método de gravação das prioridades do *Product Backlog*.

```
public function po_priorize()
{
    $this->layout = 'ajax';

    //UPDATE
    if ($this->params['form'])
    {
        $id = $this->params['form']['id'];
        $oldPosition = $this->params['form']['oldPosition'];
        $newPosition = $this->params['form']['newPosition'];

        //save priotiry of item moved
        $this->Ticket->id = $id;
        $this->Ticket->saveField('ticket_priority', $newPosition);

        //other items
        #defined the new priority
        $operator = ($oldPosition > $newPosition) ? array('+', '>=', '<=') : array('-', '<=', '>=');
        #update priority for other tickets
        $this->Ticket->updateAll(
            array('ticket_priority' => 'ticket_priority '.$operator[0].' 1'), //value
            array( //filters
                'Ticket.project_id' => $this->Session->read('Project.id'),
                'Ticket.id <>' => $id,
                'Ticket.ticket_priority '.$operator[1] => $newPosition,
                'Ticket.ticket_priority '.$operator[2] => $oldPosition
            )
        );
    }

    //SET
    $this->set('listTickets', $this->_tickets());
}
```

Figura 27: Código-fonte do método de gravação da priorização do *Product Backlog*

No item de *menu* "*Releases*", é apresentada a tela de planejamento de *releases*. A primeira ação nesta tela, será a seleção da *release* a ser planejada, conforme mostra a figura 28.

Figura 28: Seleção da *release* a ser planejada

A figura 29 apresenta, na aba "Planejar *release*", a possibilidade de planejar (associar) os itens, ainda não planejados, do *Product Backlog* ao escopo da *release* selecionada. Essa associação ocorre selecionando os itens e clicando nos botões de ação "Planejar itens para a *release*" ou "Remover itens da *release*".

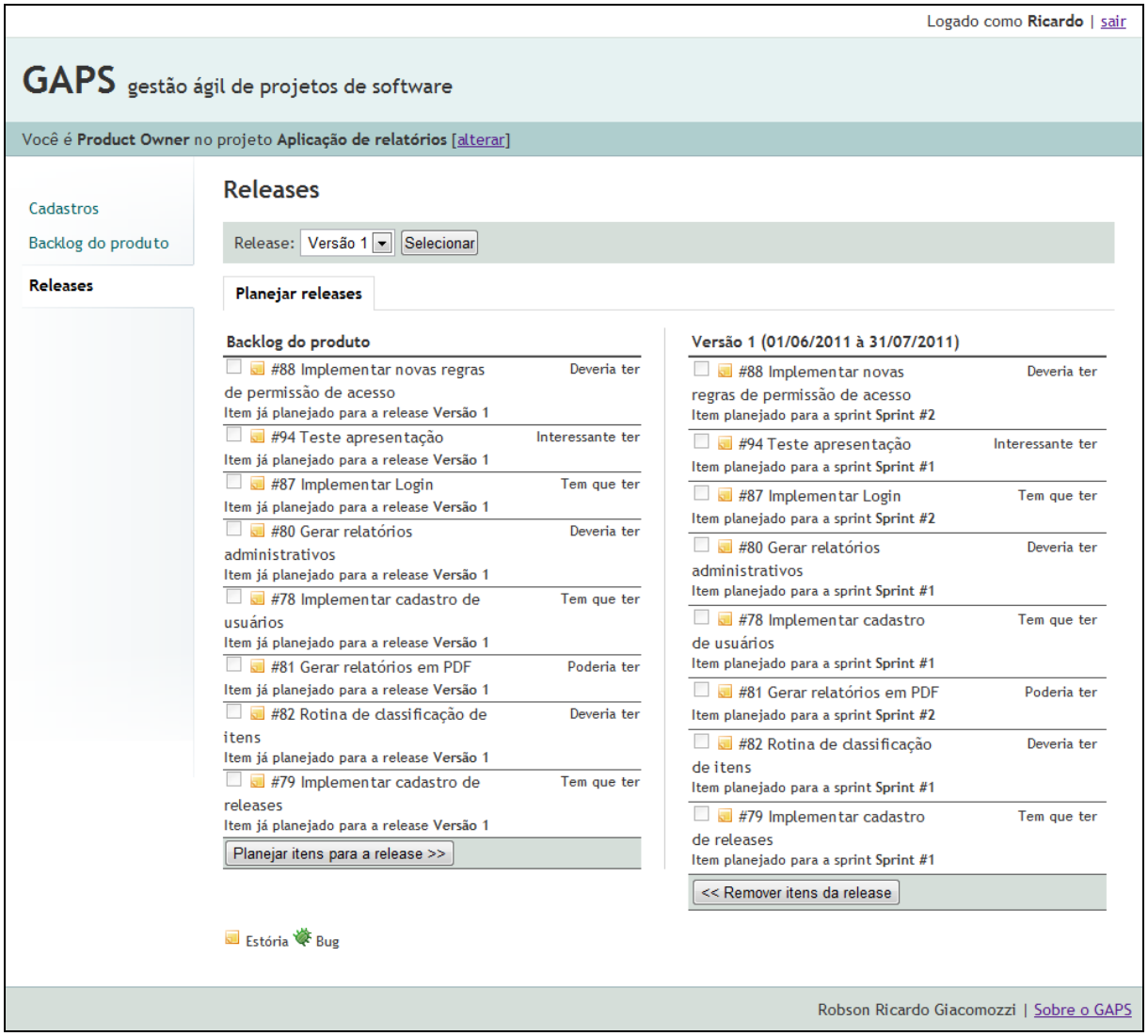


Figura 29: Tela de planejamento de uma *release*

A figura 30 demonstra a tela onde o PO terá a possibilidade editar e excluir uma

release, além de poder adicionar um novo cadastro, conforme mostra a figura 31.

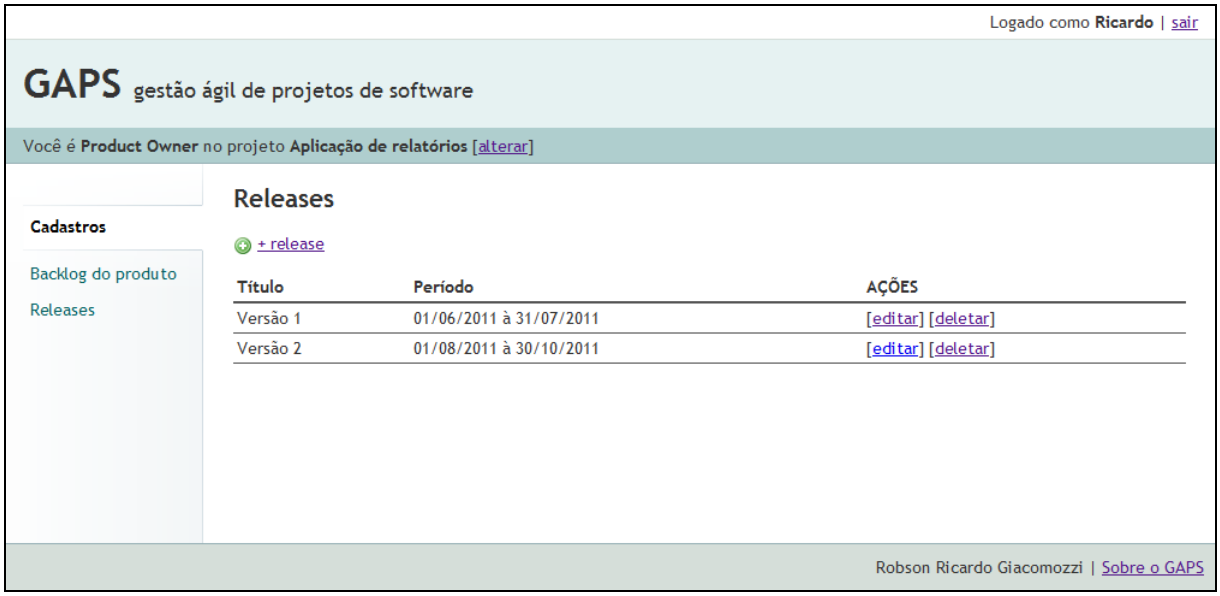


Figura 30: Tela das *releases* cadastradas

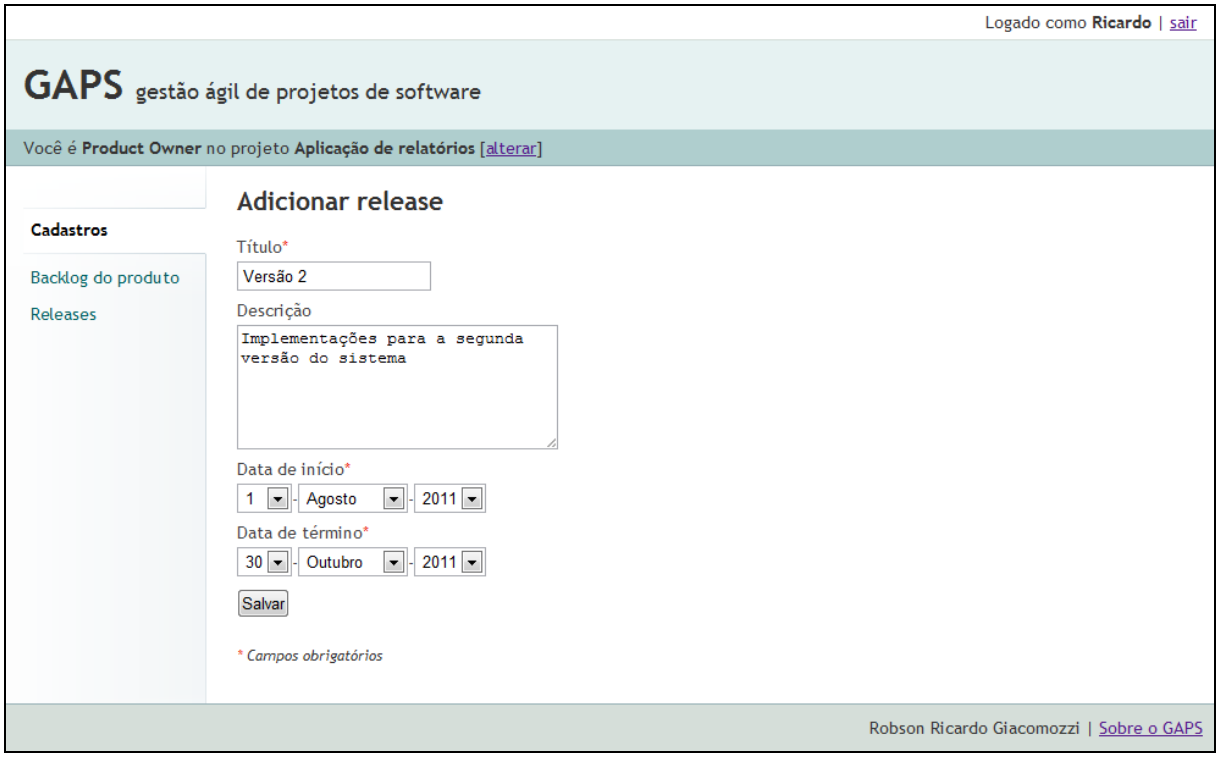


Figura 31: Tela de cadastro de uma *release*

3.3.2.4 Visão do *Scrum Master*

No item de *menu* “Cadastros”, o SM poderá gerenciar o cadastro das *sprints*, conforme mostra a figura 32. A figura 33 apresenta a tela de cadastro de uma nova *sprint*.

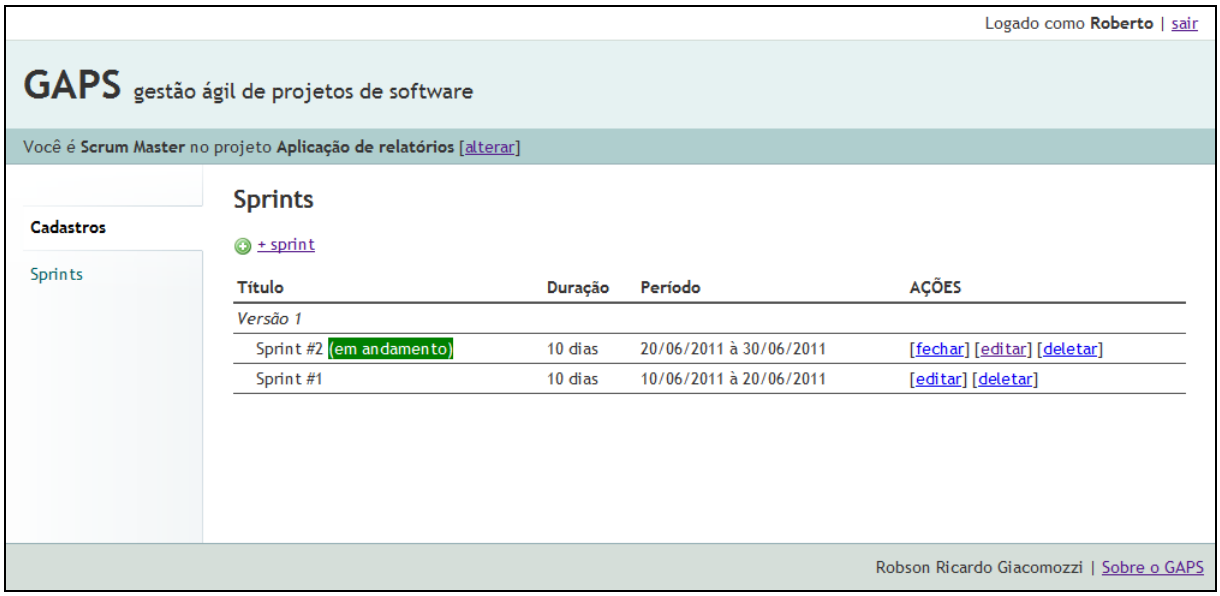


Figura 32: Tela das *sprints* cadastradas

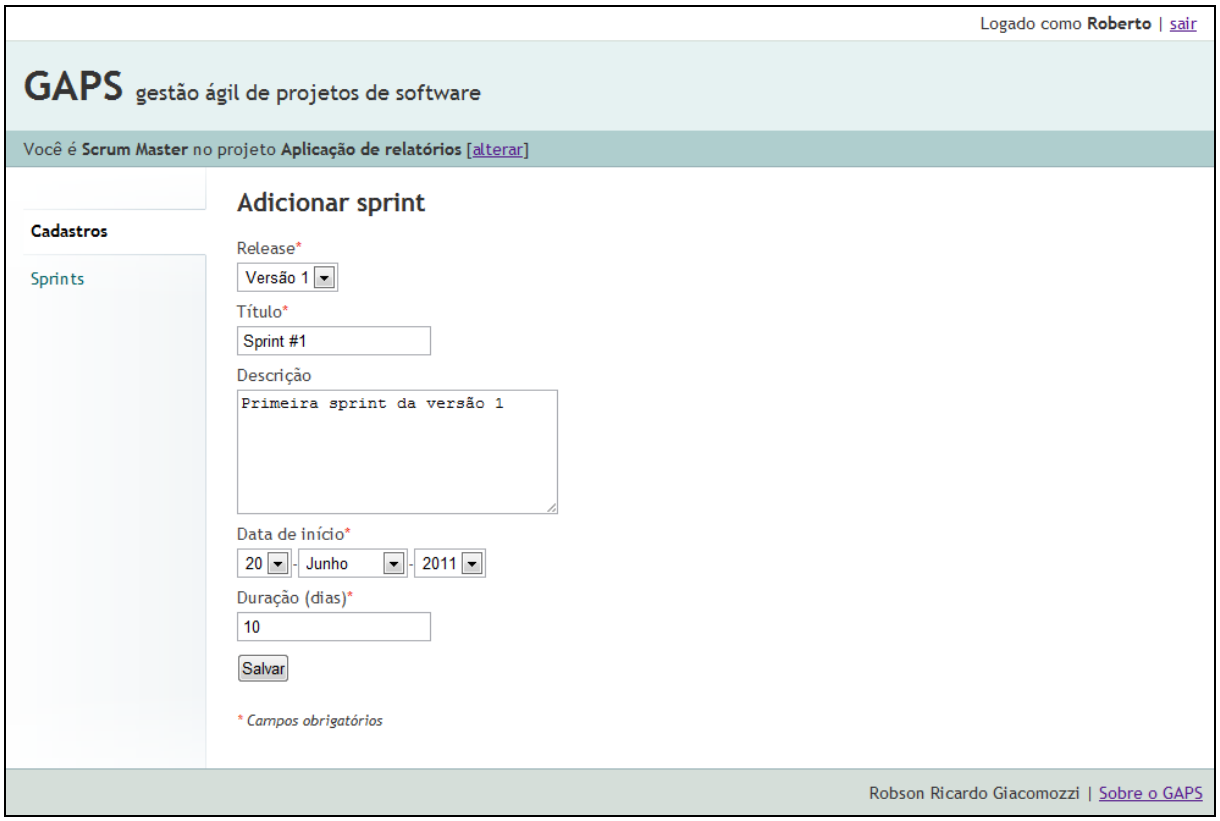


Figura 33: Tela de cadastro de uma *sprint*

No item de *menu* "*Sprints*", o SM terá a possibilidade de planejar o *Sprint Backlog*, acompanhar os impedimentos gerados, além do registro de cerimônias. A figura 34 apresenta a seleção da s*print*, que será a base para as demais opções de item de *menu*.

Figura 34: Seleção da *sprint*

Na aba "Planejar *sprint*", que também é a tela inicial do sistema para o SM, é possível definir o *Sprint Backlog*. A figura 35 mostra a tela de planejamento das *sprints*, onde o SM seleciona os itens da *release*, que ainda não foram planejados, clica no botão "Planejar itens para a *sprint*" e o sistema associa esses itens selecionados para a *sprint*. Para remover itens do *Sprint Backlog*, o SM faz o processo inverso: seleciona os itens da *sprint* e clicando no botão "Remover itens da *sprint*", onde os itens selecionados voltar para a *release*.

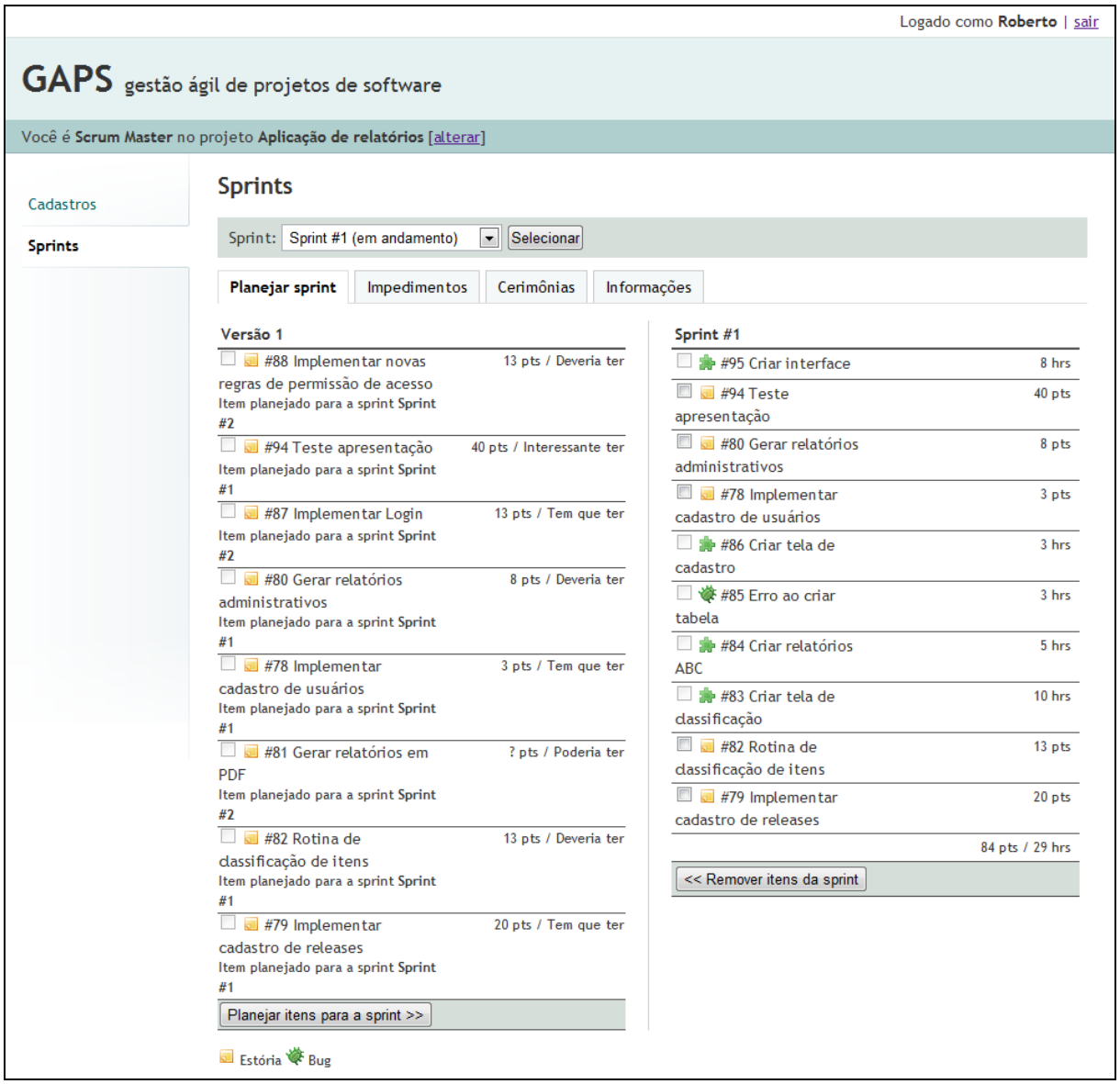


Figura 35: Tela de planejamento de uma *sprint*

A figura 36 apresenta a tela de planejamento do *Sprint Backlog*, onde nenhum item da

release foi planejado para uma outra s*print*.

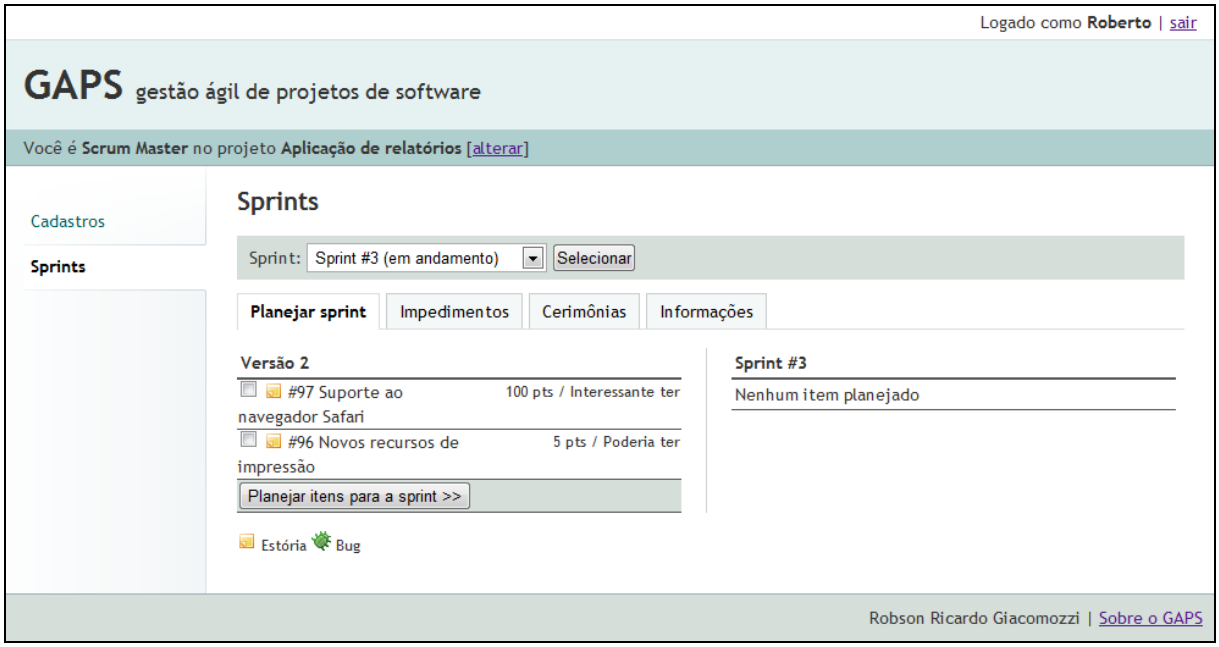


Figura 36: Tela de planejamento de uma *sprint*, onde nenhum item foi planejado

Acessando a aba "Impedimentos", o SM poderá acompanhar e gerenciar os impedimentos gerados pelo *Scrum Team*. O SM tem a responsabilidade de remover esses impedimentos, fazendo com que o ST esteja comprometido e focado apenas com o que foi planejado. A remoção ocorre quando o SM aciona a ação (link) "finalizar" no registro desejado, conforme mostra a figura 37.

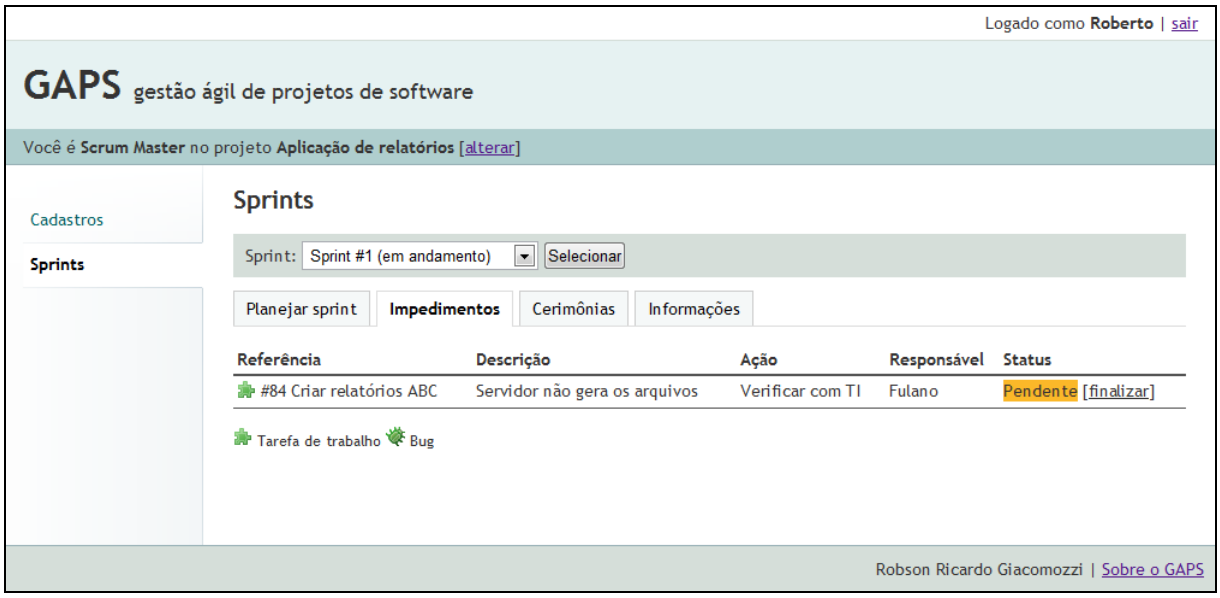


Figura 37: Tela de acompanhamento dos impedimentos de uma *sprint*

Na aba "Cerimônias", o SM pode consultar as cerimônias registradas durante a s*print* selecionada. Para as reuniões de *Sprint Planning*, *Sprint Review* e *Sprint Retrospective*, ao

criar uma *sprint*, o sistema gera automaticamente os registros para essas reuniões.

A figura 38 apresenta a tela de consulta das cerimônias cadastradas.

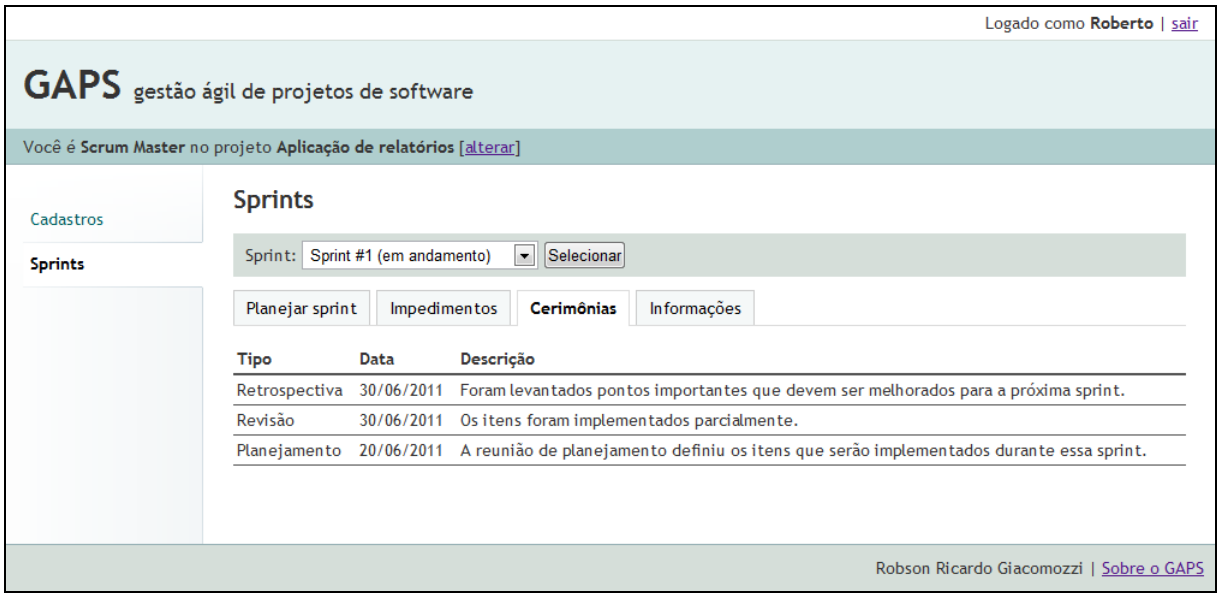


Figura 38: Tela das cerimônias cadastradas de uma *sprint*

A figura 39 apresenta a tela onde o SM poderá visualizar informações sobre a *sprint*, como a data de início, duração, total de pontos/esforço previstos e o time de trabalho.

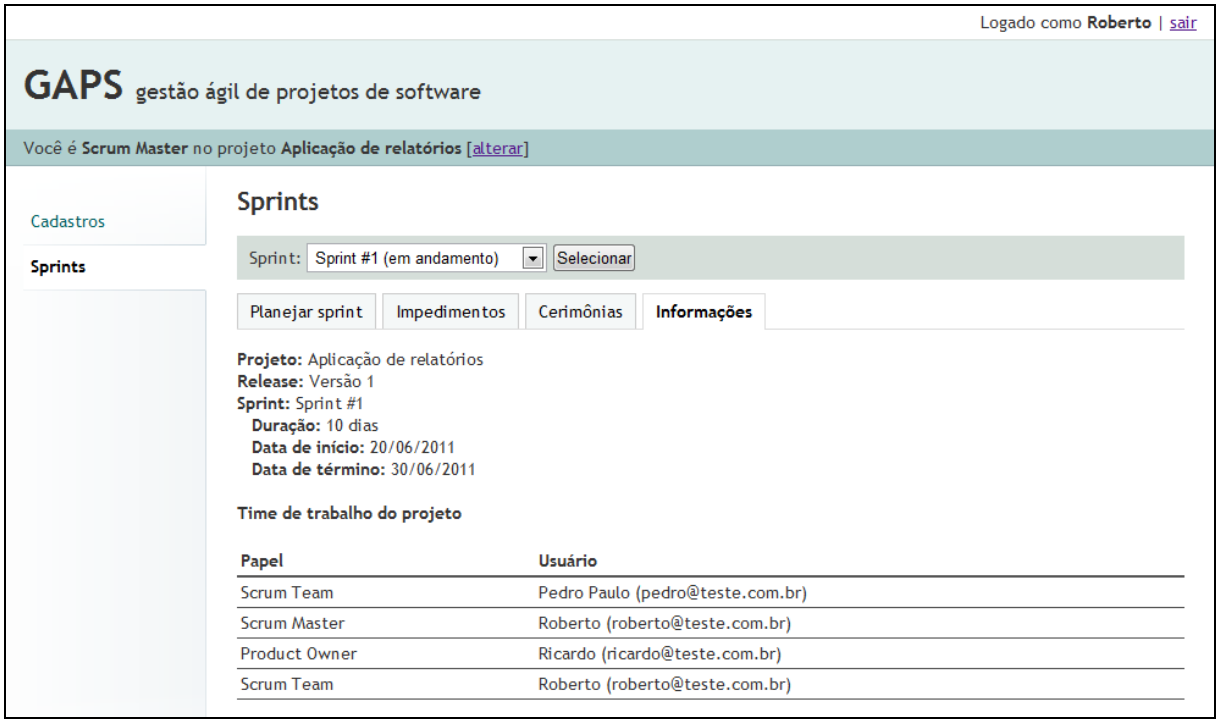


Figura 39: Tela informações de uma *sprint*

3.3.2.5 Visão do *Scrum Team*

A visão do ST é bem simplificada, focando as reais atividades e necessidades desse

tipo de usuário. Nessa visão, é apresentado através de abas, as opções para o quadro de tarefas, gráfico *burndown*, impedimentos e cerimônias da *sprint*. Na aba "Informações", são apresentados detalhes sobre a s*print*.

Na aba "Quadro de tarefas", o ST poderá consultar, cadastrar e remover tarefas de trabalho e *bugs* para uma estória de usuário. Caso necessário, poderá registrar impedimentos para estes itens. A figura 40 apresenta a tela do quadro de tarefas, conforme *sprint* selecionada, onde têm-se 4 colunas. A primeira apresenta as estórias de usuário e *bugs* que foram planejados para a *sprint*, a segunda, terceira e quarta coluna mostram o progresso das tarefas de trabalho e *bugs* da referida estória de usuário.

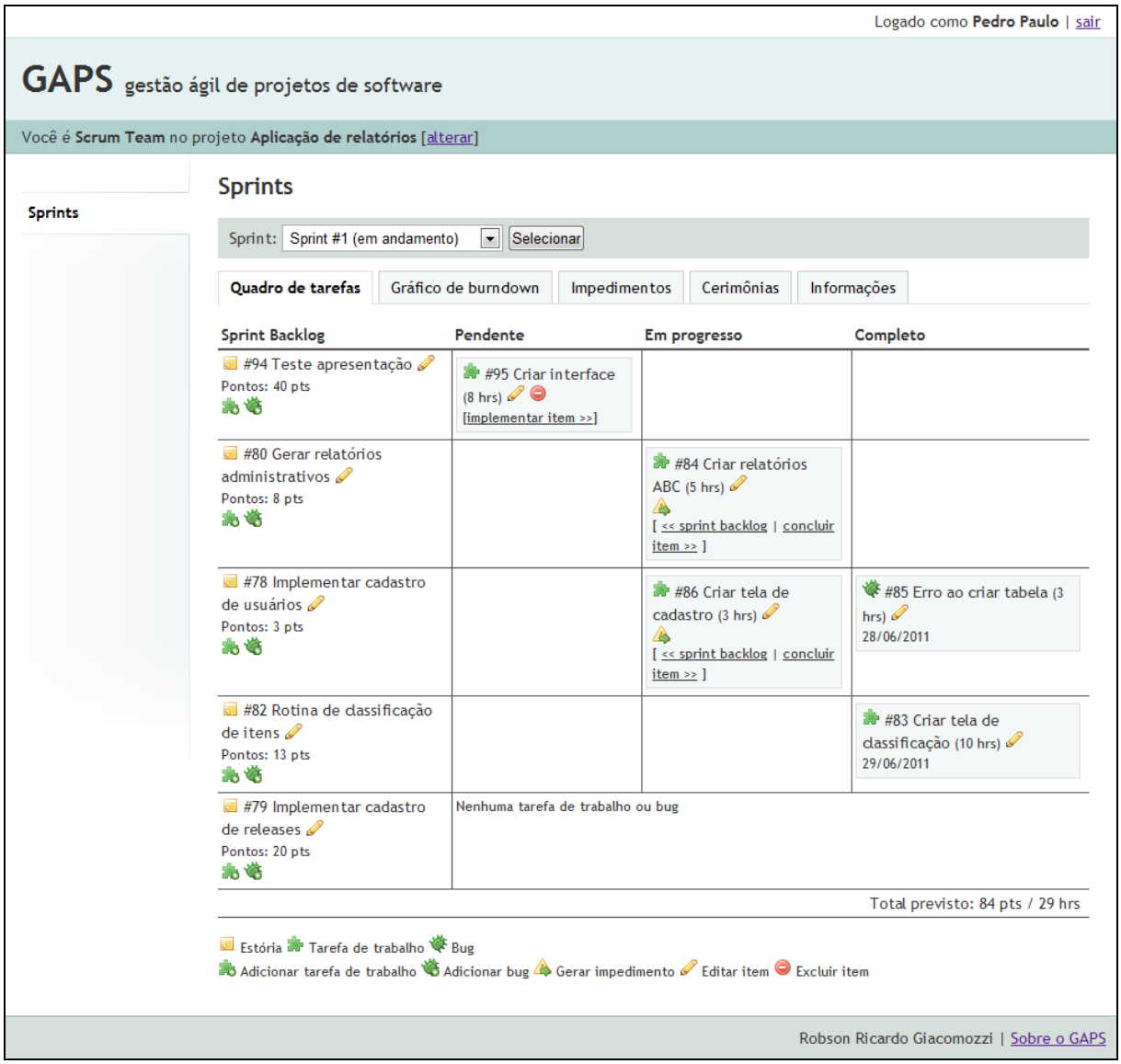


Figura 40: Tela do quadro de tarefas de uma *sprint*

Na aba "Gráfico de *burndown*", é possível acompanhar no primeiro gráfico, o andamento da *sprint* selecionada, onde são confrontadas as informações de pontos previstos

(itens planejados, conforme *Sprint Backlog*) com os pontos restantes, em uma linha do tempo (*timebox* da *sprint*). No segundo gráfico, as informações apresentadas são referentes ao esforço, em horas, previsto e restante. Nenhum dos gráficos disponíveis nesta aba, exibe o esforço realizado dos itens.

A figura 41 mostra a tela de gráficos de *burndown* com os dados da *sprint*. Na figura 42, é apresentado um trecho do código-fonte para geração dos gráficos.

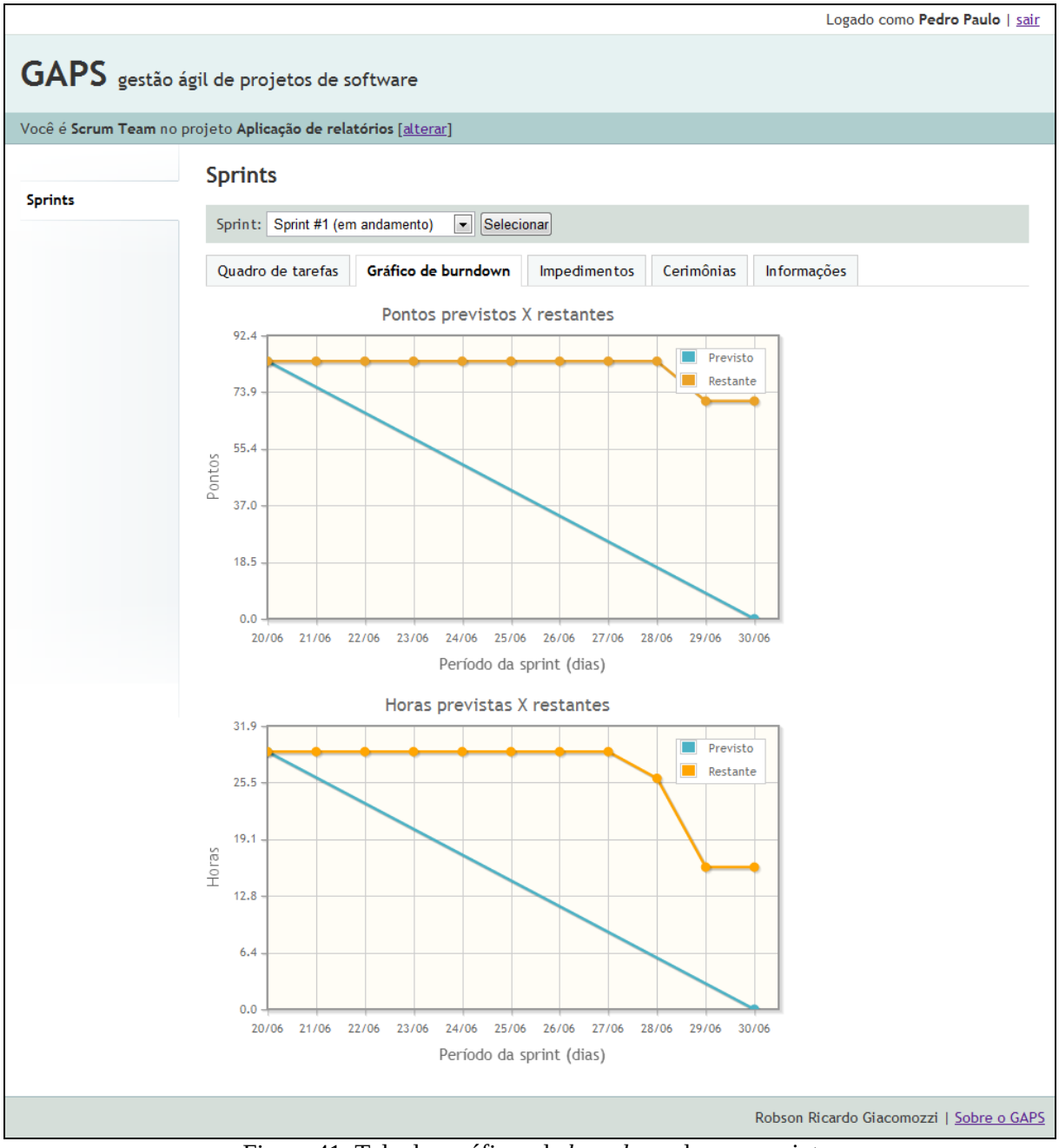


Figura 41: Tela dos gráficos de *burndown* de uma *sprint*

```
<?php echo $this->Html->script('jquery.flot'); ?>
<p class="txt_help">Pontos</p>
<div id="burndown_points" class="burndown" style="width:550px;height:350px;"></div>
<br />
<p class="txt_help">Esforço</p>
<div id="burndown_effort" class="burndown" style="width:550px;height:350px;"></div>
<script type="text/javascript">
    $(function () {
        //options
        var options = {
            series: {lines: { show: true }, points: { show: true }},
            grid: { hoverable: true }
        };
        //data points
        var data_points = [
            {
                label: '<?php echo $listDataChart['points']['estimated']['label']; ?>',
                data: <?php echo $listDataChart['points']['estimated']['data']; ?>
            },
            {
        //data effort
        var data_effort = [
            {
                label: '<?php echo $listDataChart['effort']['estimated']['label']; ?>',
                data: <?php echo $listDataChart['effort']['estimated']['data']; ?>
            },
            {
                label: '<?php echo $listDataChart['effort']['estimated_left']['label']; ?>',
                data: <?php echo $listDataChart['effort']['estimated_left']['data']; ?>
            }
        ];

        //generate burndown chart
        var plot_a = $.plot($("#burndown_points"), data_points, options);
        var plot_b = $.plot($("#burndown_effort"), data_effort, options);

        //show label
        function showTooltip(x, y, contents) {
            $('<div id="tooltip">' + contents + '</div>').css( {
                position: 'absolute',
                display: 'none',
                top: y + 5,
                left: x + 5,
                border: '1px solid #fdd',
                padding: '2px',
                'background-color': '#fee',
                opacity: 0.80
            }).appendTo("body").fadeIn(200);
        }

        var previousPoint = null;
        $(".burndown").bind("plothover", function (event, pos, item) {
            if (item) {
                if (previousPoint != item.dataIndex) {
                    previousPoint = item.dataIndex;

                    $("#tooltip").remove();
                    showTooltip(item.pageX, item.pageY, item.datapoint[1] + " horas");
                }
            } else {
                $("#tooltip").remove();
                previousPoint = null;
            }
        });
    });
</script>
```

Figura 42: Trecho do código-fonte usado na geração dos gráficos

Acessando a aba “Impedimentos”, o ST poderá acompanhar através de uma listagem, os impedimentos gerados pelo próprio *Scrum Team*, durante a execução de uma s*print*.

Na aba “Cerimônias”, o ST poderá consultar as cerimônias cadastradas e registrar as reuniões diárias (*Daily Scrum Meeting*), conforme ocorrerem durante a execução de uma s*print*. Para as cerimônias *Sprint Planning*, *Sprint Review* e *Sprint Retrospective*, que são criadas automaticamente pelo sistema no cadastro de uma *sprint*, ficarão disponíveis apenas a consulta e edição desses registros. A figura 43 apresenta o registro da reunião diária.

Figura 43: Tela de registro das reuniões diárias ocorridas durante uma *sprint*

Na aba “Informações”, o ST poderá consultar detalhes sobre a s*print* selecionada, como a data de início, duração, total de pontos/esforço previstos e o time de trabalho.

3.4 RESULTADOS E DISCUSSÃO

O desenvolvimento deste trabalho permitiu criar um sistema *web* capaz de atender aos elementos do *Scrum*, fornecendo a estrutura de planejamento, desenvolvimento e acompanhamento, além da finalização do ciclo proposto na metodologia.

Observando os trabalhos correlatos apresentados na seção 2.4, juntamente com a pesquisa inicial feita para encontrá-los, o quadro 3 apresenta a comparação das principais funcionalidades e características do sistema desenvolvido com as dos trabalhos analisados.

Funcionalidade / Característica	GAPS	*Scrum Project*	DotProject	Pronto!	MELLO (2010)
Visão do sistema para cada tipo de papel do *Scrum*	Sim	Sim	Sim	Sim	Parcial
Plataforma *web*	Sim	Sim	Sim	Sim	Sim
Língua portuguesa	Sim	Sim	Sim	Sim	Sim
Criação do *Product Backlog*	Sim	Sim	Sim	Sim	Sim
Interface amigável[13] para priorização do *Product Backlog*	Sim	Não	Não	Sim	Não
Planejamento de *releases*	Sim	Não	Não	Não	Não
Planejamento de *sprints*	Sim	Sim	Sim	Sim	Sim
Criação de times de trabalho	Sim	Sim	Não	Parcial	Sim
Registro de impedimentos	Sim	Não	Não	Não	Não
Registro de cerimônias	Sim	Sim	Parcial	Parcial	Sim
Relatórios estatísticos	Não	Sim	Sim	Não	Sim
Gráfico de *burndown*	Sim	Não	Não	Sim	Não

Quadro 3: Comparativo entre o sistema desenvolvido e os trabalhos correlatos

A funcionalidade de planejamento de *releases*, disponível apenas no GAPS, é muito importante para o planejamento, em um período de tempo mais longo, dos requisitos que serão implementados pelo ST, onde o PO define o ROI sobre esses itens.

Durante uma *sprint*, pode-se identificar impedimentos para os requisitos que estão sendo implementados. O ST registra os impedimentos e o SM gerencia-os, podendo finalizá-los ou não. Para o ST, fica disponível apenas o registro e consulta desses impedimentos.

Para validar a aderência do sistema desenvolvido com as expectativas do *Scrum*, foram entrevistadas algumas pessoas ligadas a implantação do *Scrum* na empresa, onde puderam resumir a abrangência do sistema. A avaliação foi realizada informalmente, observando aspectos da forma como se trabalha na empresa e a maneira como o sistema se comporta. Muitas situações e demandas do *Scrum*, não são aplicadas efetivamente na empresa, como por exemplo, as reuniões de revisão e retrospectiva, *Sprint Review* e *Sprint Retrospective*, respectivamente. De forma geral, os entrevistados se mostraram satisfeitos com o sistema apresentado, comentando a facilidade, simplicidade das telas além da grande utilidade do mesmo no dia-a-dia da empresa.

[13] Para este trabalho, entende-se como "amigável" as interfaces que o usuário se sinta confortável e encorajado de utilizar.

4 CONCLUSÕES

A crescente utilização dos métodos ágeis, em especial do *Scrum*, proporcionam a adaptação e criação de novas ferramentas e sistemas para gerir projetos de *software*, atuando nos objetivos e princípios da metodologia utilizada.

Neste trabalho, se propôs o desenvolvimento de um sistema *web* que fornece um conjunto de funcionalidades capazes de atentar a estrutura básica do *Scrum*. O sistema disponibiliza, aos papéis do *Scrum*, o ferramental necessário para aplicação da metodologia, como a gestão dos itens do *Product Backlog*, planejamento das *sprints*, registro de impedimentos e acompanhamento das iterações (*sprints*) através do gráfico de *burndown*.

O sistema desenvolvido tem funcionalidades diferenciadas para cada tipo de usuário (papéis do *Scrum*), permitindo restrições de acesso, aumentando a segurança das informações e disponibilizando funcionalidades específicas para cada papel.

Diante das funcionalidades implementadas, o sistema desenvolvido conseguiu atender seu objetivo principal e específico. Através do desenvolvimento de uma *interface* simples e intuitiva, empresas que adotarem o *Scrum* para gerenciamento de seus projetos, poderão utilizar uma ferramenta adequada e que comporte os procedimentos, técnicas e princípios da metodologia.

Com relação às tecnologias empregadas, o CakePHP permitiu agilizar a implementação do sistema, fornecendo a arquitetura MVC, URLs personalizadas, componentes de segurança e sessão, além da rápida curva de aprendizagem.

Pode-se dizer que a conclusão deste trabalho atingiu objetivos pessoais, transformando a experiência adquirida com os métodos ágeis em uma ferramenta que auxilie e contribua para a adoção do *Scrum* nas empresas. Na implantação de uma metodologia ágil, não é obrigatório a utilização de ferramentas para gestão de projetos. Entretanto, o sistema desenvolvido é capaz de alinhar, de forma simples e fácil, as suas funcionalidades com as expectativas e necessidades do *Scrum*.

4.1 EXTENSÕES

Embora a implementação do sistema desenvolvido neste trabalho ofereça as funcionalidades básicas do *Scrum*, outras implementações poderiam ser desenvolvidas. Dentre elas destacam-se:

a) integração com ferramentas de *bug tracker*[14], como por exemplo MantisBT[15];
b) implementação do quadro de tarefas conforme o *Sprint Backlog*, para o acompanhamento visual da evolução da *sprint*, usando quadros de *Kanban*;
c) usar técnicas para a retrospectiva da *sprint*, como por exemplo Seis Chapéus[16];
d) controle de horas realizadas das tarefas de trabalhos e *bugs*;
e) relatórios gerenciais mais aprimorados, como por exemplo: tarefas por usuários, estórias de usuários e *bugs* relacionados, relação entre pontos previstos e horas previstas.

[14] *Bug tracker* é uma ferramenta para acompanhamento, controle e gestão de erros.

[15] Mais informações sobre o MantisBT podem ser acessadas em http://www.mantisbt.org/.

[16] O método dos Seis Chapéus foi criado por Edward de Bono como uma ferramenta para tornar discussões em grupo mais eficazes. A ideia consiste em convergir o pensamento do grupo alinhando a visão e o humor das pessoas sob um mesmo aspecto (BASTOS, 2009).

REFERÊNCIAS BIBLIOGRÁFICAS

ABRAHAMSSON, Pekka et al. **New directions on agile methods: a comparative analysis**. Portland: IEEE Computer Society, 2003. Disponível em: <http://www.informatik.uni-trier.de/~ley/db/conf/icse/icse2003.html#abrahamssonWSR03>. Acesso em: 29 abr. 2011.

ADAPTWORKS. **Scrum**. [S.l.], 2011. Disponível em: <http://www.adaptworks.com.br/scrum>. Acesso em: 30 abr. 2011.

AGILE MANIFESTO. **Manifesto for Agile Software Development**. [S.l.], 2001. Disponível em: <http://agilemanifesto.org/iso/ptbr/>. Acesso em: 06 abr. 2011.

ALBUQUERQUE, Nikolai. **Entendendo o gerenciamento ágil de projetos com Scrum**. [S.1.], 2007. Disponível em <http://www.fumsoft.softex.br/ccomp/Entendendo_gerenciamento_agil_de_projetos_com_scrum.pdf>. Acesso em: 11 mar. 2011.

BASSI, Dairton. **Experiências com desenvolvimento ágil**. 2008. 170f. Dissertação (Mestrado em Ciências da Computação) – Instituto de Matemática e Estatística da Universidade de São Paulo, São Paulo, 2008.

BASTOS, Henrique. **Reuniões produtivas e objetivas com a técnica dos Seis Chapéus**. [S.1.], 2009. Disponível em: <http://henriquebastos.net/2009/12/23/reunioes-produtivas-e-objetivas-com-a-tecnica-dos-seis-chapeus/>. Acesso em: 14 jun. 2011.

BECK, Kent; FOWLER, M. **Extreme programming applied**. Boston: Addison-Wesley, 2001.

BROD, Cesar. **Uma introdução ao Scrum**. [S.l.], 2009. Disponível em: <www.centrosoftware.com.br/wp-content/uploads/2009/09/IntroducaoScrum.pdf>. Acesso em: 14 mar. 2011.

CAKEPHP. **Framework PHP para desenvolvimento rápido**. [S.l.], 2011. Disponível em: <http://www.cakephp.com.br>. Acesso em: 30 abr. 2011.

CASTRO, Flavio Steffens de. **Retorno sobre investimento: você sabe o que é?**. [S.l.], 2010. Disponível em: <http://www.agileway.com.br/2010/01/06/retorno-sobre-investimento-voce-sabe-o-que-e/>. Acesso em: 18 jun. 2011.

DÓRIA, Hugo. **Modelo de Desenvolvimento Ágil SCRUM**. [S.l.], 2009. Disponível em: <http://www.devin.com.br/modelo-scrum/>. Acesso em: 29 abr. 2011.

ECLIPSE. **PHP Development Tools Project**. [S.1], 2011. Disponível em: <http://www.eclipse.org/pdt/>. Acesso em: 10 jun. 2011.

FERREIRA, Renata Bastos; LIMA, Francisco de Paula Antunes. **Metodologias ágeis: Um Novo Paradigma de desenvolvimento de software**. [S.l.], 2006. Disponível em: <www.cos.ufrj.br/~handrade/woses/.../10-Artigo10WOSES-2006.pdf>. Acesso em: 09 abr. 2011.

GOMES, André Faria; FAIAS JUNIOR, Luiz dos Santos. **Pronto! - Software para gestão de projetos ágeis**. 2009. 66f. Trabalho de conclusão de curso (Bacharelado em Sistemas de Informação) - Faculdade de Informática e Administração Paulista, São Paulo, 2009.

GRAY, Clifford F; LARSON, Erik W. **Gerenciamento de projetos: o processo gerencial**. 4 ed. São Paulo: McGraw-Hill, 2009.

IMPROVEIT. **Scrum**. [S.l.], 2009. Disponível em: <http://improveit.com.br/scrum>. Acesso em: 09 abr. 2011.

KLUGE, Monike Roberta. **Scrum Project: Ferramenta de apoio ao gerenciamento de projetos de software baseada nos princípios da metodologia ágil Scrum**. 2009. 72f. Trabalho de conclusão de curso (Graduação em Ciências da Computação) - Centro de Ciências Tecnológicas da Terra e do Mar, Universidade do Vale do Itajaí, Itajaí, 2009.

LEITE, Jair C. **Engenharia de Software**. [S.l.], 2007. Disponível em: <http://engenhariadesoftware.blogspot.com>. Acesso em: 09 abr. 2011.

LUDVIG, Diogo; REINERT, Jonatas Davson. **Estudo do uso de metodologias ágeis no desenvolvimento de uma aplicação de governo eletrônico**. 2007. 96f. Trabalho de conclusão de curso (Graduação em Sistemas de Informação) – Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, 2007.

MARÇAL, Ana Sofia. **Entendendo Scrum para Gerenciar Projetos de Forma Ágil**. [S.1.], 2007. Disponível em: <http://www.cesar.org.br/files/file/SCRUM_MundoPM-Abril-Maio-2007.pdf >. Acesso em: 10 abr. 2011.

MARTINS, Jose Carlos Cordeiro. **Técnicas para gerenciamento de projetos de software**. Rio de Janeiro: Brasport, 2007.

MELLO, Vanessa. **Ferramenta web para gerenciamento de projetos de software baseado no Scrum**. 2010. 65f. Trabalho de conclusão de curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais - Curso de Ciências da Computação, Fundação Universidade Regional de Blumenau, Blumenau, 2010.

MENEZES, Lucas. **Terminando a tempo!**. [S.l.], 2011. Disponível em: <http://grupohaw.com.br/2011/01/16/terminando-a-tempo/>. Acesso em: 01 maio 2011.

NETO, Oscar Nogueira de Souza. **Análise comparativa das metodologias de desenvolvimento de softwares tradicionais e ágeis**. Trabalho de conclusão de curso. Universidade da Amazônia, Belém, 2004.

ORTH, Afonso Inácio; PRIKLADNICKI, Rafael. **Planejamento e gerência de projetos**. Porto Alegre: EDIPUCRS, 2009.

PEREIRA, Jhony Alceu. **Ambiente web para gerenciamento de processo de software baseado no Scrum**. 2005. 70f. Trabalho de conclusão de curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais - Curso de Ciências da Computação, Fundação Universidade Regional de Blumenau, Blumenau, 2005.

PRESSMAN, Roger S. **Engenharia de software**. 6.ed. São Paulo: MCGrall - Hill, 2006.

SCHWABER, Ken. **Agile Project Management with Scrum**. Redmond, Wash: Microsoft Press, 2004.

SILVEIRA, Antonio Carlos. **O papel do Product Owner e priorização do Product Backlog**. [S.1.], 2008. Disponível em <http://www.acarlos.com.br/blog/2008/02/papel-do-product-owner-e-priorizacao-do-product-backlog/>. Acesso em: 11 jun. 2011.

SOMMERVILLE, Ian. **Engenharia de software**. 8.ed. São Paulo: Pearson Addison Wesley, 2007.

STANDISHGROUP. **Chaos Report: The Standish Group Internacional, Inc**. [S.l.], 2009. Disponível em: <http://secure.standishgroup.com>. Acesso em: 30 abr. 2011.

TOMÁS, Mário Rui. **Métodos ágeis: características, pontos fortes e fracos e possibilidades de aplicação**. [S.l.], 2009. Disponível em: <http://run.unl.pt/handle/10362/2003>. Acesso em: 09 abr. 2011.

APÊNDICE A – Detalhamento dos casos de uso

Nos quadros 4, 5, 6, 7, 8 e 9 são apresentados os detalhes dos principais casos de uso previstos nos diagramas da seção 3.2.2.

Caso de uso: UC004 – Manter *Product Backlog.*
Objetivo: Permite incluir estórias de usuário e/ou *bugs* no *Product Backlog*. **Ator:** *Product Owner.* **Pré-condições:** Usuário autenticado no sistema, ter ao menos um projeto cadastrado, o usuário deve ser do tipo *Product Owner* no projeto. **Pós-condição:** Estória de usuário ou *bug* cadastrado.
Cenário principal 1. Usuário acessa o item "*Backlog*" do *menu* "*Backlog* do produto"; 2. Sistema carrega tela de itens cadastrados; 3. Sistema apresenta links de opções; 4. Usuário opta por adicionar uma estória de usuário ou *bug*; 5. Sistema persiste os dados na base de dados; 6. Sistema apresenta a mensagem "Item gravado com sucesso".
Cenário alternativo – Estória No passo 4, o usuário pode optar por adicionar uma estória de usuário: 4.1. Sistema carrega a tela de cadastro de estórias de usuário; 4.2. Usuário preenche os campos "Título, Descrição, Valor de negócio e Pontos"; 4.3. Volta ao cenário principal.
Cenário alternativo – Bug No passo 4, o usuário pode optar por adicionar um *bug*: 4.1. Sistema carrega a tela de cadastro de *bugs*; 4.2. Usuário preenche os campos "Título, Descrição, Estória de referência, Valor de negócio e Esforço (horas)"; 4.3. Volta ao cenário principal.
Cenário de exceção No passo 4.2, caso o usuário não preencher os campos obrigatórios, o sistema exibe o nome do campo que falta informar.

Quadro 4: Descrição do caso de uso "Manter *Product Backlog*"

Caso de uso: UC005 - Priorizar *Product Backlog*
Objetivo: Permite a priorização dos itens cadastrados no *Product Backlog*. **Ator:** *Product Owner.* **Pré-condição:** Ter ao menos dois itens cadastrados no *Product Backlog*. **Pós-condição:** Itens do *Product Backlog* priorizados.
Cenário principal 1. Usuário acessa o item "Priorizar itens" do *menu* "*Backlog* do produto"; 2. Sistema carrega tela de priorização de itens; 3. Usuário prioriza os itens, arrastando-os de forma a ordená-los conforme necessidade; 4. Sistema persiste os dados na base de dados.

Quadro 5: Descrição do caso de uso "Priorizar *Product Backlog*"

Caso de uso: UC007 – Planejar *releases.*
Objetivo: Permite planejar estórias de usuários e/ou *bugs* para uma *release*, com base nos itens do *Product Backlog*, devidamente priorizados. **Ator:** *Product Owner.* **Pré-condições:** Ter ao menos um projeto cadastrado e ter ao menos um item cadastrado no *Product Backlog*. **Pós-condição:** *Release* planejada.
Cenário principal 1. Usuário acessa o item "Planejar *release*" do *menu "Releases";* 2. Sistema carrega tela de planejamento da *release;* 3. Sistema apresenta as opções para o usuário; 4. Usuário opta por adicionar ou retirar itens da *release*; 5. Sistema persiste os dados no banco de dados; 6. Sistema apresenta a mensagem "Os itens selecionados foram planejados na *release* com sucesso".
Cenário alternativo – Adicionar itens No passo 4, o usuário pode optar por adicionar itens do *Product Backlog* ao *release*: 4.1. Usuário seleciona os itens ainda não planejados do *Product Backlog;* 4.2. Usuário clica no botão "Planejar itens para a *release*"*;* 4.3. Volta ao cenário principal.
Cenário alternativo – Retirar itens No passo 4, o usuário pode optar por retirar itens da *release*: 4.1. Usuário seleciona os itens já planejados na *release*; 4.2. Usuário clica no botão "Remover itens da *release*"; 4.3. Volta ao cenário principal.
Cenário de exceção No passo 4.1, caso o usuário não selecionar nenhum item e clicar nos botões de ação, o sistema apresenta a mensagem "Nenhum item foi selecionado".

Quadro 6: Descrição do caso de uso "Planejar *releases*"

Caso de uso: UC009 – Planejar *sprints.*
Objetivo: Permite planejar estórias de usuários e/ou *bugs* para uma s*print*, com base nos itens que foram planejados na *release* da referida *sprint*. **Ator:** *Scrum Master.* **Pré-condições:** Ter ao menos um projeto cadastrado e ter ao menos um item planejado para a *release*. **Pós-condição:** *Sprint* planejada.
Cenário principal 1. Usuário acessa o item "Planejar *sprint*" do *menu "Sprints";* 2. Sistema carrega tela de planejamento da *sprint;* 3. Sistema apresenta as opções para o usuário; 4. Usuário opta por adicionar ou retirar itens da *sprint*; 5. Sistema persiste os dados no banco de dados; 6. Sistema apresenta a mensagem "Os itens selecionados foram planejados com sucesso".
Cenário alternativo – Adicionar itens No passo 4, o usuário pode optar por adicionar itens da *release* à *sprint*: 4.1. Usuário seleciona os itens ainda não planejados da *release;* 4.2. Usuário clica no botão "Planejar itens para a *sprint*"*;* 4.3. Volta ao cenário principal.

Cenário alternativo – Retirar itens No passo 4, o usuário pode optar por remover os itens da *sprint*: 4.1. Usuário seleciona os itens já planejados na *sprint*; 4.2. Usuário clica no botão "Remover itens da sprint"; 4.3. Volta ao cenário principal.
Cenário de exceção No passo 4.1, caso o usuário não selecionar nenhum item e clicar nos botões de ação, o sistema apresenta a mensagem "Nenhum item foi selecionado".

Quadro 7: Descrição do caso de uso "Planejar *sprints*"

Caso de uso: UC012 - Fechar *sprint.*
Objetivo: Permite fechar uma *sprint*, bloqueando as operações de planejamento e cadastro de novos itens (estórias de usuário, tarefas de trabalho e/ou *bugs*) para o referido *Sprint Backlog*. **Ator:** *Scrum Master.* **Pré-condição:** Ter ao menos uma *sprint* não encerrada. **Pós-condição:** *Sprint* encerrada.
Cenário principal 1. Usuário acessa o *menu* de "Cadastros"; 2. Sistema carrega tela de *sprints* cadastradas; 3. Usuário seleciona a opção "Fechar *sprint*", no registro desejado; 4. Sistema salva as informações na base de dados.

Quadro 8: Descrição do caso de uso "Fechar *sprint*"

Caso de uso: UC013 - Cadastrar tarefa de trabalho.
Objetivo: Permite incluir tarefas de trabalho para as estórias de usuário planejadas no *Sprint Backlog*. **Ator:** *Scrum Team.* **Pré-condição:** Ter ao menos uma estória de usuário planejada no *Sprint Backlog*. **Pós-condição:** Tarefa de trabalhou ou *bug* cadastrado.
Cenário principal 1. Usuário acessa o item "Quadro de tarefas" do *menu* de "*Sprints*"; 2. Sistema carrega tela das tarefas planejadas para o *Sprint Backlog*; 3. Usuário acessa o link "+ tarefa" para adicionar uma tarefa de trabalho para a estória de usuário desejada; 4. Sistema carrega a tela de cadastro de uma tarefa de trabalho; 5. Usuário preenche os dados "Título, Descrição, Estória de referência e Esforço (horas)"; 6. Sistema seta o status do item cadastrado como "Pendente"; 7. Sistema salva as informações no banco de dados.
Cenário de exceção No passo 5, caso o usuário não preencher os campos obrigatórios, o sistema exibe o nome do campo que falta informar.

Quadro 9: Descrição do caso de uso "Cadastrar tarefa de trabalho"

APÊNDICE B – Dicionário de dados

Este apêndice apresenta a descrição detalhada das entidades da modelagem de banco de dados previstas no diagrama da seção 3.2.3. Os tipos de dados de cada campo são descritos a seguir:

a) *varchar*: tipo de campo para armazenamento de *strings* de caracteres e seu tamanho é definido em *bytes* com largura variável, os valores entre parênteses definem o comprimento máximo em *bytes* de caracteres;
b) *int*: tipo de campo para armazenamento de números inteiros;
c) *datetime*: tipo de campo para armazenamento de datas e horas;
d) *text*: tipo de campo para armazenamento de grandes *strings* ou binários.

Os campos "*created*" e "*modified*", disponível em todas as entidades (tabelas) do sistema, é alimentado pelo *framework* CakePHP. Neles são armazenados as datas de criação e edição de um registro.

No quadro 10 pode-se observar o dicionário de dados da tabela "*projects*".

Tabela: *projects*			
Campo	**Tipo**	**Descrição**	**Observação**
id	int	Número sequencial do projeto.	Chave primária.
project_title	varchar(225)	Título do projeto.	
created	datetime	Data de criação do registro.	
modified	datetime	Data de alteração do registro.	

Quadro 10: Dicionário de dados da tabela "*projects*"

No quadro 11 pode-se observar o dicionário de dados da tabela "*releases*".

Tabela: *releases*			
Campo	**Tipo**	**Descrição**	**Observação**
id	*int*	Id sequencial da *release.*	Chave primária.
project_id	*int*	Id do projeto vinculado.	Chave estrangeira.
release_title	*varchar(225)*	Título da *release.*	
release_description	*text*	Descrição da *release.*	
release_start_date	*datetime*	Data de início da *release.*	
release_end_date	*datetime*	Data de término da *release.*	
created	*datetime*	Data de criação do registro.	
modified	*datetime*	Data de alteração do registro.	

Quadro 11: Dicionário de dados da tabela "*releases*"

No quadro 12 pode-se observar o dicionário de dados da tabela "*teams*".

Tabela: *teams*			
Campo	**Tipo**	**Descrição**	**Observação**

id	*int*	Id sequencial do time.	Chave primária.
user_id	*int*	Id do usuário vinculado.	Chave estrangeira.
project _id	*int*	Id do projeto vinculado.	Chave estrangeira.
member_type	*varchar(3)*	Tipo de membro: “po”, *Product Owner*; “sm”, *Scrum Master*; ou “st”, *Scrum Team*.	
created	*datetime*	Data de criação do registro.	
modified	*datetime*	Data de alteração do registro.	

Quadro 12: Dicionário de dados da tabela "*teams*"

No quadro 13 pode-se observar o dicionário de dados da tabela “*users*”.

Tabela: *users*			
Campo	**Tipo**	**Descrição**	**Observação**
id	*int*	Id sequencial do usuário.	Chave primária.
user_name	*varchar(60)*	Nome do usuário.	
user_email	*varchar(60)*	E-mail do usuário.	
user_login	*varchar(20)*	*Login* do usuário.	
user_pass	*varchar(40)*	Senha criptografada do usuário.	
user_admin	*int(1)*	*Flag* indicando se o usuário é administrador: “1”, sim; ou “0”, não.	
user_active	*int(1)*	*Flag* indicando se o usuário está ativo: “1”, sim; ou “0”, não.	
created	*datetime*	Data de criação do registro.	
modified	*datetime*	Data de alteração do registro.	

Quadro 13: Dicionário de dados da tabela "*users*"

No quadro 14 pode-se observar o dicionário de dados da tabela “*tickets*”.

Tabela: *tickets*			
Campo	**Tipo**	**Descrição**	**Observação**
id	*int*	Id sequencial do *ticket.*	Chave primária.
project_id	*int*	Id do projeto vinculado.	Chave estrangeira.
release_id	*int*	Id da *release* vinculada.	Chave estrangeira não obrigatória.
sprint_id	*int*	Id da *sprint* vinculada.	Chave estrangeira não obrigatória.
ticket_ref	*int*	Id do *ticket* referenciado.	Chave estrangeira não obrigatória.
ticket_title	*varchar(255)*	Título do *ticket*.	
ticket_type	*varchar(15)*	Tipo do *ticket*: “us”, estória; “*task*”, tarefa de trabalho; ou “*bug*”, erro.	
ticket_priority	*int*	Prioridade do *ticket*.	
ticket_description	*text*	Descrição do *ticket*.	
ticket_bv	*varchar(15)*	Valor de negócio do *ticket*: "*must*", tem que ter; "*should*", deveria ter; "*could*", poderia	

		ter; ou "*want*", interessante ter.	
ticket_estimative	*varchar(4)*	Estimativa para o *ticket*, podendo ser em pontos ou horas.	
ticket_status	*varchar(15)*	*Status* do *ticket*: "*todo*", pendente; "*doing*", em progresso; ou "*done*", completo.	
created	*datetime*	Data de criação do registro.	
modified	*datetime*	Data de alteração do registro.	

Quadro 14: Dicionário de dados da tabela "*tickets*"

No quadro 15 pode-se observar o dicionário de dados da tabela "*sprints*".

Tabela: *sprints*			
Campo	**Tipo**	**Descrição**	**Observação**
id	*int*	Id sequencial da *sprint*.	Chave primária.
release_id	*int*	Id da *release* vinculada.	Chave estrangeira.
sprint_title	*varchar(255)*	Título da *sprint*.	
sprint_description	*text*	Descrição da *sprint*.	
sprint_start_date	*datetime*	Data de início da *sprint*.	
sprint_duration	*int*	Duração, em dias, da *sprint*.	
sprint_finish	*int*	*Flag* indicando se a *sprint* está finalizada: "1", sim; ou "0", não.	
created	*datetime*	Data de criação do registro.	
modified	*datetime*	Data de alteração do registro.	

Quadro 15: Dicionário de dados da tabela "*sprints*"

No quadro 16 pode-se observar o dicionário de dados da tabela "*impediments*".

Tabela: *impediments*			
Campo	**Tipo**	**Descrição**	**Observação**
id	*int*	Id sequencial do impedimento.	Chave primária.
ticket_id	*int*	Id do *ticket* vinculado.	Chave estrangeira.
impediment_description	*text*	Descrição do impedimento.	
impediment_action	*text*	Ação para o impedimento.	
impediment_responsible	*varchar(100)*	Nome do responsável pelo impedimento.	
impediment_status	*varchar(15)*	Situação do impedimento: "*todo*", pendente; ou "*done*", finalizada.	
created	*datetime*	Data de criação do registro.	
modified	*datetime*	Data de alteração do registro.	

Quadro 16: Dicionário de dados da tabela "*impediments*"

No quadro 17 pode-se observar o dicionário de dados da tabela “*ceremonies*”.

Tabela: *ceremonies*			
Campo	**Tipo**	**Descrição**	**Observação**
id	*int*	Id sequencial da cerimônia.	Chave primária.
sprint_id	*int*	Id da *sprint* vinculada.	Chave estrangeira.
ceremony_type	*int(1)*	Tipo da cerimônia: “sp”, planejamento da *sprint*; “srw”, revisão; “sr”, retrospectiva; ou “dm”, reunião diária.	
ceremony_date	*datetime*	Data de realização da cerimônia.	
ceremony_description	*text*	Descrição da cerimônia.	
created	*datetime*	Data de criação do registro.	
modified	*datetime*	Data de alteração do registro.	

Quadro 17: Dicionário de dados da tabela "*ceremonies*"