UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

FERRAMENTA WEB PARA GERÊNCIA DE PROJETOS BASEADOS EM METODOLOGIAS ÁGEIS COM ÊNFASE NA GESTÃO DE REQUISITOS

TRABALHO DE GRADUAÇÃO

Nathan Leidemer

Santa Maria, RS, Brasil

2013

FERRAMENTA WEB PARA GERÊNCIA DE PROJETOS BASEADOS EM METODOLOGIAS ÁGEIS COM ÊNFASE NA GESTÃO DE REQUISITOS

Por

Nathan Leidemer

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (USFM, RS), como requisito parcial para a obtenção de grau de
Bacharel em Ciência da Computação

Orientador: Profª. Drª. Lisandra Manzoni Fontoura

Trabalho de Graduação Nº 357
Santa Maria, RS, Brasil
2013

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

FERRAMENTA WEB PARA GERÊNCIA DE PROJETOS BASEADOS EM METODOLOGIAS ÁGEIS COM ÊNFASE NA GESTÃO DE REQUISITOS

elaborado por
Nathan Leidemer

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Profª. Drª. Lisandra Manzoni Fontoura
(Presidente/Orientador)

Profª. Drª. Ana Trindade Winck (UFSM)

Prof. Dr. Cesar Tadeu Pozzer (UFSM)

Santa Maria, 05 de Março de 2013.

“Caminhar sobre a água e desenvolver software a partir de uma especificação são fáceis se ambos estão congelados.”

—EDWARD V. BERARD

RESUMO

Trabalho de Graduação

Curso de Ciência da Computação

Universidade Federal de Santa Maria

FERRAMENTA WEB PARA GERÊNCIA DE PROJETOS BASEADOS EM METODOLOGIAS ÁGEIS COM ÊNFASE NA GESTÃO DE REQUISITOS

Autor: Nathan Leidemer

Orientador: Profª. Drª. Lisandra Manzoni Fontoura

Local e data da defesa: Santa Maria, 05 de março de 2013.

A tarefa de desenvolver softwares é geralmente bastante complexa, pois envolve mudanças constantes nos requisitos inicialmente definidos pelo cliente. Buscando controlar a constante mudança nos requisitos durante a execução do projeto foram criadas as metodologias ágeis de desenvolvimento. As metodologias ágeis possibilitam lidar com a dinamicidade dos requisitos dividindo o projeto em pequenas iterações. Para que uma constante comunicação entre o cliente e a equipe de desenvolvimento seja possível, é fundamental uma ferramenta que gerencie os requisitos informados pelo cliente e na qual a equipe possa controlar as atividades do projeto que estão sendo executadas em cada iteração. Esse trabalho tem como objetivo desenvolver uma ferramenta web para gerenciamento de requisitos e acompanhamento das atividades, que, sobretudo, facilite a comunicação entre o cliente, o gerente de projetos e os membros da equipe.

Palavras-chave: Gerência de Projetos, Gerência de Requisitos, Metodologias Ágeis.

ABSTRACT

Undergraduate Final Work

Graduation in Computer Science

Federal University of Santa Maria

PROJECT MANAGEMENT TOOL BASED ON AGILE DEVELOPMENT WITH EMPHASIS ON REQUIREMENTS MANAGEMENT

Author: Nathan Leidemer

Adviser: Profª. Drª. Lisandra Manzoni Fontoura

Place and date: March 2013, Santa Maria

The task of developing software is generally quite complex because it involves constant change in requirements initially defined by the customer. Seeking to control the constant change in requirements during the project execution were created agile development methodologies with which you can deal with the dynamic requirements by dividing the project into small iterations. For a constant communication between the client and the development team, it is essential a tool that manages the requirements informed by the customer and where the team can monitor the project activities being performed in each iteration. This study aims to develop a web tool that facilitates the communication between the client, the project manager and team members.

Keywords: Project Management, Requirements Management, Agile Methodologies.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

HTML	*HyperText Markup Language*
IDE	*Integrated Development Environment*
MVC	*Model-View-Controller*
PHP	*Hypertext Preprocessor*
SGBD	Sistema de Gerenciamento de Banco de Dados
UML	*Unified Modeling Language*
XP	*Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO

A tarefa de desenvolver softwares é geralmente bastante complexa, pois envolve mudanças constantes nos requisitos inicialmente definidos pelo cliente. No estudo *Chaos Report*, realizado pelo Standish Group (2010), foi constatado que apenas 37% dos projetos foram concluídos com êxito e o que dentre os principais fatores que causaram esse insucesso estão requisitos incompletos e a falta de envolvimento do usuário.

Buscando controlar a constante mudança nos requisitos durante a execução do projeto foram criadas as metodologias ágeis de desenvolvimento com as quais é possível lidar com a dinamicidade dos requisitos utilizando diversas técnicas como, por exemplo, o desenvolvimento iterativo. Dentre as principais metodologias ágeis que se propõem a resolver esse problema estão o *Scrum* e o *Extreme Programming*.

Para que uma constante interação entre o cliente e a equipe de desenvolvimento seja possível, é fundamental uma ferramenta que gerencie os requisitos informados pelo cliente e onde a equipe possa controlar as atividades do projeto que estão sendo executadas em cada iteração. Levando em conta essa necessidade, neste trabalho é proposto o desenvolvimento de uma ferramenta que permita a interação e colaboração entre os membros do projeto.

O sucesso da gerência de projetos está fortemente ligado com a engenharia de requisitos, se os requisitos estiverem definidos corretamente e as mudanças forem bem controladas, as chances do projeto ser concluído sem extrapolar prazo ou custos aumentam. A ferramenta proposta se destaca das demais ferramentas de projetos em metodologias ágeis existentes, pois possui uma abordagem mais detalhada do gerenciamento de requisitos como o conceito de rastreabilidade de requisitos que é pouco abordada em outras ferramentas.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral deste trabalho consiste em desenvolver uma ferramenta que permita o gerenciamento de projetos com metodologias ágeis, suportando as práticas de gerência de requisitos e de planejamento e acompanhamento de projetos contidas nas metodologias *Scrum* e *Extreme Programming*.

1.1.2 Objetivos Específicos

- Revisar os conceitos relacionados ao trabalho como metodologias ágeis, gerência de projetos e gerência de requisitos;
- Identificar as funcionalidades importantes em uma ferramenta de gerência baseada em metodologias ágeis;
- Projetar uma ferramenta que facilite a interação entre os membros do projeto durante a execução das atividades;
- Realizar a implementação da ferramenta proposta.

1.2 Estrutura do Texto

O texto está organizado como segue. No Capítulo 2 é apresentada uma fundamentação teórica sobre os assuntos relacionados ao trabalho, incluindo conceitos de processos de software, metodologias ágeis, *Scrum*, *Extreme Programming* e gerência de requisitos. No Capítulo 3 são detalhadas as tecnologias utilizadas e a ferramenta proposta por meio de diagrama de casos de uso, interfaces, modelo de classes e modelo de dados. Por fim, no Capítulo 4 são descritas as conclusões e contribuições do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo trata dos conceitos teóricos acerca da engenharia de software, incluindo os conceitos de processos de software, gerência de projetos e engenharia de requisitos, dando ênfase sempre para a aplicação destes conceitos nas metodologias ágeis.

2.1 Processos de software

Segundo SOMMERVILLE (2007), um processo de software é o conjunto de atividades desenvolvidas para gerar um produto de software. Essas atividades envolvem o desenvolvimento de software propriamente dito além de outras ações que dão suporte à criação do software. Embora existam muitos processos de software, o autor define quatro atividades fundamentais que são comuns a todos os processos:

- Especificação de software: envolve a definição das funcionalidades e restrições do software.
- Projeto e implementação: é a etapa em que o software em questão é modelado e implementado.
- Validação de software: o software é verificado para garantir que atende as especificações previamente definidas.
- Evolução de software: possibilita que o software evolua para atender as necessidades do cliente que mudam com o decorrer do uso.

Conforme descrito por HUMPHREY (1989), o desenvolvimento de software pode ser uma tarefa extremamente complexa, pois existem diversas maneiras de executar uma mesma tarefa. Um processo de software bem definido pode ajudar os profissionais de desenvolvimento a fazerem suas

escolhas de maneira ordenada, entendendo melhor quais são suas atribuições e o que esperar dos demais colaboradores do projeto permitindo assim que mantenha o foco no seu trabalho.

Apesar de cada projeto de software ser único, HUMPHREY (1989) cita quatro principais razões para a adoção da padronização dos processos de software. São elas:

- A padronização dos processos ajuda a reduzir os problemas com treinamento, revisão e suporte;
- Com métodos padrões, as experiências adquiridas em cada projeto podem contribuir para a melhoria global do processo;
- Os processos de software dão o suporte para as medidas de qualidade e processos;
- Como a definição do processo leva tempo e esforço para ser produzido, é impraticável definir um processo novo a cada projeto.

2.1.1 Abordagens tradicionais

Desde a formalização do conceito de processos de software surgiram diversos métodos que buscando viabilizar a maior produtividade no desenvolvimento de aplicações e aumentar a qualidade do produto final.

O primeiro modelo definido foi modelo em cascata (ROYCE, 1970) que dá ênfase ao planejamento, suas atividades são divididas em etapas onde cada etapa depende do término da etapa anterior. As etapas de levantamento de requisitos, análise dos requisitos, projeto, implementação, teste e implantação devem ser executadas de maneira ordenada não tratando bem, por exemplo, mudanças nos requisitos durante a etapa de desenvolvimento.

Posteriormente surgiram também outras abordagens, como o modelo espiral proposto por BOEHM (1986), que aborda diretamente o tratamento de riscos do projeto. Cada volta da espiral é dividida em quatro setores (definição dos objetivos, avaliação e redução de riscos, desenvolvimento e validação, planejamento) que são executadas sequencialmente produzindo protótipos cada vez mais próximos do produto final.

2.1.2 Metodologias ágeis

Em fevereiro de 2001, um grupo de 17 metodologistas formou a *Agile Software Development Alliance* e criou um manifesto para encorajar melhores práticas para o desenvolvimento de software. Nesse manifesto o grupo formulou um conjunto de princípios que constituem a base para os processos de desenvolvimento ágil de software (AMBLER, 2004). Abaixo é apresentado um trecho do manifesto, que define seus principais valores:

> Estamos descobrindo melhores maneiras de desenvolver software, fazendo-o e ajudando outros a fazê-lo. Através desse trabalho, valorizamos mais: Indivíduos e interações do que processos e ferramentas; Software em funcionamento do que documentação abrangente; Colaboração do cliente do que negociação de contrato; Respostas a mudança do que seguir um plano. Ou seja, embora itens à direita sejam importantes, valorizamos mais os que estão à esquerda. (BECK et al., 2001).

Além dos quatro valores básicos, foram definidos 12 princípios às quais as metodologias de desenvolvimento ágeis devem seguir, os princípios estão listados abaixo conforme apresentado em (BECK et al, 2001):

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.
2. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
7. Software funcionando é a medida primária de progresso.
8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.

9. Contínua atenção à excelência técnica e bom design aumenta a agilidade.
10. Simplicidade - a arte de maximizar a quantidade de trabalho não realizado - é essencial.
11. As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Esses são os conceitos básicos que dão suporte para todas as metodologias de desenvolvimento ágil. Nesse trabalho serão detalhadas as metodologias Scrum e Extreme Programming que servem como base teórica da ferramenta proposta.

2.1.2.1 Scrum

Conforme descrito por SCHWABER e SUTHERLAND (2001) o Scrum é um *framework* que está sendo usado para gerenciar o desenvolvimento de produtos complexos desde o início de 1990. O Scrum não é um processo ou uma técnica, mas sim um *framework* no qual podem ser empregados vários processos e técnicas. Os componentes básicos do Scrum são: papéis, eventos e artefatos.

No *Scrum* os membros do projeto são divididos em três papéis básicos:

- *Scrum Master*: é o responsável por garantir que as práticas e regras do *Scrum* estejam sendo corretamente aplicadas.
- Equipe de desenvolvimento: são os profissionais responsáveis por criarem os incrementos do produto ao final de cada *Sprint*. Idealmente as equipes possuem entre 3 e 9 integrantes e são responsáveis por organizarem e gerenciarem o seu próprio trabalho.
- *Product Owner*: é o responsável por aumentar o valor do produto e do trabalho da equipe de desenvolvimento. Ele é a única pessoa responsável por gerenciar o *Backlog* do Produto.

Existem cinco eventos que ocorrem durante o decorrer das atividades como descritos a seguir:

- *Sprint*: é o tempo levado para que cada incremento do produto esteja pronto, deve durar entre duas semanas e um mês. Cada *Sprint* é composta por uma reunião de planejamento, reuniões diárias, o trabalho de desenvolvimento, uma revisão e retrospectiva da *Sprint*.

- Reunião de planejamento da *Sprint*: durante essa reunião, que leva em média 8 horas, é definido o que será feito até o fim da *Sprint* e como o trabalho escolhido será realizado. A equipe de desenvolvimento é quem decide quantos itens, previamente priorizados pelo *Product Owner* no *Backlog* do Produto, serão implementados durante a *Sprint* e os inclui no *Backlog* da *Sprint*.

- Reunião diária: é um evento de 15 minutos realizado diariamente para a equipe sincronizar as atividades e criar um plano para as próximas 24 horas.

- Revisão da *Sprint*: é uma reunião geralmente de 4 horas onde o produto criado na *Sprint* será inspecionado.

- Retrospectiva da *Sprint*: é o momento em que a equipe avalia seu desempenho e cria um plano de melhorias para a próxima *Sprint*. Essa reunião leva em média 3 horas e ocorre após a revisão da *Sprint*.

Na Figura 1 pode ser visualizada a organização das atividades do projeto, com o tempo estimado para a realização de cada uma.

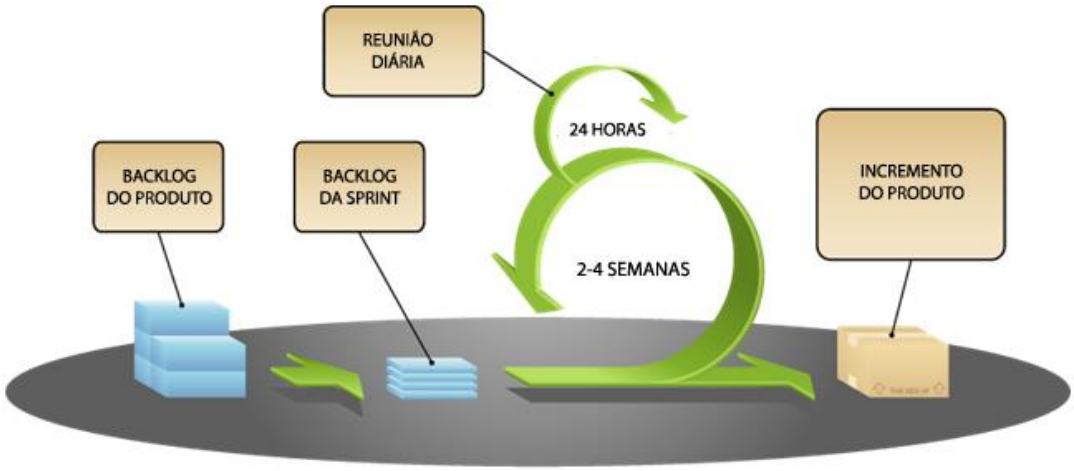

Figura 1. Processo Scrum (adaptado de COHN 2005).

O *framework* define ainda quatro artefatos:

- *Backlog* do produto: é uma lista ordenada definida pelo *Product Owner*, nela estão definidos os requisitos, características, funções, melhorias e correções que devem estar presentes no produto nas próximas versões. Essa lista sofre alterações constantemente durante o

desenvolvimento do produto e no topo da lista são colocados os itens com maior prioridade e que já estão mais refinados.

- *Backlog* da *Sprint*: é um conjunto de itens do *Backlog* do Produto selecionados para que estejam prontos ao final da *Sprint*. Esse *Backlog* contém também a estimativa da equipe do esforço necessário para realizar cada tarefa.

- Incremento: o incremento é a soma de todos os itens do *Backlog* do produto completados durante a *Sprint* e nas *Sprints* anteriores.

2.1.2.2 Extreme Programming (XP)

O Extreme Programming, ou XP, é um processo de desenvolvimento de software voltado para projetos cujos requisitos são vagos e mudam com frequência; desenvolvimento de sistemas voltado a objetos; equipes pequenas, preferencialmente até 12 desenvolvedores; desenvolvimento incremental, onde o sistema começa a ser implementado logo no início do projeto e vai ganhando novas funcionalidades ao longo do tempo. Os quatro valores do XP são *feedback*, comunicação, simplicidade e coragem (TELES, 2004).

O XP é guiado por um conjunto de 13 práticas como pode ser visto na Figura 2, adaptada de JEFFRIES (2012):

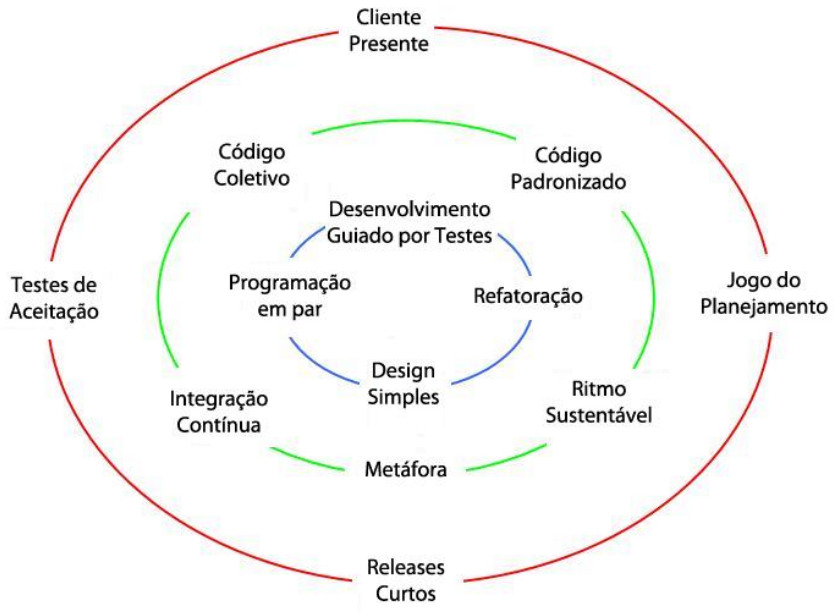

Figura 2. Práticas do XP (adaptado de JEFFRIES 2012).

Conforme apresentado por TELES (2004) as 12 práticas que dão a fundamentação ao XP estão descritas na Tabela 1.

Tabela 1. Práticas do XP (adaptada de TELES, 2004).

Práticas	Descrição resumida
Cliente Presente	O cliente deve conduzir o desenvolvimento dando constante feedback à equipe de desenvolvimento.
Jogo do Planejamento	É a reunião onde o cliente avalia as funcionalidades que devem ser implementadas naquela iteração e as prioriza. Cada funcionalidade é descrita em forma de estória e a equipe de desenvolvimento deve estimar o esforço para desenvolvê-la para que o cliente possa conhecer o custo de implementação de cada uma delas. As tarefas que tiverem sido definidas com maior prioridade pelo cliente farão parte do *Backlog* da *Sprint*.
Programação em Par	Os desenvolvedores sempre implementam as funcionalidades em dupla, isso permite que o código seja revisado enquanto ainda é construído.
Desenvolvimento Guiado por Testes	Antes de codificar cada funcionalidade os desenvolvedores escrevem testes para elas, isso resulta em um maior entendimento das necessidades do cliente e de como a funcionalidade deve ser implementada.
Testes de Aceitação	São testes construídos pelo cliente em conjunto com membros da equipe para validar um determinado requisito do sistema.
Refatoração	Consiste na alteração do código desenvolvido anteriormente sem modificar sua funcionalidade com o intuito de manter o sistema simples e facilitar sua manutenção.
Código Coletivo	No XP o código não tem dono, todos os desenvolvedores podem revisar e alterar o código produzido por outras duplas.
Código Padronizado	Para que todos possam alterar com mais facilidade o código produzido pelas demais duplas a equipe define padrões de codificação.
Design Simples	A equipe deve sempre optar por um código que seja suficiente para atender às necessidades somente da funcionalidade que está sendo implementada sem se preocupar com implementações futuras.
Metáfora	A equipe utiliza metáforas para facilitar a criação de um design simples, pois elas possibilitam transmitir ideias complexas de forma simples.
Ritmo Sustentável	O processo recomenda que os desenvolvedores trabalhem apenas 8 horas por dia e evitem fazer horas-extras para que mantenham-se sempre atentos, criativos e dispostos a solucionar problemas.
Integração Contínua	Quando uma nova funcionalidade é incorporada ao sistema os pares devem integrar seus códigos ao restante do sistema.
Releases Curtos	Para gerar um fluxo contínuo de valor para o cliente, a equipe produz um conjunto reduzido de funcionalidades a cada iteração para que o cliente possa usar o software em seu dia-a-dia e beneficiar-se dele.

A equipe XP é composta por cinco papéis básicos, são eles:

- Gerente de Projeto: é o responsável por assuntos administrativos, pelo relacionamento com o cliente e por fornecer informações essenciais sobre o projeto.
- *Coach*: é o responsável técnico que deve assegurar que a equipe está seguindo as técnicas do XP garantindo também o bom funcionamento do processo e buscando formas de melhorá-lo.
- Analista de Teste: é o responsável por auxiliar o cliente a escrever os testes e garantir que eles sejam realizados diversas vezes durante o processo para identificar eventuais defeitos logo que apareçam.
- Redator Técnico: ele ajuda a equipe a documentar o sistema possibilitando que os desenvolvedores se foquem na implementação do software.
- Desenvolvedor: é o responsável por analisar, projetar e codificar o sistema. No XP não existem as funções de analista, projetista e programador, cada desenvolvedor exerce essas funções durante o projeto.

Na Figura 3, adaptada de SOMMERVILLE (2007), pode ser visto a ordem das atividades realizadas durante o ciclo de um release em XP.

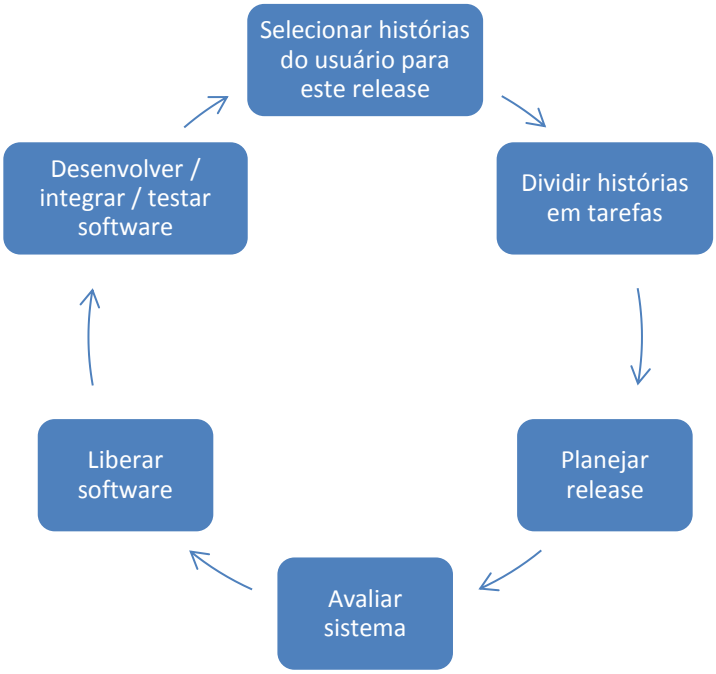

Figura 3. Ciclo de um release em XP (adaptado de SOMMERVILLE 2007).

2.2 Engenharia de requisitos

Conforme definição de SOMMERVILLE (2007), os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que ele oferece e as restrições a seu funcionamento. O processo de descobrir, analisar, documentar e verificar esses serviços e restrições é chamado de engenharia de requisitos. Os processos da engenharia de requisitos podem ser divididos em quatro grupos de atividades que serão tratadas nas próximas subseções.

2.2.1 Especificação de requisitos

Nessa etapa os integrantes do projeto devem interagir para definir quais funcionalidades deverão estar contidas no produto. No XP, conforme descrito por TELES (2004), os requisitos são levantados através de estórias que devem ser escritas pelo cliente onde ele descreve uma estória de uso do sistema e qual a importância dessa funcionalidade.

A Tabela 2, adaptada de WILDT e LACERDA (2010), mostra um modelo de estória de usuário definida em XP:

Tabela 2. Exemplo de estória de usuário (adaptada de WILDT e LACERDA, 2010).

Sendo	uma pessoa que precisa de crédito
Posso	acessar o simulador de crédito ACME
Pois assim	saberei das opções que tenho para requisitar

2.2.2 Análise e negociação

Nessa etapa o cliente deve ser capaz de identificar quais histórias são mais importantes no momento e devem estar já no próximo release. TELES (2004) explica que essa tarefa deve ser realizada pelo cliente levando em conta as suas necessidades, mas também deve ser levados em conta aspectos técnico de implementação que a equipe deverá informar, como por exemplo, dependências técnicas.

Apesar de ser o cliente quem define a prioridade das estórias a serem implementadas, a equipe pode não ser capaz de implementar no próximo release todas as funcionalidades que o cliente desejava. Isso pode ocorrer tanto por questões técnicas quanto pelo custo de implementação

e a capacidade da equipe. Essa etapa envolve a negociação entre equipe e cliente para solucionar os possíveis conflitos.

2.2.3 Validação

Conforme definição de SOMMERVILLE (2007), a validação é o processo pelo qual se verifica se os requisitos previamente levantados realmente definem o sistema que o cliente quer. Essa é uma etapa importante, pois erros nos requisitos podem gerar grandes custos com retrabalho quando descobertos durante a etapa de desenvolvimento ou quando o sistema já estiver em uso pelo usuário.

No XP utiliza-se a técnica de geração de casos de testes para descobrir possíveis erros nas estórias definidas pelo usuário anteriormente. O processo considera que se for difícil ou impossível projetar um teste, isso normalmente significa que os requisitos também serão difíceis de serem implementados e devem ser reconsiderados.

2.2.4 Gerenciamento de requisitos

Os requisitos para um sistema de grande porte estão sempre mudando. No começo do projeto geralmente o escopo ainda não está completamente definido então os requisitos do sistema devem evoluir para refletir essa constante mudança.

Na Figura 4, adaptada de SOMMERVILLE (2007), pode ser vista a evolução dos requisitos durante o andamento do projeto.

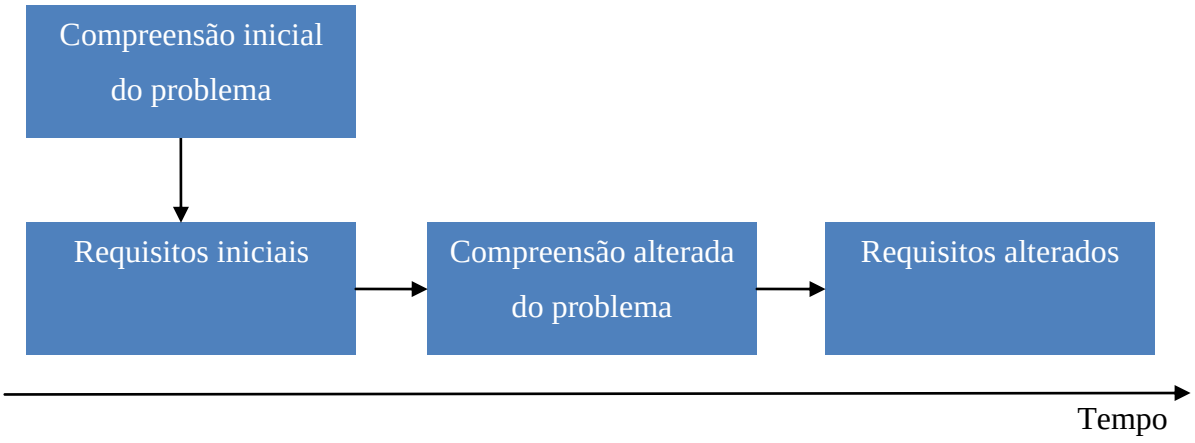

Figura 4. Evolução dos requisitos (adaptado de SOMMERVILLE 2007).

Os processos ágeis de desenvolvimento, como o XP, foram concebidos para lidar com requisitos que mudam durante o processo de desenvolvimento. Nesses processos, quando um usuário propõe uma mudança nos requisitos, a mudança não passa por um processo formal de gerenciamento de mudanças. Pelo contrário, o usuário tem de priorizar essa mudança e, em caso de alta prioridade, ele deve decidir quais recursos planejados para a próxima iteração devem ser abandonados (SOMMERVILLE, 2007).

2.2.4.1 Rastreabilidade de Requisitos

Conforme apresentado por SOMMERVILLE (2007) a rastreabilidade de requisitos é uma atividade importante pois ajuda na compreensão e gerência das informações fornecidas sobre os requisitos. Entre as principais finalidades da rastreabilidade podem ser destacadas:

- Compreender a origem dos requisitos;
- Gerenciar o escopo do projeto;
- Gerenciar mudanças de requisitos;
- Avaliar o impacto no projeto da mudança de um requisito;
- Avaliar o impacto da falha de um teste nos requisitos.

Na Figura 5, adaptada de WIEGERS (2003), pode ser visto um modelo de matriz de rastreabilidade que relaciona os casos de uso do sistema com os requisitos funcionais.

Requisitos Funcionais	Casos de Uso			
	UC-1	UC-2	UC-3	UC-4
FR-1	↵			
FR-2	↵			
FR-3			↵	
FR-4			↵	
FR-5		↵		↵
FR-6			↵	

Figura 5. Matriz de rastreabilidade (adaptado de WIEGERS 2003).

Os itens típicos de rastreabilidade incluem diferentes tipos de requisitos, elementos de modelo de *design* e análise, artefatos de teste e material de treinamento e documentação de suporte ao

usuário final. A relação entre os itens geralmente é exposta ao usuário no formato de uma matriz que apresenta a dependência de um item com o outro.

3 PROPOSTA DE FERRAMENTA

Este trabalho propôs a criação de uma ferramenta *web* para a gerência de projetos baseados em metodologias ágeis, unindo os conceitos das metodologias Scrum e XP. A ferramenta dá ênfase para a gerência de requisitos possibilitando o levantamento de requisitos através de estórias do usuário, a priorização dessas estórias, a criação do *Backlog* da *Sprint* e a geração de matrizes de rastreabilidade.

3.1 Ambiente de Desenvolvimento (Ferramentas)

Para o desenvolvimento da ferramenta proposta no trabalho foram utilizadas as seguintes ferramentas e tecnologias:

- Apache 2.2: O Apache (http://www.apache.org/) é um servidor web livre criado em 1995 e mantido pela Apache Software Fundation.
- HTML: O HTML é uma linguagem de marcação utilizada para produzir páginas web.
- jQuery 1.8: jQuery (http://jquery.com/) é uma biblioteca Javascript de código aberto lançada em 2006 e desenvolvida por John Resig. Essa biblioteca é usada para tornar as interfaces dinâmicas facilitando a interação do usuário com o HTML.
- PHP 5.3: O PHP (http://www.php.net/) é uma linguagem de programação livre para o desenvolvimento de aplicações web. Ela foi criada em 1995 e é mantida atualmente pelo The PHP Group.
- CakePHP 2.2: O CakePHP (http://www.cakephp.org/) é um framework livre escrito em PHP criado em 2005 e mantido pela Cake Software Foundation. O framework baseia-se em conceitos de engenharia de software e padrões de projeto como o MVC.

- MySQL 5.5: O MySQL (http://www.mysql.com/) é um SGBD que utiliza a linguagem SQL como interface, foi criado em 1994 e é atualmente mantido pela Oracle Corporation.
- MySQL Workbench 5.2: O MySQL Workbench (http://mysqlworkbench.org/) é uma ferramenta para a modelagem visual de bases de dados MySQL. Foi desenvolvida em 2002 e é mantida pela Oracle Corporation.
- phpMyAdmin 3.5: O phpMyAdmin (http://www.phpmyadmin.net/) é uma aplicação web desenvolvida em PHP para a administração do MySQL criado em 1998.
- NetBeans 7.1: O NetBeans (http://netbeans.org/) é uma IDE livre para o desenvolvimento em diversas linguagens como PHP, Java, C e C++. Foi lançada em 2000 e é mantida pela Oracle Corporation.

3.2 Especificação

A ferramenta proposta foi modelada utilizando-se a Linguagem de Modelagem Unificada (UML). Foram elaborados o diagrama de casos de uso para modelar a interação dos usuários com o sistema e o diagrama de classes para modelar a estrutura do sistema.

A ferramenta foi concebida a partir de práticas/atividades contidas nas metodologias *Scrum* e *XP*, levando em conta as áreas de gerência de projetos e gerência de requisitos. Serão implementados cadastro de equipe, cadastro de projetos, *Backlog* do Produto, *Backlog* da *Sprint*, cadastro de estórias, priorização de estórias, divisão de estórias em tarefas, quadro de tarefas, casos de testes e matrizes de rastreabilidade.

3.2.1 Diagrama de casos de uso

Conforme visto em SOMMERVILLE (2007) os casos de uso são uma técnica baseada em cenários que representam as situações em que os usuários interagem com o sistema. A técnica é usada para identificar os requisitos do sistema. O modelo de casos de uso para a ferramenta proposta pode ser visto na Figura 6.

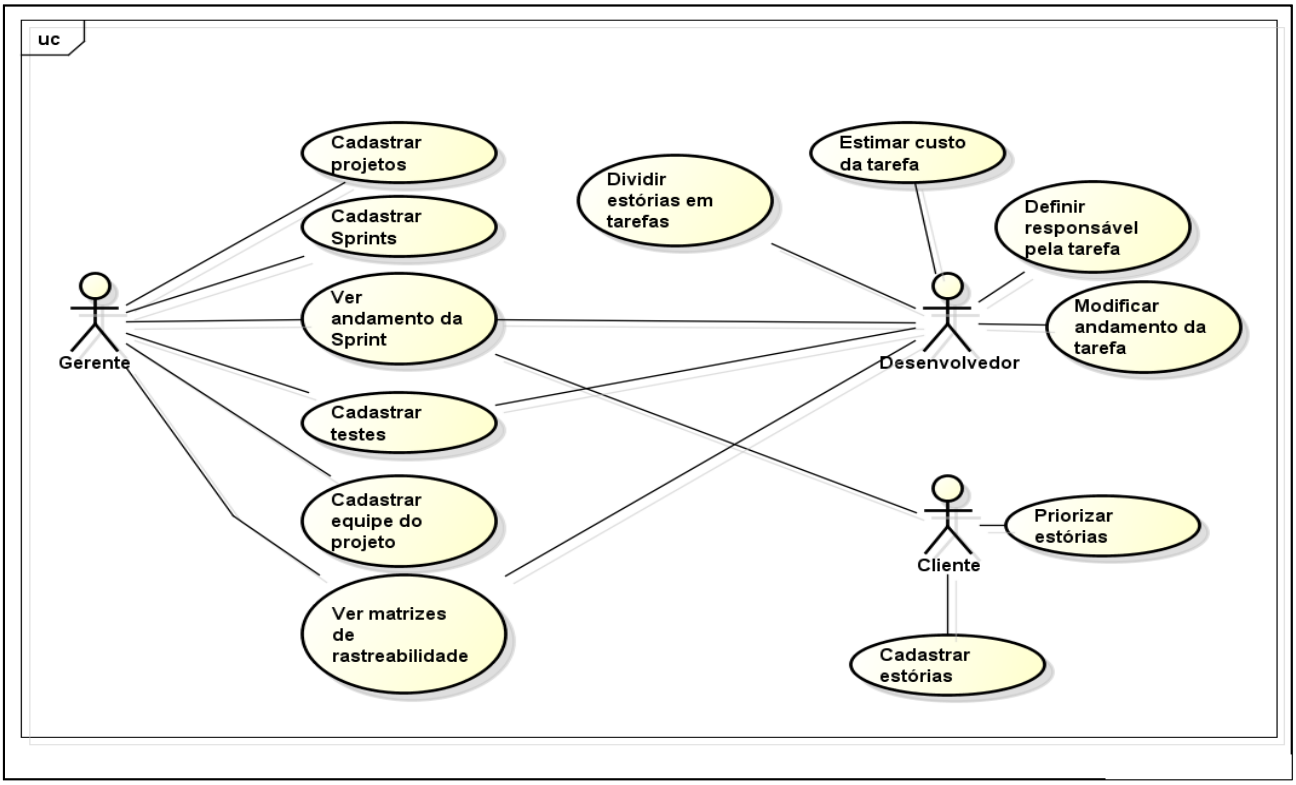

Figura 6. Diagrama de casos de uso

O sistema é composto por três atores que interagem com o sistema: gerente de projetos, desenvolvedor e cliente. Entre as atividades realizadas pelo gerente estão o cadastro de projetos, cadastro de *Sprints*, acompanhamento da *Sprint*, cadastro de testes, cadastro da equipe do projeto e visualização das matrizes de rastreabilidade.

As atividades do desenvolvedor são dividir estórias em tarefas, estimar o custo de cada tarefa, definir o responsável pela tarefa e modificar o seu andamento. Além disso, o desenvolvedor também acompanha o andamento da *Sprint* e visualiza as matrizes de rastreabilidade. Por fim, cabe ao cliente cadastrar e priorizar as estórias e acompanhar o andamento da *Sprint*.

3.2.2 Diagrama de classes

A ferramenta foi modelada com sete classes: Projetos, Rastreabilidade, Sprints, ProductBacklogs, Usuários, Tarefas e Testes. Os métodos básicos presentes na maioria das classes são index, add, edit, delete e ordenar que são utilizados respectivamente para listar os registros cadastrados, adicionar um novo registro, editar o item escolhido, remover o item e salvar a ordenação dos itens conforme a interação do usuário.

Na classe Sprints existem dois métodos específicos, um deles é utilizado para que seja montado o *Backlog* da *Sprint* e o outro responsável pelo quadro de andamento das tarefas (Kanban). A classe rastreabilidade só conta com o método responsável pela listagem, pois os itens exibidos nas matrizes de rastreabilidades são adicionados em outras classes.

O diagrama de classes para a ferramenta proposta pode ser visto na Figura 7.

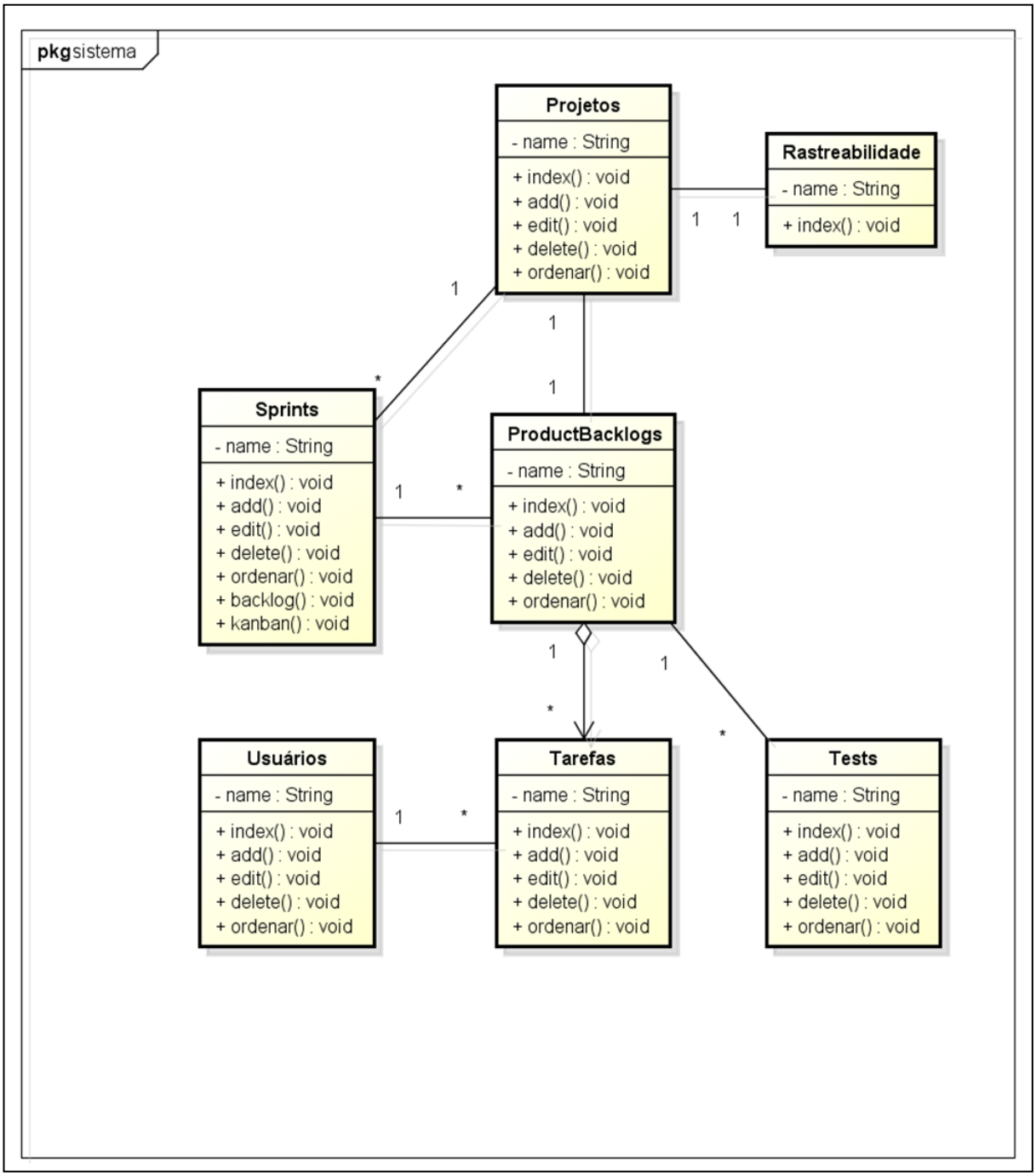

Figura 7. Diagrama de classes

3.2.3 Modelo de dados

A base de dados MySQL foi modelada com seis tabelas relacionadas entre si, são elas: projetos, sprints, usuários, tarefas, tests, e product_backlogs. Em comum a todas as tabelas estão os campos id que guarda o identificador do registro, os campos created e updated que guardam as datas de criação e atualização do registro e o campo posição que guarda a ordem do registro na interface após o usuário fazer a priorização dos dados.

Na figura 8 pode ser visto o modelo de dados completo da ferramenta com relacionamentos entre as tabelas e os campos específicos contidos em cada uma delas.

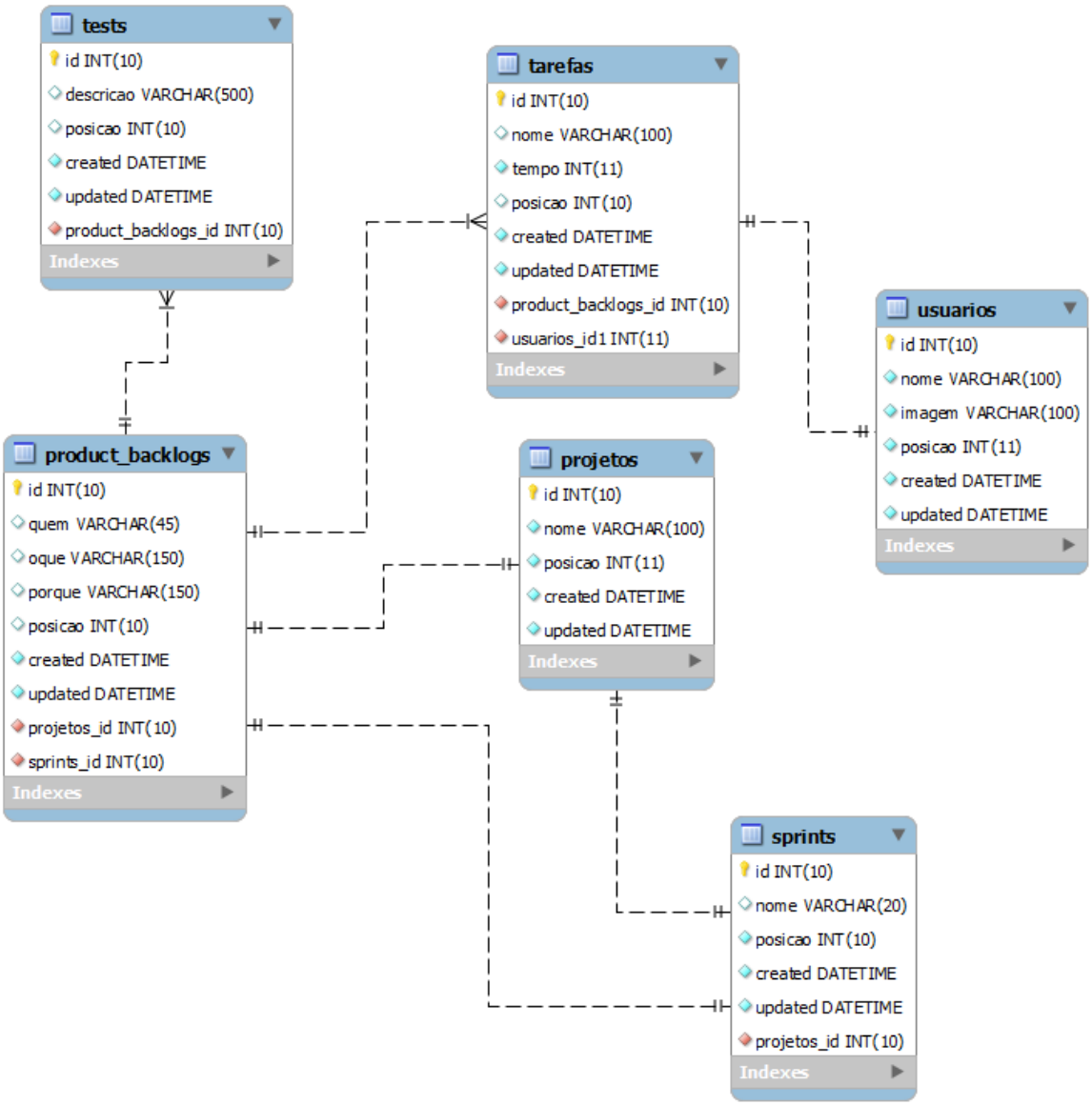

Figura 8. Modelo de dados

3.2.4 Funcionalidades implementadas

Nesta subseção serão apresentadas as funcionalidades implementadas mostrando capturas das telas do sistema. Para cada funcionalidade será especificado se ela faz parte da gerência de projetos ou requisitos e se faz parte da metodologia XP ou Scrum.

A primeira interface do sistema, mostrada na Figura 9, exibe os projetos cadastrados no sistema com botões que levam para a adição, edição e exclusão dos projetos. Nessa interface também estão contidos os botões que levam para as telas de *Sprints*, *Backlog* do Produto e Matrizes de rastreabilidade.

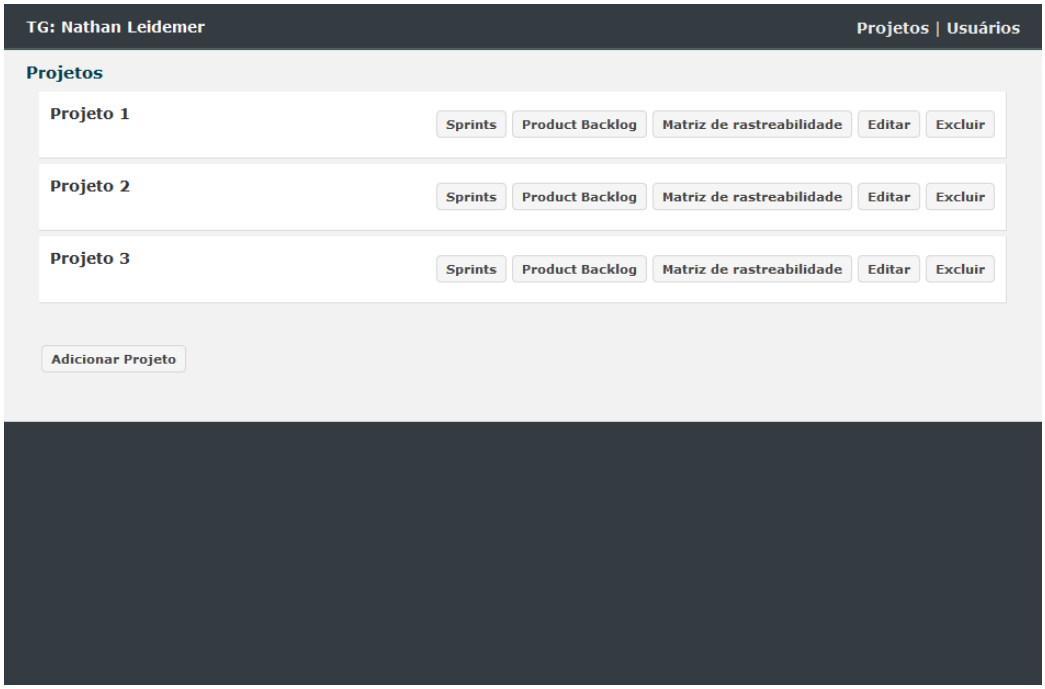

Figura 9. Interface para o controle dos projetos

O formulário, exibido na Figura 10, possibilita a edição dos dados referentes aos projetos. Todos os demais formulários de adição e edição seguem o mesmo padrão e, portanto não serão mostrados aqui.

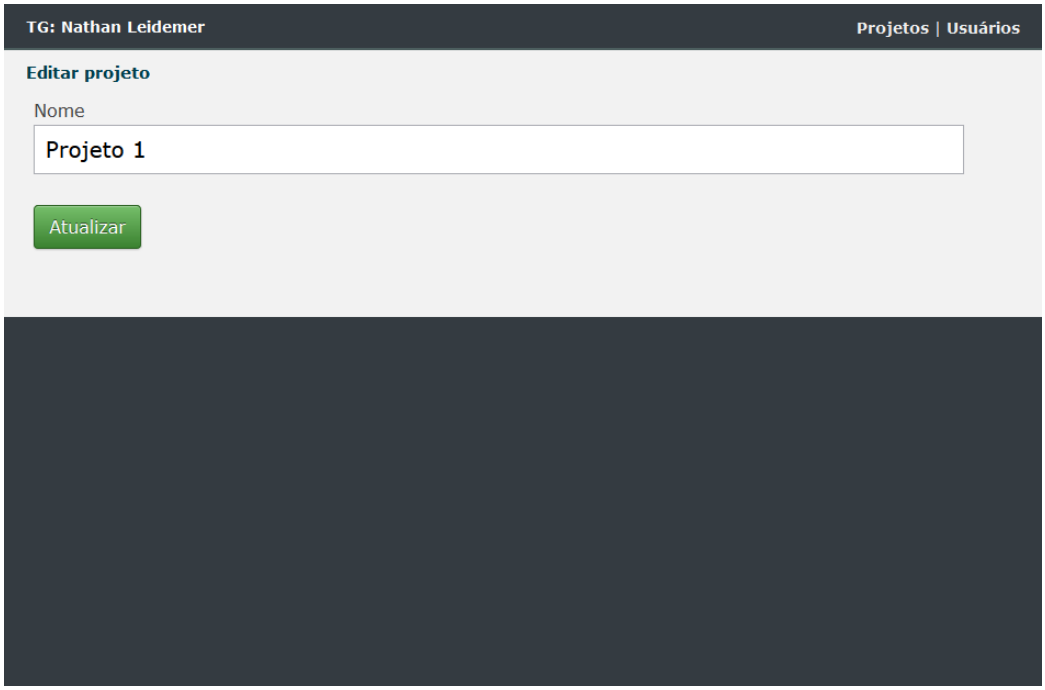

Figura 10. Formulário de edição dos projetos

Na interface mostrada na Figura 11, está o *Backlog* do Produto que faz parte das atividades de gerência de projetos em Scrum. Para o levantamento dos requisitos foi usado o método de estórias do usuário que está descrito na metodologia XP.

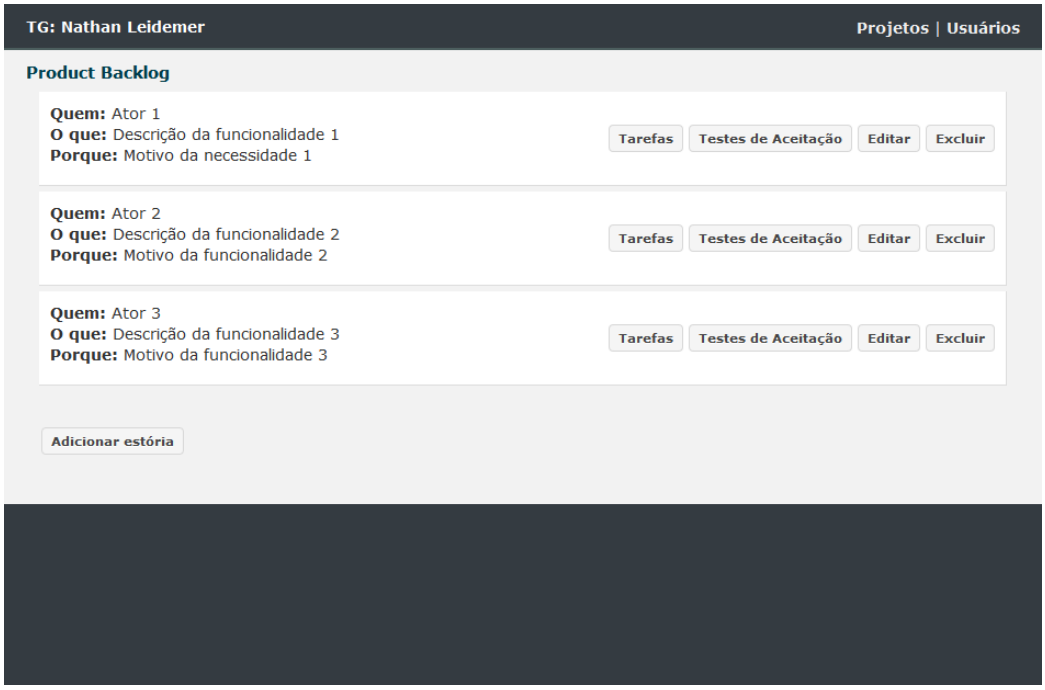

Figura 11. Interface do *Backlog* do Produto

Ainda dentro da gerência de requisitos a interface, mostrada na Figura 11, possibilita o controle das tarefas do projeto. Nas metodologias ágeis cada estória definida pelo usuário é dividida em tarefas pela equipe de desenvolvimento que também deve estimar qual o tempo necessário para que ela esteja concluída.

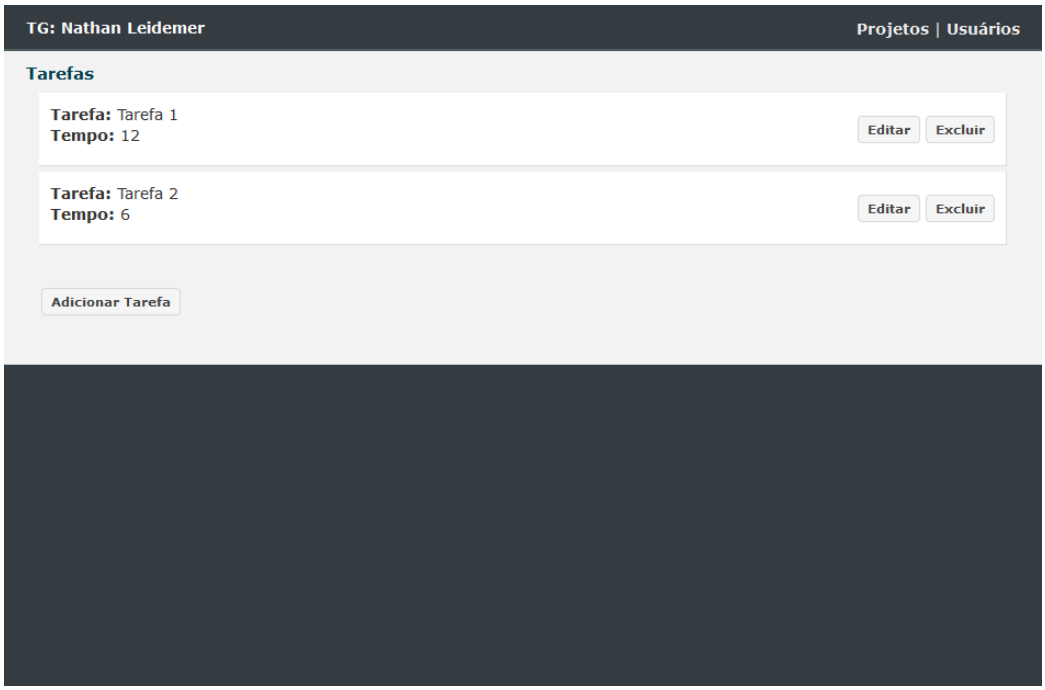

Figura 12. Interface de controle de tarefas

Para cada estória definida pelo usuário também são definidos testes de aceitação que deverão ser executados para garantir que o sistema atenda aos requisitos do usuário. Essa é uma atividade característica da metodologia XP que considera que para cada estória deve haver ao menos um teste de aceitação escrito pelo usuário.

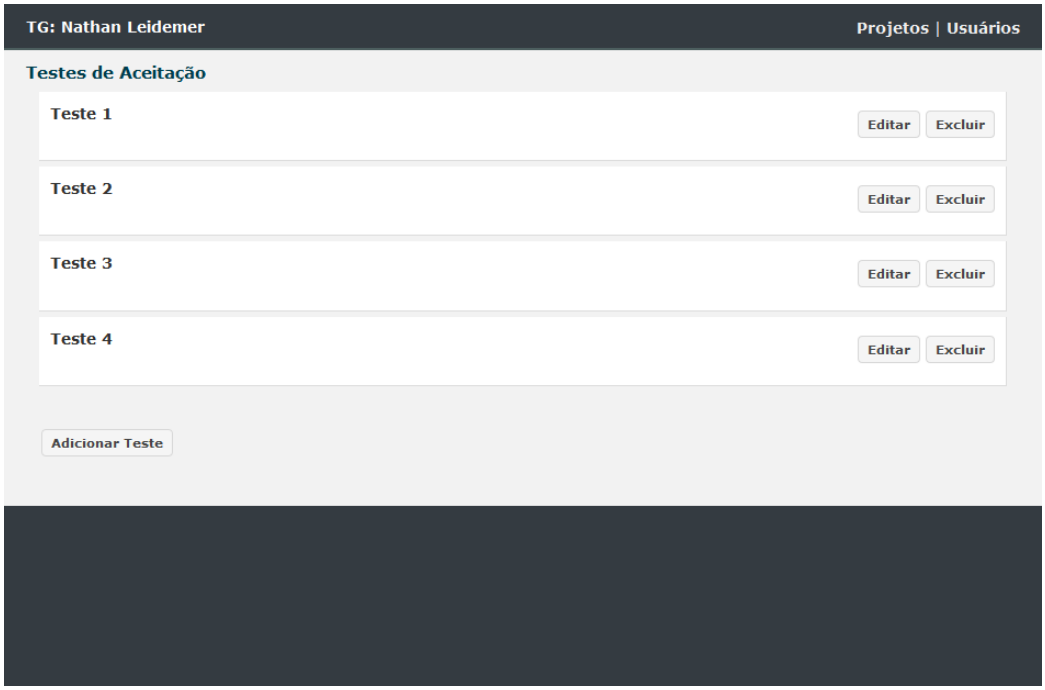

Figura 13. Interface de controle dos testes

A Figura 14 mostra a forma como são exibidas as matrizes de rastreabilidade que são uma importante ferramenta da gerência de requisitos. Nessa interface foi implementada uma tabela que mostra a dependência entre as tarefas e as estórias e outra que mostra a dependência dos testes com as estórias.

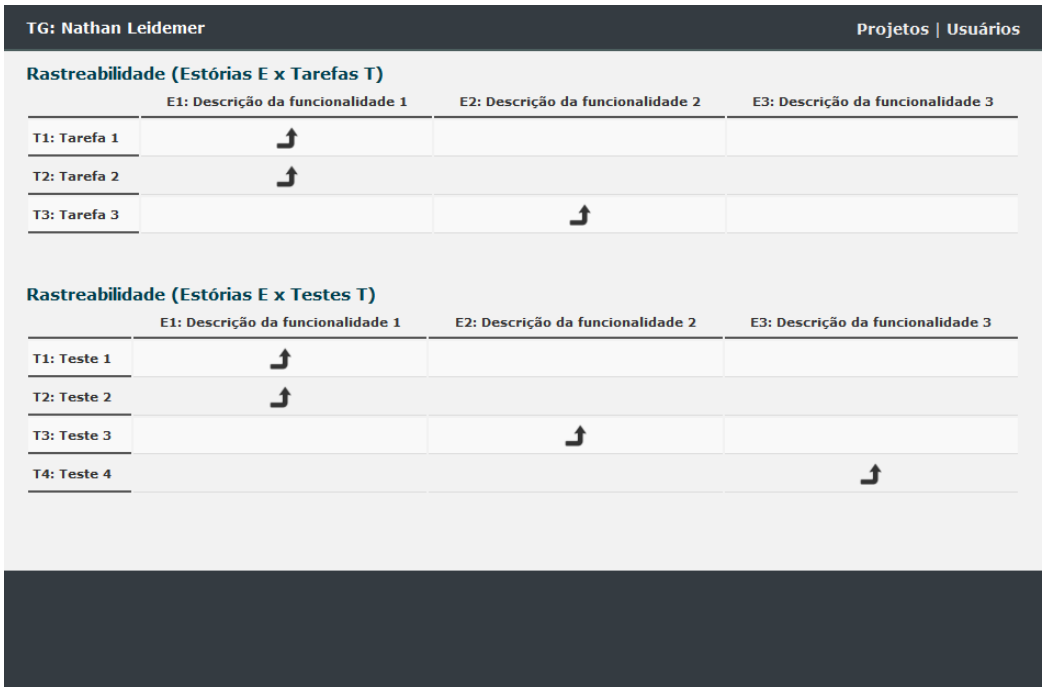

Figura 14. Interface das matrizes de rastreabilidade

Na interface da Figura 15 são cadastrados os membros da equipe do projeto. Esse cadastro é importante, pois posteriormente cada tarefa será atribuída a um membro da equipe para a sua realização.

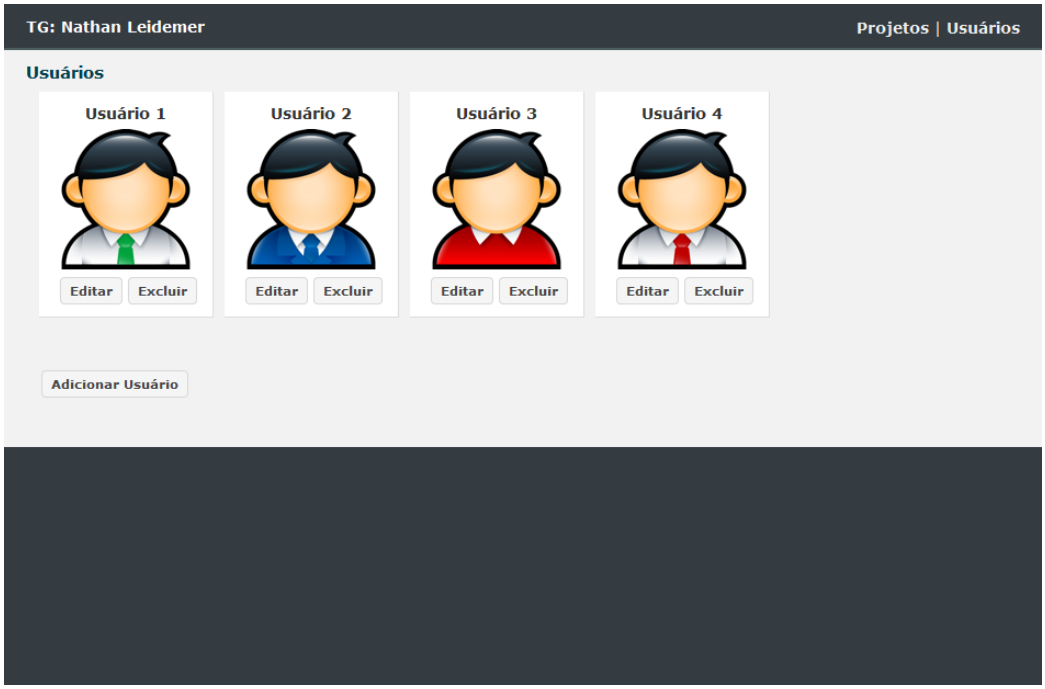

Figura 15. Interface de controle da equipe do projeto

Através da interface, exibida na Figura 16, o gerente pode realizar o controle das *Sprints* do projeto. As *Sprints* fazem parte da gerência do projeto sendo definidas na metodologia Scrum. Na metodologia XP as *Sprints* também estão presentes com a denominação de iterações.

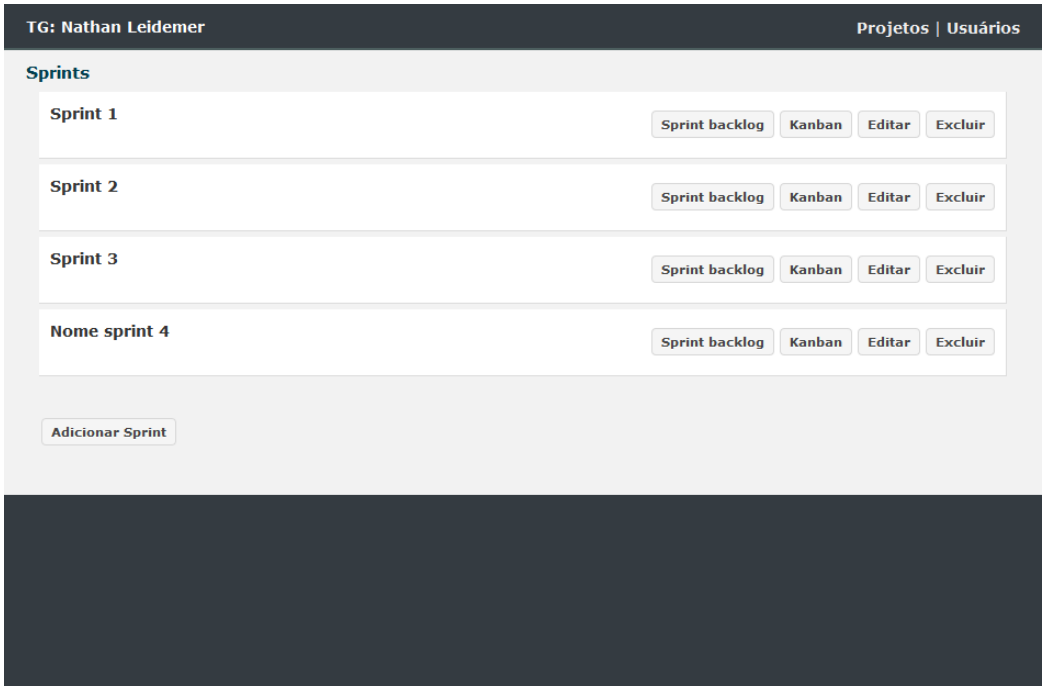

Figura 16. Interface de controle das Sprints

O *Backlog* da *Sprint*, exibido na Figura 17, é onde a equipe define quais tarefas que estão na lista de pendências poderão ser implementadas durante a *Sprint* atual. Para definir quais tarefas serão realizadas na próxima *Sprint* o usuário deve arrastá-la da lista de tarefas para a lista do *Backlog* da *Sprint*.

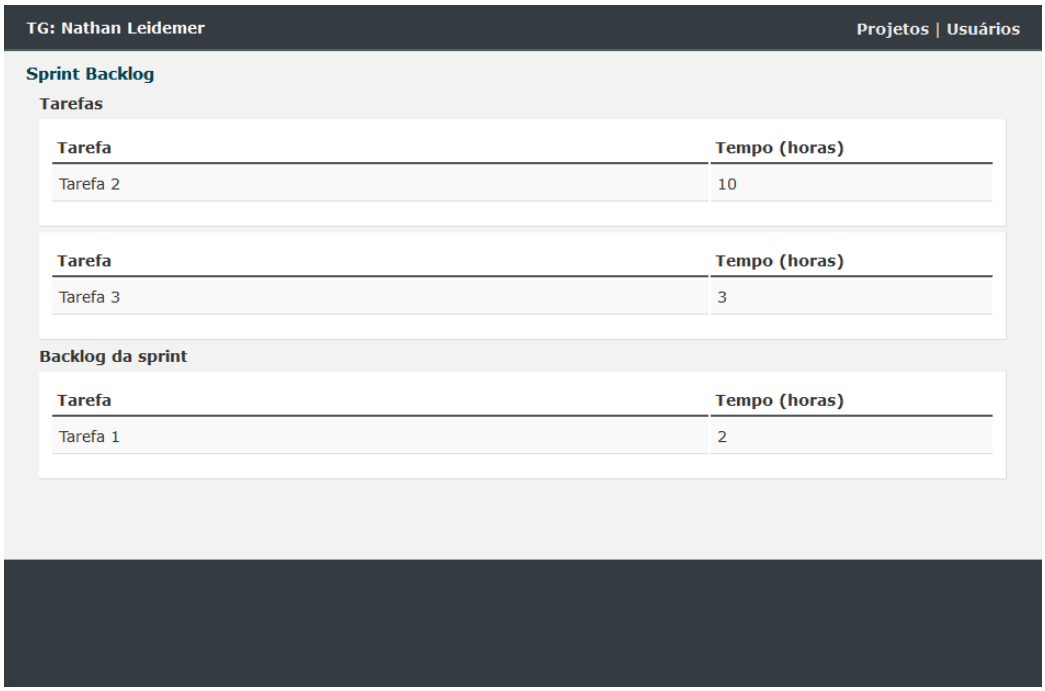

Figura 17. Interface de controle do *Backlog* da *Sprint*

Na interface da Figura 18 é controlado o andamento das tarefas definidas para a *Sprint*. Para cada tarefa é definido o usuário responsável e ela é passada de coluna conforme o seu andamento. Essa também é uma tarefa da gerência de projetos.

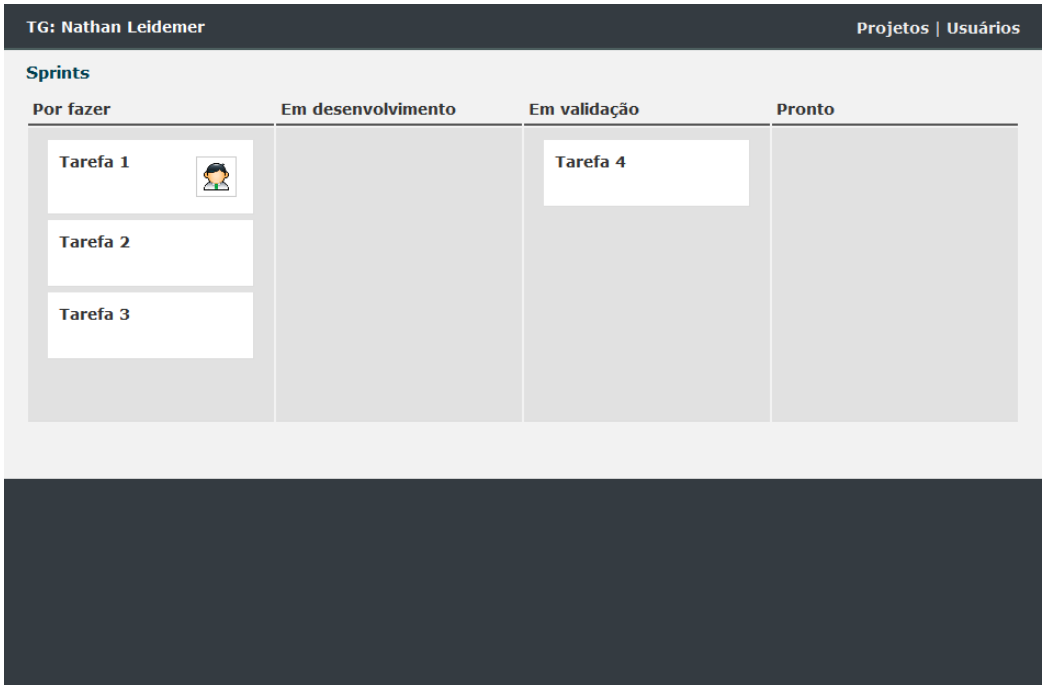

Figura 18. Interface de andamento da *Sprint* (*Kanban*)

Por fim, a Figura 19 mostra como diversas interações do usuário são feitas com o sistema. Para priorizar as estórias, definir o *Backlog* da *Sprint* e controlar o andamento das tarefas o usuário arrasta os elementos na tela e tais ações são salvas em tempo real no banco de dados.

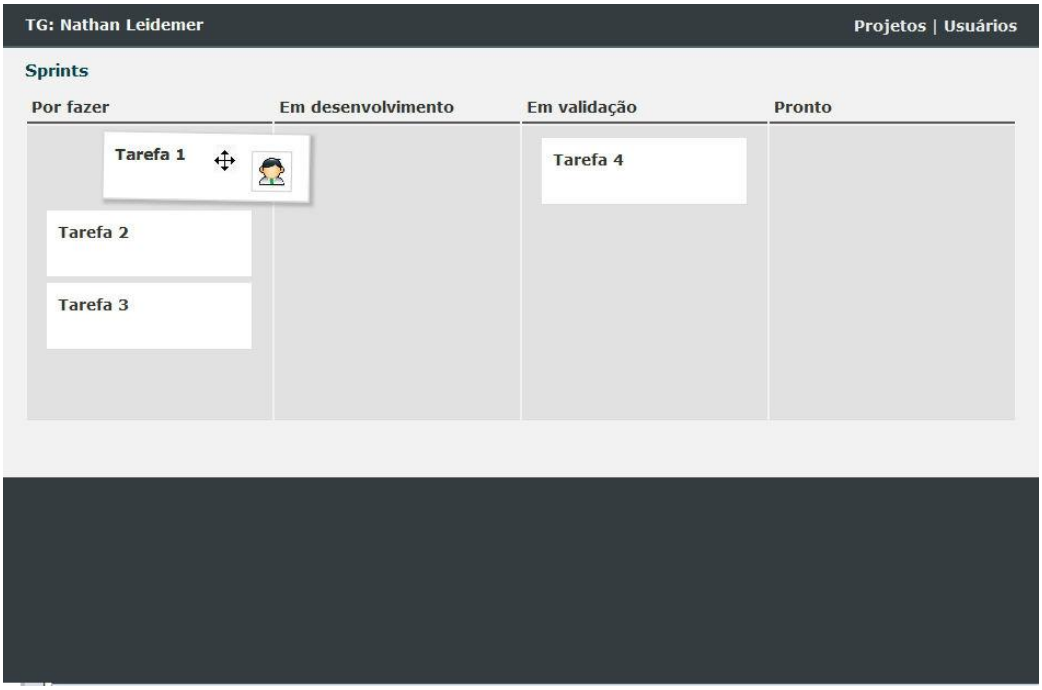

Figura 19. Interface de interação do usuário com o sistema

4 CONCLUSÃO

Nesse trabalho foi apresentada uma revisão teórica dos processos de software destacando-se a importância das áreas de gerência de projetos e a gerência de requisitos para que os projetos sejam concluídos com sucesso. Foram abordadas também as metodologias ágeis de desenvolvimento que tentam resolver o problema da constante mudança dos requisitos durante o desenvolvimento do projeto.

Baseando-se nesses conceitos foi proposta e desenvolvida uma ferramenta web que permite a gerência de projetos e engloba especialmente conceitos da área de engenharia de requisitos. Para o desenvolvimento foi utilizada a linguagem PHP com o *Framework* orientado a objetos CakePHP, além de outras tecnologias como base de dados MySQL e a biblioteca jQuery para permitir as interações do usuário com a interface.

Os gerentes de projetos em metodologias ágeis muitas vezes deixam de lado um controle detalhado dos requisitos preocupando-se exclusivamente com o controle das atividades executadas durante projeto. Com esse trabalho foi possível constatar que as áreas de gerência de projetos e gerência de requisitos podem ser combinadas em uma única ferramenta para aumentar as chances de sucesso do projeto sem que a equipe precise realizar muito esforço para criar a documentação.

Como trabalhos futuros, sugere-se a implementação de um controle de acesso e permissões de usuários para que cada membro do projeto tenha acesso somente às partes do sistema pelo qual é responsável. Também sugere-se a adição de outras técnicas da gerência de projetos ou gerência de requisitos como, por exemplo, o gráfico *Burndown* que permite o acompanhamento estatístico do andamento da *Sprint*.

REFERÊNCIAS

AMBLER, SCOTT W. **Modelagem Ágil: Práticas eficazes para a Programação eXtrema e o Processo Unificado**. 1ª. ed. Porto Alegre: Bookman, 2004.

BECK et al. **Manifesto para Desenvolvimento Ágil de Software**, 2001. Disponível em: <http://agilemanifesto.org/iso/ptbr/>. Acesso em: 05 jan. 2013.

BOEHM, BARRY. **A Spiral Model of Software Development and Enhancement**. ACM Software Engineering Notes, Vol. 11, ago. 1986, p. 22-42.

COHN, MIKE. An **Overview of Scrum for Agile Software Development**, 2005. Disponível em: <http://www.mountaingoatsoftware.com/scrum/overview>. Acesso em: 05 jan. 2013.

HUMPHREY, WATTS S. **Managing the Software Process**. 1ª. ed. Massachussets: Addison-Wesley, 1989.

JEFFRIES, RONALD E. **What is Extreme Programming?**, 2012. Disponível em: <http://xprogramming.com/what-is-Extreme-programming/>. Acesso em: 05 jan. 2013.

ROYCE, WINSTON W. **Managing the Development of Large Software Systems**. In: Proceedings of IEEE WESCON. Piscataway, NJ, Estados Unidos: IEEE Press, ago. 1970, p. 1-9.

SCHWABER, KEN; SUTHERLAND JEFF. **Guia do Scrum,** 2011. Disponível em: <http://www.scrum.org/Portals/0/Documents/Scrum Guides/Scrum Guide - Portuguese BR.pdf>. Acesso em: 05 jan. 2013.

SOMMERVILLE, IAN. **Engenharia de Software**. 8ª. ed. São Paulo: Pearson Addison-Wesley, 2007.

STANDISH GROUP. **The Chaos Report on Project Management**. 2010.

TELES, VINÍCIUS MANHÃES. **Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade.** 1ª. ed. São Paulo: Novatec Editora Ltda, 2004.

WIEGERS, KARL EUGENE. **Software Requirements.** 2ª. ed. Redmond: Microsoft Press, 2003.

WILDT, DANIEL; LACERDA GUILHERME. **Conhecendo o Extreme Programming (XP).** In: Coletânea dos Trabalhos de CMP102 – Engenharia de Software 2010, PPGC-UFRGS, 31p.