UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA

Departamento de Ciências Exatas

Graduação em Ciência da Computação

Ferramenta para Gestão de Projetos Scrum em HTML5

Jorge Farias Herculano

Vitória da Conquista – BA
Setembro de 2011

Ferramenta para Gestão de Projetos Scrum em HTML5

Jorge Farias Herculano

Orientador: Prof. Dr. Fábio Moura Pereira

Trabalho Monográfico apresentado ao Departamento de Ciências Exatas – DCE – UESB – para obtenção do título Bacharel em Ciência da Computação.

Área de concentração: Engenharia de Software.

Vitória da Conquista – BA,
Setembro de 2011

JORGE FARIAS HERCULANO

Ferramenta para Gestão de Projetos Scrum em HTML5

Trabalho Monográfico aprovado como requisito parcial para obtenção do grau de bacharel, no curso de graduação em Ciência da Computação da Universidade Estadual do Sudoeste da Bahia.

Banca Examinadora

Orientador

_____

Prof. Dr. Fábio Moura Pereira

Universidade Estadual do Sudoeste da Bahia

Membros:

_____

Prof. Ms. José Carlos Martins Oliveira

Universidade Estadual do Sudoeste da Bahia

_____

Prof. Ms. Maísa Soares dos Santos Lopes

Universidade Estadual do Sudoeste da Bahia

Vitória da Conquista, BA
Setembro de 2011

A Deus por ter me criado na simplicidade e na ignorância e prover os caminhos e obstáculos necessários ao meu crescimento.

AGRADECIMENTOS

Este trabalho só foi possível devido o apoio encontrado nos amigos, colegas, professores e familiares.

A minha mãe por sempre estar ao meu lado nos momentos de angústia e *stress*, sempre me incentivando.

Ao meu pai (sempre presente) que sempre me apoiou e incentivou a continuar diante as dificuldades.

As minhas irmãs com quem aprendi a dividir e a compartilhar.

Aos meus sobrinhos que tanto amo.

Aos amigos e colegas, em especial a Diogo, Cátia, Zó, Railda, Kelly, Cris, Cássia, Déa, Stella Cristiane que formam a tão famosa “galera do busão”.

Aos ”*Power Rangers*” e a Marcos Santiago, meus grandes amigos, que sempre compartilharam comigo os melhores e os piores momentos na Universidade.

Aos professores, em especial Cátia Khouri e Fábio Moura, personalidades singulares a quem muito adimiro.

As amigas do colegiado, Celina e Valquíria, duas “criaturas” adoráveis e sempre dispostas a ajudar.

Aos familiares pelo apoio, principalmente a minha prima Jeane. Aos colegas de trabalho, em especial Neide e Jacy.

E Principalmente a **Deus**, por permitir a vida, os desafios e os obstáculos que nos fazem crescer.

Enfim, a todos que participaram comigo nessa fase de minha vida e que infelizmente não foram citados neste curto agradecimento.

RESUMO

As ferramentas de apoio aos métodos para desenvolvimento de *software* são de grande relevância para a área da Engenharia de *Software.* Na intenção de facilitar o processo de gestão dos projetos de software, ferramentas tem sido criadas para este fim. Este trabalho teve por objetivo o desenvolvimento de uma ferramenta para gestão projetos *Scrum* em HTML5, a partir da análise de métodos para desenvolvimento de software.

O sistema foi desenvolvido sob os conceitos de Rich Internet Application (RIA), que propõem um novo modelo de aplicativos Web com característica de aplicativos desktop. Para isso, foi utilizado a nova versão do HTML: o HTML5, um novo padrão de desenvolvimento de aplicações Web com recursos que dispensa a utilização de *plugins* para a execução das mesmas.

PALAVRAS-CHAVE: *Rich Internet Application* (RIA) , *Scrum*, HTML5.

ABSTRACT

The tools to support software development methods are of great relevance to the field of Software Engineering. In the intention to ease the process of project management software tools have been developed for this purpose. This study aimed to develop a tool for managing Scrum projects in HTML5, from the analysis methods for software development.

The system was developed under the concept of Rich Internet Application (RIA), which proposes a new model of Web applications with features of desktop applications. For this we used the new version of HTML: HTML 5, a new standard for Web application development with features that eliminates the need of plugins for their implementation.

KEYWORDS: *Rich Internet Application* (RIA) , *Scrum*, HTML5.

SUMÁRIO

Índice de ilustrações

1 INTRODUÇÃO

Este capítulo tem por objetivo apresentar o contexto deste trabalho, bem como os objetivos, a motivação e a metodologia utilizada.

1.1 CONTEXTO DA PESQUISA

Segundo Smeets, Boness e Bankras (2009), os sistemas de software estão inseridos no nosso cotidiano há várias décadas, mas só atualmente pode-se afirmar que estes estão sendo utilizados por milhões de pessoas pelo mundo. Isto deve-se ao fato de que, até então, tais sistemas eram utilizados somente por profissionais altamente treinados. No entanto, com a popularização de sistemas computacionais, o número de usuários cresceu e houve necessidade de mudanças nos tipos de *software* para atender a nova clientela no mercado.

Dentre estes novos tipos de software, os que estão em constante crescimento são as Aplicações *Web.* Segundo a Wikipédia (2011), “Aplicação *Web* é o termo utilizado para designar, de forma geral, sistemas de informática projetados para utilização através de um navegador, na Internet ou em redes privadas (Intranet)”. Estes sistemas são hospedados em servidores e executados em um navegador.

De toda forma, criar e manter o software seja ele d*esktop* ou uma aplicação W*eb* é uma atividade árdua e difícil devido à complexidade e constante mudança de requisitos. Como solução, a engenharia de software entra em cena como meio de minimizar essa complexidade. Apesar de ainda ser muito nova comparada à engenharia civil, que tem mais de três mil anos, ela fornece subsídios que auxiliam no processo de criação de *software.*

Atualmente há um grande número de métodos de desenvolvimento voltadas à construção de *software,* e estes métodos estão divididos em dois grupos: os baseados em modelos, como o *Rational Unified Process* (*RUP*) e os métodos ágeis, como o *Extreme Programming* e o *Scrum*.

Sistemas para gestão de projetos baseadas nestes métodos são ferramentas indispensáveis no auxílio às empresas de desenvolvimento de software.

Principalmente, no que concerne às ferramentas Web, já que estas possibilitam o controle dos projetos através da Internet.

No entanto, tais aplicações devem possuir as mesmas características de um sistema *desktop* no quesito interface, o que não é possível utilizando apenas os padrões HTML (*HyperText Markup Language*). Para solucionar este tipo de problema, surge o conceito de *RIA* (*Rich Internet Application* – Aplicações de Internet Rica) que fornece meios de desenvolvimento de aplicações Web com as características de uma aplicação *desktop.*

O HTML5 é a mais recente especificação do HTML para o desenvolvimento de RIA, proporcionando aos sistemas Web as principais características dos sistemas desktop.

1.2 MOTIVAÇÃO

A gestão de projetos de software necessita de uma método para que as empresas possam garantir o desenvolvimento de produtos de qualidade. Os métodos ágeis estão cada vez mais presentes nestas organizações como principal ferramente de apoio.

Com a globalização da informação e a contribuição da Internet para a mesma, vem crescendo a necessidade da concepção de sistemas que possam ser acessados remotamente, possibilitando aos empresários gerenciar seus projetos de *software* sem a necessidade de estarem no mesmo local de execução do mesmo.

Há um grande número de ferramentas voltadas para gestão de projetos. No entanto, tais ferramenta quando não são aplicações desktop, exigem que *plugins* sejam instalados para serem executados nos navegadores do cliente, ou não atendem todos os requisitos dos métodos de desenvolvimento.

A motivação deste trabalho está na necessidade de uma ferramenta Web baseada em um método ágil para gerenciamento de projetos de software e que possua as principais características de um sistema desktop, sem a necessidade do uso de *plugins* e que atenda aos requisitos do método escolhido.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Este trabalho tem por objetivo o desenvolvimento de uma ferramenta Web RIA para gestão de projetos de software que fazem uso de um método ágil de desenvolvimento.

1.3.2 Objetivos específicos

- Analise dos métodos ágeis de desenvolvimento de *software*;
- Analise das tecnologias que permitem o desenvolvimento de Aplicações de Internet Rica, utilizando uma que forneça melhor desempenho e baixo custo financeiro;
- Construir uma ferramenta Web de gerenciamento de projeto baseada no método escolhido.

1.4 METODOLOGIA

Para o desenvolvimento dessa monografia foi utilizado o método indutivo com os seguintes passos:

1. Estudo dos métodos de desenvolvimento de *software;*
2. Com base nos estudos levantados, foi escolhido um método para servir de modelo para o desenvolvimento da ferramenta.
3. Estudo e análise das tecnologias utilizadas no desenvolvimento de RIA;
4. Desenvolvimento de uma ferramenta para o método escolhido utilizando a tecnologia que forneça maior desempenho e baixo custo.

1.5 ORGANIZAÇÃO DA MONOGRAFIA

Esta monografia possui outros 4 capítulos organizados da seguinte forma: o capítulo 2 apresenta os principais métodos de desenvolvimento de *software*; o capítulo 3 apresenta os conceitos e tecnologias de RIA; o capítulo 4 apresenta um estudo de caso, a ferramenta para gestão de projetos; e por fim no capítulo 5 são apresentadas as considerações finais e sugestões para trabalhos futuros.

2 METODOLOGIAS PARA DESENVOLVIMENTO DE SOFTWARE

Este capítulo tem por objetivo apresentar e analisar os principais métodos para desenvolvimento de *software*.

2.1 CONTEXTUALIZAÇÃO

A falta de organização no processo de desenvolvimento podem fazer com que projetos de *software* sejam abandonados antes do seu término ou, quando concluídos, extrapolem prazos e custos. É de senso comum pensar que *software* é apenas um amontoado de código e a falta de um método no processo de desenvolvimento tem levado muitas empresas de produção de *software* a abandonarem projetos depois de ter havido grandes investimentos.

Para o bom desenvolvimento de *software* se faz necessário a escolha de um método de desenvolvimento que se adeque ao escopo do projeto. As seções seguintes apresentam alguns desses métodos, enfatizando as suas características, dividindo-os em métodos baseados em modelos e métodos ágeis.

2.2 MÉTODO BASEADO EM MODELO: *RATIONAL UNIFIED PROCESS*

O *Rational Unified Process* (RUP), é um método de processo de software criado pela *Rational Software Corporation*, atualmente uma divisão da IBM[1] - *International Business Machines*. Este processo usa a UML (*Unified Modeling Language*) como notação para projeto e documentação do sistema. Além de possuir um ciclo de vida iterativo, seu objetivo é garantir que projetos não pereçam diante a variação do escopo e à falta de planejamento, fornecendo um *software* de qualidade, que atendendo os requisitos do cliente não ultrapasse os prazos e os custos previstos.

A arquitetura geral do RUP é apresentada na Figura 1. Observe que o RUP possui duas dimensões: a horizontal, que representa o aspecto dinâmico do

1 http://www-01.ibm.com/software/awdtools/rup/

processo, e a vertical, que representa o aspecto estático.

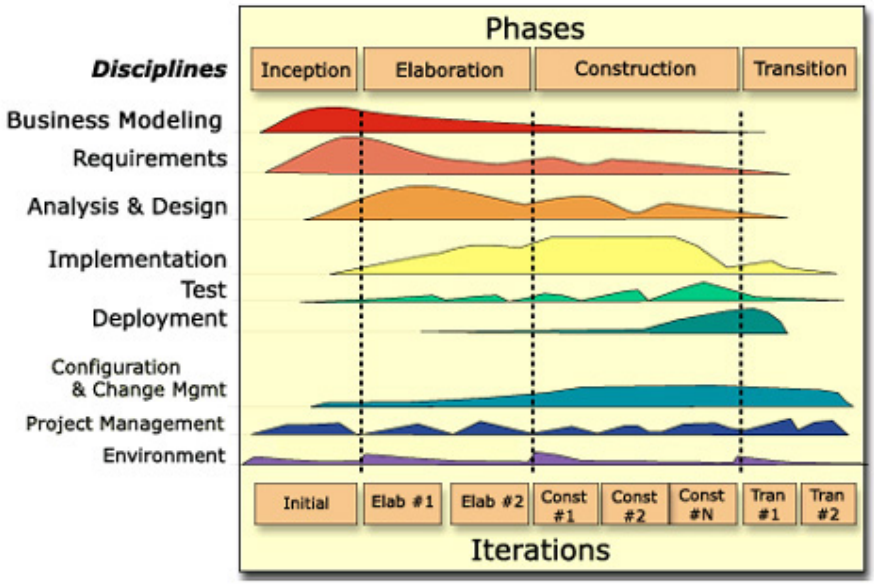

Figura 1: Arquitetura geral do RUP (AKED, 2003)

O aspecto dinâmico do processo é definido em termos conceituais de:

– **Fases*:*** O RUP é composta por quatro fases (concepção ou iniciação, elaboração, construção e transição) que diferentemente do fluxo tradicional de desenvolvimento não representa uma execução sequencial e estática que vai desde a análise de requisitos aos teste. Neste sentido, de acordo com Kruchten (2003, p. 53),

> As quatro fases (C, E, C e T) constituem um *ciclo* de desenvolvimento e produzem uma geração de software. Um produto de software é criado em um *ciclo de desenvolvimento* inicial. A menos que o ciclo de vida do produto pare neste momento, um produto evoluirá em sua próxima geração por uma repetição da sequência das fases concepção, elaboração, construção e transição, mas com ênfase diferentes nas várias fases.

– **Marcos:** Representam a conclusão de uma fase, e por sua vez é o subconjunto de um produto finalizado. Este subconjunto pode ser um documento ou

uma unidade de *software*, por exemplo uma versão beta.

– **Iterações:** Como um processo iterativo, cada fase do RUP é composta por uma ou mais iterações e ao final de cada iteração é feito um acompanhamento do progresso do projeto.

A Tabela 1 descreve as fases do RUP e seus respectivos marcos, dando-nos uma ideia do ciclo de vida do método.

Tabela 1: Fases e Marcos do RUP (CARVALHO, 2006)

Fase	Descrição	Marco
Iniciação	Essa fase tem por objetivo um acordo com todos os envolvidos no processo quanto ao entendimento dos requisitos de forma generalizada, delimitando o escopo do projeto, a estimação de custo e riscos, e as condições limites e prazos.	O marco dessa fase é o Objetivo do Ciclo de Vida (*Life-Cycle Objective*), onde se encontram especificados os prazos, custos e o escopo do projeto
Elaboração	Na fase de elaboração é feita a captura dos requisitos ainda não identificados, além de definir uma arquitetura estável para as próximas fases.	O marco dessa fase é a Arquitetura do Ciclo de Vida (*Life-Cycle Architecture*), onde o escopo e a arquitetura do sistema são detalhados.
Construção	Esta fase é considerada como um processo de manufatura. E tem por objetivos: construir e testar o sistema em incrementos, desenvolvimento completo dos componentes, evoluir a visão, a arquitetura e o plano do projeto, minimizar os custos e o prazo, desenvolver versões úteis (alfa, beta e outras versões para testes).	O marco dessa fase é a Capacidade Operacional Inicial (*Initial Operational Capability*), que consiste de uma versão executável do software.
Transição	É nessa fase que se transmite o produto para os usuários, pelos quais uma avaliação é feita.	O marco dessa fase é a *Release* do Produto (*Product Release*).

O aspecto estático do RUP pode ser descrito em termos de:

– **Disciplinas:** é um conjunto de atividades para produzir um resultado. O RUP possui 6 disciplinas de engenharia: Modelagem de Negócio, Requisitos, Análise e Projeto, Implementação, Testes e Implantação; e 3 disciplinas de suporte: Gerenciamento do Projeto, Ambiente e Gerenciamento de Configuração e Mudança.

– **Artefatos:** Os artefatos são informações produzidas ou usadas durante um projeto. Pode ser um modelo da UML, um documento contendo a especificação do projeto ou arquitetura, ou um elemento do modelo, por exemplo, um subsistema

ou classe.

– **Papéis:** São as atribuições deferidas às pessoas envolvidas durante o processo.

– **Atividades:** São um conjunto de tarefas realizados em cada disciplina. Uma atividade é realizada por um papel específico.

O RUP parte do princípio de que o sucesso do projeto encontra-se na gerência e no planejamento detalhado do mesmo. Sendo aconselhado para projetos com escopo definido, mas sem um nível de detalhamento elevado e, que necessite de um time grande para desenvolvimento, pois formalidade é um traço marcante do método.

2.3 MÉTODOS BASEADOS NO CONCEITO ÁGIL

Em 2001, um grupo de profissionais da engenharia de s*oftware*, interessados nos vários relatos de equipes de desenvolvimento que se encontravam presos por processos de desenvolvimento pesados e cheios de burocracia, reuniram-se e deram início ao Manifesto Ágil[2]. Para Abrahamsson (2002), um método pode ser considerado ágil quando possibilita o desenvolvimento de *software,* de forma, incremental, colaborativa, direta e adaptativa.

Não diferentemente desse conceito, Pressman (2006, p. 58) afirma que,

> A engenharia de software ágil combina uma filosofia e um conjunto de diretrizes de desenvolvimento. A filosofia encoraja a sofisticação do cliente e entrega incremental do software logo de início; equipes de projeto pequenas, altamente motivadas; métodos informais; produtos de trabalho de engenharia de software mínimos e simplicidade global do desenvolvimento. As diretrizes de desenvolvimento enfatizam a entrega em contraposição à análise e ao projeto (apesar dessas atividades não serem desencorajadas) e a comunicação ativa e contínua entre desenvolvedores e clientes.

Baseados nessa filosofia, existem vários métodos, dentre os quais os mais populares são: *Scrum*, *Extreme Programming* e *Feature Driver Development.* Estes

2 http://www.agilemanifesto.org

métodos são apresentados nas subseções seguintes.

2.3.1 *Extreme Programming*

Dentre os métodos ágeis, o *Extreme Programming* (XP) é o que mais se destaca. Sua popularização se deve ao fato de maximizar a participação do cliente no processo gerando rápidos *feedbacks* e garantindo a satisfação do cliente.

O XP é baseado em **valores** e **práticas,** e cada pessoa envolvida no processo representa um **papel**.

Segundo Martins (2007), o XP é uma filosofia de desenvolvimento que tem base fundamentada nos seguintes valores: comunicação, *feedback*, simplicidade, coragem e respeito. Além de possuir um conjunto de práticas que norteiam o desenvolvimento do *software*.

Os valores que o XP agrega são:

– **Comunicação**: Existem várias formas de comunicar-se em uma equipe de desenvolvimento. O XP dá preferência à comunicação falada ao invés da comunicação formal escrita. Acredita-se que com este tipo de comunicação a interação entre os membros da equipe é maior e o *feedback* é mais rápido. Isto não quer dizer que o XP utiliza apenas a coamunicação falada, mas sim que ela é a mais utilizada.

– **Simplicidade**: A equipe de projeto concentra-se na implementação de soluções simples e que funcionem. Ou seja, o projeto é implementado em partes pequenas de forma simples e de fácil documentação.

– ***Feedback***: Um dos valores mais importantes do XP, pois é através de *feedbacks* que o cliente aprende sobre o sistema. Martins (2007, pág. 282) acrescenta que “As equipes devem tentar gerar tanto *feedback* quanto elas puderem, e o mais rápido possível”. Quanto maior a quantidade de *feedback* produzido, maior será a adaptabilidade do produto.

– **Coragem**: Por ser um método novo, a coragem é fundamental nas práticas do XP. Faz-se necessário coragem para desenvolver o software de forma simples e incremental, coragem para comunicar, etc. Porém, para Martins (2007, p. 283), “(...) a coragem sozinha pode ser perigosa. Fazer as coisas sem uma

avaliação das consequências vai contra o trabalho de equipe".

– **Respeito:** Segundo o *Improve It* (2011), o repeito é necessário para que os membros da equipe comuniquem-se melhor, importando uns com os outros. Além do mais, compreender e respeitar a opinião alheia faz com que o sucesso do projeto seja alcançado.

Além dos valores apresentados acima, o XP possui um conjunto de práticas para garanti-los. Dentre estas práticas podem se destacar:

– **O cliente presente:** O XP prega que colocar o cliente junto aos desenvolvedores durante o processo de desenvolvimento facilita a compreensão de requisitos e simplifica o processo.

– **As metáforas:** São aspectos do sistema descritos numa linguagem mais compreensível. O XP procura utilizar ao máximo as metáforas para facilitar a comunicação entre os desenvolvedores e os clientes.

– **O jogo do planejamento:** É uma reunião realizada no início de cada ciclo semanal, onde as estórias[3], são estimadas pelos desenvolvedores e colocadas em cartões para serem fixadas em mural para serem acompanhadas. As estórias são funcionalidades do sistema escritas de forma simples e objetiva.

– **O s*tand up meeting*:** É uma reunião rápida, de aproximadamente dez minutos, que inicia o dia de trabalho e são discutidas as atividades realizadas no dia anterior, bem como o andamento do projeto.

– **A programação em par:** Umas das práticas mais populares do XP que sugere que um par de programadores sentem em um computador e fiquem alternando na codificação, facilitando assim a detecção e correção de erros.

Outras práticas do XP são: desenvolvimento guiado pelos testes, refatoração, código coletivo, código padronizado, integração contínua, entre outras práticas.

O XP prevê um time de pessoas para a execução das práticas e manter os valores. Cada pessoa envolvida no processo desempenha um dos papéis descritos a seguir:

– **Analista de testes:** atua junto ao cliente na escrita de casos de testes para as estórias antes de suas implementações. É responsável pela orientação do programador a respeito da automatização dos testes.

3 Descrição das funcionalidades do sistema em uma linguagem mais simples e próxima do cliente (TELES, 2006).

– **Gerentes de projeto:** têm por objetivo facilitar a comunicação entre todos os envolvidos no projeto guiando a equipe para que construa um projeto de alta qualidade, e mantendo todos informados a respeito do andamento do projeto.

– ***Coach*:** possui total conhecimento do método e tem por objetivo guiar a equipe de desenvolvimento nas práticas do XP.

– **Redator técnico**: É o responsável por atuar juntamente com equipe na redação na documentação técnica do projeto.

– **Desenvolvedor:** é o responsável pela análise, projeto e codificação do sistema.

O XP é fortemente marcado pelo foco nos testes, facilitando a prevenção de erros, além de ser um método mais “leve”, pois não utiliza muitos artefatos, ao contrário do RUP, por exemplo. Porém, pode ser utilizado juntamente com o mesmo, tornando o método mais preditivo.

2.3.2 *Feature Driven Development*

O *Feature Driven Development*[4] (FDD), é um método ágil para gerenciamento e desenvolvimento de *software*, criado por Peter Coad, Eric Lefebvre e Jeff De Luca, em 1997 em Singapura, mas só foi publicado em 1999 no livro *Java Modeling in Color with UML*, dos mesmos autores. O nome *Feature Driven Development* significa, em Português, Desenvolvimento Dirigido por Funcionalidades. Uma funcionalidade (*feature*) é uma característica do *software* que possua algum valor para o cliente e que possa ser implementada em um curto espaço de tempo, por volta de duas semanas.

Segundo o Heptagon (2011), O FDD tornou-se popular devido algumas características, tais com: resultados úteis a cada duas semanas, funcionalidades em blocos pequenos valorizadas pelo cliente, planejamento detalhado, precisão nos relatórios, etc.

A arquitetura do FDD é simples e objetiva, sendo composta por apenas duas fases: a concepção e planejamento e a construção, como apresentado na Figura 2.

4 http://www.featuredrivendevelopment.com

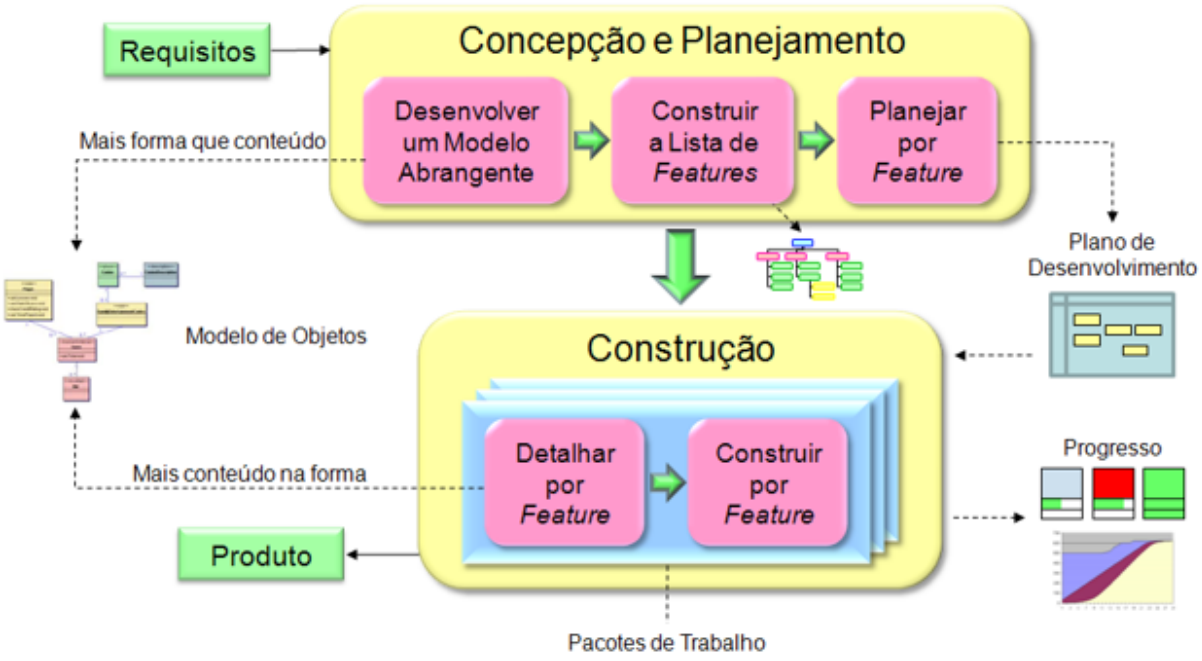

Figura 2: Arquitetura do FDD (HEPTAGON, 2011)

A primeira fase, segundo Pimentel (2010), "(..) atua com o objetivo de proporcionar um entendimento inicial e abrangente do escopo, bem como, possibilitar um planejamento macro de entregas durante as iterações". Como se observa na Figura 2 é nessa fase que acontecem os seguintes processos:

– **Desenvolver um Modelo Abrangente**: nesse processo inicial é feito um estudo acerca do escopo do sistema, dividindo-o por áreas de negócio, e a equipe é subdivida em grupos nessas áreas. Cada grupo modelará cada área e depois estes modelos serão integrados, fazendo ajustes caso necessário.

– **Construir a Lista de *Features***: Cada área modelada no processo anterior é decomposta, formando assim uma lista com as funcionalidades do modelo. Estas funcionalidades, por sua vez, são categorizadas em ordem de prioridade na lista.

– **Planejar por *Feature***: é feito um planejamento sobre a ordem em que as funcionalidades serão implementadas, priorizando as dependências entre elas, carga de trabalho da equipe e a complexidade das funcionalidades.

A segunda fase do FDD, segundo Pimentel (2010), "(...) o time poderá focar

de maneira iterativa no desenvolvimento e entrega das funcionalidades da FBS(*Feature Breakdown Structure*)". Esta fase é composta por dois processos:

– **Detalhar por *Feature***: nesse processo cada funcionalidade é modelada de acordo com a arquitetura do produto. A equipe, segundo Pimentel (2010), "(...) poderá prototipar telas, esboçar ou especificar diagramas de classes, de sequência ou qualquer outra ferramenta de modelagem."

– **Construir por *Feature***: nesse processo é onde acontece a escrita do código de cada funcionalidade pela equipe.

O FDD mantém, além de fases e processos, um conjunto de práticas e papéis, mas não serão discutidos aqui, por não fazerem parte do escopo deste trabalho.

2.3.3 *Scrum*

O *Scrum* é um método ágil para gerenciamento e planejamento de projetos de *software*. Este método pressupõe que o time de desenvolvimento seja auto-organizado e auto dirigido, ou seja, a própria equipe que define a maneira pela qual desenvolverá o projeto. A equipe utiliza os princípios do *Scrum* que s*egundo* Pressman (2006, p. 69), são "(..)usados para guiar as atividades de desenvolvimento dentro de um processo que incorpora as seguintes atividades de arcabouço: requisitos, análise, projeto, evolução e entrega".

Analisando a Figura 3, pode-se definir o ciclo de vida do *Scrum* em três fases: preparação, desenvolvimento e distribuição.

O processo do *Scrum* se inicia na fase de preparação, na qual se faz o planejamento e é definida a arquitetura para a construção do *software*. Esta arquitetura é definida em termos de estórias ou itens, que comporão o ***product backlog***. As estórias ou itens poderão ser subdivididos em uma ou mais tarefas.

A fase de Desenvolvimento consiste no desenvolvimento das estórias detalhadas no *product backlog* objetivando a concepção do produto. É composta por um conjunto de práticas que serão descritas mais a frente.

A fase de Distribuição começa quando o produto alcança maturidade para ser lançado.

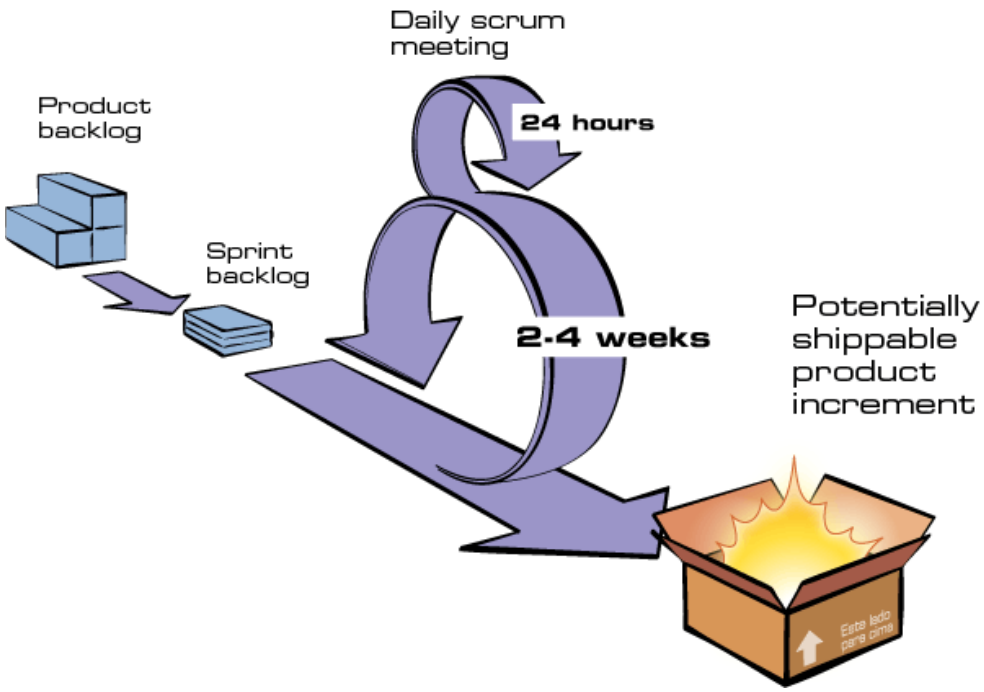

Figura 3: Fluxo do *Scrum* (IMPROVE IT, 2011).

O *product backlog* é um documento contendo os requisitos iniciais do sistema, ou seja, neste artefato é descrito tudo o que o cliente deseja que seja implementado. Segundo Kniberg (2007) o *product backlog* possui as seguintes informações:

– **Id**: identificador de um item, deve ser único e auto incrementado.

– **Nome**: nome para um item do *backlog*, de preferencia que seja curto para facilitar a visualização.

– **Importância**: a importância que o item tem para o *product owner*. Quanto mais alto, maior a importância.

– **Estimativa inicial**: a equipe faz uma estimativa de quanto tempo levará para implementar cada item. Esta estima é a soma das estimativas de cada tarefa na qual o item foi subdividido.

– **Como demonstrar**: uma descrição em linguagem informal de como poderá ser executado aquele item depois de implementado.

– **Notas**: são utilizadas para maiores esclarecimentos sobre os itens.

Na Figura 4 tem-se um exemplo de um *product backlog*:

PRODUCT BACKLOG (exemplo)					
ID	**Nome**	**Imp**	**Est**	**Como demonstrar**	**Notas**
1	Depósito	30	5	Logar-se, abrir a página de depósito, depositar R$ 10,00, ir para a página do meu saldo e verificar que este aumentou em R$ 10,00.	Precisa de uma diagrama UML de sequência. Não é necessário se preocupar com criptografia por enquanto.
2	Verificar seu próprio histórico de transações	10	8	Logar-se, clicar em "transações". Fazer um depósito. Voltar para transações, verificar se o novo depósito é listado.	Usar paginação para evitar consultas muito grandes ao banco de dados. Projetar de forma similar à página de visualização de usuários.

Figura 4: Exemplo de um Product Backlog (KINIBERG, 2007. p. 10)

No *Scrum* existe um conjunto de práticas e papéis. Os papéis são:

– ***Product Owner*****:** este é o dono do produto e é responsável por apresentar os itens do *product backlog*, bem como as prioridades destes itens.

– ***Scrum Master*****:** ou Mestre *Scrum*, é responsável por manter os valores e as práticas do *Scrum* dentro da equipe. Tem papel similar ao *Coach* do *Extreme Programming*.

– ***Team Members*****:** compõem a equipe de desenvolvimento do projeto. Como mencionado anteriormente a equipe do *Scrum* é auto-dirigida, auto-organizada e multifuncional, ou seja, neste ultimo caso os membros da equipe executam diferentes atividades durante o processo.

Além do *product backlog* o *Scrum* mantém alguns artefatos que são utilizado para fim de documentação e organização do processo. Estes artefatos são produzidos durante a execução do processo os quais pode-se destacar:

– ***Product Backlog Burndown***: contém dados do que já foi produzido do

projeto, dando uma visão do seu *status*.

– ***Sprint Backlog**:* contém uma lista de itens retirados do backlog do produto para serem implementados em um *sprint.*

– ***Sprint Backlog Burndown***: contém informações sobre o que já foi implementado em um *sprint* específico.

– ***Impediment List**:* Lista contendo as dificuldades e impedimentos encontrados durante o *sprint*.

As práticas do *Scrum* são:

– **Plano de Jogo:** É aqui que é feito o preenchimento do *product backlog* com as funcionalidades que o sistema irá adquirir. Nessa reunião de aproximadamente oito horas todos os envolvidos no projeto participam na definição do escopo do projeto e estimativas de custos e prazos.

– ***Sprint*:** De acordo com *Improve It* (2011) o *Sprint* é um espaço de tempo que varia de duas a quatro semanas em que um conjunto de funcionalidades do produto é implementado. Cada *Sprint* é uma iteração do produto.

– **Planejamento do Sprint:** São reuniões que antecedem cada *Sprint*. Nessas reuniões são discutidas as funcionalidades que serão implementadas no próximo *Sprint.* O *sprint backlog* é preenchido com os itens com maiores prioridades definidos pela equipe.

– **Equipes auto-organizadas e auto dirigidas:** Uma característica de um projeto *Scrum* é que a equipe define a forma de trabalhar em cada *Sprint*. Esta característica torna a equipe auto dirigida. Quando os membros da equipe passam a ter uma visão comum dos objetivos a serem alcançados, ela desperta a auto-organização e a capacidade de decompor problemas para que possam solucioná-los.

– **Reunião diária do *Scrum*:** São reuniões curtas de mais ou menos 10 minutos realizadas antes de começar o dia de trabalho. Nelas são discutidas o que foi feito no dia anterior priorizando o que será feito no dia que se inicia.

– **Equipes de até 7 pessoas:** o processo fluirá com mais naturalidade quando as atividades forem executadas em equipe. O *Scrum* prevê um número de

até 7 membros por equipe, mas este poderá ser maior, tudo dependerá da necessidade do projeto.

– **Reunião de revisão do *Sprint*:** Nessa reunião os membros da equipe mostra ao dono do produto o que foi implementado durante o Sprint.

– **Reunião de retrospectiva do *Sprint***: Segundo Martins (2007, pág. 277) ”Nesta reunião, o Mestre *Scrum* encoraja a Equipe a revisar o seu processo de trabalho, tendo em vista o framework e as práticas do *Scrum*, para ter um melhor desempenho na próxima iteração.”

Como pôde ser observado, o *Scrum* é um método leve comparado ao RUP ou ao XP. Possui poucas práticas e artefatos. Portanto, a equipe de projeto se sente mais à vontade em relação à forma de trabalho, gerando maior produtividade.

3 RICH INTERNET APPLICATION

Este capítulo objetiva mostrar os conceitos do HTML5, bem como os conceitos da Rich Internet Application (Aplicação de Internet Rica), e as suas linguagens de suporte.

3.1 RICH INTERNET APPLICATION – RIA

Antes de contar com as interfaces arrojadas e criativas, os usuários de computadores utilizavam sistemas baseados em modo de comando, que geralmente eram terminais com fundo preto, azul ou verde e os comandos a serem executados tinham que ser digitados. A Figura 5 mostra a tela do *Sensible Cinema Software*, um programa para gerenciamento de vídeo locadoras.

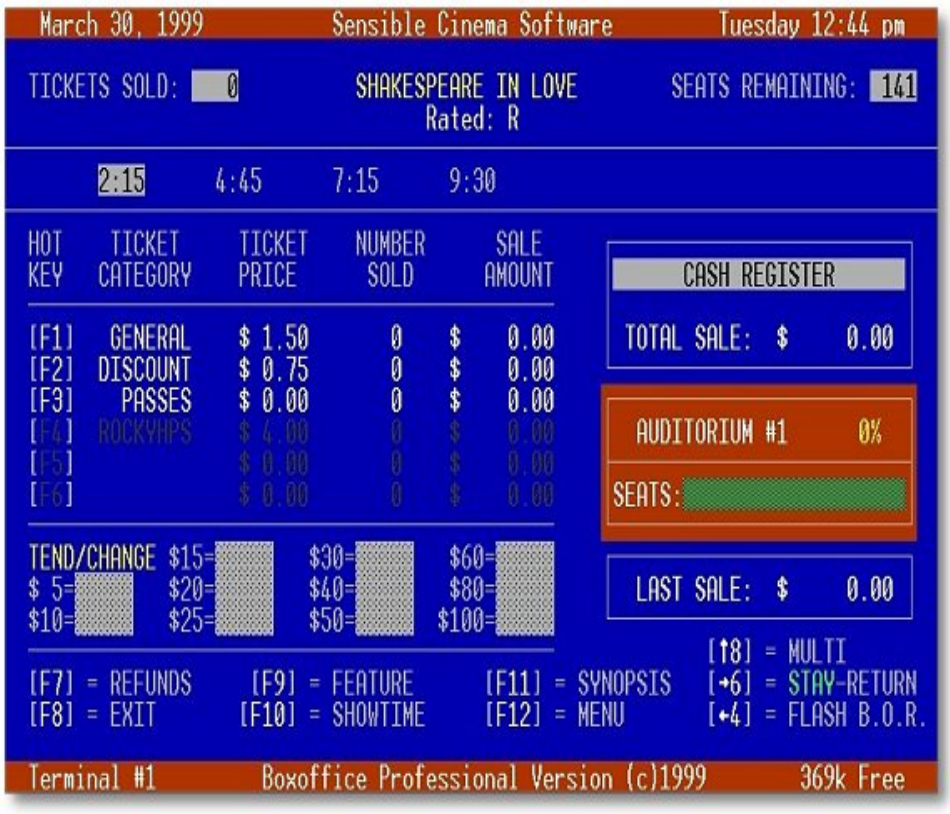

Figura 5: Exemplo de aplicação em modo de comando.(SENSIBLE CINEMA SOFTWARE, 1999)

Até pouco tempo atrás os sistemas de software eram restritos a aplicações do tipo *desktop*, ou seja, aquelas aplicações que necessitam ser instaladas no computador para então serem utilizadas pelos usuários. A Figura 6 mostra uma tela de cadastro de empresa de uma aplicação *desktop* de controle de estoque.

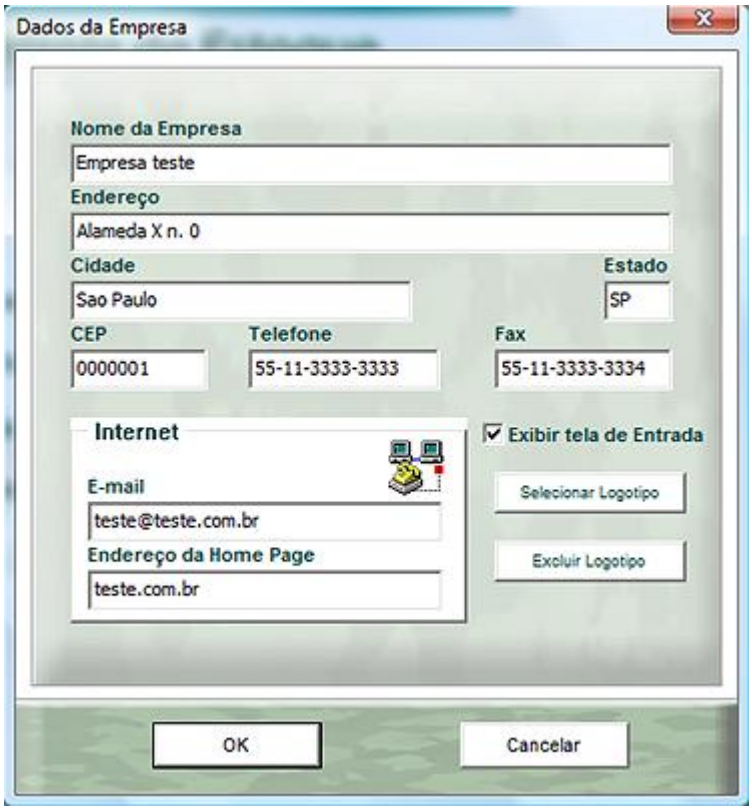

Figura 6: Exemplo de uma aplicação *desktop* (INFO EXAME, 2009)

Com o advento da Internet o campo de possibilidades de atuação do software aumentou consideravelmente. Hoje já é possível realizar tarefas em sistemas Web que até pouco tempo tempo atrás era restrito a *software desktop*.

Contudo, os sistemas Web tradicionais, segundo a Wikipédia (2011), "(...)centralizam todo o seu código em torno de uma arquitetura Cliente-Servidor e um cliente magro". Isto quer dizer que, todo o processamento dos dados era até então, realizado pelo servidor, enquanto que o cliente (*browser*) fica apenas responsável em apresentar estaticamente os dados ao usuário.

Diante disto houve a necessidade de tornar as interfaces de aplicações Web mais dinâmicas e que pudessem realizar tarefas até então inerentes a aplicações do

tipo *desktop.*

Segundo Staley (2007), em 2002 a Adobe utilizou o termo “Rich Internet Application” quando apresentou uma grande atualização da *Adobe Flash Platform.* Desde então o número de sites que passaram a utilizar esta plataforma cresceu enormemente.

As RIA's transferem para os clientes a tarefa de processamento da interface, enquanto que o servidor mantém a maioria dos dados. Desta forma, possibilita que características de software do tipo *desktop* (*drag-and-drop*, alta interatividade, robustez) possam estar incluídos nestas aplicações além de manter a portabilidade e as características de uma aplicação Web tradicional.

As aplicações Web típicas são do tipo solicitação-resposta, ou seja, o modelo de interação é síncrono. Como mostra a Figura 7, a interação nesse tipo de aplicação começa com o usuário realizando solicitação em uma página Web, esta solicitação é enviada ao servidor que processará a informação e então retornará o resultado para o usuário, seguindo assim com o fluxo de informação.

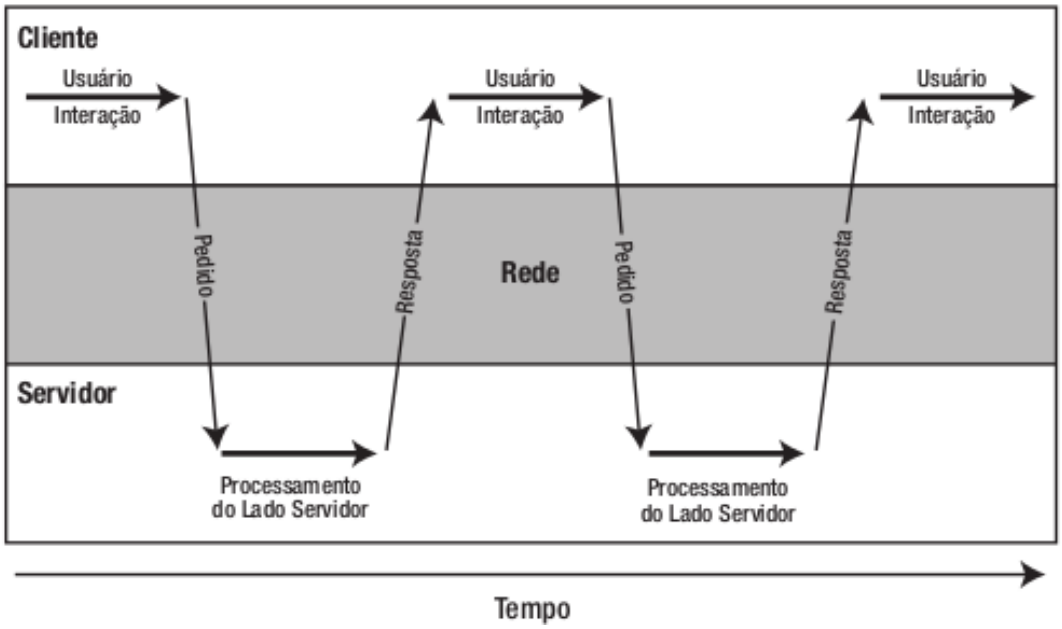

Figura 7: Modelo de interação de uma aplicação web típica.(SMEETS, BONESS E BANKRAS, 2009)

As RIA's por outro lado utilizam o modelo de interação assíncrona, ou seja, o usuário não necessita mais esperar que o servidor termine o processamento dos

dados para que ele continue utilizando a aplicação porque as solicitação ao servidor são feitas ao servidor de modo implícito sem a necessidade de atualização de toda a página.

Uma forma de conseguir esse modelo é utilizando o AJAX, um acrônimo para *Asynchronous Javascript and XML*[5]. Segundo o W3School (2011), o AJAX permite que páginas Web sejam atualizadas de forma assíncrona através da troca de pequenas quantidades de dados com o servidor nos bastidores. Isto significa que é possível atualizar partes de uma página Web, sem recarregar a página inteira. A Figura 8 ilustra o modelo de comunicação assíncrono com AJAX.

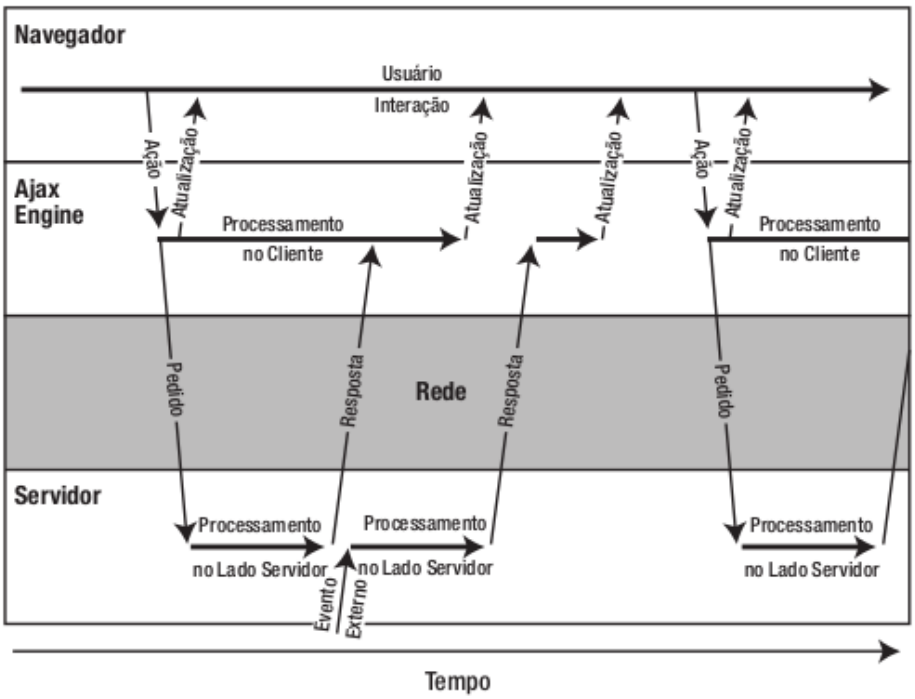

Figura 8: Modelo de interação com AJAX.(SMEETS, BONESS E BANKRAS, 2009)

Com isto, como mostra a Figura 9, pode-se afirmar que as RIA's combinam a riqueza das aplicações desktop como a robustez e alta interatividade com a conectividade e disponibilidade das aplicações web.

5 http://www.ajaxprojects.com/ajax/

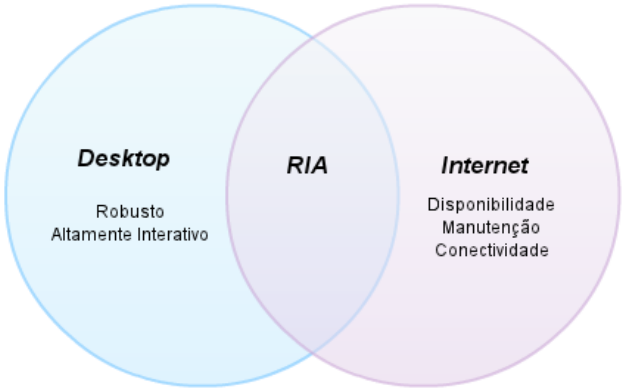

Figura 9: Rich Internet Application (BRUST, 2007)

A Tabela 2 faz uma comparação entre as aplicações *desktop*, Web tradicionais e RIA.

Tabela 2: Comparação entre aplicações *Desktop*, Web e RIA (BOZZON et al 2006).

Característica	*Desktop*	Web	RIA
Cliente universal (*browser*)	Não	Sim	Sim
Instalação do Cliente	Complexo	Simples	Simples
Capacidade de Interação	Rica	Limitada	Rica
Lógica de negócios ao lado do servidor	Sim	Sim	Sim
Lógica de negócios ao lado do cliente	Sim	Limitada	Sim
Requisição de atualização de toda a página	Não	Sim	Não
Servidor com *round-trips* frequentes	Não	Sim	Não
Comunicação Cliente-Servidor	Sim	Não	Sim
Funciona desconectado	Sim	Não	Sim

Em suma pode-se definir que as RIA's objetivam uma junção das principais características das aplicações *desktop* com as características das aplicações Web.

3.2 LINGUAGENS DE SUPORTE A RIA

Nas subseções seguintes serão tratadas as principais linguagens que dão suporte ao desenvolvimento de aplicações de Internet rica.

3.2.1 Adobe Flash

De acordo com a Adobe Systems Incorporated (2011),

> A plataforma Adobe Flash é um conjunto integrado de tecnologias de programação de aplicativo envolvidas por um ecossistema estabelecido de programas de suporte, parceiros comerciais e comunidades de usuários interessados. Juntas, elas fornecem tudo que é necessário para criar e fornecer aplicativos, conteúdos e vídeos mais atrativos para o mais amplo público possível.

A plataforma flash utiliza como linguagem de script o ActionScript que permite gerar interações da aplicação com o usuário de forma assíncrona, ou seja, o usuário não necessita esperar que o servidor termine o processamento dos dados para que que ele possa realizar outra tarefa. Para que aplicativos desenvolvidos em Flash sejam executados, faz-se necessário a instalação no navegador de um *plugin* chamado *Adobe Flash Player*.

3.2.2 Flex

De acordo com a *Adobe Systems Incorporated* (2011),

> O Flex é uma estrutura altamente produtiva, gratuita e de código aberto para a criação de aplicativos expressivos móveis, da Web e para desktop. Permite que você crie aplicativos móveis e da Web que compartilham uma base de código comum, reduzindo o tempo e custo da criação de aplicativos e a manutenção de longo prazo.

O Flex utiliza o MXML, uma linguagem baseada no XML e é utilizada principalmente para arranjar declarativamente a interface das aplicações. Além disso, conta com uma biblioteca de componentes com mais de 100 componentes de interface.

3.2.3 Silverlight

Para competir com o *Adobe Flash* a *Microsoft* criou uma plataforma para

aplicativos RIA chamada se *Silverlight*. Segundo a Microsoft (2011), “O Silverlight é uma poderosa plataforma de desenvolvimento para a criação de experiências de usuário interativas e atraentes para a Web, estações de trabalho e dispositivos móveis, online ou offline”.

Utiliza como linguagem o XAML (*eXtensible Application Markup Language*), uma linguagem declarativa baseada no XML, sendo a principal linguagem de interface da *Microsoft*. Uma das característica do *Silverlight*, segundo a Wikipédia (2011), “é a de poder criar e trabalhar com gráficos vetorizados, assim como textos, animações e sobreposições que interagem com gráficos e efeitos de alta qualidade”.

3.2.4 *Applets Java* e *JavaFX*

Applets Java são códigos java executados em página HTML que tem como responsável a *Oracle*[6], é uma outra abordagem no desenvolvimento de aplicações de *internet* rica. Segundo a Oracle (2010), o *applet* é um programa escrito na linguagem de programação Java que pode ser incluído em uma página HTML. Quando uma página que contém um *applet* é carregada pelo browser, o código Java é transferido para o sistema e executado pela máquina virtual do Java (Java Virtual Machine – JVM).

Recentemente foi criado o *JavaFX*, que segundo Castillo (2011), é a evolução da plataforma do cliente *Java* projetada para permitir que desenvolvedores de aplicativos possam criar e implementar facilmente aplicações de Internet rica (RIAs) que se comportam de forma consistente em múltiplas plataformas.

Segundo Smeets, Boness e Bankras (2009), “o principal problema com applets Java e JavaFX é a necessidade de um Java Runtime Environment (JRE) a ser instalado no cliente ”. O que muitas vezes torna essas aplicações indesejadas para os usuários.

3.2.5 HTML 5

O HTML 5 é versão mais recente do HTML (*Hypertext Markup Language* –

6 http://www.oracle.com

Linguagem de Marcação de Hypertexto). Segundo o W3C (2007), o HTML é a *lingua franca*[7] para a publicação de hipertexto na Web. É um formato não proprietário e pode ser criado e processado por inúmeras ferramentas, desde simples editores de textos a sofisticadas IDE's de desenvolvimento.

De acordo com o Tableless (2011), “um dos principais objetivos do HTML5 é facilitar a manipulação do elemento possibilitando o desenvolvedor modificar as características dos objetos de forma não intrusiva e de maneira que seja transparente para o usuário final”.

Quando decidiu-se criar a nova versão do HTML, segundo o W3school (2011), algumas regras foram estabelecidas:

– Novos recursos devem ser baseados em HTML, CSS[8], DOM[9] e *JavaScript*[10];

– Reduzir a necessidade do uso de *plugins* externos (como o Flash, por exemplo);

– Melhorar o tratamento de erros;

– O processo de desenvolvimento deve ser transparente ao público

Com base nessa regras foi lançado o HTML5. Apesar de ainda estar em desenvolvimento muitos recursos já estão disponíveis, tais como:

– O elemento *canvas* para desenho;

– Os elementos de áudio e para reprodução de arquivos de mídia;

– Melhor suporte a armazenamento local *offline;*

– Novos elementos para conteúdos específicos tais como: *article*, *footer, header*, etc, entre outras funcionalidades;

A Figura 10 mostra a estrutura básica de uma página Web escrita em HTML5. Observe que houve poucas alterações, com exceção do *doctype*, no entanto, com esta alteração a tarefa de identificar qual DTD (*Document Type Definition*) deverá ser usado foi transferida para o browser. Com isso, cabe ao browser determinar qual versão de código a marcação foi escrita.

7 **Lingua franca** (sem acento no ‘i’) é uma expressão latina para língua de contato ou língua de relação resultante do contato e comunicação entre grupos ou membros de grupos linguisticamente distintos para o comércio internacional e outras interações mais extensas.(BORTOLLETO, 2009).

8 http://www.w3.org/Style/CSS/

9 http://www.w3.org/DOM/

10 http://www.w3schools.com/js/

```html
<!DOCTYPE HTML>

<html lang="pt-br">
  <head>
    <meta charset="UTF-8">
      <link rel="stylesheet" type="text/css" href="estilo.css">
      <title></title>
  </head>

  <body>
    <p>Estrutura básica de uma página web HTML5</p>

  </body>

</html>
```

Figura 10: Estrutura básica de uma página HTML5.

Alguns elementos do HTML 4.1 foram removidos ou reescritos por se tornarem obsoletos ou não serem utilizados pelos desenvolvedores. A Tabela 3 apresenta alguns elementos que foram adicionados na nova versão do HTML5.

Tabela 3: Novos elementos no HTML5 (TABLELESS, 2011)

Tag	**Descrição**
<section>	A *tag section* define uma nova seção genérica no documento. Por exemplo, a home de um *website* pode ser dividida em diversas seções: introdução ou destaque, novidades, informação de contato e chamadas para conteúdo interno.
<nav>	O elemento *nav* representa uma seção da página que contém links para outras partes do *website.*
<article>	O elemento *article* representa uma parte da página que poderá ser distribuído e reutilizável em FEEDs por exemplo.
<aside>	O elemento *aside* representa um bloco de conteúdo que referência o conteúdo envolta do elemento *aside*. O *aside* pode ser representado por conteúdos em *sidebars* em textos impressos, publicidade ou até mesmo para criar um grupo de elementos *nav* e outras informações separados do conteúdo principal do website.
<header>	O elemento *header* representa um grupo de introdução ou elementos de navegação. O elemento *header* pode ser utilizado para agrupar índices de conteúdos, campos de busca ou até mesmo logos.
<footer>	O elemento *footer* representa literalmente o rodapé da página. Seria o último elemento do último elemento antes de fechar a tag HTML.
<time>	Este elemento serve para marcar parte do texto que exibe um horário ou uma data precisa no calendário gregoriano.

Além de novos elementos o HTML5 conta com um conjunto de API. Para a maior parte das *tags* do HTML5 há um API que é baseada em JavaScript que permite aos desenvolvedores ter maior controle sobre o elemento, tornando as páginas em HTML5 mais dinâmicas.

A Tabela 4 apresenta uma comparação entre as principais tecnologias que dão suporte ao desenvolvimento de RIA.

Tabela 4: Comparação entre as principais tecnologias que dão suporte ao desenvolvimento de RIA.

Característica	***Flash***	***Flex***	**JavaFX**	**HTML5**
Riqueza Gráfica	Alta	Alta	Alta	Alta
Suporte a Áudio e Vídeo	Sim	Sim	Sim	Sim
Requer Plugin/Runtime no cliente	Sim	Sim	Sim	Não
Drag-and-Drop	Sim	Sim	Sim	Sim
Local Data Storage	Sim	Sim	Sim	Sim
Aplicações offline	Sim	Sim	Sim	Sim
Avançar e voltar no navegador	Sim	Sim	Não	Sim
Compatibilidade entre navegadores	Sim	Sim	Sim	Não

Como pode ser observado na tabela acima, o HTML5 apresenta-se como um poderoso recurso na implementação de RIA's. A principal vantagem do HTML5 em relação às outras tecnologias é que esta não necessita de plugins ou *runtimes* instalados no cliente, aumentando assim a portabilidade entre as plataformas, inclusive as móveis. No entanto, o HTML5 possui a desvantagem de não ser compatível com todos os navegadores no mercado. Isso deve-se ao fato de os navegadores ainda não terem implementado todas as funcionalidades propostas pelo HTML5.

Portanto, para o desenvolvimento do estudo de caso deste trabalho foi escolhido o HTML5 pois este apresenta-se como uma ótima alternativa no desenvolvimento de RIA apesar de ainda estar em desenvolvimento.

4 ESTUDO DE CASO

Neste capítulo é apresentando o estudo de caso deste trabalho: o projeto e o desenvolvimento da Ferramenta para Gestão de Projetos Scrum, a partir daqui denominada de *WSCRUM*, uma redução da expressão *Scrum on the Web*[11].

4.1 PROCESSO DE DESENVOLVIMENTO

Com base nos estudos realizados nos capítulos anteriores deu-se início ao desenvolvimento da ferramenta, que seguiu as seguintes fases: Análise de requisitos, Modelagem de Requisitos, Implementação e Testes.

4.1.1 Análise de Requisitos

Nesta etapa do projeto de sistema foi levantado os requisitos funcionais e não funcionais com base nas informações contidas na seção que trata do *Scrum*, no capítulo 2 deste trabalho. Este requisitos serviram para delimitar o escopo do projeto bem como guiar no desenvolvimento e implementação do mesmo. Nas seções seguintes os requisitos são listados e descritos.

4.1.1.1 Requisitos Funcionais

Os requisitos funcionais,

> Estão intimamente ligados às funcionalidades propostas pelo sistema, e que serão usadas na resolução do problema do contratante, e atenderá todas as suas necessidades funcionais. Resumidamente, são os requisitos que objetivamente cumprem as reais necessidades do usuário do sistema. (LONGO & BEZERRA, 2011)

Os requisitos foram divididos em duas categorias: Administrativo e Projeto. A primeira descreve as tarefas que poderão ser realizadas pelo administrador do

11 Scrum na Web

sistema. A segunda, descreve as tarefas que podem ser desenvolvidas pela equipe no escopo de um projeto específico.

Requisitos Administrativo

– RF.01: Permitir ao administrador o cadastro e manutenção dos projetos, bem como, o controle de todas as ações pertinentes aos outros módulos de projeto.

– RF.02: Permitir ao administrador cadastrar e manter os usuários do sistema, bem como delegar tarefas administrativas a outros usuários.

– RF.03: Permitir ao administrador cadastrar e manter as equipes de desenvolvimento.

Requisitos de Projeto

– RF.04: Permitir ao *Scrum Master* iniciar e encerrar os *sprints*.

– RF.05: Permitir ao *Scrum Master* a manutenção do cadastro das equipes com seus respectivos membros.

– RF.06: Permitir ao *Scrum Master* a manutenção do cadastro do *product backlog* com seus respectivos itens (estórias e tarefas).

– RF.07: Permitir a priorização dos itens que comporão o *product backlog*.

– RF.08: Permitir a seleção das estórias que comporão o *sprint backlog* durante os *sprints*.

– RF.09: Permitir ao seleção de responsáveis de cada estória durante os *sprints*.

– RF.10: Permitir a adição de tarefas às estórias do backlog do produto.

– RF.11: Permitir o cadastro das estimativas (horas e dias) de cada tarefa.

– RF.12: Permitir ao *Team Member* lançar o resumo das atividades diárias desenvolvidas durante o sprint.

– RF.13: Permitir ao *Scrum Master* a manutenção do cadastro das reuniões de retrospectivas dos *sprints*.

– RF.14: Permitir ao *Scrum Master* e ao T*eam Member* manter cadastro da lista de impedimentos.

– RF.15: Permitir ao *Product Owner* acompanhar o andamento das atividades do projeto.

– **RF.16:** Permitir ao *Scrum Master* manter cadastro da categoria de estórias.

4.1.1.2 Requisitos Não-Funcionais

Os requisitos não funcionais, segundo Longo & Bezerra (2011) estão, "geralmente ligados à qualidade do produto como, por exemplo, robustez, segurança ou integridade". Abaixo são listados os requisitos não-funcionais do WSCRUM.

– **RNF.01:** O sistema terá como plataforma a web.

– **RNF.02:** A interface do sistema deverá está sob os conceitos da RIA.

– **RNF.03:** O banco de dados do sistema será implementado utilizando o *MySql*[12].

– **RNF.04:** A interface será implementada em HTML5, *JavaScript* e CSS, sem o uso de *Flash, JavaFX, Silverlight* ou *Flex.*

– **RNF.05:** A linguagem de programação utilizada no lado servidor da aplicação deve ser o PHP[13] (*Hypertext Preprocessor*).

– **RNF.06:** Login e senha deverão ser solicitados na tela inicial da aplicação para acesso aos recursos da mesma.

4.1.2 Modelagem dos Requisitos

Os requisitos foram modelados utilizando diagramas *UML.* A *UML* é segundo Medeiros (2004), uma maneira de transmitir uma ideia. Indicando a forma que poderá ser utilizada para representar um software durante os seus estágios de desenvolvimento.

4.1.2.1 Diagramas de casos de uso

12 http://www.mysql.com
13 http://www.php.net/

A modelagem dos requisitos foram divididos em dois módulos no diagrama de casos de uso para facilitar o entendimento do sistema. Estes módulos são: Administrativo e Projeto.

Módulo Administrativo

A Figura 11 apresenta o diagrama de casos de uso do módulo administrativo do sistema bem como a correspondência dos casos de uso com os requisitos funcionais do sistema. O administrador poderá cadastrar e manter o cadastro de projetos e equipes de desenvolvimento, além de poder cadastrar e manter cadastros de outros usuários do sistema.

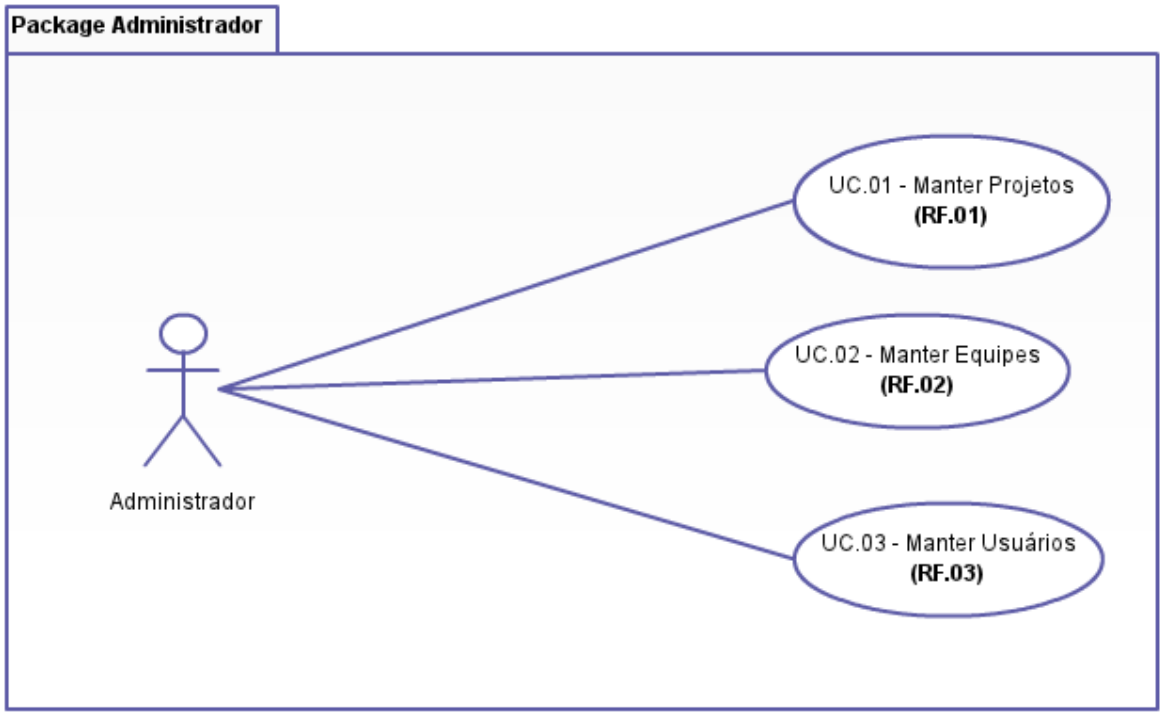

Figura 11: Casos de uso do módulo administrativo do WSCRUM.

Módulo de Projeto

A Figura 12 mostra as atividades que o *Scrum Master*, o *Product Owner* e o Team Member poderão executar no sistema. O *Scrum Master* tem a função de cadastrar e manter cadastro dos *sprints*, equipes, *product backlog*, reunião de

retrospectiva, impedimentos e categorias. O *Product Owner* poderá acompanhar o andamento das atividades. O Team Member poderá lançar no sistema as informações das atividades diárias que foram realizadas.

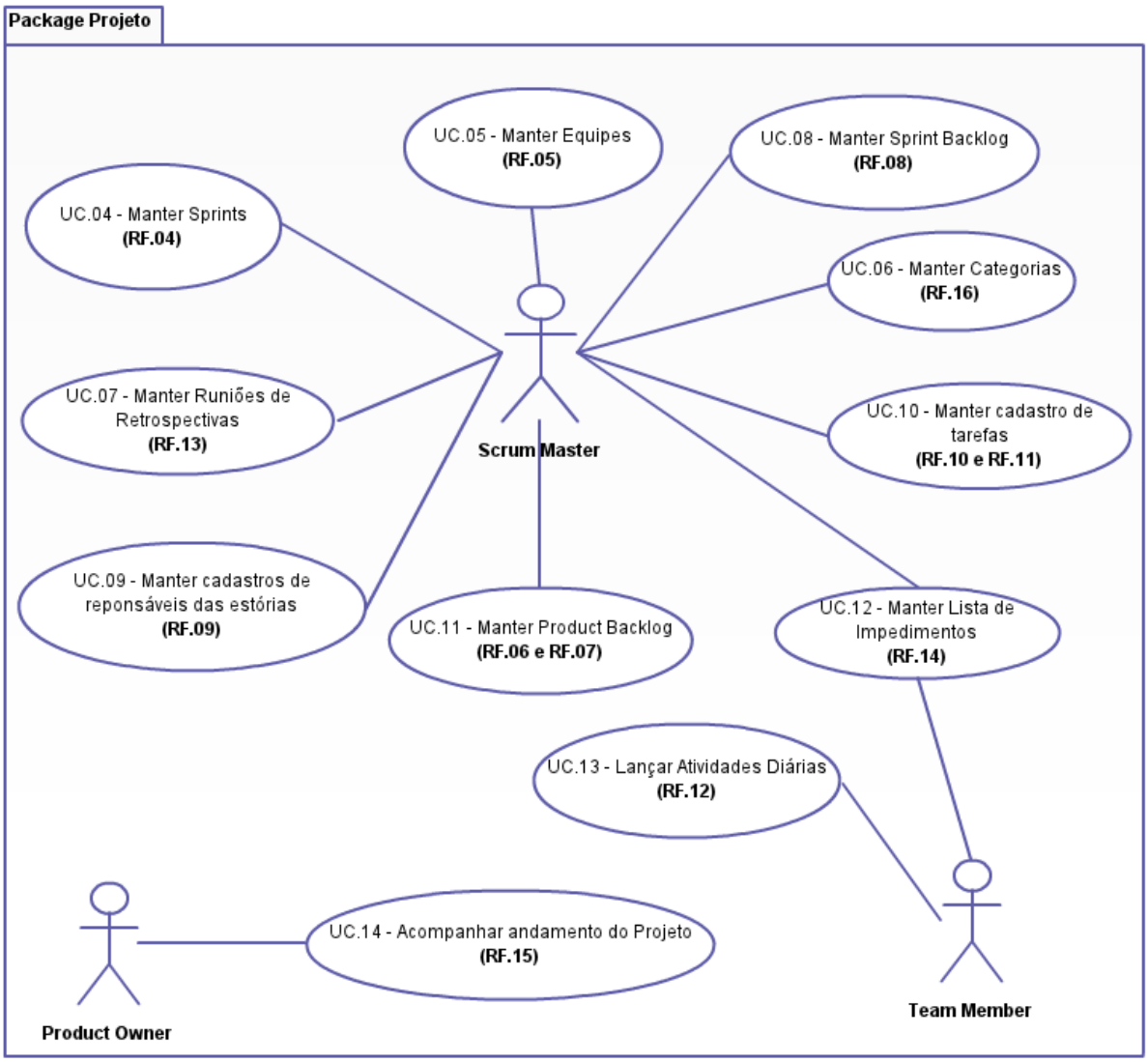

Figura 12: Casos de uso do módulo Projeto do WSCRUM.

4.1.2.2 Diagrama de Classes

O diagrama de classes tem por objetivo descrever os vários tipos de objetos do sistema e os relacionamentos existentes entre eles. É através de um diagrama de classes que segundo Medeiros (2004), nos informa as grande áreas que serão

tratadas no *software*. A Figura 13 mostra o diagrama de classes do WSCRUM.

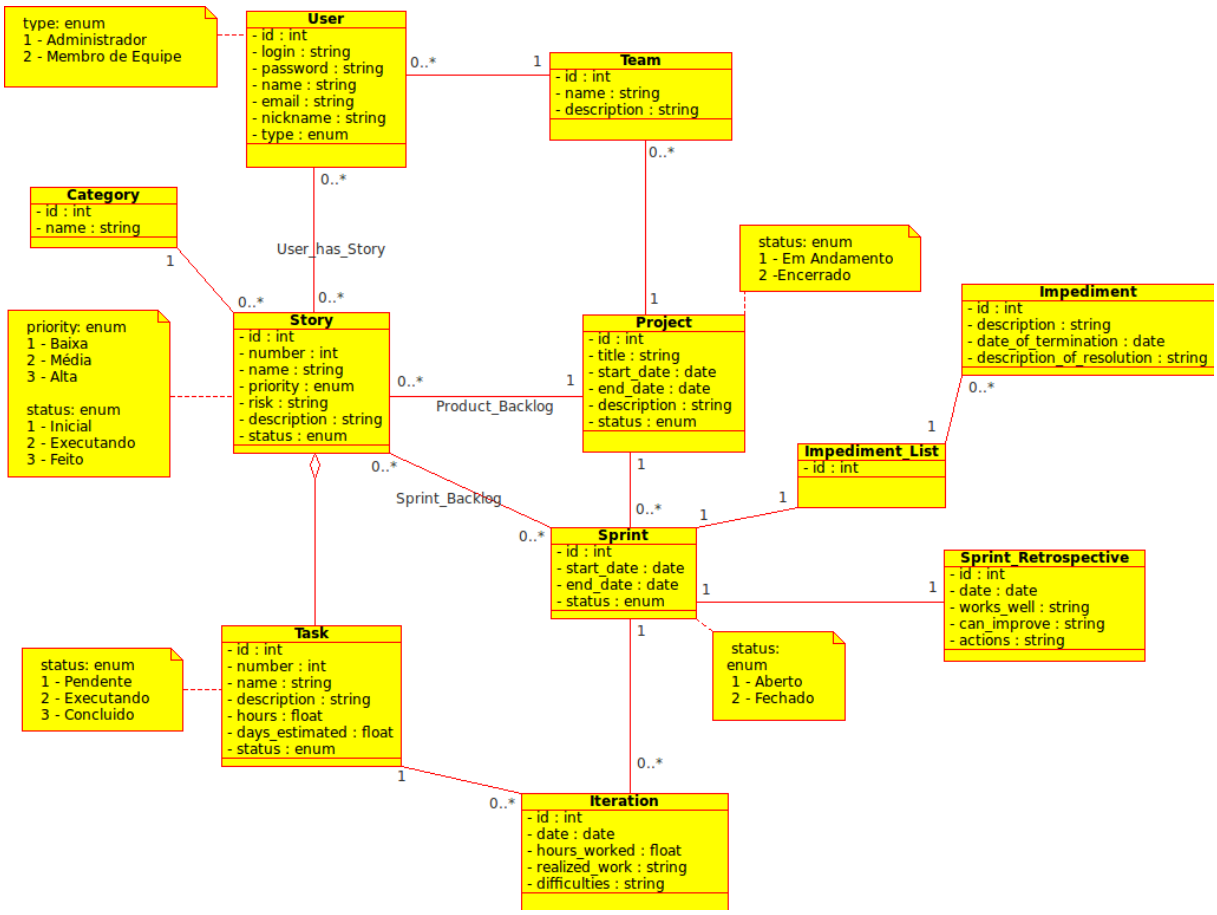

Figura 13: Diagrama de Classes do WSCRUM

Uma breve descrição do diagrama de classes do WSCRUM é apresentada a seguir:

– ***User***: classe que representa os usuários do sistema. Estabelece o nível de acesso ao sistema que o usuário terá;

– ***Team***: classe que representa as equipes de um determinado projeto e o objetivo destas neste projeto;

– ***Project***: classe que representa os projetos que serão gerenciados com o apoio desta ferramenta;

– *Story*: classe que representa as estórias do projeto, compondo assim o *product backlog*;

– *Category*: classe que representa as categorias de estórias em um projeto;

– *Task*: classe que representa as tarefas de cada estória. Além disso, armazenas as estimativas de tempo de cada tarefa;

– *Sprint*: classe que representa os *sprints* de cada projeto. Bem como o período de duração de cadas *sprint*;

– *Iteration*: classe que representa cada dia de iteração dos sprints. Armazena as atividades desenvolvidas pelo membro da equipe para cada tarefa;

– *Impediment_List*: classe que representa a lista de impedimentos que poderão ocorrer em cada *sprint*;

– *Impediment*: classe que representa os impedimentos que comporão a lista de impedimento, bem como as informações de como e quando foi solucionado;

– *Sprint_Retrospective*: classe que representa a reunião de retrospectiva no final de cada sprint, bem como armazena as informações acerca do que funcionou, o que pode ser melhorado e o as ações a serem tomadas para melhorar.

– *User_has_Story:* relacionamento que permite uma estória estar vinculada um ou mais usuários.

– *Product_Backlog:* relacionamento que permite uma estória estar vinculada a um projeto, criando assim a lista de backlog do produto.

– Sprint*_Backlog:* relacionamento que permite uma estória estar vinculada a um *Sprint*, criando assim a lista de backlog do *Sprint*.

4.1.2.3 Diagrama Entidade-Relacionamento

Após terminada a modelagem dos requisitos, foi produzido o Diagrama de Entidade e Relacionamento - DER representando a estrutura conceitual do banco de dados. Este diagrama pode ser observado na Figura 14 e traduz para o banco de dados relacional as classes persistentes do diagrama de classes bem como seus relacionamentos.

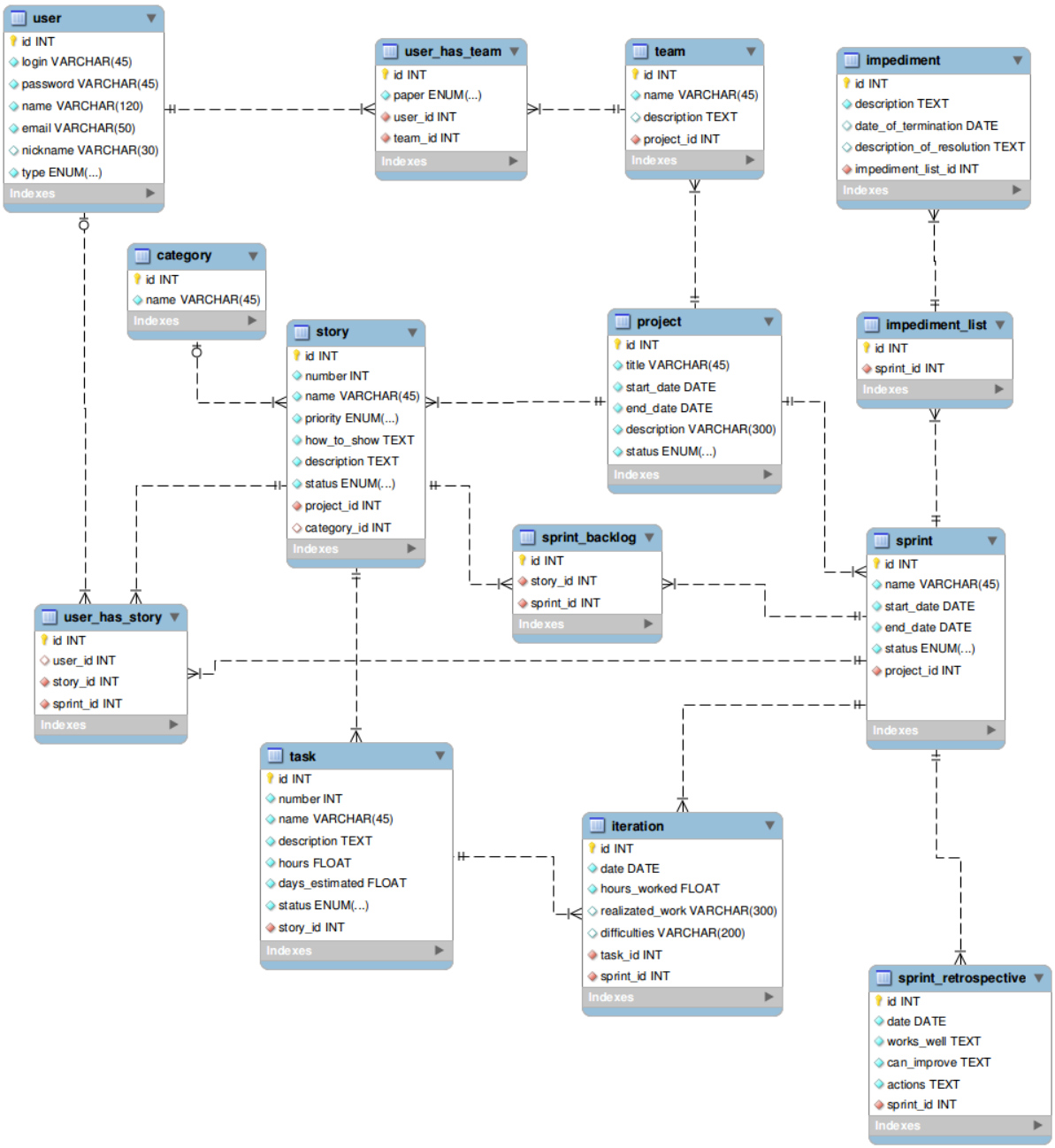

Figura 14: Diagrama de Entidade e Relacionamento do WSCRUM

4.1.3 Implementação e Testes

A ferramenta foi desenvolvida para a plataforma Web (RNF.01)na arquitetura cliente-servidor sob os conceitos de RIA (RNF.02). Para o lado cliente da plicação foi

utilizado HTML5, CSS3 e *JavaScript* (RNF.04). Devido o HTML5 ser uma versão do HTML ainda em desenvolvimento, o navegador utilizado para testes foi o Mozilla Firefox 6.0[14], pois este mostrou-se ser o mais adequado no uso da ferramenta.

Do lado servidor da aplicação, foi utilizado o *PHP* para manipulação dos dados (RNF.05) e o *MySql* como software gerenciador de banco de dados (RNF.03). A Figura 15 mostra a arquitetura do sistema.

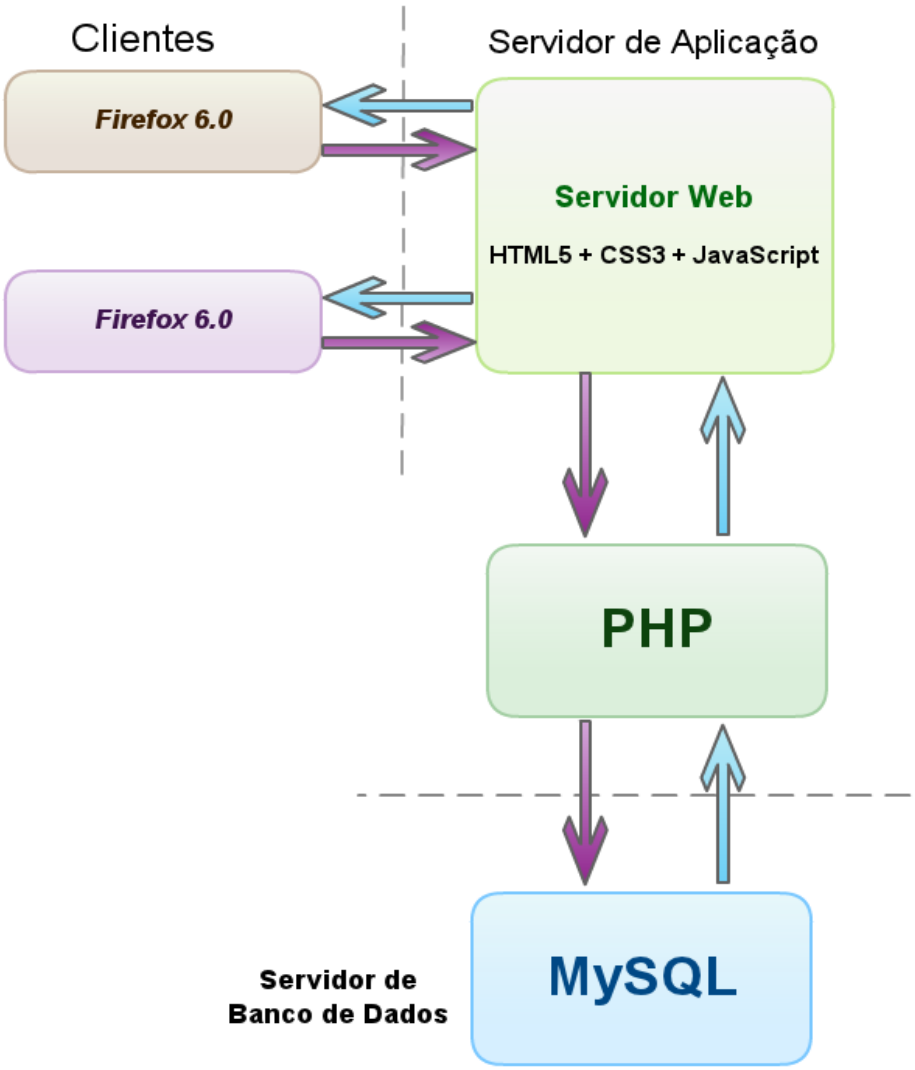

Figura 15: Arquitetura do WSCRUM

Testes de validação e de verificação foram realizados durante todo o processo de desenvolvimento, de forma sucinta e objetiva.

Na próxima seção é apresentada uma lista de ferramentas utilizadas no

14 http://www.mozilla.org/

processo de implementação da ferramenta.

4.1.3.1 Ferramentas de desenvolvimento

– *Geany*: IDE que dá suporte às principais linguagens de programação.

– *MySQL Workbench*: é uma ferramenta visual para gerenciamento de banco de dados MySQL que integra projeto, desenvolvimento e gerência de banco de dados.

– *Inkscape*: editor de imagens e documentos vetoriais;

– *GIMP*: cria e edita imagens, é similar ao *Photoshop.*

4.2 A FERRAMENTA WSCRUM

Com base nos requisitos levantados e analisados o sistema foi desenvolvido. A Figura 16 mostra a tela inicial da ferramenta. Como pode ser observado, nesta tela o usuário deverá fornecer seu login e senha para acesso ao sistema (RNF.06). O usuário padrão do sistema é o *admin*. Este usuário possui privilégios de administrador e pode criar projetos, além de cadastrar outros usuários e outorgar privilégios a estes.

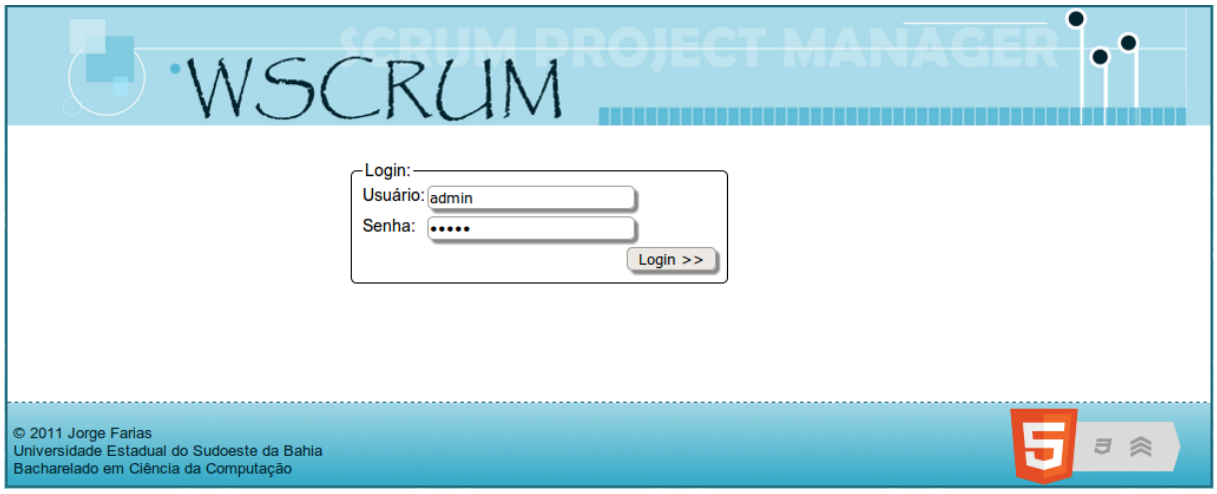

Figura 16: Tela de login do WSCRUM

Após o login o usuário encontrará a página de projetos cadastrados do sistema. Ainda nesta tela, um menu é apresentado ao usuário (Figura 17).

Figura 17: Menu inicial e projetos cadastrados no sistema

Se o usuário possuir privilégios de administrador, este poderá cadastrar novos projetos e, além disso, todos os projetos cadastrados são apresentados a ele. Caso o usuários não possua privilégios de administrador, apenas é listado os projetos aos quais o usuário faça parte. A Figura 18 apresenta a tela de cadastro de projeto, aqui deve-se informar o nome do projeto, a data de início, a previsão de término e o objetivo do mesmo (UC.01).

O calendário presente na Figura 18 é um componente que faz parte da biblioteca *JQuery*[15], chamado de *Datepicker.* O JQuery, é uma biblioteca para HTML que utiliza *JavaScript* e CSS em seus componentes.

15 http://jquery.com/

Figura 18: Cadastro de um novo projeto

Após o projeto ser cadastrado o usuário poderá retornar à tela em que os projetos são listados (Figura 17) e clicar no nome do projeto para utilizá-lo. Ao escolher o projeto o usuário será redirecionado para a tela que apresenta as informações básicas do projeto (Figura 19). As informações são: nome, data de início, previsão de término, objetivo e estórias pendentes, executando e concluídas do projeto.

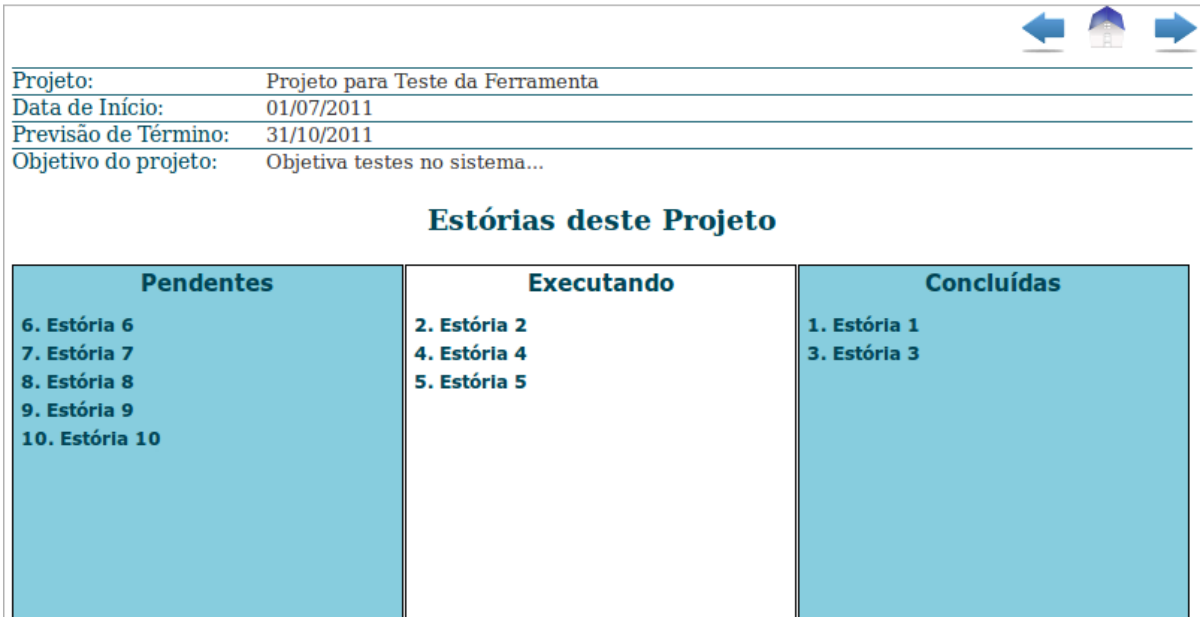

Figura 19: Informações básicas do projeto.

Caso o usuário seja um administrador ou o *Scrum Master* ele poderá cadastrar outros usuários no sistema (UC.03), como mostra a Figura 20. Neste caso apenas o usuário *admin* poderá conceder privilégios de administrador a outros usuários.

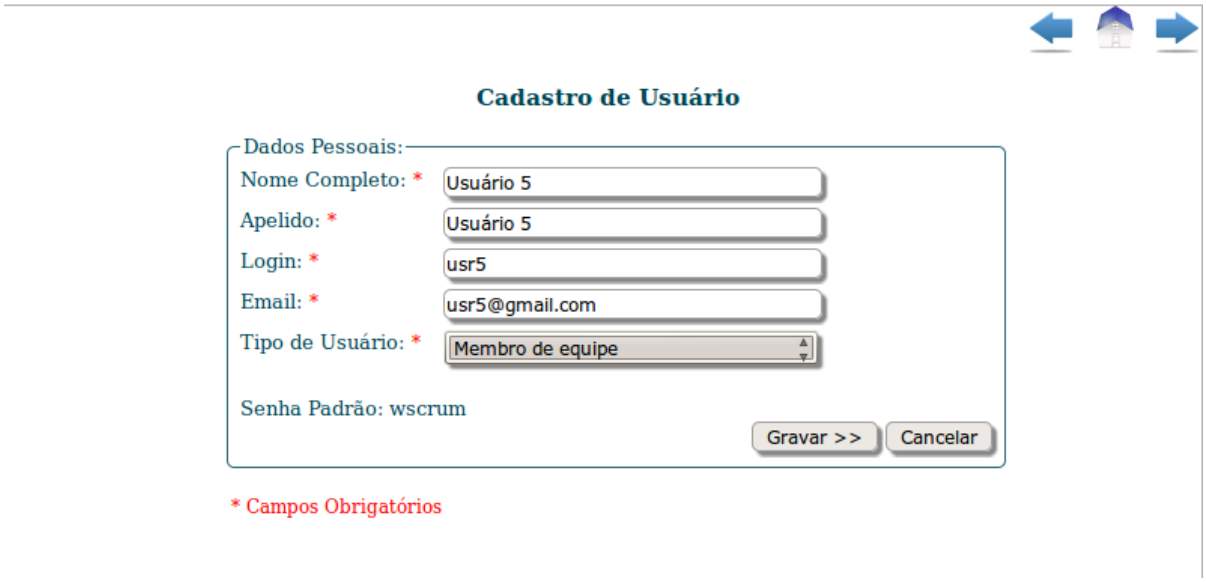

Figura 20: Cadastro de usuários do sistema.

Como apresentado Figura 21 Figura 22 respectivamente, após os usuários serem cadastrados o administrador ou o Scrum Master poderá cadastrar equipes para o projeto e vincular os usuários nas equipes criadas definindo os seus devidos papéis (UC.02 e UC.05).

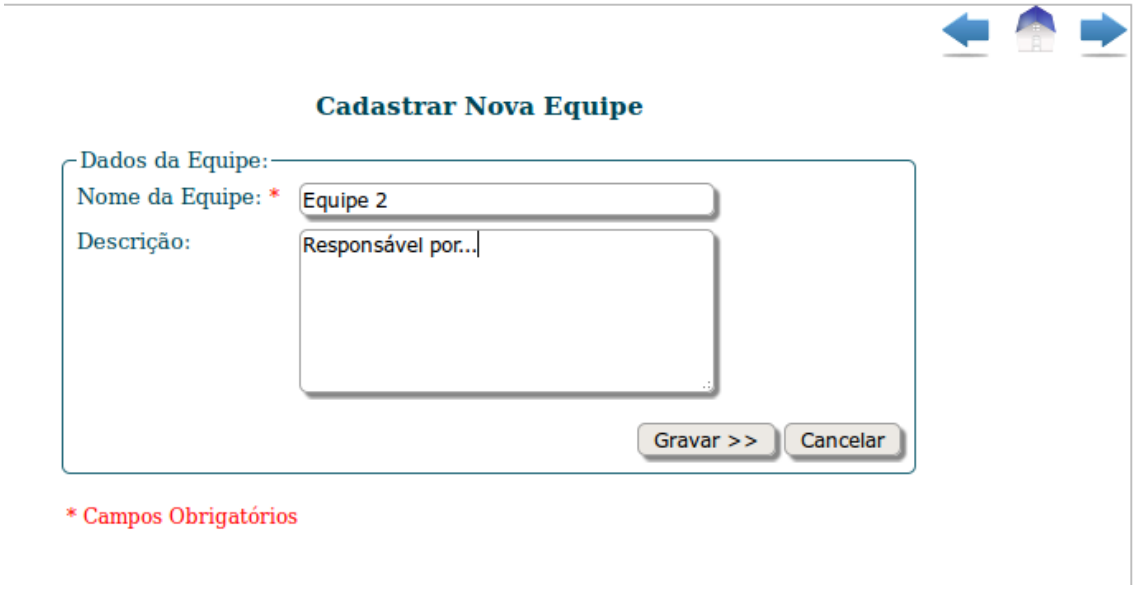

Figura 21: Cadastro de equipe.

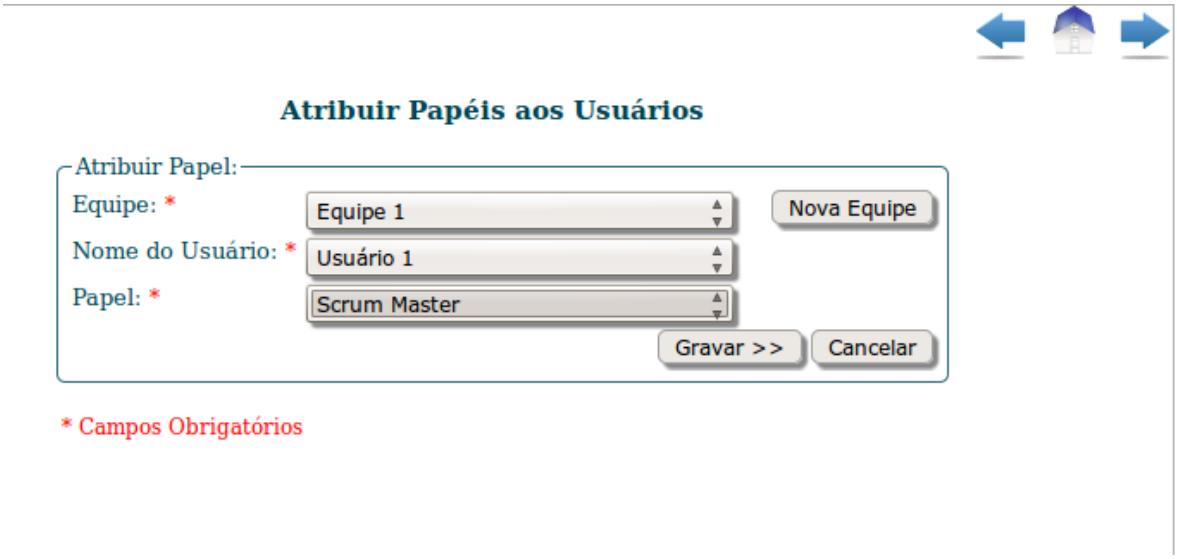

Figura 22: Adicionar usuário a uma equipe e definir papel.

O backlog do produto, como dito anteriormente, é composto por estórias. A Figura 23 apresenta tela de cadastro de uma estória (UC.11). Aqui deverá ser informado o nome da estória, a categoria, a prioridade, o exemplo de como demonstrá-la e a descrição do que ela representa.

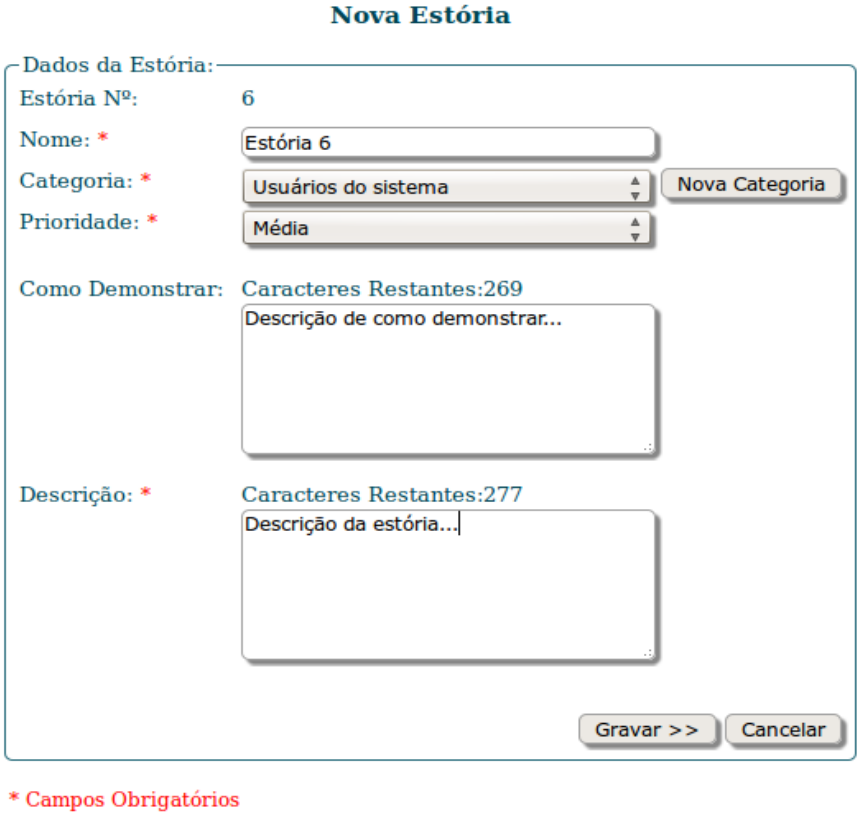

Figura 23: Cadastro de nova estória.

Cada estória poderá ser dividida em uma ou mais tarefas. As tarefas são as atividades que serão executadas em um *Sprint.* No cadastro de uma tarefa deve-se especificar (Figura 24) o nome, dias estimados, quantidade de horas a serem trabalhadas por dia estimado e o objetivo da tarefa (UC.10).

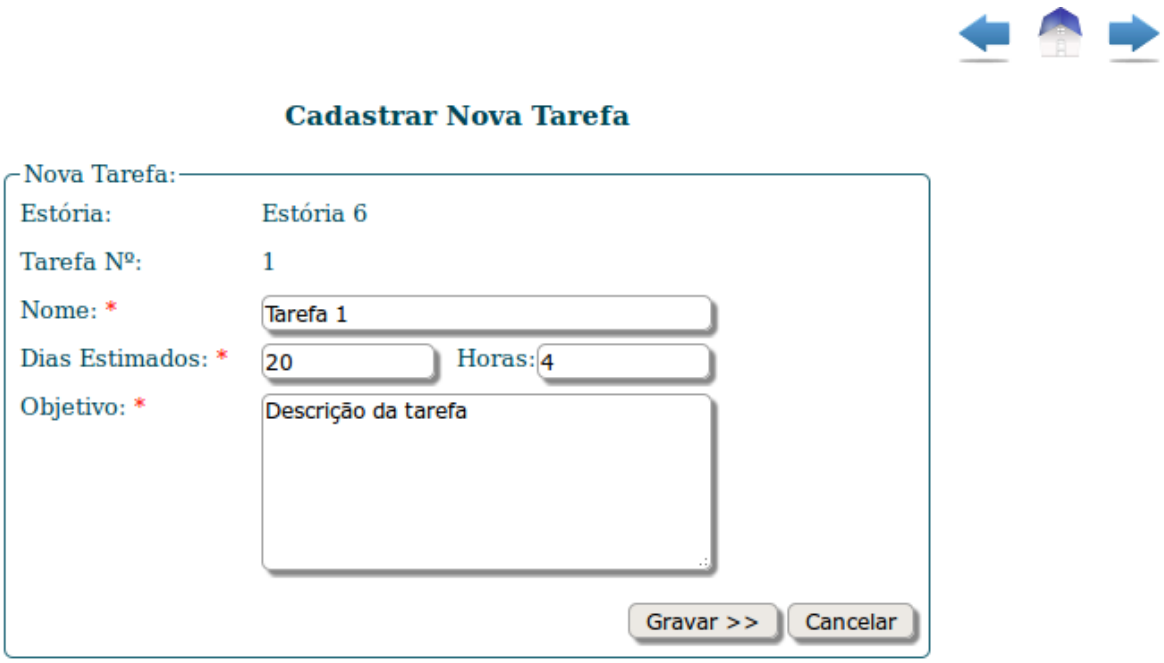

Figura 24: Cadastrar tarefas para uma determinada estória.

Na Figura 25 é apresentado a listagem de tarefas para uma determinada estória, bem como a quantidade de horas estimadas e pendentes para cada tarefa.

Figura 25: Listagem de tarefas de uma determinada estória.

As tarefas possuem um histórico de iterações (Figura 26). O WSCRUM permite que uma estória possa participar de mais de um *Sprint,* caso um *Sprint* não seja suficiente para que esta seja concluída.

Figura 26: Histórico de iterações para uma tarefa.

O administrador do sistema ou o Mestre *Scrum* pode cadastrar um novo *Sprint* como apresenta a Figura 27 (UC.04). Ao iniciar um novo *Sprint* o usuário deve informar o nome do *Sprint,* data de início e data de término do mesmo.

Figura 27: Cadastro de *Sprint.*

Inciado o *Sprint* o Mestre *Scrum* poderá adicionar as estórias que comporão o *Sprint Backlog* como mostrado na Figura 28 (UC.08). Ao adicionar as estórias no *Backlog* do *Sprint* o usuário deverá informar os membros de equipes responsáveis pela estória durante o *Sprint* (UC.09). No WSCRUM poderá escolher até dois responsáveis por estória já que o sistema está de acordo a prática de programação em par do XP.

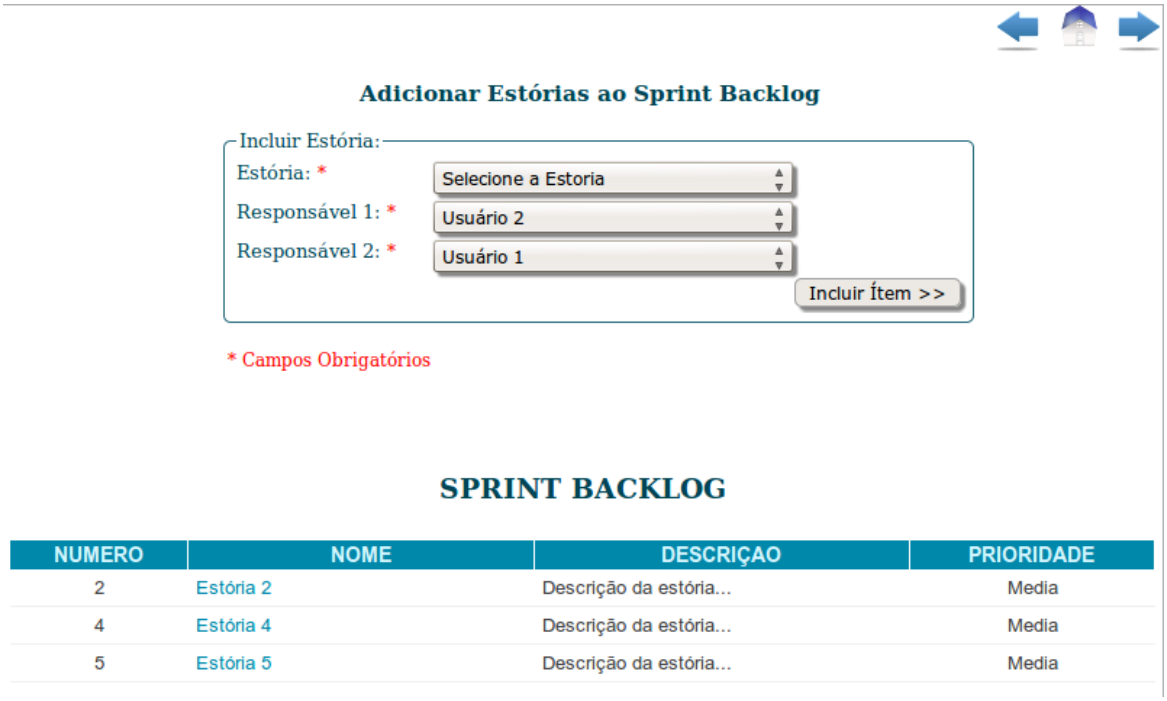

Figura 28: S*print Backlog.*

Durante um *Sprint* os os membros das equipes devem preencher uma lista de impedimentos, como apresenta a Figura 29, para que o Mestre *Scrum* possa obter *feedback* e resolver problemas que estejam interfirindo no andamento das atividades (UC.12).

Figura 29: Lista de impedimentos

Durante o *Sprint* os responsáveis da estória devem lançar diariamente informações sobre horas trabalhas, trabalho realizada e dificuldades encontradas para cada tarefa da estória (UC.13), como mostra a Figura 30.

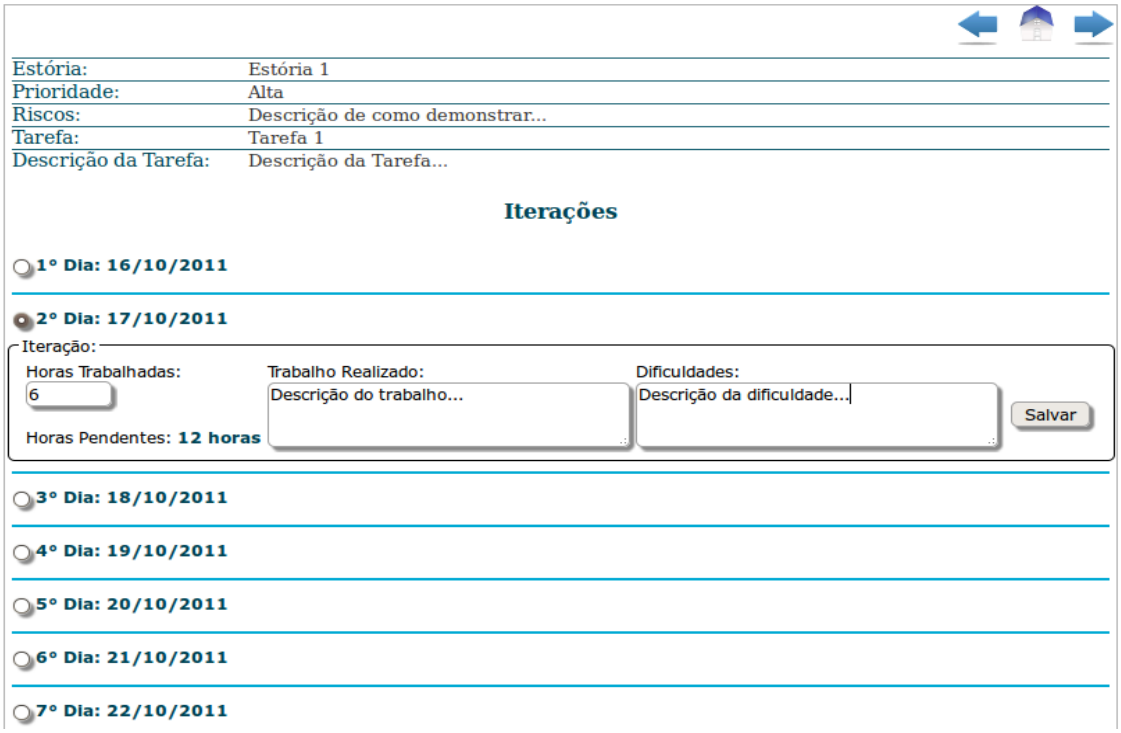

Figura 30: Iterações da tarefa em um *Sprint*.

Após concluído um *Sprint* o usuário deverá encerrá-lo informando no sistema a data e os dados da reunião de retrospectiva do *Sprint* (UC.07). Os dados devem ser baseados nas seguintes perguntas:

– O que funcionou bem?

– O que pode melhorar?

– Quais ações serão tomadas?

Após respondidas as questões o Mestre *Scrum* poderá encerrar o *Sprint* (Figura 31).

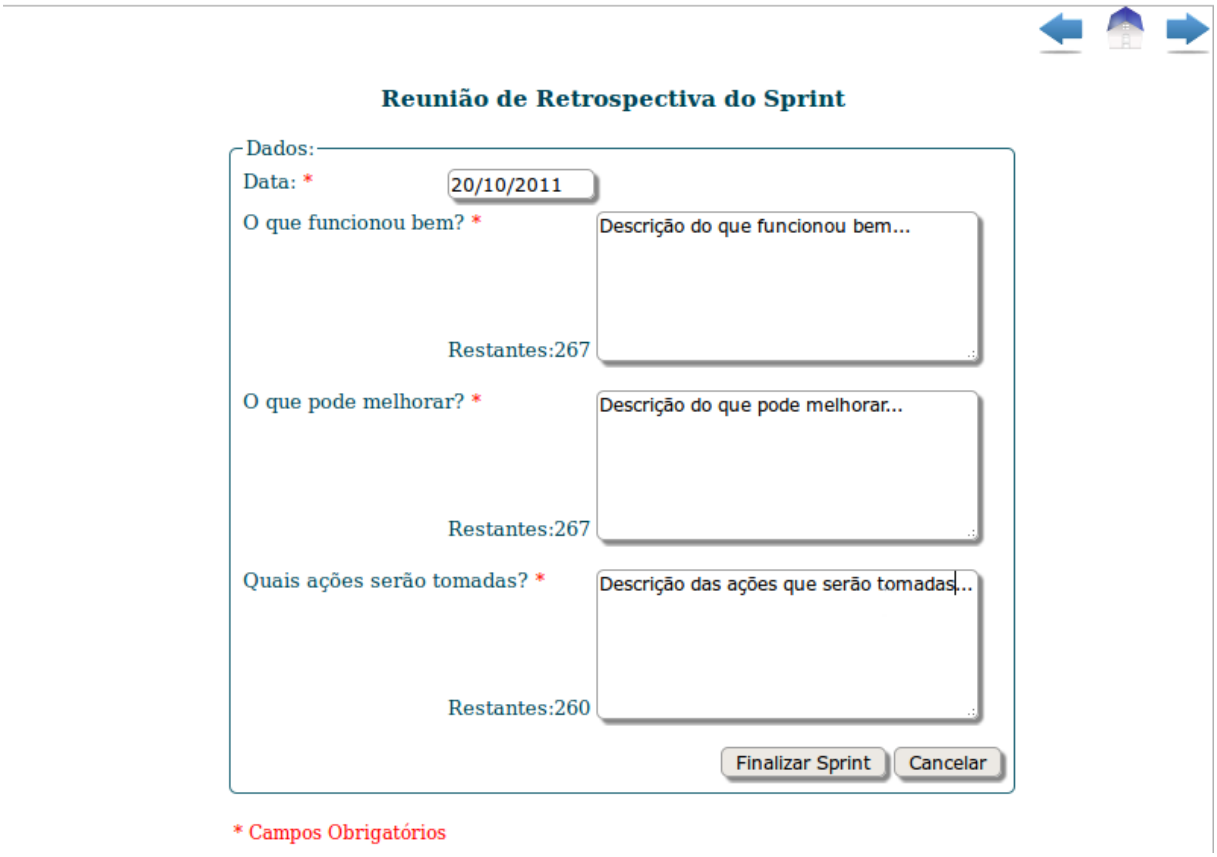

Figura 31: Cadastrar os dados da reunião de retrospectiva e encerrar o *Sprint.*

A Figura 32 mostra o cadastro de categorias de estorias (UC.06).

Figura 32: *Burndown Chart* de um *Sprint.*

O andamento das atividades da equipe do projeto em um *Sprint* poderá ser acompanhada através do *Dashboard* (Figura 33), o qual mostras as estórias que ainda estão pendentes no *backlog* do produto, as que estão sendo executas e as finalizadas (UC.14).

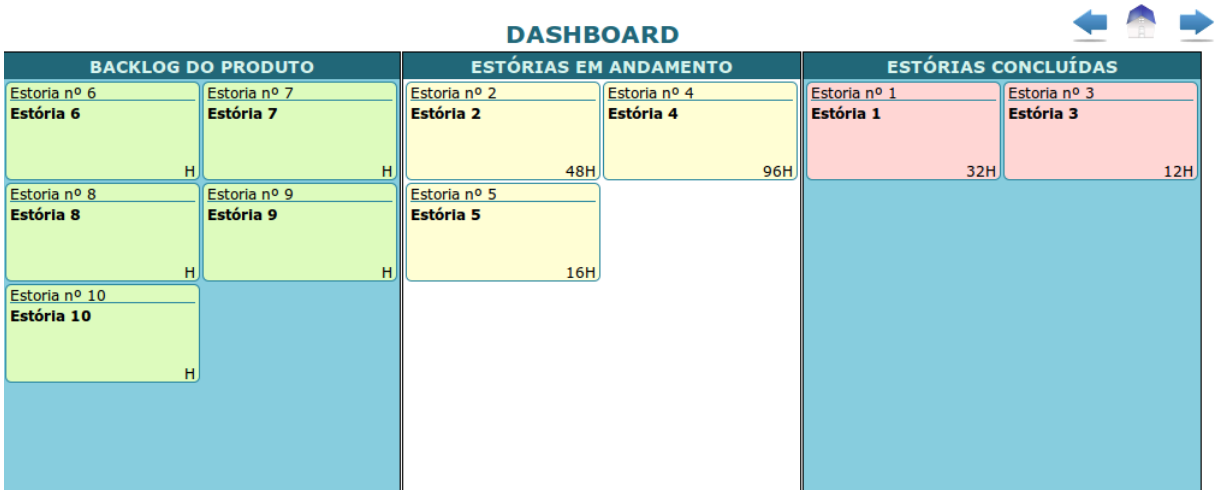

Figura 33: *Dashboard*

Além do *dashboard*, as atividades dos *Sprints* poderão ser acompanhadas através do gráfico *Burndown* (Figura 34). Este gráfico apresenta o somatório de horas pendentes das estórias que estão no *Backlog* do *Sprint* atual (UC.14).

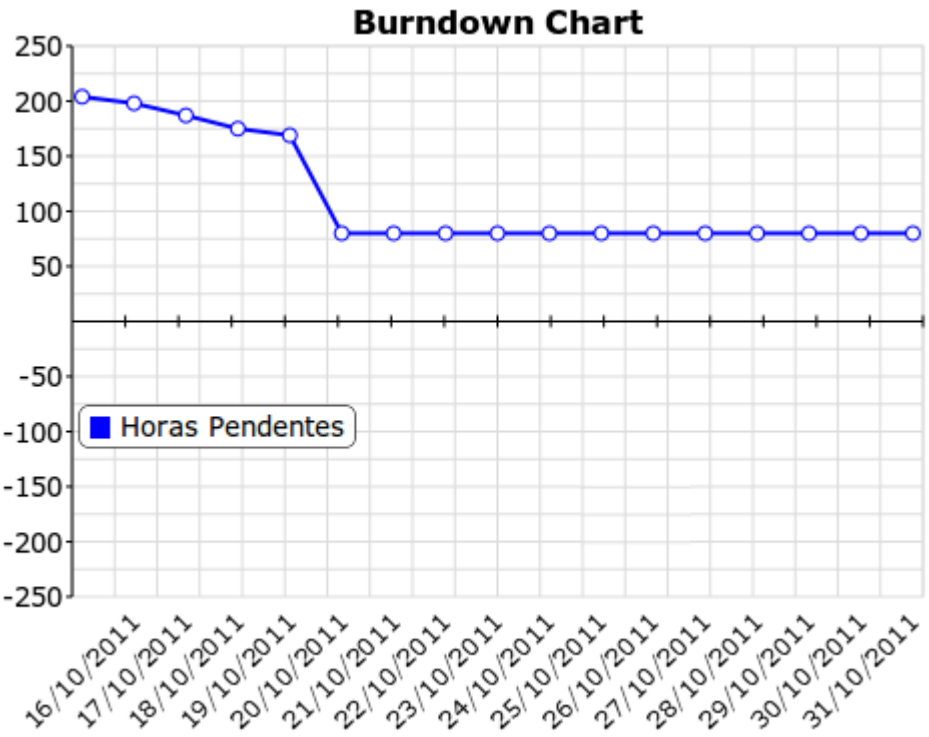

Figura 34: Gráfico *Burndown*

O gráfico *Burndown* é gerado utilizando uma API do HTML5 chamada de RGraph[16].

4.3 CONSIDERAÇÕES DO CAPÍTULO

A ferramenta para gestão WSCRUM, desenvolvida em HTML5, permite às empresas gerenciarem seus projetos de forma fácil e ágil. Norteada pelo método de desenvolvimento *Scrum*, apresenta-se como importante ferramenta de suporte na gestão de projetos de *software*, contribuindo na garantia de qualidade dos produtos

16 http://www.rgraph.net/

desenvolvidos.

O WSCRUM contempla os principais papéis artefatos do *Scrum*, tais como o *Scrum Master*, *Product Owner*, *Team Member*, *Product Backlog*, *Sprint Backlog*, estórias, tarefas, gráfico *Burndown*, além disso, permite que as atividades diárias sejam lançadas, contribuindo assim para a manutenção e suporte dos projetos desenvolvidos utilizando a ferramenta como apoio.

Para o desenvolvimento foi escolhido o HTML5 por este ser uma alternativa não proprietária no desenvolvimento de RIA's, além de dispensar a instalçao de plugins para a execução do sistema tornando-o mais portável.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve por objetivo o desenvolvimento do WSCRUM, uma ferramenta para gestão de projetos de software *Scrum* utilizando o HTML5.

O HTML5 ainda está em desenvolvimento, ou seja, os *browsers* ainda não suportam totalmente este novo padrão. No entanto, alguns navegadores, como o *Mozilla Firefox*, já suportam grande parte das funcionalidades do HTML5, permitindo assim que a ferramenta pudesse ser desenvolvida utilizando esta tecnologia.

Entre as várias vantagens do WSCRUM pode-se destacar que por ser uma ferramenta Web, possui grande disponibilidade e manutenibilidade. Além disso, como dispensa o uso de *plugins*, a portabilidade é garantida, mostrando-se eficaz tanto em ambiente Windows quanto ambientes Linux.

Outra vantagem desta ferramenta é que esta poderá ser utilizada como recurso nas disciplinas da Engenharia de *Software*, contribuindo para o ensino de metodologias ágeis.

Durante o desenvolvimento da ferramenta surgiram algumas dificuldades no uso do HTML5, principalmente em relação aos navegadores que o suportam, pois nem todas funcionalidades ainda foram implementadas. Além disso, casos reais de testes não puderam ser feitos devido ao *Scrum* determinar uma equipe com no mínimo três componentes. Outra dificuldade é o fator tempo, pois a implementação da ferramenta tomou mais tempo que o planejado.

Diante disso, como sugestão para trabalhos futuros poderá ser realizados testes reais com a ferramenta, além de poder acrescentar no sistema novas funcionalidades tais como o *drag-and-drop*, para torná-la mais rica, além de permitir a escolha de qual métrica de *software* poderá ser utilizada nos projetos.

BIBLIOGRAFIA

ABRAHAMSSON, Pekka et al. *Agile software development methods*: Review and analysis.VTT Publications 478. Oulu, Finland: VTT Publications, 2002.

ADOBE SYSTEMS INCORPORATED. *Plataforma Adobe Flash*. Disponível em: <http://www.adobe.com/br/flashplatform/> Acesso em: 02 de setembro de 2011.

_____. *Visão geral do Flex*. Disponível em: <http://www.adobe.com/br/products/flex/overview/> Acesso em: 02 de setembro de 2011.

AKED, Mark. Risk reduction with the RUP phase plan. Disponível em: <http://www.ibm.com/developerworks/rational/library/1826.html> Acesso em: 06 de julho de 2011.

BOZZON, Alessandro . *et al*. *Conceptual Modeling and Code Generation for Rich Internet Applications*. Departamento de Eletrônica e Informação Politécnica de Milão, Milão – Itália. 2006.

BORTOLLETO, Galaor. *Entenda o que é Lingua Franca*. 2009. Disponível em: < http://www.galaor.com.br/lingua-franca> Acesso em: 02 de setembro de 2011.

BRUST, Saulo S. *Rich Internet Application (RIA) para auxílio ao levantamento radiométrico de instalações e equipamentos radiológicos.* Trabalho de Graduação. Faculdade de Computação e Informática. Da Universidade Presbiteriana Mackenzie, São Paulo, 2007.

CARVALHO, Gustavo H. P. de. *Um Modelo Preditivo para Desenvolvimento de Jogos de Computador*. Trabalho de Graduação. Centro de Informática da UFPE. Pernambuco. 2006

CASTILLO, Cindy. *JavaFX Architecture and Framework*. Disponível em: < http://download.oracle.com/javafx/2.0/architecture/jfxpub-architecture.htm > Acesso em: 02 de setembro de 2011.

_____. What is JavaFX. Disponível em: <

http://download.oracle.com/javafx/2.0/overview/jfxpub-overview.htm > Acesso em: 02 de setembro de 2011.

HEPTAGON. *FDD - Feature Driven Development*. Disponível em: < http://www.heptagon.com.br/fdd > Acesso em: 04 de abril de 2011.

IMPROVE IT. *Extreme Programing*: Método de Desenvolvimento ágil. Disponível em: <http://improveit.com.br/xp> Acesso em: 04 de abril de 2011.

_____. *Scrum*: Método Ágil para Gestão e Planejamento de Projetos Disponível em: <http://improveit.com.br/scrum> Acesso em: 04 de abril de 2011.

INFO EXAME. Controle de Estoques. Disponível em: <http://info.abril.com.br/downloads/controle-de-estoque. Acesso em: 04 de abril de 2011.

KNIBERG, Henrik. *Scrum e XP direto das trincheiras*. [S.l.], 2007. Disponível em: <http://www.infoq.com/br/minibooks/scrum-xp-from-the-trenches>. Acesso em: 05 de maio de 2010.

KRUCHTEN, Phillippe. *Introdução ao RUP*: Rational Unified Process, Primeira Edição, Ciência Moderna, 2003.

LONGO, Fernando & BEZERRA, Wesley R. Engenharia de Requisitos, Disponível em: < http://www.inf.ufsc.br/~wesley/engSoft/ > Acesso em: 10 de agosto de 2011.

MARTINS, José C. C. *Técnicas para Gerenciamento de Projetos de Software*. Editora: Brasport. 2007.

MEDEIROS. Ernani Sales de. *Desenvolvendo software com UML 2.0*. São Paulo. Editora: Makron Books, 2004.

MICROSOFT. *Silverlight*. Disponível em: <http://www.microsoft.com/brasil/silverlight/> Acesso em: 02 de setembro de 2011.

ORACLE. *Applets.* Disponível em: < http://java.sun.com/applets/ > Acesso em: 11 de setembro de 2011.

PIMENTEL, Manoel. *Fluxo de Processo da FDD*. 2010. Disponível em < http://visaoagil.wordpress.com/2010/02/25/fluxo-de-processos-da-fdd/ > Acesso em: 5 de maio de 2011.

PRESSMAN, Roger S. *Engenharia de Software.* 6. Edição, Editora: McGraw-Hill, 2006.

SENSIBLE CINEMA SOFTWARE. *Sensible Cinema Software 1999*. Disponível em: <http://www.sensiblecinema.com/scsmain.htm> Acesso em: Acesso em: 07 de agosto de 2011

SMEETS Bram, BONESS Uri & BANKRAS Roald. Programando Google Web Toolkit: Do Iniciante Ao Profissional. Editora: Alta Books. 2009.

STALEY, Ted. *Planning for RIA Success*. Las Vegas: Adobe Systems Incorporated. 2007.

TABLELESS, *HTML5*: Um guia de referência para os desenvolvedores web. Disponível em: < http://tableless.com.br/html5/ > Acesso em: 02 de setembro de 2011.

TELES, Vinicius M. *Extreme Programimng*: Aprenda a encantar seus usuários desenvolvendo software com agilidade e alta qualidade. Editora: Novatec, 2006.

WIKIPÉDIA. Aplicações Web. Disponível em < http://pt.wikipedia.org/wiki/Aplica %C3%A7%C3%A3o_Web > Acesso em: 07 de agosto de 2011.

_____. *Internet Rica*. Disponível em < http://pt.wikipedia.org/wiki/Internet_rica#Interface_com_o_Usu.C3.A1rio > Acesso em: 07 de agosto de 2011.

_____. *Silverlight*. Disponível em < http://pt.wikipedia.org/wiki/Silverlight > Acesso em: 02 de setembro de 2011.

W3C. XHTML2 *Working Group Home Page*, 2007 Disponível em: < http://www.w3.org/MarkUp/ > Acesso em: 02 de setembro de 2011.

W3SCHOOL. *Ajax Introduction*. Disponível em:

<http://www.w3schools.com/ajax/ajax_intro.asp > Acesso em: 31 de agosto de 2011.

_____. *HTML5 Introduction*. Disponível em: < http://www.w3schools.com/html5/html5_intro.asp > Acesso em: 31 de agosto de 2011.