**Serviço Nacional de Aprendizagem Comercial**

**Faculdade Senac Porto Alegre**

**Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas**

**MARCELO BALDISSERA CURE**

**RELATÓRIO DE PROJETO PARCIAL**

**SISTEMA DE COLETA DE MÉTRICAS DE PROJETOS ÁGEIS PARA PROVER ESTIMATIVAS PRECISAS**

Porto Alegre

2014

**MARCELO BALDISSERA CURE**

**RELATÓRIO DE PROJETO PARCIAL**

**SISTEMA DE COLETA DE MÉTRICAS DE PROJETOS ÁGEIS PARA PROVER ESTIMATIVAS PRECISAS**

Relatório de Projeto Parcial, apresentado como requisito parcial à obtenção da aprovação do projeto de TCC1 do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, pela Faculdade Senac Porto Alegre.

Orientador: Prof. Me Luciano Zanuz

Porto Alegre

2014

**LISTA DE ILUSTRAÇÕES**

[Figura 1 – Processo básico do *Scrum* 9](#_Toc389309623)

[Figura 2 - Estrutura básica do *TDD* 13](#_Toc389309624)

[Figura 3 - Estrutura básica do sistema 15](#_Toc389309625)

[Figura 4 – Diagrama de casos de uso. 20](#_Toc389309626)

[Figura 5 – Diagrama de classes. 21](#_Toc389309627)

[Figura 6 – Diagrama Entidade Relacionamento 22](#_Toc389309628)

[Figura 7 – Protótipo da tela de cadastro de *sprints*. 22](#_Toc389309629)

[Figura 8 – Protótipo da tela de Scan do projeto. 23](#_Toc389309630)

[Figura 9 – Protótipo da tela de gráficos. 23](#_Toc389309631)

[Figura 10 – Protótipo da tela de cadastro de perfis. 24](#_Toc389309632)

[Figura 11 – Protótipo da tela de cadastro de usuários. 24](#_Toc389309633)

[Figura 12 – Protótipo da tela de cadastro de clientes. 25](#_Toc389309634)

[Figura 13 – Protótipo da tela de cadastro de projetos. 25](#_Toc389309635)

[Figura 14 – Protótipo da tela de scans pendentes. 26](#_Toc389309636)

[Figura 15 – Protótipo da tela de gráficos. 26](#_Toc389309637)

[Figura 16 – Processo de negócio do sistema 27](#_Toc389309638)

[Figura 17 – Tela de cadastro de *sprints*. 28](#_Toc389309639)

[Figura 18 – Tela de Scan do projeto. 28](#_Toc389309640)

[Figura 19 – Tela de cadastro de perfis. 29](#_Toc389309641)

[Figura 20 – Tela de cadastro de usuários. 30](#_Toc389309642)

[Figura 21 – Tela de cadastro de clientes. 30](#_Toc389309643)

[Figura 22 – Tela de cadastro de projetos. 31](#_Toc389309644)

**LISTA DE TABELAS**

[Tabela 1 – Sprint 1 18](#_Toc389309645)

[Tabela 2 – Sprint 2 18](#_Toc389309646)

[Tabela 3 – Sprint 3 19](#_Toc389309647)

[Tabela 4 – Sprint 4 19](#_Toc389309648)

[Tabela 5 – Sprint 5 20](#_Toc389309649)

[Tabela 6 – Sprint 6 20](#_Toc389309650)

[Tabela 7 - Cronograma 34](#_Toc389309651)

**LISTA DE SIGLAS**

UML Unified Modeling Language

TDD Test Driven Development

ER Entity Relationship

API Application Programming Interface

**RESUMO**

Este trabalho visa facilitar as estimativas de esforço de *sprints* para projetos que utilizam metodologias ágeis. O sistema incluirá cadastros de clientes, usuários, projetos, perfis e sprints, provendo uma visão de todos os projetos que estão acontecendo na organização. O sistema irá coletar a quantidade de pontos entregues ao final da sprint e saberá quantos pontos o time de desenvolvimento prometeu entregar. Com estes dados coletados, o sistema será capaz de produzir gráficos com informações históricas relevantes que servirão de apoio a tomada de decisão no momento de estimar uma *sprint*. Isto acarretará em segurança para o time de desenvolvimento e para o cliente que saberá exatamente o custo e tempo das *sprints* evitando imprevistos de tempo e financeiros.

SUMÁRIO

[1 APRESENTAÇÃO GERAL DO PROJETO 8](#_Toc389309712)

[2 DEFINIÇÃO DO PROBLEMA 9](#_Toc389309713)

[3 OBJETIVOS 11](#_Toc389309714)

[3.1 Objetivo Geral 11](#_Toc389309715)

[3.2 Objetivos Específicos 11](#_Toc389309716)

[4 ANÁLISE DE TECNOLOGIAS/FERRAMENTAS 12](#_Toc389309717)

[4.1 Linguagens de programação 12](#_Toc389309718)

[4.2 Ferramenta de Desenvolvimento 12](#_Toc389309719)

[4.3 Banco de Dados 12](#_Toc389309720)

[4.4 Controle de Atividades 12](#_Toc389309721)

[4.5 Versionamento do Código 13](#_Toc389309722)

[4.6 Qualidade do Código 13](#_Toc389309723)

[4.7 Ferramentas de Modelagem 14](#_Toc389309724)

[4.8 Protótipos de tela 14](#_Toc389309725)

[5 DESCRIÇÃO DA SOLUÇÃO 15](#_Toc389309726)

[6.1 Artefatos 16](#_Toc389309727)

[6.1.1 Product Backlog 16](#_Toc389309728)

[6.1.2 Sprint Backlog 16](#_Toc389309729)

[6.1.3 User Stories (estórias) 16](#_Toc389309730)

[6.1.4 Diagrama de Caso de Uso 17](#_Toc389309731)

[6.1.5 Diagrama de Classes 17](#_Toc389309732)

[7 ARQUITETURA DO SISTEMA 18](#_Toc389309733)

[7.1 Modelagem Funcional 18](#_Toc389309734)

[7.1.1 Sprint 1 18](#_Toc389309735)

[7.1.2 Sprint 2 18](#_Toc389309736)

[7.1.3 Sprint 3 19](#_Toc389309737)

[7.1.4 Sprint 4 19](#_Toc389309738)

[7.1.5 Sprint 5 20](#_Toc389309739)

[7.1.6 Sprint 6 20](#_Toc389309740)

[7.2 Modelos UML 20](#_Toc389309741)

[7.3 Modelagem de Dados 22](#_Toc389309742)

[7.4 Interface Gráfica 22](#_Toc389309743)

[7.4.1 Cadastro de Sprints 22](#_Toc389309744)

[7.4.2 Scan de projeto 23](#_Toc389309745)

[7.4.3 Gráficos 23](#_Toc389309746)

[7.4.4 Cadastro de perfis 24](#_Toc389309747)

[7.4.5 Cadastro de usuários 24](#_Toc389309748)

[7.4.6 Cadastro de clientes 25](#_Toc389309749)

[7.4.7 Cadastro de projetos 25](#_Toc389309750)

[7.4.8 Consulta de scans pendentes 26](#_Toc389309751)

[7.4.9 Gráficos 26](#_Toc389309752)

[7.5 Processo de Negócio 27](#_Toc389309753)

[8 FUNCIONAMENTO DO SISTEMA 28](#_Toc389309754)

[8.1 Membro do time 28](#_Toc389309755)

[8.1.1 Cadastro de Sprints 28](#_Toc389309756)

[8.1.2 Scan de projeto 28](#_Toc389309757)

[8.1.3 Gráficos 28](#_Toc389309758)

[8.2 Administrador 29](#_Toc389309759)

[8.2.1 Cadastro de perfis 29](#_Toc389309760)

[8.2.2 Cadastro de usuários 29](#_Toc389309761)

[8.2.3 Cadastro de clientes 30](#_Toc389309762)

[8.2.4 Cadastro de projetos 31](#_Toc389309763)

[8.2.5 Consulta de scans pendentes 31](#_Toc389309764)

[8.2.6 Gráficos 31](#_Toc389309765)

[9 VALIDAÇÃO 33](#_Toc389309766)

[10 CRONOGRAMA 34](#_Toc389309767)

[REFERÊNCIAS BIBLIOGRÁFICAS 36](#_Toc389309768)

[COMPONENTES REUTILIZADOS 38](#_Toc389309769)

# 1 APRESENTAÇÃO GERAL DO PROJETO

No desenvolvimento de software em geral, a estimativa de esforço para implementar um requisito pode afetar o projeto em alguns fatores, como o prazo que pode ser comprometido, podendo acarretar em atrasos de entrega. As estimativas são peça fundamental para o sucesso de um projeto.

Atualmente, as metodologias ágeis estão se tornando cada vez mais populares na indústria do desenvolvimento de software (PLANBOX, 2013). Estas metodologias tendem a diminuir o risco do projeto fazendo entregas periódicas chamadas de iterações.

Seguindo na linha das metodologias ágeis, uma das mais conhecidas e utilizadas é o *Scrum*. Esta metodologia consiste em fazer entregas frequentes de funcionalidades, reuniões diárias para sincronização da equipe, entre outras práticas. (METHODOLOGY, 2009) Ao final de cada iteração, chamada de *sprint*, funcionalidades completas são entregues, possibilitando ao cliente prover um feedback e planejar as próximas *sprints.* (KEN SCHWABER, 2013).

Projetos executados com *Scrum* normalmente possuem *sprints* de períodos curtos, onde os requisitos são quebradas em estórias. Cada estória é estimada, normalmente, utilizando uma metodologia chamada *Planning Poker*, onde cada membro do time estima cada estória em pontos*.*

Mesmo utilizando metodologias ágeis, as estimativas dificilmente são precisas, então os projetos carecem de métodos para prover qualidade ao estimar estórias e *sprints.*

A proposta deste sistema é coletar a quantidade de pontos entregues ao final de cada *sprint.* Com estes dados coletados, o sistema poderá prover a quantidade média de pontos que a equipe é capaz de entregar em uma *sprint* futura.Isto acarreta em mais segurança para o fornecedor e para o cliente. A equipe de desenvolvimento saberá quantos pontos poderá entregar e o cliente, juntamente com a equipe, poderá priorizar os requisitos para a *sprint* evitando atrasos e custo não planejado.

# 2 DEFINIÇÃO DO PROBLEMA

Um importante ingrediente para o sucesso de um projeto é a estimativa de esforço para a execução das tarefas, pois ela está diretamente ligada com o custo e com o prazo do projeto. De acordo com Arnaut (2008) apenas 35% dos projetos tiveram sucesso em 2006. Quando os requisitos de um projeto são definidos e quebrados em tarefas menores, estes necessitam de uma estimativa de esforço para executá-los, esta define quanto tempo e quanto dinheiro o cliente irá gastar.

Se o time de desenvolvimento de software pudesse prover estimativas de esforço precisas, este seria um cenário ideal tanto para o cliente quanto para o fornecedor do *software*. Os projetos poderiam ser vendidos com mais segurança para ambos os lados, cliente e time, estipulando o prazo e o custo sem surpresas ao final do projeto.

Projetos executados com metodologias ágeis, como o *Scrum*, tendem a ter entregas periódicas de funcionalidades específicas acordadas entre o cliente e o time de desenvolvimento, conforme ilustrado na Figura 1. Estes ciclos são chamados de *sprints*.

Normalmente *sprints* são períodos curtos, como por exemplo, duas semanas. Ao final de cada *sprint* as funcionalidades são entregues ao cliente, e uma reunião entre cliente e equipe é feita para definir as estórias que deverão ser entregues no próximo ciclo.

Figura 1 – Processo básico do *Scrum*

Fonte: (Eclipse Org, 2009)

Estórias são os itens que vão ficar, inicialmente, no *backlog* do projeto. O time escreve as estórias baseando-se no ponto de vista do cliente. Pode-se considerar que a estória é um requisito do sistema. (METHODOLOGY, 2009). Por exemplo, ao desenvolver uma calculadora, podemos definir as estórias como: subtração, adição, etc.

As estórias priorizadas para serem executadas na *sprint* devem ser estimadas para a equipe poder planejar de forma organizada a execução das mesmas. Normalmente as estimativas são feitas utilizando a metodologia *Planning Poker*, onde cada estória é estimada em pontos, que variam de 0 até 100 pontos (planningpoker.com, 2014).

O *Planning Poker* é uma metodologia utilizada para auxiliar nas estimativas de esforço. Depois que a estória é apresentada para os membros do time, estes discutem a estória para ter certeza de que todos entenderam. Cada membro do time escolhe uma pontuação representando o esforço que será necessário para implementar a estória, e não mostra para nenhum outro membro. Após cada integrante do time ter escolhido uma pontuação, denominada *story point*, os mesmos discutem os resultados justificando-os e chegam a um denominador comum. Normalmente, diferentes membros do time provém estimativas diferentes, por isto a discussão é importante para que todos cheguem a um consenso. Então o time passa a ter a estimativa para aquela estória específica.

É comum que a pontuação estimada no *Planning Poker* para a estória não seja precisa. Por exemplo, o time estimou a estória em 5 pontos, mas esta poderia ser estimada em 3 pontos, pois o esforço gasto foi proporcional a uma estória de 3 pontos.

Este trabalho visa facilitar as estimativas de *sprints* baseando-se em dados históricos de *sprints* passadas. Ao final de cada ciclo o sistema deverá coletar quantos pontos a equipe prometeu entregar ao final da *sprint* e quantos pontos realmente foram entregues. Estes dados coletados são métricas relacionadas ao desempenho do time e a capacidade de abstrair o problema do cliente e codificar as funcionalidades de maneira eficiente. É possível extrair informações inteligentes destes dados, pois estes são o histórico das *sprints* que já passaram, isto significa que a equipe pode extrair um aprendizado destas estatísticas que poderão ser utilizados como suporte para futuras estimativas.

Mesmo utilizando técnicas como o *Planning Poker*, as estimativas nem sempre são precisas. Isto se torna um problema para o time de desenvolvimento sempre que for necessário estimar o esforço de estórias para uma *sprint*, e o cliente que não terá segurança para prever um *release* e custo.

# 3 OBJETIVOS

Este capítulo descreve os objetivos gerais e específicos do projeto.

## 3.1 Objetivo Geral

O objetivo geral deste trabalho é melhorar as estimativas de *sprints*, onde as estórias são baseadas no *planning poker,* a partir de dados históricos referentes a estimativas de *sprints* anteriores.

## 3.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

1. Prover confiança para o time de desenvolvimento e para o cliente ao planejar a *sprint*;
2. Prover condições para melhorar a estimativa da *sprint*;
3. Integrar o sistema com o software *Trello* ([trello.com](http://www.trello.com/), 2014);
4. Coletar informações pertinentes dos *boards* do *Trello*;
5. Gerar gráficos para auxiliar nas estimativas futuras;
6. Prover, através de quantidade de testes unitários, uma idéia geral da qualidade do código.

# 4 ANÁLISE DE TECNOLOGIAS/FERRAMENTAS

Este capítulo descreve as tecnologias que serão utilizadas para o desenvolvimento do projeto, incluindo tecnologias de desenvolvimento, modelagem e técnicas.

## 4.1 Linguagens de programação

Este projeto será desenvolvido utilizando a linguagem de programação *Python*, pois ela provém facilidade no desenvolvimento, possui bibliotecas poderosas e as APIs são simples e bem documentadas, assim como a própria linguagem. É fácil de configurar bibliotecas externas. Python é uma linguagem de programação grátis, onde não é necessário pagar licença. (python.org.br, 2014).

Será utilizado o framework *Django* para auxiliar no desenvolvimento das páginas, onde o desenvolvedor pode definir o modelo de dados utilizando a linguagem *Python*. Este *framework* também provém uma API poderosa de integração com o banco de dados. *Django* é um *framework* para desenvolvimento web com *Python* e serve, principalmente, para agilizar o desenvolvimento. (djangoproject.com, 2014).

## 4.2 Ferramenta de Desenvolvimento

O desenvolvimento será feito utilizando uma ferramenta chamada *Sublime*. Esta ferramenta facilita o desenvolvimento pois ela oferece muitas funcionalidades úteis como *auto complete*, lista de arquivos do projeto, multi seleção, entre outras. Existem muitas extensões para o *Sublime* auxiliar no desenvolvimento, inclusive o próprio desenvolvedor pode criar *plug ins* e publicá-los. O *Sublime* é uma ferramenta extremamente leve e grátis. ([sublimetext.com](http://www.sublimetext.com/), 2014).

## 4.3 Banco de Dados

Os dados serão guardados no banco de dados PostgreSQL por ser poderoso e grátis. O PostgreSQL provém a capacidade de escrever consultas complexas e retornar os dados em um formato específico para que o sistema possa utilizá-los de maneira mais facil (postgresql.org, 2014).

O PostgreSQL é um banco de dados relacional grátis de fácil instalação e manutenção. Ele pode ser instalado na maioria dos sistemas operacionais.

## 4.4 Controle de Atividades

Para a parte de controle de atividades e *sprints*, será utilizado o software *Trello* ([trello.com](http://www.trello.com/), 2014), o qual é usado como um *Kanban*. O *Trello* é um software grátis que possibilita a criação de *boards*, *lists*, *cards*. Além disto também é possível adicionar anexos aos *cards* e outras funcionalidades interessantes. Este possibilitará fazer o controle das *sprints* de forma organizada e visual, e as tarefas de *sprints* passadas podem ser arquivadas de forma a manter um histórico.

Explicar rest

## 4.5 Versionamento do Código

Para controle de versão de código e da própria documentação do projeto, será utilizada a ferramenta *GitHub* ([github.com](http://www.github.com/), 2014). É um software grátis, onde, registrando um usuário, este pode cadastrar vários projetos, possibilitando a criação de *bugs*, melhorias, etc. Outros usuarios podem contribuir para o projeto, etc. O *GitHub* será importante para o projeto na parte de versionamento de código, onde tudo será salvo na nuvem. Isto evitará problemas como perder o código e a documentação em caso de catástrofes.

## 4.6 Qualidade do Código

Falando de qualidade de código, o sistema será implementado utilizando um método de desenvolvimento chamado *TDD*, que basicamente consiste em desenvolver um sistema baseando-se em testes. Antes de desenvolver uma funcionalidade, escreve-se testes unitários para esta. Estes testes, inicialmente, estarão falhando, pois as funcionalidades ainda não foram implementadas. O primeiro passo seria fazer estes testes passar, implementando as funcionalidades de acordo com tal necessidade. Assim o desenvolvedor tem uma segurança em saber que aquela funcionalidade já possui testes unitários, provendo confiança também para refatorar o código. A qualidade do código pode ser medida pela cobertura de testes para cada módulo ou classe do sistema. O processo básico do *TDD* está descrito na Figura 2. (SCOTT W. AMBLER, 2006).

Figura 2 - Estrutura básica do *TDD*



Fonte: (SCOTT W. AMBLER, 2006)

## 4.7 Ferramentas de Modelagem

A criação de modelos e diagramas do sistema será feita utilizando a ferramenta online *Cacoo*. Esta é uma ferramenta que possibilita a criação de diversos tipos de diagramas, como por exemplo diagrama ER e UML. Cacoo é uma ferramenta gratis. (cacoo.com, 2014).

Serão gerados os diagramas de casos de uso e diagrama e classe. A modelagem de banco será representada pelo diagrama entidade relacionamento.

## 4.8 Protótipos de tela

Para a criação dos protótipos de tela será utilizado o *NinjaMock*. Esta ferramenta possibilita criar protótipos de telas de várias plataformas, como *Web*, *Android*, *iOS*, *WindowsPhone*, etc. O *NinjaMock* é grátis e possibilita que as imagens possam ser exportadas em PDF ou PNG. (ninjamock.com, 2014).

# 5 DESCRIÇÃO DA SOLUÇÃO

Esta solução se baseará em dados históricos extraídos de estimativas de *sprints*. É um sistema específico para o auxílio de estimativas em projetos ágeis, onde as estórias são estimadas utilizando o *planning poker*. As estimativas das *sprints*, medidas em pontos, serão extraídas e salvas em um banco de dados. Estes dados históricos serão utilizados para geração de relatórios com a finalidade de prover auxílio ao time no momento de planejar uma nova *sprint*.

O sistema será restringido somente para pessoas que possuem acesso. Os membros do time terão acesso somente aos projetos que participam, onde terão a opção de fazer a coleta de dados e visualizar relatórios. O sistema incluirá a área do administrador, onde este poderá cadastrar perfis, usuários, clientes, projetos e adicionar recursos a estes projetos. O usuário com perfil administrador poderá consultar se existe algum projeto com o scan pendente. Esta consulta verificará se algum dos projetos tem uma *sprint* sendo finalizada naquele dia e o time ainda não executou o processo de scan para coletar os dados. Assim o administrador poderá ter um controle melhor sobre o sistema e os projetos que estão em progresso.

Considera-se que os projetos tenham acompanhamento na ferramenta online *Trello*. Esta ferramenta é utilizada para montar quadros *Kanban*, que possibilita a criação de *boards* onde o usuário pode criar *lists*, *cards*, *labels*, adicionar informações, anexar arquivos, etc. (trello.com, 2014).

O processo de coleta de métricas irá acessar a *API* do *Trello*. O usuário selecionará um projeto e iniciará a coleta. O sistema irá buscar os *cards* que estiverem na última coluna do *board*, que é a coluna “*Done*”, onde estarão as tarefas que foram concluídas na *sprint*.

O sistema irá coletar os dados de cada *card*, como a sua identificação, tempo de duração da tarefa, *Story Points*, linguagem de programação utilizada e a quantidade de testes unitários criados para aquela tarefa. A última etapa da coleta de dados será salvá-los no banco de dados *PostgreSQL*. Cada execução referente a um projeto será considerada uma *sprint*.

Serão disponibilizados gráficos com métricas referentes a um projeto específico ou a todos os projetos da empresa. Baseando-se nos dados coletados de *sprints* passadas, o sistema poderá prover estes gráficos com diferentes visões destes dados, gerando informações que apoiam a tomada de decisão. A estrutura básica do sistema está detalhada na Figura 3.

Figura 3 - Estrutura básica do sistema



Fonte: Autoria própria

**6 ABORDAGEM DE DESENVOLVIMENTO**

O projeto será desenvolvido utilizando alguma partes do *Scrum* e a utilização do *Kanban*. O *Scrum* é uma metodologia ágil de desenvolvimento de software que normalmente é utilizada em times pequenos, então algumas partes dele passam a ser relevantes para o cenário do projeto.

Utilizando o *Kanban*, será possível ter, de forma visual, um acompanhamento preciso do progresso da *sprint* corrente. O *Kanban* possibilita a identificação de gargalos no desenvolvimento, e através de algumas técnicas no gerenciamento do quadro, ele possibilita a melhoria no fluxo das atividades. O objetivo principal é finalizar as tarefas que já estão mais no lado direito do quadro, focando o trabalho nos itens que já estão sendo executados, resultando em mais itens entregues e maior fluxo de atividades.

Os ciclos, conhecidos como *sprints*, serão utilizados priorizando itens do *backlog* para entregas periódicas. Serão executadas *sprints* de duas semanas, onde ao final de cada ciclo as funcionalidades priorizadas serão entregues.

O projeto basicamente será dividido nas etapas descritas abaixo:

1. Criação do modelo de dados;
2. Estrutura básica do projeto com *Django*;
3. Criação de telas principais;
4. Integração com a *API* do *Trello*;
5. Processo de coleta de dados;
6. Geração dos gráficos.

## 6.1 Artefatos

A seguir, segue uma descrição dos artefatos gerados pelo projeto.

### 6.1.1 Product Backlog

O *Product Backlog* é uma lista, ordenada por prioridade, de tudo que é necessário para a construção do produto. O *Product Backlog* nunca está completo, ele é dinâmico e está mudando constantemente identificando o que o produto precisa para ser completo e apropriado.

### 6.1.2 Sprint Backlog

O *Sprint Backlog* é uma lista de itens do *Product Backlog* priorizados para serem finalizados na *sprint* corrente. Os itens são colocados na lista de forma ordenada indicando a prioridade de cada ítem.

### 6.1.3 User Stories (estórias)

Estórias baseiam-se no ponto de vista do cliente. Pode-se considerar que a estória é um requisito do sistema. As estórias basicamente podem ser definidas pelo time, baseando-se na perspectiva de quem vai utilizar a funcionalidade gerada por esta estória. Por exemplo, ao desenvolver uma calculadora, podemos definir as estórias como: subtração, adição, etc. (METHODOLOGY, scrum, 2009).

### 6.1.4 Diagrama de Caso de Uso

O digrama de caso é um artefato da UML, que visa fazer a representação dos casos de uso do projeto, indicando os atores e as ações feitas de modo claro a identificar papéis e funcionalidades do sistema.

### 6.1.5 Diagrama de Classes

O digrama de classes é um artefato da UML, que visa mostrar de maneira detalhada a estrutura de classes e a comunicação entre elas. Este diagrama é baseado em um sistema escrito em uma linguagem orientada a objetos.

# 7 ARQUITETURA DO SISTEMA

Neste capitulo será demonstrada a arquitetura básica do sistema, bem como os artefatos que serão gerados ajudando a descrever as funcionalidades do sistema.

## 7.1 Modelagem Funcional

Na modelagem funcional serão utilizadas as *User Stories*, *Product Backlog* e *Sprint Backlog*. Os ítens abaixo descreverão as *User Stories* com sua descrição detalhada priorizadas para cada *sprint*.

### 7.1.1 Sprint 1

Tabela 1 – Sprint 1

|  |  |
| --- | --- |
| Cadastros Básicos | |
| Cadastro de usuários | Tela para o perfil administrador. Mostrar os campos:  name, username, password, email, profile. Este formulário  ficará na parte esquerda da tela. No outro lado aparecerá  a lista de usuários já cadastrados. |
| Cadastro de perfis | Tela para o perfil administrador. Mostrar o campo:  descrição. Este formulário ficará na parte esquerda da  tela. No outro lado aparecerá a lista de perfis já cadastrados. |
| Tela de login | Tela para o perfil administrador. O processo de  login irá encaminhar o usuário para a tela especifica de  acordo com o seu perfil (admin ou time). Será  desenvolvido um controle de sessão utilizando uma  tabela no banco de dados e cookies do browser,  criptografando a senha utilizando a biblioteca PyCrypto. |
| Cadastro de clientes | Tela para o perfil administrador. Mostrar os campos:  name, country e Operation area. Este formulário ficará na  parte esquerda da tela. No outro lado aparecerá a  lista de clientes já cadastrados. |
| Cadastro de projetos | Tela para o perfil administrador. Mostrar os campos:  name, trello board name, customer e listar todos os usuários com perfil  “time”, possibilitando que estes sejam alocados para o  projeto. Este formulário ficará na parte esquerda da  tela. No outro lado aparecerá a lista de projetos já  cadastrados. |

### 7.1.2 Sprint 2

Tabela 2 – Sprint 2

|  |  |
| --- | --- |
| Scan e Cadastro de sprints | |
| Cadastro de sprint | Tela para o perfil time. Mostrar os campos:  Ptoject, description, start date, end date, estimated  points. Este formulário ficará na parte esquerda da  tela. |
| Scan parte 1 | Tela para o perfil time. Mostrar o campo: project,  onde o usuário selecionará o projeto e o sistema fará o  scan de acordo com o projeto. O processo de scan inicial  será simples, este acessará o *board* do projeto no  *Trello,* as informações coletadas serão mostradas na  tela de maneira simples. Ao final do processo os  dados referentes ao processo de scan serão exibidos na tela  com uma mensagem de confirmação. |

### 7.1.3 Sprint 3

Tabela 3 – Sprint 3

|  |  |
| --- | --- |
| Scan final e gráficos do administrador | |
| Scan parte 2 | Parte final do scan. Modificar o processo para salvar os  dados no banco de dados, este passo deve atualizar o  registro da *sprint* que foi previamente cadastrada. |
| Quantidade de pontos  entregues X *sprint* | Tela para o perfil administrador. Gerar um gráfico de  linhas para cada time mostrando a quantidade de  pontos entregues por sprint. |
| Quantidade de testes  unitários criados X *sprint* | Tela para o perfil administrador. Gerar um gráfico de  linhas para cada time mostrando a quantidade de  testes unitários criados por sprint. |

### 7.1.4 Sprint 4

Tabela 4 – Sprint 4

|  |  |
| --- | --- |
| Gráficos do administrador final | |
| Quantidade de pontos  estimados X pontos  entregues | Tela para o perfil administrador. Gerar um gráfico de  linhas para cada time mostrando a quantidade de  pontos estimados por quantidade de pontos entregues. |
| Quantidade de pontos  entregues X testes unitários  para cada *sprint* | Tela para o perfil administrador. Gerar um gráfico de  linhas para cada time mostrando a quantidade de  testes unitários criados por pontos entregues por *sprint*. |
| Quantidade de  estórias entregues X  sprint | Tela para o perfil administrador. Gerar um gráfico de  linhas para cada time mostrando a quantidade de  pontos entregues por *sprint*. |

### 7.1.5 Sprint 5

Tabela 5 – Sprint 5

|  |  |
| --- | --- |
| Gráficos do time | |
| Quantidade de pontos  entregues X *sprint* | Tela para o perfil time. Gerar um gráfico de linhas  mostrando a quantidade de pontos entregues por *sprint*. |
| Quantidade de testes  unitários criados X *sprint* | Tela para o perfil time. Gerar um gráfico de linhas  mostrando a quantidade de testes unitários criados por  *sprint*. |
| Quantidade de pontos  estimados X pontos  entregues | Tela para o perfil time. Gerar um gráfico de linhas  mostrando a quantidade de pontos estimados por  quantidade de pontos entregues. |
| Quantidade de pontos  entregues X testes unitários  para cada *sprint* | Tela para o perfil time. Gerar um gráfico de linhas  mostrando a quantidade de testes unitários criados por  pontos entregues por *sprint*. |

### 7.1.6 Sprint 6

Tabela 6 – Sprint 6

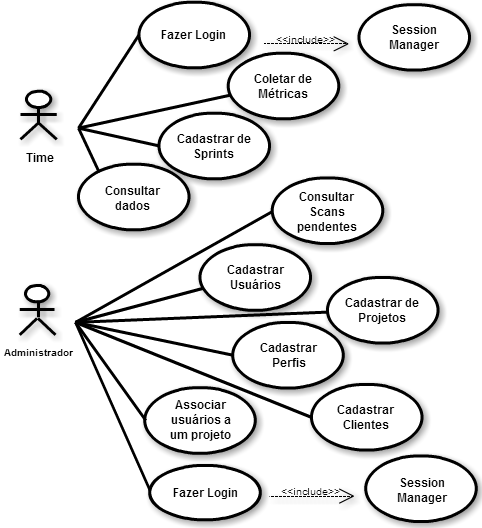
|  |  |
| --- | --- |
| Gráficos time final e consulta scans pendentes | |
| Quantidade de pontos  entregues X testes unitários  para cada *sprint* | Tela para o perfil time. Gerar um gráfico de linhas  mostrando a quantidade de testes unitários criados por  pontos entregues por *sprint*. |
| Quantidade de  estórias entregues por  *sprint* | Tela para o perfil time. Gerar um gráfico de linhas  mostrando a quantidade de estórias criadas por *sprint*. |
| Consulta times que  ainda não rodaram o  scan no final da *sprint* | Tela para o perfil administrador. Criar uma tela que  listará as sprints que estão sendo finalizadas no dia  corrente e que o time ainda não executou o processo de  scan. |

## 7.2 Modelos UML

Na modelagem UML, serão gerados diagrama de classe e diagrama de casos de uso. Inicialmente será criada uma versão simples, estes modelos evoluirão conforme as *sprints* são finalizadas.

O diagrama de casos de uso está descrito na figura 4.

Figura 4 – Diagrama de casos de uso.



Fonte: Autoria própria.

O diagrama de classes está descrito na figura 5.

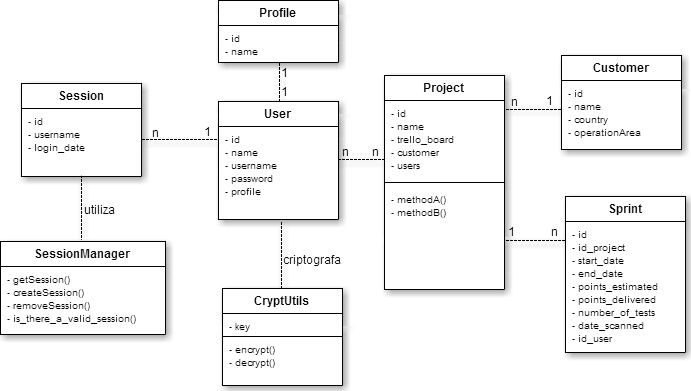
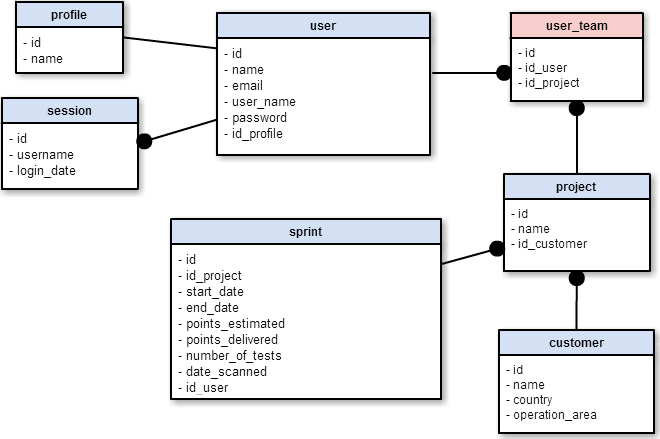


Figura 5 – Diagrama de classes.

## 7.3 Modelagem de Dados

O modelo Entidade Relacionamento será gerado para demonstrar a modelagem de dados do sistema. Inicialmente será criada uma versão simples, este diagrama evoluirá conforme as *sprints* são finalizadas.

Figura 6 – Diagrama Entidade Relacionamento



Fonte: Autoria própria

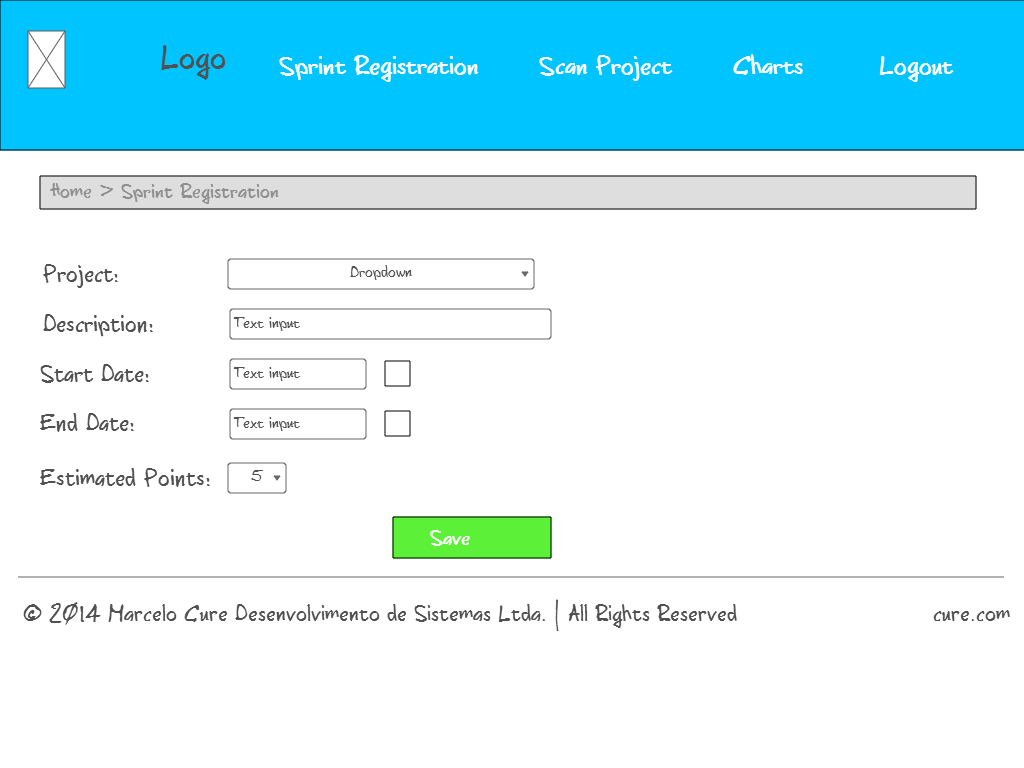
## 7.4 Interface Gráfica

Serão criados protótipos de telas do sistema para ter, de uma maneira visual, uma idéia de como as telas serão construídas. Desta maneira será possível fazer uma análise de usabilidade para deixar a navegação/execução das tarefas clara e objetiva.

### 7.4.1 Cadastro de Sprints

A figura 7 mostra o protótipo da tela de cadastro de *sprints*.

Figura 7 – Protótipo da tela de cadastro de *sprints*.

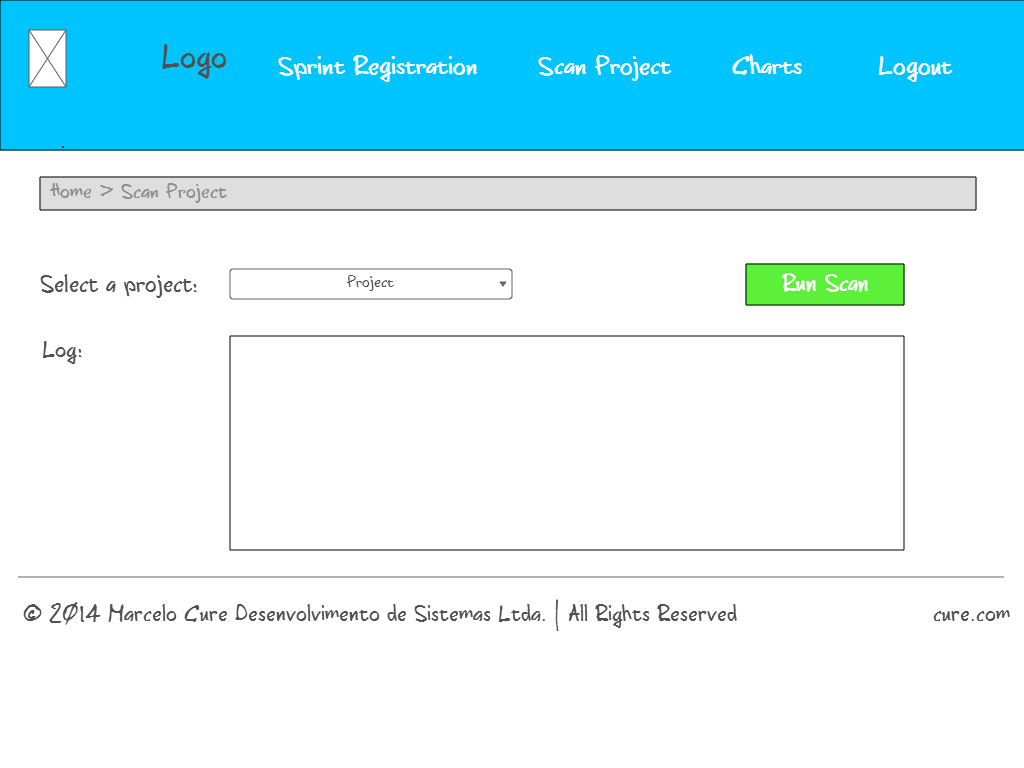


Fonte: Autoria própria.

### 7.4.2 Scan de projeto

A figura 8 mostra o protótipo da tela de scan do projeto.

Figura 8 – Protótipo da tela de Scan do projeto.

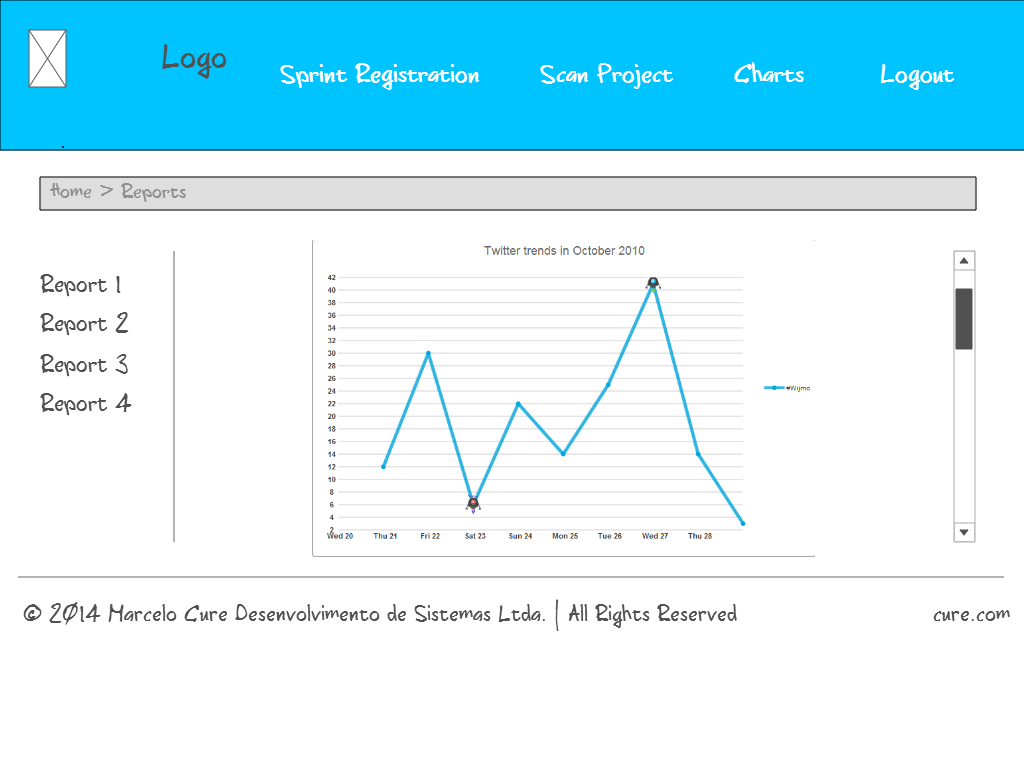


Fonte: Autoria própria.

### 7.4.3 Gráficos

A figura 9 mostra o protótipo da tela de gráficos.

Figura 9 – Protótipo da tela de gráficos.

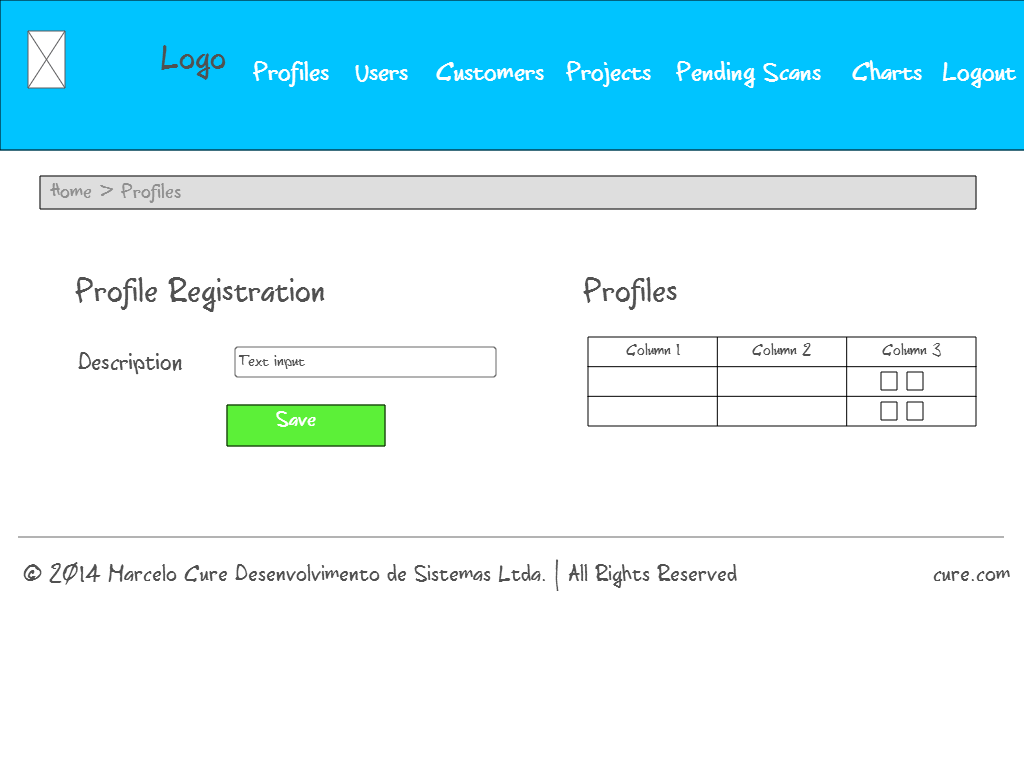


Fonte: Autoria própria.

### 7.4.4 Cadastro de perfis

A figura 10 mostra o protótipo da tela de cadastro de perfis.

Figura 10 – Protótipo da tela de cadastro de perfis.

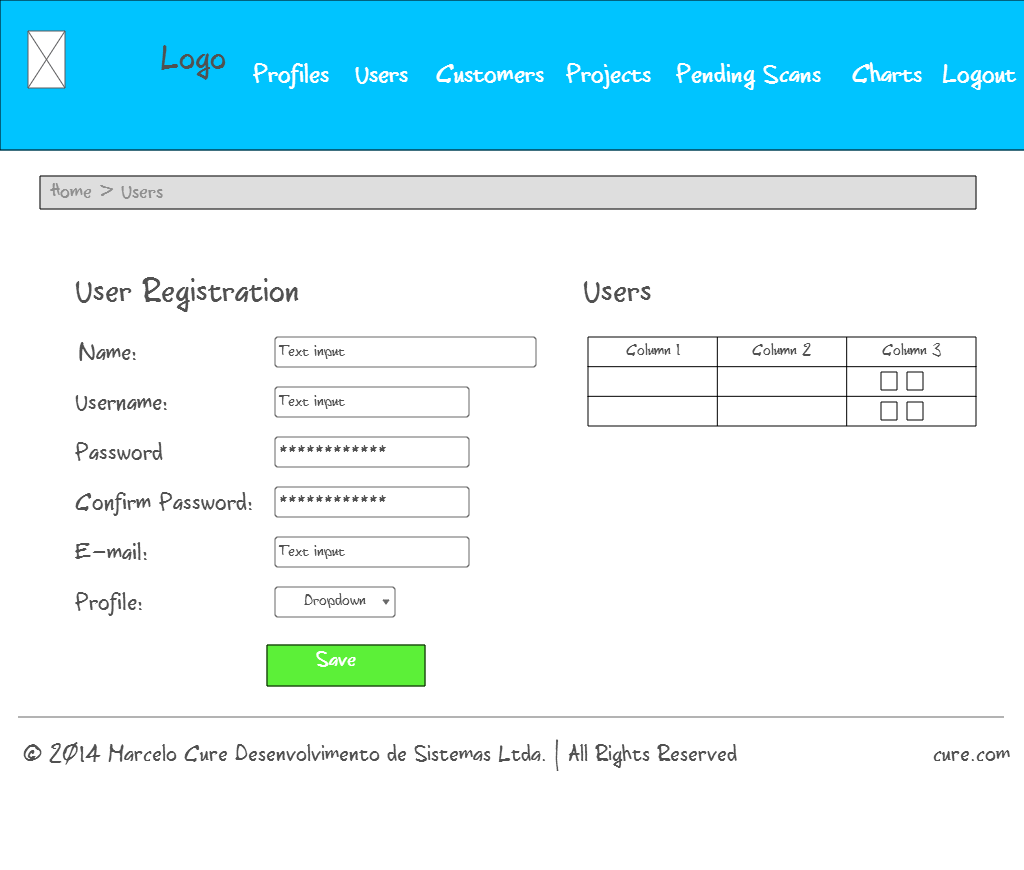


Fonte: Autoria própria.

### 7.4.5 Cadastro de usuários

A figura 11 mostra o protótipo da tela de cadastro de usuários.

Figura 11 – Protótipo da tela de cadastro de usuários.

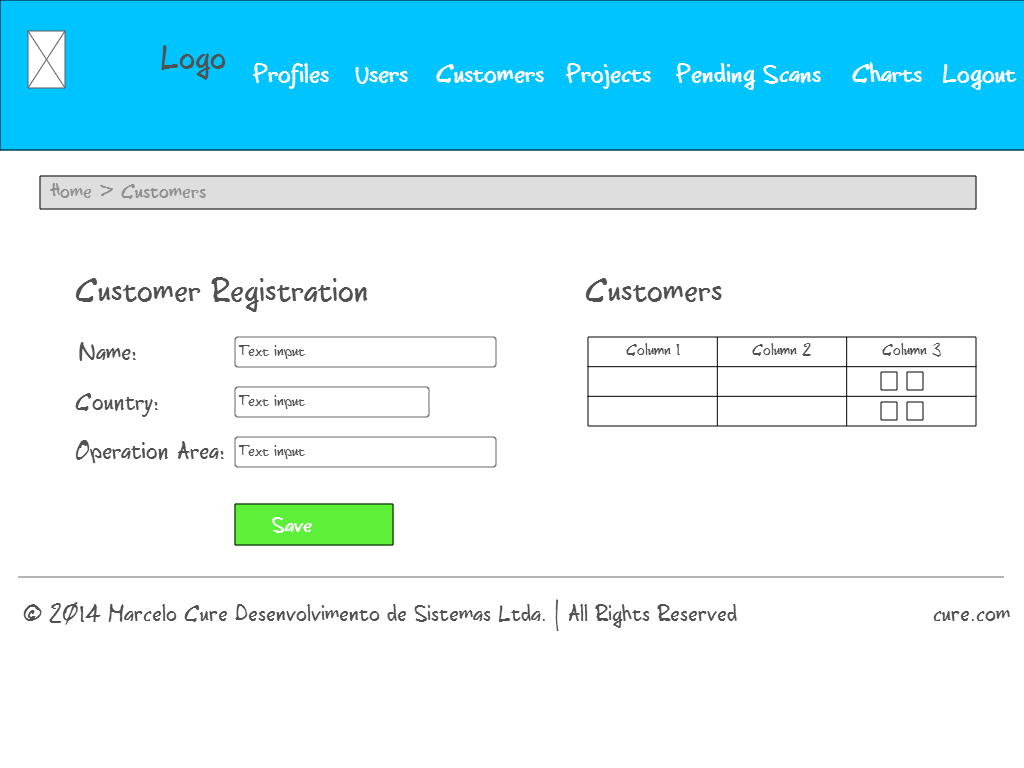


Fonte: Autoria própria.

### 7.4.6 Cadastro de clientes

A figura 12 mostra o protótipo da tela de cadastro de clientes.

Figura 12 – Protótipo da tela de cadastro de clientes.

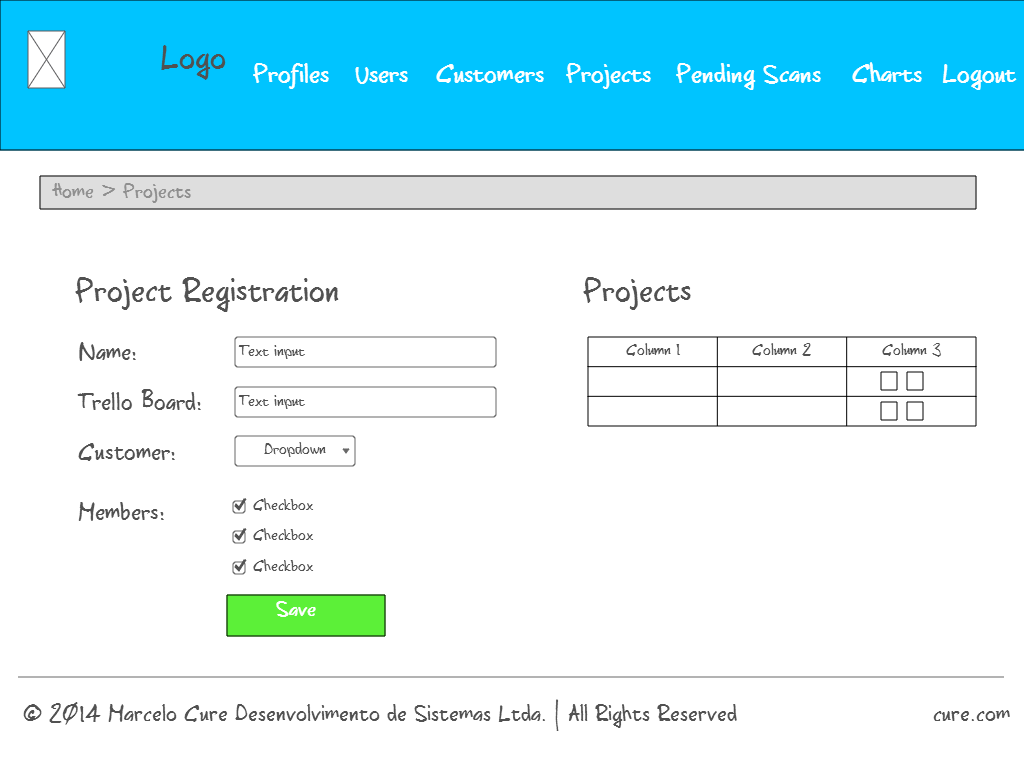


Fonte: Autoria própria.

### 7.4.7 Cadastro de projetos

A figura 13 mostra o protótipo da tela de cadastro de projetos.

Figura 13 – Protótipo da tela de cadastro de projetos.

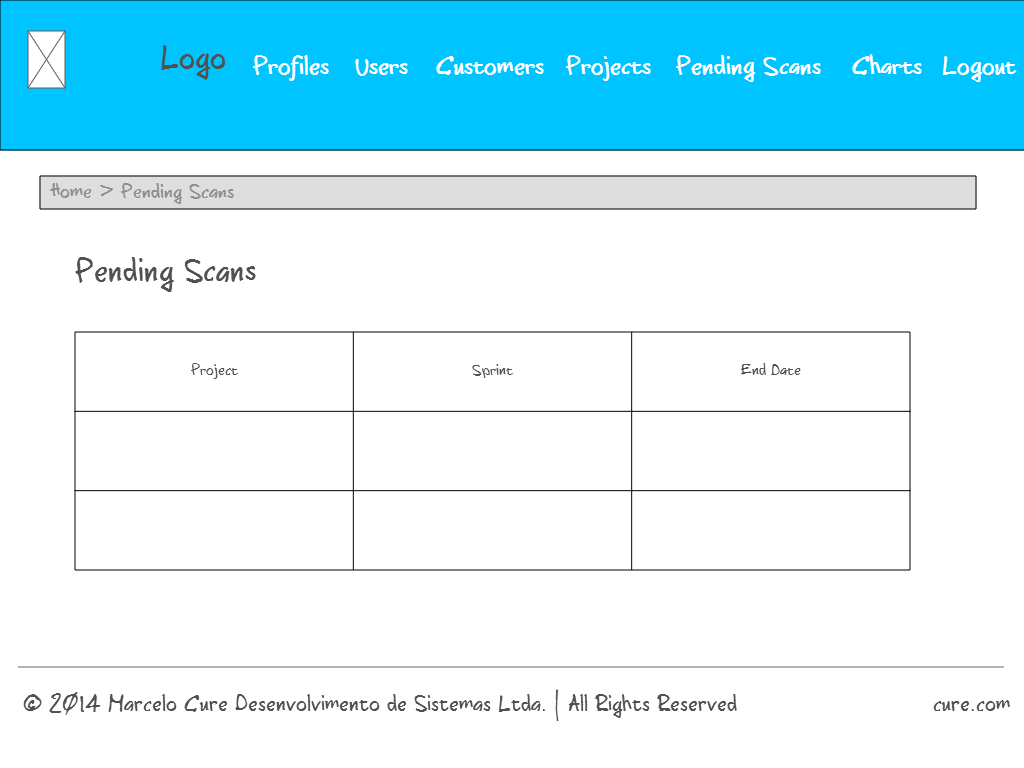


Fonte: Autoria própria.

### 7.4.8 Consulta de scans pendentes

A figura 14 mostra o protótipo da tela de scans pendentes.

Figura 14 – Protótipo da tela de scans pendentes.

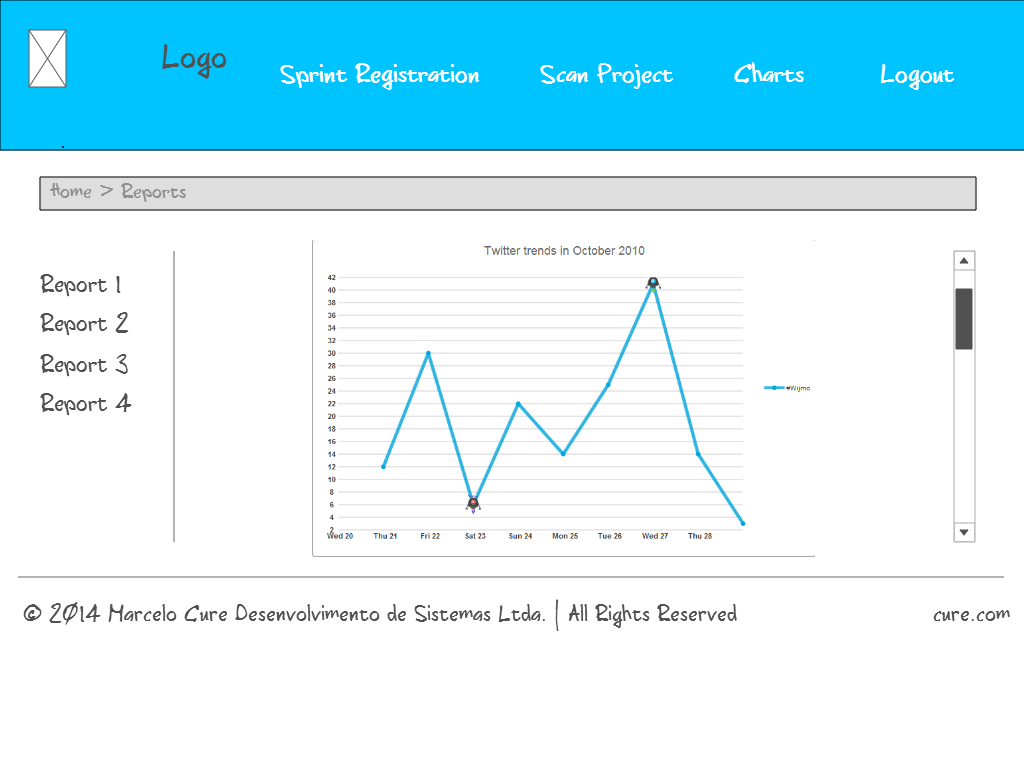


Fonte: Autoria própria.

### 7.4.9 Gráficos

A figura 15 mostra o protótipo da tela de gráficos.

Figura 15 – Protótipo da tela de gráficos.

.

Fonte: Autoria própria.

## 7.5 Processo de Negócio

O processo de negócio está descrito na figura 16.

Figura 16 – Processo de negócio do sistema



Fonte: Autoria própria.

# 8 FUNCIONAMENTO DO SISTEMA

O sistema terá, inicialmente, dois perfis de acesso, o membro do time e o administrador.

## 8.1 Membro do time

O membro do time poderá participará de um ou mais projetos e suas funcionalidades estão descritas nos ítens.

### 8.1.1 Cadastro de Sprints

Usuários com o perfil do time de desenvolvimento serão capazes de cadastrar as *sprints* do projeto. O cadastro de *sprints* incluirá o projeto em questão, operíodo de duração, quantidade de pontos estimados e a descrição das mesma. A figura 17 mostra a tela de cadastro de *sprints*.

Figura 17 – Tela de cadastro de *sprints*.

Fonte: Autoria própria.

### 8.1.2 Scan de projeto

Outra funcionalidade disponibilizada para o membro do time é a opção de fazer o scan do projeto, onde o usuário selecionará um projeto e o sistema acessará o *board* do projeto do *Trello* e fará uma leitura de cada *card* da coluna *done*. Cada um destes cards representará uma estória da *sprint,* contendo informações relevantes da estória como quantidade de dias que o time demorou para implementar a estória, quantidade de testes unitários criados, número de pontos estimados e descrição. Esta leitura coletará a quantidade total de pontos entregues na *sprint*. Com isto, o sistema irá concluir a *sprint*, colocando em seu cadastro os dados coletados que serão utilizados para gerar os gráficos do sistema. A figura 18 mostra a tela de scan do projeto.

Figura 18 – Tela de Scan do projeto.

Fonte: Autoria própria.

### 8.1.3 Gráficos

Com os dados coletados de *sprints* passadas, o sistema disponibilizará gráficos sumarizando-os conforme descrito abaixo:

1. Quantidade de pontos entregues X *sprint*;
2. Quantidade de testes unitários criados X *sprint*;
3. Quantidade de pontos estimados X quantidade de pontos entregues para cada *sprint*;
4. Quantidade de pontos entregues X testes unitários para cada *sprint*;
5. Quantidade de estórias entregues por *sprint* por projeto.

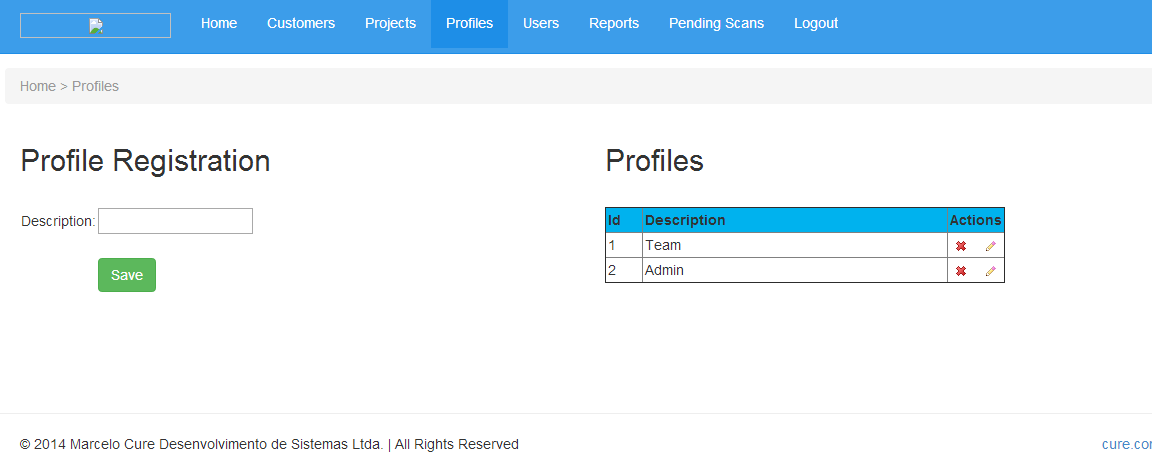
## 8.2 Administrador

O administrador terá uma visão geral de todos os projetos da organização e fará os cadastros básicos do sistema, as funcionalidades do administrador estão descritas nos ítens abaixo.

### 8.2.1 Cadastro de perfis

Na tela de cadastro de perfil o administrador será capaz de cadastrar os perfis de acesso do sistema. Estarão pré-cadastrados os perfis de administrador e membro do time. A figura 19 mostra a tela de cadastro de perfis.

Figura 19 – Tela de cadastro de perfis.



Fonte: Autoria própria.

### 8.2.2 Cadastro de usuários

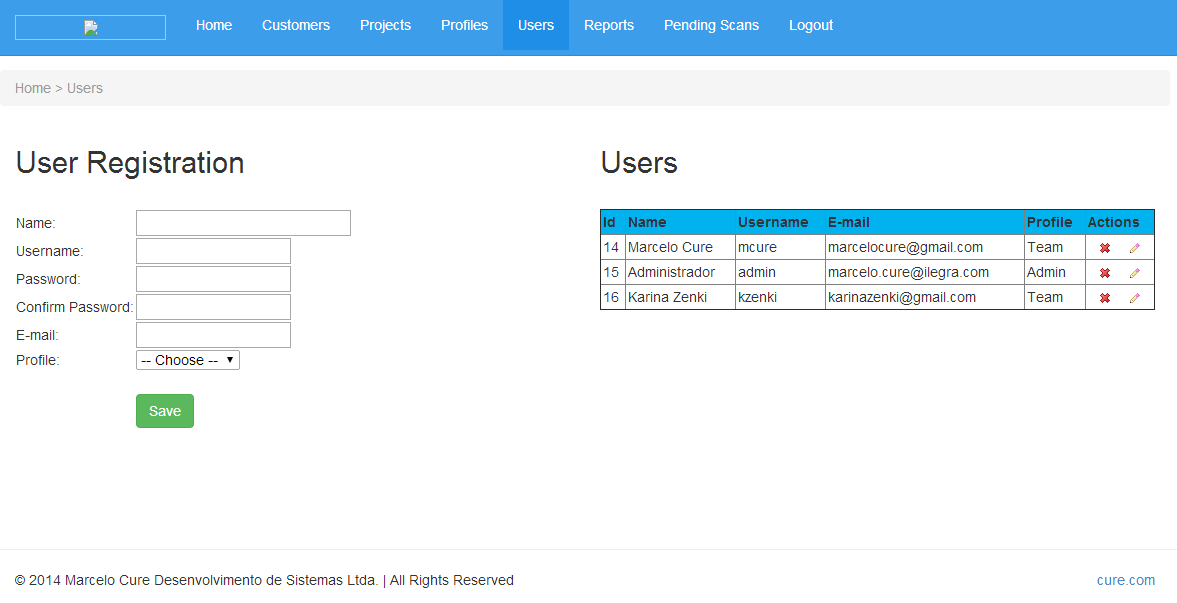
Nesta tela o administrador será capaz de cadastrar os usuários do sistema, onde serão cadastrados os dados básicos para um usuário do sistema, estes estão descritos abaixo:

1. Nome;
2. Username;
3. Password, onde incluirá a confirmação do mesmo;
4. E-mail;
5. Perfil de acesso.

As senhas serão criptografadas no banco de dados utilizando a biblioteca PyCrypto (pypi.python.org/pypi/pycrypto/2.6.1, 2014).

A figura 20 mostra a tela de cadastro de usuários.

Figura 20 – Tela de cadastro de usuários.



Fonte: Autoria própria.

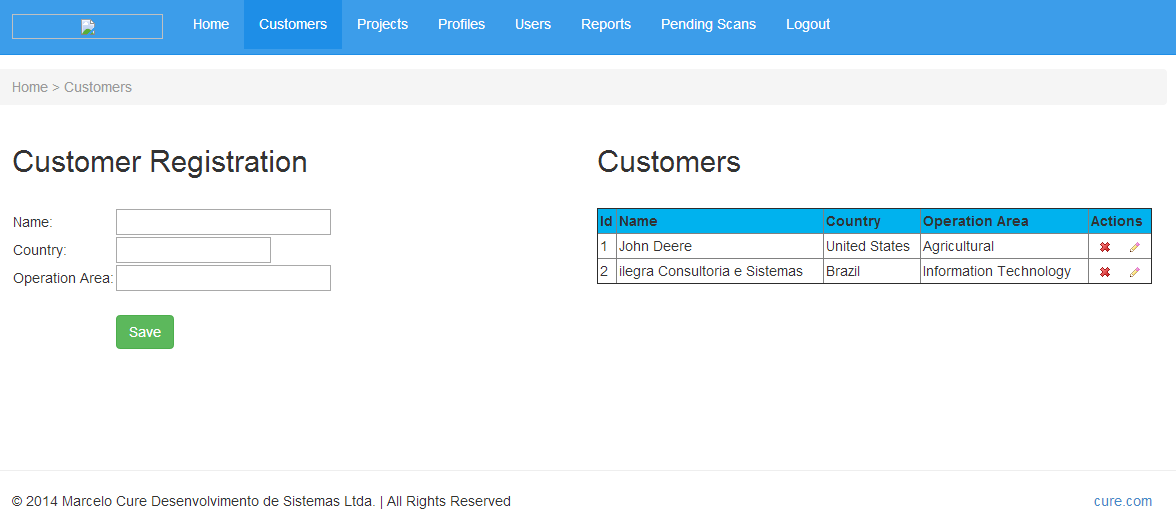
### 8.2.3 Cadastro de clientes

Nesta tela o administrador será capaz de cadastrar os clientes da organização. Os dados que serão cadastrados estão descritos abaixo:

1. Nome;
2. País;
3. Área de operação.

A figura 21 mostra a tela de cadastro de clientes.

Figura 21 – Tela de cadastro de clientes.



Fonte: Autoria própria.

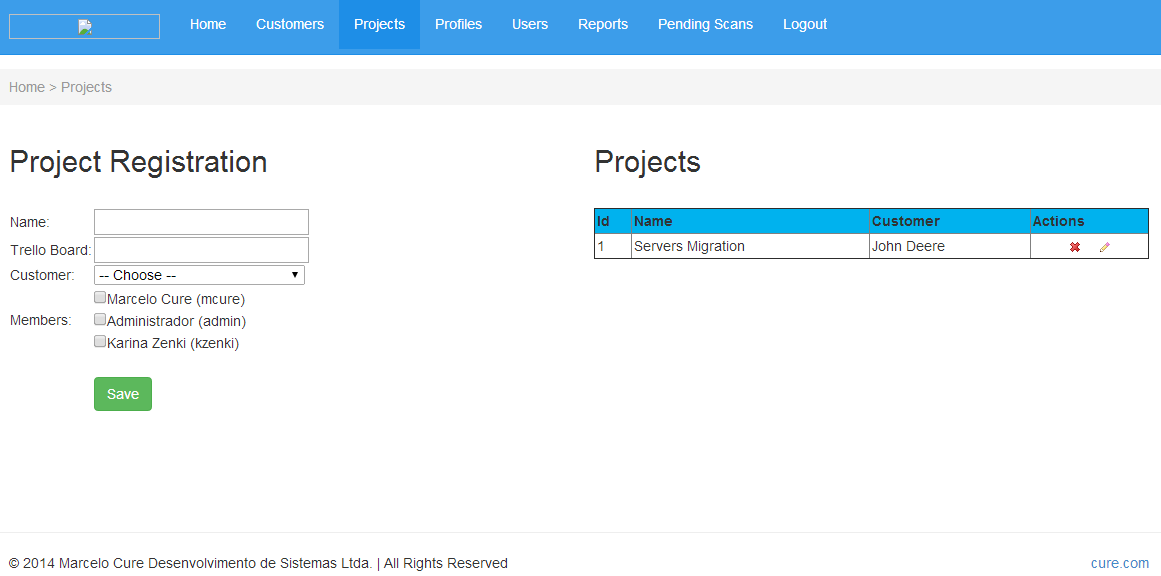
### 8.2.4 Cadastro de projetos

Nesta tela o administrador será capaz de cadastrar os projetos de um cliente. Os dados que serão cadastrados estão descritos abaixo:

1. Descrição;
2. Cliente;
3. Membros do time alocados no projeto.

A figura 22 mostra a tela de cadastro de projetos.

Figura 22 – Tela de cadastro de projetos.



Fonte: Autoria própria.

### 8.2.5 Consulta de scans pendentes

Nesta tela o administrador poderá consultar os projetos que estão finalizado a *sprint* no dia corrente e que o time ainda não executou o processo de scan.

### 8.2.6 Gráficos

Nesta tela o administrador poderá visualizar os gráficos referentes a um projeto específico ou a todos os projetos da organização.

1. Quantidade de pontos entregues por *sprint* X time;
2. Quantidade de testes unitários criados por *sprint* X time;
3. Quantidade de pontos estimados X quantidade de pontos entregues para cada *sprint*;
4. Quantidade de pontos entregues X testes unitários para cada *sprint*;
5. Quantidade de estórias entregues X *sprint* para cada projeto.

# 9 VALIDAÇÃO

A validação do projeto será dada através da implantação do sistema na empresa de desenvolvimento de software ilegra ([ilegra.com](http://www.ilegra.com/), 2014). Este poderá ser utilizada por um projeto específico pra mostrar se realmente foram registradas melhoras nas estimativas e planejamento das sprints.

Serão disponibilizados questionários aos membros do time, onde poderão ser preenchidos após 4 sprints, quando o sistema terá dados históricos suficientes para prover estimativas precisas. Os questionários disponibilizados estão descritos abaixo:

a) Avaliação das funcionalidade do sistema;

b) Avaliação da navegabilidade;

c) Avaliação da segurança do sistema.

**Iso9126 -> Iso25000**

# 10 CRONOGRAMA

Tabela 7 - Cronograma

|  |  |  |  |
| --- | --- | --- | --- |
| Descrição da Atividade | Produto | Data | Descrição |
| Entrega do formuário de aceite | Formulário de aceite | 17/03/2014 | Entrega do formuário de aceite |
| Entrega do plano de trabalho | Plano de trabalho | 31/03/2014 | Entrega do plano de trabalho |
| Produção do Product Backlog | Product Backlog | 10/04/2014 | Produção do Product Backlog |
| Modelagem ER | Diagrama ER | 15/04/2014 | Desenvolvimento do diagrama entidade relacionamento |
| Projeção das telas | Protótipo de telas | 20/04/2014 | Projeção das telas do sistema, será entregue uma idéia das telas. |
| Produção do diagrama de classes | Diagrama de Classes | 25/04/2014 | Produção do diagrama de classes |
| Produção do diagrama de casos de uso | Diagrama de Casos de Uso | 30/04/2014 | Produção do diagrama de casos de uso |
| Sprint 1 | Sistema | 14/05/2014 | Primeira iteração do projeto |
| Sprint 2 | Sistema | 01/06/2014 | Segunda iteração do projeto |
| Entrega do Relatório Parcial | Relatório Parcial de Projeto | 02/06/2014 | Entrega do Relatório Parcial |
| **Bancas de TCC I** | **Projeto Parcial** | **09 a 11 e 13/06/2014** | **Avaliação de TCC I** |
| Revisão da Implementação | Anotações | 04/08/2014 | Revisão do andamento do projeto |
| Sprint 3 | Sistema | 22/08/2014 | Terceira iteração do projeto |
| Sprint 4 | Sistema | 05/09/2014 | Quarta iteração do projeto |
| Ajustes no relatório do projeto | Relatório do projeto | 07/09/2014 | Ajustes no relatório do projeto |
| Sprint 5 | Sistema | 21/09/2014 | Quinta iteração do projeto |
| Entrega do relatório atualizado do projeto | Relatório atualizado do projeto | 22/09/2014 | Entrega do relatório atualizado do projeto |
| Seminário de andamento do projeto | Seminário | 29/09 a 03/10/2014 | Seminário de andamento do projeto |
| Sprint 6 | Sistema | 17/10/2014 | Sexta iteração do projeto |
| Validação | Questionários de validação | 03/11/2014 | Validação |
| Entrega do relatório final do projeto | Relatório Final do Projeto | 17/11/2014 | Entrega do relatório final do projeto |
| Banca Final | Projeto | 24 a 28/11/2014 | Apresentação do projeto |
| Entrega da versão final do Relatório do projeto | Projeto | 18/12/2014 | Entrega da versão final do Relatório do projeto com correções |

# 

# REFERÊNCIAS BIBLIOGRÁFICAS

METHODOLOGY, scrum: **Scrum Effort Estimation and Story Points**. 2009. Disponível em: <<http://scrummethodology.com/scrum-effort-estimation-and-story-points/>>. Acesso em: 18 mar. 2014. **(POSTAGEM RETIRADA DE SITE)**

PLANBOX (Montreal) (Ed.). 2013. **Agile by the numbers 2013: Planbox recently sponsored a global survey by Actuation Consulting that looked into why some teams excel while others struggle**. Disponível em: <https://www.planbox.com/agile-by-the-numbers-2013-performance-study/>. Acesso em: 26 mar. 2014. **(POSTAGEM RETIRADA DE SITE)**

KEN SCHWABER. **Guia do Scrum: Um guia definitivo para o Scrum: As regras do jogo.** 2013. Disponível em: <https://www.scrum.org/Portals/0/Documents/Scrum Guides/2013/Scrum-Guide-Portuguese-BR.pdf>. Acesso em: 27 mar. 2014. **(POSTAGEM RETIRADA DE SITE)**

WAGNER LINDBERG BACCARIN ARNAUT. **O Fim do Empirismo no Desenvolvimento de Software.** 2008. Disponível em: <https://www.ibm.com/developerworks/community/blogs/tlcbr/entry/fim\_do\_empirismo?lang=en>. Acesso em: 29 mar. 2014. **(POSTAGEM RETIRADA DE SITE)**

METHODOLOGY, scrum: **Scrum User Stories**. 2009. Disponível em: < [http://scrummethodology.com/scrum-user-stories/](%20http://scrummethodology.com/scrum-user-stories/)>. Acesso em: 19 mar. 2014. **(POSTAGEM RETIRADA DE SITE)**

SCOTT W. AMBLER, 2006. **Introduction to Test Driven Development (TDD)**. Disponível em: <http://www.agiledata.org/essays/tdd.html>. Acesso em: 26 mar. 2014.

jQuery. Disponível em: < http://jquery.com/>. Acesso em: 20 mar. 2014.

Eclipse Org. Disponível em: < http://epf.eclipse.org/wikis/scrumpt/Scrum/guidances/supportingmaterials/scrum\_overview\_610E45C2.html/>. Acesso em: 20 abr. 2014.

Planningpoker. Disponível em <[http://www.planningpoker.com/>/](http://www.planningpoker.com/%3e/). Acesso em: 17 mar. 2014.

Trello. Disponível em <[http://www.trello.com](http://www.trello.com/)/>. Acesso em: 17 mar. 2014.

Trello API. Disponível em <[http://www.trello.com](http://www.trello.com/)/docs>. Acesso em: 31 mar. 2014.

Python. Disponível em <<http://www.python.org.br/>>. Acesso em: 19 mar. 2014.

Django. Disponível em <<https://www.djangoproject.com/>>. Acesso em: 26 mar. 2014.

SublimeText. Disponível em <[http://www.sublimetext.com](http://www.sublimetext.com/)/>. Acesso em: 24 mar. 2014.

Postgresql. Disponível em <<http://www.postgresql.org/>>. Acesso em: 27 mar. 2014.

Github. Disponível em <[http://www.github.com](http://www.github.com/)/>. Acesso em: 21 mar. 2014.

NinjaMock. Disponível em <<http://ninjamock.com/>>. Acesso em: 28 mar. 2014.

Cacoo. Disponível em <<https://cacoo.com/>>. Acesso em: 25 mar. 2014.

ilegra. Disponível em <[http://www.ilegra.com](http://www.ilegra.com/)/>. Acesso em: 26 mar. 2014.

PyCrypto. Disponível em <https://pypi.python.org/pypi/pycrypto/2.6.1>. Acesso em: 25 mai. 2014.

# COMPONENTES REUTILIZADOS

***DJANGO***

Será utilizado o *framework* *Django* para auxiliar no desenvolvimento das páginas e integração com banco de dados, onde o desenvolvedor pode definir o modelo de dados utilizando a linguagem *Python*. Este *framework* também provém uma *API* poderosa de integração com o banco de dados. *Django* é um framework para desenvolvimento web com *Python* e serve, principalmente, para agilizar o desenvolvimento. (djangoproject.com, 2014)

***API do Trello***

O software *Trello* oferece uma *API* para acesso dos *boards*, onde possibilita a coleta de informações onde o usuário possui acesso. Através de uma chave de segurança gerado pelo site, o desenvolvedor utiliza esta chave de segurança para autenticação e coleta de informações. (trello.com/docs, 2014)

***PyCrypto***

PyCrypto é uma biblioteca escrita em Python utilizada para criptografia de dados. Esta biblioteca será utilizada para criptografar as senhas dos usuários do sistema. (pypi.python.org/pypi/pycrypto/2.6.1, 2014).