**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 ATUALIZADO**

**SISTEMA DE COLETA DE MÉTRICAS DE PROJETOS ÁGEIS PARA APOIAR ESTIMATIVAS**

Porto Alegre

2014

**MARCELO BALDISSERA CURE**

**RELATÓRIO DE PROJETO ATUALIZADO**

**SISTEMA DE COLETA DE MÉTRICAS DE PROJETOS ÁGEIS PARA APOIAR ESTIMATIVAS**

Relatório de Projeto Atualizado, apresentado como requisito parcial à obtenção da aprovação do projeto de TCC2 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](#_Toc398674861)

[Figura 2 - Estrutura básica do *TDD.* 13](#_Toc398674862)

[Figura 3 - Camadas e componentes do sistema. 15](#_Toc398674863)

[Figura 4 - Processo de login. 16](#_Toc398674864)

[Figura 5 – Diagrama de casos de uso. 23](#_Toc398674865)

[Figura 6 – Diagrama de classes. 24](#_Toc398674866)

[Figura 7 – Diagrama Entidade Relacionamento. 25](#_Toc398674867)

[Figura 8 – Protótipo da tela de cadastro de *sprints*. 25](#_Toc398674868)

[Figura 9 – Protótipo da tela de Scan do projeto. 26](#_Toc398674869)

[Figura 10 – Protótipo da tela de gráficos. 26](#_Toc398674870)

[Figura 11 – Protótipo da tela de projetos do usuário. 27](#_Toc398674871)

[Figura 12 – Protótipo da tela de *sprints* dos projetos do usuário. 28](#_Toc398674872)

[Figura 13 – Protótipo da tela de *cards* de *sprints* dos projetos do usuário. 28](#_Toc398674873)

[Figura 14 – Protótipo da tela de cadastro de perfis. 29](#_Toc398674874)

[Figura 15 – Protótipo da tela de cadastro de usuários. 29](#_Toc398674875)

[Figura 16 – Protótipo da tela de cadastro de clientes. 30](#_Toc398674876)

[Figura 17 – Protótipo da tela de cadastro de projetos. 30](#_Toc398674877)

[Figura 18 – Protótipo da tela de scans pendentes. 31](#_Toc398674878)

[Figura 19 – Protótipo da tela de gráficos. 31](#_Toc398674879)

[Figura 20 – Processo de negócio do sistema. 32](#_Toc398674880)

[Figura 21 – Tela de cadastro de *sprints*. 33](#_Toc398674881)

[Figura 22 – Tela de Scan do projeto. 34](#_Toc398674882)

[Figura 23 – Tela de cadastro de perfis. 35](#_Toc398674883)

[Figura 24 – Tela de cadastro de usuários. 35](#_Toc398674884)

[Figura 25 – Tela de cadastro de clientes. 36](#_Toc398674885)

[Figura 26 – Tela de cadastro de projetos. 37](#_Toc398674886)

**LISTA DE TABELAS**

[Tabela 1 – Sprint 1. 20](#_Toc398674887)

[Tabela 2 – Sprint 2. 21](#_Toc398674888)

[Tabela 3 – Sprint 3. 21](#_Toc398674889)

[Tabela 4 – Sprint 4. 21](#_Toc398674890)

[Tabela 5 – Sprint 5. 23](#_Toc398674891)

[Tabela 6 – Sprint 6. 23](#_Toc398674892)

[Tabela 7 - Cronograma 39](#_Toc398674893)

**LISTA DE SIGLAS**

UML Unified Modeling Language

TDD Test Driven Development

ER Entity Relationship

API Application Programming Interface

JSON JavaScript Object Notation

**RESUMO**

Este trabalho visa facilitar as estimativas de esforço de *sprints* para projetos que utilizam metodologias ágeis. O sistema inclui cadastros de clientes, usuários, projetos, perfis e sprints, provendo uma visão de todos os projetos que estão acontecendo na organização. O sistema coleta 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 servem de apoio a tomada de decisão no momento de estimar uma *sprint*. Isto acarreta em segurança para o time de desenvolvimento e para o cliente que saberá o custo e tempo das *sprints* evitando imprevistos de tempo e financeiros.

SUMÁRIO

[1 APRESENTAÇÃO GERAL DO PROJETO 8](#_Toc398674902)

[2 DEFINIÇÃO DO PROBLEMA 9](#_Toc398674903)

[3 OBJETIVOS 11](#_Toc398674904)

[3.1 Objetivo Geral 11](#_Toc398674905)

[3.2 Objetivos Específicos 11](#_Toc398674906)

[4 ANÁLISE DE TECNOLOGIAS/FERRAMENTAS 12](#_Toc398674907)

[4.1 Linguagens de programação 12](#_Toc398674908)

[4.2 Ferramenta de Desenvolvimento 12](#_Toc398674909)

[4.3 Banco de Dados 12](#_Toc398674910)

[4.4 Controle de Atividades 12](#_Toc398674911)

[4.5 Versionamento do Código 13](#_Toc398674912)

[4.6 Qualidade do Código 13](#_Toc398674913)

[4.7 Ferramentas de Modelagem 14](#_Toc398674914)

[4.8 Protótipos de tela 14](#_Toc398674915)

[5 DESCRIÇÃO DA SOLUÇÃO 15](#_Toc398674916)

[5.1 Gerenciador de Sessões 16](#_Toc398674917)

[5.2 Gerador de Gráficos 17](#_Toc398674918)

[5.3 Scan 17](#_Toc398674919)

[6 ABORDAGEM DE DESENVOLVIMENTO, 18](#_Toc398674920)

[6.1 Artefatos 18](#_Toc398674921)

[6.1.1 Product Backlog 18](#_Toc398674922)

[6.1.2 Sprint Backlog 19](#_Toc398674923)

[6.1.3 User Stories (estórias) 19](#_Toc398674924)

[6.1.4 Diagrama de Caso de Uso 19](#_Toc398674925)

[6.1.5 Diagrama de Classes 19](#_Toc398674926)

[6.1.6 Gráfico de cobertura de tests 19](#_Toc398674927)

[7 ARQUITETURA DO SISTEMA 20](#_Toc398674928)

[7.1 Modelagem Funcional 20](#_Toc398674929)

[7.1.1 Sprint 1 20](#_Toc398674930)

[7.1.2 Sprint 2 21](#_Toc398674931)

[7.1.3 Sprint 3 21](#_Toc398674932)

[7.1.4 Sprint 4 21](#_Toc398674933)

[7.1.5 Sprint 5 23](#_Toc398674934)

[7.1.6 Sprint 6 23](#_Toc398674935)

[7.2 Modelos UML 23](#_Toc398674936)

[7.3 Modelagem de Dados 25](#_Toc398674937)

[7.4 Interface Gráfica 25](#_Toc398674938)

[7.4.1 Cadastro de Sprints 25](#_Toc398674939)

[7.4.2 Scan de projeto 26](#_Toc398674940)

[7.4.3 Gráficos 26](#_Toc398674941)

[7.4.4 Projetos do usuário 27](#_Toc398674942)

[7.4.5 Projetos do usuário - *Sprints* 27](#_Toc398674943)

[7.4.6 Projetos do usuário - *Sprint cards* 28](#_Toc398674944)

[7.4.7 Cadastro de perfis 28](#_Toc398674945)

[7.4.8 Cadastro de usuários 29](#_Toc398674946)

[7.4.9 Cadastro de clientes 30](#_Toc398674947)

[7.4.10 Cadastro de projetos 30](#_Toc398674948)

[7.4.11 Consulta de scans pendentes 31](#_Toc398674949)

[7.4.12 Gráficos 31](#_Toc398674950)

[7.5 Processo de Negócio 32](#_Toc398674951)

[8 FUNCIONAMENTO DO SISTEMA 33](#_Toc398674952)

[8.1 Membro do time 33](#_Toc398674953)

[8.1.1 Cadastro de Sprints 33](#_Toc398674954)

[8.1.2 Scan de projeto 33](#_Toc398674955)

[8.1.3 Gráficos 34](#_Toc398674956)

[8.2 Administrador 34](#_Toc398674957)

[8.2.1 Cadastro de perfis 34](#_Toc398674958)

[8.2.2 Cadastro de usuários 35](#_Toc398674959)

[8.2.3 Cadastro de clientes 36](#_Toc398674960)

[8.2.4 Cadastro de projetos 36](#_Toc398674961)

[8.2.5 Consulta de scans pendentes 37](#_Toc398674962)

[8.2.6 Gráficos 37](#_Toc398674963)

[9 VALIDAÇÃO 38](#_Toc398674964)

[9.1 ESTRATÉGIA 38](#_Toc398674965)

[10 CRONOGRAMA 39](#_Toc398674966)

[REFERÊNCIAS BIBLIOGRÁFICAS 40](#_Toc398674967)

[COMPONENTES REUTILIZADOS 42](#_Toc398674968)

# 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 histórias. Cada história é estimada, normalmente, utilizando uma método chamado *Planning Poker* (Molokken-Ostvold, K., Haugen, N.C. 2007), onde cada membro do time estima cada história em pontos*.*

Mesmo utilizando metodologias ágeis, as estimativas dificilmente são precisas, então os projetos carecem de métodos para prover qualidade ao estimar histórias 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. As tarefas que serão executadas necessitam de uma estimativa precisa. Com isto, estimativa é um fator que delimita o prazo para a finalização do projeto e o valor que o cliente deverá desembolsar.

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 histórias que deverão ser entregues no próximo ciclo.

Figura 1 – Processo básico do *Scrum*.

Fonte: (Eclipse Org, 2009).

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

As histórias 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 história é estimada em pontos, que variam de 0 até 100 pontos (Molokken-Ostvold, K., Haugen, N.C., 2007).

O *Planning Poker* é uma metodologia utilizada para auxiliar nas estimativas de esforço. Depois que a história é apresentada para os membros do time, estes discutem a história para ter certeza de que todos entenderam. Cada membro do time escolhe uma pontuação representando o esforço que será necessário para implementar a história, 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 história específica.

É comum que a pontuação estimada no *Planning Poker* para a história não seja precisa. Por exemplo, o time estimou a estória em 5 pontos, mas esta poderia ser estimada em 3 pontos já que 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 a quantidade de pontos que 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 poderá extrair um aprendizado destas estatísticas que poderão ser utilizadas 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 histórias para uma *sprint*, e para o cliente que não tem segurança para prever a data de um *release* e o 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 histórias 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*, baseando-se em um histórico de estimativas e resultados obtidos;
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 estão sendo utilizadas para o desenvolvimento do projeto, incluindo tecnologias de desenvolvimento, modelagem e técnicas.

## 4.1 Linguagens de programação

Este projeto está sendo desenvolvido utilizando a linguagem de programação *Python*, pois ela provém facilidade no desenvolvimento, possui uma variedade de bibliotecas e as APIs são simples e bem documentadas, assim como a própria linguagem. A facilidade na instalação de bibliotecas e na configuração do ambiente de desenvolvimento são qualidades essenciais. (python.org.br, 2014).

Está sendo utilizado o framework *Django* para auxiliar no desenvolvimento das páginas, onde o desenvolvedor pode definir o modelo de dados utilizando a linguagem *Python*. Django é um *framework* que serve para o desenvolver aplicações web com *Python.* Este *framework* também provém uma API para integração com o banco de dados que abstrai as tabelas e através dos modelos podem ser executadas todas as operações necessárias. *Django*. (djangoproject.com, 2014).

## 4.2 Ferramenta de Desenvolvimento

O desenvolvimento está sendo 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. Existe uma diversidade de *plug ins* que auxiliam no desenvolvimento. O *Sublime* é uma ferramenta extremamente leve. ([sublimetext.com](http://www.sublimetext.com/), 2014).

## 4.3 Banco de Dados

Os dados estão sendo armazenados no banco de dados relacional PostgreSQL. Este banco de dados provê a capacidade de escrever consultas complexas e retornar os dados em um formato específico para que o sistema possa utilizá-los de maneira a suprir as necessidades das funcionalidades (postgresql.org, 2014).

## 4.4 Controle de Atividades

O controle de atividades e *sprints* está sendo controlado utilizando o *software* *Trello* ([trello.com](http://www.trello.com/), 2014), o qual também é utilizado 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.

## 4.5 Versionamento do Código

O controle de versão de código e da própria documentação do projeto está sendo feito utilizando a ferramenta *GitHub* ([github.com](http://www.github.com/), 2014). Registrando um usuário, este software possibilita o cadastramento de projetos, criação de *bugs*, melhorias, etc.. O *GitHub* será importante para o projeto na parte de versionamento de código, onde tudo será salvo na nuvem e o código pode ser recuperado em qualquer máquina na internet.

## 4.6 Qualidade do Código

O sistema está sendo 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 está sendo 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.com, 2014).

Estão sendo gerados os diagramas de casos de uso, diagrama de classes e diagrama entidade relacionamento utilizando o Cacoo.

## 4.8 Protótipos de tela

Os protótipos de tela estão sendo criados utilizado o *NinjaMock*. Esta ferramenta possibilita criar protótipos de telas de várias plataformas, como *Web*, *Android*, *iOS*, *WindowsPhone*, etc. O *NinjaMock* possibilita que os protótipos das telas sejam exportados nos formatos PDF ou PNG. (ninjamock.com, 2014).

# 5 DESCRIÇÃO DA SOLUÇÃO

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

O sistema é restringido somente para usuários que possuem acesso. Os membros do time possuem acesso somente aos projetos que participam, onde tem as opções de cadastrar *sprints*, fazer a coleta de dados, visualizar dados referentes a *sprints* que passaram e visualizar gráficos. O sistema inclui a área do administrador, onde este pode cadastrar perfis, usuários, clientes, projetos e alocar recursos a estes projetos. O usuário com perfil administrador pode consultar se existe algum projeto com o processo de *scan* pendente. Esta consulta verifica se algum dos projetos tem uma *sprint* sendo finalizada naquele dia e o time ainda não executou o processo de *scan* para coletar as métricas. Assim o administrador pode ter um controle melhor sobre o sistema e os projetos que estão em progresso.

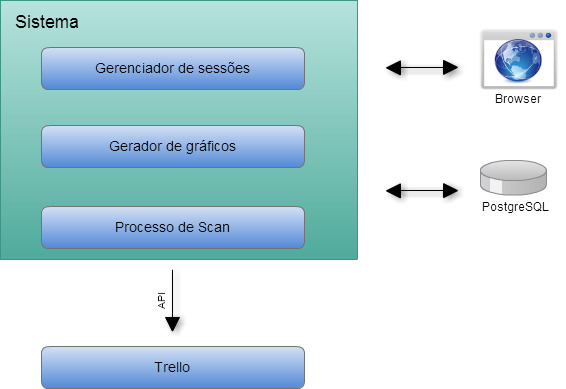
Considera-se que os projetos tenham acompanhamento na ferramenta online *Trello*. Esta ferramenta é utilizada para montar quadros *Kanban*, que possibilita a criação de *boards* onde o usuário pode criar *lists*, *cards*, *labels*, adicionar informações, anexar arquivos, etc. (trello.com, 2014).

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

O sistema coletará os dados de cada *card*, como a sua identificação, tempo de duração da tarefa, *Story Points*, 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* e finalizar a *sprint* corrente.

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.

Figura 3 - Camadas e componentes do sistema.



Fonte: Autoria própria.

A Figura 3 mostra as camadas e componentes principais da aplicação, incluindo a comunicação com o *browser* (navegador web), banco de dados e com o *software* Trello. Foram criados módulos para controlar funcionalidades específicas do sistema, que são descritos nas seções a seguir.

## 5.1 Gerenciador de Sessões

O módulo Gerenciador de Sessões é responsável por controlar os acessos dos usuários. Quando o usuário faz a autenticação no sistema, os dados da sessão são armazenados no banco de dados e em *cookies* do *browser*.

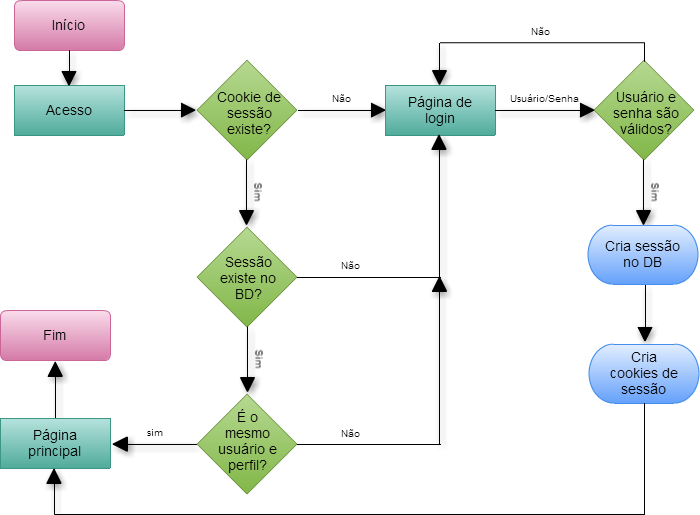
O *cookie* é utilizado para guardar informações referentes a sessão caso o usuário feche o *browser* sem clicar em Logout, no próximo acesso ao sistema, este identificará que já existe uma sessão para este usuário, não sendo necessário passar pelo processo de autenticação novamente.

As informações da sessão também são guardadas no banco de dados para evitar que usuários mal intencionados criem o *cookie* de sessão manualmente para ter acesso a informações referentes a outros perfis de usuário.

Este módulo também encaminha o usuário para a página adequada ao seu perfil.

A figura 4 descreve com detalhes o processo de login de usuário.

Figura 4 - Processo de login.



Fonte: Autoria própria.

## 5.2 Gerador de Gráficos

O módulo Gerador de Gráficos auxilia o sistema na criação dos gráficos baseados nos dados coletados. Os gráficos são gerados utilizando o ChartJS (chartjs.org, 2014), ChartJS é uma biblioteca escrita em javascript que gera gráficos baseando-se em um documento JSON. Este módulo do sistema busca busca as informações do banco de dados e retorna um documento JSON com estes dados incluindo detalhes sobre a formatação do gráfico.

## 5.3 Scan

O módulo de *Scan* é responsável por fazer a leitura do Trello. Este processo acessa o Trello através da API Trollop (pypi.python.org/pypi/trollop, 2014) fazendo a busca e sumarização dos dados referentes a *sprint* corrente do projeto. Ao final do processo de *scan* estes dados são salvos no banco de dados PostgreSQL e finaliza a *sprint*.

O acesso ao Trello se dá através de um *token* de autenticação. Este *token* tem como responsabilidade controlar o acesso ao Trello para um usuário específico que tenha acesso aos *boards* da organização. O *token* de autenticação é gerado através do próprio Trello, que identifica o usuário autenticado no momento.

# 6 ABORDAGEM DE DESENVOLVIMENTO,

FALAR QUE USA TDD E QUAIS ARTEFATOS SERÃO GERADOS.

O projeto está sendo desenvolvido utilizando alguma práticas 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 que tem uma equipe de um único membro.

As tarefas são consideradas *user stories,*e sua descrição apresenta uma ação a ser feita pelo usuário e o que o usuário espera que aconteça. Estão sendo utilizadas as *sprints* para representar cada iteração do projeto, as *sprints* tem a duração de três semanas. Ao final de cada *sprint* é feita uma retrospectiva trazendo coisas boas e dificuldades enfrentadas ao longo do desenvolvimento das tarefas na *sprint*.

Utilizando o *Kanban* é 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* estão sendo utilizados, priorizando itens do *backlog* para entregas periódicas. Estão sendo executadas *sprints* de duas semanas, onde ao final de cada ciclo as funcionalidades priorizadas serão entregues.

O projeto está sendo 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.

### 6.1.6 Gráfico de cobertura de tests

O sistema está sendo construído utilizando o TDD. Esta técnica visa desenvolver as unidades de programa baseando-se em testes unitários. Cria-se um teste unitário simples, inicalmente ele irá falhar, pois a funcionalidade ainda não está implementada, o próximo passo é fazer com que este teste passe, implementando o pedaço da funcionalidade esperada naquele teste. Após os teste estar passando, o desenvolvedor poderá refatorar aquele trecho de código, o teste unitário recém criado irá validar se qualquer comportamento foi modificado após as mudanças. E assim este processo se repetirá até a funcinalidade estar completa.

Os testes unitários são uma boa métrica de qualidade do código, porém, uma grande quantidade de testes unitários não garante a qualidade do sistema. Por isso, será gerado um gráfico com a cobertura de testes no sistema. A cobertura de testes serve para mostrar qual a porcentagem de linhas de código que está sendo testada.

# 7 ARQUITETURA DO SISTEMA

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

## 7.1 Modelagem Funcional

Na modelagem funcional estão sendo 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 | Eu como administrador devo ser capaz de cadastrar os  usuários com os seguintes 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 | Eu como administrador devo ser capaz de cadastrar os perfis  com os seguintes campos: descrição. Este formulário ficará na  parte esquerda da tela. No outro lado aparecerá a lista de  perfis já cadastrados. |
| Tela de login | Eu como administrador devo ser capaz de fazer login no  sistema e acessar a área do 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 | Eu como administrador devo ser capaz de cadastrar os clientes  com os seguintes 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 | Eu como administrador devo ser capaz de cadastrar os projetos  com os seguintes 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 | Eu como membro de um time devo ser capaz de cadastrar as  *Sprints* com os seguintes campos: Project, description, start  date, end date, estimated points. Este formulário ficará na  parte esquerda da tela. |
| Scan parte 1 | Eu como membro de um time devo ser capaz de executar o processo de scan de um projeto específico, 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 | Eu como membro de um time devo ser capaz de executar o  processo de *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* | Eu como administrador devo ser capaz de gerar um gráfico de  linhas para cada time mostrando a quantidade de pontos  entregues por *sprint*. |
| Quantidade de testes  unitários criados X *sprint* | Eu como administrador devo ser capaz de 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 | Eu como administrador devo ser capaz de 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* | Eu como administrador devo ser capaz de 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 | Eu como administrador devo ser capaz de 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* | Eu como membro de um time devo ser capaz de gerar um  gráfico de linhas mostrando a quantidade de pontos  entregues por *sprint*. |
| Quantidade de testes  unitários criados X *sprint* | Eu como membro de um time devo ser capaz de gerar um  gráfico de de linhas mostrando a quantidade de testes  unitários criados por *sprint*. |
| Quantidade de pontos  estimados X pontos  entregues | Eu como membro de um time devo ser capaz de gerar um  gráfico de de linhas mostrando a quantidade de pontos estimados  por quantidade de pontos entregues. |
| Quantidade de pontos  entregues X testes unitários  para cada *sprint* | Eu como membro de um time devo ser capaz de gerar um  gráfico de de linhas mostrando a quantidade de testes  unitários criados por pontos entregues por *sprint*. |

### 7.1.6 Sprint 6

Tabela 6 – Sprint 6.

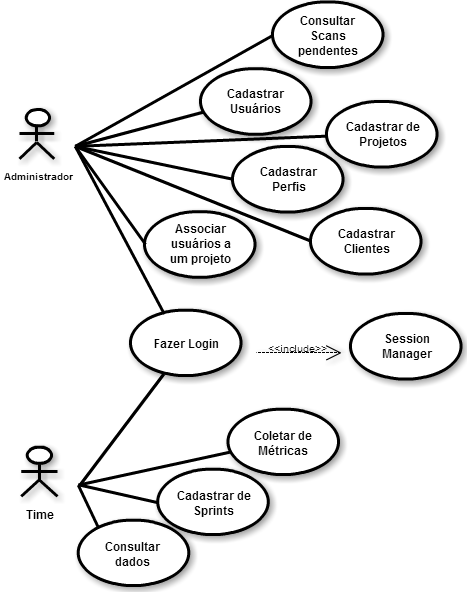
|  |  |
| --- | --- |
| Gráficos time final e consulta scans pendentes | |
| Quantidade de pontos  entregues X testes unitários  para cada *sprint* | Eu como membro de um time devo ser capaz de gerar um  gráfico de de linhas mostrando a quantidade de testes  unitários criados por pontos entregues por *sprint*. |
| Quantidade de  estórias entregues por  *sprint* | Eu como membro de um time devo ser capaz de gerar um  gráfico de de linhas mostrando a quantidade de estórias  criadas por *sprint*. |
| Consulta times que  ainda não rodaram o  scan no final da *sprint* | Eu como administrador devo ser capaz de ver 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, estão sendo gerados diagrama de classe e diagrama de casos de uso. Inicialmente foi criada uma versão simples, estes modelos evoluirão conforme as *sprints* são finalizadas.

O diagrama de casos de uso está descrito na figura 5.

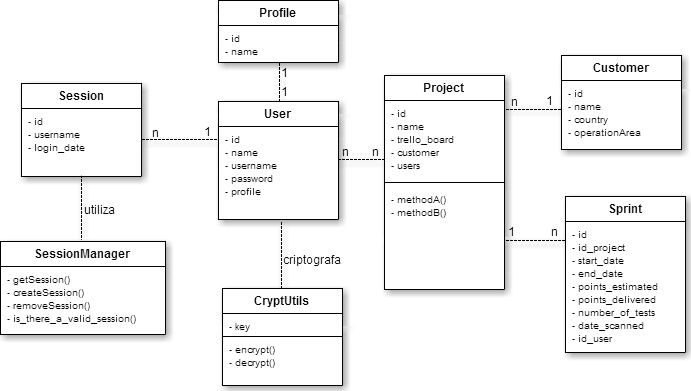
Figura 5 – Diagrama de casos de uso.



Fonte: Autoria própria.

O diagrama de classes está descrito na figura 6.

Figura 6 – Diagrama de classes.

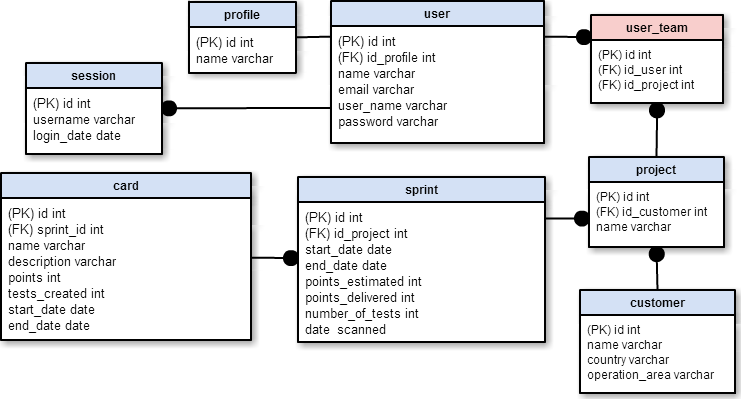


Fonte: Autoria própria.

## 7.3 Modelagem de Dados

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

Figura 7 – Diagrama Entidade Relacionamento.



Fonte: Autoria própria.

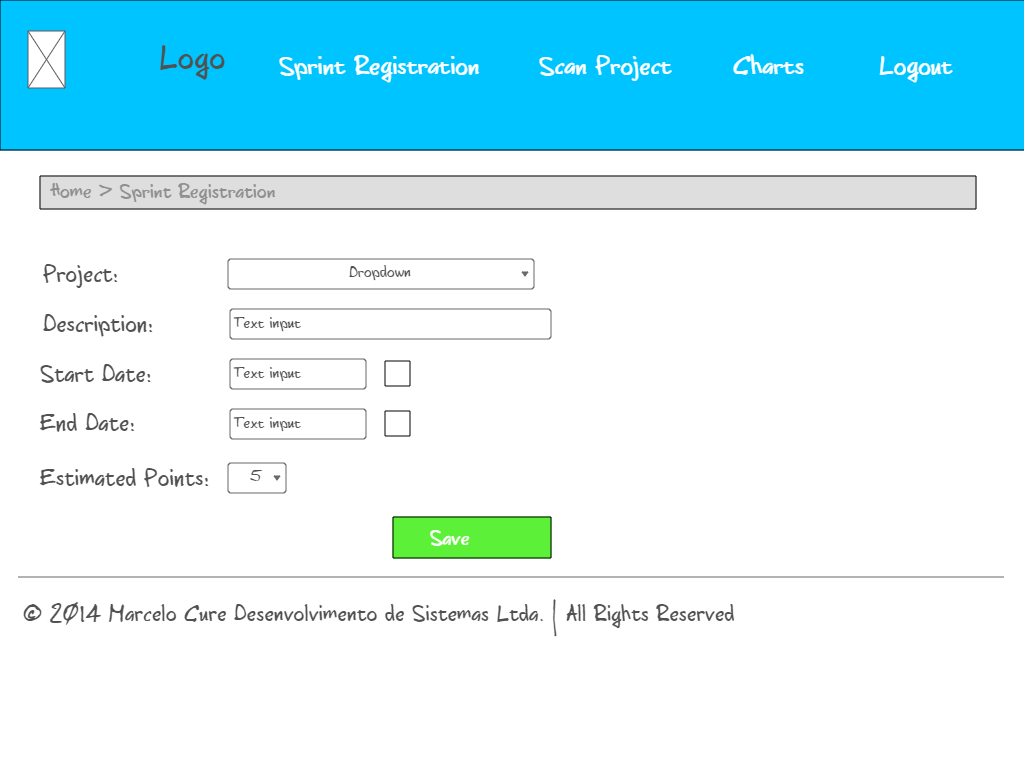
## 7.4 Interface Gráfica

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

### 7.4.1 Cadastro de Sprints

A figura 8 mostra o protótipo da tela de cadastro de *sprints*.

Figura 8 – Protótipo da tela de cadastro de *sprints*.

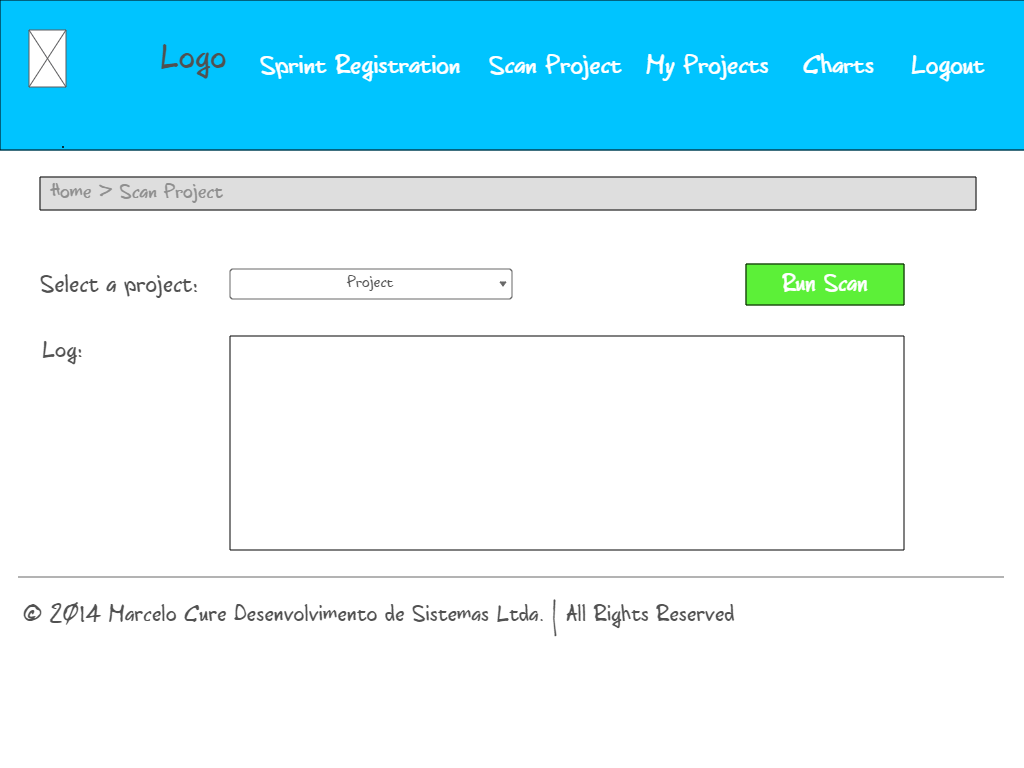


Fonte: Autoria própria.

### 7.4.2 Scan de projeto

A figura 9 mostra o protótipo da tela de scan do projeto.

Figura 9 – Protótipo da tela de Scan do projeto.

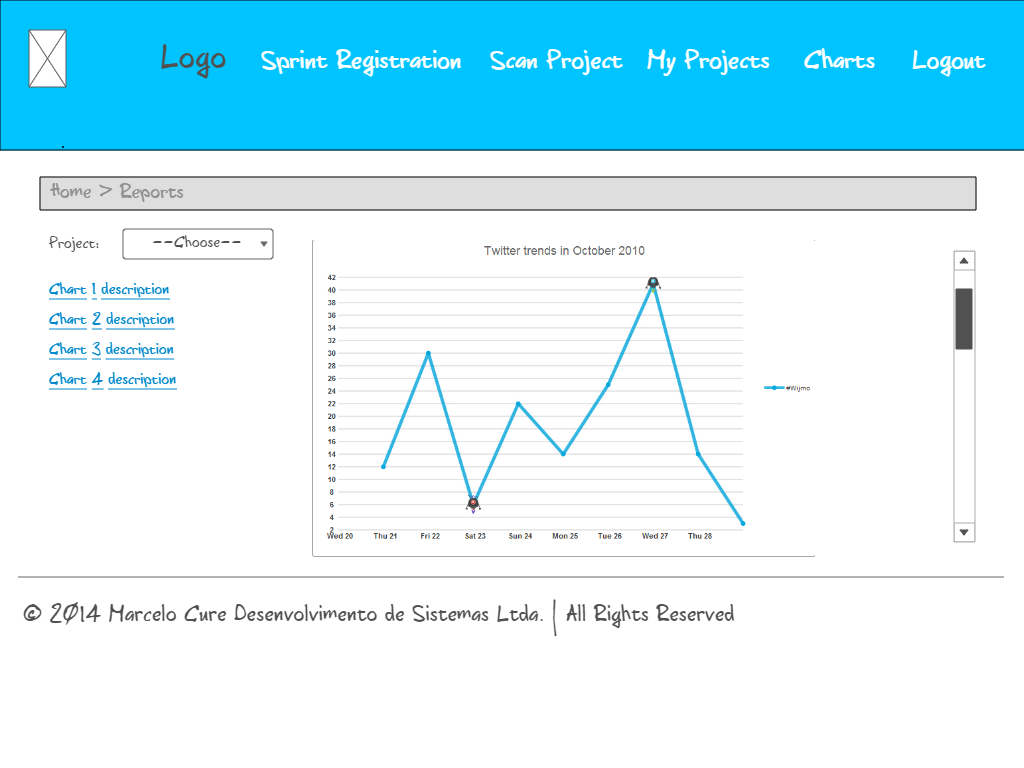


Fonte: Autoria própria.

### 7.4.3 Gráficos

A figura 10 mostra o protótipo da tela de gráficos.

Figura 10 – Protótipo da tela de gráficos.

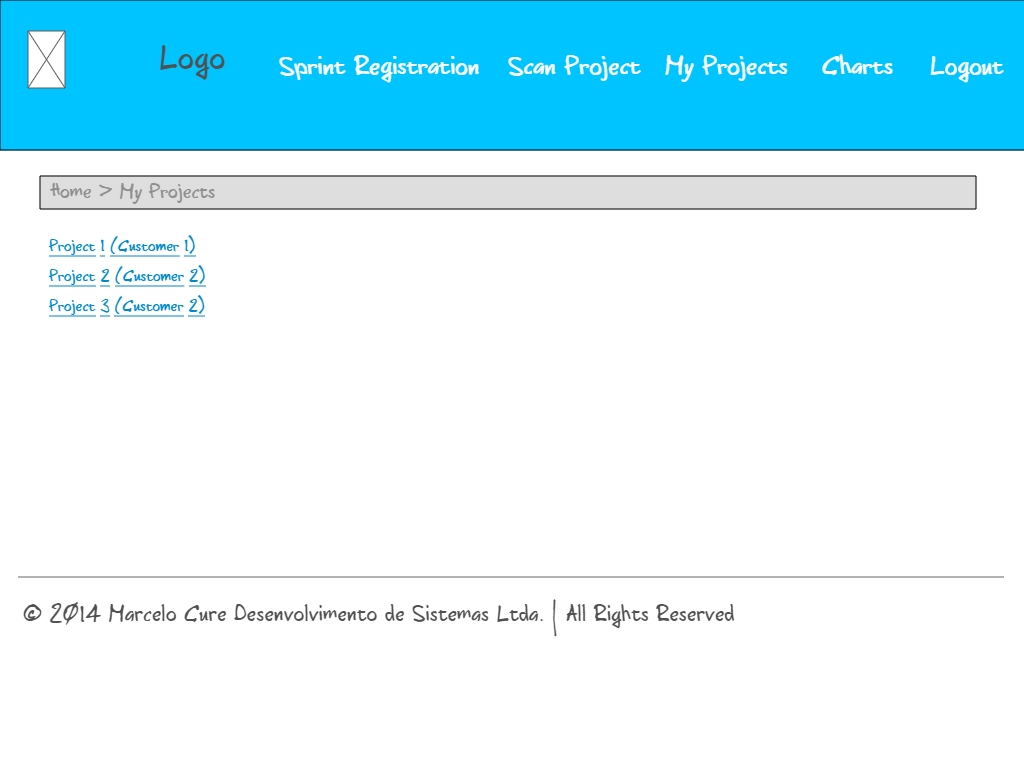


Fonte: Autoria própria.

### 7.4.4 Projetos do usuário

A figura 11 mostra o protótipo da tela de projetos do usuário.

Figura 11 – Protótipo da tela de projetos do usuário.

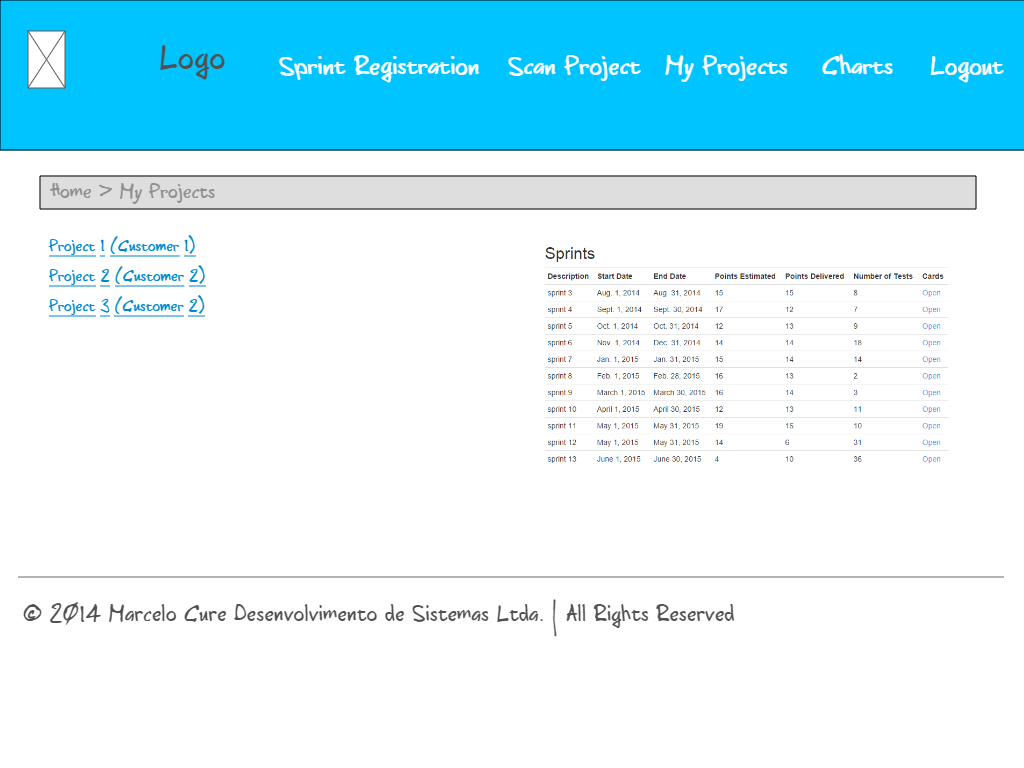


Fonte: Autoria própria.

### 7.4.5 Projetos do usuário - *Sprints*

A figura 12 mostra o protótipo da tela de *sprints* dos projetos do usuário.

Figura 12 – Protótipo da tela de *sprints* dos projetos do usuário.

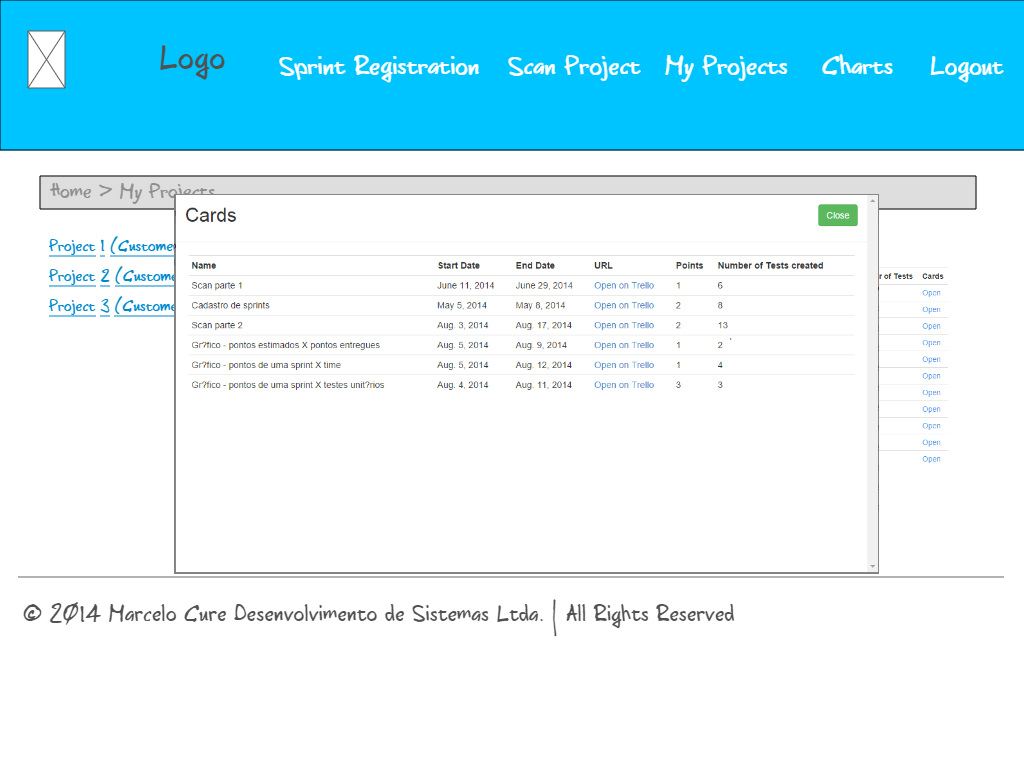


Fonte: Autoria própria.

### 7.4.6 Projetos do usuário - *Sprint cards*

A figura 13 mostra o protótipo da tela de *cards* de *sprints* dos projetos do usuário.

Figura 13 – Protótipo da tela de *cards* de *sprints* dos projetos do usuário.

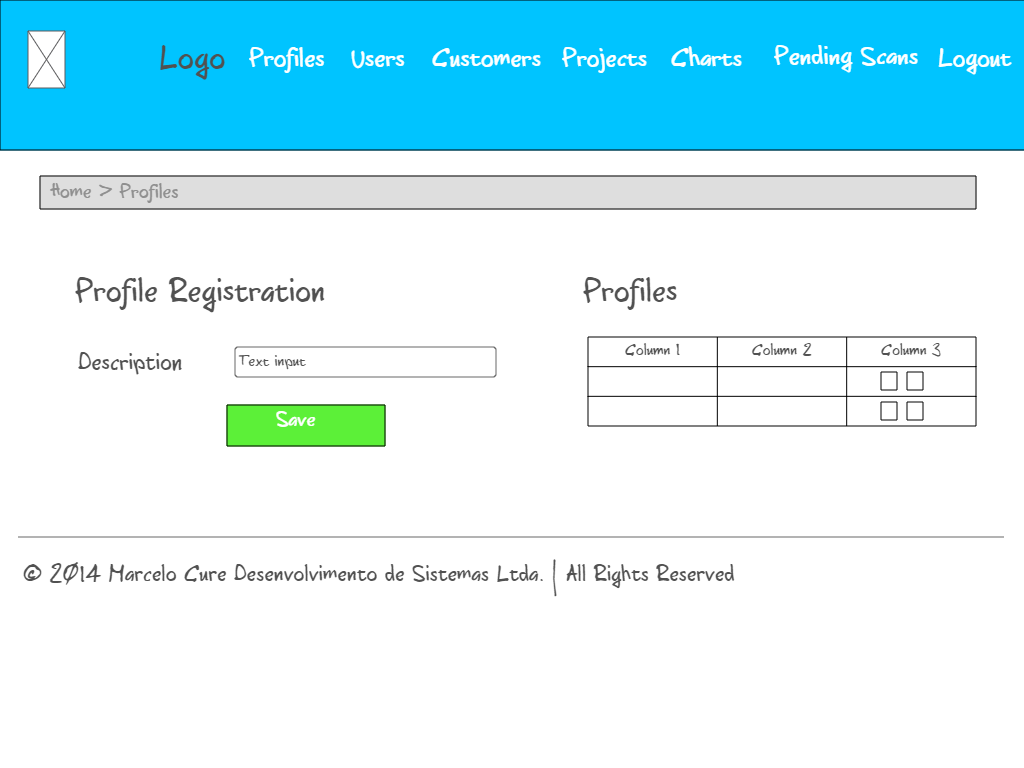


Fonte: Autoria própria.

### 7.4.7 Cadastro de perfis

A figura 14 mostra o protótipo da tela de cadastro de perfis.

Figura 14 – Protótipo da tela de cadastro de perfis.

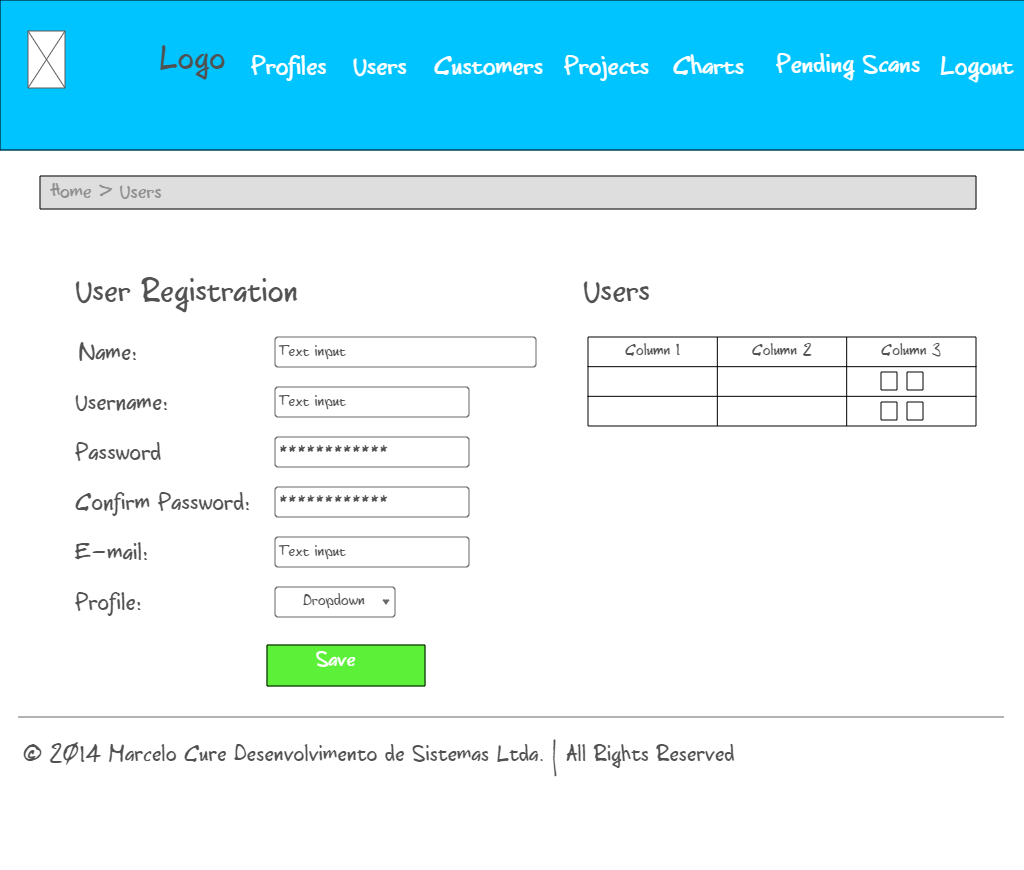


Fonte: Autoria própria.

### 7.4.8 Cadastro de usuários

A figura 15 mostra o protótipo da tela de cadastro de usuários.

Figura 15 – Protótipo da tela de cadastro de usuários.

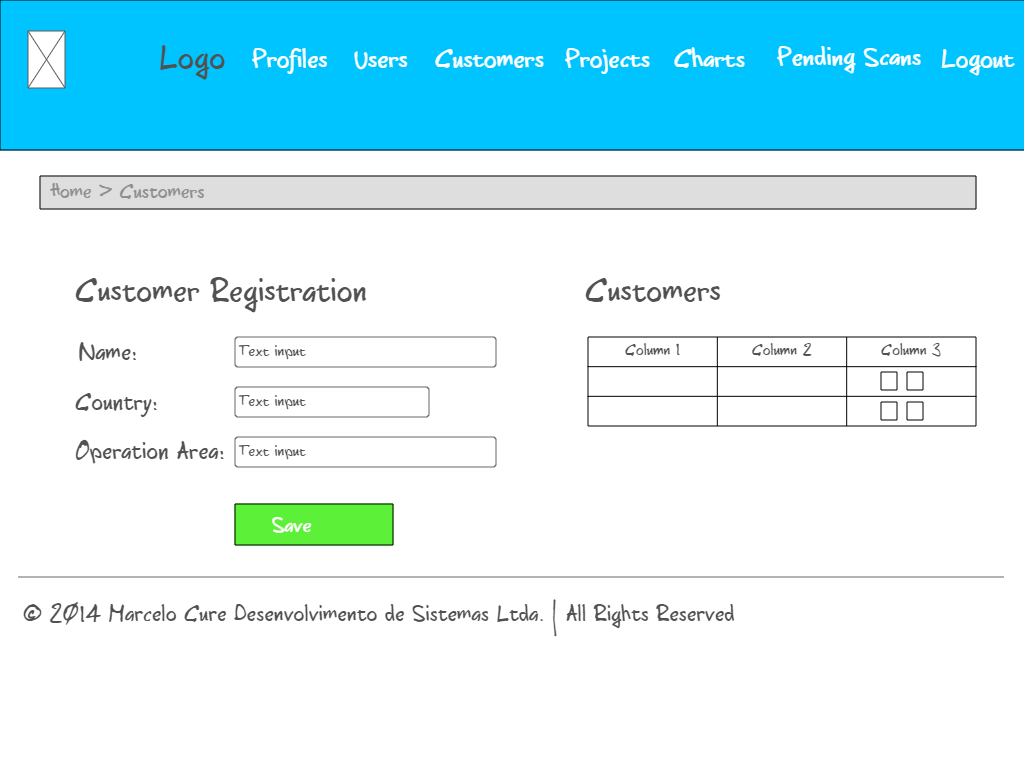


Fonte: Autoria própria.

### 7.4.9 Cadastro de clientes

A figura 16 mostra o protótipo da tela de cadastro de clientes.

Figura 16 – Protótipo da tela de cadastro de clientes.

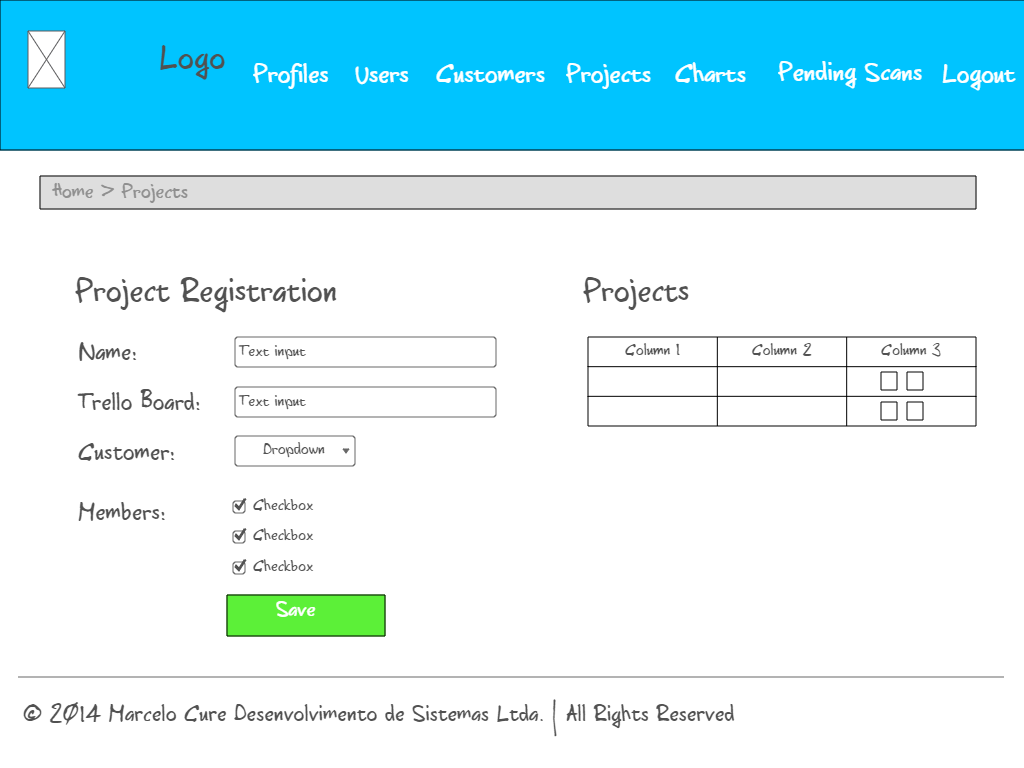


Fonte: Autoria própria.

### 7.4.10 Cadastro de projetos

A figura 17 mostra o protótipo da tela de cadastro de projetos.

Figura 17 – Protótipo da tela de cadastro de projetos.

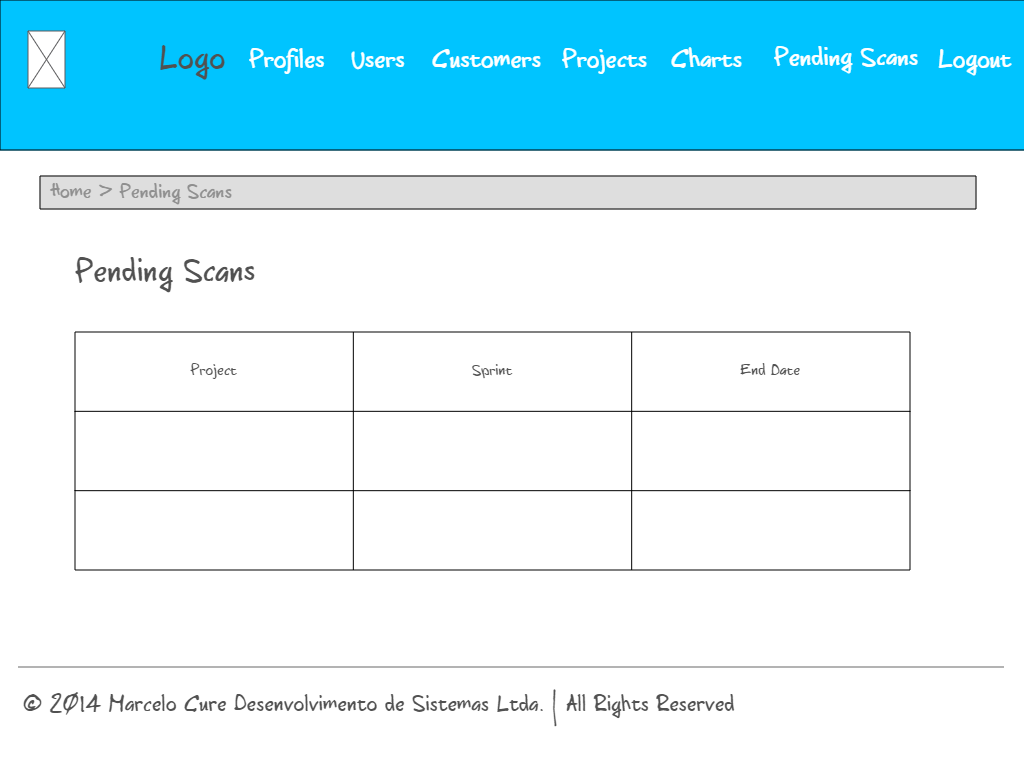


Fonte: Autoria própria.

### 7.4.11 Consulta de scans pendentes

A figura 18 mostra o protótipo da tela de scans pendentes.

Figura 18 – Protótipo da tela de scans pendentes.

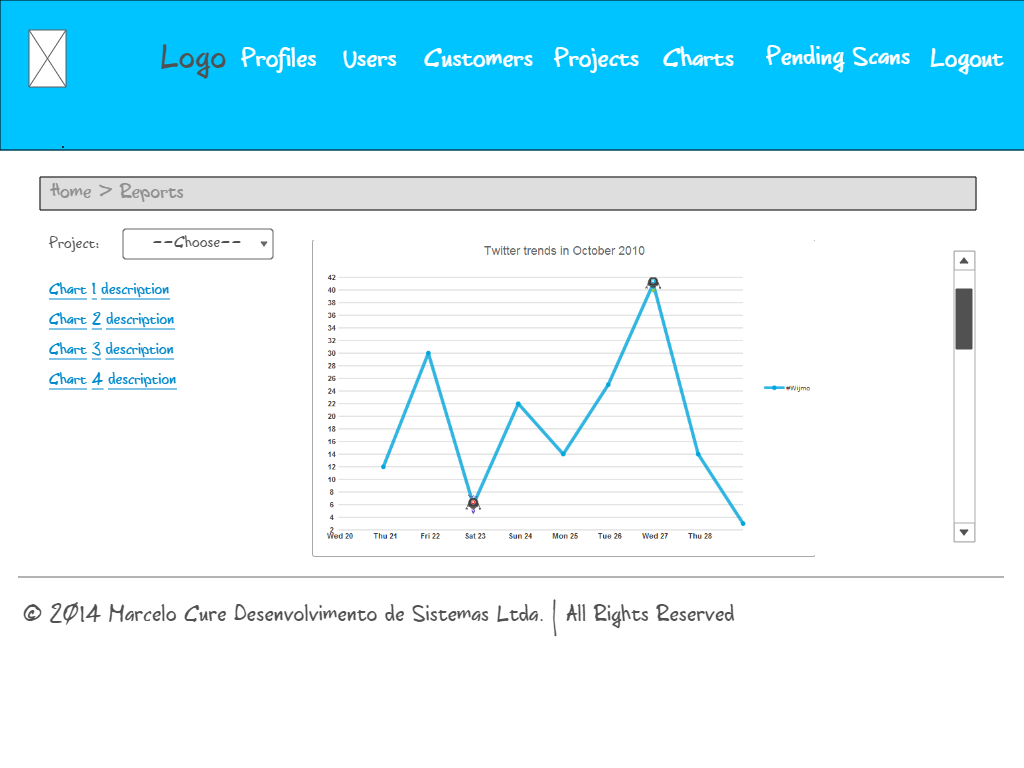


Fonte: Autoria própria.

### 7.4.12 Gráficos

A figura 19 mostra o protótipo da tela de gráficos.

Figura 19 – Protótipo da tela de gráficos.

.

Fonte: Autoria própria.

## 

## 7.5 Processo de Negócio

O processo de negócio está descrito na figura 20.

Figura 20 – Processo de negócio do sistema.



Fonte: Autoria própria.

# 8 FUNCIONAMENTO DO SISTEMA

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

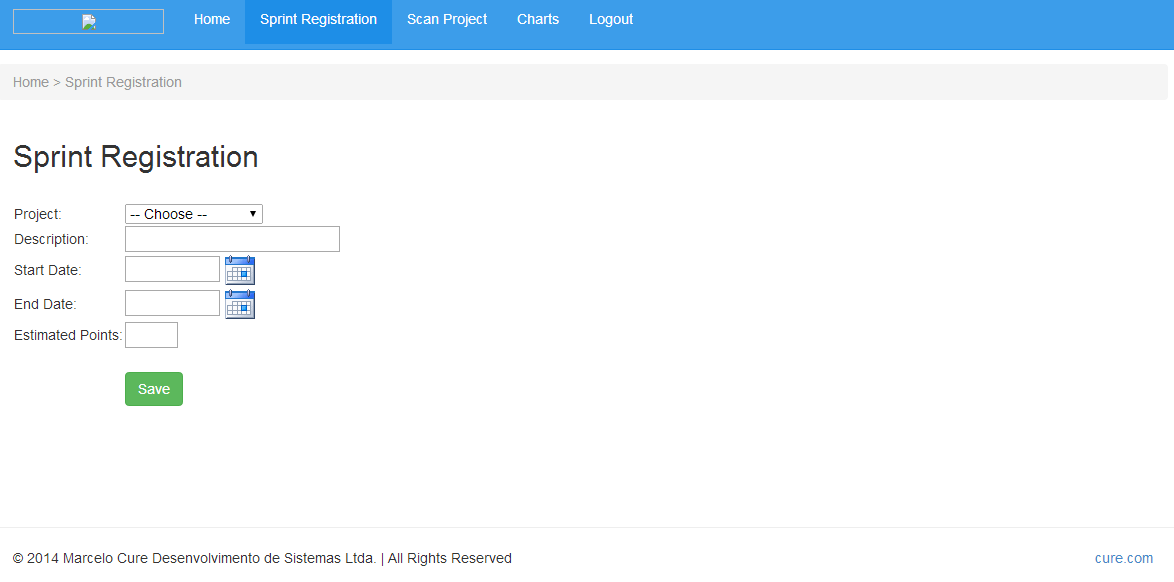
## 8.1 Membro do time

O membro do time pode participar 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 são capazes de cadastrar as *sprints* do projeto. O cadastro de *sprints* inclui o projeto em questão, operíodo de duração, quantidade de pontos estimados e a descrição das mesma. A figura 21 mostra a tela de cadastro de *sprints*.

Figura 21 – Tela de cadastro de *sprints*.

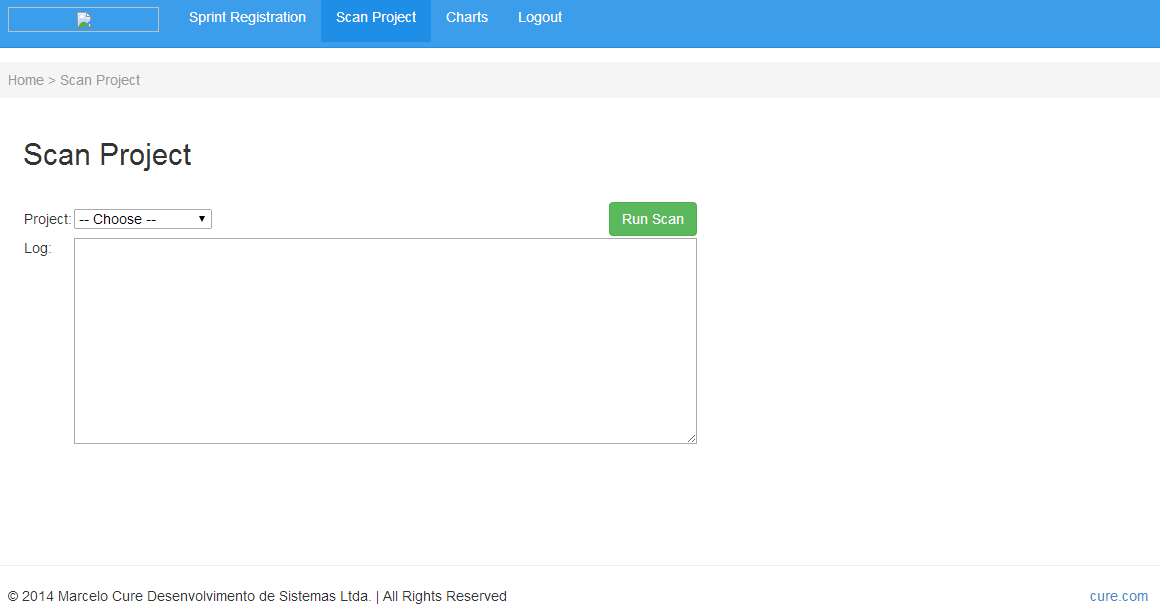


Fonte: Autoria própria.

### 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* representa 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 irá coletar 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 22 mostra a tela de scan do projeto.

Figura 22 – 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.

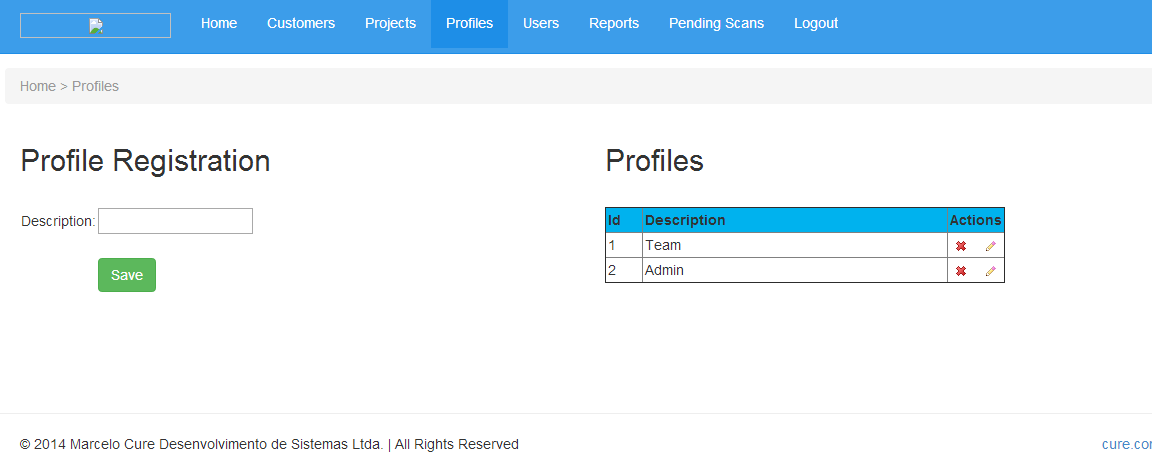
## 8.2 Administrador

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 poderá cadastrar os perfis de acesso do sistema. O sistema possui os perfis de administrador e membro do time pré-cadastrados. A figura 23 mostra a tela de cadastro de perfis.

Figura 23 – Tela de cadastro de perfis.



Fonte: Autoria própria.

### 8.2.2 Cadastro de usuários

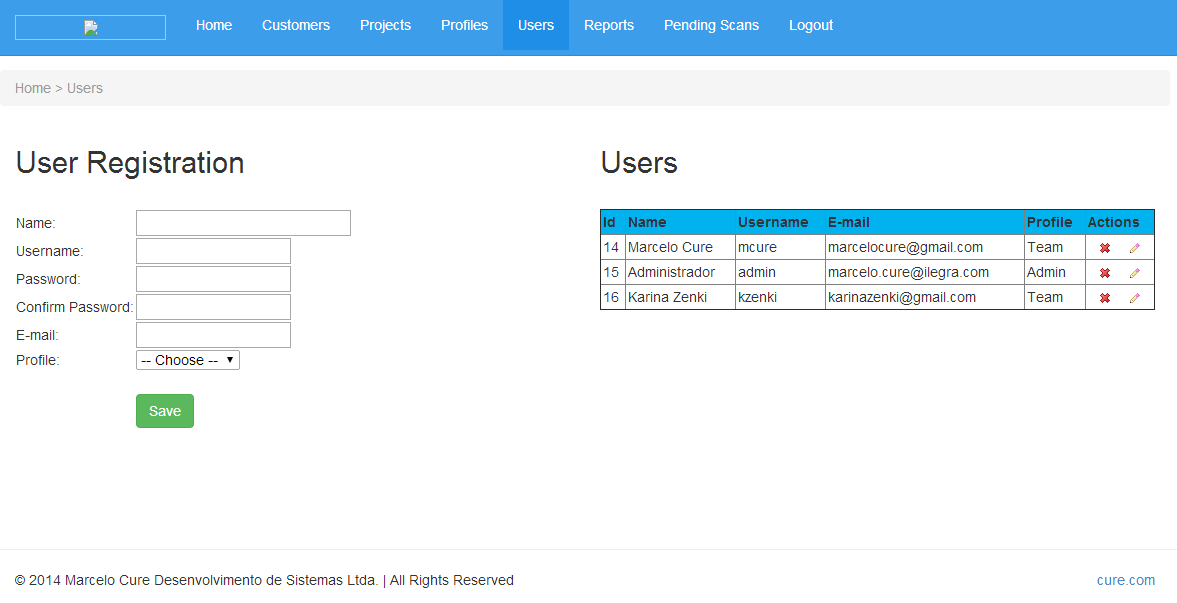
Nesta tela o administrador poderá cadastrar os usuários do sistema, onde serão cadastrados os dados básicos para um usuário do sistema, estes estão descritos abaixo:

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

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

A figura 24 mostra a tela de cadastro de usuários.

Figura 24 – Tela de cadastro de usuários.



Fonte: Autoria própria.

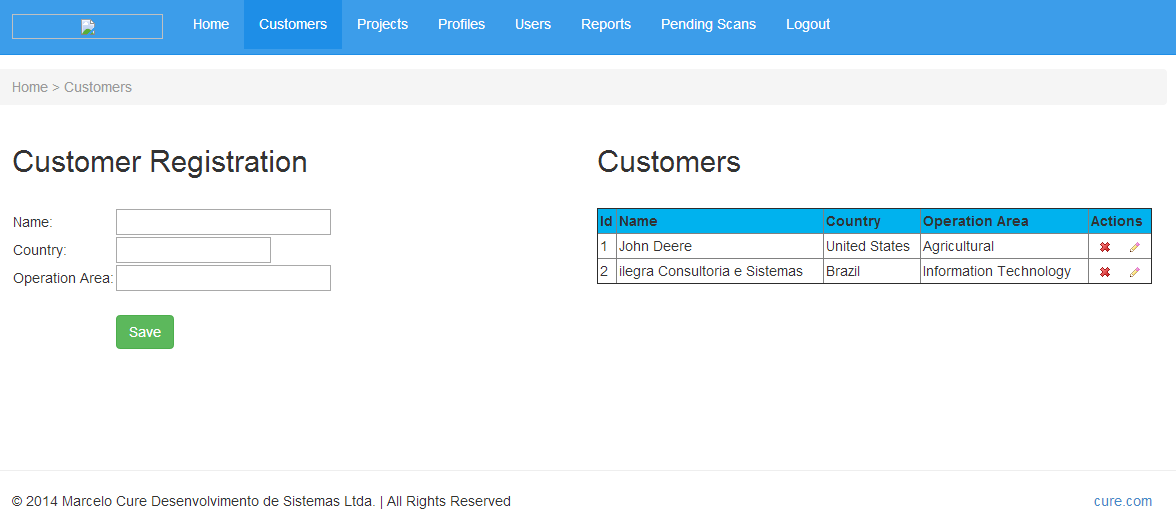
### 8.2.3 Cadastro de clientes

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 25 mostra a tela de cadastro de clientes.

Figura 25 – Tela de cadastro de clientes.



Fonte: Autoria própria.

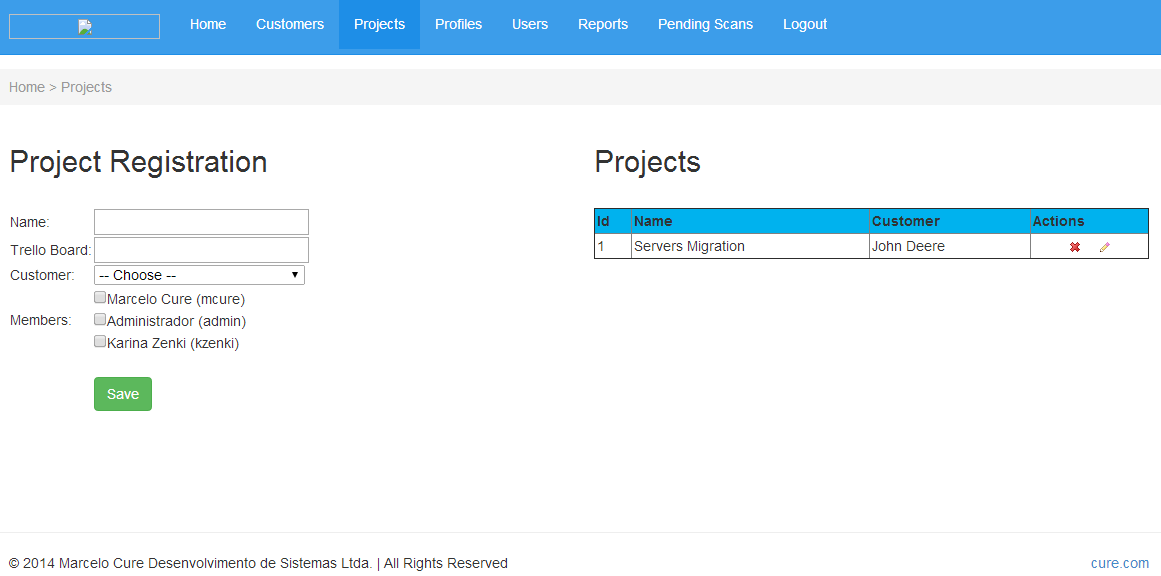
### 8.2.4 Cadastro de projetos

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 26 mostra a tela de cadastro de projetos.

Figura 26 – 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

Este capítulo apresenta a estratégia de validação deste projeto, bem como a análise dos resultados obtidos.

## 9.1 ESTRATÉGIA

O sistema será implantado na empresa de desenvolvimento de *software* ilegra (ilegra.com, 2014). Inicialmente, somente um projeto utilizará o sistema. O sistema será alimentado durante 3 *sprints*. A partir da quarta *sprint* o sistema terá dados suficientes para prover informações de apoio a decisão, ou seja, a equipe será capaz de ver dados históricos de maneira clara e objetiva, de modo a poder avaliar estimativas passadas e quantos pontos será capaz de entregar.

A validação será baseada no modelo de qualidade do produto de *software* da norma ISO/IEC 25000, também chamada SquaRE (ISO Org.).

Ao final da terceira *sprint,* quando o sistema terá dados históricos suficientes para prover informações de apoio às estimativas, os membros do time receberão um questionário baseado nos seguintes atributos de qualidade:

1. Funcionalidade: Evidencia a capacidade do *software* de atender as necessidades do usuário e ter conformidade com os requisitos do sistema.
2. Confiabilidade: Evidencia a capacidade do *software* de se manter funcional com capacidade de tratar erros.
3. Usabilidade: Evidencia a capacidade do *software* de ser compreendido, atraente para o usuário e ser intuitivo.
4. Eficiência: Evidencia se o tempo de execução das tarefas do sistema são aceitaveis e compatíveis com o cenário do sistema.

MAPEAR OS OBJETIVOS EM ATRIBUTOS/REQUISITOS, DENTRO DAS CARACTERÍSITICAS DO MODELO, IDENTIFICANDO UMA MÉTRICA PARA MEDI-LO,

INCLUIR TESTES UNITÁRIOS NA ARQUITETURA.

# 10 CRONOGRAMA

Tabela 7 - Cronograma

|  |  |  |  |
| --- | --- | --- | --- |
| Descrição da Atividade | Produto | Data | Descrição |
| 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.

ISO Org. Disponível em: <http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=64764>. Acesso em 01 jun. 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.

Molokken-Ostvold, K., Haugen, N.C. (2007). **Combining Estimates with Planning Poker--An Empirical Study**.

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 está sendo utilizada para criptografar as senhas dos usuários do sistema. (pypi.python.org/pypi/pycrypto/2.6.1, 2014)

***ChartJS***

ChartJS é uma biblioteca escrita em javascript para gerar gráficos em um ambiente web. Esta ferramenta possibilida a criação de vários tipos de gráficos. O ChartJS trabalha com os dados no formato JSON, assim fica fácil de passar os conjuntos de dados e outras configurações, como cor de fundo, cor das linhas, etc. (chartjs.org/docs/, 2014)