**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](#_Toc388716944)

[Figura 2 - Estrutura básica do *TDD* 14](#_Toc388716945)

[Figura 3 - Estrutura básica do sistema 16](#_Toc388716946)

[Figura 4 – Processo de negócio do sistema 19](#_Toc388716947)

[Figura 5 – Protótipo da tela de login 20](#_Toc388716948)

[Figura 6 – Protótipo da tela principal 20](#_Toc388716949)

[Figura 7 – Protótipo da tela de *Scan* de projetos 20](#_Toc388716950)

[Figura 8 – Protótipo da tela de contato 21](#_Toc388716951)

[Figura 9 – Protótipo da tela de relatórios 21](#_Toc388716952)

[Figura 10 – Protótipo da tela principal do administrador 22](#_Toc388716953)

**LISTA DE TABELAS**

[Tabela 1 - Cronograma 24](#_Toc388716959)

**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. Projetos de software que utilizam a metodologia *Scrum* possuem entregas frequentes chamadas de *sprints.* As *sprints* representam cada iteração do projeto, estas iterações são planejadas previamente em uma reunião chamada *Sprint Planning,* onde as estórias são selecionadas e priorizadas de acordo com necessidade do negócio do cliente. Cada estória é estimada utilizando uma técnica chamada de *Planning Poker*, onde os integrantes do time fazem a estimativa de cada estória indicando Story Points, que é a representação de esforço do *Planning Poker.*

Este sistema irá guardar as informações básicas da *sprint,* como a quantidade total de pontos estimados. Serão coletadas as informações referentes a cada estória finalizada no final da *sprint,* podendo assim totalizar a quantidade de pontos entregues ao final da mesma*.*

Conforme cada *sprint* é finalizada, estes dados serão guardados em uma base de dados. E conforme as *sprints* passam, o sistema será capaz de prover relatórios baseando-se em dados históricos mostrando estatísticas de *sprints* passadas. O time será capaz de se basear nestes dados para prover estimativas mais precisas. Isto acarretará em mais confiança por parte do time de desenvolvimento e para o cliente, sem surpresas com datas e sem custo extra para o projeto.

SUMÁRIO

[1 APRESENTAÇÃO GERAL DO PROJETO 8](#_Toc388716979)

[2 DEFINIÇÃO DO PROBLEMA 9](#_Toc388716980)

[3 OBJETIVOS 11](#_Toc388716981)

[3.1 Objetivo Geral 11](#_Toc388716982)

[3.2 Objetivos Específicos 11](#_Toc388716983)

[4 ANÁLISE DE TECNOLOGIAS/FERRAMENTAS 13](#_Toc388716984)

[4.1 Linguagens de programação 13](#_Toc388716985)

[4.2 Ferramenta de Desenvolvimento 13](#_Toc388716986)

[4.3 Banco de Dados 13](#_Toc388716987)

[4.4 Controle de Atividades 13](#_Toc388716988)

[4.5 Versionamento do Código 14](#_Toc388716989)

[4.6 Qualidade do Código 14](#_Toc388716990)

[4.7 Ferramentas de Modelagem 15](#_Toc388716991)

[4.7.1 Protótipos de tela 15](#_Toc388716992)

[4.7.2 Modelo ER 15](#_Toc388716993)

[4.7.3 Diagrama de Classes 15](#_Toc388716994)

[4.7.4 Diagrama de Casos de Uso 15](#_Toc388716995)

[5 DESCRIÇÃO DA SOLUÇÃO 16](#_Toc388716996)

[6.1 Artefatos 17](#_Toc388716997)

[6.1.1 *Product Backlog* 17](#_Toc388716998)

[6.1.2 *Sprint Backlog* 17](#_Toc388716999)

[6.1.3 *User Stories* (estórias) 17](#_Toc388717000)

[6.1.4 *Diagrama de Caso de Uso* 18](#_Toc388717001)

[6.1.5 *Diagrama de Classes* 18](#_Toc388717002)

[7 ARQUITETURA DO SISTEMA 19](#_Toc388717003)

[7.1 Modelos UML 19](#_Toc388717004)

[7.2 Modelagem de Dados 19](#_Toc388717005)

[7.3 Interface Gráfica 19](#_Toc388717006)

[7.4 Processo de Negócio 19](#_Toc388717007)

[7.5 Prototipação 19](#_Toc388717008)

[7.5.1 Tela de Login 20](#_Toc388717009)

[7.5.2 Tela principal 20](#_Toc388717010)

[7.5.3 Tela de *Scan* de projetos 20](#_Toc388717011)

[7.5.4 Tela de contato 21](#_Toc388717012)

[7.5.5 Tela de relatórios 21](#_Toc388717013)

[7.5.6 Tela principal do administrador 22](#_Toc388717014)

[8 VALIDAÇÃO 23](#_Toc388717015)

[9 CRONOGRAMA 24](#_Toc388717016)

[REFERÊNCIAS BIBLIOGRÁFICAS 27](#_Toc388717017)

[COMPONENTES REUTILIZADOS 29](#_Toc388717018)

# 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

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. De acordo com Arnaut (2008) apenas 35% dos projetos tiveram sucesso em 2006. 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.

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. Garantir a entrega da *sprint*;
3. Garantir, de forma geral, a entrega de um release;
4. Integrar o sistema com o software *Trello* ([trello.com](http://www.trello.com/), 2014);
5. Coletar informações pertinentes dos *boards* do *trello*;
6. Gerar gráficos para auxiliar nas estimativas futuras;
7. 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

Abaixo estão relacionadas as ferramentas utilizadas para fazer as modelagens necessárias para o projeto.

### 4.7.1 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).

### 4.7.2 Modelo ER

Para a criação do modelo Entidade Relacionamento será utilizado o *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).

### 4.7.3 Diagrama de Classes

Para a criação do diagrama de classes será utilizado o *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).

### 4.7.4 Diagrama de Casos de Uso

Para a criação do diagrama de casos de uso será utilizado o *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).

# 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

A arquitetura do sistema será dividida em Modelos UML, Modelagem de Dados, Interface Gráfica e Processo de Negócio.

## 7.1 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.

## 7.2 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.

## 7.3 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 Processo de Negócio

O processo de negócio está descrito na figura 4.

Figura 4 – Processo de negócio do sistema



Fonte: Autoria própria

## 7.5 Prototipação

Os protótipos das telas estão descritos abaixo.

### 7.5.1 Tela de Login

Figura 5 – Protótipo da tela de login

### 7.5.2 Tela principal

Figura 6 – Protótipo da tela principal

### 7.5.3 Tela de *Scan* de projetos

Figura 7 – Protótipo da tela de *Scan* de projetos

### 7.5.4 Tela de contato

Figura 8 – Protótipo da tela de contato

### 7.5.5 Tela de relatórios

Figura 9 – Protótipo da tela de relatórios

### 7.5.6 Tela principal do administrador

Figura 10 – Protótipo da tela principal do administrador

# 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 Sprint

Dentro de cada projeto, o integrante do time será capaz de cadastrar as *sprints* do projeto. O cadastro de *sprints* incluirá o período de duração, quantidade de pontos estimados e a descrição das mesma.

### 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.

### 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;
6. Burndown para a sprint corrente.

## 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 perfil

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.

### 8.2.2 Cadastro de usuário

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.

### 8.2.3 Cadastro de cliente

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.

### 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.

### 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;
6. Burndown para a *sprint* corrente de cada projeto;

# 8 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**

# 9 CRONOGRAMA

Tabela 1 - 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 | - Cadastro de usuários  - Cadastro de perfis  - Tela de login  - Cadastro de clientes  - Cadastro de projetos | 14/05/2014 | Desenvolvimento da primeira iteração do projeto |
| Sprint 2 | - Cadastro de sprints  - Scan parte 1 | 01/06/2014 | Desenvolvimento da 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 | - Scan parte 2  - Quantidade de pontos entregues por sprint X time  - Quantidade de testes unitários criados por sprint X time | 22/08/2014 | Desenvolvimento da terceira iteração do projeto |
| Sprint 4 | - Quantidade de pontos estimados X quantidade de pontos entregues para cada sprint  - Quantidade de pontos entregues X testes unitários para cada sprint  - Quantidade de estórias entregues por sprint por projeto  - Burndown para a sprint corrente de cada projeto | 05/09/2014 | Desenvolvimento da quarta iteração do projeto |
| Ajustes no relatório do projeto | Relatório do projeto | 07/09/2014 | Ajustes no relatório do projeto |
| Sprint 5 | - Quantidade de pontos entregues X sprint  - Quantidade de testes unitários criados X sprint  - Quantidade de pontos estimados X quantidade de pontos entregues para cada sprint | 21/09/2014 | Desenvolvimento da 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 | - Quantidade de pontos entregues X testes unitários para cada sprint  - Quantidade de estórias entregues por sprint  - Burndown para a sprint corrente  - consulta times que já rodaram o scan no final da sprint | 17/10/2014 | Desenvolvimento da 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.

# 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)

Alerta pro administrador ao final de cada sprint para avisar se algum gambeiro não rodou a coleta

***PyCrypto***

Um monte de blablabla