CENTRO UNIVERSITÁRIO UNIFACVEST
CURSO DE CIÊNCIA DA COMPUTAÇÃO
RONALDO SCOTTI DOS SANTOS

USPEED.ME: SOFTWARE PARA GESTÃO ÁGIL DE PROJETOS

LAGES
2015

RONALDO SCOTTI DOS SANTOS

USPEED.ME: SOFTWARE PARA GESTÃO ÁGIL DE PROJETOS

Projeto apresentado à Banca Examinadora do Trabalho de Conclusão de Curso II de Ciência da Computação para análise e aprovação.

Orientador: Prof. Joaquim Rodrigo de Oliveira

LAGES

2015

RONALDO SCOTTI DOS SANTOS

USPEED.ME: SOFTWARE PARA GESTÃO ÁGIL DE PROJETOS

Trabalho de Conclusão de Curso de Ciência da Computação apresentado ao Centro Universitário UNIFACVEST como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação.

Orientador: Prof. Joaquim Rodrigo de Oliveira

Lages, SC ___/___/2015.

Nota _____

_____

Márcio José Sembay

Coordenador do curso de graduação

LAGES

2015

RESUMO

O trabalho a seguir evidencia um estudo e pesquisa sobre os métodos ágeis, suas vantagens, práticas e aplicação dentro do gerenciamento de projetos por equipes de desenvolvimento de *software*. Neste projeto foi aplicada a pesquisa buscando entender e solucionar problemas que equipes enfrentam ao gerenciar de forma manual o andamento e os procedimentos presentes dentro de um projeto de *software.* Além de fazer uma análise das técnicas e ferramentas utilizadas atualmente para tentar solucionar os mesmos problemas, seus pontos fortes e os pontos em que não estão atendendo as expectativas dos usuários. Para alcançar tais metas foram realizadas pesquisas bibliográficas em livros, artigos acadêmicos, pesquisa *online* e entrevistas com especialistas no assunto. O desenvolvimento de programas e sistemas *web* promove a facilidade de acesso do usuário ao sistema, bastando apenas ter uma conexão com a *internet*. As novas tecnologias citadas no trabalho ajudam a criar uma melhor experiência ao usuário final, com boa usabilidade e mobilidade proporcionada pela programação responsiva. Com base nos dados bibliográficos e fazendo uso das ferramentas de desenvolvimento citadas dentro deste trabalho, pode-se analisar e desenvolver uma plataforma onde seja possível automatizar processos e ajudar equipes de desenvolvimento a alcançar melhores resultados, diminuindo a preocupação com os processos e possibilitando maior foco na qualidade do *software* a ser desenvolvido.

Palavras Chave: *Métodos ágeis. Sistemas web. Gerenciamento de projetos.*

ABSTRACT

That paper presents a study and research on agile methods, their advantages, practices and application in the project management for software development teams. In this project will be applied research in order to understand and solve problems that teams face in manual management of the progress and procedures present in a software project. There was also an analysis of the techniques and tools currently used to try to solve the same problems, their strengths and points that are not meeting user expectations. To achieve these goals it was conducted bibliographic searches in books, academic articles, online research and interviews with experts on agile methodologies. The development of web programs and systems promote for the user easy access to the system by simply having an internet connection. New technologies cited at work help to create a better user experience, with good usability and mobility provided by responsive programming. Based on the bibliographic information and making use of development tools cited in this paper, it is possible analyze and develop a platform where it is more easy to automate processes and to help development teams achieve better results, reducing the concern with processes and enabling more focused on the quality of the software being developed.

Keywords: *Agile methods. Web systems. Project management.*

RESUMEN

Ese trabajo se presenta un estudio y la investigación sobre los métodos ágiles, sus ventajas, las prácticas y la aplicación de la gestión de proyectos para los equipos de desarrollo de software. En este proyecto se aplicará la investigación que busca comprender y resolver los problemas que enfrentan los equipos en la gestión manual de los avances y procedimientos presentes en un proyecto de software. Había también un análisis de las técnicas y herramientas que se utilizan actualmente para tratar de resolver los mismos problemas, sus puntos fuertes y los puntos que no están cumpliendo con las expectativas del usuario. Para alcanzar estos objetivos se realizó búsquedas bibliográficas en libros, artículos académicos, la investigación en línea y entrevistas con expertos en metodologías ágiles. El desarrollo de programas y sistemas web promover para el usuario un fácil acceso al sistema por el simple hecho de tener una conexión a Internet. Las nuevas tecnologías citadas en esto trabajo ayudan a crear una mejor experiencia de usuario, con buena usabilidad y la movilidad proporcionada por la programación para dispositivos móviles. Sobre la base de la información bibliográfica y haciendo uso de las herramientas de desarrollo citados en este documento, es posible analizar y desarrollar una plataforma en la que es más fácil automatizar los procesos y ayudar a los equipos de desarrollo a lograr mejores resultados, reduciendo la preocupación por los procesos y permitiendo ser más centrado en la calidad del software que está siendo desarrollado.

Palabras Clave: *Métodos ágiles. Sistemas web. Gestión de proyectos.*

LISTA DE FIGURAS

Figura 1. Razões para adotar *agile* 19
Figura 2. Metodologias ágeis mais usadas 20
Figura 3. Visão geral do framework Scrum 21
Figura 4. Exemplo de História de Usuário 24
Figura 5. Quadro Visual Kanban 25
Figura 6. Diferença entre Tabelas SQL e Documentos NoSQL 28
Figura 7. Exemplo de criação de servidor NodeJS 29
Figura 8. Nível de conhecimento dos entrevistados 34
Figura 9. Metodologias utilizadas 34
Figura 10. Ferramentas mais citadas 35
Figura 11. Testes unitários no server-side 38
Figura 12. Tela de cadastro 39
Figura 13. Tela de *login* 40
Figura 14. Tela de dashboard 40
Figura 15. Tela de criação e edição de projetos 41
Figura 16. Tela de listagem de projetos 42
Figura 17. Exibição de um projeto 43
Figura 18. Tela de cadastro e edição de tarefas 43
Figura 19. Listagem de tarefas - Modo tabela 44
Figura 20. Listagem de tarefas com filtro aplicado 45
Figura 21. Quadro Kanban 45
Figura 22. Chat 46
Figura 23. Diagrama de Sequência Criar Tarefa 47
Figura 24. Diagrama de componentes 48
Figura 25. Diagrama de implementação 48
Figura 26. Documento tarefa 50

LISTA DE SIGLAS

BSON	–	*Binary Serialization Object Notation*
CERN	–	*Organização Europeia para a Pesquisa Nuclear*
CSS	–	*Cascade Style Sheet*
DOM	–	*Document Object Model*
ECC	–	*Error Correcting Code*
ED	–	*Equipe de desenvolvimento*
FDD	–	*Feature Driven Development*
GB	–	*Giga Byte*
GHz	–	*Giga Hertz*
HTML	–	*Hipertext Markup Language*
HTTP	–	*Hipertext Transfer Protocol*
I/O	–	*Input/Output*
IDE	–	*Integrated Developed Environment*
JS	–	*JavaScript*
JSON	–	*JavaScript Object Notation*
MA	–	*Modelagem Ágil*
MEAN	–	*MongoDB, ExpressJS, AngularJS e NodeJS*
MVC	–	*Model View Controller*
MVP	–	*Minumum Viable Product*
PDCA	–	*Plan, Do, Check, Act*
PHP	–	*PHP: Hypertext Preprocessor*
PO	–	*Product Owner*
RAM	–	*Random-Access Memory*
SM	–	*Scrum Master*
SSD	–	*Solid-State Drive*
TB	–	*Tera Byte*
TI	–	*Tecnologia da Informação*
URL	–	*Uniform Resource Locator*
XHR	–	*XML HTTP Request*
XP	–	*Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO 12

1.1 Justificativa 12

1.2 Importância 13

1.3 Objetivos gerais 13

1.3.1 Objetivos específicos 13

2 GERENCIAMENTO ÁGIL DE PROJETOS 15

2.1 Engenharia de Software 15

2.2 Breve Histórico 15

2.3 O Manifesto Ágil 16

2.3.1 Valores do manifesto ágil 16

2.3.2 Princípios por trás do manifesto ágil 17

2.3.3 Benefícios dos métodos ágeis 18

2.4 Métodos ágeis no Brasil 19

2.5 Metodologias de desenvolvimento ágil 19

2.6 Scrum 21

2.6.1 Os papéis no Scrum 21

2.6.2 Os artefatos do Scrum 22

2.6.3 As cerimônias do Scrum 23

2.7 Extreme Programming (XP) 23

2.7.1 Histórias de usuário 24

2.7.2 Testes automatizados 24

2.8 Kanban 24

3 PROGRAMAÇÃO WEB 26

3.1 JavaScript 26

3.2 Desenvolvimento web com MEAN 27

3.2.1 MongoDB 27

3.2.2 ExpressJS 28

3.2.3 AngularJS 28

3.2.4 NodeJS 28

3.3 Socket.io 29

3.4 HTML5 com EJS 30

3.5 CSS3 30

3.6 Design Responsivo com Bootstrap 30

3.7 Padrões de Projeto com MVC 30
3.8 Ferramentas do Projeto 31
3.8.1 Atom 31
3.8.2 Controle de versão com git 31
4 METODOLOGIA 32
4.1 Documentação 32
4.2 Natureza da Pesquisa 32
4.3 Tipo da Pesquisa 32
4.4 Técnicas de Pesquisa 32
4.5 Coleta de Dados 33
4.6 Validação do Projeto 33
5 PROJETO 37
5.1 Hardware 37
5.2 Qualidade do código e padrões de projeto 37
5.2.1 Padrões no código 37
5.2.2 Testes unitários 38
5.3 Telas do Sistema 38
5.3.1 Cadastro de usuário 39
5.3.2 Login 39
5.3.3 Dashboard 40
5.3.4 Criação e edição de projetos 41
5.3.5 Listagem de projetos 42
5.3.6 Visualização de um projeto 42
5.3.7 Criação e edição de tarefas 43
5.3.8 Listagem de tarefas 44
5.3.9 Chat 46
5.4 Interface Responsiva 46
5.5 Modelagem Ágil 47
5.5.1 Diagrama de Sequência 47
5.5.2 Diagrama de Componentes 48
5.5.3 Diagrama de implementação 48
5.5.4 Histórias de usuário do *uSpeed.me* 49
5.6 Data Model Design 49
6 CONCLUSÃO 51
REFERÊNCIAS 52

ANEXO A – Código do servidor NodeJS 56
ANEXO B – Código de criação do módulo principal com AngularJS 57
ANEXO C – Função de criação de novo usuário 58
ANEXO D – Criação de rotas do módulo projeto 59
ANEXO E – Configuração de sessões no Socket.io 60

1 INTRODUÇÃO

Desenvolver *software* é uma atividade recente na história da humanidade, se comparada com outras ciências existentes. Ainda estamos aprendendo como tudo isso funciona e qual a melhor forma de lidar com esses processos.

No início, não se tinham metodologias definidas para desenvolvimento de *software*. Com o passar do tempo criaram-se métodos cada vez mais sofisticados, até chegarem ao ponto de ficarem excessivos, prejudicando o rendimento de equipes, ocasionando atrasos e sugando energia de desenvolvedores em atividades que não eram necessárias.

Para resolver esse problema as chamadas Metodologias Ágeis foram criadas, representando visões distintas de cada um de seus criadores. Divergem na forma de alcançar os objetivos, mas convergem na essência. Todas colocam as pessoas no centro da questão, refletindo um melhor entendimento sobre as virtudes e fraquezas humanas. E a partir daí, propõem mecanismos para potencializar o desempenho de quem escreve códigos, detectar e corrigir falhas mais rapidamente e assegurar uma comunicação entre as pessoas de forma mais eficaz (TELES, 2014).

Atualmente existem diversas metodologias sendo usadas em larga escala por empresas do mundo todo, cada uma com suas peculiaridades e processos, porém sempre embasadas no mesmo princípio: desenvolver *software* funcionando e com qualidade. Embora isso pareça óbvio, a história nos mostra que nem sempre isso foi uma prioridade.

Tendo como base o conhecimento de que se torna oneroso controlar todos esses processos e interações entre membros da equipe, e ainda prezar e atentar-se constantemente pela qualidade do *software*, é originado o questionamento que motiva este trabalho: não seria mais cômodo que uma ferramenta auxiliasse automatizando esses processos?

1.1 Justificativa

Este estudo justifica-se pelo fato de que alavancar um processo de mudança é uma tarefa extremamente difícil. Cada equipe é distinta. O contexto de cada processo é único. Pessoas tem habilidades diferentes, os níveis de maturidades são diversos.

No entanto, é possível encontrar elementos comuns a uma boa quantidade de processos em diferentes metodologias. Invariavelmente, equipes precisam gerar valor para o cliente. Pessoas precisam colaborar e fazer o trabalho fluir (VALE, 2014).

A ferramenta desenvolvida possibilita uma gestão visual do planejamento e do acompanhamento do projeto, facilitando e diminuindo o número de preocupações da equipe de

desenvolvimento, permitindo assim que programadores e demais membros da equipe deem maior foco à qualidade do *software* a ser entregue.

Mas, se o Manifesto ágil (2001) diz “Indivíduos e interação entre eles mais que processos e ferramentas [...]”, por que eu desenvolvi uma ferramenta para ser utilizada neste contexto?

O manifesto não decreta que não se deve usar ferramentas, apenas diz que não podemos deixar que ferramentas substituam o contato e a interação entre os membros da equipe, que é muito importante durante o andamento do projeto. Ferramentas que são criadas com o intuito de automatizar processos e melhorar o rendimento serão sempre relevantes (REHEM, 2015).

1.2 Importância

Os projetos contemporâneos são caracterizados por alta velocidade, grandes mudanças, busca por menores custos, alta complexidade, muita incerteza e uma série de outros fatores. Isto representa um desafio assustador para o gerente de projetos (WYSOCKI, 2014).

O objetivo da ferramenta que criei não é substituir este contato entre os integrantes da equipe, mas sim contribuir para que essa interação ocorra da melhor forma possível. Tudo isso sem deixar nenhuma tarefa para trás, e tendo total controle do andamento e cumprimento de prazos do projeto.

Estes fatores aumentam a cada dia a necessidade de adoção do gerenciamento de projetos em pequenas, médias e grandes empresas. Sua importância está relacionada à redução de custos no desenvolvimento de projetos, cumprimento de prazos, eficácia no resultado final e mensuração de resultados.

1.3 Objetivos gerais

Desenvolvimento de uma ferramenta que facilite o gerenciamento de projetos de *software*, mais especificamente em equipes que usem metodologias ágeis de desenvolvimento, fazendo a distribuição de tarefas e controle visual do andamento do projeto. Tudo isso embasado massivamente nos fundamentos das metodologias ágeis XP, *Scrum* e *Kanban.*

1.3.1 Objetivos específicos

Para um projeto obter sucesso é necessário que atenda às necessidades do usuário final e elimine, ou pelo menos minimize os problemas que ele encontra ao realizar determinadas atividades. Automatizar e facilitar a gestão e acompanhamento de projetos de software será o

principal objetivo deste trabalho, provendo assim, que projetos sejam entregues com mais facilidade e com menos percalços durante o seu andamento.

Abaixo serão listadas as principais funcionalidades desenvolvidas neste projeto:

a) **Área restrita segura:** Sistema de cadastro de usuários e *login* seguro com dados criptografados.
b) **Criação de Projetos:** Descrever e definir seu tempo estimado para entrega, permitindo posteriormente anexar tarefas a ele.
c) **Criação de Tarefas:** Nomear tarefas para membros da equipe, com a descrição do que deverá ser executado, tempo máximo para entrega e qual sua situação no projeto. É possível também adicionar uma história de usuário a tarefa.
d) **Possibilitar gestão visual:** Permite ao usuário uma visão limpa e ampla do que precisa ser concluído e de como o projeto está fluindo. Nesta parte o método *Kanban* está presente, com seus gráficos e quadros que provem uma visão clara do andamento do projeto.
e) **Elaborar formas de relacionamento entre membros da equipe:** Um *chat* em tempo real está disponível parar conversas entre usuários.
f) **Filtros e buscas:** Possibilita ao usuário a filtragem dos resultados apresentados nas telas de projetos e tarefas.

Na sessão seguinte serão explicadas de forma ampla e clara as metodologias ágeis, desde sua criação até sua utilização nos dias atuais, bem como as tecnologias que serão adotadas para desenvolvimento do projeto.

2 GERENCIAMENTO ÁGIL DE PROJETOS

2.1 Engenharia de Software

A Engenharia de *Software* nasceu e vem se desenvolvendo junto do aumento da complexidade no desenvolvimento de *softwares*. A maior causa desta complexidade é em razão das máquinas estarem atingindo níveis de magnitude e processamento maiores no decorrer dos anos (CASSEZ, et al. 2009).

É o processo em que cada *software* é planejado, modelado, desenvolvido, implementado e gerenciado. Também se considera Engenharia de *Software* o replanejamento de *softwares* já existentes, quando se tem o intuito de mudar as regras de negócio, funções ou performance, por exemplo (FOSTER, 2014).

2.2 Breve Histórico

A criação de *software* evolui a cada dia, e se tornou uma das mais importantes indústrias da era moderna. Porém, muitas vezes a qualidade do *software* não acompanha este crescimento, fazendo com que algumas funcionalidades, requisitos e desempenho acabem sendo sacrificados ao longo da jornada do desenvolvimento.

Durante muito tempo, a Engenharia de *Software* inspirou-se em processos de manufatura para aplicar seus métodos de trabalho. Tendo início em meados do século XX, buscou em pontos da indústria da época que estavam em crescimento boa parte de suas teorias e métodos de produção. Inspirava-se principalmente em indústrias do campo automobilístico para a criação da nova indústria de TI. Graças ao modelo de produção em série de Henry Ford, altamente inspirado por Frederick Taylor, todo o pensamento tradicional da ciência do desenvolvimento de *software* foi se expandindo com grande foco na padronização de componentes e processos e na mecanização do movimento (GOMES; WILLI; REHEM, 2014).

Devido a isso no início da década de 90, visando diminuir a burocracia dos processos de desenvolvimento de *software* foram criadas novas abordagens mais leves, que se mostraram muito mais úteis e efetivas que tentativas anteriores (GOMES, 2013).

Essas novas metodologias passaram a ser chamadas de ágeis a partir de 2001, quando um grupo de 17 especialistas se reuniu em Utah, no Estados Unidos, para discutir maneiras de desenvolver *softwares* de uma forma mais leve, rápida e centralizada em pessoas (GOMES; WILLI; REHEM, 2014).

Pela primeira vez, a indústria de *software* encontrou uma forma real e sustentável para resolver problemas que sufocavam várias gerações de times de desenvolvimento (STELLMAN; GREENE, 2014).

2.3 O Manifesto Ágil

O Manifesto Ágil é composto por uma série de valores e princípios, porém a agilidade não está tão relacionada com o seguimento de normas específicas ou protocolos pré-definidos de produção, e sim com mudanças de atitude e de comportamento.

Agile é uma maneira de desenvolver *softwares* que nos faz lembrar que apesar de serem os computadores que rodam o código, são pessoas que o criam e o mantém. São atitudes e abordagens para entregar *software* de maneira simples, rápida e objetiva. Aumentando drasticamente as chances de sucesso, por trazer o melhor que a equipe de desenvolvimento pode oferecer (RASMUSSON, 2010).

Os profissionais que deram origem ao manifesto ágil foram Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin,Steve Mellor, Ken Schwaber, Jeff Sutherlande e Dave Thomas (GOMES, 2013).

2.3.1 Valores do manifesto ágil

Vejamos o que diz o Manifesto ágil (2001) sobre os valores a serem seguidos no desenvolvimento de *software*:

> Estamos descobrindo maneiras melhores de desenvolver *software* fazendo-o nós mesmos e ajudando outros a fazê-lo. Através deste trabalho, passamos a valorizar: Indivíduos e interação entre eles mais que processos e ferramentas; Software em funcionamento mais que documentação abrangente; Colaboração com o cliente mais que negociação de contratos; Responder a mudanças mais que seguir um plano; Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.

Analisando a última frase, podemos notar que de fato há sim valor nos itens da direita. Porém, por muito tempo a engenharia de *software* focava suas soluções demasiadamente nestes itens, deixando em segundo plano outros itens que também era importantes – os itens da esquerda (GOMES; WILLI; REHEM, 2014).

Abaixo um detalhamento melhor do que são, e qual o sentido de se adotar os valores ágeis no dia a dia do desenvolvimento de *software*.

a) **Indivíduos e interações:** As pessoas na equipe são mais importantes do que seus papéis em diagramas de processos. Assim, embora descrições de processos possam

ser necessárias para se começar o trabalho, as pessoas envolvidas nos procedimentos não podem ser trocadas como peças (COCKBURN, 2007).

b) **Software em funcionamento:** Clientes se interessam por resultados, ou seja, *software* em funcionamento que entregue um valor real ao negócio (HIGHSMITH, 2002).
c) **Colaboração com o cliente:** No desenvolvimento Ágil não há nós e eles, há apenas nós, significando que clientes e desenvolvedores estão do mesmo lado, colaborando para desenvolver o *software* que possa trazer mais valor para o usuário. Ambos são necessários para que se produza um software de qualidade (COCKBURN, 2007).
d) **Responder a mudanças:** Equipes de desenvolvimento precisam estar sempre à procura de mudanças e buscando saber se estão respondendo de forma apropriada as transformações das necessidades do usuário. Se as circunstâncias mudarem, o projeto precisará de um novo plano (STELLMAN; GREENE, 2014).

Falando em outras palavras, uma equipe que aplica os valores ágeis para ajudar a atingir seus objetivos de criar um *software* que traga valor para seus usuários, maior colaboração e resposta a mudanças, vai conseguir desenvolver melhor seus projetos do que uma equipe que simplesmente planeja, programa e documenta.

2.3.2 Princípios por trás do manifesto ágil

Os princípios do Manifesto Ágil, juntos dos valores, formam os pilares sobre os quais são estabelecidos os denominados Métodos Ágeis. São eles:

Quadro 1. Os princípios do Manifesto Ágil

a) Nossa maior prioridade é satisfazer ao cliente com entregas contínuas e adiantadas de *software* com valor agregado.
b) Mudanças nos requisitos são bem vindas, mesmo tardiamente no desenvolvimento. Os processos ágeis tiram vantagem das mudanças, visando à vantagem competitiva para o cliente.
c) Entregar frequentemente *software* funcionando, de poucas semanas a poucos meses, com preferência a menor escala de tempo.
d) Pessoas de negócio e desenvolvedores deve trabalhar diariamente em conjunto por todo o projeto.

e) Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessários e confie neles para realizar o trabalho.
f) O método mais eficiente e eficaz de transmitir informação para a equipe e entre a equipe de desenvolvimento é a conversa frente a frente.
g) *Software* funcional é a medida primária de progresso.
h) Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante sempre.
i) Contínua atenção à excelência técnica e bom *design* aumentam a agilidade.
j) Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial.
k) As melhores arquiteturas, requisitos e design emergem de times auto-organizáveis.
l) Em intervalos regulares, o time reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Fonte: GOMES; WILLI; REHEM (2014)

Agile é baseado principalmente em entrega contínua de *software* de qualidade para um cliente que busca mudar constantemente, visando se manter competitivo no mercado. O trabalho de uma equipe ágil é manter seu cliente feliz (LANGR; OTTINGER, 2011).

Esta ideia de desenvolvimento contínuo, é difundida em métodos ágeis. Ele inclui o próprio ciclo de vida de desenvolvimento, mas também habilidades de aprendizagem, levantamento de requisitos, desenvolvimento de produtos, treinamento de usuários, e tudo o mais. Ela engloba todas as atividades, em todos os níveis (SUBRAMANIAM; HUNT, 2006).

2.3.3 Benefícios dos métodos ágeis

Uma das motivações mais expressivas para a transição para métodos ágeis são os benefícios trazidos para a organização devido ao valor que é agregado ao cliente com velocidade e qualidade (GOMES, 2013).

Em uma pesquisa realizada pela VersionOne.com em 2014 que envolveu mais de 3.000 mil pessoas e organizações com variados perfis na indústria de *software* e espalhados por todo o mundo, são mostradas algumas das principais vantagens alcançadas, e o que buscavam após a transição para métodos ágeis. Dentre os principais benefícios, estão:

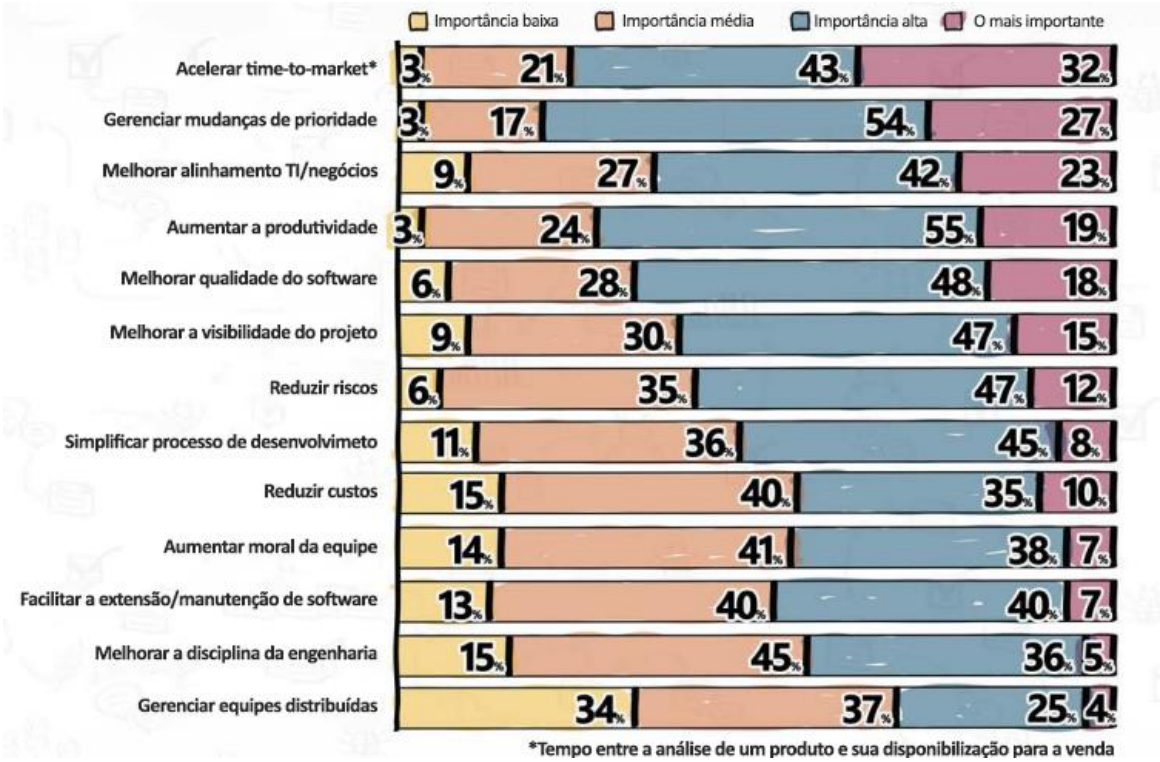

Figura 1. Razões para adotar *agile*
Fonte: VERSIONONE (2014)

2.4 Métodos ágeis no Brasil

Vários profissionais reportaram que o modo de pensar ágil já era adotado no desenvolvimento de *software* no Brasil bem ates da criação do Manifesto ágil.

Com a formalização do Manifesto, o movimento teve início no Brasil em 2001, introduzido principalmente por professores universitários e por profissionais da área que tiveram contato com o evento internacional iniciado em 2001 nos Estados Unidos da América (GOLDMAN et al., 2014).

Desde então, a disciplina foi introduzida em universidades de todo pais. Eventos também vem sendo organizados ao longo dos anos, sendo *Programação Extrema'2002* o primeiro evento ágil realizado no país. Atualmente o evento mais importante que trata sobre desenvolvimento ágil é o *AgileBrazil*, que teve início em 2010 e a partir daí vem sendo realizado todos os anos (GOLDMAN et al., 2014).

2.5 Metodologias de desenvolvimento ágil

Para facilitar o aprendizado e a prática de metodologias ágeis no desenvolvimento de *software*, foram criados vários *frameworks*. Segundo o dicionário Collins, *framework* é um

conjunto particular de regras, ideias e crenças que você usa com o intuito de lidar com problemas reais ou decidir o que fazer (COLLINS, 2011).

A pesquisa realizada pela VersionOne.com em 2014, mostrou quais são os *frameworks* mais utilizados atualmente:

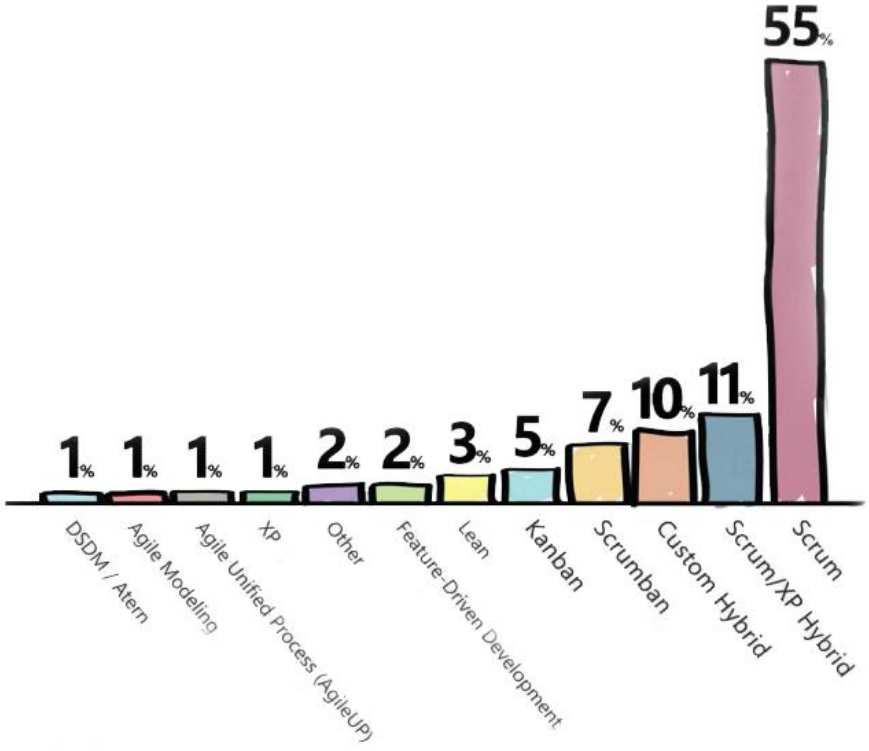

Figura 2. Metodologias ágeis mais usadas
Fonte: VERSIONONE (2014)

Como podemos ver, são variadas as metodologias utilizadas, porém o *framework Scrum* vem sendo o mais utilizado para aplicar as metodologias ágeis, obtendo um total de 73%, se forem somados os percentuais de uso de suas variações.

Nenhum método é completo e nenhum é perfeito, por isso, podemos nos sentir livres para combinar as ferramentas e aproveitar o que funcionar melhor para cada contexto, considerando a cultura da organização em questão, as habilidades da equipe e as restrições de projeto que estiver em andamento. Não se deve buscar o melhor método, e sim o método mais adequado para cada contexto (GOMES, 2013).

No presente trabalho, serão aprofundados os estudos no *framework Scrum*, por ser o mais utilizado por equipes ágeis, O *Extreme Programming* por ter uma forte base na qualidade do código escrito e o *Kanban*, por ser uma poderosa ferramenta de representação visual do projeto. Foram também essas as metodologias usadas como base no desenvolvimento do sistema de gerenciamento de projetos.

2.6 Scrum

Seu surgimento ocorreu na empresa *Easel Corporation* em 1993, quando Jeff Sutherland, John Scumniotales e Jeff McKenna o conceberam, documentaram e implantaram. Tudo isso incorporando os estilos de gerenciamento observados por Takeuchi e Nokata, em um artigo publicado na *Harvard Business Review* em 1986, onde comparavam equipes de alto rendimento e multidisciplinares com a formação *Scrum*, existente em jogadas de *rugby* (VISCARDI, 2013).

O *Scrum* trata de problemas bem específicos no desenvolvimento de *software*, como a grande variação nos requisitos e o alto grau de imprevisibilidade. As funcionalidades consideradas de maior valor são desenvolvidas de forma antecipada, enquanto se reflete sobre a necessidade ou não das menos prioritárias. Se mudanças forem necessárias, a equipe poderá modificar facilmente suas prioridades (PRIKLADNICKI; MAGNO, 2014).

A figura 3 apresenta uma visão geral do *framework* do *Scrum*:

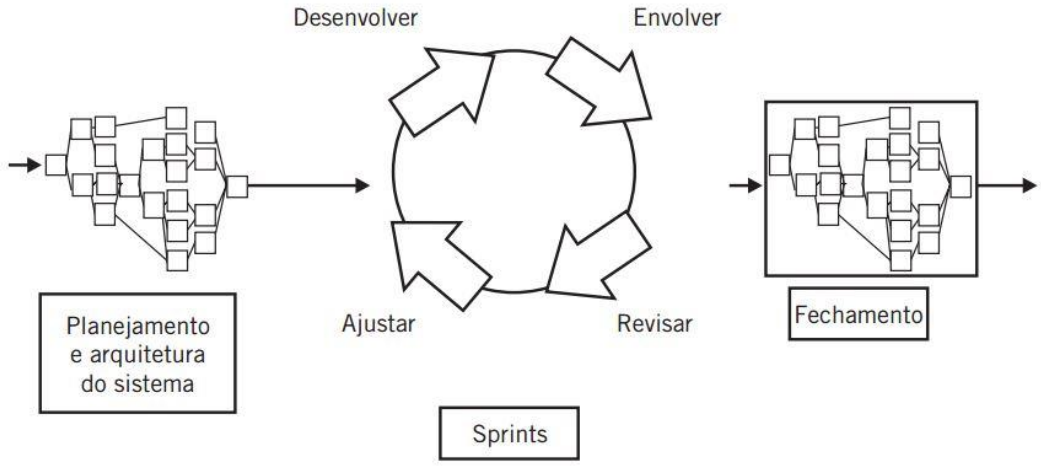

Figura 3. Visão geral do framework Scrum
Fonte: SCHWABER (2004)

A primeira e a última fase (Planejamento e Fechamento) são compostas por processos com entradas e saídas bem definidas. As *sprints* são processos baseados na experiência, e seguem o ciclo PDCA (*Plan, Do, Check, Act*) (PRIKLADNICKI; MAGNO, 2014).

2.6.1 Os papéis no Scrum

Para facilitar o gerenciamento, o *Scrum* distribui a gestão do projeto entre três papéis distintos: O *Product Owner* (PO), o *ScrumMaster* (SM) e a Equipe de Desenvolvimento (ED). Veremos abaixo uma descrição mais detalhada sobre cada papel:

a) ***Product Owner*:** É o responsável pelo sucesso do produto. Trabalha com os clientes, e com qualquer outra parte que possa estar interessada ou que possa contribuir com o melhor entendimento da Visão do Produto, inclusive com os próprios usuários, que de tempo em tempo receberão partes prontas do produto para serem utilizadas (SABBAGH, 2014).
b) ***ScrumMaster*:** É considerado o cão-de-guarda do processo. Entende as razões por traz de um processo empírico, e da seu melhor para manter o desenvolvimento do produto fluindo da melhor maneira possível. Busca manter o time focado em atingir os objetivos da *Sprint* e ajuda o PO a manter o *Backlog* organizado. É quem mais conhece sobre *Scrum*, e deve garantir que as regras e os valores ágeis estejam sendo seguidos (VISCARDI, 2013)
c) **Equipe de Desenvolvimento:** É o grupo de pessoas responsável por entregar funcionalidades com qualidade no final de cada *Sprint*. São pessoas multitarefas e auto-organizáveis, porém isso não significa que todos podem executar qualquer tarefa, e sim que não existe apenas um especialista de cada área na equipe, mas várias pessoas que entendem daquele assunto, complementando as habilidades um do outro. São indicados de três até nove membros em uma equipe de desenvolvimento (HUNDERMARK, 2014).

2.6.2 Os artefatos do Scrum

O *Scrum* possui alguns artefatos que nos dão uma visão mais detalhada sobre o andamento do projeto e das *Sprints*. Abaixo serão descritos de forma detalhada quais são os principais artefatos:

a) ***Backlog* do Produto:** É uma lista ordenada de tudo que deve ser realizado no produto, sendo a origem única dos requisitos e de qualquer mudança a ser feita no projeto. É onde ficam todas as tarefas a serem realizadas durante o projeto, aguardando para serem executadas (SCHWABER; SUTHERLAND, 2014).
b) ***Backlog* da *Sprint*:** É o conjunto de itens do *Backlog* do produto a serem realizados na *Sprint* mais o plano para transformá-los em um incremento. Seu objetivo é tornar visível o trabalho que será necessário para que a Equipe de Desenvolvimento atinja a meta da *Sprint* (PRIKLADNICKI; MAGNO, 2014).
c) **Incremento do Produto:** É a soma de todos os itens do *Backlog* do Produto completados durante a *Sprint* e o valor dos incrementos de todas as *Sprints* já

realizadas até o momento. Ao final de cada *Sprint*, a equipe de desenvolvimento entrega um novo incremento do produto (SCHWABER; SUTHERLAND, 2014).

2.6.3 As cerimônias do Scrum

O *Scrum* possui alguns eventos de duração fixa, realizados em intervalos regulares com o objetivo de inspeção e adaptação.

Sprints são ciclos completos de desenvolvimento com duração fixa que, ao final, resultam em incrementos potencialmente entregáveis do produto. Cada *Sprint* tem duração de até um mês, e permite *feedbacks* constantes do *Product Owner*. No início de cada *Sprint*, há a Reunião de Planejamento da *Sprint*, onde a equipe se reúne para definir o que será entregue naquele ciclo. Diariamente, a equipe de desenvolvimento se reúne por no máximo 15 minutos, com o intuito de revelar o que fez no último dia, o que pretende fazer hoje e qual problema está enfrentando para realizar suas tarefas, essa reunião é chamada de *Scrum* Diária. Ao final de cada *Sprint*, existe a cerimonia chamada de Revisão da *Sprint*, onde são apresentadas e inspecionadas as novas funcionalidades desenvolvidas. Logo após a Revisão, é realizada a Retrospectiva da *Sprint*, onde o foco é o aprimoramento dos processos, identificação de problemas e medidas a serem tomadas para a melhoria do processo para a próxima *Sprint* (PRIKLADNICKI; MAGNO, 2014).

2.7 Extreme Programming (XP)

É a combinação de uma abordagem colaborativa, livre de desconfianças, com um conjunto de boas práticas de engenharia de *software* que são eficientes independentemente do contexto.

Foi criado por Kent Beck, durante um projeto crítico na *Chrysler*. Cada uma dessas práticas contribui para o aumento da qualidade do produto entregue ao cliente. Entre estas práticas podem ser citadas: revisão de código, integração rápida, testes automatizados, *feedback* do cliente e design simples (BASSI, 2014).

O *Extreme Programming* assume que a variação dos requisitos pode ocorrer, e ao invés de tentar eliminá-la, trata esta mudança de maneira flexiva e colaborativa, colocando a equipe de desenvolvimento e o cliente no mesmo time com o propósito de criar *software* com alto valor agregado.

2.7.1 Histórias de usuário

Ao contrário do que muitas pessoas acreditam, o desenvolvimento de *software* não tem início no momento que o programador começa a digitar o código. O desenvolvimento começa na análise e principalmente na especificação de requisitos.

Segundo Helm e Wildt (2014) “Se a especificação do *software* é ruim, o resultado do trabalho provavelmente será um software igualmente ruim”. É possível usar uma forma de especificação mais efetiva e eficiente, chamada Histórias de Usuário (*user stories*).

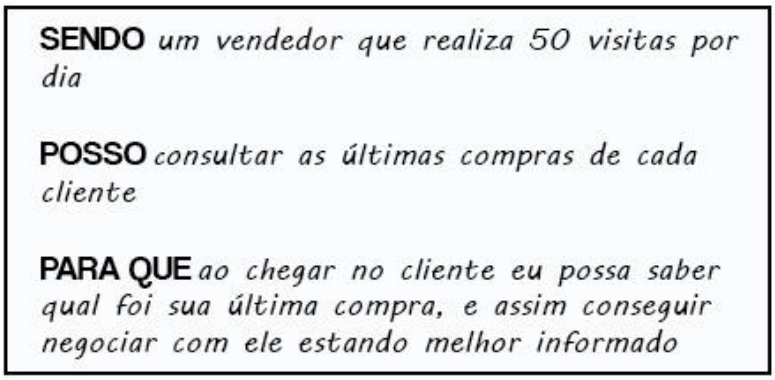

Figura 4. Exemplo de História de Usuário
Fonte: HELM; WILDT (2014)

Histórias de usuário são descrições curtas de funcionalidades que determinado público-alvo gostaria de ver no sistema. Devem fazer sentido para o negócio, e serem escritos de forma curta e objetiva, com uma linguagem simples para que tanto o time de desenvolvedores entenda o que está descrito, quanto o usuário final (RASMUSSON, 2010).

2.7.2 Testes automatizados

Quanto mais cedo os erros forem encontrados, mais barato será corrigi-los e menos impacto eles causarão no *software*.

Desenvolvedores escrevem testes automatizados para cada elemento do sistema e os executam sempre que o código é alterado para garantir que as modificações não ocasionem erros em funcionalidades antigas. Portanto, como o teste faz parte do processo de desenvolvimento, um código sem testes não é considerado concluído (BASSI, 2014).

2.8 Kanban

Kanban tem um foco diferente de metodologias ágeis como *Scrum* e XP. Enquanto *Scrum* foca no gerenciamento do projeto, e XP no desenvolvimento, *Kanban* busca ajudar uma equipe a melhorar sua maneira de visualizar o *software*.

Uma equipe que utiliza *Kanban* tem uma imagem clara das ações necessárias para o desenvolvimento, da sua interação com a equipe, onde estão perdendo tempo e como melhorar seu rendimento (STELLMAN; GREENE, 2014).

Equipes costumam posicionar quadros em áreas visíveis do ambiente de desenvolvimento, preenchendo-o com cartões que simbolizam as tarefas selecionadas para uma dada iteração. Geralmente, os cartões são posicionados conforme seu estado presente: não iniciado, em andamento e finalizado (VALE, 2014).

Figura 5. Quadro Visual Kanban
Fonte: VALE (2014)

Assim como todas as ferramentas, *Scrum*, XP e *Kanban* não são perfeitos ou completos. Não nos dizem tudo que devemos fazer, apenas nos fornecem um rumo para seguir. Juntos se completam, formando uma maneira poderosa de gerenciar projetos, tanto por meio das regras do *Scrum*, quanto pela facilidade de visualizar os acontecimentos provida pelo *Kanban.* Juntando essa capacidade de visualizar e gerenciar o andamento de projetos provida pelas metodologias citadas acima, à preocupação que o *Extreme Programming* tem com a qualidade do código que está sendo criado, temos uma metodologia global, capaz de gerir todos os aspectos de um projeto de *software*, desde sua especificação inicial, até a entrega do *software* funcionando ao fim do projeto.

3 PROGRAMAÇÃO WEB

A *Web* nasceu em um laboratório de física de partículas (CERN), em Genebra, Suíça, em 1989. Um especialista em computação chamado Tim Berners-Lee propôs pela primeira vez um sistema de gerenciamento de informações que usou um processo *hipertexto* para conectar os documentos relacionados em uma rede (DAN CONNOLLY, 2000).

A internet de hoje é diferente do que era em décadas passadas. Antes, a interação entre um usuário e um site era muito orientada no sentido do consumo. O servidor geraria páginas, e o usuário navegaria por elas. Hoje, ao invés de apenas consumir dados, os usuários estão consumindo dados de uma forma nunca antes imaginada, o foco agora está em colocar as pessoas no controle de suas experiências, usando os dados que elas criam para modificar, melhorar e aprimorar sua experiência em tempo real (WILSON, 2013).

No desenvolvimento do presente projeto, serão utilizados como base vários *frameworks* feitos com a linguagem JavaScript, CSS para o *design*, com um banco de dados NoSql, o MongoDB.

3.1 JavaScript

Apesar do nome, JavaScript não tem relação nenhuma com a linguagem de programação Java.

Sua criação deu-se em 1995 na *Netscape*, sendo inicialmente chamado de *LiveScript*. Atualmente, é utilizado tanto para aplicações *client-side* (navegador) quando *server-side* (servidor), e também em algumas aplicações *desktop* (MARDAN, 2014).

Iniciou sendo empregado como a linguagem que navegadores usavam para executar sua lógica no *client-side*. A partir dos anos 2000, desenvolvedores iniciaram o desenvolvimento de bibliotecas e ferramentas que diminuíam o ciclo de desenvolvimento, dando início a uma geração de aplicações *web* ainda mais avançadas. A maior revolução se iniciou no ano de 2008, quando a *Google* lançou o navegador *Google Chrome*, juntamente com uma nova e rápida *engine*, a JavaScript V8. A *engine* V8 da *Google* fez o JavaScript ficar tão rápido, que revolucionou o modo de se desenvolver aplicações *web*. E o mais importante, o lançamento da *engine* V8 como *open-source* permitiu que desenvolvedores trouxessem o JavaScript para fora do navegador (HAVIV, 2014).

Com a possibilidade de se usar JavaScript também no *server-side*, elimina-se a necessidade de combinar outras linguagens como PHP, Java ou Perl para desenvolver grandes

aplicações na *web.* A partir daí, surgiu um novo conceito, o MEAN: MongoDB, ExpressJS, AngularJS e NodeJS.

3.2 Desenvolvimento web com MEAN

Usando estas quatro ferramentas juntas, desenvolvedores podem criar aplicações eficientes, bem organizadas e interativas rapidamente.

Pode-se usar JavaScript no *server-side* (NodeJS e ExpressJS), no *client-side* (AngularJS), armazenar objetos no formato JSON (*JavaScript Object Notation*) em banco de dados MongoDB e usar objetos JSON para transferir dados facilmente da base de dados para o *server-side* e para o *client-side* (SEVILLEJA; LLOYD, 2015).

No decorrer deste projeto, veremos mais detalhadamente um pouco sobre cada ferramenta que foi utilizada no desenvolvimento, dando início pelas ferramentas que fazem parte da *stack* MEAN.

3.2.1 MongoDB

MongoDB é um sistema gerenciador de banco de dados feito para aplicações *web* e infraestrutura de internet.

Sua estratégia de modelagem foi construída para grandes quantidades de leituras e escritas com facilidade e grande escalabilidade. Foi criado em meados de 2007, em uma *startup* chamada *10gen* (BANKER, 2012).

É NoSQL (não-relacional), diminuindo sua complexidade, especialmente por não precisar lidar com *querys* complexas e repletas de *joins* entre tabelas, que acabam sobrecarregando a aplicação e causando lentidões. Hoje em dia, grandes empresas como *Google*, *Facebook*, *Yahoo!*, *Twitter*, além de muitas outras usam bancos de dados NoSQL (VAISH, 2013).

Uma base de dados relacional é muito estruturada, com múltiplas tabelas que armazenam partes individuais dos dados. MongoDB, por outro lado, armazena tudo em um único documento.

Funciona como um arquivo *JSON*, que oferece uma leitura fácil por humanos, e uma excelente performance por manter todos os dados que são relacionados no mesmo local. O formato de arquivo armazenado na base de dados foi criado pela própria equipe de desenvolvimento do MongoDB, e é chamado *BSON*, que é um *JSON* binário, tornando muito mais fácil para um computador processar e pesquisar por documentos (HOWS; MEMBREY; PLUGGE, 2014).

A figura abaixo exemplifica a facilidade de leitura de um arquivo *JSON*, mostrando-nos como é sua estrutura. Funciona como se constituísse um *array* multidimensional.

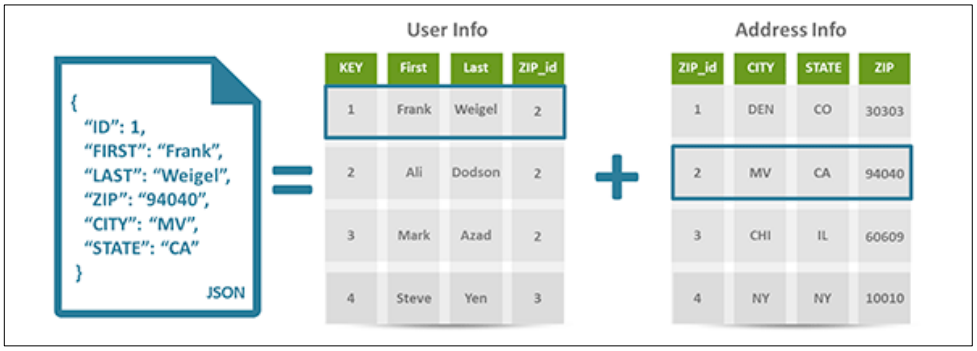

Figura 6. Diferença entre Tabelas SQL e Documentos NoSQL
Fonte: IBM (2014)

3.2.2 ExpressJS

ExpressJS é um *framework* baseado no núcleo do módulo HTTP e nos componentes de conexão do NodeJS. Suas principais funções são fazer análise de requisições HTTP, análise de *cookies*, gerenciamento de sessões, organizar rotas com uma cadeia de condições *if* baseadas em caminhos de URL e métodos HTTP dos pedidos, além da determinação de cabeçalhos de resposta adequadas com base nos tipos de dados recebidos. Também facilita a construção de aplicações com base na estrutura MVC (*Model View Controller*) (YAAPA, 2013).

3.2.3 AngularJS

Foi criado por Miško Hevery e Adam Abrons em 2009. É um *framework client-side* JavaScript e de código aberto. Tem como objetivo prover uma programação mais limpa, reutilizável e de fácil manutenção, por trabalhar com base na estrutura MVC. Atualmente AngularJS é mantido e desenvolvido pela *Google* (BRANAS, 2014).

Existem alguns conceitos principais no AngularJS. *Data Binding*, é um estilo de programação que elimina o código escrito entre o *view* e o *model*. A maior parte dos dados movidos de um para outro acontece automaticamente. *Dependency Injection* nos permite seguir um estilo de desenvolvimento em que, em vez de criar dependências, as classes podem pedir apenas o que elas precisam. *Directives* permitem estender a sintaxe HTML, por meio de uma DOM incluída no núcleo do AngularJS (GREEN; SESHADRI, 2013).

3.2.4 NodeJS

O NodeJS foi criado no final de 2009, por uma equipe liderada por Ryan Dahl, tendo como base a *engine* V8 do *Google Chrome*. É uma plataforma altamente escalável e de baixo

nível, pois é programada diretamente com protocolos de rede e internet, possuindo bibliotecas que permitem acessar recursos de sistemas operacionais, principalmente os baseados em Unix. Esta tecnologia possui um modelo inovador, sua arquitetura é totalmente *non-blocking thread* (não-bloqueante), apresentando uma boa performance e podendo fazer uso de todo o poder de processamento dos servidores. Diferentemente de outras plataformas, como .NET, Java, PHP, Ruby ou Python, que enfileiram requisições para processá-las uma a uma, causando gargalos com o aumento de acessos simultâneos ao sistema (PEREIRA, 2013).

```
1  var http = require('http');
2
3  http.createServer(function(request, response){
4      response.writeHead(200, {'Content-Type': 'text/html'});
5      response.end('<h2>Servidor online!</h2>');
6  }).listen(3000);
7
8  console.log('Servidor iniciado em localhost:3000.');
```

Figura 7. Exemplo de criação de servidor NodeJS
Fonte: *Print screen* de arquivo no editor *Sublime Text* 3.

Aplicações NodeJS são *single-thread*, ou seja, cada aplicação terá instância de um único processo, permitindo explorar ao máximo a programação assíncrona. Segue a mesma filosofia de orientação a eventos do JavaScript *client-side*, tendo como única diferença o fato de não existirem eventos como o *click* do mouse, ou o *keyup* do teclado. NodeJS trabalha com eventos de I/O do servidor, como o evento *connect* de um banco de dados ou um evento *data* de um *streaming* de dados, por exemplo. O *Event-Loop* é o agente responsável por escutar e emitir eventos no sistema (TEIXEIRA, 2013).

Foi a principal plataforma usada durante o desenvolvimento do projeto, pelas vantagens apresentadas em relação a linguagens mais tradicionais, como o PHP. Uma dessas vantagens é a possibilidade de se trabalhar com sistemas *real-time*, por meio do novo protocolo *WebSockets*. O projeto desenvolvido tem uma grande demanda por requisições no banco de dados, e obteve um melhor desempenho sendo programado de modo não-bloqueante.

3.3 Socket.io

Foi criado em 2010 por Guillermo Rauch, tem como objetivo o desenvolvimento de aplicações em tempo real utilizando NodeJS. Funciona utilizando os protocolos HTTP e XHR, primeiro abrindo uma comunicação *long-polling* XHR e, em seguida, tentando atualizar a conexão em um canal de *WebSockets* (HAVIV, 2014).

3.4 HTML5 com EJS

HTML é a sigla em inglês para *Hyper Text Markup Language*, que em português, significa linguagem para marcação de hipertexto. Foi criado por Tim Berners-Lee e atualmente está na versão HTML5 (SILVA, 2011).

EJS é um pré-processador de código HTML, que funciona da mesma maneira que o PHP, exceto pelo fato de não ser executado do lado do servidor. EJS é executado dentro do navegador *web*. Isso permite que o código do lado do cliente processe modelos de interface baseados em *JavaScript* para inserção na página atual sem consultar o servidor (HAVIV, 2014).

3.5 CSS3

CSS é a abreviação para o termo em inglês *Cascading Style Shee*t, traduzido para o português como folhas de estilo em cascata. Cabem ao CSS todas as funções de apresentação de uma aplicação web, como cores, fontes e todos os elementos visuais (SILVA, 2012).

3.6 Design Responsivo com Bootstrap

O número de usuários que utilizam dispositivos móveis para acessar a *internet* vem aumentando exponencialmente nos últimos anos. Com isso surgiu a importância de se otimizar o código para que funcione de forma agradável em todas as resoluções.

Basicamente, o *Design* Responsivo permite que uma *interface* se adapte a diferentes tamanhos de telas, pequenas ou grandes. Isso é possível fazendo o uso das *media-queries* do CSS3 (FIRDAUS, 2013).

Neste projeto foi usado um *framework* de código aberto chamado *Bootstrap*, que tem por objetivo facilitar o desenvolvimento de aplicações *web* responsivas. Foi criado por Mark Otto e Jacob Thornto em 2011, enquanto trabalhavam no *Twitter*. Funciona com um *grid* de 12 colunas, que se ajustam dependendo da resolução em que a *interface* está sendo visualizada (SPURLOCK, 2013).

Uma das possíveis desvantagens deste *framework* é a semelhança que as aplicações que a usam obtém. Porém, este problema é facilmente resolvido com a personalização via CSS.

3.7 Padrões de Projeto com MVC

Este projeto fez referência frequente ao padrão de projetos MVC, tanto no lado do servidor quanto no lado do cliente, por ser uma forma eficiente de organizar a arquitetura da aplicação, facilitando uma manutenção posterior. As tecnologias utilizadas facilitam a

utilização do MVC, por serem criadas buscando a otimização e melhor organização da arquitetura do projeto.

Foi criado para uso na plataforma *Smalltalk* na década de 1970. Promove o desacoplamento do sistema em três componentes: *Model*, que contém os dados a serem lidos ou utilizados, *View*, que é a *interface* pela qual o usuário interage com o *Model* e *Controller*, que delega as ações solicitadas pelo usuário através da *View* para o *Model* subjacente (WILSON, 2013).

3.8 Ferramentas do Projeto

Na sequência do trabalho, serão descritas as principais ferramentas a serem utilizadas para auxiliar no desenvolvimento do projeto.

3.8.1 Atom

Atom é um editor de código fonte *open-source*, criado em 2015 e mantido pela equipe do *Git-Hub*. É construído tendo como base tecnologias *web*, como NodeJS e HTML. Sendo assim, é altamente personalizável e possui uma grande gama de *plug-ins* e ferramentas que estendem suas funcionalidades nativas (GITHUB, 2015).

É nativamente integrado com *Git*, o que facilita o *backup* de repositórios inteiros na nuvem, utilizando apenas uma linha de comando. Outra grande vantagem do Atom, é a sua velocidade se comparado com IDE's mais consagradas, como Netbeans e Eclipse

3.8.2 Controle de versão com git

Soluciona problemas como o de equipes trabalhando em paralelo no mesmo arquivo, evitando que quando algum membro salve o arquivo, as modificações feitas por outro membro sejam perdidas.

Controle de versão é um sistema que recorda mudanças em arquivos ou conjunto de arquivos ao longo do tempo para que possamos buscar versões especificas mais tarde. Git é um sistema de controle de versão distribuído com ênfase em velocidade, que foi inicialmente projetado e desenvolvido por Linus Torvalds para o desenvolvimento do *kernel Linux* (CHACON; STRAUB, 2014).

O presente projeto foi hospedado em um repositório *git* provido pela empresa *Bitbucket.* Tendo sido controlado todo o processo de desenvolvimento desta forma.

4 METODOLOGIA

4.1 Documentação

Em pesquisa bibliográfica, a documentação é o acervo de textos usados para a explicação, ou para a demonstração do problema escolhido pelo pesquisador. Este projeto foi iniciado com uma pesquisa prévia, em busca de materiais que pudessem contribuir para o esclarecimento do problema em questão, de forma criteriosa e seletiva (RUIZ, 2002).

4.2 Natureza da Pesquisa

O presente trabalho apresenta suas informações por meio de uma pesquisa de caráter exploratório e teórico.

O objetivo da pesquisa exploratória consiste em uma caracterização inicial do problema, de sua classificação e sua definição. Sendo assim, constitui o primeiro estágio de toda pesquisa científica, não tendo como objetivo resolver um problema, apenas defini-lo. Pesquisa teórica, tem por objetivo ampliar generalizações e leis, relacionando diferentes conceitos e gerando novas hipóteses (RUIZ, 2002).

A pesquisa utilizada pode ser definida como exploratória, pelo fato de ter sido um levantamento das informações mais relevantes sobre o tema abordado, buscando uma maior compreensão do que engloba o problema a ser resolvido. E pode também ser definida como teórica, pela sugestão da utilização de metodologias diferentes como base no gerenciamento de projetos, sendo essas metodologias *Scrum, Extreme Programming* e *Kanban.*

4.3 Tipo da Pesquisa

A pesquisa aqui se define como bibliográfica e de campo, utilizada buscando justificar os objetivos e motivações para tal projeto.

Pesquisa bibliográfica abrange toda a bibliografia disponibilizada que esteja relacionada ao tema de estudo. Abrange desde publicações, livros, monografias, pesquisas, teses, etc. Pesquisa de campo, é aquela utilizada com o objetivo de conseguir informações e conhecimentos sobre um determinado problema, utilizando a observação de fatos e fenômenos e a coleta de dados referentes ao tema (MARCONI; LAKATOS, 2003).

4.4 Técnicas de Pesquisa

A observação ajuda o pesquisador a identificar e obter provas a respeito do objetivo que estão buscando alcançar, formando uma fonte rica para a construção de hipóteses. Observar

é prestar atenção em um fenômeno ou problema, captá-lo e retratá-lo tal como se manifesta (RUIZ, 2002).

4.5 Coleta de Dados

Para a coleta de informações sobre viabilidade, e tópicos importantes a serem abordados no presente projeto, foram aplicadas na pesquisa as técnicas de entrevista e de questionário. A entrevista, foi feita com Rafael Helm. Questionários foram aplicados com Daniel Wildt, Serge Normando Rehem, Alercio Bressano e Bruno Souza (JavaMan). Todos especialistas em desenvolvimento ágil e ligados diretamente com o cenário nacional e internacional de eventos sobre o assunto. Com esses especialistas tive a oportunidade de definir quais as melhores metodologias ágeis a serem usadas em um projeto como este, além de receber opiniões acerca da qualidade dos tópicos aqui tratados.

Questionários também foram realizados com Diego Simon, especialista em *marketing* digital e em lançamento de produtos *online*, com quem tirei dúvidas sobre a viabilidade comercial do *software* em questão.

Entrevista consiste em um diálogo com o objetivo de colher dados relevantes para a pesquisa em andamento. O questionário ao invés de o informante falar, o mesmo responde de forma escrita a uma série de questões elaboradas cuidadosamente e com clareza, proporcionando assim uma maior chance de se obter boas respostas (RUIZ, 2002).

4.6 Validação do Projeto

Buscando validar a ideia e a viabilidade do projeto para o público-alvo escolhido, foi realizada uma pesquisa *online*, através do sistema gratuito *Typeform.* A pesquisa foi respondida por 105 desenvolvedores de *software* de todo o Brasil. Nesta pesquisa foram feitos alguns questionamentos, sendo os dois primeiros tópicos de múltipla escolha:

a) O nível de conhecimento de cada desenvolvedor em relação a Métodos ágeis em uma escala de um até cinco;
b) Com quais metodologias ágeis já haviam trabalhado;
c) Se utilizavam algum *software* para auxiliar no gerenciamento de projetos e qual era esta ferramenta;
d) Sugestões de funcionalidades para *softwares* de gerenciamento de projeto;

Os resultados da primeira pergunta, podem ser visualizados na figura 10.

Figura 8. Nível de conhecimento dos entrevistados
Fonte: *Print screen* do sistema *Typeform*.

Foi obtida a média de 3.74 em relação ao conhecimento que os entrevistados acreditavam ter sobre o assunto. Com relação as metodologias utilizadas pelos entrevistados nos projetos em que trabalham ou já trabalharam, o resultado é reproduzido na imagem abaixo. É importante ressaltar que nesta questão os entrevistados podiam selecionar mais do que uma metodologia, e que as três metodologias mais utilizadas são as mesmas abordadas com mais ênfase neste trabalho. Outra observação importante, é que os 3% que selecionaram a opção "Outro", utilizam o método FDD (*Feature Driven Development*).

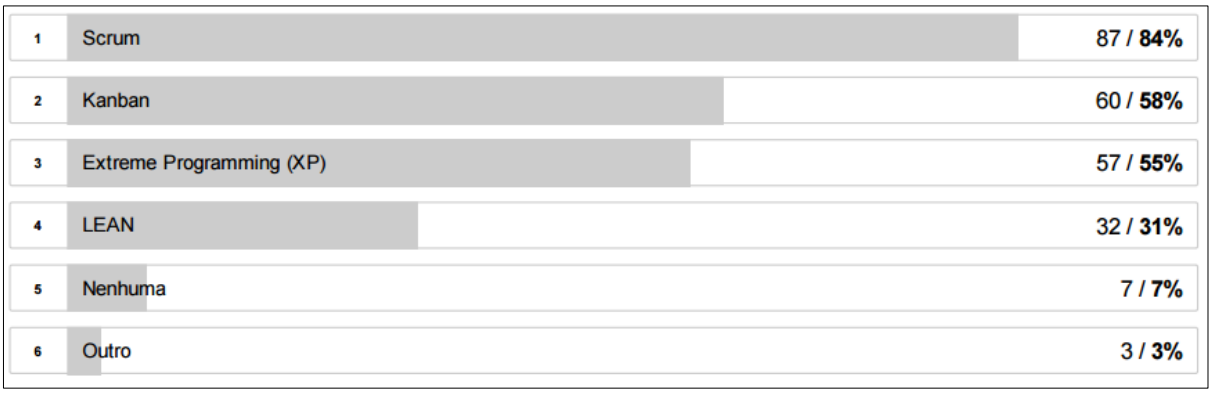

Figura 9. Metodologias utilizadas
Fonte: *Print screen* do sistema *Typeform*.

As respostas para a terceira pergunta foram variadas, totalizando uma quantia de 52 ferramentas citadas. A grande maioria foi citada apenas uma ou duas vezes. Também vale observar que cada entrevistado podia citar a quantidade de ferramentas que desejasse.

Estas quatro ferramentas foram estudadas mais a fundo e na sequência do trabalho pode ser visualizada uma descrição resumida de seu funcionamento, pontos positivos e negativos. As quatro ferramentas que se destacaram pela quantidade de vezes em que foram mencionadas, podem ser vistas na imagem apresentada na sequência do trabalho.

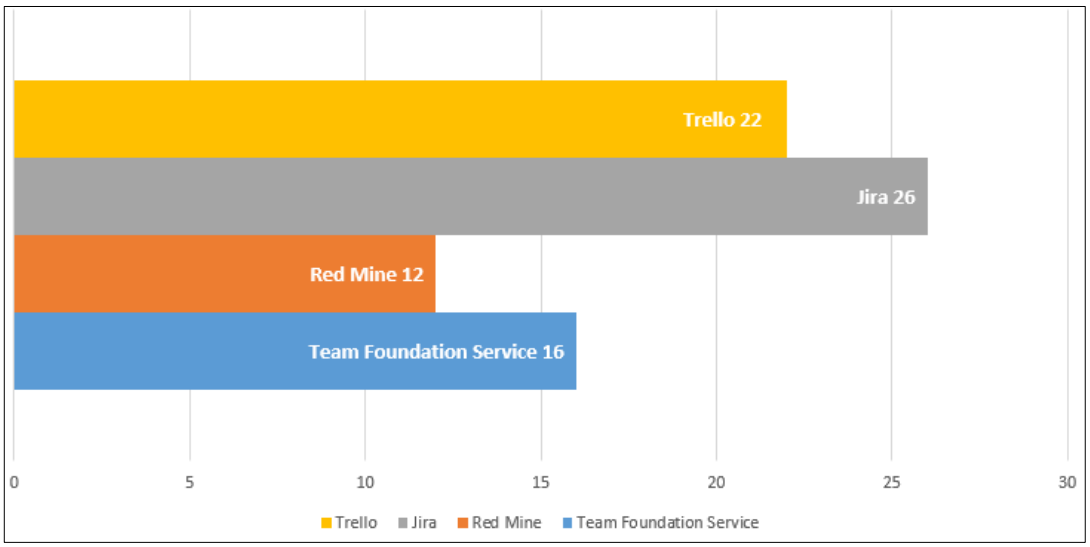

Figura 10. Ferramentas mais citadas
Fonte: Próprio Autor.

Trello é um *software online* simples para gerenciamento de tarefas. Possui as funcionalidades mais básicas, como adicionar membros, selecionar a data de entrega e anexar arquivos. É útil para pequenos projetos de maneira geral, o que o torna insuficiente quando se trata de projetos de *software*, por serem necessárias funcionalidades mais específicas.

Jira fornece acompanhamento de projetos e *bugs* para equipes de desenvolvimento de *software*. É uma das ferramentas de seu ramo mais utilizada ao redor do mundo, e possui uma série de *plug-ins* para adicionar novas funcionalidades. Abrange todas as etapas de um projeto, porém sua usabilidade é um tanto complexa e o tempo para se ter habilidade com a ferramenta é bastante grande (LI, 2013).

Red Mine é *open-source* e oferece basicamente as mesmas funcionalidades que o *Jira*. Alguns problemas encontrados no seu uso são a *interface* de difícil compreensão e dependência de um servidor próprio para hospedar a aplicação (LESYUK, 2013).

Team Foundation Service é uma plataforma baseada em nuvem que fornece ferramentas para o planejamento ágil e acompanhamento de trabalho. Possui um repositório de código que pode ser usado a partir do *Visual Studio*, Java e Mac. É uma ferramenta completa, porém sua utilização requer uma grande curva de aprendizado. Outro fato a se destacar é o alto custo da ferramenta, que em seu plano mínimo custa 20 dólares mensais por membro da equipe (OLAUSSON, 2013).

A última questão da pesquisa, na qual os entrevistados tinham espaço para dizer quais funcionalidades eram importantes para eles em uma ferramenta de gerenciamento de projetos, as respostas foram variadas, mas em resumo as principais foram: Gestão visual do andamento do projeto, delegação de tarefas, gestão do *Backlog*, facilidade de comunicação tanto entre

membros quanto com o cliente, métricas de evolução individual e da equipe, repositório de requisitos, estatísticas do projeto, gráfico *burndown*, notificações, *bug tracker*, facilidade de personalização e configuração, cronograma, gerenciamento de tempo, *gantt* e base de conhecimento.

Com base nas respostas obtidas serão selecionadas as principais funcionalidades do sistema, para que atenda as reais necessidades do usuário sem que haja desperdício com recursos que não resolverão o problema proposto neste trabalho. A partir do estudo de pontos fortes e fracos das ferramentas já existentes no mercado, e mais usadas atualmente, serão levantados quais os aprimoramentos necessários para a criação do *uSpeed.me.*

5 PROJETO

5.1 Hardware

Pelo fato de todas as tecnologias que foram utilizadas serem multiplataforma, o desenvolvimento ocorre em dois estágios.

O primeiro estágio do desenvolvimento foi a concepção do projeto, os primeiros passos. Essa fase foi desenvolvida e hospedada em uma máquina pessoal, com sistema operacional *Ubuntu* 15.03, processador *Intel Core* I5, 8GB de memória RAM e capacidade de armazenamento de 1TB. Tendo em vista que a aplicação não está sendo utilizada por múltiplos usuários, sua execução irá ocupar apenas uma parcela mínima do *hardware* utilizado.

A partir do momento em que o *software* for aprovado pela banca e puderem ser iniciados os testes com usuários, será migrada para um servidor terceirizado na nuvem. A empresa responsável pela terceirização será a *Digital Ocean*, situada em Nova Iorque. Será executado em um servidor com sistema operacional *Ubuntu* 14.04, processador *Intel Core* i7-980X (12M *Cache*, 3.33 GHz, 6.40 GT/s *Intel* QPI), 1GB de memória RAM ECC dedicada e 30GB de armazenamento RAID SSD.

5.2 Qualidade do código e padrões de projeto

Com o intuito de aplicar todos os conhecimentos adquiridos sobre a qualidade do código durante os estudos sobre métodos ágeis, alguns conceitos e cuidados foram aplicados durante o desenvolvimento da aplicação. Tais conceitos serão listados na sequência.

5.2.1 Padrões no código

Buscando atingir um padrão global de qualidade no código, o mesmo foi escrito utilizando os mesmos padrões utilizados pela comunidade *NodeJS* ao redor do mundo. Entre os padrões utilizados, destacam-se:

a) **Comentários:** Os comentários foram feitos em todos os métodos e funções, explicando seu funcionamento e os parâmetros utilizados. Foram feitos no idioma Inglês, por ser o idioma adotado na comunidade *open-source*.

b) **Nomenclatura:** Funções, métodos e variáveis seguiram o padrão utilizando letra minúscula no início da primeira palavra, e maiúscula no início das palavras seguintes. Também no idioma Inglês.

c) **Banco de dados:** A nomenclatura de modelos e atributos no banco de dados seguiram o mesmo padrão utilizado em variáveis. Também sendo utilizado o idioma Inglês, buscando ter consistência entre os módulos do projeto.

Tais padrões poderão ser observados nos códigos disponibilizados em anexo ao trabalho.

5.2.2 Testes unitários

Buscando uma melhor qualidade no código e na aplicação, foram utilizados testes unitários no *server-side* utilizando o *framework Mocha*. Estes testes têm como intuito garantir que todo o código está funcionando da forma devida.

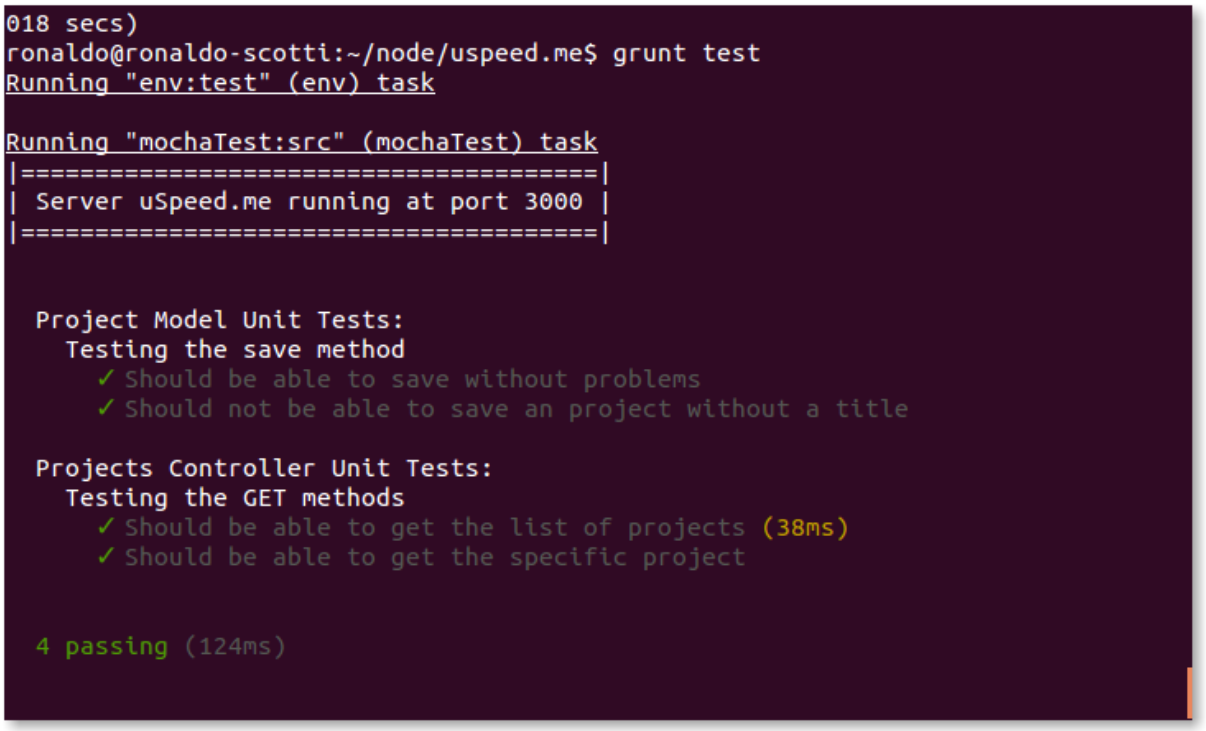

Figura 11. Testes unitários no server-side
Fonte: Próprio autor.

Na figura acima podemos visualizar a forma clara e objetiva com que foram feitos os métodos utilizados nos testes. Os testes são executados rodando o comando *grunt test* dentro do *prompt* de comandos do servidor.

5.3 Telas do Sistema

Atendendo as sugestões feitas pelos entrevistados, o princípio seguido em todas as páginas do sistema foi o de diminuir o número de passos necessários para o usuário realizar

determinada função, podendo assim criar uma melhor experiência e uma baixa curva de aprendizado para o uso do sistema.

Na sequência, serão apresentadas algumas das principais telas do sistema, buscando exemplificar de forma visual como ocorrerá o seu funcionamento, e qual será a experiência do usuário.

5.3.1 Cadastro de usuário

A fase inicial do processo de utilização do sistema por um novo usuário é o cadastro do mesmo. Este processo pode ser visualizado na figura abaixo.

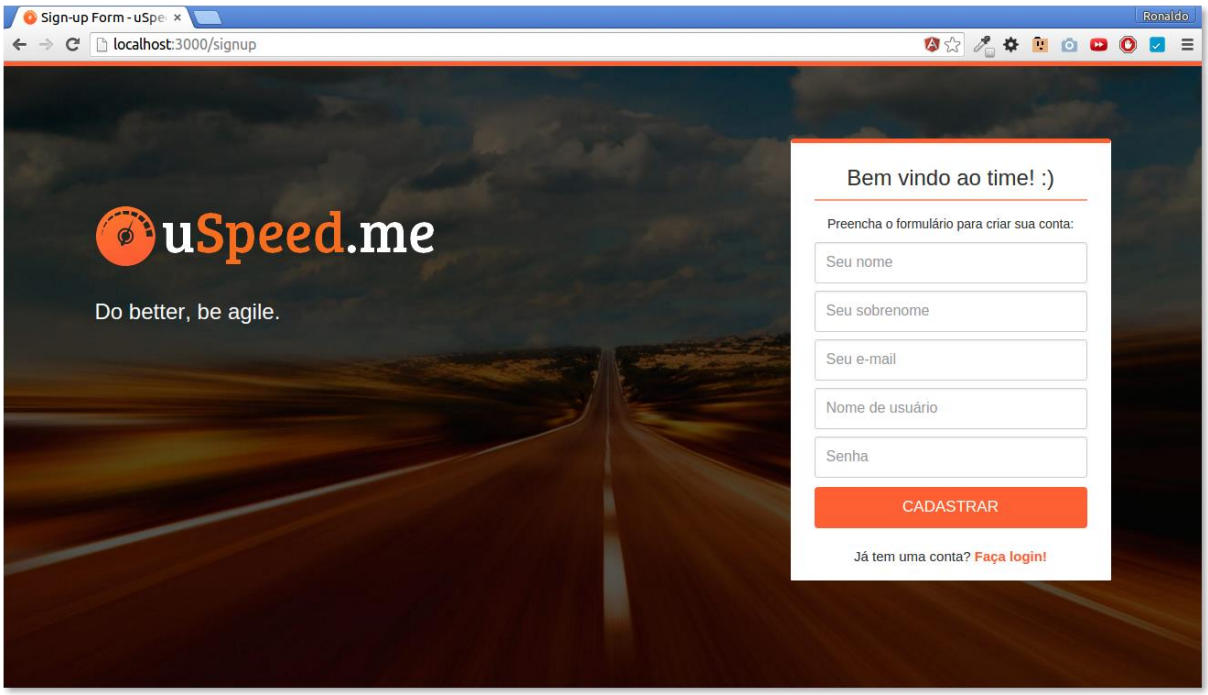

Figura 12. Tela de cadastro
Fonte: Próprio autor

Após preencher os dados, os mesmos são salvos no banco de dados criptografando sua senha. Dessa forma, os dados do usuário estão seguros, não podendo ser acessados nem mesmo pelo desenvolvedor do sistema ou por pessoas que tenham acesso ao banco de dados.

5.3.2 Login

O processo de entrada no sistema é feito através da tela de *login*, que poderá ser visualizada na figura seguinte. Tentativas de acesso direto ao sistema por meio de URL, são redirecionadas a esta mesma tela, caso o usuário não esteja autenticado.

Figura 13. Tela de *login*
Fonte: Próprio autor

Após o envio do formulário, o sistema irá criptografar a senha, e comparar com os dados existentes no banco de dados e efetuar a autenticação da sessão, armazenando-a no banco de dados e autenticando o usuário.

5.3.3 Dashboard

Esta é a tela inicial do sistema. O resultado pode ser visto na próxima figura.

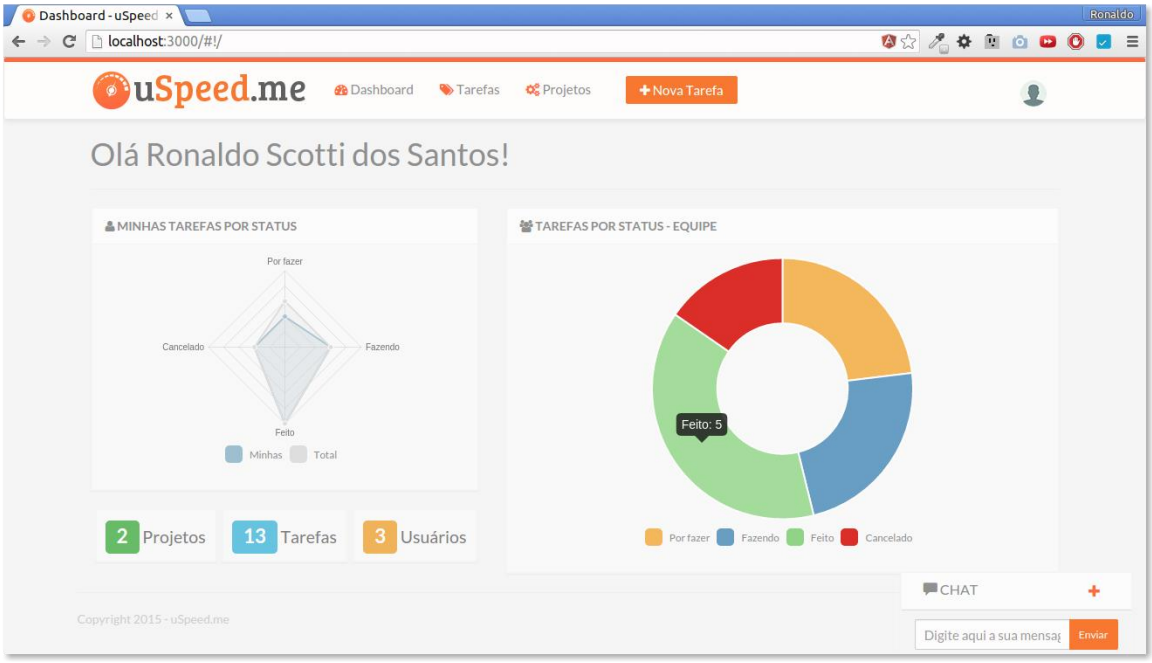

Figura 14. Tela de *dashboard*
Fonte: Próprio autor.

Esta tela traz resumos importantes em forma gráfica e numérica. Como podemos observar, são trazidas as quantidades de projetos, tarefas e usuários cadastrados no sistema. Também é gerado um gráfico *Doughnut* trazendo uma segmentação visual de tarefas por status.

Foi utilizado um gráfico *Radar* que compara informações entre as tarefas pessoais e o total de tarefas do sistema, fazendo a segmentação por *status*. Para criação destes gráficos, foi utilizada a biblioteca JavaScript *Charts.js*.

5.3.4 Criação e edição de projetos

O item inicial a ser cadastrado pelo usuário no sistema, é o projeto. A tela que contém o formulário de criação e edição de projetos, é exibida abaixo.

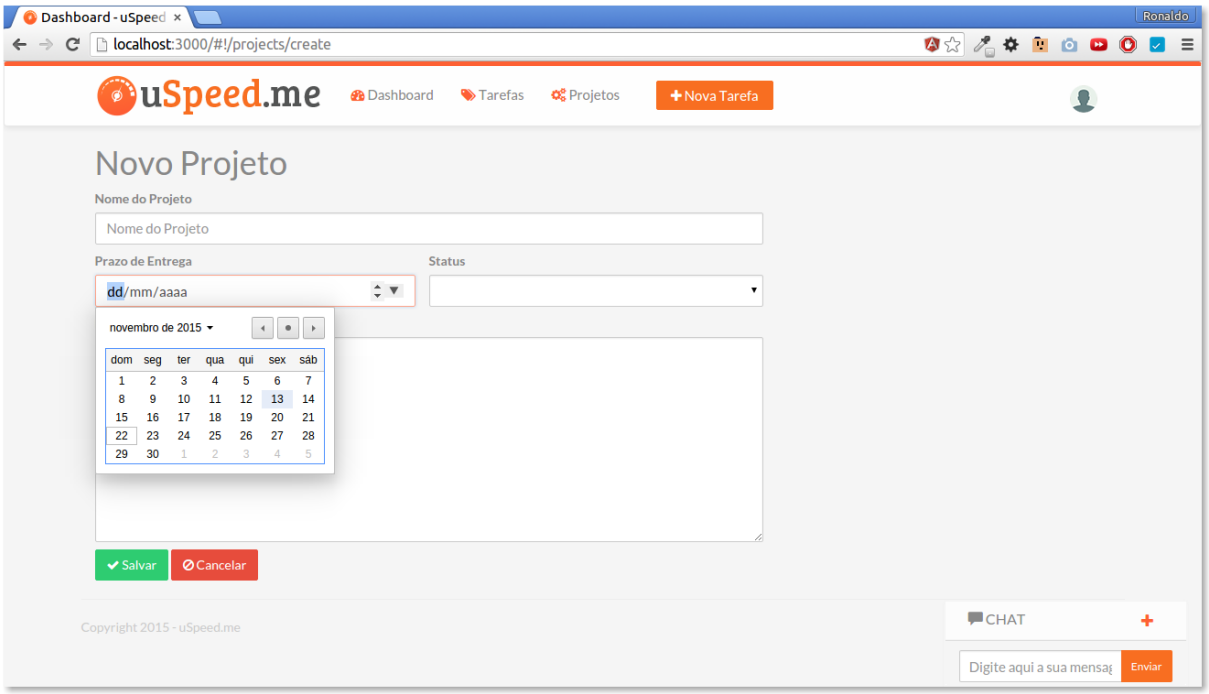

Figura 15. Tela de criação e edição de projetos
Fonte: Próprio autor

Um formulário é exibido na tela para a inserção dos dados referentes ao projeto. Campos a se destacar são o prazo de entrega, que quando clicado exibe um calendário para proporcionar uma maior facilidade ao usuário, e o *status*, que exibe um *select* com 3 opções, em andamento, finalizado e cancelado.

No momento em que o usuário acessar a URL de edição do projeto, os dados sobre o mesmo serão carregados do banco de dados, e exibidos como pré-definidos no formulário. O mesmo acontecerá no formulário de edição de uma tarefa, que será exibido posteriormente neste mesmo trabalho.

5.3.5 Listagem de projetos

Para os projetos, uma listagem simples e objetiva foi desenvolvida. O resultado pode ser visto na figura abaixo.

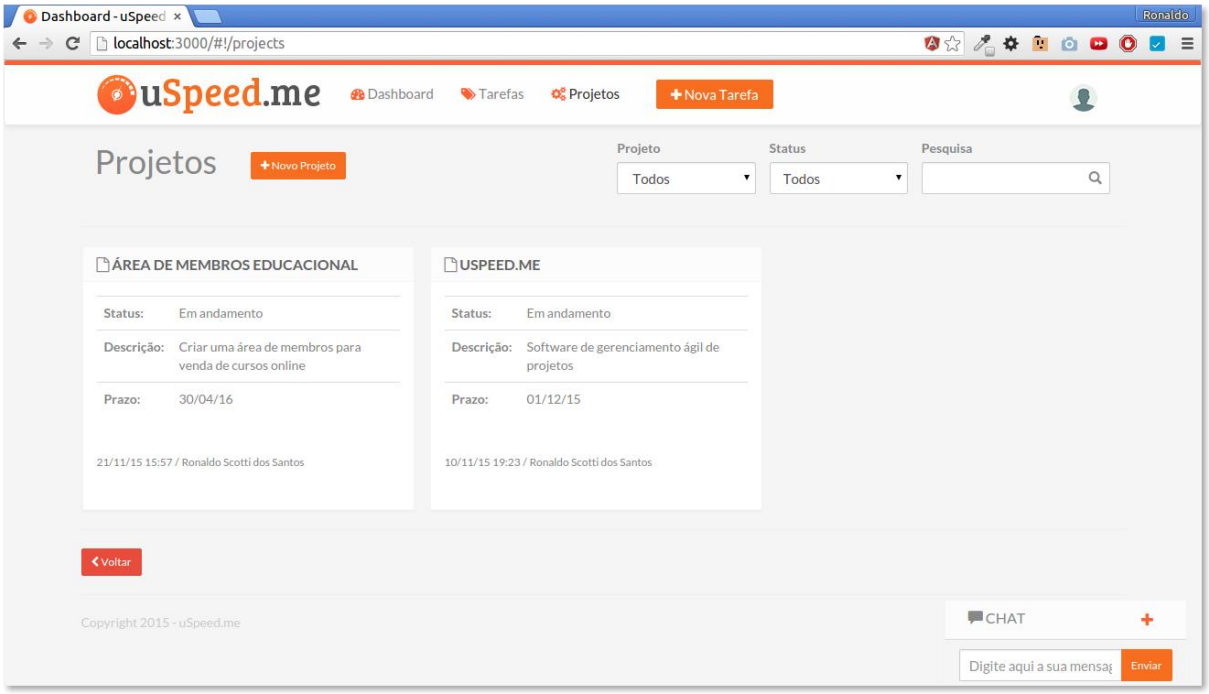

Figura 16. Tela de listagem de projetos
Fonte: Próprio autor.

Os projetos são listados em forma de blocos, exibindo as principais informações sobre o mesmo em uma tabela. Há a possibilidade de o usuário utilizar filtros para personalizar a sua visualização, sendo esses filtros: a escolha de um projeto diretamente, a escolha de um *status* de andamento específico, ou a pesquisa direta por meio de um formulário.

Outro fator observado, foi a navegabilidade do sistema. Telas internas possuem a opção de voltar sendo exibida no final do conteúdo, e o menu está sempre disponível para a navegação entre páginas.

5.3.6 Visualização de um projeto

Além de criar e editar um único projeto, o usuário pode também sentir a necessidade de apenas o visualizar. A exibição das tarefas que fazem parte daquele projeto também é um fator importante. O projeto é buscado no banco de dados, através de seu *_id* que é passado como parâmetro na URL. Para esse fim, foi criada a tela exibida na figura abaixo.

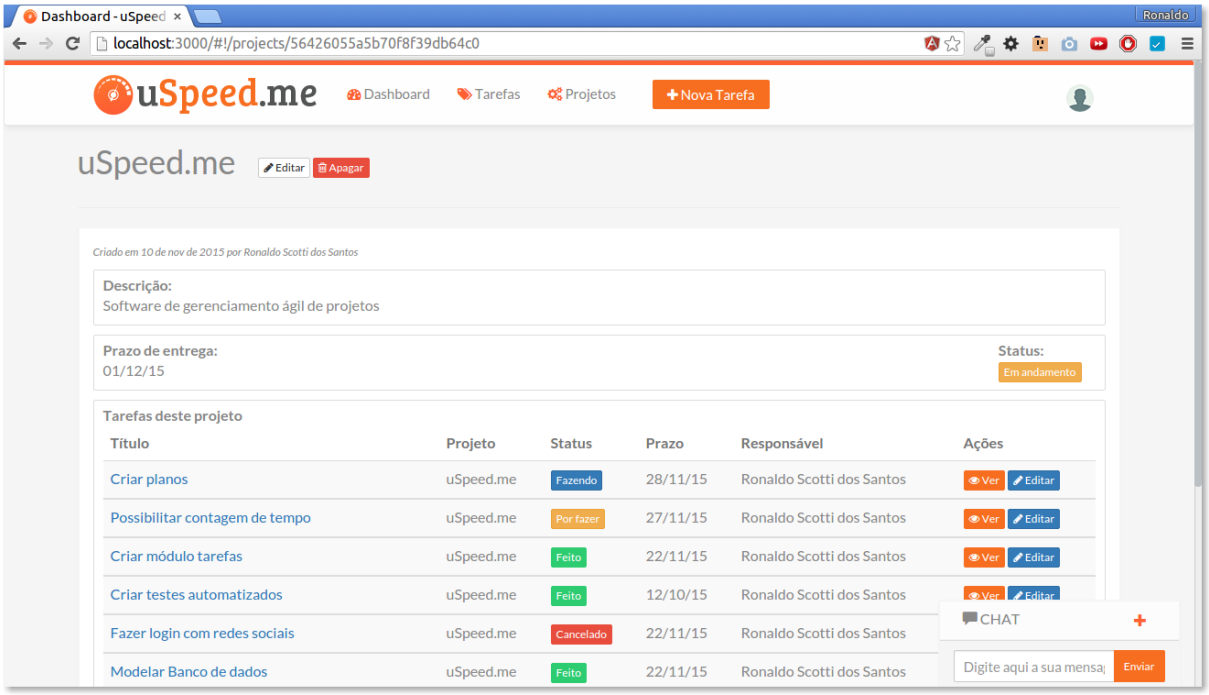

Figura 17. Exibição de um projeto
Fonte: Próprio autor

Nesta tela o usuário tem também a possibilidade de acessar a tela de edição ou excluir o projeto, caso tenha sido ele o criador.

5.3.7 Criação e edição de tarefas

A tela utilizada para o cadastro e edição de tarefas, é exibida na figura seguinte.

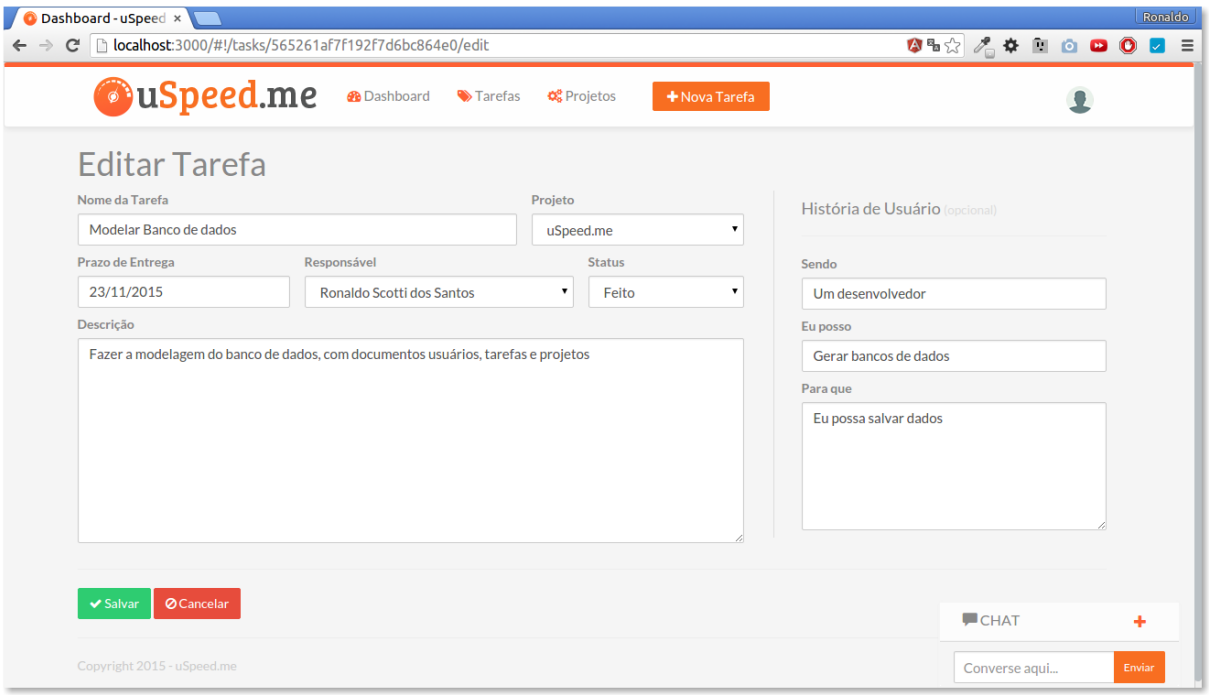

Figura 18. Tela de cadastro e edição de tarefas
Fonte: Próprio autor

Nesta visualização, foi escolhida a edição de uma tarefa já existente para descrever seu funcionamento. Além do *_id* que é passado na URL, é passado também o parâmetro *edit*, que faz com que o sistema renderize a página de edição. O usuário tem a possibilidade de adicionar a tarefa a um projeto, escolher um usuário responsável por sua execução e determinar um *status* inicial para a tarefa, podendo ser: por fazer, fazendo, feito ou cancelado. Além também de preencher as outras informações, como prazo de entrega e história de usuário.

5.3.8 Listagem de tarefas

Seguindo na apresentação do sistema, temos a página de listagem de tarefas. Esta é uma página importante, pois será uma das mais acessadas pelos usuários. O usuário tem duas possibilidades de visualização: a listagem comum, em uma tabela, e a exibição de um quadro Kanban.

A exibição em listagem será a primeira a ser apresentada, na figura seguinte.

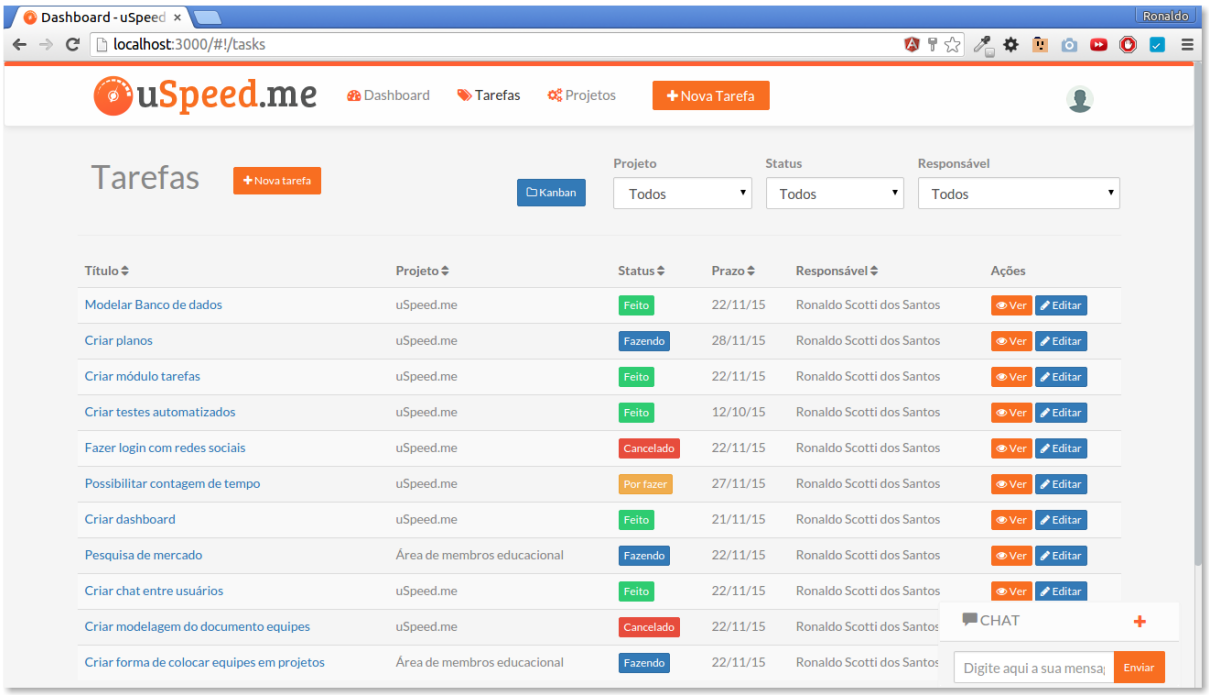

Figura 19. Listagem de tarefas - Modo tabela
Fonte: Próprio autor

Nesta tela, o usuário tem uma visão geral, tendo *links* diretos para a edição e visualização de uma tarefa.

Outro cuidado tomado na exibição das tarefas foi a criação de filtros de listagem. Em um momento em que existam muitas tarefas criadas pelo sistema, a visualização poderia ser confusa e dificultada pela grande quantidade de informações em uma única tela. Na próxima figura podem ser visualizados os filtros sendo aplicados a listagem.

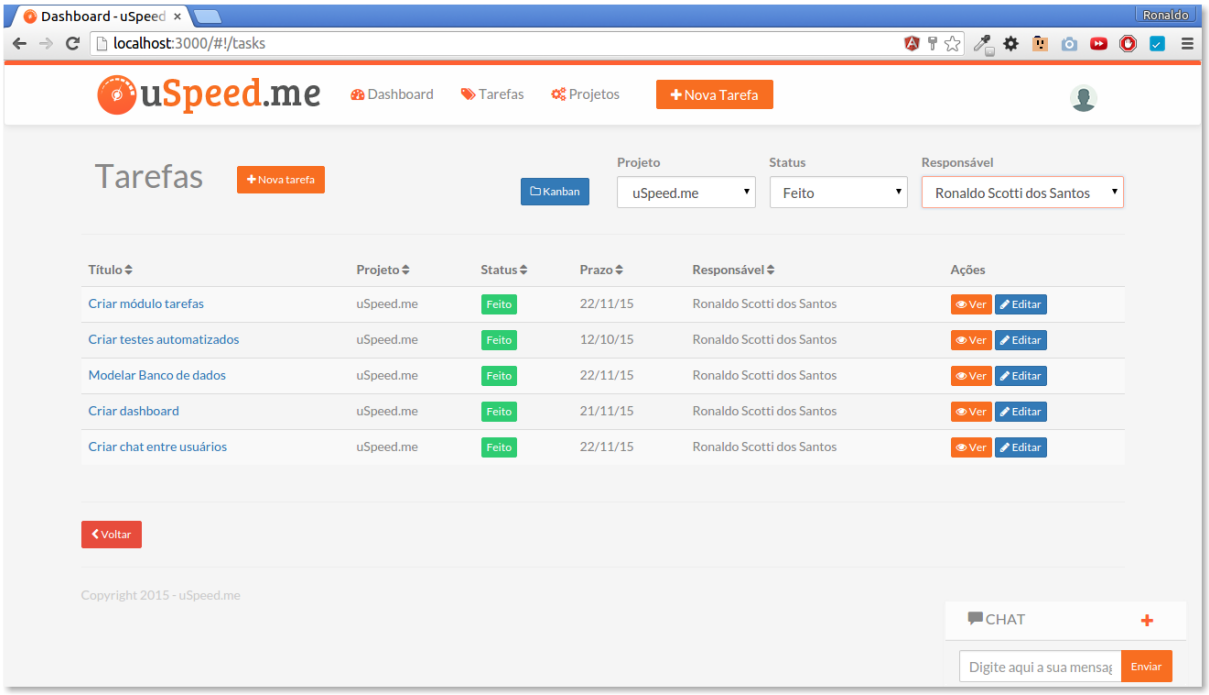

Figura 20. Listagem de tarefas com filtro aplicado
Fonte: Próprio autor

Como pôde ser visualizado, estão sendo exibidas apenas as tarefas do projeto *uSpeed.me* com *status* igual a feito e designadas a Ronaldo Scotti dos Santos.

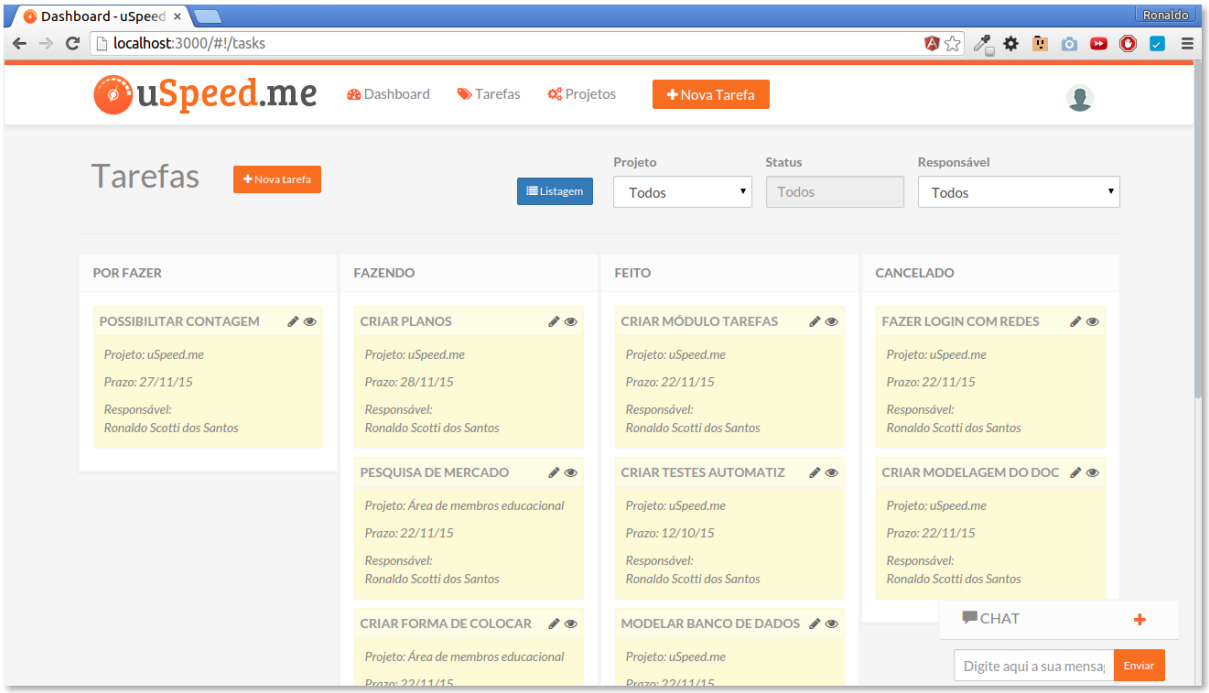

Figura 21. Quadro Kanban
Fonte: Próprio autor

Outra funcionalidade adicionada à esta tela, foi a possibilidade de se visualizar um quadro Kanban com as tarefas cadastradas. Esta funcionalidade é ativada ao clicar no botão Kanban, e pode ser visualizada na figura acima.

5.3.9 Chat

Uma funcionalidade adicional desenvolvida foi a criação de um *chat*, em que todos os usuários podem conversar e trocar informações sobre o projeto. O *chat* foi desenvolvido utilizando o método *handshake*, através dos protocolos HTTP e XHR, com o módulo *Socket.io* do NodeJS.

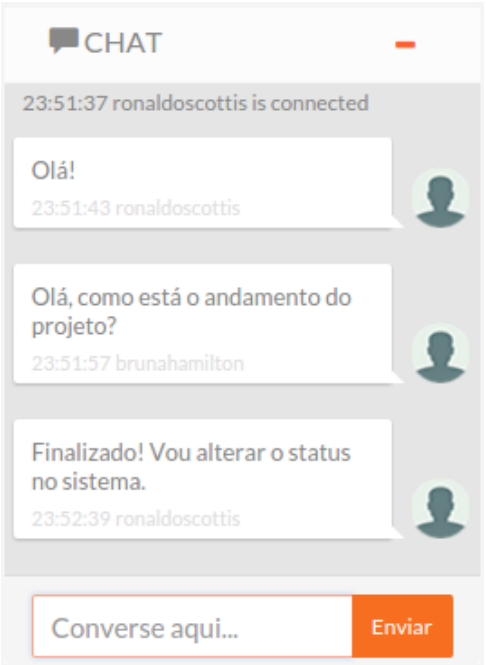

Figura 22. Chat
Fonte: Próprio autor

A figura acima apresenta o elemento em que é exibida a conversa do *chat.* Este elemento está presente em todas as páginas internas do sistema, como pôde ser visto nas figuras anteriores. Tem a possibilidade de ser minimizado para não comprometer a visualização de outras partes do sistema.

5.4 Interface Responsiva

Conforme citado no referencial teórico, toda a *interface* do sistema foi desenvolvida de tal forma que proporcione uma boa experiência aos usuários que acessem o sistema a partir qualquer dispositivo, independente da resolução do mesmo.

Além de oferecer esta vantagem aos usuários, este estilo de programação favorecerá o posicionamento do sistema nos mecanismos de busca. Pelo fato de, a partir de abril de 2015 o *Google* ter começado a priorizar sites responsivos em buscas realizadas a partir de *smartphones*, devido ao crescente aumento do uso de dispositivos móveis para navegar na internet (FOLHA DE SÃO PAULO, 2015).

Assim se proporciona uma melhor experiência e maior rendimento para quem estiver o utilizando. Mobilidade será um atributo de importância e prioridade alta dentro deste projeto. Poder utilizar o sistema e acompanhar o projeto em que está trabalhando em diversas plataformas é um dos principais diferenciais do *uSpeed.me*.

5.5 Modelagem Ágil

A Modelagem Ágil (MA) é uma metodologia baseada na prática para modelagem e documentação eficazes de sistemas. Tendo como objetivos colocar em prática um conjunto de valores e princípios relativos a uma modelagem eficaz e leve. Seus princípios são derivados dos princípios do Manifesto Ágil, buscando assim a melhoria contínua e maior qualidade no *software* final (AMBLER, 2004).

5.5.1 Diagrama de Sequência

É um diagrama que tem o objetivo de mostrar como as mensagens entre os objetos são trocadas no decorrer do tempo para a realização de uma operação (HAMILTON; MILES, 2006).

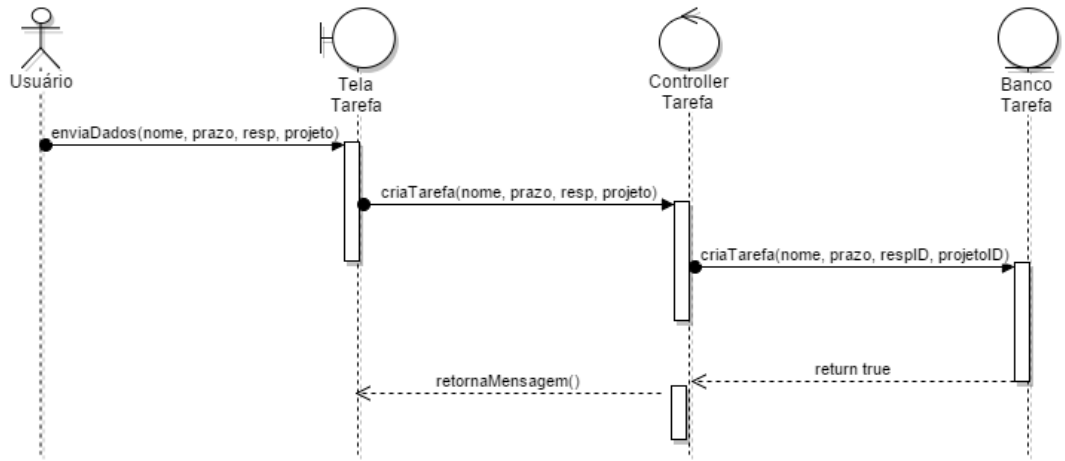

Figura 23. Diagrama de Sequência Criar Tarefa
Fonte: Próprio Autor.

A figura acima nos mostra o diagrama de sequência de uma criação de tarefa. Note que os dados foram reduzidos e as funções inseridas no idioma português para facilitar uma rápida visualização.

5.5.2 Diagrama de Componentes

Na figura abaixo, tem-se um diagrama de componentes que nos dá a noção de qual será o caminho percorrido pelos dados dentro da arquitetura do sistema.

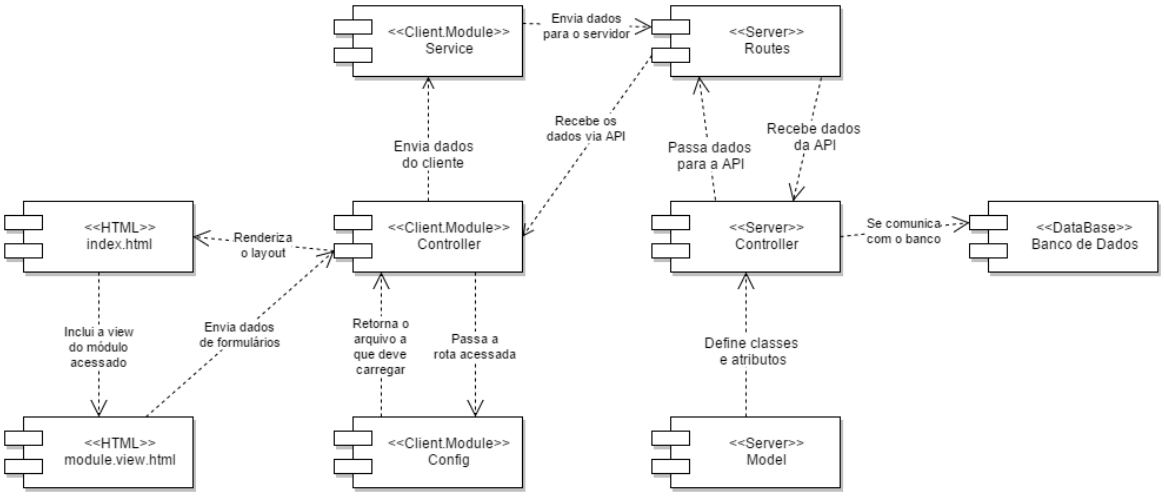

Figura 24. Diagrama de componentes
Fonte: Próprio Autor.

Um diagrama de componentes lustra como as classes deverão se encontrar organizadas através da noção de componentes de trabalho (AMBLER, 2004).

5.5.3 Diagrama de implementação

Na UML, os diagramas de implementação modelam a arquitetura física de um sistema. Os diagramas de implementação mostram os relacionamentos entre os componentes de software e hardware no sistema e a distribuição física do processamento (HAMILTON; MILES, 2006).

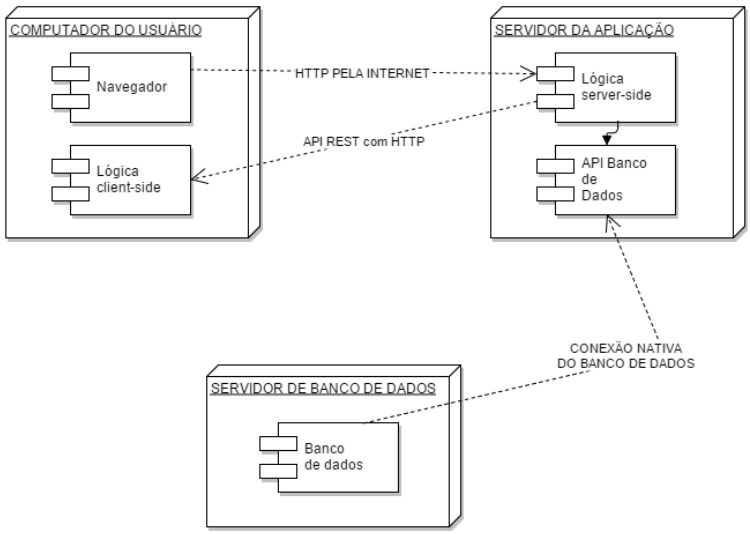

Figura 25. Diagrama de implementação
Fonte: Próprio autor

A figura acima nos mostra como funciona a arquitetura do sistema quando integrada ao *hardware*, que neste caso são os servidores da aplicação e de banco de dados.

5.5.4 Histórias de usuário do *uSpeed.me*

Como já citado no referencial teórico, histórias de usuário são descrições curtas de funcionalidades que um certo grupo de usuários gostaria de ver no sistema.

Sua utilização no sistema será constante e incentivada, pois dá uma visão ampla para o desenvolvedor do real problema a ser resolvido com aquela tarefa. Na sequência deste trabalho serão descritos dois cenários por meio de histórias de usuário.

O primeiro, descrito abaixo, faz referência a visão que um gerente de projetos terá ao visualizar os indicadores de desempenho de sua equipe:

SENDO um gerente de projetos que lidera uma equipe de desenvolvimento.

POSSO consultar o desempenho da equipe e de cada colaborador individualmente.

PARA QUE eu saiba se o projeto está sendo executado dentro do prazo, e possa valorizar o membro da equipe que esteja tendo um bom rendimento, assim como cobrar melhor desempenho de quem não estiver atendendo as expectativas do projeto.

O segundo cenário, descreve a percepção que um desenvolvedor terá ao ter a sua disposição uma lista de todas as tarefas que foram designadas a ele.

SENDO um desenvolvedor de *software* que faz parte de uma equipe ágil.

POSSO visualizar todas as tarefas designadas para mim com descrição completa e em ordem de prioridade.

PARA QUE eu tenha noção do que devo fazer a seguir, atendendo as expectativas do cliente, e assim não prejudique o rendimento da equipe e nem ocasione um atraso na entrega do projeto.

Histórias de usuário são um dos pilares da modelagem ágil e são também indispensáveis em qualquer projeto que faça o uso da metodologia XP, como será o caso do *uSpeed.me*.

5.6 Data Model Design

Em bancos de dados relacionais, a modelagem de dados normalmente progride como uma série de tabelas, consistindo de linhas e colunas, que definem coletivamente o esquema de seus dados.

Com NoSQL, se tem a noção de que seus dados não precisam ser tabulares, em MongoDB, os dados são armazenados em documentos. Isto significa que, enquanto em bancos

de dados relacionais é necessário que cada interseção linha-coluna contenha exatamente o mesmo valor, MongoDB permite armazenar uma matriz de valores distintos (COPELAND, 2013).

Assim sendo, o processo de *design* de um banco de dados NoSQL não ocorre da mesma forma que de um relacional, pois os dados podem variar de um documento para outro.

Como forma de exemplificação da forma de visualização de um banco de dados NoSQL, foi selecionado um *print-screen* do sistema *Robomongo*, utilizado como interface gráfica do *shell* utilizado pelo MongoDB.

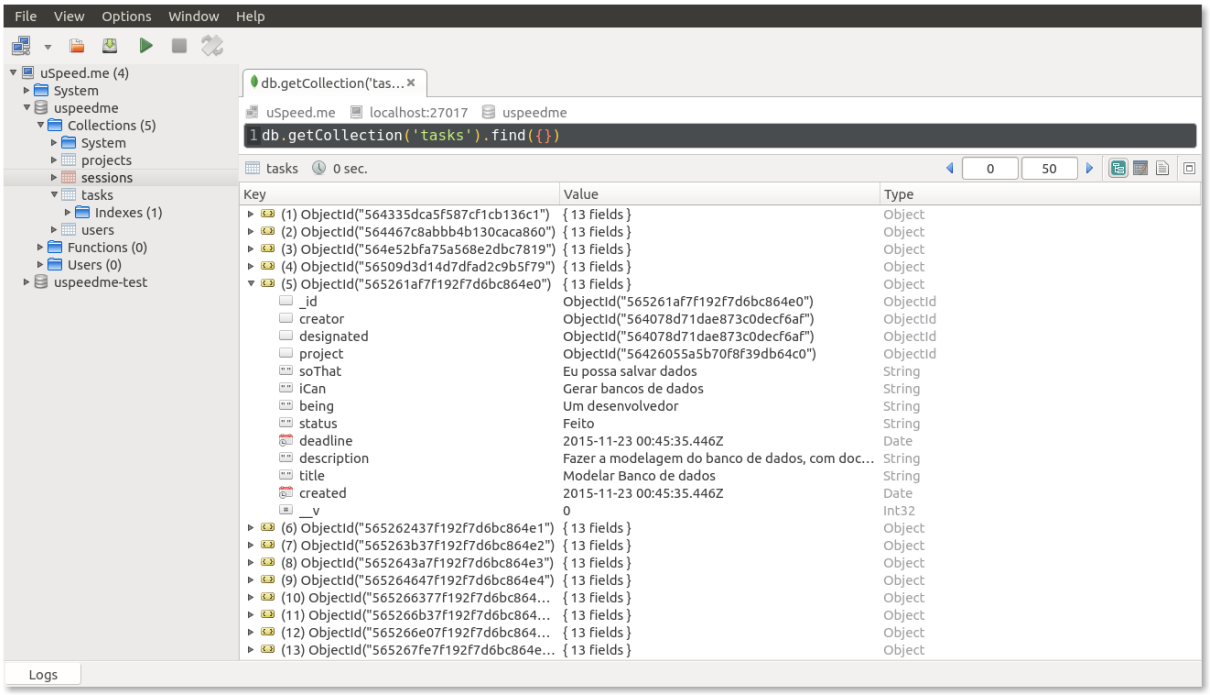

Figura 26. Documento tarefa
Fonte: Próprio autor.

Na figura acima pode ser vista a estrutura do documento tarefas. Sendo armazenados o _id de outros documentos no caso de criador, responsável e projeto, para que estejam vinculados a itens cadastrados fora do documento atual.

6 CONCLUSÃO

O objetivo a partir do momento da aprovação, é disponibilizar a ferramenta de forma gratuita na *web* para testes e indicações de melhorias. Após um período de testes e de melhorias contínuas, a ferramenta deve lançada no mercado com intuito comercial.

Dentro do conceito das *Startups*, a criação dessa primeira versão é chamada de MVP (*Minimum Viable Product*) e é altamente recomendada para que se obtenha uma direção correta e não se desperdice tempo e recursos com funcionalidades não utilizadas ou mal projetadas (GOTHELF, 2013).

Pode-se concluir a partir do planejamento e criação desse projeto, que os métodos ágeis vêm se tornando cada vez mais úteis e indispensáveis em um mundo competitivo como o que vivemos. E que ferramentas que nos ajudem a organizar as pendencias e manter o foco no desenvolvimento são de extrema importância.

REFERÊNCIAS

AMBLER, Scott W.. **Modelagem Ágil:** Práticas eficazes para a Programação Extrema e o Processo Unificado. Porto Alegre: Bookman, 2004.

BANKER, Kyle. **MongoDB in Action.** Shelter Island: Manning Publications, 2012.

BASSI, Dairton. Programação Extrema (XP). In: PRIKLADNICKI, Rafael; WILLI, Renato; MILANI, Fabiano (Org.). **Métodos Ágeis para Desenvolvimento de Software.** Porto Alegre: Bookman, 2014. Cap. 4. p. 37-57.

BRANAS, Rodrigo. **AngularJS Essentials.** Birmingham: Packt Publishing, 2014.

CASSEZ, Franck et al. **Automatic Synthesis of Robust and Optimal Controllers – An Industrial Case Study:** 12th International Conference, HSCC 2009, San Francisco, CA, USA, April 13-15, 2009. Proceedings. San Francisco: Springer Berlin Heidelberg, 2009. (Lecture Notes in Computer Science).

CHACON, Scott; STRAUB, Ben. **Pro Git**. 2. ed. Nova Yorque: Apress, 2014.

COCKBURN, Alistair. **Agile Software Development: The Cooperative Game.** 2. ed. Boston: Addison-wesley, 2007.

COLLINS. **Collins COBUILD Advanced Dictionary of American English.** 2. ed. Nova Iorque: Harpercollins Publishers, 2011.

COPELAND, Rick. **MongoDB Applied Design Patterns.** Sebastopol: O'reilly, 2013.

DAN CONNOLLY. W3c. **A Little History of the World Wide Web.** 2000. Disponível em: <http://www.w3.org/History.html>. Acesso em: 16 mar. 2015.

FIRDAUS, Thoriq. **Responsive Web Design by Example:** Beginner's Guide. Birmingham: Packt Publishing, 2013.

FOLHA DE SÃO PAULO. **Google passa a priorizar sites pensados para o celular e sofre críticas.** 2015. Disponível em: <http://www1.folha.uol.com.br/tec/2015/04/1619316-google-passa-a-priorizar-sites-pensados-para-o-celular-e-sofre-criticas.shtml>. Acesso em: 05 jun. 2015.

FOSTER, Elvis C.. **Software Engeneering**: A Methodical Approach. Nova Iorque: Apress, 2014.

GITHUB. **Atom:** flight manual. San Francisco: Creative Commons, 2015.

GOLDMAN, Alfredo et al. A História dos Métodos Ágeis no Brasil. In: PRIKLADNICKI, Rafael; WILLI, Renato; MILANI, Fabiano (Org.). **Métodos Ágeis para Desenvolvimento de Software.** Porto Alegre: Bookman, 2014. Cap. 2. p. 16-21.

GOMES, Alexandre; WILLI, Renato; REHEM, Serge. O Manifesto Ágil. In: PRIKLADNICKI, Rafael; WILLI, Renato; MILANI, Fabiano (Org.). **Métodos Ágeis para Desenvolvimento de Software.** Porto Alegre: Bookman, 2014. Cap. 1. p. 3-15.

GOMES, André Faria. **Agile:** Desenvolvimento de software com entregas frequentes e foco no valor de negócio. São Paulo: Casa do Código, 2013.

GOTHELF, Jeff. **Lean UX.** Sebastopol: O'reilly, 2013.

GREEN, Brad; SESHADRI, Shyam. **AngularJS.** Sebastopol: O'reilly Media, 2013.

HAMILTON, Kim; MILES, Russel. **Learning UML 2.0.** Sebastopol: O'reilly, 2006.

HAVIV, Amos Q.. **MEAN Web Development:** Master real-time web application development using a mean combination of MongoDB, Express, AngularJS, and Node.js. Birmingham: Packt Publishing, 2014.

HELM, Rafael; WILDT, Daniel. **Histórias de Usuário:** Por que e como escrever requisitos de forma ágil?. 3. ed. Porto Alegre: Wildtech, 2014.

HIGHSMITH, Jim. **Agile Software Development Ecosystems.** Boston: Addison-wesley, 2002.

HOWS, David; MEMBREY, Peter; PLUGGE, Eelco. **MongoDB Basics:** A Quick Introduction to MobgoDB. Nova Yorque: Apress, 2014.

HUNDERMARK, Peter. **Do Better Scrum:** An unofficial set of tips and insights into how to implement Scrum well. 3. ed. Cidade do Cabo: Scrumsense, 2014.

IBM. **Build a Java EE app on Bluemix Using Watson and Cloudant.** 2014. Escrito por Jeff Sloyer. Disponível em: <https://developer.ibm.com/bluemix/2014/10/17/building-java-ee-app-ibm-bluemix-using-watson-cloudant/>. Acesso em: 20 jun. 2015.

LANGR, Jeff; OTTINGER, Tim. **Agile in a Flash: Speed-Learning Agile Software Development:** Agile Cards for Agile Teams. Dallas: Pragmatic Bookshelf, 2011. (Pragmatic Programmers).

LESYUK, Andriy. **Mastering Redmine**. Birmingham: Packt Publishing, 2013.

LI, Patrick. **JIRA 5.2 Essentials**. Birmingham: Packt Publishing, 2013.

MANIFESTO ÁGIL (Utah). **Manifesto para desenvolvimento ágil de software**. 2001. Disponível em: <http://www.manifestoagil.com.br/>. Acesso em: 08 mar. 2015.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. **Fundamentos de metodologia cientifica.** 5. ed. São Paulo: Atlas, 2003.

MARDAN, Azat. **Rapid Prototyping with JS**: Agile JavaScript Development. 4. ed. San Francisco: Webapplog.com, 2014.

OLAUSSON, Mathias et al. **Pro Team Foundation Service.** Nova Iorque: Apress, 2013.

PEREIRA, Caio Ribeiro. **Aplicações web real-time com Node.js.** São Paulo: Casa do Código, 2013.

PRIKLADNICKI, Rafael; MAGNO, Alexandre. O Framework do Scrum. In: PRIKLADNICKI, Rafael; WILLI, Renato; MILANI, Fabiano (Org.). **Métodos Ágeis para Desenvolvimento de Software.** Porto Alegre: Bookman, 2014. Cap. 3. p. 22-35.

RASMUSSON, Jonathan. **The Agile Samurai:** How Agile Masters Deliver Great Software. Dallas: Pragmatic Bookshelf, 2010. (Pragmatic Programmers)

REHEM, Serge. **Feedback TCC.** [mensagem pessoal] Mensagem recebida por: <ronaldoscottis@hotmail.com>. em: 13 maio 2015.

RODEN, Ted. **Building the Realtime User Experience.** Sebastopol: O'reilly Media, 2010.

RUIZ, João Álvaro. **Metodologia Científica:** Guia para eficiência nos estudos. 5. ed. São Paulo: Atlas, 2002.

SABBAGH, Rafael. **Scrum: Gestão Ágil para Projetos de Sucesso.** São Paulo: Casa do Código, 2014.

SCHWABER, Ken. **Agile Project Management with Scrum.** Redmond: Microsoft Press, 2004.

SCHWABER, Ken; SUTHERLAND, Jeff. **Guia do Scrum™:** Um guia definitivo para o Scrum: As regras do jogo. Estados Unidos: Scrum.org And Scruminc, 2014. Tradução de: Fábio Cruz, Caio Cestari Silva, Eduardo Rodrigues Sucena e Daniel Racowsky.

SEVILLEJA, Chris; LLOYD, Holly. **MEAN Machine:** A beginner's practical guide to the JavaScript stack. Vancouver: Leanpub, 2015.

SILVA, Maurício Samy. **CSS3:** desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das css3. 2. ed. São Paulo: Novatec, 2012.

SILVA, Maurício Samy. **HTML 5:** A Linguagem de Marcação que Revolucionou a Web. São Paulo: Novatec, 2011.

SPURLOCK, Jake. **Bootstrap.** Sebastopol: O'reilly Media, 2013.

STELLMAN, Andrew; GREENE, Jennifer. **Learning Agile:** Understanding Scrum, XP, Lean and Kanban. Sebastopol: O'reilly Media, 2014.

SUBRAMANIAM, Venkat; HUNT, Andy. **Practices of an Agile Developer:** Working in the Real World. Dallas: Pragmatic Bookshelf, 2006. (Pragmatic Programmers).

TEIXEIRA, Pedro. **Hands-onNode.js.** Funchal, Portugal: Leanpub, 2013.

TELES, Vinícius. Prefácio. In: PRIKLADNICK, Rafael; WILLI, Renato; MILANI, Fabiano (Org.). **Métodos Ágeis para Desenvolvimento de Software.** Porto Alegre: Bookman, 2014. p. 8-9.

VALE, Alisson. Kanban. In: PRIKLADNICKI, Rafael; WILLI, Renato; MILANI, Fabiano (Org.). **Métodos Ágeis para Desenvolvimento de Software.** Porto Alegre: Bookman, 2014. Cap. 8. p. 119-145.

VAISH, Gaurav. **Getting Started with NoSQL.** Birmingham: Packt Publishing, 2013.

VERSIONONE. **State of Agile Survey.** 8. ed. Estados Unidos: Versionone.com, 2014.

VISCARDI, Stacia. **The Professional ScrumMaster's Handbook.** Birmingham: Packt Publishing, 2013. (Professional Expertise Distilled).

WILSON, Mike. **Construindo Aplicações Node com MongoDB e Backbone.** São Paulo: Novatec, 2013.

WYSOCKI, Robert K.. **Effective Project Management:** Traditional, Agile, Extreme. 7. ed. Indianapolis: John Wiley & Sons, Inc., 2014.

YAAPA, Hage. **Express Web Application Development.** Birmingham: Packt Publishing, 2013.

ANEXO A – Código do servidor NodeJS

```
// Invoke 'strict' JavaScript mode
'use strict';

// Set the 'NODE_ENV' variable
process.env.NODE_ENV = process.env.NODE_ENV || 'development';

// Load the module dependencies
var mongoose = require('./config/mongoose'),
       express = require('./config/express'),
       passport = require('./config/passport');

// Create a new Mongoose connection instance
var db = mongoose();

// Create a new Express application instance
var app = express(db);

// Configure the Passport middleware
var passport = passport();

// Use the Express application instance to listen to the '3000' port
app.listen(3000);

// Log the server status to the console
console.log('|=====================================|');
console.log('| Server uSpeed.me running at port 3000 |');
console.log('|=====================================|');

// Use the module.exports property to expose our Express application instance for
external usage
module.exports = app;
```

ANEXO B – Código de criação do módulo principal com AngularJS

```
// Invoke 'strict' JavaScript mode
'use strict';

// Set the main application name
var mainApplicationModuleName = 'uspeedme';

// Create the main application
var mainApplicationModule = angular.module(mainApplicationModuleName,
['ngResource', 'ngRoute', 'angularMoment', 'chart.js', 'users', 'dashboard',
'projects', 'chat', 'tasks']);

// Configure the hashbang URLs using the $locationProvider services
mainApplicationModule.config(['$locationProvider',
	function($locationProvider) {
		$locationProvider.hashPrefix('!');
	}
]);

// Manually bootstrap the AngularJS application
angular.element(document).ready(function() {
	angular.bootstrap(document, [mainApplicationModuleName]);
});
```

ANEXO C – Função de criação de novo usuário

```
// Create a new controller method that creates new 'OAuth' users
exports.saveOAuthUserProfile = function(req, profile, done) {
// Try finding a user document that was registered using the current OAuth provider
	User.findOne({
		provider: profile.provider,
		providerId: profile.providerId
	}, function(err, user) {
		// If an error occurs continue to the next middleware
		if (err) {
			return done(err);
		} else {
			// If a user could not be found, create a new user
			if (!user) {
				// Set a possible base username
				var possibleUsername = profile.username || ((profile.email) ? profile.email.split('@')[0] : '');
				// Find a unique available username
				User.findUniqueUsername(possibleUsername, null, function(availableUsername) {
					// Set the available user name
					profile.username = availableUsername;
					// Create the user
					user = new User(profile);
					// Try saving the new user document
					user.save(function(err) {
						// Continue to the next middleware
						return done(err, user);
					});
				});
			} else {
				// Continue to the next middleware
				return done(err, user);
			}
		}
	});
};
```

ANEXO D – Criação de rotas do módulo projeto

```
// Invoke 'strict' JavaScript mode
'use strict';

// Load the module dependencies
var users = require('../../app/controllers/users.server.controller'),
    tasks = require('../../app/controllers/tasks.server.controller'),
    projects = require('../../app/controllers/projects.server.controller');

// Uses the Express app.route() method to define the base routes for CRUD operations
module.exports = function(app) {
  app.route('/api/projects')
     .get(projects.list)
     .get(tasks.list)
     .post(users.requiresLogin, projects.create);

  app.route('/api/projects/:projectId')
     .get(projects.read)
     .put(users.requiresLogin, projects.hasAuthorization, projects.update)
     .delete(users.requiresLogin, projects.hasAuthorization, projects.delete);

  app.param('projectId', projects.projectByID);
};
```

ANEXO E – Configuração de sessões no Socket.io

```
// Load the module dependencies
var config = require('./config'),
    cookieParser = require('cookie-parser'),
    passport = require('passport');

// Define the Socket.io configuration method
module.exports = function(server, io, mongoStore) {
      // Intercept Socket.io's handshake request
    io.use(function(socket, next) {
      // Use the 'cookie-parser' module to parse the request cookies
       cookieParser(config.sessionSecret)(socket.request, {}, function(err) {
            // Get the session id from the request cookies
           var sessionId = socket.request.signedCookies['connect.sid'];

           // Use the mongoStorage instance to get the Express session information
           mongoStore.get(sessionId, function(err, session) {
            // Set the Socket.io session information
               socket.request.session = session;

               // Use Passport to populate the user details
               passport.initialize()(socket.request, {}, function() {
                  passport.session()(socket.request, {}, function() {
                        if (socket.request.user) {
                              next(null, true);
                        } else {
                              next(new Error('Usuário não autenticado'), false);
                        }
                  });
               });
           });
       });
    });
      // Add an event listener to the 'connection' event
    io.on('connection', function(socket) {
      // Load the chat controller
     require('../app/controllers/chat.server.controller')(io, socket);
    });
};
```