



UFOP

Universidade Federal
de Ouro Preto

**Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas**

***BOPE SYSTEM: Sistema de
Acompanhamento para o processo
BOPE***

Lorena Souza Costa

**TRABALHO DE
CONCLUSÃO DE CURSO**

**ORIENTAÇÃO:
Prof. Igor Muzetti Pereira**

**Setembro, 2017
João Monlevade—MG**

Lorena Souza Costa

***BOPE SYSTEM: Sistema de Acompanhamento
para o processo BOPE***

Orientador: Prof. Igor Muzetti Pereira

Monografia apresentada ao curso de Engenharia de Computação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina “Trabalho de Conclusão de Curso II”.

Universidade Federal de Ouro Preto

João Monlevade

Setembro de 2017

C837b

Costa, Lorena Souza.

Bope System [manuscrito]: sistema de acompanhamento para o processo BOPE / Lorena Souza Costa. - 2017.

83f.:

Orientador: Prof. Me. Igor Muzetti Pereira.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Ciências Exatas e Aplicadas.

1. Software de aplicação - desenvolvimento. 2. Administração de projetos. 3. Gerenciamento de recursos de informação. I. Pereira, Igor Muzetti . II. Universidade Federal de Ouro Preto. III. Título.

CDU: 004.9

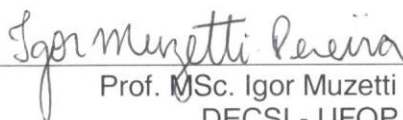
ANEXO IV – Folha de Aprovação
Curso Engenharia de Computação

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

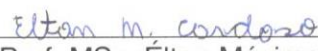
BOPE System: Sistema de Acompanhamento para o processo BOPE

Lorena Souza Costa

Monografia apresentada ao Departamento de Computação e Sistemas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CSI496 – Trabalho de Conclusão de Curso II, do curso de Bacharelado em Engenharia de Computação e aprovada pela Banca Examinadora abaixo assinada:


Prof. MSc. Igor Muzetti Pereira
DECSI - UFOP


Prof. MSc. Vicente José Peixoto de Amorim
DECSI - UFOP


Prof. MSc. Elton Máximo Cardoso
DECSI - UFOP

ANEXO IV - Ata de Defesa

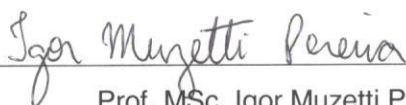
ATA DE DEFESA

Aos 04 dias do mês de setembro de 2017, às 14 horas, na sala C203 do Instituto de Ciências Exatas e Aplicadas, foi realizada a defesa de Monografia pelo aluno **Lorena Souza Costa**, sendo a Comissão Examinadora constituída pelos professores: Prof. MSc. Igor Muzetti Pereira, Prof. MSc. Vicente José Peixoto de Amorim e Prof. MSc. Elton Máximo Cardoso.

O candidato apresentou a monografia intitulada: "*BOPE System: Sistema de Acompanhamento para o processo BOPE*". A comissão examinadora deliberou, por unanimidade, pela aprovação do candidato, com nota 9,5 (nove e meio), concedendo-lhe o prazo de 15 dias para incorporação das alterações sugeridas ao texto final.

Na forma regulamentar, foi lavrada a presente ata que é assinada pelos membros da Comissão Examinadora e pelo graduando.

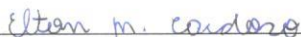
João Monlevade, 04 de setembro de 2017.



Prof. MSc. Igor Muzetti Pereira
Professor Orientador/Presidente



Prof. MSc. Vicente José Peixoto de Amorim
Professor Convidado



Prof. MSc. Elton Máximo Cardoso
Professor Convidado



Lorena Souza Costa
Graduando

TERMO DE RESPONSABILIDADE

Eu, **Lorena Souza Costa** declaro que o texto do trabalho de conclusão de curso intitulado “**BOPE SYSTEM: Sistema de Acompanhamento para o processo BOPE**” é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos respectivos autores.

João Monlevade, 04 de setembro de 2017



Lorena Souza Costa

*Este trabalho é dedicado aos meus pais, irmãos e amigos que me ajudaram nesta imensa
conquista*

Agradecimentos

Agradeço, primeiramente, a Deus por ter me guiado e me proporcionado esta conquista.

Sou grata também à minha família, em especial aos meus pais, Luiz Carlos e Maria de Fátima, que sempre me incentivaram e acreditaram em mim, principalmente, nos momentos mais difíceis.

Aos meus irmãos Pedro e Thais que também me incentivaram e me ajudaram sempre que precisei.

Aos meus amigos pelo companheirismo e paciência que tiveram comigo durante essa jornada.

E, por fim, agradeço à toda equipe iMobilis pela confiança que depositaram em mim, pela ajuda nos momentos difíceis e companheirismo com o decorrer deste árduo trabalho.

Obrigada a todos que contribuíram de alguma forma para que eu concluísse esta etapa de suma importância em minha vida.

“Ter sucesso é falhar repetidamente, mas sem perder o entusiasmo.”

— Winston Churchill

Resumo

O atual trabalho relata a implementação de um sistema *web* responsivo em *Ruby on Rails* para tornar o gerenciamento de projetos mais fácil no processo BOPE, dentro do laboratório de computação Móvel e Sistemas Embarcados, o iMobilis-JM. Foi proposto o desenvolvimento de um aplicativo onde sua principal contribuição é auxiliar esse gerenciamento além de agilizar a geração de relatórios de acompanhamento BOPE. Para alcançar esse objetivo, foi necessário entender o processo BOPE e adaptar o sistema para acompanhar este processo. As funcionalidades do sistema foram identificadas e, posteriormente, implementadas no software de acordo com as suas necessidades. Este trabalho relata as etapas seguidas para a criação desse sistema juntamente com as boas práticas de Engenharia de Software utilizadas em cada uma destas fases. Logo, o principal efeito deste projeto foi a implantação do software BOPE SYSTEM que se resume num sistema de acompanhamento para o processo BOPE.

Palavras-chaves: Sistema *web*. *Ruby on Rails*. Gerenciamento de projetos. Boas práticas de Engenharia de Software. BOPE SYSTEM. iMobilis.

Abstract

The current work reports the implementation of a responsive web system in Ruby on Rails to make project management easier in the BOPE process inside of the Mobile and Embedded Systems lab, iMobilis-JM. It was proposed the development of an application where its main contribution is to assist this management besides to speed up the generation of BOPE monitoring reports. To achieve this goal, it was necessary to understand the BOPE process and adapt the system to follow this process. The system features were identified and then implemented in the software according to your needs. This work reports the steps followed to create this system together with the good Software Engineering practices used in each of these phases. Therefore, the main effect of this project was the implementation of the BOPE SYSTEM software, which is summarized in a monitoring system for the BOPE process.

Keywords: Web system. Ruby on Rails. Project management. Good Software Engineering practices. BOPE SYSTEM. iMobilis.

Lista de ilustrações

Figura 1 – Padrão MVC usado no Rails	26
Figura 2 – Protótipo da tela de Visualização e Criação de Atividade	38
Figura 3 – Diagrama ER do sistema BOPE SYSTEM	40
Figura 4 – Estrutura das pastas e arquivos geradas automaticamente no Rails	41
Figura 5 – Configuração do Banco de Dados PostgreSQL	43
Figura 6 – Modelo <i>user_profile</i>	44
Figura 7 – Controlador do <i>status</i>	45
Figura 8 – Autenticação do usuário	46
Figura 9 – Autotização do usuário	47
Figura 10 – Visão Geral do sistema do usuário colaborador	47
Figura 11 – Visão Geral do sistema do usuário administrador	48
Figura 12 – Associação <i>belongs_to</i> entre <i>status</i> e <i>color</i>	48
Figura 13 – Associação <i>has_one</i> entre <i>team</i> e <i>project</i>	49
Figura 14 – Associação <i>has_many</i> entre <i>label</i> e <i>activities</i>	49
Figura 15 – Tela de login do sistema	51
Figura 16 – Visão do administrador com breve descrição do sistema	52
Figura 17 – Visualização para o administrador da lista de usuários	52
Figura 18 – Visualização para o administrador da criação de um usuário	53
Figura 19 – Visualização para o administrador do perfil de um usuário específico	53
Figura 20 – Visualização para o administrador da edição de um usuário específico	54
Figura 21 – Visualização para o administrador da exclusão de um usuário específico	54
Figura 22 – Visualização para o administrador da lista das equipes cadastradas	55
Figura 23 – Visualização para o administrador da criação de uma equipe	55
Figura 24 – Visualização para o administrador de uma equipe específica	56
Figura 25 – Visualização para o administrador da edição dos usuários de uma equipe específica	56
Figura 26 – Visualização para o administrador da lista de projetos cadastrados	57
Figura 27 – Visualização para o administrador da criação de um novo projeto	57
Figura 28 – Visualização para o administrador de um projeto e suas informações	58
Figura 29 – Visualização para o administrador do progresso de um projeto	58
Figura 30 – Visualização para o administrador da listagem dos <i>Sprints</i> dentro de um projeto	59
Figura 31 – Visualização para o administrador da listagem dos rótulos para caracte- rizar uma atividade	59
Figura 32 – Visualização para o administrador da listagem das cores para serem usadas nos rótulos e estados das atividades	60

Figura 33 – Visualização para o administrador da listagem dos estados para caracterizar uma atividade	60
Figura 34 – Visualização para o administrador da listagem dos relatórios	61
Figura 35 – Visualização para o administrador da busca de um relatório específico ao selecioná-lo	61
Figura 36 – Visualização para o administrador dos possíveis relatórios de Sprint . . .	62
Figura 37 – Visualização para o administrador dos possíveis relatórios semanais . . .	62
Figura 38 – Relatório de dedicação relativa dos usuários	63
Figura 39 – Relatório de Acompanhamento de equipe: dedicação relativa de uma equipe específica	63
Figura 40 – Relatório de Acompanhamento de equipe: burndown de uma equipe específica	64
Figura 41 – Relatório de Acompanhamento de equipe: velocidade de uma equipe específica	64
Figura 42 – Relatório de atividades para uma equipe específica	65
Figura 43 – Relatório de avanço e completude para uma equipe específica	65
Figura 44 – Relatório de produtividade relativa para uma equipe específica	66
Figura 45 – Visão do colaborador com a aba “Projetos” selecionada	66
Figura 46 – Autenticar e adicionar usuário	72
Figura 47 – Editar e excluir usuário	72
Figura 48 – Visualizar perfil do usuário	73
Figura 49 – Cadastrar e editar equipe	73
Figura 50 – Excluir equipe e vincular usuário a uma equipe	74
Figura 51 – Excluir usuário da equipe e visualizar equipe	74
Figura 52 – Cadastrar e editar atividade	75
Figura 53 – Excluir e visualizar atividade	75
Figura 54 – Fechar atividade	75
Figura 55 – Cadastrar e editar projeto	76
Figura 56 – Excluir e visualizar projeto	76
Figura 57 – Vincular equipe a um projeto	77
Figura 58 – Adicionar e editar Sprint	77
Figura 59 – Excluir e visualizar Sprint	78
Figura 60 – Visualizar pontos atualizados e cadastrados no Sprint	78
Figura 61 – Adicionar e editar semana no Sprint	79
Figura 62 – Excluir semana do Sprint	79
Figura 63 – Logar e deslogar em horas nas macroatividades	80
Figura 64 – Visualizar porcentagem das macroatividades	80
Figura 65 – Gerar Relatórios de dedicação relativa de cada colaborador e equipe, e relatório de velocidade de cada equipe por semana	81

Figura 66 – Gerar relatório do burndown de cada equipe por semana, e relatórios de atividades, completude e avanço de cada projeto por Sprint	81
Figura 67 – Instalando o Ruby	82
Figura 68 – Instalando o Rails e PostgreSQL	83
Figura 69 – Executando a aplicação	83

Lista de tabelas

Tabela 1 – <i>Product Backlog</i>	35
---	----

Lista de abreviaturas e siglas

PMBOK	Project Management Body of Knowledge
ICEA	Instituto de Ciências Exatas e Aplicadas
UFOP	Universidade Federal de Ouro Preto
RoR	Ruby on Rails
SGBDOR	Sistema Gerenciador de Banco de Dados Objeto-Relacional
DRY	Don't Repeat Yourself
CoC	Convention Over Configuration
MVC	Model-View-Controller
RAD	Rapid Application Development
MVCC	Multiversion Concurrency Control

Sumário

1	INTRODUÇÃO	18
1.1	Visão Geral	18
1.2	O problema de pesquisa	19
1.3	Objetivos	19
1.4	Organização do trabalho	20
2	REVISÃO BIBLIOGRÁFICA	21
2.1	Conceitos básicos	21
2.1.1	Métricas	21
2.1.1.1	Dedicação relativa	21
2.1.1.2	Velocidade	22
2.1.1.3	Burndown	22
2.1.1.4	Compleitude e avanço	23
2.1.1.5	Produtividade relativa	23
2.1.2	A linguagem <i>Ruby</i>	24
2.1.3	O framework <i>Ruby on Rails</i>	25
2.1.4	Gerenciador de Banco de Dados Objeto-Relacional PostgreSQL	27
2.1.5	Gems	27
2.1.5.1	Bootstrap	28
2.1.5.2	Devise	28
2.1.5.3	CanCan	28
2.1.5.4	Devise i18n	28
2.2	Trabalhos correlatos	29
2.2.1	<i>RedMine</i>	29
2.2.2	<i>Confluence</i>	30
2.2.3	<i>Trello</i>	31
2.2.4	<i>Pivotal Tracker</i>	31
2.2.5	<i>Slack</i>	32
3	METODOLOGIA	34
3.1	Concepção e Elaboração	34
3.1.1	<i>Product Backlog</i>	34
3.1.2	<i>Histórias de usuários</i>	35
3.1.3	<i>Cenários de teste</i>	36
3.1.4	<i>Protótipos de tela</i>	37
3.1.5	<i>Projeto de Banco de Dados</i>	39

3.2	<i>Implementação</i>	40
3.2.1	Criação do Projeto	40
3.2.2	Autenticação e autorização de usuários	46
3.2.3	Associação de tabelas no Rails	48
3.2.4	Geração de PDF	49
3.2.5	<i>Testes</i>	50
3.3	Implantação do sistema no laboratório iMobilis	50
4	APRESENTAÇÃO DOS RESULTADOS	51
5	CONSIDERAÇÕES FINAIS	68
5.1	Conclusão	68
5.2	Trabalhos Futuros e Limitações	69
6	REFERÊNCIAS BIBLIOGRÁFICAS	70

APÊNDICES **71**

APÊNDICE A – HISTÓRIAS DE USUÁRIO E CENÁRIOS DE TESTE **72**

APÊNDICE B – TUTORIAL DE IMPLANTAÇÃO DO SISTEMA NO LABORATÓRIO IMOBILIS **82**

1 Introdução

Este capítulo de Introdução é dividido nas seguintes seções: elaboração do capítulo, o problema da pesquisa, objetivos e, por último, organização do trabalho.

1.1 Visão Geral

Conforme PMBOK (2008), um projeto pode se resumir em um conjunto de atividades, onde estas são temporárias, uma vez que contêm um início e fim definidos. Esse conjunto de atividades tem o intuito de produzir ou aperfeiçoar produtos, serviços ou resultados específicos. O gerenciamento de projetos é feito para atingir os objetivos específicos utilizando conhecimentos, habilidades e técnicas e obedecendo um cronograma e custos definidos.

De acordo com Sommerville (2011), o gerenciamento de projetos de software é vital na engenharia de software, uma vez que os mesmos necessitam ser gerenciados devido a limitações de cronograma e orçamentos organizacionais. Além disso o papel do gerente de projetos é assegurar que o projeto ofereça softwares de qualidade boa e, além disso, atenda e supere tais limitações. Vale ainda ressaltar que garantir um bom gerenciamento de projeto não garante a obtenção de sucesso, mas quando se tem um mau gerenciamento, geralmente, isso resulta na falha do projeto.

Os grupos de processos do gerenciamento de projetos são: início, planejamento, execução, monitoramento e controle e encerramento. O fato de gerenciar um projeto provê que o objetivo seja alcançado no prazo correto, com o custo e da forma que foram anteriormente planejados.

É interessante ressaltar os critérios a serem seguidos para se ter êxito no gerenciamento de projetos. Ainda de acordo com Sommerville (2011), tais critérios são: entrega do software no prazo estabelecido, custos devem estar dentro do orçamento feito, exige-se que o software atenda às expectativas do cliente e, por fim, a equipe de desenvolvimento precisa trabalhar bem e feliz.

O laboratório iMobilis-JM utiliza o processo de software híbrido, conhecido como BOPE, em Pereira et al., (2013). Este processo mistura práticas de engenharia de software oriundas de processos tradicionais e ágeis para se adaptarem ao contexto acadêmico de um laboratório de pesquisa e desenvolvimento de software. Através de tal processo, é possível adaptar às mudanças no planejamento de forma rápida, corrigindo ações das equipes com eficiência. Vale ressaltar que no BOPE, um artefato é gerado a cada atividade do processo e para cada necessidade do cliente mapeia-se uma história de usuário e cenários de teste,

conduzindo o desenvolvimento do software do cliente para o desenvolvedor.

Conforme Sommerville (2011), o desenvolvimento ágil deve ser gerenciado de forma com que se faça o melhor uso do tempo e dos recursos disponíveis para cada equipe. Com isso, exige-se uma abordagem diferente para o gerenciamento de projetos, uma vez que esta tem que ser adaptada para o desenvolvimento incremental e para os pontos cruciais dos métodos ágeis.

Diante deste contexto, o presente trabalho retrata a criação do BOPE SYSTEM que é um sistema de acompanhamento para o processo BOPE. Este sistema tem como principal intuito tornar o gerenciamento de projetos mais fácil, onde sua principal funcionalidade é a geração automatizada de relatórios para aperfeiçoar esse processo. O sistema contabiliza as horas e registra as atividades com suas devidas pontuações para cada usuário e equipe, e por fim, gera os relatórios de acordo com estes dados obtidos. Nestes relatórios são especificadas as seguintes métricas: dedicação relativa de cada usuário e equipe; velocidade, *burndown* e produtividade relativa de cada equipe; e além disso, a completude e avanço de cada projeto dentro do iMobilis.

1.2 O problema de pesquisa

O laboratório iMobilis do campus ICEA/UFOP é um laboratório de pesquisa em computação, onde se desenvolve sistemas para dispositivos móveis, *web* e embarcados usando boas práticas da engenharia de software. Os projetos dentro do laboratório seguem práticas de engenharia de software provenientes do BOPE com o intuito de acelerar o desenvolvimento de software focando na melhoria contínua do processo e organização da equipe.

Ao levar em consideração esta perspectiva, o BOPE SYSTEM auxilia no gerenciamento de projetos para o processo de desenvolvimento BOPE, de forma a gerar os relatórios automaticamente, uma vez que, anteriormente, eles eram feitos a mão e isso gerava um custo maior e os tornavam mais propensos a erros. Estes relatórios contêm as métricas necessárias para se fazer uma análise do comprometimento e desempenho dos membros do laboratório. Além disso, com a construção deste sistema *web*, é possível manter o registro de informações dos projetos ao longo do tempo. Vale ressaltar que este presente aplicativo não se restringe apenas ao laboratório iMobilis, podendo também ser usado por qualquer instituição que adira o processo BOPE.

1.3 Objetivos

O presente trabalho consiste em definir, construir e implantar um sistema *web* em *RoR* (*Ruby on Rails*) para apoiar o gerenciamento de projetos no processo BOPE. Para

isso ocorrer, foi necessário que o BOPE SYSTEM seguisse o processo BOPE que utiliza práticas de engenharia de software, priorizando a interação entre todos os colaboradores nos projetos em execução.

Este trabalho possui os seguintes objetivos específicos:

- Pesquisar e entender as ferramentas de gestão ágil de projetos de software.
- Projetar um sistema que se adeque a cultura do BOPE.
- Desenvolver o BOPE SYSTEM.
- Implantar o BOPE SYSTEM no iMobilis.

1.4 Organização do trabalho

Este trabalho está estruturado em mais três seções: revisão bibliográfica, metodologia e, por fim, considerações finais.

A Seção 2 aborda a revisão bibliográfica, a qual apresenta duas subseções: conceitos básicos e trabalhos correlatos. A primeira subseção retrata os conceitos básicos para melhor entendimento deste presente trabalho. São eles: métricas, linguagem *Ruby*, *framework Ruby on Rails*, *SGBDOR PostgreSQL* e *Gems*. A segunda subseção fala sobre os trabalhos correlatos a este estudo que são as seguintes ferramentas: *RedMine*, *Confluence*, *Trello* e *Pivotal Tracker*.

A Seção 3, referente à metodologia, apresenta cada etapa realizada para a criação do presente projeto e os procedimentos metodológicos essenciais para a criação do sistema.

A seção 4 retrata os resultados do presente trabalho obtidos com a implantação do sistema no laboratório.

Por último, a Seção 5 relata as considerações finais que se resume na conclusão, limitações do software e trabalhos futuros.

2 Revisão bibliográfica

Este capítulo de revisão bibliográfica é composto por duas seções: conceitos básicos e trabalhos correlatos. Primeiramente, os conceitos básicos relacionados ao trabalho desenvolvido são abordados para um melhor entendimento deste. Posteriormente, serão apresentados os trabalhos correlatos, ou seja, os trabalhos relacionados a pesquisa em questão.

2.1 Conceitos básicos

Nesta seção serão abordadas as métricas utilizadas para auxiliar o Gerenciamento de Processos no laboratório iMobilis, assim como as tecnologias utilizadas para o desenvolvimento deste trabalho.

2.1.1 Métricas

A medição é muito utilizada na engenharia de software e, segundo Pressman (2016), ela permite um melhor entendimento do processo e projeto, tornando possível a obtenção de um mecanismo para uma avaliação objetiva. No entanto, há divergências sobre o que exatamente medir e como avaliar o resultado obtido das medições. O processo de medição resulta na transformação de atributos de qualidade em dados numéricos, estes dados são classificados em métricas de software. As métricas de software podem ser consideradas parâmetros e estatísticas exibidas quantitativamente com relação ao desempenho de um produto de software ou processo. Escolher bem as métricas para se utilizar em um determinado projeto é de suma importância, já que é através delas que se evolui e aperfeiçoa-se o modo de trabalhar. A seguir, são abordadas as métricas usadas neste trabalho.

2.1.1.1 Dedicção relativa

Esta métrica é usada para medir a dedicação de cada usuário e equipe no laboratório em relação as horas realizadas e esperadas destes. A dedicação relativa utiliza duas outras métricas para ser calculada: dedicação real e esperada. A dedicação real representa as horas realizadas pelos usuários para executarem suas devidas atividades, enquanto que a dedicação esperada se resume às horas esperadas que cada usuário deve alcançar. Logo, a dedicação relativa pode ser expressa através da divisão das horas realizadas pelas horas esperadas. Um outro ponto a evidenciar é que a métrica em questão é calculada para

usuários e equipes tanto semanalmente quanto por *Sprint*. Deste modo, se pode descrever as seguintes fórmulas:

$$\text{Dedicação relativa} = \frac{\text{dedicação real}}{\text{dedicação esperada}}$$

$$\text{Dedicação real} = \text{horas realizadas}$$

$$\text{Dedicação esperada} = \text{horas esperadas}$$

2.1.1.2 Velocidade

A velocidade é uma métrica que calcula a quantidade de trabalho que uma equipe pode completar em um determinado período. Neste caso específico, a velocidade representa a quantidade de pontos realizados por uma equipe durante a semana e também durante um *Sprint*. Vale ressaltar que tais pontos são provenientes das atividades que cada usuário tem que realizar. É interessante ressaltar ainda que, um *Sprint* não tem um tempo fixo, podendo variar entre quatro e cinco semanas. Ela é calculada através da seguinte expressão:

$$\text{velocidade} = \text{quantidade de pontos realizados}$$

2.1.1.3 Burndown

O *burndown* é uma métrica que representa o progresso de cada equipe para completar todos os pontos que se comprometeram a fazer em um *Sprint*. Ele é representado através de um gráfico que começa com o número total de pontos que uma equipe precisa realizar em um determinado *Sprint* e, a cada semana, subtrai-se desse total, a quantidade de pontos que foram realizados. Vale ressaltar que esse gráfico representa o *burndown* real, mas existe ainda o *burndown* ideal.

Segundo Cohn (2010), *burndown* é um gráfico que mostra a quantidade de trabalho cumulativo restante de um *Sprint*, dia por dia. Neste gráfico, o eixo y (eixo das ordenadas) indica a quantidade de tarefas do *Sprint Backlog* não completadas, e o eixo x (eixo das abscissas) são os dias. Este gráfico é um dos principais recursos de medição do processo de desenvolvimento e um diferencial para a metodologia *Scrum*. Quanto mais horizontal, melhor.

O *burndown* ideal representa a subtração entre os pontos desejados e a divisão correta de pontos para cada semana levando em consideração os pontos totais e quantas semanas possui aquela determinado *Sprint*. Por exemplo, existe um *Sprint* composto por cinco semanas, onde uma equipe tem que completar sessenta pontos. Logo, o ideal é de que eles realizem doze pontos por semana e o *burndown* é calculado através da subtração desses pontos desejados e o ideal de pontos que precisam realizar a cada semana.

Outro aspecto importante é o fato de que o *burndown*, tanto o ideal quanto o real, é um gráfico que decai com o passar do tempo à medida que os pontos são realizados. Além disso, através dele se tem um maior controle ao realizar as tarefas requeridas, uma vez que se é possível saber o esforço restante de acordo com o tempo disponível.

Deste modo, o *burndown* ideal e real podem ser descritos através das seguintes fórmulas:

$$Ideal = \text{pontos planejados no Sprint} - \left(\frac{\text{pontos planejados no Sprint}}{\text{quantidade de semanas}} \right) * \text{número da semana atual}$$

$$Real = \text{pontos planejados no Sprint} - \text{pontos realizados até o momento}$$

2.1.1.4 Completude e avanço

Outra métrica importante usada no laboratório iMobilis é a completude do projeto, onde ela mostra o progresso de cada projeto levando em consideração todas as macro atividades deste. As macro atividades representam as divisões do seu projeto em relação às atividades que são desenvolvidas nele. Deste modo, a completude pode ser calculada através da porcentagem correspondente à quantidade de macro atividades do projeto.

Através da completude, calcula-se o avanço do projeto, o qual se resume na subtração entre a completude do *Sprint* atual e a completude do *Sprint* anterior. Vale lembrar que o avanço também é dado em porcentagem.

$$Completude \text{ do projeto} = \text{média das completudes de todas as macro atividades}$$

$$Avanço \text{ da Sprint atual} = \text{completude do Sprint atual} - \text{completude do Sprint anterior}$$

2.1.1.5 Produtividade relativa

Por fim, a última métrica utilizada é a produtividade relativa que relaciona tanto as horas quanto as pontuações de cada equipe em uma determinado *Sprint*. Esta métrica envolve outras duas métricas: velocidade relativa e dedicação relativa. A dedicação relativa já foi abordada anteriormente. Percebe-se que a velocidade relativa é a velocidade descrita também, anteriormente, dividida pela velocidade esperada. A velocidade esperada abrange a quantidade de pontos desejados naquele *Sprint*. Então, após calcular a velocidade e a dedicação relativas, simplesmente se faz a divisão da primeira pela segunda para obter essa produtividade.

$$Produtividade \text{ relativa} = \frac{\text{velocidade relativa}}{\text{dedicação relativa}}$$

$$Velocidade \text{ relativa} = \frac{\text{pontos realizados}}{\text{pontos planejados}}$$

2.1.2 A linguagem *Ruby*

*Ruby*¹ é uma linguagem de programação dinâmica e de código aberto onde seu objetivo é a simplicidade e produtividade. Ela foi projetada e desenvolvida em 1995 por Yukihiro “Matz” Matsumoto no Japão. Cabe salientar que tal linguagem tem uma sintaxe que é fácil de compreender e escrever. Outro ponto a evidenciar é que o *Ruby* suporta vários paradigmas de programação (multiparadigma), abrangendo a programação funcional, orientada a objetos, imperativa e reflexiva.

Esta linguagem dinâmica também chamada de linguagem de *scripting*, como o próprio nome já diz, é caracterizada pela tipagem dinâmica. Este tipo de tipagem permite que o desenvolvedor não necessite de especificar o tipo de dados de uma variável, retorno de método ou atributo.

Sabe-se que a linguagem imperativa é orientada a ações, sendo que a computação é feita através de uma sequência de instruções onde estas manipulam os valores das variáveis presentes no código. Por outro lado, a programação funcional, como o próprio nome já diz, tem como foco o conceito de funções, onde simplesmente avaliam-se as expressões contidas em um determinado código. Desta forma, o programador cria uma função para realizar determinada tarefa e passa tal função para o computador, lembrando que esta função pode conter outras funções. Com isso, o computador avalia as expressões no programa e chega a uma conclusão, a qual é mostrada para o programador. Em suma, enquanto a programação imperativa foca nas mudanças no estado do programa, a funcional enfatiza o uso de funções, não possuindo o conceito de “variáveis de estado”.

Para um melhor entendimento de ambos os paradigmas, deve-se lembrar que na linguagem imperativa, o desenvolvedor escreve o código como uma sequência de ordens, onde o sistema deve se comportar da forma como ele espera. Com isso, o software executará as ações na forma definida pelo desenvolvedor. Já na programação funcional, o desenvolvedor escreve as funções e o software funcionará de acordo com a interação entre essas expressões matemáticas, sendo que o resultado de uma determinada função irá servir como parâmetro para outras e assim sucessivamente.

Outro aspecto fundamental é o fato do *Ruby* suportar também a linguagem orientada a objetos, onde o foco desta é tentar aproximar o mundo real do mundo virtual com a utilização de objetos. O programador precisa moldar o mundo dos objetos e como eles interagem entre si. Neste paradigma, é possível a reutilização de código o que facilita muito o trabalho dos programadores.

Adicionalmente, o *Ruby* também é uma linguagem reflexiva, que, como o próprio nome diz, usa a reflexão que é um processo onde um programa é capaz de ler seus próprios metadados em tempo de execução. Com isso, uma aplicação pode alterar seu

¹ <<https://www.ruby-lang.org/pt/>>

comportamento pelo fato de ter adquirido novas informações durante a execução desta. O termo reflexão é usado pelo fato de que o sistema é capaz de raciocinar sobre si mesmo, a fim de solucionar problemas em tempo de execução. Lembrando que, a linguagem reflexiva é a implementação deste conceito dentro da orientação à objetos.

Por fim, o código *Ruby* é muito legível e auto-documentado, facilitando o entendimento para os futuros desenvolvedores de um determinado projeto e melhorando a sua produtividade.

2.1.3 O framework *Ruby on Rails*

Em 2003, o *Ruby on Rails* ² surgiu como um *framework* criado pelo David H. Hansson. *Rails* é um *framework* de desenvolvimento de aplicações *web* escrito na linguagem *Ruby* que foi projetado para facilitar a programação *web*, possuindo uma plataforma independente, completa e *open source*.

Segundo Vinícius Fuentes (2014), pode-se dizer que o *Ruby on Rails* segue duas filosofias: *DRY* (*Don't Repeat Yourself*) e o *CoC* (*Convention Over Configuration*). O *DRY* se resume em não repetir o código na aplicação, de forma que, se há a necessidade de utilizar o mesmo código novamente, o ideal é colocá-lo em um único local e, assim, toda vez que for necessário usá-lo, ele é buscado neste ponto único. Então, se for necessário alterar o código que foi reaproveitado, a edição deste só ocorrerá neste único lugar. Por não escrever repetidamente a mesma informação, o código se torna mais enxuto, sustentável, extensível e com menos *bugs*.

A outra filosofia é o *CoC*, onde o *Rails* sugere uma boa forma de padronização em uma aplicação *web* para garantir uma maior facilidade na configuração do ambiente de trabalho. Deste modo, deve-se, preferencialmente, aceitar as convenções do *Rails* ao invés de configurá-las, já que seguindo-as o *framework* saberá o que fazer. Um exemplo de convenção no *Rails* são as tabelas que têm seus nomes no plural, enquanto que os nomes dos *models* estão no singular.

Outra característica muito importante a ser citada é o padrão de arquitetura de software *MVC* (*Model-View-Controller*) utilizada pelo *Rails* que separa a representação da informação e a interação dos dados no sistema, dividindo a aplicação em três partes: modelo, visão e controlador. Deste modo, se consegue, através do *MVC*, representar a forma como o usuário interage com a aplicação.

O *model* (modelo) consiste nos dados da aplicação, assim como suas validações, associações e transações. Além disso, é onde se tem a lógica da aplicação e regras de negócio que administram o acesso e a alteração dos dados. Em suma, o modelo sabe como executar as tarefas do aplicativo, mas este não precisa saber qual é o momento de serem

² <<http://rubyonrails.org>>

realizadas e nem como se deve mostrar os dados. Deste modo, ele sabe o que a aplicação quer fazer e pode ser considerado a parte principal da arquitetura, já que é ele quem modela o problema a ser resolvido.

A visão, de forma geral, tem como intuito apresentar os dados em um determinado formato para o usuário. Logo, a visão pode se resumir como a camada de interação com o usuário onde ela irá exibir os dados para ele. Baseado nesse contexto, se pode afirmar que a *view* não sabe o que a aplicação está fazendo, já que esta camada só irá receber as instruções do controlador e as informações presentes no modelo e, deste modo, exibi-las. Então, resumidamente, na visão não há uma preocupação em como ou onde se obtém a informação, ela simplesmente a exibe.

O controlador pode ser considerado o administrador da interação entre a visão e o modelo, onde ele manipula o fluxo e os dados do banco para serem apresentados na visão. Diante disso, percebe-se que o *controller* controla qual *model* vai ser usado e qual *view* será mostrada na tela. Neste sentido, ele não precisa saber de qual forma os dados devem ser obtidos e nem como mostrá-los para o usuário, simplesmente quando se deve fazer isso.

Vale ressaltar que as ideias centrais por trás do MVC são a reusabilidade de código e a separação de conceitos (*Model*, *View* e *Controller*), onde as responsabilidades são divididas em camadas de apresentação, negócio e acesso aos dados. Além disso, o modelo e a visão não conversam diretamente, já que eles usam o controlador como intermediário para isso, como mostra a figura a seguir:

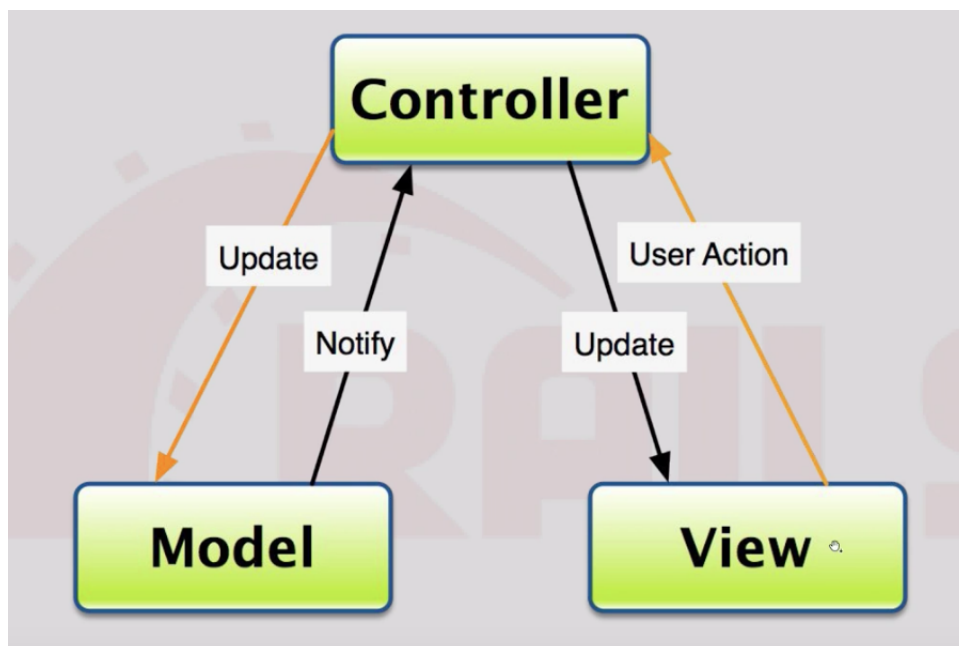


Figura 1 – Padrão MVC usado no Rails

Figura 1: Fonte: PIRES, Jackson. Ruby on Rails - Curso Completo. 2016.

Algumas vantagens de usar o *Ruby on Rails* é que o processo de programar é muito mais rápido do que em outros *frameworks* e linguagens, ou seja, usando-o se consegue trabalhar de forma mais rápida onde o resultado é mais imediato. Além disso, as convenções Rails tornam fácil para os desenvolvedores se moverem entre diferentes projetos, onde cada um segue as mesmas estruturas e práticas de codificação. Ele também é bom para *RAD* (*Rapid Application Development*) que é o desenvolvimento rápido de aplicações e é fácil de incluir mudanças, ou seja, consegue-se produzir muito rapidamente conteúdos e se tiver mudanças no projeto, isso é feito de uma maneira fácil.

Um outro aspecto a ser levado em conta é a sua facilidade no desenvolvimento de códigos complexos de forma rápida usando Gems, as quais serão abordadas adiante. Por fim, o *Rails* utiliza conceitos do desenvolvimento ágil com intuito de se ter uma criação do sistema *web* de forma rápida e com uma maior qualidade, combinando a linguagem de programação *Ruby* com HTML, CSS e JavaScript.

2.1.4 Gerenciador de Banco de Dados Objeto-Relacional PostgreSQL

Sabe-se que PostgreSQL é um *SGBDOR* (*Sistema Gerenciador de Banco de Dados Objeto-Relacional*) de código aberto que permite a integração de seus próprios tipos de dados e métodos personalizados ao banco de dados. Vale ressaltar que ele é muito robusto, confiável, flexível e contém inúmeros recursos. Destes recursos, se pode citar: consultas complexas, chaves estrangeiras, facilidade de acesso, *views*, *triggers*, linguagem procedural, indexação por texto, integridade transacional, suporte ao modelo híbrido objeto-relacional, controle de concorrência multi-versão, entre outros.

Percebe-se que o PostgreSQL é avançado pelo fato dele conter muitos recursos. Segundo André Milani (2008), ele possui o MVCC (*Multiversion Concurrency Control*) que é um mecanismo de bloqueio, onde ele trata problemas de ambientes multiusuários no banco de dados. Além disso, esse SGBDOR suporta tabelas e banco de dados de grande porte. Convém salientar que ele também abrange um bom mecanismo de FAILSAVE (à prova de falhas) que fornece segurança contra falhas.

Logo, o PostgreSQL foi escolhido por ser gratuito, suportar bases de dados muito grandes e complexas, além de oferecer confiabilidade e escalabilidade, já que tais características são essenciais neste presente trabalho. Adicionalmente, ele é multiplataforma e é compatível com várias linguagens de programação, dentre elas, o Ruby.

2.1.5 Gems

De um modo geral, as Gems são bibliotecas ou conjunto de arquivos *Ruby* reutilizáveis etiquetados com um nome e uma versão. Ou seja, se tem um código *Ruby* onde este pode ser reutilizado em outros projetos, basta “empacotar” tal código colocando um

nome e uma versão para ele. Existem diversas Gems no *Ruby on Rails* que possuem uma série de funcionalidades, facilitando muito o desenvolvimento de uma aplicação. A seguir, listam-se as principais Gems utilizadas neste projeto.

2.1.5.1 Bootstrap

O *bootstrap* é um *framework web front-end* de código aberto e livre utilizado para criar sites e aplicativos *web*. Ele é um dos mais populares *frameworks HTML, CSS e JS* para o desenvolvimento *web* rápido e fácil. Em oposição a maioria dos *frameworks web*, o *bootstrap* lida apenas com a criação *front-end*. E ao considerar essa perspectiva, é importante enfatizar que o objetivo principal do *bootstrap* é agilizar e facilitar o desenvolvimento *web* e, por estes motivos, além da flexibilidade que ele possibilita, esta gem foi utilizada no atual trabalho. Cabe salientar que, neste projeto, foi utilizado um *template* do *bootstrap* chamado *cyborg*, caracterizado pela sua cor de fundo preta.

2.1.5.2 Devise

O *devise* é uma gem usada para autenticação do usuário que verifica se o mesmo está cadastrado no sistema ou não. Ela é muito utilizada no Rails no quesito de autenticação, já que permite a implementação do *login*, *logout* e *register* com poucas linhas de código, refletindo o principal motivo para sua utilização no desenvolvimento deste software. No caso deste trabalho, só foram utilizadas as funcionalidades de *login* e *logout*, uma vez que o administrador já é previamente cadastrado no sistema e terá a função de cadastrar os demais colaboradores.

2.1.5.3 CanCan

Esta gem é uma biblioteca de autorização contida no Rails que define o que o usuário pode ou não ter acesso, restringindo ou não os recursos existentes dentro do sistema. As permissões dadas ou não ao usuário, são definidas em um único local no código, de forma a não serem duplicadas nos *controllers*, *models* e *views*. Devido a sua facilidade de utilização, esta gem foi aplicada ao software em questão. A autorização é um ponto muito importante para a maioria das aplicações e, normalmente, acontece após um usuário ser autenticado.

2.1.5.4 Devise i18n

O *devise i18n* é responsável pela internacionalização do *Rails* em relação aos diferentes idiomas existentes. Logo, ela é uma biblioteca usada para traduzir para o idioma desejado, atendendo a necessidade de cada desenvolvedor. Esta é a gem mais utilizada para este propósito, por isso ela foi escolhida para ser usada no aplicativo.

2.2 Trabalhos correlatos

2.2.1 *RedMine*

A ferramenta *RedMine* ³ tem como objetivo gerenciar projetos e subprojetos de forma flexível. Ela opera em serviço *web*, não tendo a necessidade de instalá-la em cada máquina, e também foi feita em *Ruby on Rails*. De maneira geral, se pode dizer que esta aplicação *web* é um software livre, multiplataforma e multi-idíomas, tem suporte para vários projetos, e por fim, é de fácil configuração.

Nos dias atuais, existem muitos problemas dentro dos projetos dos quais se pode citar: não cumprimento de prazos, problemas de comunicação entre os membros da equipe, falha no controle das tarefas especificadas e nas organizações das documentações, e além disso, dificuldade na recuperação de informações. Isso tudo ocorre, principalmente, pelo fato de não se ter um gerenciamento de projeto eficiente. Portanto, o *RedMine* veio com o intuito de minimizar ou até mesmo extinguir tais problemas.

Então, o *RedMine* possui aspectos relevantes que podem tratar esses tipos de problemas, dos quais se pode citar: gerenciamento de tarefas e de tempo do projeto, publicação de notícias do projeto para todos os membros das equipes, fóruns para participação e troca de idéias entre os colaboradores, calendário básico para controle das datas dos projetos, gráfico de *Gantt* para mostrar o avanço das atividades exibindo as datas inicial e final para cada uma delas, e por último, o registro das atividades.

A correlação desta ferramenta com o presente projeto se resume basicamente no fato de ambos os trabalhos auxiliarem no gerenciamento de projetos definindo tarefas para cada colaborador dentro de uma equipe e definindo prazos para cada projeto em questão. Ambos fornecem a visualização do projeto, além do seu progresso e prazo para se ter um maior controle na entrega.

No entanto, se pode perceber algumas diferenças entre eles, como por exemplo, o *RedMine* fornece: mais alternativas de comunicação entre os membros de uma equipe como o módulo de notícias que se tem para cada projeto; prioridade para cada atividade requerida; um local para armazenar documentos (atas e outros tipos de documentos); calendário para um maior controle sobre as datas; e além disso, se é possível saber exatamente como está o andamento de tarefas de cada membro, já que é informado tudo o que foi feito e alterado em cada parte do projeto. Além disso, outra funcionalidade presente no *RedMine* é o gráfico de *Gantt* que ordena as atividades de acordo com as datas de início e fim, conscientizando a equipe sobre a importância do cumprimento do prazo para que não ocorra impactos no cronograma.

Por outro lado, o BOPE SYSTEM fornece a geração de relatórios, evidenciando as

³ <<http://www.redmine.org>>

métricas relatadas anteriormente, com o intuito de exibir quantitativamente o desempenho de cada usuário, equipe e projeto. Deste modo, eles também se diferem nesse quesito, onde este presente projeto usa métricas para aperfeiçoar e evoluir o modo de se trabalhar.

2.2.2 Confluence

O software de colaboração de conteúdo *Confluence* ⁴ é escrito em Java e muito usado em ambientes corporativos, a fim de mudar a forma como as equipes trabalham. Ele é uma das melhores ferramentas de documentação existentes hoje, onde este permite a organização de toda a documentação de uma empresa que se encontra em um único local.

Ele também é um aplicativo baseado na *web* e não é *free*, mas se pode experimentar a ferramenta por sete dias sem custo algum. O Confluence resolve alguns tipos de problemas em relação a documentação, como por exemplo, documentos duplicados e várias versões deste. Para isso, esta ferramenta guarda os documentos de forma centralizada, ou seja, em um único local e, com isso, só existe uma versão para eles. Vale ressaltar ainda que, como no BOPE SYSTEM, também se é possível gerenciar uma equipe usando este aplicativo.

Esta ferramenta permite que os membros das equipes trabalhem juntos nos projetos de forma a criar, compartilhar e colaborar em um só lugar, mantendo assim, seus projetos em andamento e num ritmo mais rápido. Além disso, com ela, se torna fácil publicar, organizar e acessar as informações de projetos e equipes, revelando um trabalho mais organizado e acessível. Com tudo isso, é possível se obter uma maior produtividade e flexibilidade, uma vez que o *Confluence* trabalha de forma colaborativa e suporta a maneira como qualquer equipe funciona, de forma a atender seu tipo de necessidade.

Um outro aspecto a mencionar é que com este aplicativo se pode criar vários tipos de documentos, como por exemplo, notas de reunião, planos de projetos, requisitos de produtos e muito mais. Através dele, é possível criar espaços para equipes, departamentos ou projetos com o intuito de compartilhar conhecimento, informações e, ainda, preservar um trabalho organizado. Existem outros espaços para: informações pessoais de cada usuário; documentações para registrar, por exemplo, um serviço ou produto; base de conhecimento que foca também em documentação e boas práticas em solução de problemas.

De forma correlata ao presente trabalho, a ferramenta *Confluence* pode ajudar as equipes trabalharem de forma padronizada e organizada, documentando todas as informações relevantes do projeto. Além disso, através de tais registros, é possível ter um maior controle do prazo de entregas.

⁴ <<https://www.atlassian.com/software/confluence>>

2.2.3 *Trello*

O *Trello* ⁵ é um aplicativo de gerenciamento de projetos baseado na *web* permitindo que o trabalho seja feito de forma colaborativa. Com isso, ele se torna uma maneira flexível, gratuita, fácil e visual para organizar e priorizar seus projetos, podendo também gerenciar as tarefas e as equipes de cada projeto. Existe também o compartilhamento de informações entre os membros da equipe, atualizando-os sobre as informações relevantes de seus projetos, como por exemplo, o seu progresso.

Esta ferramenta funciona através da criação de quadros, onde estes podem ser tidos como projetos. E, dentro desses quadros, criam-se listas como se fossem listas de tarefas de cada usuário. Os cartões dentro destas listas representam as atividades que devem ser cumpridas e possuem estados que podem ser criados pelo próprio usuário. São exemplos de estados de tarefas: pendente, em andamento e realizada. Vale lembrar que, estes cartões, podem progredir de uma lista para outra, ou seja, uma atividade pendente pode migrar para a lista de tarefas realizadas. Outro aspecto relevante é o fato de que tais cartões podem conter datas, deste modo, as tarefas têm prazo para serem concluídas, tendo então um maior controle sobre os limites de entrega destas.

Assim como o BOPE SYSTEM, o *Trello* é um aplicativo gratuito para auxiliar no gerenciamento de projetos que se baseia na *web*. Ele também possui o gerenciamento de projetos, tarefas e equipes, de forma a facilitar e flexibilizar a forma de trabalhar.

2.2.4 *Pivotal Tracker*

O *Pivotal Tracker* ⁶ é uma ferramenta *web* de desenvolvimento ágil que também auxilia no gerenciamento de projetos de forma simples e colaborativa. Diferentemente dos outros trabalhos correlatos abordados anteriormente, este aplicativo calcula a velocidade da equipe em cada *Sprint*, fornece o *product backlog*, gera o gráfico *burndown* e, por fim, informa e calcula quanto tempo o colaborador está atrasado ou adiantado em um determinado projeto. Vale ressaltar que esta ferramenta é paga, mas se pode experimentá-la por trinta dias sem custo algum.

Este aplicativo oferece uma visão compartilhada das prioridades de cada equipe, promove também a colaboração entre os membros de uma equipe e oferece ferramentas dinâmicas para analisar o progresso de um determinado projeto. Além disso, é de suma importância o *backlog* compartilhado do *Tracker*, o qual permite que as prioridades dentro de um projeto se tornem claras, promovendo uma melhor organização das equipes. Outro fato importante é que o fluxo de trabalho dele ajuda cada equipe a manter o ritmo e se adaptarem quando as necessidades mudam.

⁵ <<https://www.atlassian.com/software/trello>>

⁶ <<https://www.pivotaltracker.com>>

Esta ferramenta foca no *backlog* de histórias, onde isso ajuda a equipe a se desenvolver melhor e a acompanhar seu projeto enquanto este progride. Desta maneira, ele calcula a velocidade de cada equipe com base no número médio de pontos das histórias que foram concluídas, prevendo o futuro da equipe de acordo com os dados obtidos anteriormente. Com este aplicativo, também é possível identificar o desempenho da equipe e o progresso do projeto através de gráficos e cálculos realizados, de forma a estimular um ritmo de trabalho consistente em cada equipe. Um outro fator relevante é que cada equipe pode ver claramente o trabalho que precisa ser concluído, colocando a equipe ciente das tarefas que precisam ser finalizadas, de modo que tais atividades têm suas prioridades e seus níveis de dificuldade.

Em suma, com tudo isso, a equipe se torna previsível, já que o *Tracker* calcula de forma automática quanto de trabalho a equipe precisa completar em qualquer iteração dada baseando no seu desempenho anterior. Logo, o fluxo de trabalho simples e, ao mesmo tempo, poderoso do *Tracker*, permite que toda uma equipe seja corretamente orientada, mantendo-se ágil quando as circunstâncias do projeto mudam.

Percebe-se que, assim como o BOPE SYSTEM, o *Tracker* também auxilia no gerenciamento de projetos, onde as tarefas contidas no primeiro correspondem às histórias do segundo. Tais tarefas também recebem pontuações e têm seus *status* para um maior controle do que foi e o que falta ser realizado. Ambos permitem calcular o desempenho da equipe e o progresso do projeto, o que promove uma maior consciência por parte dos integrantes do que falta ser finalizado e, ao mesmo tempo, motiva a equipe a trabalhar num ritmo adequado ao projeto em questão.

2.2.5 *Slack*

O *Slack* ⁷ é uma ferramenta onde diversas pessoas podem conversar de forma privada ou em grupos e prioriza o compartilhamento de arquivos entre os times que ficam salvos ao longo do tempo. Vale ressaltar que se pode fazer chamadas dentro do próprio aplicativo. Uma das vantagens deste *software* é manter a organização das informações em um único ambiente de trabalho. São duas as principais funcionalidades deste aplicativo: canais e integrações.

Canais são *feed* de conversas e compartilhamento de arquivos com várias pessoas de forma simultânea. Como se fossem um grupo de discussão mas que vai ser estruturado similarmente ao *feed* do *facebook*, ou seja, sempre se pode resgatar o que foi conversado.

As integrações são aplicativos de serviços externos, muitas vezes já conhecidos, que podem ser instalados dentro do *Slack* para aumentar o seu impacto, como por exemplo, *Google Agenda*, *Hangouts*, *Google Drive*, *Skype*, entre outros.

⁷ <<https://slack.com>>

Além disso, esta ferramenta, do mesmo modo que no BOPE SYSTEM, tem a visualização do perfil do usuário e dos membros de sua equipe. Deste modo, se tem o gerenciamento das equipes e o controle de tarefas através da comunicação entre os membros. Este aplicativo também é multiplataforma e

A principal diferença entre o BOPE SYSTEM e o *Slack* é essa possibilidade de instalar outros aplicativos fornecendo serviços existentes neles para esta ferramenta e, deste modo, aumentando seu potencial. Além disso, a comunicação deste *software* é abrangente e feita de forma rápida.

3 Metodologia

Neste capítulo, a metodologia utilizada na construção do sistema *web* BOPE SYSTEM é descrita, onde o planejamento do software em todas as suas etapas é relatado. Tais fases se resumem em: treinamento na linguagem de programação escolhida para a criação do sistema *web*, identificação dos requisitos do projeto, elaboração das histórias de usuário e dos cenários de teste, prototipação das telas e modelagem do sistema, implementação, testes e implantação.

3.1 Concepção e Elaboração

O presente projeto iniciou a partir do treinamento da linguagem Ruby e do *framework Rails*, de tal modo que, isso proporcionou um melhor entendimento destes e conhecimentos básicos requeridos para a implementação do sistema em questão. Dando seguimento ao projeto, na etapa de elaboração, o escopo, objetivos principais do trabalho e os requisitos foram especificados.

3.1.1 *Product Backlog*

O *product backlog* é um conjunto composto por todas as funcionalidades almejadas de um produto que descreve em alto nível o escopo do aplicativo. Na Tabela 1 a seguir, são identificados os serviços do software com seus requisitos específicos. Neste caso, o sistema foi dividido em oito partes, onde cada uma delas desenvolve um tipo de serviço com suas devidas funcionalidades.

Tabela 1 – *Product Backlog*.

SERVIÇOS	REQUISITOS ESPECÍFICOS
Serviço de gerenciamento de usuário	Autenticar usuário; Cadastrar novo usuário; Editar perfil do usuário; Excluir usuário; Visualizar perfil do usuário.
Serviço de gerenciamento da equipe	Cadastrar equipe; Editar equipe; Excluir equipe; Vincular usuário a equipe; Excluir usuário da equipe; Visualizar equipe.
Serviço de Gerenciamento de Atividades	Cadastrar atividade; Editar atividade; Excluir atividade; Visualizar atividade; Validar atividade.
Serviço de Gerenciamento de <i>Sprint</i>	Adicionar <i>Sprint</i> ; Editar <i>Sprint</i> ; Excluir <i>Sprint</i> ; Visualizar <i>Sprint</i> ; Visualizar pontos atualizados; Visualizar pontos cadastrados;
Serviço de Gerenciamento de Projeto	Cadastrar novo projeto; Editar projeto; Excluir projeto; Visualizar projeto; Vincular equipe a um projeto;
Serviço de Gerenciamento de Semana	Adicionar Semana; Editar Semana; Excluir semana; Adicionar semana ao <i>Sprint</i> ; Excluir semana do <i>Sprint</i> ; Visualizar pontos atualizados; Visualizar horas realizadas; Visualizar horas esperadas.
Serviço de controle de horas	Logar em horas no projeto; Deslogar no projeto; Visualizar porcentagem de horas no projeto na semana corrente;
Serviço de Geração de Relatórios	Gerar relatórios de dedicação relativa de cada colaborador por semana e por <i>Sprint</i> ; Gerar relatórios de acompanhamento por equipe para cada semana e <i>Sprint</i> : velocidade, burndown e dedicação relativa; Gerar relatórios de atividades gerais do laboratório e de cada projeto por <i>Sprint</i> ; Gerar relatórios de avanço e completude de cada projeto por <i>Sprint</i> ; Gerar relatórios de produtividade relativa para cada equipe por <i>Sprint</i> .

3.1.2 Histórias de usuários

Após definido o *Product Backlog*, as histórias de usuários foram criadas para cada item contido nele, tal que estas histórias apontam como o sistema deve ser operado.

Segundo Beck e Fowler (2001), a história de usuário em um ambiente ágil, é um objeto de escrita empregado no levantamento de requisitos para relatar a especificação de uma funcionalidade do sistema. Vale ressaltar que essas histórias descrevem de forma sucinta as necessidades do software, fornecendo às equipes, a capacidade de resposta rápida sobre o que o usuário deseja e necessita.

Como sugerida por Cohn (2004), a sintaxe de uma história de usuário necessita conter o papel, a funcionalidade e o objetivo como mostrado a seguir:

- COMO UM (papel ou ator - Quem?)
- EU GOSTARIA (funcionalidade necessária ou capacidade - O que?)
- PARA CONSEGUIR (objetivo - Por que?)

A história de usuário para o cadastro de usuários é mostrada abaixo como exemplo:

Serviço de Gerenciamento de Usuários

1. Cadastrar novo usuário

- COMO um administrador cadastrado
- EU GOSTARIA DE cadastrar novos usuários fornecendo nome, senha e tipo de usuário
- PARA CONSEGUIR adicionar usuários ao sistema para que estes tenham acesso às funcionalidades do sistema

As demais histórias para todos os outros serviços foram documentadas em uma ferramenta chamada *DokuWiki* que é utilizada no iMobilis e, estas são mostradas no Apêndice A deste presente trabalho.

3.1.3 *Cenários de teste*

Depois de concluir as histórias de usuário, se pode testar as suas funcionalidades através da criação dos cenários de testes para a validação destas. Resumidamente, o cenário pode ser considerado uma narrativa que irá ilustrar como o software que foi planejado atenderá a necessidade do usuário. De maneira geral, se pode dizer que estes cenários descrevem o comportamento do sistema, onde eles devem ser coerentes com a linguagem das histórias de usuário criadas anteriormente.

A sintaxe usada para representar os cenários é descrita a seguir:

- DADO QUE (pré-condição)

- QUANDO (evento)
- ENTÃO (pós-condição)

Percebe-se que essas expressões apoiam a criação dos cenários de forma que eles têm que ser independentes uns dos outros e, além disso, se pode ter cenários alternativos para a mesma história de usuário. Adicionalmente, as expressões E, OU e MAS podem ser utilizadas para aperfeiçoar o cenário de teste.

A seguir, são apresentados os cenários de teste para o cadastro de usuários:

1. Cadastrar novo usuário

- Cenário principal
 - DADO QUE um administrador forneça nome, senha e tipo de usuário
 - QUANDO o botão “criar usuário” for pressionado
 - ENTÃO o usuário é cadastrado com sucesso
- Cenário alternativo
 - DADO QUE um administrador clique no botão “novo usuário”, mas ele desiste de criá-lo por algum motivo
 - QUANDO o botão “cancelar” for pressionado
 - ENTÃO o administrador é redirecionado para a tela anterior de usuários cadastrados

Os outros cenários de testes para todos os outros serviços foram documentados também na *DokuWiki* e, estas são mostradas no Apêndice A juntamente com as histórias de usuário.

3.1.4 Protótipos de tela

A prototipação facilita o entendimento do software que vai ser criado, de forma a apoiar a definição dos requisitos, apresentação dos conceitos e funcionalidades do sistema. Através disso, se pode propor uma versão mais fiel do software em questão. A prototipação é muito importante uma vez que é uma forma mais rápida, econômica e eficiente de se definir e experimentar um projeto.

Portanto, protótipos de telas do BOPE SYSTEM foram criados durante a concepção do sistema para se ter uma idéia de como será a interação real do usuário com este, onde os detalhes deste software são evidenciados. Com isso, essas telas tem como objetivo prever o máximo de situações possíveis que o usuário pode experimentar durante a utilização

Início	Atividade 1
Usuários	
Projetos ✓	Descrição:
Equipes	Pontos:
	Label:
	Status:
	Editar Excluir Voltar

Início	Adicionar atividade
Usuários	
Projetos ✓	Descrição:
Equipes	Pontos:
	Label:
	Status:
	Adicionar Cancelar

Figura 2 – Protótipo da tela de Visualização e Criação de Atividade

do sistema *web*. Sendo assim, de forma geral, esses protótipos revelam a simulação da interação entre o usuário e o aplicativo.

Essas representações foram desenhadas à mão de maneira rápida e superficial, feitas em paralelo com as histórias de usuário e cenários de teste para ajudar na identificação de funcionalidades do sistema. A seguir, as telas de visualização de atividade e de criar atividade são mostradas na Figura 2.

Como apresentada anteriormente, a tela de visualização de atividade mostra as informações sobre cada atividade: descrição, pontos, *label* e *status*. Além disso, se pode editá-la ou excluí-la. Posteriormente, na tela de criação de atividade, se deve preencher as mesmas informações contidas na tela anterior.

3.1.5 Projeto de Banco de Dados

A partir das informações obtidas anteriormente nas histórias de usuários, cenários de testes e protótipos de telas, é possível desenvolver um diagrama para conduzir o desenvolvimento do sistema. Este diagrama é o ER ou DER (Diagrama Entidade-Relacionamento) que pode ser descrito como um tipo de fluxograma utilizado na Engenharia de Software para representar os objetos (entidades) contidos no software, com suas características (atributos) e a forma como eles se relacionam uns com os outros (relacionamentos).

As entidades ou objetos são as partes envolvidas no sistema, sendo nomeadas com substantivos abstratos ou concretos que irão representar de forma objetiva e clara sua função dentro do software. Os atributos servem para descrever e caracterizar as entidades. Por fim, os relacionamentos definem como os objetos se relacionam entre si, sendo classificados em três tipos: relacionamentos “um para um”, “um para muitos” e (muitos para muitos).

O relacionamento 1..1 (um para um) é onde uma entidade se relaciona com outra e elas referenciam apenas uma unidade da outra. Já no relacionamento 1..n (um para muitos) também é quando duas entidades se relacionam, mas enquanto uma referencia apenas uma unidade da outra, essa outra referencia várias unidades da primeira. E por último, o relacionamento n..n (muitos para muitos) ambas as entidades referenciam várias unidades umas das outras.

O diagrama entidade-relacionamento do BOPE SYSTEM pode ser visualizado a seguir:

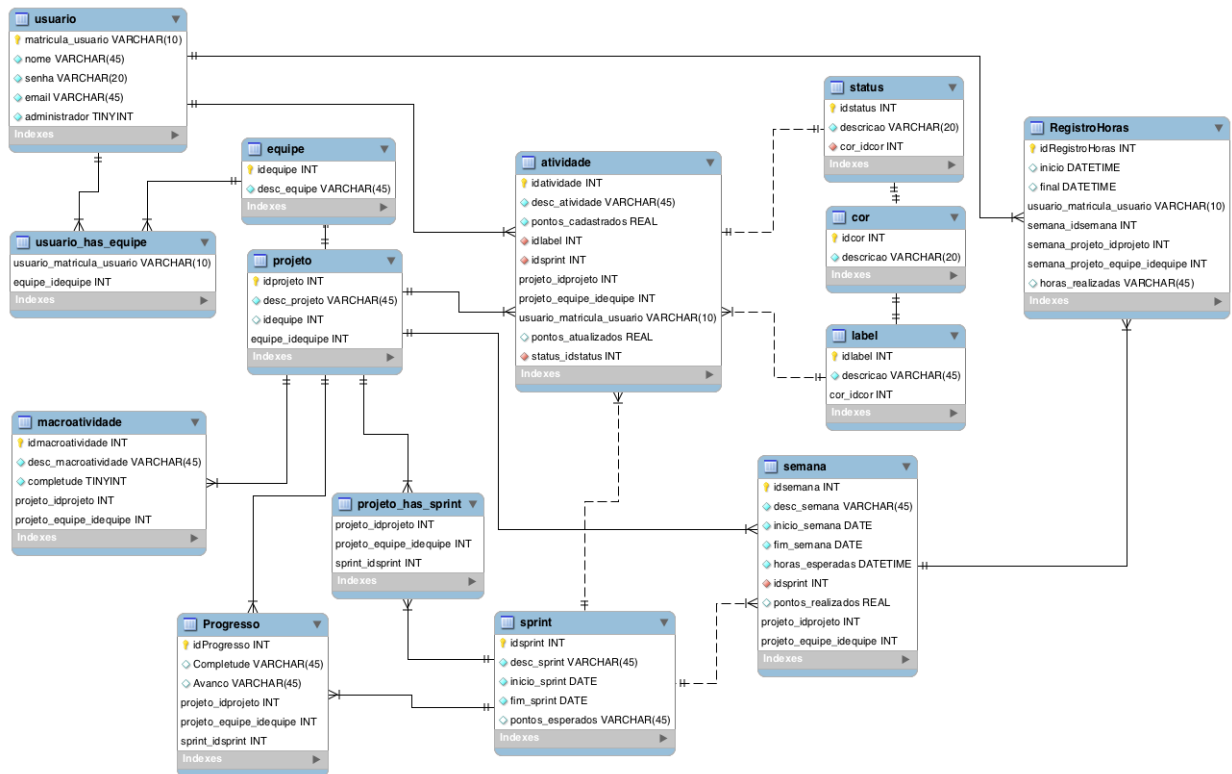


Figura 3 – Diagrama ER do sistema BOPE SYSTEM

3.2 Implementação

Durante o desenvolvimento do sistema *web* foram usadas as versões 2.4.0 do *Ruby* e 5.0.2 do *Rails*. Além disso, o banco de dados *PostgreSQL* foi aplicado ao sistema em sua versão 9.6.2. É importante salientar também que o aplicativo foi desenvolvido no sistema operacional *Mac OS X Yosemite* versão 10.10.5 e o editor de texto *ATOM*¹ em sua versão 1.18.0 foi utilizado.

3.2.1 Criação do Projeto

Posteriormente, com a linguagem *Ruby* e *framework Rails* instalados corretamente, juntamente com o *postgresql*, foi necessário criar o projeto para iniciar a implementação do sistema. Deste modo, a aplicação foi criada com o seguinte comando:

```
rails new BOPE --database=postgresql
```

Após o comando acima, a estrutura padrão do *Ruby on Rails* foi criada automaticamente, onde se tem uma série de arquivos (pt-br) e pastas dentro do projeto, como mostra a figura a seguir:

¹ <<https://atom.io>>

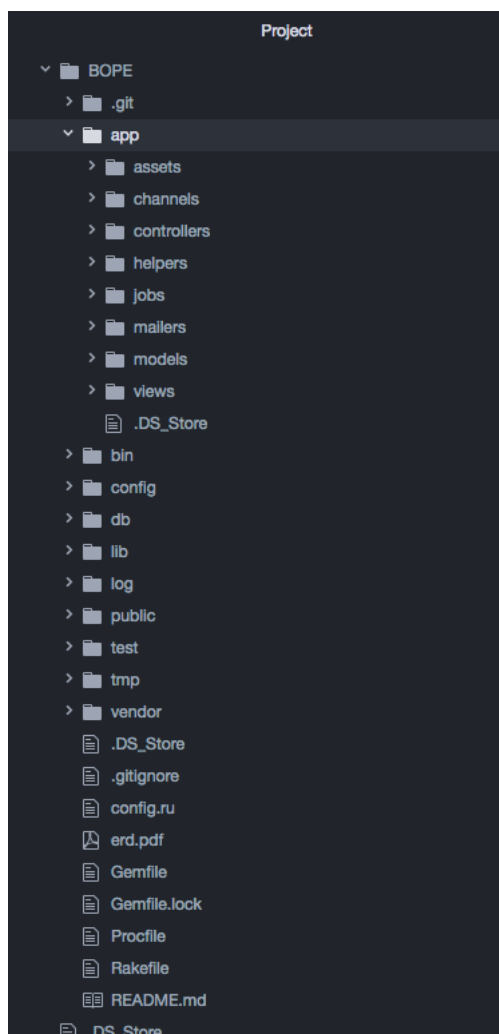


Figura 4 – Estrutura das pastas e ficheiros geradas automaticamente no Rails

Com isso, foi necessário acessar a pasta desta aplicação criada através do seguinte comando:

```
cd BOPE
```

Após isso, foram instaladas todas as dependências contidas na pasta Gemfile (onde ficam as Gems) digitando no terminal:

```
bundle install
```

Abaixo, são relatadas as pastas e arquivos presentes na imagem anterior, juntamente com seus propósitos:

- `app/`: Esta pasta é aquela onde o desenvolvedor vai acessar a maior parte do tempo, pois é nela que se tem os *controllers*, *models*, *views*, *helpers*, *mailers* e *assets* (contém

os arquivos CSS e JavaScript) da aplicação.

- bin/: Pasta que contém os *scripts* do rails e onde pode ser adicionado outros scripts que o desenvolvedor necessitar durante o desenvolvimento da aplicação.
- config/: Onde se configuram as rotas, banco de dados do aplicativo, ambientes, arquivos de inicialização, arquivos de internacionalização da língua Portuguesa, entre outros.
- config.ru: *Rack* de configuração para *Rack* de Servidores, utilizados para dar início ao aplicativo.
- db/: O esquema e as migrações do banco de dados ficam nesta pasta, além do arquivo *seeds.rb* que pode ser usado para preencher informações iniciais no banco de dados.
- lib/: São os módulos de extensão para sua aplicação e tarefas que se deseja gerar de forma automática no banco de dados.
- log/: Arquivos de log da aplicação.
- public/: Armazena arquivos estáticos e CSS/JS compilados, sendo que esta é a única pasta que pode ser acessada via *browser*.
- test/: A pasta padrão para testes de unitários, fixtures, entre outros tipos de testes no *Rails*.
- tmp/: Onde se localizam os arquivos temporários, como por exemplo, arquivos de *cache*, *pid*, entre outros.
- vendor/: Onde estão armazenados códigos e arquivos de terceiros. No caso deste presente trabalho, é onde se encontra a logo do iMobilis e outras imagens usadas na aplicação, além dos arquivos do *bootstrap* para colocá-lo de forma estática. Nesta pasta também pode se encontrar gems terceirizadas.
- Gemfile/: Um arquivo que contém todas as Gems utilizadas no desenvolvimento da aplicação, como por exemplo, *bootstrap*, *devise-i18n*, *foreman*, *rails*, entre várias outras.
- Gemfile.lock: Arquivo que permite especificar quais gems são necessárias para a sua aplicação Rails.
- Rakefile: Arquivo que localiza e carrega as tarefas que podem ser realizadas a partir da linha de comando.

Deve-se salientar que a configuração do banco de dados PostgreSQL é feita no arquivo `database.yml` dentro da pasta `config`, como mostra a figura a seguir:

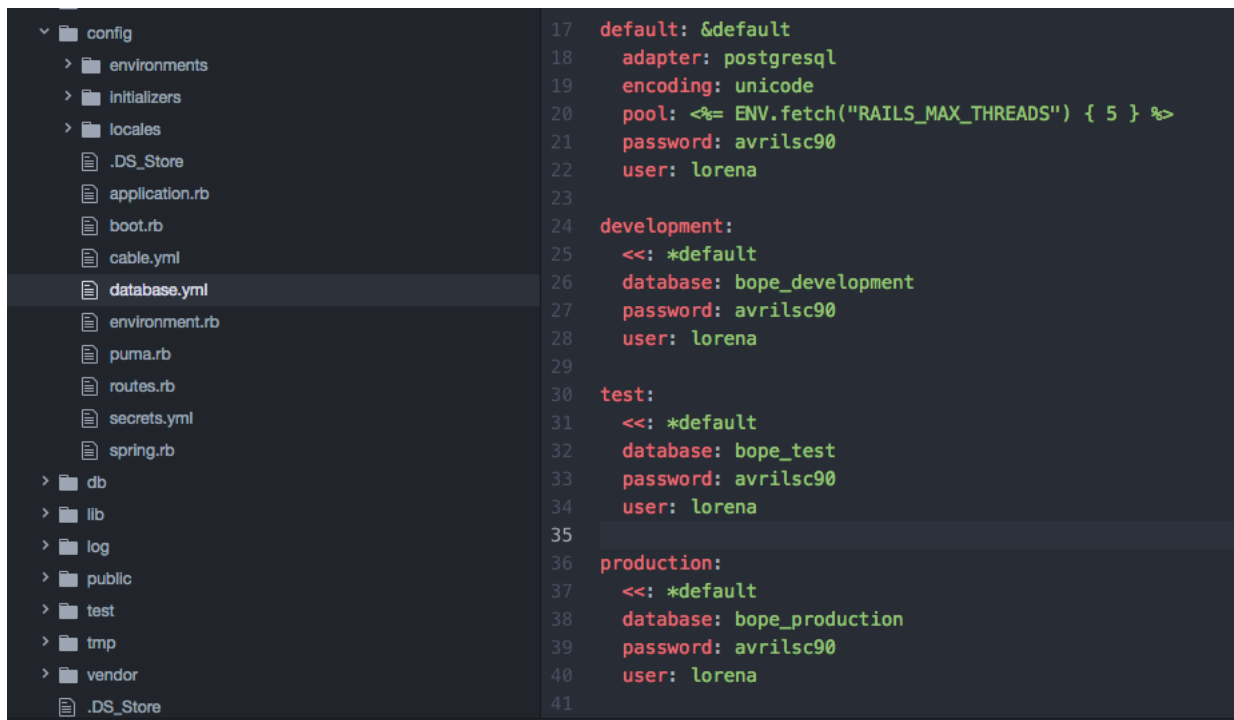


Figura 5 – Configuração do Banco de Dados PostgreSQL

De acordo com a imagem anterior, o banco de dados é configurado com o nome de usuário “lorena” e a senha “avrilsc90”. É importante observar que essa configuração é feita para o ambiente de desenvolvimento, teste e produção.

Depois da configuração do banco de dados, se fez necessário apagar todas as informações do mesmo e depois criar, migrar as novas informações deste. E, por último, o sistema foi inicializado com um administrador padrão para ter acesso ao aplicativo. Os comandos usados para isso são listados a seguir:

```
rails db:drop
```

```
rails db:create
```

```
rails db:migrate
```

```
rails db:seed
```

É importante saber como criar cada tabela no banco de dados com seus respectivos atributos. Neste caso, é mostrado a seguir um exemplo de como criar a tabela usuário e seus atributos:

```
rails g model user name:string student_number:string
```

Observa-se que foi criado um modelo (correspondente a uma tabela no banco de dados) com nome `user_profile` contendo seus atributos `name` e `student_number`, onde

ambos são do tipo string. Com isso, esse atributo é adicionado à essa tabela de forma automática. Para que tal tabela seja adicionada ao banco, foi necessário migrá-la através do comando *migrate* abordado anteriormente.

Desta forma, observou-se que o modelo criado apareceu no diretório `app/models` e, além disso, foram criadas uma unidade de teste e migração para ele, como visto na Figura 6 a seguir:

```
1 class UserProfile < ApplicationRecord
2   # Associations
3   enum role: { :Total => 0, :Parcial => 1 }
4   belongs_to :user, optional: true
5
6   validates_presence_of :name, :student_number
7   validates_length_of :student_number, maximum: 10, allow_blank: false
8
9   #paperclip
10  # "#" instead ">": cut from the middle
11  has_attached_file :picture, #styles: { medium: '300x300#', thumb: '100x100#' },
12                        default_url: "usuario_padrao.png" #configura tamanho da image
13  validates_attachment_content_type :picture, content_type: /\Aimage\/.*\z/ #ace
14
15 end
```

Figura 6 – Modelo *user_profile*

Vale lembrar que um modelo também pode ser criado usando como atributo a referência de outra tabela. Por exemplo, ao criar o modelo *status*, um dos seus atributos é a cor e, deste modo, se pode usar a tabela *color* como sua referência, pegando o seu *id*. Para isso, o comando abaixo foi utilizado:

```
rails g model status description color:references
```

No comando acima, é importante observar que não foi definido um tipo para o atributo descrição, uma vez que, por padrão, o *Rails* estabelece o tipo *string*.

Vale ressaltar que se pode inserir novos atributos para um modelo já criado, uma vez que se pode verificar a necessidade de mais atributos conforme o desenvolvimento do projeto. Deste modo, o exemplo a seguir mostra a adição do atributo *access* feito para a tabela *user*, identificando se o acesso do usuário é parcial ou total.

```
rails g migration AddAccessToUser access:string
```

Por outro lado, se o desenvolvedor necessita deletar um atributo de alguma tabela, o comando será:

```
rails g migration RemoveAcessoToUser access:string
```

Depois da criação do modelo, foi necessário adicionar um *controller* para ele. Então, o comando a seguir foi utilizado:

```
rails g controller statuses index show new edit
```

Observa-se que, após a inserção do comando acima, o controlador para o *status* foi criado no diretório `app/controller`, como na figura abaixo:

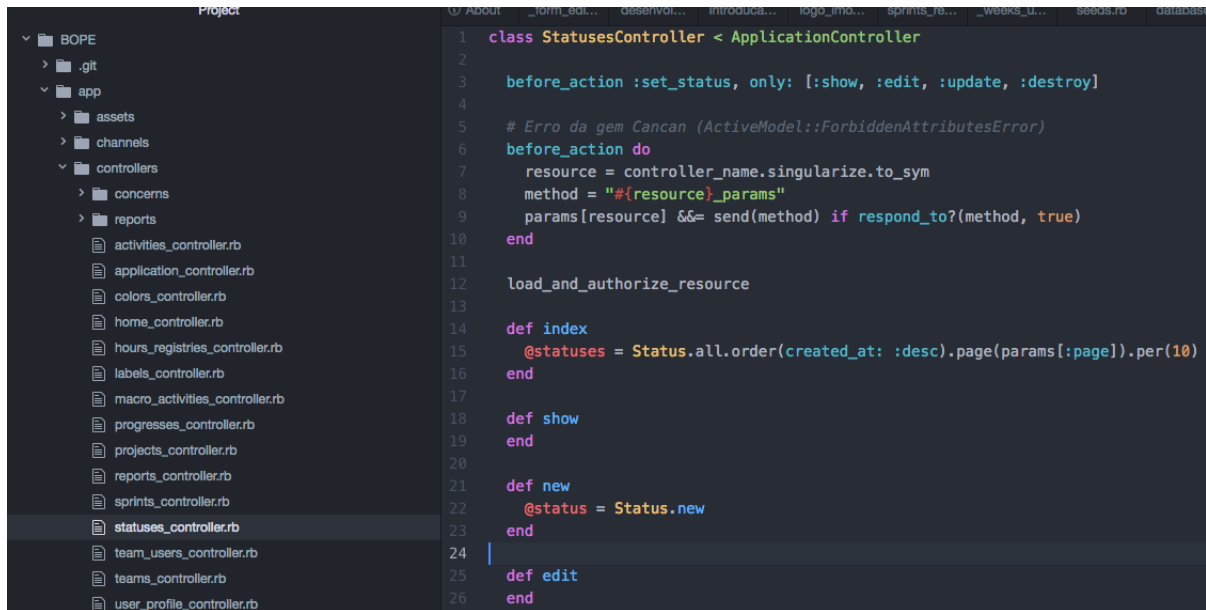


Figura 7 – Controlador do *status*

É importante salientar que o *controller* foi criado com quatro ações: *index*, *edit*, *show* e *new*. Enquanto o *index* permite a visualização da lista de todos os *status*, o *show* exibe apenas um *status* específico. A ação *new* permite o retorno de um formulário HTML para criar um novo *status*. E, por fim, o *edit* retorna um formulário HTML para alterar um *status*. Além dessas ações, outras são usadas: *create*, *update* e *destroy*. O *create* cria um novo *status*, o *update* atualiza um *status* específico e o *destroy* tem a função de destruir um determinado *status*.

Os comandos, anteriormente listados, foram usados em todas as tabelas do sistema: *activity*, *burndown*, *color*, *hours_registry*, *label*, *macro_activity*, *progress*, *project_sprint*, *project*, *sprint*, *status*, *team_user*, *team*, *user_profile*, *user*, *users_team* e *week*. É de suma importância observar que algumas tabelas foram criadas pelo fato de outras duas tabelas representarem uma relação “muitos para muitos”. Um exemplo disso é a tabela *users_team*, uma vez que um usuário pode pertencer a várias equipes e uma equipe tem vários usuários. Outro aspecto importante a ser notado é o fato de que as informações do usuário foram divididas em duas tabelas: *users_profile* e *user*, onde a primeira contém o nome e número de matrícula do colaborador, enquanto que a segunda possui o e-mail, senha e confirmação da senha. Com isso, é necessário unir as informações de uma com a outra para apresentar o perfil completo do usuário. E, para isso ocorrer, criou-se uma associação *belongs_to :user*, onde esta faz a tabela *user_profile* pertencer a *user*. Esses tipos de associações serão abordados posteriormente neste presente trabalho.

3.2.2 Autenticação e autorização de usuários

A autenticação no sistema foi feita através da Gem *devise*, onde esta permitiu o *login* e *logout* dos usuários cadastrados. Vale ressaltar que o registro e exclusão dos usuários são realizados pelo administrador padrão do sistema que, como dito anteriormente, foi criado de forma automática através do arquivo *seed.rb*.

Os módulos dessa Gem que foram utilizados na construção do software são citados abaixo:

- *Database Authenticatable*: Através dele se pode armazenar uma senha para cada usuário no banco de dados com o intuito de validar a autenticidade ao logar no sistema.
- *Rememberable*: permite salvar o usuário e sua senhas de login.

A figura abaixo, apresenta a autenticação do usuário feita através da Gem *devise*:

```
1 class User < ApplicationRecord
2   EMAIL_REGEX = /\A[^\s@]+@[^\s@]+\.[^\s@]+\z/
3   # Include default devise modules. Others available are:
4   # :confirmable, :lockable, :timeoutable and :omniauthable
5   devise :database_authenticatable, :registerable,
6         :recoverable, :rememberable, :trackable, :validatable
7
8   # Associations
9   has_one :user_profile, dependent: :destroy
10  has_many :team_users, dependent: :destroy
11  has_many :teams, through: :team_users
12  has_many :activities, dependent: :destroy
13  has_many :hours_registries, dependent: :destroy
14
15  # Validations
16  validates_presence_of :email, :password, :password_confirmation
17  validates_confirmation_of :password
18  validates_format_of :email, with: EMAIL_REGEX
19
20  # Nested Attributes
21  accepts_nested_attributes_for :user_profile, allow_destroy: true # f
22
23 end
```

Figura 8 – Autenticação do usuário

Logo após a autenticação, a autorização no *Rails* foi feita através da Gem *CanCan*, onde esta define as permissões e restrições dos usuários. De modo geral, ela determina quais recursos, presentes no sistema, os usuários poderão acessar. No caso do atual trabalho, existem dois tipos de usuários: administrador e colaborador. O colaborador é aquele que tem acesso parcial do sistema, enquanto que o administrador tem o acesso total. Deste modo, o sistema foi implementado de forma a evidenciar essas duas visões. É importante saber

que essas permissões e restrições são definidas em um único local (app/models/ability.rb) para que elas não sejam replicadas no código. Essa classe *ability* é mostrada na Figura 9:

```
1 class Ability
2   include CanCan::Ability
3
4   def initialize(user)
5     # Define abilities for the passed in user here. For example:
6     #
7     # user ||= User.new # guest user (not logged in)
8     if user.user_profile.role == "Total"
9       can :manage, :all
10    else
11      can [:read, :update], UserProfile, id: user.id
12      # can :update, UserProfile, id: user.id
13      can :read, Team
14      can :read, Project
15      can :read, MacroActivity
16      can :read, Sprint
17      can :manage, Activity
18    end
19    #
20    # The first argument to 'can' is the action you are giving the user
```

Figura 9 – Autorização do usuário

É preciso citar que, o usuário do tipo colaborador, não poderá criar ou excluir nada no sistema, com exceção de suas próprias atividades. Além disso, ele poderá alterar suas informações de perfil, como por exemplo, e-mail, senha, nome, foto e matrícula. Por outro lado, o administrador poderá usufruir de todos os recursos do sistema. Vale ainda lembrar que a visão de colaborador só permite que ele acesse a barra de projetos, equipe, início e o próprio perfil, sendo restrito de ter acesso às definições de cores, estados, rótulos e relatórios. A seguir, são exibidas ambas as visões:



Figura 10 – Visão Geral do sistema do usuário colaborador



Figura 11 – Visão Geral do sistema do usuário administrador

3.2.3 Associação de tabelas no Rails

No Rails, se faz necessário a associação dos modelos criados, estabelecendo o tipo de relação entre eles. Os tipos de associações existentes são: *belongs_to*, *has_one*, *has_many*, *has_one :through*, *has_many :through* e *has_and_belongs_to_many*.

A associação *belongs_to* representa a conexão de um para um de um modelo com outro, onde cada instância do modelo em que se declara este tipo de associação “pertence” a uma instância do outro modelo. Um exemplo disso é *status* que tem a instância da tabela *color* como mostra a figura seguir:

```
1 class Status < ApplicationRecord
2   belongs_to :color
3   has_many :activities, dependent: :nullify
4 end
5
```

Figura 12 – Associação *belongs_to* entre *status* e *color*

Outro tipo de associação é a *has_one* que também permite a conexão de um para um com outro modelo, mas ela indica que cada instância de um modelo contém ou possui uma instância de outro modelo. Por exemplo, uma equipe tem um projeto como apresentado na figura abaixo:

```
1 class Team < ApplicationRecord
2   has_many :team_users, dependent: :destroy
3   has_many :users, through: :team_users
4   has_one :project, dependent: :destroy # adicionado
5
6   #has_and_belongs_to_many :users
7   validates_presence_of :description
8 end
9
```

Figura 13 – Associação *has_one* entre *team* e *project*

Por outro lado, a associação *has_many* indica uma conexão de um para muitos com outro modelo, onde cada instância do modelo tem uma ou mais instâncias de outro modelo. Um exemplo disso é encontrado na relação entre *label* e *activities*, onde uma mesma *label* pode ser encontrada em diversas atividades.

```
1 class Label < ApplicationRecord
2   belongs_to :color
3   has_many :activities, dependent: :nullify
4 end
5
```

Figura 14 – Associação *has_many* entre *label* e *activities*

A diferença do *has_one* e *has_one :through* se dá pelo fato de que, este último, indica que a instância do modelo declarante pode ser combinado com uma instância de um outro modelo através de um terceiro modelo. Vale ressaltar que esse tipo de associação não foi usada nesse sistema.

Do mesmo modo, a diferença do *has_many* e *has_many :through* se dá pelo fato de que, este último, indica que a instância do modelo declarante pode ser combinado com uma ou mais instâncias de um outro modelo através de um terceiro modelo. Na Figura 13, mostrada em um dos exemplos anteriores, se observa esse tipo de associação, onde uma equipe tem vários usuários através da tabela *team_users*.

Por fim, a associação *has_and_belongs_to_many* cria uma conexão direta de muitos para muitos com outro modelo, sem modelo intermediário. É importante saber que esse tipo de associação não foi usada nesse sistema.

3.2.4 Geração de PDF

A geração de PDF se faz necessário devido aos relatórios requeridos com seus respectivos gráficos para apresentar as métricas estabelecidas no processo BOPE. Nestes relatórios são especificadas as seguintes métricas: dedicação relativa de cada usuário e equipe; velocidade, *burndown* e produtividade relativa de cada equipe; e além disso, a completude e avanço de cada projeto dentro do iMobilis.

Na geração de PDF a gem Wicked PDF foi utilizada para que fosse possível gerar os relatórios requeridos pelo sistema. Esta gem contém o utilitário *wkhtmltopdf-binary* que também é considerado uma gem. Neste caso, um documento HTML é enviado para *wkhtmltopdf* que o converte em um documento PDF, de modo que este pode ser baixado pelo usuário. E, por fim, para gerar os gráficos foi utilizada a gem ZingChart, onde esta permite a visualização dos dados em aplicativos *web*.

3.2.5 Testes

Os testes no sistema foram feitos durante todo o processo de desenvolvimento e, principalmente, ao final da implementação com todas as funcionalidades prontas.

3.3 Implantação do sistema no laboratório iMobilis

A implantação do sistema *web* foi feita após vários testes realizados no software e a documentação referente ao seu tutorial encontra-se no Apêndice B deste trabalho. Neste tutorial, são apresentados todos os passos necessários para tal implantação no servidor Mariana dentro do laboratório iMobilis.

4 Apresentação dos Resultados

Neste capítulo, serão mostradas as telas do sistema, assim como os relatórios gerados por ele após a sua implantação no laboratório iMobilis. As telas iniciais mostram o sistema na visão do administrador e, posteriormente, na visão do colaborador.

Inicialmente, tem-se a tela de *login* do sistema como mostrada a seguir:

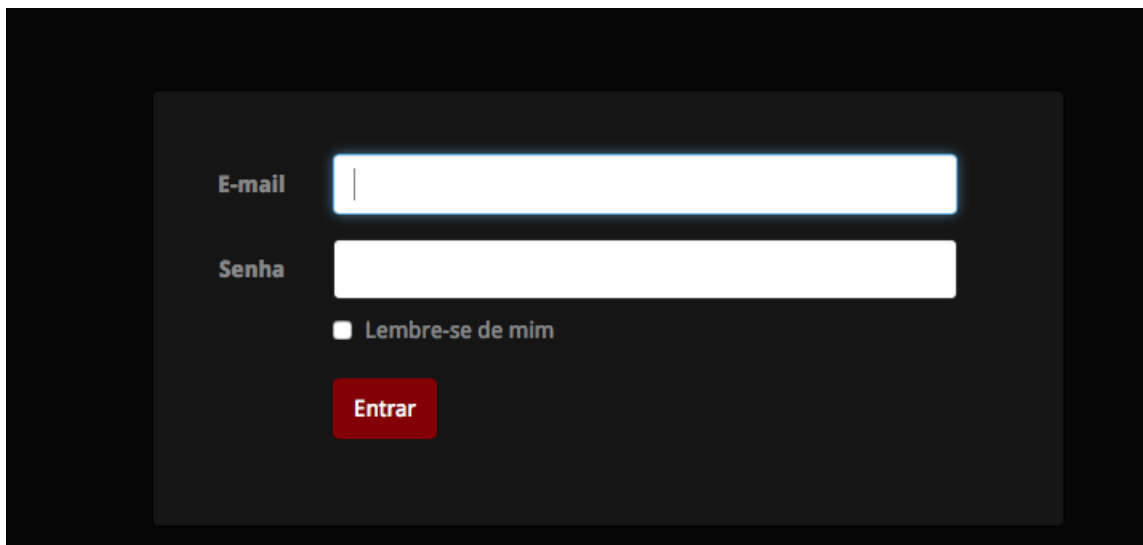


Figura 15 – Tela de login do sistema

Ao fazer o *login* no sistema, se tem tela inicial do sistema com sua breve descrição como mostrada na figura abaixo. Além disso, as opções de sair do sistema e visualizar o perfil também são mostradas.



Figura 16 – Visão do administrador com breve descrição do sistema

A seguir, a tela de visualização de todos os usuários cadastrados no sistema é mostrada ao clicar na aba “Usuários”. Com isso, o administrador tem as opções de criar novo usuário, apagar, editar ou visualizar um usuário específico.

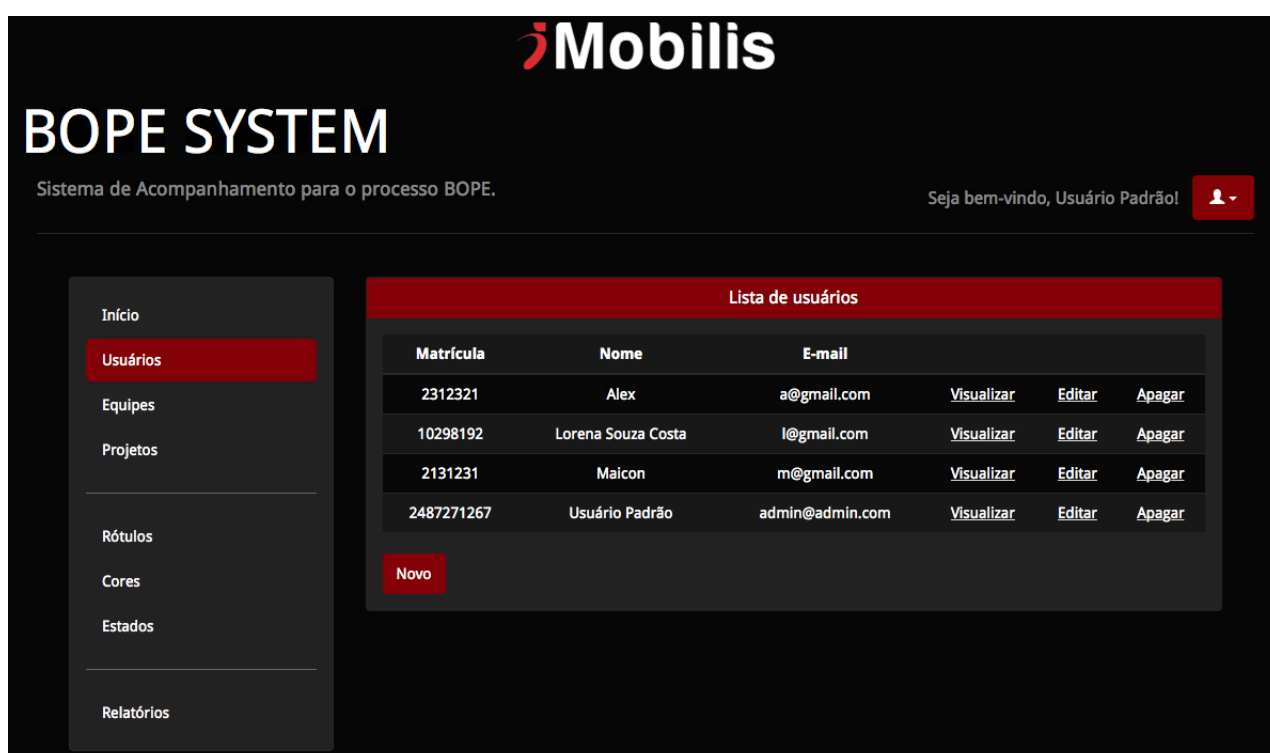


Figura 17 – Visualização para o administrador da lista de usuários

Ao clicar no botão “Novo”, a tela abaixo é apresentada, onde o administrador deve

fornecer o e-mail, senha, confirmação da senha, matrícula, nome, acesso e foto do perfil para o usuário a se criar.

Figura 18 – Visualização para o administrador da criação de um usuário

Ao clicar no link “Visualizar”, o perfil de um usuário específico é apresentado como na imagem seguinte:

Figura 19 – Visualização para o administrador do perfil de um usuário específico]

Ao clicar no link “Editar”, o perfil de um usuário específico pode ser editado como na figura abaixo:

Seja bem-vindo, Usuário Padrão!

Editar usuário

E-mail:

Senha:

Confirme sua senha:

Matrícula:

Nome:

Acesso:

Foto de perfil: no file selected

Figura 20 – Visualização para o administrador da edição de um usuário específico

Ao clicar no link “Excluir”, uma janela aparece para o administrador para que ele confirme a exclusão de um usuário específico. Isso é mostrado abaixo:

Sistema de Acompanhamento para o processo BOPE. Seja bem-vindo, Usuário Padrão!

Tem certeza que deseja excluir?

2312321	Alex	a@gmail.com	Visualizar	Editar	Apagar
10298192	Lorena Souza Costa	l@gmail.com	Visualizar	Editar	Apagar
2131231	Maicon	m@gmail.com	Visualizar	Editar	Apagar
2487271267	Usuário Padrão	admin@admin.com	Visualizar	Editar	Apagar

Figura 21 – Visualização para o administrador da exclusão de um usuário específico

Ao selecionar a aba “Equipes”, a listagem de todas as equipes cadastradas no sistema são apresentadas como mostra na imagem a seguir:



Figura 22 – Visualização para o administrador da lista das equipes cadastradas

Da mesma forma que ocorre com os usuários, se pode criar uma nova equipe, visualizar, editar ou excluir uma equipe específica. As telas de criação, visualização de uma equipe e edição de usuários de uma equipe são mostradas a seguir:

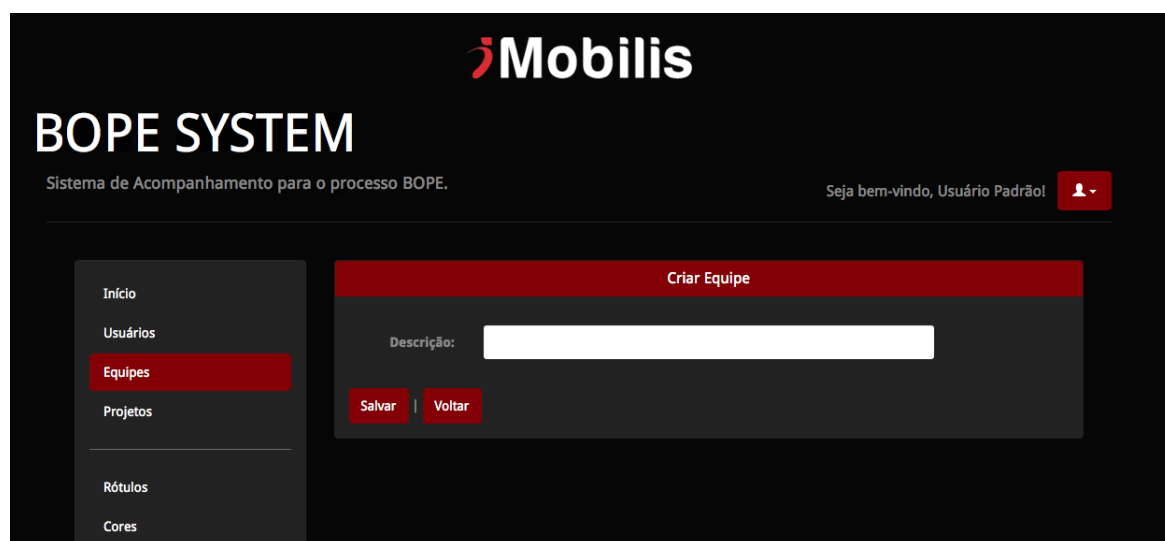


Figura 23 – Visualização para o administrador da criação de uma equipe

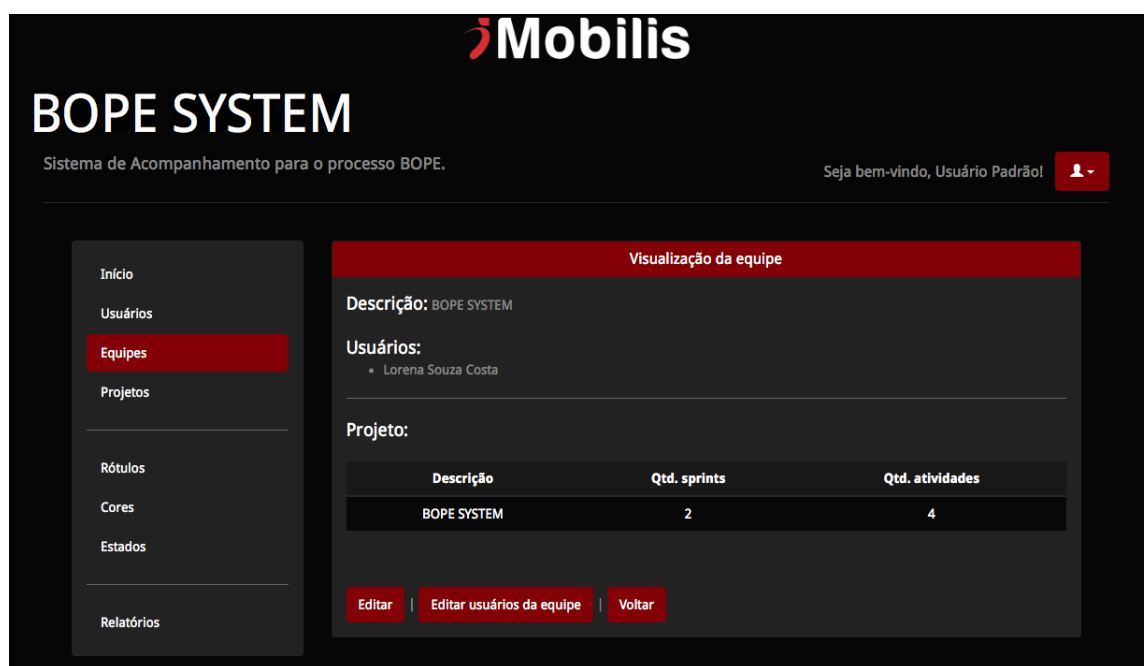


Figura 24 – Visualização para o administrador de uma equipe específica

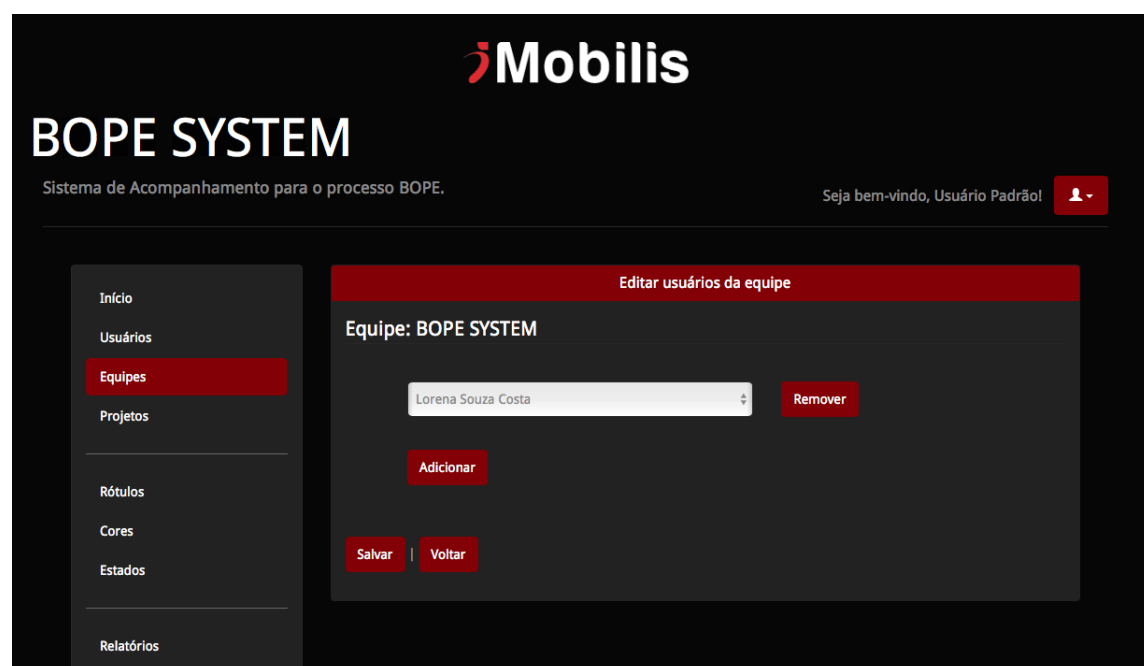


Figura 25 – Visualização para o administrador da edição dos usuários de uma equipe específica

Ao clicar na aba “Projetos”, listam-se todos os projetos cadastrados e também podem criar novos projetos, assim como excluir, alterar ou visualizar um projeto específico. Deste modo, a tela de visualização de projetos é apresentada abaixo:

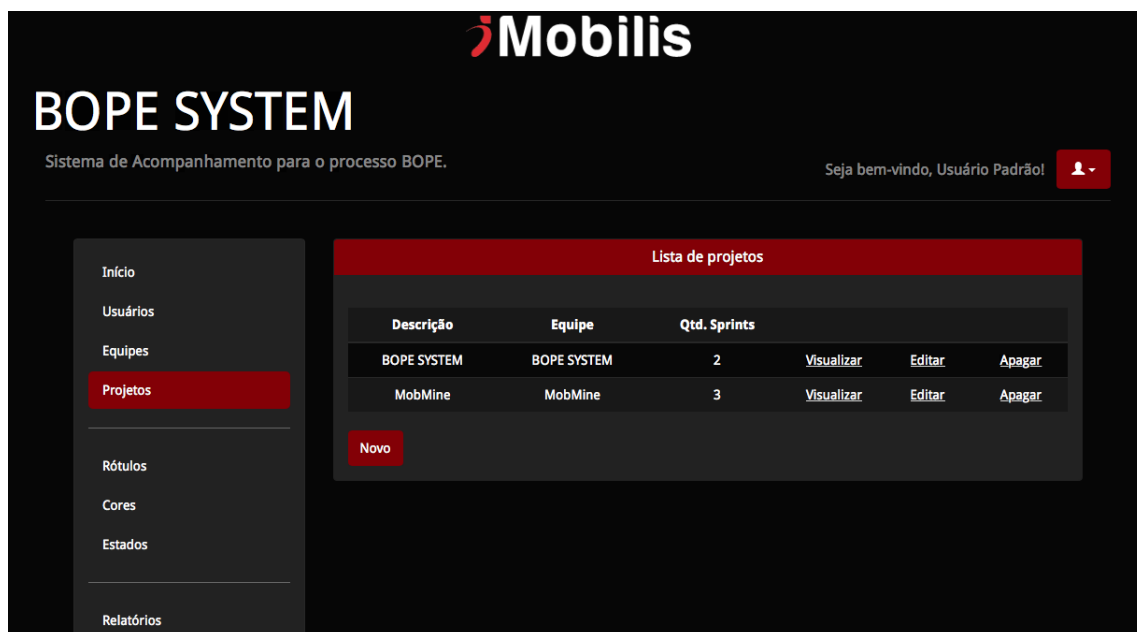


Figura 26 – Visualização para o administrador da lista de projetos cadastrados

Ao pressionar o botão de “Novo”, o administrador deseja criar um novo projeto como mostra a seguir:

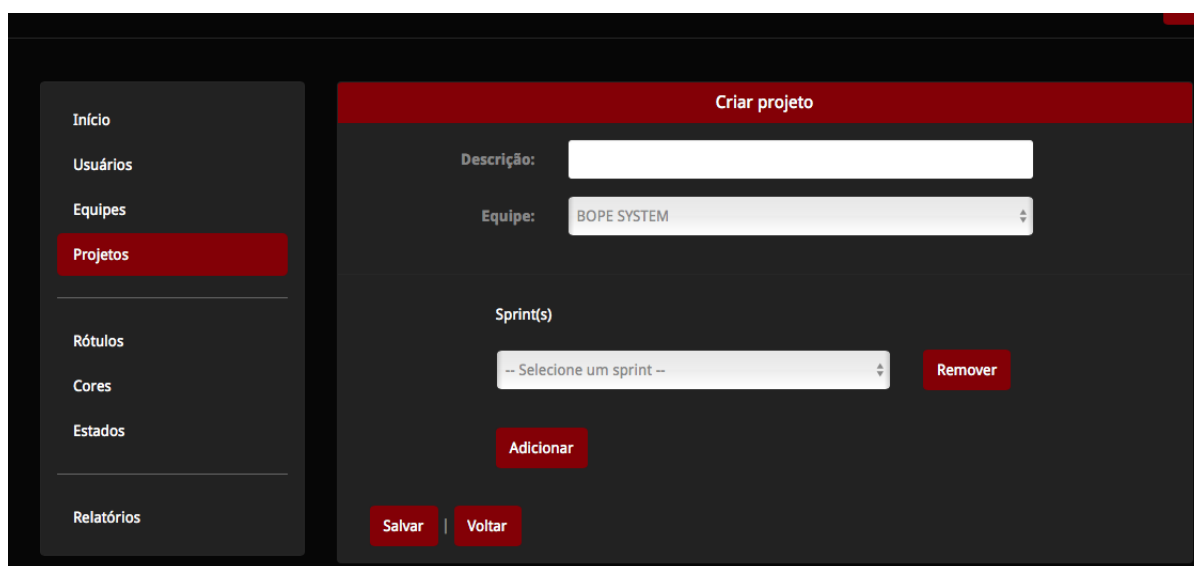


Figura 27 – Visualização para o administrador da criação de um novo projeto

Ao visualizar um projeto, são mostradas as informações do progresso, dos *Sprints* e macro atividades deste como na seguinte figura:

Figura 28 – Visualização para o administrador de um projeto e suas informações

Percebe-se que se pode criar um novo *Sprint* e macro atividade, assim como editar, excluir ou visualizar um *Sprint* ou macro atividade existente. Também é possível visualizar o progresso de um projeto, assim como criar uma nova completude ou apagar alguma existente, como é visto na imagem a seguir:

Figura 29 – Visualização para o administrador do progresso de um projeto

Ao selecionar a opção de visualizar um *Sprint*, são mostrados os *Sprints* daquele projeto juntamente com suas semanas e atividades dos colaboradores de acordo com a seleção de tal. A figura a seguir apresenta a visualização dos *Sprints*:



Figura 30 – Visualização para o administrador da listagem dos *Sprints* dentro de um projeto

Percebe-se que se pode criar uma nova atividade, assim como editar, excluir ou visualizar uma atividade já existente. Além disso, é permitido apenas editar uma semana de forma que só as horas esperadas e realizadas podem ser modificadas. Vale lembrar que, quando se cria um *Sprint*, estabelecendo início e fim para este, as semanas são automaticamente criadas.

Ao selecionar a aba “Rótulos”, “Cores” e “Estados”, são apresentadas as imagens a seguir:

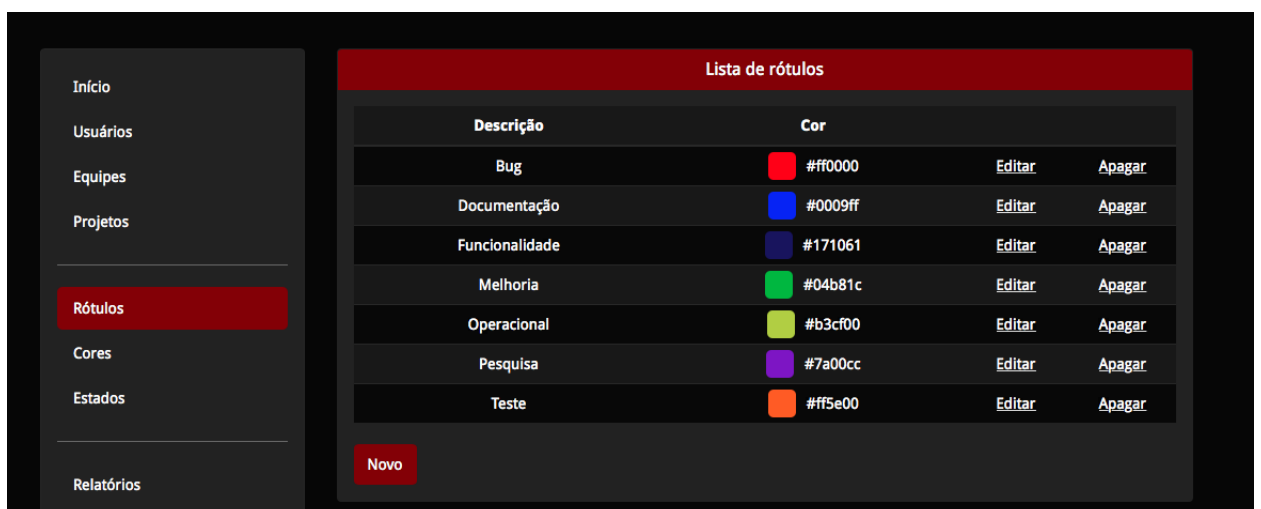


Figura 31 – Visualização para o administrador da listagem dos rótulos para caracterizar uma atividade

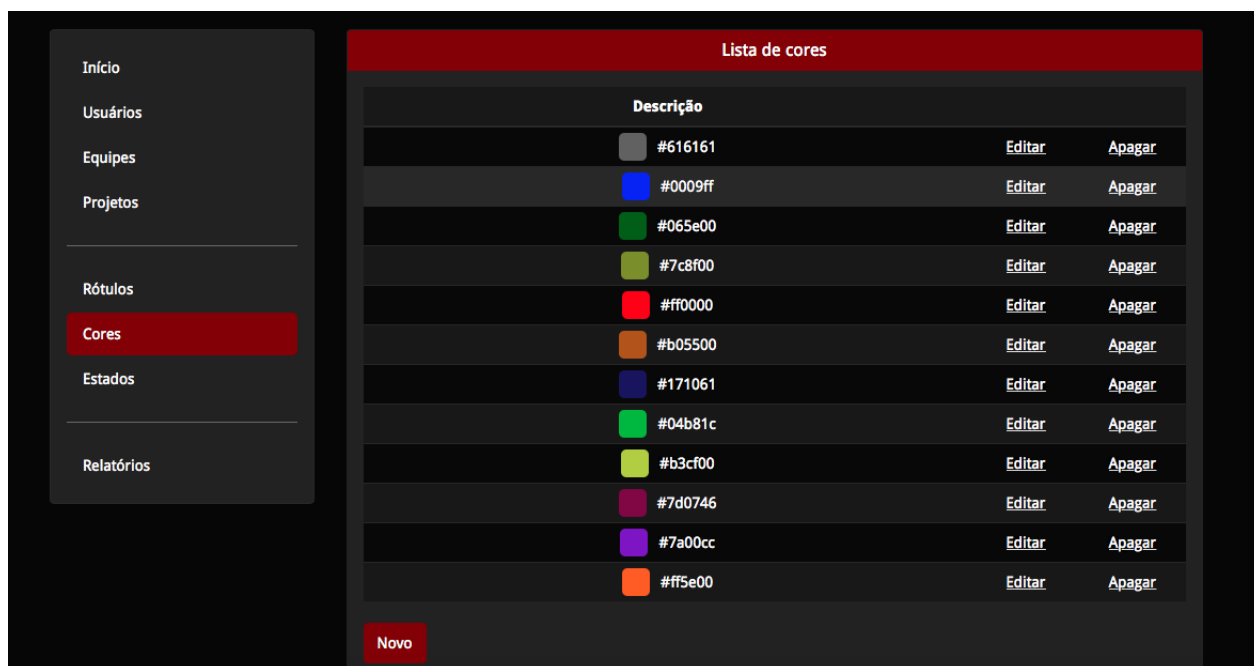


Figura 32 – Visualização para o administrador da listagem das cores para serem usadas nos rótulos e estados das atividades



Figura 33 – Visualização para o administrador da listagem dos estados para caracterizar uma atividade

Por fim, ao clicar na aba “Relatórios”, estes são divididos em: relatórios de Sprint e semanais, como mostrada na figura abaixo:

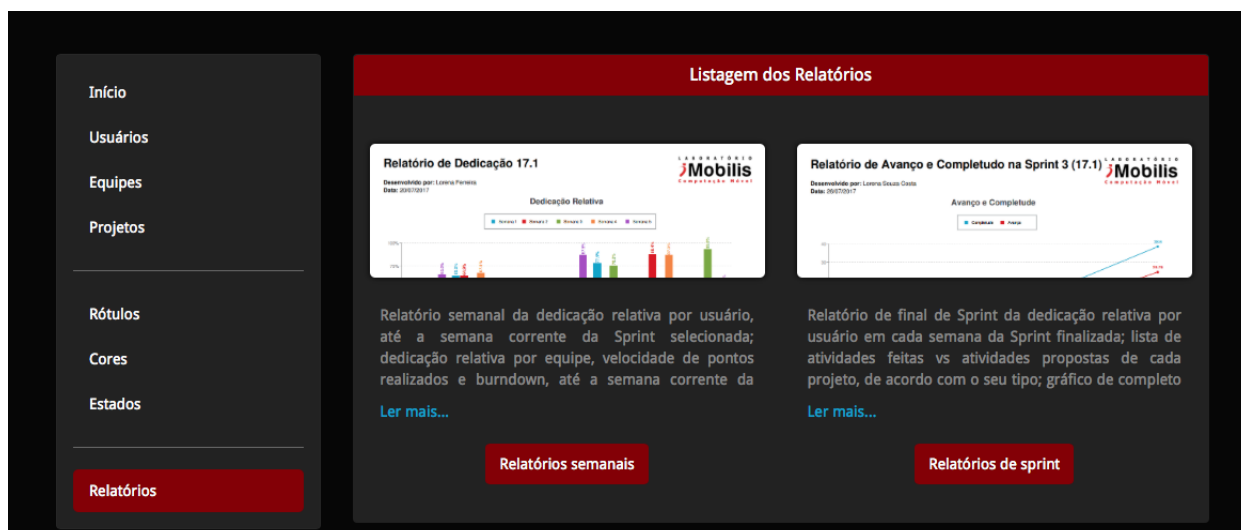


Figura 34 – Visualização para o administrador da listagem dos relatórios

Ao seleccionar o relatório semanal por exemplo, se faz necessário seleccionar qual tipo entre estes relatórios se deseja gerar e clicar no botão “Buscar” como na imagem seguinte:

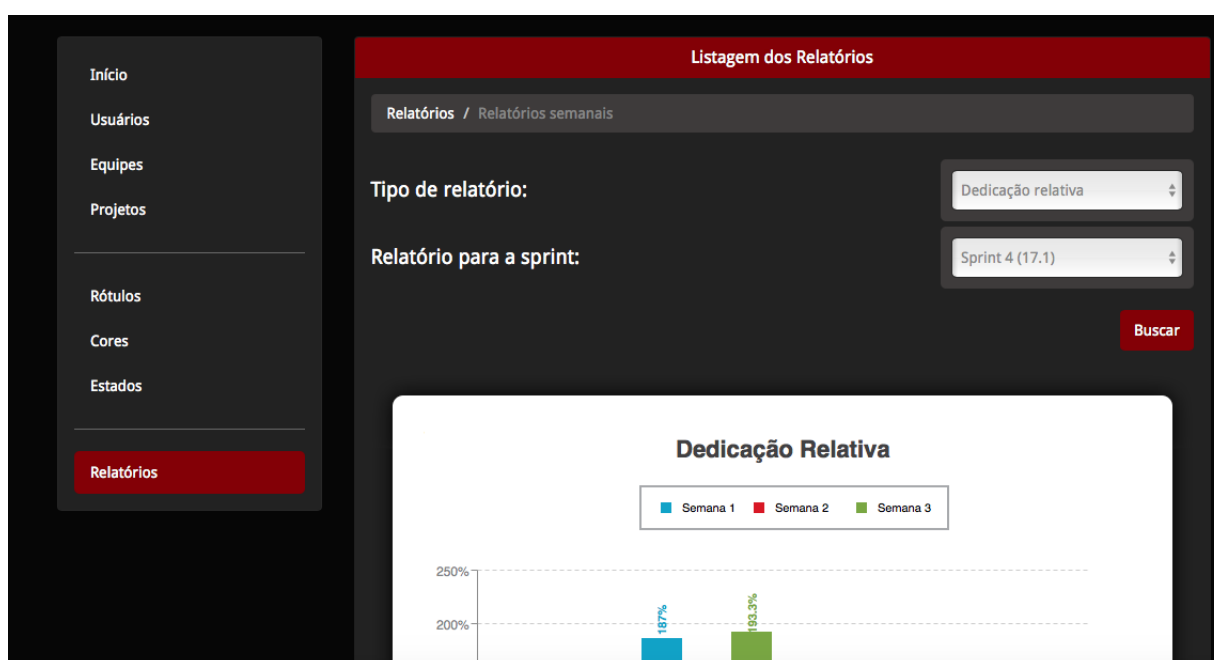


Figura 35 – Visualização para o administrador da listagem dos relatórios

A seguir, são apresentadas as telas de listagem dos relatórios de Sprint e semanais com seus respectivos tipos existentes:

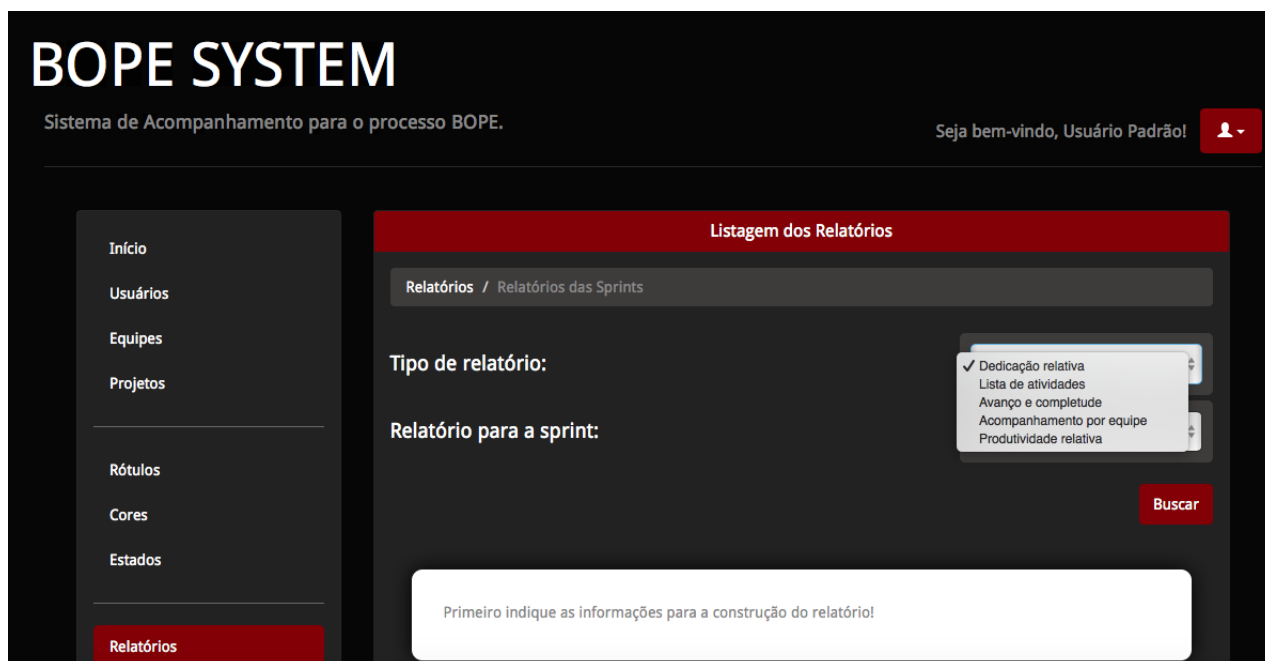


Figura 36 – Visualização para o administrado dos possíveis relatórios de Sprint



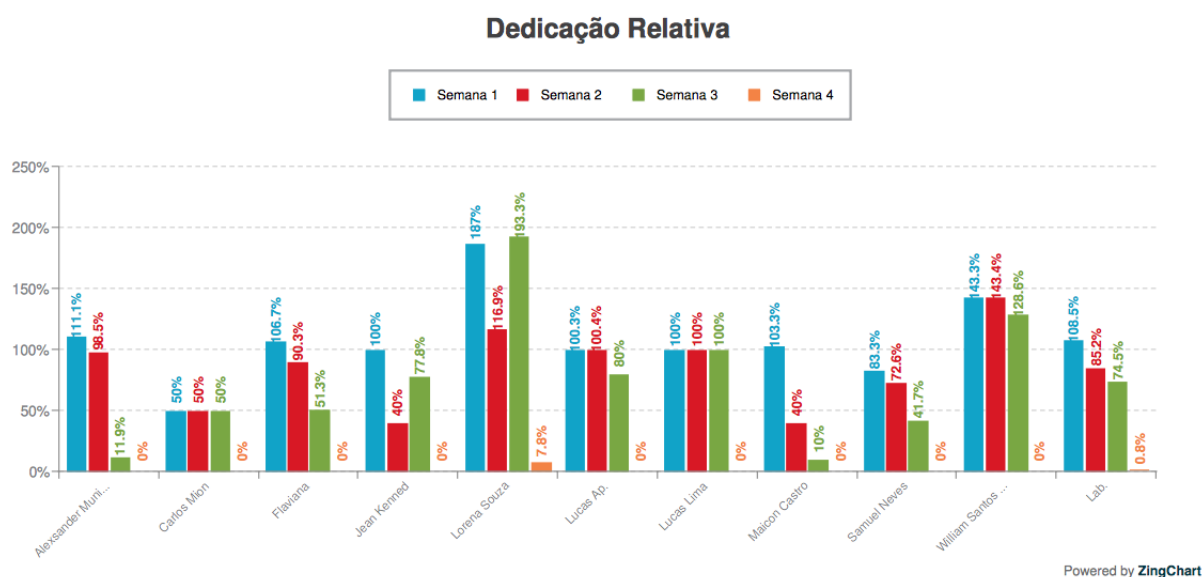
Figura 37 – Visualização para o administrado dos possíveis relatórios semanais

Para fazer o download destes relatórios buscados, se faz necessário clicar no botão “Download PDF” e assim eles são gerados como nas imagens seguintes:

Relatório de Dedicção Semanal 17.1

Desenvolvido por: Lorena Souza Costa

Data: 01/09/2017



* Integrantes da equipe: Alexsander Muniz da Costa, Carlos Mion, Flaviana, Jean Kenned, Lorena Souza, Lucas Ap., Lucas Lima, Maicon Castro, Samuel Neves, William Santos Nunes.

Figura 38 – Relatório de dedicação relativa dos usuários

Acompanhamento por equipe 17.1

Desenvolvido por: Lorena Souza Costa

Data: 30/08/2017



BOPE SYSTEM

* Integrantes da equipe: Lorena Souza.

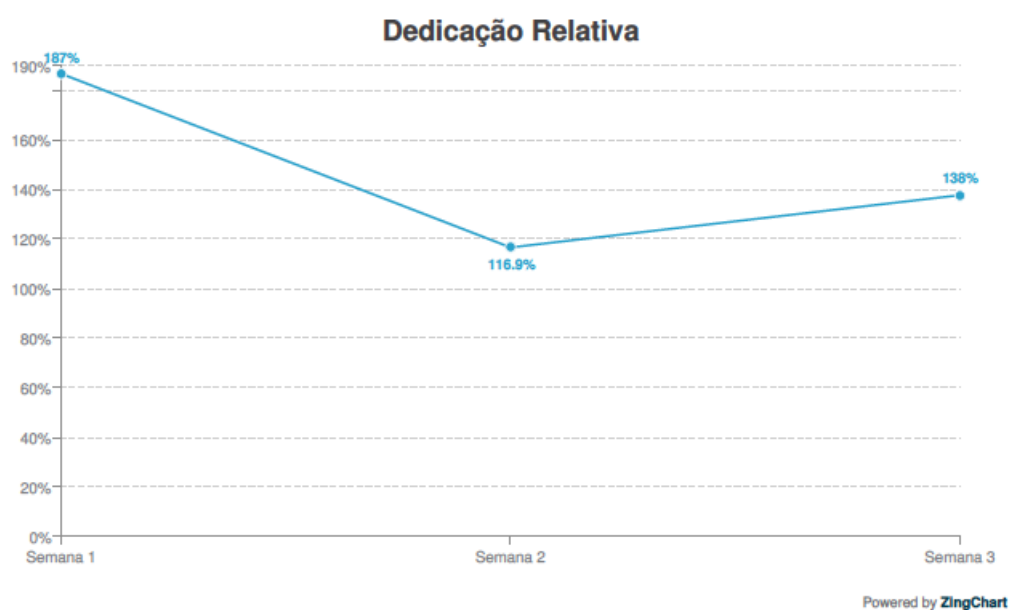


Figura 39 – Relatório de Acompanhamento de equipe: dedicação relativa de uma equipe específica

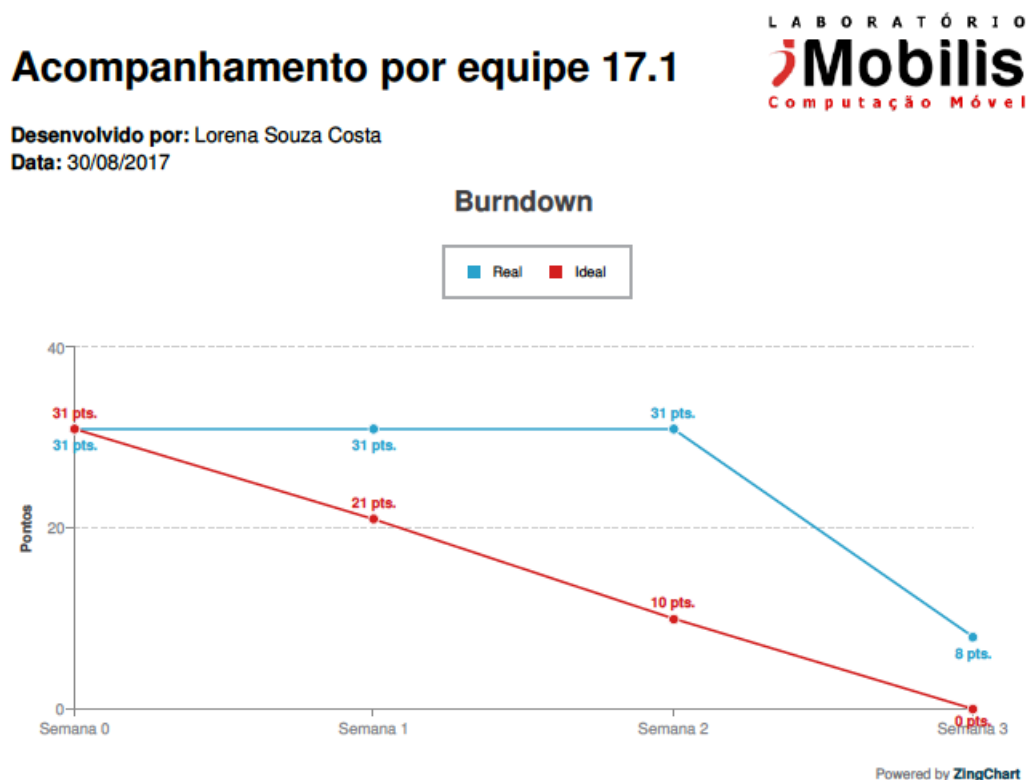


Figura 40 – Relatório de Acompanhamento de equipe: burndown de uma equipe específica

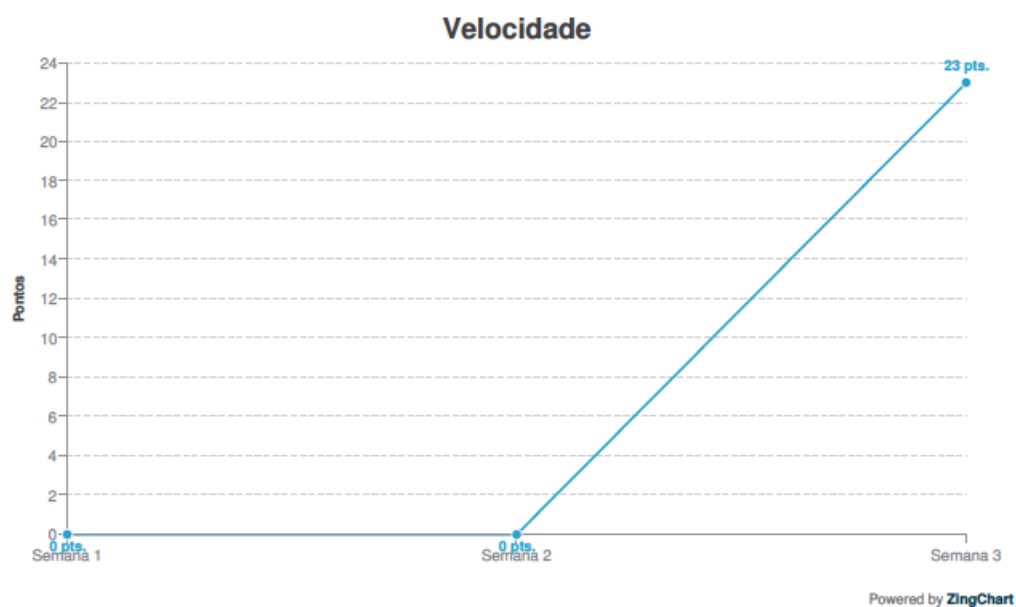
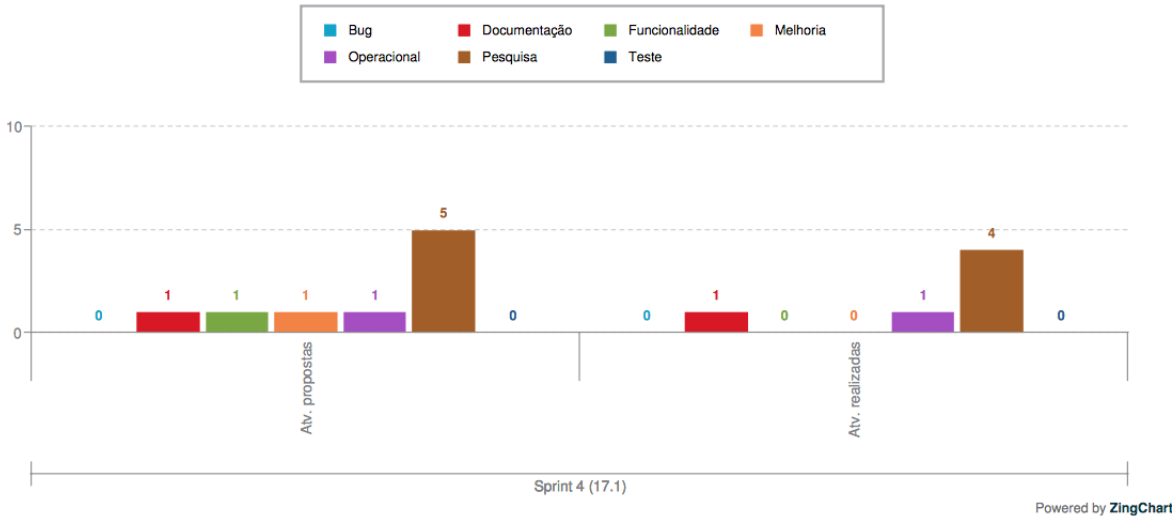


Figura 41 – Relatório de Acompanhamento de equipe: velocidade de uma equipe específica

Desenvolvido por: Lorena Souza Costa
Data: 03/09/2017

BOPE SYSTEM

Atividades Propostas vs Atividades Realizadas



* Integrantes da equipe: Lorena Souza.

Figura 42 – Relatório de atividades para uma equipe específica

Avanço e Completude

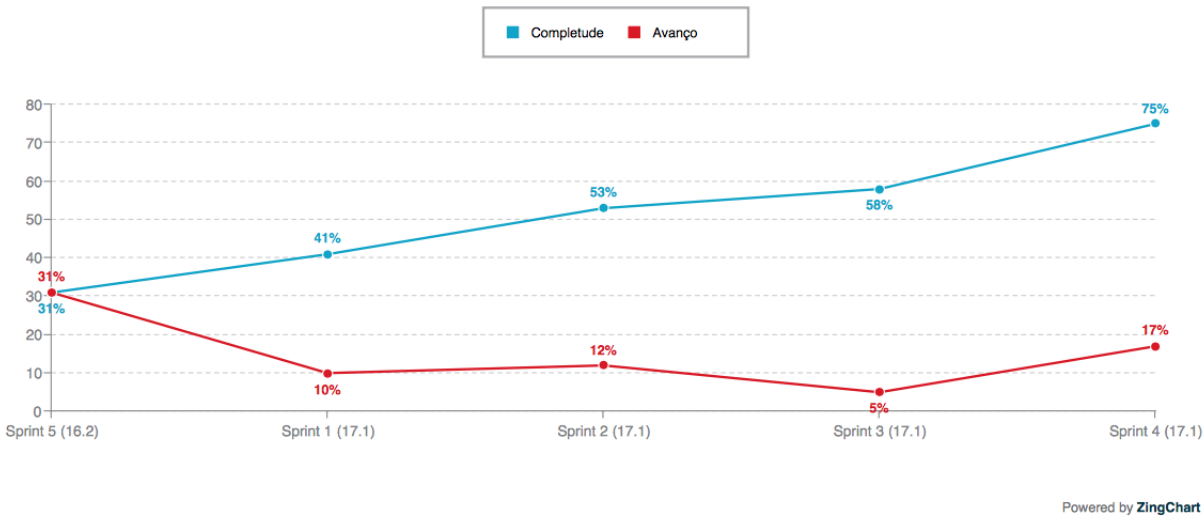


Figura 43 – Relatório de avanço e completude para uma equipe específica

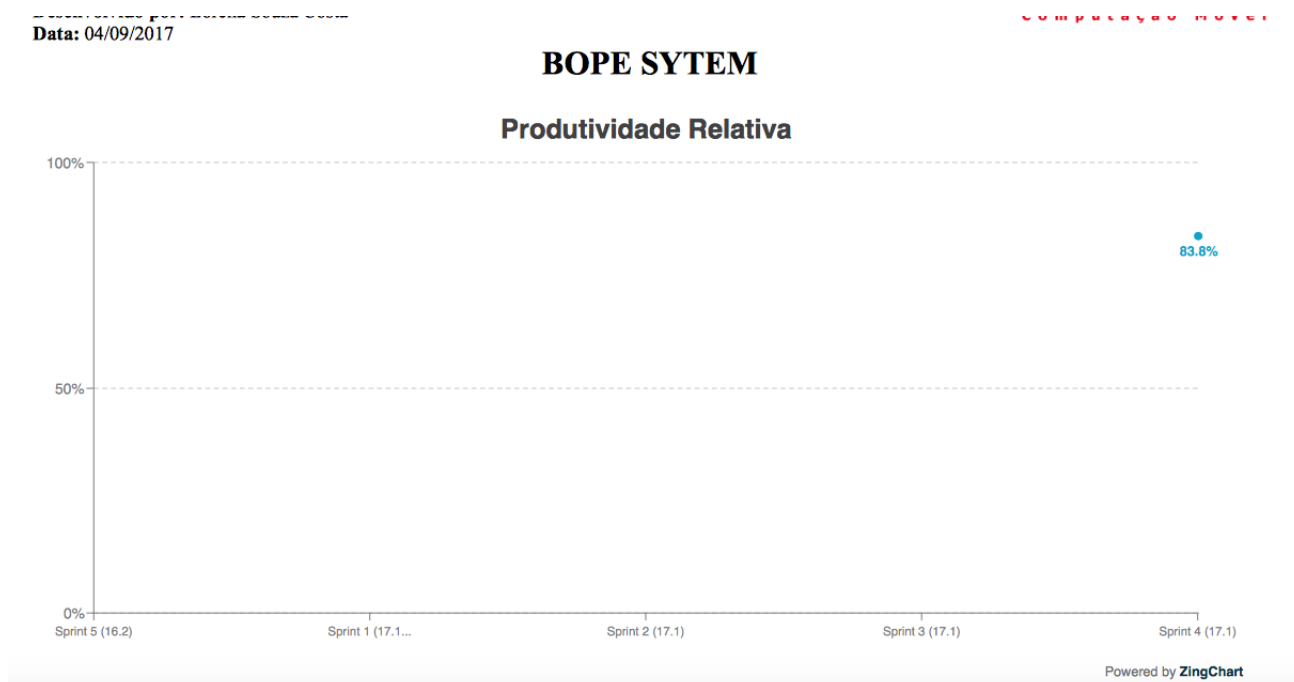


Figura 44 – Relatório de produtividade relativa para uma equipe específica

Na visão do usuário colaborador, apenas as abas de “Início”, “Equipes” e “Projetos” estarão disponíveis para ele, como mostrado a seguir:

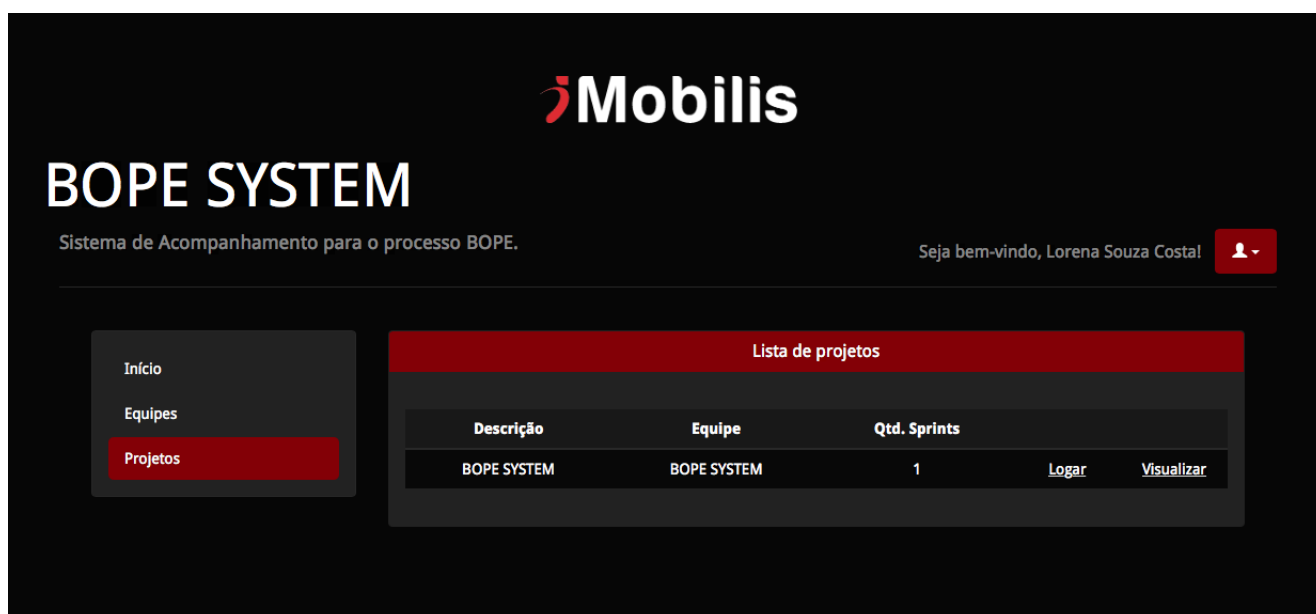


Figura 45 – Visão do colaborador com a aba “Projetos” selecionada

Percebe-se que para o usuário colaborador, a aba de “Projetos” vem automaticamente selecionada. As demais telas são idênticas às outras mostradas anteriormente na visão do administrador, mas com alterações nos recursos disponíveis para o colaborador. Isso acontece porque este tipo de usuário não pode criar, editar e apagar nada no sistema,

com exceção do fato de poder alterar seu perfil e adicionar atividades para ele mesmo durante um *Sprint*.

5 Considerações Finais

Este capítulo apresenta, como dito anteriormente, a conclusão, trabalhos futuros e limitações do sistema.

5.1 Conclusão

Este trabalho surgiu, principalmente, com a necessidade de melhorar o processo BOPE, de forma a auxiliar o gerenciamento de projetos e gerar os relatórios requeridos de forma automatizada. Para a construção deste software, as métricas estabelecidas dentro do processo BOPE foram devidamente empregadas revelando uma análise do comprometimento e desempenho dos membros do laboratório. Além disso, através deste sistema *web*, se tornou possível a preservação do registro de informações dos projetos dentro do iMobilis com o decorrer do tempo. Logo, este software pode ser utilizado por qualquer instituição que segue o processo BOPE e que tenha a necessidade de apoiar o gerenciamento de projetos.

Durante este trabalho, as práticas ágeis de engenharia de software foram usadas, acelerando o desenvolvimento do sistema e possibilitando uma visão das etapas a serem definidas, executadas e concluídas juntamente com a implementação, testes e implantação do sistema *web*. Vale ressaltar que, durante o desenvolvimento do aplicativo, ocorreram algumas mudanças na definição de requisitos, histórias de usuários, cenários de testes e na modelagem do banco de dados. Tais mudanças foram evidenciadas à medida que o sistema foi sendo implementado.

Percebe-se também que as funcionalidades exigidas pelo sistema foram cumpridas, de forma que a conclusão destas foi realizada de forma contínua e, cada uma delas, apresentava um nível de complexidade no desenvolvimento. É importante salientar que, na criação desse software, a busca por proporcionar uma interface fácil e simples foi de suma importância para prover uma melhor interação com o usuário e uma facilidade ao utilizá-lo. Além disso, o aplicativo procura também fornecer dados corretos na geração de relatórios e evitar falhas durante o uso deste. Ele também pretende ser flexível de forma a ser fácil de modificar, podendo corrigir erros e incluir melhorias. E por fim, o presente projeto é um sistema *web* fácil de instalar e responsivo, de forma que este pode ser acessado em diferentes plataformas sendo necessário apenas o acesso à internet.

Através deste aplicativo, se torna mais fácil gerenciar os projetos existentes no iMobilis, auxiliando no controle das tarefas especificadas para cada colaborador, no cumprimento de prazo para tais tarefas e, conseqüentemente, entrega do projeto, na

visualização do progresso de cada projeto, na recuperação de informações e, por fim, no desempenho da equipe para resultar numa maior produtividade e dedicação desta.

5.2 Trabalhos Futuros e Limitações

Um ponto a evidenciar sobre a limitação encontrada no BOPE SYSTEM, é o fato de que os relatórios devem ser gerados na data correta, tanto no último dia da semana, quanto no último dia do *Sprint*, para não gerar dados inconsistentes nos relatórios. Portanto, se faz necessário gerar os relatórios no dia exato para obtenção de dados concisos ao revelar as métricas estabelecidas no processo BOPE.

Deseja-se que, nos trabalhos futuros, a tela de visualização de atividades proporcione uma melhor visão destas de acordo com o *status* de cada uma. Isso irá facilitar o usuário a distinguir melhor quais atividades ainda não foram iniciadas, estão em andamento, esperando validação e que já foram validadas. Além disso, seria viável ter um fórum dentro do aplicativo, onde este possibilitaria uma melhor interação entre os usuários da equipe e promoveria, possivelmente, um melhor desempenho. Outras melhorias para o sistema seria adicionar prioridade para as atividades propostas e um local para armazenar todos os relatórios gerados como uma melhor forma de organização da documentação gerada pelo software.

6 Referências bibliográficas

PIRES, Jackson. Ruby on Rails - Curso Completo. [udemy.com](https://www.udemy.com/rubyonrails/learn/v4/overview). 2016. Disponível em: <<https://www.udemy.com/rubyonrails/learn/v4/overview>> Acessado em: 17/02/2017.

PRESSMAN, Roger S. Engenharia de Software - Uma Abordagem Profissional. 18a Ed. Amgh Editora, 2016.

Fuentes, Vinícius Baggio. Ruby on Rails: Coloque sua aplicação web nos trilhos. Editora Casa do Código, 2014.

PEREIRA, Igor Muzetti; CARNEIRO, Tiago Garcia de Senna; PEREIRA, Rodrigo Reis. Developing innovative software in Brazilian public universities : Tailoring agile processes to the reality of research and development laboratories. 4a Annual International Conference on Software Engineering Applications. - Phuket : GSTF, 2013.

SOMMERVILLE, Ian. Engenharia de Software. 9o Edição. PEARSON EDUCATION - BR, 2011.

Cohn, Mike. Succeeding with agile: software development using Scrum. Pearson Education, 2010.

PMBOK, GUIA. “Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos (Guia PMBOK®). Em português.” Project Management Institute, Inc. EUA. Versão em Pdf para associado PMI (2008).

Milani, André. PostgreSQL-Guia do Programador. Novatec Editora, 2008.

Cohn, Mike. User stories applied: For agile software development. Addison-Wesley Professional, 2004.

Beck, Kent, and Martin Fowler. Planning extreme programming. Addison-Wesley Professional, 2001.

Apêndices

APÊNDICE A – Histórias de Usuário e Cenários de Teste

1. Serviço de Gerenciamento de Usuário

Autenticar usuário

- * História
 - COMO um usuário cadastrado
 - EU GOSTARIA DE logar no sistema mediante fornecimento do "nome do usuário" e "senha"
 - PARA CONSEGUIR acesso às funcionalidades do sistema de acordo com o tipo de usuário (administrador ou colaborador)
- * Cenário principal
 - DADO que um usuário cadastrado forneça "senha" e "nome de usuario" válidos
 - QUANDO o botão "Entrar" for pressionado
 - ENTÃO a aplicação deve transitar para a tela principal
- * Cenário alternativo
 - DADO que um usuário cadastrado forneça "senha" inválida
 - QUANDO o botão "Entrar" for pressionado
 - ENTÃO (a) a mensagem "Senha inválida" deve ser apresentada e (b) a aplicação deve permanecer na tela de login
- * Cenário alternativo
 - DADO que um usuário cadastrado forneça "nome do usuário" inválido
 - QUANDO o botão "Entrar" for pressionado
 - ENTÃO (a) a mensagem "Usuário desconhecido" deve ser apresentada e (b) a aplicação deve permanecer na tela de login

Cadastrar novo usuário

- * História
 - COMO um administrador cadastrado
 - EU GOSTARIA DE cadastrar novos usuários fornecendo nome, senha e tipo de usuário
 - PARA CONSEGUIR adicionar usuários ao sistema para que estes tenham acesso às funcionalidades do sistema
- * Cenário principal
 - DADO que um administrador forneça nome, senha e tipo de usuário
 - QUANDO o botão "criar usuário" for pressionado
 - ENTÃO o usuário é cadastrado com sucesso
- * Cenário alternativo
 - DADO que um administrador clique no botão "novo usuário", mas ele desiste de criá-lo por algum motivo
 - QUANDO o botão "cancelar" for pressionado
 - ENTÃO o administrador é redirecionado para a tela anterior de usuários cadastrados

Figura 46 – Autenticar e adicionar usuário

Editar usuário

- * História
 - COMO usuário cadastrado
 - EU GOSTARIA DE alterar informações fornecidas no cadastro: nome, matrícula, projeto, senha e e-mail
 - PARA CONSEGUIR atualizar minhas informações no sistema como a redefinição da minha senha, por exemplo
- * Cenário principal
 - DADO que um usuário selecione um usuário cadastrado no sistema com suas devidas informações
 - Quando o botão "editar usuário" e, posteriormente, "salvar usuário" forem pressionados
 - Então uma mensagem "tem certeza que deseja salvar as alterações feitas no usuário" é emitida
 - E ao pressionar o botão "Sim", as alterações feitas em relação às informações do usuário são salvas com sucesso.
- * Cenário alternativo
 - DADO que um usuário cadastrado clique no botão "editar perfil do usuário", mas ele desiste de editar seu perfil por algum motivo
 - QUANDO o botão "cancelar" for pressionado
 - ENTÃO o usuário é redirecionado para a tela anterior de visualização do usuário

Excluir usuário

- * História
 - COMO administrador cadastrado
 - EU GOSTARIA DE deletar um usuário cadastrado anteriormente que não fará mais parte do time iMobilis
 - PARA CONSEGUIR que tal usuário não tenha mais acesso às funcionalidades do sistema
- * Cenário principal
 - DADO que um usuário cadastrado tenha suas devidas informações presentes no sistema
 - QUANDO o administrador pressionar o botão "excluir usuário"
 - ENTÃO uma mensagem "tem certeza que deseja excluir este usuário" é emitida
 - E ao pressionar o botão "Sim", o usuário é excluído com sucesso.
- * Cenário alternativo
 - DADO que um administrador cadastrado clique no botão "excluir usuário", mas ele desiste de excluí-lo por algum motivo
 - QUANDO o botão "não" for pressionado ao responder a pergunta: "tem certeza que deseja excluir este usuário"
 - ENTÃO o administrador é redirecionado para a tela anterior de visualização do usuário

Figura 47 – Editar e excluir usuário

Visualizar perfil do usuário

```

* História
* COMO um usuário cadastrado
* EU GOSTARIA DE ter acesso às minhas informações cadastradas anteriormente: nome, matrícula, senha,
projeto e e-mail (exceto tipo de usuário, pois um usuário comum não pode alterá-lo)
* PARA CONSEGUIR ver as minhas informações e/ou editar meu perfil
* Cenário principal
* DADO que um usuário cadastrado tenha suas devidas informações presentes no sistema
* QUANDO o usuário selecionar um usuário e o botão "Visualizar perfil do usuário" for pressionado
* ENTAO o usuário é direcionado para uma página onde contém todas as suas informações cadastradas
(exceto o tipo de usuário)
* Cenário alternativo
* DADO que um usuário cadastrado tenha suas devidas informações presentes no sistema
* QUANDO o administrador selecionar um usuário e o botão "Visualizar perfil do usuário" for pressionado por ele
* ENTAO o administrador é direcionado para uma página onde contém todas as suas informações cadastradas, podendo
alterar ou excluir tal usuário

```

Figura 48 – Visualizar perfil do usuário

2. Serviço de gerenciamento da equipe

Cadastrar equipe

```

* História
* COMO administrador cadastrado
* EU GOSTARIA DE criar uma equipe informando apenas a descrição (nome) da equipe
* PARA CONSEGUIR acesso às funcionalidades do sistema
* Cenário principal
* DADO que um administrador cadastrado clique em "adicionar equipe" e forneça descrição (nome) da equipe
* QUANDO o botão "criar" for pressionado
* ENTAO a equipe é cadastrada com sucesso
* Cenário alternativo
* DADO que um administrador clique no botão "criar equipe", mas ele desiste de criá-la por algum motivo
* QUANDO o botão "cancelar" for pressionado
* ENTAO o administrador é redirecionado para a tela anterior de equipes cadastradas

```

Editar equipe

```

* História
* COMO administrador cadastrado
* EU GOSTARIA DE alterar as informações da equipe podendo mudar a descrição e/ou projeto e/ou membros cadastrados
* PARA CONSEGUIR atualizar as informações referentes às equipes cadastradas
* Cenário principal
* DADO que um administrador tenha cadastrado uma equipe
* QUANDO o botão "Editar equipe" for pressionado e, posteriormente, "salvar equipe" forem pressionados
* Então uma mensagem "tem certeza que deseja salvar as alterações feitas na equipe" é emitida
* E ao pressionar o botão "Sim", as alterações feitas em relação às informações da equipe são salvas com sucesso.
* Cenário alternativo
* DADO que um administrador cadastrado clique no botão "editar equipe", mas ele desiste de editar seu perfil
por algum motivo
* QUANDO o botão "voltar" for pressionado
* ENTAO o administrador é redirecionado para a tela anterior de visualização da equipe
* Cenário alternativo
* DADO que um administrador cadastrado clique no botão "editar equipe"
* QUANDO o administrador selecionar um membro daquela determinada equipe, ele pode adicioná-lo ou excluí-lo de
tal equipe
* ENTAO o administrador escolhe qual operação realizar

```

Figura 49 – Configuração do Banco de Dados PostgreSQL

Excluir equipe

```

* História
* COMO administrador cadastrado
* EU GOSTARIA DE apagar a equipe, uma vez que ela não pertence mais ao sistema
* PARA CONSEGUIR que tal equipe não tenha mais acesso às funcionalidades do sistema
* Cenário principal
* DADO que uma equipe cadastrada tenha suas devidas informações presentes no sistema
* QUANDO o administrador pressionar o botão "excluir usuário"
* ENTAO uma mensagem "tem certeza que deseja excluir este equipe" é emitida
* E ao pressionar o botão "Sim", a equipe é excluída com sucesso.
* Cenário alternativo
* DADO que um administrador cadastrado clique no botão "excluir equipe", mas ele desiste de excluí-la por
algum motivo
* QUANDO o botão "não" for pressionado ao responder a pergunta: "tem certeza que deseja excluir este usuário"
* ENTAO o administrador é redirecionado para a tela anterior de visualização da equipe

```

Vincular usuário a equipe

```

* História
* COMO administrador cadastrado
* EU GOSTARIA DE agrupar os membros de uma determinada equipe
* PARA CONSEGUIR coletar informações por equipe
* Cenário principal
* DADO que um administrador cadastrado clique no botão "editar equipe"
* QUANDO o botão "adicionar membro a equipe" for pressionado
* ENTAO o administrador é redirecionado para uma tela com todos os usuários cadastrados, onde este seleciona
qual/quais usuário(s) se deseja adicionar a equipe
* E ao clicar em "adicionar", o(s) membro(s) é/são adicionado(s) a equipe
* Cenário alternativo
* DADO que um administrador cadastrado clique no botão "adicionar membro a equipe", mas ele desiste de fazê-lo
por algum motivo
* QUANDO o botão "cancelar" for pressionado
* ENTAO o administrador é redirecionado para a tela anterior de editar equipe

```

Figura 50 – Excluir usuário e vincular usuário a uma equipe

Excluir usuário da equipe

```

* História
* COMO administrador cadastrado
* EU GOSTARIA DE deletar um usuário de uma determinada equipe
* PARA CONSEGUIR que tal usuário não tenha mais acesso às informações referentes a tal equipe
* Cenário principal
* DADO que um administrador cadastrado clique no botão "editar equipe"
* QUANDO o administrador selecionar um ou mais membros daquela determinada equipe e o botão "excluir membro
da equipe" for pressionado
* ENTAO o(s) membro(s) é/são excluído(s) da equipe após confirmação da exclusão
* Cenário alternativo
* DADO que um administrador cadastrado clique no botão "excluir membro da equipe", mas ele desiste de fazê-lo
por algum motivo
* QUANDO o botão "cancelar" for pressionado
* ENTAO o administrador é redirecionado para a tela anterior de editar equipe

```

Visualizar equipe

```

* História 1
* COMO usuário cadastrado
* EU GOSTARIA DE ter acesso às informações de cada equipe cadastradas anteriormente
* PARA CONSEGUIR visualizar e averiguar se todas as informações referentes a minha equipe estão corretas
* História 2
* COMO administrador cadastrado
* EU GOSTARIA DE ter acesso às informações de cada equipe cadastradas anteriormente
* PARA CONSEGUIR visualizar e/ou alterar e/ou excluir usuários da equipe
* Cenário principal
* DADO que uma equipe cadastrada tenha suas devidas informações presentes no sistema
* QUANDO o botão "Visualizar equipe" for pressionado por um usuário
* ENTAO o usuário é direcionado para uma página onde contém todas as informações daquela equipe
* Cenário alternativo
* DADO que uma equipe cadastrada tenha suas devidas informações presentes no sistema
* QUANDO o administrador selecionar uma equipe e o botão "Visualizar perfil do usuário" for pressionado por ele
* ENTAO o administrador é direcionado para uma página onde contém todas as suas informações cadastradas daquela
equipe, podendo alterar ou excluir tal equipe

```

Figura 51 – Excluir usuário da equipe e visualizar equipe

3. Serviço de Gerenciamento de Atividades

Cadastrar atividades

- * História
 - * COMO um usuário cadastrado
 - * EU GOSTARIA DE inserir uma nova atividade mediante o fornecimento da descrição, pontos, label e status da atividade
 - * PARA CONSEGUIR cadastrar atividades no sistema a serem feitas pelos usuários
- * Cenário principal
 - * DADO que um usuário clique no botão "adicionar atividade" e forneça descrição, pontos, label e status
 - * QUANDO o botão "adicionar" for pressionado
 - * ENTÃO a atividade é cadastrada com sucesso
- * Cenário alternativo
 - * DADO que um usuário clique no botão "adicionar atividade", mas ele desiste de fazê-lo por algum motivo
 - * QUANDO o botão "cancelar" for pressionado
 - * ENTÃO o usuário é redirecionado para a tela anterior de atividades

Editar atividades

- * História
 - * COMO um usuário cadastrado
 - * EU GOSTARIA DE editar a atividade ao selecioná-la
 - * PARA CONSEGUIR alterar qualquer incoerência nas informações das atividades como: descrição, pontos, label e status
- * Cenário principal
 - * DADO que um usuário selecione uma atividade cadastrada no sistema com suas devidas informações
 - * Quando o botão "editar atividade" e, posteriormente, "salvar" forem pressionados
 - * Então uma mensagem "tem certeza que deseja salvar as alterações feitas na atividade" é emitida
 - * E ao pressionar o botão "Sim", as alterações feitas em relação às informações de tal atividade são salvas com sucesso.
- * Cenário alternativo
 - * DADO que um usuário cadastrado clique no botão "editar atividade", mas ele desiste de fazê-lo por algum motivo
 - * QUANDO o botão "cancelar" for pressionado
 - * ENTÃO o usuário é redirecionado para a tela anterior de visualização da atividade

Figura 52 – Cadastrar e editar atividade

Excluir atividades

- * História
 - * COMO um usuário cadastrado
 - * EU GOSTARIA DE deletar alguma atividade que não faça mais parte da minha Sprint
 - * PARA CONSEGUIR ter maior controle sobre quais atividades fazem parte da minha Sprint
- * Cenário principal
 - * DADO que um usuário cadastrado selecione uma atividade cadastrada no sistema com suas devidas informações
 - * QUANDO o usuário pressionar o botão "excluir atividade"
 - * ENTÃO uma mensagem "tem certeza que deseja excluir esta atividade" é emitida
 - * E ao pressionar o botão "Sim", a atividade é excluído com sucesso.
- * Cenário alternativo
 - * DADO que um usuário cadastrado clique no botão "excluir atividade", mas ele desiste de excluí-la por algum motivo
 - * QUANDO o botão "não" for pressionado ao responder a pergunta: "tem certeza que deseja excluir esta atividade"
 - * ENTÃO o usuário é redirecionado para a tela anterior de visualização da atividade

Visualizar atividades

- * História
 - * COMO um usuário cadastrado
 - * EU GOSTARIA DE ver todas as informações referentes às minhas atividades numa determinada Sprint
 - * PARA CONSEGUIR visualizar e/ou alterar e/ou excluir atividades da Sprint
- * Cenário principal
 - * DADO que uma atividade cadastrada tenha suas devidas informações presentes no sistema
 - * QUANDO o usuário selecionar uma atividade e o botão "Visualizar atividade" for pressionado
 - * ENTÃO o usuário é direcionado para uma página onde contém todas as informações cadastradas sobre aquela atividade, podendo alterar ou excluir tal atividade

Figura 53 – Configuração do Banco de Dados PostgreSQL

Fechar atividade

- * História
 - * COMO um usuário cadastrado
 - * EU GOSTARIA DE encerrar uma atividade uma vez que esta já foi completada
 - * PARA CONSEGUIR controlar melhor todas as atividades que já foram ou não encerradas
- * Cenário principal
 - * DADO que um usuário tenha sua atividade cadastrada no sistema
 - * QUANDO o botão "fechar atividade" for pressionado
 - * ENTÃO a atividade é encerrada, dada tal atividade como concluída e validada
- * Cenário alternativo
 - * DADO que um usuário cadastrado clique no botão "fechar atividade", mas ele desiste de fechá-la por algum motivo
 - * QUANDO o botão "não" for pressionado ao responder a pergunta: "tem certeza que deseja fechar esta atividade"
 - * ENTÃO o usuário é redirecionado para a tela anterior de visualização da Sprint, onde se tem todas as atividades cadastradas

Figura 54 – Fechar atividade

4. Serviço de Gerenciamento de Projeto

Cadastrar novo projeto

- * História
 - * COMO um administrador cadastrado
 - * EU GOSTARIA DE cadastrar novos projetos fornecendo a descrição do projeto
 - * PARA CONSEGUIR adicionar projetos em andamento dentro do iMobilis
- * Cenário principal
 - * DADO que um administrador forneça descrição do projeto
 - * QUANDO o botão "criar projeto" for pressionado
 - * ENTÃO o projeto é criado com sucesso
- * Cenário alternativo
 - * DADO que um administrador clique no botão "criar novo projeto", mas ele desiste de criá-lo por algum motivo
 - * QUANDO o botão "cancelar" for pressionado
 - * ENTÃO o administrador é redirecionado para a tela anterior de projetos cadastrados

Editar projeto

- * História
 - * COMO um usuário cadastrado
 - * EU GOSTARIA DE alterar a descrição do projeto
 - * PARA CONSEGUIR atualizar tal informação do meu projeto
- * Cenário principal
 - * DADO que um usuário selecione um projeto cadastrado no sistema com suas devidas informações
 - * Quando o botão "editar projeto" e, posteriormente, "salvar projeto" forem pressionados
 - * Então uma mensagem "tem certeza que deseja salvar as alterações feitas no projeto" é emitida
 - * E ao pressionar o botão "Sim", as alterações feitas em relação às informações do projeto são salvas com sucesso.
- * Cenário alternativo
 - * DADO que um usuário cadastrado clique no botão "editar projeto", mas ele desiste de editá-lo por algum motivo
 - * QUANDO o botão "cancelar" for pressionado
 - * ENTÃO o usuário é redirecionado para a tela anterior de projetos cadastrados

Figura 55 – Cadastrar e editar projeto

Excluir projeto

- * História
 - * COMO administrador cadastrado
 - * EU GOSTARIA DE deletar um projeto cadastrado anteriormente
 - * PARA CONSEGUIR que tal projeto não faça mais parte dos projetos existentes dentro do iMobilis
- * Cenário principal
 - * DADO que um projeto cadastrado tenha suas devidas informações presentes no sistema
 - * QUANDO o administrador pressionar o botão "excluir projeto"
 - * ENTÃO uma mensagem "tem certeza que deseja excluir este projeto" é emitida
 - * E ao pressionar o botão "Sim", o projeto é excluído com sucesso.
- * Cenário alternativo
 - * DADO que um administrador cadastrado clique no botão "excluir projeto", mas ele desiste de excluí-lo por algum motivo
 - * QUANDO o botão "não" for pressionado ao responder a pergunta: "tem certeza que deseja excluir este projeto"
 - * ENTÃO o administrador é redirecionado para a tela anterior de projetos cadastrados

Visualizar projeto

- * História
 - * COMO um usuário cadastrado
 - * EU GOSTARIA DE ter acesso ao(s) meu(s) projetos cadastrados anteriormente
 - * PARA CONSEGUIR ver as Sprints referentes a tal projeto
- * Cenário principal
 - * DADO que um projeto cadastrado tenha suas devidas informações presentes no sistema
 - * QUANDO o usuário selecionar um projeto e o botão "Visualizar projeto" for pressionado
 - * ENTÃO o usuário é direcionado para uma página onde contém todas as informações cadastradas de tal projeto
- * Cenário alternativo
 - * DADO que um projeto cadastrado tenha suas devidas informações presentes no sistema
 - * QUANDO o administrador selecionar um usuário e o botão "Visualizar perfil do usuário" for pressionado por ele
 - * ENTÃO o administrador é direcionado para uma página onde contém todas as informações cadastradas de tal projeto, podendo alterar ou excluir tal projeto

Figura 56 – Excluir e visualizar projeto

Vincular equipe a um projeto

- * História
 - * COMO administrador cadastrado
 - * EU GOSTARIA DE ligar a equipe a um determinado projeto
 - * PARA CONSEGUIR que cada equipe seja responsável por um determinado projeto a ser implementado
- * Cenário principal
 - * DADO que um administrador cadastrado clique na aba "Projetos"
 - * QUANDO o botão "vincular equipe a projeto" for pressionado
 - * ENTÃO o administrador é redirecionado para uma tela com todas as equipes cadastradas, onde este seleciona qual equipe deve ser vinculada ao projeto
 - * E ao clicar em "vincular", tal equipe é vinculada ao projeto
- * Cenário alternativo
 - * DADO que um administrador cadastrado clique no botão "vincular equipe a projeto", mas ele desiste de fazê-lo por algum motivo
 - * QUANDO o botão "cancelar" for pressionado
 - * ENTÃO o administrador é redirecionado para a tela anterior de projetos cadastrados

Figura 57 – Vincular equipe a um projeto

5. Serviço de Gerenciamento de Sprint

Adicionar Sprint

- * História
 - * COMO um administrador cadastrado
 - * EU GOSTARIA DE criar nova Sprint dentro de um projeto informando apenas sua descrição
 - * PARA CONSEGUIR cadastrar novas atividades e semanas em cada Sprint
- * Cenário principal
 - * DADO que um administrador forneça descrição da Sprint
 - * QUANDO o botão "criar Sprint" for pressionado
 - * ENTÃO uma nova Sprint é cadastrada com sucesso
- * Cenário alternativo
 - * DADO que um administrador clique no botão "Adicionar Sprint", mas ele desiste de criá-la por algum motivo
 - * QUANDO o botão "cancelar" for pressionado
 - * ENTÃO o administrador é redirecionado para a tela anterior de Sprints cadastradas

Editar Sprint

- * História
 - * COMO um usuário cadastrado
 - * EU GOSTARIA DE alterar a descrição da Sprint criada
 - * PARA CONSEGUIR atualizar as informações de cada Sprint evitando que elas estejam erradas
- * Cenário principal
 - * DADO que um usuário selecione uma Sprint cadastrada no sistema com suas devidas informações
 - * Quando o botão "editar Sprint" e, posteriormente, "salvar Sprint" forem pressionados
 - * Então uma mensagem "tem certeza que deseja salvar as alterações feitas na Sprint" é emitida
 - * E ao pressionar o botão "Sim", as alterações feitas em relação às informações da Sprint são salvas com sucesso.
- * Cenário alternativo
 - * DADO que um usuário cadastrado clique no botão "editar Sprint", mas ele desiste de editá-la por algum motivo
 - * QUANDO o botão "cancelar" for pressionado
 - * ENTÃO o usuário é redirecionado para a tela anterior de Sprints cadastradas

Figura 58 – Adicionar e editar Sprint

Excluir Sprint

- * História
 - * COMO um administrador cadastrado
 - * EU GOSTARIA DE deletar uma Sprint
 - * PARA CONSEGUIR que Sprints inválidas sejam descartadas do projeto
- * Cenário principal
 - * DADO que uma Sprint cadastrada tenha suas devidas informações presentes no sistema
 - * QUANDO o administrador pressionar o botão "excluir Sprint"
 - * ENTAO uma mensagem "tem certeza que deseja excluir esta Sprint" é emitida
 - * E ao pressionar o botão "Sim", a Sprint é excluído com sucesso.
- * Cenário alternativo
 - * DADO que um administrador cadastrado clique no botão "excluir Sprint", mas ele desiste de excluí-la por algum motivo
 - * QUANDO o botão "não" for pressionado ao responder a pergunta: "tem certeza que deseja excluir esta Sprint"
 - * ENTAO o administrador é redirecionado para a tela anterior de Sprints cadastradas

Visualizar Sprint

- * História
 - * COMO um usuário cadastrado
 - * EU GOSTARIA DE ter acesso às Sprints cadastradas anteriormente
 - * PARA CONSEGUIR ver as atividades referentes a determinada Sprint
- * Cenário principal
 - * DADO que uma Sprint cadastrada tenha suas devidas informações presentes no sistema
 - * QUANDO o usuário selecionar uma Sprint e o botão "Visualizar Sprint" for pressionado
 - * ENTAO o usuário é direcionado para uma página onde contém todas as informações da Sprint cadastrada

Figura 59 – Excluir e visualizar Sprint

Visualizar pontos atualizados

- * História
 - * COMO um administrador cadastrado
 - * EU GOSTARIA DE visualizar os pontos atualizados para cada usuário e equipe
 - * PARA CONSEGUIR gerar os relatórios semanais e finais
- * Cenário principal
 - * DADO que um administrador cadastrado selecione uma ou mais semanas e clique em "visualizar semanas"
 - * QUANDO o botão "Visualizar pontos atualizados" for pressionado por ele
 - * ENTAO o administrador é direcionado para uma página onde contém todas as informações sobre os pontos atualizados (pontos feitos até então) por usuário e por equipe, podendo alterá-los
- * Cenário alternativo
 - * DADO que um administrador cadastrado selecione uma ou mais semanas e clique em "visualizar semanas"
 - * QUANDO o botão "Visualizar pontos atualizados" for pressionado e o administrador terminar de visualizar os pontos atualizados daquela(s) semana(s)
 - * ENTAO ele clica em "voltar" e ele é redirecionado para a tela anterior de semanas cadastradas

Visualizar pontos cadastrados

- * História
 - * COMO um administrador cadastrado
 - * EU GOSTARIA DE visualizar os pontos cadastrados para cada usuário e equipe
 - * PARA CONSEGUIR gerar os relatórios semanais e finais
- * Cenário principal
 - * DADO que um administrador cadastrado selecione uma ou mais semanas e clique em "visualizar semanas"
 - * QUANDO o botão "Visualizar pontos cadastrados" for pressionado por ele
 - * ENTAO o administrador é direcionado para uma página onde contém todas as informações sobre os pontos cadastrados por usuário e por equipe, podendo alterá-los
- * Cenário alternativo
 - * DADO que um administrador cadastrado selecione uma ou mais semanas e clique em "visualizar semanas"
 - * QUANDO o botão "Visualizar pontos cadastrados" for pressionado e o administrador terminar de visualizar os pontos cadastrados daquela(s) semana(s)
 - * ENTAO ele clica em "voltar" e ele é redirecionado para a tela anterior de semanas cadastradas

Figura 60 – Visualizar pontos atualizados e cadastrados no Sprint

6. Serviço de Gerenciamento de Semana

Adicionar semana a Sprint

- * História
 - * COMO um administrador cadastrado
 - * EU GOSTARIA DE incluir uma semana a determinada Sprint
 - * PARA CONSEGUIR ter mais controle sobre os pontos cadastrados e obtidos até então
- * Cenário principal
 - * DADO que um administrador forneça descrição da semana
 - * QUANDO o botão "adicionar semana a Sprint" for pressionado
 - * ENTÃO a semana é cadastrada com sucesso
- * Cenário alternativo
 - * DADO que um administrador clique no botão "adicionar semana a Sprint", mas ele desiste de adicioná-la por algum motivo
 - * QUANDO o botão "cancelar" for pressionado
 - * ENTÃO o administrador é redirecionado para a tela anterior de semanas cadastradas

Editar semana da Sprint

- * História
 - * COMO um usuário cadastrado
 - * EU GOSTARIA DE alterar a descrição da semana cadastrada numa Sprint
 - * PARA CONSEGUIR atualizar descrição de cada semana evitando que ela esteja errada
- * Cenário principal
 - * DADO que um usuário selecione uma semana cadastrada no sistema com suas devidas informações
 - * Quando o botão "editar semana" e, posteriormente, "salvar" forem pressionados
 - * Então uma mensagem "tem certeza que deseja salvar as alterações feitas na semana" é emitida
 - * E ao pressionar o botão "Sim", as alterações feitas em relação às informações da semana são salvas com sucesso.
- * Cenário alternativo
 - * DADO que um usuário cadastrado clique no botão "editar semana", mas ele desiste de editá-la por algum motivo
 - * QUANDO o botão "cancelar" for pressionado
 - * ENTÃO o usuário é redirecionado para a tela anterior de semanas cadastradas

Figura 61 – Adicionar e editar semana no Sprint

Excluir semana da Sprint

- * História
 - * COMO um administrador cadastrado
 - * EU GOSTARIA DE deletar semana da Sprint
 - * PARA CONSEGUIR que tal semana não faça mais parte da Sprint em questão
- * Cenário principal
 - * DADO que um administrador selecione uma semana cadastrada no sistema com suas devidas informações
 - * QUANDO o administrador pressionar o botão "excluir semana"
 - * ENTÃO uma mensagem "tem certeza que deseja excluir esta semana da Sprint" é emitida
 - * E ao pressionar o botão "Sim", a semana é excluída com sucesso.
- * Cenário alternativo
 - * DADO que um administrador cadastrado clique no botão "excluir semana", mas ele desiste de excluí-la por algum motivo
 - * QUANDO o botão "não" for pressionado ao responder a pergunta: "tem certeza que deseja excluir esta semana da Sprint"
 - * ENTÃO o administrador é redirecionado para a tela anterior de semanas cadastradas

Figura 62 – Excluir semana do Sprint

7. Serviço de controle de horas

Logar em horas nas macro atividades

* História

- * COMO um usuário cadastrado
- * EU GOSTARIA DE dar início a uma macro atividade
- * PARA CONSEGUIR contabilizar minhas horas durante meu trabalho em tal macro atividade

* Cenário principal

- * DADO que uma macro atividade tenha suas devidas informações presentes no sistema
- * QUANDO o usuário clicar em "macro atividades" e, posteriormente, selecionar uma macro atividade e clicar em "logar"
- * ENTAO o usuário dá início ao ponto eletrônico, logando em uma macro atividade e, assim, contabilizando suas horas naquela determinada macro atividade

* Cenário alternativo

- * DADO que um usuário cadastrado clique no botão "macro atividades", mas ele desiste de acessá-las por algum motivo
- * QUANDO o botão "voltar" for pressionado
- * ENTAO o usuário é redirecionado para a tela anterior de projetos cadastrados

Deslogar em horas nas macro atividades

* História

- * COMO um usuário cadastrado
- * EU GOSTARIA DE dar fim ao login em uma macro atividade
- * PARA CONSEGUIR encerrar a contabilização de horas quando eu não estiver trabalhando mais na macro atividade

* Cenário principal

- * DADO que uma macro atividade tenha suas devidas informações presentes no sistema
- * QUANDO o usuário clicar em "macro atividades" e, posteriormente, selecionar uma macro atividade e clicar em "deslogar"
- * ENTAO o usuário dá fim ao ponto eletrônico, deslogando em uma macro atividade e, assim, finalizando a contabilização das suas horas naquela determinada macro atividade

* Cenário alternativo

- * DADO que um usuário cadastrado clique no botão "macro atividades", mas ele desiste de acessá-las por algum motivo
- * QUANDO o botão "voltar" for pressionado
- * ENTAO o usuário é redirecionado para a tela anterior de projetos cadastrados

Figura 63 – Logar em horas nas macroatividades

Visualizar porcentagem das macro atividades

* História

- * COMO um usuário cadastrado
- * EU GOSTARIA DE visualizar a porcentagem de horas com relação a determinada macro atividade
- * PARA CONSEGUIR ter acesso a quantidade de horas já trabalhadas em tal macro atividade

* Cenário principal

- * DADO que uma macro atividade tenha suas devidas informações presentes no sistema e seja selecionada
- * QUANDO o botão "porcentagem" for pressionado
- * ENTAO mostrará para o usuário o percentual daquela determinada macro atividade, ou seja, mostra a porcentagem do que já foi feito e o que falta fazer em relação às macro atividades do projeto

* Cenário alternativo

- * DADO que uma macro atividade tenha suas devidas informações presentes no sistema e seja selecionada
- * QUANDO o botão "porcentagem" for pressionado e o usuário terminar de visualizar a porcentagem de tal macro atividade
- * ENTAO ele clica em "voltar" e ele é redirecionado para a tela de macro atividades cadastradas

Figura 64 – Visualizar porcentagem das macroatividades

8. Serviço de Geração de Relatórios

Gerar Relatório de dedicação relativa de cada colaborador

- * História
 - * COMO um administrador cadastrado
 - * EU GOSTARIA DE produzir um gráfico de dedicação relativa de cada colaborador no intervalo de tempo escolhido
 - * PARA CONSEGUIR coletar informações importantes para o BOPE
- * Cenário principal
 - * DADO que um usuário selecione a aba "relatórios" e selecione tanto o(s) usuário(s) quanto o intervalo de tempo desejado para geração de tal relatório
 - * QUANDO o link "gerar relatório de dedicação relativa de cada colaborador" for acessado
 - * ENTAO um pdf será gerado com tais informações

Gerar Relatório de dedicação relativa de cada equipe

- * História
 - * COMO um administrador cadastrado
 - * EU GOSTARIA DE produzir um gráfico de dedicação relativa de cada equipe
 - * PARA CONSEGUIR coletar informações importantes para o BOPE
- * Cenário principal
 - * DADO que um usuário selecione a aba "relatórios" e selecione tanto o(s) usuário(s) quanto o intervalo de tempo desejado para geração de tal relatório
 - * QUANDO o link "gerar relatório de dedicação relativa de cada equipe" for acessado
 - * ENTAO um pdf será gerado com tais informações

Gerar Relatório de velocidade de cada equipe por semana

- * História
 - * COMO um administrador cadastrado
 - * EU GOSTARIA DE produzir um gráfico de velocidade de cada equipe a cada semana
 - * PARA CONSEGUIR coletar informações importantes para o BOPE a cada semana
- * Cenário principal
 - * DADO que um usuário selecione a aba "relatórios"
 - * QUANDO o link "gerar relatório de velocidade de cada equipe por semana" for acessado
 - * ENTAO um pdf será gerado com tais informações

Figura 65 – Gerar Relatório de dedicação relativa de cada colaborador e equipe, e relatório de velocidade de cada equipe por semana

Gerar Relatório do burndown de cada equipe por semana

- * História
 - * COMO um administrador cadastrado
 - * EU GOSTARIA DE produzir um gráfico do burndown de cada equipe a cada semana
 - * PARA CONSEGUIR coletar informações importantes para o BOPE a cada semana
- * Cenário principal
 - * DADO que um usuário selecione a aba "relatórios"
 - * QUANDO o link "gerar relatório de burndown de cada equipe por semana" for acessado
 - * ENTAO um pdf será gerado com tais informações

Gerar Relatório para mostrar a quantidade de atividades de cada projeto por Sprint(acumulativo)

- * História
 - * COMO um administrador cadastrado
 - * EU GOSTARIA DE produzir um gráfico que mostra a quantidade de atividades de cada projeto a cada Sprint
 - * PARA CONSEGUIR coletar informações importantes para o BOPE a cada Sprint
- * Cenário principal
 - * DADO que um usuário selecione a aba "relatórios"
 - * QUANDO o link "gerar relatório de quantidade de atividades de cada projeto por Sprint" for acessado
 - * ENTAO um pdf será gerado com tais informações

Gerar Relatório final para mostrar a completude e avanço de cada projeto por Sprint(acumulativo)

- * História
 - * COMO um administrador cadastrado
 - * EU GOSTARIA DE produzir um gráfico que mostra a completude de cada projeto a cada Sprint
 - * PARA CONSEGUIR coletar informações importantes para o BOPE
- * Cenário principal
 - * DADO que um usuário selecione a aba "relatórios"
 - * QUANDO o link "gerar relatório de completude e avanço de cada projeto por Sprint" for acessado
 - * ENTAO um pdf será gerado com tais informações

Figura 66 – Gerar relatório do burndown de cada equipe por semana, e relatórios de atividades, completude e avanço de cada projeto por Sprint

APÊNDICE B – Tutorial de implantação do sistema no laboratório iMobilis



Universidade Federal de Ouro Preto - UFOP
Departamento de Computação e Sistemas de Informação - DECSI



TUTORIAL DE IMPLANTAÇÃO DO SISTEMA BOPE SYSTEM

Desenvolvido por: Lorena Souza Costa (lsouzaco@hawk.iit.edu)
Data: 03/08/2017
Professor Orientador: Igor Muzetti Pereira (igormuzetti@gmail.com)

Implantação do sistema BOPE SYSTEM

Configuração Ruby on Rails no Ubuntu 14.04

1. Instalando o Ruby em sua versão 2.4.0

a. Instalando algumas dependências para o Ruby

```
$ sudo apt-get update

$ sudo apt-get install git-core curl zlib1g-dev build-essential libssl-dev libreadline-dev
libyaml-dev libsqlite3-dev sqlite3 libxml2-dev libxslt1-dev libcurl4-openssl-dev python-
software-properties libffi-dev nodejs
```

b. Instalando o Ruby usando o método rbenv.

```
$ cd
$ git clone https://github.com/rbenv/rbenv.git ~/.rbenv
$ echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
$ echo 'eval "$(rbenv init -)"' >> ~/.bashrc
$ exec $SHELL
$ git clone https://github.com/rbenv/ruby-build.git ~/.rbenv/plugins/ruby-build
$ echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc
$ exec $SHELL
$ rbenv install 2.4.0
$ rbenv global 2.4.0
$ ruby -v
```

c. Instalando o Bundler

```
$ gem install bundler
```

Figura 67 – Instalando o Ruby

2. Instalando o Rails em sua versão 5.0.2

a. Instalando um runtime JavaScript como NodeJS

```
$ curl -sL https://deb.nodesource.com/setup_4.x | sudo -E bash -  
$ sudo apt-get install -y nodejs
```

b. Instalando o Rails

```
$ gem install rails --v 5.0.2
```

c. Executar o seguinte comando para tornar os executáveis disponíveis:

```
$ rbenv rehash
```

d. Conferindo a versão instalada:

```
$ rails --v
```

3. Instalando e Configurando o Banco de Dados PostgreSQL

a. Para instalar uma versão recente do PostgreSQL, basta adicionar um novo repositório:

```
$ sudo sh -c "echo 'deb http://apt.postgresql.org/pub/repos/apt/ precise-pgdg  
main' > /etc/apt/sources.list.d/pgdg.list"  
wget --quiet -O - http://apt.postgresql.org/pub/repos/apt/ACCC4CF8.asc | sudo  
apt-key add -  
$ sudo apt-get update  
$ sudo apt-get install postgresql-common  
$ sudo apt-get install postgresql-9.5 libpq-dev
```

b. Para configurar um usuário no banco de dados:

```
$ sudo -u postgres createuser lorena -s  
$ sudo -u postgres psql  
postgres=# \password avrisc90
```

Figura 68 – Instalando o Rails e PostgreSQL

Lembrando que, após essa configuração do PostgreSQL, foi necessário atualizar o arquivo `config/database.yml` da aplicação BOPE SYSTEM para ter o mesmo nome de usuário e senha do banco de dados configurado no passo anterior. Para acessar o sistema basta entrar na pasta `bope` através do comando `cd BOPE SYSTEM` e levantar o servidor com outro comando `foreman start -p 8001`. Agora, já se pode acessar o sistema BOPE SYSTEM colocando no browser o seguinte endereço: `200.239.154.218:8001`.

Figura 69 – Executando a aplicação