

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Matheus Gaya Scandiffio

**Sistema de controle de prazos para os alunos
de pós-graduação**

Uberlândia, Brasil

2017

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Matheus Gaya Scandiffio

**Sistema de controle de prazos para os alunos de
pós-graduação**

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito exigido parcial à obtenção do grau
de Bacharel em Ciência da Computação.

Orientador: Professora Maria Camila Nardini Barioni

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2017

Matheus Gaya Scandiffio

Sistema de controle de prazos para os alunos de pós-graduação

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Uberlândia, Brasil, 02 de agosto de 2017:

**Professora Maria Camila Nardini
Barioni**
Orientador

**Professora Maria Adriana Vidigal
Lima**

Professor Silvio Bacala

Uberlândia, Brasil
2017

“Há dois tipos de pessoas que vão te dizer que você não pode fazer a diferença neste mundo: as que têm medo de tentar e as que têm medo de que você se dê bem” – Ray Goforth, executivo

Resumo

Este trabalho aborda os aspectos teóricos e práticos necessários para o desenvolvimento de um sistema de alertas e controle de alunos matriculados na pós-graduação da Universidade Federal de Uberlândia. Apesar de ser um trabalho mais técnico, este TCC aborda vários aspectos de engenharia de software, que são necessários para o início do desenvolvimento do sistema. Serão discutidos diversos pontos-chave, como metodologias, arquitetura, funcionalidades e outros aspectos relevantes para a criação de um software com qualidade.

Palavras-chave: engenharia de software, desenvolvimento web, metodologias, requisitos.

Lista de ilustrações

Figura 1 – Visão geral dos diagramas da UML - Martinez (2016)	15
Figura 2 – Visão da arquitetura do sistema	20
Figura 3 – Exemplo de uma aplicação utilizando o material design. Destaque para as cores vivas, botões de ação chamativos e as sombras no card superior, dando a impressão de elevação em relação ao restante do conteúdo. . .	23
Figura 4 – Diagrama de caso de uso do sistema mostrando as funcionalidades do sistema do ponto de vista do usuário	28
Figura 5 – Diagrama de classe do sistema, mostrando os atributos, métodos e relacionamento entre as classes do sistema.	29
Figura 6 – Detalhamento das collections do sistema, seus atributos e suas relações	30
Figura 7 – Simulação de requisição para o servidor. Em vermelho (458,00ms), temos destacado o tempo que o servidor levou para receber, processar e responder a requisição.	32
Figura 8 – Tela de apresentação e login do sistema de controle para os alunos de pós-graduação da FACOM.	45
Figura 9 – Tela inicial do sistema.	47
Figura 10 – Tela de orientadores do sistema.	47
Figura 11 – Modal para cadastro de novo orientador.	48
Figura 12 – Tela 1/2 de alunos do sistema.	49
Figura 13 – Tela 2/2 de alunos do sistema.	49
Figura 14 – Modal para criação/edição de um aluno.	51
Figura 15 – Tela de alarmes do sistema.	52
Figura 16 – Tela de cadastro de alarmes do sistema.	53
Figura 17 – Tela de emails enviados.	55

Lista de tabelas

Tabela 1 – Requisito Funcional - Acessar o sistema.	33
Tabela 2 – Requisito Funcional - Gerar token de acesso.	34
Tabela 3 – Requisito Funcional - Cadastrar professor orientador.	34
Tabela 4 – Requisito Funcional - Listar professores orientadores.	35
Tabela 5 – Requisito Funcional - Buscar professor orientador.	35
Tabela 6 – Requisito Funcional - Editar professor orientador.	35
Tabela 7 – Requisito Funcional - Remover professor orientador.	36
Tabela 8 – Requisito Funcional - Cadastrar alarme.	36
Tabela 9 – Requisito Funcional - Listar alarmes.	37
Tabela 10 – Requisito Funcional - Buscar alarmes.	37
Tabela 11 – Requisito Funcional - Editar alarme.	37
Tabela 12 – Requisito Funcional - Remover alarme.	38
Tabela 13 – Requisito Funcional - Cadastrar aluno.	38
Tabela 14 – Requisito Funcional - Listar alunos.	39
Tabela 15 – Requisito Funcional - Buscar aluno.	40
Tabela 16 – Requisito Funcional - Editar aluno.	40
Tabela 17 – Requisito Funcional - Enviar alerta por e-mail.	40
Tabela 18 – Requisito Funcional - Salvar logs.	41
Tabela 19 – Requisito Funcional - Cadastrar administrador.	41
Tabela 20 – Requisito Funcional - Listar administradores.	41
Tabela 21 – Requisito Funcional - Buscar administrador.	42
Tabela 22 – Requisito Funcional - Editar administrador.	42
Tabela 23 – Requisito Funcional - Remover administrador.	42
Tabela 24 – Requisito Funcional - Requisitos não-funcionais gerais.	44

Lista de abreviaturas e siglas

API	<i>Application Programming Interface</i>
CSS	<i>Cascading Style Sheets</i>
HTML	<i>HyperText Markup Language</i>
JS	<i>Javascript</i>
MVC	<i>Model View Controller</i>
REST	<i>Representational State Transfer</i>
SQL	<i>Structured Query Language</i>
UFU	Universidade Federal de Uberlândia
UML	<i>Unified Modeling Language</i>
PPGCO	Programa de Pós-Graduação em Computação

Sumário

1	INTRODUÇÃO	10
	Introdução	10
1.1	Visão Geral	10
	Visão Geral	10
1.2	Justificativas	10
	Justificativas	10
1.3	Objetivos	11
	Objetivos	11
2	REFERENCIAL TEÓRICO	12
2.1	A Engenharia de Software	12
2.2	A UML (<i>Unified Modeling Language</i>)	13
2.2.1	Diagramas Comportamentais	13
2.2.2	Diagramas Estruturais	14
3	TECNOLOGIAS UTILIZADAS	16
3.1	Front-end	16
3.1.1	Javascript	16
3.1.2	Angular.JS	17
3.1.3	HTML 5	17
3.1.4	CSS 3	17
3.2	Back-end	17
3.2.1	Node.JS	18
3.2.2	Express.JS	18
3.2.3	MongoDB	18
3.3	Versionamento	18
4	DESENVOLVIMENTO DO SISTEMA	20
4.1	Arquitetura da aplicação	20
4.1.1	Camada de apresentação	21
4.1.1.1	Importância da UX	21
4.1.1.2	Importância da UI	21
4.1.1.3	O Material Design	22

4.1.2	Camada de aplicação	23
4.1.3	Camada de dados	24
4.2	Metodologia de desenvolvimento	25
4.2.1	O Scrum	25
4.2.1.1	Primeiro <i>sprint</i>	25
4.2.1.2	Segundo <i>sprint</i>	26
4.2.1.3	Terceiro <i>sprint</i>	26
4.3	Requisitos do sistema	26
4.3.1	Diagrama de caso de uso	28
4.3.2	Diagrama de classes	28
4.3.3	Modelagem do banco de dados	29
4.3.4	Requisitos, testes e implantação	30
4.3.4.1	Teste de funcionalidades	31
4.3.4.2	Teste de desempenho	32
4.3.4.3	Teste unitário	32
5	APRESENTAÇÃO DO SOFTWARE	45
5.1	Tela de apresentação e autenticação de usuário	45
5.2	Tela inicial - Dashboard	46
5.3	Gestão de orientadores	47
5.4	Gestão de alunos	49
5.5	Gestão de alarmes	52
5.6	Emails enviados	54
6	CONCLUSÃO	56
6.1	Próximos passos	56
	REFERÊNCIAS	57

1 Introdução

Ano após ano a quantidade de alunos matriculados nos cursos de pós-graduação na UFU (Universidade Federal de Uberlândia) cresce cada vez mais. Devido as várias regras e prazos estabelecidos pela norma da pós-graduação, os alunos acabam ficando confusos e perdidos em relação aos seus prazos e datas limite para entrega de seus trabalhos.

O aluno que não cumpre com os prazos estipulados na norma do programa de pós-graduação, pode até ser desligado do programa, fazendo com que a taxa de desistência fique elevada. Um dos responsáveis pela comunicação com os alunos é o secretário da pós-graduação, que alerta os mesmos para que eles cumpram com as suas obrigações e consigam concluir o curso dentro do prazo estipulado pela norma.

1.1 Visão Geral

Atualmente, é feito um controle manual do vencimento do prazo das atividades dos discentes de pós-graduação. Um secretário fica encarregado de enviar e-mails/alertas aos alunos quando o prazo de vencimento de alguma atividade está próximo, evitando que o aluno seja expulso do programa caso não entregue as atividades devidas. Porém, como a quantidade de alunos matriculados na pós-graduação tem crescido bastante, este controle está ficando complicado e trabalhoso para o secretário.

Este trabalho visa facilitar o controle para o secretário ao criar uma aplicação para controlar o gerenciamento de prazos de atividades de discentes do programa de Pós-graduação em Ciência da Computação da Universidade Federal de Uberlândia (UFU).

1.2 Justificativas

Tanto no programa de mestrado quanto no programa de doutorado, o aluno e o professor responsável precisam se preocupar com diversas datas, que quando não são cumpridas, podem gerar o desligamento do discente do programa.

Este trabalho foi desenvolvido justamente para evitar que isso ocorra. Com o envio de alertas automatizados quando algum prazo estiver próximo do vencimento, o aluno e seu professor responsável vão ficar atentos e cumprir os requerimentos para continuar no programa.

Apesar do secretário já fazer este controle de forma manualmente, um controle automatizado vai resultar em mais tranquilidade para o secretário (que não vai precisar mais ficar se preocupando com os prazos de cada aluno), e também uma maior confiança

no envio dos e-mails, uma vez que os alertas rodam automaticamente e não precisam ser enviados manualmente pelo secretário.

1.3 Objetivos

O objetivo deste trabalho de conclusão de curso foi aplicar o conhecimento adquirido nas disciplinas cursadas ao longo do curso na criação de uma aplicação web para gerenciar os prazos de vencimento das atividades dos alunos da pós-graduação e enviar alertas automáticos pré-configurados, avisando os discentes e os professores responsáveis pelas atividades do vencimento dos prazos. Sendo assim, o secretário de pós-graduação terá uma forma simples e eficiente de alertar os alunos.

Espera-se que a criação deste sistema e a automatização do envio de e-mail aos alunos e professores possa contribuir para diminuir a taxa de desistência e desligamento, pois os alunos estarão cientes dos prazos e deveres que devem cumprir (uma vez que o regulamento é extenso e algumas vezes acaba confundindo o aluno).

No sistema o secretário poderá acompanhar a quantidade de alertas enviados, orientadores e alunos cadastrados. Dessa forma ele consegue visualizar e parametrizar a quantidade de e-mails que o sistema está enviando para os alunos e professores.

Além da parte de programação necessária para o desenvolvimento do sistema, o trabalho também possui um enfoque na área de engenharia de software para que seja desenvolvido um sistema bem documentado, de qualidade e de fácil expansão caso alguém queira adicionar novas funcionalidades ao portal.

2 Referencial Teórico

Neste capítulo serão apresentados os principais conceitos utilizados no desenvolvimento do sistema, bem como a sua análise de requisitos. Também serão explicadas algumas definições para o entendimento da modelagem do sistema.

2.1 A Engenharia de Software

A Engenharia de Software é a área da computação que tem como objetivo especificar, desenvolver, e projetar como será feito um determinado sistema. O objetivo final é a gerência de projetos, visando deixar os processos bem definidos e organizados, buscando otimizar a produtividade e qualidade do software a ser desenvolvido.(Falbo (2005)). Os seus fundamentos envolvem a utilização de modelos abstratos precisos que permitem ao gerente de projetos ou desenvolvedor especificar, projetar, implementar e manter sistemas de software, sempre avaliando e garantindo a sua qualidade.

A engenharia de software não se limita apenas aos aspectos técnicos de desenvolvimento de um software, mas também engloba as atividades de gerenciamento de projetos de software e o desenvolvimento de métodos e teorias que ajudem e otimizem a produção do software a ser realizado.

Segundo Sommerville (2007), a engenharia de software se depara com 3 desafios chaves, sendo eles:

1. **Desafio de entrega:** Existem muitas técnicas tradicionais para o desenvolvimento de software, porém elas precisam de muito tempo para obter a qualidade desejada. O desafio da entrega se preocupa justamente com isso, ou seja, reduzir os tempos de entrega de sistemas grandes e complexos sem prejudicar a qualidade final.
2. **Desafio da confiança:** Atualmente os softwares estão presentes em praticamente todos os aspectos de nossas vidas. Diariamente fazemos a utilização de vários tipos distintos de software. O desafio da confiança se preocupa em desenvolver técnicas que façam com que os usuários se sintam confortáveis e confiem nos software a sua volta.
3. **Desafio da heterogeneidade:** Os sistemas de software precisam interagir com sistemas distribuídos e sistemas legados (sistemas antigos que ainda estão em produção). O desafio da heterogeneidade se preocupa em criar técnicas para o desenvolvimento de sistemas mais flexíveis e confiáveis que se adaptem facilmente aos outros sistemas já existentes.

A qualidade de um software não é medida somente pela complexidade ou utilidade de um sistema. A qualidade é refletida por seu comportamento quando em produção, a estrutura de pastas e arquivos, a organização do código fonte e toda a documentação relacionada com o seu desenvolvimento. Tais atributos de um sistema são classificados como atributos não funcionais. Uma estrutura bem definida, um código bem escrito e uma documentação atualizada são essenciais e indispensáveis para a manutenção e qualidade de software.

2.2 A UML (*Unified Modeling Language*)

A UML, ou em português "Linguagem de Modelagem Unificada" é um padrão de linguagem estabelecido para a modelagem de diversos aspectos que abrangem o estágio anterior ao desenvolvimento de software. (OOCH G; RUMBAUGH (2008))

Antigamente, eram utilizados 3 métodos para fazer esse tipo de modelagem, que era a utilização dos métodos do BOOCH, OMT (*Rumbaugh*) e OOSE (*Jacobson*). A UML unificou esses 3 métodos.

Dessa forma, a UML tem como principal objetivo ajudar a visualizar de forma mais clara como os objetos irão se comunicar entre si, ajudando os desenvolvedores a visualizarem os produtos de seu trabalho em diversos diagramas e esquemas visuais.

Com a UML é possível fazer uma representação do software através dos diagramas e modelos, porém ela não possui um processo que define como o trabalho deve ser desenvolvido. O foco é descrever "como fazer", "o que fazer", e "por que deve ser feito". (Vargas (2016))

Os diagramas utilizados para a representação do software pela UML são divididos em duas partes, os diagramas estruturais e os diagramas comportamentais. Os principais diagramas que fazem parte da UML estão representados abaixo na figura 1.

2.2.1 Diagramas Comportamentais

1. **De caso de uso:** É um diagrama mais geral e informal, mais utilizado para fases de levantamento e análise de requisitos e funcionalidades do sistema.
2. **De Máquina de Estados:** É um diagrama que tem como foco acompanhar as mudanças que um objeto irá sofrer dentro de um determinado processo.
3. **De Atividades:** É um diagrama que descreve todos os passos que precisam ser realizados para a conclusão de uma certa atividade, fazendo com que todo o processo fique bem claro.
4. **De Interação:** O diagrama de interação é dividido em quatro partes, sendo elas:

- a) **De Sequência:** Mostra a ordem com que os objetos trocam mensagens entre si.
- b) **Geral interação:** Fornece uma visão geral dentro do sistema. É uma variação dos diagramas de atividades.
- c) **De comunicação:** É um diagrama que concentra-se em como os objetos estão vinculados.
- d) **De tempo:** É um diagrama que descreve as mudanças sofridas por uma classe ou instância durante o tempo.

2.2.2 Diagramas Estruturais

1. **De Classe:** É um dos diagramas mais utilizados na UML. O objetivo dele é organizar e demonstrar o conjunto de classes com seus atributos e métodos, além do relacionamento entre as diversas classes do sistema.
2. **De Objeto:** É um complemento do diagrama de classes. O diagrama de objetos mostra os valores guardados pelos diferentes objetos de um diagrama de classe.
3. **De Componentes:** É um diagrama focado nos componentes do software e como eles se relacionam entre si.
4. **De implantação:** Lista todas as necessidades para se fazer a implantação do software, ou seja, todos os requisitos de hardware e características do sistema para suportar o software.
5. **De Pacotes:** É um diagrama que mostra os subsistemas, facilitando a visualização das partes que o compõem.
6. **De Estrutura:** É um diagrama que descreve a estrutura geral do sistema.

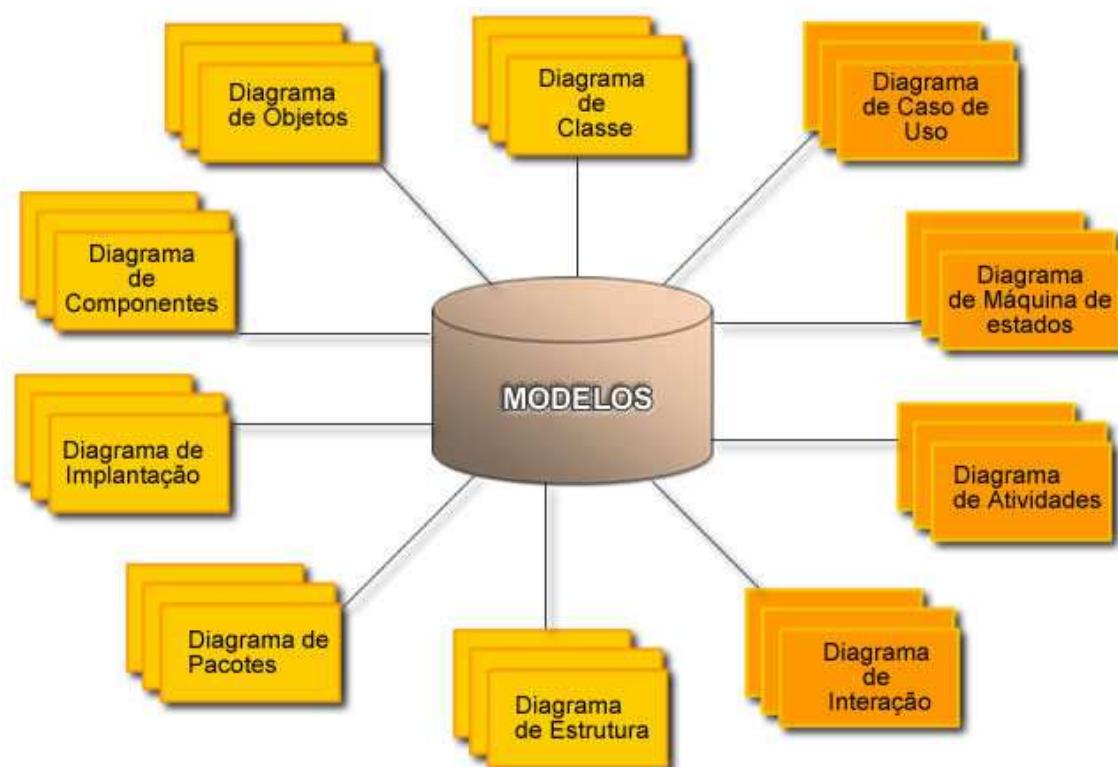


Figura 1 – Visão geral dos diagramas da UML - [Martinez \(2016\)](#)

3 Tecnologias utilizadas

Neste capítulo serão explicadas todas as tecnologias que foram utilizadas para o desenvolvimento do sistema de controle dos alunos de pós-graduação em Ciência da Computação da UFU.

Para a escolha das tecnologias, foi levado em conta a performance, facilidade de uso, facilidade de instalação e facilidade de manutenção da aplicação, uma vez que usuários leigos deverão ter total capacidade de instalar o sistema facilmente.

Todas as tecnologias utilizadas são recentes e *Open Source*, uma vez que depois de desenvolvido, outro aluno ou desenvolvedor poderá realizar manutenção evolutiva no software, ou seja, desenvolvimento de novas funcionalidades caso o secretário ache necessário após um período de uso.

As tecnologias serão divididas em duas categorias, sendo elas:

1. *front-end* - O *front-end* da aplicação é toda a parte do sistema que fica visível para usuário (tela, botões, elementos, etc...) e é responsável pelas interações do mesmo com o sistema.
2. *back-end* - O *back-end* da aplicação é responsável por processar os dados enviados pelo usuário através do *front-end*. O *back-end* é o servidor, que traz os dados solicitados e concentra todo o banco de dados da aplicação.([Lamim \(2014\)](#))

3.1 Front-end

Para o desenvolvimento do *front-end* do sistema de controle dos alunos de pós-graduação em Ciência da Computação da UFU, foram utilizadas as seguintes tecnologias:

1. Javascript
2. Angular.JS
3. HTML 5
4. CSS 3

3.1.1 Javascript

O *Javascript* foi inicialmente desenvolvida por *Brendan Eiché* no ano de 1995 e logo em seguida padronizada pela ECMA, associação internacional especializada na padronização de sistemas). O *Javascript* é uma linguagem de programação interpretada

baseada em *script's*. Uma das principais vantagens do *Javascript* é que ela foi implementada como parte dos navegadores web, fazendo com que os *scripts* sejam executados do lado do cliente (tirando a carga do servidor). (Silva (2017))

3.1.2 Angular.JS

O Angular.JS é um *framework* desenvolvido na linguagem *javascript* que auxilia e facilita o desenvolvimento de sites e sistemas web's. O Angular.JS utiliza o padrão MVC (*Model*, *View* e *Controller*), onde a aplicação é dividida em 3 camadas.

A primeira camada (*Model*) é responsável pela manipulação dos dados da aplicação, ou seja, ela é quem manipula a leitura e escrita de dados nas variáveis e as suas validações. A segunda camada (*View*) provê a interação com o sistema e apenas exibe os dados para o usuário. Por fim, a terceira camada (*Controller*) é a camada responsável por controlar a aplicação. Esta camada recebe todas as solicitações do usuário e executa as ações necessárias. (ALBANI (2011))

O grande diferencial do Angular.JS é que além de ser um projeto *Open Source* apoiado pelo Google, ele permite o desenvolvedor utilizar *tag's* especiais que estão ligadas a diretivas da biblioteca, facilitando e agilizando muito o desenvolvimento de uma aplicação web, uma vez que as diretivas utilizadas pelo Angular.JS deveriam ser implementadas pelo desenvolvedor caso ele não use nenhum *framework* e trabalhe apenas com *javascript* puro.

3.1.3 HTML 5

O HTML 5 (*Hypertext Markup Language*) é uma linguagem utilizada para estruturar e apresentar conteúdo na internet. Os navegadores como *Google Chrome* e *Firefox* por exemplo, interpretam os arquivos HTML's gerando assim as páginas da internet. O HTML 5 trouxe novas funcionalidades em relação a semântica e acessibilidade e ele ajuda muita na manipulação dos dados além de agilizar o desenvolvimento para os desenvolvedores.

3.1.4 CSS 3

O CSS 3 (*Cascading Style Sheets*) é uma linguagem utilizada para manipular a apresentação de documentos escritos em linguagem de marcação, como HTML 5 por exemplo. Através do CSS, os desenvolvedores conseguem criar efeitos, animações, mudar as cores e estilizar os seus sites como quiserem.

3.2 Back-end

Para o desenvolvimento do *back-end* do sistema de controle dos alunos de pós-graduação em Ciência da Computação da UFU foram utilizadas as seguintes tecnologias:

1. Node.JS
2. Express.JS
3. MongoDB

3.2.1 Node.JS

Node.JS é um *framework* desenvolvido em *javascript* com o objetivo de facilitar a construção de aplicações de rede rápidas e escaláveis. Com o Node.JS podemos criar aplicações web utilizando somente código em *javascript*.

Uma das vantagens do Node.JS é que ele utiliza um modelo de entrada/saída direcionada a eventos e não bloqueante, tornando a aplicação leve e eficiente. Por essa agilidade, o Node.JS é ideal para construir aplicações que possuem uma intensa troca de dados em tempo real.

3.2.2 Express.JS

O Express.JS é um *framework* desenvolvido em *javascript* com o objetivo de facilitar o desenvolvimento de aplicações que utilizam Node.JS.

Um dos grandes benefícios do Express.JS, é que além de ser muito flexível e robusto, ele fornece vários recursos voltados para o desenvolvimento de aplicações móveis e web.

3.2.3 MongoDB

MongoDB é um banco de dados orientado a documentos que possui alta performance. Por ser baseado em documentos, o MongoDB fornece aos desenvolvedores uma maneira mais simples de fazer as consultas necessárias no banco (quando comparado a um banco SQL por exemplo).

Outros fatores que levaram a escolha do MongoDB, foi o fato dele ser altamente escalável, possuir uma ótima performance em comparação a bancos SQL, ser *Open Source*, e possuir uma comunidade grande e ativa, com várias documentações e materiais para consulta e aprendizado.

3.3 Versionamento

Visando gerenciar diferentes versões no desenvolvimento de qualquer arquivo do sistema, foi utilizado um sistema de controle de versões de software. Com o versionamento, é possível buscar todas as alterações feitas no projeto, além de garantir uma cópia de

segurança caso o computador em que se está desenvolvendo o sistema pare de funcionar ou se perca. As principais vantagens de se utilizar um controle de versão no desenvolvimento são as seguintes:

1. Trabalho em equipe - Várias pessoas conseguem trabalhar no mesmo projeto, pois o mesmo está na nuvem e as outras pessoas conseguem acompanhar as alterações assim que enviadas pelo outro desenvolvedor.
2. Histórico de alterações - Todas as alterações são gravadas no repositório, fazendo com que uma eventual restauração ou correção seja feita de maneira rápida e eficiente.
3. Criação de ramos - É possível criar diversos ramos no repositório, permitindo com que os desenvolvedores trabalhem simultaneamente em diferentes ramos sem conflito.

Para este projeto, foi criado um repositório privado no *GitHub* através da ferramenta *SourceTree*, que é um software voltado para controle de versão.

4 Desenvolvimento do sistema

Este capítulo irá abordar com mais detalhes como foi feito todo o desenvolvimento do sistema. Será explicado como funcionará a arquitetura da aplicação, ou seja, como foi feita toda a comunicação do *front-end* com o *back-end*, os protocolos e tipos de dados a serem trafegados, os prazos para cada etapa do desenvolvimento e também será explicado todos os requisitos que foram desenvolvidos para a criação do sistema.

4.1 Arquitetura da aplicação

A arquitetura da aplicação consiste em definir quais vão ser os componentes de software e como eles vão interagir entre si para obter o resultado esperado. Não possuir uma arquitetura bem definida pode trazer um re-trabalho muito grande na etapa de desenvolvimento, então é de extrema importância que a arquitetura esteja bem feita e compreendida, pois um dos grandes problemas da ciência da computação é lidar com a complexidade de informações a serem tratadas de modo eficiente.([Shallit \(1995\)](#))

Para o desenvolvimento da aplicação e relacionamento das tecnologias do *front-end* com o *back-end*, foi utilizado uma arquitetura separada em 3 camadas, sendo elas:

1. Camada de apresentação
2. Camada de aplicação
3. Camada de dados

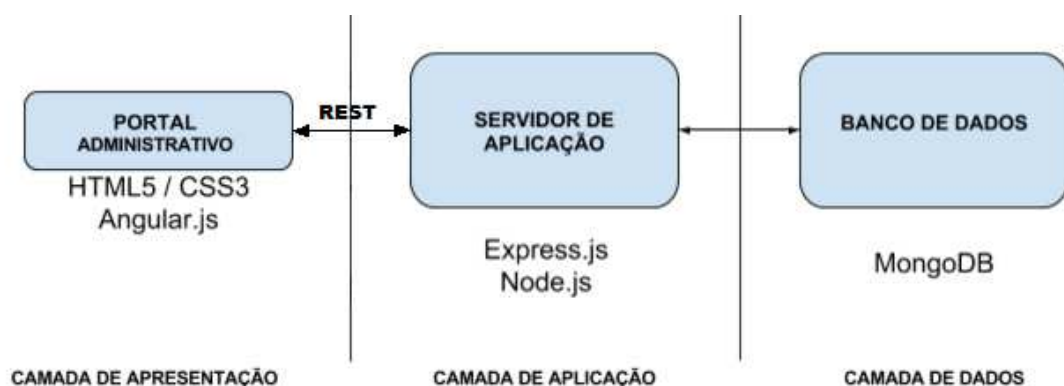


Figura 2 – Visão da arquitetura do sistema

4.1.1 Camada de apresentação

A camada de apresentação é responsável por mostrar os dados na tela para o usuário, sendo a interface que o usuário do sistema vai utilizar para executar as ações desejadas. Essa camada impacta diretamente o usuário. É necessário tomar bastante cuidado ao desenvolver a camada de aplicação, pois o mal desenvolvimento da mesma faz com que o usuário tenha dificuldades ao utilizar o sistema. Quando o usuário tem dificuldades na utilização, ele não vai realizar o fluxo esperado e vai deixar de utilizar o sistema, uma vez que ele vai ter a impressão que o mesmo não funciona ou que é muito complexo de manusear. Na camada de apresentação, temos duas grandes áreas relevantes, sendo elas a *UX (User Experience)* e *UI (User Interface)*, que serão abordadas nas próximas sessões.

4.1.1.1 Importância da UX

A *UX* é responsável pela experiência que o usuário vai sentir ao utilizar o software. Quando um usuário consegue realizar o que deseja sem grandes dificuldades e tudo flui muito naturalmente, dizemos que o software possui uma boa *UX*, pois o usuário vai ter uma experiência boa ao utilizá-lo.

Por exemplo, diversas vezes ao utilizar um determinado software ou ao navegar em alguma página web, nós ficamos "perdidos", buscando onde está o botão para realizar uma certa ação desejada. Quando isso acontece, nós associamos essa dificuldade a realizar a ação como uma má experiência, pois o usuário não está conseguindo fazer a tarefa que ele deseja dentro do software. Caso o usuário tenha uma experiência negativa ao utilizar o seu software, as chances dele voltar e continuar usando ele são muito baixas.

Então além de todo o software estar funcionando como deveria, a interface para o usuário efetivamente usar as funcionalidade devem ser muito claras e objetivas, de forma que o usuário se sinta bem ao utilizar o seu sistema. (Matiola (2017))

4.1.1.2 Importância da UI

A *UI* é toda a parte que vai atingir o usuário ao utilizar o sistema. É tudo aquilo que é lançado a ele como informação, ou seja, os elementos, cores, animações e até sons que o sistema emite ao ser utilizado. São os meios com que o usuário irá interagir com o sistema. Um sistema com cores e elementos bem feitos e desenhados, impacta diretamente na *UX*, pois irá chamar a atenção do usuário para utilizar o sistema e irá facilitar o uso.

O grande objetivo da *UI* é desenvolver uma interface que seja ao mesmo tempo simples, porém única e bonita, que se destaque das demais e chame a atenção do usuário. Ao desenvolver a *UI* de um sistema, é importante levar alguns pontos em consideração:

1. Sempre buscar juntar informações semelhantes, de forma que o usuário saiba dis-

tinguir rapidamente onde está cada coisa que ele deseja.

2. Deixar os botões de ação destacados e acessíveis, de forma a estimular o usuário a completar o fluxo desejado no software
3. Usar cores para facilitar o entendimento, como o verde para ações de confirmação, e vermelho para ações de cancelamento.
4. Destacar as principais informações e mostrar somente o necessário para não poluir a tela com muitas informações e confundir o usuário.
5. Ser objetivo e simples.

O modo com que as pessoas se relacionam com a tecnologia sempre está em constante evolução, então é muito importante que a interface também acompanhe essas mudanças. Não existe uma regra definida de como fazer um sistema com boa *UI*. Uma boa interface hoje, pode parecer antiga e confusa em alguns anos. (LOVI (2017))

4.1.1.3 O Material Design

Visando desenvolver um sistema com boa *UX* e *UI*, foi utilizado o *Material Design* como referência para desenvolvimento da interface. O *Material Design* é um conceito recente, que foi apresentado pelo Google em 25 de Junho de 2014. Apesar de ser relativamente novo, o *Material Design* tem sido utilizado em muitas aplicações.

A principal ideia por trás deste conceito, é a que os elementos dispostos nas telas e suas animações devem ser semelhantes ao mundo real, então toda a parte de sombras e posicionamento é muito bem trabalhada. Por exemplo, caso você queira mostrar uma informação específica para o usuário em forma de alerta, este alerta deve dar a sensação de estar "elevado" na tela. Para isso, as sombras e elevações precisam ser feitas para que o usuário tenha essa impressão.

Todo o sistema foi desenvolvido seguindo as principais recomendações e regras do *Material Design*. Além de ser visualmente agradável (dessa forma obtemos uma boa *UI*), ele também impacta na *UX*, pois como hoje muitas sistemas estão utilizando este padrão, ao entrar no sistema, o usuário já vai se sentir confortável e terá a impressão de estar em um ambiente familiar, ou seja, mesmo acessando pela primeira vez, ele vai ter a impressão de que já sabe onde ficam os elementos e como deve fazer para interagir com o sistema.

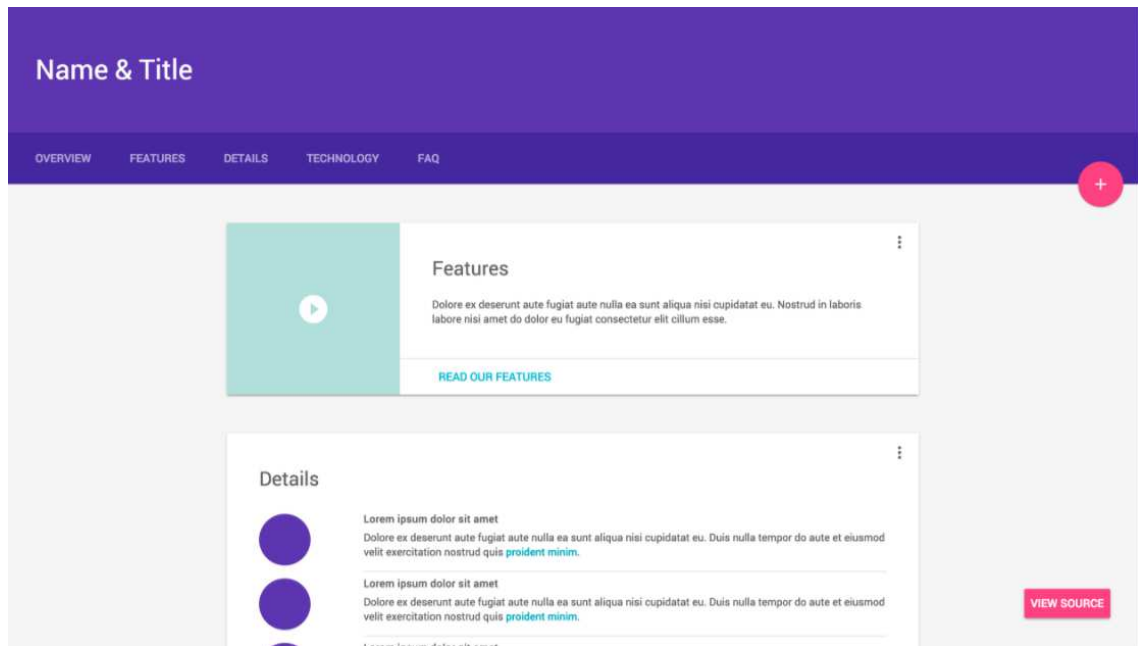


Figura 3 – Exemplo de uma aplicação utilizando o material design. Destaque para as cores vivas, botões de ação chamativos e as sombras no card superior, dando a impressão de elevação em relação ao restante do conteúdo.

4.1.2 Camada de aplicação

A camada intermediária (aplicação) é responsável por ligar a camada de apresentação com a camada de dados. Ela fornece interfaces que são consumidas pela camada de aplicação para carregar informações e mostrar dados para o usuário. Essas interfaces são chamadas de *webservices*.

Por exemplo, ao entrar no sistema, o usuário deseja carregar os usuários cadastrados. Essa informação não se encontra no computador do usuário. Ela se encontra no banco de dados da aplicação. Para obter estes dados, a camada de aplicação disponibiliza um *webservice* a ser consumido. Sendo assim, é feita uma requisição para esta interface disponibilizada.

Ao receber esta requisição, a camada de aplicação dispara uma consulta para a camada de dados, que é onde está o banco de dados do sistema. Ao obter estes dados, a camada de aplicação organiza a informação da forma que for necessária e responde a requisição com os dados obtidos.

Sempre que o sistema precisa obter novos dados ou validar informações com o servidor, são disparadas essas requisições que foram disponibilizadas pela camada de aplicação. Dessa forma, o usuário pode acessar a aplicação de qualquer computador ou qualquer parte do mundo, pois ao acessar, o sistema irá fazer uma requisição e buscar a informação unificada no banco de dados.

Essas requisições feitas para a camada de aplicação demoram alguns mili-segundos

para serem respondidas. Esse tempo pode variar dependendo da quantidade de informação a ser trafegada em cada requisição e dependendo da velocidade de conexão da internet do usuário.

Dessa forma, é sempre necessário buscar trafegar a menor quantidade de informação possível. Sendo assim, todas as consultas para listar dados são paginadas no sistema de 10 em 10 registros. Com isso, o sistema se torna escalável, e não vai demorar muitos segundos para carregar uma página ou mostrar uma informação específica para o usuário.

É muito importante a camada de apresentação levar esse tempo em conta ao fazer uma requisição. Sempre que for disparada uma requisição, é importante mostrar para o usuário um *feedback* visual, como uma tela de carregamento, para ele perceber que a ação solicitada está sendo realizada e não ter a impressão de que o sistema está travado ou que não está funcionando corretamente.

4.1.3 Camada de dados

Para finalizar o *back-end*, existe a camada de dados, que é onde se encontra o banco de dados que armazena as informações necessárias para o funcionamento do sistema. As consultas e operações no banco estão definidas na camada de aplicação.

Por exemplo, quando a camada de aplicação disponibiliza uma interface "listar alunos" para a camada de apresentação consumir, na camada de aplicação temos como os dados devem ser tratados e também a consulta a ser realizada diretamente no banco de dados. A camada de dados então busca os dados baseado na consulta e retorna para a camada de aplicação devolver para a camada de apresentação.

É importante deixar todas as consultas pré-definidas e bloquear qualquer outro tipo de acesso e interação com a camada de dados, visando proteger as informações do banco de dados. Dessa forma, é bloqueado que algum usuário mal intencionado tente roubar alguma informação sigilosa do banco.

4.2 Metodologia de desenvolvimento

A metodologia de desenvolvimento utilizada no desenvolvimento do software é de extrema importância. A metodologia consiste em um conjunto de práticas e métodos que são replicados durante toda a etapa de produção, até a conclusão do software.

Para o desenvolvimento do sistema proposto neste trabalho, foi utilizada uma metodologia de desenvolvimento ágil denominado *Scrum*. Esta foi a metodologia escolhida pois a base do *Scrum* é a transparência durante o processo de desenvolvimento. Outro ponto positivo desta metodologia é a facilidade para adaptação conforme as dificuldades e possíveis alterações de requisitos do sistema. O *Scrum* tem como objetivo realizar entregas rápidas e contínuas, de forma que seja possível acompanhar a evolução e crescimento do projeto de forma mais próxima e detalhada. (Stefanini (2017))

4.2.1 O Scrum

O *scrum* é uma metodologia ágil para o planejamento e gestão do desenvolvimento de um projeto de software. No *scrum*, todo o projeto é dividido em ciclos que são chamados de *sprints*. O *sprint* define um conjunto de tarefas que devem ser realizados até o fim de seu prazo, que costuma variar entre 15 e 30 dias.

Todas as funcionalidades a serem desenvolvidas ficam em uma lista chama *product backlog*. No início de cada *sprint*, é realizada uma reunião para definir quais funcionalidades do *product backlog* entrarão no *sprint* atual. Nessa reunião o chamado *product owner*, que é a pessoa responsável por definir as prioridades no desenvolvimento do sistema, diz quais são os itens a serem priorizados do *product backlog* e que entrarão no *sprint* atual. Baseado nessa lista de demandas apontada pelo *product owner*, os desenvolvedores apontam quais demandas conseguirão atender dentro do prazo do *sprint*. Após a definição das atividades a serem feitas, é iniciado o novo *sprint*, e as atividades desse *sprint* são removidas da lista de *product backlog*.

Ao final de cada *sprint*, os desenvolvedores apresentam as funcionalidades desenvolvidas durante o *sprint* para validação. Também é discutido como foi o processo de desenvolvimento durante o *sprint*, para conseguir identificar o que funcionou bem e o que precisa ser melhorado para o próximo *sprint*.

Ao finalizar um *sprint*, já é iniciado o próximo ciclo com um novo *sprint*. Esse ciclo se repete até a finalização do desenvolvimento do software desejado. (Franco (2017))

4.2.1.1 Primeiro *sprint*

O primeiro *sprint* para o desenvolvimento do sistema teve duração de 30 dias, e consistia em preparar todo o ambiente de desenvolvimento, modelar o banco e também

começar o desenvolvimento dos *webservices* do lado do servidor. Para iniciar o desenvolvimento dos *webservices* foi necessário já definir a modelagem do banco e os relacionamentos entre as *collections* do sistema. Ao finalizar a parte do banco, foram desenvolvidos os *webservices* que o *front-end* precisa consumir para o funcionamento do sistema. Ao finalizar este primeiro *sprint*, por se tratar apenas de aspectos mais técnicos e não visíveis para o usuário, foi feita uma reunião apenas para conferir se os dados armazenados e as relações atendiam a necessidade do sistema. Ao finalizar o primeiro *sprint*, foi iniciado na semana seguinte o segundo *sprint* para o desenvolvimento do sistema.

4.2.1.2 Segundo *sprint*

Com o banco modelo, e os *webservices* desenvolvidos, o segundo *sprint*, que também teve duração de 30 dias, foi mais focado na parte do desenvolvimento do *front-end* do sistema. Sendo assim, foi iniciada a página que o secretário vai acessar para fazer a gestão do sistema, cadastrar alunos, cadastrar professores, cadastrar alarmes e acompanhar os e-mails disparados. Foi desenvolvido o layout do sistema e já foram feitas as integrações com os *webservices* desenvolvidos no primeiro *sprint*. Ao finalizar este *sprint*, foi marcada uma reunião com o secretário da pós para ele visualizar como estava ficando o sistema e colher alguns *feedback's*. Baseado nesses *feedback's*, foi elaborado o terceiro *sprint* para o desenvolvimento do sistema.

4.2.1.3 Terceiro *sprint*

Após colher os *feedback's* do secretário da pós, o terceiro *sprint* foi mais focado em aplicar os ajustes e finalizar algumas partes do sistema que ficaram faltando. O terceiro *sprint*, da mesma forma que os anteriores, também teve duração de 30 dias. O resultado deste *sprint* foi uma primeira versão beta do sistema, onde o secretário já poderia utilizar o sistema.

4.3 Requisitos do sistema

Considerando as necessidades apresentadas pelo secretário da pós-graduação e pela coordenadora do PPGCO professora Maria Camila, foram especificados todos os requisitos funcionais e não funcionais do sistema de controle de alunos da pós-graduação.

O levantamento dos requisitos é de extrema importância, pois deixa claro para o cliente (nesse caso o secretário da pós-graduação) todas as funcionalidades do sistema. Apenas os requisitos não são suficientes para o desenvolvimento do sistema, mas eles são muito úteis e importantes para o correto entendimento do funcionamento do sistema.

Os requisitos de um sistema podem ser separados em dois grupos, os requisitos funcionais e requisitos não funcionais.

1. Requisitos funcionais definem as funcionalidades do sistema e como o sistema deve reagir em determinadas situações.
2. Requisitos não funcionais são restrições sobre os serviços ou funções existentes no sistema, como por exemplo restrições de tempo, sobre o processo de desenvolvimento. (Fialho (2000)).

No mestrado, o aluno precisa honrar com os seguintes prazos:

1. Tempo máximo no programa (24 meses, podendo solicitar extensão e aumentar o prazo em até 6 meses, totalizando 30 meses).
2. Projeto de dissertação (O aluno precisa concluir o projeto até o final do segundo semestre letivo).
3. Solicitação para defesa (O aluno precisa solicitar com 30 dias de antecedência do prazo final).
4. Aprovação com restrições (O aluno terá até 30 dias após a defesa para entregar o projeto corrigido).
5. Exame de língua estrangeira (Deve ser feito antes do prazo máximo do programa).

Já no doutorado, o aluno precisa honrar com os seguintes prazos:

1. Tempo máximo no programa (48 meses, podendo solicitar extensão e aumentar o prazo em até 12 meses, totalizando 60 meses).
2. Projeto de tese (O aluno precisa concluir o projeto até o final do terceiro semestre letivo).
3. Exame de qualificação (O aluno precisa concluir o exame até 36 meses após o ingresso no programa).
4. Exame de qualificação caso reprove (O aluno precisa concluir o exame novamente em até 3 meses).
5. Solicitação para defesa (O aluno precisa solicitar com 30 dias de antecedência).
6. Reprovação (O aluno precisa esperar 3 meses para realizar a nova defesa e não pode estourar o prazo máximo do programa).
7. Exame de língua estrangeira (Deve ser feito antes do prazo máximo do programa).

4.3.1 Diagrama de caso de uso

Visando representar as funcionalidades do sistema desenvolvido do ponto de vista do usuário, foi criado o diagrama de caso de uso do sistema, representado na figura 4. Através do diagrama de caso de uso, o usuário consegue ver de maneira rápida e simples as principais funcionalidades do sistema e como ele funciona.

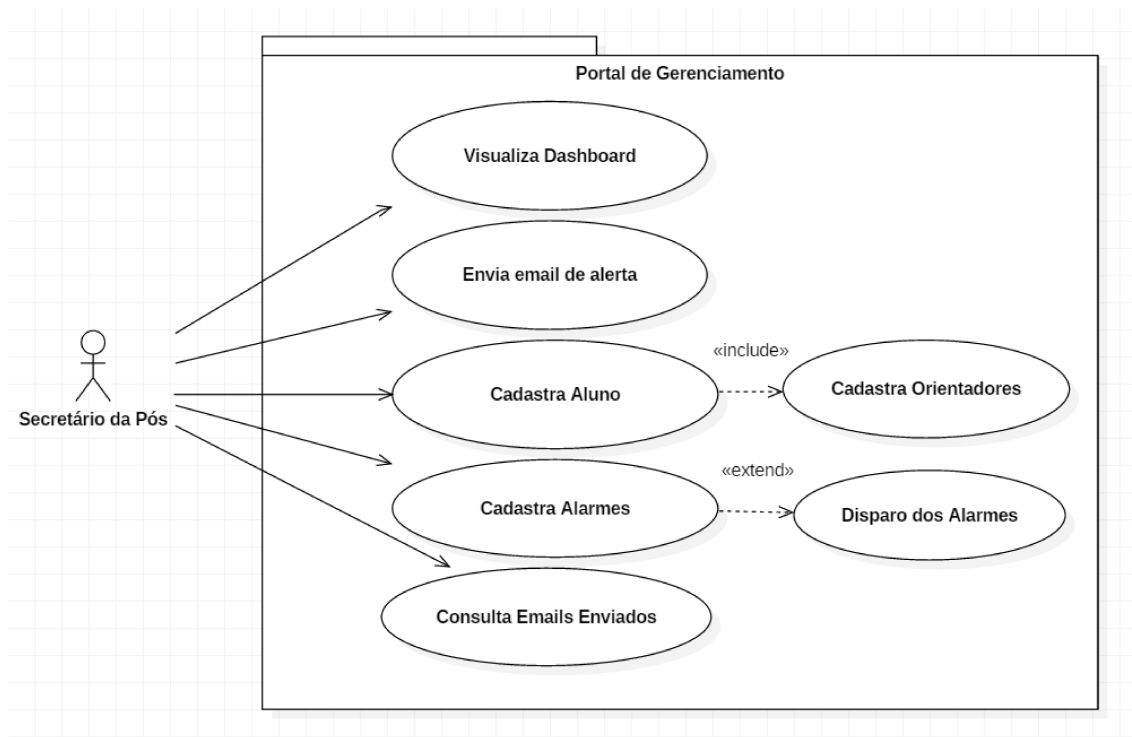


Figura 4 – Diagrama de caso de uso do sistema mostrando as funcionalidades do sistema do ponto de vista do usuário

4.3.2 Diagrama de classes

Para representar as classes do sistema desenvolvido e as suas relações entre si, foi criado o diagrama de classes, representado na figura 5. Através do diagrama de classes, o usuário consegue identificar como o sistema está estruturado e os relacionamentos entre as classes. Dessa forma, ele saberá exatamente quais os métodos e atributos de cada classe.

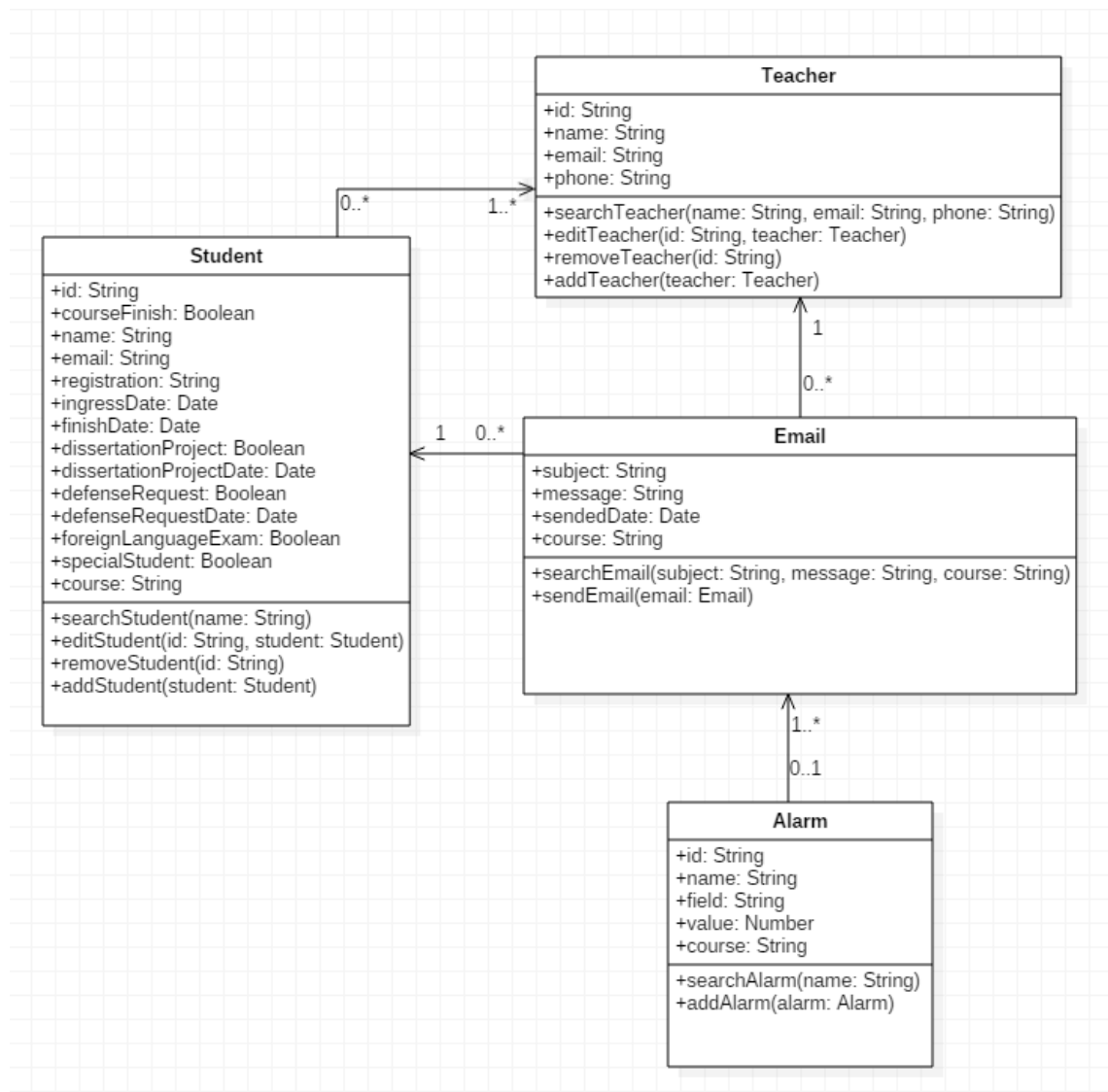


Figura 5 – Diagrama de classe do sistema, mostrando os atributos, métodos e relacionamento entre as classes do sistema.

4.3.3 Modelagem do banco de dados

Para modelar o banco de dados, primeiro foi necessário entender exatamente quais seriam os dados a serem armazenados. Baseado na descrição do sistema, foi possível criar um modelo que atendesse as necessidades.

De acordo com a conversa com o secretário da pós e com a coordenadora do PPGCO professora Maria Camila, foi definido que cada aluno precisaria ter um nome, email para contato e envio de alarmes, curso, data de ingresso no curso, data prevista de término do curso, professor orientador, e as respectivas datas para conclusão das etapas de cada curso. O orientador do aluno precisa conter o nome do orientador, email para contato e envio de alarmes e telefone para contato caso o secretário precise conversar algo mais urgente. Os alarmes são disparados automaticamente, porém são cadastrados

pelo usuário do sistema com nome para fácil identificação, curso que vai ser enviado, qual etapa do programa que vai ser notificada (por exemplo: projeto de dissertação, solicitação de qualificação, entre outros...), dias antes do vencimento da etapa para notificar, título do email a ser enviado, e o conteúdo do email. Para cada email enviado, é necessário armazenar o título do email enviado, o corpo do email, o curso para que foi enviado e os alunos que receberam este email.

Baseado nos requisitos do sistema listados abaixo, e nas funcionalidades que o sistema precisa atender, foi modelado o banco de dados segundo a figura 6. A figura mostra as *collections* presentes no sistema, os dados de cada uma, e também as suas relações entre si.

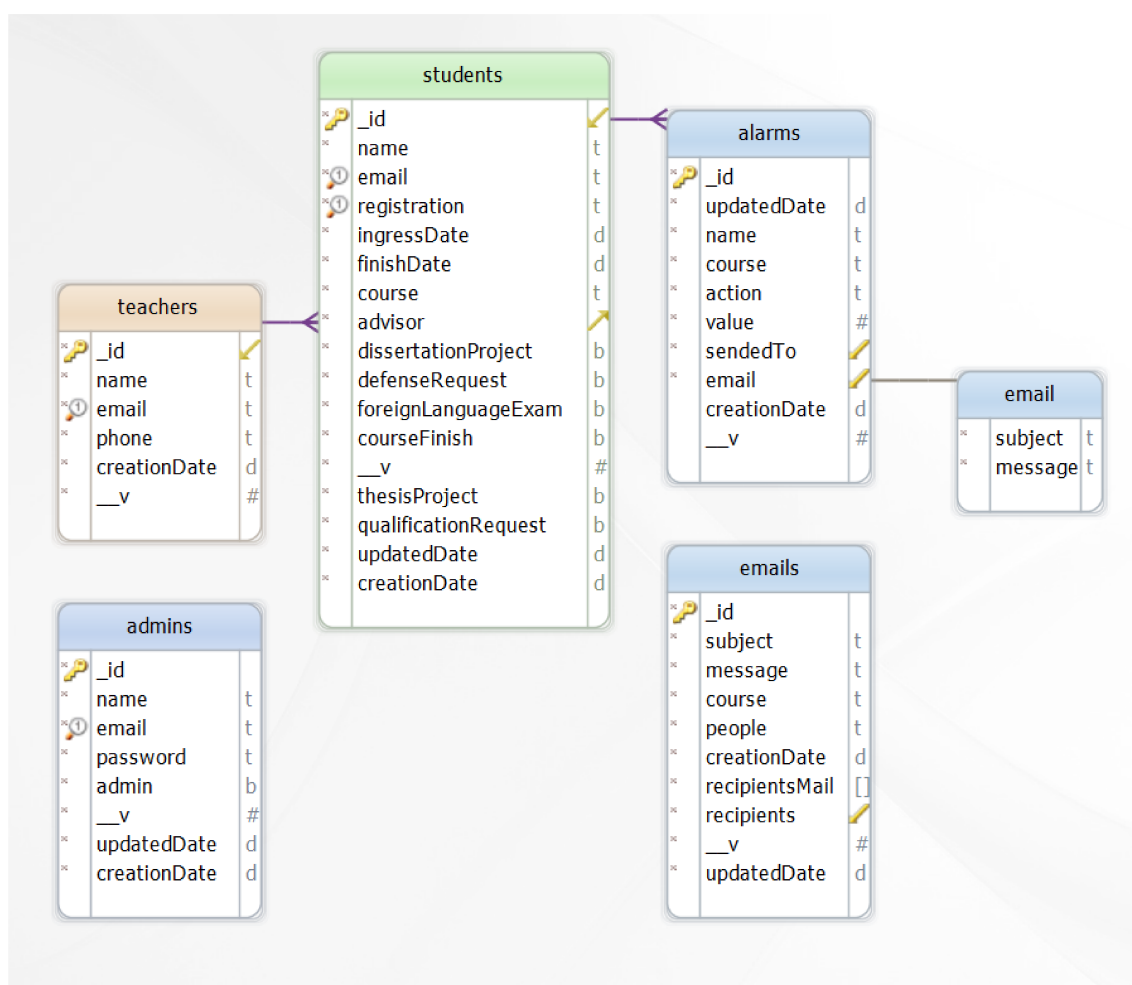


Figura 6 – Detalhamento das collections do sistema, seus atributos e suas relações

4.3.4 Requisitos, testes e implantação

Após o desenvolvimento do sistema, foi repassado cada um dos requisitos do sistema que foram detalhados nas tabelas abaixo (tabelas 1 a 14) durante a utilização do sistema. Esse repasse tinha como objetivo verificar se os detalhes e regras dos requisi-

tos foram respeitados e obedecidos durante a implementação. Para considerar o requisito como válido, foi necessário verificar se o funcionamento era igual ao descrito no requisito e se o sistema também estava respeitando os requisitos não funcionais de cada requisito.

Para disponibilizar o sistema, foi necessário hospedar o servidor em alguma máquina. Para facilitar este processo, o sistema foi hospedado na *Amazon*. O serviço utilizado para isso foi o *Amazon Elastic Compute Cloud* (EC2). O EC2 é uma solução de *cloud computing* da *Amazon* que permite o aluguel de máquinas virtuais para rodar a aplicação desejada. Como todo o código estava em um controle de versão, foi necessário apenas criar a instância na *Amazon*, puxar o código fonte e rodar a aplicação. ([Amazon EC2 \(2017\)](#))

Foram executados 3 tipos de testes para validação do sistema, sendo eles:

1. Teste de funcionalidades
2. Teste de desempenho
3. Teste unitário

4.3.4.1 Teste de funcionalidades

O principal objetivo ao se executar um teste de funcionalidades é avaliar toda a interface do sistema desenvolvido. Durante este teste, foi avaliado os textos, as ações dos botões, links entre as telas, entre outros. Ao executar este teste, garantimos que a interface está funcionando corretamente, e que o usuário não irá enfrentar problemas como links que não funcionam e botões que não executam ação alguma. O tempo para resposta de uma requisição pode ser conferido na figura abaixo.

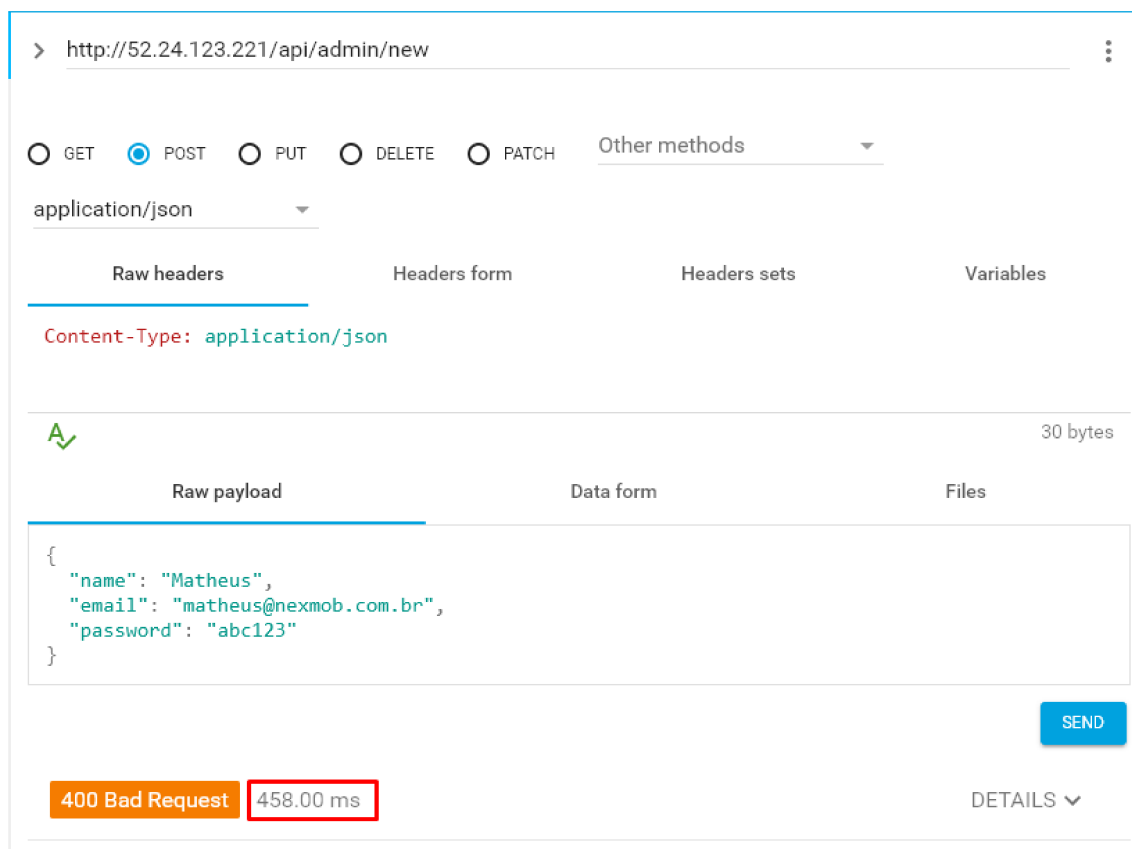


Figura 7 – Simulação de requisição para o servidor. Em vermelho (458,00ms), temos destacado o tempo que o servidor levou para receber, processar e responder a requisição.

4.3.4.2 Teste de desempenho

O principal objetivo ao se executar um teste de desempenho é avaliar a performance do sistema. Durante este teste, foi verificado se as requisições para o servidor estão respondendo em tempo hábil. Por se tratar de uma aplicação simples, que não vai possuir um grande número de usuários simultâneos ou sobrecarga de requisições, uma instância *micro* da *Amazon* suportou tranquilamente o fluxo. Caso a máquina fique sobrecarregada, será necessário melhorar a instância para obter um melhor desempenho. Em média, as requisições demoram 450 ms para serem respondidas para o cliente, ou seja, um tempo razoável e que não gera uma espera grande para o usuário.

4.3.4.3 Teste unitário

O principal objetivo ao se executar um teste de unitário é avaliar todas as unidades que formam o software. Ou seja, foram executados testes nas ações de professor (criar, editar e buscar por exemplo), alunos, alarmes e emails. Ao executar este teste, foi verificado que todas as partes do software estão funcionando conforme o esperado.

Após a execução das rotinas de testes e validação, o sistema foi liberado para o secretário da pós começar a utilização.

Requisito Funcional 1	Acessar o sistema (Login)	Oculto	()	
Descrição				
Este requisito funcional permite ao usuário do sistema se autenticar na plataforma. Este login será feito a partir de um e-mail e senha pré-cadastrados. Caso o usuário digite um e-mail ou senha incorretos, será mostrada a mensagem “Usuário e/ou senha incorretos. Por favor, tente novamente.”.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF1.1 - Formulário único	Todos os dados para o login do usuário no sistema devem ser apresentados para o usuário em um formulário único.	Interface		X
NF1.2 – Criptografia	A senha do usuário no banco de dados é criptografada, provendo maior segurança ao sistema.	Segurança		X

Tabela 1 – Requisito Funcional - Acessar o sistema.

Requisito Funcional 2	Gerar token de acesso	Oculto	(X)	
Descrição				
Este requisito funcional permite que o sistema crie um Token de acesso ao usuário após a confirmação da autenticação no Login. Este token identifica o usuário e provê acesso as funcionalidades do sistema. Este Token é passado para o servidor em todas as requisições que exigem o usuário autenticado.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF2.1 – Criptografia	O Token é gerado a partir de um algoritmo criptográfico, provendo segurança ao sistema.	Segurança		X

Tabela 2 – Requisito Funcional - Gerar token de acesso.

Requisito Funcional 3	Cadastrar professor orientador	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário cadastre um novo professor orientador no sistema. Para o cadastro de um novo professor orientador, o usuário precisa informar os seguintes dados (Nome completo e e-mail).				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF3.1 – Formulário único	Todos os dados para o cadastro de um novo professor orientador no sistema devem ser apresentados para o usuário em um formulário único.	Interface		X

Tabela 3 – Requisito Funcional - Cadastrar professor orientador.

Requisito Funcional 4	Listar professores orientadores	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário visualize todos os professores orientadores cadastrados no sistema em forma de lista.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF4.1 – Paginação dos resultados	A lista será feita de forma paginada visando uma utilização mais simples para o usuário e melhor desempenho para a aplicação.	Interface / Desempenho		X

Tabela 4 – Requisito Funcional - Listar professores orientadores.

Requisito Funcional 5	Buscar professor orientador	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário pesquise por um professor orientador cadastrado no sistema. Essa busca poderá ser feita por nome ou e-mail do professor orientador.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
Não há	-	-		

Tabela 5 – Requisito Funcional - Buscar professor orientador.

Requisito Funcional 6	Editar professor orientador	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário edite as informações de um professor orientador cadastrado.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF6.1 – Formulário único	Todos os dados para a edição de um professor orientador no sistema devem ser apresentados para o usuário em um formulário único.	Interface		X

Tabela 6 – Requisito Funcional - Editar professor orientador.

Requisito Funcional 7	Remover professor orientador	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário remova um professor orientador cadastrado no sistema.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
Não há	-	-		

Tabela 7 – Requisito Funcional - Remover professor orientador.

Requisito Funcional 8	Cadastrar alarme	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário cadastre um novo alarme. O alarme irá enviar um e-mail para os alunos e professores baseado em sua regra. Ao cadastrar o alarme o usuário precisará informar o nome do alarme, a mensagem a ser exibida, se o alarme é para aluno especial de mestrado, aluno regular de mestrado, aluno regular de doutorado ou uma combinação desses tipos de alunos, a quantidade de alarmes a serem enviados e o prazo com que cada alarme será disparado. Por exemplo, 4 alarmes para os alunos e seus orientadores de mestrado quando faltar 60, 45, 30 e 15 dias para o vencimento do prazo máximo do aluno no programa. O usuário precisará relacionar o alarme com alguma das variáveis que serão mostradas pelo sistema (prazo para vencimento do tempo do aluno no programa, prazo para entrega do projeto de dissertação, prazo para entrega do projeto de tese, prazo para solicitação da defesa de qualificação de doutorado, prazo para solicitação da defesa, prazo para entrega das correções no caso de aprovação com restrições, prazo para conclusão do exame de língua estrangeira e prazo para matrícula e ajuste de matrícula).				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF8.1 – Formulário único	Todos os dados para o cadastro de um novo alarme no sistema devem ser apresentados para o usuário em um formulário único.	Interface		X

Tabela 8 – Requisito Funcional - Cadastrar alarme.

Requisito Funcional 9	Listar alarmes	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário visualize todos os alarmes previamente cadastrados no sistema em forma de lista.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF9.1 – Paginação dos resultados	A lista será feita de forma paginada visando uma utilização mais simples para o usuário e melhor.	Interface / Desempenho		X

Tabela 9 – Requisito Funcional - Listar alarmes.

Requisito Funcional 10	Buscar alarmes	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário pesquise por um alarme cadastrado no sistema. Essa busca será feita pelo nome do alarme.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
Não há	-	-		X

Tabela 10 – Requisito Funcional - Buscar alarmes.

Requisito Funcional 11	Editar alarme	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário edite as informações de um alarme cadastrado.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF11.1 – Formulário único	Todos os dados para a edição de um alarme no sistema devem ser apresentados para o usuário em um formulário único.	Interface		X

Tabela 11 – Requisito Funcional - Editar alarme.

Requisito Funcional 12	Remover alarme	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário remova um alarme cadastrado no sistema.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
Não há	-	-		

Tabela 12 – Requisito Funcional - Remover alarme.

Requisito Funcional 13	Cadastrar aluno	Oculto	()	
Descrição				
Este requisito funcional permite ao usuário do sistema cadastrar um novo aluno. Para efetuar o cadastro, basta informar os dados do aluno (nome, e-mail, matrícula, se é especial ou não, data de ingresso, data de conclusão, professor orientador, status do projeto de dissertação, status da solicitação de defesa, status do exame de curso de língua e curso). Além dos dados do aluno, o usuário precisa informar se é um aluno de mestrado ou doutorado. O usuário poderá relacionar o aluno com algum regulamento previamente cadastrado no sistema para que o sistema preencha automaticamente as datas que o aluno necessita cumprir. No cadastro o usuário também deverá relacionar o aluno com algum professor orientador previamente cadastrado no sistema. Também será opcional a inclusão de um co-orientador para o aluno durante o seu cadastro.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF13.1 – Formulário único	Todos os dados para o cadastro de um novo aluno devem ser apresentados para o usuário em um formulário único.	Interface / Desempenho		X

Tabela 13 – Requisito Funcional - Cadastrar aluno.

Requisito Funcional 14	Listar alunos	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário visualize todos alunos cadastrados no sistema em forma de lista.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF14.1 – Paginação dos resultados	A lista será feita de forma paginada visando uma utilização mais simples para o usuário e melhor desempenho para a aplicação.	Interface / Desempenho		X

Tabela 14 – Requisito Funcional - Listar alunos.

Requisito Funcional 15	Buscar aluno	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário pesquise por um aluno cadastrado no sistema. Essa busca poderá ser feita por nome, e-mail ou matrícula do aluno.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
Não há	-	-		X

Tabela 15 – Requisito Funcional - Buscar aluno.

Requisito Funcional 16	Editar aluno	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário edite as informações de um aluno cadastrado. Neste requisito o usuário também terá a opção de editar o tempo máximo do aluno no programa (Caso ele solicite extensão).				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF16.1 – For- mulário único	Todos os dados para a edição de um aluno no sistema devem ser apresentados para o usuário em um formulário único.	Interface		X

Tabela 16 – Requisito Funcional - Editar aluno.

Requisito Funcional 17	Enviar alerta por e-mail	Oculto	()	
Descrição				
Este requisito funcional permite que o sistema faça o envio automático dos alertas necessários ao alunos e professores orientadores. A mensagem, prazo e quantidade dos alertas são previamente cadastrados pelo usuário da plataforma.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF17.1 – Tempo de envio.	O e-mail deverá ser enviado para os alunos e professores responsáveis em até 30 segundos.	Desempenho	X	

Tabela 17 – Requisito Funcional - Enviar alerta por e-mail.

Requisito Funcional 18	Salvar logs	Oculto	(X)	
Descrição				
Este requisito funcional permite que o sistema armazene no banco de dados logs de ações feitas pelo administrador do sistema. Com o log de dados, será possível consultar quando foi enviado cada alerta, e pra quem foi enviado.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
Não há	-	-		

Tabela 18 – Requisito Funcional - Salvar logs.

Requisito Funcional 19	Cadastrar administrador	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário cadastre um novo administrador no sistema. Para o cadastro de um novo administrador, o usuário precisa informar os seguintes dados (Nome completo, e-mail e senha para acesso).				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF19.1 – Formulário único	Todos os dados para o cadastro de um novo administrador no sistema devem ser apresentados para o usuário em um formulário único.	Interface		X

Tabela 19 – Requisito Funcional - Cadastrar administrador.

Requisito Funcional 20	Listar administradores	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário visualize todos os administradores cadastrados no sistema em forma de lista.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF20.1 – Paginação dos resultados	A lista será feita de forma paginada visando uma utilização mais simples para o usuário e melhor desempenho para a aplicação.	Interface / Desempenho		X

Tabela 20 – Requisito Funcional - Listar administradores.

Requisito Funcional 21	Buscar administrador	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário pesquise por um administrador cadastrado no sistema. Essa busca poderá ser feita por nome ou e-mail do administrador.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
Não há	-	-		

Tabela 21 – Requisito Funcional - Buscar administrador.

Requisito Funcional 22	Editar administrador	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário edite as informações de um administrador cadastrado.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
NF22.1 – For- mulário único	Todos os dados para a edição de um administra- dor no sistema devem ser apre- sentados para o usuário em um formulário único.	Interface		X

Tabela 22 – Requisito Funcional - Editar administrador.

Requisito Funcional 23	Remover administrador	Oculto	()	
Descrição				
Este requisito funcional permite que o usuário remova um administrador cadastrado no sistema.				
Requisitos Não-Funcionais				
Nome	Restrições	Categoria	Desejável	Permanente
Não há	-	-		

Tabela 23 – Requisito Funcional - Remover administrador.

Requisitos Não-Funcionais Gerais				
Nome	Restrições	Categoria	Desejável	Permanente
RNFG 1 – Tipo de interface	As interfaces do sistema devem ser implementadas usando HTML 5 e CSS 3.	Interface		X
RNFG 2 – Perfis	O sistema terá um único perfil, que será utilizado pelo secretário da pós-graduação. Este perfil tem acesso total a todas as funcionalidades do sistema.	Regra de negócio		X
RNFG 3 – Linguagem de programação	Para a implementação do sistema foi utilizada a linguagem de programação Javascript.	Software		X
RNFG 4 – Framework's de desenvolvimento	Para a implementação do front-end será utilizado o framework Angular.JS, e para o back-end será utilizado o framework Node.JS.	Software		X
RNFG 5 – Banco de dados	O sistema utilizará o banco de dados não relacional MongoDB.	Software		X
RNFG 6 – Licenças	Todas as tecnologias utilizadas para o desenvolvimento da aplicação são grátis e Open Source, ou seja, livres de qualquer custo.	Software		X

Requisitos Não-Funcionais Gerais				
Nome	Restrições	Categoria	Desejável	Permanente
RNFG 7 – Segurança	Todas as informações sigilosas estarão protegidas no banco via criptografia.	Segurança		X
RNFG 8 – Escalabilidade	O sistema suporta uma grande quantidade de alunos cadastros sem uma perda significativa em seu desempenho.	Desempenho		X

Tabela 24 – Requisito Funcional - Requisitos não-funcionais gerais.

5 Apresentação do Software

Este capítulo irá apresentar todas as telas e funcionamento do sistema de controle para os alunos de pós-graduação da FACOM.

Serão apresentadas e explicadas todas as funcionalidades, junto com as suas interfaces e detalhamento para utilização.

5.1 Tela de apresentação e autenticação de usuário

Logo ao acessar a URL onde o sistema está hospedado, o usuário irá se deparar com um formulário de autenticação. Dessa forma, apenas o administrador do sistema pode entrar para manipular as informações e enviar os alertas que forem necessários.

Por conter todos os dados para comunicação com os alunos, é importante que o secretário guarde essa senha em local seguro.



Figura 8 – Tela de apresentação e login do sistema de controle para os alunos de pós-graduação da FACOM.

5.2 Tela inicial - Dashboard

Após a autenticação do usuário no sistema, ele é direcionado para a primeira tela do portal administrativo, chamada de dashboard.

Nesta tela, ele consegue ver 3 informações quantitativas sobre o uso do sistema. Ele acompanha a quantidade de orientadores que estão cadastrados na plataforma, a quantidade de alunos cadastrados e quantos emails já foram disparados pelo sistema. Este contador de emails leva em consideração tanto os emails enviados de maneira individual quando os enviados via alerta automático.

Nesta primeira tela ele tem a opção de enviar um email específico em caso de emergência. Por exemplo, ocorreu algum imprevisto, e as aulas serão canceladas no dia seguinte. Como não é possível cadastrar um alarme para isso (até por que não é um evento que tem uma repetição e deve acontecer várias vezes), ele tem a opção de enviar um email específico. Ele será disparado para todo o grupo de usuários que ele filtrar.

Os filtros disponíveis para o campo "Curso" são:

1. Todos
2. Mestrado
3. Doutorado

Os filtros disponíveis para o campo "Pessoas" são:

1. Todos
2. Somente alunos
3. Somente orientadores

Após selecionar o grupo que deseja enviar a mensagem, o usuário pode escrever o conteúdo do email. Para o conteúdo, foi disponibilizado um editor HTML. Dessa forma, ele consegue enviar imagens, vídeos, gifs e textos para os alunos.

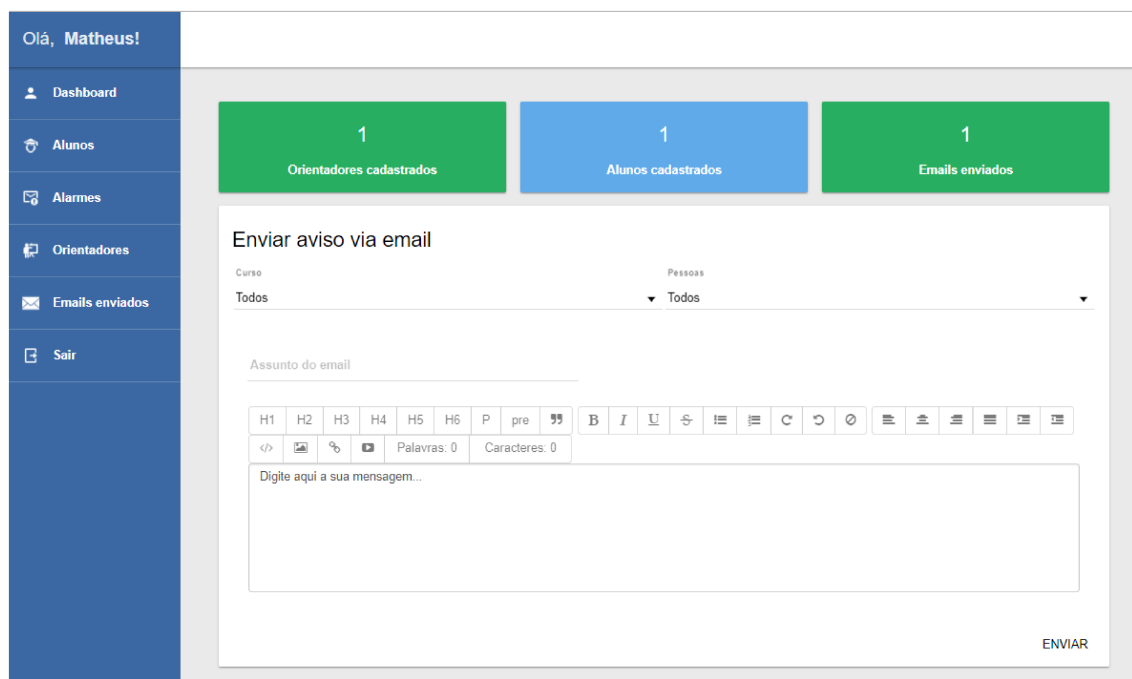


Figura 9 – Tela inicial do sistema.

5.3 Gestão de orientadores

Ao abrir a sessão orientadores, o usuário consegue visualizar todos os orientadores cadastrados na plataforma.

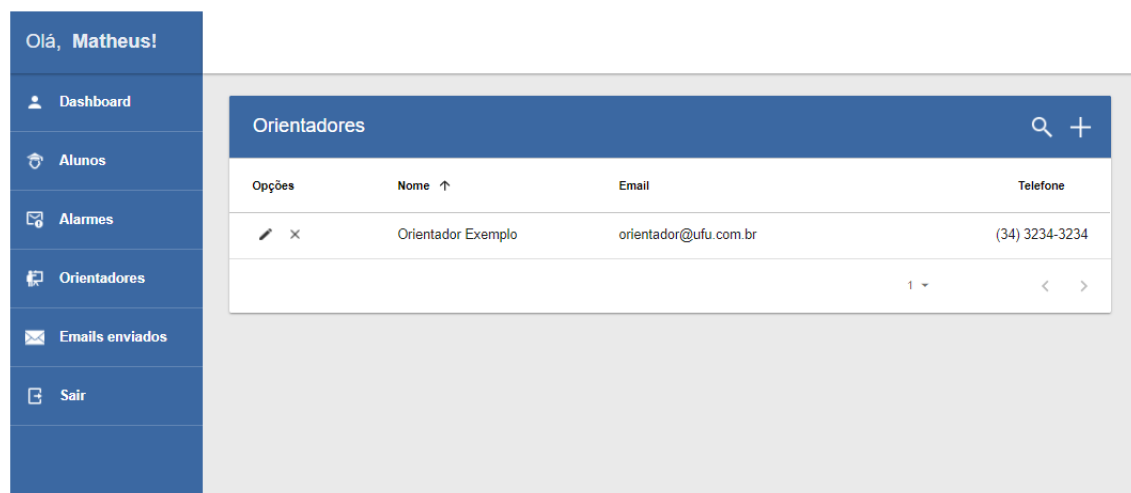


Figura 10 – Tela de orientadores do sistema.

São listados os dados do orientador, sendo eles:

1. Nome
2. Email
3. Telefone para contato

O usuário possui a opção de editar e excluir um orientador previamente cadastrado, ou ele também pode criar um novo administrador.

Ao clicar para criar um novo administrador, será mostrado uma nova janela sobrepondo a tela atual do sistema, requisitando os dados para cadastro de um novo orientador.

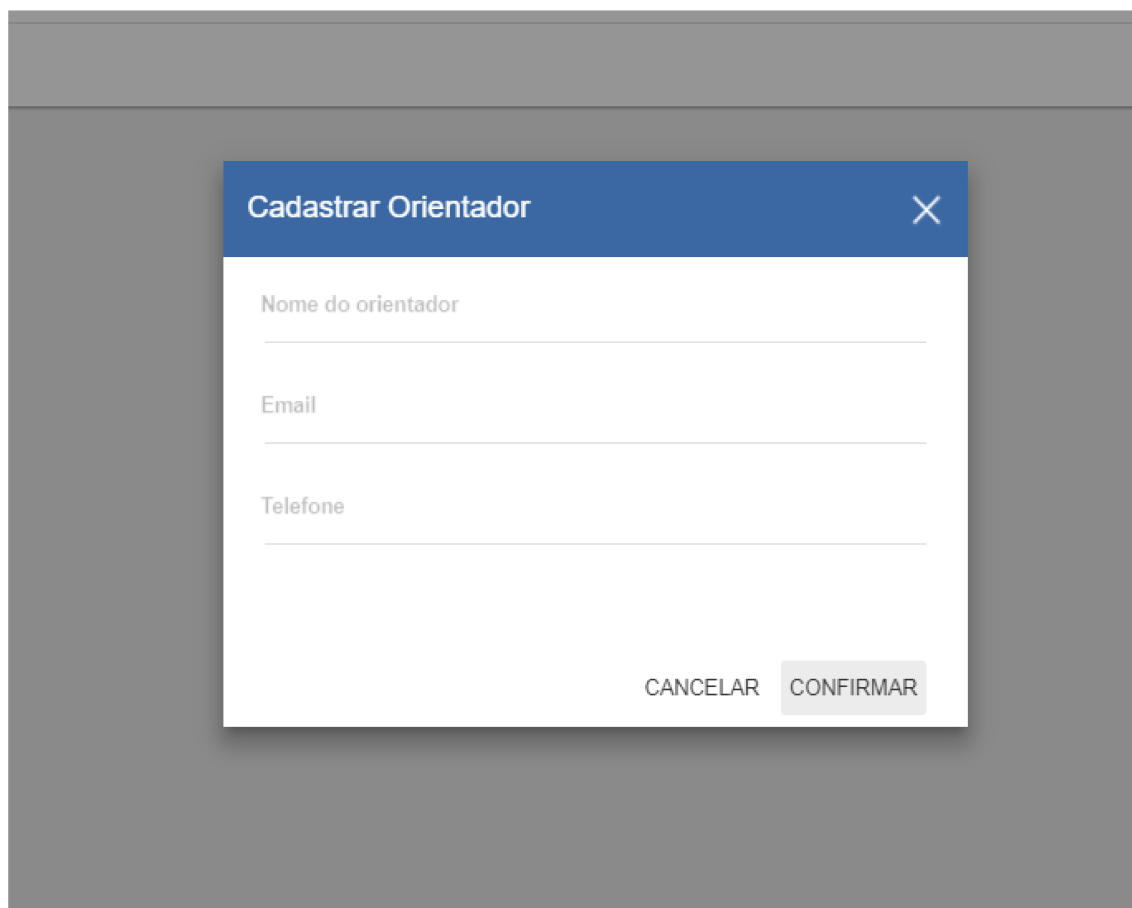
A imagem mostra um modal de cadastro de novo orientador. O modal tem um cabeçalho azul com o título "Cadastrar Orientador" e um ícone de fechar (X) no canto superior direito. O corpo do modal é branco e contém três campos de texto com rótulos "Nome do orientador", "Email" e "Telefone". Abaixo dos campos, há dois botões: "CANCELAR" e "CONFIRMAR". O modal está sobreposto a uma tela cinza.

Figura 11 – Modal para cadastro de novo orientador.

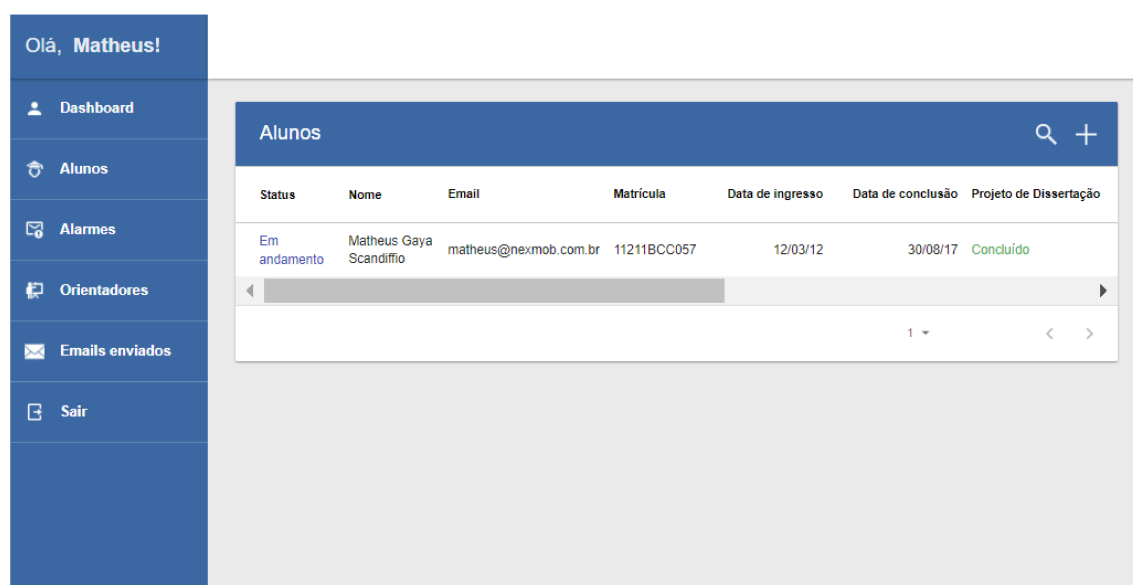
Todas as colunas das tabelas podem ser ordenadas de forma crescente ou decrescente, para facilitar a busca de registro.

Ao clicar na lupa no canto superior direito, o usuário pode filtrar os resultados pesquisando pelo nome, telefone ou email do orientador cadastrado.

5.4 Gestão de alunos

Ao abrir a sessão alunos, o usuário do sistema consegue visualizar todos os alunos cadastrados na plataforma. Ao cadastrar um aluno na plataforma, o usuário possui o controle das datas e pendências que o aluno precisa cumprir para finalizar o curso.

Ao cadastrar um aluno, ele estará apto a receber um alerta caso ele se enquadre na regra definida pelo alerta. Os detalhes do funcionamento dos alertas serão descritos logo abaixo.



Status	Nome	Email	Matrícula	Data de ingresso	Data de conclusão	Projeto de Dissertação
Em andamento	Matheus Gaya Scandiffo	matheus@nexmob.com.br	11211BCC057	12/03/12	30/08/17	Concluído

Figura 12 – Tela 1/2 de alunos do sistema.

Olá, Matheus!

Dashboard

Alunos

Alarmes

Orientadores

Emails enviados

Sair

Alunos

+

Matrícula	Projeto de Dissertação	Solicitação de Defesa	Exame de Língua Estrangeira	Aluno especial	Curso	Orientador	Opções
8/17	Concluído	Pendente - 15/07/17	Concluído	Não	Mestrado	Orientador Exemplo	

1

Figura 13 – Tela 2/2 de alunos do sistema.

Nesta tela, são listadas as seguintes informações para cada aluno cadastrado no sistema:

São listados os dados do orientador, sendo eles:

1. Status do aluno (Se ainda está cursando ou se já concluiu o curso).
2. Nome completo.
3. Email.
4. Matrícula.
5. Data de ingresso no curso.
6. Data prevista para conclusão.
7. Status do projeto de dissertação (Concluído ou data máxima para conclusão).
8. Status da solicitação de defesa (Concluído ou data máxima para conclusão).
9. Status do exame de língua estrangeira (Concluído ou data máxima para conclusão).
10. Se é um aluno especial ou não.
11. Curso.
12. Orientador responsável.
13. Opções (Marcar como curso concluído, editar ou excluir aluno).

Caso o usuário clique para cadastrar um novo aluno, ou editar um já existente, será aberto um modal para que o usuário preencha as informações necessárias do aluno.

The image shows a modal window titled "Novo aluno" with a close button (X) in the top right corner. The modal contains several input fields and checkboxes. The fields are: "Nome", "Email", "Matricula", "Data de ingresso", and "Data de conclusão". Below these is a "Professor orientador" field. There is a "Curso" dropdown menu. Below the dropdown are four checkboxes, each with a label and a "Data Limite" field to its right: "Projeto de Dissertação não entregue" (with "Data Limite para entrega"), "Solicitação de defesa não realizada" (with "Data Limite para Solicitação"), "Exame de língua não concluído", and "Aluno não especial". At the bottom right of the modal are two buttons: "FECHAR" and "CRIAR".

Figura 14 – Modal para criação/edição de um aluno.

Os filtros disponíveis para o campo "Curso" são:

1. Mestrado
2. Doutorado

As pendências a serem cumpridas são alteradas conforme o usuário seleciona o curso do aluno, uma vez que o mestrado e o doutorado possuem diferentes etapas a serem concluídas.

Após preencher todos os dados, basta clicar em Criar (caso esteja criando um novo aluno), ou Salvar (caso esteja editando um aluno já existente) para persistir as informações.

Todas as colunas das tabelas podem ser ordenadas de forma crescente ou decrescente, para facilitar a busca de registro.

Ao clicar na lupa no canto superior direito, o usuário pode filtrar os resultados pesquisando pelo nome do aluno cadastrado.

5.5 Gestão de alarmes

Ao abrir a sessão alarmes, o usuário do sistema consegue visualizar todos os alarmes cadastrados na plataforma. Ao cadastrar um novo alarme, o sistema irá automaticamente rodar uma rotina diariamente para disparar este alarme para os alunos que realmente precisam receber este alarme, ou seja, alunos que ainda não cumpriram com as suas pendências no prazo pré-definido do alarme.

Assim que um alarme for cadastrado, ele já vai ser disparado para os alunos que se enquadram na condição definida. Após isso, todo dia será rodada uma rotina automaticamente para disparar os alarmes necessários.

Figura 15 – Tela de alarmes do sistema.

Os filtros disponíveis para o campo "Campo" são:

1. Data de Conclusão do Curso
2. Projeto de Dissertação
3. Solicitação de Defesa
4. Exame de Língua Estrangeira
5. Projeto de Tese (Doutorado)
6. Solicitação de Defesa de Qualificação (Doutorado)

Os filtros disponíveis para o campo "Curso" são:

1. Mestrado
2. Doutorado

Todas as colunas das tabelas podem ser ordenadas de forma crescente ou decrescente, para facilitar a busca de registro.

Ao clicar na lupa no canto superior direito, o usuário pode filtrar os resultados pesquisando pelo nome do alarme cadastrado.

5.6 Emails enviados

Ao abrir a sessão emails, o usuário do sistema consegue visualizar todos os emails já disparados pela plataforma. Essa sessão serve para caso o usuário deseje conferir se um certo aluno foi notificado, ou apenas acompanhar o disparo dos emails.

Nesta tela, são listadas as seguintes informações para cada email enviado:

1. Assunto do email.
2. Mensagem do email do alarme.
3. Data do envio do email.
4. Para qual curso foi enviado (Apenas mestrado, doutorado ou ambos).
5. Para qual publico foi enviado (Apenas alunos, orientadores ou ambos).
6. Emails que receberam o email.

Emails enviados					
Assunto	Mensagem	Data submissão	Curso	Recipientes	Emails enviados
Teaset	testes	02:36 - 03/07/17	Doutorado	Somente alunos	['matheus@nexmob.com.br']

Figura 17 – Tela de emails enviados.

Todas as colunas das tabelas podem ser ordenadas de forma crescente ou decrescente, para facilitar a busca de registro.

Ao clicar na lupa no canto superior direito, o usuário pode filtrar os resultados pesquisando por assunto, mensagem, curso ou recipientes de cada email enviado.

6 Conclusão

O trabalho desenvolvido teve como principais objetivos a especificação, desenvolvimento e implantação de um sistema de controle de prazos para os alunos de pós-graduação da FACOM - UFU. De maneira resumida, o sistema tem como objetivo automatizar o processo de envio de e-mails aos discentes e seus professores orientadores quando alguma data de vencimento do programa de pós está próximo do vencimento, que hoje é realizado manualmente pelo secretário da pós-graduação. O resultado esperado é um maior controle dos alertas enviados, evitar com que discentes sejam expulsos do programa por perderem alguma data limite, e liberar o secretário da pós para executar outras atividades mais relevantes.

O sistema foi desenvolvido como uma plataforma web, para que o secretário consiga acessar ela através de qualquer dispositivo que possua conexão com a internet. A linguagem de programação utilizada para o servidor foi Javascript, utilizando os frameworks Node.js e Express.JS. O banco escolhido para o desenvolvimento da aplicação foi o MongoDB. O sistema já está em produção, e foi hospedado na Amazon.

Este trabalho também explica como foi feito o desenvolvimento, aprofundando nas tecnologias e metodologias utilizadas. Dessa forma, caso o secretário queira incrementar o sistema com novas funcionalidades, algum outro desenvolvedor consegue criar essas novas funcionalidades de maneira simples após a leitura deste trabalho.

6.1 Próximos passos

Está sendo avaliada a possibilidade de migrar o sistema para os servidores da UFU. Dessa forma, não seria mais necessário ter uma conta na Amazon para hospedar o sistema e o controle ficaria inteiramente sobre o domínio da FACOM.

Ao utilizar o sistema, o secretário poderá ter novas ideias para facilitar o controle de outras partes relacionadas ao programa de pós da FACOM. Como o sistema foi desenvolvido de uma forma totalmente modularizada, a criação de novas funcionalidades e manutenção do sistema se torna simples para um desenvolvedor terceiro. Baseado nestes *feedback's*, o sistema poderá ir evoluindo para se tornar uma plataforma cada vez mais completa.

Referências

- ALBANI, J. S. S. R. *SISTEMA PARA GERENCIAMENTO DE VINÍCOLAS*: uma perspectiva de Arquitetura da Informação da Escola de Brasília. PATO BRANCO: [s.n.], 2011. Citado na página 17.
- AMAZON EC2. *Introdução ao Amazon EC2*. [S.l.], 2017. Disponível em: <<https://aws.amazon.com/pt/ec2/>>. Acesso em: 13 jul. 2017. Citado na página 31.
- FALBO, R. de A. *Notas de aula sobre engenharia de software*. UFES: [s.n.], 2005. Citado na página 12.
- FIALHO, W. de P. P. *Engenharia de Software: fundamentos, métodos e padrões*. 1. ed. São Paulo: LTC Editora, 2000. Citado na página 27.
- FRANCO, D. V. *7 benefícios do Scrum no desenvolvimento dos produtos Digisystem*. [S.l.], 2017. Disponível em: <<http://www.digisystem.com.br/7-beneficios-do-scrum-no-desenvolvimento-dos-produtos-digisystem/>>. Acesso em: 13 jul. 2017. Citado na página 25.
- LAMIM, J. *Afinal, o que é Frontend e o que é Backend?* 2014. Afinal, o que é Frontend e o que é Backend? Disponível em: <<https://www.oficinadanet.com.br/post/13541-afinal-o-que-e-frontend-e-o-que-e-backend->>. Acesso em: 13 mai. 2016. Citado na página 16.
- LOVI, R. *O QUE É UI – USER INTERFACE?* [S.l.], 2017. Disponível em: <<http://www.raffcom.com.br/blog/o-que-e-ui/>>. Acesso em: 14 jul. 2017. Citado na página 22.
- MARTINEZ, M. *UML*. 2016. UML. Disponível em: <<http://www.infoescola.com/engenharia-de-software/uml/>>. Acesso em: 16 jun. 2016. Citado 2 vezes nas páginas 5 e 15.
- MATIOLA, W. *7 benefícios do Scrum no desenvolvimento dos produtos Digisystem*. [S.l.], 2017. Disponível em: <<http://designculture.com.br/o-que-e-ui-design-e-ux-design/>>. Acesso em: 07 jul. 2017. Citado na página 21.
- OOCH G; RUMBAUGH, J. e. J. I. *UML, Guia do Usuário: tradução*. 2. ed. Rio de Janeiro: Campus, 2008. Citado na página 13.
- SHALLIT, J. A very brief history of computer science. *University of Waterloo (2006)*, v. 1, n. 1, 1995. Citado na página 20.
- SILVA, G. *O que é e como funciona a linguagem JavaScript?* [S.l.], 2017. Disponível em: <<https://canaltech.com.br/internet/O-que-e-e-como-funciona-a-linguagem-JavaScript/>>. Acesso em: 14 jul. 2017. Citado na página 17.
- SOMMERVILLE, I. *Engenharia de Software*. 8. ed. Reino Unido: PEARSON EDUCATION, 2007. Citado na página 12.

STEFANINI, M. *SCRUM E SUAS VANTAGENS*. [S.l.], 2017. Disponível em: <<https://stefanini.com/br/2014/01/scrum-vantagens/>>. Acesso em: 13 jul. 2017. Citado na página 25.

VARGAS, T. C. de S. *A história de UML e seus diagramas*. [S.l.], 2016. Disponível em: <https://projetos.inf.ufsc.br/arquivos_projetos/projeto_721/artigo.tcc.pdf>. Acesso em: 16 jun. 2016. Citado na página 13.