



Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Campus Campina Grande
Coordenação do Curso Superior de Tecnologia em Telemática

OrientaIFPB: um sistema web para gestão das atividades de orientação acadêmica

Diego Martins Duarte

Orientador: Anderson Fabiano Batista Ferreira da Costa

Campina Grande, Setembro de 2022

©Diego Martins Duarte



Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Campus Campina Grande
Coordenação do Cursos Superior de Tecnologia em Telemática

OrientaIFPB: um sistema web para gestão das atividades de orientação acadêmica

Diego Martins Duarte

Monografia apresentada à Coordenação do
Curso de Telemática do IFPB - Campus
Campina Grande, como requisito parcial para
conclusão do curso Superior de Tecnologia em
Telemática.

Orientador: Anderson Fabiano Batista Ferreira da Costa

Campina Grande, Setembro de 2022

D812o Duarte, Diego Martins.

OrientalFPB: um sistema *web* para gestão das atividades de orientação acadêmica / Diego Martins Duarte. - Campina Grande, 2022.

46 f. : il.

Trabalho de Conclusão de Curso (Curso de Tecnologia em Telemática) - Instituto Federal da Paraíba, 2022.

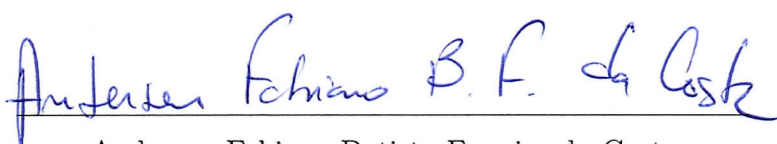
Orientador: Prof. Dr. Anderson Fabiano Batista Ferreira da Costa.

1. Sistema *web* 2. Gestão de orientação 3. *Spring Boot*
I. Título.

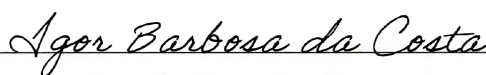
CDU 004.3

OrientaIFPB: um sistema web para gestão das atividades de orientação acadêmica

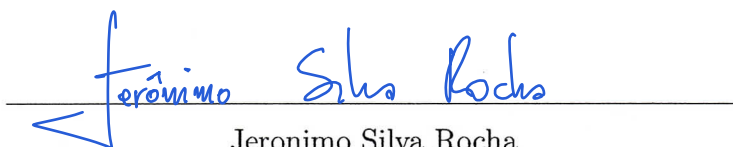
Diego Martins Duarte



Anderson Fabiano Batista Ferreira da Costa
Orientador



Igor Barbosa da Costa
Membro da Banca



Jeronimo Silva Rocha
Membro da Banca

Campina Grande, Paraíba, Brasil

Setembro /2022

Dedico esse trabalho aos meu familiares e professores que me ajudaram ao longo desta caminhada. E ao meu orientador que prestou todo suporte para que eu conseguisse concluir esta difícil tarefa .

"Se alguém te oferecer uma oportunidade incrível, mas você não tem certeza de que consegue fazer, diga sim - e depois aprenda como fazer."

Richard Branson

Agradecimentos

Em primeiro lugar, Deus, que fez com que meus objetivos fossem alcançados, durante todos esses anos de estudos.

Aos meus familiares por todo o apoio e pela ajuda, que muito contribuíram para a realização deste trabalho.

Ao professor Anderson, por ter sido meu orientador e ter desempenhado tal função com dedicação e amizade. E aos professores do Curso Superior de Tecnologia em Telemática agradeço com profunda admiração pelo vosso profissionalismo.

Resumo

A *internet* faz parte do cotidiano da sociedade, sendo uma ferramenta de trabalho poderosa para todo tipo de serviço. Neste cenário, as aplicações *Web* são poderosas ferramentas que visam facilitar o dia-dia com soluções completas e robustas. O IFPB como a maioria das instituições de ensino superior, tem como requisito para obtenção do diploma o desenvolvimento de um TCC, porém existe a dificuldade de encontrar um tema e orientador disponível para o desenvolvimento do trabalho de conclusão de curso, dificuldade essa enfrentada pela maioria dos alunos em fase de conclusão de curso. Com o objetivo de resolver esse problema foi desenvolvido o sistema *Web OrientaIFPB*, uma aplicação que permite que o coordenador do curso possa acompanhar o desenvolvimento das atividades relacionadas ao trabalho de conclusão de curso, possibilitando também o registro de reuniões de orientação realizados, entre o aluno e orientador, e posteriormente a possibilidade de agendar a defesa da monografia. Foi realizada uma pesquisa bibliográfica para identificar a melhor abordagem possível para o desenvolvimento da aplicação, ficando decidido por desenvolver uma aplicação *front-end*, utilizando o *framework* Angular responsável pela interface gráfica da aplicação e uma API em Java com o *framework* Spring Boot sendo o *back-end* da aplicação. Pode-se concluir que o sistema atende todos os requisitos necessários já que facilita a gestão de dados referentes à orientação, auxiliando o aluno na escolha do tema e orientador e o professor na gestão das atividades de orientação.

Palavras-chave: *Angular*; Aplicações *Web*; Gestão de Orientação; *Spring Boot*.

Abstract

With Internet being part of the daily life of society, a tool for powerful work for every kind of service. In this scenario the Web applications are powerful tools that aim to facilitate the day-to-day, with complete and robust solutions. IFPB, like most higher education institutions, require the development of a term paper by graduating students, but it is difficult to find a theme and advisor, this difficulty faced by most students in the course conclusion phase. As In order to solve this problem, the OrientaIFPB Web system was developed. This application will allow the course coordinator to follow the development activities related to the term papers, also enabling the record of meetings between the student and advisor and the presentations scheduling. A bibliographic research was carried out to identify the best possible approach. for the development of the application, after research it was decided to develop a front-end application developed using the Angular framework responsible for the interface application graphics and put an end to an API in Java with the Spring Boot framework being the application backend. Finally, we can conclude that the system meets all the necessary requirements since facilitates the management of data related to guidance, helping the student to choose the topic and advisor and the teacher in the management of guidance activities.

Keywords: Framework; Web Applications; Java; Spring Boot; API; Frontend; Job of course completion; Backend.

Sumário

Lista de Abreviaturas	xi
Lista de Figuras	xii
1 Introdução	1
1.1 Objetivos	2
1.1.1 Objetivo Geral	2
1.1.2 Objetivos Específicos	2
1.2 Metodologia	3
2 Fundamentação Teórica	4
2.1 Tecnologias de front-end	5
2.1.1 HTML5	5
2.1.2 CSS3	5
2.1.3 JavaScript	6
2.1.4 Angular	7
2.1.5 Two-way data binding	8
2.2 Tecnologias de back-end	9
2.2.1 Java	9
2.2.2 Spring Boot	9
3 Desenvolvimento do Sistema OrientaIFPB	11
3.1 Levantamento de requisitos	11
3.1.1 Envolvidos no sistema	11
3.1.2 Requisitos funcionais	12
3.1.3 Requisitos não funcionais	13
3.2 Desenvolvimento back-end	13
3.2.1 Camada de Controle	14
3.2.2 Camada de Manipulação de Dados	15
3.2.3 Pacote <i>Config</i>	15
3.3 Desenvolvimento front end	18
3.3.1 Estrutura do projeto	19

3.3.2	Camada de serviços	20
4	Resultados e Discussões	21
4.1	Cenário	21
4.2	Cadastro no sistema	21
4.3	Página <i>Home</i>	22
4.4	Cadastro de Professores	23
4.5	Cadastro de Alunos	24
4.6	Cadastro Trabalho de Conclusão de Curso	25
4.6.1	Cadastro de Reuniões	26
4.6.2	Agendamento de defesa	27
5	Conclusão	29
5.1	Trabalhos Futuros	29
A	Instalação OrientaIFPB APP	31
A.1	<i>Clone dos repositórios</i>	31
A.2	<i>Front-end</i>	31
A.3	<i>Back-end</i>	32
	Referências Bibliográficas	33

Lista de Abreviaturas

API	<i>Application Programming Interface</i>
DOM	<i>Document Object Model</i>
JVM	<i>Java Virtual Machine</i>
JDK	<i>Kit de desenvolvimento Java</i>
DAO	<i>Data Access Object</i>
MVC	<i>Model View Controller</i>
CORS	<i>Cross-origin Resource Sharing</i>
JSON	<i>JavaScript Object Notation</i>
JWT	<i>JSON Web Token</i>
SPA	<i>Single Page Applications</i>
IEEE	<i>Instituto de Engenheiros Eletricistas e Eletrônicos</i>
CSRF	<i>Cross-site Request Forgery</i>
SSO	<i>Single Sign On</i>
MVC	<i>Model-View-Controller</i>

Lista de Figuras

2.1	Estrutura básica documento html	5
3.1	Arquitetura MVC	14
3.2	Diagrama de sequência aluno <i>controller</i>	15
3.3	Classe WebConfig	16
3.4	Fluxograma de autenticação com <i>oauth2</i>	17
3.5	Token JWT	18
3.6	Token JWT decodificado	18
3.7	Estrutura do projeto	19
3.8	Diagrama de sequência, para o serviço de alunos	20
4.1	Tela de <i>login</i>	21
4.2	Tela de cadastro	22
4.3	Página <i>Home</i>	22
4.4	Formulário de cadastro de professores	23
4.5	Lista de professores	23
4.6	Lista de alunos	24
4.7	Formulário de cadastro de alunos	24
4.8	Opções de atualizar/deletar dados	25
4.9	Tela de cadastro TCC	25
4.10	Lista de orientações	26
4.11	Formulário de cadastro de reuniões	26
4.12	Conteúdo de reuniões realizadas	27
4.13	Realizar agendamento	27
4.14	Agendando defesa de monografia	28

Capítulo 1

Introdução

Com a evolução da Tecnologia da Informação, é possível realizar tarefas diárias de forma cada vez mais simples e rápida. Nesse contexto, qualquer organização, seja pública ou privada, para aprimorar seus processos e melhorar a gestão das informações, necessitam de ferramentas automatizadas por meio de Sistemas de Informação (SI) [Silva 2017].

No intuito de facilitar a realização dessas tarefas, as aplicações *Web* surgem como ferramentas que auxiliam na execução de diversos tipos de atividades, desde a realização de compras online, reservas em hotéis a pagamentos de contas.

Aplicações Web são soluções executadas diretamente no navegador, sem a necessidade de instalação no computador do usuário, como um programa de computador normal, exigindo-se, portanto, conexão com a Internet [Roveda 2020].

As instituições de ensino, a exemplo do IFPB, possuem diversas demandas para desenvolvimentos de SI específicos para melhorar a gestão de suas atividades de rotina. Nesse sentido, a gestão de atividades, como o controle das orientações acadêmicas, é uma das informações que precisam ser administradas pelas as coordenações de curso e unidades acadêmicas e esses dados nem sempre são de fácil ou de imediato acesso.

O IFPB utiliza o Sistema Unificado de Administração Pública (SUAP) para realizar diversos serviços de âmbito acadêmico e administrativo, mas esse sistema carece de relatórios ou opções que auxiliem os gestores acadêmicos em algumas tarefas. Por exemplo quando uma coordenação necessita, alocar carga horária de ensino ou de orientação acadêmica, o SUAP não dispõe de um módulo específico para auxiliar o gestor nessa tarefa. Além disso, trabalhos de orientações acadêmica, a exemplo de um trabalho de conclusão de curso, apresentam alguns trâmites como formação de banca e geração de ata, que o sistema também não contempla.

Este trabalho, portanto, visa o desenvolvimento de um sistema Web, nomeado de OrientaIFPB, que tem o intuito de auxiliar o coordenador de curso ou de unidade acadêmica na gerência de processos que envolvem orientações de trabalhos acadêmicos dos docentes lotados na unidade/cursos, de modo a centralizar informações, e conseqüentemente, facilitar a gerência de informações de todos os envolvidos (alunos, docentes e coordenadores).

A ideia é que esse sistema matenha informações sobre atividades de orientação acadêmica dos docentes, orientações de Trabalho de Conclusão de Curso (TCC), Pesquisa e Extensão, por meio de um sistema Web com uma interface amigável. Além disso, o sistema deve manter informações, no caso específico de TCCs, sobre bancas avaliadoras, datas de defesas e geração de atas de defesa.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver a aplicação Web OrientaIFPB, para auxílio à coordenação de curso ou unidade acadêmicas na gestão de atividades de orientações acadêmica dos docentes do IFPB.

1.1.2 Objetivos Específicos

Para alcançar o objetivo geral neste trabalho de conclusão de curso, os seguintes objetivos específicos serão considerados:

- possibilitar o cadastro e acompanhamento, por parte do coordenador de curso/unidade acadêmica, de dados relacionados a atividades de orientação acadêmica dos docentes.
- disponibilizar, a diferentes perfis (gestor, professor e aluno), o acesso aos dados de orientação acadêmica.
- disponibilizar interface para que o coordenador possa gerenciar a geração de atas de TCC, alocação e agendamento de bancas.

1.2 Metodologia

Foi realizada uma pesquisa bibliográfica para identificar qual a melhor abordagem possível para o desenvolvimento da interface Web da aplicação.

Um dos pontos essenciais que o desenvolvedor tem que se preocupar é a interface gráfica ou *front-end* da aplicação, em que se desenvolve a aplicação com a qual o usuário interage diretamente.

Após essa busca para a melhor abordagem, foi decidida a utilização do framework Angular, que é uma plataforma para construção de interface de aplicações utilizando HTML, CSS e JavaScript [Afonso 2018].

Alguns elementos básicos que devem fazer parte dessa construção são os componentes, templates, diretivas de roteamento, módulos, serviços e injeção de dependências, além de ferramentas de infraestrutura que automatizam tarefas, como execução de testes unitários das aplicações [Afonso 2018].

Para a persistência de dados, na fase de testes foi utilizado o H2, um banco de dados relacional executado diretamente no navegador. Posteriormente, o Mysql foi utilizado como banco de dados final da aplicação.

Para o *back-end* foi desenvolvida uma API *rest* em java 8, com o objetivo de simplificar e agilizar o desenvolvimento da funcionalidade de receber e processar todas as requisições vindas do *front-end*. Nesse caso, foi utilizado o *framework Spring Boot*.

Capítulo 2

Fundamentação Teórica

Voltada para a construção de *sites*, aplicativos, bancos de dados e qualquer outra ferramenta, para Roveda [Roveda 2020] a área de desenvolvimento *Web* é que constrói a Internet como conhecemos. No desenvolvimento *Web* há especialidades como o *front-end* e o *back-end*.

O desenvolvimento *front-end* envolve a parte visual de um *site* ou aplicação, existindo tecnologias voltadas para o desenvolvimento da interface gráfica de tudo que é apresentado pelo navegador [Roveda 2020].

As principais tecnologias *front-end* são HTML, CSS e JavaScript, existindo uma grande variedade de bibliotecas muito populares que auxiliam no desenvolvimento. Essas tecnologias foram utilizadas no desenvolvimento da aplicação orientada a objetos, junto do *framework Angular*.

Segundo [Roveda 2020], o desenvolvimento *back-end*, ao contrário do *front-end*, é a parte interna da aplicação ou *site*, sendo responsável pelo funcionamento e armazenamento de informações.

Quando ocorre uma interação do usuário através de sua interface gráfica todas as informações ali inseridas são processadas por um servidor ou armazenado em um banco de dados, é o *back-end* que processa essas solicitações do usuário [Roveda 2020].

Existe várias linguagens de programação que podem ser utilizadas no desenvolvimento *back-end*, dentre elas Python, Java, PHP e Ruby. No projeto em questão foi utilizado Java 8 e o *framework spring boot*.

Neste Capítulo são abordados conceitos importantes relacionados ao trabalho sendo organizado da seguinte forma. Na seção 2.1, são abordadas, de forma mais abrangente, as tecnologias utilizadas no desenvolvimento da interface gráfica da aplicação. Na Seção 2.2, são apresentadas as tecnologias referentes ao *back-end* da aplicação.

2.1 Tecnologias de front-end

2.1.1 HTML5

HTML (*HyperText Markup Language*) é uma linguagem de marcação de Hipertexto utilizada no desenvolvimento de páginas *Web* [W3schools 1999].

Documentos HTML são interpretados pelos navegadores e apresentam um conjunto de *tags* e atributos [Lima 2021]. *Tags* são os marcadores ou os comandos de formatação da linguagem, uma *tag* indica para o navegador onde começa e termina determinado elemento. Os atributos descrevem as características do elemento.

A estrutura básica de um documento HTML apresenta marcações, como as mostradas na Figura 2.1.

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="utf-8" />
    <meta name="description" content="a descrição do seu site em no máximo 90 caracteres">
    <meta name="keywords" content="escreva palavras-chaves curtas, máximo 150 caracteres">
    <title>Título do Documento</title>
  </head>
  <body>
    <!-- Aqui fica a página que será visível para todos, onde pode-se inserir
    textos, imagens, links para outras páginas, etc, geralmente usa-se: -->

    <div>
      Tag para criar-se uma 'caixa', um bloco, mais utilizada com "Cascading Style Sheets
      (Folhas de Estilo em Cascata)
    </div>

    <span>Tag para modificação de uma parte do texto da página</span>

    <a href="http://www.wikipedia.org">Wikipedia, A Enciclopédia Livre</a>
  </body>
</html>
```

Figura 2.1: Estrutura básica documento html [W3schools 1999].

A quinta versão da HTML apresenta funcionalidades novas de semântica e acessibilidade [Silva 2017]. Possibilitando o uso de recursos que só eram possíveis com a aplicação de outras tecnologias, com suporte para multimídias recentes.

2.1.2 CSS3

Cascading Style Sheet ou folha de estilo em cascata, é utilizada para estilizar elementos escritos em uma linguagem de marcação como HTML [W3schools 1999]. CSS funciona como uma camada de estilização de páginas HTML, é uma maneira de dar estilo ao código.

Criada em 1996 pelo W3C, para suprir a necessidade que os desenvolvedores tinham de criar padrões de formatação de páginas HTML, o CSS veio com uma forma de melhorar a estética dos sites.

CSS possibilita a edição do *layout* como definição de cores, espaçamentos, tamanhos e tipos de fontes, e ajuda na redução de repetição na estrutura do código.

Bootstrap

O *Bootstrap* é *Framework* de código aberto que fornece estruturas de CSS para criação de *sites* e aplicações responsivas [Lima 2021], para *sites* de desktop e páginas de dispositivos móveis.

Foi desenvolvido originalmente para o *Twitter* pelos desenvolvedores Mark Otto e Jacob Thornton e se tornou uma das estruturas de *front-end* mais populares [Lima 2021].

Ainda que o *Bootstrap* poupe o tempo que os desenvolvedores gastariam gerenciando seus templates, seu principal objetivo é criar *sites* responsivos. Permitindo que a interface do usuário seja otimizada para qualquer tipo de tela, desde dispositivos móveis até telas maiores.

Assim não é preciso desenvolver muitas versões de um mesmo *site* para que se adeque a todos os tamanhos de telas disponíveis [Lima 2021].

2.1.3 JavaScript

Criada em meados da década de 90 pela *Netscape Communications*, o *JavaScript* é uma linguagem de programação interpretada, tendo como objetivo atender a demanda crescente por sites mais interativos e dinâmicos, já que até então as páginas HTML continham nenhuma ou pouca interatividade além dos *hyperlinks* [Melo 2021].

Essa linguagem controla elementos da página localmente sem ter a necessidade de receber dados ou uma resposta do servidor.

É uma linguagem multiparadigma, de tipagem dinâmica, sintaxe com recursos avançados como orientação a objetos e APIs, podendo trabalhar com textos, matrizes, datas e expressões regulares [Melo 2021].

O uso primário do *JavaScript* é escrever funções e *scripts* que são embarcados em uma página HTML, atualizando informações e interagindo dinamicamente com o conteúdo da página [Melo 2021].

Um código *JavaScript* envia e recebe dados do servidor de forma síncrona e assíncrona, processando, validando e exibindo as informações em tempo real [Silva 2017]. Apesar de algumas semelhanças com o Java, inclusive seu nome o *JavaScript* não é uma tecnologia derivada do Java [Melo 2021].

JavaScript Object Notation - JSON

Segundo [Puluceno 2012], JSON é um mecanismo de codificação para troca de dados de forma simples e rápida. Apresenta uma sintaxe de alto nível nativa de JavaScript, e fácil de ser entendida, o que facilita o trabalho dos desenvolvedores com um modelo para intercâmbio de dados de linguagem textual leve, fácil de ser analisado.

A aplicação OrientaIFPB utiliza arquivos no formato JSON para troca de informações entre o *front-end* e o *back-end* da aplicação. Para [Santos 2018] as camadas de serviços

expõem informações via JSON, fazendo requisições via protocolo HTTP.

Com a capacidade de representar estruturas de dados de uma forma geral como: registros, listas e árvores, os arquivos em formato JSON de tem sido adotado justamente ao fato de ser produtivo em aplicações distribuídas e no desenvolvimento de serviços [Puluceno 2012].

2.1.4 Angular

É uma plataforma voltada para construção da interface de aplicações, usando HTML, CSS e JavaScript, tendo foco em aplicações *Web* de código-fonte aberto e *front-end* baseadas em *TypeScript* [Totvs 2020].

TypeScript é uma linguagem que permite escrever o código JavaScript fazendo programações orientadas a objetos sem perder suas vantagens [Afonso 2018].

Ao compilar o código em *TypeScript*, é gerado um código JavaScript que é executado no navegador [Afonso 2018].

Para [Totvs 2020], entre os elementos básicos de Angular podemos destacar:

- Templates;
- Componentes;
- Roteamento;
- Diretivas;
- Módulos;
- Serviços;
- Injeção de dependências;
- Ferramentas de infraestrutura;

Segundo [Totvs 2020], com Angular o desenvolvimento das aplicações *front-end* é otimizado, no HTML consegue-se criar páginas estáticas, porém em uma aplicação *Web* é necessário mais do que isso.

O Angular adapta e estende o HTML tradicional otimizando sua experiência, com conteúdo dinâmico e ligação direta de dados, conhecido como *two-way-data-binding* que possibilita a sincronização automática de modelos e visualizações [Totvs 2020].

Provendo recursos para o desenvolvimento de *Single Page Applications*, ou seja, uma única página *Web* tendo o objetivo de fornecer a experiência ao usuário parecida com a de uma aplicativo *desktop*, com o código carregado na página única de forma dinâmica [Totvs 2020].

A produtividade aumenta, o desenvolvedor tem a opção de dividir o código em partes, com o uso de componentes, módulos e outras funcionalidades. Com essa forma de estrutura, há um ambiente para o desenvolvimento mais organizado.

Componentes

Um conceito fundamental quando se fala de *framework front-end* é o conceito de componentes, por permitirem a criação de códigos que podem ser reutilizados e testados sem maiores riscos [Totvs 2020].

O *AppComponent* é o componente principal de uma aplicação Angular, a partir dele há uma hierarquia de outros componentes ao modelo de objeto de documento (DOM) de página [Totvs 2020].

Módulos

Um aplicativo é definido por uma junção de módulos quando se utiliza Angular. Esses módulos são como blocos que podem ser utilizados na construção interfaces no Angular. Essa ação pode ser traduzida como agrupar, exportar e esconder componentes, diretivas, *pipes* e serviços relacionados.

Esses módulos servem para formar uma aplicação e são chamados de *NgModules* [Totvs 2020]. Cada aplicação é composta por pelo menos uma categoria dessa classificação, que é o módulo principal da aplicação. As principais categorias são:

- Imports: arranjos com outros módulos, necessários para utilizar componentes declarados dentro da aplicação;
- Declarations: arranjos de componentes, diretivas e *pipes*, que fazem parte do módulo;
- Exports: conjunto de componentes e *pipes*, disponíveis para outros módulos;
- Providers: declaração dos serviços, em que, se um módulo for o principal, eles estarão disponíveis para toda a aplicação.

2.1.5 Two-way data binding

Pode-se definir uma das principais características do *framework*, como uma associação bidirecional de dados, onde a informação entra através da visualização ou *template* passando para uma propriedade de classe do componente. Com isso o dado já é exibido automaticamente pelo elemento do DOM no *template* do componente.

Sua principal proposta é automatizar a circulação dos dados, facilitando a vida do desenvolvedor ao não exigir a criação de *handlers* para automatizar a visualização [Totvs 2020].

Desta forma quando o valor de um componente mudar, o próprio *framework* realiza a atualização do valor na página, a ligação bidirecional combina a entrada e saída em um único processo [Totvs 2020].

2.2 Tecnologias de back-end

2.2.1 Java

Java é linguagem de programação e ambiente computacional criado pela *Sun Microsystems* na década de 90.

Se tornou popular graças à possibilidade de escrever o código apenas uma vez e rodá-lo em diferentes dispositivos, passando a ser implementada em praticamente qualquer aplicação, desde sites a datacenters, celulares e videogames [Melo 2020].

O código Java é baseado em classes e orientado a objetos, com seu foco voltado para segurança, portabilidade e alto desempenho.

Tendo como característica uma sintaxe similar a C/C++, possuindo uma extensa biblioteca de rotinas e APIs para trabalhar com recursos de rede, e também um poderoso gerenciamento de memória de forma automática.

O código Java é compilado em *bytecode*, que então é interpretado e executado pela Máquina Virtual Java (JVM).

Dessa forma o sistema, ou aplicação em Java se torna portátil, podendo ser rodado em praticamente qualquer ambiente ou dispositivo no qual a JVM esteja instalado.

Composta por um grande número de tecnologias, a plataforma se divide em ambiente de desenvolvimento e de execução do software.

Existem três plataformas que englobam as ferramentas necessárias para criação e execução de software e sistemas, a máquina virtual, o kit de desenvolvimento JDK, o compilador e outras ferramentas utilitárias [Melo 2020].

Sendo responsável por compilar, executar e gerenciar os programas, a JVM é onde os usuários finais costumam interagir com os programas desenvolvidos em java.

2.2.2 Spring Boot

O *Spring Boot* é um *framework* Java que tem como objetivo facilitar a etapa de criação e configuração de uma aplicação, o que anteriormente era um processo demorado, pois exigia a manipulação de vários arquivos e dependências [Rossalli 2021].

Com o *Spring Boot* conseguimos abstrair etapas de configuração, como exemplo:

- Servidores;
- Gerenciamento de dependências;
- Configurações de bibliotecas.

Segundo Afonso [Afonso 2017] o *Spring Boot* consegue isso por favorecer a convenção sobre a configuração, basta informar quais módulos utilizar (Web, template, persistência) que ele vai reconhecer.

No desenvolvimento da aplicação *OrientalFPB* foram utilizados os módulos do *Spring Boot*: *Spring Data JPA* e *Spring Security*.

Spring Data JPA

Esse módulo facilita, a criação dos repositórios do projeto, liberando o desenvolvedor de implementar as interfaces referentes aos repositórios (DAOs) pré-implementando algumas funcionalidades como ordenação de consultas e de paginação [Afonso 2017].

O JPA faz parte de um projeto maior o Spring Data, que tem por objetivo facilitar o trabalho com persistência de dados de uma forma geral [Afonso 2017].

Spring Security

Segundo [Candioli 2020], o *Spring Security* é uma biblioteca para proteção, por meio de autenticação, autorização e armazenamento de senhas. Para autenticação pode trabalhar com vários protocolos, para armazenar senhas em um banco de dados tendo como opção vários encoders. Podendo ser utilizado em vários projetos desde Java EE, *Spring Webflux* e Kotlin, protegendo também de ataques mais comuns como *Cross-site Request Forgery* (CSRF).

Como porta de entrada da aplicação temos a autenticação e essa porta precisa estar protegida, tanto para uma aplicação interna como externa [Candioli 2020].

Para isso são disponibilizados alguns mecanismos, como:

- *Username and Password*: quando se utiliza usuário e senha para logar. Para esse mecanismos podemos utilizar três formas: *Form login*, *Basic Authentication* e *Digest Authentication*;
- *OAuth2*: quando se utiliza *OpenID connect*, é possível se autenticar em uma aplicação usando uma conta e servidor externo, como Github;
- *SAML2*: é um padrão que surgiu antes do OAuth e que serve para compartilhamento de informações de login, podendo também atender o padrão *Single Sign On* (SSO);
- *Remember me*: mecanismo utilizado para salvar a sessão do usuário logado na aplicação;

Como mecanismo de autenticação, foi utilizado o OAuth2 em conjunto de *tokens* JWT (JSON Web Token), transmitindo de forma compacta e segura objetos JSON entre diferentes aplicações, assinado digitalmente com uma chave secreta HMAC ou pares de chaves pública RSA ou ECDSA. Consistindo em três partes separadas por pontos, sendo *header*, *payload* e *signature*. É uma tecnologia bastante utilizada em cenários reais, o seu conhecimento por parte dos desenvolvedores é de suma importância.

Capítulo 3

Desenvolvimento do Sistema OrientaIFPB

Este capítulo tem como objetivo demonstrar o processo de desenvolvimento da aplicação OrientaIFPB.

Na Seção 3.1 é apresentado o levantamento de requisitos da aplicação, será demonstrado os envolvidos no sistema, os requisitos funcionais e não funcionais.

Na Seção 3.2 é mostrado o processo para o desenvolvimento do *back-end* da aplicação, como mencionado anteriormente foi desenvolvida a API utilizando a linguagem de programação Java em sua versão 8, e o *framework Spring Boot*.

Já na Seção 3.3 é mostrada a implementação da interface com o cliente da aplicação, que foi desenvolvida utilizando o *framework* para desenvolvimento *front-end*, Angular.

3.1 Levantamento de requisitos

3.1.1 Envolvidos no sistema

Como envolvidos no sistema temos dois grupos que podemos destacar como usuários e stakeholders, as pessoas interessadas na gestão do projeto.

Sendo usuários do sistema todos aqueles que utilizam a aplicação com o objetivo de adquirir informações e orientações importantes referente ao desenvolvimento de projetos de orientação.

Como usuários do sistema temos:

- Coordenadores - realizam acompanhamento e o cadastro de dados relacionados a orientações acadêmicas dos docentes;
- Docentes/Orientador - realizam acompanhamento do discente/aluno registrando reuniões de orientação no sistema, podendo também realizar o agendamento de defesa do TCC;

- Alunos - conseguem realizar o acompanhamento de todas as reuniões que já participou;

Já como stakeholders da aplicação OrientaIFPB temos:

- Orientador - também usuário do sistema;
- Orientando - também usuário do sistema;
- Desenvolvedor - com a responsabilidade de criação e implementação do sistema garantindo sua segurança e testes;

A aplicação está sendo desenvolvida utilizando um perfil de administrador (coordenador do curso), em que o usuário tem permissão para cadastrar alunos, professores e orientações, além de conseguir excluir e editar o andamento da orientação, posteriormente pode-se implementar níveis de acesso, de acordo do perfil de usuário do sistema, para mais controle de uso do sistema.

O professor poderá inserir/salvar os conteúdos de orientação relacionados ao seu perfil, já o aluno será o que terá menor nível de acesso, podendo apenas visualizar os conteúdos das reuniões com seu respectivo orientador.

3.1.2 Requisitos funcionais

Os requisitos funcionais tem o objetivo de mostrar as funcionalidades do software, descrever as funções que deve executar, especificando de forma precisa as funcionalidades do sistema, como o sistema lida com entradas específicas, como será seu comportamento em algumas situações.

Como requisitos funcionais do sistema OrientaIFPB deve-se destacar:

- O sistema deve permitir o cadastro de professores, alunos e orientações;
- O sistema deve listar alunos, professores e orientações já cadastradas no sistema;
- O sistema tem como função por meio de uma página home, mostrar orientações em andamento, concluídas ou trancadas;
- No sistema, com orientação em andamento, deve ter a possibilidade de registro das reuniões de orientações entre alunos e professores, podendo registrar o conteúdo da reunião;
- O sistema deve listar posteriormente esses registros das reuniões para funcionar como consulta, tanto do orientador quanto para o aluno que está sendo orientado;
- O sistema deve disponibilizar a função de alocação e agendamento de bancas, além de possibilitar a geração de atas de TCC.

3.1.3 Requisitos não funcionais

São todos aqueles que não estão diretamente ligados a funcionalidades do sistema. Podendo também ser chamados de atributos de qualidade, tem um papel de suma importância durante o desenvolvimento do sistema, e podem ser usados como critérios de seleção na escolha de alternativas do projeto, estilo de arquitetura e forma de implantação.

Ao desenvolver um novo sistema de software, apresenta-se um conjunto de atributos de qualidade ou requisitos não funcionais que o sistema deve suportar. Podem ser citados como exemplos desses requisitos o desempenho do sistema, portabilidade, manutenibilidade e escalabilidade, dentre os quais se destacam:

- Usabilidade: o sistema deve possuir uma interface amigável e simples, permitindo que o usuário tenha uma experiência de uso intuitiva, garantindo uma maior aceitação dos usuários do sistema;
- Segurança: O acesso dos usuários ao sistema, deve ser feito por meio de um usuário e senha cadastrados, além de *token* de acesso JWT. O sistema garante que somente usuários cadastrados, tenham acesso a conteúdo e informações internas do sistema;
- Manutenibilidade: com o intuito de facilitar a manutenção e atualização do sistema, a aplicação foi desenvolvida seguindo toda uma padronização de *interface* e de códigos utilizados no desenvolvimento do *software*;
- Desempenho: devendo possuir um desempenho eficiente o sistema deve assegurar que todos os usuários tenham uma mesma experiência de uso, de modo que independente de taxa de transmissão de conexão ou do equipamento físico, não permita que essa experiência venha a ser afetada reduzindo a sua produtividade ao ponto de não atender às suas necessidades;

3.2 Desenvolvimento back-end

Sendo desenvolvida utilizando a arquitetura *Model-View-Controller* (MVC) criada em 1979 pelo cientista da computação norueguês Trygve Reenskaug, que na época trabalhava na Xerox PARC [Silva 2020].

A proposta do cientista era criar um padrão de arquitetura que separasse o projeto em camadas, reduzindo assim a dependência entre elas, apesar de existir a bastante tempo, foi com o desenvolvimento *Web* que a metodologia ganhou vários adeptos.

Segundo [Silva 2020], a arquitetura MVC é um padrão de arquitetura de software que divide a aplicação em três camadas.

- Camada de manipulação de dados;
- Camada de visualização;

- Camada de controle;

Quando o usuário acessa um *site*, a camada de controle (*Controller*), se comunica com a camada de visualização (*View*) e com a camada de manipulação de dados (*Model*) para gerar a requisição.

A camada de manipulação de dados não se comunica diretamente com a de visualização, cabendo assim a função de renderização à camada de controle. A camada de manipulação avisa quando as solicitações são atendidas para que a de visualização enfim mostre os dados ao usuário. Na Figura 3.1 temos de forma ilustrada o funcionamento da arquitetura MVC.

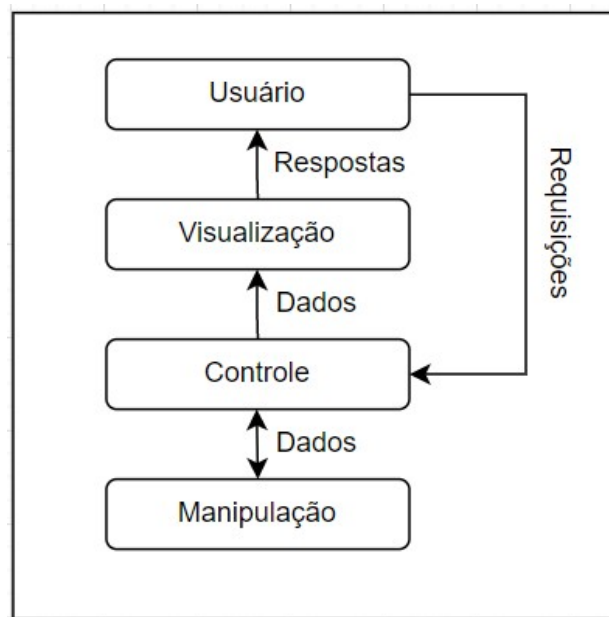


Figura 3.1: *Arquitetura MVC*(Imagem: [Silva 2020]).

Pode-se destacar algumas vantagens na utilização dessa arquitetura como, agilidade na atualização da interface da aplicação, facilidade na manutenção do código, ajuda na implementação de camadas de segurança.

3.2.1 Camada de Controle

Conforme mencionado, a camada de controle é responsável por receber as requisições vindas do *front-end* da aplicação, processar essa requisição realizar o acesso à camada de manipulação de dados e devolver a informação para a camada de visualização que por sua vez mostra os dados ao cliente.

Podemos destacar os *Controllers* *AlunoController*, *ProfessorController* e *TccController* responsáveis por criar, atualizar, listar e buscar informações no banco de dados da aplicação, respectivamente para alunos, professores e para o trabalho de conclusão de curso.

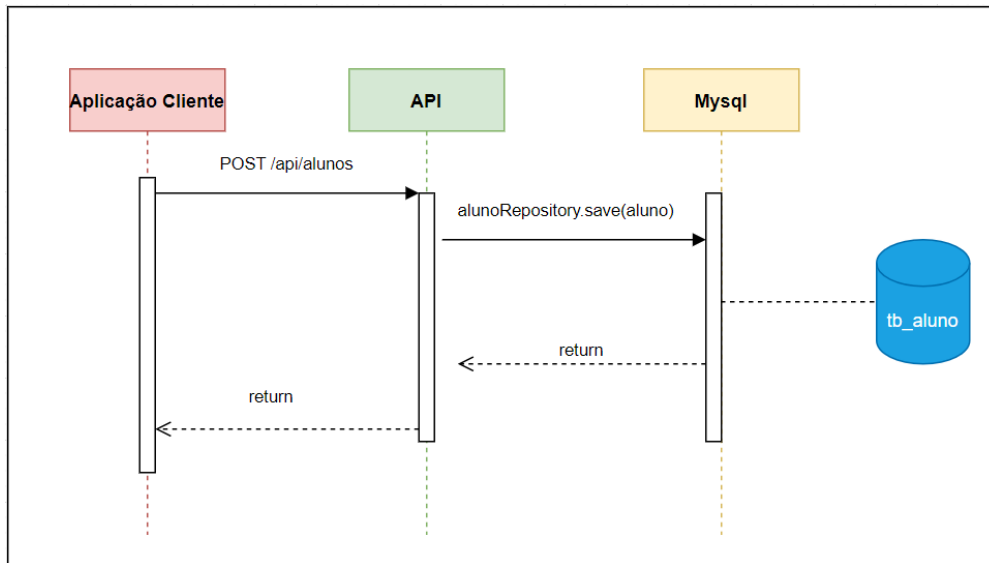


Figura 3.2: Diagrama de sequência aluno controller (Imagem: Autoria própria).

Na Figura 3.2 é apresentado o digrama de sequência para a classe alunoController, mostrando o fluxo para salvar as informações sobre o aluno na base de dados do sistema, a aplicação cliente faz uma requisição *POST* para o alunoController, chamando o método *save*, que por sua vez tem os dados salvos no banco de dados na tabela de aluno (*tb_aluno*).

Após salvos os dados, retorna-se uma mensagem de sucesso ou de erro para a camada de controle e posteriormente para o *front-end* da aplicação (camada de visualização).

3.2.2 Camada de Manipulação de Dados

Nessa camada são manipulados os dados da aplicação, como o acesso a informações dos alunos, professores e das orientações em andamento, é aqui que se destaca o uso do *Spring Data JPA*, que facilita a implementação dos repositórios do sistema, já que apresenta suporte aprimorado para camadas de acesso a dados.

Na camada de manipulação da aplicação há dois pacotes: o *Entity* referente às entidades da camada de dados da aplicação, e o pacote *Repository* responsável pelos repositórios das entidades da camada de manipulação de dados.

3.2.3 Pacote *Config*

Outro pacote que podemos destacar é o pacote de *Config*, que não faz parte da arquitetura MVC, porém é de suma importância para o funcionamento da aplicação.

Neste pacote há classes referentes a configurações importantes e que possibilitam a comunicação entre o *front-end* e a API.

Cross-origin Resource Sharing - CORS

Na classe *WebConfig* é realizada a configuração de CORS, um mecanismo utilizado por navegadores para o compartilhamento de recursos entre diferentes origens. Sendo uma especificação do W3C [W3schools 1999], CORS faz uso de *Headers* do HTTP para informar aos navegadores se determinado recurso pode ser utilizado ou não.

Se tratando de duas aplicações distintas o *front-end* (camada de visualização da arquitetura MVC) e o *back-end*, sem essa configuração todas as requisições vindas do *front-end* da aplicação seriam recusadas pela API.

Na Figura 3.3 é mostrada a classe *WebConfig* com a configuração de *CORS* da aplicação, permitindo assim que todas as requisições de origem do *front-end* sejam recebidas e processadas pelo *back-end*.

```
@Configuration
public class WebConfig {

    @Bean
    public FilterRegistrationBean<CorsFilter> corsFilterRegistrationBean(){
        List<String> all = Arrays.asList("*");

        CorsConfiguration corsConfiguration = new CorsConfiguration();
        corsConfiguration.setAllowedOrigins(all);
        corsConfiguration.setAllowedHeaders(all);
        corsConfiguration.setAllowedMethods(all);
        corsConfiguration.setAllowCredentials(false);

        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration( pattern: "**", corsConfiguration);

        CorsFilter cors = new CorsFilter(source);

        FilterRegistrationBean<CorsFilter> filter = new FilterRegistrationBean<>(cors);
        filter.setOrder(Ordered.HIGHEST_PRECEDENCE);

        return filter;
    }
}
```

Figura 3.3: Classe *WebConfig* (Imagem: Autoria própria).

Fluxo de Autenticação com Oauth 2.0

Outro recurso da aplicação, que também está configurado no pacote, é referente à autenticação entre aplicações. Foi utilizado o protocolo de autorização Oauth 2 que permite que uma aplicação se autentique na outra. Isso é possível quando uma aplicação pede permissão de acesso para um usuário, mesmo que pra isso ela não tenha acesso a nenhuma senha do usuário, podendo conceder ou não o acesso à aplicação.

O cliente acessa a aplicação Angular fornecendo seu nome de usuário e senha. A aplicação Angular acessa o servidor de autorização, utilizando o *clientId* e o *clientSecret*, conforme mostrado na Figura 3.4.

Por sua vez, o servidor de autorização, implementado na classe *AuthorizationServerConfig*, valida o *clientId* e o *clientSecret*, além de verificar se as informações do cliente fornecidas são válidas. Caso as informações sejam validas, é enviado um *Token* de acesso (detalhado na próxima seção) para a aplicação Angular.

O *Token* de acesso à aplicação faz requisições para a API ou servidor de recursos, também implementada em uma classe `ResourceServerConfig`, no pacote `config`. É feita a validação desse *Token* que contém dados do cliente, dados da aplicação Angular. Caso validado corretamente é retornado o recurso para a aplicação Angular em formato JSON.

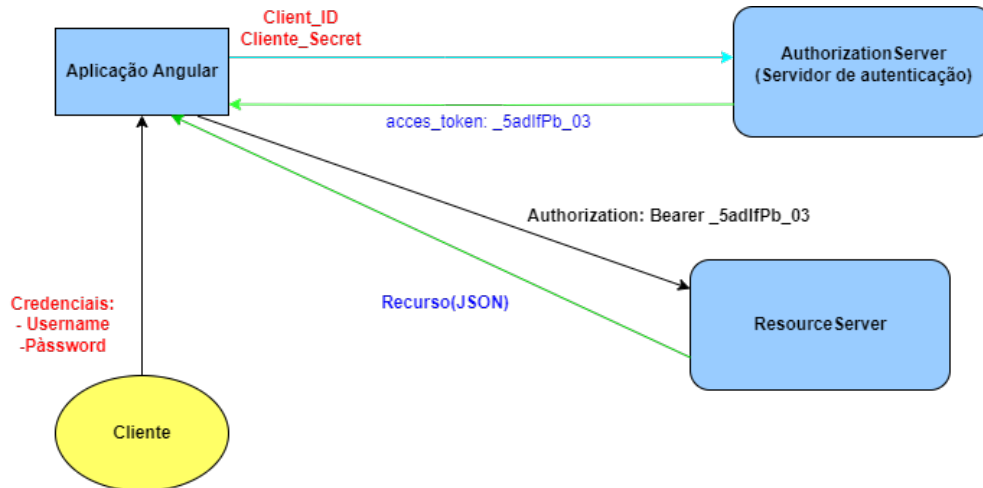


Figura 3.4: Fluxograma de autenticação com *oauth2* (Imagem: Autoria própria).

Token de Acesso

Como descrito, foi decidido utilizar *Token* JWT, um padrão aberto, que define uma forma compacta e autocontida de transmissão de informações de forma segura entre partes com um objeto JSON. Essas informações são confiáveis e podem ser verificadas porque são assinadas digitalmente. Os JWT são assinados usando uma chave secreta com o algoritmo HMAC (Código de autenticação de mensagem baseado em Hash) ou um par de chaves pública/privada usando *Rivest-Shamir-Adleman* (RSA) ou *Elliptic Curve Digital Signature Algorithm* (ECDSA) [Auth 2022].

Sendo formado por cabeçalho, carga útil e assinatura, com o cabeçalho em duas partes com o de token, que no caso é JWT, e o algoritmo de assinatura usado.

A segunda parte do token é a carga útil, contendo as declarações sobre uma entidade como dados do usuário e dados adicionais. A assinatura é gerada com o cabeçalho e a carga útil, ambas codificadas com uma chave secreta.

Assim um token JWT, com cabeçalho e a carga útil, codificados e assinados, apresenta o formato mostrado na Figura 3.5, nas cores vermelha temos as informações do cabeçalho do *token* como o algoritmo de encriptação que está sendo utilizado e o tipo do *token*. Em roxo, as informações do *Payload*, como nome de usuário (a identificação do cliente) e as permissões que esse usuário possui. Por fim, na cor azul, há a assinatura do *token*.

```

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
rG99IiwiaXNjb2NpYWwiOnRydWV9.
4pcPyMD09o1PSyXnrXCjTwXyr4BsezdI1AVTmud2fU4

```

Figura 3.5: *Token JWT (Imagem: Site da documentação JWT).*

Na Figura 3.6 é apresentado um *Token JWT* utilizado na aplicação OrientaIFPB, para login do usuário do sistema. O *Token* pode ser decodificado no *Site* <https://jwt.io/>.

The image shows the JWT.io website interface. On the left, under the 'Encoded' tab, a long alphanumeric string is pasted. Below it, a warning message states: 'Warning: Looks like your JWT signature is not encoded correctly using base64url (https://tools.ietf.org/html/rfc4648#section-5). Note that padding ("=") must be omitted as per https://tools.ietf.org/html/rfc7515#section-2'. On the right, under the 'Decoded' tab, the token is broken down into three parts: 'HEADER: ALGORITHM & TOKEN TYPE', 'PAYLOAD: DATA', and 'VERIFY SIGNATURE'. The header shows 'alg': 'HS256' and 'typ': 'JWT'. The payload is a JSON object containing 'exp', 'user_name', 'authorities', 'jti', 'client_id', and 'scope'. The 'scope' array includes 'read' and 'write'. The signature verification section shows the HMACSHA256 algorithm and the resulting signature string.

Figura 3.6: *Token JWT decodificado (Imagem: site JWT).*

Nesta Seção foram apresentados alguns pontos importantes, relacionados ao desenvolvimento do *back-end* da aplicação. Na próxima Seção serão apresentados pontos importantes do *front-end*.

3.3 Desenvolvimento front end

Nesta Seção são mostrados alguns pontos importantes no desenvolvimento da interface da aplicação, assim como as funcionalidades da página.

Para o desenvolvimento da interface da aplicação foi escolhido o *framework* Angular, devido a sua estrutura e organização, que de forma modularizada torna mais fácil organizar e estruturar o sistema.

Sendo uma aplicação SPA *Single Page Applications*, a aplicação OrientaIFPB foi desenvolvida utilizando esse método relativamente novo, que apesar do nome não significa que a página tenha apenas uma página.

O que diferencia uma página SPA para uma *multi-page* é como ela é carregada no navegador do cliente. As páginas tradicionais são renderizadas do lado do servidor em que, a cada nova página que precisa ser carregada, é realizada uma nova requisição ao servidor. Nas aplicações SPA, todos os recursos necessários são carregados de uma vez em uma única requisição ao servidor [Guedes 2020].

3.3.1 Estrutura do projeto

Conforme mencionado, a estrutura de uma aplicação Angular pode ser construída de forma modularizada. Na Figura 3.7 é apresentado um diagrama com a estrutura do projeto, onde é possível identificar os módulos da aplicação. Dentro desses módulos são criados os componentes renderizados na página.

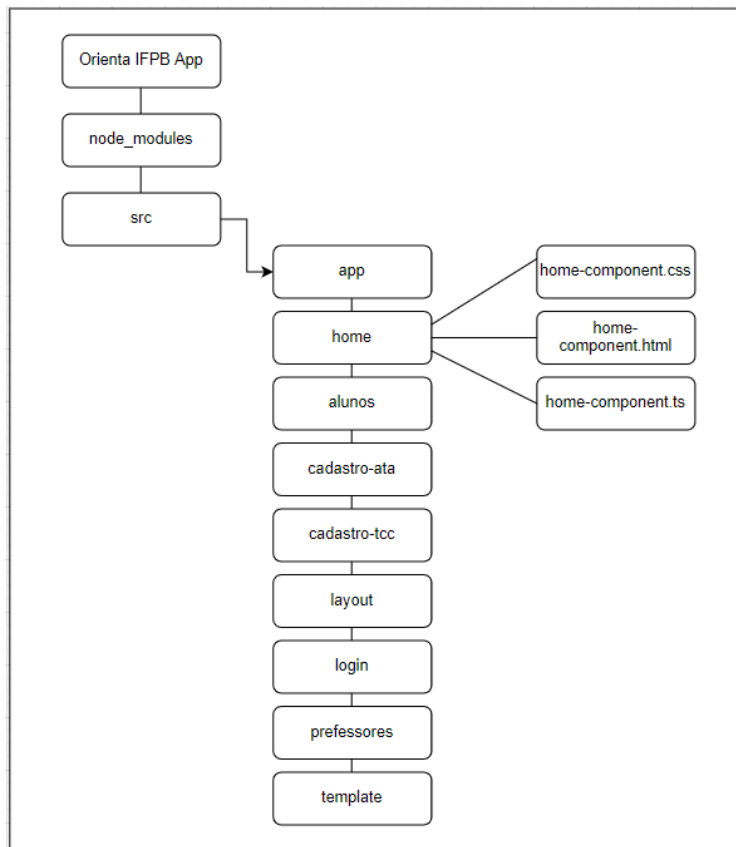


Figura 3.7: Estrutura do projeto (Imagem: Autoria própria).

Podemos verificar os módulos de alunos, cadastro-ata, cadastro-tcc, login, professores layout e home, dentro de cada componente há três arquivos principais, sendo eles um *HTML* responsável pela estrutura, um *CSS* responsável pela estilização do componente e um arquivo *Typescript*, responsável pela lógica necessária para o funcionamento do componente.

3.3.2 Camada de serviços

Com a necessidade de se conectar com a API ou o *back-end* da aplicação, foram implementadas classes de serviços para cada componente do sistema que tem essa necessidade de conexão.

As requisições são enviadas para o *back-end* utilizando verbos HTTP, requisições essas que batem na camada de controle da API, a partir desse momento são processadas e devolvidas para o *front-end* da aplicação.

Na Figura 3.8 é mostrado o digrama de seqüência para o serviço de alunos, para a comunicação com o controller, por meio de requisições get, post, put e push na classe `alunos.service`. Para os demais componentes é seguida a mesma lógica de implementação.

Na seqüência evidenciada, é apresentado o fluxo para salvar dados do aluno, quando o usuário preenche o formulário e salva essas informações, a camada de serviços é acionada realizando comunicação com a API, seguindo o fluxo já evidenciado na seção 3.2.1.

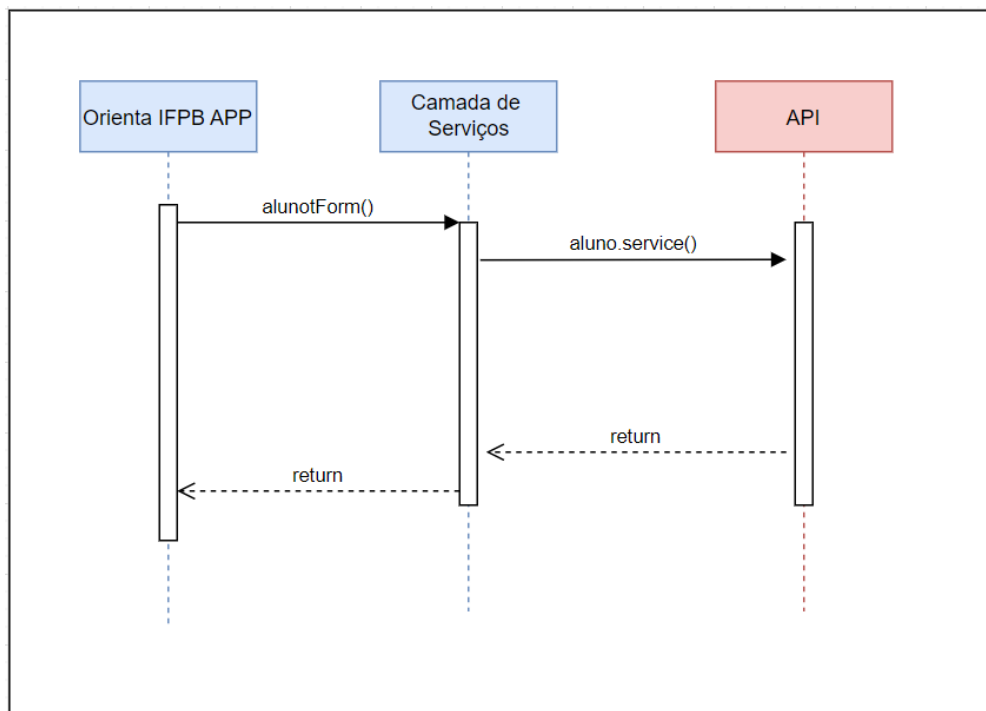


Figura 3.8: Diagrama de seqüência, para o serviço de alunos (Imagem: Autoria própria).

Nesta Seção foram abordados alguns pontos importantes no desenvolvimento da interface do sistema, também foi apresentada a estrutura do sistema, e suas principais funcionalidades.

Capítulo 4

Resultados e Discussões

4.1 Cenário

Com o objetivo de mostrar os resultados obtidos com o sistema, a aplicação foi instalada na instituição de ensino superior IFPB, com o intuito de auxiliar os coordenadores de área/curso nos processos referentes a orientação de alunos.

Neste capítulo é apresentado o sistema, em sua versão atual e funcionalidades implementadas, bem como alguns testes realizados, através da captura das telas da plataforma e seus respectivos casos de uso.

4.2 Cadastro no sistema

Acessando o endereço da aplicação, a tela de *login* é apresentada para o usuário conforme mostrado na Figura 4.1. Em um primeiro acesso é necessária a criação de um usuário do sistema, pressionando o botão de novo cadastro, o usuário é direcionado para a página de cadastro.

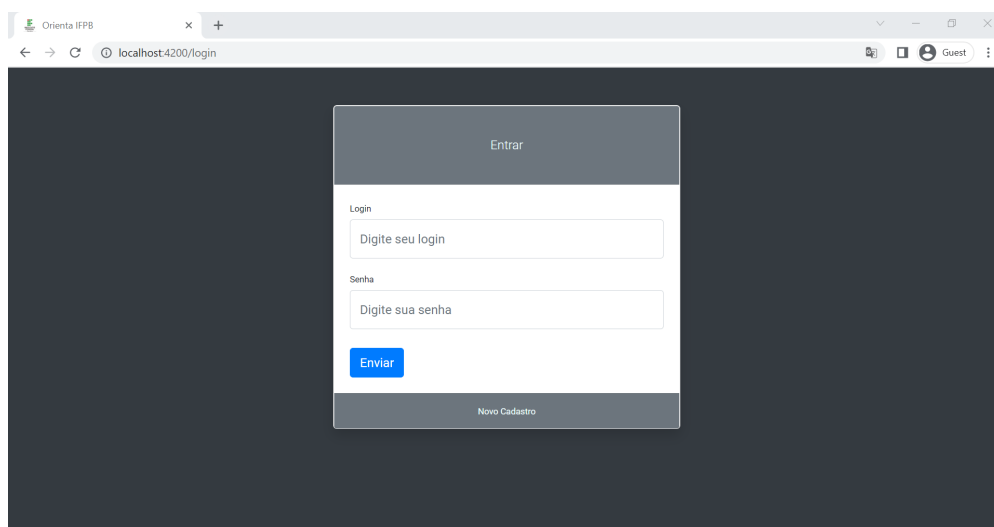


Figura 4.1: Tela de login (Imagem: Autoria própria).

Na Figura 4.2 é mostrada a tela de cadastro para um novo usuário do sistema. Nesse primeiro momento o cadastro está sendo feito como perfil de administrador. Após realizar o cadastro, o usuário já pode efetuar seu acesso na tela de *login*.

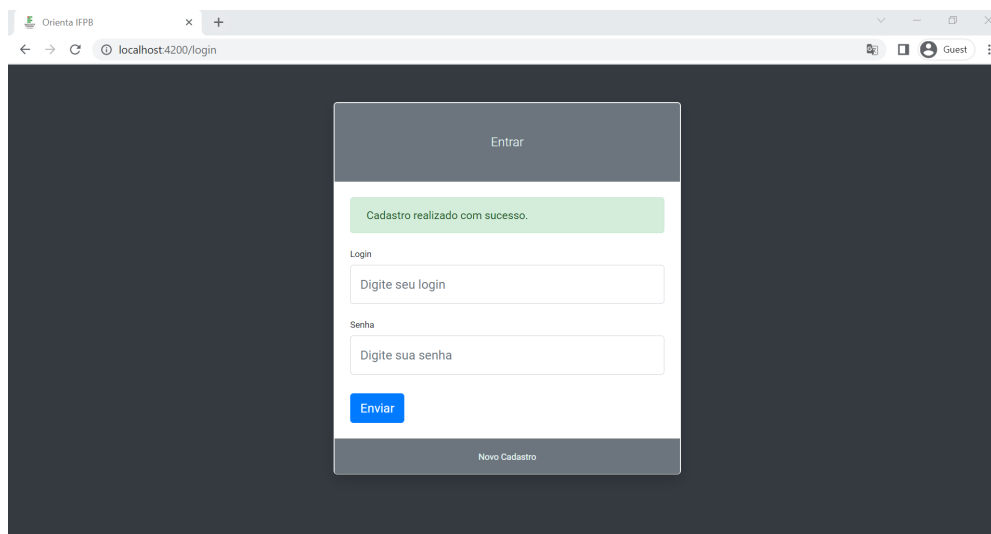


Figura 4.2: Tela de cadastro (Imagem: Autoria própria).

4.3 Pagina Home

Ao realizar *login* na aplicação o usuário é direcionado para a página principal da aplicação, em que é apresentada a quantidade de orientações em andamento, concluídas e/ou trancadas.

Há também um campo de pesquisa, em que é possível realizar buscas por professores cadastrados no sistema com o intuito de verificar se ele está disponível para novas orientações ou com orientações em andamento. Também é apresentado o número de horas acumuladas pelo professor.

Na Figura 4.3 é mostrada a página *home* da aplicação.

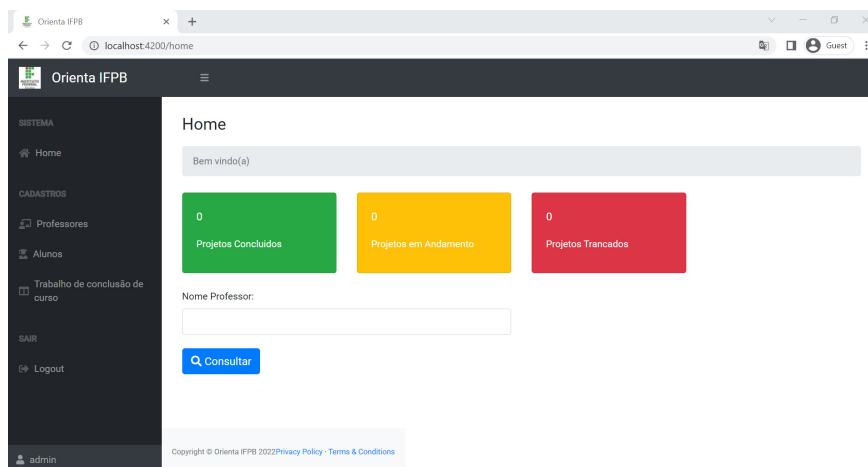
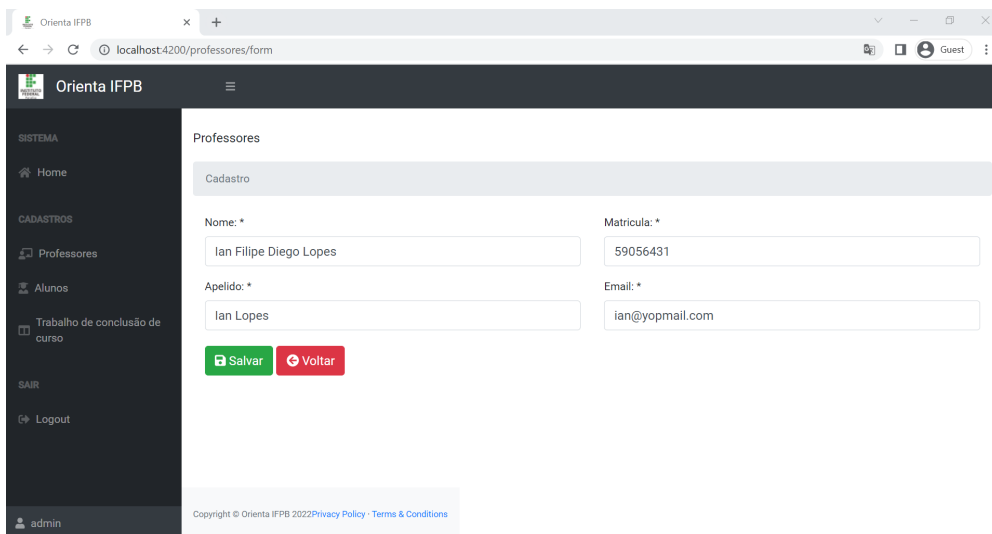


Figura 4.3: Página Home (Imagem: Autoria própria).

No lado esquerdo da tela *home*, há o menu da aplicação, em que é possível realizar os cadastros de alunos, professores e orientações, além do botão de *logout* do sistema.

4.4 Cadastro de Professores

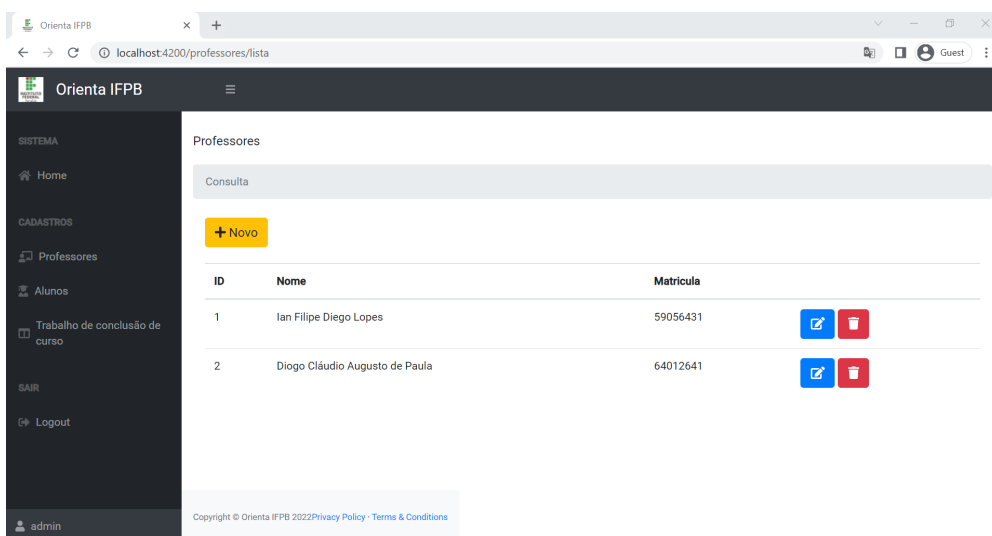
Na Figura 4.4 é apresentado o formulário de cadastro dos professores do sistema, sendo inseridas informações como nome, matrícula e “apelido” esse sendo formado pelo primeiro e último nome do professor, e por último o e-mail.



The screenshot shows a web browser window with the URL `localhost:4200/professores/form`. The page title is "Orienta IFPB" and the main heading is "Professores". Below the heading is a "Cadastro" section with four input fields: "Nome: *" (filled with "Ian Filipe Diego Lopes"), "Matricula: *" (filled with "59056431"), "Apelido: *" (filled with "Ian Lopes"), and "Email: *" (filled with "ian@yopmail.com"). At the bottom of the form are two buttons: "Salvar" (green) and "Voltar" (red). The left sidebar contains a menu with options like "Home", "Professores", "Alunos", "Trabalho de conclusão de curso", and "Logout". The user is logged in as "admin".

Figura 4.4: *Formulário de cadastro de professores (Imagem: Autoria própria).*

Após cadastro, o sistema lista todos os professores cadastrados no sistema, conforme é mostrado na Figura 4.5. Vale salientar que todos os dados cadastrados são dados fictícios, qualquer semelhança com dados ou informações reais é uma mera coincidência.



The screenshot shows a web browser window with the URL `localhost:4200/professores/lista`. The page title is "Orienta IFPB" and the main heading is "Professores". Below the heading is a "Consulta" section with a yellow "+ Novo" button. A table lists the registered teachers with columns for "ID", "Nome", and "Matricula". Each row has two action buttons: a blue edit button and a red delete button.





ID	Nome	Matricula		
1	Ian Filipe Diego Lopes	59056431		
2	Diogo Cláudio Augusto de Paula	64012641		

Figura 4.5: *Lista de professores (Imagem: Autoria própria).*

4.5 Cadastro de Alunos

Ao selecionarmos a opção de cadastro de aluno o usuário é direcionado para a página de listagem de alunos, o sistema apresenta o mesmo comportamento da página de listagem de professores.

A Figura 4.6 faz referência a essa página que tem a função de listar os alunos cadastrados no sistema.

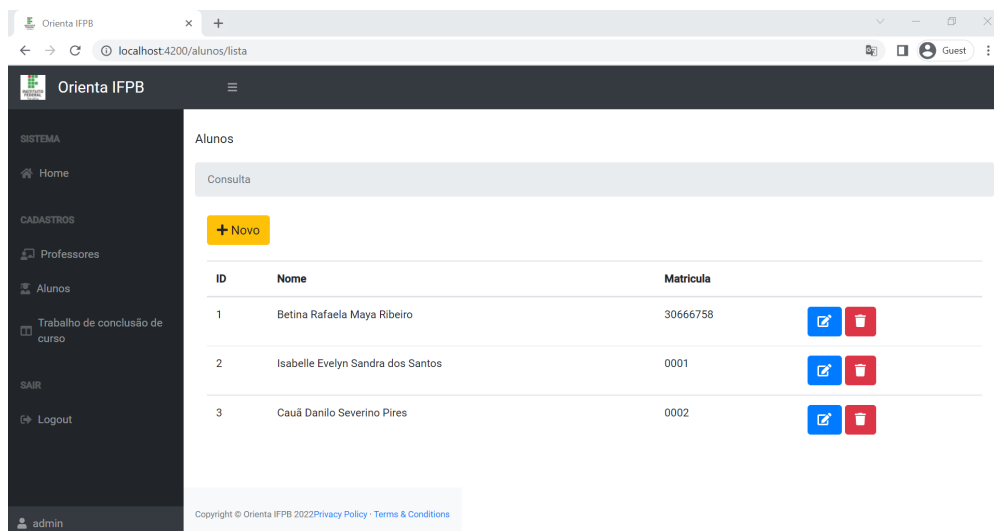


Figura 4.6: Lista de alunos (Imagem: Autoria própria).

Com a necessidade de cadastrar os alunos que estão em período de final de curso que defenderão seu projeto/trabalho de conclusão de curso, ao selecionarmos a opção de novo cadastro, nos deparamos com o formulário de cadastrado do aluno. Realizado o cadastro do aluno conforme mostrado na Figura 4.7.

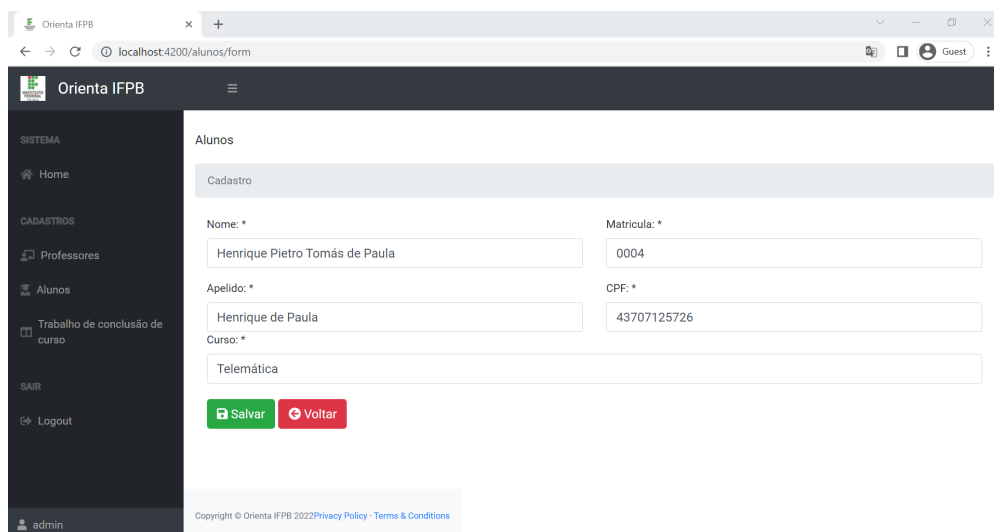


Figura 4.7: Formulário de cadastro de alunos (Imagem: Autoria própria).

Caso exista a necessidade de alterar alguma informação cadastrada, o sistema fornece a opção de atualização (botão azul) em destaque na Figura 4.8, e caso tenha a necessidade de

deletar informações do sistema há a opção de deleção (botão vermelho).

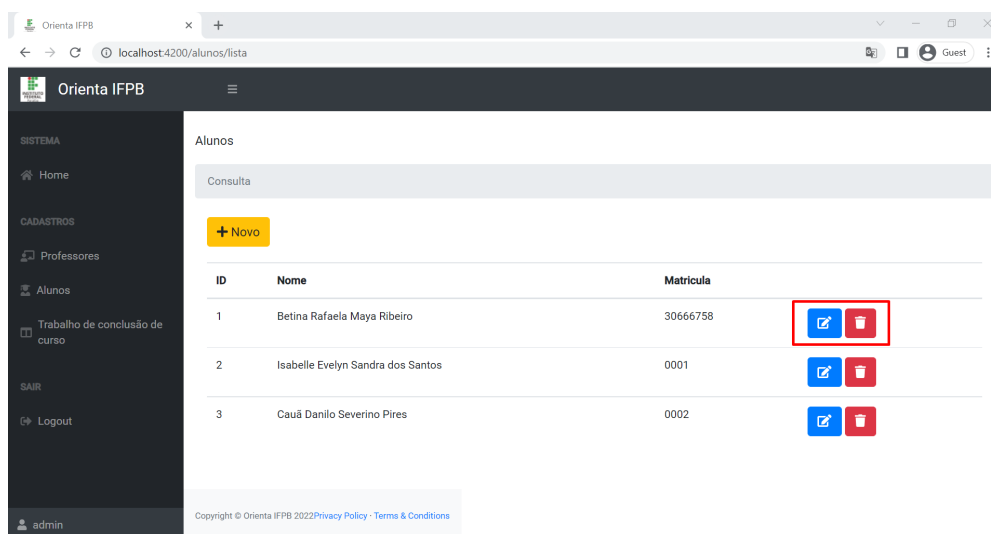


Figura 4.8: Opções de atualizar/deletar dados (Imagem: Autoria própria).

4.6 Cadastro Trabalho de Conclusão de Curso

Como o objetivo da aplicação é o cadastro de orientações para trabalhos de conclusão de cursos, há a aba para cadastro de orientação. Ao selecionar a opção de um novo cadastro de orientação, é apresentado o formulário de cadastro mostrado na Figura 4.9.

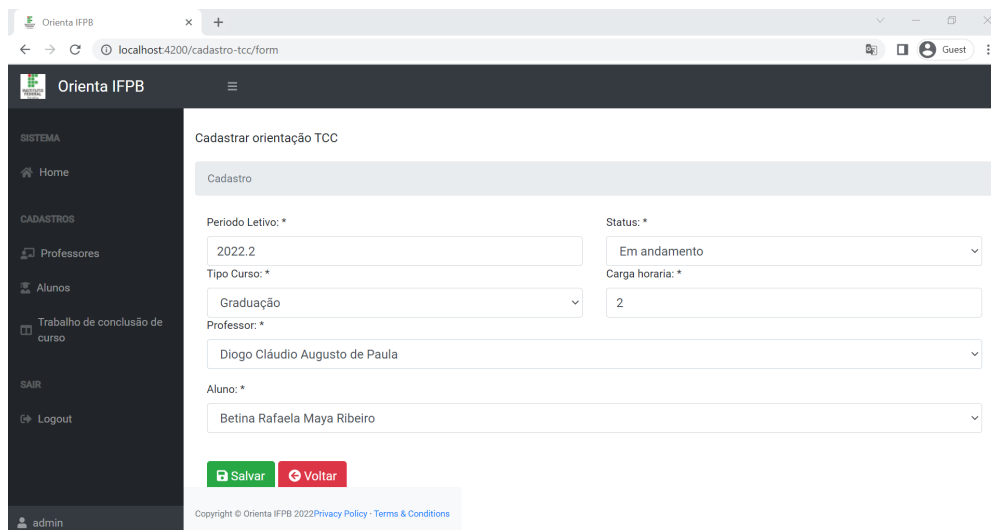


Figura 4.9: Tela de cadastro TCC (Imagem: Autoria própria).

No formulário são inseridas as informações de período letivo, status referente a orientação posteriormente podendo ser atualizado para concluído ou trancado, é cadastrado também o tipo do curso, onde temos as opções de graduação, técnico ou pós-graduação.

Neste formulário também é cadastrada a carga horaria referente à orientação, e por fim o professor/orientador e o aluno.

Após cadastro, o sistema lista todas as orientações salvas na base de dados. Nas próximas seções são mostradas as funcionalidade de cadastro de reuniões e agendamento de defesa.

Na Figura 4.10 está evidenciada a opção de reuniões, apresentada na próxima seção.

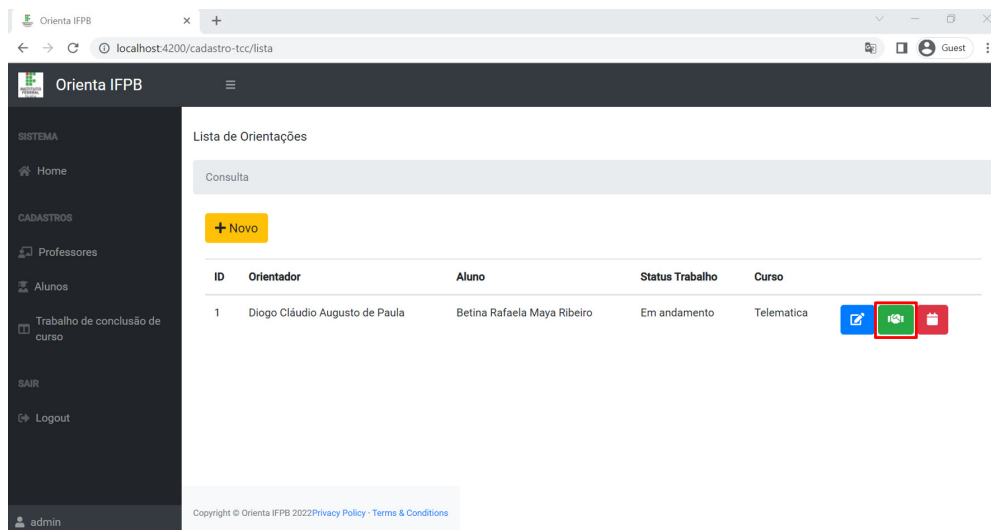


Figura 4.10: Lista de orientações (Imagem: Autoria própria).

4.6.1 Cadastro de Reuniões

Nesta seção é apresentada a tela de cadastro de reuniões, para o acompanhamento do aluno por parte do orientador, o sistema possui a função de cadastro de reuniões, que possibilita ao professor registrar toda reunião de orientação que teve com o aluno.

Essas reuniões ficam salvas na base de dados da aplicação, servindo como base de consulta tanto para o aluno quanto para o professor.

É possível observar o formulário para o cadastro dessas reuniões na Figura 4.11. Onde todo o conteúdo da reunião pode ser salvo no sistema, assim como a data da reunião.

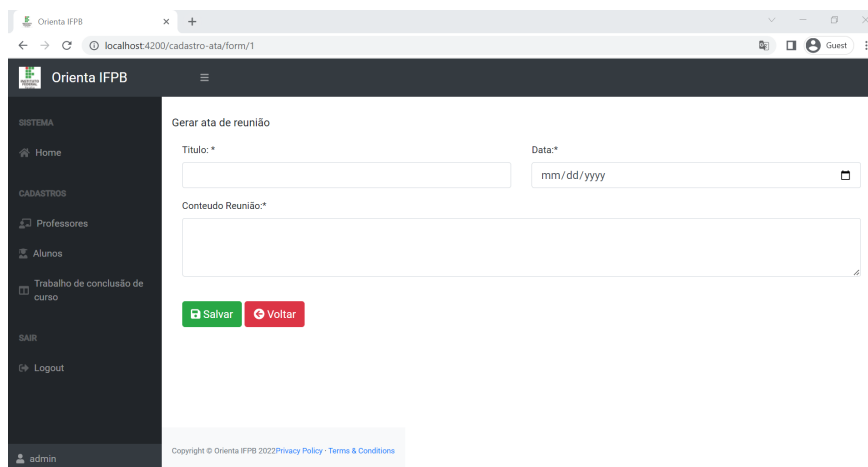


Figura 4.11: Formulário de cadastro de reuniões (Imagem: Autoria própria).

Com um comportamento semelhante aos demais fluxos de cadastro, a aplicação também lista as reuniões realizadas entre aluno/orientador, ficando evidenciado na Figura 4.12.

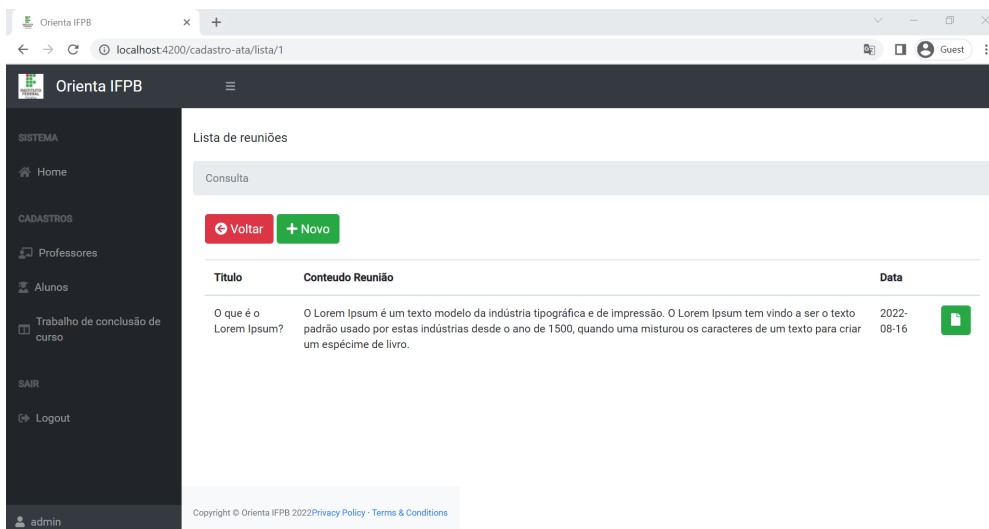


Figura 4.12: *Conteúdo de reuniões realizadas (Imagem: Autoria própria).*

4.6.2 Agendamento de defesa

Nesta seção é mostrada a opção de agendamento de defesa da monografia, podendo ser realizada ao pressionar o botão destacado na Figura 4.13.

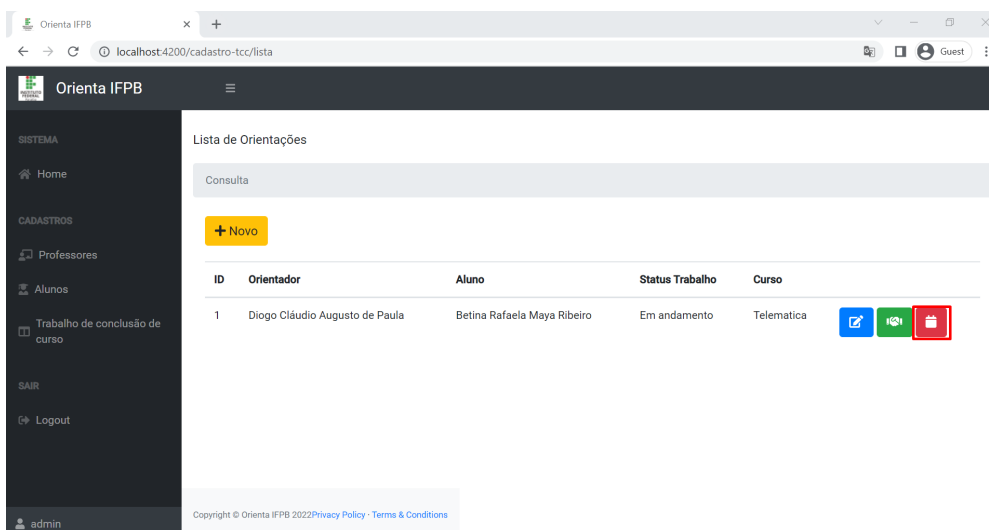


Figura 4.13: *Realizar agendamento (Imagem: Autoria própria).*

Após todos os encontros referentes à orientação, pode-se realizar o agendamento da defesa da monografia, sendo essa a última etapa para a conclusão do curso ao qual o aluno está matriculado.

Ao selecionar a opção destacada na Figura 4.13, o usuário é direcionado para a página de agendamento de defesa, apresentado na Figura 4.14.

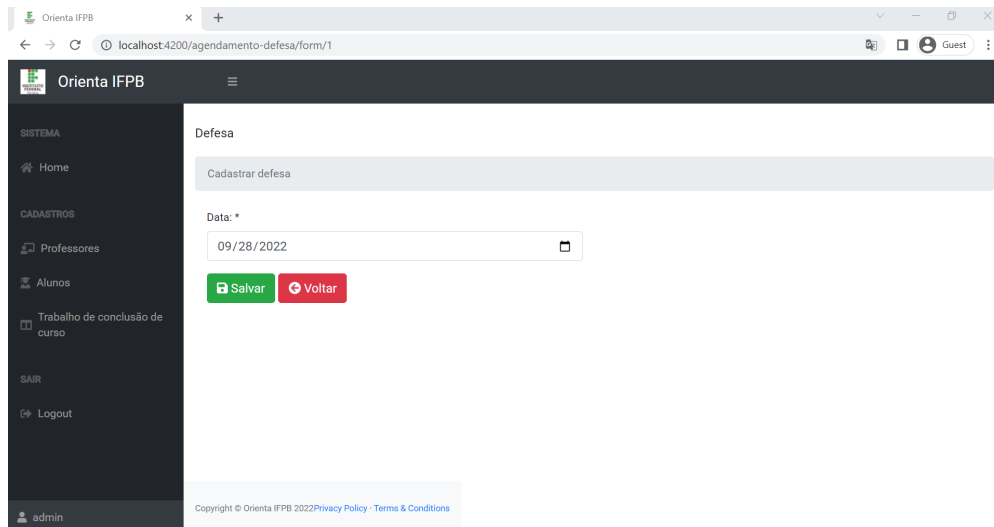


Figura 4.14: *Agendando defesa de monografia (Imagem: Autoria própria).*

Nesta seção é mostrada a atual versão da aplicação e suas funcionalidades que estão prontas e funcionais. Pode-se concluir que a aplicação cumpre os requisitos que foram propostos no início do desenvolvimento do sistema.

Capítulo 5

Conclusão

O objetivo principal deste trabalho é a implementação do OrientaIFPB: um sistema que auxilia no acompanhamento da evolução dos trabalhos de conclusão de curso de instituições de ensino superior. Logo, o orientador do aluno e o coordenador de curso supervisionam as etapas do andamento das atividades.

A priori, foi realizado um levantamento para definir quais tecnologias seriam usadas ao desenvolver o sistema, sendo decidido o uso *frameworks*: Angular, para evolução da interface gráfica, visando agilizar a organização e estruturação do projeto; e *Spring Boot*, para criação de uma API usando Java 8, por este último ser uma linguagem madura e de ampla adoção na comunidade. Além disso, a aplicação foi construída utilizando a arquitetura de *Software MVC*, visando padronização no código e na identidade visual, e agilidade na estruturação do sistema.

Após o desenvolvimento, foram feitos testes os quais constataram que tanto requisitos funcionais quanto específicos estabelecidos inicialmente foram alcançados. Desta forma, a aplicação atende à maioria dos processos relacionados às atividades de desenvolvimento do TCC.

5.1 Trabalhos Futuros

Nesta Seção serão apresentadas algumas sugestões de possíveis melhorias para o sistema, e mostrada algumas ideias que foram pensadas no início do projeto, porém não foram incluídas no sistema atual, mas que podem ser implementadas em trabalhos futuros.

- Implementar níveis de acesso a aplicação de acordo com o perfil do usuário, no momento a aplicação está com apenas o perfil de administrador configurado.
- Finalizar a integração do *front-end* com o *back-end*, referente a busca de professores para assim conseguir gerar relatórios do sistema, referente a quantidade de horas e disponibilidade de professores para o papel de orientação.

- Concluir o desenvolvimento da funcionalidade de geração de atas do sistema, o desenvolvimento foi iniciado, faltando apenas finalizar o *Template* da ata e integrar com o *front-end* da aplicação.
- Envio automático de e-mails para membros da banca com detalhes sobre a defesa e com as atas geradas.
- Atualmente o sistema permite o acompanhamento para trabalhos de conclusão de curso, podendo ser disponibilizada a opção para projetos de extensão e pesquisa.
- Adaptar o sistema para dispositivos moveis, o que será facilitado devido o uso do *Bootstrap* no desenvolvimento do sistema.

Apêndice A

Instalação OrientaIFPB APP

Este capítulo tem como objetivo, o de mostrar o processo de instalação da aplicação para testes. O código fonte da aplicação *front-end* e da API estão salvos em seus respectivos repositórios no GitHub.

A.1 *Clone dos repositórios*

A aplicação OrientaIFPB App está salva no repositório do github na url a seguir:

- <https://github.com/diegomardu/orienta-ifpb-app>

Bastando apenas realizar o clone do respoitorio utilizando o git-bash, o *Git Bash* é o aplicativo para ambientes do *Microsoft Windows* que oferece a camada de emulação para a experiência de linha de comando Git.

- `git clone https://github.com/diegomardu/orienta-ifpb-app.git`

Já o código referente a API, se encontra no repositório a seguir, devendo ser realizado o mesmo procedimento para o clone do repositório.

- `git clone https://github.com/diegomardu/api-cadastro-tcc.git`

A.2 *Front-end*

Como pré-requisito de instalação do *front-end* da aplicação, a máquina onde a mesma será instalada precisa ter o Node-Js instalado, podendo ser facilmente encontrado no site <https://nodejs.org/en/>.

Após a instalação na máquina teremos acesso npm, que é ao gerenciador de pacotes do Node-js, como próximo passo devemos realizar a instalação do Angular-cli utilizando o comando abaixo:

- `npm install -g @angular/cli`

Depois de realizado clone/download da aplicação, entrar na pasta criada abrindo um terminal e executando o comando.

- **npm install**

Com a execução desse comando é feito a instalação de todas as dependências para que a aplicação funcione corretamente, após a instalação de suas dependências basta executar o comando a seguir para que a aplicação seja executada.

- **ng serve --open**

A.3 *Back-end*

Após realizar o clone do código basta abrir em uma IDE (*eclipse, sts, intelij*), com o código aberto basta editar o arquivo `application.properties` da seguinte forma.

- **spring.profiles.active=test**

Depois só executar a aplicação, a mesma ira subir com o *profile* de testes ativado, dessa é utilizado o bando em memoria H2 ideal para testes.

Referências Bibliográficas

[Afonso 2017] AFONSO, A. O que é spring boot? 2017. Disponível em: <<https://blog.algaworks.com/spring-boot/>>. 9, 10

[Afonso 2018] AFONSO, A. O que é angular? 2018. Disponível em: <<https://blog.algaworks.com/o-que-e-angular/>>. 3, 7

[Auth 2022] AUTH. Introdução aos tokens da web json. 2022. Disponível em: <<https://jwt.io/introduction>>. 17

[Candioli 2020] CANDIOLLI, S. Começando com spring security. 2020. Disponível em: <<https://medium.com/cwi-software/come%C3%A7ando-com-spring-security-86a3caec8c40>>. 10

[Guedes 2020] GUEDES, M. O que são aplicações spa? 2020. Disponível em: <<https://www.treinaweb.com.br/blog/o-que-sao-aplicacoes-spa>>. 19

[Lima 2021] LIMA, G. Bootstrap - o que é, como e quando usar? 2021. Disponível em: <<https://www.alura.com.br/artigos/bootstrap>>. 5, 6

[Melo 2020] MELO, D. O que é java? [guia para iniciantes]. 2020. Disponível em: <<https://tecnoblog.net/responde/o-que-e-java-guia-para-iniciantes/>>. 9

[Melo 2021] MELO, D. O que é javascript? [guia para iniciantes]. 2021. Disponível em: <<https://tecnoblog.net/responde/o-que-e-javascript-guia-para-iniciantes/>>. 6

[Puluceno 2012] PULUCENO, T. V. Estudo de caso sobre uma api rest utilizando a abordagem de programação orientada e eventos com a plataforma node.js. 2012. 6, 7

[Rossalli 2021] ROSSALLI, B. Spring boot: como começar. 2021. Disponível em: <<https://www.zup.com.br/blog/spring-boot>>. 9

[Roveda 2020] ROVEDA, U. Desenvolvimento web: O que É e como ser um desenvolvedor web. 2020. Disponível em: <<https://kenzie.com.br/blog/desenvolvimento-web/>>. 1, 4

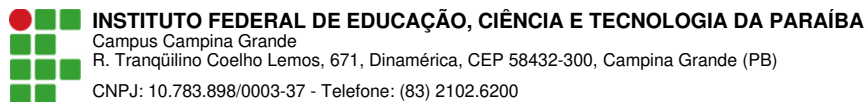
[Santos 2018] SANTOS, M. A. da Silva dos. Processo para criação de serviços web para o encapsulamento e acesso a uma api específica. 2018. 6

[Silva 2020] SILVA, G. O que é arquitetura mvc? 2020. Disponível em: <<https://coodesh.com/blog/dicionario/o-que-e-arquitetura-mvc/>>. 13, 14

[Silva 2017] SILVA, M. R. R. da. Projeto e desenvolvimento de um sistema para gerenciamento de trabalhos de conclusão de curso. 2017. 1, 5, 6

[Totvs 2020] TOTVS, E. Por que o angular é um framework tão poderoso? 2020. Disponível em: <<https://www.totvs.com/blog/developers/angular/>>. 7, 8

[W3schools 1999] W3SCHOOLS, E. Html tutorial. 1999. Disponível em: <<https://www.w3schools.com/html/>>. 5, 16



Documento Digitalizado Restrito

Entrega do Tcc

Assunto: Entrega do Tcc
Assinado por: Diego Duarte
Tipo do Documento: Projeto
Situação: Finalizado
Nível de Acesso: Restrito
Hipótese Legal: Informação Pessoal (Art. 31 da Lei no 12.527/2011)
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- **Diego Martins Duarte França, ALUNO (201721210006) DE TECNOLOGIA EM TELEMÁTICA - CAMPINA GRANDE**, em 23/09/2022 14:40:14.

Este documento foi armazenado no SUAP em 23/09/2022. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse <https://suap.ifpb.edu.br/verificar-documento-externo/> e forneça os dados abaixo:

Código Verificador: 633627

Código de Autenticação: f43db45329

