

UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Benjamin Erich Vichel

Desenvolvimento Front End UniConnect:
Plataforma de vagas para alunos da universidade

Porto Alegre
2024

Benjamin Erich Vichel

**Desenvolvimento Front End UniConnect:
Plataforma de vagas para alunos da universidade**

Relatório final de Estágio apresentado
como requisito para a aprovação na
atividade acadêmica de Estágio do Curso
de Ciência da Computação da
Universidade do Vale do Rio dos Sinos –
UNISINOS.

Professora supervisora: Profa. Dra. Andriele Busatto do Carmo

Porto Alegre

2024

LISTA DE FIGURAS

Figura 1 - Vantagem Vite	12
Figura 2 - Organização pastas.....	20
Figura 3 - Utilização BrowserRouter	21
Figura 4 - Mapeamento URLs.....	21
Figura 5 – AuthContext	22
Figura 6 - Exemplo implementação AuthProvider.....	22
Figura 7 - Exemplo código na UseApi.....	23
Figura 8 - Type Job.....	24
Figura 9 - Type State	25
Figura 10 - Type User	25
Figura 11 - Componentes	26
Figura 12 - Principais páginas	26
Figura 13 - Página Home	27
Figura 14 - Página Works	27
Figura 15 - Página Login.....	28
Figura 16 - Página Registrar.....	28
Figura 17 - Página Currículo.....	29
Figura 18 - Página Candidaturas	29
Figura 19 - Página Admin	30
Figura 20 - Componente JobDetails e JobsRemove.....	30
Figura 21 - Componente Jobs	31
Figura 22 - Componente JobsForm	31
Figura 23 - Componente Search.....	32
Figura 24 - Componente StateSelect.....	32
Figura 25 - Uso do Axios	33

LISTA DE TABELAS

Tabela 2 - Cronograma de atividades.....	18
--	----

LISTA DE SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
NBR	Normas Brasileiras de Regulação
UI	Interface do usuário
UX	Experiência do usuário
IDE	Ambiente de desenvolvimento Integrado
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
DOM	Document Object Model
MEC	Ministério da Educação
EAD	Educação a Distância

SUMÁRIO

1 INTRODUÇÃO	7
1.1 Objetivos	7
1.1.1 Objetivo geral	7
1.1.2 Objetivos específicos.....	7
2 IDENTIFICAÇÃO DA EMPRESA.....	8
3 FUNDAMENTAÇÃO TEÓRICA	10
3.1 JavaScript	10
3.2 Paradigmas de programação	10
3.2.1 Programação Orientada a Objetos	10
3.2.2 Programação Funcional	11
3.3 JSON.....	11
3.4 React.....	11
3.4.1 Funcionamento geral.....	11
3.5 Vite.....	12
3.6 IDE Visual Studio	13
3.7 HyperText Markup Language (HTML)	13
3.8 Cascading Style Sheets (CSS)	13
3.9 TypeScript.....	13
3.10 Engenharia de Software.....	14
3.10.1 Engenharia de Requisitos	14
3.10.2 Testes.....	15
4 METODOLOGIA	17
4.1 Oportunidades de Melhoria	17
5 CRONOGRAMA	18
6 ATIVIDADES REALIZADAS.....	18
6.1 Especificação dos requisitos	18
6.2 Design Front End e configuração do ambiente de desenvolvimento	18
6.2.1 Criação do projeto e organização das pastas	19
6.2.2 Criação de rotas	20
6.2.3 Criação AuthContext, AuthProvider e UseApi	21
6.2.4 Criação dos Types.....	23
6.2.5 Criação das principais páginas e componentes	25

6.2.6 Funcionamento das páginas	26
6.2.7 Funcionamento dos componentes.....	30
6.2.8 CSS e responsividade	32
6.3 Integração Front End com o Back End.....	32
6.4 Realização de Testes e Implantação.....	33
7 CONSIDERAÇÕES FINAIS	33
REFERÊNCIAS.....	35

1 INTRODUÇÃO

Em muitas instituições de ensino, surge um desafio constante: proporcionar uma plataforma eficiente que conecte seus alunos a uma variedade de oportunidades, como estágios, trabalhos acadêmicos e iniciações científicas. Com frequência, as plataformas existentes são limitadas, atendendo apenas uma dessas modalidades e deixando lacunas nas necessidades dos estudantes.

Este relatório relata o desenvolvimento da aplicação web da UniConnect, plataforma de divulgação de vagas de estágio, trabalho, iniciação científica, entre outros.

Essa atividade de estágio tem como principais atividades a especificação e codificação de interfaces de usuário (UI) e experiência do usuário (UX).

1.1 Objetivos

1.1.1 Objetivo geral

O objetivo geral deste relatório é documentar o processo de desenvolvimento da aplicação web UniConnect, com foco na criação de uma plataforma eficaz para disseminar oportunidades de estágio, emprego e iniciações científicas para alunos da universidade do vale do rio dos sinos – Unisinos. O relatório cobrirá as fases de design, implementação e testes e pretende fornecer uma visão abrangente do desenvolvimento do projeto.

1.1.2 Objetivos específicos

Por meio dos objetivos específicos a seguir, o estagiário busca aplicar os conceitos estudados até o momento no curso, visando consolidar os ensinamentos teóricos e práticos.

- a) Análise dos requisitos e especificações da aplicação UniConnect, identificando necessidades de usuários e os principais recursos a serem implementados.

- b) O desenvolvimento da codificação será utilizado o Vite como ferramenta de construção de projetos. O código será escrito em JavaScript para a lógica da aplicação, HTML para estruturação do conteúdo e CSS para estilização. O ambiente de desenvolvimento integrado (IDE) escolhido para o projeto foi o Visual Studio Code.
- c) Realizar a integração com o Back-End da UniConnect. Nessa etapa serão realizadas as conexões para comunicação eficiente com o Back-End, sendo implementadas as requisições HTTP, manipulação de dados em formato JSON ou outro definido pela Api do Back-End.
- d) Colaboração na execução de testes e na identificação e correção de falhas em conjunto com o Back-End.

2 IDENTIFICAÇÃO DA EMPRESA

A universidade do Vale do Rio dos Sinos (Unisinos) é uma instituição de ensino superior localizada em São Leopoldo, Rio grande do Sul, Brasil. A Unisinos foi fundada pela companhia de Jesus (Jesuítas) em 1969 e começou como uma instituição educacional focada principalmente em áreas como engenharia, ciências sociais e comunicações. Ao longo dos anos, expandiu sua oferta educacional para incluir uma variedade de cursos de graduação, pós-graduação e extensão em diversas áreas do conhecimento, possuindo hoje também MBAs e Especializações, Mestrados e Doutorados e Graduações EAD.

O parque Tecnológico de São Leopoldo - Tecnosinos reúne mais de 100 empresas além de startups. Esse parque está diretamente ligado ao desenvolvimento sustentável da região, gerando mais de 8 mil empregos. No tecnosinos, empresas 7 globais se unem a dezenas de startups incubadas e graduadas na Incubadora Unisinos (Unitec), impulsionando a inovação e dinamizando a economia. A unitec, como parte da Unisinos, é uma unidade de negócios que promove e implementa avanços tecnológicos, integrando expertise acadêmica e corporativa através da pesquisa aplicada.

A universidade já foi reconhecida diversas vezes pelo MEC, com destaque a melhor graduação privada da região Sul do país e como melhor universidade da região Sul.

Além dos polos de São Leopoldo e Porto Alegre, também estão presentes polos em Bento Gonçalves, Cachoeirinha, Campo Bom, Canoas, Caxias do Sul, Estrela, Montenegro, Passo Fundo, Santa Maria, e Taquara. Para modalidade EAD conta com os polos em Florianópolis (SC), Curitiba (PR), Rio de Janeiro (RJ), São Paulo (SP), Teresina (PI), Belo Horizonte e Santa Rita do Sapucaí (MG).

3 FUNDAMENTAÇÃO TEÓRICA

3.1 JavaScript

De acordo com CROCKFORD (2008, p.2),

JavaScript é uma linguagem importante porque é uma linguagem do navegador da web. Sua associação com o navegador a torna uma das linguagens de programação mais populares do mundo.

A linguagem de programação JavaScript (ou muitas vezes chamada de JS) é principalmente reconhecida como a linguagem de scripting para páginas web. O JavaScript suporta os estilos de orientação de objetos, o estilo imperativo e o funcional, por ser uma linguagem que é baseada em multi-paradigma, protótipos e dinâmica. (CROCKFORD, 2018).

É possível utilizar o JS tanto como uma linguagem procedural quanto orientada a objetos, possuindo a capacidade de construção de seus objetos em tempo de execução, diferente de outras linguagens como c++ e Java. (CROCKFORD, 2018).

3.2 Paradigmas de programação

Cada paradigma possui uma abordagem e estilo único para escrever código de computador, com cada uma possuindo as suas próprias regras, conceitos e princípios de como seus programas serão estruturados e executados. Abaixo temos alguns dos paradigmas que são utilizadas durante o projeto. (OSMANI, 2012).

3.2.1 Programação Orientada a Objetos

No paradigma orientado a objetos, usamos objetos para representar entidades do mundo real. De acordo com Osmani (2012, p. 3), *“Um objeto é uma coleção de propriedades nomeadas, uma lista de valores chaves. Algumas das propriedades podem ser funções as quais chamamos de métodos.”*

No JavaScript temos a possibilidade de usar heranças, permitindo que objetos herdem propriedades e métodos de outros objetos, facilitando o compartilhamento de comportamento entre objetos relacionados. (OSMANI, 2012).

3.2.2 Programação Funcional

Cerca de 10 linhas de código são escritos pelos desenvolvedores por dia, pois em média 70% do tempo estamos tentando entender as linhas de código. Por este motivo a programação funcional é muito importante. Aprender seus princípios leva a códigos mais reconhecíveis e rápidos de entender. Por exemplo, depois de aprender a função do `map()`, sempre que ele aparecer você já sabe o que aquele código está fazendo, mas se tiver um loop, mesmo sabendo a função de um loop, você precisa olhar o código para ver qual a sua finalidade. Portanto, quanto mais fácil é o entender de um código, menos tempo utilizaremos em manutenção e leitura, e mais em lógica de programação. (SIMPSON, 2018).

3.3 JSON

Desenvolvido por Douglas Crockford, JSON possui formato em texto, leve e fácil de ler por pessoas. Sua função sendo para trocar informações entre clientes e servidores. O JSON não é dependente do JavaScript, possuindo suporte em praticamente todas as linguagens de programação. Nas aplicações web é principalmente utilizado para a transferência de dados. (Sriparasa, 2013).

3.4 React

O React é uma biblioteca JavaScript popular criada pelo Facebook em 2013, de destacando rapidamente para o desenvolvimento de grandes aplicativos web. (State os JS Survey, 2018).

3.4.1 Funcionamento geral

O React trouxe uma grande inovação que é na manipulação do DOM. O React pega os elementos na árvore DOM que são representados como objetos e cria uma representação virtual dessa árvore, chamamos isso de “Virtual DOM”. Dessa maneira quando um estado sofre alterações, o react compara a árvore virtual com a atual e realiza as alterações somente nos elementos necessários, melhorando muito o desempenho da aplicação. (Accomazzo, Murray, Lerner, Allsopp, 2017).

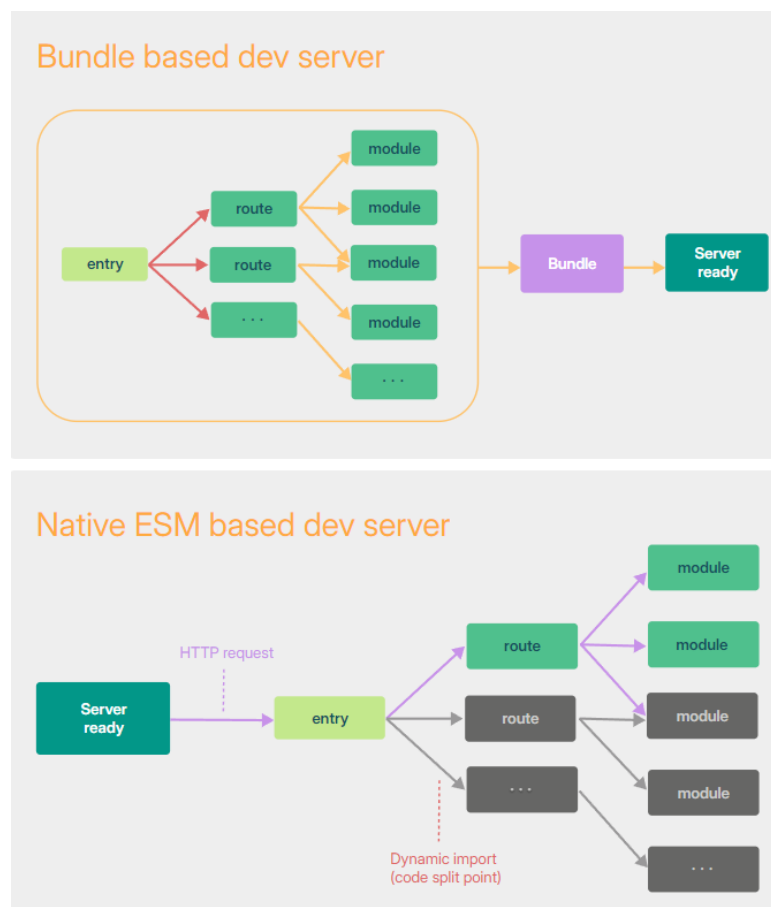
3.5 Vite

Vite é um construtor de aplicações web desenvolvido especificamente para projetos JavaScript, como o React. Serve como servidor de desenvolvimento que oferece atualizações rápidas e instantâneas do navegador.

Uma das grandes vantagens do Vite está relacionado à renderização dos módulos. Quando um módulo é alterado, é necessário recompilar ele novamente, junto também com todos outros módulos, mesmo que esses não tenham sofrido mudanças, resultando em tempos prolongados de renderização. No entanto, com o Vite, é possível realizar a recompilação apenas do módulo que foi alterado, otimizando significativamente o processo de desenvolvimento e minimizando potenciais problemas de desempenho associados à renderização de todos os módulos.

A Figura 1 nos ilustra esse funcionamento.

Figura 1 - Vantagem Vite



Fonte: Retirada site Vite

3.6 IDE Visual Studio

Ambiente de desenvolvimento integrado (IDE) desenvolvido pela Microsoft. Fornece a seus usuários uma variedade de ferramentas e recursos para ajudar a criar, depurar e implantar software em diversas plataformas, incluindo Windows, IOS e Web.

O Visual Studio conta com várias extensões para auxiliar o programador, alguns a serem utilizados nesse projeto são: 'vs code-style-components', com recursos como realce de sintaxe aprimorado, autocompletar, verificação de erros em tempo real entre outros. O 'ES7 + React/Redux/React-Native Snippets' que oferece um conjunto de snippets (fragmentos de código) para facilitar o desenvolvimento de aplicativos usando as tecnologias React, Redux e React Native com JavaScript moderno (ES7+).

3.7 HyperText Markup Language (HTML)

O HTML utiliza elementos para descrever a estrutura de uma página. Para isso utiliza tags, ou podemos pensar nelas como contêineres. Cada contêiner normalmente possui uma abertura e fechamento, fornecendo informação sobre qual tipo de conteúdo possui dentro. (Duckett, 2011).

3.8 Cascading Style Sheets (CSS)

CSS trabalha lado a lado com o HTML. O CSS nos permite criar regras com o objetivo de especificar como algum conteúdo no HTML deve ser exibido. Permite o controle de regras sobre cada um dos elementos de forma individual. Um exemplo de aplicação seria definirmos o plano de fundo da página com a cor azul, fonte Arial e todo texto em negrito. (Duckett, 2011).

3.9 TypeScript

(Cherny, 2019 apud Fenton, 2018).

'E se pudéssemos fortalecer o JavaScript com o que está faltando para o desenvolvimento de aplicações de larga escala, como

digitação estática, classes e módulos... É isso que o TypeScript é sobre.'

O TypeScript é um super conjunto de JavaScript. Ele engloba toda linguagem JavaScript e ainda possui funcionalidades a mais. (Fenton, 2018).

Todo JavaScript é válido TypeScript, ou seja, podemos transferir código JavaScript para TypeScript sem problema algum, já que todas as declarações vão ser válidas. No entanto, não quer dizer que não teremos avisos de erros, pois existe uma diferença na tipagem estática de cada um.

No JavaScript podemos declarar uma variável atribuindo um valor inteiro, logo em seguida, atribuímos para essa mesma variável um valor String e mandamos imprimir na tela. Nesse caso teremos somente um erro aparecendo na tela quando o código já estiver em execução, mas no TypeScript teremos o erro sendo informado durante a compilação do código. (Fenton, 2018).

Outra característica admirável do TypeScript é a tentativa de executar o código mesmo possuindo erros, diferente de outros compiladores que não permitem essa abordagem. (Fenton, 2018).

3.10 Engenharia de Software

Podemos nos referir a engenharia de software como um conjunto de métodos e práticas para o desenvolvimento de software. Ela está presente em todas as etapas de um software, desde a concepção até a sua manutenção. Algumas das principais fases da engenharia de software são as etapas de definição de requisitos, projeto de software, implementação, testes e manutenção.

3.10.1 Engenharia de Requisitos

Definição de requisitos: De acordo com GARCIA (2024) Os requisitos são instruções detalhadas de como um sistema deve se comportar, definindo funcionalidades e limites que esse sistema deve seguir. Também, segundo GARCIA (2024) existem dois tipos principais de requisitos, os requisitos funcionais e não funcionais.

GARCIA (2024), "Requisitos funcionais descrevem o que o sistema deve fazer, detalhando suas funções e processos."

GARCIA (2024), “já os requisitos não funcionais especificam como o sistema deve ser em relação ao desempenho, segurança, usabilidade, e outros aspectos que definem a qualidade do software.”

- Levantamento de Requisitos: O levantamento de requisitos consiste na prática de coletar as informações sobre as necessidades dos clientes. Podemos coletar essas informações através de questionários, sessões de brainstorming, observação, entre outros.
- Análise de Requisitos: Todos requisitos coletados são então analisados para definir quais são ou não viáveis ou possíveis a serem realizados. Também podem ser alterados e categorizados em lista de prioridade.
- Validação: Nessa etapa, o software é submetido a uma série de teste rigorosos para assegurar que todas as funcionalidades implementadas atendam aos requisitos definidos durante o levantamento e análise.
- Gerenciamento de Requisitos: Essa etapa acompanha o projeto desde o seu início até o seu final. Inclui o rastreamento constante dos requisitos, assegurando que sejam completamente atendidos durante o desenvolvimento, o controle das mudanças nos requisitos à medida que o projeto avança e a comunicação com o cliente.

3.10.2 Testes

O processo de teste de software de acordo com ALBUQUERQUE (2024), *O teste é composto por diversas atividades: Planejamento e Controle, Análise e Modelagem, Implementação e Execução, Avaliação dos critérios de Saída e Relatórios, e por fim as Atividades de Encerramento do teste.*

- Planejamento e controle: Aqui, são determinadas quais abordagens a serem utilizadas nos testes.
- Controle: No controle é verificado se o progresso feito no projeto está alinhado com o que foi definido no planejamento.
- Análise e Modelagem do teste: Definido o que será testado e de que forma esses testes serão conduzidos.
- Implementação e execução: Tem como foco colocar em prática os planos elaborados anteriormente para testar o software. Isso envolve

transformar condições teóricas de teste em casos práticos que serão executados.

- Avaliação dos critérios de Saída e Relatórios: Verifica se os objetivos alcançados nas últimas etapas foram alcançados. Também, nessa etapa é decidido se são necessários mais testes, ajustes nos critérios de saída ou mudanças no plano de teste.
- Encerramento do teste: Na última fase, ocorrem atividades de encerramento, onde são consolidadas as informações e experiências adquiridas ao longo do projeto.

4 METODOLOGIA

Como descrito no Capítulo 1 deste relatório, o estagiário tem como objetivo a criação de todo Front-End da plataforma UniConnect.

Para alcançar o resultado esperado, ele está realizando as seguintes tarefas: Criação da estrutura de navegação do site, incluindo menus, barra de pesquisa e links de navegação.

Desenvolvimento dos layouts das páginas principais, como página inicial, página de vagas, página de perfil, entre outros.

Implementação de formulários, como formulários de login, cadastro, candidatar-se a vagas, entre outros.

Integração com Api do Back-End para comunicação entre ambos.

Realização de testes de qualidade, incluindo testes de usabilidade, compatibilidade de navegadores e dispositivos.

4.1 Oportunidades de Melhoria

Mesmo o estágio sendo realizado com a própria universidade, o projeto a ser desenvolvido do zero não terá implementações importantes em seu desenvolvimento. Alguns fatores que colaboram para isso são a duração do tempo de estágio e a alta complexidade para implementação. Um exemplo que podemos citar é página de login: a universidade já possui sua própria página de login e, por este motivo, não seria necessário a criação de uma nova caso tivéssemos acesso à API dela. No entanto, como não temos esse acesso, é necessário a criação de uma nova UI. Além disso, por não ter acesso aos dados dos alunos da universidade, é necessário criar a página de registro de usuário, algo que não seria necessário, pois o projeto não teria autorização para cadastrar alunos na universidade, somente consultar dados.

5 CRONOGRAMA

Conforme apresentado na Tabela 2, foi elaborado um cronograma organizando as etapas a serem executadas durante o estágio para a sua conclusão.

Tabela 1 - Cronograma de atividades

ETAPA	ABR	MAI	JUN
Especificação dos Requisitos	X		
Design Front End e configuração do ambiente de desenvolvimento		X	X
Integração Front End com o Back End		X	X
Realização de testes		X	X
Implantação			X

Fonte: Autor

6 ATIVIDADES REALIZADAS

Neste capítulo será abordado o passo a passo das atividades realizadas no projeto, conforme descrito no capítulo 1.3.1.

6.1 Especificação dos requisitos

Antes de iniciar qualquer atividade, foram realizadas tarefas de engenharia de software para o levantamento de requisitos funcionais e não funcionais. Alguns dos requisitos levantados para o projeto incluem:

- a) Implementação de cadastro e login de usuários.
- b) Desenvolvimento de uma página para administração de novos pedidos de cadastro, permitindo que administradores revisem e gerenciem as solicitações na plataforma.
- c) Criação de um sistema para candidatura a vagas.
- d) Implementação de funcionalidades avançadas de pesquisa e filtragem de vagas.
- e) Integração com Back-End via APIs.

6.2 Design Front End e configuração do ambiente de desenvolvimento

Neste passo, será apresentado um passo a passo das atividades realizadas pelo estagiário no design Front End e na configuração do ambiente de desenvolvimento. Será fornecida uma breve explicação do que foi feito em cada etapa

e como essas atividades foram conduzidas para alcançar os objetivos estabelecidos no projeto.

6.2.1 Criação do projeto e organização das pastas

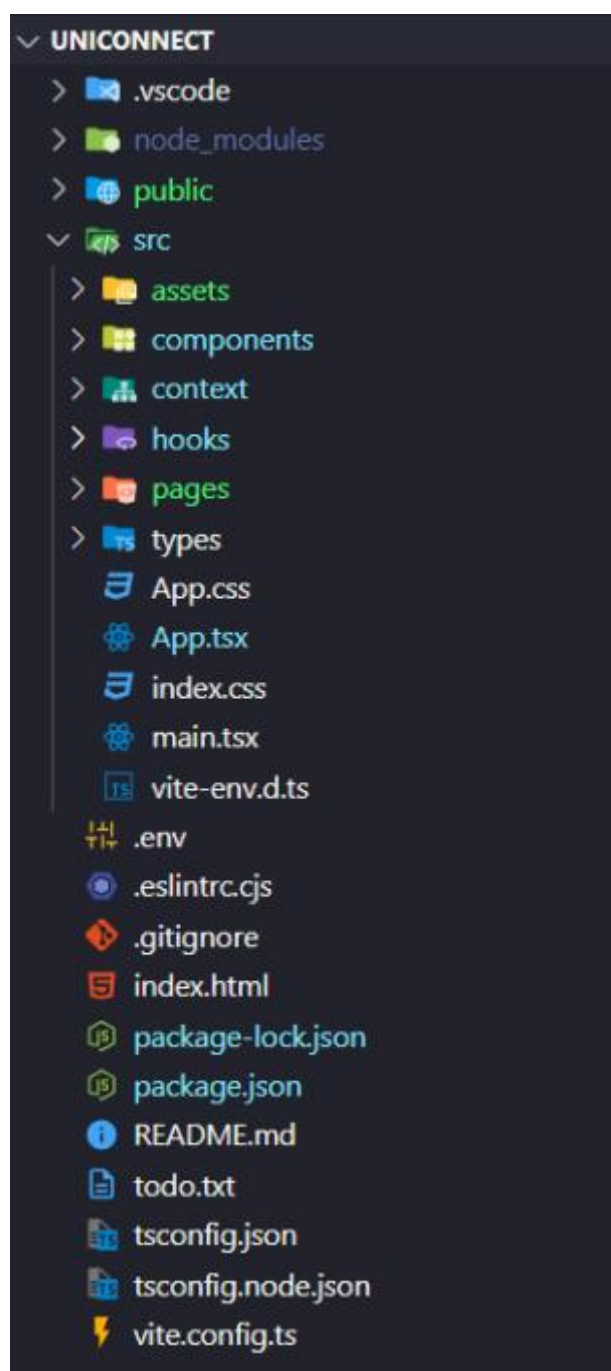
Na criação de um projeto Vite, já recebemos uma estrutura pronta com alguns códigos dentro, no entanto, muitas dessas pastas e códigos não são necessários e podemos removê-las.

Aproveitando a remoção dos itens desnecessários, já foram criadas outras no lugar que serão de uso futuro, deixando toda organização de pastas do projeto pronta.

A organização das pastas ficou o seguinte:

- a) Assets: Imagens.
- b) Components: Componentes.
- c) Context/auth: Autenticações.
- d) Hooks: Chamadas para API.
- e) Pages: Páginas.
- f) Type: Tipos de dados.

A organização das pastas é ilustrada na Figura 2.

Figura 2 - Organização pastas

Fonte: Autor

6.2.2 Criação de rotas

Após a organização das pastas era hora de criar toda lógica de navegação do site. Para isso foi utilizado o BrowserRouter importado do react-router-dom. O componente BrowserRouter do React Router é uma biblioteca popular para roteamento em aplicações React. O React Router permite que você defina diferentes

rotas para diferentes componentes em sua aplicação, permitindo a navegação entre esses componentes sem recarregar a página inteira. Resumindo, é um componente que envolve sua aplicação e habilita o roteamento, permitindo que você mapeie URLs para componentes específicos e crie uma experiência de navegação dinâmica em seu aplicativo.

A Figura 3 ilustra o Browser Router envolvendo o componente <App/> e a Figura 4 ilustra as URLs mapeadas dentro do <App/>.

Figura 3 - Utilização BrowserRouter

```
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <AuthProvider>
      <BrowserRouter>
        <App />
      </BrowserRouter>
    </AuthProvider>
  </React.StrictMode>,
)
```

Fonte: Autor

Figura 4 - Mapeamento URLs

```
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/UniConnect/Works" element={<RequireAuth><Works /></RequireAuth>} />
  <Route path="/UniConnect/Login" element={<Login />} />
  <Route path="/UniConnect/Register" element={<Register />} />
  <Route path="/UniConnect/ForgotPassword" element={<ForgotPassword />} />
  <Route path="/UniConnect/Curriculum" element={<RequireAuth><Curriculum /></RequireAuth>} />
  <Route path="/UniConnect/JobDetails/:jobId" element={<JobDetails />} />
  <Route path="/UniConnect/Applications" element={<RequireAuth><Applications /></RequireAuth>} />
  {/* Rota padrão para a página inicial */}
  <Route path="*" element={<Home />} />
</Routes>
```

Fonte: Autor

6.2.3 Criação AuthContext, AuthProvider e UseApi

O AuthContext é responsável por declarar quais métodos e campos estarão disponíveis para o contexto de autenticação. É como definir o que estará disponível para utilizar quando precisar realizar operações de autenticação.

A Figura 5 ilustra sua aplicação.

Figura 5 – AuthContext

```
export type AuthContextType = {
  user: User | null;
  signin: (email: string, password: string) => Promise<boolean>;
  signout: () => void;
  register: (name: string, email: string, password: string, phone: string, address: string, city: string, state: number) => Promise<boolean>;
  uploadPDF: (PDF: File) => Promise<boolean>;
  getPDF: () => Promise<Blob | null>;
  jobs: Job[] | null;
  getJobsList: () => Promise<Job[] | null>;
  userJobs: Job[] | null;
  getUserJobsList: () => Promise<Job[] | null>;
  addJob: (title: string, workStyle: WorkStyle, employmentType: EmploymentType, description: string, promoter: string, salary: number) => Promise<boolean>;
  subscribeToJob: (id: number) => void;
  removeJob: (id: number) => Promise<boolean>;
  //apiStates
  state: State[] | null;
  getStates: () => Promise<boolean>;
};

export const AuthContext = createContext<AuthContextType>(null!); //so preenche contexto com informações no provider
```

Fonte: Autor

Já o AuthProvider, conforme ilustrado na Figura 6, é um componente onde são definidas as implementações concretas de cada método declarado no AuthContext. Ele especifica o comportamento de cada método, como realizar a autenticação, registro, obtenção de dados do usuário etc.

Figura 6 - Exemplo implementação AuthProvider

```
const getUserJobsList = async (): Promise<Job[] | null> => {
  try {
    const data = await api.getUserJobsList(token);
    if (data.jobs) {
      console.log(data.jobs);
      setUserJobs(data.jobs);
      return data.jobs;
    }
    return null;
  } catch (error) {
    console.error("Erro ao obter a lista de trabalhos do usuário:", error);
    return null;
  }
}
```

Fonte: Autor

O UseApi, conforme ilustra a Figura 7, tem a finalidade de lidar com as chamadas ao Back End. É utilizado dentro do AuthProvider para realizar as chamadas à API e obter os dados necessários para as operações de autenticação. O UseApi serve como ponte entre o Front End e o Back End.

Figura 7 - Exemplo código na UseApi

```
export const useApi = () => ({
  // Métodos para a back-end API
  validateToken: async (token: string) => { //working
    // return {
    //   user: { id: 3, name: 'José', email: 'jose@gmail.com', role: 'ADMIN' }
    // }
    const headers = {
      Authorization: `${token}`,
      'Content-Type': 'application/json', // Especificando o tipo de conteúdo como JSON
    };
    const response = await api.get('/auth/userdata', { headers });
    return response.data;
  },
});
```

Fonte: Autor

6.2.4 Criação dos Types

Até o final do cronograma foram criados 3 types:

- a) Job: Ilustrado na Figura 8, representa um trabalho ou vaga de emprego, contendo informações como título, descrição, estilo de trabalho, tipo de emprego, entre outros detalhes.
- b) State: Ilustrado na Figura 9, representa um estado brasileiro com seu ID, nome e sigla.
- c) User: Ilustrado na Figura 10, define o usuário do sistema, incluindo informações como nome, e-mail, senha, telefone, endereço, cidade, estado, imagem de perfil e função.

Figura 8 - Type Job

```
import { State } from "../State";

export enum WorkStyle {
  HIBRIDO = "HIBRIDO",
  PRESENCIAL = "PRESENCIAL",
  REMOTO = "REMOTO"
}

export enum EmploymentType {
  TRABALHO = "TRABALHO",
  ESTAGIO = "ESTAGIO"
}

export type Job = {
  id: number;
  title: string;
  text: string; //brief summary of the job
  workStyle: WorkStyle;
  employmentType: EmploymentType;
  description: string; //job description
  promoter: string;
  salary: string;
  city: string;
  state: State;
  date: Date;
}
```

Fonte: Autor

Figura 9 - Type State

```
export type State = {
  id: number;
  nome: string;
  sigla: string;
};
```

Fonte: Autor

Figura 10 - Type User

```
import { State } from "./State";

export enum role {
  ADMIN = "ADMIN",
  USER = "USER"
}

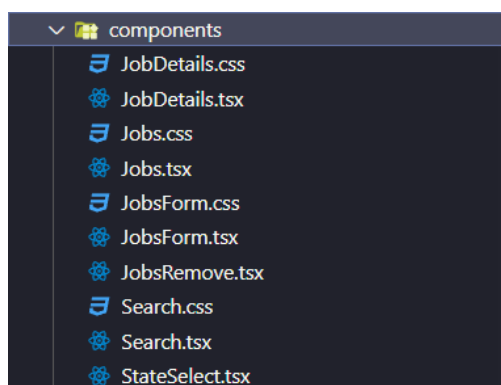
export type User = {
  id: number;
  name: string;
  email: string;
  password?: string;
  phone: string;
  address: string;
  city: string;
  state: State;
  profilePicture: Uint8Array | null;
  role: role;
}
```

Fonte: Autor

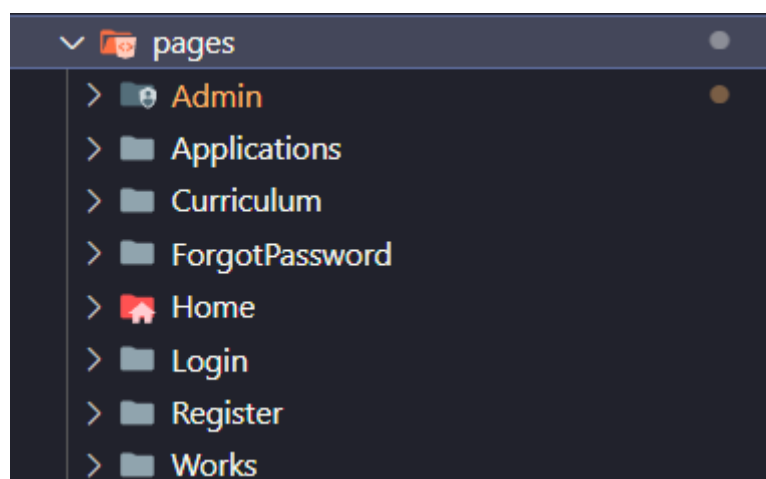
6.2.5 Criação das principais páginas e componentes

Na pasta componentes, serão armazenados componentes reutilizáveis da interface do usuário, criação, remoção e visualização de um Trabalho, filtro de pesquisa, entre outros. Pasta components ilustrada na Figura 11.

Na pasta pages, contém páginas principais da aplicação, cada uma organizada em seu próprio arquivo para facilitar navegação e manutenção. Pasta pages ilustrada na Figura 12.

Figura 11 - Componentes

Fonte: Autor

Figura 12 - Principais páginas

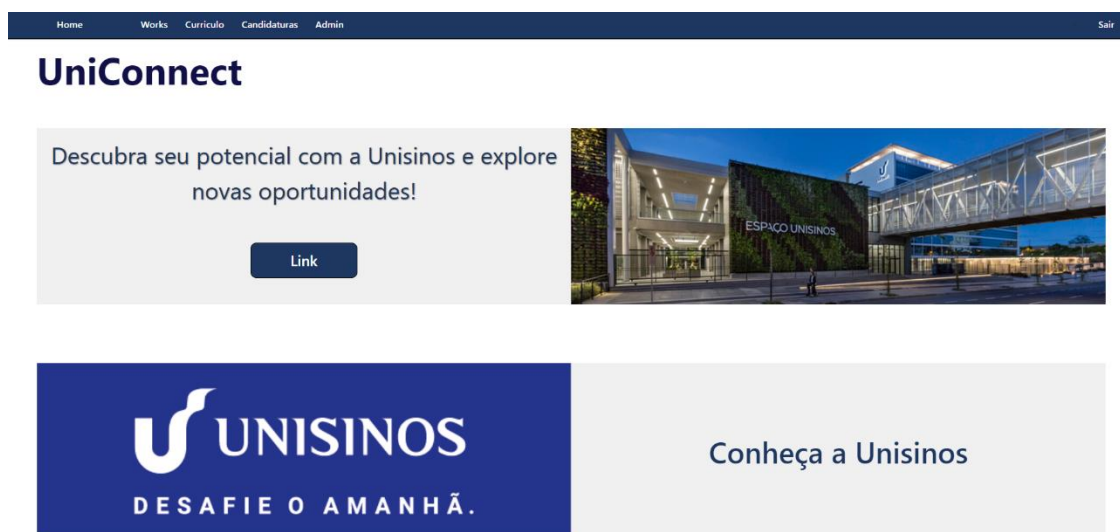
Fonte: Autor

6.2.6 Funcionamento das páginas

O componente App é o principal e envolve toda a aplicação. Ele possui a estrutura básica de interface e gerencia o estado principal da aplicação. Quando o usuário decide navegar da página Home para Candidaturas, ocorre a troca de conteúdo dentro do corpo (body) do App.

Conforme mostra a Figura 13, a página Home contém dois Links, um ainda sem destino, e outro para a página da Unisinos.

Figura 13 - Página Home

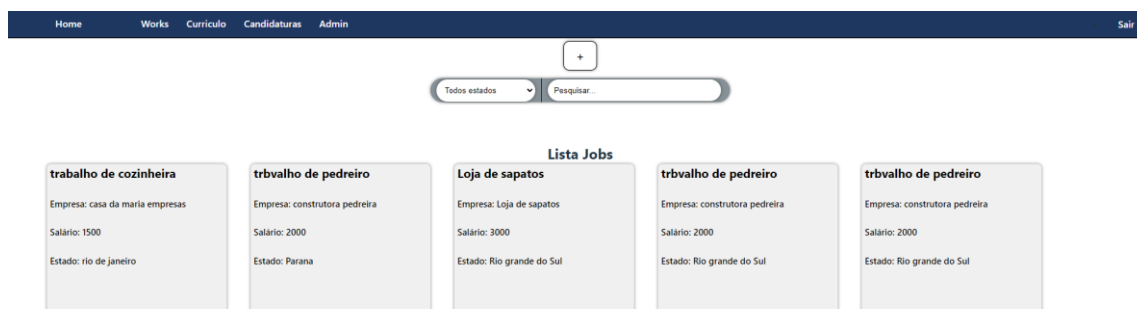


Fonte: Autor

Na página Works, conforme mostrado na figura 14, temos 3 componentes sendo exibidos, o componente com o símbolo + redireciona o usuário para o JobsForm, onde é possível criar um Job. Abaixo temos o componente para filtro de pesquisa de Jobs, podendo escolher filtrar o Job por estado, título ou ambos.

Por fim, temos o componente Jobs, que mostra uma lista com todos Jobs disponíveis, quando clicado em um, redireciona para o componente JobDetails, que mostra todos seus detalhes.

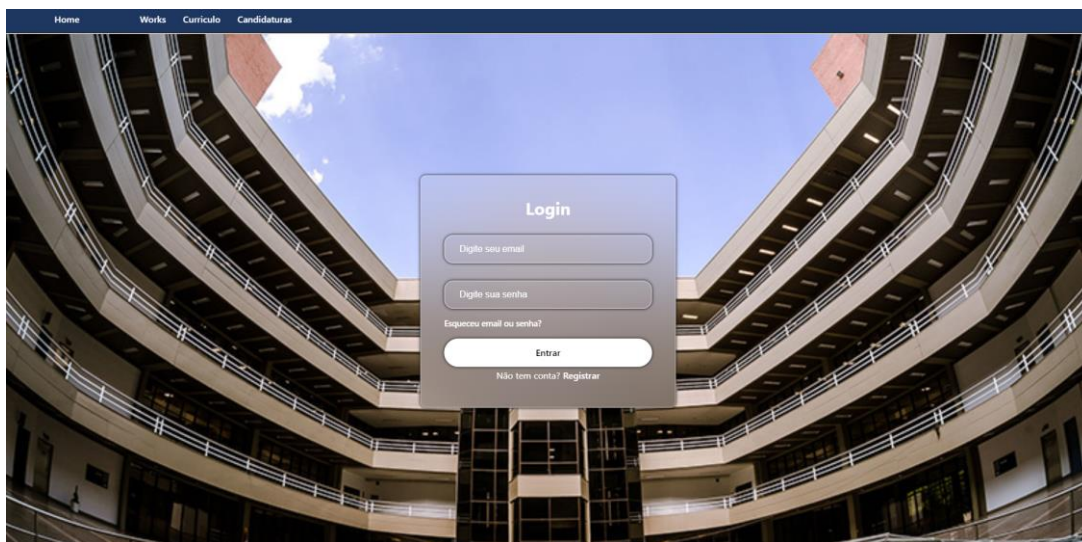
Figura 14 - Página Works



Fonte: Autor

A página Login pode ser acessada ao usuário tentar acessar o seu currículo ou os Jobs sem estar logado no sistema, conforme mostra a Figura 14. O usuário deve informar seu e-mail de login e sua senha. Caso não lembre dos dados pode utilizar a opção de 'Esqueceu e-mail ou senha?'. Se o usuário não tiver uma conta, pode clicar em se registrar para ser redirecionado a página de registro.

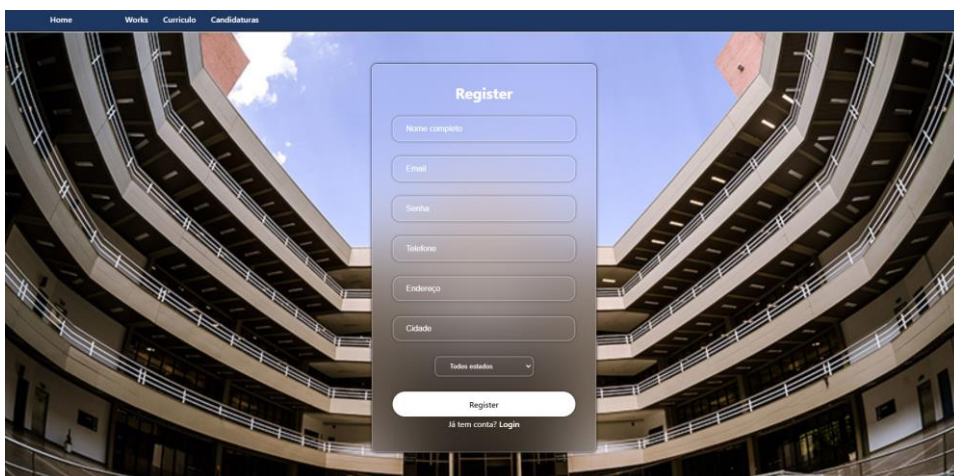
Figura 15 - Página Login



Fonte: Autor

Na página de registro, o usuário deve preencher todas as informações e então aguardar um admin para que aceite seu pedido. Após aceito ele pode voltar a tela de Login e realizar o Login para acessar o sistema. A Figura 16 mostra a página de Registrar.

Figura 16 - Página Registrar



Fonte: Autor

A página Currículo que ainda não foi finalizada, conforme apresenta a Figura 15, possui informações do Usuário. Na opção 'escolher Arquivo', o usuário pode selecionar um PDF com seu currículo e enviar ao Back End. Podendo visualizar esse PDF nessa página Currículo sempre que desejar, assim como também alterar o arquivo ou o excluir.

Figura 17 - Página Currículo



Fonte: Autor

A página Candidaturas possui a finalidade de mostrar ao usuário em quais Jobs ele se candidatou, conforme ilustrado na Figura 16, o usuário se candidatou ao Job com o título teste. Caso o usuário queira mais informações sobre esse Job, basta clicar sobre o título e assim será apresentado o componente JobDetails.

Figura 18 - Página Candidaturas



Fonte: Autor

A página Admin mostra todos os usuários que realizaram o cadastro e esperam a aprovação de um Admin.

Conforme mostra na Figura 17, o nome do usuário é apresentado a esquerda. A direita, o admin escolhe entre a opção de aceitar o usuário como USER ou ADMIN. Após isso ele escolhe se quer Aprovar ou Negar a solicitação de entrada do usuário.

Figura 19 - Página Admin

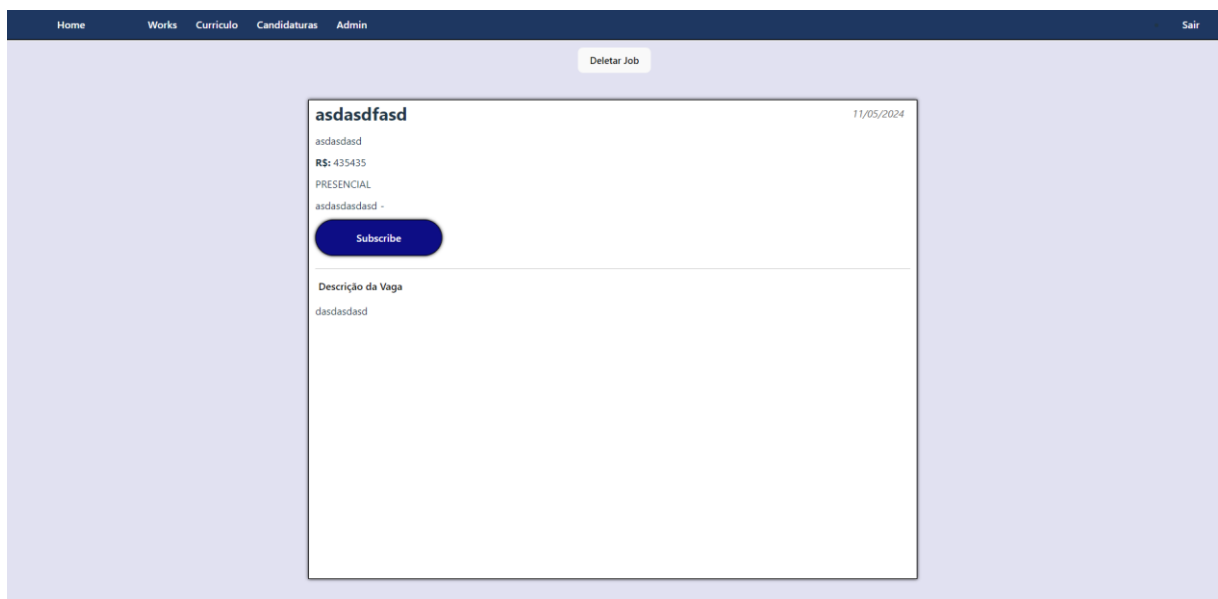


Fonte: Autor

6.2.7 Funcionamento dos componentes

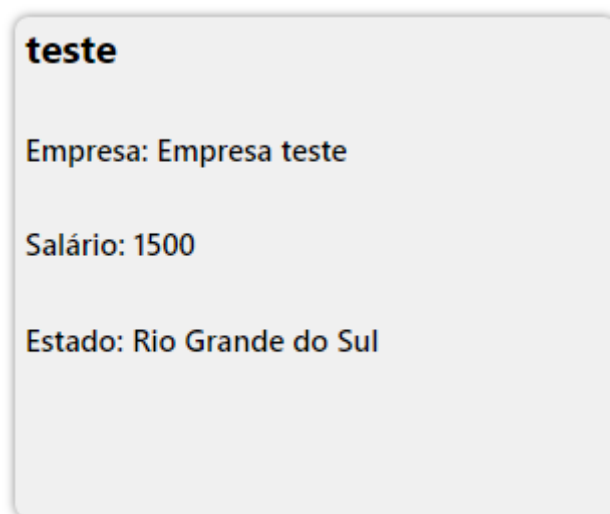
O componente JobDetails, conforme mostra a Figura 18, mostra todas as informações de um Job. Dentro desse componente temos 2 botões. O botão “Deletar Job” chama o componente JobbRemove, também ilustrado na Figura 18, e deleta esse job do banco de dados. O segundo botão na tela chamado “Subscribe” adiciona esse Job na página Candidaturas, onde o usuário pode visualizar todos Jobs em que está inscrito.

Figura 20 - Componente JobDetails e JobsRemove



Fonte: Autor

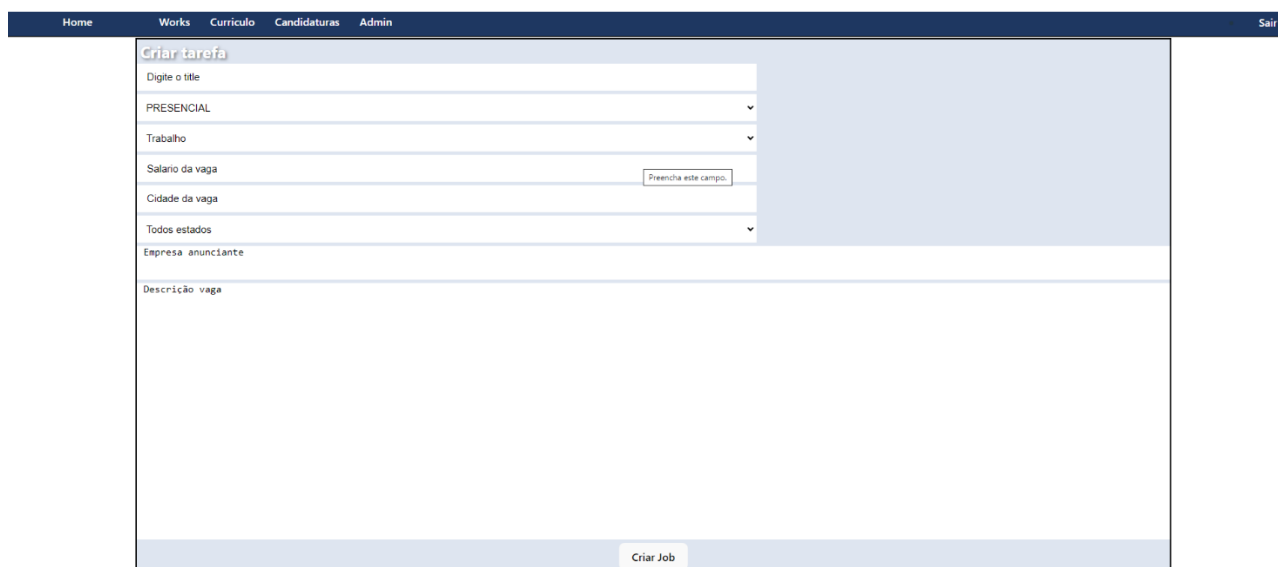
O componente Jobs, conforme mostra a Figura 19 apresenta um resumo de cada Job e fica na página Works. Ele é um componente clicável que redireciona o usuário para o JobDetails.

Figura 21 - Componente JobsA screenshot of a light gray rounded rectangle representing a job card. It contains the following text: 'teste' in bold at the top, followed by 'Empresa: Empresa teste', 'Salário: 1500', and 'Estado: Rio Grande do Sul'.

Fonte: Autor

O componente JobsForm, conforme mostra a Figura 20, nos possibilita a criação de um Job, sendo necessário o preenchimento de cada um dos campos e então clicar no botão Criar Job.

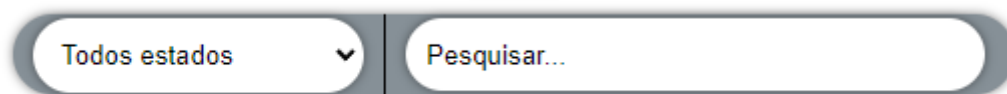
Todas as informações inseridas serão apresentadas no componente JobDetails e no componente Jobs.

Figura 22 - Componente JobsFormA screenshot of a web application interface. At the top is a dark blue navigation bar with links: 'Home', 'Works', 'Curriculo', 'Candidaturas', 'Admin', and a 'Sair' button on the right. Below the navigation bar is a form titled 'Criar tarefa'. The form has several input fields: 'Digite o title', a dropdown menu with 'PRESENCIAL' selected, a dropdown menu with 'Trabalho' selected, a text field for 'Salario da vaga' with a 'Preencha este campo.' button next to it, a text field for 'Cidade da vaga', a dropdown menu with 'Todos estados' selected, and a text field for 'Empresa anunciante'. Below these fields is a large text area for 'Descrição vaga'. At the bottom of the form is a 'Criar Job' button.

Fonte: Autor

O componente Search é utilizado para filtrar Jobs por título ou por estados, conforme apresentado na Figura 21.

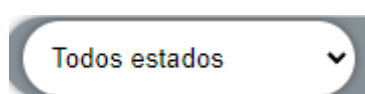
Figura 23 - Componente Search



Fonte: Autor

Por último o componente StateSelect, ilustrado na Figura 22, nos dá a opção de selecionar um estado brasileiro.

Figura 24 - Componente StateSelect



Fonte: Autor

6.2.8 CSS e responsividade

O estilo da aplicação foi todo desenvolvido em CSS, utilizando um modelo de design responsivo para garantir uma boa experiência do usuário em diferentes dispositivos e tamanhos de telas. No entanto, devido ao fator tempo, alguns não puderam ser finalizados ou iniciados. Também, o estilo aplicado é um modelo que serve como base, pois muitas das aplicações que podem vir a serem criadas ou modificadas podem levar ou exigir alterações.

6.3 Integração Front End com o Back End

Para a integração, foram utilizadas instâncias do Axios configuradas com variáveis de ambientes. O Axios é uma biblioteca JavaScript utilizada para requisições HTTP a servidores, simplificando o processo de comunicação entre o Front End e o Back End. Utilizando o Axios podemos realizar diferentes tipos de requisições, como GET, POST, PUT, DELETE etc. Até o momento desse trabalho somente o GET e POST foram usados.

Conforme ilustra a Figura 13, é utilizado o Axios para criar instâncias do Axios com configurações específicas. São, então, definidos diferentes URLs para cada um através do baseUrl. O baseUrl é um prefixo de URL que será automaticamente pré-

anexado a todas as requisições feitas com essa instância do Axios, não sendo então necessário repetir esse trecho da URL em cada chamada.

Caso seja necessário alterar a URL, basta alterar uma vez onde ela está definida, e assim todas chamadas utilizarão o novo valor. As URLs no projeto estão definidas na pasta .env, que serve somente para esse propósito.

Como mostra na Figura 13, temos 2 instâncias criadas, uma para o servidor Back End e outra para consultar todos os estados brasileiros e suas informações.

Figura 25 - Uso do Axios

```
const api = axios.create({
  baseURL: import.meta.env.VITE_API
});

const apiStates = axios.create({
  baseURL: import.meta.env.VITE_APISTATE,
});
```

Fonte: Autor

6.4 Realização de Testes e Implantação

Não foi possível realizar as etapas de testes e implantação do projeto devido a inúmeros fatores, destacando entre eles as chuvas ocorrentes no estado do Rio Grande do Sul durante o período de estágio, que afetaram o cronograma estabelecido. Algumas funcionalidades também não puderam ser construídas, entre elas a funcionalidades para o usuário recuperar sua senha e visualização do PDF enviado ao Back-End.

7 CONSIDERAÇÕES FINAIS

Apesar de não realizados todos os objetivos especificados no cronograma, o estagiário pode afirmar que obteve grande aprendizado durante o período. Além de colocar em prática conhecimentos já aprendidos em sala de aula, também adquiriu um vasto conhecimento em novas tecnologias que não possuía experiência anteriormente.

Alguns destaques de aprendizados do estagiário que podem ser citados:

- I. Adquirir habilidades em ferramentas e tecnologias mais utilizadas no mercado.
- II. Entender o passo a passo necessário para a criação de uma aplicação de Front End utilizando as tecnologias aprendidas.
- III. Experiência em gestão de prazos e demandas.
- IV. Trabalho colaborativo com Back End.
- V. Desenvolvimento de autonomia, aprendendo a trabalhar de forma mais independente, assumindo responsabilidades e tomando decisões sob supervisão.

A experiência proporcionada pelos desafios do ambiente corporativo, permitiu um crescimento tanto pessoal quanto profissional ao estagiário, podendo este concluir que o projeto contribuiu significativamente para sua formação e preparação para desafios futuros na área.

REFERÊNCIAS

ALBUQUERQUE, Gustavo. **O processo de testes de software simplificado (OneDayTesting)**. 2024. Disponível em: <https://blog.onedaytesting.com.br/o-processo-de-testes-de-software-simplificado/>. Acesso em: 26/06/2024

ATENCIO, L. **Functional Programming in JavaScript**. Sebastopol: O'Reilly Media, 2016.

CROCKFORD, Douglas. **JavaScript: The Good.Parts.1.** ed. Sebastopol: O'ReillyMedia,2008.

DUCKETT, Jon. **HTML and CSS: Design and Build Websites**. Indianapolis: Wiley, 2011.

FENTON, Steve. **Pro TypeScript: Application-Scale JavaScript Development**. Apress, 2018.

FLANAGAN, David. **JavaScript: The Definitive Guide**. 7th ed. Sebastopol: O'Reilly Media, 2020.

GARCIA, Guilherme. **Engenharia de Requisitos: O que é, Como Funciona e Tipos (mercadoonlinedigital.com)**. 2024. Disponível em: <https://mercadoonlinedigital.com/blog/engenharia-de-requisitos/>. Acesso em: 18/06/2024

MOZILLA. **Web JavaScript**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/conflicting/Web/JavaScript>. Acesso em: 20 abril 2024.

OSMANI, A. **Learning JavaScript Design Patterns**. Sebastopol: O'Reilly Media, 2012.

SIMPSON, Kyle. **Functional - Light JavaScript**. Createspace Independent Pub, 2017.

SRIPARASA, S. S. **JavaScript and JSON Essentials**. Birmingham: Pack Publishing, 2013.

State of JS. **Introdução**. Disponível em: <https://2018.stateofjs.com/introduction/>. Acesso em 21 abril 2024.

UNISINOS. Universidade do Vale do Rio dos Sinos, Disponível em: <https://www.unisinos.br/>. Acesso em: 24 abril 2024.

Vite. **Why Vite?**. Disponível em: <https://vitejs.dev/guide/why.html>. Acesso em 23 abril 2024.