FRONT-END DO SISTEMA PARA DIVISÃO DE ESTÁGIO DA ESCOLA POLITÉCNICA DE PERNAMBUCO

Trabalho de Conclusão de Curso

Engenharia da Computação

Nome do Aluno: Matheus Hernandes Andreoni
Orientador: Prof. Joabe Bezerra Jesus Júnior

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

MATHEUS HERNANDES ANDREONI

FRONT-END DO SISTEMA PARA A DIVISÃO DE ESTÁGIO DA ESCOLA POLITÉCNICA DE PERNAMBUCO

Aplicativo apresentado como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Orientador: Prof. Joabe Bezerra de Jesus Júnior

Recife
Outubro de 2022.

Andreoni, Matheus Hernandes
Front-End do sistema para a divisão de estágio da Escola Politécnica de Pernambuco / Matheus Hernandes Andreoni. Recife - PE, 2022.
xi, 44 f. : 16 il. ; 29 cm.

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) Universidade de Pernambuco, Escola Politécnica de Pernambuco, Recife, 2022.

Orientador: Prof°. Joabe Bezerra de Jesus Júnior.

Inclui referências.

1. Desenvolvimento de Software. 2. Frontend. 3. React. I. *Front-End* do sistema para a divisão de estágio da Escola Politécnica de Pernambuco. II. Júnior, Joabe Bezerra de Jesus . III. Universidade de Pernambuco.

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 21/10/2022, às 10h00min, reuniu-se para deliberar sobre a defesa da monografia de conclusão de curso do(a) discente **MATHEUS HERNANDES ANDREONI**, orientado(a) pelo(a) professor(a) **JOABE BEZERRA DE JESUS JÚNIOR**, sob título FRONT-END DO SISTEMA PARA DIVISÃO DE ESTÁGIO DA ESCOLA POLITÉCNICA DE PERNAMBUCO, a banca composta pelos professores:

HENRIQUE ALVES DINARTE DA SILVA (PRESIDENTE)
JOABE BEZERRA DE JESUS JÚNIOR (ORIENTADOR)

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

☒ Aprovada ☐ Aprovada com Restrições* ☐ Reprovada

e foi-lhe atribuída nota: 9,5 (nove e meio)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O(A) discente terá 7 dias para entrega da versão final da monografia a contar da data deste documento.

gov.br Documento assinado digitalmente
HENRIQUE ALVES DINARTE DA SILVA
Data: 26/10/2022 12:15:33-0300
Verifique em https://verificador.iti.br

AVALIADOR 1: Prof (a) **HENRIQUE ALVES DINARTE DA SILVA**

Joabe Bezerra de Jesus Júnior

AVALIADOR 2: Prof (a) **JOABE BEZERRA DE JESUS JÚNIOR**

AVALIADOR 3: Prof (a)

* Este documento deverá ser encadernado juntamente com a monografia em versão final.

ESCOLA POLITÉCNICA
DE PERNAMBUCO

Autorização de publicação de PFC

Eu, **Matheus Hernandes Andreoni** autor(a) do projeto de final de curso intitulado: **FRONT-END DO SISTEMA PARA DIVISÃO DE ESTÁGIO DA ESCOLA POLITÉCNICA DE PERNAMBUCO;** autorizo a publicação de seu conteúdo na internet nos portais da Escola Politécnica de Pernambuco e Universidade de Pernambuco.

O conteúdo do projeto de final de curso é de responsabilidade do autor.

Matheus H. Andreoni

Matheus Hernandes Andreoni

Joabe Bezerra de Jesus Júnior

Orientador(a): **Joabe Bezerra de Jesus Júnior**

Coorientador(a):

Prof, de TCC: **Daniel Augusto Ribeiro Chaves**

Data: 21/10/2022

Resumo

O estágio é o primeiro contato do estudante com o mercado de trabalho dentro da sua área de atuação escolhida. Essa experiência é uma importante aliada ao curso de graduação, auxiliando no processo de desenvolvimento e aprendizagem do aluno. Nesse cenário, a Escola Politécnica de Pernambuco, unidade mais antiga da Universidade de Pernambuco, possui a Divisão de Estágio, um setor responsável por cuidar de todos os processos relacionados aos estágios dos alunos, sejam eles obrigatórios ou não. Entretanto, não existe um sistema, um software que centralize e auxilie no desenvolvimento destes processos atualmente. Foram utilizados o *React*, *Redux* e *Material UI* para a construção do frontend do sistema, visando a usabilidade do mesmo. O trabalho demonstra e conclui como o uso de softwares podem auxiliar e centralizar o desenvolvimento de processos (entre eles a autorização, autenticação e fluxos de requerimento de estágio), sugerindo como trabalhos futuros a finalização dos outros fluxos do sistema.

Palavras-chave: Desenvolvimento de Software, Front-end, React, Redux, Material UI, Estágio.

Abstract

The internship is the student's first contact with the job market within their chosen area of expertise. This experience is an important ally to the graduation course, helping in the student's development and learning process. In this scenario, the Escola Politécnica de Pernambuco, the oldest unit of the University of Pernambuco, has the Internship Division, a sector responsible for taking care of all processes related to student internships, whether mandatory or not. However, there is no system or software that centralizes and assists in the development of these processes today. *React*, *Redux* and *Material UI* were used to build the system's frontend, aiming at its usability. The work demonstrates and concludes how the use of software can help and centralize the development of processes (including authorization, authentication and internship requirement flows), suggesting as future works the completion of the other flows of the system.

Keywords: Software Development, Front-end, React, Redux, Material UI, Internship.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
BLOB	*Binary Large Object*
CSS	*Cascade Style Sheet*
DTO	*Data Transfer Object*
HTTP	*Hypertext Transfer Protocol*
JSON	*Javascript Object Notation*
JWT	*JSON Web Token*
SPA	*Single Page Application*
UI	*User Interface*
URL	*Uniform Resource Locator*
UX	*User Experience*

Sumário

Capítulo 1

Sistema de Estágio da POLI-UPE

O trabalho em questão se refere a um sistema criado para a Divisão de Estágio da Escola Politécnica de Pernambuco. Este sistema tem como finalidade centralizar os processos existentes dentro da universidade e facilitar a forma que estes processos são executados. Para este trabalho, foram feitas as funcionalidades de autenticação, autorização e de requerimento de estágio. Funcionalidades além deste escopo como criação de relatórios, vínculos com empresas ou avaliação de alunos, não foram construídas neste projeto.

O sistema foi desenvolvido utilizando as bibliotecas *React* [1] e *Redux* [2], para a criação de componentes e gerenciamento do estado da aplicação, respectivamente. Além disso, foi utilizado o *Material UI* [3], uma biblioteca de componentes *CSS* baseadas no *Material Design* [4], visando um melhor *UX* e *UI*. Para uma melhor tipagem do projeto, foi utilizado o *TypeScript*, um superconjunto sintático estrito de *JavaScript* que adiciona tipagem estática opcional à linguagem.

1.1 Autenticação

A autenticação do sistema foi feita de duas diferentes formas. Os alunos e os professores se autenticam no sistema utilizando o e-mail institucional da universidade, através de uma biblioteca do Google chamada *Google Identity Services* [5], que fornece toda a base para realizar este tipo de login. Já os administradores e funcionários da Divisão de Estágio se autenticam através de credenciais previamente cadastradas no sistema.

1.2 Autorização

A autorização foi feita através da criação de 3 perfis de usuário. Primeiramente tem-se o perfil de aluno, perfil este que foi definido como o perfil padrão no sistema. Além do aluno, tem-se o perfil de professor e o perfil de administrador, que representa os funcionários e administradores da Divisão de Estágio. O administrador é o responsável por determinar o perfil de cada usuário corretamente, utilizando a funcionalidade de edição de usuário.

1.3 Requerimento

A funcionalidade de requerimento envolve apenas 3 cenários básicos: criação, leitura e edição. A criação do requerimento é feita através do preenchimento de um formulário com os principais dados solicitados, tais como o professor, os dados da empresa e o upload dos documentos obrigatórios. Para o *upload* dos documentos, foi utilizado a interface *File* presente no Typescript. Finalizada a criação, o requerimento fica disponível para o professor escolhido e para a Divisão de Estágio, para que ambos realizem a aprovação do mesmo.

Conforme citado no parágrafo anterior, dentro da edição do requerimento está incluída a parte de aprovação e reprovação. Essa aprovação exige uma análise, tanto dos dados inseridos, quanto dos documentos enviados pelo aluno. Para a visualização dos dados foi disponibilizada uma tela com os detalhes e documentos do requerimento.

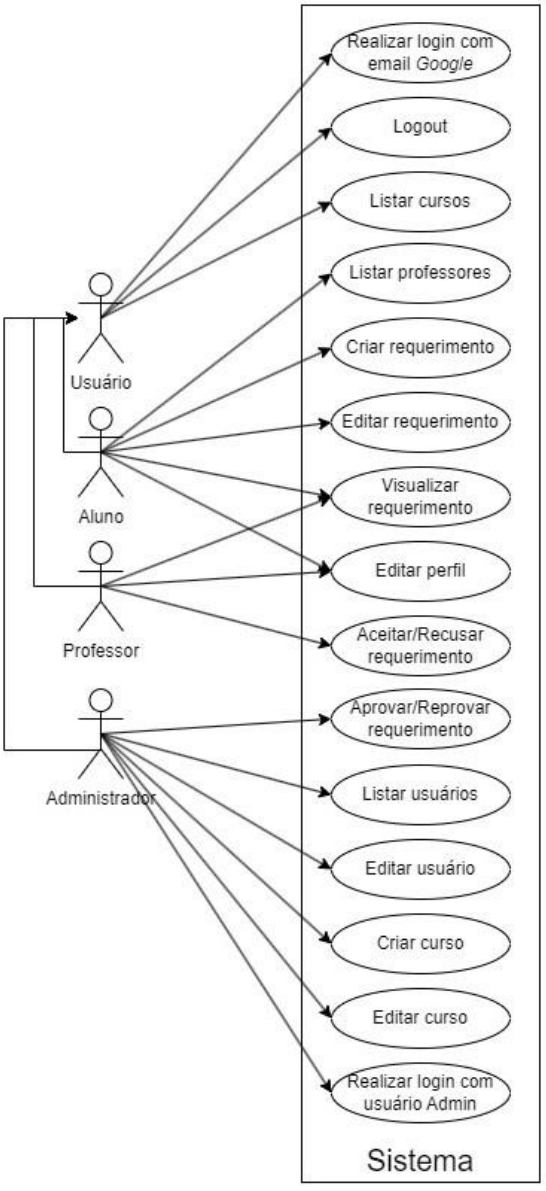
Caso haja algo errado no requerimento, tanto o professor quanto a Divisão de Estágio podem reprová-lo, inserindo um motivo de recusa. Com o requerimento recusado, o aluno poderá editá-lo e solicitar uma nova revisão para que o mesmo seja aprovado.

Capítulo 2

Modelagem dos Requisitos

Na Figura 1, observa-se um diagrama que representa os casos de uso de cada perfil dentro do sistema. Primeiramente temos os casos de uso dos perfis de aluno e professor, em seguida os casos de uso do Administrador.

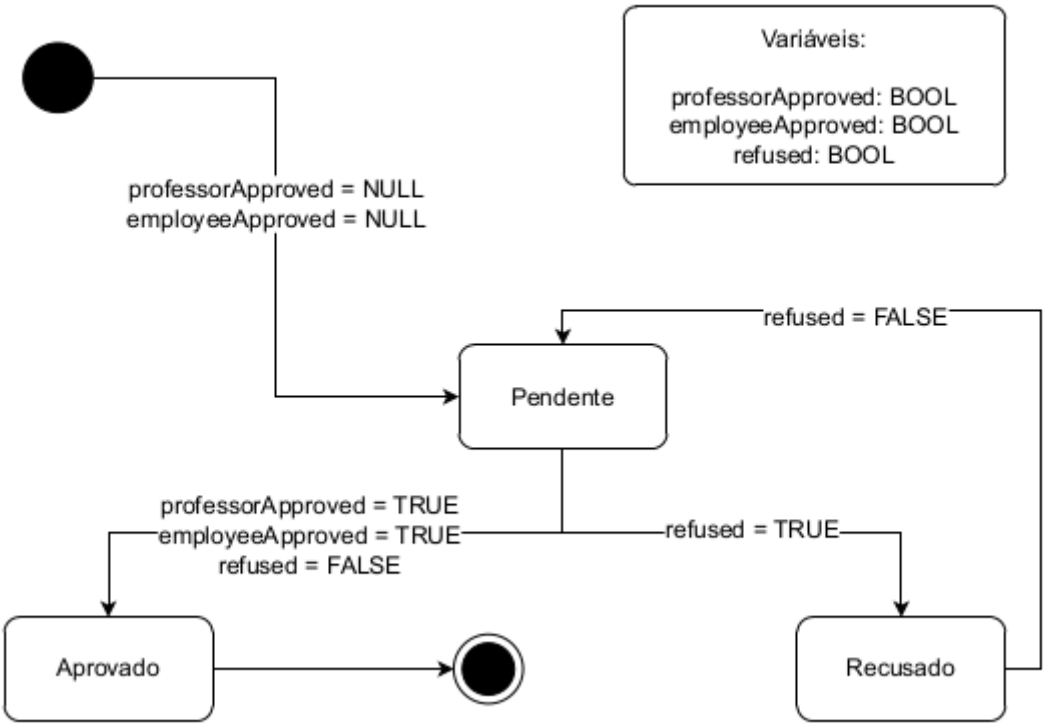
Figura 1 - Casos de uso do sistema.



Fonte: Autor / Renan Villarim

Além do diagrama de casos de uso exibido anteriormente, foi criado um diagrama que representa uma máquina de estados para os possíveis *status* de um requerimento, mostrando todos os possíveis cenários e ações no sistema que podem mudar a situação dessa entidade. Dessa forma, é possível observar na Figura 2 que o requerimento sempre se inicia no *status* de “Pendente”, podendo ser alterado para “Aprovado”, caso o professor e a Divisão de Estágio aprovem o requerimento, ou para “Recusado”, onde houve uma reprovação do requerimento por parte do professor ou da Divisão de Estágio. Após a edição do requerimento por parte do aluno e solicitada novamente a avaliação, o *status* retorna para “Pendente”.

Figura 2 - Máquina de estados de um requerimento.



Fonte: Autor / Renan Villarim

Capítulo 3

Arquitetura do Sistema de Estágio da POLI-UPE

Neste capítulo são apresentados a análise e o escopo do projeto que foram utilizados para definir o contexto do trabalho, a estruturação da SPA ilustrando o fluxo da arquitetura do *front-end*.

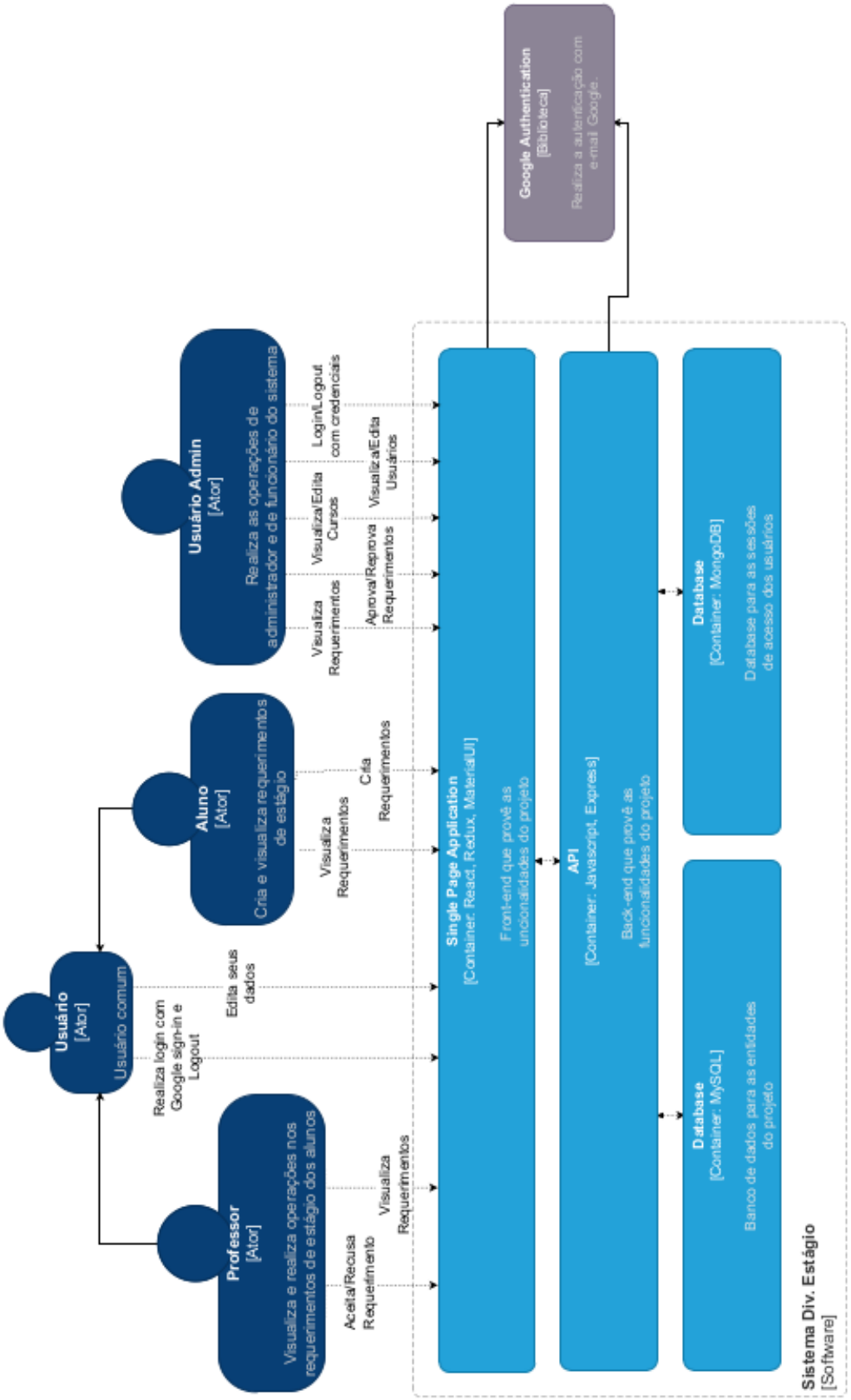
3.1 Arquitetura do Projeto

Conforme citado anteriormente, para a construção do sistema foram definidos três tipos de atores: Aluno, Professor e Administrador. A Figura 3 ilustra a arquitetura geral do projeto, onde os três atores interagem diretamente com a SPA que, por sua vez, comunica-se através de requisições HTTP com a API (*backend* construído em *JavaScript*, paralelamente a este projeto), utilizando o JSON como principal forma de comunicação entre cliente e servidor. Em seguida, ao receber as solicitações da SPA, a API realiza consultas, derivadas das requisições enviadas pelo SPA, ao *database*. Por fim, o *database* retorna as informações resultantes da consulta, que regressam às entidades da arquitetura até chegarem à interface do usuário. Além do *database* construído em MySQL (sistema de gerenciamento de banco de dados, que utiliza a linguagem SQL como interface), foi criado um *database* em MongoDB (programa de banco de dados NoSQL, ou seja, não relacional) com a função de armazenar as sessões em andamento dos usuários autenticados no sistema.

O controle da autenticação por parte dos alunos e professores se dá através dos *cookies* da sessão em andamento no sistema. Já no acesso dos funcionários e administradores o controle da sessão é feito através de *tokens JWT.*

Na parte dos documentos, mais precisamente para se obter a visualização dos mesmos, foi utilizado o *BLOB*, facilitando a manipulação do conteúdo dos arquivos provenientes do servidor. Com o *BLOB* é possível se criar uma *URL* que abre o arquivo registrado no banco de dados do servidor em um *browser*.

Figura 3 - Arquitetura do sistema.

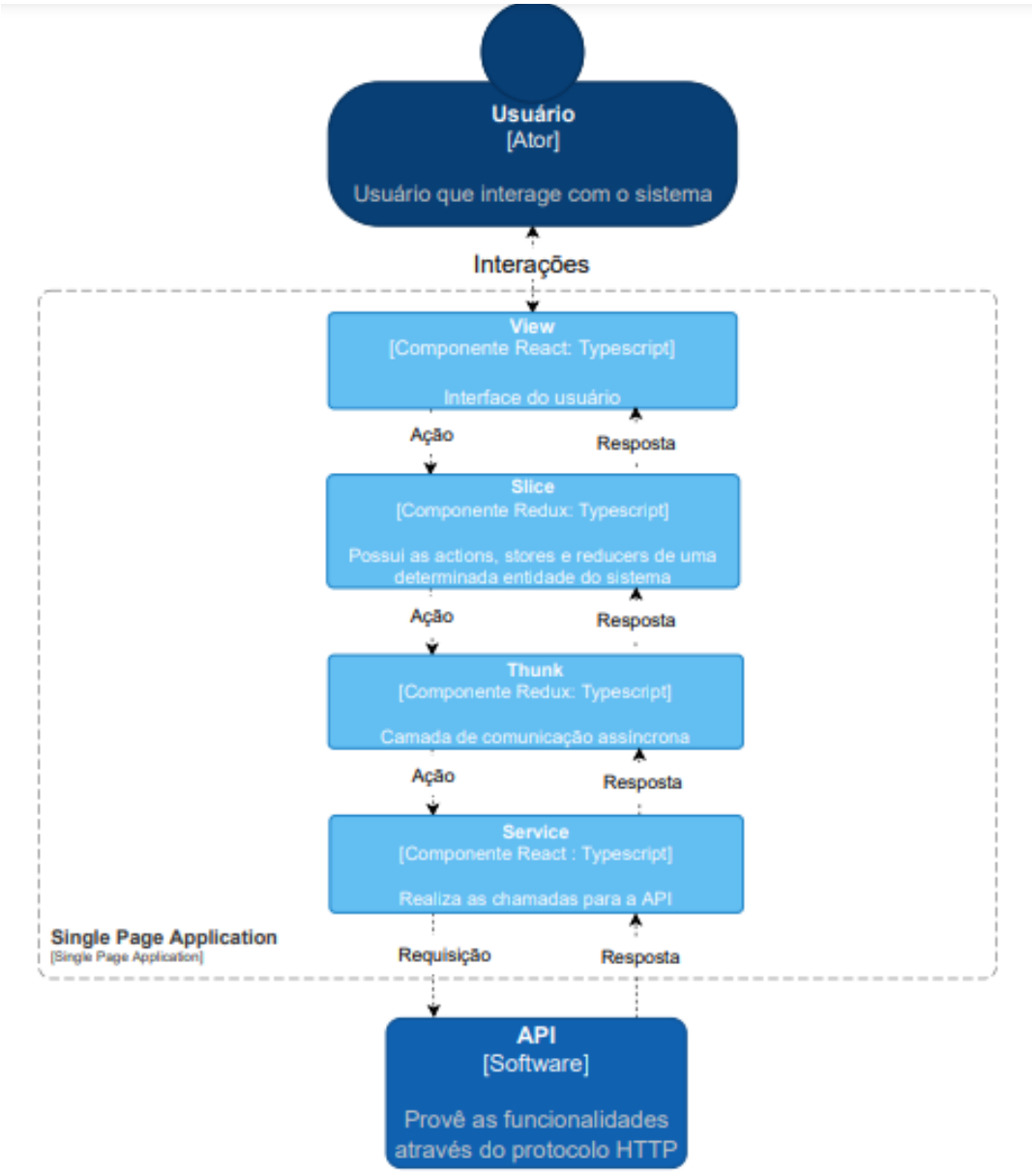


Fonte: Autor / Renan Villarim

3.2 Arquitetura da SPA

A estruturação da SPA foi baseada na arquitetura do *Redux*, escolhida por sua facilidade, previsibilidade, mantenabilidade e pelo principal fator da sua escolha, o encapsulamento dos dados em um componente externo à aplicação. Além disso, foi utilizado o *react-redux* para integrar o *React* com o *Redux*, assim como o *redux-toolkit* como ferramenta indicada para simplificar a implementação do *Redux* dentro do SPA. A composição da SPA está exposta a seguir, na Figura 4.

Figura 4 - Arquitetura da SPA.

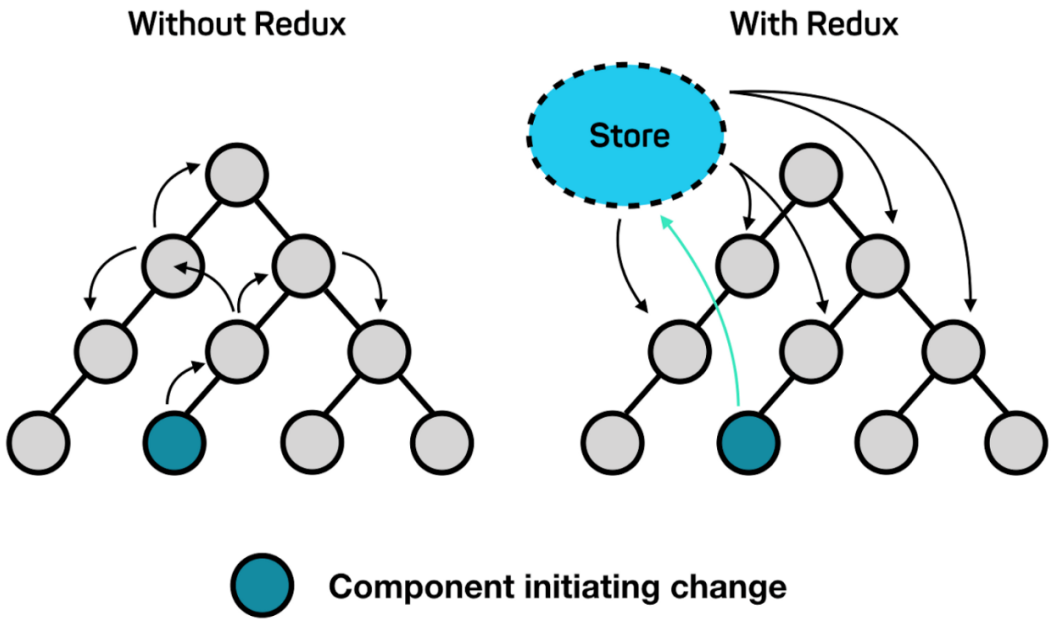


Fonte: Autor

3.2.1 Aplicação do *Redux* e *Redux-toolkit*

O *Redux* é definido como um gerenciador de estados em aplicações *JavaScript*. Dessa forma, o *Redux* tem a capacidade de centralizar todos os estados de uma aplicação dentro da *store*, que se caracteriza como um grande container ou um grande centro de informações capaz de receber e entregar as informações necessárias para a aplicação. Através deste comportamento, o *Redux* tira a responsabilidade de cada um dos componentes de armazenar os estados, sendo utilizado ao mesmo tempo por todos os componentes de forma compartilhada. Além disso, o *Redux* evita um problema muito comum denominado *Prop Drilling*, um termo não oficial para caracterizar cenários onde existe a transferência de informações para componentes em camadas mais fundas. Esse problema é exemplificado na Figura 5, onde através do comparativo é possível perceber a diferença entre duas aplicações, uma utilizando a outra não utilizando o *Redux*.

Figura 5 - Comparativo entre um sistema com e sem *Redux.*

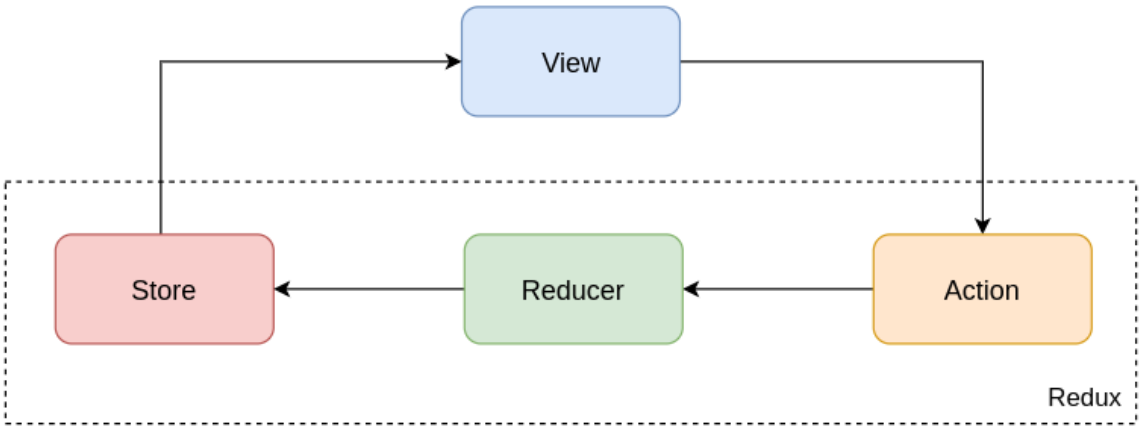


Fonte: Level Up Coding. Disponível em: <https://levelup.gitconnected.com/a-reduced-explanation-of-redux-reducers-actions-and-store-9b7819c5d064/>. Acesso em: 20 setembro 2022.

Além da *store*, citada anteriormente, o *Redux* possui mais dois recursos de suma importância no seu funcionamento: as *actions* (São ações disparadas da aplicação para o *store*) e os *reducers* (encarregado de lidar com todas as ações e especificam como o estado da aplicação irá mudar de acordo com a *action* que foi enviada para o *store*).

Dito isso, tem-se na Figura 6 uma ilustração de como esse fluxo de dados funciona dentro do *Redux*. Caso tenha-se um cenário onde é necessário incrementar um valor armazenado na *store*, primeiramente tem-se uma *view* (um componente *React*, por exemplo), que dispara uma determinada *action* (ex: incrementar o valor) direcionada para o *reducer* específico (ex: função que incrementa o valor). Esse *reducer* realiza a alteração desejada e envia o resultado a *store*. Por fim, a *view* resgata o valor novamente do *store*, porém desta vez já incrementado, conforme o cenário exemplificado.

Figura 6 - Fluxo de dados utilizando Redux.



Fonte: Autor

Diante desse contexto do *Redux*, foi utilizado na aplicação o *Redux-toolkit* [6], uma biblioteca que facilita a configuração do *Redux* dentro do projeto. Dentro desta biblioteca temos 2 conceitos exibidos no diagrama do SPA, o conceito de *slice* e o conceito de *thunk.* O *slice* se caracteriza como uma “fatia”, agrupando *actions*, *reducers* e uma parte da *store* para um determinado contexto. Ou seja, cada funcionalidade do sistema possui sua própria “fatia”, cada um com suas regras e fluxos diferentes, todos eles inseridos dentro do contexto global do *Redux.* Já o *thunk* funciona como um *middleware*, que permite retornar funções, em vez de apenas ações, dentro do *Redux*. Isso permite ações com atraso, incluindo trabalhar com *promises*.

Dentro do escopo da aplicação foram criados 4 *slices*, uma para as operações de Administrador, uma para os processos de login, uma para os dados do usuário e por fim, uma para as operações de requerimento de estágio. Dessa forma, cada *slice* se comunica com seu próprio *thunk*, que por sua vez se comunica com o serviço responsável por fazer a chamada ao *back-end*, conforme o diagrama mostrado anteriormente na Figura 4.

Capítulo 4

Especificação dos Requisitos

Será descrito neste capítulo os requisitos da aplicação, trazendo de forma detalhada cada história de usuário. Cada história será detalhada trazendo cenários de sucesso e erro, abordando por completo as funcionalidades presentes no sistema.

4.1. Autenticação

Funcionalidade: Login com e-mail institucional

Como um aluno ou professor da Escola Politécnica de Pernambuco não logado no sistema

Eu quero poder utilizar o meu e-mail Google vinculado a universidade

Para acessar o sistema

Cenário 1: Login Google com sucesso

Dado que possuo um e-mail Google vinculado a universidade

E vou realizar o login no sistema

Quando clicar no botão de login com Google e realizar os passos necessários com sucesso

Então devo ser redirecionado para a tela de home do usuário contendo a lista de requerimentos

Cenário 2: Login Google com falha

Dado que possuo um e-mail Google vinculado a universidade

E vou realizar o login no sistema

Quando houver um erro no processo de login com Google

Então deve ser exibido a mensagem “Houve um erro ao realizar o login. Tente novamente!”

Cenário 3: Primeiro acesso ao sistema

Dado que possuo um e-mail Google vinculado a universidade

E vou realizar o login no sistema pela primeira vez

Quando realizar o login com sucesso

Então devo ser redirecionado para a home e um *pop-up* deverá aparecer, solicitando o preenchimento dos meus dados

Funcionalidade: Login de administrador

Como um usuário do tipo administrador não logado no sistema

Eu quero poder utilizar as minhas credenciais

Para acessar o painel administrativo

Cenário 1: Login admin com sucesso

Dado que possuo um usuário de administrador

E vou iniciar o login de admin no sistema

E defino o “email”

E defino a “senha”

Quando clicar no botão de login

Então devo ser redirecionado para o painel administrativo contendo as listagens de usuários, cursos e requerimentos.

Cenário 2: Login admin com falha

Dado que possuo um usuário de administrador

E vou iniciar o login de admin no sistema

E defino o “email”

E defino a “senha”

Quando houver um erro de login

Então deve ser exibido a mensagem “Houve um erro ao realizar o login. Tente novamente!”

Funcionalidade: Proteção de rotas do sistema

Como um usuário não autenticado

Eu quero ser impedido de navegar

Para não acessar as rotas protegidas do sistema

Cenário 1: Redirecionamento

Dado que possuo um usuário não autenticado no sistema

E vou acessar uma rota protegida

Quando realizar o acesso a página

Então devo ser redirecionado para a tela de login

Funcionalidade: Logout

Como um usuário logado no sistema

Eu quero poder realizar o logout

Para encerrar minha sessão no sistema

Cenário 1: Logout com sucesso

Dado que possuo um usuário logado no sistema

E vou iniciar o logout

Quando clicar no botão de “Sair”

Então devo ser redirecionado para a tela de login

Cenário 2: Logout com falha

Dado que possuo um usuário logado no sistema

E vou iniciar o logout

Quando houver um erro no logout

Então deve ser exibido a mensagem “Houve um erro ao encerrar sua sessão. Tente novamente!”

4.2. Autorização

Funcionalidade: Edição de perfil de usuário

Como usuário devidamente autenticado, com o perfil de acesso de funcionário da divisão de estágio

Eu quero alterar os perfis de acesso dos usuários cadastrados

Para definir corretamente, de acordo com o atual corpo docente, quem são os usuários que devem possuir o perfil de acesso "Professor".

Cenário 1: Edição de perfil de usuário com sucesso

Dado que possuo um usuário com perfil de acesso de funcionário da Divisão de Estágio logado no sistema

E existe uma base de usuários já cadastrados no sistema

E vou editar um usuário para alterar seu perfil de acesso

Quando finalizar a edição

Então deve ser exibida uma mensagem de sucesso e a listagem de usuários deve ser atualizada

Cenário 2: Edição de perfil de usuário com falha

Dado que possuo um usuário com perfil de acesso de funcionário da Divisão de Estágio logado no sistema

E existe uma base de usuários já cadastrados no sistema

E vou editar um usuário para alterar seu perfil de acesso

Quando finalizar a edição e houver um erro

Então deve ser exibida uma mensagem de erro para o usuário

4.3. Requerimento

Funcionalidade: Criação de requerimento

Como usuário devidamente autenticado, com o perfil de acesso de aluno

Eu quero criar um requerimento de estágio

Para dar início ao processo de estágio na universidade

Cenário 1: Criação de requerimento com sucesso

Dado que possuo um usuário com perfil de acesso de aluno logado no sistema

E existe uma base de professores do curso deste aluno já cadastrados no sistema

Quando finalizar a criação do requerimento

Então deve ser exibida uma mensagem de sucesso e redirecionado para a home de aluno

Cenário 2: Criação de requerimento com falha

Dado que possuo um usuário com perfil de acesso de aluno logado no sistema

E existe uma base de professores do curso deste aluno já cadastrados no sistema

Quando finalizar a criação do requerimento com alguma falha

Então deve ser exibida uma mensagem de erro indicando a falha no formulário.

Funcionalidade: Edição de requerimento

Como usuário devidamente autenticado, com o perfil de acesso de aluno e com um requerimento criado

Eu quero editar um requerimento de estágio

Para atualizar as informações do meu requerimento em andamento

Cenário 1: Edição de requerimento com sucesso

Dado que possuo um usuário com perfil de acesso de aluno logado no sistema, com pelo menos um requerimento cadastrado no sistema.

E vou editar este requerimento

Quando finalizar a edição do requerimento

Então deve ser exibida uma mensagem de sucesso e a janela de edição deve ser fechada

Cenário 2: Edição de requerimento com falha

Dado que possuo um usuário com perfil de acesso de aluno logado no sistema, com pelo menos um requerimento cadastrado no sistema.

E vou editar este requerimento

Quando finalizar a edição do requerimento com falha

Então deve ser exibida uma mensagem de erro para o usuário

Funcionalidade: Detalhamento de requerimento

Como usuário devidamente autenticado, com o perfil de acesso de aluno ou professor, e com um requerimento criado

Eu quero visualizar os detalhes do meu requerimento

Para me certificar de que os dados estão corretos

Cenário 1: Visualização de detalhes de requerimento

Dado que possuo um usuário com perfil de acesso de aluno ou professor logado no sistema, com pelo menos um requerimento cadastrado no sistema.

E vou visualizar este requerimento

Quando clicar no botão de ver mais detalhes

Então deve ser exibida uma janela com todas as informações do requerimento, incluindo os dados do aluno, dados do estágio, dados da empresa e os documentos solicitados.

Funcionalidade: Aprovação/Aceite de requerimento

Como usuário devidamente autenticado, com o perfil de acesso de professor ou funcionário da Divisão de Estágio, e com um requerimento criado

Eu quero aprovar/aceitar um requerimento

Para avançar no processo de estágio do aluno vinculado ao requerimento

Cenário 1: Aprovação/Aceite com sucesso

Dado que possuo um usuário com perfil de acesso de funcionário da Divisão de Estágio ou professor logado no sistema, com pelo menos um requerimento cadastrado no sistema.

E vou aprovar/aceitar um requerimento

Quando clicar no botão de aprovar

Então deve ser exibida uma mensagem de sucesso

Cenário 2: Aprovação/Aceite com falha

Dado que possuo um usuário com perfil de acesso de aluno ou professor logado no sistema, com pelo menos um requerimento cadastrado no sistema.

E vou aprovar/aceitar um requerimento

Quando clicar no botão de aprovar e ocorra um erro no sistema

Então deve ser exibida uma mensagem de erro dizendo “Houve um erro ao aprovar o requerimento. Tente novamente!”

Funcionalidade: Reprovação/Recusa de requerimento

Como usuário devidamente autenticado, com o perfil de acesso de professor ou funcionário da Divisão de Estágio, e com um requerimento criado

Eu quero reprovar/recusar um requerimento

Para que o aluno vinculado ao requerimento possa editá-lo e solicitar uma nova avaliação do mesmo

Cenário 1: Reprovação/Recusa com sucesso

Dado que possuo um usuário com perfil de acesso de aluno ou professor logado no sistema, com pelo menos um requerimento cadastrado no sistema.

E vou reprovar/recusar um requerimento

Quando clicar no botão de reprovar

Então deve ser exibida uma mensagem de sucesso

Cenário 2: Reprovação/Recusa com falha

Dado que possuo um usuário com perfil de acesso de aluno ou professor logado no sistema, com pelo menos um requerimento cadastrado no sistema.

E vou reprovar/recusar um requerimento

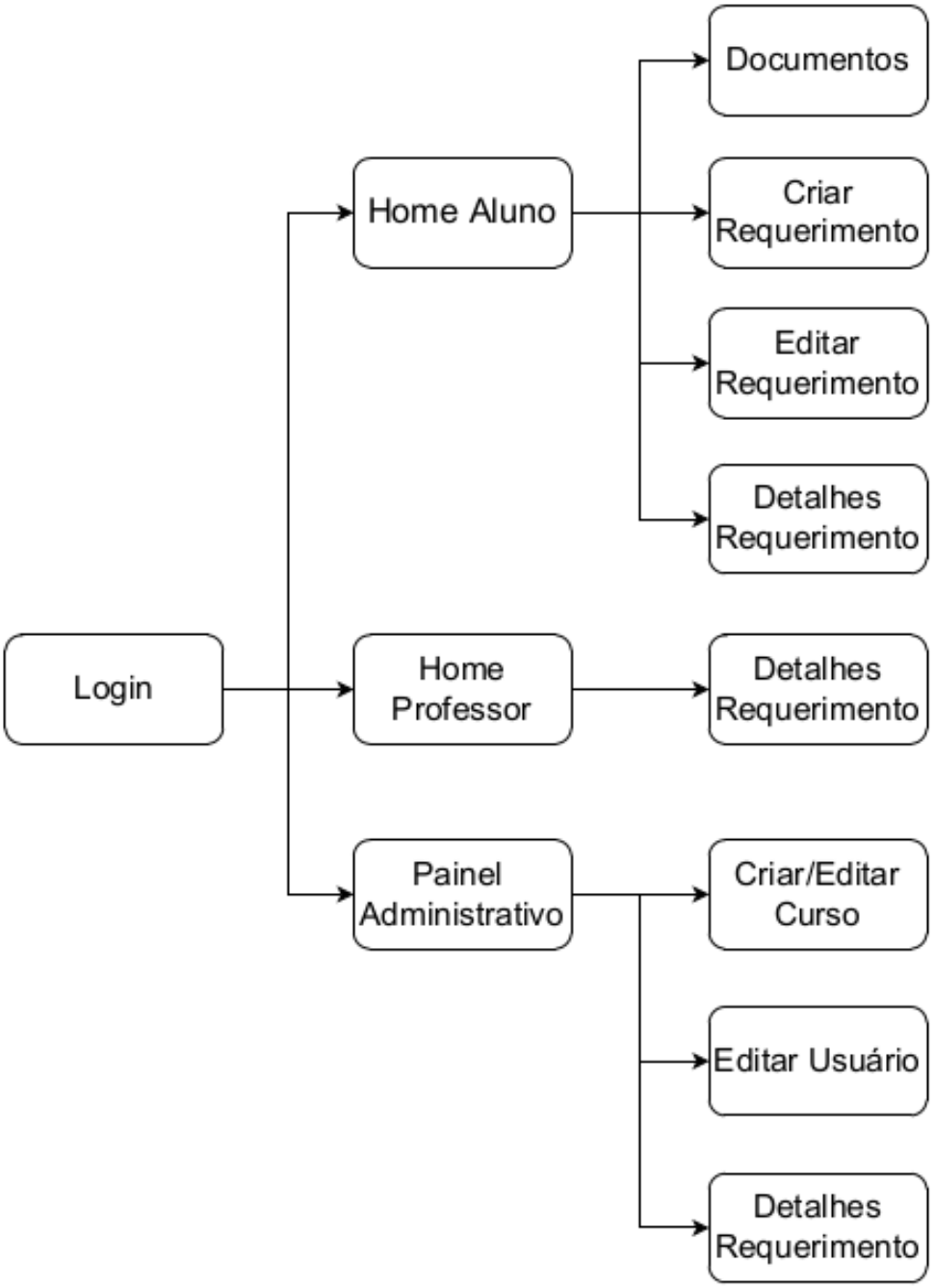
Quando clicar no botão de reprovar e ocorra um erro no sistema

Então deve ser exibida uma mensagem de erro dizendo “Houve um erro ao reprovar o requerimento. Tente novamente!”

4.4. Fluxograma de telas

De acordo com a Figura 7, é possível observar o fluxograma das telas existentes no sistema, onde o mesmo inicia-se na tela de login e a partir desta tela pode-se navegar para as telas subsequentes. Alguns *pop-ups*, como por exemplo o de detalhes de requerimento e edição de usuários, foram considerados como telas do sistema pois assumem grande importância dentro do escopo da aplicação.

Figura 7 - Fluxograma de telas do sistema



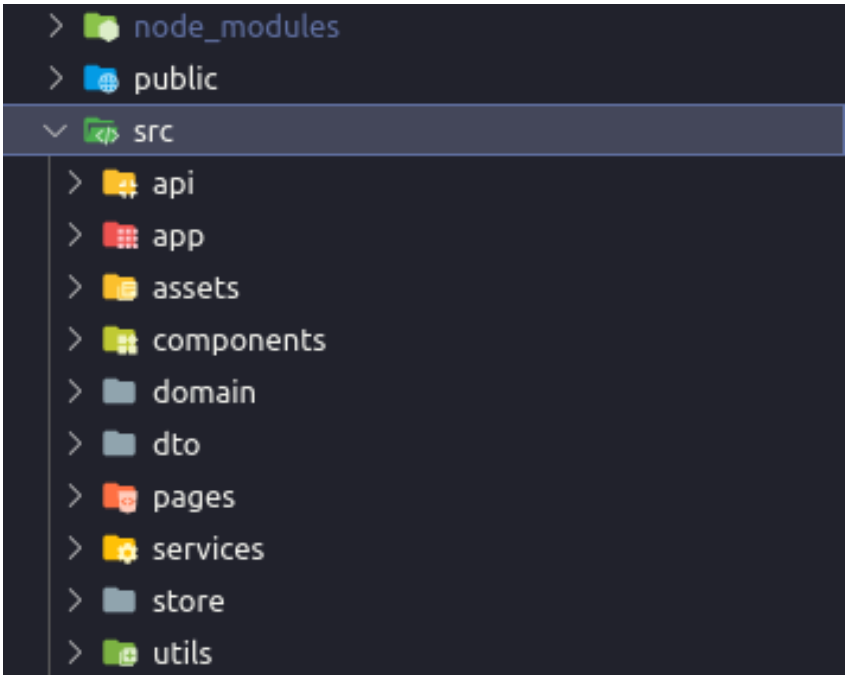
Fonte: Autor

Capítulo 5

Front-End do Sistema de Estágio da POLI-UPE

Como resultado deste trabalho, será feito uma análise da estrutura construída no projeto, além da exibição das telas referentes ao sistema construído. De acordo com a Figura 8 tem-se a disposição das pastas do projeto, onde a regra de negócio do sistema se concentra na pasta “src”. Na tabela 1, tem-se detalhadamente a definição de cada sub-pasta existente no projeto, começando pela pasta “api” até a pasta “utils”.

Figura 8 - Estrutura de pastas do sistema.



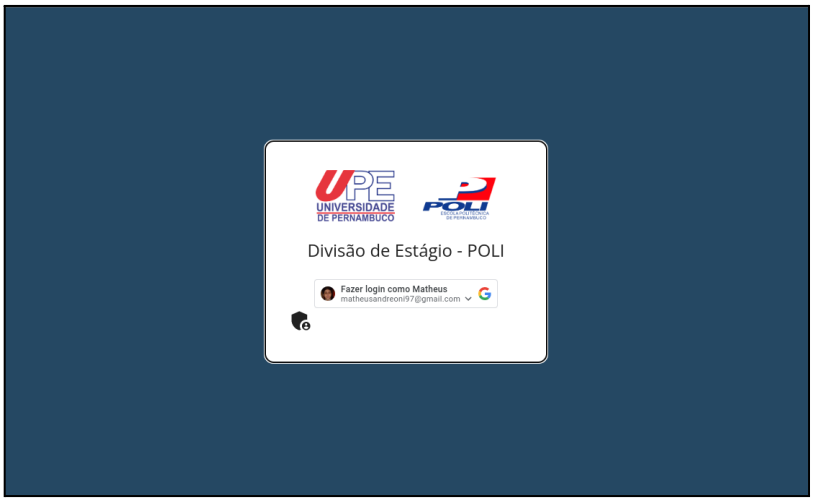
Fonte: Autor

Tabela 1 - Detalhes das pastas do sistema

Nome da pasta	Definição
api	Arquivos que compõem a estrutura de comunicação HTTP com o servidor
app	Configuração do *Redux* dentro da aplicação
assets	Arquivos de imagem que são utilizados dentro do sistema
components	Componentes que são comuns a mais de uma tela dentro do sistema, como por exemplo um componente de *Header*
domain	Armazena todas as entidades utilizadas ao longo da construção do sistema (ex: dados do usuário ou os dados de um requerimento)
dto	Entidades DTO (*Data Transfer Object*) do sistema, para se comunicar corretamente com o servidor
pages	Telas do sistema, separadas por funcionalidade (ex: telas de requerimento estão agrupadas em uma pasta)
services	Serviços responsáveis por realizar a comunicação HTTP com o servidor
store	Armazena os *slices* utilizados no sistema
utils	funções utilitárias usadas em todo o sistema

Em relação às telas desenvolvidas, tem-se na Figura 9 a tela de login para alunos e professores, utilizando os e-mails da instituição previamente cadastrados. Essa tela se caracteriza como a primeira tela ao acessar o sistema.

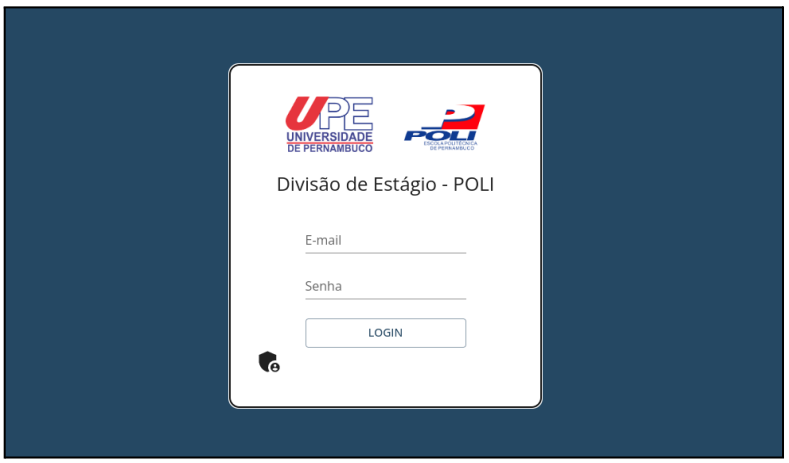
Figura 9 - Tela de login de alunos e professores.



Fonte: Autor

Já na Figura 10 tem-se o login referente aos funcionários e administradores do sistema. Esse login se difere do login exibido na Figura 9, pois o acesso se dá através de um formulário, requerendo e-mail e senha do usuário administrador.

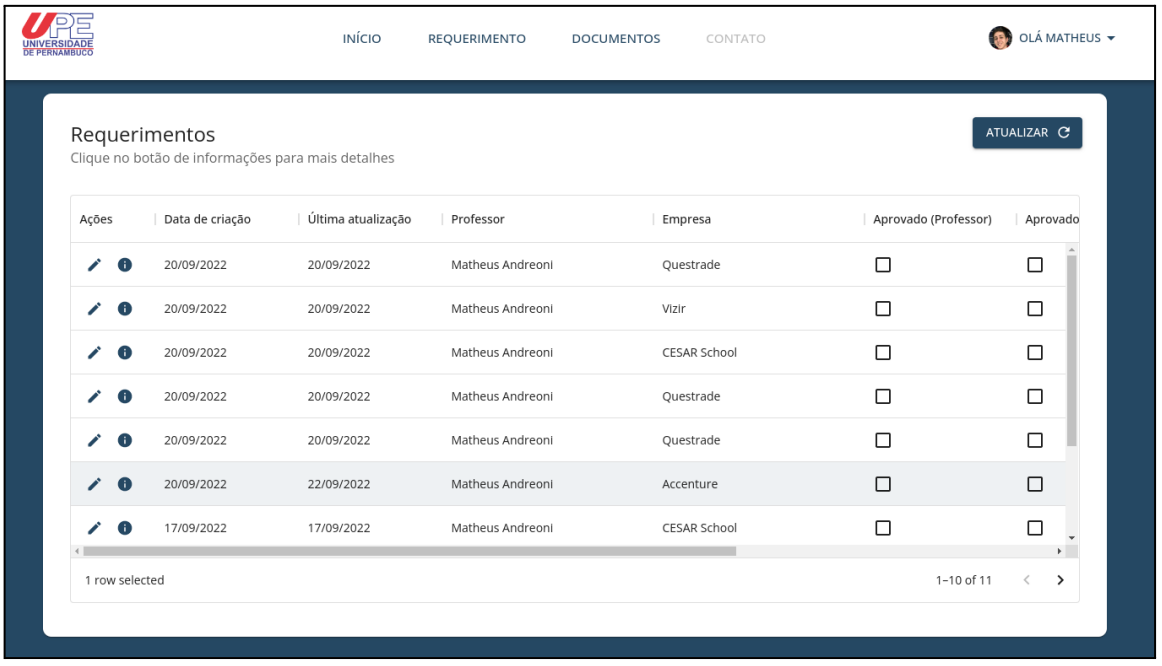
Figura 10 - Tela de login de administradores e funcionários.



Fonte: Autor

Após o login do aluno ou do professor ser realizado com sucesso, o usuário será redirecionado para a tela de home. Através da home do usuário, demonstrada na Figura 11, tem-se a listagem de requerimentos vinculados ao usuário. Essa listagem foi construída com o componente denominado *Data Grid* [8]. Através deste componente, é possível abstrair algumas implementações como regras de paginação, estilização e responsividade, todas nativas do próprio componente. Para a implementação da listagem foi necessário apenas definir as colunas e seus respectivos comportamentos. Caso o usuário seja do tipo Aluno, o mesmo terá a opção de editar os seus requerimentos e ver os detalhes do requerimento escolhido. Caso o usuário seja do tipo Professor, o mesmo poderá também visualizar os detalhes de cada requerimento, assim como aprová-lo ou recusá-lo (com um motivo de recusa).

Figura 11 - Tela de home do usuário.



Fonte: Autor

A Figura 12 mostra a tela referente a edição de dados do usuário logado no sistema. Através deste *pop-up*, construído através do componente do *Material UI* denominado *Dialog* [7], é possível alterar as informações pessoais de cada usuário,

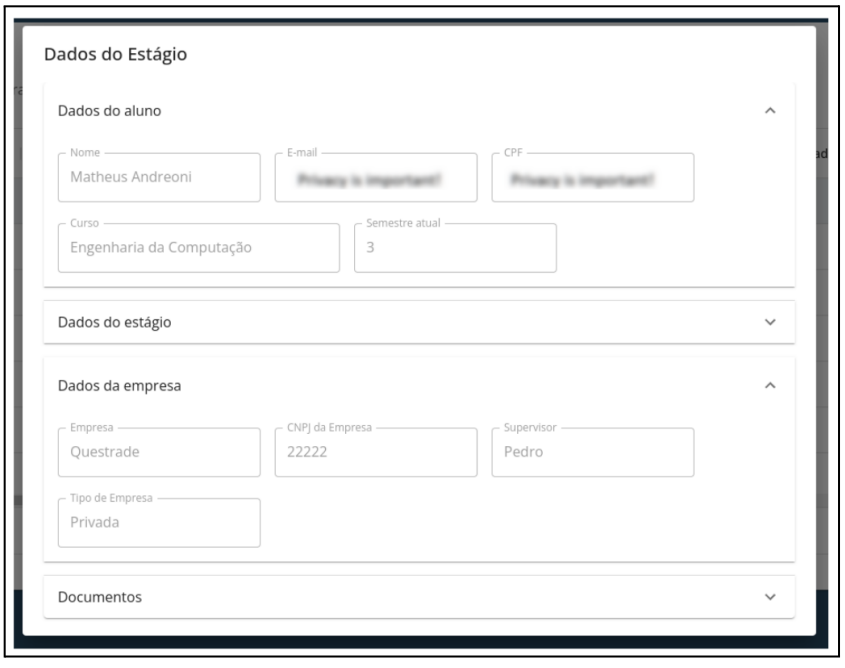
tais como nome, sobrenome e curso. Caso seja o primeiro acesso do usuário, esse *pop-up* será aberto automaticamente para que o usuário em questão possa finalizar o preenchimento de seus dados.

Figura 12 - Tela de edição de dados do usuário.

Fonte: Autor

Dentro da listagem de requerimentos exibida na Figura 11, tem-se a opção de visualizar os detalhes de cada requerimento, conforme exibido na Figura 13. Para acessar esses detalhes é necessário clicar no botão de informações presente na coluna de ações da listagem de requerimentos.

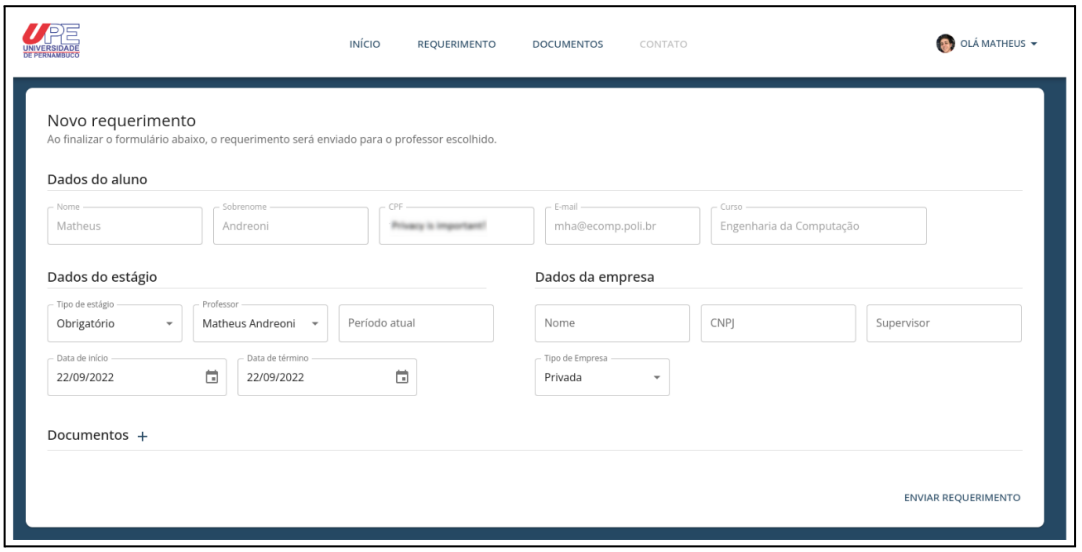
Figura 13 - Tela de detalhes de requerimento.



Fonte: Autor

No formulário da Figura 14, o usuário do tipo Aluno poderá criar um requerimento preenchendo os campos presentes na página. Além dos campos de texto, tem-se uma área destinada aos documentos, onde o aluno enviará os arquivos desejados e estes são listados para que o usuário possa avaliar e eventualmente deletar um arquivo anexado. Finalizado o preenchimento, o aluno deve clicar na opção de enviar requerimento e este é enviado ao professor. O envio se dá de forma assíncrona e em caso de sucesso, o aluno é redirecionado para a home. Em caso de erro, o sistema mostrará um alerta exibindo o que houve de errado para que o aluno possa corrigir e enviar novamente.

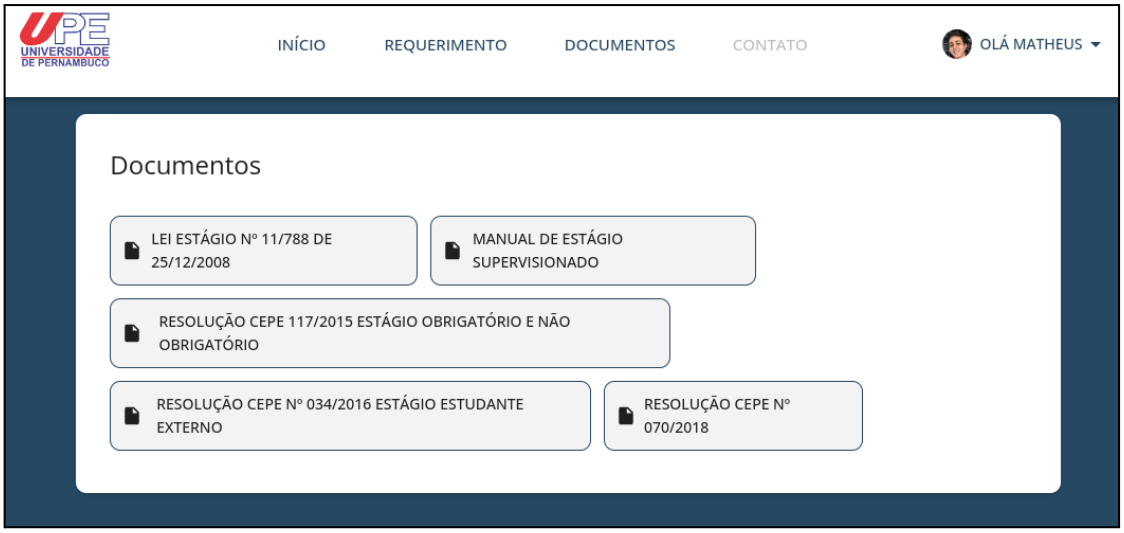
Figura 14 - Tela de formulário de novo requerimento.



Fonte: Autor

A página exibida na Figura 15 traz documentos importantes para o processo de estágio dentro da universidade. Cada documento foi construído por uma *Box* [9], um componente derivado do *Material UI* que nos fornece um encapsulador capaz de aceitar qualquer estilização necessária.

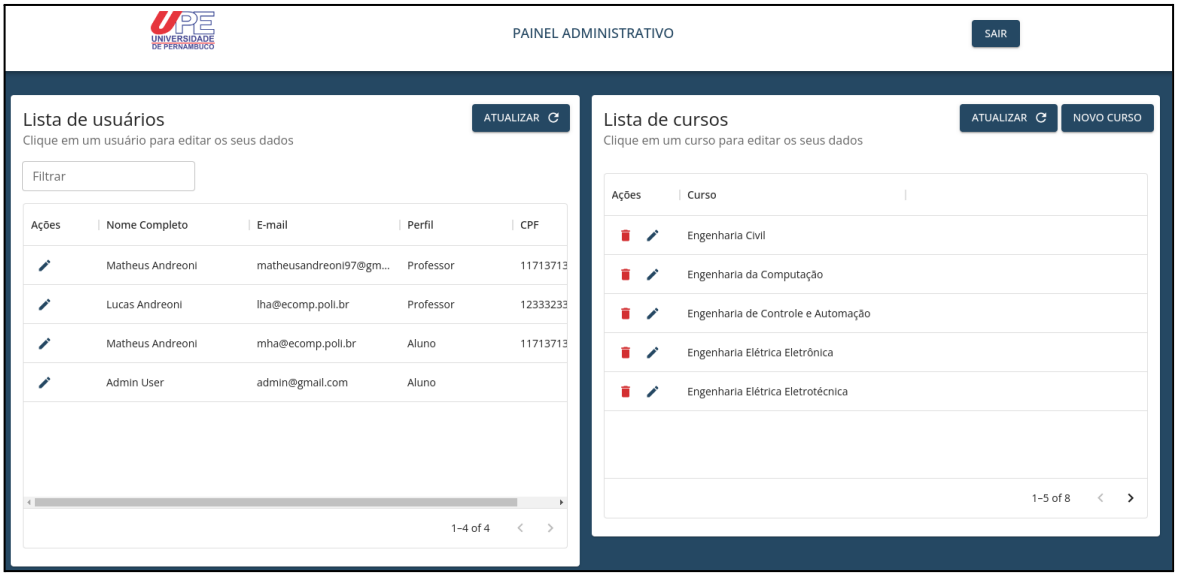
Figura 15 - Tela de documentos.



Fonte: Autor

Por fim tem-se na Figura 16 o painel administrativo, acessado pelos funcionários e pelos administradores. Este painel traz a listagem e fornece a opção de edição dos usuários do sistema além da criação, deleção e edição dos cursos presentes no sistema, O painel também traz a listagem de requerimentos para serem aprovados ou reprovados pela Divisão de Estágio, junto com a funcionalidade de visualização dos detalhes dos requerimentos de estágio.

Figura 16 - Painel administrativo do sistema.



Fonte: Autor

Com a aplicação do *Material UI* no projeto, a responsividade das telas também foram levadas em consideração. Através dela, é possível atingir uma experiência de usuário de qualidade em telas de menores tamanhos, como em celulares e tablets. No Apêndice A deste documento, são exibidas algumas telas do sistema em formato responsivo.

Capítulo 6

Conclusões

A partir do que foi exposto neste trabalho, fica evidente a necessidade da criação de um sistema para a Divisão de Estágio da POLI-UPE. Através do uso do *React*, *Redux* e do *Material UI* como recursos para a construção deste sistema, mostrou-se como soluções modernas podem auxiliar e agilizar os processos dentro da universidade.

O trabalho teve como foco a implementação do fluxo de autenticação, autorização e requerimentos de estágio, sendo manipulado pelos diferentes usuários já citados nas seções anteriores.

Após a realização deste trabalho esperam-se trabalhos futuros relacionados a implementação de novos processos da Divisão de Estágio, como por exemplo a criação de relatórios, a realização das avaliações dos mesmos dentro da plataforma e a inserção de um perfil para empresas, aproximando ainda mais os participantes do processo de estágio dentro da plataforma. Além disso, implementar uma camada de testes unitários e de integração, a fim de garantir uma maior confiabilidade nas mudanças a serem desenvolvidas no futuro.

Finalizando, também espera-se que esse sistema sirva de exemplo para contínua aplicação e inserção de novas tecnologias no contexto da universidade.

Referências

[1] **React - Uma biblioteca JavaScript para criar interfaces de usuário.** React, 2022. Disponível em: <https://pt-br.reactjs.org/>, Acesso em: 26 setembro 2022.

[2] **Redux - A Predictable State Container for JS Apps**. Redux, 2022.Disponível em: <https://redux.js.org/>, Acesso em: 25 setembro 2022.

[3] **MUI: The React component library you always wanted**. Mui, 2022. Disponível em: <https://mui.com/pt/>, Acesso em 26 setembro 2022.

[4] **Material Design**. Material, 2022. Disponível em: <https://material.io/>, Acesso em 26 setembro 2022.

[5] **Google Identity Services.** Google, 2022. Disponível em: <https://developers.google.com/identity/>, Acesso em: 5 outubro 2022.

[6] **Redux Toolkit - The official, opinionated, batteries-included toolset for efficient Redux development.** Redux Toolkit, 2022. Disponível em: <https://redux-toolkit.js.org/>, Acesso em: 23 setembro 2022.

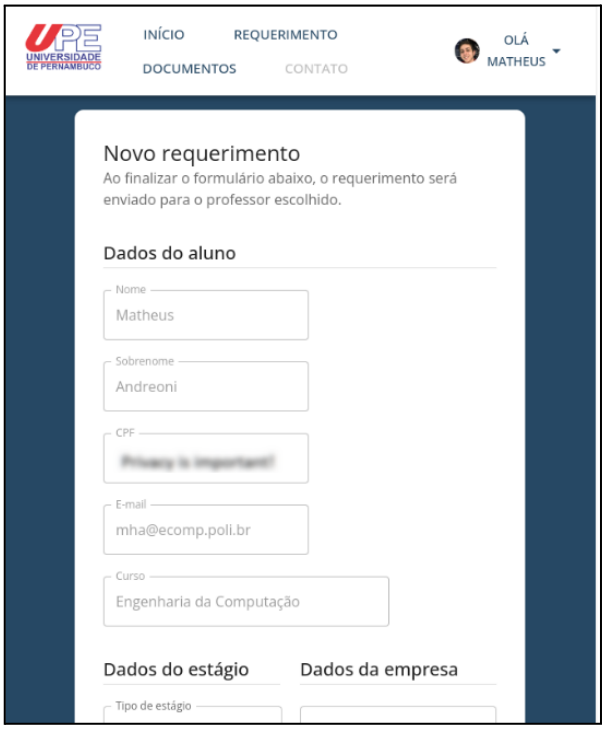
[7] **MUI Dialog**. Mui, 2022. Disponível em: <https://mui.com/pt/material-ui/react-dialog/>, Acesso em 05 outubro 2022.

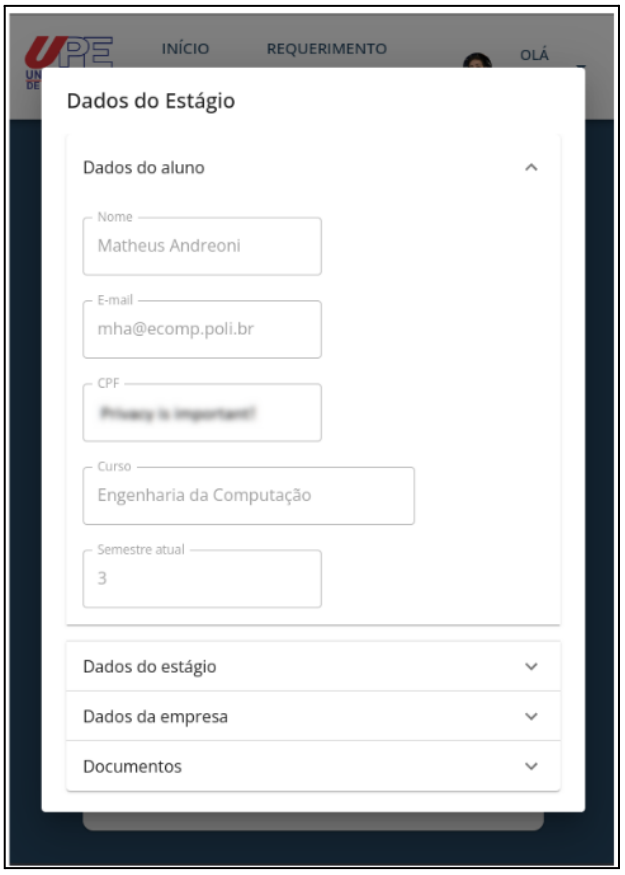
[8] **MUI Data Grid**. Mui, 2022. Disponível em: <https://mui.com/pt/x/react-data-grid//>, Acesso em 05 outubro 2022.

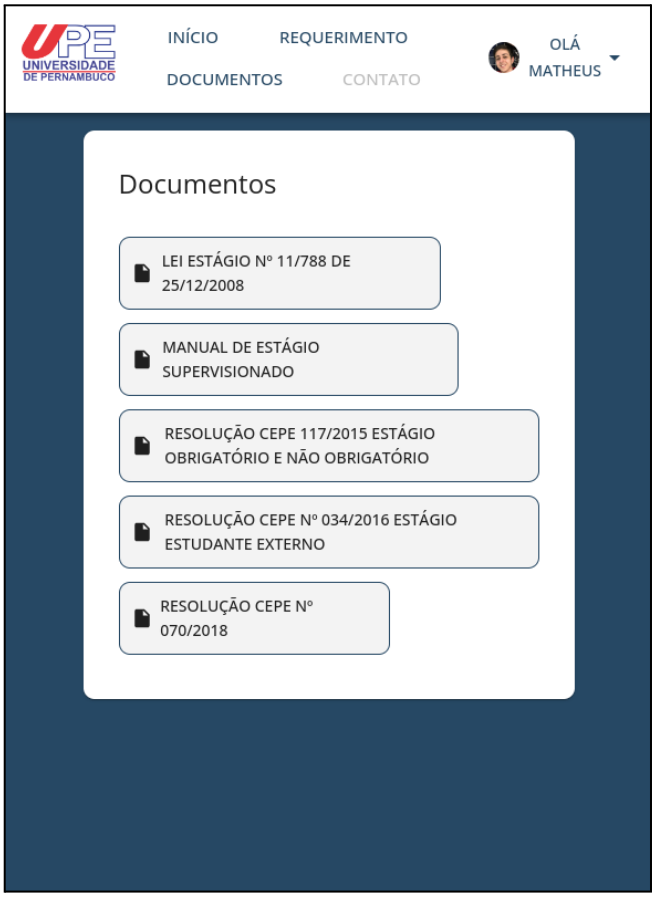
[9] **MUI Box**. Mui, 2022. Disponível em: <https://mui.com/pt/material-ui/react-box/>, Acesso em 05 outubro 2022.

Apêndice A

Telas Responsivas




INÍCIO
REQUERIMENTO
OLÁ
Dados do Estágio
Dados do aluno
Nome
Matheus Andreoni
E-mail
mha@ecomp.poli.br
CPF
Curso
Engenharia da Computação
Semestre atual
3
Dados do estágio
Dados da empresa
Documentos


UNIVERSIDADE DE PERNAMBUCO
INÍCIO
REQUERIMENTO
DOCUMENTOS
CONTATO
OLÁ MATHEUS
Documentos
LEI ESTÁGIO Nº 11/788 DE 25/12/2008
MANUAL DE ESTÁGIO SUPERVISIONADO
RESOLUÇÃO CEPE 117/2015 ESTÁGIO OBRIGATÓRIO E NÃO OBRIGATÓRIO
RESOLUÇÃO CEPE Nº 034/2016 ESTÁGIO ESTUDANTE EXTERNO
RESOLUÇÃO CEPE Nº 070/2018