REPÚBLICA FEDERATIVA DO BRASIL
MINISTÉRIO DA ECONOMIA
INSTITUTO NACIONAL DA PROPRIEDADE INDUSTRIAL
DIRETORIA DE PATENTES, PROGRAMAS DE COMPUTADOR E TOPOGRAFIAS DE CIRCUITOS INTEGRADOS

Certificado de Registro de Programa de Computador

Processo Nº: **BR512022001662-1**

O Instituto Nacional da Propriedade Industrial expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de 29/03/2022, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Título: SIGEE: SISTEMA DE GESTÃO DE ESTÁGIOS

Data de publicação: 29/03/2022

Data de criação: 25/03/2022

Titular(es): FUNDAÇÃO UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE - FUERN

Autor(es): SEBASTIÃO EMIDIO ALVES FILHO; GABRIEL NASCIMENTO JALES; JONAS SILVA RODRIGUES; JEFFERSON XIMENES ROCHA DE SOUSA; RAFAEL DA SILVA XIMENES

Linguagem: JAVA SCRIPT

Campo de aplicação: AD-01; IF-02; IF-10

Tipo de programa: AP-01; FA-01; GI-01

Algoritmo hash: SHA-512

Resumo digital hash:
616861684bfc34d8519e364fa5831e3b945cd3a069367605bf716d058e25b8f0ad98ecee3d606dd3dc94b77a6ef749a0560386f732a9c7d376f21e390bb4956a

Expedido em: 12/07/2022

Aprovado por:
Joelson Gomes Pequeno
Chefe Substituto da DIPTO - PORTARIA/INPI/DIRPA Nº 02, DE 10 DE FEVEREIRO DE 2021

UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE – UERN

FACULDADE DE CIÊNCIAS EXATAS E NATURAIS – FANAT

DEPARTAMENTO DE INFORMÁTICA – DI

Jefferson Ximenes Rocha de Sousa

SIGEE: Sistema de Gestão de Estágios

Na perspectiva do Front-End

MOSSORÓ - RN

2022

Jefferson Ximenes Rocha de Sousa

SIGEE: SISTEMA DE GESTÃO DE ESTÁGIOS

NA PERSPECTIVA DO FRONT-END

Relatório apresentado ao curso de Ciência da Computação da Universidade do Estado do Rio Grande no Norte como requisito da disciplina de Trabalho de Diplomação, sob a orientação do Prof. Dr. Sebastião Emidio Alves Filho e coorientação de Rafael da Silva Ximenes.

MOSSORÓ - RN

2022

JEFFERSON XIMENES ROCHA DE SOUSA

SIGEE: SISTEMA DE GESTÃO DE ESTÁGIOS NA PERSPECTIVA DO FRONT-END

Registro de software apresentado como pré-requisito para a obtenção do título de Bacharel em Ciência da Computação da Universidade do Estado do Rio Grande do Norte – UERN, submetida à aprovação da banca examinadora composta pelos seguintes membros:

Aprovada em: 26/04/2022

Banca Examinadora

Prof. Dr. Sebastião Emídio Alves Filho
Universidade do Estado do Rio Grande do Norte - UERN

Rafael da Silva Ximenes

Prof. Dr. Carlos Heitor Pereira Liberalino
Universidade do Estado do Rio Grande do Norte - UERN

Prof. Dr. Isaac de Lima Oliveira Filho
Universidade do Estado do Rio Grande do Norte - UERN

SUMÁRIO

1 INTRODUÇÃO

A lei n° 11.788, de 25 de setembro de 2008, dispõe sobre o estágio de estudantes. De acordo com o artigo primeiro da mesma, o estágio é definido como um ato escolar educativo, desenvolvimento em ambiente de trabalho, visando preparar o estudante para o trabalho produtivo, desde que esteja frequentando o ensino regular em instituições de educação superior, de educação profissional, de ensino médio, da educação especial e dos anos finais do ensino fundamental, na modalidade profissional da educação de jovens e adultos. Conforme o parágrafo segundo do artigo primeiro, o estágio visa o aprendizado de competências próprias da atividade profissional e à contextualização curricular, objetivando o desenvolvimento do estudante para a vida cidadã e para o trabalho. (BRASIL, 2008).

O número de estagiários no Brasil é de 900 mil segundo a última pesquisa, finalizada em fevereiro de 2021. Esse número é resultado de um levantamento feito com os agentes de integração e instituições de ensino do país. (ABRES, 2021). As informações dos estágios e/ou estagiários precisam ser geridas para que haja um maior controle, principalmente por parte da instituição de ensino a qual o estudante que participará do ato de estágio é vinculado.

Com o objetivo de facilitar e propiciar a gestão dessas informações foi criado o Sistema de Gestão de Estágios (SIGEE), que tem como finalidade ser um ambiente portável, simples e seguro.

2 OBJETIVOS

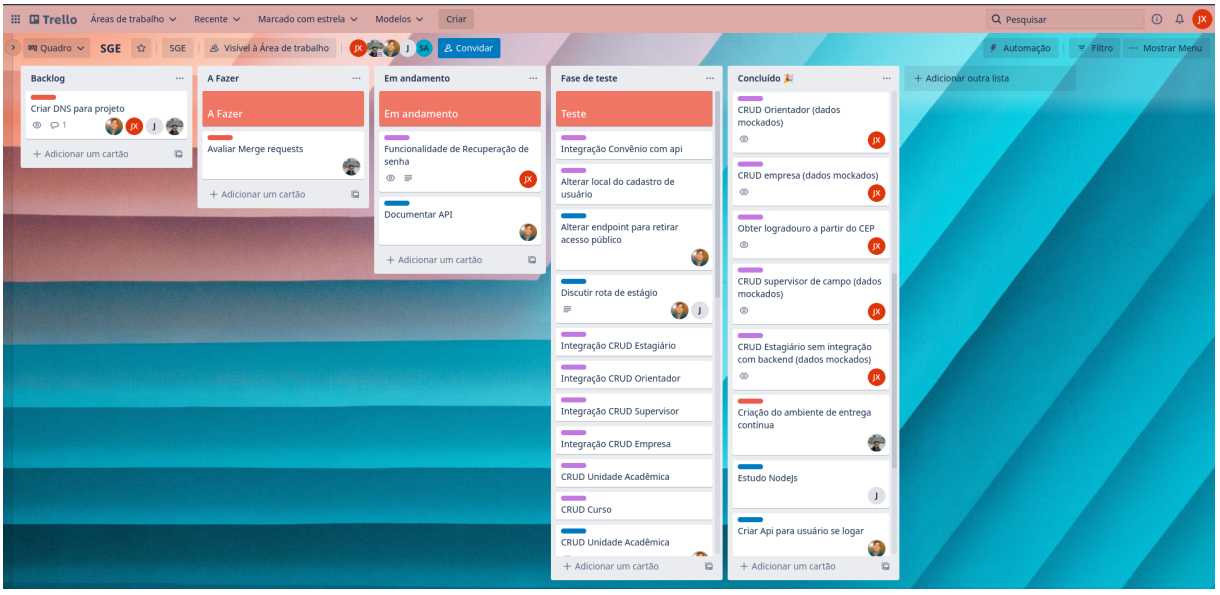
O presente trabalho tem como objetivo discutir sobre a construção da parte *Front-end* do SIGEE. Apresentando como o sistema está estruturado, seus componentes e páginas. Além de mostrar como é feito o fluxo de autenticação e validação via *JWT (JSON Web Token)* e como é consumida a *API (Application Programming Interface)* para que os dados sejam trafegados entre o *Front-end* e a *API*.

3 METODOLOGIA

A metodologia utilizada para a construção do sistema foi a SCRUM (SOMMERVILLE, 2011), que é amplamente utilizada na construção de sistemas. O início foi marcado pela definição do *backlog* do sistema, que é o conjunto de funcionalidades que se pretende ter na aplicação. Após isso foram iniciados os ciclos de desenvolvimento, cada ciclo tem uma duração de tempo e uma quantidade de atividades previamente definidas pela equipe para serem executadas. Esses ciclos são conhecidos como *Sprints*. Após o encerramento das *sprints* a equipe se reuniu para validar se o que foi proposto no *backlog* tinha sido concluído e formalizar a finalização do desenvolvimento.

Foi-se utilizado um aplicativo web de gerenciamento de projeto chamado Trello (TRELLO, 2022) para facilitar a gestão das atividades. No Trello, as atividades são representadas por quadros e cada atividade é atrelada a pessoa que irá executá-la. As atividades foram dispostas em quatro estados: a fazer, em andamento, em fase de testes e concluídas, como mostra a **figura 1**.

Figura 1 - Projeto no Trello.



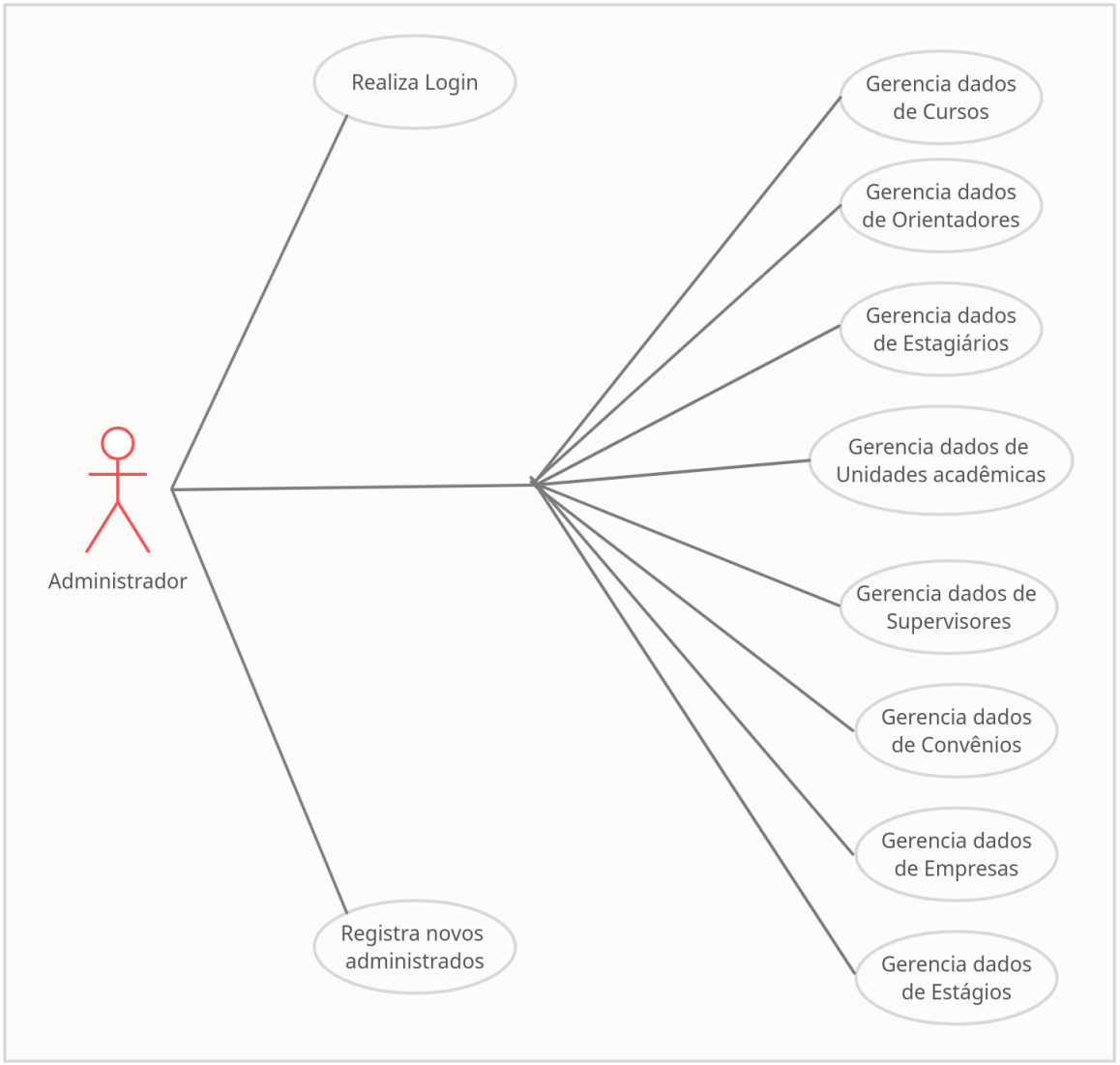
Fonte: Autoria própria.

4 DESCRIÇÃO DO SISTEMA

O SIGEE é um sistema WEB criado com o intuito de gerir as informações referentes a estágios. Ele é composto por duas aplicações principais, *back-end (API)* e *front-end*. De forma simplificada, podemos entender o *back-end* como sendo a camada responsável pela regra de negócio do projeto. Por sua vez, o *front-end* é a camada responsável por interagir com o usuário, ou seja, é a interface gráfica da aplicação.

O sistema utiliza o modelo cliente-servidor, ou seja, o cliente (*front-end*) envia requisições ao servidor (*back-end*) das informações desejadas e o servidor envia uma respostas com as informações, possibilitando ao usuário realizar operações de *CRUD (Create, Read, Update, Delete)* com os dados do estágio, desde que o usuário administrador esteja autenticado e autorizado com um token *JWT*, por exemplo: criar um estagiário, deletar um estagiário, atualizar um estagiário ou apenas visualizar um estagiário. A **figura 2** demonstra detalhadamente os casos de uso do sistema, onde o administrador gerencia os dados do sistema, ou seja, ele realiza todas as operações de CRUD de cada entidade presente no software.

Figura 2 - Diagrama de casos de uso.



Fonte: Autoria própria.

Na figura acima são evidenciadas as entidades que fazem parte do sistema, são elas:

- **Empresa:** Parte responsável por conceder e realizar o estágio ao estudante.
- **Convênio:** É o acordo jurídico firmado entre a empresa e a instituição de ensino para a concessão de estágios, obrigatórios ou não.
- **Curso:** Entidade utilizada para o registro do curso ao qual o estudante está matriculado.
- **Unidade acadêmica:** As unidades acadêmicas são partes da instituição de ensino organizadas por área de conhecimento.
- **Estagiário:** É o estudante que irá participar do ato de estágio na empresa.

- **Supervisor:** O supervisor é o profissional que irá acompanhar o estagiário nas atividades do seu dia-a-dia.
- **Orientador:** É o professor que fará o acompanhamento pedagógico do estágio.
- **Estágio:** Nessa entidade é feita a junção de todas as informações referentes ao estágio, ou seja, quem é o estagiário, a empresa, o orientador, o supervisor de campo, etc…
- **Administrador:** O administrador é responsável por gerir todas as informações das entidades citadas acima, é ele que irá fazer todas as operações no sistema.

Para se ter acesso e controle dos dados referentes ao estágio foram utilizadas 34 rotas que são divididas em 8 grupos (Estagiário, Unidade Acadêmica, Orientador, Supervisor de Campo, Empresa, Convênio, Curso e Estágios). **A tabela 1** mostra o método da requisição, a rota e o objetivo de cada uma.

Tabela 1 - Rotas utilizadas na aplicação.

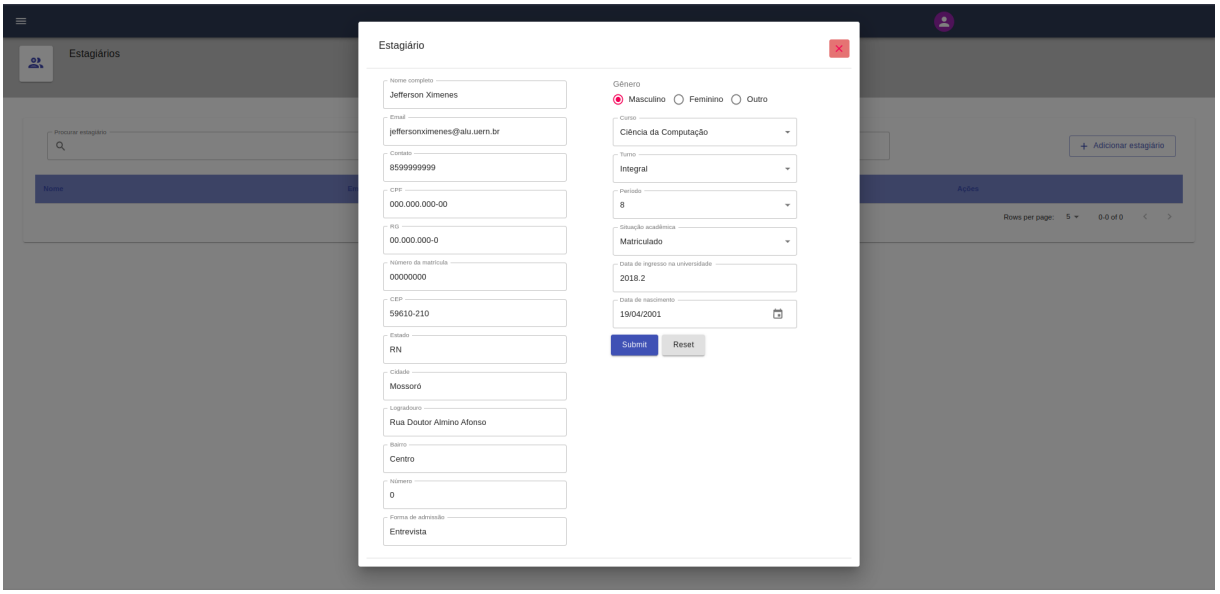
Método	**Rota**	**Objetivo**
GET	/academic_units	Retorna todas as unidades acadêmicas
POST	/academic_units	Cria uma unidade acadêmica
PUT	/academic_units/:id	Atualiza a unidade acadêmica por id
DELETE	/academic_units/:id	Deleta a unidade acadêmica por id
GET	/advisors	Retorna todos os orientadores
POST	/advisors	Cria um orientador
PUT	/advisors/:id	Atualiza o orientador por id
DELETE	/advisors/:id	Deleta o orientador por id
GET	/companies	Retorna todas as empresas
POST	/companies	Cria uma empresa
PUT	/companies/:id	Atualiza a empresa por id

DELETE	/companies/:id	Deleta a empresa por
GET	/agreements	Retorna todos os convênios
POST	/agreements	Cria um convênio
PUT	/agreements/:id	Atualiza o convênio por id
DELETE	/agreements/:id	Deleta o convênio por id
GET	/courses	Retorna todos os cursos
POST	/courses	Cria um curso
PUT	/courses/:id	Atualiza o curso por id
DELETE	/courses/:id	Deleta o curso por id
GET	/interns	Retorna todos os estagiários
POST	/interns	Cria um estagiário
PUT	/interns/:id	Atualiza o estagiário por id
DELETE	/interns/:id	Deleta o estagiário por id
GET	/internships	Retorna todos os estágios
POST	/internships	Cria um estágio
PUT	/internships/:id	Atualiza um estágio por id
DELETE	/internships/:id	Deleta um estágio por id
GET	/supervisors	Retorna todos os supervisores de campo
POST	/supervisors	Cria um supervisor de campo
PUT	/supervisors/:id	Atualiza o supervisor de campo por id
DELETE	/supervisors/:id	Deleta o supervisor de campo por id
POST	/auth/login	Realiza login do usuário
POST	/auth/register	Cria um usuário

Fonte: Autoria própria.

A **figura 3** ilustra o formulário para se criar um estagiário e a **figura 4** mostra como esse estagiário é apresentado na tela.

Figura 3 - Formulário para criar estagiário.



Fonte: Autoria própria.

Figura 4 - Visualização de estagiários.

Fonte: Autoria própria.

A **figura 5** mostra como é feito o processo para criação de um estagiário, utilizando o método POST com os dados necessário para a criação, como: nome, cpf, rg, email, telefone, gênero, data de nascimento, situação acadêmica, matrícula,

data de entrada na universidade, forma de admissão, período, turno, cep, cidade, estado, logradouro, bairro e número.

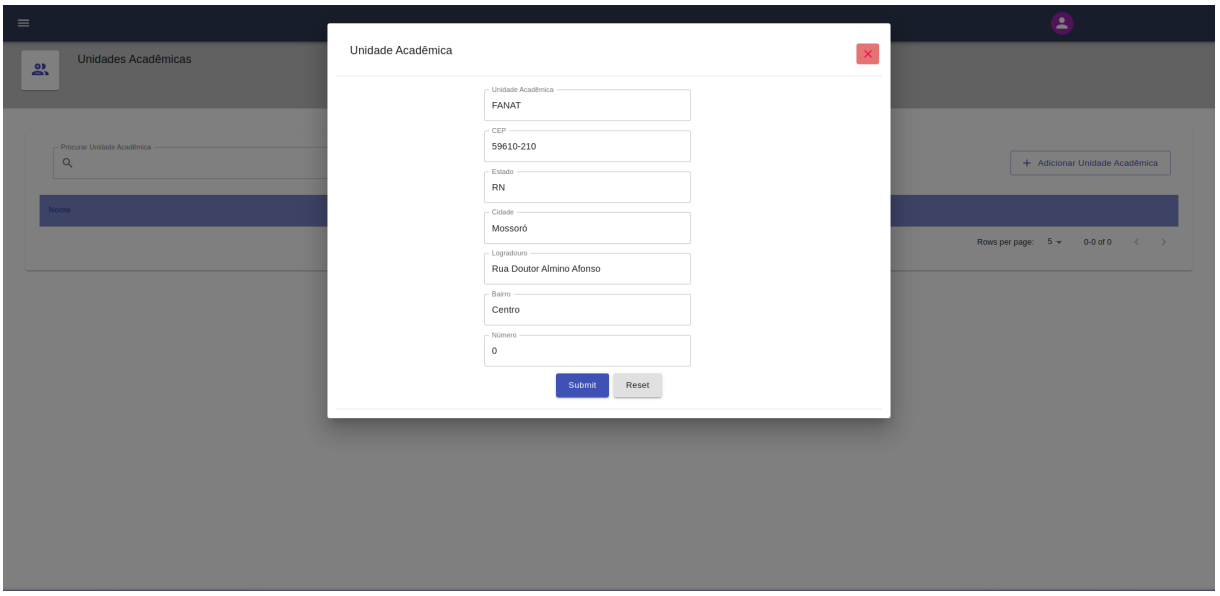
Figura 5 - Processo para criar estagiário.

```
export async function insertIntern(data: any) {

    const {
        name,
        cpf,
        rg,
        email,
        phone,
        gender,
        birth_date,
        academic_situation,
        student_registration,
        academic_entry_year,
        admission_way,
        period,
        shift,
        course_id,
        intern_address
    } = data

    const { cep, state, city, district, street, home_number } = intern_address

    const intern = {
        name: name,
        cpf: cpf,
        rg: rg,
        email: email,
        phone: phone,
        gender: gender,
        birth_date: birth_date,
        academic_situation: academic_situation,
        student_registration: student_registration,
        academic_entry_year: academic_entry_year,
        admission_way: admission_way,
        period: period,
        shift: shift,
        course_id: course_id,
        address: {
            cep: cep.replace('-', ''),
            state: state,
            city: city,
            district: district,
            street: street,
            home_number: parseInt(home_number)
        }
    }

    return await api.post('/interns/', intern)
}
```

Fonte: Autoria própria.

A **figura 6** mostra o formulário para se criar uma unidade acadêmica.

Figura 6 - Formulário para criar Unidade Acadêmica.



Fonte: Autoria própria.

A **figura 7** mostra como é feita a requisição para o servidor utilizando o método POST com os dados necessários para a criação de uma unidade acadêmica, como: nome, cep, cidade, estado, logradouro, bairro e número.

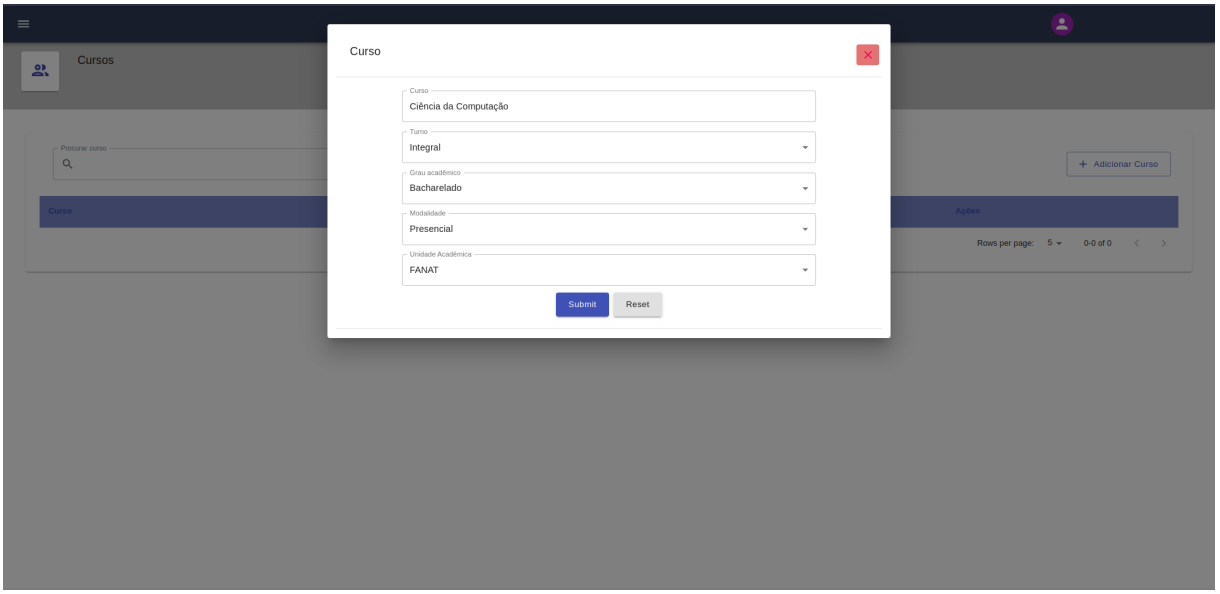
Figura 7 - Processo para criar unidade acadêmica.

```
export async function insertAcademicUnit(data: any) {

    const { name, academic_unit_address } = data
    const { cep, street, home_number, state, city, district } = academic_unit_address

    const obj = {
        name: name,
        address: {
            cep: cep.replace('-', ''),
            state: state,
            street: street,
            home_number: parseInt(home_number),
            city: city,
            district: district
        }
    }

    return await api.post('/academic_units/', obj)
}
```

Fonte: Autoria própria.

A **figura 8** mostra o formulário para se criar um curso.

Figura 8 - Formulário para criar curso.



Fonte: Autoria própria.

A **figura 9** evidencia o processo para se criar um curso. É feita uma requisição utilizando o método POST com as informações necessárias para a criação, como: nome, turno, grau acadêmico, modalidade e unidade acadêmica.

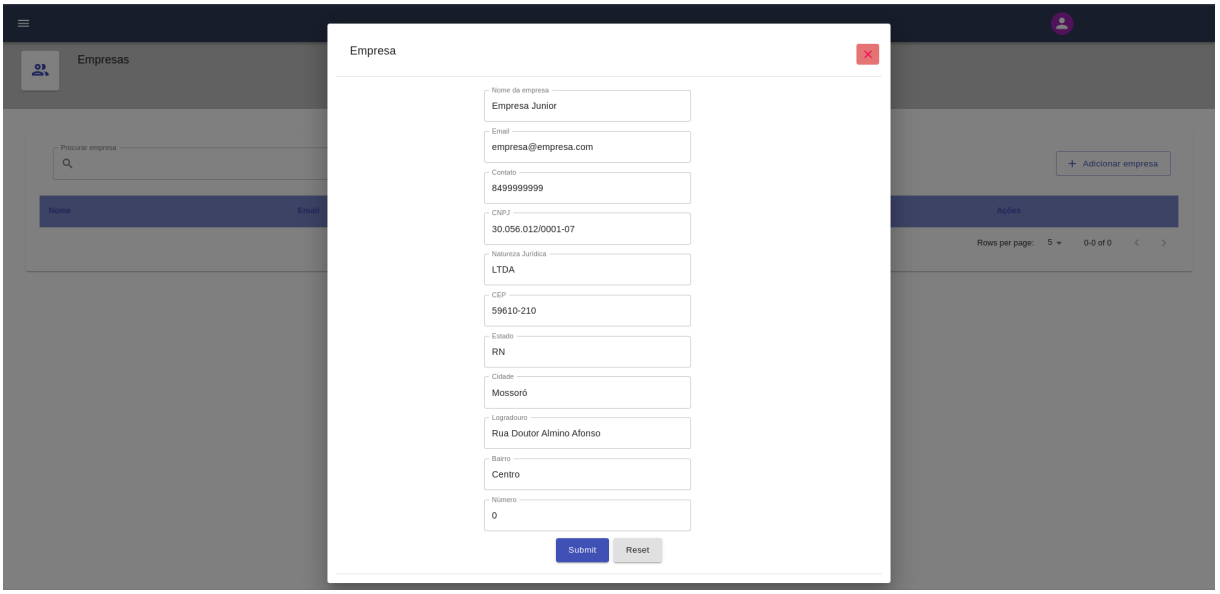
Figura 9 - Processo para criar curso.

```
export async function insertCourse(data: any) {

    const { name, shift, degree, modality, academic_unit_id } = data

    const obj = {
        name: name,
        shift: shift,
        degree: degree,
        modality: modality,
        academic_unit_id: academic_unit_id
    }

    return await api.post('/courses/', obj)
}
```

Fonte: Autoria própria.

A **figura 10** mostra o formulário para criação de uma empresa.

Figura 10 - Formulário para criar empresa.



Fonte: Autoria própria.

A **figura 11** mostra o processo para criar uma empresa. É enviada uma requisição com o método POST com as informações necessárias, como: nome, email, telefone, CNPJ, natureza jurídica, cep, estado, cidade, logradouro, bairro e número.

Figura 11 - Processo para criar empresa.

```
export async function insertCompany(data: any) {
    const { name, cnpj, business_nature, email, phone, company_address } = data
    const { cep, street, state, city, district, home_number } = company_address

    const company = {
        name: name,
        cnpj: cnpj.replace(/[^0-9]/g, ''),
        business_nature: business_nature,
        email: email,
        phone: phone,
        address: {
            cep: cep.replace('-', ''),
            street: street,
            state: state,
            city: city,
            district: district,
            home_number: parseInt(home_number)
        }
    }

    return await api.post('/companies/', company)
}
```

Fonte: Autoria própria.

4.1 TECNOLOGIAS UTILIZADAS

Para o desenvolvimento do *Front-end* foi empregada a linguagem TypeScript (TYPESCRIPT, 2022) juntamente com a biblioteca React que é usada para criar interfaces de usuário (REACT, 2022), o ambiente de execução NodeJs (NODEJS, 2022) e Docker (DOCKER, 2022) para a criação do ambiente de desenvolvimento. Além dos recursos citados anteriormente, a biblioteca Material-UI (MUI, 2022) foi usada para a criação e estilização dos componentes que são utilizados nas páginas.

O envio de requisições por parte do *Front-end* e consumo da *API* do SIGEE foi feito utilizando Axios (AXIOS, 2022), uma biblioteca baseada em *Promises* que provê um cliente http para o navegador e também para o NodeJS. Por fim, para a implantação do sistema, contamos com a *Microsoft Azure*, uma plataforma destinada à criação, execução e gerência de aplicativos e serviços em nuvem (AZURE, 2022).

4.2 ESTRUTURA DO PROJETO

Estruturar uma aplicação é uma tarefa que se bem feita auxiliará não somente a tornar a aplicação mais legível e intuitiva para que outras pessoas a entendam, mas também para facilitar a posterior manutenção ou extensão da mesma, se for o caso. Pensando nisso, o *Front-end* foi dividido nos seguinte diretórios:

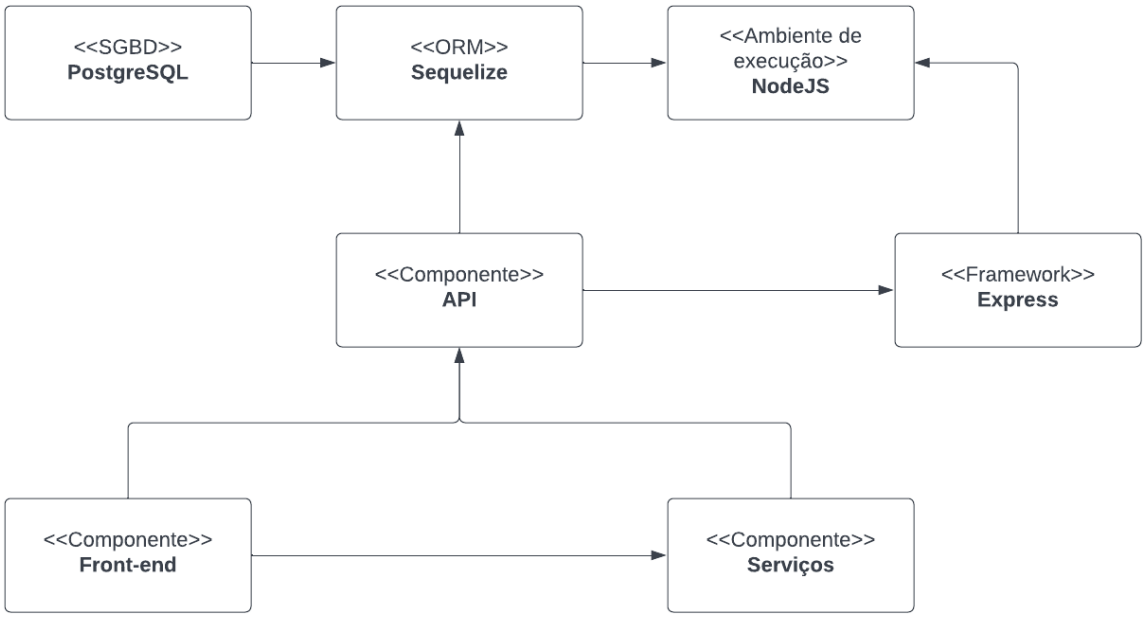
- **Components:** Nesse diretório está contido todos os componentes do *Front-end*. Componentes são partes independentes e reutilizáveis, tratando cada parte da aplicação como um container isolado, sem precisar de

dependências externas. Podemos usar como exemplo de componentes: botão, *header*, *sidebar*, *check box*, *input*, etc…

- **Context:** Tem todos os contextos necessários da aplicação. Os contextos são usados para compartilhar dados globais para uma determinada árvore de componentes. Por exemplo, o contexto de autenticação autoriza o usuário a realizar operações no sistema.
- **Pages:** Contém todas as páginas da aplicação. As páginas são formadas por componentes. Podemos dizer que uma página é um amontoado de componentes estruturados para formar um todo.
- **Services:** Esse diretório contém todos os *scripts* que farão a comunicação entre o *Front-end* e a *API*. Os *services* ou serviços são responsáveis pela lógica de negócio na parte do *Front-end*.

Abaixo, temos a **figura 12** que ilustra o diagrama de componentes do sistema. Esse diagrama mostra como estão organizadas as partes do sistema. O *Front-end*, como dito, é responsável por interagir com o usuário para fazer as operações desejadas. Após a inserção, os dados são enviados para os *services* que tratam esses dados e fazem uma requisição para a *API,* que por sua vez utiliza o framework *Express* (EXPRESS, 2022) para validar os dados e realizar a lógica de negócio da *API*. Para fazer a conexão com o PostgreSQL(POSTGRESQL, 2022), o banco de dados utilizado na aplicação, persistir e buscar todas as informações foi utilizado o *ORM* Sequelize (SEQUELIZE, 2022). O NodeJS (NODEJS, 2022) serviu como ambiente de execução da aplicação, permitindo à aplicação funcionar sem necessitar de um navegador.

Figura 12 - Diagrama de componentes.



Fonte: Autoria própria.

5 RESULTADOS

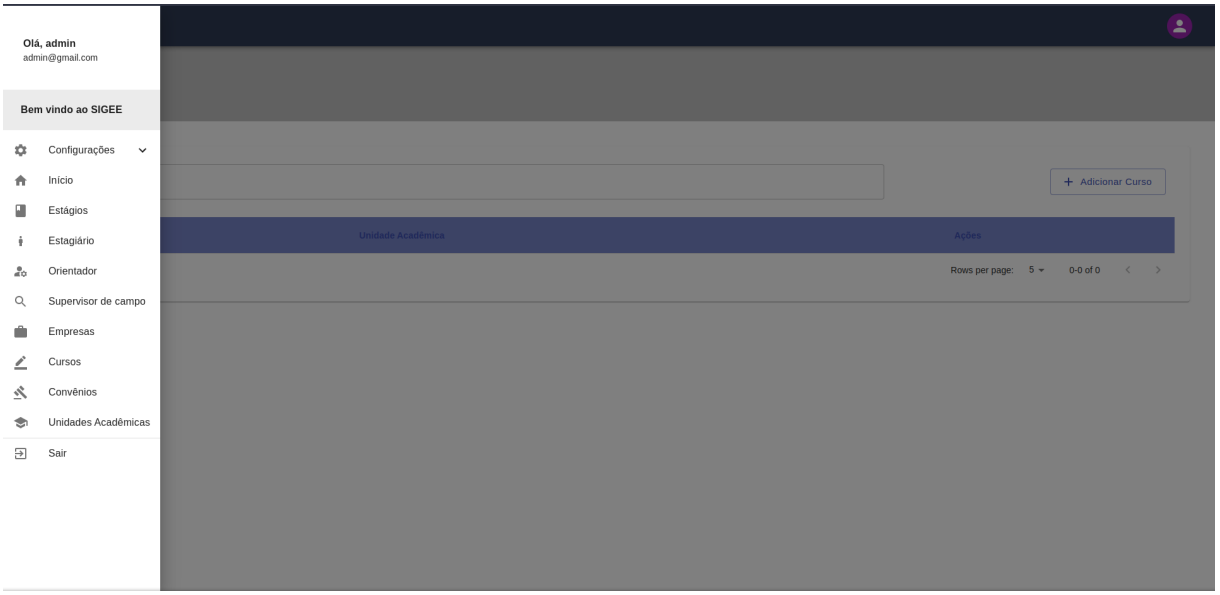
A presente seção mostra testes realizados a fim de validar as funcionalidades da aplicação. A **figura 13** mostra a página de login, onde o usuário coloca um email e senha, caso sejam válidos, é feita a autenticação do mesmo no sistema.

Figura 13 - Página de login

Fonte: Autoria própria.

Com o endereço de email e senha validados e o usuário autenticado no sistema, é permitida a navegação e a realização de operações, como mostra a **figura 14**.

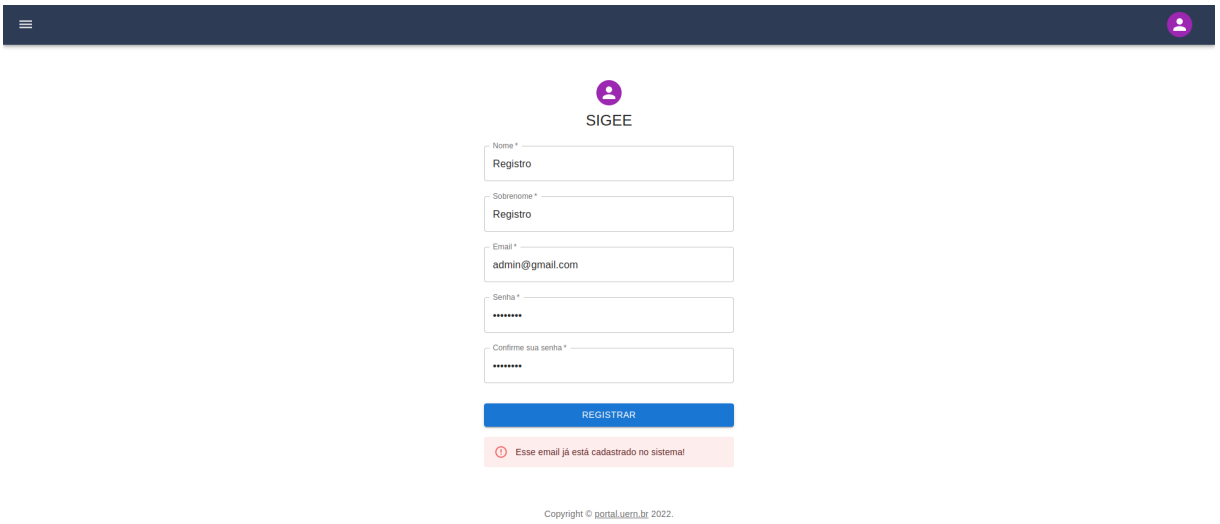
Figura 14 - Usuário validado.



Fonte: Autoria própria.

A **figura 15** ilustra o erro obtido ao tentar registrar um novo administrador com um endereço de email já cadastrado no sistema.

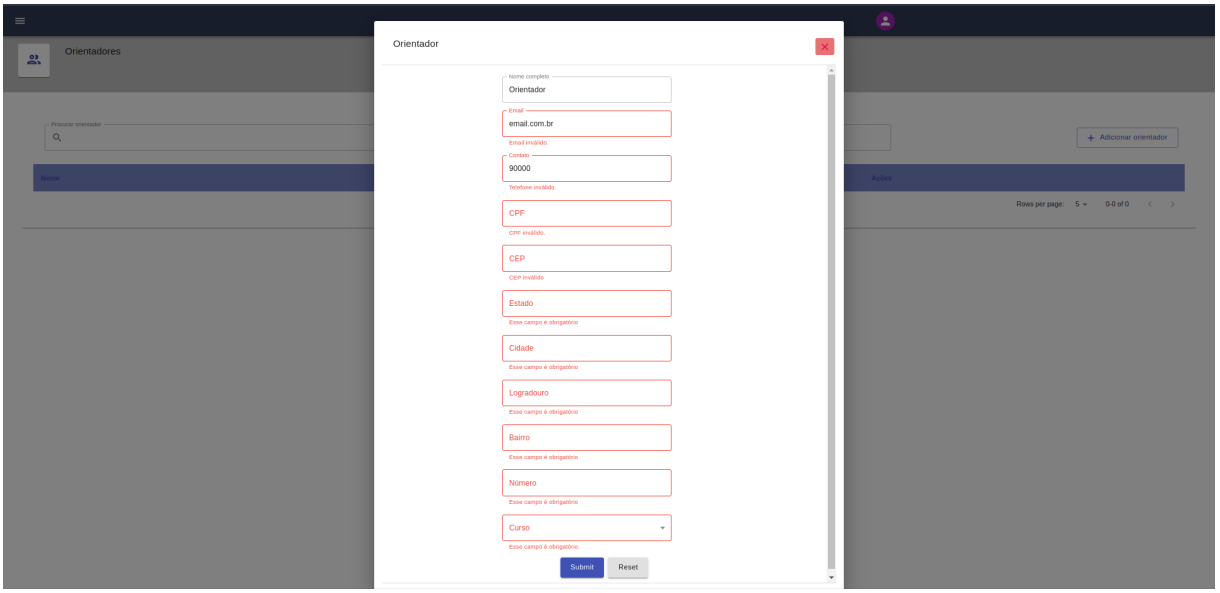
Figura 15 - Email já cadastrado no sistema.



Fonte: Autoria própria.

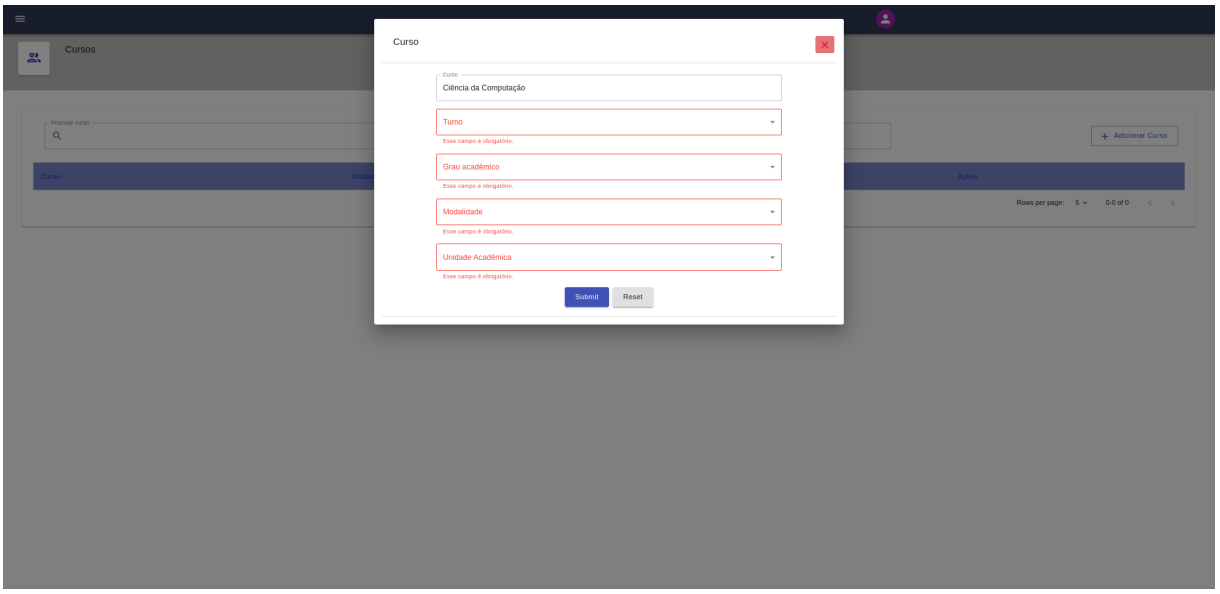
Nas **figuras 16 e 17** é demonstrada a validação dos campos preenchidos pelo usuário. Caso haja alguma informação errada ou algum campo seja deixado em branco, mensagens de erro são apresentadas na tela.

Figura 16 - Erro ao registrar orientador.



Fonte: Autoria própria.

Figura 17 - Erro ao registrar curso.



Fonte: Autoria própria.

Toda a aplicação foi implantada na plataforma de nuvem da Microsoft, a Azure, a fim de torná-la disponível para os usuários interessados.

5.1 Conclusão

Levando em consideração os fatos mencionados, o objetivo deste trabalho foi a construção de um Front-end capaz de consumir a API do SIGEE de forma que o administrador do sistema pudesse efetuar operações de inserção, deleção, atualização e visualização dos dados no sistema. O Front-end foi construído utilizando a linguagem Typescript, juntamente com a biblioteca React e atingiu o resultado esperado.

Foi-se apresentado o sistema ao setor de estágios da Pró-Reitoria de Assuntos Estudantis, para que fosse feita uma avaliação sobre o software e um *feedback* sobre pontos de melhoria do mesmo. O sistema foi validado enviando e recuperando dados de maneira segura, atingindo os objetivos esperados.

6 REFERÊNCIAS

ABRES. Estatísticas - Abres. [S. l.], 2021. Disponível em: https://abres.org.br/estatisticas/#:~:text=De%20acordo%20com%20pesquisa%20realizada,o%20ensino%20m%C3%A9dio%20e%20t%C3%A9cnico. Acesso em: 13 de março de 2022.

AXIOS. 2022. Disponível em: https://axios-http.com/. Acesso em: 19 de março de 2022.

AZURE. About. 2022. Disponível em: https://azure.microsoft.com/pt-br/overview/what-is-azure. Acesso em: 18 de março de 2022.

BRASIL. Lei nº 11.788, de 25 de setembro de 2008. Dispõe sobre o estágio de estudantes; altera a redação do art. 428 da Consolidação das Leis do Trabalho – CLT, aprovada pelo Decreto-Lei no 5.452, de 1o de maio de 1943, e a Lei no 9.394, de 20 de dezembro de 1996; revoga as Leis nos 6.494, de 7 de dezembro de 1977, e 8.859, de 23 de março de 1994, o parágrafo único do art. 82 da Lei no 9.394, de 20 de dezembro de 1996, e o art. 6o da Medida Provisória no 2.164-41, de 24 de agosto de 2001; e dá outras providências. Brasília, DF, 25 set. 2008. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2007-2010/2008/lei/l11788.htm. Acesso em: 12 de março de 2022.

DOCKER, Inc. Why docker. 2022. Disponível em: https://www.docker.com/why-docker/. Acesso em: 16 de março de 2022.

EXPRESS. 2022. Disponível em: https://expressjs.com/pt-br/. Acesso em: 12 de abril de 2022.

MUI. About us. 2022. Disponível em: https://mui.com/pt/about. Acesso em: 16 de março de 2022.

OPENJS, Foundation. About Node.js. 2022. Disponível em: https://nodejs.org/en/about/. Acesso em: 12 de abril de 2022.

POSTGRESQL, Global Development Group. About. 2022. Disponível em: https://www.postgresql.org/about/. Acesso em: 12 de abril de 2022.

SEQUELIZE. 2022. Disponível em: https://sequelize.org/. Acesso em: 12 de abril de 2022.

REACT, Meta Platforms, Inc. 2022. Disponível em: https://pt-br.reactjs.org/. Acesso em: 15 de março de 2022.

SOMMERVILLE, Ian. Engenharia de Software. 9.ed. São Paulo, Brasil: Pearson, 2011.

TRELLO, Atlassian. About. 2022. Disponível em: https://trello.com/about. Acesso em: 18 de março de 2022.

TYPESCRIPT. 2022. Disponível em: https://www.typescriptlang.org/ Acesso em: 15 de março de 2022.