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

Gabriel Nascimento Jales

API SIGEE: UMA API REST PARA GESTÃO DE ESTÁGIOS

MOSSORÓ - RN

2022

Gabriel Nascimento Jales

API SIGEE: UMA API REST PARA GESTÃO DE ESTÁGIOS

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

GABRIEL NASCIMENTO JALES

API SIGEE: UMA API REST PARA GESTÃO DE ESTÁGIOS

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: 28/09/2022

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. Francisco Chagas de Lima Júnior
Universidade do Estado do Rio Grande do Norte - UERN

SUMÁRIO

1 INTRODUÇÃO

O estágio é um ato educativo supervisionado, desenvolvido no ambiente de trabalho, que visa a preparação do estudante para o exercício do trabalho produtivo. Tem por objetivo proporcionar ao estudante a aquisição de competências próprias da atividade profissional e a contextualização curricular.

Para a atividade de estágio, há a existência de pelo menos três agentes envolvidos: a) o estudante, que é o aprendiz que se propõe a realização do ato educativo supervisionado; b) a parte concedente, que é a instituição/empresa que recebe o estudante em seu espaço profissional para proporcionar atividades compatíveis com o desenvolvimento educacional do estudante e c) a instituição de ensino, comumente chamada de proponente, é a instituição de origem do estudante e que estabelece no projeto pedagógico do curso a atividade de estágio. Pode haver ainda um quarto agente envolvido, o agente de integração, que realiza uma ação auxiliar na construção do relacionamento entre as outras três partes.

Com esta descrição, percebe-se que além do próprio processo pedagógico próprio do estágio, existe um processo administrativo que regula e rege a relação entre estudantes, instituições de ensino e concedentes. Para gerenciar a regra de negócios da aplicação e permitir que diversos sistemas acessem as mesmas informações de maneira uniforme, padronizada, segura e com uma alta disponibilidade, optou-se por desenvolver o back-end no formato de *API* (*Application Programming Interface*) *REST* (*Representational State Transfer*). Então, o Sistema de Gestão de Estágios (SIGEE) tem como objetivo principal fornecer um ambiente seguro para gerir e administrar os dados de estágios de uma instituição.

2 OBJETIVOS

O objetivo geral deste trabalho é abordar sobre a construção do SIGEE na perspectiva do back-end, ou seja: apresentar a *API* do sistema, uma *API* que segue o modelo *REST*, possui um fluxo de autenticação baseado em *JWT* (*JSON Web Token*) e possibilita o desenvolvimento de aplicações para gestão de estágios, como aplicações WEB ou *mobile*, sem que seja necessário recriar a regra de negócio múltiplas vezes. Ademais, podem ser citados como objetivos específicos: a utilização da metodologia ágil *Scrum* durante o desenvolvimento do projeto, a realização de testes para validação da *API* por meio da ferramenta Postman (POSTMAN, 2022) e a integração com o SendGrid (SENDGRID, 2022), serviço para envio de e-mails transacionais.

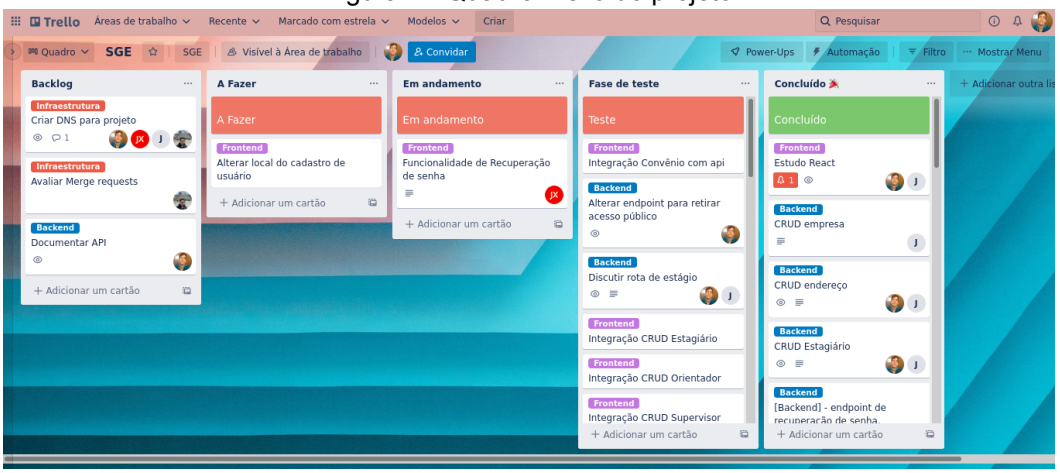
3 METODOLOGIA

A metodologia utilizada para o desenvolvimento do SIGEE foi a abordagem ágil *Scrum*, que possui três fases: planejamento geral, ciclo de *sprints* e o encerramento do projeto (SOMMERVILLE, 2011). Na primeira fase foram estabelecidos os principais objetivos do projeto, as tecnologias que seriam utilizadas, o *backlog* do sistema listando todo o trabalho a ser feito e a duração fixa dos ciclos de *sprints*, definida em quinze dias. Na segunda fase cada membro da equipe trabalhava em uma nova funcionalidade para ser incrementada no sistema, com reuniões diárias para analisar os progressos e dificuldades, além de reuniões a cada quinze dias para finalizar a *sprint*. Por fim, na última etapa encerrou-se o desenvolvimento do projeto e foram feitas avaliações sobre as lições aprendidas no projeto.

Para facilitar o gerenciamento e divisão das tarefas que precisavam ser feitas e manter a equipe alinhada, foi utilizada a ferramenta web de gerenciamento de projetos Trello. No Trello os projetos são organizados em quadros, nesses quadros são definidas listas com todas as tarefas do projeto e por sua vez, essas tarefas podem ser atribuídas a membros do quadro. A **Figura 1** mostra o quadro do projeto,

com as listas de cada etapa (Backlog, A fazer, Em andamento, Fase de teste e Concluído) e as tarefas com os membros responsáveis por realizá-las.

Figura 1 - Quadro Trello do projeto

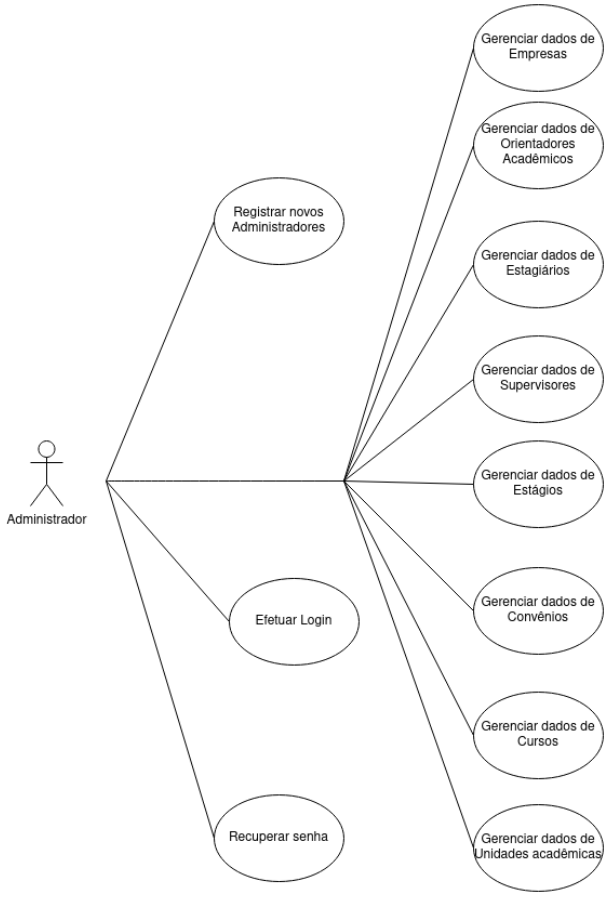


Fonte: Autoria própria

4 DESCRIÇÃO DO SISTEMA

O SIGEE trata-se de um sistema WEB para gestão de dados relacionados a estágios, onde um usuário cadastrado e que foi previamente autenticado pode realizar operações de registro, consulta, atualização e remoção de informações como dados de estágios, estagiários, unidades acadêmicas, cursos, orientadores acadêmicos, supervisores de estágio, empresas e convênios, como descrito no diagrama de caso de uso do sistema, ilustrado na **Figura 2**.

Figura 2 - Diagrama de caso de uso



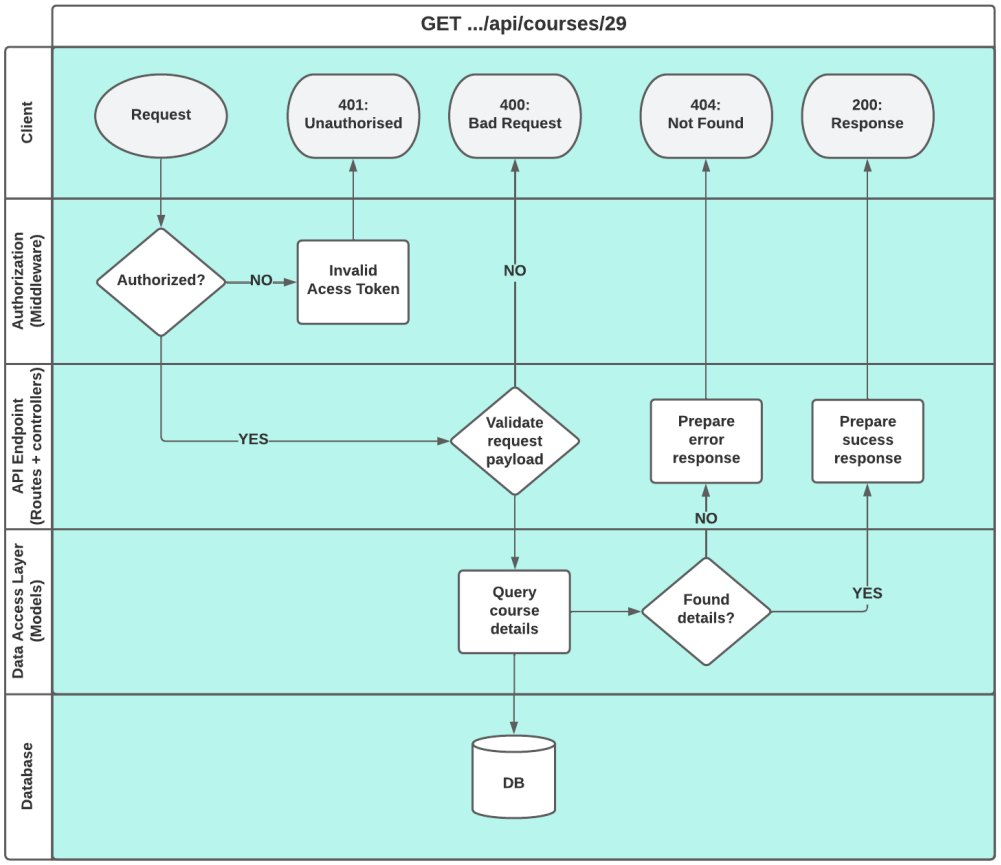
Fonte: Autoria própria

O sistema pode ser dividido em duas aplicações principais: o Front-end é a parte responsável por interagir diretamente com o usuário final do sistema e a API, por sua vez, é a camada que será abordada neste trabalho e possui a responsabilidade de gerenciar a regra de negócio e os dados do sistema. A API do SIGEE implementa a arquitetura *REST*, termo que foi introduzido pelo cientista da computação Roy Fielding e pode ser definida como um conjunto de restrições arquiteturais utilizadas para desenvolver serviços web. Quando um cliente faz uma requisição (*request*) para a *API*, a mesma fornece uma resposta (*response*) via *HTTP* (*HyperText Transfer Protocol*) com uma representação do estado do recurso que foi solicitado em um formato específico, como por exemplo o *JSON* (*Javascript Object Notation*), que foi o formato escolhido para a nossa API e é o mais utilizado no mercado por ser de fácil compreensão para humanos. Com relação às requisições, elas são compostas por três informações principais:

- **Endpoint:** é a URL/caminho que o serviço receberá a requisição do cliente;
- **Método:** indica uma ação a ser executada para um determinado recurso. Os mais utilizados são o *GET*, *POST*, *PUT* e *DELETE*. O método *GET* obtém informações de um recurso, o *POST* cria um recurso, o *PUT* atualiza as informações de um recurso e o *DELETE* exclui informações de um recurso;
- **Corpo:** informações adicionais enviadas para o servidor, como dados de um formulário;

A **Figura 3** ilustra o fluxograma da *API*, utilizando como exemplo uma requisição do tipo *GET* na rota *…/api/courses/29*.

Figura 3 - Fluxograma da API (requisição GET na rota …/api/courses/29)

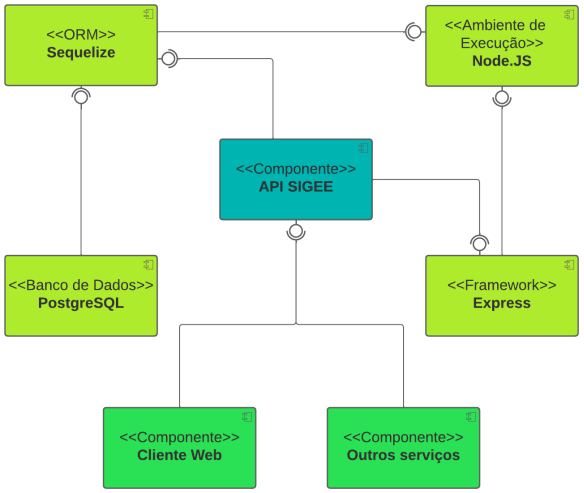


Fonte: Autoria própria

4.1 TECNOLOGIAS UTILIZADAS

Para desenvolver a *API* do SIGEE foi utilizada a linguagem Javascript (JAVASCRIPT, 2022) e o ambiente de execução Node.js (NODEJS, 2022), que permite a criação de aplicações Javascript sem depender de um navegador para executá-las. Em conjunto dessas duas tecnologias, também foram utilizados: o framework para construção de servidores web Express.js (EXPRESS, 2022), o *ORM* (*Object Relational Mapper*) baseado em *Promises* Sequelize (SEQUELIZE, 2022), o banco de dados relacional PostgreSQL (POSTGRESQL, 2022), Docker (DOCKER, 2022) para criação do ambiente de desenvolvimento e para validação da aplicação por meio de testes, foi utilizado o Postman (POSTMAN, 2022). A **Figura 4** abaixo retrata o diagrama de componentes da API com algumas das tecnologias citadas.

Figura 4 - Diagrama de componentes da API SIGEE



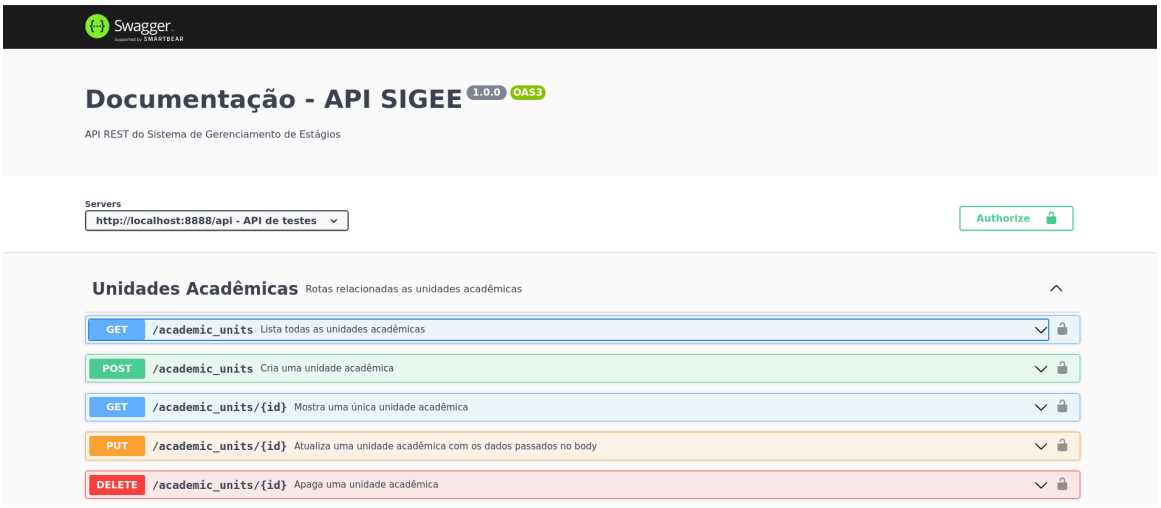
Fonte: Autoria própria

4.2 ESTRUTURA DE CAMADAS/PASTAS DO PROJETO

Um dos processos mais importantes durante o desenvolvimento de um software é a definição da estrutura de camadas e diretórios do mesmo. Quanto mais intuitiva, bem separada e organizada for a estrutura do projeto, mais simples será o processo de manutenção e testagem do software. Pensando nesses pontos, a *API* do SIGEE possui uma estrutura de diretórios simples e elementos com papéis bem definidos, como descrito abaixo:

- **Config:** Contém os arquivos de configurações gerais do projeto, como configuração de conexão com o banco de dados, os tipos de erros da aplicação e as funções responsáveis pelo envio de e-mails transacionais utilizando o serviço Sendgrid. A funcionalidade de envio de e-mails será abordada de forma mais detalhada, posteriormente, no **Tópico 4.3**;
- **Controllers:** Camada responsável por conter toda a lógica da aplicação. Pode-se afirmar que existem dois tipos de controllers na *API*, o *controller* de autenticação é responsável pelo registro e login de usuário, além da funcionalidade de recuperar senha. Já os demais *controllers* são utilizados para manipulação de dados com as funções *show, list, create, update* e *delete*. As funções list e create, respectivamente, são responsáveis por listar todos os recursos de uma rota e criar um recurso com os dados fornecidos. Já as funções show, update e delete necessitam de um ID para mostrar, atualizar ou apagar um determinado recurso.
- **Database:** Contém arquivos para manipulação no banco de dados, como por exemplo as *migrations* e as *seeds*. *Migrations,* ou migrações, são alterações diretas nas tabelas do banco de dados, além de servirem como um histórico dessas alterações. As *seeds* por sua vez, são dados pré determinados que serão inseridos no banco de dados da aplicação. O SIGEE atualmente só possui uma única *seed*, ela é usada para gerar um usuário inicial que será responsável por cadastrar os demais.
- **Docs:** Um ponto muito importante para se levar em conta no desenvolvimento de uma *API* é a sua documentação, ela será uma ferramenta muito útil para as pessoas que irão utilizar a *API* não desperdicem tempo tentando desvendar como a mesma funciona. Para facilitar e automatizar o processo de desenvolvimento da documentação, foi utilizado o *framework* de descrição e visualização de serviços *REST*, *Swagger* (SWAGGER, 2022). Como está ilustrado na **Figura 5,** o *Swagger* possui uma *UI* (*User Interface* ou Interface de Usuário) muito amigável que possibilita uma interação mais intuitiva entre os desenvolvedores e a *API*, de tal maneira que não seja necessário ter algum conhecimento da implementação da aplicação em si.

Figura 5 - Ilustração da documentação da API do SIGEE (Rotas de unidades acadêmicas)



Fonte: Autoria própria

- **Middlewares:** Basicamente são funções que possuem acesso ao objeto de requisição, o objeto de resposta e o próximo middleware a ser chamado. Até o momento a *API* do SIGEE possui dois *middlewares*, o *loginRequired* é responsável por verificar (por meio do *JWT*) se o cliente que está tentando acessar determinada rota está logado. O segundo middleware é o *handleErrors,* usado para tratar todos os erros que venham a ocorrer no sistema e evitar que a aplicação simplesmente pare de responder.
- **Models:** São classes que representam os dados da nossa aplicação e como eles se comportam, de acordo com a lógica e as regras de negócios do sistema.
- **Routes:** Rotas, também chamadas de *endpoints*, são caminhos que receberão requisições de clientes, ou outras aplicações, e devolveram respostas com algum tipo de informação.

4.3 ENVIO DE E-MAILS TRANSACIONAIS

Com o intuito de facilitar a manutenção e a escalabilidade do código, a funcionalidade de envio de e-mails foi dividida em quatro principais funções. Essas funções fazem uso do pacote "*@sendgrid/mail*" e de uma *API key* (chave API),

obtida ao realizar cadastro na plataforma Sendgrid, para possibilitar o uso da mesma. As quatro funções mencionadas são:

- ***sucessCreateUserMessage*:** Função com mensagem que confirma a criação do usuário no sistema após a realização do cadastro, ilustrada no **Código 1**;

Código 1 - Função sucessCreateUserMessage

Função:	sucessCreateUserMessage

```
const sgMail = require('@sendgrid/mail');
sgMail.setApiKey(process.env.MAIL_KEY);

const sucessCreateUserMessage = (email, name) => {
 const msg = {
   to: email,
   from: process.env.MAIL_SENDER,
   subject: 'Cadastro realizado',
   text: `Olá ${name}!\nSua conta no SIGEE está pronta para ser
usada.`,
 }
 return msg;
};
```

Fonte: Autoria própria

- ***fpMessage*:** Contém uma mensagem com o link para que o usuário possa recuperar a sua senha, esse link contém um token que posteriormente é utilizado para verificar as informações do usuário. Função ilustrada no **Código 2**;

Código 2 - Função fpMessage

Função:	fpMessage

```
// Código anterior omitido
const fpMessage = (email, token) => {
 const msg = {
   to: email,
   from: process.env.MAIL_SENDER,
   subject: 'Recuperação de senha',
   text: `Para recuperar a sua senha, por favor clique no link
```

```
abaixo.\n\nhttps://${process.env.DOMAIN}/auth/reset-password?toke
n=${encodeURIComponent(token)}&email=${email}`,
 }
 return msg;
};
```

Fonte: Autoria própria

- ***sucessFpMessage*:** Mensagem confirmando que a senha foi alterada com sucesso, ilustrada no **Código 3**;

Código 3 - Função sucessFpMessage

Função:	sucessFpMessage

```
// Código anterior omitido
const sucessFpMessage = (email) => {
 const msg = {
   to: email,
   from: process.env.MAIL_SENDER,
   subject: 'Senha alterada',
   text: `Olá! Sua senha foi alterada com sucesso!`,
 }
 return msg;
};
```

Fonte: Autoria própria

- **sendMail**: Função principal que é responsável por enviar de fato o e-mail. Utiliza a classe *MailService* do pacote "*@sendgrid/mail*", que foi importada como *sgMail no código*, e recebe as demais funções como parâmetro. A função está representada no **Código 4**;

Código 4 - Função sendMail

Função:	sendMail

```
// Código anterior omitido
const sendMail = async (msg) => {
 try {
   await sgMail.send(msg);
   console.log('Email enviado!');
```

```
  } catch (err) {
    console.error('Error sending test email');
    console.error(err);
    if (err.response) {
      console.error(err.response.body)
    }
  }
};
```

Fonte: Autoria própria

5 RESULTADOS

Na **Tabela 1** estão listadas todas as rotas do sistema, juntamente do método *HTTP* utilizado e uma descrição. As rotas destacadas são privadas e o usuário precisa estar, obrigatoriamente, autenticado.

Tabela 1 - Todas as rotas da API

Método HTTP	Rota	Descrição
GET	…/api/	Rota inicial
GET	…/api/docs	Documentação
POST	…/api/auth/register	Cria um usuário
POST	…/api/auth/login	Usuário realiza login no sistema (autenticado)
POST	…/api/auth/forgot-password	Usuário fornece um e-mail para recuperar senha
GET	…/api/auth/reset-password	Link que o usuário recebe no e-mail (e-mail do usuário e token estão contidos na URL)
POST	…/api/auth/reset-password	Rota onde a senha é efetivamente recuperada
PUT	…/api/users/	Atualizar informações de um usuário logado
DELETE	…/api/users/	Usuário logado deleta sua conta

GET	…/api/interns/	Retorna todos os estagiários
POST	…/api/interns/	Cria um estagiário
GET	…/api/interns/:id	Retorna um estagiário específico com o ID fornecido
PUT	…/api/interns/:id	Atualiza um estagiário específico com o ID fornecido
DELETE	…/api/interns/:id	Deleta um estagiário específico com o ID fornecido
GET	…/api/internships/	Retorna todos os estágios
POST	…/api/internships/	Cria um estágio
GET	…/api/internships/:id	Retorna um estágio específico com o ID fornecido
PUT	…/api/internships/:id	Atualiza um estagiário específico com o ID fornecido
DELETE	…/api/internships/:id	Deleta um estágio específico com o ID fornecido
GET	…/api/companies/	Retorna todas as empresas
POST	…/api/companies/	Cria uma empresa
GET	…/api/companies/:id	Retorna uma empresa específica com o ID fornecido
PUT	…/api/companies/:id	Atualiza uma empresa específica com o ID fornecido
DELETE	…/api/companies/:id	Deleta uma empresa específica com o ID fornecido
GET	…/api/agreements/	Retorna todos os convênios
POST	…/api/agreements/	Cria um convênio
GET	…/api/agreements/:id	Retorna um convênios específico com o ID fornecido
PUT	…/api/agreements/:id	Atualiza um convênios específico com o ID fornecido
DELETE	…/api/agreements/:id	Deleta um convênios específico com o ID fornecido
GET	…/api/advisors/	Retorna todos os orientadores
POST	…/api/advisors/	Cria um orientador

GET	…/api/advisors/:id	Retorna um orientador específico com o ID fornecido
PUT	…/api/advisors/:id	Atualiza um orientador específico com o ID fornecido
DELETE	…/api/advisors/:id	Deleta um orientador específico com o ID fornecido
GET	…/api/supervisors/	Retorna todos os supervisores
POST	…/api/supervisors/	Cria um supervisor
GET	…/api/supervisors/:id	Retorna um supervisor específico com o ID fornecido
PUT	…/api/supervisors/:id	Atualiza um supervisor específico com o ID fornecido
DELETE	…/api/supervisors/:id	Deleta um supervisor específico com o ID fornecido
GET	…/api/academic_units/	Retorna todas as unidades acadêmicas
POST	…/api/academic_units/	Cria uma unidade acadêmica
GET	…/api/academic_units/:id	Retorna uma unidade acadêmica específica com o ID fornecido
PUT	…/api/academic_units/:id	Atualiza uma unidade acadêmica específica com o ID fornecido
DELETE	…/api/academic_units/:id	Deleta uma unidade acadêmica específica com o ID fornecido
GET	…/api/courses/	Retorna todos os cursos
POST	…/api/courses/	Cria um curso
GET	…/api/courses/:id	Retorna um curso específico com o ID fornecido
PUT	…/api/courses/:id	Atualiza um curso específico com o ID fornecido
DELETE	…/api/courses/:id	Deleta um curso específico com o ID fornecido

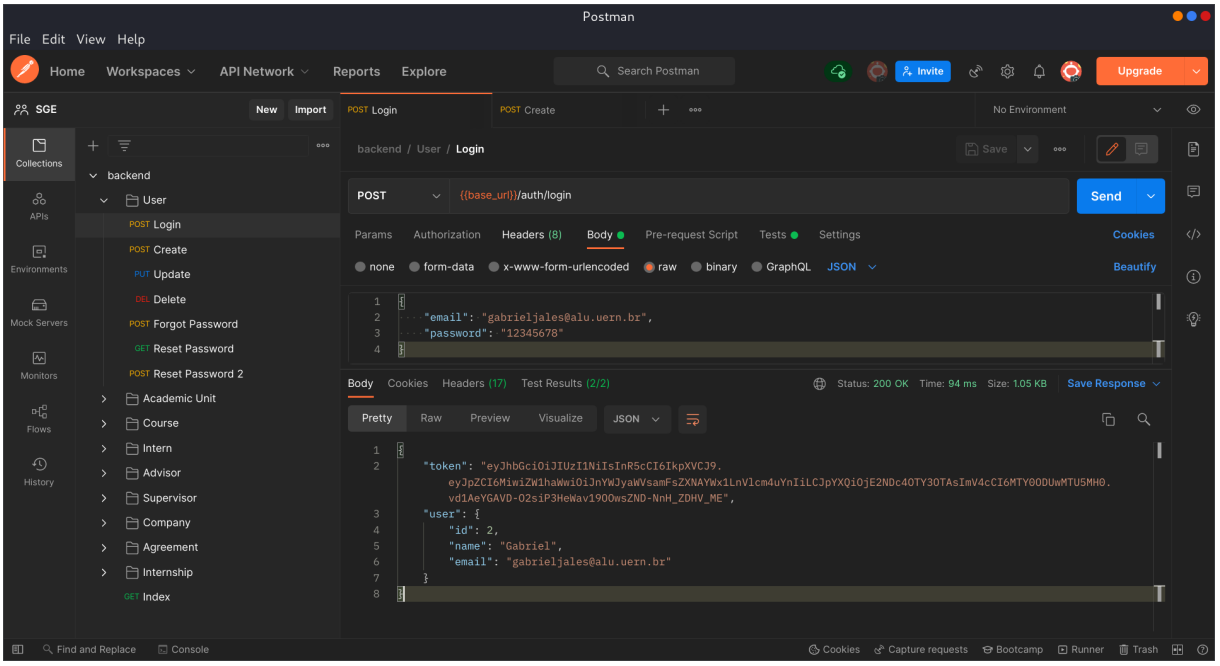
Fonte: Autoria própria

5.1 TESTES

Como mencionado anteriormente, o *API Cliente Postman* foi a ferramenta utilizada para realização e validação dos testes da *API*. Essa ferramenta gratuita possibilita que os desenvolvedores consigam criar e salvar solicitações *HTTP* e HTTPS, além de visualizar as suas respostas. O objetivo primordial dos testes foi verificar o comportamento dos *Endpoints* da *API*, validando se estavam com os métodos e valores de retorno corretos, além de testar como a aplicação lidava com situações de erros e se os códigos HTTP retornados estavam certos.

A **Figura 6** ilustra o funcionamento do envio de uma requisição do tipo POST para obtenção do token de autenticação JWT na rota */auth/login*, fornecendo os parâmetros necessários no corpo (email e senha). Se os dados fornecidos forem válidos, ou seja, o usuário de fato estiver cadastrado no sistema, a resposta da API conterá um token e o código de status 200 indicando sucesso.

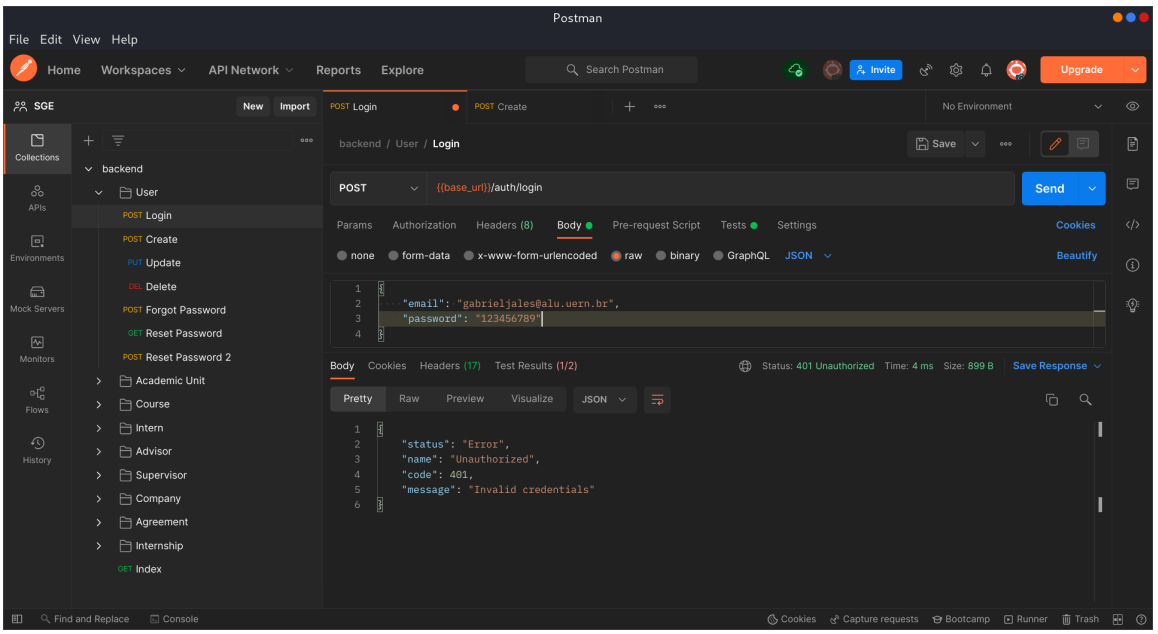
Figura 6 - Demonstração de login no Postman (sucesso)



Fonte: Autoria própria

Caso as credenciais fornecidas sejam inválidas, a *API* retornará um objeto com status igual a “*error*”, com o nome do tipo de erro, o código de status *HTTP* e uma mensagem de erro, como ilustrado na **Figura 7** abaixo.

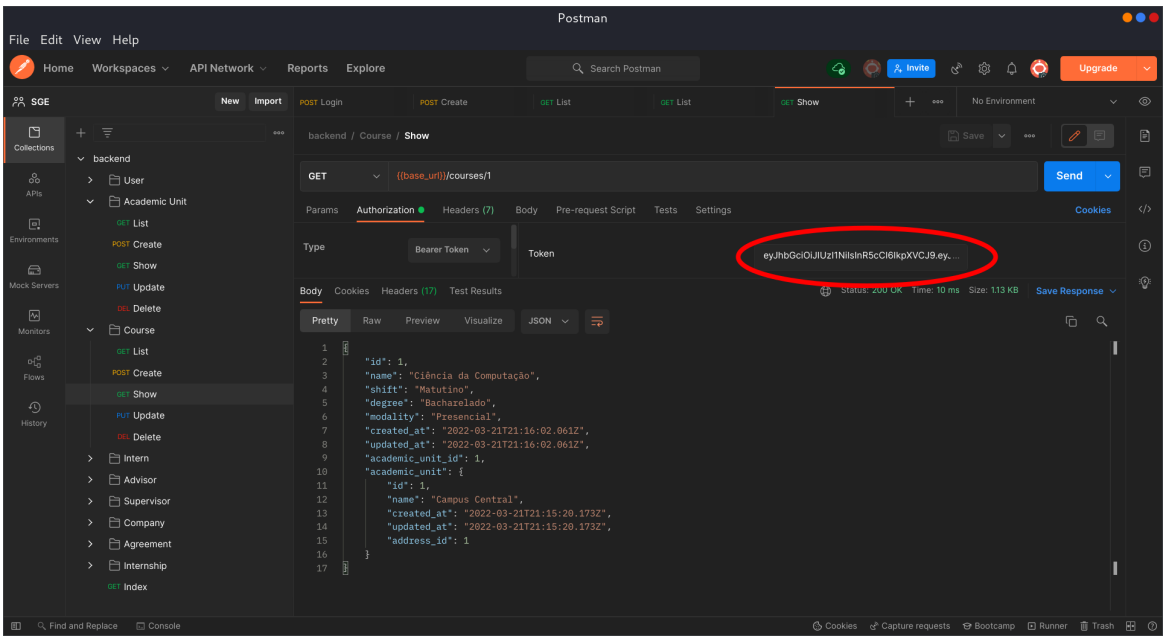
Figura 7 - Demonstração de login no Postman (erro)



Fonte: Autoria própria

Após realizar o login e obter o token o usuário possui permissão para realizar as operações do sistema enquanto esse token for válido, esse tempo é definido em uma variável de ambiente chamada “*TOKEN_EXPIRATION*” por questões de segurança. A **Figura 8** demonstra o processo de visualizar as informações de um determinado curso, perceba que o token (circulado em vermelho) recebido anteriormente no processo de login está sendo utilizado para fazer a requisição.

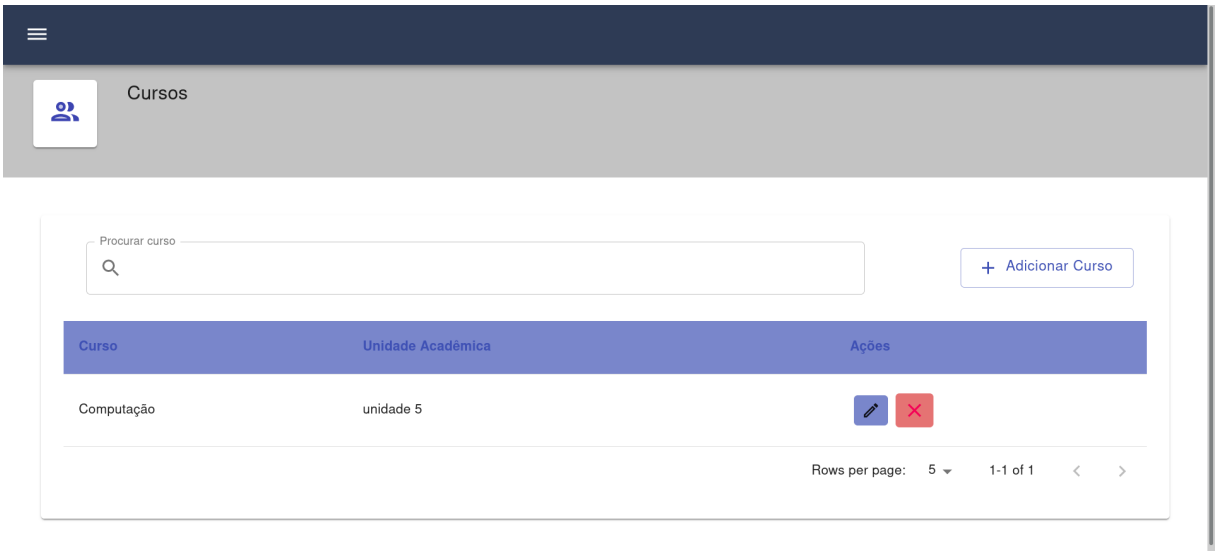
Figura 8 - Visualizando os dados de um curso no Postman (sucesso)



Fonte: Autoria própria

Em consideração ao que foi exposto, é importante mencionar que a primeira versão da *API*[1] passou nos testes da maneira esperada e fornece os recursos necessários para a criação de aplicações voltadas a gestão de dados de estágios. As **Figuras 9** e **10** ilustram um exemplo de uma aplicação feita em *React* (REACT, 2022) que utiliza a *API* como serviço e, juntos, integram o sistema como um todo.

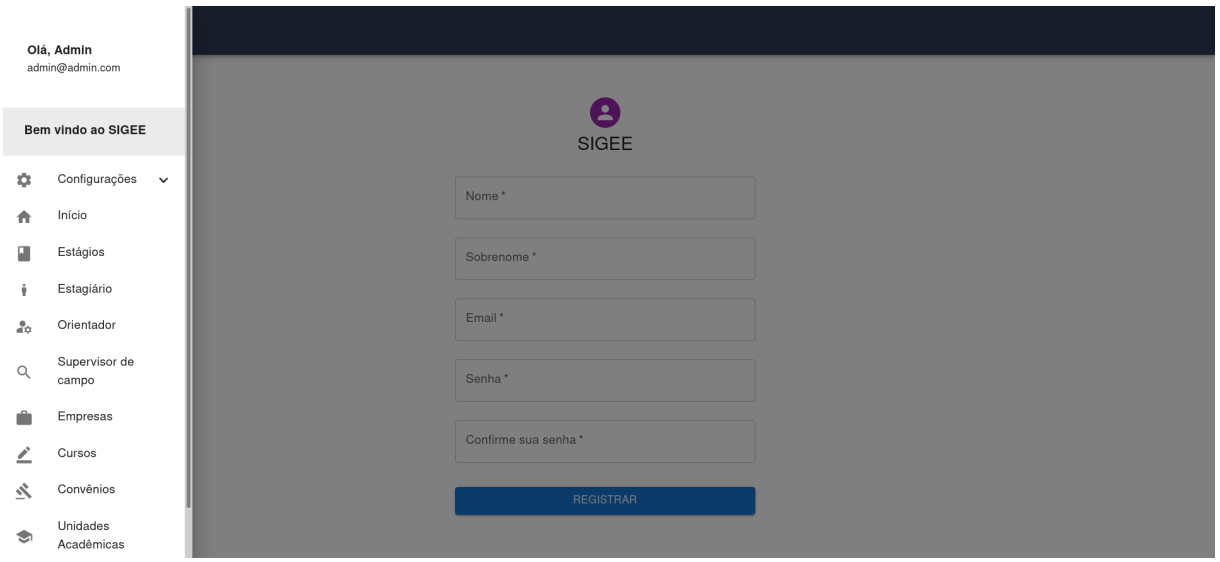
Figura 9 - Tela registro de cursos - Frontend SIGEE



Fonte: Autoria própria

[1] **API disponível em: https://api-sigee.herokuapp.com/api/docs/**

Figura 10 - Frontend do SIGEE, aplicação que utiliza a API como serviço (menu lateral)



Fonte: Autoria própria

6 CONSIDERAÇÕES FINAIS

Dado o exposto, o objetivo deste trabalho é o desenvolvimento de uma API REST, com fluxo de autenticação baseado no padrão JWT, e que é voltada para a gestão de dados relacionados a estágios. A API foi desenvolvida utilizando a linguagem de programação Javascript, em conjunto do ambiente de execução Node.js, do framework Express.js e da ferramenta Postman, utilizado para testes e validação.

Como resultado final, a primeira versão[2] da API foi validada pela aplicação Web do SIGEE, demonstrando que ela fornece de maneira segura e uniforme os recursos necessários para o funcionamento do sistema. Ademais, é importante mencionar que o sistema como um todo foi apresentado ao setor de estágios da Pró-Reitoria de Assuntos Estudantis (PRAE), onde foram sugeridas melhorias e ideias de funcionalidades adicionais, que estão descritas na próxima seção.

[2] **API disponível em: https://api-sigee.herokuapp.com/api/docs/**

7 PERSPECTIVAS FUTURAS

Como perspectivas futuras, podem ser citadas as seguintes funcionalidades: adicionar mais tipos de usuário para a aplicação, adição de rotas com estatísticas relacionadas às entidades (total de estagiários, quantos estão com o relatório em dia ou atrasado, número de estagiários por empresa e etc), possibilidade de gerar o documento de termo de compromisso do estágios, histórico de estágios feitos por um estagiário e, além disso, possibilitar que a API forneça dados em outros formatos além do *JSON* (como XML e TXT).

REFERÊNCIAS

DOCKER, Inc. Why docker. 2022. Disponível em: https://www.docker.com/why-docker/. Acesso em: 07 de março de 2022.

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

JAVASCRIPT. About. 2022. Disponível em: https://www.javascript.com/about. Acesso em: 07 de março de 2022.

OPENJS, Foundation. About Node.js. 2022. Disponível em: https://nodejs.org/en/about/. Acesso em: 07 de março de 2022.

POSTGRESQL, Global Development Group. About. 2022. Disponível em: https://www.postgresql.org/about/. Acesso em: 07 de março de 2022.

POSTMAN, Inc. What is postman. 2022. Disponível em: https://www.postman.com/product/what-is-postman/. Acesso em: 07 de março de 2022.

REACT, Meta Platforms, Inc. 2022. Disponível em: https://pt-br.reactjs.org/. Acesso em: 21 de março de 2022.

SENDGRID. Why Sendgrid. 2022. Disponível em: https://sendgrid.com/why-sendgrid/. Acesso em: 07 de março de 2022.

SEQUELIZE. 2022. Disponível em: https://sequelize.org/. Acesso em: 07 de março de 2022.

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

SWAGGER, SmartBear Software. 2022. Disponível em: https://swagger.io/about/. Acesso em: 21 de março de 2022.

TRELLO, Atlassian. About. 2022. Disponível em: https://trello.com/about. Acesso em: 25 de março de 2022.