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: **BR512022001665-6**

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 31/03/2022, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Título: Pra Nadar App

Data de publicação: 31/03/2022

Data de criação: 30/03/2022

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

Autor(es): ADALBERTO VERONESE DA COSTA; SEBASTIÃO EMIDIO ALVES FILHO; EXLLEY CLEMENTE DOS SANTOS; TIAGO DA SILVA MORAIS; JEFTER ROBERTO MOTA TARGINO; FRANCISCO CLEMENTINO MAIA JÚNIOR; MATHEUS DIOGENES DA SILVA; HÉLIO VICTOR APOLINÁRIO SOARES

Linguagem: JAVA SCRIPT; OUTROS

Campo de aplicação: AD-01; IF-01; SD-01

Tipo de programa: AP-01; GI-01; SO-05

Algoritmo hash: SHA-512

Resumo digital hash:
2c2b1f799dcd7d1f60ca267144f12e53ad34b6dfe293f543e8ca31912a76fab534c8d17ffb79ac75dbebd2e18d55a74d28d5fa02a96742c7dcbd9fe5a18fb039

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

MATHEUS DIOGENES DA SILVA

TESTE DE INTEGRAÇÃO: VALIDAÇÃO DAS FUNCIONALIDADES DA API REST PRA NADAR

MOSSORÓ - RN
2020

MATHEUS DIOGENES DA SILVA

TESTE DE INTEGRAÇÃO: VALIDAÇÃO DAS FUNCIONALIDADES DA API REST PRA NADAR

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 do M.e Exlley Clemente dos Santos.

MOSSORÓ - RN

2020

MATHEUS DIÓGENES DA SILVA

TESTE DE INTEGRAÇÃO: VALIDAÇÃO DAS FUNCIONALIDADES DA API REST PRA NADAR

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: 20 / 04 / 2022

Banca Examinadora

Sebastião Emídio Alves Filho

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

Exlley Clemente dos Santos

Me. Exlley Clemente dos Santos
Universidade do Estado do Rio Grande do Norte - UERN

Ceres Germanna Braga Morais

Profa. Ms. Ceres Germanna Braga Morais
Universidade do Estado do Rio Grande do Norte - UERN

Alysson Mendes de Oliveira

Prof. Alysson Mendes de Oliveira
Universidade do Estado do Rio Grande do Norte - UERN

SUMÁRIO

1 INTRODUÇÃO

1.1 DESCRIÇÃO GERAL DO PROJETO

O Programa Pra Nadar é um projeto de extensão fundado e dirigido pela Faculdade de Educação Física (FAEF) da Universidade do Estado do Rio Grande do Norte (UERN), oferecendo práticas de atividades aquáticas, como natação e hidroginástica. As práticas são oferecidas para a comunidade em geral, o que inclui alunos, professores e servidores da UERN, além da comunidade externa. O processo de matrícula dos praticantes passa pela necessidade da entrega de atestado médico, dando a devida condição para o exercício das atividades. O Pra Nadar funciona através das contribuições financeiras, provenientes da participação dos alunos matriculados no programa.

O programa inicialmente era organizado pelos secretários e professores da faculdade e não existia algo para contribuir na gestão dos dados que estavam sendo inseridos e tratados. O processo de gerenciamento de turmas, alunos, instrutores e secretários no programa eram feitos de forma manual, ocasionando na dificuldade do gerenciamento.

1.2 JUSTIFICATIVA

Diante da dificuldade do gerenciamento, contando com problemas de controle do número de estudantes em cada turma e vagas disponíveis para matrículas, registro das contribuições feitas durante o decorrer do programa, controle das frequências dos estudantes e instrutores. Com estas adversidades, a FAEF necessitava de um sistema de gerenciamento para ter melhor controle do programa. Foi desenvolvido um sistema que visa administrar os dados do Programa Pra Nadar.

2 OBJETIVOS

2.1 OBJETIVO GERAL

O projeto tem a finalidade de fornecer um sistema de gerenciamento de informações para auxiliar a equipe de alunos e professores da FAEF envolvidas no programa Pra Nadar.

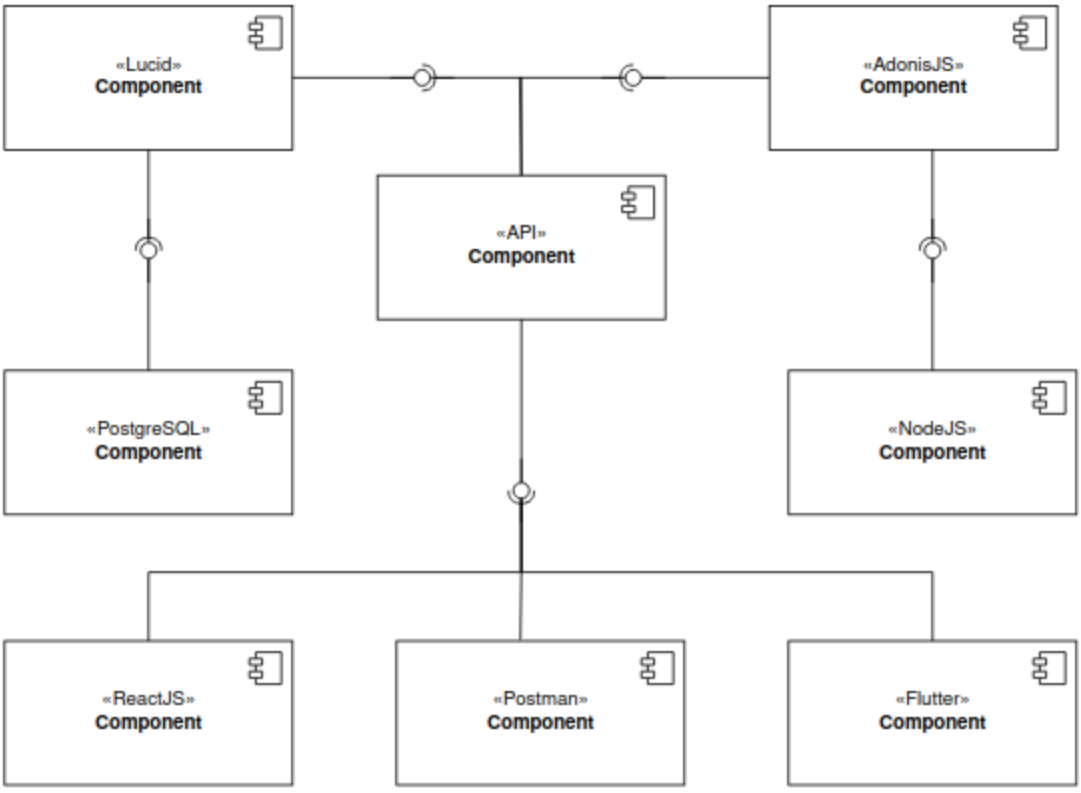
2.2 OBJETIVO ESPECÍFICO

Criar um roteiro que busque validar as funcionalidades da *Application Program Interface* (API, em português, Interface de Programa de Aplicação), visando aumentar a confiabilidade e encontrar falhas de integração entre os módulos.

3 COMPONENTES DO SISTEMA

O sistema está estruturado como mostra a figura 1. O principal componente da aplicação é a API, ela contém toda regra de negócio e serve como base de dados para a plataforma web e mobile do sistema Pra Nadar. A aplicação é formada pelos seguintes componentes:

Figura 1 - Diagrama de Componentes



Autoria própria (2022)

1. AdonisJS

O AdonisJS inclui tudo o que você precisa para criar um aplicativo da Web totalmente funcional ou um servidor de API (ADONISJS, 2022, tradução nossa). Foi utilizado para a construção da API.

2. Lucid

Lucid é o SQL ORM oficial para AdonisJS (NPMJS, 2022, tradução nossa). Ele é o ORM padrão do AdonisJS para fazer a comunicação com o banco de dados.

3. PostgreSQL

É um gerenciador de banco de dados objeto-relacional PostgreSQL, que conforme DevMedia ([202?]): “é um dos cinco SGBDs relacionais mais utilizados no mercado", por ser projetado para ser portátil, atuando em várias plataformas e é um software de código aberto e de baixo custo.(DEVMEDIA, [202?]).

4. NodeJS

Node.js é uma tecnologia usada para executar código JavaScript fora do navegador. Com ele podemos construir aplicações web em geral, desde web sites até APIs e microsserviços. (DEVMEDIA, 202?])

5. ReactJS

Uma biblioteca JavaScript para criar interfaces de usuário. (META PLATFORMS, INC. 2022). Foi utilizado para criar a interface do usuário na plataforma web.

6. Flutter

Flutter é uma estrutura de código aberto do Google para criar aplicativos multiplataforma bonitos e compilados nativamente a partir de uma única base de código (FLUTTER, 2022, tradução nossa). Foi utilizado para criar a interface do usuário na plataforma mobile.

7. Postman

Postman é uma plataforma de API para construir e usar APIs. O Postman simplifica cada etapa do ciclo de vida da API e agiliza a colaboração

para que você possa criar APIs melhores e mais rapidamente. (POSTMAN, 2022, tradução nossa). Foi usado para fazer os testes.

4 METODOLOGIA

O teste de API é tão importante quanto o teste de software. API é:

> “Uma interface, normalmente especificada como um conjunto de operações, que permite acesso a uma funcionalidade da aplicação. Isso significa que essa funcionalidade pode ser chamada diretamente por outros programas e não apenas acessada através da interface de usuário” (ENGENHARIA DE SOFTWARE, 2011, p. 515).

Com a API é possível criar um ou mais sistemas para diferentes plataformas utilizando a mesma base de dados. A API permite criar regras de negócio para a aplicação, onde qualquer software que queira utilizar a API precisa obrigatoriamente obedecer essas regras.

Os testes de API servem para nos certificar de que cada módulo está funcionando adequadamente com base nas suas regras de negócio que foram definidas.

O teste escolhido para a validação da API foi os testes de integração. O objetivo do teste de integração é:

> “[...] descobrir bugs de componente que só são revelados quando um componente é usado por outros componentes do sistema.O teste de interação também ajuda a encontrar equívocos dos desenvolvedores de componentes sobre outros componentes do sistema” (ENGENHARIA DE SOFTWARE, 2011, p. 153).

4.1 ATORES

O sistema Pra Nadar tem quatro tipos de usuários, onde cada um deles tem seu papel dentro do sistema. São eles:

1. Secretário

   O secretário é de um cargo administrativo, ele atua no gerenciamento de turmas, isso inclui excluir, atualizar, criar e visualizar os dados, gerenciar as presenças dos instrutores, turmas e estudantes. Ele também é responsável por anexar os atestados e questionários no ato da matrícula e registrar pagamentos feitos pelo estudante.

2. Instrutor

O instrutor vai ser responsável por fazer o gerenciamento das aulas da turma a qual ele está vinculado, como elaborar planos de aula, fazer a presença dos estudantes, gerenciar suas informações pessoais, ver as turmas a qual ele está vinculado, alunos que estão na turma e a visualização do perfil dos estudantes da turma.

3. Estudante

O estudante é o usuário que tem menos funcionalidades, ele poderá visualizar seu perfil, onde estarão seus dados pessoais, as turmas a qual ele está vinculado, suas notas, presenças na aulas e os pagamentos realizados e pendentes.

4. Administrador

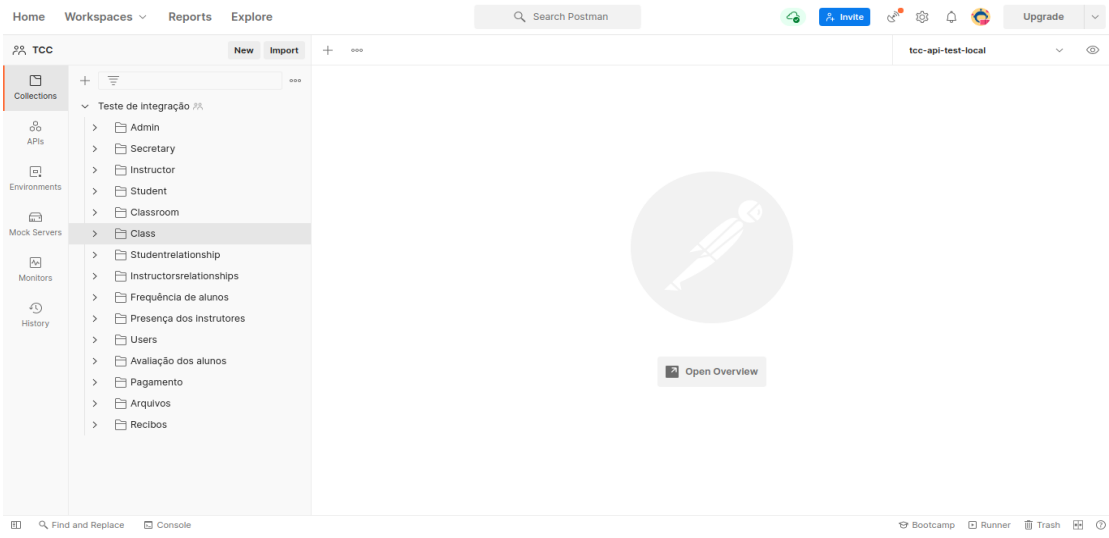
O administrador é o usuário com mais privilégios, de forma que o sistema já vem com administrador por padrão e só terá apenas ele no sistema. Ele tem a permissão de gerenciar todos os usuários, isso inclui excluir, atualizar, criar e visualizar os dados dos usuários. Além de ter todas as funcionalidades de secretário, estudante e instrutor.

4.2 AUTOMAÇÃO DOS TESTES

Para o desenvolvimento e automação de testes foi utilizado o Postman (Postman Inc., 2022). O Postman se demonstrou uma ferramenta que oferece bons recursos para automatização de testes. Um dos principais recursos que ele traz, e de estabelecer regras, e com base nessas regras, se acontecer o resultado esperado, o teste vai ter sucesso, caso contrário, o teste irá falhar.

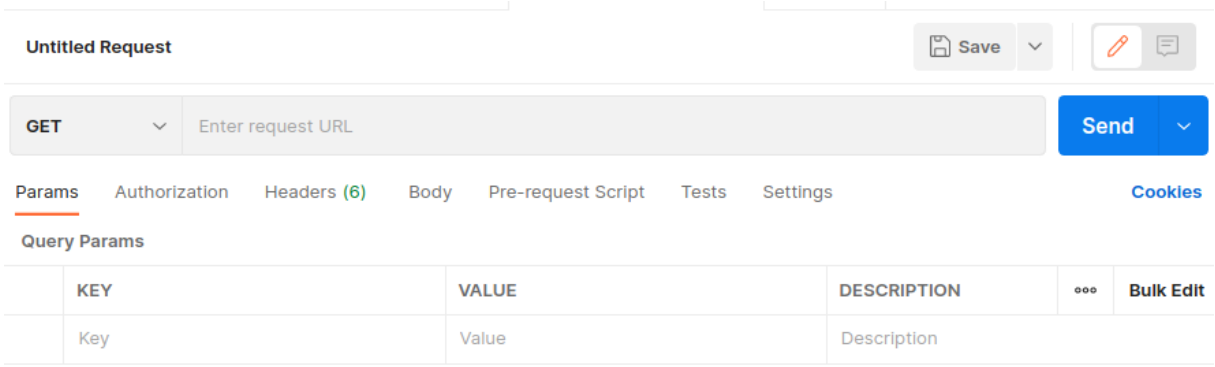
Na Figura 2 mostra a estrutura dos testes. Cada pasta é uma funcionalidade do sistema. A pasta Studentrelationship contém o caso de uso de vinculação do estudante à turma, que será usado posteriormente para demonstrar as etapas que foram feitas para validar essa funcionalidade.

Figura 2 - Estrutura do projeto



Fonte: Autoria Própria (2022)

Figura 3 - Postman



Fonte: Autoria Própria (2022)

A Figura 3 ilustra a principal área do Postman. Nessa área é possível colocar o endereço da requisição, método HTTP, os dados que serão enviados, o script para ser executado antes da requisição ser enviada e os testes. A linguagem que o postman adota para fazer os scripts é o ECMAscript, também conhecido como Javascript.

Na Figura 3, Foram utilizadas as seguintes funcionalidades:

1. Params: Como o próprio nome já diz, aqui serão enviados os parâmetros da requisição.
2. Authorization: Esse recurso permite o uso token de autorização. Token de autorização é um certificado digital que permite que o usuário só faça o que lhe é permitido dentro da API. Como a API tem vários tipo de usuários, o token vai ser usado para autenticação, ou seja, verificar se usuário é quem diz ser, e também na parte da autorização, que é a parte que autoriza o usuário a fazer somente o que lhe é permitido, por exemplo, vai evitar que um estudante execute uma tarefa de um instrutor.
3. Body: Onde ficam os dados da requisição, em formato JSON.
4. Pre-request Script: É executado um script antes da requisição ser enviada, um exemplo de uso é definir um script que coloque o token de autorização dentro da requisição. Isso evitaria eu colocar o token manualmente.
5. Tests: Onde serão feitos os testes.

4.3 TESTES DE INTEGRAÇÃO

Todos os testes de integração foram feitos seguindo a documentação da API do sistema Pra Nadar. Para cada funcionalidade foi feito um caso de uso com intuito de validar a funcionalidade. No total foram feitos 17 casos de uso. Os 17 casos de uso colocaram à prova todas as funcionalidades da API, sem exceção. A seguir é demonstrado como foi feito um dos testes de integração. Como dito anteriormente, o teste escolhido foi o de vincular o estudante à turma (Studentrelationship).

Tabela 1 - Teste de vincular estudante à turma

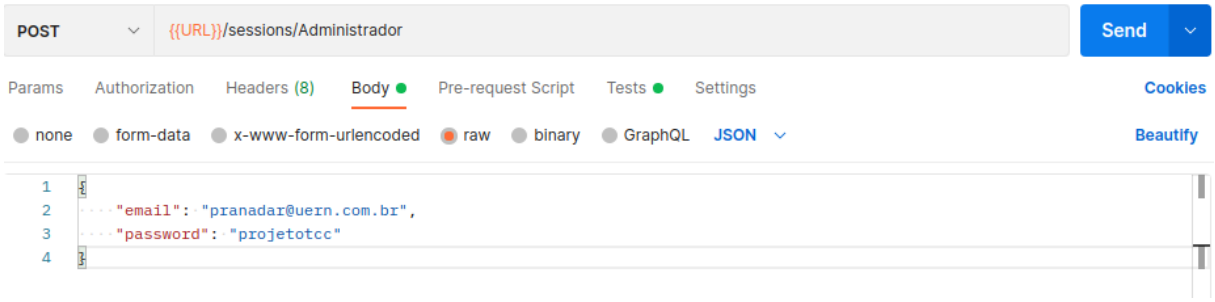
Identificador	Teste de integração
Ambiente	Postman
Objetivo	Vincular estudante a turma
Ator	Administrador
Pré Condições	O administrador deve estar logado no sistema.
Procedimentos	O administrador irá fazer o cadastro do estudante. O administrador irá criar uma turma. O administrador irá vincular o estudante à turma.

Resultados Esperados	Caso tudo ocorra como planejado, o estudante será vinculado à turma.
Teste mal sucedido	O teste deve falhar se: O estudante já esteja vinculado à turma. Turma não existe. Estudante não existe.

Fonte: Autoria Própria (2022)

Como especificado na Tabela 1, para fazer o vínculo de um estudante à turma, o administrador do sistema deve estar logado. Para fazer o login, é informado no body da requisição o email e senha do administrador, como ilustrado na Figura 4.

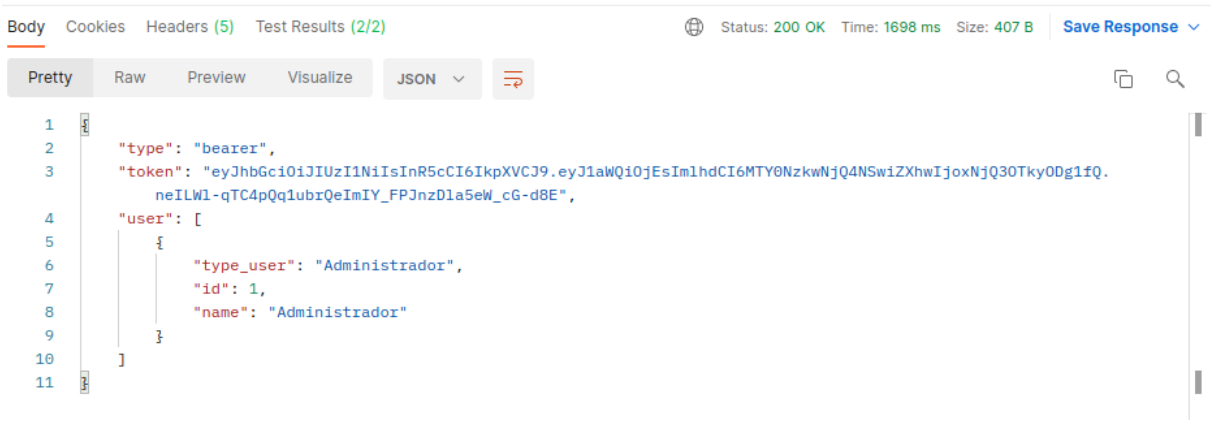
Figura 4 - Informando os dados do administrador para fazer o login



Fonte: Autoria Própria (2022)

Após enviar a requisição, a aplicação retorna um token para o cliente (Figura 5). Esse token será usado para autenticação. Qualquer atividade que ele for fazer dentro do sistema ele tem que estar autenticado.

Figura 5 - Resposta da requisição do login do administrador



Fonte: Autoria Própria (2022)

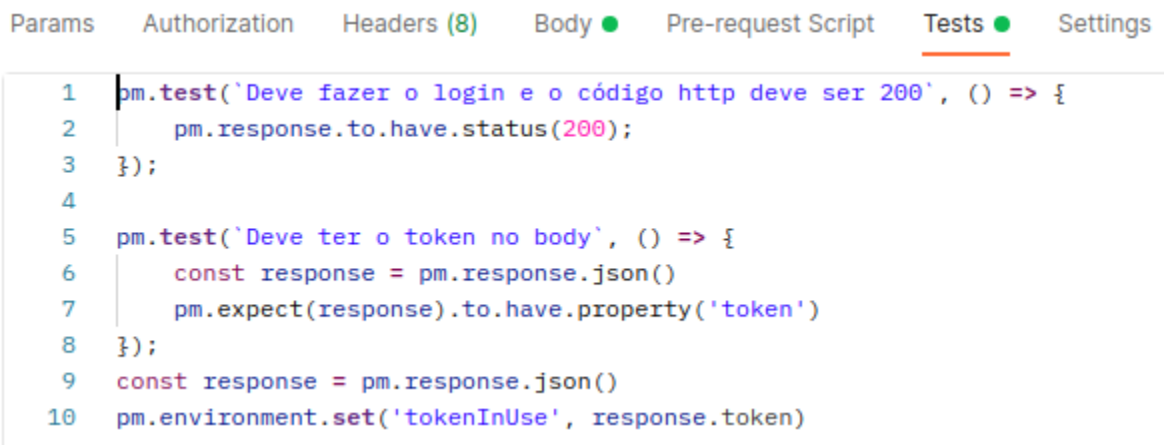
Os testes feitos para esse para o login do administrador foi:

1. Código HTTP deve ser 200.
2. A resposta da requisição deve conter um token.

A implementação dos testes para o login do administrador pode ser vista na Figura 6.

Após feitos os testes, o token é guardado em uma variável de ambiente que será usada para futuras atividades do administrador no sistema.

Figura 6 - Testes feitos para o login do administrador



Fonte: Autoria Própria (2022)

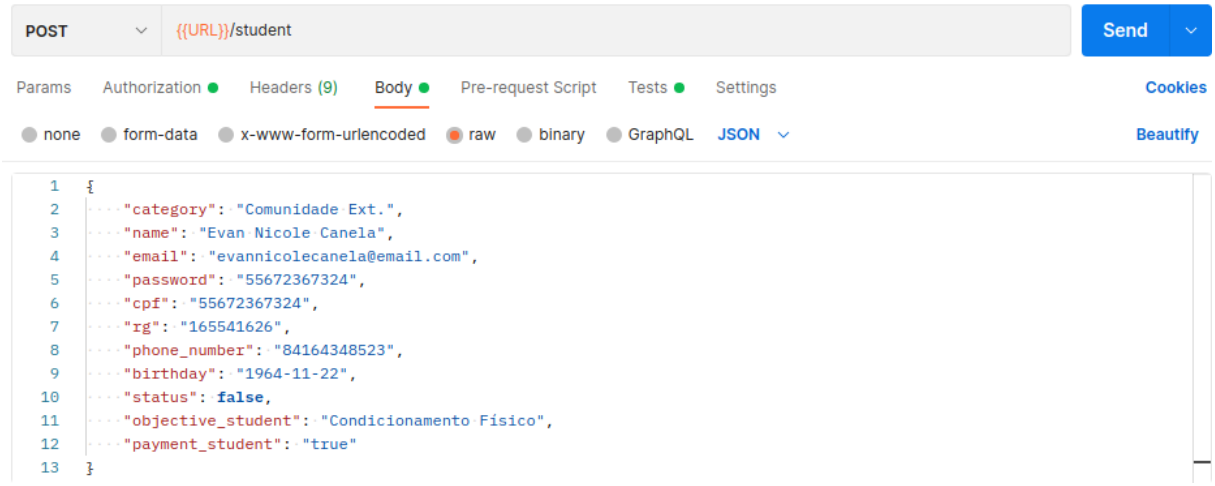
Como observado na Figura 7, os testes obtiveram sucesso.

Figura 7 - Resultados dos testes: Login do administrador

Fonte: Autoria Própria (2022)

O primeiro procedimento é fazer o cadastro do estudante (Figura 8), onde a senha padrão do usuário será o cpf desse.

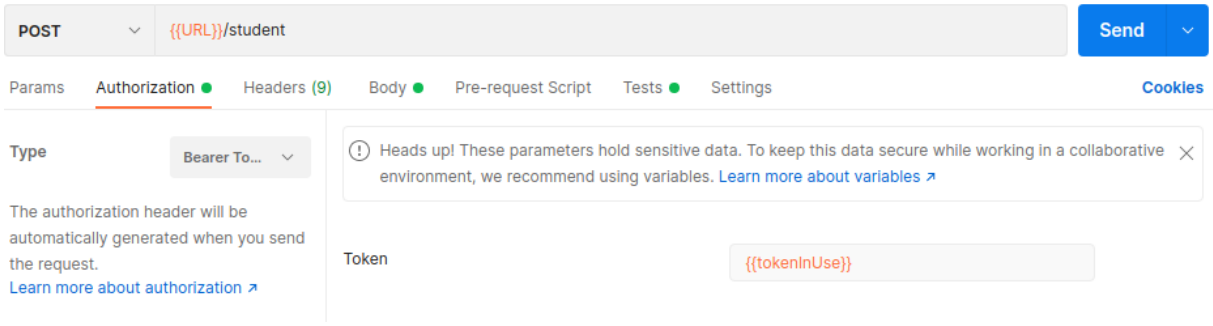
Figura 8 - Informações do cadastro do estudante



Fonte: Autoria Própria (2022)

Para fazer o cadastro do estudante, é necessário estar autenticado, para isso é informado o token do administrador (Figura 9). Como o token foi guardado dentro de uma variável de ambiente, basta colocar o nome da variável onde colocaria o token.

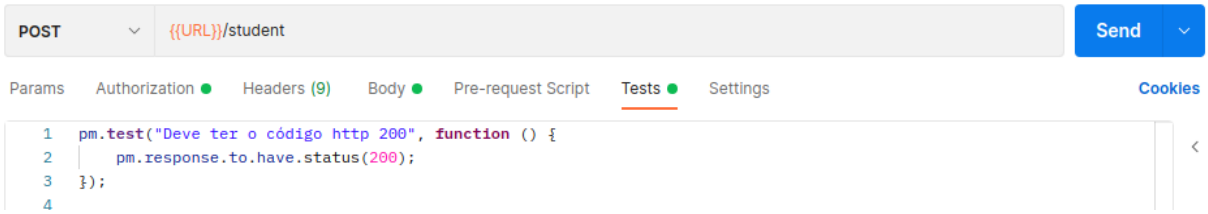
Figura 9 - Informando o token



Fonte: Autoria Própria (2022)

Para garantir que a requisição fosse entregue, o teste feito, mostrado na Figura 10, foi justamente para ter essa garantia.

Figura 10 - Teste do cadastro do estudante.



Fonte: Autoria Própria (2022)

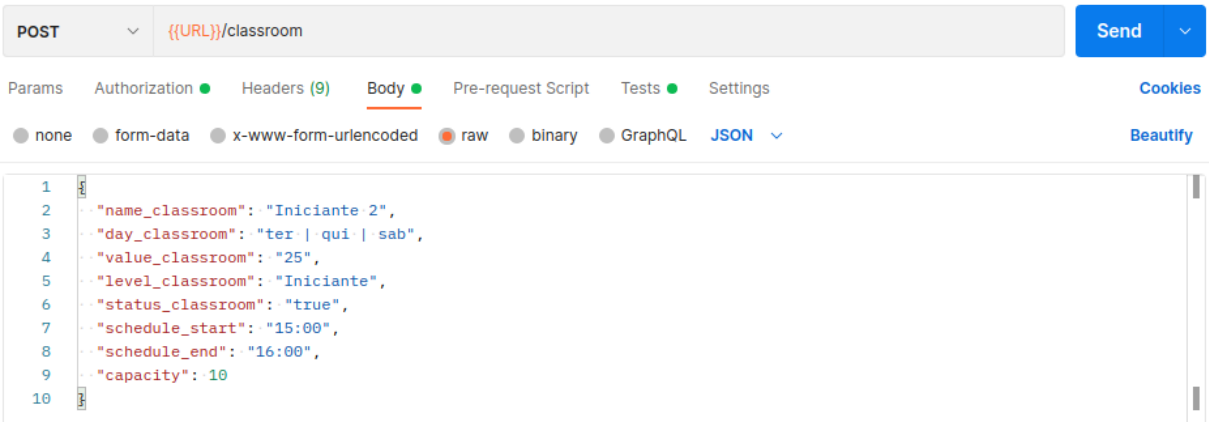
Ao enviar a requisição, o resultado do teste, visto na Figura 11.

Figura 11 - Resultado do teste do cadastro do estudante

Fonte: Autoria Própria (2022)

Para criação da turma, o administrador informa os dados da turma, como ilustrado na Figura 12.

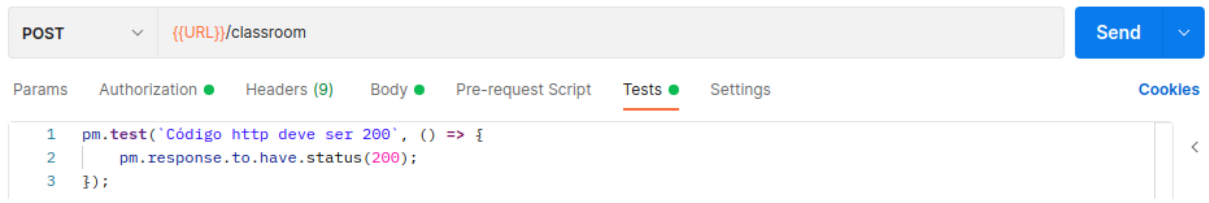
Figura 12 - Criação de turma



Fonte: Autoria Própria (2022)

Para essa tarefa de criação de turma, é preciso informar o token para a requisição da mesma forma que mostra a Figura 9. O teste feito, com visto na Figura 13, foi que o código HTTP deve ser 200.

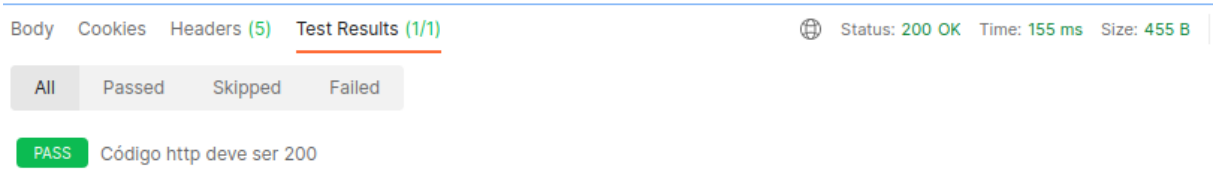
Figura 13 - Teste feito para criação de turma.



Fonte: Autoria Própria (2022)

Após enviar a requisição, é mostrado o resultado do teste, mostrado na Figura 14.

Figura 14 - Resultado do teste feito para criação de turma



Fonte: Autoria Própria (2022)

Após criar o estudante e a turma, foi feito o teste de vincular o estudante à turma criada. Para prosseguir com o teste, foi feito um script antes da requisição (Pre-request Script) ser enviada para pegar o ID do estudante (Figura 15) e o ID da turma (Figura 16) que foram criados anteriormente. Caso essa pré-requisição falhar, significa que o estudante não foi encontrado ou a turma não foi encontrada, ou

ambos. E consequentemente os testes feitos para saber se o estudante foi vinculado à turma irá falhar.

Figura 15 - Pegando o estudante que vai ser vinculado à turma.

Fonte: Autoria Própria (2022)

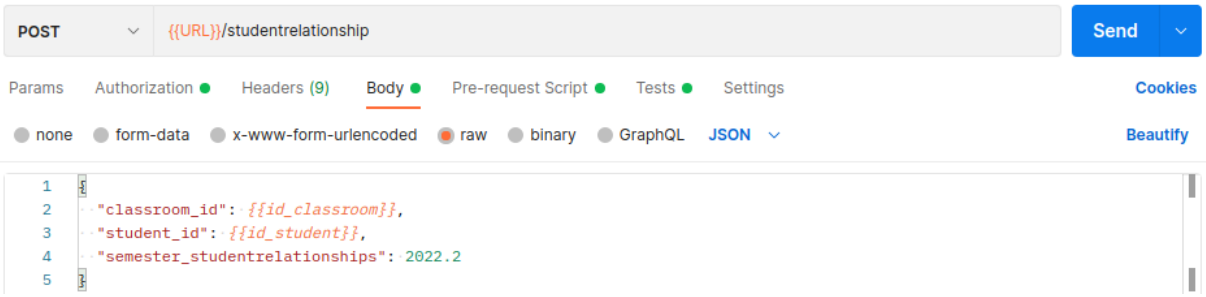
Figura 16 - Pegando a turma a qual o estudante vai ser vinculado.

Fonte: Autoria Própria (2022)

O ID de estudante e o ID de turma foram guardados dentro de uma variável de ambiente, que foi usada para vincular o estudante à turma.

Os dados para vincular o estudante à turma são informados (Figura 17), e a requisição foi enviada.

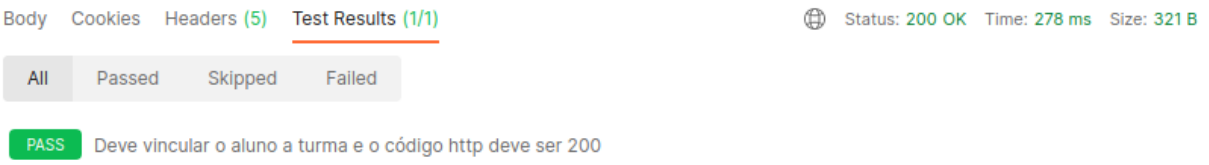
Figura 17 - Vinculação do estudante a turma



Fonte: Autoria Própria (2022)

Feito a requisição, o resultado do teste (Figura 18).

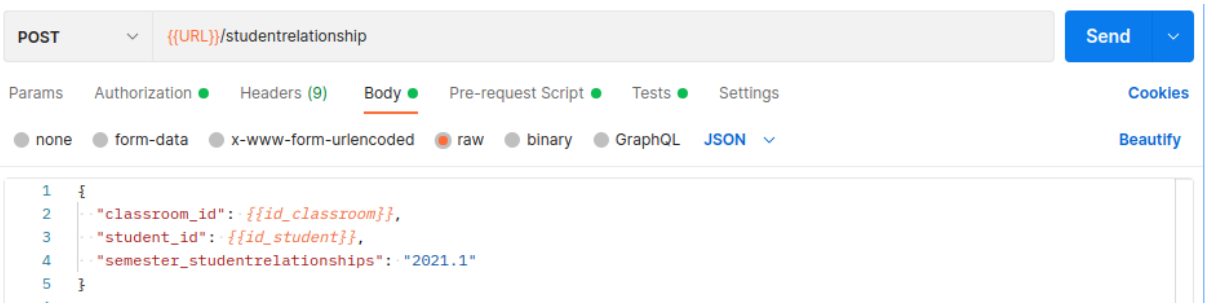
Figura 18 - Resultado do teste: vinculação do estudante à turma.



Fonte: Autoria Própria (2022)

Esses foram os casos de sucesso, tudo ocorreu como planejado. Foram feitos três casos de falha para esta funcionalidade, onde o primeiro é o caso do estudante já estar vinculado à turma. A resposta que a API deve retornar é que ao tentar vincular o estudante já vinculado, o estudante não deve ser vinculado novamente e deve retornar uma mensagem informando o que aconteceu. A vinculação do estudante pode ser visualizada na Figura 19.

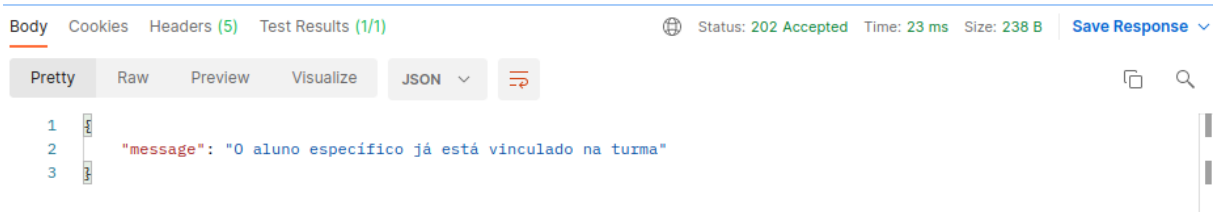
Figura 19 - Vinculação de estudante que já está vinculado



Fonte: Autoria Própria (2022)

O estudante e a turma utilizados aqui são os mesmo dos testes anteriores. Enviado a requisição, a API retorna uma mensagem informando que o estudante já foi vinculado e o código HTTP da requisição foi 202 (Figura 20).

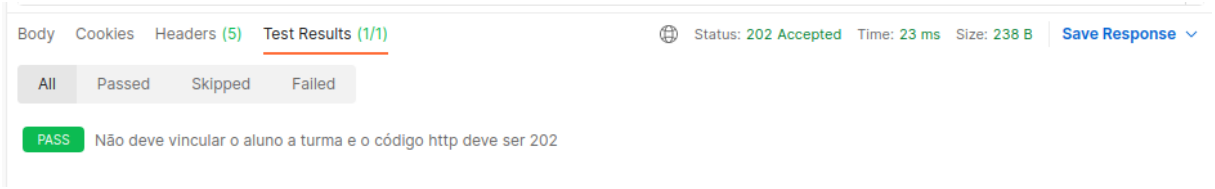
Figura 20 - Resposta da requisição



Fonte: Autoria Própria (2022)

E com a requisição já enviada, o Postman mostra o resultado do teste (Figura 21).

Figura 21 - Resultado do teste

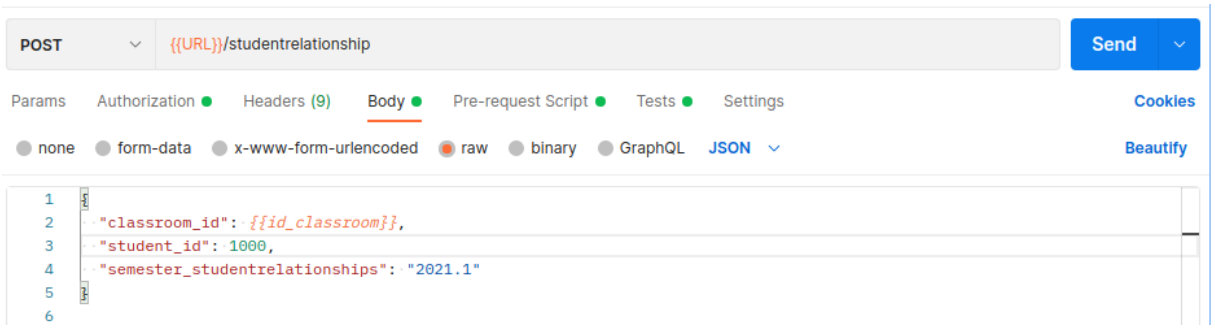


Fonte: Autoria Própria (2022)

O segundo caso de falha foi para ver se a API se comporta corretamente quando é feita a tentativa de vincular um estudante a uma turma que não existe, e o terceiro foi de vincular um estudante que não existe a uma turma já existente.

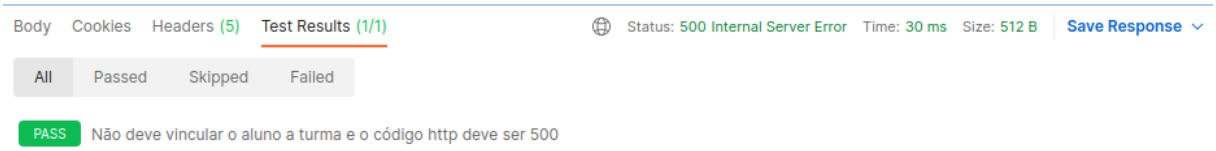
O caso de vincular um estudante a uma turma que não existe é mostrado na Figura 22. Feito a requisição é mostrado o resultado do teste (Figura 23).

Figura 22 - Vinculação de um estudante inexistente



Fonte: Autoria Própria (2022)

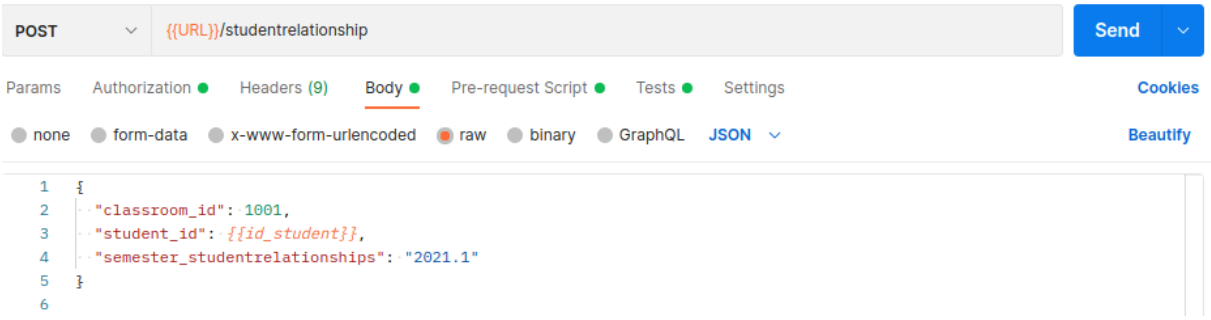
Figura 23 - Resultado do teste



Fonte: Autoria Própria (2022)

Por fim, o caso de vincular um estudante que não existe a uma turma já existente. Para esse teste foi colocado um ID de uma turma que já existe, e o estudante usado aqui foi o mesmo da Figura 8. As informações foram passadas como ilustrado na Figura 24.

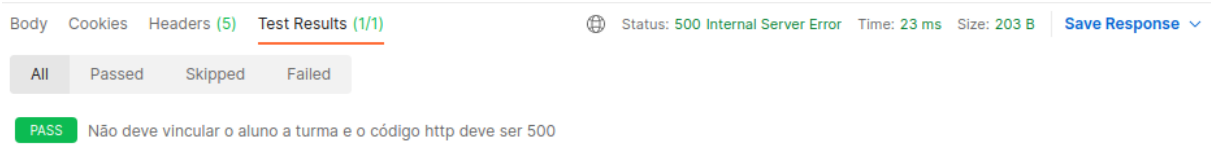
Figura 24 - Vinculação de um estudante a uma turma inexistente



Fonte: Autoria Própria (2022)

Logo após, é mostrado o resultado do teste (Figura 25) na interface do Postman.

Figura 25 - Resultado do teste



Fonte: Autoria Própria (2022)

Com esses testes, a funcionalidade de vincular o estudante à turma está válida, e se ocorrer algum imprevisto, como o estudante ou turma não existe, a API tem o comportamento correto. Os demais testes foram feitos nesses mesmos procedimentos. Os resultados desses testes podem ser vistos no Apêndice A.

5 CONSIDERAÇÕES FINAIS

Ao término dos testes, o resultado mostrou que cada funcionalidade está muito bem integrada com as demais, e que não é possível burlar o sistema pelo fato de que o sistema está funcionando de forma hierárquica e organizada. Para chegar a funcionalidade mais interna, tem que passar pela funcionalidade mais externa. O teste de integração mostrou que a API cumpriu o que foi proposto, e que os resultados obtidos nos testes realizados foram satisfatórios.

Como perspectivas futuras, além do teste de integração, outros testes que aumentariam ainda mais a confiabilidade, seriam o teste de aceitação, que trabalha com dados reais dos usuários e adicionar teste de unidade, que o desenvolvedor pode executar para testar unidades específicas do código.

REFERÊNCIAS

ADONISJS. **A fully featured web framework for Node.js.** [2021?]. Disponível em: https://adonisjs.com/ . Acesso em: 29 mar. 2022.

DEVMEDIA. **Tecnologia Node,js.** [202?]. Disponível em: https://www.devmedia.com.br/guia/node-js/40312. Acesso em: 2 ago. 2022.

DEVMEDIA. **Tecnologia PostgreSQL.** [S. l.], [202?]. Disponível em: https://www.devmedia.com.br/guia/tecnologia-postgresql/34328. Acesso em: 4 ago. 2022.

META PLATFORMS, INC. **Introdução**. [*S. l.*], 2022. Disponível em: https://pt-br.reactjs.org/docs/getting-started.html. Acesso em: 4 ago. 2022.

NPMJS, adonis/lucid. 2022. Disponível em: https://www.npmjs.com/package/@adonisjs/lucid. Acesso em: 4 ago. 2022.

POSTMAN. **Postman**. Disponível em: https://www.postman.com/. Acesso em: Março de 2022.

SOMMERVILLE, Ian. **Engenharia de software.** 9. ed. São Paulo: Pearson Prentice Hall, 2011.

APÊNDICE A - RESULTADOS DE TODOS OS TESTES

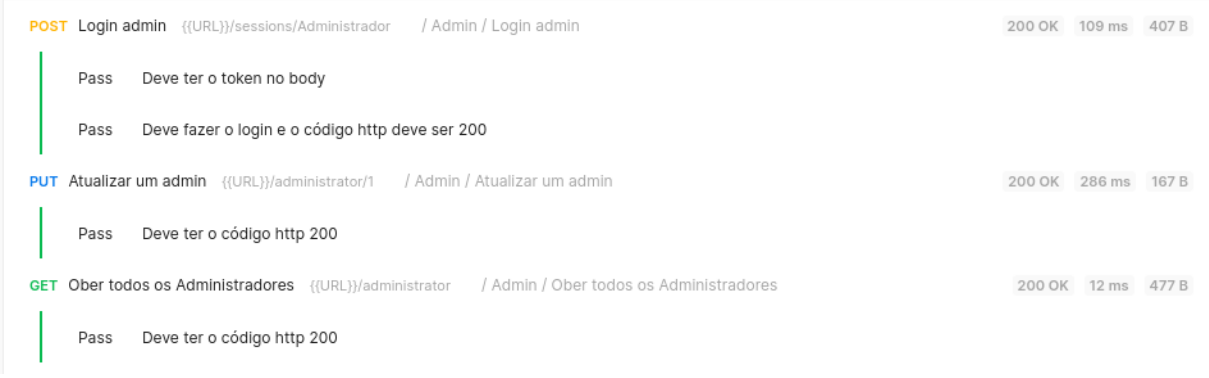
A seguir os resultados de todos os testes feitos. Os testes foram realizados para validar o funcionamento de todas as funcionalidades. Cada funcionalidades abaixo pode ser testada de forma independente, tomando como exemplo as funcionalidades de administradores que são login, atualização de dados e obter todos os administradores cadastrados no sistema, essa bateria de testes pode ser testada individualmente.

Os resultados dos testes foram gerados automaticamente pelo Postman.

Administradores

Funcionalidades:

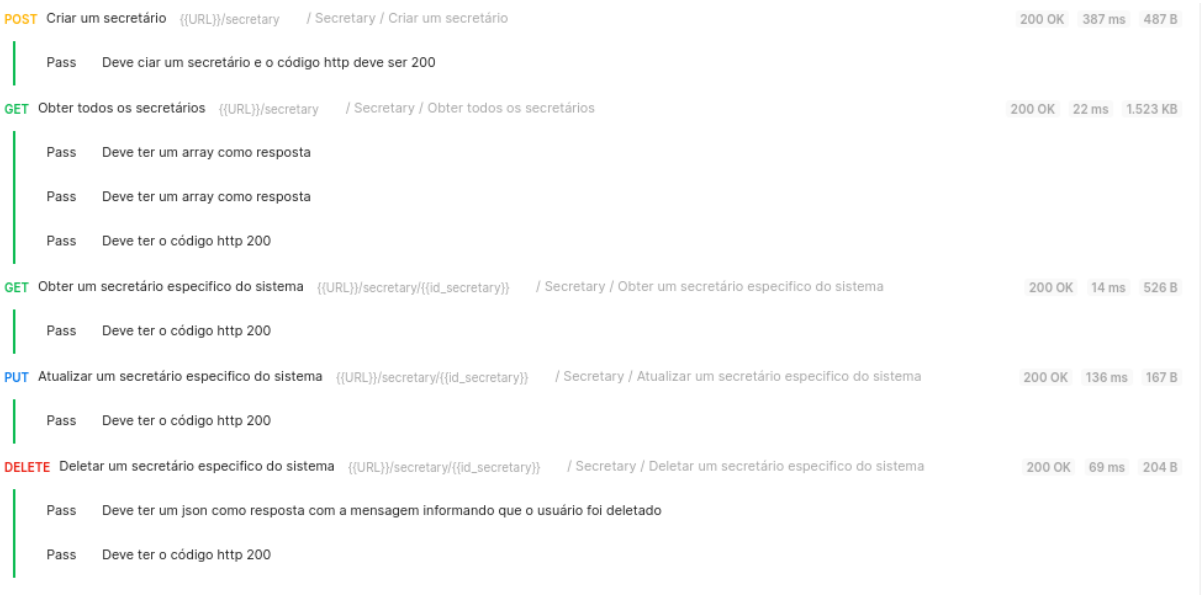
- Login
- Atualizar dados
- Obter todos os administradores do sistema



Secretários

Funcionalidades:

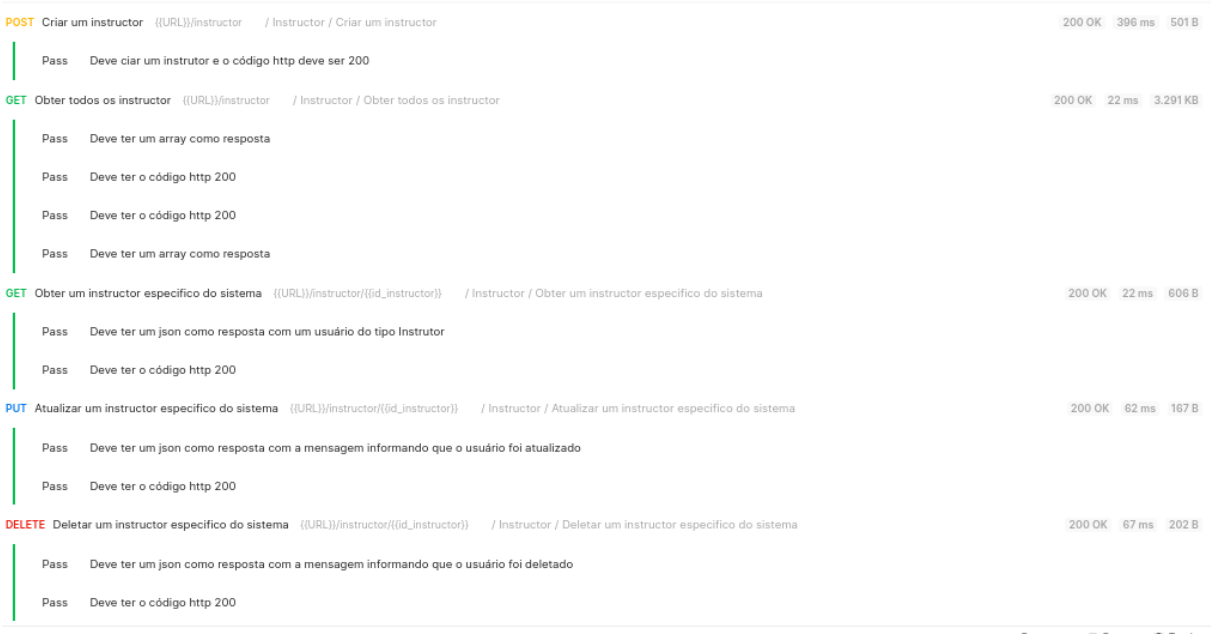
- Criação de secretários
- Obter todos os secretários do sistema
- Obter um secretário do sistema
- Atualização de secretário
- Deletar secretário



Instrutores

Funcionalidades:

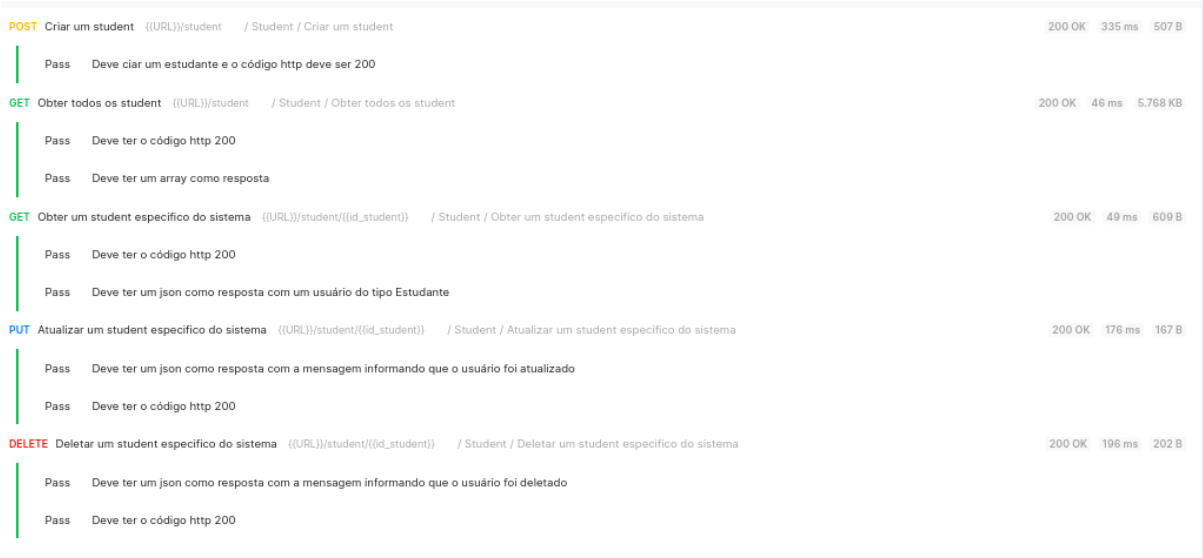
- Criação de instrutores
- Obter todos os instrutores do sistema
- Obter um instrutor do sistema
- Atualização de instrutor
- Deletar instrutor



Estudantes

Funcionalidades:

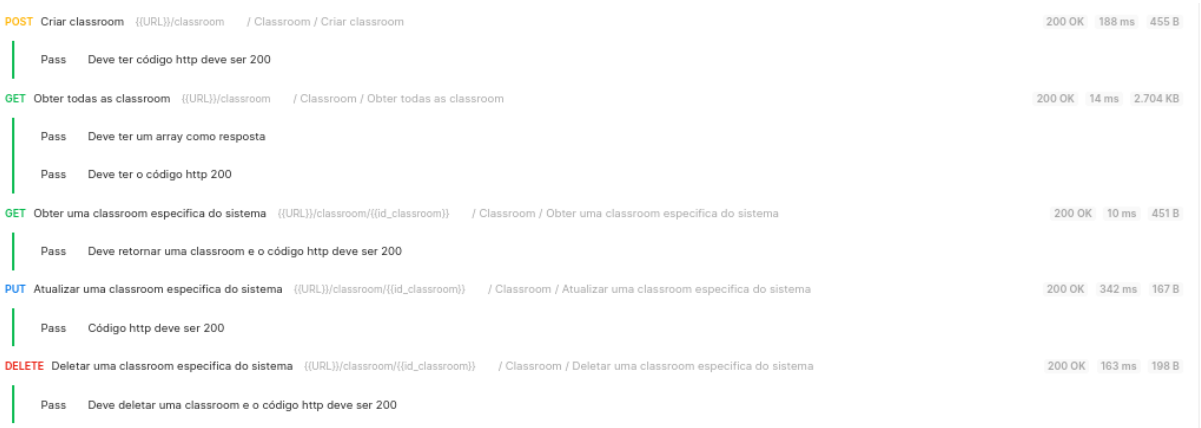
- Criação de estudante
- Obter todos os estudantes do sistema
- Obter um estudante do sistema
- Atualização de estudante
- Deletar estudante



Turmas

Funcionalidades:

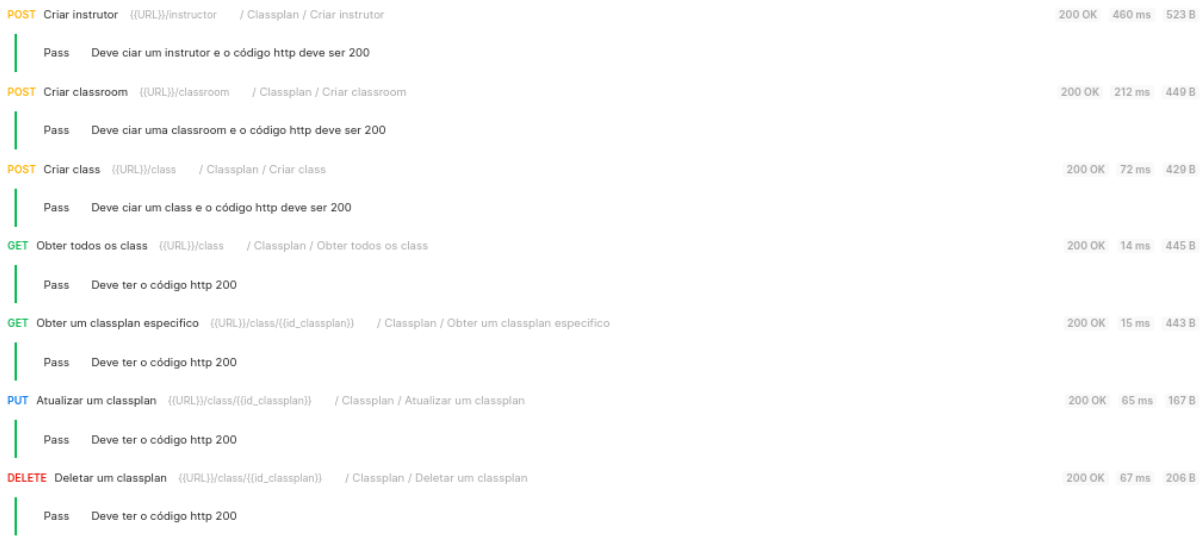
- Criação de turmas
- Obter todas as turmas do sistema
- Obter uma turma do sistema
- Atualização de turma
- Deletar turma



Planos de aula

Funcionalidades:

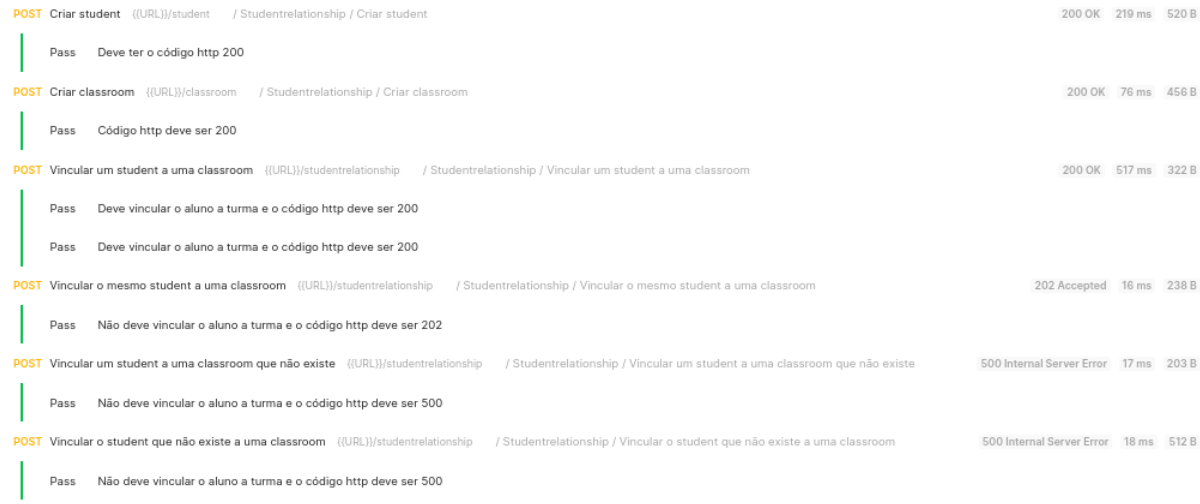
- Criação de plano de aula
- Obter todos os planos de aula do sistema
- Obter um plano de aula do sistema
- Atualização de plano de aula
- Deletar plano de aula

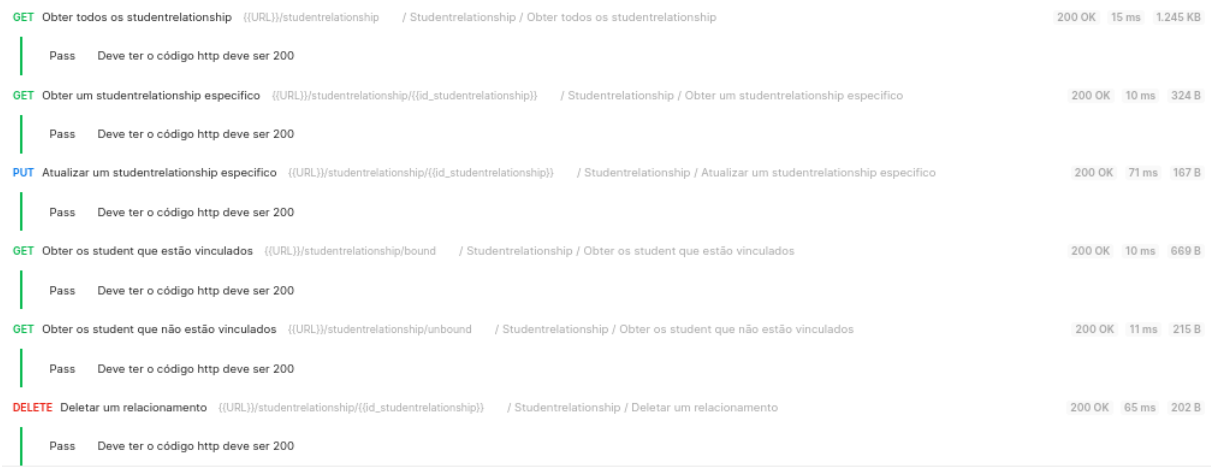


Vínculo de estudante a turma

Funcionalidades:

- Criação de vínculo
- Obter todos os vínculos do sistema
- Obter um vínculo do sistema
- Atualização de vínculo
- Deletar vínculo

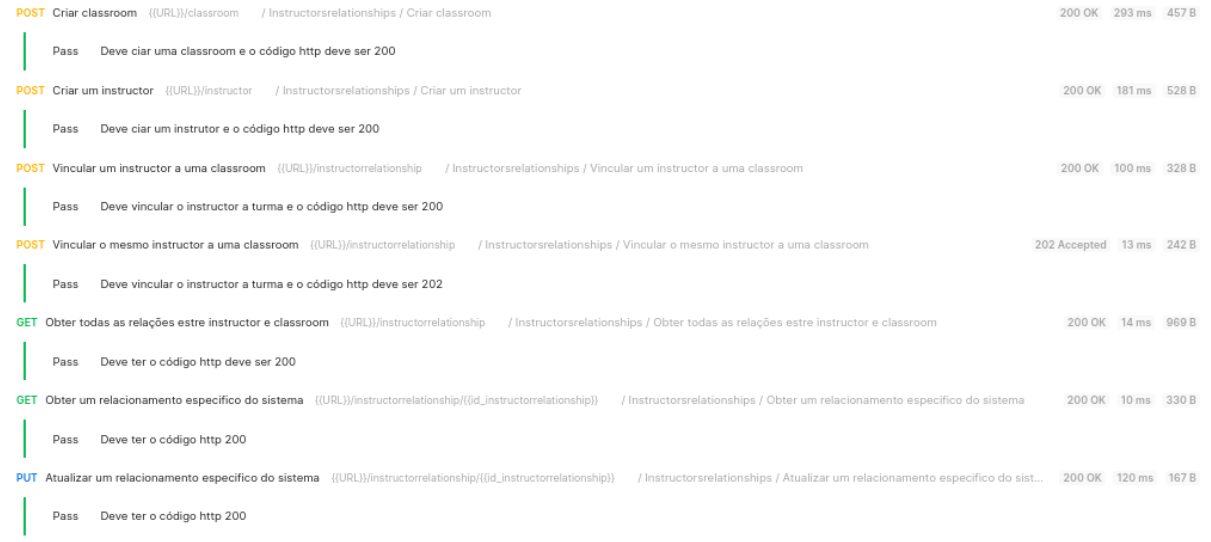




Vínculo de instrutor a turma

Funcionalidades:

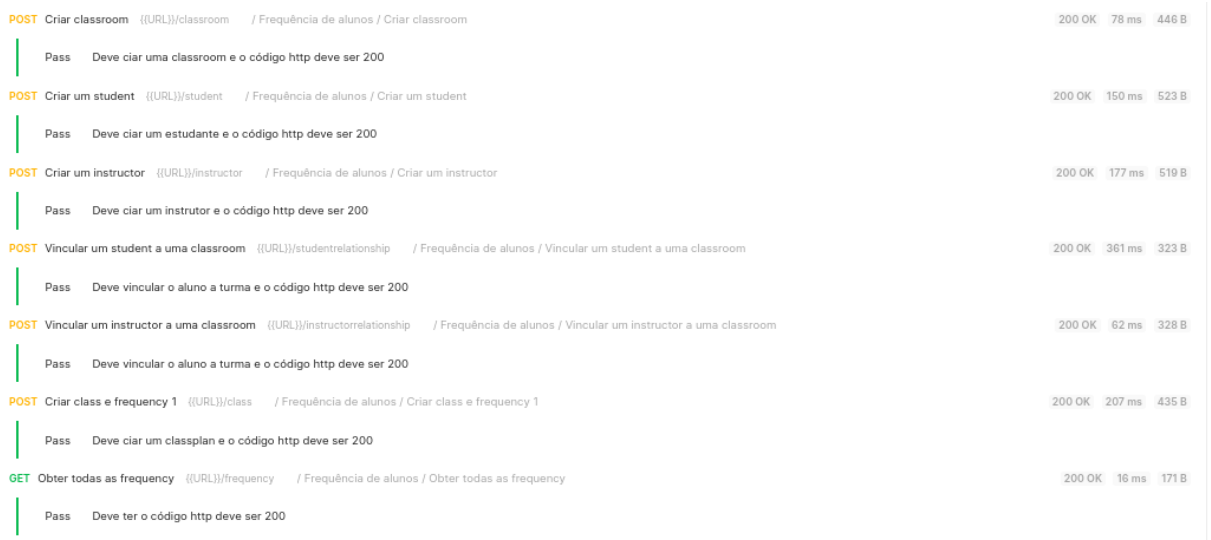
- Criação de vínculo
- Obter todos os vínculos do sistema
- Obter um vínculo do sistema
- Atualização de vínculo
- Deletar vínculo



Frequência de alunos

Funcionalidades:

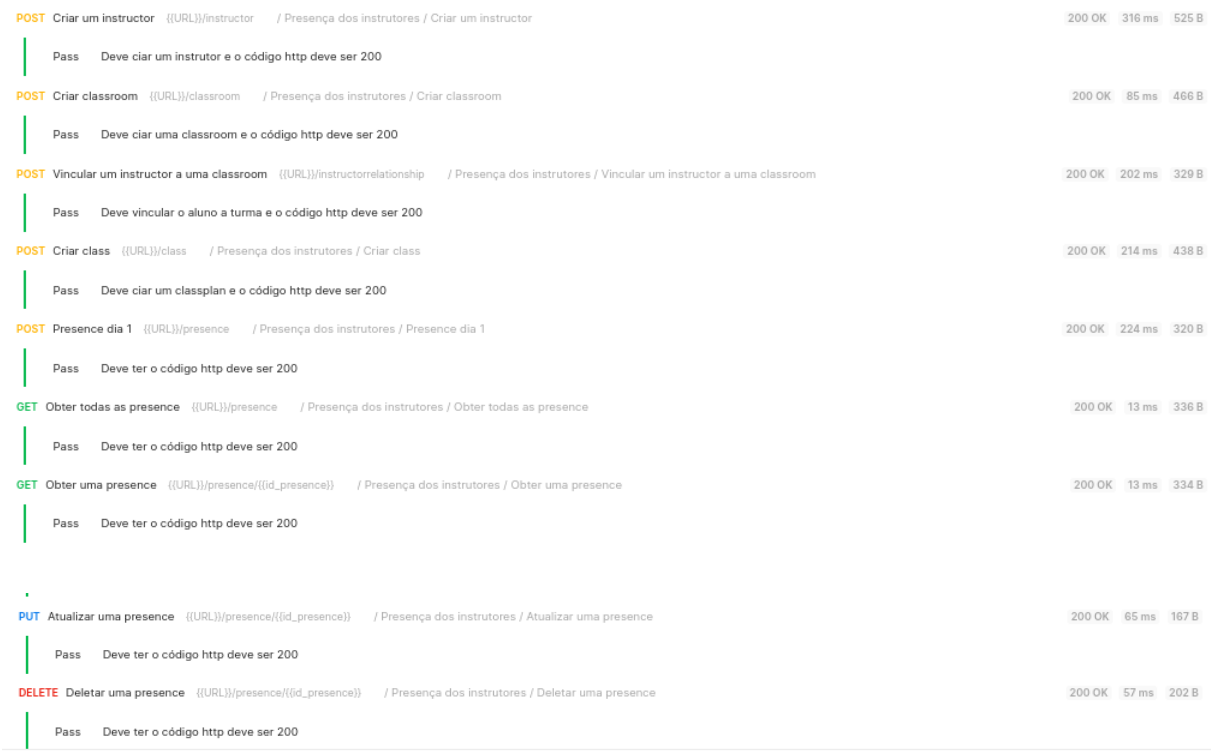
- Criação de frequência
- Obter todas as frequências do sistema
- Obter uma frequência do sistema
- Atualização de frequência
- Deletar frequência



Presença dos instrutores

Funcionalidades:

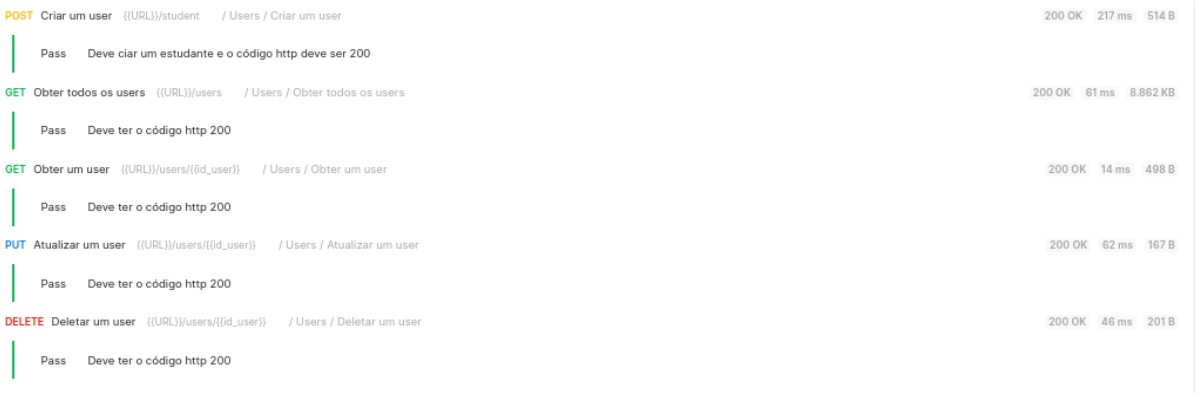
- Criação de presença
- Obter todas as presenças do sistema
- Obter uma presença do sistema
- Atualização de presença
- Deletar presença



Todos os usuários do sistema

Funcionalidades:

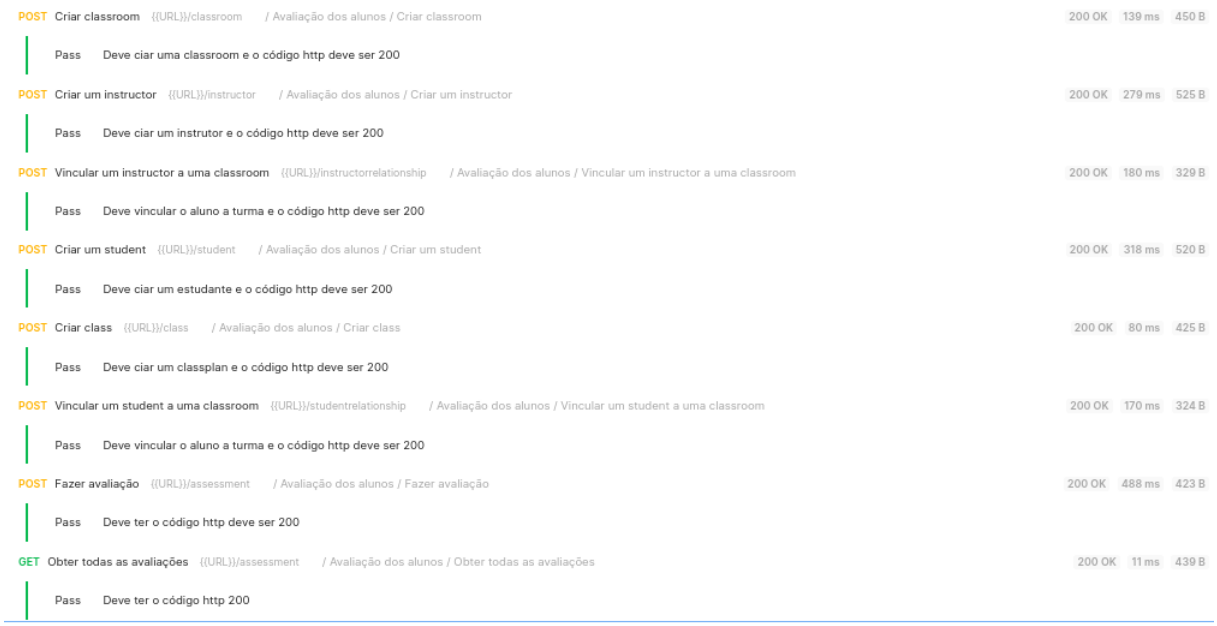
- Obter todos os usuários do sistema
- Obter um usuário do sistema
- Atualização de usuário
- Deletar usuário

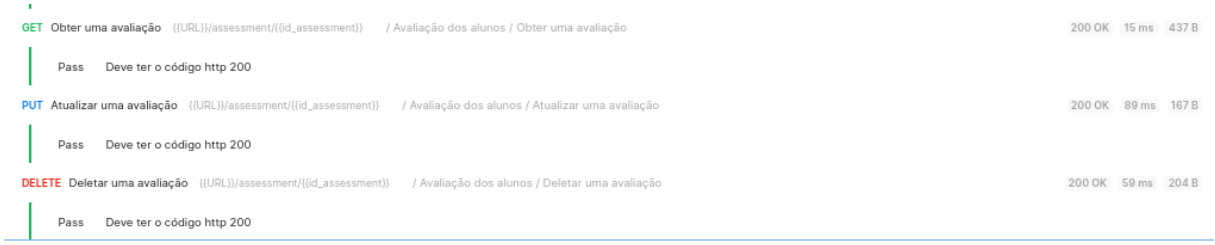


Avaliação dos alunos

Funcionalidades:

- Fazer avaliação
- Obter todas as avaliações do sistema
- Obter uma avaliação do sistema
- Atualização de avaliação
- Deletar avaliação

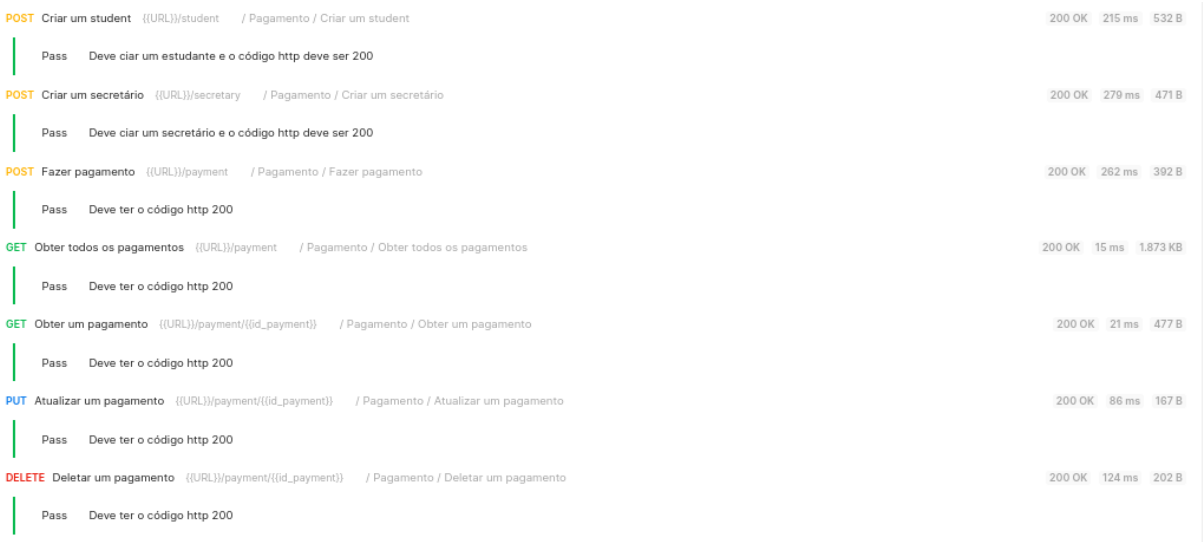




Pagamentos

Funcionalidades:

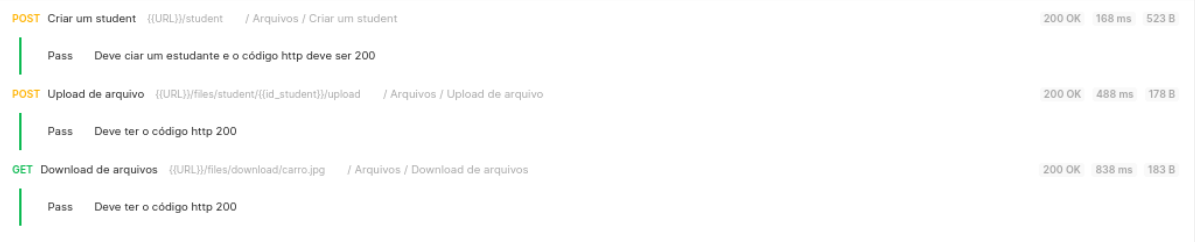
- Fazer pagamento
- Obter todos os pagamentos do sistema
- Obter um pagamento do sistema
- Atualização de pagamento
- Deletar pagamento



Upload e Download de arquivos

Funcionalidades:

- Fazer upload de um arquivo
- Fazer download de um arquivo



Upload e Download de recibos

Funcionalidades:

- Fazer upload de um recibo
- Fazer download de um recibo