

Documentação

Plataforma StudyTech -

backend

Danilo Soares da Silva

Lucas Souza Davanço

Paloma Cristina da Silva Correa

Lucas dos Santos Melo

Gabriel Antunes

Versão 1.0

Sumário

Tech Challenge - Pós-Tech FSDT – FIAP.....	3
Membros do grupo	3
Tech Challenge	3
O problema	3
Requisitos funcionais:.....	3
Requisitos técnicos:.....	4
Acesso ao projeto	5
O desenvolvimento	6
Organização das tarefas.....	6
Desafios encontrados	6
A aplicação.....	6
Arquitetura do sistema	6
Swagger	7
Uso da API.....	7
Rota POSTS - /posts	7
GetAll(GET) - /.....	7
UpdateById(PUT) - /:id.....	9
DeleteById(DELETE) - /:id	9
Rota Users- /users	12
GetAll(GET) - /.....	12
GetById(GET) - /:id	13
Create(POST) - /.....	14
Update(PUT) - /:id	14
DeleteById(DELETE) - /:id	15
Testes	15

Tech Challenge - Pós-Tech FSDT – FIAP

Este projeto foi desenvolvido como desafio proposto no Tech-Challenge da fase 2 do curso de pós-graduação em FullStack Development.

Membros do grupo

Danilo Soares da Silva - **RM:354317;**

Lucas Souza Davanso – **RM: 354925;**

Paloma Cristina da Silva Correa – **RM:355519;**

Lucas dos Santos Melo – **RM:355274;**

Gabriel Antunes – **RM: 354712.**

Tech Challenge

O problema

Atualmente, a maior parte de professores e professoras da rede pública de educação não têm plataformas onde postar suas aulas e transmitir conhecimento para alunos e alunas de forma prática, centralizada e tecnológica.

Para solucionar esse problema, nós utilizamos os conhecimentos adquiridos na última fase para auxiliar a nossa comunidade com a criação de uma aplicação de blogging dinâmico, utilizando a plataforma OutSystems. A plataforma foi um sucesso e, agora, nossa aplicação vai escalar para um panorama nacional. Portanto, precisaremos refatorar nosso Back-end, utilizando a plataforma de desenvolvimento node.js, e precisaremos persistir esses dados em um banco de dados, seja ele SQL ou NoSQL, de acordo com a decisão do grupo.

Requisitos funcionais:

Os seguintes endpoints REST serão implementados para a aplicação de blogging:

1. GET /posts - Lista de Posts:
 - a. Este endpoint permitirá aos alunos visualizarem uma lista de todos os posts disponíveis na página principal.

2. GET /posts/:id - Leitura de Posts:
 - a. Ao acessar este endpoint com um ID específico de post, os alunos poderão ler o conteúdo completo desse post.
3. POST /posts - Criação de Postagens:
 - a. Permite que professores criem novas postagens. Este endpoint aceitará dados como título, conteúdo e autor no corpo da requisição.
4. PUT /posts/:id - Edição de Postagens:
 - a. Usado para editar uma postagem existente. Professores deverão fornecer o ID do post que desejam editar e os novos dados no corpo da requisição.
5. GET /posts/admin - Listagem de Todas as Postagens (Visão Administrativa):
 - a. Este endpoint permitirá que professores vejam todas as postagens criadas, facilitando a gestão do conteúdo.
6. DELETE /posts/:id - Exclusão de Postagens:
 - a. Permite que professores excluam uma postagem específica, usando o ID do post como parâmetro.
7. GET /posts/search - Busca de Posts:
 - a. Este endpoint permitirá a busca de posts por palavras-chave. Os usuários poderão passar uma query string com o termo de busca e o sistema retornará uma lista de posts que contêm esse termo no título ou conteúdo.

Requisitos técnicos:

1. Back-end em Node.js:
 - a. Implementação do servidor usando Node.js.
 - b. Utilização de frameworks como Express para roteamento e middleware.
2. Persistência de Dados:
 - a. Utilização de um sistema de banco de dados (por exemplo, MongoDB, PostgreSQL).
 - b. Implementação de modelos de dados adequados para as postagens.
3. Containerização com Docker:
 - a. Desenvolvimento e implantação usando contêineres Docker para garantir consistência entre ambientes de desenvolvimento e produção.
4. Automação com GitHub Actions:
 - a. Configuração de workflows de CI/CD para automação de testes e deploy.
5. Documentação:
 - a. Documentação técnica detalhada do projeto, incluindo setup inicial, arquitetura da aplicação e guia de uso das APIs.
6. Cobertura de Testes:
 - a. O projeto deve garantir que pelo menos 30% do código seja coberto por testes unitários. Essa medida é essencial para assegurar a qualidade e a

estabilidade do código, especialmente em funções críticas como criação, edição e exclusão de postagens.

Todos os endpoints que modificam dados (POST, PUT, DELETE) devem incluir autenticação e autorização adequadas para garantir que apenas usuários autorizados (professores) possam realizar essas operações.

Acesso ao projeto

Como parte da entrega dos artefatos do Tech Challenge 2, está o acesso aos avaliadores à documentação e ao repositório do github.

Repositório no GitHub: <https://github.com/LpldFiap/StudyTech>

O desenvolvimento

Organização das tarefas

Para organizar e determinar as tarefas para a realização do projeto, foi utilizado o Discord para marcar reuniões e discutir as abordagens que iríamos usar no desenvolvimento. Também utilizamos o Trello para gerenciar as tarefas. No Trello, listamos todas as atividades que precisavam ser feitas, designamos responsabilidades e estabelecemos prazos.

Desafios encontrados

Nesse segundo Tech Challenge, o principal desafio foi o desenvolvimento em si e a decisão pelas tecnologias envolvidas. A necessidade de refatorar o Back-end com Node.js apresentou questões sobre arquitetura e desempenho, exigindo uma escolha que possa garantir que o backend suporte um volume maior de usuários sem perder eficiência. Além disso, a escolha entre bancos de dados SQL ou NoSQL trouxe discussões para o grupo, já que cada opção possui suas próprias vantagens e desvantagens em termos de flexibilidade, escalabilidade e manutenção.

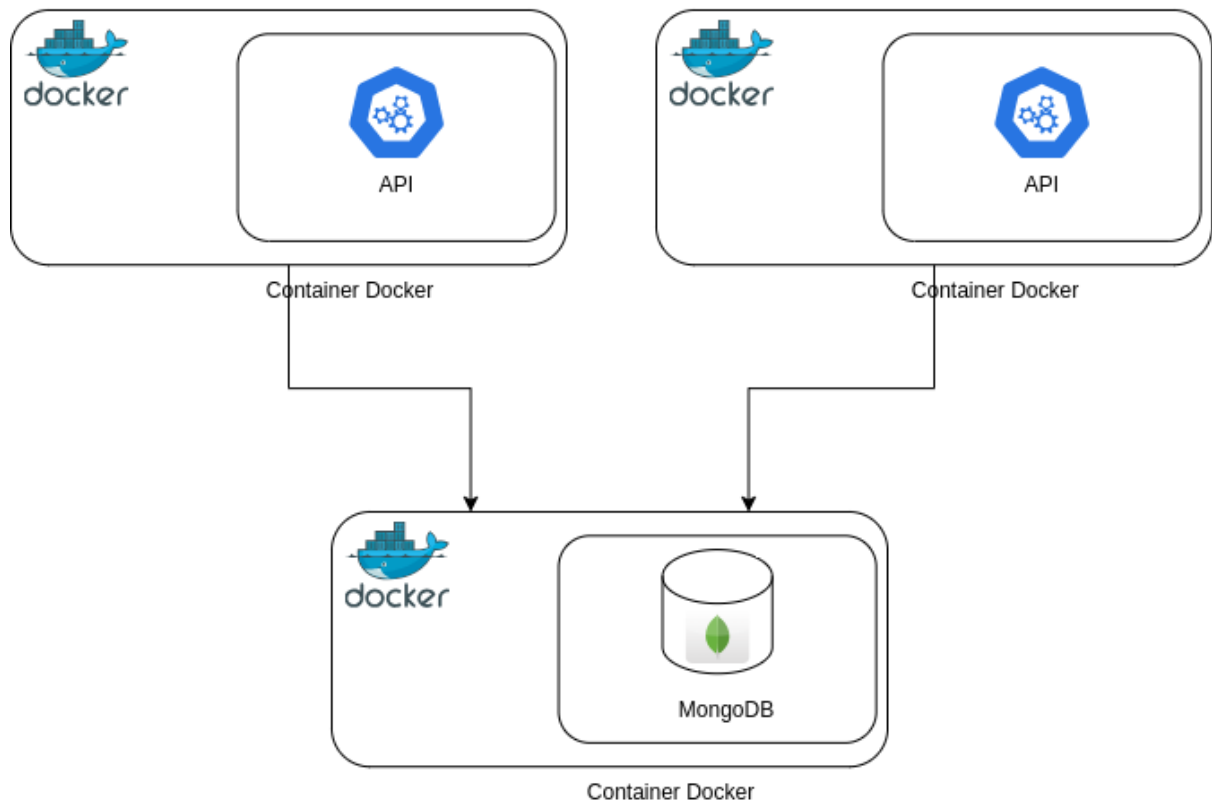
A aplicação

Nesse segundo Tech Challenge, o desafio proposto consiste em refatorar o atual backend da aplicação StudyTech que foi desenvolvido na plataforma OutSystems, tornando-o escalável, utilizando o framework Node.js e persistindo os dados no MongoDB.

Arquitetura do sistema

A API foi criada para ser escalável, de forma que quando necessário suporte um alto nível de requisições. Para isso, foi considerado a utilização do Docker como ferramenta de virtualização para aumentar ou diminuir a quantidade de instâncias de acordo com a demanda necessária.

TechChallenge 2



Swagger

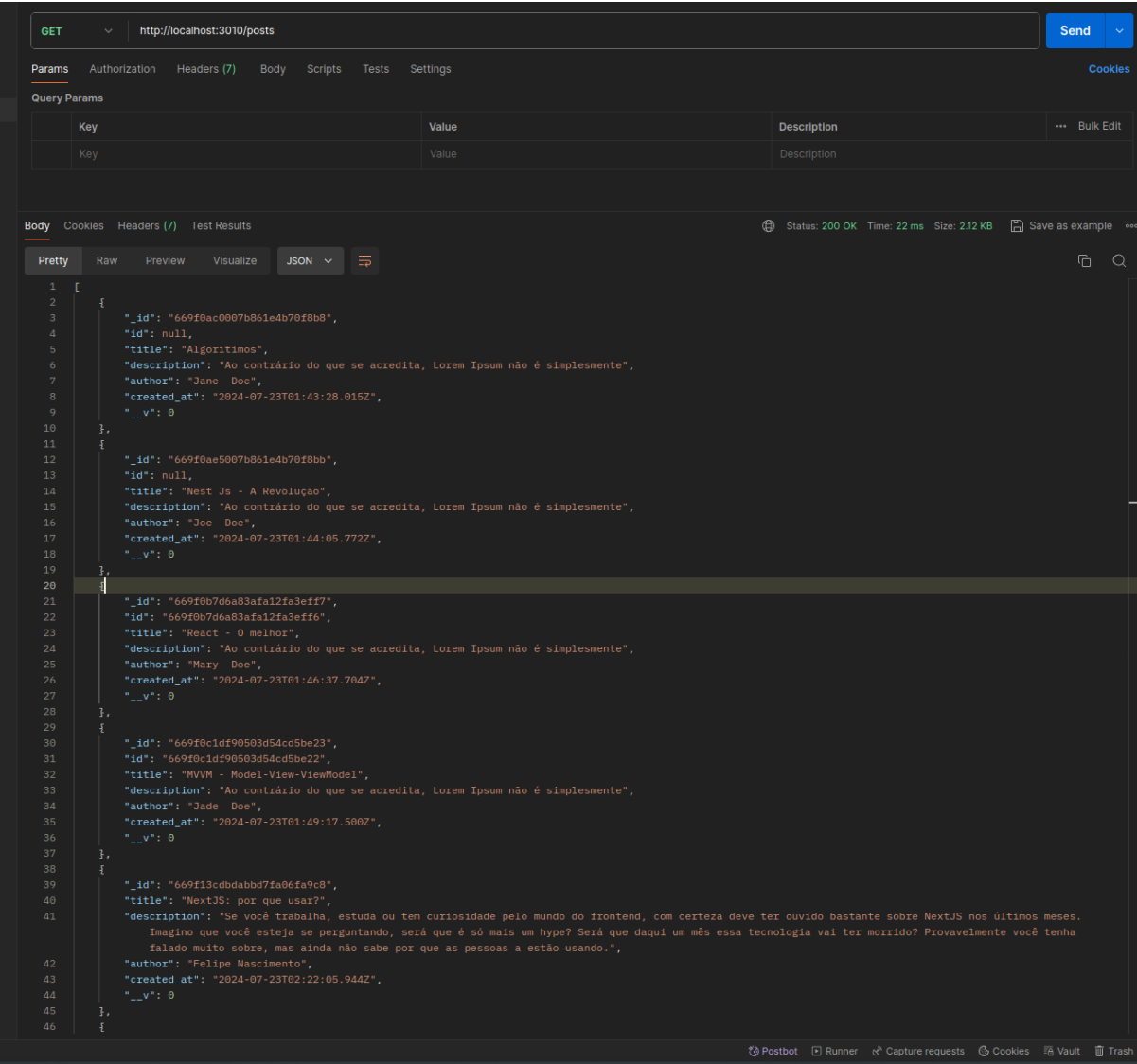
É possível acessar a documentação swagger ao carregar a rota “/api” para visualizar as chamadas que poderão ser realizadas na API

Uso da API

Rota POSTS - /posts

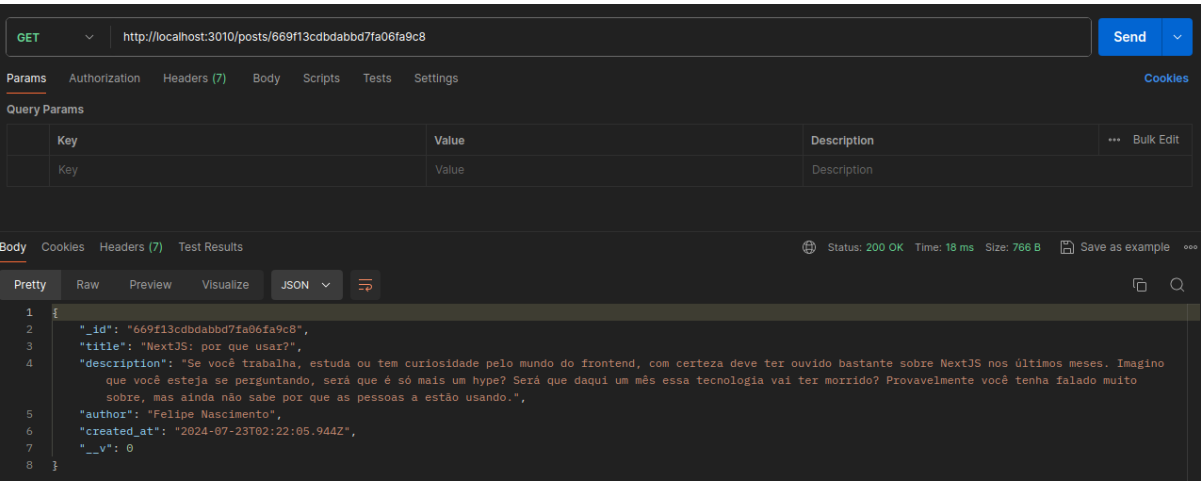
GetAll(GET) - /

Retorna todos os posts salvos na aplicação



GetById(GET) - /:id

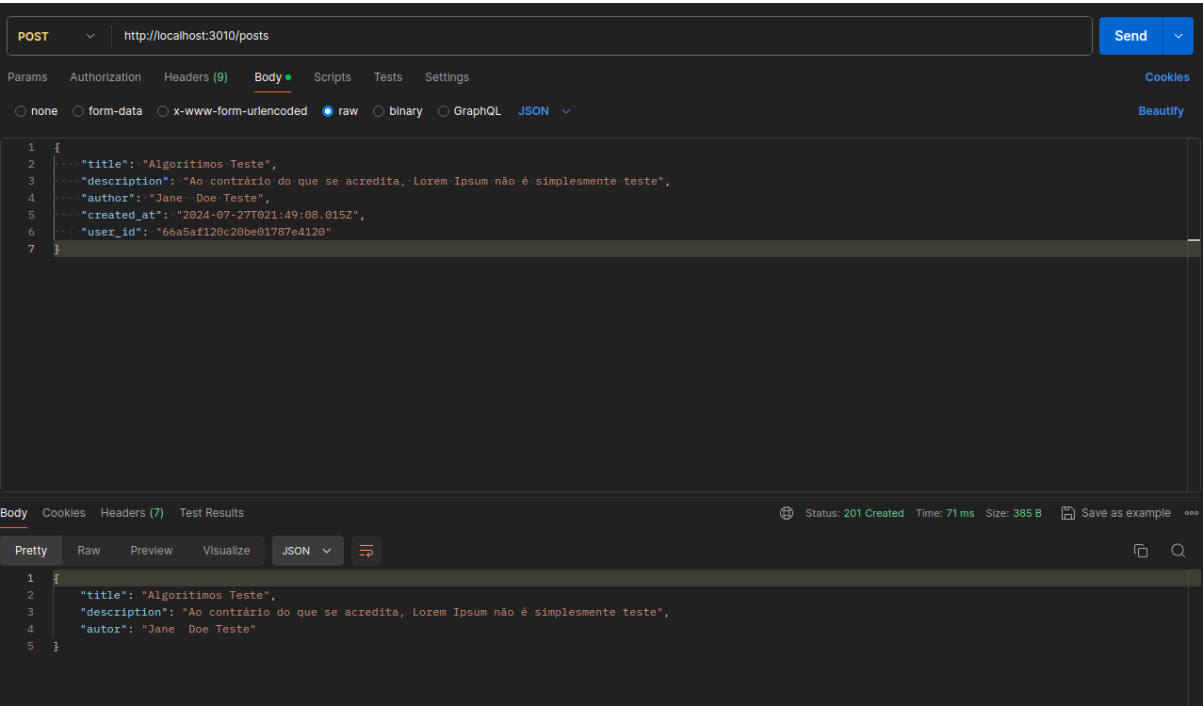
Retorna um post específico passando o id como parâmetro



Create (POST) - /

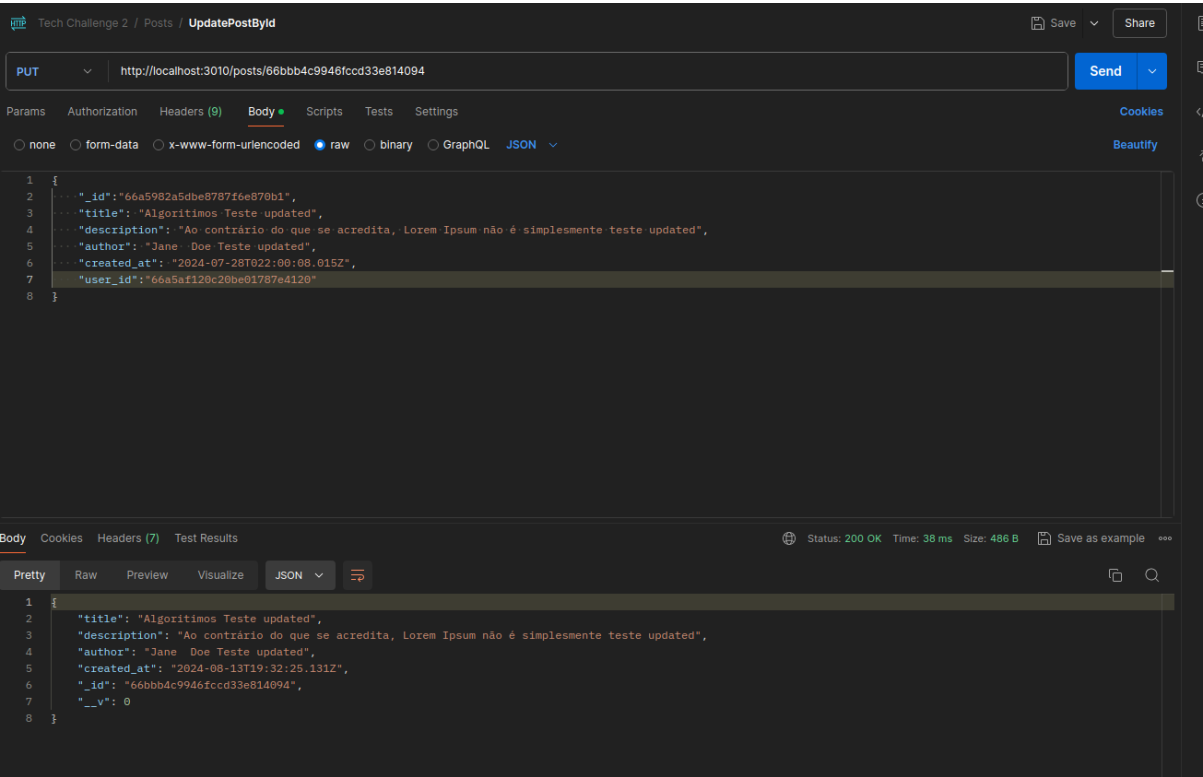
Versão 1.0

Cria um novo post.



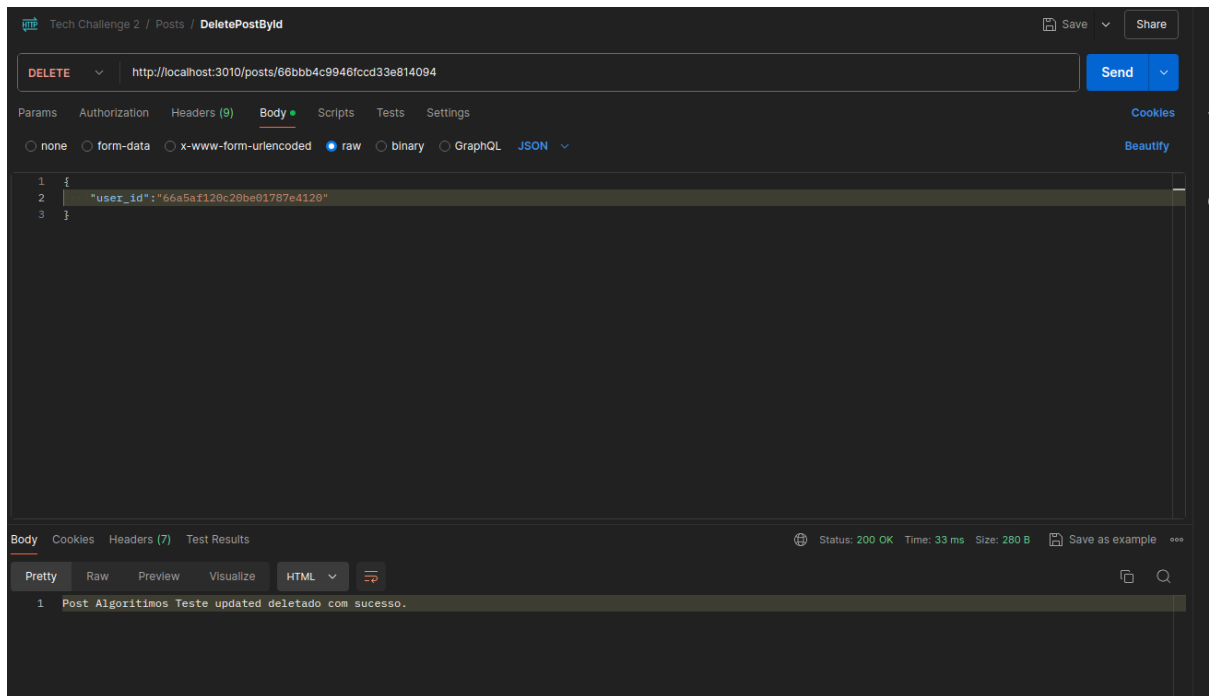
UpdateById(PUT) - /:id

Atualiza um post existente baseado no id



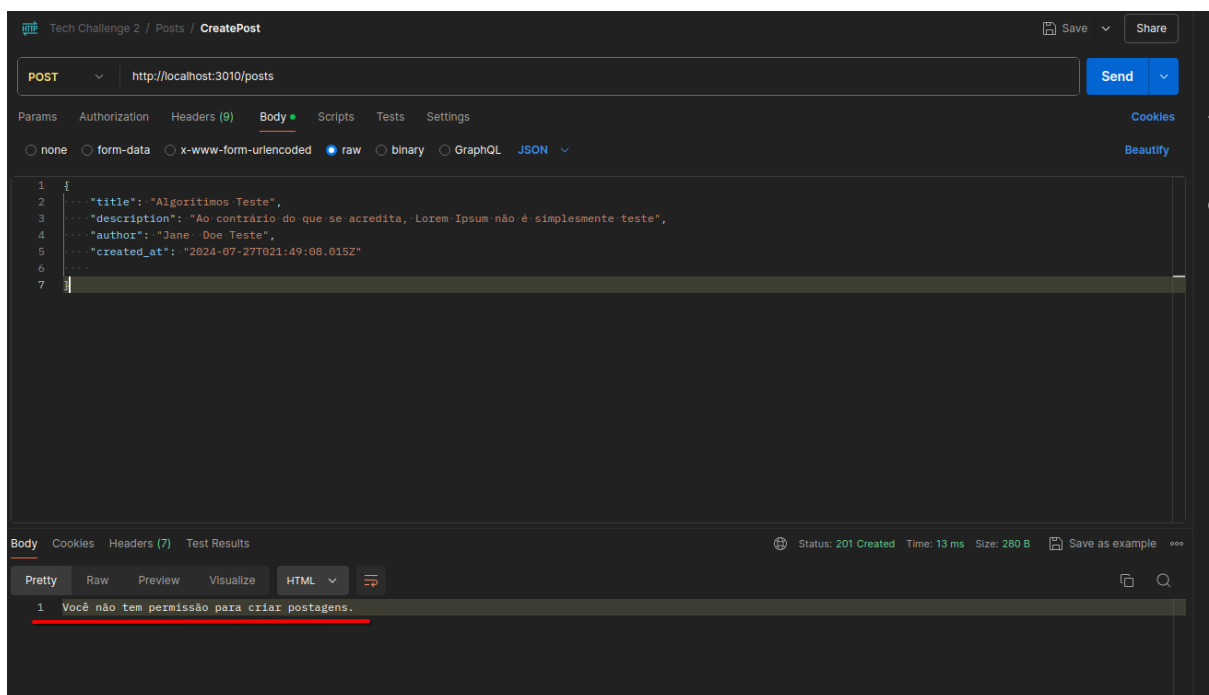
DeleteById(DELETE) - /:id

Deleta um post existente baseado no id.



Somente usuários cadastrados como professor poderá operar nas rotas de criação, deleção e atualização de posts. Para garantir que apenas professores possam utilizar essas rotas, será necessário passar o id do usuário no corpo da requisição, caso não seja passado ou o id passado seja de um aluno, a API retornará uma mensagem informando que não foi permitido realizar a operação:

Sem usuário:



Usuário com perfil de estudante:

Tech Challenge 2 / Users / GetUserById

GET http://localhost:3010/users/66bbb875946fccd33e8140a7 Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 9 ms Size: 413 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "_id": "66bbb875946fccd33e8140a7",
3   "name": "Estudante 01",
4   "email": "estudante@estudante.com",
5   "password": "123teste",
6   "role": "student",
7   "created_at": "2024-08-13T19:48:05.032Z",
8   "__v": 0
9 }
```

Tech Challenge 2 / Posts / CreatePost

POST http://localhost:3010/posts Send

Params Authorization Headers (9) Body Scripts Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautiful

```
1 {
2   "title": "Algoritmos Teste",
3   "description": "Ao contrário do que se acredita, Lorem Ipsum não é simplesmente teste",
4   "author": "Jane Doe Teste",
5   "created_at": "2024-07-27T021:49:08.015Z",
6   "user_id": "66bbb875946fccd33e8140a7",
7 }
```

Body Cookies Headers (7) Test Results

Status: 201 Created Time: 15 ms Size: 280 B Save as example

Pretty Raw Preview Visualize HTML

```
1 Você não tem permissão para criar postagens.
```

Usuário com perfil de professor:

Tech Challenge 2 / Users / GetUserById

GET http://localhost:3010/users/66bbb8da946fccd33e8140ac Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (7) Test Results Status: 200 OK Time: 13 ms Size: 413 B Save as example

Pretty Raw Preview Visualize JSON Bulk Edit

```
1 {
2   "id": "66bbb8da946fccd33e8140ac",
3   "name": "Professor 01",
4   "email": "professor@professor.com",
5   "password": "123teste",
6   "role": "teacher",
7   "created_at": "2024-08-13T19:49:46.640Z",
8   "__v": 0
9 }
```

Tech Challenge 2 / Posts / CreatePost

POST http://localhost:3010/posts Send

Params Authorization Headers (9) Body Scripts Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

```
1 {
2   "title": "Algoritmos Teste",
3   "description": "Ao contrário do que se acredita, Lorem Ipsum não é simplesmente teste",
4   "author": "Jane Doe Teste",
5   "created_at": "2024-07-27T021:49:08.015Z",
6   "user_id": "66bbb8da946fccd33e8140ac",
7 }
```

Body Cookies Headers (7) Test Results Status: 201 Created Time: 29 ms Size: 385 B Save as example

Pretty Raw Preview Visualize JSON Bulk Edit

```
1 {
2   "title": "Algoritmos Teste",
3   "description": "Ao contrário do que se acredita, Lorem Ipsum não é simplesmente teste",
4   "autor": "Jane Doe Teste"
5 }
```

Rota Users- /users

GetAll(GET) - /

Retorna todos os usuários cadastrados na aplicação

Tech Challenge 2 / Users / GetAllUsers

GET http://localhost:3010/users Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (7) Test Results

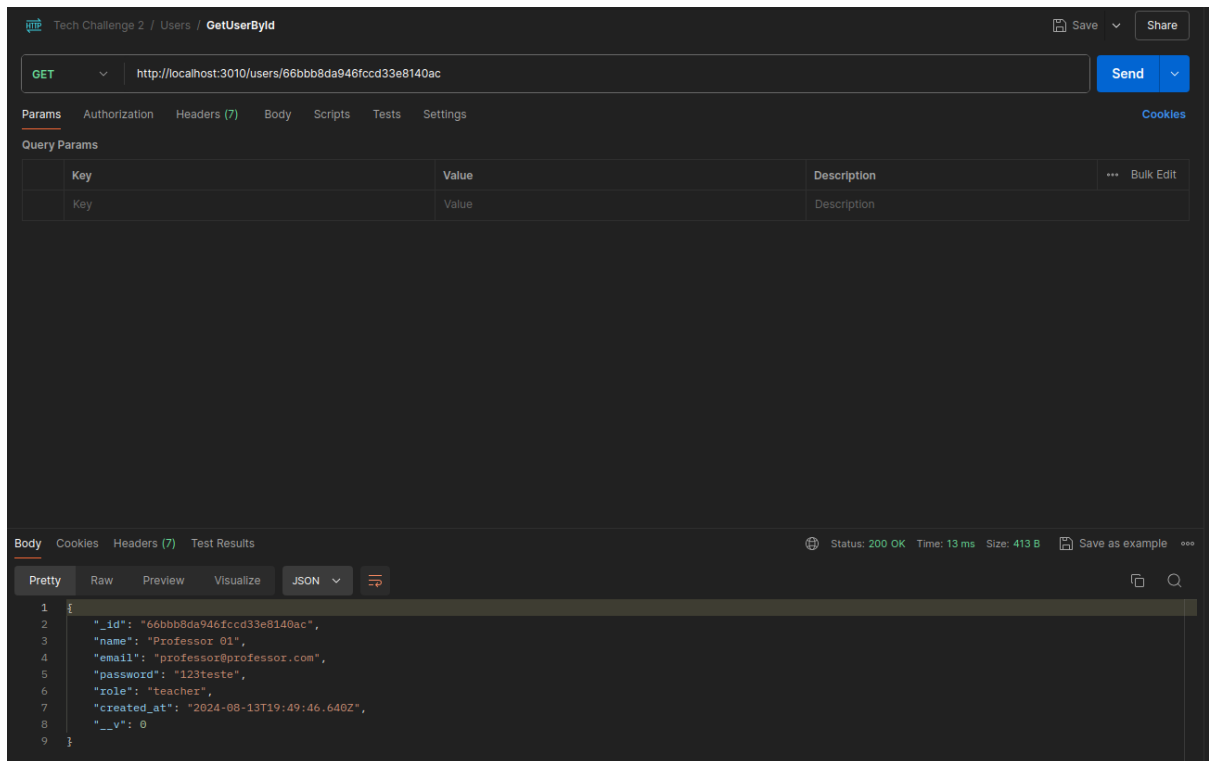
Status: 200 OK Time: 259 ms Size: 760 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "_id": "66a5af120c20be01787e4120",
4     "name": "Danilo",
5     "email": "danilo@danilo.com",
6     "password": "123teste",
7     "role": "teacher",
8     "created_at": "2024-07-28T02:38:10.223Z",
9     "__v": 0
10  },
11  {
12    "_id": "66bbb875946fcd33e8140a7",
13    "name": "Estudante 01",
14    "email": "estudante@estudante.com",
15    "password": "123teste",
16    "role": "student",
17    "created_at": "2024-08-13T19:48:05.032Z",
18    "__v": 0
19  },
20  {
21    "_id": "66bbb8da946fcd33e8140a8",
22    "name": "Professor 01",
23    "email": "professor@professor.com",
24    "password": "123teste",
25    "role": "teacher",
26    "created_at": "2024-08-13T19:49:46.640Z",
27    "__v": 0
28  }
29 ]
```

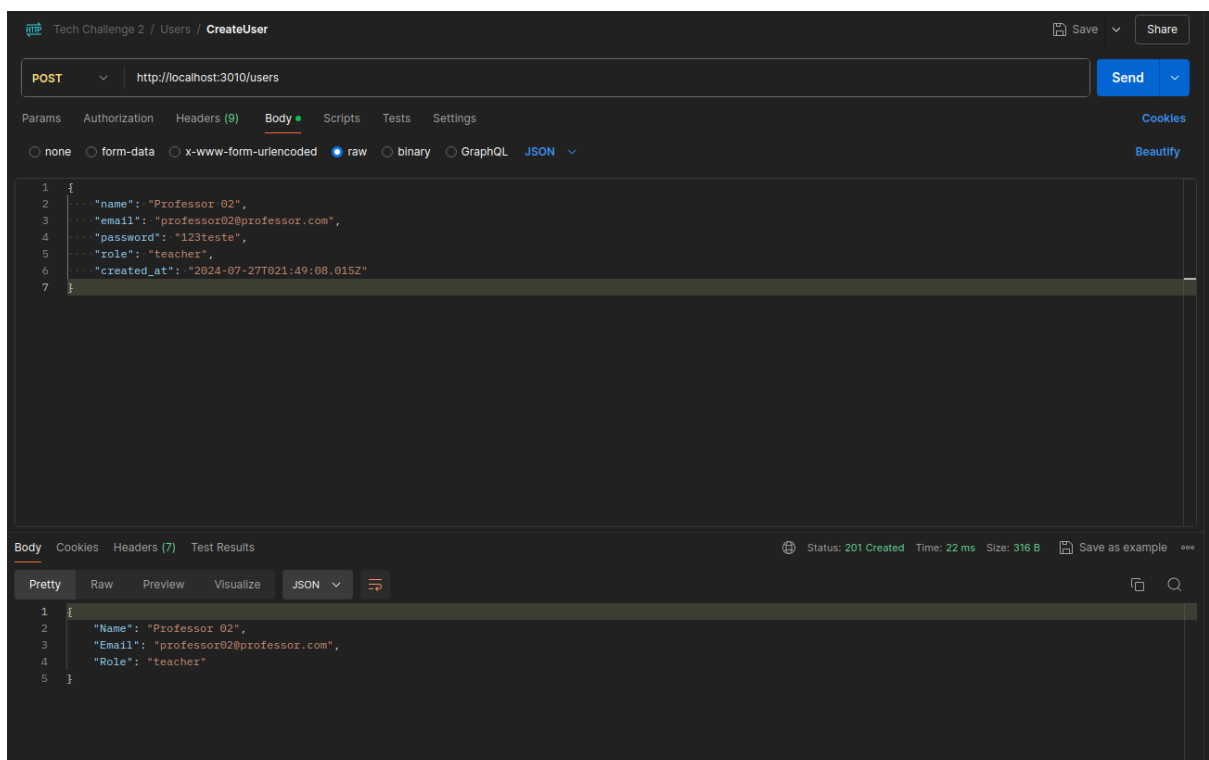
GetById(GET) - /:id

Retorna um usuário específico passando o id como parâmetro



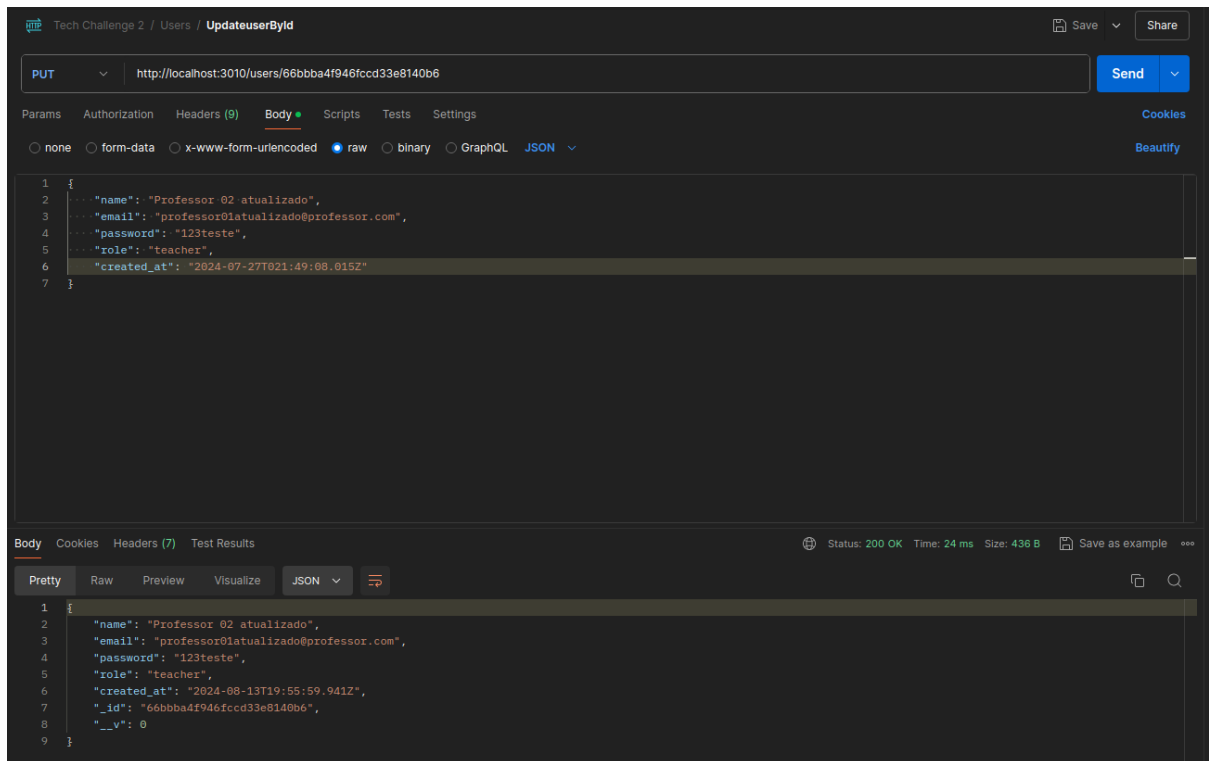
Create(POST) - /

Cria um novo usuário



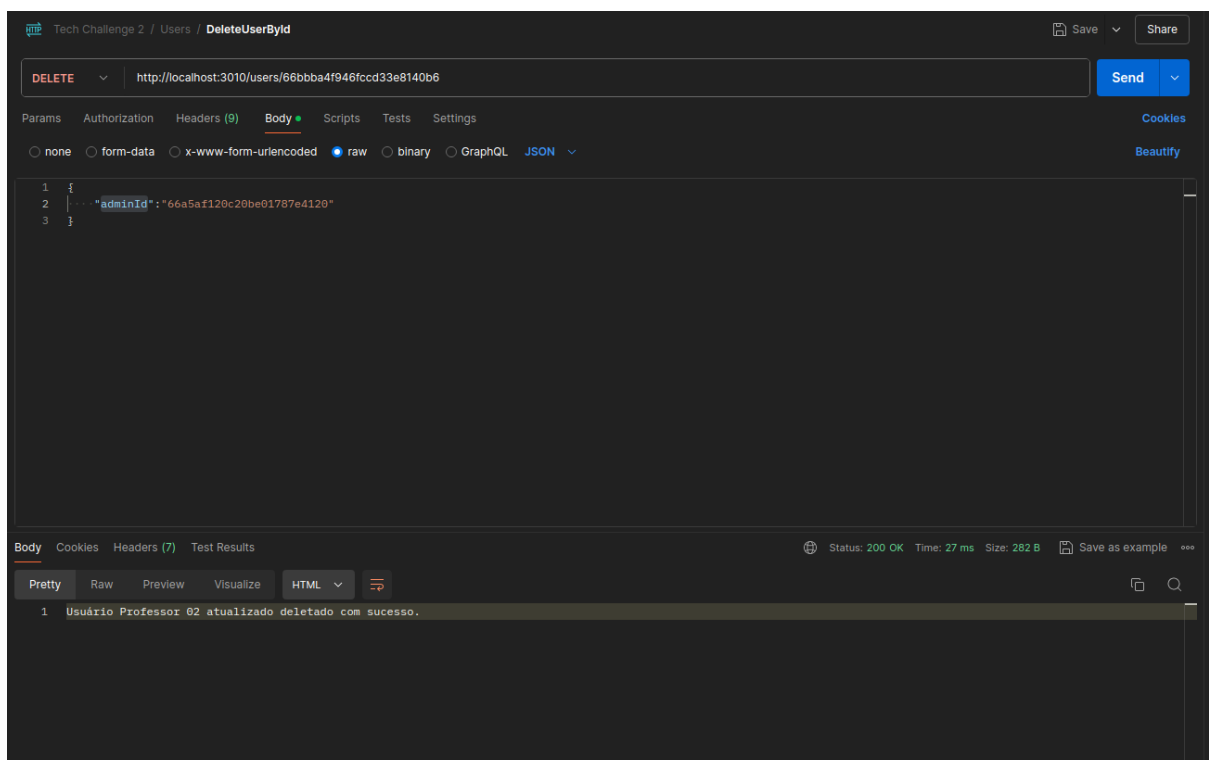
Update(PUT) - /:id

Atualiza um usuário existente baseado no id



DeleteById(DELETE) - /:id

Deleta um usuário existente baseado no id



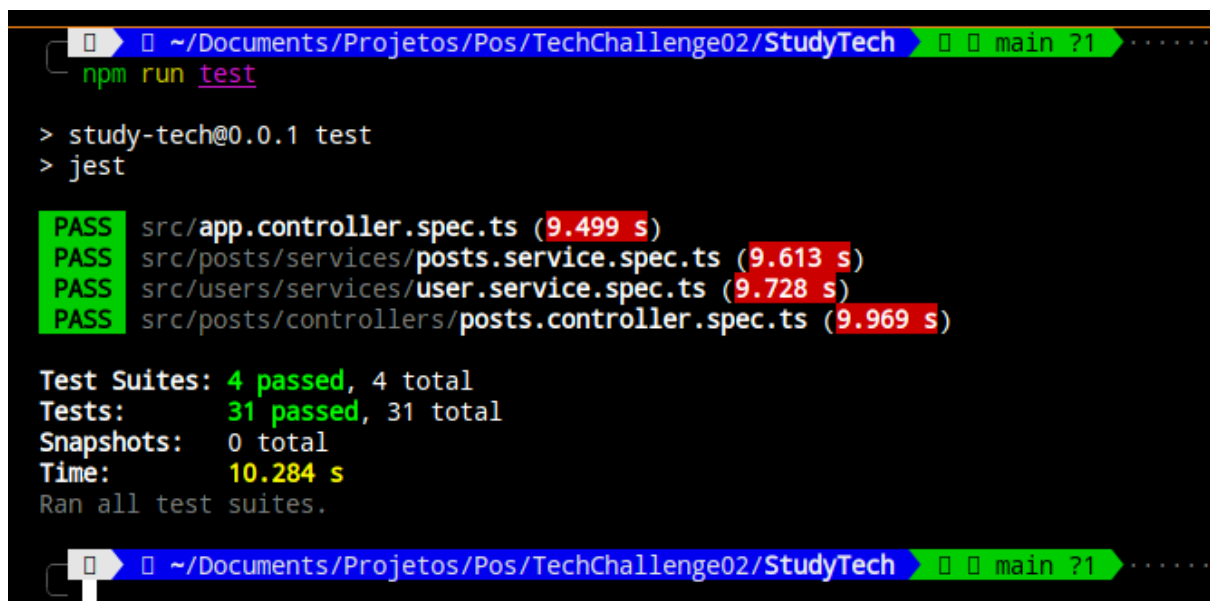
Testes

Para garantir a qualidade e a confiabilidade do código, utilizamos a ferramenta Jest para a realização de testes unitários. O Jest é amplamente reconhecido por sua

eficiência em testes de aplicações JavaScript, permitindo uma cobertura abrangente de código e a detecção rápida de possíveis regressões.

Durante o processo de desenvolvimento, implementamos uma série de testes unitários. Esses testes foram projetados para validar o comportamento esperado de cada componente, assegurando que todas as funções e métodos operem corretamente sob diferentes cenários e condições.

Os testes foram bem-sucedidos, alcançando uma cobertura de 100%.



```
~/Documents/Projetos/Pos/TechChallenge02/StudyTech main ?1
npm run test

> study-tech@0.0.1 test
> jest

PASS src/app.controller.spec.ts (9.499 s)
PASS src/posts/services/posts.service.spec.ts (9.613 s)
PASS src/users/services/user.service.spec.ts (9.728 s)
PASS src/posts/controllers/posts.controller.spec.ts (9.969 s)

Test Suites: 4 passed, 4 total
Tests: 31 passed, 31 total
Snapshots: 0 total
Time: 10.284 s
Ran all test suites.
```