

tryber / **sd-010-a-project-blogs-api** Publicgenerated from [betrybe/sd-0x-project-blogs-api](#)

★ 4 stars 🍴 3 forks

★ Star

🔔 Stop ignoring ▼

Code

Issues

Pull requests 111

Actions

Projects

Wiki

Security

Insights

🔗 master ▼

...

**GabrielCoruja** ...

on 9 Oct 🕒

[View code](#)

☰ README.md



Termos e acordos

Ao iniciar este projeto, você concorda com as diretrizes do Código de Ética e Conduta e do Manual da Pessoa Estudante da Trybe.

Boas vindas ao repositório do API de Blogs!

Você já usa o GitHub diariamente para desenvolver os exercícios, certo? Agora, para desenvolver os projetos, você deverá seguir as instruções a seguir. Fique atento a cada passo, e se tiver qualquer dúvida, nos envie por Slack! #vqv 🚀

Aqui você vai encontrar os detalhes de como estruturar o desenvolvimento do seu projeto a partir deste repositório, utilizando uma branch específica e um Pull Request para colocar seus códigos.

Sumário

- [Habilidades](#)
- [Entregáveis](#)

- O que deverá ser desenvolvido
- Desenvolvimento
- Data de entrega
- Instruções para entregar seu projeto
 - Antes de começar a desenvolver
 - Durante o desenvolvimento
 - Execução de testes unitários
- Como desenvolver
 - Linter
- Requisitos do projeto
 - Antes de começar
 - Observações importantes
 - Dicas
 - Lista de Requisitos
 - 1 - Sua aplicação deve ter o endpoint POST /user
 - 2 - Sua aplicação deve ter o endpoint POST /login
 - 3 - Sua aplicação deve ter o endpoint GET /user
 - 4 - Sua aplicação deve ter o endpoint GET /user/:id
 - 5 - Sua aplicação deve ter o endpoint POST /categories
 - 6 - Sua aplicação deve ter o endpoint GET /categories
 - 7 - Sua aplicação deve ter o endpoint POST /post
 - 8 - Sua aplicação deve ter o endpoint GET /post
 - 9 - Sua aplicação deve ter o endpoint GET post/:id
 - 10 - Sua aplicação deve ter o endpoint PUT /post/:id
 - Requisitos Bônus
 - 11 - Sua aplicação deve ter o endpoint DELETE post/:id
 - 12 - Sua aplicação deve ter o endpoint DELETE /user/me
 - 13 - Sua aplicação deve ter o endpoint GET post/search?q=:searchTerm
- Avisos Finais

Habilidades

Nesse projeto, você vai construir um back-end usando `ORM` com o pacote `sequelize` do `npm`, e será capaz de:

- Criar e associar tabelas usando `models` do `sequelize`
- Construir endpoints para consumir os `models` que criar
- Fazer um `CRUD` com o `ORM`

Entregáveis

Para entregar o seu projeto você deverá criar um Pull Request neste repositório.

Lembre-se que você pode consultar nosso conteúdo sobre [Git & GitHub](#) sempre que precisar!

O que deverá ser desenvolvido

Você vai arquitetar, desenvolver uma API de um CRUD posts de blog (com o sequelize). Começando pela API, você vai desenvolver alguns endpoints (seguindo os princípios do REST) que estarão conectados ao seu banco de dados. Lembre-se de aplicar os princípios SOLID!

Primeiro, você irá criar uma tabela para os usuários que desejam se cadastrar na aplicação. Após isso, criará também uma tabela de Categorias para seus Posts e por fim a tabela de Posts será seu foco, guardando todas as informações dos posts realizados na plataforma. Essa é apenas uma recomendação!

Desenvolvimento

Você deve desenvolver uma aplicação em `Node.js` usando o pacote `sequelize` para fazer um `CRUD` de posts.

Para fazer um post é necessário usuário e login, portanto será trabalhada a **relação entre user e post**. Também será necessário a utilização de categorias para seus posts, assim trabalhando a relação de `posts para categorias` e de `categorias para posts`.

Data de Entrega

- Projeto individual.
- Serão 3 dias de projeto.
- Data de entrega para avaliação final do projeto: 22/10/2021 - 14:00h .

Instruções para entregar seu projeto:

ANTES DE COMEÇAR A DESENVOLVER:

1. Clone o repositório

- `git clone https://github.com/tryber/sd-010-a-project-blogs-api.git` .
- Entre na pasta do repositório que você acabou de clonar:
 - `cd sd-010-a-project-blogs-api`

2. Instale as dependências [**Caso existam**]

- `npm install`

3. Crie uma branch a partir da branch `master`

- Verifique que você está na branch `master`
 - Exemplo: `git branch`
- Se não estiver, mude para a branch `master`
 - Exemplo: `git checkout master`
- Agora crie uma branch à qual você vai submeter os `commits` do seu projeto
 - Você deve criar uma branch no seguinte formato: `nome-de-usuario-nome-do-projeto`
 - Exemplo: `git checkout -b joaozinho-sd-010-a-project-blogs-api`

4. Adicione as mudanças ao `stage` do Git e faça um `commit`

- Verifique que as mudanças ainda não estão no `stage`
 - Exemplo: `git status` (deve aparecer listada a pasta *joaozinho* em vermelho)
- Adicione o novo arquivo ao `stage` do Git
 - Exemplo:
 - `git add .` (adicionando todas as mudanças - *que estavam em vermelho* - ao `stage` do Git)
 - `git status` (deve aparecer listado o arquivo *joaozinho/README.md* em verde)
- Faça o `commit` inicial
 - Exemplo:
 - `git commit -m 'iniciando o projeto x'` (fazendo o primeiro `commit`)
 - `git status` (deve aparecer uma mensagem tipo *nothing to commit*)

5. Adicione a sua branch com o novo `commit` ao repositório remoto

- Usando o exemplo anterior: `git push -u origin joaozinho-sd-010-a-project-blogs-api`

6. Crie um novo `Pull Request` (*PR*)

- Vá até a página de *Pull Requests* do [repositório no GitHub](https://github.com/tryber/sd-010-a-project-blogs-api)
- Clique no botão verde *"New pull request"*

- Clique na caixa de seleção "*Compare*" e escolha a sua branch **com atenção**
- Clique no botão verde "*Create pull request*"
- Adicione uma descrição para o *Pull Request* e clique no botão verde "*Create pull request*"
- **Não se preocupe em preencher mais nada por enquanto!**
- Volte até a [página de Pull Requests do repositório](#) e confira que o seu *Pull Request* está criado

DURANTE O DESENVOLVIMENTO

- Faça `commits` das alterações que você fizer no código regularmente
- Lembre-se de sempre após um (ou alguns) `commits` atualizar o repositório remoto
- Os comandos que você utilizará com mais frequência são:
 - i. `git status` (para verificar o que está em vermelho - fora do stage - e o que está em verde - no stage)
 - ii. `git add` (para adicionar arquivos ao stage do Git)
 - iii. `git commit` (para criar um commit com os arquivos que estão no stage do Git)
 - iv. `git push -u nome-da-branch` (para enviar o commit para o repositório remoto na primeira vez que fizer o `push` de uma nova branch)
 - v. `git push` (para enviar o commit para o repositório remoto após o passo anterior)

Execução de testes unitários

Vamos usar o Jest para executar os testes, use o comando a seguir para executar todos os testes:

```
npm test
```

Caso queira executar só um arquivo de test use o seguinte comando, considerado que quer testar o arquivo `tests/req07-createPost.test.js` :

```
npm test tests/req07-createPost.test.js
```

ou

```
npm test req07
```

Como desenvolver

Linter

Para garantir a qualidade do código, usaremos o [ESLint](#) para fazer a sua análise estática.

Este projeto já vem com as dependências relacionadas ao *linter* configuradas nos arquivos `package.json` nos seguintes caminhos:

- `sd-010-a-project-blogs-api/package.json`

Para poder rodar os `ESLint` em um projeto basta executar o comando `npm install` dentro do projeto e depois `npm run lint`. Se a análise do `ESLint` encontrar problemas no seu código, tais problemas serão mostrados no seu terminal. Se não houver problema no seu código, nada será impresso no seu terminal.

Você também pode instalar o plugin do `ESLint` no `VSCode`, bastar ir em `extensions` e baixar o [plugin ESLint](#).

 **PULL REQUESTS COM ISSUES DE LINTER NÃO SERÃO AVALIADAS. ATENTE-SE PARA RESOLVÊ-LAS ANTES DE FINALIZAR O DESENVOLVIMENTO!** 

Requisitos do projeto:

Antes de começar:

 **Leia-os atentamente e siga à risca o que for pedido.** 

Observações importantes:

Em cada requisito você encontrará uma imagem de um protótipo de como sua aplicação deve ficar. Estilo da página não será avaliado.

O não cumprimento de um requisito, total ou parcialmente, impactará em sua avaliação.

Há um arquivo `index.js` no repositório. Não remova, nele, o seguinte trecho de código:

```
app.get('/', (request, response) => {  
  response.send();  
});
```

Você irá precisar configurar as variáveis globais do MySQL. Você pode usar esse [Conteúdo de variáveis de ambiente com NodeJS](#) como referência.

Faça essas configurações também para as variáveis de ambiente usadas nesses arquivo:

sd-010-a-project-blogs-api/config/config.js

```
module.exports = {
  development: {
    username: process.env.MYSQL_USER,
    password: process.env.MYSQL_PASSWORD,
    database: 'blogs_api',
    host: process.env.HOSTNAME,
    dialect: 'mysql',
  },
  test: {
    username: process.env.MYSQL_USER,
    password: process.env.MYSQL_PASSWORD,
    database: 'blogs_api',
    host: process.env.HOSTNAME,
    dialect: 'mysql',
  },
  production: {
    username: process.env.MYSQL_USER,
    password: process.env.MYSQL_PASSWORD,
    database: 'blogs_api',
    host: process.env.HOSTNAME,
    dialect: 'mysql',
  },
};
```

(Neste arquivo é obrigatório deixar o nome do database como "database": 'blogs_api')

É essencial usar essas 3 variáveis no arquivo acima:

Variáveis:

host: process.env.HOSTNAME

user: process.env.MYSQL_USER

password: process.env.MYSQL_PASSWORD

Com elas que iremos conseguir conectar ao banco do avaliador automático

Variável JWT (opcional):

JWT_SECRET

Também poderá ser utilizada esta variável de ambiente para o **SECRET** do JWT

Dicas

Status HTTP

Tenha em mente que todas as "respostas" devem respeitar os [status do protocolo HTTP](#) com base no que o REST prega.

Alguns exemplos:

- Requisições que precisam de token mas não o receberam devem retornar um código de status 401 ;
- Requisições que não seguem o formato pedido pelo servidor devem retornar um código de status 400 ;
- Um problema inesperado no servidor deve retornar um código de status 500 ;
- Um acesso ao criar um recurso, no nosso caso usuário ou post, deve retornar um código de status 201 .

Os seguintes pontos serão avaliados:

- O seu projeto deverá usar um ORM para criar e atualizar o seu banco. A clonagem do projeto seguida de um comando de migrate deve deixá-lo em sua forma esperada.
- Deve conter uma tabela chamada **Users**, contendo dados com a seguinte estrutura::

```
{
  "id": 1,
  "displayName": "Brett Wiltshire",
  "email": "brett@email.com", // tem que ser único
  "password": "123456",
  "image": "http://4.bp.blogspot.com/_YA50adQ-7vQ/S1gfR_6ufpI/AAAAAAAAAAk/
```

- Deve conter uma tabela chamada **Categories**, contendo dados com a seguinte estrutura::

```
{
  "id": 18,
  "name": "News"
}
```


- Deve conter uma tabela chamada **PostsCategories**, contendo dados com a seguinte estrutura:

```
{
  "postId": 50,
  "categoryId": 20
}
```

- Deve conter uma tabela chamada **BlogPosts**, contendo dados com a seguinte estrutura::

```
{
  "id": 21,
  "title": "Latest updates, August 1st",
  "content": "The whole text for the blog post goes here in this key",
  "userId": 14, // esse é o id que referência usuário que é o autor do pos
  "published": "2011-08-01T19:58:00.000Z",
  "updated": "2011-08-01T19:58:51.947Z",
}
```

Os dados acima são fictícios, e estão aqui apenas como exemplo

OBS: Os testes irão rodar através do seu migrate usando os seguintes comandos:

"drop": "npx sequelize-cli db:drop \$" -- Dropa o banco

"prestart": "npx sequelize-cli db:create && npx sequelize-cli db:migrate \$" -- Cria o banco e gera as tabelas

"seed": "npx sequelize-cli db:seed:all \$" , -- Insere dados na tabela

Então preste bastante atenção se estiver errado o avaliador não irá funcionar

Haverá um arquivo na pasta /seeders dentro dela irá conter as queries para inserir no banco não remova ela o avaliador irá usar ela .

Lista de Requisitos:

1 - Sua aplicação deve ter o endpoint POST /user

Os seguintes pontos serão avaliados:

- O endpoint deve ser capaz de adicionar um novo user a sua tabela no banco de dados;

- O corpo da requisição deverá ter o seguinte formato:

```
{
  "displayName": "Brett Wiltshire",
  "email": "brett@email.com",
  "password": "123456",
  "image": "http://4.bp.blogspot.com/_YA50adQ-7vQ/S1gfR_6ufpI/AAAAAAAAAAk/"
}
```

- O campo `displayName` deverá ser uma string com no mínimo de 8 caracteres;
- O campo `email` será considerado válido se tiver o formato `<prefixo>@<domínio>` e se for único. Ele é obrigatório.
- A senha deverá conter 6 caracteres. Ela é obrigatória.
- Caso exista uma pessoa com o mesmo email na base, deve-se retornar o seguinte erro:

```
{
  "message": "User already registered"
}
```

- Caso contrário, retornar a mesma resposta do endpoint de `/login`, um token `JWT`:

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJwYXlsb2FkIjp7ImlkIjo1L"
}
```

O token anterior é fictício

Além disso, as seguintes verificações serão feitas:

[Será validado que é possível cadastrar um usuário com sucesso]

Se o usuário for criado com sucesso o resultado retornado deverá ser conforme exibido abaixo, com um status `http 201`:

POST http://localhost:3000/user Send 201 Created 13.8 ms 257 B Just Now

JSON Auth Query Header Docs Preview Header Cookie Timeline

```
1 {
2   "displayName": "usuarioteste",
3   "email": "test@email.com",
4   "password": "123556",
5   "image": "null"
6 }
```

```
1 {
2   "token":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJwYXlsb2FkI
  jpb7ImklIjo2LCJkaXNwbGF5TmFtZSI6InVzdWYyaW90ZXN0ZSI6
  ImVtYWlsIjoiaGVzdEB1bWVpY20iLCJpbWFnZSI6Im51bGw
  ifSwiaWF0IjoxNjIwMDM5LCJleHAiOjE2MjA2NzY0MzI9.C
  XYzYxAtAtrKzdZAZJ4NUMAid8KsMVy2QMihAx_nfb8"
3 }
```

[Será validado que não é possível cadastrar usuário com o campo `displayName` menor que 8 caracteres]

Se o usuário tiver o campo "displayName" menor que 8 caracteres o resultado retornado deverá ser conforme exibido abaixo, com um status http 400 :

POST http://localhost:3000/user Send 400 Bad Request 1.87 ms 71 B Just Now

JSON Auth Query Header Docs Preview Header Cookie Timeline

```
1 {
2   "displayName": "usua",
3   "email": "test@email.com",
4   "password": "123556",
5   "image": "null"
6 }
```

```
1 {
2   "message": "\"displayName\" length must be at
  least 8 characters long"
3 }
```

(As contrabarras \ estão escapando as aspas de dentro da string)

[Será validado que não é possível cadastrar usuário com o campo `email` com formato `email: rubinho`]

Se o usuário tiver o campo "email" com o formato `email: rubinho` o resultado retornado deverá ser conforme exibido abaixo, com um status http 400 :

POST http://localhost:3000/user Send 400 Bad Request 2.08 ms 45 B Just Now

JSON Auth Query Header Docs Preview Header Cookie Timeline

```
1 {
2   "displayName": "usuaarioteste",
3   "email": "test",
4   "password": "123556",
5   "image": "null"
6 }
```

```
1 {
2   "message": "\"email\" must be a valid email"
3 }
```

(As contrabarras \ estão escapando as aspas de dentro da string)

[Será validado que não é possível cadastrar usuário com o campo `email` com formato `email: @gmail.com`]

Se o usuário tiver o campo "email" com o formato `email: @gmail.com` o resultado retornado deverá ser conforme exibido abaixo, com um status http 400 :

The screenshot shows a REST client interface. The top bar indicates a POST request to `http://localhost:3000/user` with a status of **400 Bad Request**, a response time of 1.45 ms, and a body size of 45 B. The left pane shows the JSON request body:

```
1 {
2   "displayName": "usuaarioteste",
3   "email": "@email.com",
4   "password": "123556",
5   "image": "null"
6 }
```

The right pane shows the JSON response body:

```
1 {
2   "message": "\"email\" must be a valid email"
3 }
```

(As contrabarras \ estão escapando as aspas de dentro da string)

[Será validado que o campo `email` é obrigatório]

Se o usuário não tiver campo "email" o resultado retornado deverá ser conforme exibido abaixo, com um status `http 400` :

The screenshot shows a REST client interface. The top bar indicates a POST request to `http://localhost:3000/user` with a status of **400 Bad Request**, a response time of 1.72 ms, and a body size of 35 B. The left pane shows the JSON request body:

```
1 {
2   "displayName": "usuaarioteste",
3   "password": "123556",
4   "image": "null"
5 }
```

The right pane shows the JSON response body:

```
1 {
2   "message": "\"email\" is required"
3 }
```

(As contrabarras \ estão escapando as aspas de dentro da string)

[Será validado que não é possível cadastrar usuário com o campo `password` diferente de 6 caracteres]

Se o usuário tiver o campo "password" menor ou maior que 6 caracteres o resultado retornado deverá ser conforme exibido abaixo, com um status `http 400` :

The screenshot shows a REST client interface. The top bar indicates a POST request to `http://localhost:3000/user` with a status of **400 Bad Request**, a response time of 1.54 ms, and a body size of 59 B. The left pane shows the JSON request body:

```
1 {
2   "displayName": "usuaarioteste",
3   "email": "test@email.com",
4   "password": "123",
5   "image": "null"
6 }
```

The right pane shows the JSON response body:

```
1 {
2   "message": "\"password\" length must be 6 characters long"
3 }
```

(As contrabarras \ estão escapando as aspas de dentro da string)

[Será validado que o campo `password` é obrigatório]

Se o usuário não tiver campo "password" o resultado retornado deverá ser conforme exibido abaixo, com um status `http 400` :

The screenshot shows a REST client interface. The top bar indicates a POST request to `http://localhost:3000/user` with a status of **400 Bad Request**, a response time of 2.29 ms, and a body size of 38 B. The left pane shows the request body in JSON format:

```
1 {
2   "displayName": "usuaarioteste",
3   "email": "test@email.com",
4   "image": "null"
5 }
```

The right pane shows the response body in JSON format:

```
1 {
2   "message": "\"password\" is required"
3 }
```

(As contrabarras \ estão escapando as aspas de dentro da string)

[Validar que não é possível cadastrar um usuário com email já existente]

Se o usuário cadastrar o campo "email" com um email que já existe, o resultado retornado deverá ser conforme exibido abaixo, com um status http 409 :

The screenshot shows a REST client interface. The top bar indicates a POST request to `http://localhost:3000/user` with a status of **409 Conflict**, a response time of 5.68 ms, and a body size of 37 B. The left pane shows the request body in JSON format:

```
1 {
2   "displayName": "usuaarioteste",
3   "email": "test@email.com",
4   "password": "123456",
5   "image": "null"
6 }
```

The right pane shows the response body in JSON format:

```
1 {
2   "message": "User already registered"
3 }
```

2 - Sua aplicação deve ter o endpoint POST /login

Os seguintes pontos serão avaliados:

- O corpo da requisição deverá seguir o formato abaixo:

```
{
  "email": "email@mail.com",
  "password": "123456"
}
```

- Caso algum desses campos seja inválido ou não exista um usuário correspondente no banco de dados, retorne um código de status 400 com o corpo `{ message: "Campos inválidos" }`.
- Caso esteja tudo certo com o login, a resposta deve ser um token JWT , no seguinte formato:

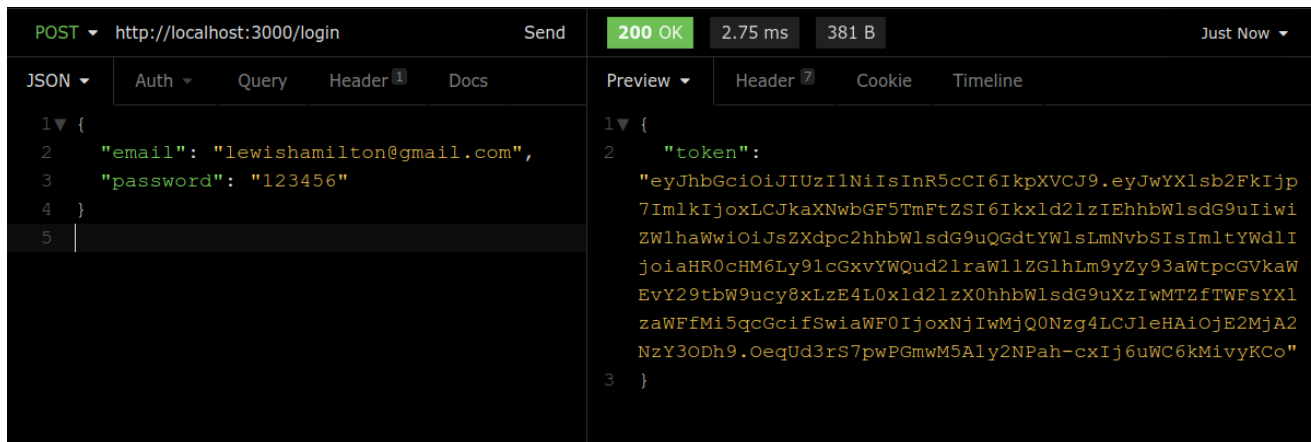
```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJwYXNwd291IjoiIiwiaWF0Ijoi1L"
}
```

O token anterior é fictício

Além disso, as seguintes verificações serão feitas:

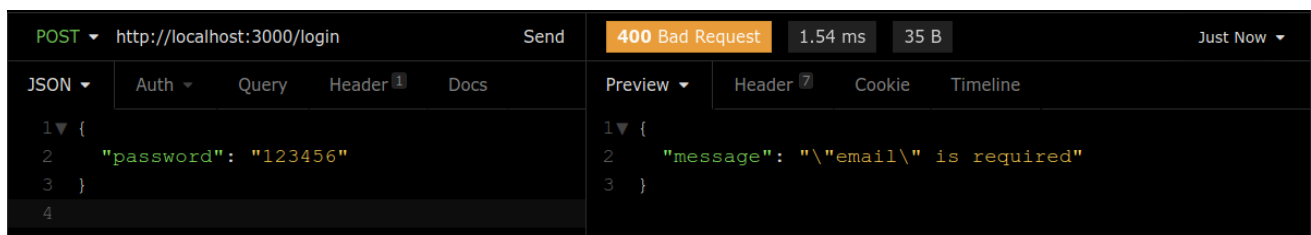
[Será validado que é possível fazer login com sucesso]

Se o login foi feito com sucesso o resultado retornado deverá ser conforme exibido abaixo, com um status http 200 :



[Será validado que não é possível fazer login sem o campo email]

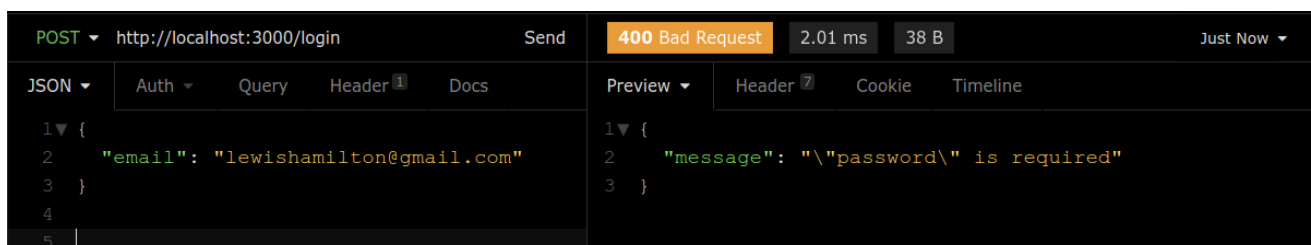
Se o login não tiver o campo "email" o resultado retornado deverá ser conforme exibido abaixo, com um status http 400 :



(As contrabarras \ estão escapando as aspas de dentro da string)

[Será validado que não é possível fazer login sem o campo password]

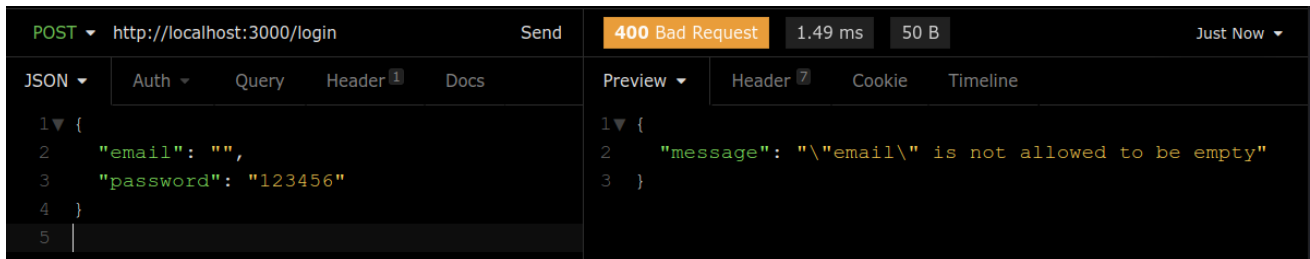
Se o login não tiver o campo "password" o resultado retornado deverá ser conforme exibido abaixo, com um status http 400 :



(As contrabarras \ estão escapando as aspas de dentro da string)

[Será validado que não é possível fazer login com o campo email em branco]

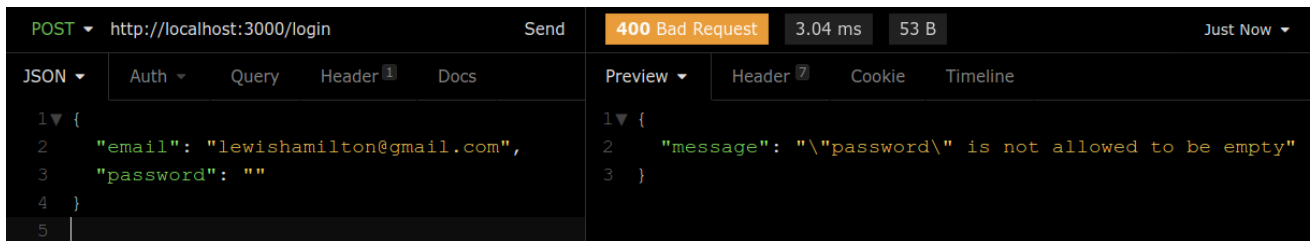
Se o login tiver o campo "email" em branco o resultado retornado deverá ser conforme exibido abaixo, com um status http 400 :



(As contrabarras \ estão escapando as aspas de dentro da string)

[Será validado que não é possível fazer login com o campo password em branco]

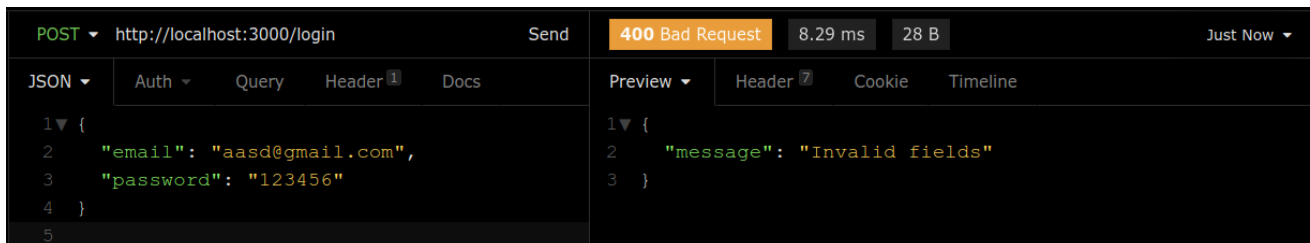
Se o login tiver o campo "password" em branco o resultado retornado deverá ser conforme exibido abaixo, com um status http 400 :



(As contrabarras \ estão escapando as aspas de dentro da string)

[Será validado que não é possível fazer login com um usuário que não existe]

Se o login for com usuário inexistente o resultado retornado deverá ser conforme exibido abaixo, com um status http 400 :



3 - Sua aplicação deve ter o endpoint GET /user

Os seguintes pontos serão avaliados:

- Deve listar todos os **Users** e retorná-los na seguinte estrutura:

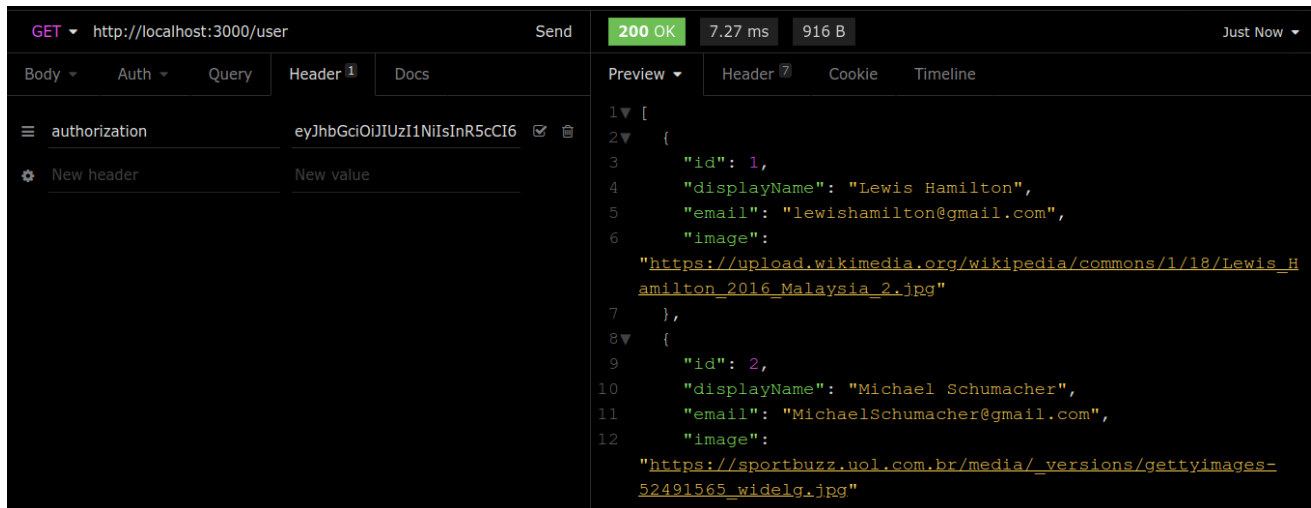
```
[
  {
    "id": "401465483996",
    "displayName": "Brett Wiltshire",
    "email": "brett@email.com",
    "image": "http://4.bp.blogspot.com/_YA50adQ-7vQ/S1gfR_6ufpI/AAAAAAAAA"
  }
]
```

- A requisição deve ter token de autenticação nos headers e, caso contrário, retorne um código de status 401 .

Além disso, as seguintes verificações serão feitas:

[Será validado que é possível listar todos os usuários]

Ao listar usuários com sucesso o resultado retornado deverá ser conforme exibido abaixo, com um status http 200 :



```
GET http://localhost:3000/user 200 OK 7.27 ms 916 B Just Now
```

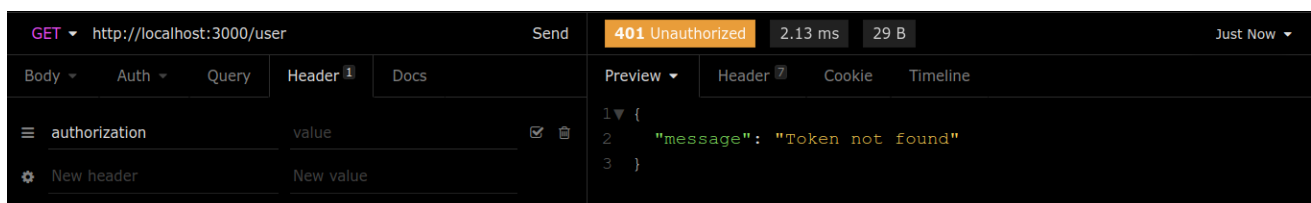
authorization eyJhbGciOiJIUzI1NiIsInR5cCI6I...

New header New value

```
1 [
2   {
3     "id": 1,
4     "displayName": "Lewis Hamilton",
5     "email": "lewishamilton@gmail.com",
6     "image": "https://upload.wikimedia.org/wikipedia/commons/1/18/Lewis_Hamilton_2016_Malaysia_2.jpg"
7   },
8   {
9     "id": 2,
10    "displayName": "Michael Schumacher",
11    "email": "MichaelSchumacher@gmail.com",
12    "image": "https://sportbuzz.uol.com.br/media/_versions/gettyimages-52491565_wide1g.jpg"
13  }
14 ]
```

[Será validado que não é possível listar usuários sem o token na requisição]

Se o token for inexistente o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



```
GET http://localhost:3000/user 401 Unauthorized 2.13 ms 29 B Just Now
```

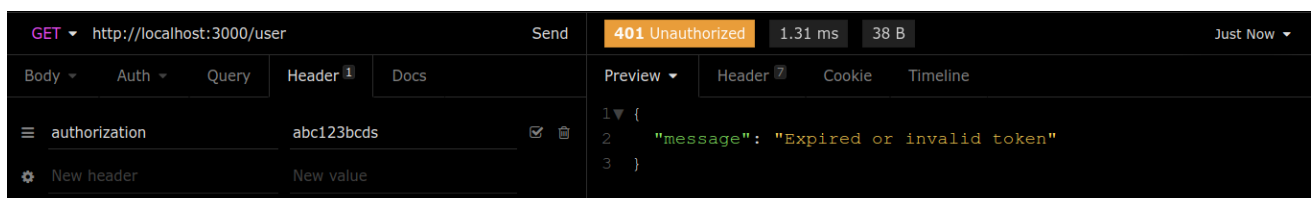
authorization value

New header New value

```
1 {
2   "message": "Token not found"
3 }
```

[Será validado que não é possível listar usuários com o token inválido]

Se o token for inválido o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



```
GET http://localhost:3000/user 401 Unauthorized 1.31 ms 38 B Just Now
```

authorization abc123bcds

New header New value

```
1 {
2   "message": "Expired or invalid token"
3 }
```

4 - Sua aplicação deve ter o endpoint GET /user/:id

Os seguintes pontos serão avaliados:

- Retorna os detalhes do usuário baseado no `id` da rota. Os dados devem ter o seguinte formato:

```
{
  "id": "401465483996",
  "displayName": "Brett Wiltshire",
  "email": "brett@email.com",
  "image": "http://4.bp.blogspot.com/_YA50adQ-7vQ/S1gfR_6ufpI/AAAAAAAAAAk/"
}
```

- A requisição deve ter token de autenticação nos headers e, caso contrário, retorne um código de status `401`.

Além disso, as seguintes verificações serão feitas:

[Será validado que é possível listar um usuario específico com sucesso]

Ao listar um usuário com sucesso o resultado retornado deverá ser conforme exibido abaixo, com um status http `200` :

GET `http://localhost:3000/user/1` Send **200 OK** 7.1 ms 170 B Just Now ▾

Body ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 7 Cookie Timeline

authorization eyJhbGciOiJIUzI1NiIsInR5cCI6I...

New header New value

```
1 {
2   "id": 1,
3   "displayName": "Lewis Hamilton",
4   "email": "lewishamilton@gmail.com",
5   "image":
6     "https://upload.wikimedia.org/wikipedia/commons/1/18/Lewis_Hamilton_2016_Malaysia_2.jpg"
7 }
```

[Será validado que não é possível listar um usuário inexistente]

Se o usuário for inexistente o resultado retornado deverá ser conforme exibido abaixo, com um status http `404` :

GET `http://localhost:3000/user/999` Send **404 Not Found** 7.62 ms 33 B Just Now ▾

Body ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 7 Cookie Timeline

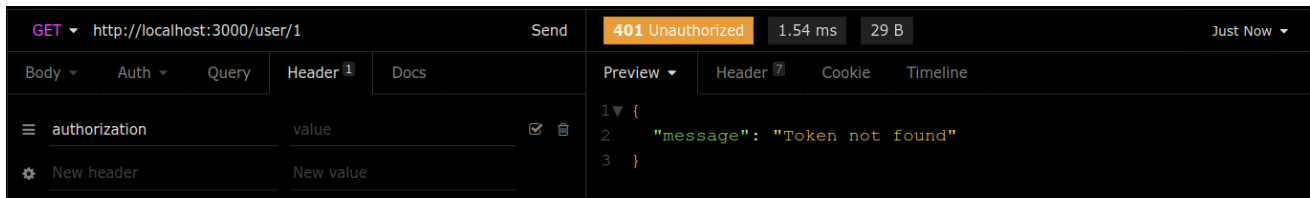
authorization eyJhbGciOiJIUzI1NiIsInR5cCI6I...

New header New value

```
1 {
2   "message": "User does not exist"
3 }
```

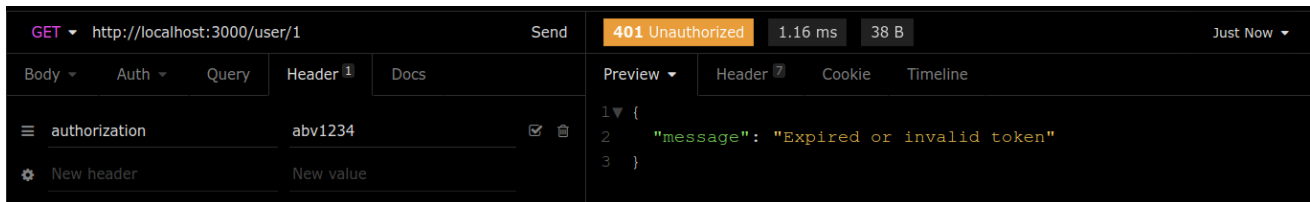
[Será validado que não é possível listar um determinado usuário sem o token na requisição]

Se o token for inexistente o resultado retornado deverá ser conforme exibido abaixo, com um status http `401` :



[Será validado que não é possível listar um determinado usuário com o token inválido]

Se o token for inválido o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



5 - Sua aplicação deve ter o endpoint POST /categories

Os seguintes pontos serão avaliados:

- Esse endpoint deve receber uma *Categoria* no corpo da requisição e criá-la no banco. O corpo da requisição deve ter a seguinte estrutura:

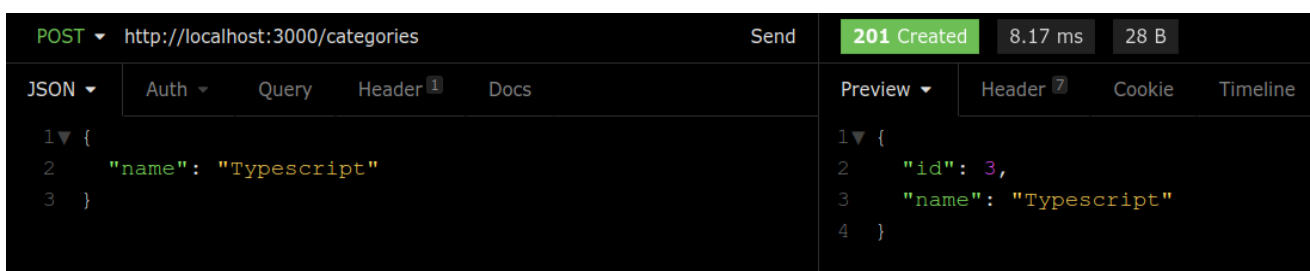
```
{
  "name": "Inovação"
}
```

- Caso a Categoria não contenha o `name` a API deve retornar um erro de status 400 .
- A requisição deve ter o token de autenticação nos headers e, caso contrário, retorne um código de status 401 .

Além disso, as seguintes verificações serão feitas:

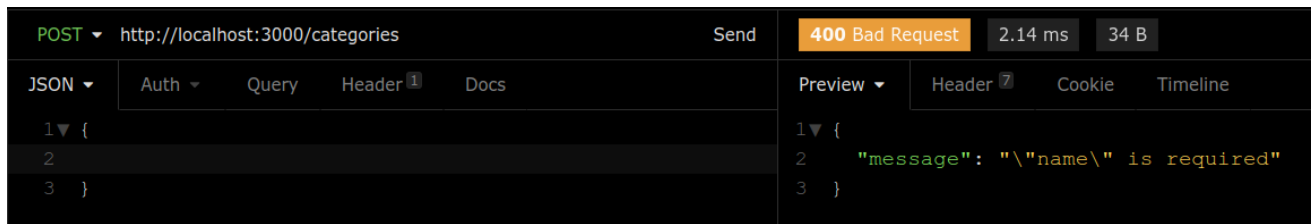
[Será validado que é possível cadastrar uma categoria com sucesso]

Se cadastrar uma categoria com sucesso o resultado retornado deverá ser conforme exibido abaixo, com um status http 201 :



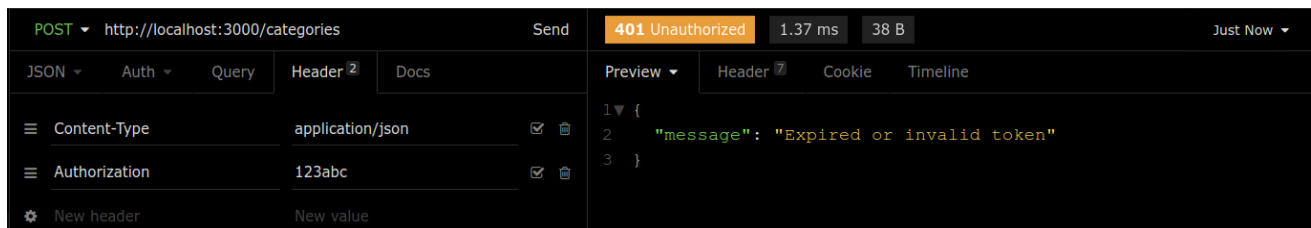
[Será validado que não é possível cadastrar uma categoria sem o campo name]

Se ao tentar cadastrar uma categoria sem o campo name o resultado retornado deverá ser conforme exibido abaixo, com um status http 400:



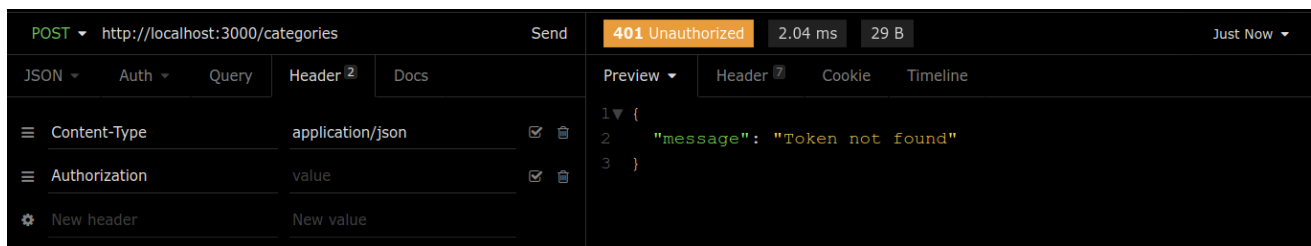
[Será validado que não é possível cadastrar uma determinada categoria com o token inválido]

Se o token for inválido o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



[Será validado que não é possível cadastrar uma determinada categoria sem o token na requisição]

Se o token for inexistente o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



6 - Sua aplicação deve ter o endpoint GET /categories

Os seguintes pontos serão avaliados:

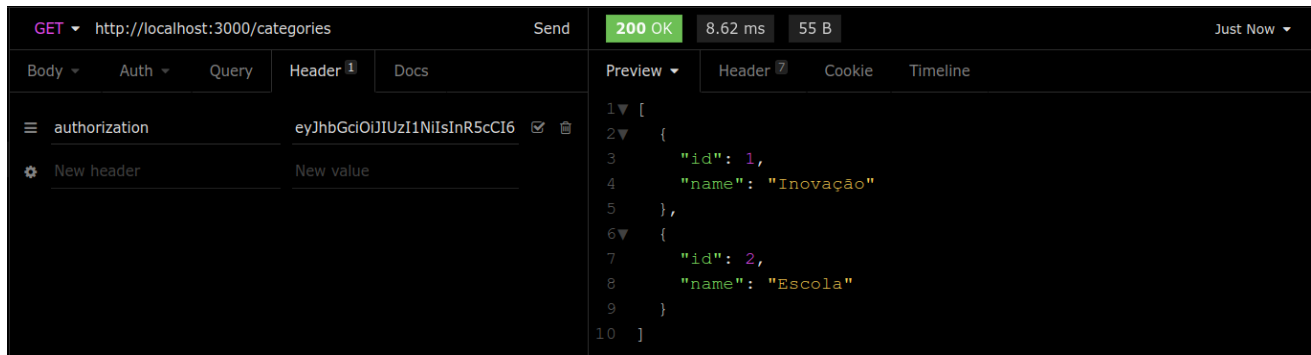
- Esse endpoint deve listar todas as Categorias e retorná-las na seguinte estrutura:

```
[
  {
    "id": 1,
    "name": "Escola"
  },
  {
    "id": 2,
```

```
"name": "Inovação"
}
```

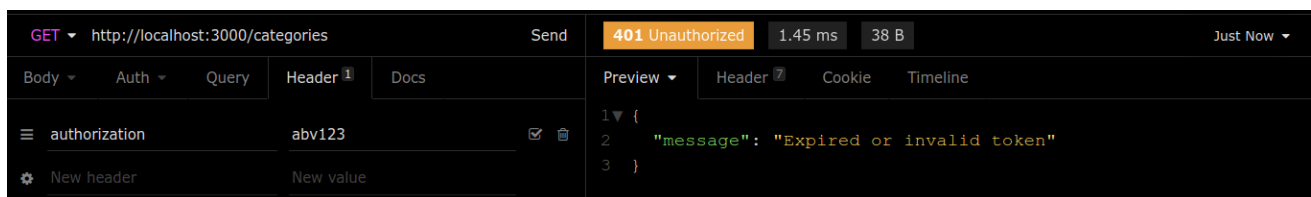
Além disso, as seguintes verificações serão feitas: [Será validado que é possível listar todas as categoria com sucesso]

Se buscar todas as categorias com sucesso o resultado retornado deverá ser conforme exibido abaixo, com um status http 200:



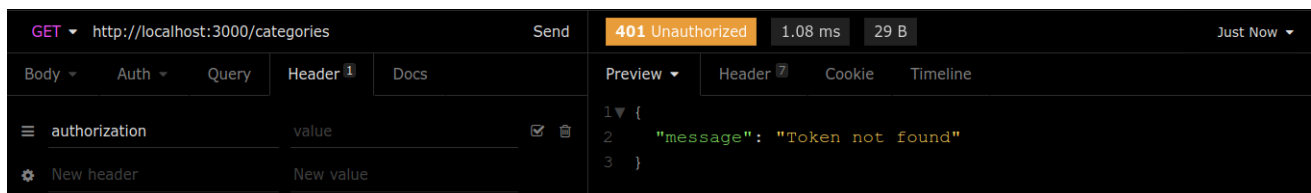
[Será validado que não é possível listar as categorias com o token inválido]

Se o token for inválido o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



[Será validado que não é possível cadastrar uma determinada categoria sem o token na requisição]

Se o token for inexistente o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



7 - Sua aplicação deve ter o endpoint POST /post

Os seguintes pontos serão avaliados:

- Esse endpoint deve receber um *BlogPost* no corpo da requisição e criá-lo no banco. O corpo da requisição deve ter a seguinte estrutura:

```
{
  "title": "Latest updates, August 1st",
  "content": "The whole text for the blog post goes here in this key",
  "categoryIds": [1, 2]
}
```

- Caso o post não contenha o `title`, `content` ou `categoryIds` a API deve retornar um erro de `status 400`.
- A requisição deve ter o token de autenticação nos headers e, caso contrário, retorne um código de `status 401`.

Além disso, as seguintes verificações serão feitas:

[Será validado que é possível cadastrar um blogpost com sucesso]

Se cadastrar um blogpost com sucesso o resultado retornado deverá ser conforme exibido abaixo, com um status `http 201`:

| Method | URL | Status | Time | Size |
|--------|----------------------------|-------------|---------|------|
| POST | http://localhost:3000/post | 201 Created | 84.6 ms | 62 B |

```
Request Body (JSON):
{
  "title": "I Love Trybe",
  "categoryIds": [1, 2],
  "content": "Love <3"
}

Response Body (JSON):
{
  "id": 4,
  "userId": 1,
  "title": "I Love Trybe",
  "content": "Love <3"
}
```

[Será validado que não é possível cadastrar um blogpost sem o campo `title`]

Se não conter o campo `title` o resultado retornado deverá ser conforme exibido abaixo, com um status `http 400`:

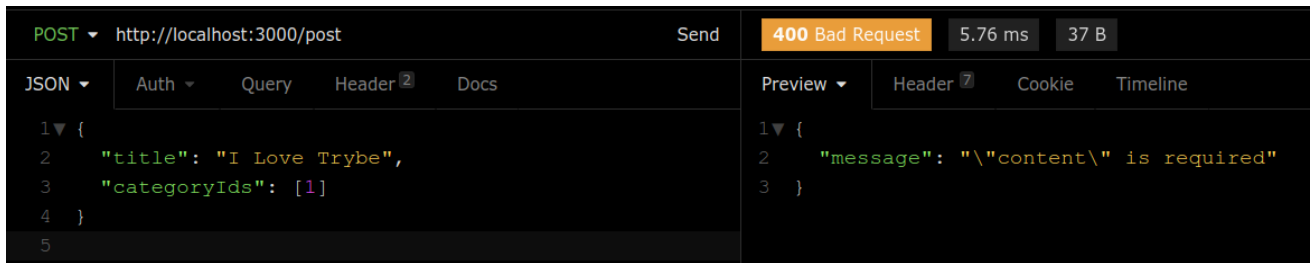
| Method | URL | Status | Time | Size |
|--------|----------------------------|-----------------|-------|------|
| POST | http://localhost:3000/post | 400 Bad Request | 10 ms | 35 B |

```
Request Body (JSON):
{
  "categoryIds": [1],
  "content": "Love <3"
}

Response Body (JSON):
{
  "message": "\"title\" is required"
}
```

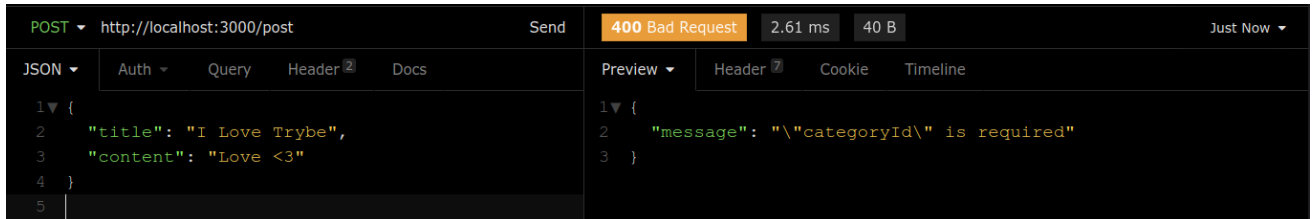
[Será validado que não é possível cadastrar um blogpost sem o campo `content`]

Se não conter o campo `content` o resultado retornado deverá ser conforme exibido abaixo, com um status `http 400`:



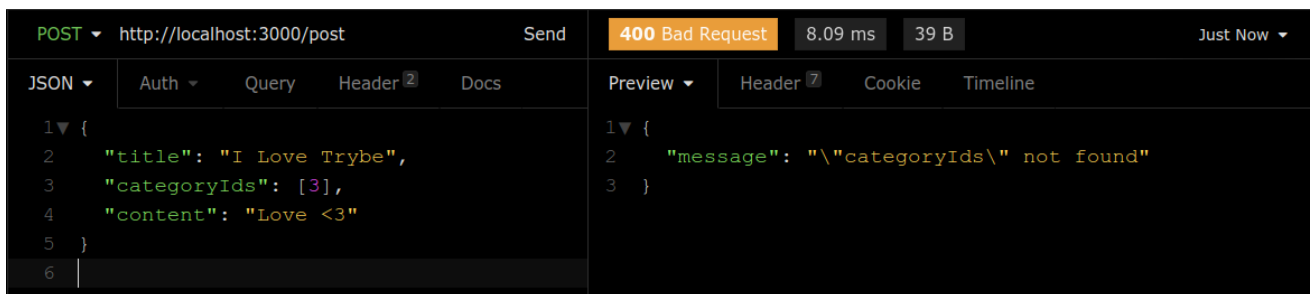
[Será validado que não é possível cadastrar um blogpost sem o campo `categoryIds`]

Se não conter o campo `categoryIds` o resultado retornado deverá ser conforme exibido abaixo, com um status http 400 :



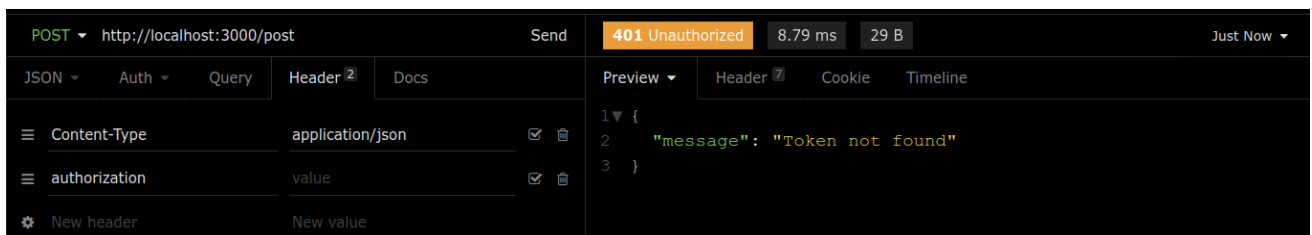
[Será validado que não é possível cadastrar um blogpost com uma `categoryIds` inexistente]

Se o campo `categoryIds` tiver uma categoria inexistente, o resultado retornado deverá ser conforme exibido abaixo, com um status http 400 :



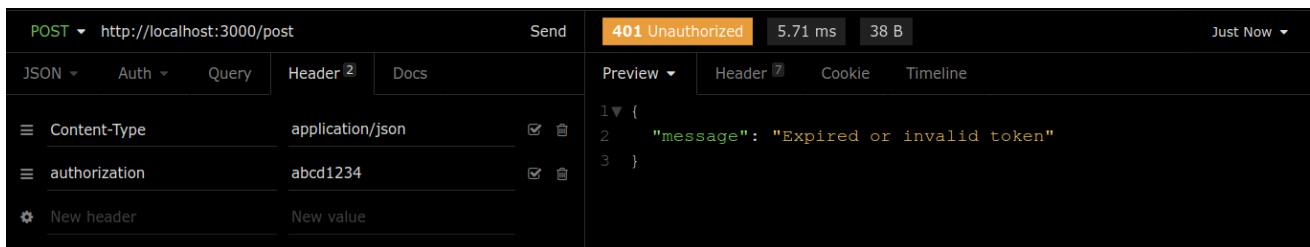
[Será validado que não é possível cadastrar um blogpost sem o token]

Se não conter o token o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



[Será validado que não é possível cadastrar um blogpost com o token inválido]

Se o token for inválido o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



8 - Sua aplicação deve ter o endpoint GET /post

Os seguintes pontos serão avaliados:

- Esse endpoint deve listar todos os *BlogPosts* e retorná-los na seguinte estrutura:

```
[
  {
    "id": 1,
    "title": "Post do Ano",
    "content": "Melhor post do ano",
    "userId": 1,
    "published": "2011-08-01T19:58:00.000Z",
    "updated": "2011-08-01T19:58:51.000Z",
    "user": {
      "id": 1,
      "displayName": "Lewis Hamilton",
      "email": "lewishamilton@gmail.com",
      "image": "https://upload.wikimedia.org/wikipedia/commons/1/18/Lewis_Hami"
    },
    "categories": [
      {
        "id": 1,
        "name": "Inovação"
      }
    ]
  }
]
```

Além disso, as seguintes verificações serão feitas:

[Será validado que é possível listar blogpost com sucesso]

Se listar os blogpost com sucesso o resultado retornado deverá ser conforme exibido abaixo, com um status http 200 :

GET http://localhost:3000/post Send 200 OK 10.4 ms 745 B Just Now

Body Auth Query Header Docs

authorization eyJhbGciOiJIUzI1NiIsInR5cCI6I...

New header New value

Preview Header Cookie Timeline

```

1 [
2   {
3     "id": 1,
4     "title": "Post do Ano",
5     "content": "Melhor post do ano",
6     "userId": 1,
7     "published": "2011-08-01T19:58:00.000Z",
8     "updated": "2011-08-01T19:58:51.000Z",
9     "user": {
10      "id": 1,
11      "displayName": "Lewis Hamilton",
12      "email": "lewishamilton@gmail.com",
13      "image": "https://upload.wikimedia.org/wikipedia/commons/1/18/Lewis_Hamilton_2016_Malaysia_2.jpg"
14    },
15    "categories": [
16      {
17        "id": 1,
18        "name": "Inovação"
19      }
20    ]
21  },

```

[Será validado que não é possível listar blogpost sem token]

Se não conter o token o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :

GET http://localhost:3000/post Send 401 Unauthorized 16 ms 29 B Just Now

Body Auth Query Header Docs

authorization value

New header New value

Preview Header Cookie Timeline

```

1 {
2   "message": "Token not found"
3 }

```

[Será validado que não é possível listar blogpost com token inválido]

Se o token for inválido o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :

GET http://localhost:3000/post Send 401 Unauthorized 1.08 ms 38 B Just Now

Body Auth Query Header Docs

authorization abc1234

New header New value

Preview Header Cookie Timeline

```

1 {
2   "message": "Expired or invalid token"
3 }

```

9 - Sua aplicação deve ter o endpoint GET post/:id

Os seguintes pontos serão avaliados:

- Retorna um **BlogPost** com o `id` especificado. O retorno deve ter os seguinte formato:

```

{
  "id": 1,

```



```

    "title": "Post do Ano",
    "content": "Melhor post do ano",
    "userId": 1,
    "published": "2011-08-01T19:58:00.000Z",
    "updated": "2011-08-01T19:58:51.000Z",
    "user": {
      "id": 1,
      "displayName": "Lewis Hamilton",
      "email": "lewishamilton@gmail.com",
      "image": "https://upload.wikimedia.org/wikipedia/commons/1/18/Lewis_Hamilt
    },
    "categories": [
      {
        "id": 1,
        "name": "Inovação"
      }
    ]
  }
}

```

Além disso, as seguintes verificações serão feitas:

[Será validado que é possível listar um blogpost com sucesso]

Se listar um blogpost com sucesso o resultado retornado deverá ser conforme exibido abaixo, com um status http 200 :

GET http://localhost:3000/post/1 Send 200 OK 10.4 ms 370 B Just Now ▾

JSON ▾ Auth ▾ Query Header 2 Docs

Content-Type application/json

authorization eyJhbGciOiJIUzI1NiIsInR5cCI6Ikpz...

New header New value

Preview ▾ Header 7 Cookie Timeline

```

1 {
2   "id": 1,
3   "title": "Post do Ano",
4   "content": "Melhor post do ano",
5   "userId": 1,
6   "published": "2011-08-01T19:58:00.000Z",
7   "updated": "2011-08-01T19:58:51.000Z",
8   "user": {
9     "id": 1,
10    "displayName": "Lewis Hamilton",
11    "email": "lewishamilton@gmail.com",
12    "image": "https://upload.wikimedia.org/wikipedia/commons/1/18/Lewis_Ha
13  },
14  "categories": [
15    {
16      "id": 1,
17      "name": "Inovação"
18    }
19  ]
20 }

```

[Será validado que não é possível listar um blogpost sem token]

Se não conter o token o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :

The screenshot shows the Chrome DevTools Network tab. The top bar indicates a GET request to http://localhost:3000/post/1, which resulted in a 401 Unauthorized status, a response time of 7.7 ms, and a size of 29 B. The response is categorized as 'Just Now'. Below the top bar, the 'Header' tab is selected, showing a table with headers: 'Content-Type' with value 'application/json', and 'authorization' with value 'value'. There is a 'New header' row at the bottom. The 'Preview' tab is also visible, showing a JSON response: { "message": "Token not found" }.

[Será validado que não é possível listar um blogpost com token inválido]

Se o token for inválido o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :

The screenshot shows the Chrome DevTools Network tab. The top bar indicates a GET request to http://localhost:3000/post/1, which resulted in a 401 Unauthorized status, a response time of 2.18 ms, and a size of 38 B. The response was received 'Just Now'. The 'Headers' tab is selected, showing the 'Content-Type' header as 'application/json' and the 'authorization' header as 'abc1234'. The 'Preview' tab shows the response body as a JSON object: { "message": "Expired or invalid token" }.

[Será validado que não é possível listar um blogpost inexistente]

Se o id do post for inválido o resultado retornado deverá ser conforme exibido abaixo, com um status http 404 :

A screenshot of the Chrome DevTools Network tab. The top bar shows a GET request to http://localhost:3000/post/999 with a status of 404 Not Found, a response time of 9.57 ms, and a size of 33 B. Below the top bar, there are tabs for JSON, Auth, Query, Header, Docs, Preview, Cookie, and Timeline. The Header tab is selected, showing two headers: Content-Type: application/json and authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6I... Both headers have checkmark and trash icons next to them. At the bottom left, there is a 'New header' button. On the right side, the Preview pane shows the response body as a JSON object: {"message": "Post does not exist"}.

10 - Sua aplicação deve ter o endpoint PUT /post/:id

Os seguintes pontos serão avaliados:

- O endpoint deve receber um **BlogPost** que irá sobrescrever o original com o `id` especificado na URL. Só deve ser permitido para o usuário que criou o **BlogPost**.
- A(s) categoria(s) do post **não** podem ser editadas, somente o `title` e `content`.
- O corpo da requisição deve ter a seguinte estrutura:

```
{
  "title": "Latest updates, August 1st",
  "content": "The whole text for the blog post goes here in this key"
}
```

- Caso uma pessoa diferente de quem criou faça a requisição, deve retornar um código status 401 .
- Caso uma requisição sem token seja recebida, deve-se retornar um código de status 401 .
- Caso o post não contenha o title e/ou o content a API deve retornar um erro de status 400 .

Além disso, as seguintes verificações serão feitas:

[Será validado que é possível editar um blogpost com sucesso]

Se editar um blogpost com sucesso o resultado retornado deverá ser conforme exibido abaixo, com um status http 200 :

PUT http://localhost:3000/post/2 Send 200 OK 11.2 ms 96 B Just Now

JSON Auth Query Header Docs Preview Header Cookie Timeline

```
1 {
2   "title": "Trybe Love I",
3   "content": "<3 Trybe"
4 }
5
```

```
1 {
2   "title": "Trybe Love I",
3   "content": "<3 Trybe",
4   "userId": 1,
5   "categories": [
6     {
7       "id": 2,
8       "name": "Escola"
9     }
10  ]
11 }
```

[Será validado que não é possível editar as categorias de um blogpost]

Só será possível editar o título ou o conteúdo de um post.

PUT http://localhost:3000/post/2 Send 400 Bad Request 64.7 ms 41 B 5 Minutes Ago

JSON Auth Query Header Docs Preview Header Cookie Timeline

```
1 {
2   "title": "Trybe Love I",
3   "content": "Edited",
4   "categoryIds": [1,2]
5 }
6
```

```
1 {
2   "message": "Categories cannot be edited"
3 }
```

[Será validado que não é possível editar um blogpost com outro usuário]

Somente o usuário que criou o post poderá editá-lo.

PUT http://localhost:3000/post/2 Send 401 Unauthorized 8.73 ms 31 B Just Now

JSON Auth Query Header Docs Preview Header Cookie Timeline

```
1 {
2   "title": "Trybe Love I",
3   "content": "<3 Trybe"
4 }
5
```

```
1 {
2   "message": "Unauthorized user"
3 }
```

[Será validado que não é possível editar um blogpost sem token]

Se não conter o token o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :

| | | | | | | | | |
|----------------------------------|------------------|-------|-------------------------------------|------|--------------------------------|----------|----------|----------|
| PUT http://localhost:3000/post/2 | | Send | 401 Unauthorized | | 1.65 ms | 29 B | Just Now | |
| JSON | Auth | Query | Header 2 | Docs | Preview | Header 7 | Cookie | Timeline |
| Content-Type | application/json | | <input checked="" type="checkbox"/> | | 1 { | | | |
| authorization | value | | <input checked="" type="checkbox"/> | | 2 "message": "Token not found" | | | |
| | | | | | 3 } | | | |
| New header | New value | | | | | | | |

[Será validado que não possível editar um blogpost com token inválido]

Se o token for inválido o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :

PUT

http://localhost:3000/post/2

Send

401 Unauthorized

1.84 ms

38 B

Just Now

JSON

Auth

Query

Header 2

Docs

Preview

Header 7

Cookie

Timeline

≡

Content-Type

application/json

✓

🗑

≡

authorization

abv1234

✓

🗑

⚙

New header

New value

1 {

2 "message": "Expired or invalid token"

3 }

[Será validado que não possível editar um blogpost sem o campo title]

Se não conter o campo title o resultado retornado deverá ser conforme exibido abaixo, com um status http 400 :

| | | | | | | |
|---|------------------------|---|------------------------------|--|--|----------|
| PUT http://localhost:3000/post/2 | | Send | 400 Bad Request 4.36 ms 35 B | | | Just Now |
| JSON | Auth Query Header Docs | Preview Header Cookie Timeline | | | | |
| <pre>1 { 2 "content": "<3 Trybe" 3 } 4</pre> | | <pre>1 { 2 "message": "\"title\" is required" 3 }</pre> | | | | |

[Será validado que não possível editar um blogpost sem o campo content]

Se não conter o campo content o resultado retornado deverá ser conforme exibido abaixo, com um status http 400 :

| | | | | | | | | |
|--|------|-------|-----------------|---|---------|--------|----------|----------|
| PUT http://localhost:3000/post/2 | | Send | 400 Bad Request | | 2.57 ms | 37 B | Just Now | |
| JSON | Auth | Query | Header | Docs | Preview | Header | Cookie | Timeline |
| <pre>1 { 2 "title": "Trybe Love I" 3 } 4</pre> | | | | <pre>1 { 2 "message": "\"content\" is required" 3 }</pre> | | | | |

Requisitos Bônus

11 - Sua aplicação deve ter o endpoint DELETE post/:id

Os seguintes pontos serão avaliados:

- Além disso, as seguintes verificações serão feitas:**

Se deletar blogpost com sucesso o resultado retornado deverá ser conforme exibido abaixo, com um status http 204 :

[Será validado que não é possível deletar um blogpost com outro usuário]

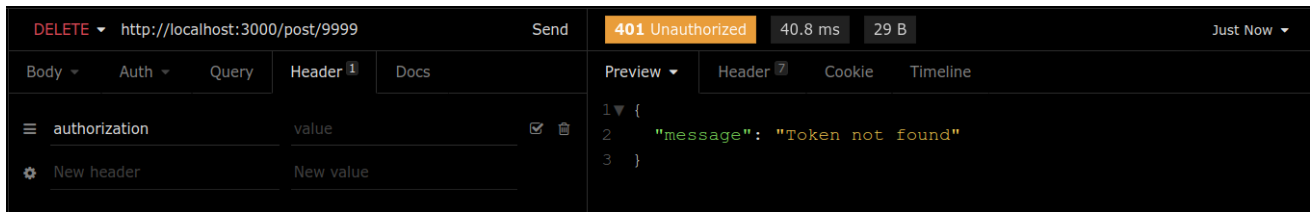
Se não for o dono do blogpost o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :

[Será validado que não é possível deletar um blogpost inexistente]

Se o blogpost nao existir o resultado retornado deverá ser conforme exibido abaixo, com um status http 404 :

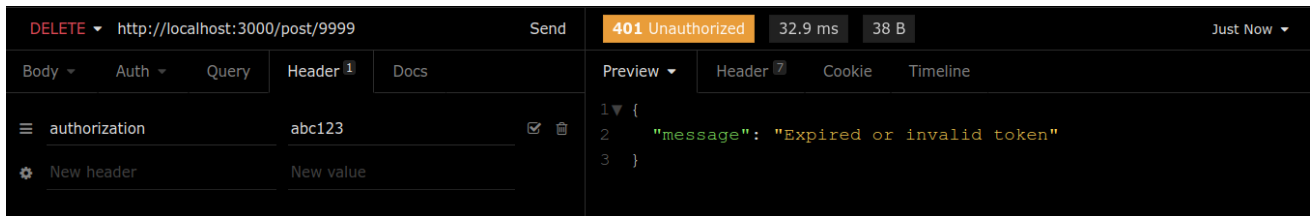
[Será validado que não é possível deletar um blogpost sem o token]

Se não contém o token o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



[Será validado que não é possível deletar um blogpost com o token inválido]

Se o token for inválido o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



12 - Sua aplicação deve ter o endpoint DELETE /user/me

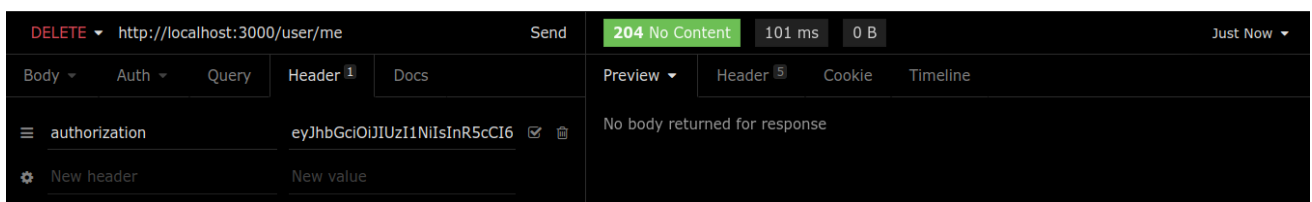
Os seguintes pontos serão avaliados:

- Utilizando o token de autenticação nos headers, o usuário correspondente deve ser apagado.

Além disso, as seguintes verificações serão feitas:

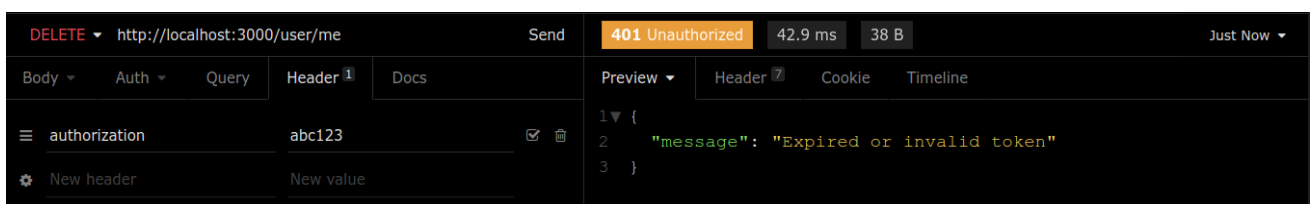
[Será validado que é possível excluir meu usuário com sucesso]

Ao deletar um usuário com sucesso o resultado retornado deverá ser conforme exibido abaixo, com um status http 204 :



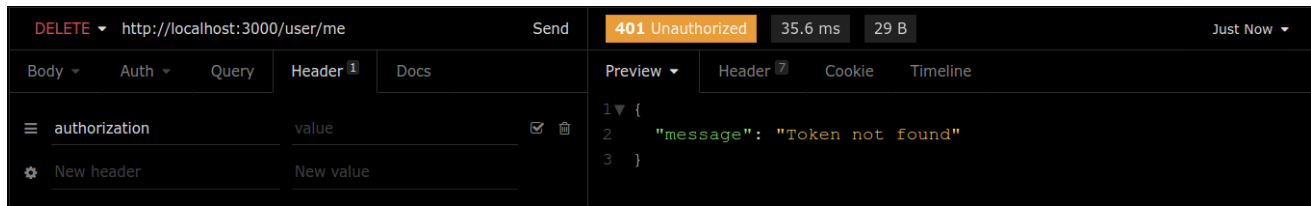
[Será validado que não é possível excluir meu usuário com token inválido]

Se o token for inválido o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



[Será validado que não é possível excluir meu usuário sem o token]

Se não conter o token o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



13 - Sua aplicação deve ter o endpoint GET post/search?q=:searchTerm

Os seguintes pontos serão avaliados:

- Retorna uma array de **BlogPosts** que contenham em seu título, ou conteúdo, o termo pesquisado no queryParam da URL. O retorno deve ter o seguinte formato:

```
[
  {
    "id": 2,
    "title": "Vamos que vamos",
    "content": "Foguete não tem ré",
    "userId": 1,
    "published": "2011-08-01T19:58:00.000Z",
    "updated": "2011-08-01T19:58:51.000Z",
    "user": {
      "id": 1,
      "displayName": "Lewis Hamilton",
      "email": "lewishamilton@gmail.com",
      "image": "https://upload.wikimedia.org/wikipedia/commons/1/18/Lewis_Hami"
    },
    "categories": [
      {
        "id": 2,
        "name": "Escola"
      }
    ]
  }
]
```

- Caso nenhum **BlogPost** satisfaça a busca, retorne um array vazio.

Além disso, as seguintes verificações serão feitas:

[Será validado que é possível buscar um blogpost pelo title]

Se a busca for pelo title o resultado retornado deverá ser conforme exibido abaixo, com um status http 200 :

GET http://localhost:3000/post/search?q=Vamos que vamos Send 200 OK 80.8 ms 374 B Just Now

Body Auth Query Header Docs

authorization eyJhbGciOiJIUzI1NiIsInR5cCI6I...

New header New value

Preview Header Cookie Timeline

```

1 [
2   {
3     "id": 2,
4     "title": "Vamos que vamos",
5     "content": "Foguete não tem ré",
6     "userId": 1,
7     "published": "2011-08-01T19:58:00.000Z",
8     "updated": "2011-08-01T19:58:51.000Z",
9     "user": {
10      "id": 1,
11      "displayName": "Lewis Hamilton",
12      "email": "lewishamilton@gmail.com",
13      "image": "https://upload.wikimedia.org/wikipedia/commons/1/18/Lewis_Hamilton_2016_Malaysia_2.jpg"
14    },
15    "categories": [
16      {
17        "id": 2,
18        "name": "Escola"
19      }
20    ]
21  }
22 ]

```

[Será validado que é possível buscar um blogpost pelo content]

Se a buscar for pelo content o resultado retornado deverá ser conforme exibido abaixo, com um status http 200 :

GET http://localhost:3000/post/search?q=foguete não tem ré Send 200 OK 8.39 ms 374 B Just Now

Body Auth Query Header Docs

authorization eyJhbGciOiJIUzI1NiIsInR5cCI6I...

New header New value

Preview Header Cookie Timeline

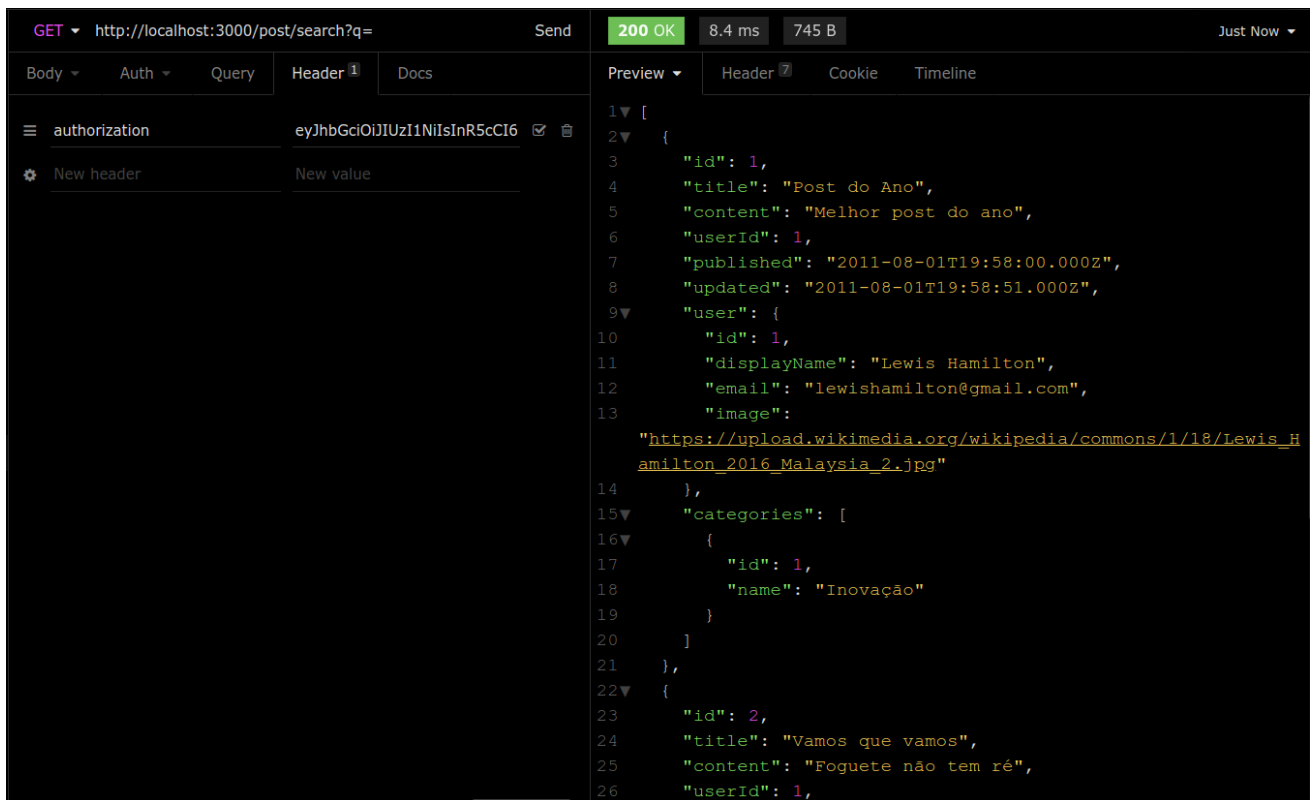
```

1 [
2   {
3     "id": 2,
4     "title": "Vamos que vamos",
5     "content": "Foguete não tem ré",
6     "userId": 1,
7     "published": "2011-08-01T19:58:00.000Z",
8     "updated": "2011-08-01T19:58:51.000Z",
9     "user": {
10      "id": 1,
11      "displayName": "Lewis Hamilton",
12      "email": "lewishamilton@gmail.com",
13      "image": "https://upload.wikimedia.org/wikipedia/commons/1/18/Lewis_Hamilton_2016_Malaysia_2.jpg"
14    },
15    "categories": [
16      {
17        "id": 2,
18        "name": "Escola"
19      }
20    ]
21  }
22 ]

```

[Será validado que é possível buscar todos os blogpost quando passa a busca vazia']

Se a buscar for vazia o resultado retornado deverá ser conforme exibido abaixo, com um status http 200 :



GET http://localhost:3000/post/search?q= Send 200 OK 8.4 ms 745 B Just Now

Body Auth Query Header Docs

authorization eyJhbGciOiJIUzI1NiIsInR5cCI6I...

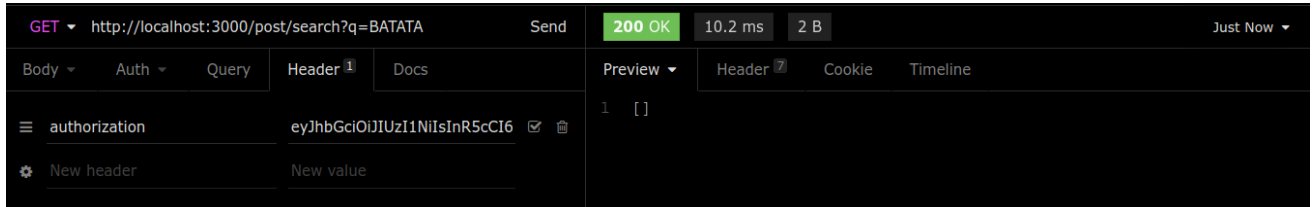
New header New value

Preview Header Cookie Timeline

```
1 [
2   {
3     "id": 1,
4     "title": "Post do Ano",
5     "content": "Melhor post do ano",
6     "userId": 1,
7     "published": "2011-08-01T19:58:00.000Z",
8     "updated": "2011-08-01T19:58:51.000Z",
9     "user": {
10      "id": 1,
11      "displayName": "Lewis Hamilton",
12      "email": "lewishamilton@gmail.com",
13      "image": "https://upload.wikimedia.org/wikipedia/commons/1/18/Lewis_Hamilton_2016_Malaysia_2.jpg"
14    },
15    "categories": [
16      {
17        "id": 1,
18        "name": "Inovação"
19      }
20    ]
21  },
22  {
23    "id": 2,
24    "title": "Vamos que vamos",
25    "content": "Foguete não tem ré",
26    "userId": 1,
```

[Será validado que é possível buscar um blogpost inexistente e retornar array vazio]

Se a buscar um post inexistente o resultado retornado deverá ser conforme exibido abaixo, com um status http 200 :



GET http://localhost:3000/post/search?q=BATATA Send 200 OK 10.2 ms 2 B Just Now

Body Auth Query Header Docs

authorization eyJhbGciOiJIUzI1NiIsInR5cCI6I...

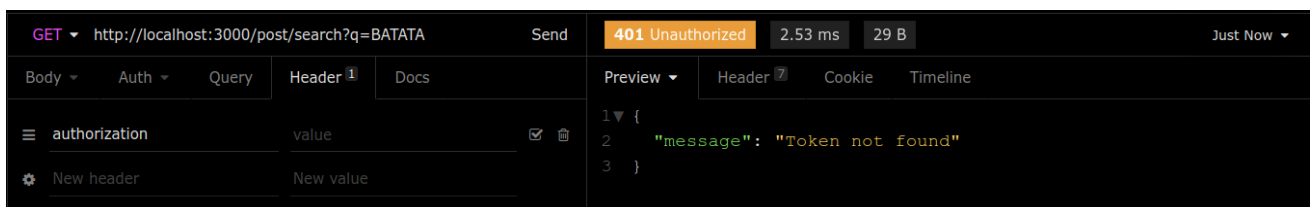
New header New value

Preview Header Cookie Timeline

```
1 []
```

[Será validado que não é possível buscar um blogpost sem o token]

Se não contém o token o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



GET http://localhost:3000/post/search?q=BATATA Send 401 Unauthorized 2.53 ms 29 B Just Now

Body Auth Query Header Docs

authorization value

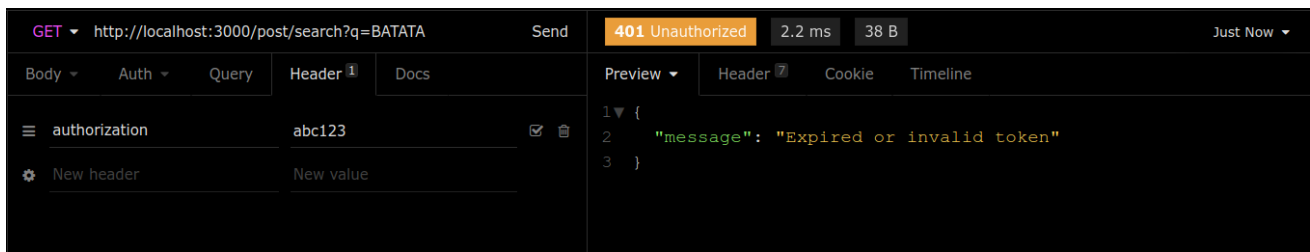
New header New value

Preview Header Cookie Timeline

```
1 {
2   "message": "Token not found"
3 }
```

[Será validado que não é possível buscar um blogpost com o token inválido]

Se o token for inválido o resultado retornado deverá ser conforme exibido abaixo, com um status http 401 :



Depois de terminar o desenvolvimento

Para **"entregar"** seu projeto, siga os passos a seguir:

- Vá até a página **DO SEU Pull Request**, adicione a label de *"code-review"* e marque seus colegas
 - No menu à direita, clique no *link "Labels"* e escolha a *label code-review*
 - No menu à direita, clique no *link "Assignees"* e escolha **o seu usuário**
 - No menu à direita, clique no *link "Reviewers"* e digite `students`, selecione o `time tryber/students-sd-010-a`

Se ainda houver alguma dúvida sobre como entregar seu projeto, [aqui tem um video explicativo](#).

⚠ Lembre-se que garantir que todas as *issues* comentadas pelo **Lint** estão resolvidas! ⚠

Revisando um pull request

À medida que você e as outras pessoas que estudam na Trybe forem entregando os projetos, vocês receberão um alerta via Slack para também fazer a revisão dos Pull Requests dos seus colegas. Fiquem atentos às mensagens do "Pull Reminders" no Slack!

Use o material que você já viu sobre [Code Review](#) para te ajudar a revisar os projetos que chegaram para você.

Avisos Finais

Ao finalizar e submeter o projeto, não se esqueça de avaliar sua experiência preenchendo o formulário. Leva menos de 3 minutos!

Link: [FORMULÁRIO DE AVALIAÇÃO DE PROJETO](#)

O avaliador automático não necessariamente avalia seu projeto na ordem em que os requisitos aparecem no readme. Isso acontece para deixar o processo de avaliação mais rápido. Então, não se assuste se isso acontecer, ok?

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Contributors 2



jeanpsv Jean Paulo Silva Vasconcelos



GabrielCoruja Gabriel Dalseco

Languages

● **JavaScript** 100.0%