



Ignite



Node.js



Desafio 03 - Corrigindo o código

Sobre o desafio

Template da aplicação

Rotas da aplicação

GET /repositories

POST /repositories

PUT /repositories/:id

DELETE /repositories/:id

POST /repositories/:id/like

Especificação dos testes

Testes de repositórios

Testes de likes

Entrega



Sobre o desafio

Nesse desafio, temos uma aplicação Node.js que está em processo de desenvolvimento mas que já possui os testes necessários para fazer toda a validação dos requisitos (você não deve mexer nos testes).

Após algumas alterações no código da aplicação, parte dos testes deixaram de passar e agora só você pode resolver esse problema. Bora lá? 🚀

Essa aplicação realiza o CRUD (**C**reate, **R**ead, **U**ppdate, **D**elte) de repositórios de projetos. Além disso, é possível dar likes em repositórios cadastrados, aumentando a quantidade de likes em 1 a cada vez que a rota é chamada.

A estrutura de um repositório ao ser criado é a seguinte:

```
{ id: uuid(), title, url, techs, likes: 0 }
```

Descrição de cada propriedade:

- **id** deve ser um uuid válido;
- **title** é o título do repositório (por exemplo "unform");
- **url** é a URL que aponta para o repositório (por exemplo ["https://github.com/unform/unform"](https://github.com/unform/unform));
- **techs** é um array onde cada elemento deve ser uma string com o nome de uma tecnologia relacionada ao repositório (por exemplo: ["react", "react-native", "form"]);
- **likes** é a quantidade de likes que o repositório recebeu (e que vai ser incrementada de 1 em 1 a cada chamada na rota de likes).

Note que a quantidade de likes deve sempre ser zero no momento de criação.


Template da aplicação

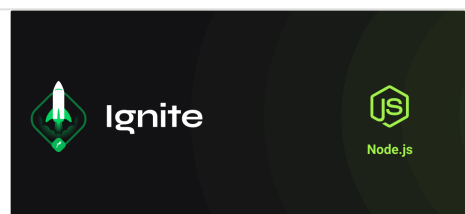
Para realizar esse desafio, criamos para você esse modelo que você deve utilizar como um template do GitHub.

O template está disponível na seguinte URL:

rocketseat-education/ignite-template-corrigindo-o-c...

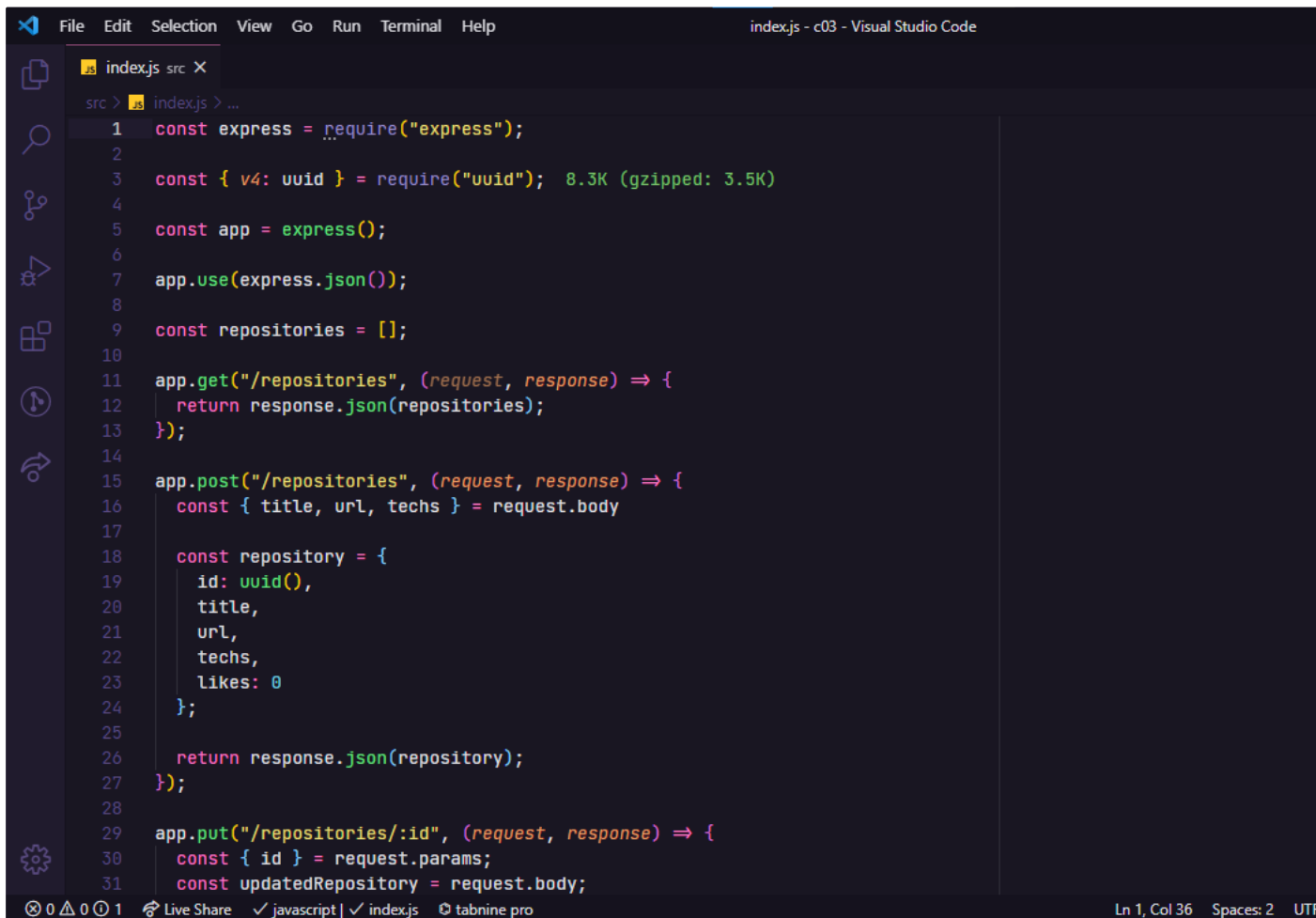
Ignite] Desafio 03 - Trilha Node.js. Contribute to rocketseat-education/ignite-template-corrigindo-o-codigo development by

 <https://github.com/rocketseat-education/ignite-template-...>



Dica: Caso não saiba utilizar repositórios do GitHub como template, temos um guia em [nosso FAQ](#).

Agora navegue até a pasta criada, abra no Visual Studio Code e por último abra o arquivo `index.js`. Lembre-se de executar o comando `yarn` no seu terminal para instalar todas as dependências e você terá o seguinte código:

A screenshot of the Visual Studio Code editor interface. The top bar shows the menu (File, Edit, Selection, View, Go, Run, Terminal, Help) and the file name 'index.js - c03 - Visual Studio Code'. The left sidebar shows the Explorer view with a file named 'index.js' selected. The main editor area displays the following JavaScript code:

```
1 const express = require("express");
2
3 const { v4: uuid } = require("uuid"); 8.3K (gzipped: 3.5K)
4
5 const app = express();
6
7 app.use(express.json());
8
9 const repositories = [];
10
11 app.get("/repositories", (request, response) => {
12   return response.json(repositories);
13 });
14
15 app.post("/repositories", (request, response) => {
16   const { title, url, techs } = request.body
17
18   const repository = {
19     id: uuid(),
20     title,
21     url,
22     techs,
23     likes: 0
24   };
25
26   return response.json(repository);
27 });
28
29 app.put("/repositories/:id", (request, response) => {
30   const { id } = request.params;
31   const updatedRepository = request.body;
```

The status bar at the bottom shows 'Ln 1, Col 36', 'Spaces: 2', and 'UTF-8'.

Rotas da aplicação

Com o template já clonado e o arquivo `index.js` aberto, você deve completar onde não possui código com o código para atingir os objetivos de cada teste.

GET `/repositories`

A rota deve retornar uma lista contendo todos os repositórios cadastrados.

POST `/repositories`

A rota deve receber `title`, `url` e `techs` pelo corpo da requisição e retornar um objeto com as informações do repositório criado e um status `201`.

PUT `/repositories/:id`

A rota deve receber `title`, `url` e `techs` pelo corpo da requisição e o `id` do repositório que deve ser atualizado pelo parâmetro da rota. Deve alterar apenas as informações recebidas pelo corpo da requisição e retornar esse repositório atualizado.

DELETE `/repositories/:id`

A rota deve receber, pelo parâmetro da rota, o `id` do repositório que deve ser excluído e retornar um status `204` após a exclusão.

POST `/repositories/:id/like`

A rota deve receber, pelo parâmetro da rota, o `id` do repositório que deve receber o like e retornar o repositório com a quantidade de likes atualizada.

Especificação dos testes

Em cada teste, tem uma breve descrição no que sua aplicação deve cumprir para que o teste passe.



Note que partes da aplicação já estão prontas e você precisará alterar apenas o que está errado (ou implementar algo que esteja faltando).

Se você achou algum trecho de código confuso ou pensou em uma melhor solução, sinta-se livre para também refatorar.



Caso você tenha dúvidas quanto ao que são os testes, e como interpretá-los, dê uma olhada em [nosso FAQ](#)

Para esse desafio, temos os seguintes testes:

Testes de repositórios

- Should be able to create a new repository

Para que esse teste passe, você deve permitir que um novo repositório seja cadastrado pela rota `POST /repositories`. Caso precise confirmar o formato do objeto, você pode olhar [aqui](#).

Também é necessário que você retorne a resposta com o código `201`.

- Should be able to list the projects

Para que esse teste passe, é necessário que você conclua o teste anterior. Se tudo ocorreu bem, os repositórios cadastrados deverão aparecerem na listagem da rota GET `/repositories` e esse teste irá passar.

- **Should be able to update repository**

Para que esse teste passe, você deve permitir que um repositório seja atualizado a partir de seu `id` pela rota PUT `/repositories/:id` usando as informações recebidas pelo corpo da requisição. Lembre-se de manter as informações que não foram passadas pelo corpo, por exemplo:

Se o usuário quiser trocar apenas o `title`, mantenha `url` e `techs` que já estavam no repositório.

- **Should not be able to update a non existing repository**

Para que esse teste passe, você deve verificar se o repositório existe antes de atualizar as informações na rota PUT `/repositories/:id`. Caso não exista, retorne um status `404` (que é o status para **Not Found**) com uma mensagem de erro no formato `{ error: "Mensagem do erro" }`.

- **Should not be able to update repository likes manually**

Para que esse teste passe, você deve impedir que a quantidade de likes de um repositório seja alterada manualmente através da rota PUT `/repositories/:id`. Por exemplo:

Errado:

```
// Repositório recém criado: { id: "c160a99b-9d3b-4669-8a35-8dce1e8196ec", title: "Umbriel", techs: ["React", "ReactNative", "TypeScript", "ContextApi"], url: "https://github.com/Rocketseat/umbriel", likes: 0 } // Requisição para alterar informações: // Rota: "/repositories/c160a99b-9d3b-4669-8a35-8dce1e8196ec" // Método: PUT // Corpo: { title: "Novo título", likes: 10 } // Retorno: { id: "c160a99b-9d3b-4669-8a35-8dce1e8196ec", title: "Novo título", techs: ["React", "ReactNative", "TypeScript", "ContextApi"], url: "https://github.com/Rocketseat/umbriel", likes: 10 }
```

Certo:

```
// Repositório recém criado: { id: "c160a99b-9d3b-4669-8a35-8dce1e8196ec", title: "Umbriel", techs: ["React", "ReactNative", "TypeScript", "ContextApi"], url: "https://github.com/Rocketseat/umbriel", likes: 0 } // Requisição para alterar informações: // Rota: "/repositories/c160a99b-9d3b-4669-8a35-8dce1e8196ec" // Método: PUT // Corpo: { title: "Novo título", likes: 10 } // Retorno: { id: "c160a99b-9d3b-4669-8a35-8dce1e8196ec", title: "Novo título", techs: ["React", "ReactNative", "TypeScript", "ContextApi"], url: "https://github.com/Rocketseat/umbriel", likes: 0 } // A quantidade de likes não mudou }
```

- Should be able to delete the repository

Para que esse teste passe, você deve permitir que um repositório seja excluído através do `id` passado pela rota **DELETE** `/repositories/:id`.

- Should not be able to delete a non existing repository

Para que esse teste passe, você deve validar se o repositório existe antes de excluí-lo. Caso o repositório não exista, retorne um status `404` com uma mensagem de erro no formato `{ error: "Mensagem do erro" }`.

Testes de likes

- Should be able to give a like to the repository

Para que esse teste passe, deve ser possível incrementar a quantidade de likes em `1` a cada chamada na rota **POST** `/repositories/:id/like`. Use o `id` passado por parâmetro na rota para realizar essa ação.

- Should not be able to give a like to a non existing repository

Para que esse teste passe, você deve validar que um repositório existe antes de incrementar a quantidade de likes. Caso não exista, retorne um status `404` com uma mensagem de erro no formato `{ error: "Mensagem do erro" }`.



Entrega

Esse desafio deve ser entregue a partir da plataforma da Rocketseat. Envie o link do repositório que você fez suas alterações. Após concluir o desafio, além de ter mandado o código para o GitHub, fazer um post no LinkedIn é uma boa forma de demonstrar seus conhecimentos e esforços para evoluir na sua carreira para oportunidades futuras.

Feito com  por Rocketseat  Participe da nossa comunidade aberta!