




# Ignite



Node.js



## Desafio 01 - Conceitos do Node.js

 [Sobre o desafio](#)

[Template da aplicação](#)

[Rotas da aplicação](#)

[POST /users](#)

[GET /todos](#)

[POST /todos](#)

[PUT /todos/:id](#)

[PATCH /todos/:id/done](#)

[DELETE /todos/:id](#)

[Especificação dos testes](#)

[Testes de usuários](#)

[Testes de todos](#)

 [Entrega](#)

[Solução do desafio](#)



### Sobre o desafio

Nesse desafio, você deverá criar uma aplicação para treinar o que aprendeu até agora no Node.js!

Essa será uma aplicação para gerenciar tarefas (em inglês *todos*). Será permitida a criação de um usuário com `name` e `username`, bem como fazer o CRUD de *todos*:

- Criar um novo *todo*;
- Listar todos os *todos*;
- Alterar o `title` e `deadline` de um *todo* existente;
- Marcar um *todo* como feito;
- Excluir um *todo*;

Tudo isso para cada usuário em específico (o `username` será passado pelo header). A seguir veremos com mais detalhes o que e como precisa ser feito 🚀


## 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-conceitos-do-nodejs

Ignite] Desafio 01 - Trilha Node.js. Contribute to rocketseat-education/ignite-template-conceitos-do-nodejs development by creating an account on GitHub.

 <https://github.com/rocketseat-education/ignite-template-conceitos-do-no...>



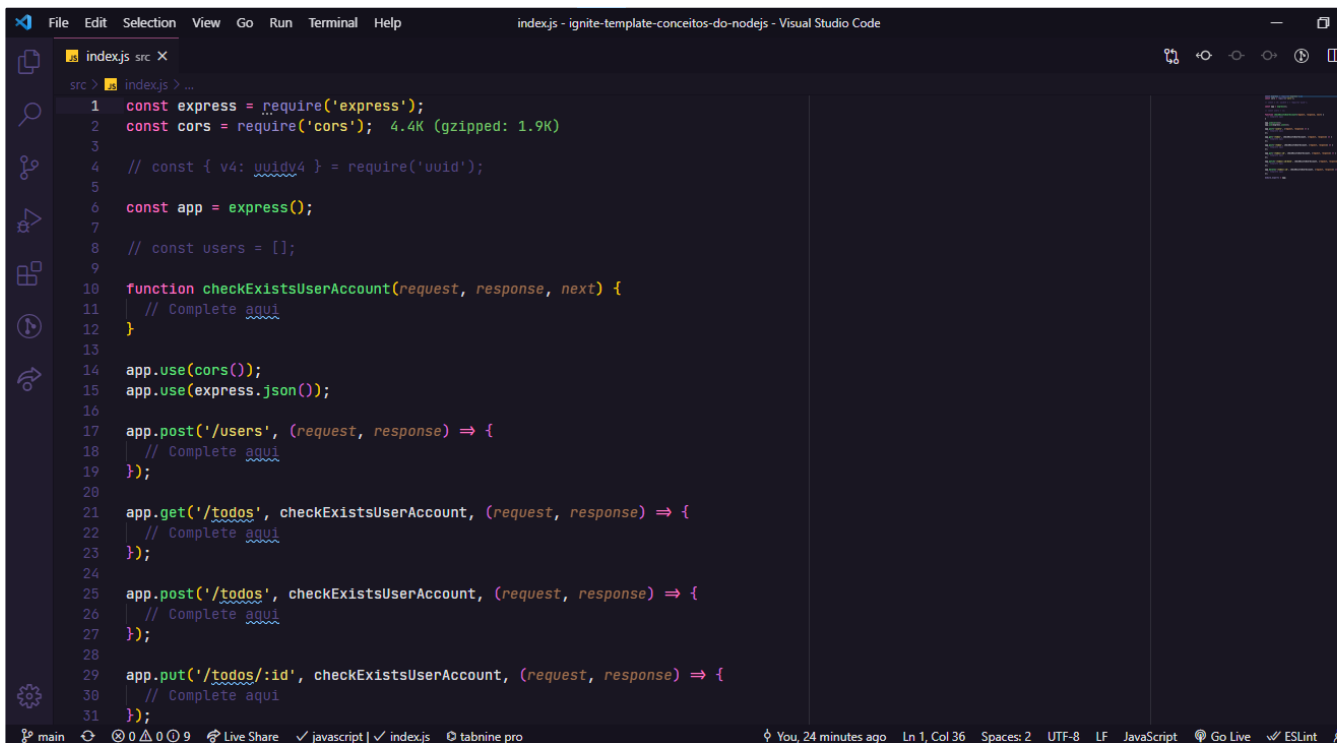
Ignite



Node.js

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:



```
1  const express = require('express');
2  const cors = require('cors'); // 4.4K (gzipped: 1.9K)
3
4  // const { v4: uuidv4 } = require('uuid');
5
6  const app = express();
7
8  // const users = [];
9
10 function checkExistsUserAccount(request, response, next) {
11   // Complete aqui
12 }
13
14 app.use(cors());
15 app.use(express.json());
16
17 app.post('/users', (request, response) => {
18   // Complete aqui
19 });
20
21 app.get('/todos', checkExistsUserAccount, (request, response) => {
22   // Complete aqui
23 });
24
25 app.post('/todos', checkExistsUserAccount, (request, response) => {
26   // Complete aqui
27 });
28
29 app.put('/todos/:id', checkExistsUserAccount, (request, response) => {
30   // Complete aqui
31 });
```

## 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.

### POST `/users`

A rota deve receber `name`, e `username` dentro do corpo da requisição. Ao cadastrar um novo usuário, ele deve ser armazenado dentro de um objeto no seguinte formato:

```
{ id: 'uuid', // precisa ser um uuid name: 'Danilo Vieira', username: 'danilo', todos:
[] }
```

Certifique-se que o ID seja um UUID, e de sempre iniciar a lista `todos` como um array vazio. O objeto do usuário deve ser retornado na resposta da requisição.

## GET `/todos`

A rota deve receber, pelo header da requisição, uma propriedade `username` contendo o username do usuário e retornar uma lista com todas as tarefas desse usuário.

## POST `/todos`

A rota deve receber `title` e `deadline` dentro do corpo da requisição e, uma propriedade `username` contendo o username do usuário dentro do header da requisição. Ao criar um novo *todo*, ele deve ser armazenada dentro da lista `todos` do usuário que está criando essa tarefa. Cada tarefa deverá estar no seguinte formato: . Certifique-se que o ID seja um UUID.

```
{ id: 'uuid', // precisa ser um uuid title: 'Nome da tarefa', done: false, deadline: '2021-02-27T00:00:00.000Z', created_at: '2021-02-22T00:00:00.000Z' }
```

**Observação:** Lembre-se de iniciar a propriedade `done` sempre como `false` ao criar um *todo*.

**Dica:** Ao fazer a requisição com o Insomnia ou Postman, preencha a data de `deadline` com o formato `ANO-MÊS-DIA` e ao salvar a tarefa pela rota, faça da seguinte forma:

```
{ id: 'uuid', // precisa ser um uuid title: 'Nome da tarefa', done: false, deadline: new Date(deadline), created_at: new Date() }
```

Usar `new Date(deadline)` irá realizar a transformação da string "ANO-MÊS-DIA" (por exemplo "2021-02-25") para uma data válida do JavaScript.

O objeto do `todo` deve ser retornado na resposta da requisição.

## PUT `/todos/:id`

A rota deve receber, pelo header da requisição, uma propriedade `username` contendo o username do usuário e receber as propriedades `title` e `deadline` dentro do corpo. É preciso alterar **apenas** o `title` e o `deadline` da tarefa que possua o `id` igual ao `id` presente nos parâmetros da rota.

## PATCH `/todos/:id/done`

A rota deve receber, pelo header da requisição, uma propriedade `username` contendo o username do usuário e alterar a propriedade `done` para `true` no *todo* que possuir um `id` igual ao `id` presente nos parâmetros da rota.

## DELETE `/todos/:id`

A rota deve receber, pelo header da requisição, uma propriedade `username` contendo o username do usuário e excluir o *todo* que possuir um `id` igual ao `id` presente nos parâmetros da rota.

## Especificação dos testes

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

💡 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 usuários

- Should be able to create a new user

Para que esse teste passe, você deve permitir que um usuário seja criado e retorne um JSON com o usuário criado. Você pode ver o formato de um usuário [aqui](#).

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

- Should not be able to create a new user when username already exists

Para que esse teste passe, antes de criar um usuário você deve validar se outro usuário com o mesmo `username` já existe. Caso exista, retorne uma resposta com status `400` e um json no seguinte formato:

```
{ error: 'Mensagem do erro' }
```

A mensagem pode ser de sua escolha, desde que a propriedade seja `error`.

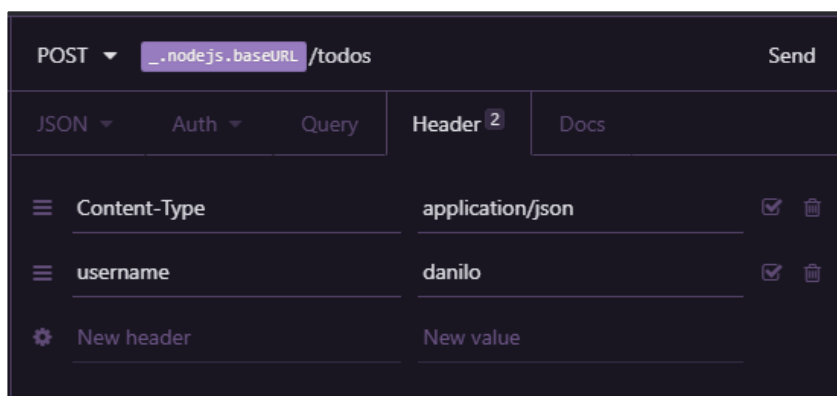
## Testes de *todos*

### Middleware

Para completar todos os testes referentes à *todos* é necessário antes ter completado o código que falta no middleware `checkExistsUserAccount`. Para isso, você deve pegar o `username` do usuário no header da requisição, verificar se esse usuário existe e então colocar esse usuário dentro da `request` antes de chamar a função `next`. Caso o usuário não seja encontrado, você deve retornar uma resposta contendo status `404` e um json no seguinte formato:

```
{ error: 'Mensagem do erro' }
```

Observação: O username deve ser enviado pelo header em uma propriedade chamada `username`:



- Should be able to list all user's todos

Para que esse teste passe, na rota GET `/todos` é necessário pegar o usuário que foi repassado para o `request` no middleware `checkExistsUserAccount` e então retornar a lista `todos` que está no objeto do usuário conforme foi criado para satisfazer o [primeiro teste](#).

- Should be able to create a new todo

Para que esse teste passe, na rota POST `/todos` é necessário pegar o usuário que foi repassado para o `request` no middleware `checkExistsUserAccount`, pegar também o `title` e o `deadline` do corpo da requisição e adicionar um novo *todo* na lista `todos` que está no objeto do usuário. Lembre-se de seguir a estrutura padrão de um *todo* como mostrado [aqui](#).

Após adicionar o novo *todo* na lista, é necessário retornar um status `201` e o *todo* no corpo da resposta.

- **Should be able to update a todo**

Para que esse teste passe, na rota PUT `/todos/:id` é necessário atualizar um *todo* existente, recebendo o `title` e o `deadline` pelo corpo da requisição e o `id` presente nos parâmetros da rota.

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

Para que esse teste passe, você não deve permitir a atualização de um *todo* que não existe e retornar uma resposta contendo um status `404` e um json no seguinte formato:

```
{ error: 'Mensagem do erro' }
```

- **Should be able to mark a todo as done**

Para que esse teste passe, na rota PATCH `/todos/:id/done` você deve mudar a propriedade `done` de um *todo* de `false` para `true`, recebendo o `id` presente nos parâmetros da rota.

- **Should not be able to mark a non existing todo as done**

Para que esse teste passe, você não deve permitir a mudança da propriedade `done` de um *todo* que não existe e retornar uma resposta contendo um status `404` e um json no seguinte formato:

```
{ error: 'Mensagem do erro' }
```

- **Should be able to delete a todo**

Para que esse teste passe, DELETE `/todos/:id` você deve permitir que um *todo* seja excluído usando o `id` passado na rota. O retorno deve ser apenas um status `204` que representa uma resposta sem conteúdo.

- **Should not be able to delete a non existing todo**

Para que esse teste passe, você não deve permitir excluir um *todo* que não exista e retornar uma resposta contendo um status `404` e um json no seguinte 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!](#)

## Solução do desafio