

 [rocketseat-education](#) / [bootcamp-gostack-desafios](#)**Code**

Issues 5

Pull requests 3


Actions

Projects

Wiki

Security

...

 master ▾

...

[bootcamp-gostack-desafios](#) / [desafio-reactjs-crud](#) /

danilo-vieira ...

on 18 Aug



..



assets

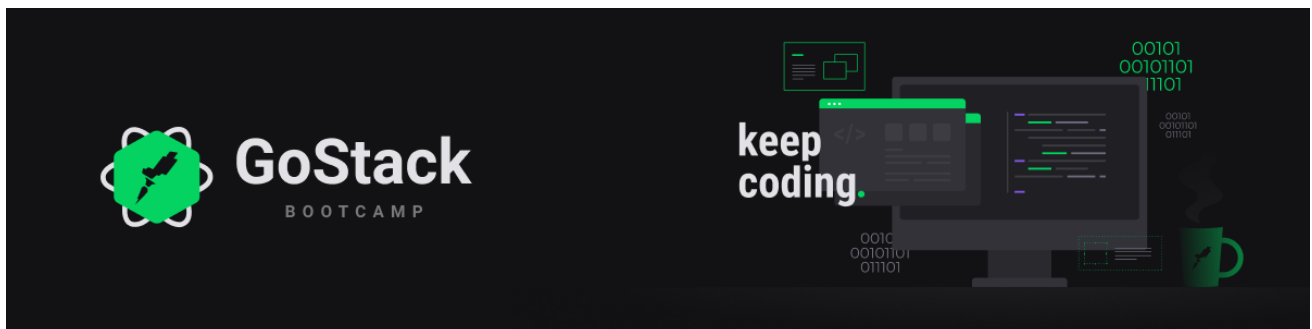
last month



README.md

last month

README.md



Desafio 10: GoRestaurant Web

“O tempo que leva para realizar seus sonhos vai passar de qualquer forma”!

languages 0

made by Rocketseat

license MIT

 Stars

810

[Sobre o desafio](#) | [Entrega](#) | [Licença](#)

Sobre o desafio

Nesse desafio, você irá desenvolver mais uma aplicação, a GoRestaurant. Agora você irá praticar o que você aprendeu até agora no React.js junto com TypeScript, praticando o conceito de CRUD (Create, Read, Update, Delete).

Essa será uma aplicação que irá se conectar a uma fake API, e exibir os pratos de comida criados e permitir a criação, remoção e atualização desses pratos.

Template da aplicação

Para te ajudar nesse desafio, criamos para você um modelo que você deve utilizar como um template do Github.

O template está disponível na seguinte url: [Acessar Template](#)

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

Agora navegue até a pasta criada e abra no Visual Studio Code, lembre-se de executar o comando `yarn` no seu terminal para instalar todas as dependências.

Utilizando uma fake API

Antes de tudo, para que você tenha os dados para exibir em tela, criamos um arquivo que você poderá utilizar como fake API para te prover esses dados.

Para isso, deixamos instalado no seu `package.json` uma dependência chamada `json-server`, e um arquivo chamado `server.json` que contém os dados para uma rota `/foods`. Para executar esse servidor você pode executar o seguinte comando:

```
yarn json-server server.json -p 3333
```

Layout da aplicação

Essa aplicação possui um layout que você pode seguir para conseguir visualizar o seu funcionamento.

O layout pode ser acessado através da página do Figma, no [seguinte link](#).

Você precisará uma conta (gratuita) no Figma pra inspecionar o layout e obter detalhes de cores, tamanhos, etc.

Funcionalidades da aplicação

Agora que você já está com o template clonado e pronto para continuar, você deve verificar os arquivos da pasta `src` e completar onde não possui código, com o código para atingir os objetivos de cada rota.

- **Listar os pratos de comida da sua API**: Sua página `Dashboard` deve ser capaz de exibir uma listagem, com o campo `title`, `value`, e `description` e `available` de todos os pratos de comida que estão cadastrados na sua API.

Dica: Para exibir se o prato de comida está disponível ou não, você pode validar o campo `available` que é retornado da API e exibir `Disponível` caso seja `true`, e `Indisponível` caso seja `false`.

- **Adicionar novos pratos de comida a sua API :** Em sua página Dashboard você deve abrir um modal ao clicar no botão `Novo Prato` no Header. Esse modal deve ser responsável por cadastrar uma nova `food` passando os campos `image` , `name` , `description` , `value` .

Dica 1: O campo `image` deve ser uma URL, deixamos três URL de imagens como exemplo no arquivo `server.json`.

Dica 2: Ao enviar o request para sua API para salvar a `food` , lembre-se sempre de setar o campo `available` como `true`.

- **Editar pratos de comida da sua API :** Em sua página Dashboard você deve abrir um modal ao clicar no botão `Editar Prato` . Esse modal deve ser responsável por editar uma `food` passando os campos `image` , `name` , `description` , `value` .

Dica: Ao editar um item, quando for envia-lo para o backend, lembre de copiar os dados anteriores como o `available` e o `id` , ou eles serão perdidos do seu arquivo `server.json`.

- **Remover pratos de comida da sua API :** Em sua página Dashboard você deve remover um prato de comida ao clicar no botão com ícone de lixeira no componente `Food`.

Dica: Após remover o item da sua API, lembre-se de remover ele também da listagem.

- **Alterar disponibilidade dos pratos de comida da sua API :** Em sua página Dashboard você deve alterar a disponibilidade de um prato de comida ao clicar no switch que é controlado pelo valor de `available` .

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, dé uma olhada em [nosso FAQ](#).

Para esse desafio, temos os seguintes testes:

- **should be able to list all the food plates from your api :** Para que esse teste passe, sua aplicação deve permitir que sejam listados, toda os pratos de comidas que são retornadas da sua fake API.

- **should be able to add a new food plate** : Para que esse teste passe, você deve permitir que um prato de comida seja adicionado a sua api, adicionando-o também à listagem.
- **should be able to edit a food plate** : Para que esse teste passe, você deve permitir que um prato de comida seja editado na sua api, editando-o também na listagem.
- **should be able to remove a food plate** : Para que esse teste passe, você deve permitir que um prato de comida seja removido da sua api, removendo-o também da listagem.
- **should be able to update the availability of a food plate** : Para que esse teste passe, em sua dashboard você deve permitir que o status do prato de comida seja alterado entre Disponível e Indisponível ;



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, fazer um post no LinkedIn e postar o código no Github é uma boa forma de demonstrar seus conhecimentos e esforços para evoluir na sua carreira para oportunidades futuras.

Solução do desafio

Caso você queira ver como resolver o desafio, fizemos um video explicando o passo a passo para cumprir com todos os requisitos da aplicação:





Licença

Esse projeto está sob a licença MIT. Veja o arquivo [LICENSE](#) para mais detalhes.

Feito com  by Rocketseat  [Entre na nossa comunidade!](#)
