Rocketseat / **bootcamp-gostack-desafios**

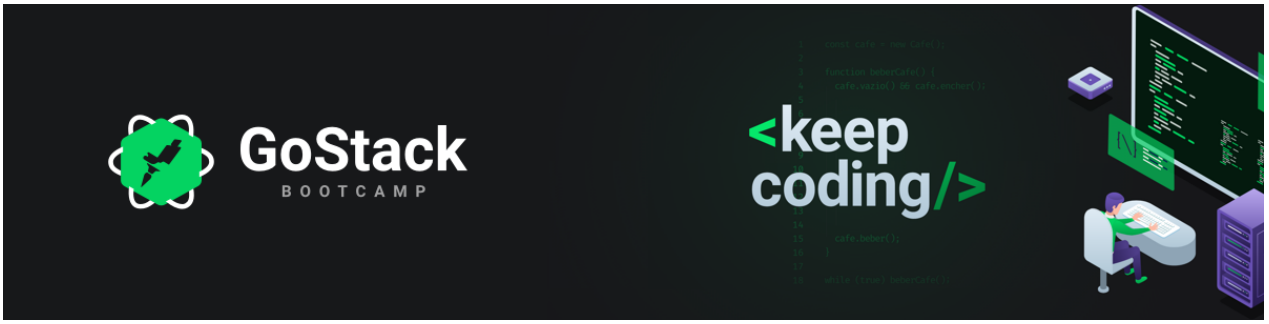
master | bootcamp-gostack-desafios / desafio-fundamentos-reactjs / | ... | Gitpod

StefanoSaffran committed on 25 May ... | History

..

assets	2 months ago
README.md	last month

README.md



Desafio 07: GoFinances Web

> "Não espere resultados brilhantes se suas metas não forem claras"!

Sobre o desafio | Entrega | Licença

Sobre o desafio

Nesse desafio, você deve continuar desenvolvendo a aplicação de gestão de transações, a GoFinances. Agora você irá praticar o que você aprendeu até agora no React.js junto com TypeScript, utilizando rotas e envio de arquivos por formulário.

Essa será uma aplicação que irá se conectar ao seu backend do Desafio 06, e exibir as transações criadas e permitir a importação de um arquivo CSV para gerar novos registros no banco de dados.

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.

Preparando o backend

Antes de tudo, para que seu frontend se conecte corretamente ao backend, vá até a pasta do seu `backend` e execute os comandos `yarn add cors` e depois `yarn add @types/cors -D`.

Depois disso vá até o seu `app.ts` ainda no backend, e importe o `cors` e adicione `app.use(cors())` antes da linha que contém `app.use(routes)`;

Além disso, tenha certeza que as informações da categoria, estão sendo retornadas junto com a transação do seu backend no formato como o seguinte:

```
{
  "id": "c0512e43-6589-4dc8-bd0c-2a3f71b347aa",
  "title": "Loan",
  "type": "income",
  "value": "1500.00",
  "category_id": "d93eccc7-64bb-4268-b825-9200103f3a8b",
  "created_at": "2020-04-20T00:00:49.620Z",
  "updated_at": "2020-04-20T00:00:49.620Z",
  "category": {
    "id": "d93eccc7-64bb-4268-b825-9200103f3a8b",
    "title": "Others",
    "created_at": "2020-04-20T00:00:49.594Z",
    "updated_at": "2020-04-20T00:00:49.594Z"
  }
}
```

Para isso, você pode utilizar a funcionalidade de eager loading do TypeORM, adicionando o seguinte na sua model de transactions:

```
@ManyToOne(() => Category, category => category.transaction, { eager: true })
@JoinColumn({ name: 'category_id' })
category: Category;
```

Lembre também de fazer o mesmo na model de Category, mas referenciando a tabela de Transaction.

```
@OneToMany(() => Transaction, transaction => transaction.category)
transaction: Transaction;
```

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 as transações da sua API`**: Sua página `Dashboard` deve ser capaz de exibir uma listagem através de uma tabela, com o campo `title`, `value`, `type` e `category` de todas as transações que estão cadastradas na sua API.

Dica: Você pode utilizar a função Intl para formatar os valores. Dentro da pasta `utils` no template você encontrará um código para te ajudar.

- **`Exibir o balance da sua API`**: Sua página `Dashboard`, você deve exibir o balance que é retornado do seu backend, contendo o total geral, junto ao total de entradas e saídas.

- **`Importar arquivos CSV`**: Na sua página `Import`, você deve permitir o envio de um arquivo no formato `csv` para o seu backend, que irá fazer a importação das transações para o seu banco de dados. O arquivo csv deve seguir o seguinte modelo.

Dica: Deixamos disponível um componente chamado `Upload` na pasta `components` para você ter já preparado uma opção de drag-n-drop para o upload de arquivos. PS: Caso você esteja no windows e esteja sofrendo com algum erro ao tentar importar CSV, altere o tipo de arquivo dentro do arquivo `components/upload/index.ts` de `text/csv` para `.csv, application/vnd.ms-excel, text/csv`.

Dica 2: Utilize o FormData() para conseguir enviar o seu arquivo para o seu backend.

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 the total balance inside the cards`**: Para que esse teste passe, sua aplicação deve permitir que seja exibido na sua Dashboard, cards contendo o total de `income`, `outcome` e o total da subtração de `income - outcome` que são retornados pelo balance do seu backend.

- **`should be able to list the transactions`**: Para que esse teste passe, sua aplicação deve permitir que sejam listados dentro de uma tabela, toda as transações que são retornadas do seu backend.

Dica: Para a exibição dos valores na listagem de transações, as transações com tipo `income` devem exibir os valores no formado `R$ 5.500,00`. Transações do tipo `outcome` devem exibir os valores no formado `- R$ 5.500,00`.

- **`should be able to navigate to the import page`**: Para que esse teste passe, você deve permitir a troca de página através do Header, pelo botão que contém o nome `Importar`.

Dica: Utilize o componente `Link` que é exportado do `react-router-dom`, passando a propriedade `to` que leva para a página `/import`.

- **`should be able to upload a file`**: Para que esse teste passe, você deve permitir que um arquivo seja enviado através do componente de drag-n-drop na página de `import`, e que seja possível exibir o nome do arquivo enviado para o input.

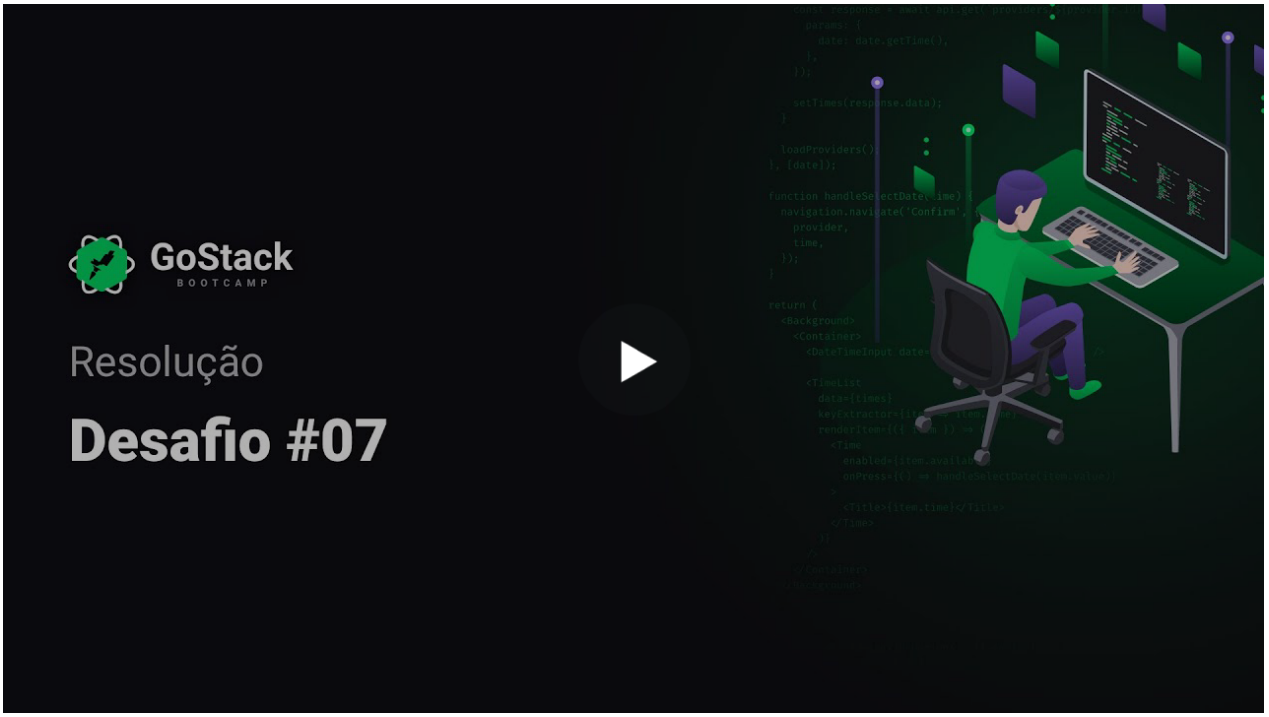
Dica: Deixamos disponível um componente chamado `FileList` na pasta `components` para ajudar você a listar os arquivos que enviar pelo componente de `Upload`, ele deve exibir o título do arquivo e o tamanho dele.

🗓 Entrega

Esse desafio deve ser entregue a partir da plataforma Skylab, 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

Feito com ❤ by Rocketseat 👌 Entre na nossa comunidade!