Rocketseat / **bootcamp-gostack-desafios**

Code | Issues 7 | Pull requests 1 | Actions | Projects | Wiki | Security

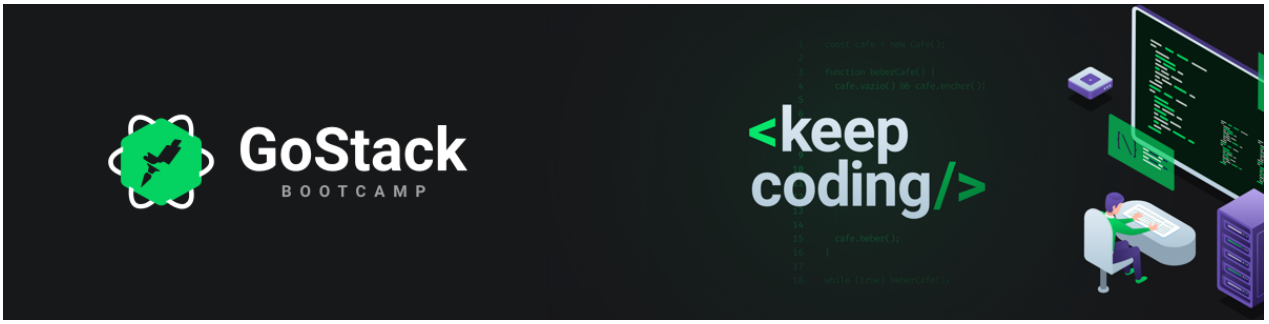
master | bootcamp-gostack-desafios / desafio-database-upload / | ... | Gitpod

StefanoSaffran committed on 25 May ... History

..

assets	2 months ago
README.md	last month

README.md



Desafio 06: Banco de dados e upload de arquivos no Node.js

> "Só deseje as coisas as quais você está disposto a lutar"!

languages 0 | made by Rocketseat | license MIT | Stars 488

Sobre o desafio | Entrega | Licença

Sobre o desafio

Nesse desafio, você deve continuar desenvolvendo a aplicação de gestão de transações, treinando o que você aprendeu até agora no Node.js junto ao TypeScript, mas dessa vez incluindo o uso de banco de dados com o TypeORM e envio de arquivos com o Multer!

Essa será uma aplicação que deve armazenar transações financeiras de entrada e saída e permitir o cadastro e a listagem dessas transações, além de permitir a criação de novos registros no banco de dados a partir do envio de um arquivo csv.

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.

Rotas 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.

- **`POST /transactions`**: A rota deve receber `title`, `value`, `type`, e `category` dentro do corpo da requisição, sendo o `type` o tipo da transação, que deve ser `income` para entradas (depósitos) e `outcome` para saídas (retiradas). Ao cadastrar uma nova transação, ela deve ser armazenada dentro do seu banco de dados, possuindo os campos `id`, `title`, `value`, `type`, `category_id`, `created_at`, `updated_at`.

Dica: Para a categoria, você deve criar uma nova tabela, que terá os campos `id`, `title`, `created_at`, `updated_at`.

Dica 2: Antes de criar uma nova categoria, sempre verifique se já existe uma categoria com o mesmo título. Caso ela exista, use o `id` já existente no banco de dados.

```
{
  "id": "uuid",
  "title": "Salário",
  "value": 3000,
  "type": "income",
  "category": "Alimentação"
}
```

- **`GET /transactions`**: Essa rota deve retornar uma listagem com todas as transações que você cadastrou até agora, junto com o valor da soma de entradas, retiradas e total de crédito. Essa rota deve retornar um objeto o seguinte formato:

```
{
  "transactions": [
    {
      "id": "uuid",
      "title": "Salário",
      "value": 4000,
      "type": "income",
      "category": {
        "id": "uuid",
        "title": "Salary",
        "created_at": "2020-04-20T00:00:49.620Z",
        "updated_at": "2020-04-20T00:00:49.620Z"
      },
      "created_at": "2020-04-20T00:00:49.620Z",
      "updated_at": "2020-04-20T00:00:49.620Z"
    },
    {
```

```
      "id": "uuid",
      "title": "Freela",
      "value": 2000,
      "type": "income",
      "category": {
        "id": "uuid",
        "title": "Others",
        "created_at": "2020-04-20T00:00:49.620Z",
        "updated_at": "2020-04-20T00:00:49.620Z"
      },
      "created_at": "2020-04-20T00:00:49.620Z",
      "updated_at": "2020-04-20T00:00:49.620Z"
    },
    {
      "id": "uuid",
      "title": "Pagamento da fatura",
      "value": 4000,
      "type": "outcome",
      "category": {
        "id": "uuid",
        "title": "Others",
        "created_at": "2020-04-20T00:00:49.620Z",
        "updated_at": "2020-04-20T00:00:49.620Z"
      },
      "created_at": "2020-04-20T00:00:49.620Z",
      "updated_at": "2020-04-20T00:00:49.620Z"
    },
    {
      "id": "uuid",
      "title": "Cadeira Gamer",
      "value": 1200,
      "type": "outcome",
      "category": {
        "id": "uuid",
        "title": "Recreation",
        "created_at": "2020-04-20T00:00:49.620Z",
        "updated_at": "2020-04-20T00:00:49.620Z"
      },
      "created_at": "2020-04-20T00:00:49.620Z",
      "updated_at": "2020-04-20T00:00:49.620Z"
    }
  ],
  "balance": {
    "income": 6000,
    "outcome": 5200,
    "total": 800
  }
}
```


Dica: Dentro de balance, o income é a soma de todos os valores das transações com `type` income. O outcome é a soma de todos os valores das transações com `type` outcome, e o total é o valor de `income - outcome`.

Dica 2: Para fazer a soma dos valores, você pode usar a função reduce para agrupar as transações pela propriedade `type`, assim você irá conseguir somar todos os valores com facilidade e obter o retorno do `balance`.

- `DELETE /transactions/:id`: A rota deve deletar uma transação com o `id` presente nos parâmetros da rota;

- `POST /transactions/import`: A rota deve permitir a importação de um arquivo com formato `.csv` contendo as mesmas informações necessárias para criação de uma transação `id`, `title`, `value`, `type`, `category_id`, `created_at`, `updated_at`, onde cada linha do arquivo CSV deve ser um novo registro para o banco de dados, e por fim retorne todas as `transactions` que foram importadas para seu banco de dados. O arquivo csv, deve seguir o seguinte modelo

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:

⚠ **Antes de rodar os testes, crie um banco de dados com o nome "gostack_desafio06_tests" para que todos os testes possam executar corretamente** ⚠

- `should be able to create a new transaction`: Para que esse teste passe, sua aplicação deve permitir que uma transação seja criada, e retorne um json com a transação criado.

- `should create tags when inserting new transactions`: Para que esse teste passe, sua aplicação deve permitir que ao criar uma nova transação com uma categoria que não existe, essa seja criada e inserida no campo category_id da transação com o `id` que acabou de ser criado.

- `should not create tags when they already exists`: Para que esse teste passe, sua aplicação deve permitir que ao criar uma nova transação com uma categoria que já existe, seja atribuído ao campo category_id da transação com o `id` dessa categoria existente, não permitindo a criação de categorias com o mesmo `title`.

- `should be able to list the transactions`: Para que esse teste passe, sua aplicação deve permitir que seja retornado um array de objetos contendo todas as transações junto ao balanço de income, outcome e total das transações que foram criadas até o momento.

- `should not be able to create outcome transaction without a valid balance`: Para que esse teste passe, sua aplicação não deve permitir que uma transação do tipo `outcome` extrapole o valor total que o usuário tem em caixa (total de income), retornando uma resposta com código HTTP 400 e uma mensagem de erro no seguinte formato: `{ error: string }`.

- `should be able to delete a transaction`: Para que esse teste passe, você deve permitir que a sua rota de delete exclua uma transação, e ao fazer a exclusão, ele retorne uma resposta vazia, com status 204.

- `should be able to import transactions`: Para que esse teste passe, sua aplicação deve permitir que seja importado um arquivo csv, contendo o seguinte modelo. Com o arquivo importado, você deve permitir que seja criado no banco de dados todos os registros e

categorias que estavam presentes nesse arquivo, e retornar todas as transactions que foram importadas.

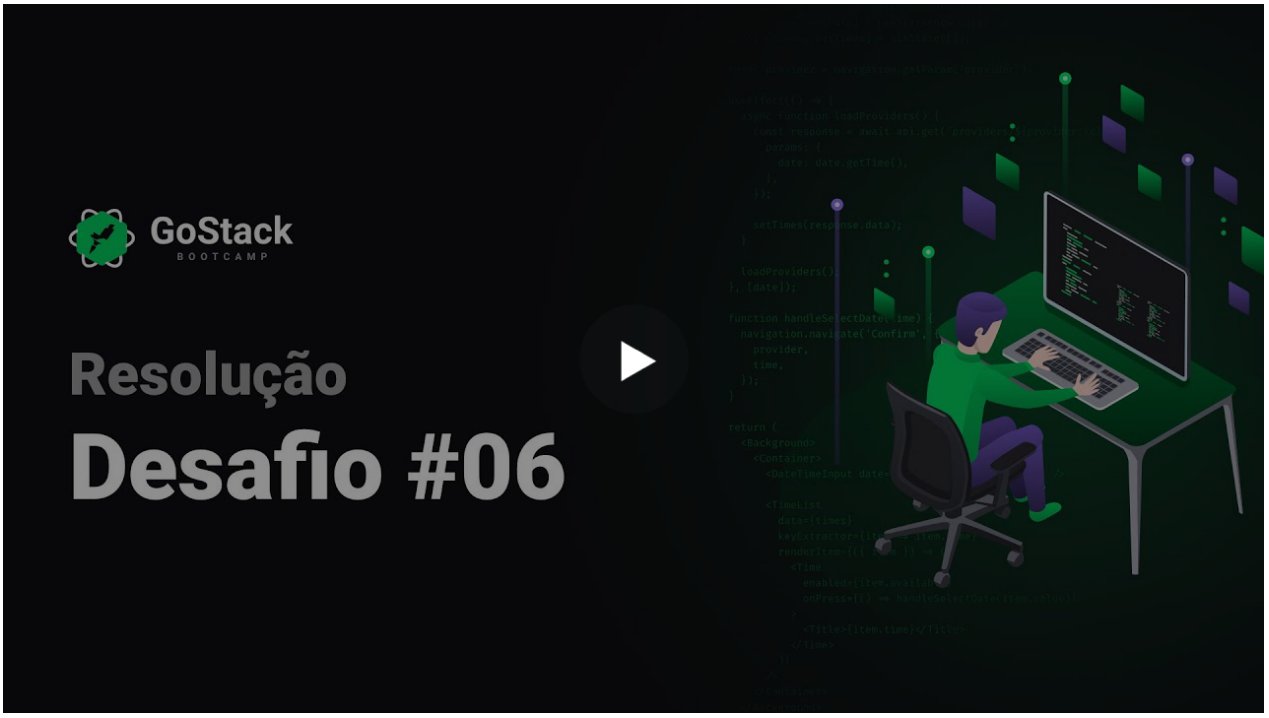
Dica: Caso você tenha dificuldades com a leitura de CSV, temos um guia no Notion.

📆 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

Esse projeto está sob a licença MIT. Veja o arquivo LICENSE para mais detalhes.

Feito com ❤ by Rocketseat 👋 Entre na nossa comunidade!