Adicionando Novas Rotas e Funcionalidades

Agora que conseguimos obter todas as transações do banco de dados, vamos continuar adicionando novas funcionalidades à nossa API. Nesta aula, vamos aprender como adicionar novas transações.

Objetivos da Aula:

- Compreender como criar rotas para adicionar novas transações
- Configurar controladores para manipular as requisições de criação de transações

1. **Configurando a Rota para Adicionar Transações**

Atualizar o Arquivo transactions.js

1. Abra o arquivo **transactions.js** dentro da pasta **routes**:

Vamos adicionar uma nova rota **POST** para adicionar transações.

2. Adicione o seguinte código ao arquivo **transactions.js**:

Código JavaScript

const express = require('express'); // Importa o framework Express
const router = express.Router(); // Cria um novo roteador
const transactionsController = require('../controllers/transactionsController'); // Importa o controlador de transações

// Definindo uma rota para obter todas as transações
router.get('/', transactionsController.getAllTransactions);

// Definindo uma rota para adicionar uma nova transação
⇨ **router.post('/', transactionsController.addTransaction);**

// Exportando o roteador
module.exports = router;

Explicação do Código:

- **const express = require('express');:** Importa o framework Express e armazena-o em uma constante chamada express.
- **const router = express.Router();:** Cria um novo roteador com o Express e armazena-o em uma constante chamada **router**.
- **const transactionsController = require('../controllers/transactionsController');:** Importa o controlador de transações e armazena-o em uma constante chamada **transactionsController**.

- **router.get('/', transactionsController.getAllTransactions);:** Define uma rota **GET** para a URL raiz das transações (/). Quando um cliente faz uma requisição **GET** para essa URL, a função **getAllTransactions** do controlador será executada.
- **router.post('/', transactionsController.addTransaction);:** Define uma rota **POST** para a URL raiz das transações (/). Quando um cliente faz uma requisição **POST** para essa URL, a função **addTransaction** do controlador será executada.
- **module.exports = router;:** Exporta o roteador para que possa ser utilizado em outros arquivos.

Configurar o Controlador para Adicionar Transações

1. Abra o arquivo **transactionsController.js** dentro da pasta **controllers**:

Vamos adicionar uma nova função para adicionar transações.

2. Adicione o seguinte código ao arquivo **transactionsController.js**:

Código JavaScript

```
const db = require('../config/db'); // Importa a conexão com o banco de dados

// Função para obter todas as transações
const getAllTransactions = (req, res) => {
  db.query('SELECT * FROM transactions', (err, results) => {
    if (err) {
      console.error('Erro ao obter transações:', err);
      res.status(500).send('Erro ao obter transações');
      return;
    }
    res.json(results);
  });
};

// Função para adicionar uma nova transação
const addTransaction = (req, res) => {
  const { date, amount, description, category, account, user_id } = req.body;
  db.query(
    'INSERT INTO transactions (date, amount, description, category, account, user_id) VALUES
(?, ?, ?, ?, ?, ?)',
    [date, amount, description, category, account, user_id],
    (err, results) => {
      if (err) {
        console.error('Erro ao adicionar transação:', err);
        res.status(500).send('Erro ao adicionar transação');
        return;
      }
      res.status(201).send('Transação adicionada com sucesso');
    }
  );
};
```

```
module.exports = {
  getAllTransactions,
  addTransaction
};
```

Explicação do Código:

- **const db = require('../config/db');:** Importa a conexão com o banco de dados.
- **const getAllTransactions = (req, res) => {...};:** Define uma função arrow chamada **getAllTransactions** que recebe dois parâmetros, **req** (requisição) e **res** (resposta). Esta função será usada para obter todas as transações do banco de dados.
- **const addTransaction = (req, res) => {...};:** Define uma função arrow chamada **addTransaction** que recebe dois parâmetros, **req** (requisição) e **res** (resposta). Esta função será usada para adicionar uma nova transação no banco de dados.
- **const { date, amount, description, category, account, user_id } = req.body;:** Esta linha está usando uma funcionalidade do JavaScript chamada desestruturação. Isso significa que estamos pegando esses campos **(date, amount, description, category, account, user_id)** diretamente do corpo da requisição **(req.body)** e armazenando-os em variáveis separadas com os mesmos nomes.
- **db.query('INSERT INTO transactions (date, amount, description, category, account, user_id) VALUES (?, ?, ?, ?, ?, ?)', [...], (err, results) => {...});:** Esta linha está executando uma **query SQL** para inserir uma nova transação na tabela **transactions**. Os valores a serem inseridos são passados como um **array: [date, amount, description, category, account, user_id]**.
- **(err, results) => {...}:** Esta é uma **função de callback** que é executada após a **query** ser executada. Se houver um erro **(err)**, ele será tratado e uma mensagem de erro será enviada ao cliente. Se a **query** for bem-sucedida, uma mensagem de sucesso será enviada ao cliente.

Parte Prática (1 hora):

Ação:

1. Atualizar o arquivo **transactions.js** para definir a rota de adição de transações.
2. Atualizar o arquivo **transactionsController.js** para definir o controlador de adição de transações.
3. Testar a rota **POST** para **/api/transactions** para garantir que novas transações podem ser adicionadas.

Teste Prático com Insomnia:

Vamos usar o **Insomnia** para testar a rota **POST** para **/api/transactions**.

1. **Abra o Insomnia**.
2. **Crie uma Nova Requisição**:

- No menu lateral esquerdo, clique no botão "+" ao lado de "Debug" para adicionar uma nova requisição.

- Dê um nome à sua requisição, por exemplo, "**Add Transaction**".

3. **Configure a URL da Requisição**:

- Na barra de URL, insira **http://localhost:3000/api/transactions**.

4. **Configure o Corpo da Requisição**:

- Na aba "**Body**", selecione "**JSON**".
- Adicione o seguinte **JSON**:

Código json

```
{
  "date": "2023-07-08",
  "amount": 250.00,
  "description": "Viagem",
  "category": "Lazer",
  "account": "Cartão de Crédito",
  "user_id": 1
}
```

5. **Envie a Requisição**:

- Clique no botão **"Send"** para enviar a requisição.

6. **Verifique a Resposta**:

- Verifique o painel de resposta no **Insomnia**.
- A resposta deve indicar que a transação foi adicionada com sucesso.

Exemplo de Resposta Esperada

Código json

```
{
  "message": "Transação adicionada com sucesso"
}
```

Resolução de Problemas Comuns

Erro 400 (Bad Request)

- Verifique se o corpo da requisição está correto e contém todos os campos necessários.

Erro 500 (Internal Server Error)

- Verifique os logs do servidor para identificar a causa do erro.
- Certifique-se de que a conexão com o banco de dados está configurada corretamente.