Sistemas Distribuídos

PROJETO A3

Relatório

Equipe NESK:

Enrico Falcão - 1272216661

Ignácio Nunes Piva - 1272212685

Lucas Yann Santos Silva -1272215507

Lucas Ribeiro Santiago Nunes - 12722118292

Lúcio Santiago Marques de Queiroz – 12722123508

Relatório da Atividade da A3 (criação de uma aplicação que simule a captação de dados de venda de uma rede de lojas).

Salvador
2023

OBJETIVO

Este relatório tem como objetivo pontuar as principais informações do projeto, evidenciando nossa linha de raciocínio para o desenvolvimento da aplicação e do CRUD, sinalizando os cumprimentos dos requisitos propostos pelos orientadores.

RESUMO

A aplicação é uma simulação de captação de dados de venda de uma rede de lojas. O sistema funciona como um serviço de gerenciar cliente, estoque, vendas e geração de relatórios estatísticos de acordo com os dados captados, toda desenvolvida em Javascript.

SOFTWARES NECESSÁRIOS

PARA EXCUÇÃO DA APLICAÇÃO

Para a execução correta da aplicação proposta, serão necessários os seguintes softwares:

- Javascript
- NodeJS
- Express (framework)
- SQLite (banco de dados)

Justificativa:

Decidimos utilizar as tecnologias citadas acima pelo fato de serem extremamente acessíveis, as tornando mais fáceis para encontrar soluções a dúvidas e problemas. Também optamos por elas por conta de sua praticidade e familiaridade. Juntamente tais fatores, essas ferramentas possuem um grande apoio da comunidade, resultando em um eficiente suporte e viabilizando mais ainda o progresso e desenvolvimento do projeto.

INSTRUÇÕES PARA INSTALAÇÃO

E EXECUÇÃO DA APLICAÇÃO

- Primeiro passo:

Abra um terminal na pasta ***src*** e instale a suas dependências (sqlite3 e express), utilizando o comando *no* ***node: npm install***;

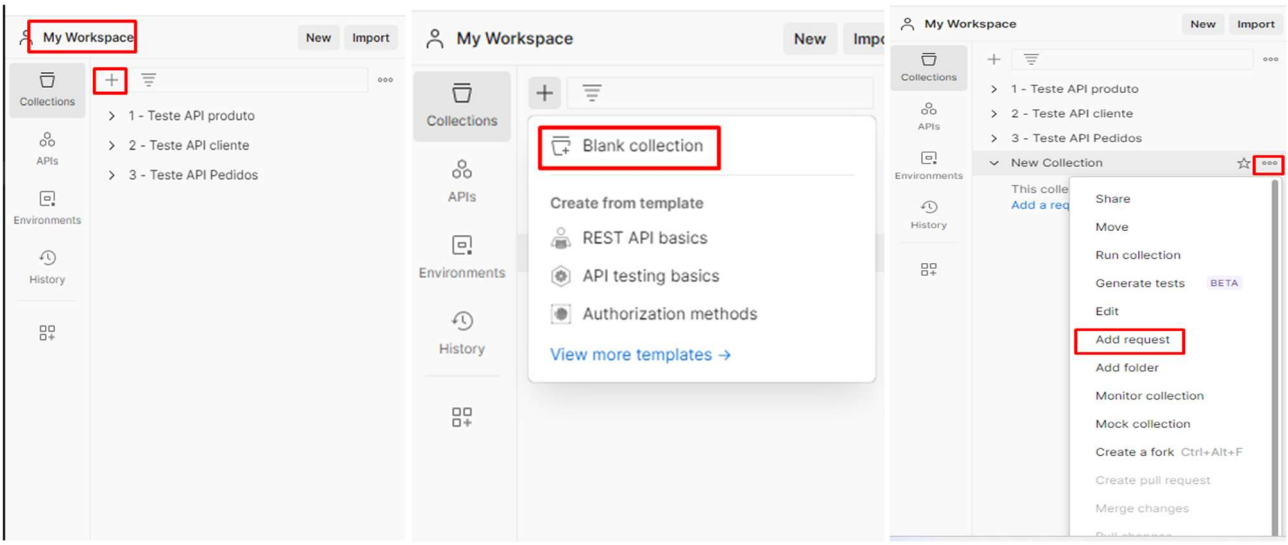
- Segundo passo:

Para iniciar o servidor e inicializar a API, é necessário inserir no terminal na pasta "*src*": *node app.j*s

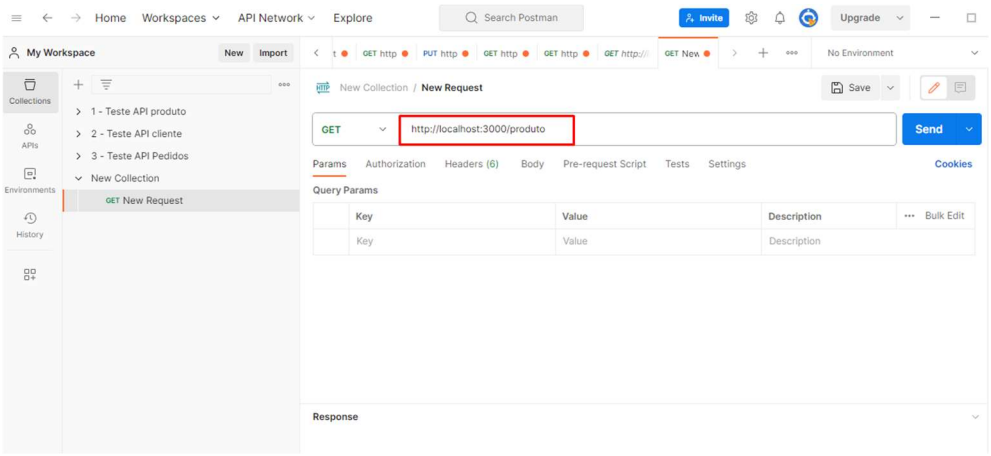
- Terceiro:

Caso deseje testar as funções de CRUD, basta utilizar o programa/software Postman;

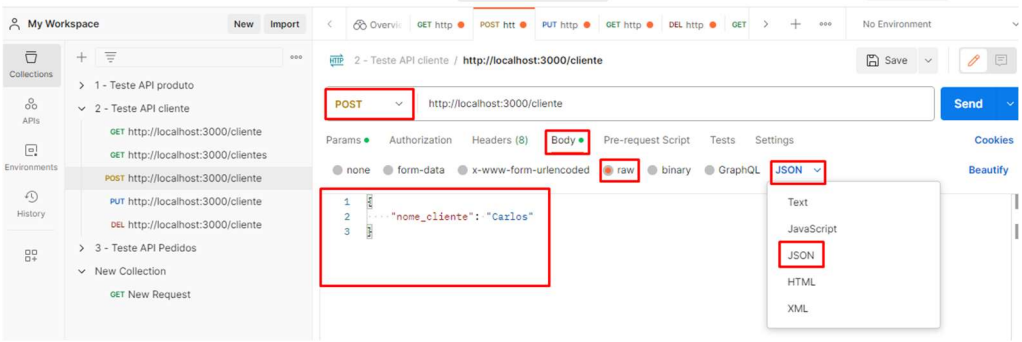
Vá para a área My Workspace, crie uma nova *colection* e adicione um *request*.



Na *request*, escreva a URL da API, o *endpoint* que deseja testar e clique em "Send". O resultado será apresentado no canto inferior da tela. (Neste exemplo, utilizamos o método GET).



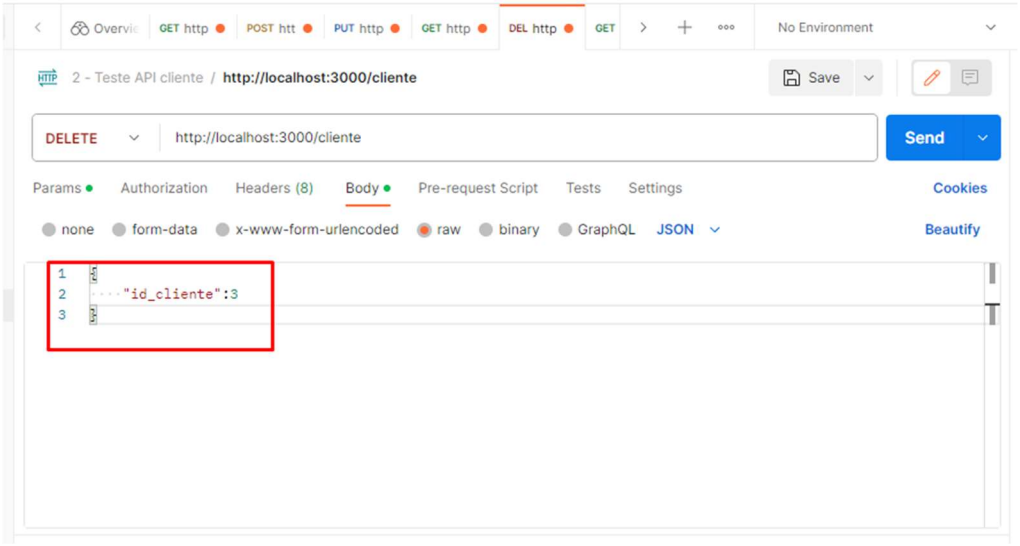
Obs[1].: Para métodos que precisam de parâmetros como POST, PUT e DELETE. É necessário trocar o tipo de requisição e ir até a categoria “Body”, selecionar o tipo “Raw” e selecionar o tipo “JSON”. Em seguida, escreva em formato JSON os atributos que desejar manipular.



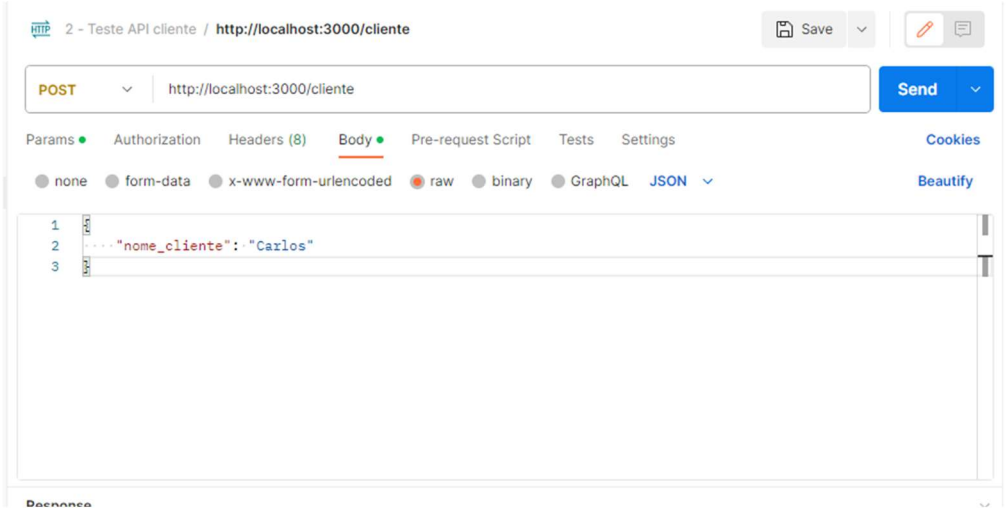
Obs[2].: Segue os atributos para cada tipo de requisição.

- Cliente

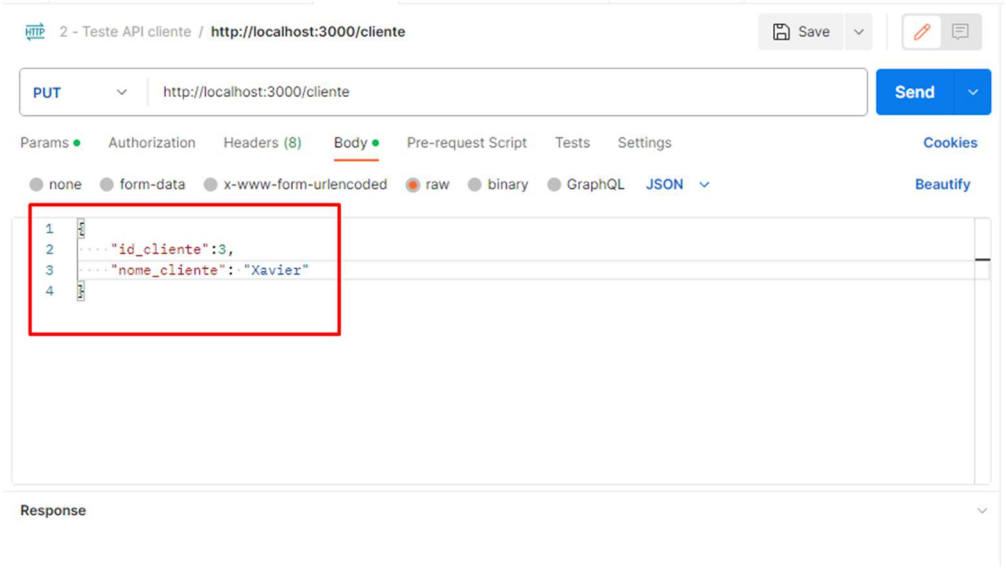
Busca de Cliente por ID e/ou deletar:



Adicionar novo Cliente:

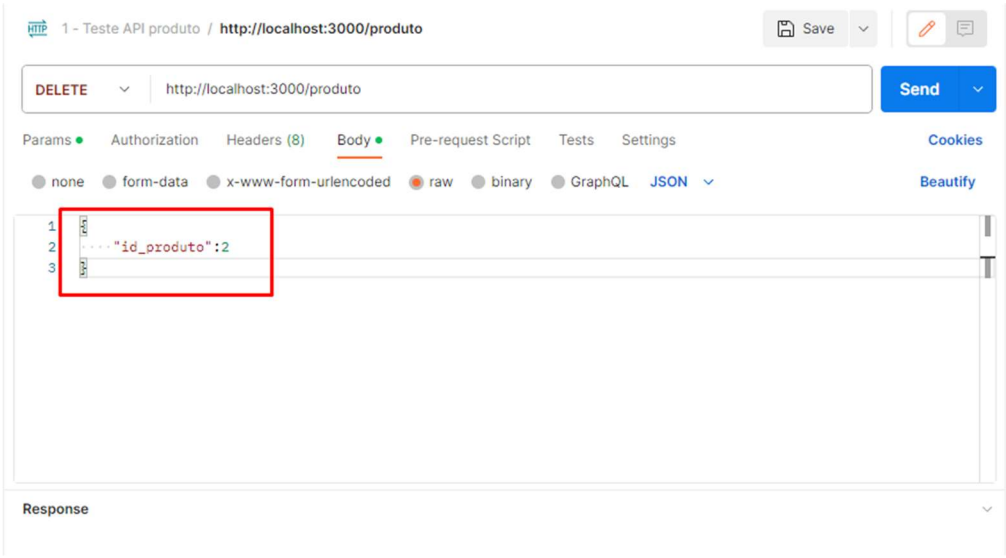


Editar Cliente:

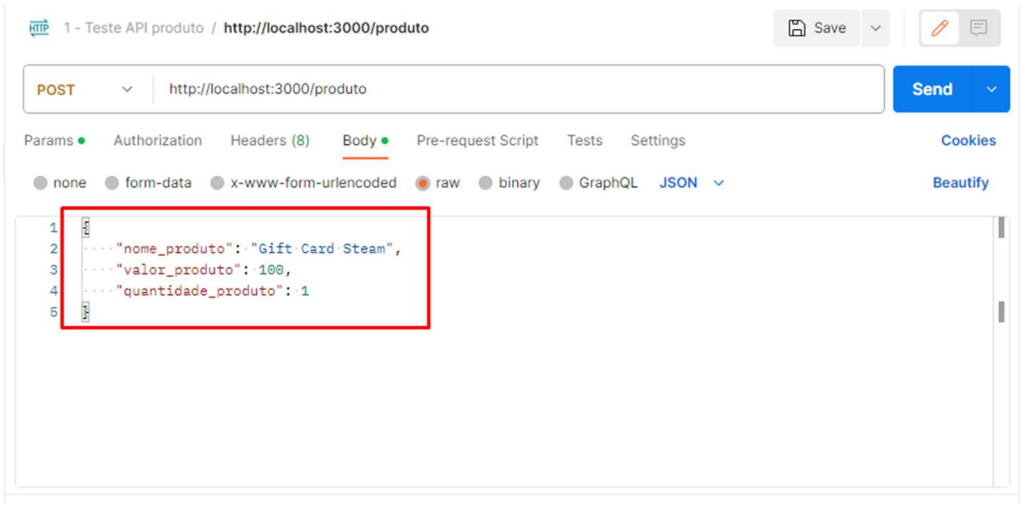


- Produto

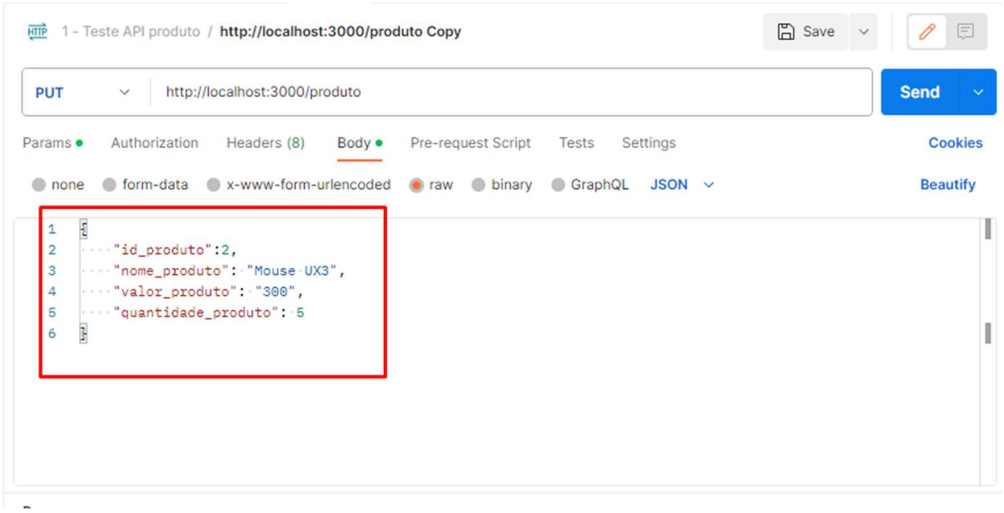
Buscar produto por ID e/ou deletar:



Criar um novo produto:

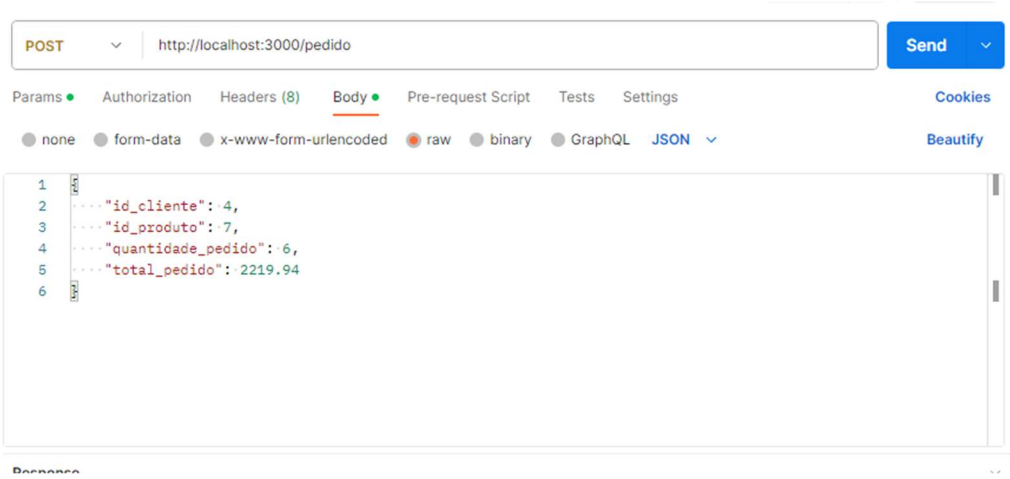


Editar um produto por ID:



- Pedido

Criar um novo pedido:



- Quarto passo:

Caso deseje visualizar os bancos de dados SQLite, instale o *sqlite3* na sua máquina, baixe a extensão MySQL no VSCode. Após baixado a extensão, selecione SQLite e o arquivo "*database.dd*". Porém, todos os dados cadastrados nos bancos podem ser visualizados na pasta "screens", onde há páginas HTML apresentando as informações cadastradas.

APRESENTAÇÃO E DETALHAMENTO SOBRE A

ARQUITETURA, ESTRATÉGIA E ALGORITMOS UTILIZADOS

No projeto, foi criado um arquivo chamado *app.js,* onde se encontra:

- Funções de criações de tabela;

- Funções de importação das rotas para API;

- Função de teste para o bom funcionamento da API.

1. **Rotas em routes.js:**
    - As rotas da aplicação estão definidas no arquivo **routes.js**. Este arquivo é responsável por gerenciar todas as diferentes URLs (endpoints) da sua aplicação.
2. **Biblioteca router do Express:**
    - A biblioteca **router** do Express é utilizada para organizar e modularizar as rotas. O Express Router é uma funcionalidade que permite criar manipuladores de rota modulares, tornando mais fácil dividir as rotas em partes menores e mais gerenciáveis.
3. **Modularização com Controllers:**
    - As funções associadas aos endpoints (URLs) não estão diretamente definidas nas rotas, mas estão sendo importadas da pasta **controller**. Isso indica que a lógica de manipulação das requisições HTTP (como processar dados do banco de dados) está separada em arquivos chamados controllers.
4. **Controllers e Bancos de Dados:**
    - As funções nos controllers são responsáveis por lidar com operações específicas relacionadas aos bancos de dados. Cada função está associada a um endpoint específico e realiza operações como consultar, criar, atualizar ou excluir dados no banco de dados.

Como foi feito o projeto:

Inicialmente, foram feitos todos os *endpoints* no arquivo "*app.js*", porém para melhorar a legibilidade do código e o torná-lo mais limpo, foi criado uma pasta chamada "*routes"*, justamente para armazenar esses *endpoints*.

Após isso, foi criada a pasta "*controler"* para armazenar as funções para os *endpoints*. Lá está o arquivo de:

- **Cliente.js:** com as funções de CRUD de um cliente a partir do ID;

- **Pedidos.js:** com as funções de criação de tabela de pedidos, para o pedido de compra e para ordenar os produtos mais vendidos

- **Produto.js:** com funções para a criação de tabela, criação de um novo produto, edição de um produto existente a partir do ID, para visualizar todos os produtos cadastrados, para visualizar UM produto cadastrado a partir do ID, deletar um produto e exibir produtos de baixo estoque.

Utilizamos NodeJS Express no back-end, para criar a API RESTful para manipular solicitações HTTP e desenvolver o CRUD, além do manuseio de pacotes e dependências.

Para o banco de dados, foi utilizado o SQLite para armazenar e manipular dados em um arquivo local, sem a necessidade de um servidor dedicado.

No arquivo *app.js*, está configurando um servidor de API usando o framework Express em JavaScript, juntamente com algumas operações relacionadas ao banco de dados SQLite. Esse código cria um servidor Express, configura middleware para lidar com JSON, realiza operações relacionadas ao banco de dados SQLite (como a criação de tabelas) e define rotas para o servidor. Por fim, inicia o servidor na porta 3000 e exibe uma mensagem indicando que a API está em execução.

Por fim, ao passar para o *router.js,* definimos as rotas para um servidor Express, utilizando o módulo **Router** do Express, para operações CRUD em entidades como Cliente, Produto e Pedidos. Além disso, há rotas específicas para geração de relatórios estatísticos.

Bibliotecas utilizadas:

- Express: utilizada para criar o servidor Web;

- SQLite3: driver SQL que interage com o banco de dados;

- Cors: mecanismo que permite que uma aplicação web possa acessar recursos de outra origem diferente da sua.

Configuração Inicial:

- O servidor é configurado para escutar na porta 3000.

- E um banco de dados é criado usando o módulo SQLite3

Endpoints:

Cliente (/cliente):

Endpoint GET para recuperar apenas um cliente no banco de dados;

Endpoint PUT para adicionar um novo cliente ao banco de dados;

Endpoint POST para atualizar um cliente existente no banco de dados;

Endpoint DELETE para excluir um cliente do banco de dados.

Clientes (/clientes):

Endpoint GET para recuperar todos os clientes no banco de dados;

Produto(/produto):

Endpoint GET para recuperar apenas um produto no banco de dados;

Endpoint PUT para adicionar um novo produto ao banco de dados;

Endpoint POST para atualizar um produto existente no banco de dados;

Endpoint DELETE para excluir um produto do banco de dados.

Produtos (/produtos):

Endpoint GET para recuperar todos os produtos no banco de dados;

Pedido(/pedido):

Endpoint POST para criar um pedido no banco de dados;

Pedidos(/pedidos):

Endpoint GET para visualizar todos os pedidos cadastrados no banco de dados;

Produtos Baixo Estoque (/produtos-baixa-quantidade):

Endpoint GET para recuperar todos os produtos com o estoque abaixo ou igual a 3 no banco de dados;

Produtos Mais Vendidos (/produtos-mais-vendidos):

Endpoint GET para mostrar os produtos mais vendidos no banco de dados;