Trabalho 02 - Banco de Dados

Introdução

Este trabalho tem como objetivo a criação de uma API para gerenciar um sistema de pedidos. A API será desenvolvida utilizando Node.js e se conectará a um banco de dados **MySQL**, com o gerenciamento de esquemas e dados realizado através da biblioteca **Knex.js**.

Requisitos Técnicos

O trabalho deve ser desenvolvido em um ambiente Node.js, e a comunicação com o banco de dados deve ser feita exclusivamente através do **Knex.js**. A API deve ser capaz de realizar operações de **CRUD (Create, Read, Update, Delete)** em diversas tabelas, bem como de gerenciar relacionamentos entre elas.

Etapas do Trabalho

1. **Configuração Inicial**:
    - Configurar o ambiente de desenvolvimento Node.js.
    - Instalar e configurar as dependências necessárias, incluindo **Knex.js**, o driver do MySQL e o framework da API (como Fastify).
    - Criar o arquivo de configuração do Knex.js (knexfile.js), especificando a conexão com o banco de dados MySQL.
2. **Migrations e Seeds**:
    - Criação das **migrations** para as seguintes tabelas, com atenção especial aos relacionamentos:
        - marcas
        - produtos
        - clientes
        - pedidos
        - itens_pedidos
    - As tabelas deverão conter os campos necessários para armazenar os dados dos arquivos CSV fornecidos (por exemplo, marcas.csv, produtos.csv).
    - Criação dos **seeds** para popular as tabelas. Os dados do seed estão contidos nos arquivos .csv.
    - **Atenção aos Relacionamentos**: Certifique-se de que os relacionamentos entre as tabelas estejam corretamente definidos nas migrations (ex: produtos se relacionando com marcas, pedidos com clientes, e itens_pedidos com pedidos e produtos).

3. **Desenvolvimento das Rotas da API**:
    - Após a configuração inicial, as migrations e os seeds estarem prontos e executados, o código deve ser **commitado** com a mensagem: "feat: Configurações iniciais, migrations e seeds".
    - Em seguida, a API deverá implementar as seguintes rotas, utilizando o Knex.js para todas as interações com o banco de dados:
    - A estrutura de resposta deverá conter além do HTTP CODE(200, 204, 400, 412, 500, etc..) referente a resposta, o seguinte formato

      {

      message: “”,

      data: [Array] ou {Object},

      error: false

      }
    - **Tabela de marcas**:

      GET /marcas: Lista todas as marcas.

      GET /marcas/:id: Lista a marca com o id especificado.

      DELETE /marcas/:id: Exclui a marca com o id especificado.
    - **Tabela de produtos**:

      GET /produtos: Lista todos os produtos.

      GET /produtos/:id: Lista o produto com o id especificado.

      POST /produtos: Cadastra um novo produto.
    - **Tabela de clientes**:

      GET /clientes: Lista todos os clientes.

      GET /clientes/:id: Lista o cliente com o id especificado.

      POST /clientes: Cadastra um novo cliente.
    - **Tabela de pedidos**:

      GET /pedidos: Lista todos os pedidos. Cada pedido deve incluir uma propriedade itens que contenha todos os itens_pedidos relacionados a ele.

      GET /pedidos/:id: Lista o pedido com o id especificado. O pedido deve incluir a propriedade itens com os itens_pedidos relacionados.

      GET /pedidos/:cidade: Lista todos os pedidos da cidade especificada. Cada pedido deve incluir a propriedade itens com os itens_pedidos relacionados.

      POST /pedidos: Gera um novo pedido. Os itens do pedido enviados na requisição devem ser inseridos na tabela itens_pedidos e associados ao novo pedido.

Entrega

- O trabalho deve ser entregue através do classroom com o repositório GIT do seu projeto.
- O nome do aluno ou equipe deverá estar contido no arquivo README.md.

- Na segunda-feira alguns alunos ou equipes serão sorteados para comentar sobre o trabalho desenvolvido.
- O histórico de commits deve refletir o processo de desenvolvimento, incluindo o commit mencionado no passo 3.
- O código deve ser bem estruturado e seguir as boas práticas de programação.

Qualquer dúvida sobre a utilização do Knex.js, a documentação oficial pode ser consultada em: **https://knexjs.org/**.