Hands-o): Construindo uma API de Gerenciamento de Loja

Introdução

Prezados(as) alunos(as),

Esta atividade prática (Hands-on) foi cuidadosamente elaborada para consolidar o conhecimento adquirido em sala de aula sobre o desenvolvimento de APIs robustas e organizadas. A proposta é que vocês apliquem os conceitos de Node.js, TypeScript, Express, TypeORM, validação com Zod e as boas práticas de arquitetura de software.

Vocês terão a liberdade de trabalhar **individualmente, em duplas ou em trios**, fomentando a colaboração e a troca de experiências. O objetivo final é a entrega de um **documento bem estruturado** que detalhe a solução desenvolvida, além do código-fonte da API.

Cenário de Negócio: A Loja "Mundo Geek"

Recentemente, a equipe de desenvolvimento foi procurada pelo Sr. Osvaldo, proprietário da **"Mundo Geek"**, uma loja especializada em produtos colecionáveis, jogos de tabuleiro e itens de cultura pop. A "Mundo Geek" está crescendo rapidamente e o Sr. Osvaldo percebeu que o controle manual de seu estoque e vendas está se tornando insustentável. Ele precisa de um sistema que o ajude a organizar seus produtos por categorias e a gerenciar o inventário de forma mais eficiente.

O Sr. Osvaldo solicitou uma API que permita o gerenciamento de suas **Categorias** de produtos e dos **Produtos** em si. Ele enfatizou a importância de um sistema confiável e fácil de usar, que possa ser expandido no futuro.

Entidades Essenciais

Para atender à demanda da "Mundo Geek", vocês deverão modelar e implementar as seguintes entidades principais, estabelecendo um relacionamento de **Um para Muitos (1:N)** entre elas:

1. **Categoria:** Representa uma categoria de produtos na loja (ex: "Jogos de Tabuleiro", "Action Figures", "Livros").
    - id (UUID ou numérico, chave primária)
    - nome (string, obrigatório, único)
    - descricao (string, opcional)
    - dataCriacao (data e hora, gerado automaticamente)
    - dataAtualizacao (data e hora, atualizado automaticamente)
2. **Produto:** Representa um item disponível na loja.
    - id (UUID ou numérico, chave primária)

- nome (string, obrigatório)
- descricao (string, opcional)
- preco (número decimal, obrigatório, maior que zero)
- estoque (número inteiro, obrigatório, maior ou igual a zero)
- dataCriacao (data e hora, gerado automaticamente)
- dataAtualizacao (data e hora, atualizado automaticamente)
- categoria (relacionamento 1:N com a entidade Categoria, indicando a qual categoria o produto pertence).

Requisitos Técnicos

O projeto deve ser uma API RESTful e seguir as seguintes especificações:

- **Linguagem:** TypeScript
- **Runtime:** Node.js
- **Framework Web:** Express
- **ORM:** TypeORM utilizando um banco de dados PostgreSQL,
- **Validação:** Zod para validação de esquemas de entrada (request body, params, query)
- **Estrutura de Projeto:** O projeto deve ser organizado de forma modular, seguindo a estrutura de pastas indicada abaixo.
- **Boas Práticas:** Aplicação dos princípios divisão de responsabilidade buscando um código limpo, coeso e desacoplado.

Estrutura de Pastas Sugerida

Para manter a organização e facilitar a manutenção, o projeto deve seguir a seguinte estrutura dentro da pasta src:

```
src/
├── server.ts          # Ponto de entrada da aplicação, configuração do Express
├── controllers/       # Lógica de manipulação de requisições e respostas
├── routes/            # Definição das rotas da API
├── services/          # Regras de negócio e orquestração
├── database/          # Configuração da conexão com o banco de dados e TypeORM
├── config/            # Variáveis de ambiente e outras configurações gerais
|--- middlewares/
├── entities/          # Definição das entidades do TypeORM (Categoria, Produto)
├── validates/         # Esquemas de validação com Zod
```

```
├── models/             # (Opcional) Interfaces ou classes para modelos de dados que não são entidades TypeORM
```

Orientações e Dicas

3. **Início do Projeto:**
   - Crie uma nova pasta para o seu projeto.
   - Inicialize o projeto Node.js: npm init -y
   - Instale o TypeScript: npm install typescript ts-node @types/node --save-dev
   - Crie o arquivo tsconfig.json através do comando: npx tsc --init
4. **Instalação de Dependências Essenciais:**
   - Express: npm install express @types/express
   - TypeORM e driver do banco de dados (ex: PostgreSQL): npm install typeorm pg @types/pg
   - Zod: npm install zod
   - nodemon e tsx: npm i nodemon tsx –save-dev
   - Dotenv (para variáveis de ambiente): npm install dotenv
   - No arquivo package.json adicionem o script : “dev”: “`nodemon --watch src/**/*.ts --exec tsx src/server.ts`
     ”
5. **Base do Projeto:** Utilizem como ponto de partida o projeto base desenvolvido em aula, adaptando-o para as entidades e funcionalidades desta atividade.
6. **Implementação CRUD:** Para cada entidade (Categoria e Produto), implementem as operações básicas de CRUD (Create, Read, Update, Delete) através dos seus respectivos controllers, services e routes.
7. **Validação com Zod:** Garanta que todas as entradas da API (criação e atualização de categorias/produtos) sejam validadas utilizando esquemas Zod. Por exemplo, o preco de um produto deve ser um número e maior que zero, e o nome da categoria deve ser único.
8. **Relacionamento 1:N:** Implementem o relacionamento onde uma Categoria pode ter muitos Produtos, mas um Produto pertence a apenas uma Categoria. Pesquise em como o TypeORM gerencia essa associação (chaves estrangeiras).
9. **Tratamento de Erros:** Implementem um tratamento de erros adequado para a API, retornando mensagens claras e códigos de status HTTP apropriados

OBS: UTILIZEM COMO BASE O PROJETO CRIADO DURANTE O CURSO.

Entrega

Ao final da atividade, vocês deverão entregar:

10. **Código-fonte:** Um repositório Git (GitHub, GitLab, etc.) contendo todo o código da API.
11. **Documento de Solução:** Um arquivo Markdown (.md) ou PDF (.pdf) detalhando:
    - A arquitetura da solução (diagrama simples, se possível).
    - As decisões de design tomadas (por que escolheram certas abordagens).
    - Como os princípios de divisão de responsabilidades foram aplicados.
    - Instruções para configurar e rodar a API localmente.

Bom trabalho a todos!