Teste para Programador Backend Sênior

Objetivo:
Desenvolver um CRUD (Create, Read, Update, Delete) completo para uma entidade de sua escolha (por exemplo, Produto, Usuário, Pedido, etc.) utilizando o framework NestJS, demonstrando domínio de boas práticas de arquitetura, testes, segurança e documentação.

Requisitos Obrigatórios

1. API RESTful com NestJS

- Implemente endpoints para:
    - Cadastrar (Create) a entidade.
    - Listar todas as entidades (Read - listagem).
    - Consultar uma entidade específica por ID (Read - detalhe).
    - Atualizar uma entidade específica por ID (Update).
    - Excluir uma entidade específica por ID (Delete).
- Utilize o padrão Service + Repository do NestJS.
- Estruture o projeto de forma modular (usar módulos, services, controllers, DTOs, schemas/entidades separados).

2. Validação e Tratamento de Erros

- Garanta que todos os campos obrigatórios sejam validados com decorators do NestJS e class-validator.
- Implemente tratamento adequado de erros (HTTPException, filtros globais, etc.).
- Faça logging dos erros e principais eventos da aplicação (preferencialmente usando o sistema de logger do NestJS).

3. Documentação

- Utilize o @nestjs/swagger para documentar todos os endpoints, exemplos de payloads e respostas.

- Inclua instruções de uso da API no README.

4. Testes Automatizados

- Implemente testes automatizados de unidade (unit) e de integração (integration) usando Jest.

- Cubra ao menos os principais fluxos do CRUD e os casos de erro.

5. Autenticação e Autorização

- Implemente autenticação JWT protegendo os endpoints do CRUD.

- Implemente autorização básica (ex: só usuários autenticados podem acessar ou modificar dados).

6. Persistência de Dados

- Use banco de dados **não relacional** (preferencialmente MongoDB).

- Utilize o @nestjs/mongoose para abstração dos modelos e acesso ao banco.

- Justifique a escolha do banco de dados não relacional no README.

7. Containerização

- Prepare a aplicação para ser executada em ambiente Docker (inclua Dockerfile e docker-compose.yml para facilitar o setup com MongoDB).

- README deve conter instruções para executar localmente e via Docker.

Diferenciais (Desejáveis)

- **Padrões Avançados:**
  Uso de patterns como CQRS, Event Sourcing, Repository Pattern, Service Pattern, etc.

- **Clean Architecture:**
  Separação clara entre camadas de domínio, aplicação e infraestrutura.

- **Observabilidade:**
  Implementação de métricas básicas (prometheus, health check, readiness/liveness, tracing).

- **Rate Limiting:**
  Limitar requisições para evitar abusos na API.

- **Integração com CI/CD:**
  Scripts de integração contínua (Github Actions, Gitlab CI, etc.).

- **Frontend (opcional):**
  Interface simples (SPA) para consumir e demonstrar a API. Pode usar React, Angular, Vue, etc.

- **Deploy na Nuvem (opcional):**
  Demonstre deploy em algum ambiente cloud (Heroku, AWS, GCP, etc.).

Instruções de Entrega

- Suba o projeto em um repositório público (GitHub, GitLab ou Bitbucket).

- O README deve conter:

    - Descrição técnica do projeto.

    - Justificativa das principais escolhas de arquitetura, patterns e tecnologias.

    - Passo a passo para rodar o projeto localmente e via Docker.

    - Instruções para rodar os testes.

    - (Opcional) Demonstração do frontend e/ou link para o deploy na nuvem.

- Inclua exemplos de requisições (curl ou Postman Collection).

Critérios de Avaliação

- Organização, clareza e manutenibilidade do código.
- Domínio dos conceitos e boas práticas do NestJS.
- Arquitetura e padrões de projeto aplicados.
- Segurança, observabilidade e robustez da solução.
- Cobertura de testes e qualidade dos mesmos.
- Documentação, justificativas e facilidade de uso/execução.
- Atenção a detalhes e diferenciais implementados.

!