DESAFIO TÉCNICO - DESENVOLVEDOR(A) FRONTEND (NEXT.JS) / FULLSTACK

OBJETIVO

Construir um blog completo utilizando Next.js no frontend e uma API backend na linguagem de sua preferência. A aplicação deve conter quatro páginas principais:

- Página Inicial: lista de notícias publicadas com imagem de destaque.
- Página de Detalhes: visualização completa da notícia e sua imagem.
- Página de Criação: formulário para cadastrar uma nova notícia (título, resumo, corpo e imagem).
- Página de Edição: formulário para editar uma notícia existente.

REGRAS PARA O FRONTEND

- Framework obrigatório: Next.js (App Router ou Pages Router).
- Estilo obrigatório: Tailwind CSS.
- Layout responsivo utilizando classes utilitárias do Tailwind.
- Boas práticas de acessibilidade (atributo alt nas imagens, contraste adequado, foco por teclado).

REGRAS PARA O BACKEND

- Pode ser implementado em Node.js/Express, Python/Django ou FastAPI, Ruby on Rails, Go etc.
- Endpoints mínimos (REST ou GraphQL equivalente):
  - GET /news -> lista de notícias
  - GET /news/{id} -> detalhes de uma notícia
  - POST /news -> cria nova notícia
  - PUT /news/{id} -> edita notícia
- Banco de dados livre (PostgreSQL, SQLite, MongoDB...). Os dados devem ser persistidos.
- Upload/armazenamento da imagem de destaque (disco local ou serviço de storage).

FUNCIONALIDADES OBRIGATÓRIAS

- Listagem na homepage (título, resumo e imagem).
- Página de detalhes com conteúdo completo e imagem.
- Página de criação de notícia.
- Página de edição de notícia:
  * Caso NÃO haja autenticação, deve estar acessível a qualquer usuário.
  * Caso haja autenticação, apenas o autor da notícia pode editar seu próprio conteúdo.

AUTENTICAÇÃO (EXTRA)

- Se implementar autenticação:
  - Criar página de login.
  - Somente usuários autenticados podem cadastrar notícias.
  - Uma notícia só pode ser editada pelo usuário que a criou.

AVALIAÇÃO

Serão avaliados:

- UX / UI e uso consistente do Tailwind.
- Qualidade do código: organização, clareza e boas práticas em front e back.
- API completa e tratamento de erros.
- Implementação correta de todas as funcionalidades.

PONTOS EXTRAS

- Autenticação com JWT ou sessão.
- Testes automatizados (frontend e/ou backend).
- Deploy (ex.: Vercel para frontend e AWS para backend).
- Documentação de API (Swagger ou Postman).

ENTREGA

1. Repositório GitHub com pastas frontend/ e backend/.
2. README contendo:
   - Passos para rodar frontend e backend localmente.
   - Migrações ou seed de dados.
   - Decisões técnicas principais.
   - (Opcional) link de deploy e documentação da API.