Teste Técnico — Fullstack (React / Next.js / .NET / IA)

Objetivo

Avaliar a capacidade do(a) candidato(a) em desenvolver uma aplicação fullstack moderna, utilizando React.js + Next.js, API em C# .NET e integração com IA, aplicando boas práticas de desenvolvimento, organização de código e arquitetura.

Escopo Geral

O teste consiste na criação de uma aplicação com três páginas distintas, cada uma explorando um modelo diferente de renderização do Next.js:

1. SSR (Server-Side Rendering)

2. SSG (Static Site Generation)

3. CSR (Client-Side Rendering) com integração de IA (NL2SQL)

Além disso, será necessário criar um backend em C# .NET com um CRUD que será consumido pela página SSR do frontend.

Requisitos Técnicos

Frontend

- React + Next.js

- TypeScript

- Uso explícito de SSR, SSG e CSR

- Organização clara de componentes e páginas

Backend

- API REST em C# .NET

- CRUD completo (Create, Read, Update, Delete)

- Persistência em banco de dados (livre escolha)

- A API será consumida pela página SSR do frontend

Páginas Obrigatórias

1. Página SSR — Dados Dinâmicos

- Renderizada via Server-Side Rendering

- Consome dados do CRUD da API em C# .NET

- Domínio livre (ex: produtos, usuários, pedidos, cursos)

- Tratamento de loading, erro e lista vazia

2. Página SSG — Conteúdo Estático

- Página 100% estática

- Gerada via Static Site Generation

- Não consome API

- Conteúdo livre (Sobre, FAQ, Documentação, Arquitetura)

3. Página CSR — IA com Agente NL2SQL

- Página renderizada via Client-Side Rendering (CSR)

- Interface onde o usuário pode realizar perguntas em linguagem natural sobre os dados disponíveis

Funcionalidade esperada:

1. Receber a pergunta em linguagem natural fornecida pelo usuário

2. Utilizar um modelo de IA (LLM) para interpretar a pergunta e gerar uma consulta SQL correspondente

3. Executar a consulta SQL gerada no banco de dados

4. Retornar os dados obtidos para exibição na interface do usuário

Observação sobre arquitetura:

A arquitetura utilizada para integrar a aplicação com o modelo de IA é livre.
O(a) candidato(a) deve decidir onde essa integração ocorre e justificar suas escolhas durante a apresentação da solução.

Documentação

O projeto deve conter um README explicando:

- Como rodar o frontend e o backend

- Arquitetura da solução

- Decisões e limitações técnicas

Diferenciais (não obrigatórios)

- Docker

- Testes automatizados

- Melhorias de UX

Avaliação

Serão considerados:

- Organização do código

- Clareza arquitetural

- Uso correto de SSR, SSG e CSR

- Qualidade da integração com IA

- Boas práticas de segurança e engenharia de software

Considerações Finais

O foco do teste não é complexidade, mas clareza, boas decisões e domínio técnico. Soluções simples, bem justificadas e organizadas são preferíveis a soluções excessivamente complexas.