Teste Técnico – Desenvolvedor(a) Fullstack

Introdução

Olá! Agradecemos seu interesse em fazer parte da nossa equipe. Este teste foi projetado para simular um desafio real e nos ajudar a entender suas habilidades em arquitetura de software, desenvolvimento fullstack com **Node.js/Next.js/React** e sua atenção a boas práticas como componentização e testes.

O Desafio: Plataforma de Gestão para Grupos de Networking

Um cliente, que lidera um grupo de networking focado em geração de negócios, precisa de uma plataforma para digitalizar e otimizar a gestão dos membros e suas interações. O objetivo é substituir planilhas e controles manuais por um sistema centralizado e eficiente.

Sua tarefa é **projetar a arquitetura completa** para esta plataforma e, em seguida, **implementar uma parte crucial** dela para demonstrar suas habilidades práticas.

Prazo Sugerido: 3 a 5 dias.

Tarefa 1: Desenho da Arquitetura (40% da Avaliação)

Antes de escrever qualquer código, você deve criar um **Documento de Arquitetura** (em formato Markdown, dentro do próprio repositório) que detalhe sua visão para o sistema completo. Este documento é fundamental para avaliarmos seu pensamento estratégico e sua capacidade de planejar soluções escaláveis.

O documento deve cobrir **todas as funcionalidades listadas abaixo**:

Funcionalidades a Considerar na Arquitetura:

1. **Gestão de Membros:**
   - Formulário público de intenção de participação.
   - Área de gestão para administradores aprovarem/recusarem intenções.
   - Formulário de cadastro completo para membros aceitos.
2. **Comunicação e Engajamento:**
   - Área de avisos e comunicados para os membros.

- Controle de presença em reuniões (check-in).

3. **Geração de Negócios:**
   - Sistema de indicações e referências de negócios entre membros.
   - Avaliação e acompanhamento do status das indicações (ex: `Enviada`, `Em Negociação`, `Fechada`).
   - Registro de "obrigados" (agradecimentos públicos por negócios fechados).
4. **Acompanhamento e Performance:**
   - Controle de reuniões 1 a 1 entre membros.
   - Dashboards de desempenho individual e do grupo.
   - Relatórios por período (semanal, mensal, acumulado).
5. **Financeiro:**
   - Módulo de controle de mensalidades (geração, status de pagamento).

Entregáveis do Documento de Arquitetura:

1. **Diagrama da Arquitetura:** Um diagrama simples (pode ser em ASCII, Mermaid ou uma imagem) mostrando os principais componentes da solução (ex: Frontend, Backend API, Banco de Dados, etc.) e como eles se comunicam.
2. **Modelo de Dados:** O esquema do banco de dados (SQL ou NoSQL) que suportaria todas as funcionalidades. Detalhe as tabelas/coleções, campos e relacionamentos. Justifique a escolha do banco de dados.
3. **Estrutura de Componentes (Frontend):** Uma breve descrição de como você organizaria os componentes React no Next.js, pensando em reutilização, estado global (se necessário) e pastas (ex: `components/ui`, `components/features`, `containers`).
4. **Definição da API:** A especificação dos principais endpoints da API (REST ou GraphQL). Descreva as rotas, métodos HTTP, e os schemas de request/response para pelo menos 3 funcionalidades (ex: `POST /applications`, `GET /admin/applications`, `POST /referrals`).

Tarefa 2: Implementação Prática (60% da Avaliação)

Com a arquitetura planejada, é hora de codificar. Você deverá implementar os módulos descritos abaixo, aplicando as melhores práticas de desenvolvimento.

Stack Técnica Obrigatória:

- **Frontend:** Next.js e React

- **Backend:** Node.js (com qualquer framework, como Express, NestJS, ou os próprios API Routes do Next.js)
- **Banco de Dados:** SQLite, PostgreSQL ou MongoDB (à sua escolha).
- **Testes:** Jest e React Testing Library (ou equivalentes).

Módulo Obrigatório: Fluxo de Admissão de Membros

Esta é a porta de entrada do sistema e deve ser totalmente funcional.

1. **Página de Intenção:** Crie uma página pública com um formulário simples contendo campos como `Nome`, `Email`, `Empresa` e `Por que você quer participar?`.
2. **Área do Administrador:** Crie uma área privada (pode ser protegida por uma variável de ambiente, sem a necessidade de um sistema de login completo) onde um administrador possa:
   - Ver a lista de todas as intenções submetidas.
   - Aprovar ou Recusar cada intenção.
3. **Cadastro Completo:** Ao aprovar uma intenção, o sistema deve gerar um "convite" (pode ser um registro no banco de dados com um token único). Crie uma segunda página de cadastro, mais completa, que só possa ser acessada com esse token válido. A implementação do envio de e-mail pode ser simulada (um `console.log` do link gerado é suficiente).

Módulo Opcional (Escolha um para implementar)

Escolha **uma** das duas opções abaixo para demonstrar sua habilidade em outras áreas.

- **Opção A - Sistema de Indicações:**
  - Implemente a funcionalidade para um membro (logado) criar uma indicação de negócio para outro membro.
  - O formulário deve conter campos como `Membro Indicado`, `Empresa/Contato Indicado`, `Descrição da Oportunidade`.
  - Crie uma página onde um membro possa ver as indicações que fez e as que recebeu, e atualizar o status de cada uma (`Nova`, `Em Contato`, `Fechada`, `Recusada`).
- **Opção B - Dashboard de Performance:**
  - Crie uma página privada que exiba um dashboard simples com os seguintes indicadores (os dados podem ser "mockados" ou vir do banco de dados se você implementou outras partes):
    - Número total de membros ativos.

- Total de indicações feitas no mês.
- Total de "obrigados" registrados no mês.

Critérios de Avaliação

- **Componentização e Qualidade de Código (30%):** Código limpo, bem estruturado e legível. Componentes React reutilizáveis e bem definidos. Lógica de backend clara e organizada.
- **Testes (30%):** Cobertura e relevância dos testes unitários e de integração. Queremos ver que você sabe o que testar e como testar de forma eficaz.
- **Integração Fullstack (25%):** A comunicação entre o frontend e o backend deve ser eficiente e segura. Bom uso do estado no frontend e manipulação correta dos dados.
- **Boas Práticas Gerais (15%):** Uso correto do Git (histórico de commits claro), um `README.md` bem escrito explicando como rodar o projeto, e uso de variáveis de ambiente para configurações sensíveis.

Entregáveis

- O link de um **repositório Git público** (GitHub, GitLab, etc.) contendo:
  1. Todo o código-fonte do projeto.
  2. O `README.md` com as instruções de instalação e execução.
  3. O documento `ARQUITETURA.md` que você criou na Tarefa 1.

Boa sorte!