Desafio Técnico

Módulo de Clientes - CRM Corporativo

Cenário de Negócio

A empresa está iniciando a construção de um novo CRM para gerenciar sua base global de clientes. Este módulo é crítico e servirá de base para futuras integrações (faturamento, suporte e marketing). Esperamos uma solução que não apenas funcione, mas que seja resiliente, auditável e fácil de manter.

Premissas Tecnológicas

- **Back-End:** .NET 8.0 ou superior.
- **Front-End:** React (com TypeScript preferencialmente).
- **Banco de Dados:** Relacional (PostgreSQL) ou NoSQL, justificando a escolha.
- **Arquitetura:** Domain-Driven Design (DDD) com separação clara de responsabilidades.
- **Padrões:** CQRS e Event Sourcing para garantir o histórico imutável de alterações do cliente.
- **Validações:** FluentValidation.

Requisitos de Mínimos

O candidato deve implementar o cadastro e gestão de clientes com os seguintes campos e regras:

Dados Cadastrais:

- Nome/Razão Social, CPF/CNPJ, Data de Nascimento/Fundação, Telefone e E-mail.
- **Endereço:** CEP, logradouro, número, bairro, cidade e estado.

Regras de Negócios:

1. **Unicidade:** Bloquear cadastros duplicados por CPF/CNPJ ou E-mail.
2. **Compliance:** Pessoa Física deve ter idade mínima de 18 anos.
3. **Tributação:** Pessoa Jurídica deve obrigatoriamente informar a Inscrição Estadual (IE) ou marcar explicitamente a condição de “Isento”.
4. **Consistência de Endereço:** Recomenda-se integração com API externa (ex: ViaCEP) para autocompletar e validar o endereço a partir do CEP.

Critérios de Avaliação

Avaliaremos, além do básico, outros critérios como:

- **Auditabilidade:** O Event Sourcing deve permitir visualizar quem alterou qual dado e quando (log de eventos).
- **Resiliência:** Tratamento de erros globais, uso de *Retry Policies* em chamadas externas e logs estruturados.
- **Testes:** Cobertura de testes unitários de domínio e, ao menos, um teste de integração do fluxo completo (Command -> Handler -> Repository).
- **Documentação (ADRs):** Além de explicar o "como rodar", documente o "porquê" das decisões arquiteturais tomadas (ex: por que escolheu tal banco? Como estruturou o Event Store?).

Entrega e Prazos

- **Containerização:** Disponibilizar docker-compose que suba a aplicação/API, o Banco de Dados e o Front-End em um único comando.
- **Repositório:** Disponibilizar código no GitHub com histórico de commits organizado e nos encaminhar o link.
- **Prazo:** 6 dias corridos.

Este desafio foi desenhado para simular dilemas reais do nosso dia a dia. Mais do que a perfeição técnica, buscamos entender sua linha de raciocínio, sua capacidade de tomar decisões e como você lida com a evolução constante da tecnologia. Independentemente do resultado ou do nível de senioridade, encare cada linha de código como uma oportunidade de crescimento.

Boa sorte, faça o seu melhor e lembre-se: a vida, assim como a tecnologia, é uma eterna evolução.