GRUPO DIRECIONAL

Teste para Analista/Desenvolvedor Sênior

Objetivo

O candidato deverá demonstrar habilidades na criação de uma estrutura de banco de dados e no desenvolvimento de uma API para interagir com as tabelas.

Requisitos

1. **Criação do banco de dados**
   - Criar um banco de dados utilizando SQL Server Express.
   - Criar 4 tabelas:
     - **clientes**: Tabela para armazenar informações dos clientes.
     - **apartamentos**: Tabela para armazenar informações dos apartamentos.
     - **vendas**: Tabela para armazenar informações sobre vendas de apartamentos para os clientes.
     - **reservas**: Tabela para armazenar informações sobre reservas de apartamentos para os clientes.
2. **Criação da API**
   - Desenvolver uma API utilizando .NET 9.
   - Criar endpoints para interagir com as tabelas do banco de dados:
     - Criar, listar, atualizar e excluir clientes.
     - Criar, listar, atualizar e excluir apartamentos.
     - Criar, listar, atualizar e excluir vendas.
   - Utilizar Entity Framework como ORM.
   - Implementar autenticação e autorização utilizando JWT (JSON Web Token).
     - Apenas usuários autenticados devem ter acesso aos endpoints.
     - Criar um endpoint de login para gerar o token JWT.
3. **Entrega de um Docker Compose**
   - Criar um docker-compose.yml para subir toda a estrutura do projeto.
   - O Compose deve iniciar:
     - O banco de dados SQL Server Express.
     - O serviço da API.
4. **Documentação**
   - Criar um arquivo README.md com as seguintes informações:

        - Como rodar o projeto localmente utilizando Docker.
        - Exemplos de requisições para a API.
        - Estrutura das tabelas no banco de dados.
        - Como gerar e utilizar o token JWT para autenticação.
        - Considerações e decisões técnicas tomadas durante o desenvolvimento.
5. **Front-end (Opcional)**
    - O desenvolvimento do front-end não é obrigatório, mas será considerado um bônus na avaliação final.
    - Caso o candidato opte por criar um front-end, deve escolher entre **React** ou **Next.js**.
    - O front-end deve consumir a API criada e apresentar uma interface funcional para cadastro, visualização e reserva de apartamentos.
6. **Testes Unitários (Opcional - Bônus na Avaliação)**
    - A criação de testes unitários não é obrigatória, mas será considerada um bônus na avaliação final.
    - Caso o candidato opte por criar testes, deve utilizar frameworks adequados como **xUnit** ou **NUnit**.
    - Os testes devem cobrir os principais fluxos da API, como autenticação, criação e manipulação de clientes, apartamentos e reservas.

Cenário de Uso

Um cliente chega em um stand de vendas e é atendido por um corretor de imóveis. O corretor apresenta diversos apartamentos disponíveis e suas respectivas características. Após conhecer os detalhes, o cliente se interessa por um dos apartamentos e decide comprá-lo imediatamente. Para registrar essa transação, o corretor realiza as seguintes ações no sistema:

1. **Autenticação:**
    - O corretor realiza login.
2. **Cadastro do cliente:**
    - O corretor cadastra os dados do cliente.
3. **Verificação da disponibilidade do apartamento:**
    - A API é consultada para verificar se o apartamento ainda está disponível.
4. **Venda do apartamento:**
    - O corretor cria uma reserva para garantir que o apartamento não seja vendido para outra pessoa.
5. **Finalização da compra:**
    - O cliente realiza o pagamento da entrada e a reserva é confirmada como uma venda definitiva no sistema.
    - A API atualiza o status do apartamento como "Vendido" e associa a compra ao cliente.

Esse cenário exemplifica como a API pode ser utilizada no processo de compra de um imóvel, garantindo o controle de clientes, apartamentos e transações em tempo real.

Critérios de Avaliação

- Estrutura e organização do código.
- Qualidade da implementação da API.
- Uso correto do Docker e Docker Compose.
- Implementação segura da API.
- Clareza e completude da documentação.
- Boas práticas de desenvolvimento.
- **(Bônus)** Se houver um front-end, será avaliado quanto à sua usabilidade, design e integração com a API.
- **(Bônus)** Se houver implementação de testes unitários cobrindo os principais fluxos da API.

Como Entregar

- O código deve ser disponibilizado em um repositório público no GitHub.
- O link do repositório deve ser enviado para o e-mail **arquitetura.sistemas@direcional.com.br**
- **Assunto:** Testes (Nome do candidato) – (Vaga)