Teste Técnico Fullstack .NET - AUVO Tecnologia

Desenvolvedor Fullstack utilizando .NET e AWS Cloud

Orientações Gerais

O objetivo deste teste técnico é avaliar a capacidade do candidato em desenvolver uma aplicação fullstack usando o ecossistema .NET e a AWS Cloud, integrando práticas modernas de desenvolvimento, como microserviços, segurança com JWT, cloud deployment, arquitetura de microsserviços ou monolítica, e design responsivo para o front-end. A aplicação deverá ser desenvolvida de modo a simular um cenário real de negócios, integrando uma API pública gratuita para fornecer dados dinâmicos ao usuário final.

Case de Aplicação: Sistema de Clima e Países Favoritos

Você está sendo contratado por uma empresa fictícia chamada "GloboClima", que quer desenvolver uma aplicação que permita aos usuários consultar informações climáticas e dados de países (como população, idiomas, moedas, etc.) e salvar suas cidades e países favoritos para futuras consultas. O sistema será composto por um backend REST API e uma interface web para exibir e interagir com os dados. A aplicação precisa ser hospedada na AWS, utilizando AWS Lambda ou EC2/ECS para o backend, e um front-end responsivo.

Funcionalidades Principais

Backend:

1. Consumo de APIs Públicas:
   - A aplicação deve consultar informações de clima usando a OpenWeatherMap API e dados de países usando a REST Countries API.

2. Gerenciamento de Favoritos:
   - A aplicação deve permitir que os usuários autenticados salvem cidades e países como favoritos. Esses dados devem ser armazenados em DynamoDB.

3. Autenticação e Segurança:

- Implementar JWT para autenticação. Somente usuários autenticados poderão salvar, listar ou deletar cidades e países favoritos.

4. API REST:
- Criar endpoints RESTful que permitem:
  - Consultar o clima de uma cidade e informações de um país via API pública.
  - Salvar, listar e deletar cidades e países favoritos.

5. Documentação com Swagger:
- Toda a API deve ser documentada com Swagger, incluindo descrições detalhadas das rotas e exemplos de uso.

Frontend:

1. Interface Responsiva:
- O front-end deve ser implementado em Blazor ou ASP.NET MVC, e exibir os dados climáticos e de países consumidos pela API.

2. Gerenciamento de Favoritos:
- A interface deve permitir que os usuários salvem, visualizem e removam cidades e países favoritos, além de acessar essas informações posteriormente.

3. Autenticação:
- O usuário deve se autenticar usando JWT para acessar funcionalidades protegidas (como salvar favoritos).

4. UI Responsiva:
- A aplicação web deve ser totalmente responsiva e agradável de usar, tanto em dispositivos móveis quanto em desktop.

Requisitos Técnicos

1. Backend - API e Microserviços (Lambda ou EC2/ECS)

- Desenvolver uma API RESTful usando .NET Core que consuma dados das APIs públicas mencionadas.
- Implementar autenticação com JWT para proteger as rotas de gerenciamento de favoritos.
- A API deve ser implementada usando AWS Lambda ou hospedada em EC2/ECS.
- Documentar a API com Swagger para que seja fácil de entender e testar.

2. Frontend - Aplicação Web (Blazor ou ASP.NET MVC)

- O front-end deve ser desenvolvido em Blazor ou ASP.NET MVC.
- A interface deve consumir a API REST, exibir informações de clima e países, e permitir o gerenciamento de favoritos.
- A interface deve ser totalmente responsiva e atender aos padrões de design web moderno.

3. Integração com AWS

- O backend deve ser hospedado na AWS utilizando serviços como Lambda (para funções serverless) ou EC2/ECS (para uma abordagem mais tradicional).
- Implementar deploy contínuo com AWS CodePipeline ou GitHub Actions para automatizar o deploy das mudanças no backend e front-end.
- Configurar CloudWatch para monitoramento de logs e desempenho.

4. Armazenamento de Dados

- Armazenar as cidades e países favoritos dos usuários em um banco DynamoDB.
- Implementar consultas e operações CRUD com Entity Framework Core ou ADO.NET.

5. Segurança e Autenticação

- Utilizar JWT para autenticação de usuários.
- Implementar HTTPS para garantir a segurança no ambiente de produção.

6. Testes Unitários e Integração

- Implementar testes unitários com xUnit ou NUnit para cobrir no mínimo 50% do backend.
- Criar testes de integração para verificar o funcionamento correto das rotas da API e da persistência no DynamoDB.

7. CI/CD e Automação

- Configurar um pipeline de CI/CD com AWS CodePipeline ou GitHub Actions.
- Usar Infraestrutura como Código (IaC) com CloudFormation ou Terraform para

provisionar a infraestrutura AWS necessária.

8. Microservices e Containerização

- O backend pode ser containerizado usando Docker para facilitar a implantação em ECS ou EKS.
- Usar ECS/EKS para gerenciar containers, se optar por não usar AWS Lambda.

Critérios de Avaliação

1. Funcionalidade Completa: Todas as funcionalidades propostas devem ser implementadas, como consumo de API pública, autenticação JWT, e CRUD de favoritos.
2. Qualidade do Código: O código deve seguir boas práticas de desenvolvimento (modularidade, legibilidade, padrões de design).
3. Documentação: A API deve ser bem documentada com Swagger, detalhando cada rota e seus parâmetros.
4. Segurança: A autenticação JWT deve ser implementada corretamente e as rotas sensíveis devem estar protegidas.
5. Desempenho e Otimização: O candidato deve demonstrar preocupação com o desempenho da aplicação, utilizando boas práticas como cacheamento de dados e otimização assíncrona (async/await).
6. Automação e DevOps: O pipeline de CI/CD deve ser corretamente configurado para automatizar o deploy contínuo do projeto.