📌 Descrição do Case

Desenvolver um sistema simples de **gestão de pedidos** (Order Management System), onde será possível criar, listar e visualizar pedidos. Cada pedido terá um **status**, e toda vez que um pedido for criado, uma **mensagem** será enviada para o **Azure Service Bus**, simulando um processamento assíncrono.

🛠 Tecnologias Obrigatórias

- **Backend:** C# (.NET 7 ou superior) + Entity Framework + PostgreSQL
- **Frontend:** React + TailwindCSS
- **Mensageria:** Azure Service Bus
- **Infra:** Docker (opcional)

🚀 Requisitos

1️⃣ Backend (API em C#)

Criar uma **API REST** com os seguintes endpoints:

- POST /orders → Criar um pedido
- GET /orders → Listar todos os pedidos
- GET /orders/{id} → Obter detalhes de um pedido

Regras de Negócio:

- Cada pedido terá os seguintes atributos:
    - Id (Guid)
    - Cliente (string)
    - Produto (string)

    - Valor (decimal)
    - Status (string) → Pode ser: Pendente, Processando, Finalizado
    - DataCriacao (DateTime)
- Ao **criar um pedido**, o backend deve:
    - Persistir os dados no PostgreSQL usando Entity Framework
    - Enviar um evento para o Azure Service Bus com o pedido recém-criado
- Criar um **worker (background service)** em C# que irá consumir as mensagens do Azure Service Bus e **atualizar o status do pedido** para Processando e, após 5 segundos, alterar para Finalizado.

2️⃣ Frontend (React + Tailwind)

Criar um sistema em React que permita:

- Listar pedidos em uma **tabela responsiva**
- Criar um novo pedido com um **formulário**
- Visualizar detalhes do pedido ao clicar nele

3️⃣ Extras (Diferencial)

- Criar um **Docker Compose** para facilitar a inicialização do ambiente
- Implementar um **CRUD completo** no frontend
- Exibir **notificações em tempo real** quando um pedido mudar de status (usando SignalR ou WebSockets)

🔥 **Entrega**

- O candidato deve fornecer:
    - Link para o repositório no GitHub/GitLab
    - Instruções claras de como rodar o projeto (README.md)
    - Código limpo e bem estruturado (seguir boas práticas)

🎯 Critérios de Avaliação

1. **Qualidade do Código:** Estrutura, organização, uso de boas práticas (SOLID, Clean Code)
2. **Domínio das Tecnologias:** Uso correto de C#, Entity Framework, PostgreSQL, Azure Service Bus, React e Tailwind
3. **Funcionalidade:** A aplicação deve atender aos requisitos básicos
4. **Documentação:** README.md explicando setup, execução e detalhes técnicos