



## 🚩 Objetivo do Desafio

Desenvolver um sistema simples de gestão de pedidos, onde será possível criar, listar e visualizar pedidos. Sempre que um pedido for criado, o sistema deve enviar uma mensagem para o Azure Service Bus, simulando um processamento assíncrono. Um worker será responsável por consumir as mensagens, processar o pedido e atualizar seu status.

## 🔗 Tecnologias Obrigatórias

- Backend: C# (.NET 7 ou superior) + Entity Framework + PostgreSQL
- Frontend: React + TailwindCSS
- Mensageria: Azure Service Bus
- Infraestrutura: Docker / Docker Compose (preferencial)

## 🔗 Requisitos do Sistema

### 1🔗 Backend (API em C#)

Criar uma API REST com os seguintes endpoints:

- POST /orders → Cria um novo pedido
- GET /orders → Lista todos os pedidos
- GET /orders/{id} → Obtém detalhes de um pedido

Regras de Negócio:

- Cada pedido deve conter os atributos: id, cliente, produto, valor, status, data\_criacao.
- Status: Pendente, processando ou finalizado.
- Ao criar um pedido, persistir os dados no PostgreSQL usando EF e publicar uma mensagem no Azure Service Bus.
- Criar um worker que consome mensagens, atualiza para 'Processando' e, após 5 segundos, altera para 'Finalizado'.

Regras adicionais:

- Sequência de status obrigatória: Pendente → Processando → Finalizado.
- Consumidor do Service Bus deve ser idempotente.
- Incluir CorrelationId=OrderId e EventType=OrderCreated.
- Implementar health checks para API, banco e fila.

### 2🔗 Frontend (React + TailwindCSS)

Criar uma interface para:

- Listar pedidos em tabela responsiva.
- Criar novos pedidos via formulário.
- Visualizar detalhes do pedido.
- Exibir feedback visual para mudanças de status.

### 3🔗 Infraestrutura

- Criar Docker Compose com API, Worker, Frontend, PostgreSQL e PgAdmin.
- Usar .env para variáveis sensíveis.
- Configurar migrations automáticas.
- Implementar healthchecks no Compose.

#### 4 Módulo Opcional – Pergunte sobre os Pedidos (IA/Analytics)

Este módulo é opcional, mas vale pontos extras.

Permite que os usuários façam perguntas em linguagem natural sobre os pedidos, usando uma LLM (OpenAI, Azure OpenAI, Gemini, etc.).

A LLM deve interpretar a pergunta, consultar o banco e responder de forma amigável, usando dados reais.

Exemplos de perguntas:

- Quantos pedidos temos hoje?
- Qual o tempo médio para aprovar os pedidos?
- Quantos pedidos estão pendentes?
- Qual o valor total de pedidos finalizados este mês?

#### 5 Diferenciais Técnicos (Bônus)

- Outbox Pattern (mensageria transacional) [+3]
- Histórico de status do pedido [+3]
- SignalR/WebSockets com fallback [+3]
- Testcontainers [+1]
- Tracing ponta-a-ponta [+2]
- Golden Tests [+2]
- Módulo IA/Analytics com LLM [+5]

#### 6 Critérios de Avaliação

- Qualidade do Código: 30%
- Mensageria & Confiabilidade: 20%
- Funcionalidade: 15%
- Documentação & DX: 15%
- Frontend & UX: 10%
- Testes Automatizados: 10%

#### 7 Entrega

O candidato deve fornecer um repositório GitHub:

- Código-fonte completo
- README.md com instruções claras para rodar o projeto
- Exemplo de .env
- Diagramas simples de arquitetura (diferencial)