Documentação – Teste Técnico - AmbevTech

Autor: Jairo Alves da Silva Data: 21/06/2025

Micro serviço: OrderService

Serviço responsável pelo recebimento, processamento e exposição de pedidos vindos de um sistema externo A e consultado por um sistema externo B.

Este projeto foi desenvolvido para atender a um teste técnico com foco em alta volumetria, consistência de dados, cache e resiliência.

Tecnologias Utilizadas

- .NET 8 C#
- Dapper como micro-ORM
- SQL Server para armazenamento dos pedidos
- Azure Service Bus para mensageria assíncrona
- Redis como cache distribuído
- Swagger / OpenAPI para documentação da API
- Testes com xUnit, Moq e FluentAssertions
- Arquitetura Hexagonal

Estrutura do Projeto

OrderService/
— OrderService.Api/ # API REST para consulta dos pedidos (Sistema B)
├— OrderService.Application/ # Casos de uso e interfaces (UseCases)
├— OrderService.Domain/ # Entidades de negócio (Order, OrderItem)
— OrderService.Infrastructure/ # Acesso a dados (Dapper) + Redis Cache
├— OrderService.Worker/ # Worker que consome fila do Azure Service Bus (Sistema A)
├— OrderService.Tests/ # Testes unitários

4. Configurar as strings de conexão

OrderService.Api/appsettings.json

OrderService.Worker/appsettings.json

String conexão do Service Bus:

Endpoint=sb://service-

order.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=96ZB6HwDLYIT/s7y/Ca1Wwq3H4smK2NUx+ASbGX7F20=

5. Exemplo de mensagem na fila (mensagem enviada pelo Sistema A)

```
{
"ExternalOrderId": "EX123456",

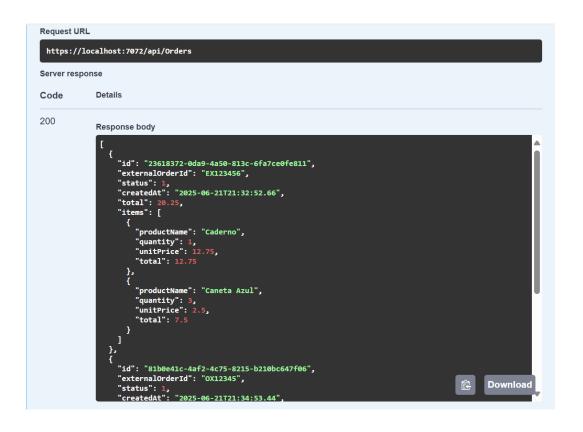
"Items": [
    {"ProductName": "Caneta", "Quantity": 3, "UnitPrice": 2.5 },
    {"ProductName": "Caderno", "Quantity": 1, "UnitPrice": 12.75 }
]
}
```

6. Endpoints Disponíveis (API a ser consumido pelo Sistema B)

Verbo Rota Descrição

GET /api/orders Retorna pedidos por status

GET /api/orders/{externalId} Retorna pedido específico



7. Desenho da Arquitetura

