Projeto da Aceleração Global Dev #3
Digital Innovation One
avanade
18/12/2020

Azure Service Bus

API REST – Simulando a troca de mensagens entre dois sistemas utilizando o serviço de mensagens em fila da Microsoft Azure

I. Apresentação:

- Marcelo França
- 7 anos de experiência
- Graduado em Sistemas de Informação
- Atuação no desenvolvimento e manutenção de sistemas corporativos

II. Resumo

Foi desenvolvido uma API REST:

- Utilizando o Visual Studio 2019 Community
- Criado um Aplicativo Web ASP NET Core - API REST
- Utilizando um banco de dados em memória
- Utilizando o Formato de mensagens – JSON
- Utilizando o serviço de mensagens: Microsoft Azure Service Bus

Objetivo do projeto:

- Criar dois módulos para simular a troca de mensagens utilizando o serviço Azure Service Bus.
- Criar 2 bases de dados independentes que serão gerenciadas através das mensagens trocadas utilizando o serviço em cloud.
- Criar regras de validação de cadastro.

  Criar o fluxo de comunicação entre os módulos aplicando os conceitos de fila e troca de mensagens entre sistemas.

III. Introdução

Para a solução foi criado um único projeto simulando a separação dos sistemas em duas pastas:

- Uma pasta para simular o sistema de estoque: (Estoque)
- Uma pasta para simular o sistema de vendas: (Vendas)

Cada módulo possui uma estrutura com a seguinte organização:

- Controllers
    - A classe utilizando os verbos http da API (GET, POST, PUT)
    - O Swagger para gerar a documentação e testes da API.
    - Os Filters do .NET Core para as validações.
- Repository
    - Utilizando o EntityFrameworkCore para realizar operações na base de dados com base no modelo de dados.
- Models
    - O modelo de dados – uma classe de produto com (Id, Codigo, Nome, Preço, Quantidade)
    - Uma classe representado o conexto para conexão com o banco de dados (O módulo de estoque – base: estoque , O módulo de vendas – base: vendas)
- Services
    - Uma classe de serviços responsável pelas regras de negócio.
    - Um classe responsável pela produção das mensagens.
    - As classes de serviços em background (`IHostedService/Consumer`), responsáveis pelo consumo das mensagens do serviço de barramento – ficam escutando as filas e processando as mensagens recebidas.
- ViewModels
    - Uma classe com base no modelo do negócio para utilização de regras de validação e customização de acordo com o comportamento desejado.
    - Classes de validação e filters utilizados no Controller.

IV. Necessidades/Problemas

O sistema foi desenvolvido para simular ao mesmo tempo a produção e o consumo das mensagens trocadas através do barramento Azure Service Bus.

O que gerou um problema inicial de injeção de dependencia dentro do IhostedService (background service), provocando uma InvalidOperationException quando ele era iniciado.

A solução foi criar uma injeção de dependencia com o IserviceScopeFactory dentro do IhostedService. Assim ao receber a mensagem um serviço é chamado e a base de vendas ou de estoque é atualizada.

O plano escolhido dentro da plataforma Microsoft Azure foi o Básico que não possui o modelo de tópicos. Então foi utilizado o modelo filas com duas políticas de acesso: Gerenciar(Produtor) e Ouvir(Consumidor).

V. Avaliação

O projeto possibilitou uma melhor visão das vantagens e desvantagens da solução:

- A facilidade na utilização da plataforma e do serviço de mensagens.
- O funcionamento dos processos assíncronos, o baixo acoplamento, redundância, resiliência, a capacidade de escalabilidade e a facilidade de criação de novos recursos.
- A necessidade de uma boa definição da interface de comunicação e regras para manter a integridade dos sistemas.