Pedro Antunes Negrão 21086916 11/08/2024

Sistemas Distribuídos - UFABC
Prof. Emilio Francesquini

Projeto 1 - MeuQoelhoMQ

Sobre

Este projeto implementa um sistema de mensageria, semelhante ao RabbitMQ. O servidor e um cliente foi desenvolvido em .NET, o outro cliente foi desenvolvido em Python. Algumas funcionalidades do projeto são: criação, exclusão e listagem de filas criadas, salvamento das mensagens em arquivos usando a serialização do gRPC, backup do servidor em caso de falha, usando a desserialização do gRPC, publicar mensagens isoladamente, publicar uma lista de mensagens, resgatar somente uma mensagem da fila ou realizar a assinatura de filas para receber as mensagens que são publicadas.

Como utilizar o código

Primeiro é necessário instalar o Git (sudo app install git). Depois é preciso baixar o .NET (No Linux, basta instalar o sdk do .NET via APT: sudo apt update && \ sudo apt install -y dotnet-sdk-8.0). Depois instalar o Python (caso ainda não esteja instalado), também via APT: sudo apt install python3. Instalar também o pip para instalação dos pacotes em Python: sudo apt install python3-pip. Por fim, instalar os pacotes de gRPC do Python, descritos igual está na página do grpc Python:

```
$ sudo apt install python3-venv
$ python -m venv venv
$ source venv/bin/activate
$ python -m pip install grpcio
$ python -m pip install grpcio-tools
```

Com tudo instalado, basta fazer o clone do projeto na máquina. O projeto foi separado em 5 pastas principais: Client (Cliente em C#), ClientPython, Proto (Pasta que contém o arquivo .proto que é compartilhado com todos os agentes do projeto), Server e Tests. Para baixar as dependências do projeto, basta executar o comando **dotnet restore** na raíz do projeto. Para buildar, basta executar o comando **dotnet build** na raíz também. Para executar os testes automáticos, basta executar o comando **dotnet test** na raíz do projeto. Com tudo isso preparado, basta executar primeiramente o servidor, acessando a pasta Server/src e executando o comando **dotnet run**. Após isso, basta acessar a pasta Client/src e executar o mesmo comando. Para testar o cliente em Python, basta acessar a pasta ClientPython e executar o comando **python client.py**.

Dificuldades e Supresas

Ao longo do código eu coloquei alguns comentários que explicam algumas dificuldades enfrentadas durante a implementação do projeto. Por exemplo, eu quebrei a cabeça por 1 dia inteiro para tentar resolver um problema que eu estava enfrentando com a lista de filas de mensagens. Basicamente o servidor não estava guardando as filas que eram criadas pelo cliente, então a cada nova execução de um cliente parecia que uma nova instância do servidor era criada, mesmo sem interromper a execução do servidor! Depois de muito tempo, eu consegui resolver definindo um Singleton para o meu Server. Dessa forma ele começou a funcionar corretamente.

Porém, nem tudo foi um caos. Tive uma surpresa boa quando eu percebi a facilidade de utilizar a serialização e a desserialização do gRPC. Além disso, implementar um servidor resiliente que consegue inicializar usando o backup para recriar as filas e as mensagens foi um desafio, mas vê-lo funcionando é muito legal.

Link do video explicativo: https://youtu.be/aNeMCYLYNBU