Pedro Antunes Negrão 21086916 16/08/2024

Sistemas Distribuídos - UFABC
Prof. Emilio Francesquini

# Projeto 2 - PokemonDHT

## Sobre

Este projeto, desenvolvido em C# (.NET), implementa uma tabela hash distribuída (DHT), semelhante ao Chord, cuja a topologia é de um anel, onde cada nó possui informação apenas de seu sucessor e predecessor. O objeto que foi abstraído para ser a informação que será transportada na rede são imagens de cartas de Pokemon. Além do projeto implementar a DHT, também foi criado uma aplicação CLI para interagir com as funcionalidades da DHT, como entrar na rede, realizar o armazenamento de uma carta, resgatar uma carta e deixar a rede. Ou seja, cada instância da aplicação também é um nó na rede, e só é possível realizar certas ações se a aplicação fizer parte da rede. Como é uma rede P2P, cada nó atua como cliente e servidor simultaneamente.

## 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).

Com tudo instalado, basta fazer o clone do projeto na máquina. O projeto foi separado em várias pastas, desde Domain e Interfaces até Exceptions e Helpers. 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 o comando **dotnet run** na pasta src/App.

No vídeo de apresentação foi demonstrado o uso do programa usando módulo de 16 na hora de calcular os hashs, para facilitar a visualização dos IDs dos nós e das cartas. No repositório eu irei retirar o módulo para manter os hashs enormes calculados pelo SHA-256, porém eu vou manter com uma anotação de **//TODO** as linhas onde eu aplico o módulo, caso queria testar igual o video de apresentação (Arquivos: TypeHelper.cs e SHA256Hash.cs).

## Dificuldades e Surpresas

No começo, a maior dificuldade foi tentar estruturar a arquitetura do projeto e modelar como funcionaria cada Nó. Como recomendado no PDF do projeto, eu segui os Milestones para tentar me ajudar a organização do código, definindo o Domínio e a interface CLI. Porém eu não tinha entendido inicialmente que cada nó teria que implementar tanto o Client quanto o Server do protocolo implementado via gRPC, então eu comecei a desenvolver a aplicação implementando somente o Client, porque quando se pensa numa rede P2P, é intuitivo pensar na conexão entre clientes, sem servidor. Quando eu cheguei na parte de definir um novo Nó na rede, foi aí que eu percebi que estava faltando alguma coisa. E foi nesse momento que as coisas ficaram complicadas, porque no projeto 1 tivemos que criar um servidor e um cliente separado, e o bootstrap do servidor é configurado automaticamente no ecossistema .NET. Nessa aplicação eu tive que realizar essa configuração manualmente, e salvar a instancia de um servidor rodando dentro um objeto do tipo Nó. É meio loucura pensar assim, mas deu certo!

Por fim, eu gostei demais de conseguir implementar uma DHT. Pude aprender tantas configurações na minha linguagem de programação escolhida que eu mal conhecia. Eu acredito que esses projetos, mesmo sendo bastante complicados de implementar, é gratificante ver funcionando, porque dá pra perceber que é possível implementar qualquer coisa, o céu é o limite!

**Link do video explicativo: https://youtu.be/iQsqAJnYdvA**