Instituto Superior de Engenharia de Lisboa

Mestrado em Engenharia Informática e de Computadores

Computação Distribuída

Semestre de Inverno 2020/2021

Laboratório #2

Docente: Luís Assunção

Realizado por: Grupo 6

- Edgar Alves nº 33017
- Paulo Pimenta nº 47972
- João Silva nº 42086

22.11.2020

Exercício 01

O objetivo neste exercício era desenvolver duas aplicações (Aplicação Cliente e Aplicação Servidor) em Google RPC (gRPC) com execução da aplicação Servidor na máquina virtual na Google Cloud Platform, sendo que a aplicação Servidor iria comunicar comunicar com o *Central Server* com o endereço IP: 35.230.146.225 no port 7500. Neste CentralServer é disponibilizado um contrato gRPC como é indicado no enunciado, e cujo mesmo contrato nos foi disponibilizado para implementação na aplicação Cliente e na aplicação Servidor. O contexto deste exercicio foi baseado num cenário de uma estrada com 5 pontos de acesso que permite entradas e/ou saídas de veículos, que conforme o percurso efetuado por esse veículo iria se aplicar uma tarifa ao mesmo, sendo que a nossa aplicação Cliente iria ser o veículo. Seguindo os slides das aulas e as instruções recomendadas, desenvolveu-se o respetivo Contract-1.0.jar com a interface indicada pelo enunciado. Após isso, desenvolveu-se a aplicação Cliente, começando pela definição do pom.xml para o maven importar as respetivas dependencies necessárias e para se implementar o respetivo Contract-1.0.jar. Na aplicação Cliente, instanciou-se um canal que iria ser a conexão para a nossa aplicação Servidor, instanciou-se um stub (*Non Blocking Stub*) e instanciou-se a classe StreamObserverClient, em que esta classe implementa *StreamObserver*. A partir daqui, a nossa aplicação Cliente conseguia gerar um request (em que este indicava a matrícula e o ponto de entrada) e realizou-se esta chamada (request) para a nossa aplicação Servidor.

Na aplicação Servidor, iniciou-se o desenvolvimento da mesma através da definição do pom.xml para o maven importar as dependencies necessárias e implementou-se o Contract-1.0.jar e o CentralContract-1.0.jar. Após isso, a partir da aplicação Servidor, recebiamos essa chamada com o Initial e guardamos a informação num *ConcurrentHashMap* onde a informação da matricula e do ponto de entrada eram guardados, após serem guardados, a StreamObserver era encerrada chamando o método *onCompleted().* Na aplicação Cliente, quando o utilizador deseja enviar um aviso, o mesmo *stub* foi utilizado para se criar um novo request à aplicação Servidor, em que neste request se enviou o objeto instanciado do tipo *WarnMsg,* em que se definiu a matricula (*setId(matricula)*) e a mensagem de aviso (setWarning(msg)). No lado da aplicação Servidor, era recebido este *StreamObserver* do tipo *WarnMsg,* em que o Servidor iria guardá-los num *ArrayList*. Outros clientes que tivessem enviado mensagens e que estivessem presentes neste *ArrayList*, iriam também receber as mensagens que iriam vir de outros clientes e assim sucessivamente.

Quando um cliente decidisse escolher um ponto de saída para o seu veículo, foi necessário instanciar um objeto da classe *FinalPoint*, onde neste se definiu a matrícula (*setId(matricula)*) e o ponto de saída (*setOutPoint(endPoint)*) e, também foi necessário instanciar um objeto do tipo *StreamObserverPayment* que implementava o *StreamObserver* do tipo *Payment*. Através do método *leave,* que recebe como parâmetros o objeto *FinalPoint* e o objeto *StreamObserverPayment,* a aplicação Servidor recebe este request do Cliente e iria criar uma conexão ao *Central Server*. Após criar a conexão, a nossa aplicação Servidor iria percorrer a ArrayList e procurar pelo ID (neste caso, matricula) para se saber o *In Point* do veículo em questão, após isso, o Servidor iria instanciar um objeto *Track* onde se definiu o *In Point* e o *Out Point* (que veio no request do Cliente, no objeto *FinalPoint)*, criou-se um stub (neste caso um *Blocking Stub)*, de forma a criar um request para se enviar ao *Central Server,* onde se enviou o nosso objeto *Track*. Após o Central Server enviar a resposta para

a nossa aplicação Servidor, a mesma iria instanciar um objeto do tipo *Payment* onde se iria definir o valor de pagamento que foi devolvido pelo *Central Server,* em que depois o Servidor envia este mesmo objeto como resposta para a nossa aplicação Cliente, e após enviar a nossa aplicação Servidor encerra este *StreamObserver* com o cliente e removendo este mesmo do *ArrayList* (Onde se guardava a informação relativa aos veículos).

Durante este exercício, não houve dificuldades registadas.