

## Computação Distribuída

### Síntese de atividades de laboratório

Laboratório nº: 2

Data: Domingo, 22 de Novembro 2020

Turma: MI1N

Grupo: 24

Integrantes:

Número	Nome
41548	Guilherme Arede
44021	Nuno Gomes

#### Atividade proposta:

Desenvolvimento de aplicações Cliente/Servidor em Google RPC e execução da aplicação servidora em máquina virtual na Google Cloud Platform.

##### 1. Objetivo da atividade:

Esta atividade prática foi fazer a implementação de um sistema de gestão de estradas, tendo um ponto de entrada e um ponto de saída. Conforme o percurso realizado, é calculada a tarifa de circulação sobre o mesmo. Existe ainda a hipótese de enviar mensagens de aviso sobre o sistema, de forma a informar outros utentes da via sobre possíveis perigos que possam ser encontrados.

##### 2. Descrição da solução:

A solução desenvolvida, engloba quatro componentes: Cliente, dois Contratos, Servidor de Controlo e Servidor Central. O Servidor Central e o Contrato não são descritos no relatório pois a sua execução não fazia parte do trabalho, apenas a sua utilização.

O servidor Central, fornecido em anexo à atividade de laboratório, já se encontra implementado, sendo apenas necessário estabelecer uma ligação ao seu endereço de IP, também fornecido.

O contrato entre o servidor de Controlo e o servidor central apenas é utilizado para fazer pedidos de tarifas.

O contrato entre o servidor de controlo e o cliente, já é usado para sinalizar quando um carro entra numa estrada da nossa rede, sai de uma estrada, ou tenciona avisar os outros utilizadores de algo.

---

#### *Servidor de Controlo*

Nesta componente, desenvolvida por nós, considera-se que o servidor funciona como um cliente do servidor central. Seguindo esta abordagem, foi necessário estabelecer a ligação ao Servidor Central, de forma a conseguir comunicar com o mesmo, para poder obter os preços das tarifas a pagar por cada percurso efetuado na estrada. Implementou-se ainda o contrato fornecido em anexo, de forma a obter acesso aos métodos “enter”, “warning” e “leave”. Foi criada ainda uma classe designada de “WarningObserver”, de forma a implementar um contrato de comunicação entre os clientes e este servidor, para se obter suporte ao envio e distribuição de mensagens de aviso para todos os utentes da estrada. Em todas as operações realizadas pelo servidor, antes do mesmo concluir a mesma, é enviada uma notificação sobre o resultado da sua execução, bem como mais alguma informação necessária, através da utilização de um “StreamObserver”. O comportamento das operações suportadas pelo servidor, descreve-se da seguinte forma:

##### *- Operação “enter”:*

Quando um utilizador entra na via, esta operação é invocada. É recolhido o ponto inicial da entrada na estrada e criado um tuplo com a informação do ponto de entrada associada à matrícula do carro que iniciou o percurso. É devolvido um aviso, notificando que o registo foi efetuado com sucesso.

### - Operação “leave”:

Quando um utilizador sai da via, esta operação é invocada. É recolhido o ponto de saída na estrada e através da matrícula que vem embebida no pedido, obtém-se o tuplo criado anteriormente para obter o respetivo ponto de entrada. Com essa informação, é efetuado um pedido ao Servidor Central, informando ambos os pontos, e obtém-se por parte deste, o valor da tarifa correspondente ao percurso realizado. Este valor é devolvido para o utilizador, notificando-o do custo do trajeto, e o tuplo criado com a informação da matrícula e do ponto de entrada é removido.

### - Operação “warning”:

Quando um utilizador entra na via, este tem possibilidade de subscrever a notificação de avisos sobre possíveis perigos na via. Se o utilizador assim desejar, é feita a chamada a esta operação. Inicialmente, é feita a geração de um código aleatório associado ao utilizador em questão, código esse que é devolvido ao mesmo. A esse código, é associado uma instância de “*StreamObserver*”, para existir a possibilidade de enviar notificações para esse cliente. O Servidor de Controlo fica à espera de que o cliente faça um novo pedido, utilizando esse código, para poder obter a matrícula do mesmo. Quando o cliente informa o Servidor de Controlo do código recebido, é feito um novo mapeamento entre a matrícula que vem embebida no envio do pedido que tem o respetivo código gerado, e a instância de “*StreamObserver*” previamente criada, agora associada à matrícula do cliente. Esta operação apenas é efetuada uma única vez, de forma a permitir a subscrição de avisos por parte dos utilizadores da via. Quando existe um utilizador que deseje informar sobre um aviso, o Servidor envia a notificação recebida para todos os outros clientes, através das instâncias de “*StreamObserver*” armazenadas.

---

## Cliente

---

O cliente implementa as operações de *sendWarning()*, *leaveAccessPoint()* e *enterAccessPoint()* também. As operações *enterAccessPoint()* e *leaveAccessPoint()*, são simples, na medida em que são unárias e apenas passam ao servidor a informação de entrada, ou saída de um cliente em dado ponto de acesso ou saída. A operação *leaveAccessPoint()* tem um pouco mais de granularidade pois também se encarrega de desligar o observer do serviço de *Warnings*. Isto é feito pois foi interpretado que um cliente apenas deve ter avisos das nossas redes, caso ele se encontre a usufruir das mesmas.

A operação *sendWarning()* verifica se o utilizador se encontra numa estrada de forma a garantir que ele está a usar os nossos serviços. Caso esteja e não esteja ainda a utilizar o serviço de mensagens, é criado o observer para tal, e nesta criação, é feita uma primeira mensagem com a chave dada pelo servidor, de forma ao servidor ficar a saber a matrícula do carro e associar ao observer que é passado do cliente. Por fim, caso tenha sido feita a configuração, é perguntado ao cliente se ele tenciona enviar uma mensagem, ou se apenas chamou o método para receber os avisos. Caso o utilizador escolha enviar mensagem, esta é enviada com o observer recebido do servidor.

Também houve um cuidado especial de forma a garantir que sempre que o Cliente se desligava, os recursos do servidor eram libertados. Isto foi feito mesmo em casos que o cliente feche de forma excecional, é feita uma tentativa de libertação de recursos. Caso o cliente queira sair de forma normal, também é verificado se ele saiu da estrada, e no caso de não ter saído é pedido que indique a saída antes de sair.

### 3. Indicação se a solução final é apresentável e demonstrável:

A solução final é apresentável e demonstrável, no caso do servidor central numa aplicação de consola, e no caso das restantes partes num IDE compatível, que neste caso foi o IntelliJ.

### 4. Conclusão e lições aprendidas:

Foi aprendida a diferença entre chamadas bloqueantes, não bloqueantes e também unárias, ou com streams de cliente, servidor ou ambos. Conclui-se que usando o caso de stream de cliente e de servidor é um pouco mais trabalhoso, mas permite uma maior flexibilidade para evolução de um projeto, mesmo que não se faça uso pleno dessa implementação.

Foi compreendida a importância e a utilidade do uso de gRPC para a comunicação entre partes de uma aplicação. Isto porque foi feita uma implementação mais fácil e legível do que uma idêntica feita com uma API REST, que seria o que normalmente usaríamos neste caso. O gRPC também permite utilizar