Laboratório 4

Objetivos:

- Desenvolver serviços distribuídos com tecnologia *Google Remote Procedure Call* (gRPC)
- Chamadas com *stream* de cliente e stream de servidor

Nota: Como resultado terão de submeter no Moodle a estrutura dos projetos Maven/Intellij desenvolvidos, isto é, só o ficheiro pom.xml e a diretoria src, sem ficheiros a partir da diretoria target

Considere um cenário, onde uma autarquia pretende implementar um sistema de gestão e controlo de painéis informativos instalados numa cidade.

Cada painel pode mostrar mensagens de texto (anúncios/avisos), bem como ler, reportar e receber valores de sensores de ambiente (por exemplo temperatura ou poluição).

Os painéis enviam e recebem informação de um serviço com uma API gRPC, sendo simulados neste laboratório através de uma aplicação cliente de consola (*gRPCpanel*) que permite o envio e receção de mensagens de texto com alertas (por exemplo, “este painel foi vandalizado”), bem como o envio e receção de valores de temperatura.

O serviço com API gRPC (servidor *gRPCpanelManager*) tem os seguintes requisitos:

- Aceitar o pedido de um painel para atribuição de um ID único;
- Receber periodicamente os dados de temperatura existente na zona do painel;
- Retornar todas as temperaturas previamente enviadas por um painel;
- Receber mensagens de alerta enviadas a partir dos painéis;
- Quando o servidor receber mensagens de alerta vindas de qualquer painel deve enviar essa mensagem para todos os painéis que tenham feito o registo para receberem esses alertas.

Implemente os 3 projetos IntelliJ para: definição do contrato, servidor, e cliente consola de simulação do painel.

Sugestão esboço do Contrato

rpc init(Local) returns (PanelID);
rpc sendTemperatures(stream Temp) returns(Void);
rpc getAllTemperatures(PanelID) returns (AllTemps);
rpc sendAlert(Alert) returns (Void)
rpc registerForAlerts(PanelID) returns (stream Alert)
	Local: string
	PanelID: número inteiro único
	Temperature: PanelID, *double*
	AllTemps: PanelID, Array (repeated) de *double*
	Alert: PanelID, local do painel, texto
	Void: mensagem vazia

Opcionalmente pode implementar o contrato só com a operação register e mais uma operação de caso 4 com stream de cliente e stream de servidor para que, após o registo do painel no servidor, toda a comunicação bidirecional possa ser realizada com uma única operação e mensagens com campos variáveis (*oneof*).

```
message BidirecionalMessage  {
    int32 PanelID = 1;
    oneof MsgOptions {
        Temperature tValue = 2;
        ListTemps allTemps = 3;
        Alert alert = 4;
    }
}

message ListTemps {
    repeated Temperature tValues = 1;
}
```