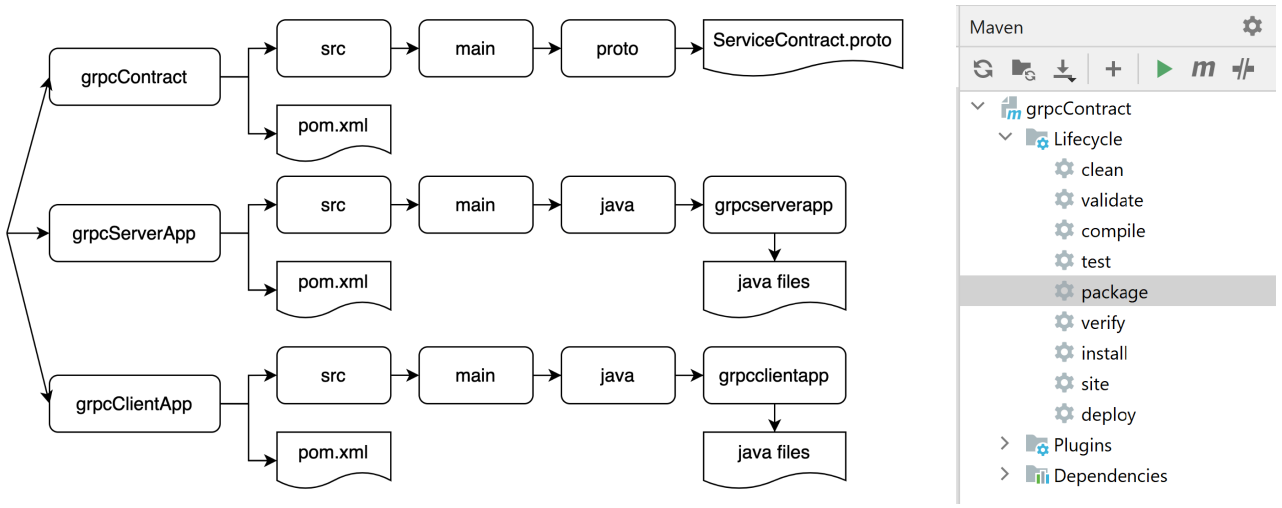
Laboratório 2

Objetivos:

- Desenvolver serviços distribuídos com tecnologia *Google Remote Procedure Call* (gRPC)
- Distinguir entre chamadas unárias e chamadas com *stream* de servidor

Parte 1

1) Considere o exemplo dos três projetos gRPC apresentados e demonstrados na aula (anexo `grpcBase-Slides.zip`), com a estrutura apresentada na Figura 1a). Os projetos `grpcServerApp` e `grpcClientApp` têm a estrutura de diretorias Maven com o respetivo pom.xml. O projeto `grpcContract`, para além do pom.xml, tem o contrato definido no ficheiro `serviceContract.proto` que, por convenção do compilador de `protobuf`, está localizado na diretoria `\src\main\proto`. Veja abaixo (secção anexo) como abrir os projetos Maven no Intellij.



a) b)

Fig. 1: a) Estrutura de directorias e ficheiros dos projetos fornecidos; b) Janela Maven no IntelliJ

No projeto `grpcContract`, usando o menu Maven no ambiente IntelliJ, gere o JAR com o contrato (ação ***package***) e instale no repositório local do Maven (ação ***install***).

2) Acrescente ao contrato e realize a respetiva implementação no servidor da seguinte operação rpc:

```
rpc findPrimes(IntervalNumbers) returns (stream IntNumber)
message IntervalNumbers { int32 start = 1; int32 end = 2; }
```

a) Realize uma nova aplicação cliente que, usando um *stub* não bloqueante, realiza 5 chamadas à operação `findPrimes` para calcular os números primos entre 1 e 500, em intervalos de 100 números ([1,100], [101,200], …).

b) Após testar na sua máquina local, crie o artefacto do projeto servidor (opção *package* no menu Maven), copie o artefacto para uma VM no seu projeto GCP e teste as aplicações cliente a conectarem-se ao servidor em execução na VM.

Parte 2

1) Considere um sistema para troca de mensagens (*Forum*) cujo envio e receção de mensagens está organizado em tópicos. O contrato do serviço a disponibilizar pela aplicação servidora é apresentado em seguida:

```
service Forum {
   // subscribe a topic
   rpc topicSubscribe(SubscribeUnSubscribe) returns (stream ForumMessage);
   // unsubscribe a topic
   rpc topicUnSubscribe(SubscribeUnSubscribe) returns (google.protobuf.Empty);
   // get all topics in server
   rpc getAllTopics(google.protobuf.Empty) returns (ExistingTopics);
   // send a message to a topic
   rpc publishMessage(ForumMessage) returns (google.protobuf.Empty);
}

message SubscribeUnSubscribe {
   string usrName = 1;
   string topicName = 2;
}
message ExistingTopics {
   // list of topic names
   repeated string topicName = 1;
}
message ForumMessage {
   string fromUser = 1;
   string topicName = 2;
   string txtMsg = 3;
}
```

a) Implemente a aplicação servidora que disponibiliza o serviço *Forum*. Assuma que o *usrName* é único para cada utilizador e que um tópico é criado no servidor após a primeira subscrição do mesmo por parte de um utilizador.

   Sugere-se que a gestão do estado do servidor relativo a tópicos e utilizadores conectados seja suportada pela seguinte estrutura de dados, a qual consiste em ter um *hashmap* cuja chave é o nome de um tópico e cujo valor é outro *hashmap* cuja chave é o nome dos utilizadores, que subscreveram o tópico, e o valor o respetivo *stream observer* onde cada utilizador receberá as mensagens do tópico:

```
// topic -> { map: username -> streamObserver }
ConcurrentMap<String, ConcurrentMap<String, StreamObserver<ForumMessage>>>
```

b) Implemente uma aplicação cliente, em modo consola, que permita aos utilizadores trocarem mensagens em vários tópicos.

ANEXO

Como abrir projetos no *IntelliJ* a partir da descrição Maven (*pom.xml*).