Laboratório 2

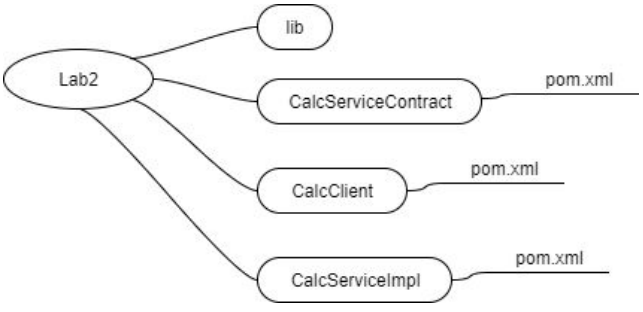
Objectivos:

- Prática com implementação de serviços com gRPC
- Prática de acesso a serviços implementado com gRPC

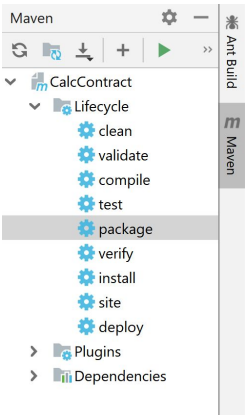
1. Considere a seguinte definição de contrato em linguagem protocol buffers. O contrato é de um serviço com duas operações: **add** - que recebe dois números `double` e retorna o resultado da soma; e **findPrimes** que calcula uma sequência de *n* (*`numOfPrimes`*) números primos existentes a partir de um dado número inicial (*startNum*).

```
syntax = "proto3";
option java_multiple_files = true;
option java_package = "calcstubs";

package calcservice; // package do proto

service CalcService {
   rpc add(OperationRequest) returns (OperationReply);
   rpc findPrimes (SeriesOfPrimes) returns (stream Prime);
}
message OperationRequest {
   double op1 = 1;
   double op2 = 2;
}
message OperationReply {
   double res = 1;
}
message SeriesOfPrimes {
   int32 numOfPrimes=1; // número de primos a calcular
   int32 startNum=2;    // número inicial para procurar primos
}
message Prime {
   int32 prime = 1;     // um número primo
}
```

Considere o ficheiro Lab2.zip com a seguinte estrutura de diretorias:



a. Considere a diretoria *CalcServiceContract* com o projeto *maven* (em `pom.xml`) e o contrato do serviço em `src\main\proto\CalcService.proto`. Importe para o IntelliJ o projeto maven (selecionando o `pom.xml`), e gere o artefato `CalcContract-1.0.jar` com os stubs e classes de serialização, fazendo duplo click em *package* no menu Maven.



b. Implemente o serviço *CalcService*, o qual depende do artefato gerado na alínea anterior. Considere a diretoria CalcServiceImpl na qual está um projeto *maven* com as dependências do middleware gRPC e uma dependência explícita para o .jar gerado na alínea anterior. O pom.xml assume que o .jar está na directoria `lib` na raiz do projeto.

c. Implemente um cliente que acede ao serviço com *stubs*:
   i. Bloqueantes
   ii. Não bloqueantes

d. Acrescente uma nova operação ao serviço que permita enviar um *stream* de números inteiros e obter como resposta a soma de todos os inteiros. Realize um cliente para testar a nova operação.

e. Acrescente uma nova operação ao serviço que permita enviar um *stream* de operações de adição (*message OperationRequest*) e obter um *stream* onde o serviço escreve o resultado de cada adição (*message OperationReply*). Realize um cliente para testar a nova operação.

f. A operação *findPrimes*, permite obter N (`numOfPrimes`) primos a partir de um número inicial (*startNum*). Acrescente ao serviço uma nova operação que retorna um stream de números primos no intervalo [n_1, n_2]. Realize um cliente que divide e obtém em paralelo os número primos do intervalo [1, 100].