

Laboratório 02

Objetivos: Desenvolver serviços distribuídos com tecnologia *Google Remote Procedure Call* (gRPC), distinguindo entre chamadas unárias e chamadas com *stream* de cliente e *stream* de servidor

Pretende-se desenvolver um serviço para vários cálculos de números, em que os clientes do serviço podem enviar pedidos de cálculo retornando resultados de forma síncrona ou assíncrona.

- 1) Em anexo ao enunciado, no ficheiro gRPCCalc.zip são fornecidos 3 projetos Maven (contrato, servidor e cliente), com a estrutura apresentada na Figura 1a). O projeto CalcContract, para além do pom.xml, tem o contrato definido no ficheiro CalcService.proto que, por convenção do compilador de protobuf, está localizado na diretoria \src\main\proto. Os projetos CalcServerApp e CalcClientApp têm apenas a estrutura de diretorias de projetos Maven e o respetivo pom.xml.

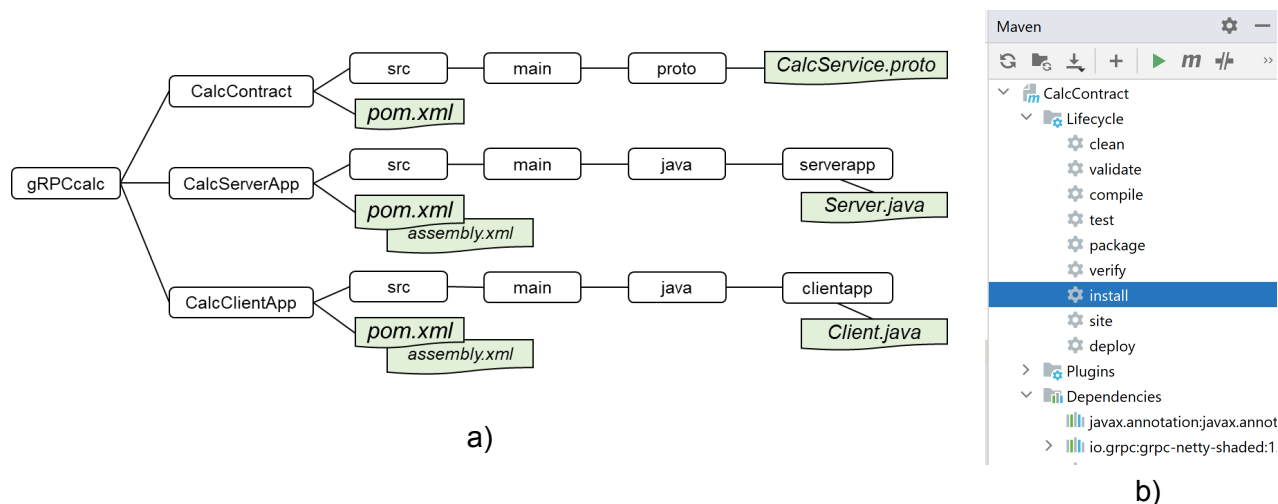
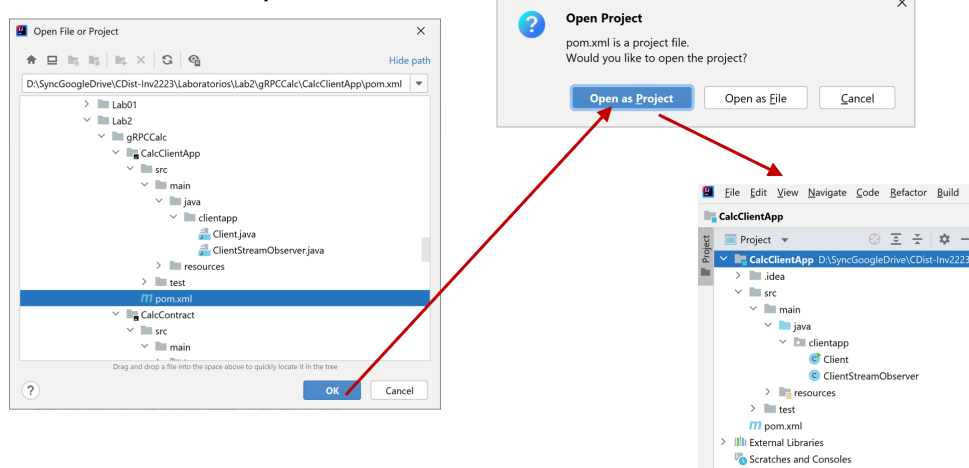


Fig. 1: a) Estrutura de directorias e ficheiros dos projetos fornecidos; b) Janela Maven no IntelliJ

Para abrir no IntelliJ os projetos Maven:

Abrir o projeto selecionando na estrutura de diretorias o ficheiro *pom.xml*



Unidade Curricular de Computação Distribuída

- 2) No projeto CalcContract, usando o menu Maven no ambiente IntelliJ, gere o JAR com o contrato (ação **package**) e instale no repositório local do Maven (ação **install**).
- 3) No projeto do **servidor** (CalcServerApp) verifique que já existe no ficheiro pom.xml uma dependência para o artefacto do contrato gerado na alínea 2. Implemente no ficheiro Server.java as operações do contrato. Note que a inicialização do servidor já está concretizada.
- 4) No projeto do **cliente** (CalcClientApp) verifique que já existe no ficheiro pom.xml uma dependência para o artefacto da contrato gerado na alínea 2. No ficheiro Client.java já existe a inicialização do cliente para acesso ao servidor (criação do *Channel* e *Stubs* síncronos e assíncronos).
 - a) Usando um *stub* bloqueante chame a operação *add* para verificar a correta conectividade ao serviço para realizar uma operação unária (caso 1).
 - b) Implemente a chamada às restantes operações do contrato usando um *stub* não bloqueante.
- 5) Teste as aplicações CalcServerApp e CalcClientApp invocando os JAR existentes na diretoria target após fazer *package* na janela Maven dos projetos no IntelliJ:
 - a) Na diretoria do projeto CalcServerApp:
java -jar target\CalcServerApp-1.0-jar-with-dependencies.jar 8500
 - b) Na diretoria do projeto CalcClientApp:
java -jar target\CalcClientApp-1.0-jar-with-dependencies.jar localhost 8500
- 6) Após os testes na máquina local, execute a aplicação servidor CalcServerApp-1.0-jar-with-dependencies.jar numa VM do seu projeto GCP, mantendo a aplicação cliente na sua máquina local. Garanta que na *firewall* da VM está aberto o porto usado pelo servidor.
- 7) Tal como fez no Lab01, crie um ficheiro *Dockerfile*, que permita criar uma imagem Docker de nome cd-lab2-calc-server:v1 com o servidor CalcServerApp-1.0-jar-with-dependencies.jar. Execute um ou mais *containers* que disponibilizam o serviço de cálculos com números, por exemplo: docker run -d -p 8000:8500 cd-lab2-calc-server:v1
 - a) Teste a funcionalidade com múltiplos clientes concorrentes e ligados a diferentes portos da máquina virtual para aceder ao mesmo serviço em diferentes *containers*.
- 8) Publique (PUSH) a imagem de base do container no Docker Hub e remova localmente a imagem anterior.
- 9) Realize a operação PULL e volte a testar a funcionalidade como fez na questão 7a)