Q1.1 - gRPC 1

Assinale a resposta correta em cada alínea, ou deixe em branco em caso de dúvida (a resposta errada desconta 50% do valor atribuído à alínea)

Considere a seguinte operação gRPC e as respetivas mensagens:

```
rpc oper(MsgX) returns (MsgY)
message MsgX { int32 x = 1; }
message MsgY { int32 y = 1; }
```

Verdadeiro ✔ **No cliente a chamada à operação pode ser feita usando qualquer tipo de stub**

Falso ✔ **O formato da mensagem MsgX teria de ser modificado caso a operação passasse a ser stream de cliente**

Q1.2 - gRPC 1

Assinale a resposta correta em cada alínea, ou deixe em branco em caso de dúvida (a resposta errada desconta 50% do valor atribuído à alínea)

Considere a seguinte operação gRPC e as respetivas mensagens:

```
rpc oper(MsgX) returns (MsgY)
message MsgX { int32 x = 1; }
message MsgY { int32 y = 1; }
```

Falso ✔ **No servidor, a implementação da operação determina se o cliente pode ou não usar um stub bloqueante**

Falso ✔ **No servidor, a implementação da operação 'oper' tem a assinatura: public MsgY oper(MsgX) {...}**

Q2.1 - gRPC 2

Assinale a resposta correta em cada alínea, ou deixe em branco em caso de dúvida (a resposta errada desconta 50% do valor atribuído à alínea)

No contexto do serviço Forum realizado no laboratório 2, e assumindo o contrato:

```
service Forum {
  rpc topicSubscribe(SubscribeUnSubscribe) returns (stream ForumMessage);
  rpc topicUnSubscribe(SubscribeUnSubscribe) returns (google.protobuf.Empty);
  rpc getAllTopics(google.protobuf.Empty) returns (ExistingTopics);
  rpc publishMessage(ForumMessage) returns (google.protobuf.Empty);
}
```

Falso ✔ **Usando um stub bloqueante, o cliente chama a operação publishMessage passando como parâmetro um objeto stream que implementa StreamObserver<ForumMessage>, obtendo como retorno um objeto do tipo Empty.**

Verdadeiro ✔ **A implementação no servidor referente à operação publishMessage tem a seguinte assinatura:**

```
public void publishMessage(ForumMessage msg, StreamObserver<Empty> responseObserver)
```

Q2.2 - gRPC 2

Assinale a resposta correta em cada alínea, ou deixe em branco em caso de dúvida (a resposta errada desconta 50% do valor atribuído à alínea)

No contexto do serviço Forum realizado no laboratório 2, e assumindo o contrato:

```
service Forum {
  rpc topicSubscribe(SubscribeUnSubscribe) returns (stream ForumMessage);
  rpc topicUnSubscribe(SubscribeUnSubscribe) returns (google.protobuf.Empty);
  rpc getAllTopics(google.protobuf.Empty) returns (ExistingTopics);
  rpc publishMessage(ForumMessage) returns (google.protobuf.Empty);
}
```

Verdadeiro ✔ **Após a chamada à operação topicSubscribe, o middleware gRPC garante que as mensagens (ForumMessage) chegam ao cliente pela ordem enviada pelo servidor.**

Falso ✔ **A aplicação cliente do serviço Forum só pode usar um stub bloqueante para invocar a operação getAllTopics porque é uma operação unária.**

Q3.1 - gRPC 3

Assinale a resposta correta em cada alínea, ou deixe em branco em caso de dúvida (a resposta errada desconta 50% do valor atribuído à alínea)

Sobre o middleware gRPC:

Verdadeiro ✔ **O JAR que resulta da compilação de um contrato protobuf tem de estar acessível ao cliente e ao servidor.**

Verdadeiro ✔ **No caso das operações com stream de servidor (e sem stream de cliente), se o cliente usar um stub não bloqueante, é o cliente que tem de definir uma classe que implemente a interface StreamObserver<T>**

Q3.2 - gRPC 3

Assinale a resposta correta em cada alínea, ou deixe em branco em caso de dúvida (a resposta errada desconta 50% do valor atribuído à alínea)

Sobre o middleware gRPC:

Falso ✔ **Numa operação com stream de cliente e stream de servidor a quantidade de mensagens enviadas em cada stream é sempre igual.**

Falso ✔ **No laboratório 2 sobre o Forum, se a estrutura de dados usada no servidor para guardar as mensagens for alterada (por exemplo, para ser usada uma lista), o contrato protobuf tem obrigatoriamente de ser alterado também.**