Laboratório 2

Objectivos:

- Desenvolver serviço distribuído com tecnologia *Java Remote Method Invocation* (RMI)
- Analisar as possibilidades e as limitações da tecnologia RMI para cenários de pedidos com iniciativa do servidor (*callback*)
- Configuração dos portos TCP/IP nas VM GCP através de regras de *firewall*

Pretende-se desenvolver em RMI um serviço para cálculo de números primos. A aplicação cliente indica um intervalo de números inteiros o qual é usado pela aplicação servidora para encontrar quais os números primos existentes nesse intervalo.

1) Desenvolva o serviço implementando projetos independentes para o contrato (gerando um JAR como *library*), para o servidor e para o cliente (como dois JAR executáveis):

   a) Considere o seguinte contrato, entre cliente e servidor, composto por duas interfaces: `IPrimesService` e `ICallback`. Na interface `IPrimesService` o método `findPrimes` recebe dois números [*startNumber*, *endNumber*] que definem um intervalo para encontrar números primos e o *callback* para ir enviando os números primos encontrados. Crie o artefacto do contrato (JAR *library*) que será utilizado pelas aplicações cliente e servidor.

```
public interface IPrimesService extends Remote {
   void findPrimes(int startNumber,
                   int endNumber,
                   ICallback callback) throws RemoteException;
}

public interface ICallback extends Remote {
   void nextPrime(int prime) throws RemoteException;
}
```

   b) Implemente o servidor RMI de forma a que a operação *findPrimes* não bloqueie a chamada do cliente até encontrar o número de primos no intervalo especificado, retornando cada número primo que vai sendo encontrado no intervalo, através do *callback* passado pelo cliente na chamada. Na implementação, sugere-se a seguinte função para determinar se um número é primo.

```
static boolean isPrime(int num) {
  if (num <= 1) return false;
  if (num == 2 || num == 3) return true;
  if (num % 2 == 0) return false;
  for (int i=3; i <= Math.sqrt(num); i+=2) {
     if (num % i == 0) return false;
  }
  return true;
}
```

c) Desenvolva o cliente do serviço, o qual lê do *standard input* o número inicial (`startNumber`) e o número final (`endNumber`) do intervalo, faz a chamada ao servidor indicando o seu objecto de recepção das notificações de callback que mostra no *standard output* a sequência de primos retornados pelo servidor.

2) Dado que temos concorrência implícita no servidor haveria vantagem do cliente em procurar números primos dividindo o espaço de pesquisa em múltiplos intervalos, por exemplo, entre [1, 500], [501, 5000], ..., [10000, 20000]. Sem alteração dos contratos e da implementação do servidor, implemente uma nova aplicação cliente que permita obter em simultâneo os números primos existentes em múltiplos intervalos.

3) Após testar a correta funcionalidade da solução no seu computador pessoal, crie os artefactos JAR executáveis das aplicações desenvolvidas (servidor e clientes) .

   a) Execute o servidor numa VM1 do seu projeto GCP;

   b) Execute uma das aplicações cliente numa segunda VM2 e aceda ao servidor em execução na VM1 (alínea a) verificando o correto funcionamento.

   c) Execute uma instância da aplicação cliente no seu computador pessoal e aceda ao servidor em execução na VM1 conforme alínea a). Note que neste caso vai encontrar um problema que não tem solução fácil de resolver no contexto deste laboratório. Constate a existência do problema e encontre uma explicação conceptual para o mesmo.