Escola de Ciências e Tecnologia
Departamento de Informática
Licenciatura em Engenharia Informática
Unidade Curricular Sistemas Distribuídos
Ano letivo 2020/2021

Relatório Trabalho 1

Docente:
Professor José Saias
Discentes:
José Santos - 43017
Filipe Alfaiate - 43315

May 1, 2021

Contents

1 Introdução

No âmbito da unidade curricular de Sistemas Distribuídos lecionada pelo professor José Saias, foi solicitado, como primeiro trabalho, que os alunos implementassem um sistema de apoio ao cidadão para inscrição voluntária em fila de espera para vacinação e para o reporte de efeitos secundários após a mesma.

Utilizando uma aplicação `servidor` e uma aplicação `cliente`, comunicando com uma solução de `Middleware, Remote Method Invocation (RMI)`.

Para este fim, foram criadas inicialmente a interface remota e a implementação da mesma, nomeadas `Vacinacao` e `VacinacaoImpl` respetivamente. Foram tambem criadas as classes `T1_cliente` e `T1_server`.

2 Implementação

2.1 Cliente

Para a aplicação cliente foram implementados os seguintes métodos:

- `printLista`
- `showMenu`
- `main`

2.1.1 printLista

Este método recebe como argumentos um `ArrayList<String>` com o número de vacinados e número de efeitos secundários por tipo de vacina, posteriormente realizando o `output` para o utilizador.

2.1.2 showMenu

Este método tem como funcionalidade, mostrar todos os comandos para realizar todas as funcionalidades ao utilizador, e como executá-las.

2.1.3 main

No método `main` são recebidos os inputs do utilizador, e consoante o comando introduzido, são invocados remotamente, os métodos referentes aos mesmos da classe remota.

2.2 Servidor

O servidor faz a conexão à base de dados e faz o bind com o serviço e o objeto remoto.

2.3 Vacinacao

Esta é a `interface` remota da classe remota, que tem a declaração dos métodos da classe `vacinacaoImpl`. Cada método será explicado indivualmente no ponto `2.4`.

2.4 VacinacaoImpl

2.4.1 Construtor

O construtor recebe como argumento o `Statement` referente à base de dados, de modo a realizar as `querys` e os `updates`.

2.4.2 getCentrosVacinacao

Este método não recebe nenhum argumento, e devolve uma `ArrayList` que contém o o nome de todos os centros de vacinação na base de dados.

2.4.3 getTamanhoFilaEspera

Este método recebe o nome de um centro de vacinação, e devolve o número de pessoas em fila de espera.

2.4.4 inscricaoVacinacao

Este método tem como objetivo inserir um utilizador na fila de espera, recebe o nome do centro de vacinação, o cartão de cidadão, o nome, a idade e o género do utilizador. Sendo que por fim, devolve um código único de vacinação ao utilizador.

2.4.5 registarVacinacao

Este método tem como funcionalidade registar a vacinação dada a um utilizador, é recebido como argumentos, o código referente à vacinação e o tipo de vacina administrado. O utilizador é removido da fila de espera e adicionado à tabela de vacinados.

2.4.6 registarEfeitosSecundarios

Este método recebe um código de vacinação e uma descrição dos efeitos secundários do utilizador. Este efeito é adicionado à tabela de efeitos secundários e associado ao código de vacinação.

2.4.7 getDadosVacinados

Este método é repetido, com e sem argumentos:

- O método com argumentos, recebe o tipo de vacina, e devolve o número de vacinados e número de efeitos secundários associados a esse tipo de vacina.
- O método sem argumentos, vai guardar todos os tipos de vacinas, e iterando por cada um deles, invoca o método com o mesmo nome, usando o tipo de vacina como argumento. Por fim devolve todos os dados associados a todos os tipos de vacina.

3 Comandos

3.1 Criar Base de Dados

Os seguintes comandos são referentes à criação das tabelas da base de dados:

```
create table centros(nomeC varchar(128) primary key);

create table pessoa (cc varchar(12) primary key,
nomeP varchar(128), idade integer, genero char(1));

create table fila(nomeC varchar(128), cc varchar(12),
codigo varchar(4) primary key, foreign key (nomeC) references
centros, foreign key (cc) references pessoa);

create table vacinados ( codigo varchar(4) primary key,
data timestamp without time zone, modeloVac varchar(50));

create table efeitoS (codigo varchar(4), descricao varchar(500),
modeloVac varchar(50), primary key (codigo, descricao),
foreign key (codigo) references vacinados);
```

3.2 Compilar

```
$ javac -d build/classes/ src/t1/*java
```

3.3 RMI

```
$ rmiregistry -J-classpath -Jbuild/classes (Port)
```

3.4 Servidor

```
$ java -classpath build/classes:resources/postgresql.jar
t1.T1_server (Port) (Host) (DataBase) (User) (Password)
```

3.5 Cliente

```
$ java -classpath build/classes t1.T1_client (Host) (Port)
```

4 Conclusão

Após a realização deste trabalho, conseguimos verificar a importância e a praticabilidade de usar JavaRmi na interação cliente-servidor, como também uma base de dados para armazenamento persistente.

Gostariamos de salientar uma situação, como não estava descrito no enunciado um comando para adicionar centros de vacinação, e como não nos pareceu coerente o utilizador conseguir adicionar centros de vacinação, estes têm de ser adicionados diretamente através dos comandos da base de dados.

Por fim, consideramos que este trabalho se encontra bem sintetizado, organizado e simples, demonstrando objetivamente os conceitos e conteúdos interiorizados no decorrer das aulas.