Especificação do Trabalho II – ELC1018 Sistemas Distribuídos
Programação de uma aplicação Sala de Chat utilizando Java RMI
Data de entrega (código e documentação) e apresentação: 30/05/2019
Trabalho deve ser realizado em dupla

O trabalho consiste em implementar uma aplicação **Sala de Chat** utilizando Java RMI.

A aplicação distribuída Sala de Chat deve ter um servidor central (classe `ServerChat`) e permitir diversos clientes (classe `UserChat`) e diversas salas (classe `RoomChat`).

Requisitos funcionais absolutos

RFA1: O servidor central deve manter uma lista de salas (`roomList`). A lista de salas deve declarada como `private arrayList<String> roomList`.

RFA2: Cada sala (`RoomChat`) da `roomList` deve manter uma lista de usuários (`userList`). A lista de usuários deve declarada como `private Map<String, IUserChat> userList`.

RFA3: No servidor central, não deve haver limite de salas, tampouco de usuários por sala.

RFA4: No início, todo cliente, identificado pelo seu nome (`usrName`), deve contatar o servidor e solicitar a lista de salas `roomList`.

RFA5: A solicitação da lista de salas deve ser realizada através da invocação ao método remoto `getRooms()` da lista de salas `roomList`.

RFA6: A lista de salas deve ser exibida na interface do usuário (GUI), para permitir a escolha da sala.

RFA7: Sempre que um usuário desejar entrar numa sala já existente ele deve solicitar a referência ao objeto remoto ao RMI Registry usando o nome da sala e, após conhecer o objeto, deve invocar o método remoto `joinRoom(String usrName)` da respectiva sala.

RFA8: Caso o usuário não encontre no servidor a sala desejada ele deve poder solicitar a criação de uma nova sala. Isto deve ser feito através da invocação ao método remoto `criateRoom(String roomName)` do servidor. A vinculação do usuário a esta sala não deve ser automática. Ele deve solicitar a entrada invocando o método remoto `joinRoom()` da sala.

RFA9: Após pertencer a uma sala, o usuário deve enviar mensagens de texto à sala através da invocação ao método remoto `sendMsg(String usrName, String msg)` da sala.

RFA10: Para receber mensagens, o processo do usuário deve implementar um método remoto `deliverMsg(String senderName, String msg)`.

RFA11: O controlador da sala (sala) é quem deve controlar o envio das mensagens aos membros da sala.

RFA12: Os usuários devem sair da sala invocando o método remoto `leaveRoom(String usrName)` da sala.

RFA13: Uma sala só deve poder ser fechada pelo servidor. O servidor deve fechar a sala invocando o método remoto `closeRoom()` do controlador de sala. Caso haja usuários na sala, antes de ser finalizado o controlador da sala deve enviar uma mensagem "*Sala fechada pelo servidor.*" aos usuários.

RFA14: Após fechar a sala o servidor deve eliminar a sala da lista de salas. Cada usuário deve fazer o mesmo ao receber a mensagem "*Sala fechada pelo servidor.*" do controlador.

RFA15: A formatação da GUI para o usuário e servidor é de livre escolha, mas deve contar no mínimo com um quadro para visualização das mensagens, com a possibilidade de seleção da sala pelo usuário e servidor, além de conter botões apropriados para os principais comandos (Send, Close, Join e Leave).

RFA16: O servidor deve ser registrado no registro de RMI (rmiregistry) com o nome "Servidor" e usar a porta "2020" para escutar clientes. O registro deve executar na máquina do servidor.

RFA17: As classes do servidor, usuário e controlador de sala devem implementar as interfaces `IServerChat`, `IUserChat` e `IRoomChat`, respectivamente.

```
public interface IServerChat extends java.rmi.Remote {
     public roomList getRooms();
     public void createRoom(String roomName);
}

public interface IUserChat extends java.rmi.Remote {
     public void deliverMsg(String senderName, String msg);
}

public interface IRoomChat extends java.rmi.Remote {
     public void sendMsg(String usrName, String msg);
     public void joinRoom(String usrName, IUserChat user);
     public void leaveRoom(String usrName);
     public void closeRoom();
     public String getRoomName();
}
```