UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG
ENGENHARIA DE COMPUTAÇÃO

IGOR OLIVEIRA DE SOUSA - 118301
THIAGO JANSEN SAMPAIO COSTA - 116502

RELATÓRIO

Implementação de um Chat usando Sockets em Java

RIO GRANDE
2020

IGOR OLIVEIRA DE SOUSA - 118301
THIAGO JANSEN SAMPAIO COSTA - 116502

RELATÓRIO
Implementação de um Chat usando Sockets em Java

Trabalho apresentado à disciplina de Sistemas Distribuídos como requisito para obtenção de nota.

Prof. Bruno Lopes Dalmazo

RIO GRANDE
2020

1. INTRODUÇÃO

O trabalho consiste na implementação em Java de um chat cliente-servidor usando sockets, no qual permite diversos usuários realizarem um bate-papo em tempo real utilizando threads. A troca de mensagens é centralizada no servidor, sendo assim todas as mensagens enviadas pelos usuários do chat são enviadas primeiramente ao servidor e, por fim, o servidor fica responsável por repassá-las aos usuários envolvidos no chat.

Este relatório visa explicar detalhadamente nossa aplicação. O documento inclui decisões de projeto, detalhes da implementação, interface gráfica e um pequeno manual.

2. DECISÕES DE PROJETO

Para o desenvolvimento da aplicação, foi utilizada a linguagem de programação Java e interface Swing. O Java foi escolhido devido a sua grande comunidade e proativa com diversos tipos de documentação, além de ter fácil manuseio do sistema de threads. Enquanto o Swing, pela praticidade na criação da interface numa dinâmica de “arrastar e soltar”.

3. IMPLEMENTAÇÃO

Para que haja conexão entre usuários foram utilizados sockets para realizar conexão, no qual é necessário obter o endereço IP do servidor e o número da porta.

Foram criadas duas aplicações: uma chamada *Client,* responsável por enviar e receber mensagens do servidor, e uma segunda chamada *Server*, responsável pela conexão entre os usuários, recebendo e enviando as mensagens.

3.1. SERVIDOR

No que diz respeito à implementação do lado do servidor, foi desenvolvido utilizando conhecimentos básicos do funcionamento de sockets. Ao receber uma requisição do cliente, é chamado o construtor, onde é inicializado o buffer referente ao socket, sendo que cada nova instância do servidor é inicializada em uma nova thread. Em seguida, o servidor fica rodando na função “run”, responsável pela checagem dos canais de entradas, e chamando a função “sendToAll” ao receber uma requisição de nova mensagem, ou desconectando clientes quando requisitado.

```
public void sendToAll(BufferedWriter bufWriOut, String msg) throws
IOException {
      for(BufferedWriter bw : Users){
       if(!(bufWriOut == bw)){
         bw.write(nameUser + ": " + msg+"\r\n");
         bw.flush();
       }
      }
    }
```

Código 1. Função sendToAll().

A “sendToAll” realiza a tarefa de enviar a todos os clientes conectados a nova mensagem (menos o cliente que enviou tal mensagem).

3.2. CLIENTE

No cliente, como dito anteriormente, foi utilizado o framework Swing para a interface, com isso a implementação vem com partes já geradas pelo o próprio framework. No entanto, não é apenas nisso que ele interfere, para que seja viável, é necessário a criação de um arquivo onde toda a implementação decorrerá, sobrando ao arquivo “main” do projeto “mainClient” a tarefa de unicamente inicializar utilizando o atributo “setVisible(true)”, conforme o código 2.

```
java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
           new Client().setVisible(true);
        }
     });
```

Código 2. Função main.run().

O exemplo do código 2 está implementado na função “main” do arquivo “mainClient”.

Já o outro arquivo da implementação conterá todas as funções necessárias para o funcionamento. Primeiro, tem a função “connectServer” que recebe como parâmetros o nome de usuário, ip e a porta passadas pelo o usuário, a partir disso inicia a conexão do socket local com o servidor, criando uma primeira requisição com o nome do usuário que será guardado pelo servidor. Tal método com auxílio da função “sendMsg” passa para a interface de todos os usuários a mensagem: "Você

foi conectado com sucesso!!!", e ao servidor a mensagem: "Entrei!!!", podendo ser observado na Imagem 1. Ao receber a requisição oriunda do botão enviar a função sendMsg entra em funcionamento realizando a simples tarefa de enviar para a interface e servidor a mensagem digitada pelo usuário, como observado no código 3.

```
public void sendMsg(String msg) throws IOException{
        if(this.first){
            this.writerBuf.write("Entrei !!\r\n");
            this.feedMsg.append( "Você foi conectado com
sucesso!!!\r\n");
            this.writerBuf.flush();
            this.first = false;
        }else{
            this.writerBuf.write(msg+"\r\n");
            this.feedMsg.append( this.inpName.getText() + ":
"+this.inpMsg.getText()+"\r\n");
            this.writerBuf.flush();}}
```

Código 3. Função sendMsg().

Por fim, tem a função "out" e a função "run", a primeira serve basicamente para finalizar a conexão fechando o socket, enquanto a outra foi necessário a implementação de uma classe dentro do corpo da classe atual para poder acrescentar a funcionalidade "Runnable", fazendo com que seja possível receber as mensagens enviadas pelo servidor enquanto estiver conectado.

4. INTERFACE

Para a construção da interface foi utilizado o framework Swing que disponibiliza um conjunto de elementos gráficos para ser utilizado na plataforma Java. Conforme imagens abaixo:

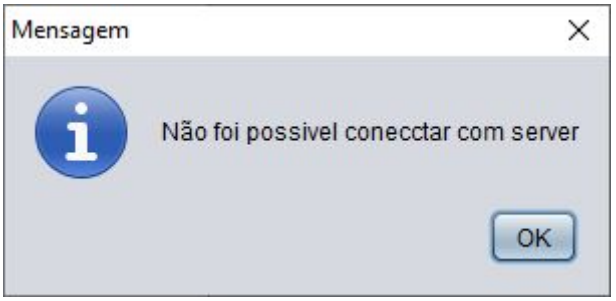

Imagem 1. Aviso para caso a conexão não tenha sido estabelecida.

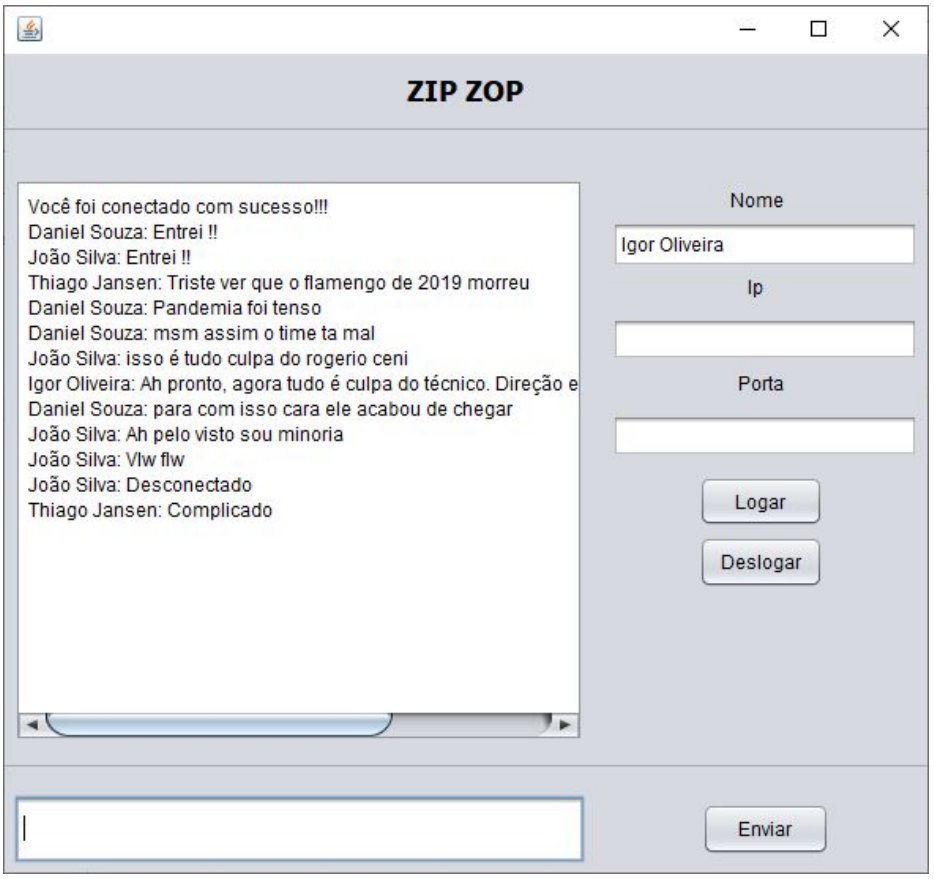

Imagem 2. Interface do chat.

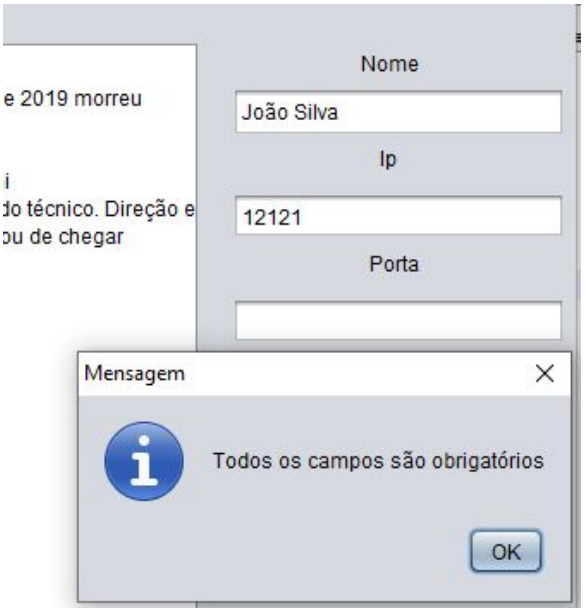

Imagem 3. Aviso para caso não o usuário não preencha todos os campos.

5. COMO RODAR

Recomendamos a utilização do NetBeans IDE para rodar nossa aplicação.

Para inicializar nossa aplicação, é necessário rodar a aplicação do servidor chamada “Server” e informar a porta que deseja colocar o servidor, conforme a Imagem 4.

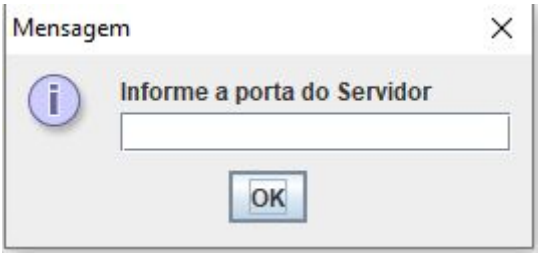

Imagem 4. Tela de solicitação da porta do servidor

Após inicializar o servidor, deve-se inicializar a aplicação cliente chamada “mainClient” e informar seu nome, IP e a porta do servidor. O IP local pode ser facilmente identificado utilizando o comando “ipconfig” no prompt de comando do Windows. Caso queira se conectar com usuários des outras redes, será necessário a realização de um desses dois métodos a seguir: 1º utilizando um software que simula uma rede local, como o famoso Hamachi; 2º o host do servidor terá que criar um virtual server abrindo assim o seu modem para acesso externo (a porta do virtual server deverá ser a mesma do servidor).

Recomendamos que deslogue o chat antes de fechar a janela para que a conexão seja desfeita da forma correta.