**Ministerul Educaţiei al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Raport**

Lucrare de laborator **№5**

Disciplina: Programarea in Retea

A îndeplinit: st.gr.FI-161 Panfil Nicolae

A controlat: Gavrilita Mihai

Chișinău 2019

**Lucrare de Laborator Nr.5**

**Scop:**

**1. Înțelegerea protocolului de transfer de date.**

**2. Inițializarea comunicarii intre Client-Server.**

**3. Efectuarea unui chat între clienți.**

**4. Conceptele de bază ale protocolului TCP.**

**Sarcina:**

Efectuarea aplicației Client-Server bazată pe Sockets.

Scopui acestei lucrari de laborator este de a efectua comunicarea între două părți ( Client – Server ) bazate pe Sockets. Un Socket este un punct final al comunicării bidirecționale între mașini. Există mai multe tipuri de Sockets:

* Stream Sockets - utilizează TCP.
* Datagram Sockets - utilizează UDP.
* Raw Sockets
* Sequenced Packet Socket

Prin urmare, am efectuat conexiunea prin TCP (Transmission Control Protocol), acest protocol va garanta că datele sunt recepționate de către receiver și vor trimite mesajul de confirmare înapoi către expeditor.

Pentru realizarea sarcinilor am două clase principale: Client si Server.

În clasa Client am implementat partea Client în acest chat și am efectuat conexiunea la server. Pentru conexiune din partea Client trebuie să știm: ip și port. Am inițializat câteva obiecte pentru a lucra cu transferul de mesaje sau de date. Apoi, am metoda ListenThread(), care are ca scop pornirea noului thread pentru fiecare client conectat la Server.

public class client\_frame extends javax.swing.JFrame

{

String username, address = "localhost";

ArrayList<String> users = new ArrayList();

int port = 2222;

Boolean isConnected = false;

Socket sock;

BufferedReader reader;

PrintWriter writer;

public void ListenThread()

{

Thread IncomingReader = new Thread(new IncomingReader());

IncomingReader.start();

}

public void userAdd(String data)

{

users.add(data);

}

public void userRemove(String data)

{

ta\_chat.append(data + " is now offline.\n");

}

public void sendDisconnect()

{

String bye = (username + ": :Disconnect");

try

{

writer.println(bye);

writer.flush();

} catch (Exception e)

{

ta\_chat.append("Could not send Disconnect message.\n");

}

}

Metodele userAdd(), userRemove() sunt pentru a lucra cu utilizatorii și a afișa informații despre utilizatorii conectați și numele fiecăruia. Metoda sendDisconnect() este pentru partea UI, aceasta este logica pentru butonul Disconnect , iar Disconnect() va închide threadul deschis și încă o abordare este că numele utilizatorului nu va fi schimbat, isConnected(false), tf\_username.setEditable(true).

public void Disconnect()

{

try

{

ta\_chat.append("Disconnected.\n");

sock.close();

} catch(Exception ex) {

ta\_chat.append("Failed to disconnect. \n");

}

isConnected = false;

tf\_username.setEditable(true);

}

IncomingReader() implementează Runnable, această metodă va funcționa cu toate mesajele primite de la clienți și de la server și toate mesajele vor apărea pe textView pe partea UI.

În continuare, în clasa Client sunt elementele bazate pe partea UI și legătura dintre UI și implementare.

Pe partea de server este implementarea pentru comportamentul serverului, serverul este realizat pentru a urma comunicarea dintre clienți.

La implementarea serverului, am o server class care începe implementarea cu ClientHandler() care implementează Runnable, aceasta înseamnă că această metodă va avea ceva legat de threads. Deci, această metodă va avea grijă de utilizatorii care se vor conecta la Server și vor crea pentru fiecare dintre un nou thread separat, ClientHandler per utilizator.

Apoi am metoda run() care vine automat din interfața Runnable, scopul acestei metode este de a afișa unele informații despre data, conexiuni la server și starea utilizatorului.

@Override

public void run()

{

String message, connect = "Connect", disconnect = "Disconnect", chat = "Chat" ;

String[] data;

try

{

while ((message = reader.readLine()) != null)

{

System.out.println("Received message: " + message);

ta\_chat.append("Received: " + message + "\n");

data = message.split(":");

System.out.println("Data: " + data);

for (String token:data)

{

System.out.println("Token: " + token);

ta\_chat.append(token + "\n");

}

if (data[2].equals(connect))

{

tellEveryone((data[0] + ":" + data[1] + ":" + chat));

userAdd(data[0]);

}

else if (data[2].equals(disconnect))

{

tellEveryone((data[0] + ":has disconnected." + ":" + chat));

userRemove(data[0]);

}

else if (data[2].equals(chat))

{

tellEveryone(message);

}

else

{

ta\_chat.append("No Conditions were met. \n");

}

}

}

Metoda ServerStart() va gestiona conexiunea la server și va furniza firele pentru fiecare client care se va conecta la server. Conexiunea este deschisă astfel încât Clientul vine la ServerStart și acest lucru va crea un nou thread pentru acest client.

public class ServerStart implements Runnable

{

@Override

public void run()

{

clientOutputStreams = new ArrayList();

users = new ArrayList();

try

{

ServerSocket serverSock = new ServerSocket(2222);

while (true)

{

Socket clientSock = serverSock.accept();

PrintWriter writer = new PrintWriter(clientSock.getOutputStream());

clientOutputStreams.add(writer);

Thread listener = new Thread(new ClientHandler(clientSock, writer));

listener.start();

ta\_chat.append("Got a connection. \n");

}

}

catch (Exception ex)

{

ta\_chat.append("Error making a connection. \n");

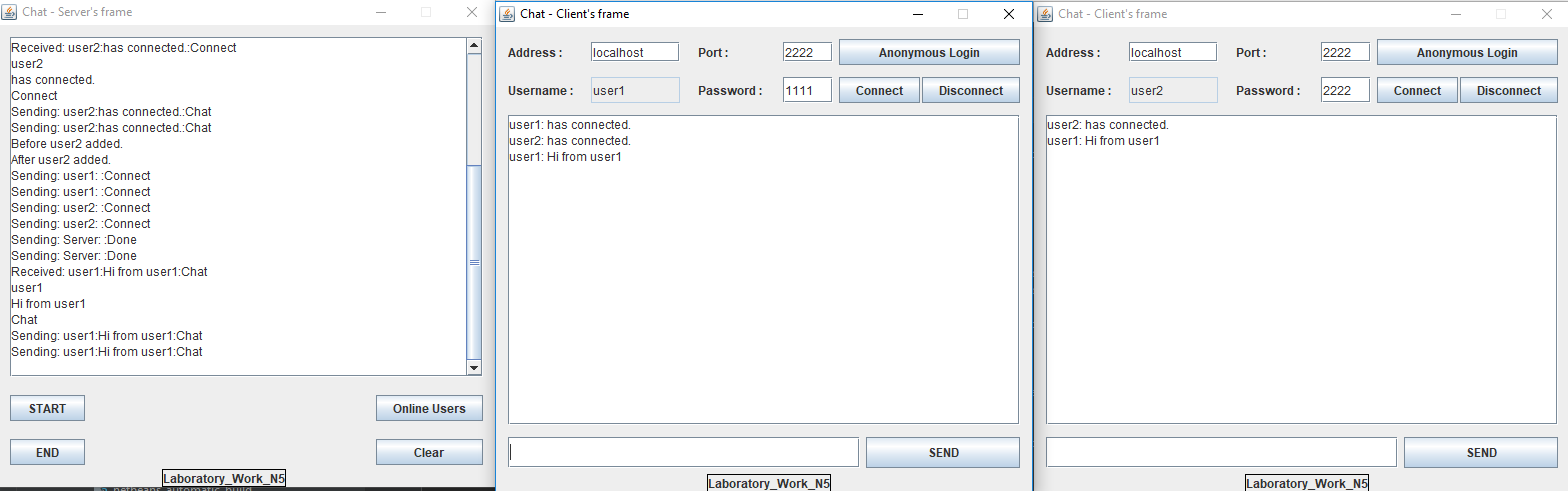
}

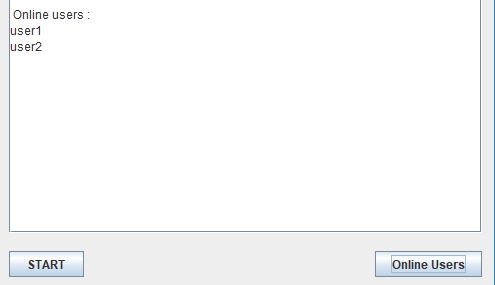
}

}

Următoarele sunt metodele pentru partea UI.

**Output:**

Pentru a porni aplicația, ar trebui mai întâi de rulat server\_frame și apoi client\_frame, dacă se va rula mai întâi client\_frame , server\_frame nu se va rula . De asemenea, pentru a efectua anumi[](https://user-images.githubusercontent.com/43058513/56769284-6efa0300-67b9-11e9-969f-b39b4597b3d6.PNG)ți clienți, trebuie să se ruleze din nou clasa client\_frame.

[](https://user-images.githubusercontent.com/43058513/56769282-6efa0300-67b9-11e9-80d6-6901013070ee.PNG)

**Concluzie:**

În această lucrare de laborator am dobândit abilități de lucru cu conexiune între două laturi (Client - Server )și câteva noțiuni de bază pe partea UI. In concluzie pot să spun că serverul are relația many-to-many cu clienții, deoarece are o mulțime de conexiuni, iar clientul are relația one-to-one cu serverul.