Logotipo

Descrição gerada automaticamente

**Curso de Ciência da Computação**

**UNIVERSIDADE PAULISTA**

**Gabriel Macedo Ramos RA:F281060**

**Guilherme Augusto Sbizero Correa RA:F235289**

**Matheus Moreira Dias RA: F2265B8**

**DESENVOLVIMENTO DE UMA FERRAMENTA PARA COMUNICAÇÃO EM REDE**

**SÃO PAULO – SP**

**2022**

**RESUMO**

Redes de computadores é um conjunto de dispositivos (normalmente conhecido como nós) conectados por links de comunicação. Um nó pode ser um computador, uma impressora ou outro dispositivo de envio e/ou recepção de dados, que estejam a outros nós da rede. Uma rede de computadores permite que o usuário compartilhe uma enorme quantidade de informações e enviem mensagens uns aos outros, incluindo textos, imagens, áudios e vídeos. A operacionalização de uma rede de computadores tem como objetivos básicos prover a comunicação confiável entre os vários sistemas de informação, melhorar o fluxo e o acesso às informações, bem como agilizar a tomada de decisões administrativas facilitando a comunicação entre seus usuários. O objetivo deste trabalho é avaliar de forma clara o conceito de redes, sua utilização e sua aplicabilidade, por meio de uma ferramenta de comunicação em rede desenvolvida na linguagem Java para atender as necessidades do Conselho Nacional do Meio Ambiente (CONAMA) que deseja saber quais atividades industriais estão gerando poluição do Rio Tietê desde sua nascente em Salesópolis (SP) até a sua passagem pela região da grande São Paulo. Para tal ela precisa trocar informações das equipes de inspetores treinados e capacitados que se revezarão dentro de cada indústria, controlando os processos e passando informações online para a Secretaria. O trabalho realizado teve caráter qualitativo, fundamentado em obras publicadas por estudiosos especialistas, que já apontaram os conceitos, usos, aplicações, desempenho e falhas de certos tipos de redes.

Palavras-chave: comunicação rede; Berkeley; TCP/IP; sockets.

**Sumário**

**1 Introdução**.......................................................................................................**4**

**2 Objetivo e motivação do trabalho**................................................................**6**

**3 Fundamentos da Comunicação em rede**....................................................**7**

3.1 Arpanet.....................................................................................................**8**

3.2 O que é TCP/IP........................................................................................**8**

3.3 O que são e como funcionam os sockets................................................**8**

3.4 Comunicação de Dados.........................................................................**11**

3.5 Componentes.........................................................................................**12**

3.6 Tipos de Conexão..................................................................................**14**

**4 Plano de desenvolvimento da Aplicação** ................................................**15**

**5 Projeto** ........................................................................................................**20**

**6 Relatório com as linhas de código do programa** ...................................**23**

**7 Apresentação do programa em funcionamento em computador** ........**49**

**8 Bibliografia** ................................................................................................**55**

**9 Ficha de Atividades Práticas Supervisionadas** ......................................**56**

# Introdução

Rede de computadores é um conjunto de computadores interligados entre si através de um cabo, rádio, satélite, entre muitos outros, de forma que se comuniquem e troquem informações entre si. Todo equipamento conectado a uma rede de computadores que tenha autonomia para transmitir e receber dados são chamados de Host ou hospedeiro. O host de origem prepara o pacote e envia para o host de destino que recebe o pacote da rede, assim criando uma comunicação entre host ou entre usuários de locais diferentes. Uma rede não se compõe sempre de apenas dois usuários, ela pode partir de uma simples rede doméstica a vários continentes. Uma rede de computadores ela pode ser:

• LAN (Local Area Network), que é uma rede local;

• MAN (Metropolitan Area Network), este tipo de rede abrange o espaço físico de uma cidade;

• WLAN (Wireless Local Area Network), rede através de uma onda de rádio de alta frequência;

• CAN (Campus Area Network), rede composta de vários edifícios;

• SAN (Storage Area Network), para recursos de armazenamento;

• PAN (Personal Area Network), pequena rede local (Doméstica);

• WANs (Wide Area Network), este tipo de rede abrange vários municípios e países. A comunicação de dados entre locais remotos pode ser realizada através de um processo denominado conectividade, que envolve desde a conexão de computadores, meios e dispositivos de rede. Em nosso trabalho, apresentaremos um exemplo de comunicação entre duas ou várias outras máquinas, através da arquitetura Cliente-servidor, utilizando protocolos TCP/IP e utilizando a API de Sockets de Berkeley. A camada cliente, neste trabalho, terá a função de prover a interface para que os usuários possam manipular as informações, ou seja, através dela realizase a interação entre o usuário e o sistema. É desenvolvida para se conectar diretamente ao banco de dados, tendo como responsabilidade fazer as solicitações dos dados necessários ao servidor, sendo que este os processa e 6 devolve o resultado. A Secretaria de Estado do Meio Ambiente, que deseja saber e controlar o nível das atividades industriais e domésticas que estão gerando poluição do Rio Tietê, utilizarão o programa no qual desenvolvemos, para trocar informações das equipes de inspetores dentro de cada indústria. O cliente (client), será responsável por prover a interface para que as equipes e dar às equipes todo o tipo de informação necessária e disponível. O servidor será responsável por armazenar as mensagens, cadastros, transmitir e receber informações das diversas equipes de inspetores que controlam os processos de poluição do Tietê e passando informações online para a Secretaria de Estado do Meio Ambiente.

1. **Objetivo e motivação do trabalho**

O objetivo do trabalho desse trabalho consistente em estudar e desenvolver uma ferramenta para comunicação em rede e toda teoria computacional envolvida. Após pesquisa bibliográfica sobre o assunto em questão o grupo de deverá criar uma aplicação que permita que duas ou mais pessoas possam se comunicar em uma rede, utilizando o protocolo TCP/IP. A utilização para a comunicação de dados implementando a ferramenta que foi utilizada as primitivas dos sockets de Berkeley, os quais serão implementados na linguagem de programação JAVA.

**3 Fundamentos da comunicação em rede**

A palavra rede tem várias definições. Aplicada aos computadores, rede é uma maneira de conectar computadores para que eles tenham consciência um do outro e possam unir e compartilhar seus recursos. (Nobrega Filho, 2016).

Entre os anos 70 e 80 ocorreu uma fusão dos campos de Ciência da Computação e Comunicação de Dados e isto trouxe vários fatos relevantes. Nas empresas, escolas e em muitos outros tipos de organização, as redes de comunicação de dados, em seus diversos tipos oferecem vários benefícios:

Permitir acesso simultâneo a programas e dados importantes;

Permitir às pessoas compartilhar dispositivos periféricos;

Facilitar o processo de realização de cópias de segurança (backup);

Agilizar as comunicações pessoais com correio eletrônico ou mensagens instantâneas;

Na área comercial, por exemplo, as redes revolucionaram o uso da tecnologia dos computadores. Muitas empresas que costumavam depender de um sistema centralizado em mainframe com uma série de terminais agora usam redes de computadores, em que cada empregado tem um computador em sua mesa. A tecnologia e a perícia dos computadores não estão mais centralizadas no mainframe de uma companhia de e em seu departamento de tecnologia, elas estão distribuídas por toda a organização, entre uma rede de computadores e usuários preparados. (Nobrega Filho, 2016).

A indústria da computação, por assim dizer, teve um processo espetacular em um curto período e as redes de comunicação também fazem parte desse crescimento tendo um papel fundamental nos processos de comunicação para o mundo globalizado. As redes de comunicação, cada vez mais rápidas e eficiente, permitiram a comunicação e o acesso rápido a qualquer parte do globo de forma praticamente instantânea.

**3.1 Arpanet**

A Arpanet foi a primeira rede computadores, construída em 1969 como um meio robusto para transmitir dados militares sigilosos e para interligar os departamentos de pesquisa de todos os Estados Unidos. Arpanet primeiro executou NCP (Network Control Protocol em português Protocolo de Controle de Rede) e posteriormente a primeira versão do protocolo de internet ou conjuntos de protocolos TCP/IP.

**3.2 O que é TCP/IP**

O TCP/IP é o principal protocolo de envio e recebimento de dados TCP significa Transmission Control Protocol (Protocolo de Controle Transmissão) e o IP Internet Protocol (Protocolo de Internet).

Protocolo é uma espécie de linguagem utilizada para que dois computadores consigam de comunicar. Por mais que duas máquinas estejam conectadas à mesma rede, se não “falarem” a mesma língua, não há como estabelecer uma comunicação. Então, o TCP/IP é uma espécie de idioma que permite às aplicações conversarem entre si.

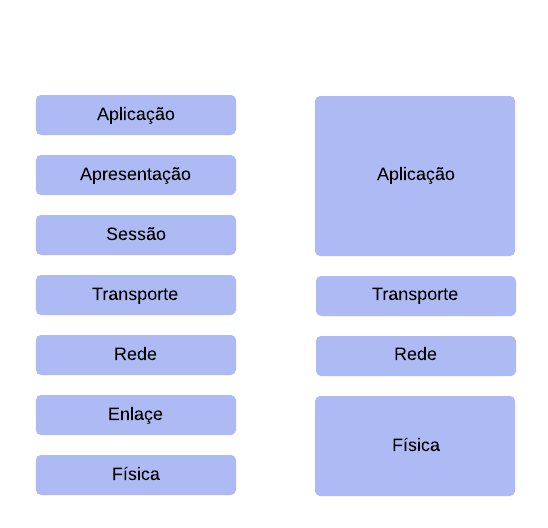
**3.3 O que são e como funcionam os sockets**

Formalmente falando os sockets foram a forma de permitir que dois processos se comuniquem. Esses processos podem ou não estar na mesma máquina.

Diversas aplicações que utilizamos no dia a dia fazem uso de sockets para se comunicar. Nosso navegar web utiliza sockets para requisitar páginas. Quando um sistema se integra com um banco de dados abre um socket. Quando fazemos um SSH em um servidor estamos abrindo e utilizando um socket.

Originalmente a implementação dos sockets foi feita no BSD em 1983. Essa implementação foi portada par o Linux com poucas modificações. Na forma de uma API, os sockets abstraem a camada de rede para a que uma aplicação possa se comunicar com outra sem ter que se preocupar com detalhes da pilha TCP/IP que gere a rede abaixo dessa aplicação.

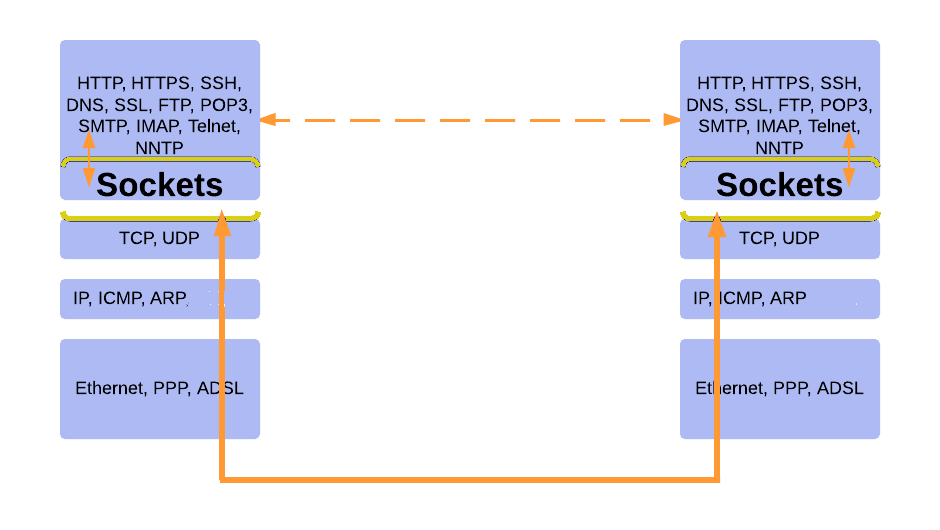
A internet utiliza como base para todas as suas comunicações a pilha TCP/IP que possui apenas 4 camadas. Baseado no modelo OSI, o TCP/IP é, atualmente, o padrão de comunicação em redes. A imagem abaixo apresenta uma comparação de modelo OSI com sete camadas em relação as respectivas 4 camadas do modelo TCP/IP:



Fonte: Blog Pantuza, 2017

Considerando a internet e o TCP/IP, os sockets estão entre a camada de transporte e a aplicação. Estando nesse ponto de intercessão, eles conseguem fazer uma interface entre aplicações e rede de maneira bem transparente. Assim, aplicações são implementadas através de uma comunicação lógica. Lógica no sentido de que para esses programas, eles estão se comunicando diretamente um com o outro, mas na prática eles estão passando pela rede para trocar mensagens.

A imagem a seguir mostra essa divisão lógica e física de comunicação dentro da pilha TCP/IP e onde exatamente entre as camadas de transporte e aplicação se encaixa a API de sockets.



Fonte: Blog Pantuza, 2017

Até agora vimos que os sockets foram criados na forma de uma API que possibilita aplicações/processos se comunicarem. Abaixo contém algumas das principais funções ao criar um programa utilizando sockets.

Texto

Descrição gerada automaticamente

fonte: DevMedia,2008.

Quando se programa utilizando sockets, uma arquitetura muito comum para esses programas, é utilizar o Cliente/Servidor (client/server). Para essa arquitetura temos que implementar um programa cliente e um programa servidor. Ambos fazem uso da mesma API sockets.

**3.4 Comunicação de Dados**

A comunicação de dados é uma disciplina da ciência da computação que se trata da comunicação entre computadores e dispositivos de calculadoras analógicas antigas sem utilização de nenhum protocolo do modelo OSI ou da arquitetura TCP/IP diferentes através de um meio de transmissão incomum.

A eficiência de um sistema de comunicação de dados depende fundamentalmente de três características:

1. Entrega: o sistema deve entregar os dados ao destino correto. Os dados devem ser recebidos somente pelo dispositivo ou usuário de destino.
2. Confiabilidade: o sistema deve garantir a entrega dos dados. Dados modificados ou corrompidos em uma transmissão são pouco úteis.
3. Tempo de atraso: o sistema deve entregar dados em um tempo finito e predeterminado. Dados entregues tardiamente são pouco úteis. Por exemplo, no caso de transmissões multimídia, como vídeo, os atrasos não são desejáveis, de modo que eles devem ser entregues praticamente no mesmo instante em que foram produzidos, isto é, sem atrasos significativos.

**3.5 Componentes**

Um sistema básico de comunicação de dados é composto de cinco elementos

**1. Mensagem:** é a informação a ser transmitida. Pode ser constituída de texto, números, figuras, áudio e vídeo.

**2. Transmissor:** é o dispositivo que envia a mensagem de dados. Pode ser um computador, uma estação de trabalho, um telefone, uma câmera de vídeo, etc.

**3. Receptor:** é o dispositivo que recebe a mensagem. Pode ser um computador, uma estação de trabalho, uma câmera de vídeo e assim por diante.

**4. Meio:** é o caminho físico por onde viaja uma mensagem originada e dirigida ao receptor.

**5. Protocolo:** é um conjunto de regras que governa a comunicação de dados. Ele representa um acordo entre os dispositivos que se comunicam.

Gráfico

Descrição gerada automaticamente com confiança baixa

Fonte: Brasil Escola,2022.

**Direção do fluxo de dados:** Uma comunicação entre dois dispositivos pode acontecer de três maneiras diferentes: simplex, Half-duplex ou full duplex.

**Simplex:** no modo simplex, s comunicação é unidirecional, como em uma rua de mão única. Somente um dos dois dispositivos no link é capaz de transmitir; logo o outro só será capaz de receber.

Texto

Descrição gerada automaticamente

Fonte: Brasil Escola,2022.

**Half-duplex:** Neste modo, cada estação pode transmitir e receber, mas nunca ao mesmo tempo. Quando um dispositivo está transmitindo o outro está recebendo e vice-versa. Em uma transmissão Half-duplex, toda a capacidade do canal é dada ao dispositivo que estive transmitindo no momento.

Texto

Descrição gerada automaticamente com confiança média

Fonte: Brasil Escola,2022.

**Full-duplex:** Neste modo, ambas as estações podem transmitir e receber simultaneamente. Sinais em direções opostas compartilham a capacidade do link ou canal.

Interface gráfica do usuário, Diagrama, Aplicativo

Descrição gerada automaticamente

Fonte: Brasil Escola,2022.

**3.6 Tipos de Conexão**

**Ponto a ponto:** proporciona um link dedicado entre dispositivos.

Texto

Descrição gerada automaticamente com confiança média

Fonte: Brasil Escola,2022.

**Multi-ponto:** É aquela na qual mais de dois dispositivos compartilham um único link

Diagrama

Descrição gerada automaticamente

Fonte: Brasil Escola,2022.

**4 Plano de desenvolvimento da aplicação**

Seguindo os tópicos de desenvolvimento do trabalho, onde primeiramente pesquisamos sobre sockets de Berkeley. Esse programa foi concebido para suportar diversos tipos de protocolos, logo ela suporta diversos formatos de dados, onde são armazenados em uma estrutura especial genérica. Para cada família de protocolos, existem estruturas mais especificas e adequadas, como o protocolo IP, que vai usar uma estrutura que pertence à família AF\_NET, que contêm também os protocolos UDP, TCP e ICMP, facilitando assim o desenvolvimento do programa.

Pesquisamos também sobre os protocolos que utilizaríamos no desenvolvimento do trabalho, pesquisamos sobre o protocolo TCP, que é um dos principais protocolos da camada da internet do modelo TCP/IP, que é um dos principais protocolos da camada da internet de modelo TCP/IP, ele permite gerenciar os dados vindo da camada inferior, ou seja, o protocolo IP. Quando os dados são liberados ao protocolo IP, este guarda em datagramas IP (datagrama é o nome dado a uma mensagem sem conexão e sem confirmação). Ele funciona entregando em ordem (já que utiliza a transmissão de socket baseado em conexões) os datagramas do protocolo IP, ele verifica o fluxo de dados para o evitar uma saturação na rede e, permitindo o início e o fim de uma comunicação de maneira correta, onde ele é um dos principais protocolos que consiste esse programa. Temos também o protocolo IP, que foi bem pesquisado, já que assim com o TCP faz parte dos principais protocolos da camada de internet, pois permite a composição e o transporte dos datagramas, assegurando a entrega de pacotes de dados, onde determina o destinatário da mensagem graças a três campos que são, o de endereço IP, máscara de sub rede gateway, sendo um dos principais protocolos que fizeram parte do trabalho.

Pesquisamos também sobre diversos códigos que viriam a ajudar a trazer um trabalho mais completo, trazendo assim todo o potencial da comunicação entre usuários na mesma rede, onde não ocorresse nenhuma falha. Essa etapa do trabalho foi realizada por todos.

A maior parte dos testes foi realizado por pelo integrante do grupo: Guilherme Augusto, sendo todos eles realizado em sua residência, onde foram estabelecidos um computados com cliente e o outro como cliente/servidor, assim testamos as mensagens de textos, onde funcionaram com êxito, testamos também a conexão entre um computador e outro onde também funcionou com êxito. Os testes seguirão até a entrega do trabalho, para evitarmos futuras falhas realizada por todos.

A parte escrita segue o cronograma que foi passado para os grupos, onde consiste em 6 etapas, objetivo do trabalho, introdução, fundamentos da comunicação de dados em rede, plano de desenvolvimento da aplicação, projeto (estrutura) do programa e relatório com linhas de código.

**5 Projeto**

Precisávamos de uma ferramenta que fosse possível realizar a comunicação dos membros do Conselho Nacional do Meio Ambiente (CONAMA), com o objetivo de trocar informações via rede local. Com isto, foi desenvolvido um software que tornou possível a troca de mensagens a partir do local host, sendo possível interagir com outros usuários.

Para estabelecermos um servidor local, foi utilizado sockets que são compostos por um conjunto de primitivas do sistema operacional e foram originalmente desenvolvidos para o BSD Linux.

Foi concebida uma interface gráfica onde o usuário poderá interagir e iniciar o servidor ou se conectar a um servidor que esteja ativo. Ao iniciar um servidor é possível definir qual porta TCP/IP estará disponível para aceitar novas conexões. Após a inicialização do servidor, ele estará disponível para que os usuários façam login.

Para que seja feita a conexão com sucesso, o usuário que deseja se conectar, deverá informar o IP do servidor da rede local e informar a porta que deseja se conectar, do contrário não será possível estabelecer a conexão, inviabilizando a troca de mensagens na ferramenta. O usuário poderá definir um usuário de identificação na ferramenta, que será exibido na troca de mensagens.

Dito isto, foi desenvolvido diversas classes para manter o devido funcionamento da ferramenta, como também, tratar os devidos erros que possam surgir, além de tornar a ferramenta intuitiva a partir da interface gráfica. Foi utilizado a classe JFrame do pacote Swing do JAVA para construção da interface do usuário, possibilitando criar as janelas, botões e as devidas interações na ferramenta.

A estrutura da ferramenta acontece da seguinte maneira, no início da execução, o usuário é apresentado a duas possibilidades, Servidor e Cliente:

**5.1 Cliente**

O cliente é a parte que interage com o usuário, é considerada a estação de trabalho em nosso projeto

Interface gráfica do usuário

Descrição gerada automaticamente

fonte: próprio autor.

Seguindo por Servidor, o programa irá chamar pela classe SetupServer que definirá os parâmetros para inicialização do servidor local. Após a definição desses parâmetros, iniciará o método responsável por receber esses parâmetros e partir dos sockets mencionados anteriormente, iniciar o servidor e deixá-lo disponível para receber novas conexões na porta definida.

Após esse processo de inicialização do servidor, é possível retornar a tela inicial da ferramenta e seguir para a opção Cliente, o programa ficará responsável por chamar pela classe Login onde será feito o login do Cliente para iniciar a troca de mensagens. Será necessário informar o servidor ativo e a respectiva porta, além de um usuário de identificação na ferramenta. Feito o login, o usuário estará apto a trocar mensagens com os demais usuários, em tempo real. Nesse ambiente, é possível a conexão e troca de mensagens de diversos usuários diferentes ao mesmo tempo. Não há um limite de conexões simultâneas definido, até que o servidor seja devidamente encerrado, continuará a aceitar novas conexões.

Após realizado o login, será iniciado a classe Cliente responsável pela conexão do usuário ao servidor e pela para troca de mensagens entre usuários. Nesse momento, o usuário poderá trocar mensagens com os demais usuários conectados na ferramenta. Caberá ao usuário a encerrar a conexão com o servidor e encerrar o bate-papo. Como mencionado anteriormente, o encerramento da conexão com o servidor por parte dos usuários não afetará na disponibilidade do servidor, até que ele seja encerrado novos usuários poderão se conectar e utilizar da ferramenta para trocar dados e informações.

Essa conexão entre usuário e servidor, se deve aos sockets do tipo TCP/IP que viabilizam essa comunicação entre processos através de uma rede computadores. Sendo possível garantir a ordem dos pacotes, considerado confiável e sem perdas.

Abaixo temos uma representação desse fluxo:

Diagrama

Descrição gerada automaticamente

fonte: próprio autor.

Nessa representação é possível notar a constante conexão do Cliente com o Servidor, o que possibilita a troca de dados e informações com os demais usuários conectados a mesma rede interna, garantindo a segurança e integridade dos dados e usuários.

**6 Relatório com as linhas de código do programa**

**Run.java**

package main;

import chatform.Start;

public class Run {

public static void main(String[] args) {

// TODO Auto-generated method stub

Start.main(null);

}

}

**Server.java**

package connection;

import java.io.\*;

import java.net.Socket;

import java.util.ArrayList;

public class Server extends Thread {

private static ArrayList<BufferedWriter> clients = new ArrayList<BufferedWriter>();

private static ArrayList<String> users = new ArrayList<String>();

private Socket connection;

private BufferedReader bufferReader;

private String currentUser = "";

public Server(Socket connection) {

this.connection = connection;

try {

// Reader

bufferReader = new BufferedReader(new InputStreamReader(connection.getInputStream()));

} catch (IOException e) {

e.printStackTrace();

}

}

@Override

public void run() {

try {

BufferedWriter bufferWriter = new BufferedWriter(new OutputStreamWriter(this.connection.getOutputStream()));

clients.add(bufferWriter);

currentUser = this.bufferReader.readLine();

users.add(currentUser);

String msg = "Text&" + currentUser + " Conectado";

while (!("Text&Disconnect " + currentUser).equalsIgnoreCase(msg) && msg != null) {

broadCast(msg);

System.out.println(currentUser + " [Server(run)] " + msg);

msg = this.bufferReader.readLine();

}

removeUser(currentUser);

broadCast("Text&Usuário " + currentUser + " Desconectado");

} catch (Exception e) {

e.printStackTrace();

removeUser(currentUser);

}

}

private void broadCast(String msg) {

for (BufferedWriter bufferClient : clients) {

try {

bufferClient.write(msg + "\r\n");

System.out.println("[Broadcast] " + msg);

bufferClient.flush();

} catch (Exception e) {

e.printStackTrace();

}

}

}

private void removeUser(String user) {

int index = users.indexOf(user);

clients.remove(index);

users.remove(index);

}

}

**Start.java**

package chatform;

import java.awt.Font;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.awt.event.WindowListener;

import java.awt.event.ActionEvent;

import java.awt.Color;

public class Start extends JFrame {

/\*

\* Variáveis da classe Start

\*/

private static final long serialVersionUID = 5602644222765201727L;

private final JPanel contentPanel = new JPanel();

/\*

\* Executa a aplicação

\*/

public static void main(String[] args) {

try {

Start dialog = new Start();

dialog.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

/\*

\* Definido as propriedades dos componentes utilizado e ação dos botões

\*/

public Start() {

getContentPane().setBackground(Color.BLACK);

setForeground(Color.BLACK);

setBackground(Color.WHITE);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 450, 300);

getContentPane().setLayout(null);

contentPanel.setBackground(Color.WHITE);

contentPanel.setBounds(0, 0, 434, 261);

contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));

getContentPane().add(contentPanel);

contentPanel.setLayout(null);

setLocationRelativeTo(null);

setTitle("Aplicação de Conversa");

{

JButton btnServidor = new JButton("Servidor");

btnServidor.setBackground(new Color(72, 209, 204));

btnServidor.setFont(new Font("Arial Black", Font.PLAIN, 11));

btnServidor.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

SetupServer server = new SetupServer();

server.setVisible(true);

Start.this.dispose();

}

});

btnServidor.setBounds(10, 126, 182, 61);

contentPanel.add(btnServidor);

}

{

JButton btnCliente = new JButton("Cliente");

btnCliente.setBackground(new Color(72, 209, 204));

btnCliente.setFont(new Font("Arial Black", Font.PLAIN, 11));

btnCliente.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

SetupServer s1 = new SetupServer();

Login login = new Login();

login.setVisible(true);

}

});

btnCliente.setBounds(242, 126, 182, 61);

contentPanel.add(btnCliente);

}

{

JLabel lblInicializar = new JLabel("Como deseja inicializar o programa?");

lblInicializar.setForeground(Color.BLACK);

lblInicializar.setBounds(67, 40, 305, 34);

contentPanel.add(lblInicializar);

lblInicializar.setFont(new Font("Arial Black", Font.PLAIN, 15));

}

}

}

**SetupServer.java**

package chatform;

import connection.Server;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import java.awt.Font;

import javax.swing.JTextField;

import javax.swing.JButton;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.awt.CardLayout;

import java.net.InetAddress;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.UnknownHostException;

import java.awt.Color;

public class SetupServer extends JFrame {

/\*\*

\* Server variáveis

\*/

private static ServerSocket server;

/\*\*

\* Form variáveis

\*/

private static final long serialVersionUID = 4998717362394143017L;

private JPanel contentPane;

private JTextField inputPort;

private JButton btnOk;

private JButton btnBack;

private JPanel panelConfig;

private JPanel panelStatus;

private JLabel lblIp;

private JLabel lblPort;

private JLabel lblValueIP;

private JLabel lblValuePort;

private JButton btnStopConnection;

private InetAddress inetAddress;

private String hostaddress;

private JButton btnBack\_1;

private String localhost = "";

/\*\*

\* Executa a aplicação

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

SetupServer frame = new SetupServer();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Aqui é definido as propriedades dos componentes utilizados nessa tela

\* além de definir a ação dos botões

\*/

public SetupServer() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 366, 158);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(new CardLayout(0, 0));

this.setLocationRelativeTo(null);

this.setTitle("Aplicação de Conversa (Servidor)");

panelConfig = new JPanel();

contentPane.add(panelConfig, "panelConfig");

panelConfig.setLayout(null);

JLabel lblNumeroDaPorta = new JLabel("Número da porta:");

lblNumeroDaPorta.setBounds(10, 35, 113, 18);

panelConfig.add(lblNumeroDaPorta);

lblNumeroDaPorta.setFont(new Font("Arial Black", Font.PLAIN, 11));

inputPort = new JTextField();

inputPort.setBounds(135, 35, 86, 20);

panelConfig.add(inputPort);

inputPort.setText("");

inputPort.setColumns(10);

btnOk = new JButton("Ok");

btnOk.setBackground(new Color(72, 209, 204));

btnOk.setFont(new Font("Arial Black", Font.PLAIN, 11));

btnOk.setBounds(90, 86, 73, 23);

panelConfig.add(btnOk);

btnOk.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

startServer();

}

});

btnBack = new JButton("Voltar");

btnBack.setFont(new Font("Arial Black", Font.PLAIN, 11));

btnBack.setBackground(new Color(72, 209, 204));

btnBack.setBounds(177, 86, 73, 23);

panelConfig.add(btnBack);

panelStatus = new JPanel();

contentPane.add(panelStatus, "panelStatus");

panelStatus.setLayout(null);

lblIp = new JLabel("IP:");

lblIp.setFont(new Font("Arial Black", Font.PLAIN, 15));

lblIp.setBounds(38, 11, 46, 14);

panelStatus.add(lblIp);

lblPort = new JLabel("Porta:");

lblPort.setFont(new Font("Arial Black", Font.PLAIN, 15));

lblPort.setBounds(38, 36, 57, 14);

panelStatus.add(lblPort);

lblValueIP = new JLabel();

lblValueIP.setForeground(new Color(0, 100, 0));

lblValueIP.setFont(new Font("Arial Black", Font.PLAIN, 13));

lblValueIP.setBounds(67, 11, 171, 14);

panelStatus.add(lblValueIP);

lblValuePort = new JLabel();

lblValuePort.setForeground(new Color(0, 100, 0));

lblValuePort.setFont(new Font("Arial Black", Font.PLAIN, 13));

lblValuePort.setBounds(97, 36, 121, 14);

panelStatus.add(lblValuePort);

btnStopConnection = new JButton("Encerrar Conexão");

btnStopConnection.setBackground(new Color(72, 209, 204));

btnStopConnection.setFont(new Font("Arial Black", Font.PLAIN, 11));

btnStopConnection.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

System.exit(0);

}

});

btnStopConnection.setBounds(38, 79, 144, 30);

panelStatus.add(btnStopConnection);

btnBack\_1 = new JButton("Voltar");

btnBack\_1.setFont(new Font("Arial Black", Font.PLAIN, 11));

btnBack\_1.setBackground(new Color(72, 209, 204));

btnBack\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

SetupServer.this.dispose();

Start start = new Start();

start.setVisible(true);

}

});

btnBack\_1.setBounds(212, 79, 75, 30);

panelStatus.add(btnBack\_1);

btnBack.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

SetupServer.this.dispose();

Start start = new Start();

start.setVisible(true);

}

});

}

/\*

\* Aqui é iniciado o servidor, o inetadress pega as informações do localhost

\* e inicia o servidor, deixando disponivel para novas conexões

\*/

private void startServer() {

try {

server = new ServerSocket(Integer.parseInt(inputPort.getText()));

new Thread(new Runnable() {

@Override

public void run() {

CardLayout c (CardLayout)(contentPane.getLayout());

c.show(contentPane, "panelStatus");

try {

inetAddress InetAddress.getLocalHost();

hostaddress inetAddress.getHostAddress();

localhost = hostaddress;

lblValueIP.setText(hostaddress);

} catch (UnknownHostException e1) {

e1.printStackTrace();

}

lblValuePort.setText(inputPort.getText());

while (true) {

System.out.println("Waiting connection...");

try {

Socket connection = server.accept();

Thread serverThread = new Server(connection);

serverThread.start();

} catch (Exception e) {

e.printStackTrace();

}

}

}

}).start();

} catch (Exception e) {

e.printStackTrace();

}

}

}

**Login.java**

package chatform;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.JButton;

import javax.swing.JLabel;

import java.awt.Font;

import javax.swing.JTextField;

import java.awt.event.ActionListener;

import java.net.InetAddress;

import java.net.UnknownHostException;

import java.awt.event.ActionEvent;

import java.awt.Color;

public class Login extends JFrame {

/\*\*

\*

\*/

private static final long serialVersionUID = 948747724372712259L;

private JPanel contentPane;

private JTextField inputIP;

private JTextField inputPort;

private JTextField inputUser;

SetupServer s1 = new SetupServer();

private static InetAddress inetAddress;

private static String hostaddress;

/\*\*

\* Executa a aplicação

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

Login frame = new Login();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Cria o frame

\*/

public Login() {

setResizable(false);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 374, 277);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

setLocationRelativeTo(null);

setTitle("Aplicacao de Conversa (Cliente)");

try {

inetAddress = InetAddress.getLocalHost();

hostaddress = inetAddress.getHostAddress();

} catch (UnknownHostException e1) {

e1.printStackTrace();

}

JButton btnLogin = new JButton("Login");

btnLogin.setBackground(new Color(72, 209, 204));

btnLogin.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

Login.this.dispose();

setFocusableWindowState(false);

connect();

}

});

btnLogin.setFont(new Font("Arial Black", Font.PLAIN, 12));

btnLogin.setBounds(5, 198, 83, 39);

contentPane.add(btnLogin);

JButton btnBack = new JButton("Voltar");

btnBack.setFont(new Font("Arial Black", Font.PLAIN, 12));

btnBack.setBackground(new Color(72, 209, 204));

btnBack.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Login.this.dispose();

Start start = new Start();

start.setVisible(true);

}

});

btnBack.setBounds(98, 198, 83, 39);

contentPane.add(btnBack);

JLabel lblServerIp = new JLabel("IP do servidor:");

lblServerIp.setFont(new Font("Arial Black", Font.PLAIN, 12));

lblServerIp.setBounds(5, 11, 151, 39);

contentPane.add(lblServerIp);

inputIP = new JTextField();

inputIP.setText(hostaddress);

inputIP.setFont(new Font("Arial Black", Font.PLAIN, 15));

inputIP.setBounds(138, 15, 222, 30);

contentPane.add(inputIP);

inputIP.setColumns(10);

JLabel lblPortaDoServidor = new JLabel("Porta do servidor:");

lblPortaDoServidor.setFont(new Font("Arial Black", Font.PLAIN, 12));

lblPortaDoServidor.setBounds(5, 72, 151, 39);

contentPane.add(lblPortaDoServidor);

inputPort = new JTextField();

inputPort.setText("");

inputPort.setFont(new Font("Arial Black", Font.PLAIN, 15));

inputPort.setColumns(10);

inputPort.setBounds(138, 76, 222, 30);

contentPane.add(inputPort);

JLabel lblUsurio = new JLabel("Nome de usuário:");

lblUsurio.setFont(new Font("Arial Black", Font.PLAIN, 12));

lblUsurio.setBounds(5, 134, 151, 39);

contentPane.add(lblUsurio);

inputUser = new JTextField();

inputUser.setText("unip1");

inputUser.setFont(new Font("Arial Black", Font.PLAIN, 15));

inputUser.setColumns(10);

inputUser.setBounds(138, 138, 222, 30);

contentPane.add(inputUser);

}

private void connect() {

try {

Thread thread = new Thread(new Client(new String[] {getUser(), getIP(), getPort()}));

thread.start();

} catch (Exception e) {

e.printStackTrace();

}

}

private String getIP() {

return inputIP.getText();

}

private String getPort() {

return inputPort.getText();

}

private String getUser() {

return inputUser.getText();

} }

**Client.java**

package chatform;

import java.awt.BorderLayout;

import java.awt.event.\*;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.JTextField;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.JButton;

import javax.swing.ScrollPaneConstants;

import javax.swing.text.DefaultCaret;

import java.io.\*;

import java.net.ConnectException;

import java.net.Socket;

import java.awt.Font;

import java.awt.Color;

public class Client extends JFrame implements Runnable {

/\*\*

\* Variáveis do Socket

\*/

private Socket socket;

private BufferedWriter bufferWriter;

/\*\*

\* Variáveis da clsse Client

\*/

private String user;

private String serverIP;

private int serverPort;

Login login = new Login();

/\*\*

\* Variáveis do Form

\*/

private static final long serialVersionUID = 5391582161763137020L;

private JTextField inputText;

private JTextArea output;

/\*\*

\* Execução da aplicação

\*/

public void run() {

try {

this.setTitle(this.user);

this.establishConnection();

this.listenConnection();

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* Cria o frame, aqui é verificado a conexão do socket,

\* como também, se necessário encerra o programa ou retorna para tela anterior

\*/

public Client(String[] args) {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

if(socket == null)

dispose();

else if(!socket.isClosed()) {

try {

disconnect();

} catch (Exception ex) {

ex.printStackTrace();

}

}

Start start = new Start();

start.setVisible(true);

dispose();

}

}

);

/\* Aqui é definido as propriedades da interface e seus componentes

\*

\*/

setBounds(100, 100, 600, 500);

JPanel contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(new BorderLayout(0, 0));

setLocationRelativeTo(null);

setTitle("Aplicação de Conversa (Cliente)");

output = new JTextArea();

output.setBackground(Color.WHITE);

output.setFont(new Font("Arial Black", Font.PLAIN, 12));

output.setEditable(false);

output.setLineWrap(true);

JScrollPane messages = new JScrollPane(output);

messages.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL\_SCROLLBAR\_ALWAYS);

DefaultCaret caretOutput = (DefaultCaret) output.getCaret();

caretOutput.setUpdatePolicy(DefaultCaret.ALWAYS\_UPDATE);

contentPane.add(messages, BorderLayout.CENTER);

JPanel interactive = new JPanel();

contentPane.add(interactive, BorderLayout.SOUTH);

interactive.setLayout(new BorderLayout(0, 0));

inputText = new JTextField();

inputText.setFont(new Font("Arial Black", Font.PLAIN, 12));

interactive.add(inputText, BorderLayout.CENTER);

inputText.setColumns(10);

JPanel buttons = new JPanel();

interactive.add(buttons, BorderLayout.EAST);

buttons.setLayout(new BorderLayout(0, 0));

JButton btnSend = new JButton("Enviar");

btnSend.setBackground(new Color(72, 209, 204));

btnSend.setFont(new Font("Arial Black", Font.PLAIN, 11));

btnSend.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String text = inputText.getText();

if(!text.equals("")) {

sendMessage(concatMsg(text));

inputText.setText("");

}

}

});

buttons.add(btnSend, BorderLayout.NORTH);

inputText.addKeyListener(new KeyListener() {

@Override

public void keyTyped(KeyEvent e) {

// TODO Auto-generated method stub

}

@Override

public void keyReleased(KeyEvent e) {

// TODO Auto-generated method stub

}

@Override

public void keyPressed(KeyEvent e) {

if (e.getKeyCode() == KeyEvent.VK\_ENTER) {

String text = inputText.getText();

if(!text.equals("")) {

sendMessage(concatMsg(text));

inputText.setText("");

}

}

}

});

JButton btnExit = new JButton("Desconectar");

btnExit.setBackground(new Color(72, 209, 204));

btnExit.setFont(new Font("Arial Black", Font.PLAIN, 11));

btnExit.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

disconnect();Client.this.dispose();

login.setVisible(true);

} catch (Exception e1) {

e1.printStackTrace();

}

}

});

buttons.add(btnExit, BorderLayout.SOUTH);

JButton btnBack = new JButton("Voltar");

btnBack.setBackground(new Color(72, 209, 204));

btnBack.setFont(new Font("Arial Black", Font.PLAIN, 11));

btnBack.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Client.this.dispose();

login.setVisible(true);

}

});

buttons.add(btnBack, BorderLayout.CENTER);

setLocationRelativeTo(null);

setVisible(true);

setClientInfo(args[0], args[1], args[2]);

}

private void setClientInfo(String user, String serverIP, String serverPort) {

this.user = user;

this.serverIP = serverIP;

this.serverPort = Integer.parseInt(serverPort);

}

private void sendMessage(String msg) {

try {

bufferWriter.write(msg);

bufferWriter.flush();

} catch (Exception e) {

writeOutput("Desconectado");

}

}

private String concatMsg(String msg) {

return "Text&" + "[" + user + "]" + " ~> " + msg + "\r\n";

}

private void writeOutput(String phrase) {

output.append(phrase + "\r\n");

}

/\*Aqui verifica o IP e a Porta informados no login e estabelece a conexão com o servidor,

\* caso contrário, apresenta a mensagem de erro

\*

\*/

private void establishConnection() {

try {

socket = new Socket(this.serverIP, this.serverPort);

bufferWriter = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

bufferWriter.write(user + "\r\n");

bufferWriter.flush();

} catch (ConnectException e) {

JOptionPane.showMessageDialog(null, "Nao foi possível criar a conexão, servidor indisponível na porta e IP indicados.");

Client.this.dispose();

login.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

disconnect();

}

}

/\*

\* Aqui é verificado a mensagem, se ocorrer algum problema na mensagem, apresenta erro e desconeta o usuário

\*/

private void listenConnection() {

try {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

String msg = "";

String command;

String textMsg;

do {

if (bufferedReader.ready()) {

msg = bufferedReader.readLine();

command = msg.split("&", 2)[0];

textMsg = msg.split("&", 2)[1];

if(command.equals("Text")) {

writeOutput(textMsg);

} else {

writeOutput("Algo está errado na mensagem recebida do servidor");

}

}

} while (!("Disconnect " + user).equalsIgnoreCase(msg));

} catch (Exception e) {

System.out.println("Impossível escutar servidor. O mesmo possívelmente está indisponível.");

}

}

private void disconnect() {

writeOutput("Desconectado");

sendMessage("Disconnect " + this.user);

try {

bufferWriter.close();

socket.close();

} catch (Exception e) {

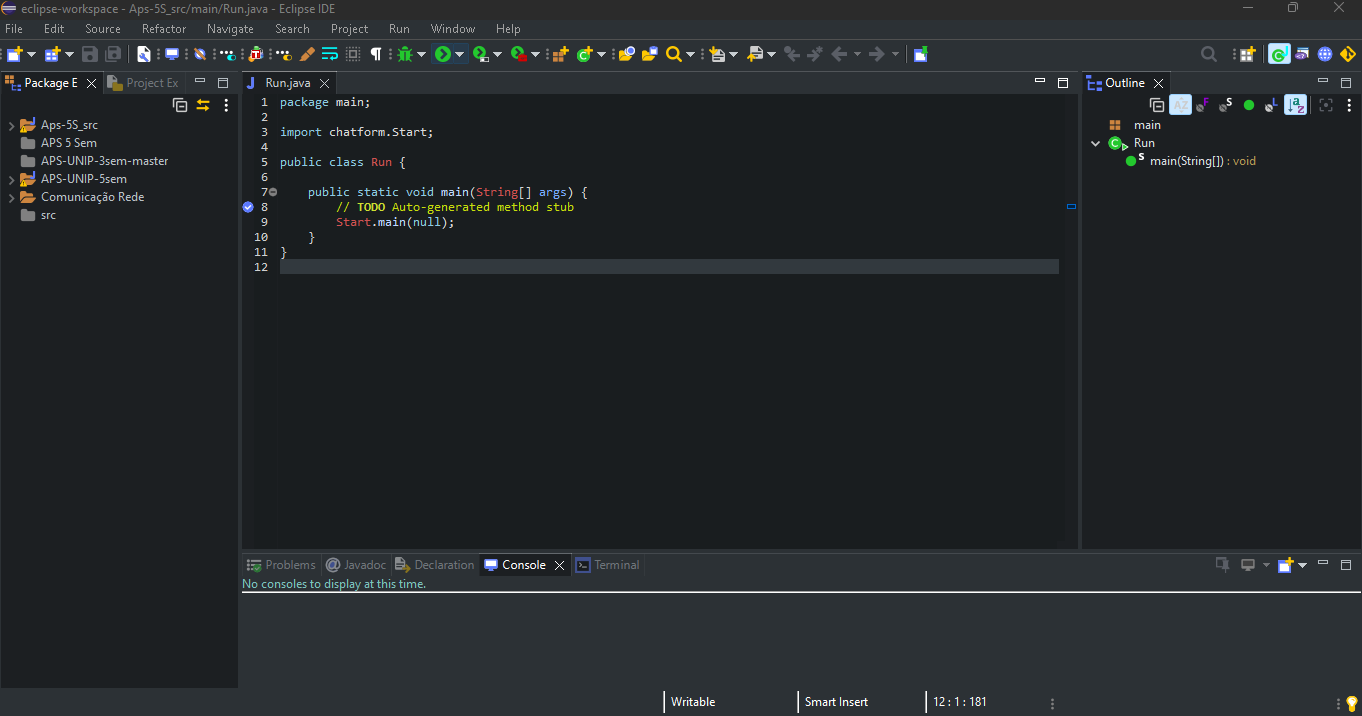
System.out.println("Nao é possível fechar conexão.");

}

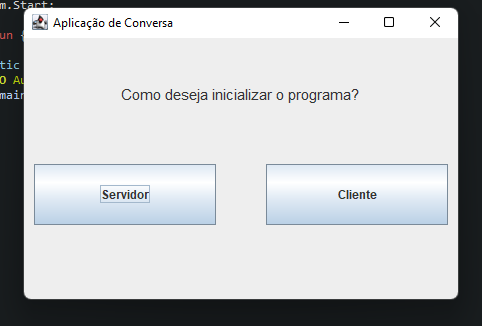
}

}

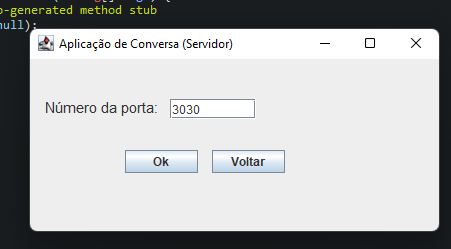
**7 Apresentação do programa funcionando em um computador**



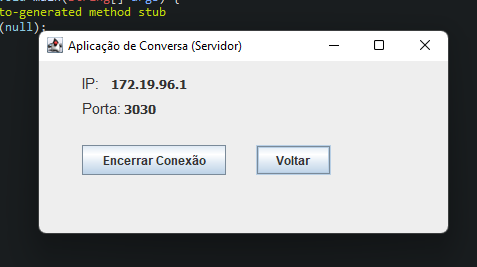
fonte: Guilherme Augusto,2022.



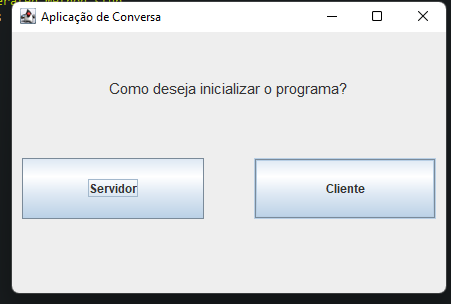
fonte: Guilherme Augusto,2022.



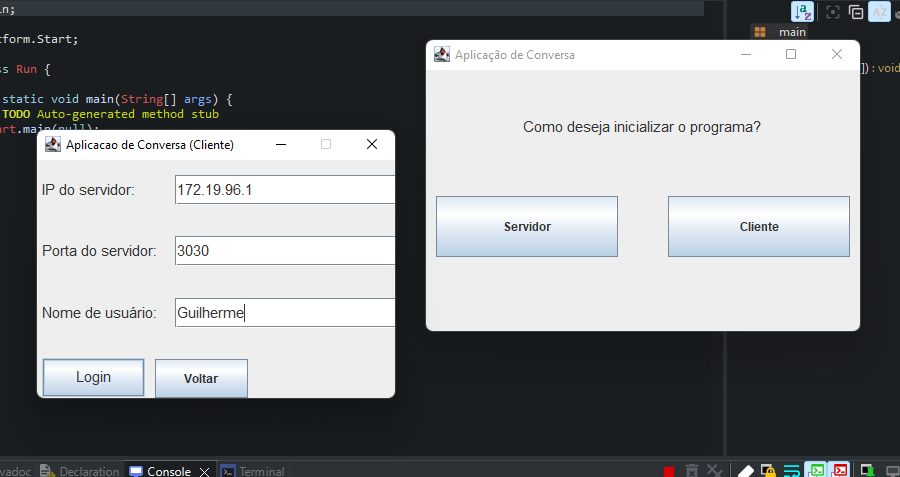
fonte: Guilherme Augusto,2022.



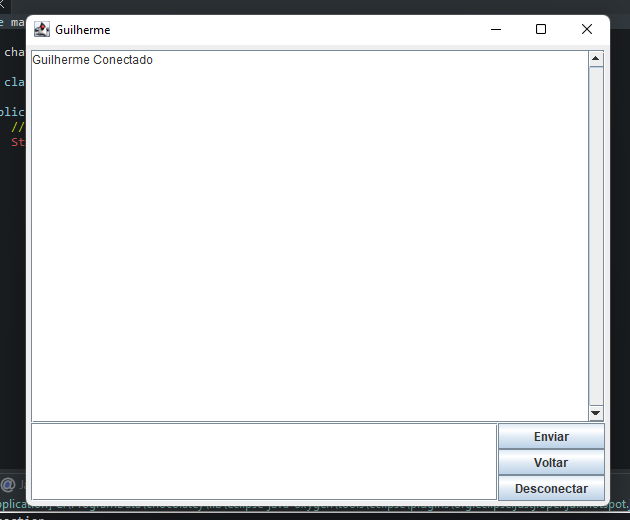
fonte: Guilherme Augusto,2022.



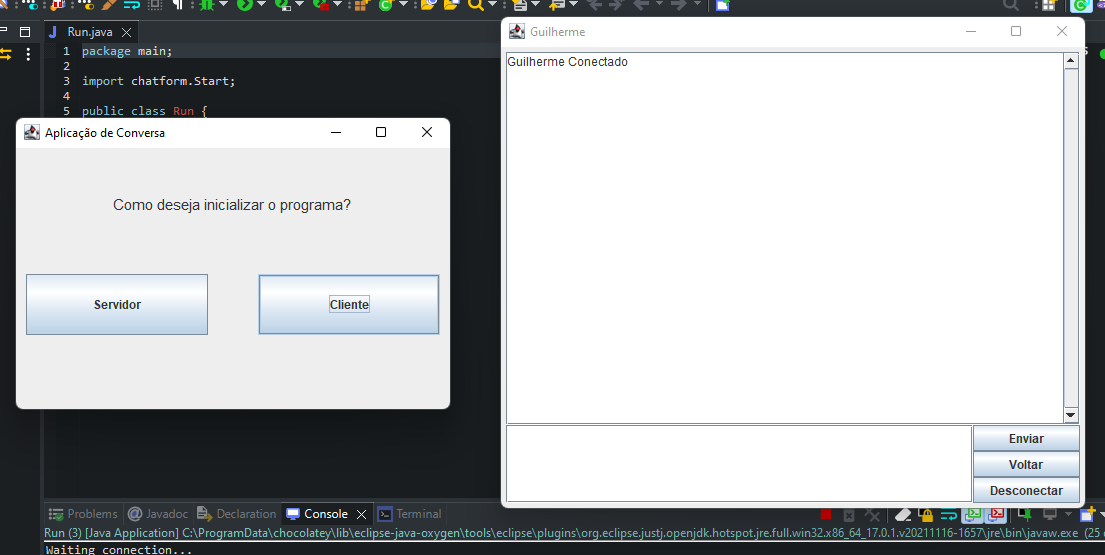
fonte: Guilherme Augusto,2022.



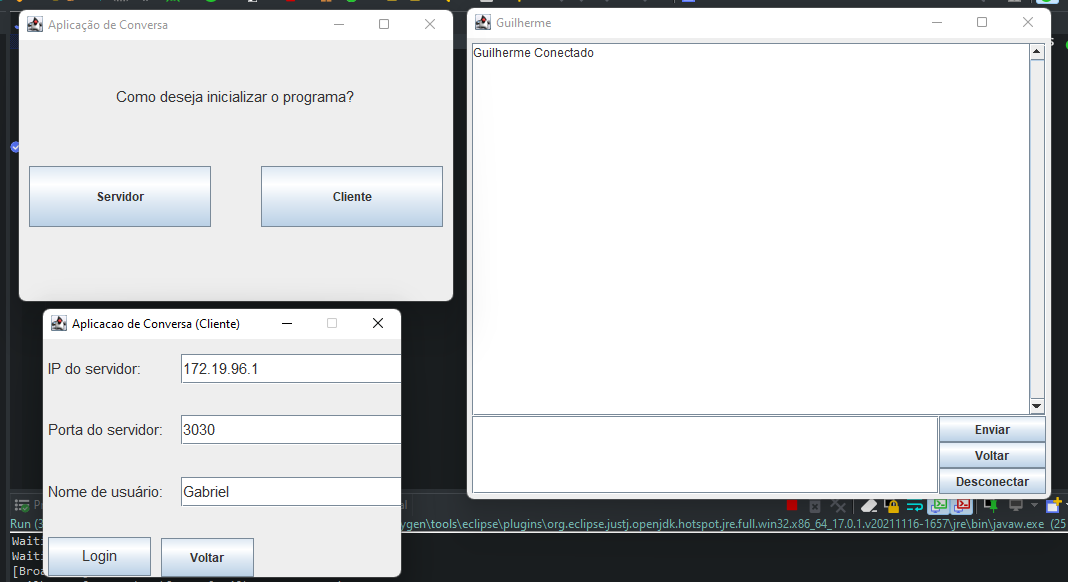
fonte: Guilherme Augusto,2022.



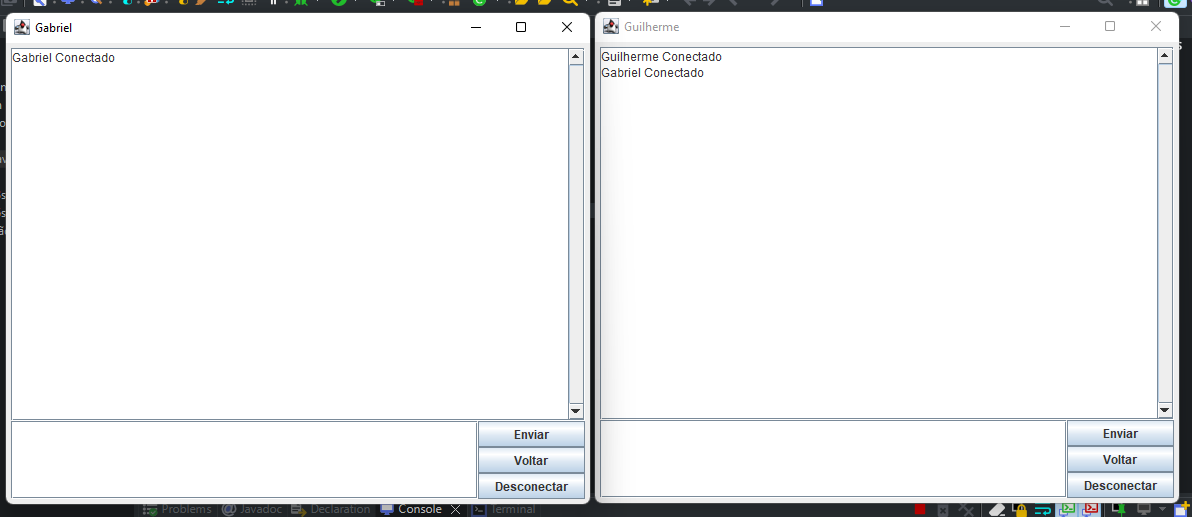
fonte: Guilherme Augusto,2022.



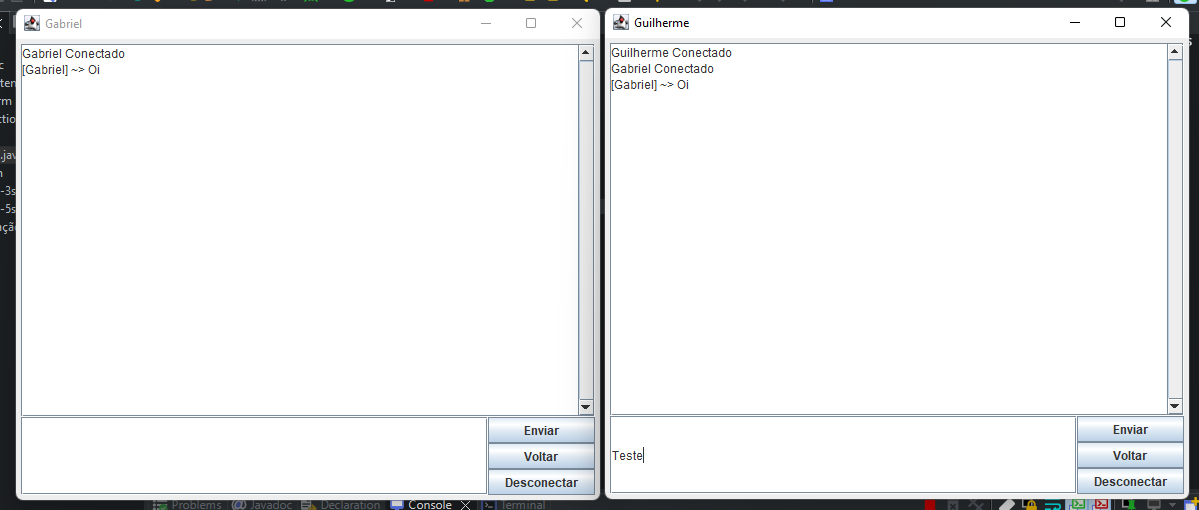
fonte: Guilherme Augusto,2022.



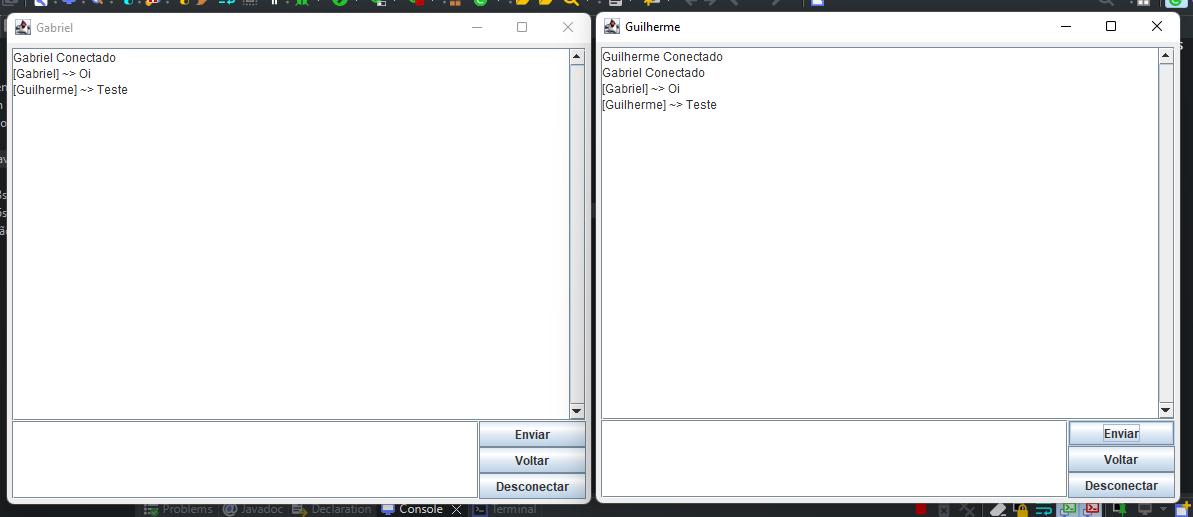
fonte: Guilherme Augusto,2022.

****

fonte: Guilherme Augusto,2022.



fonte: Guilherme Augusto,2022.



fonte: Guilherme Augusto,2022.

**8 Bibliografia**

Mozilla Developer 2022. Disponível em:

[Arpanet - | MDN (mozilla.org)](https://developer.mozilla.org/pt-BR/docs/Glossary/Arpanet) Acesso em 28/04/2022

Tecmundo 2012 Disponível em:

[O que é TCP/IP? - TecMundo](https://www.tecmundo.com.br/o-que-e/780-o-que-e-tcp-ip-.htm) Acesso em 28/04/2022

Pantuza Blog 2017 Disponível em:

[O que são e como funcionam os Sockets (pantuza.com)](https://blog.pantuza.com/artigos/o-que-sao-e-como-funcionam-os-sockets) Acesso em 28/04/2022

GOV CRTRJ 2021 Disponível em:

[Redes de Comunicação de Dados | Principais Conceitos – CRT-RJ (crtrj.gov.br)](https://www.crtrj.gov.br/redes-de-comunicacao-de-dados-principais-conceitos/#:~:text=Para%20que%20a%20comunica%C3%A7%C3%A3o%20seja,envia%20a%20mensagem%20de%20dados.) Acesso em 29/04/2022

ESCOLA, Brasil 2020 Disponível em:

[Comunicação de Dados - Brasil Escola (uol.com.br)](https://brasilescola.uol.com.br/informatica/comunicacao-dados.htm) Acesso em 07/05/2022

Bibliografias consultadas

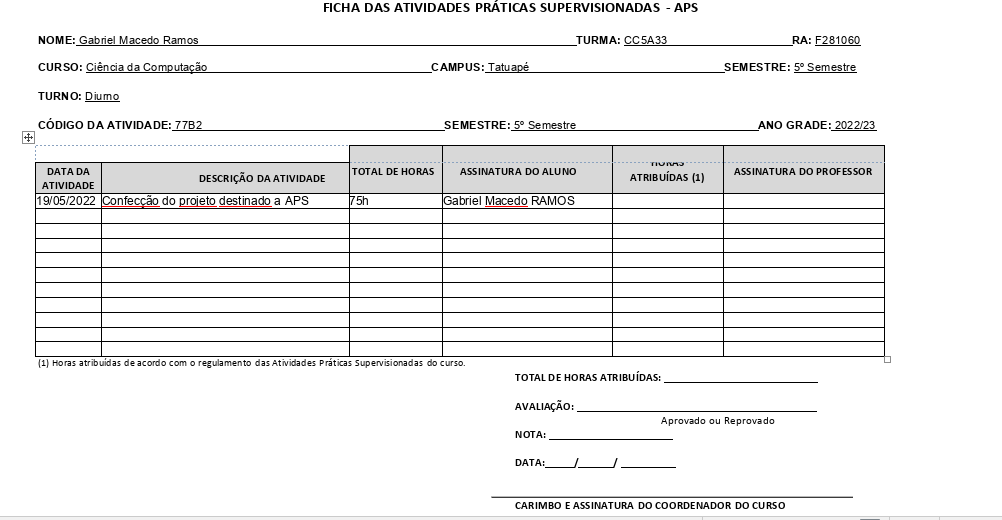
COMER, Douglas E. Internet Working with TCP/IP. 4a. Ed. Prentice Hall, 2000.

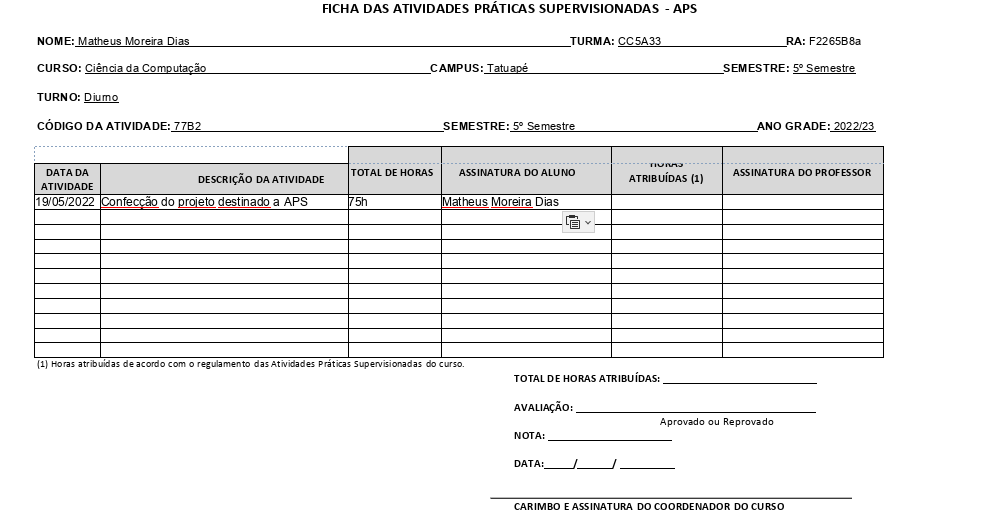
FOROUZAN, Behrouz A. Comunicação de dados e redes de computadores. 3a. ed. Porto Alegre: Bookman, 2006.

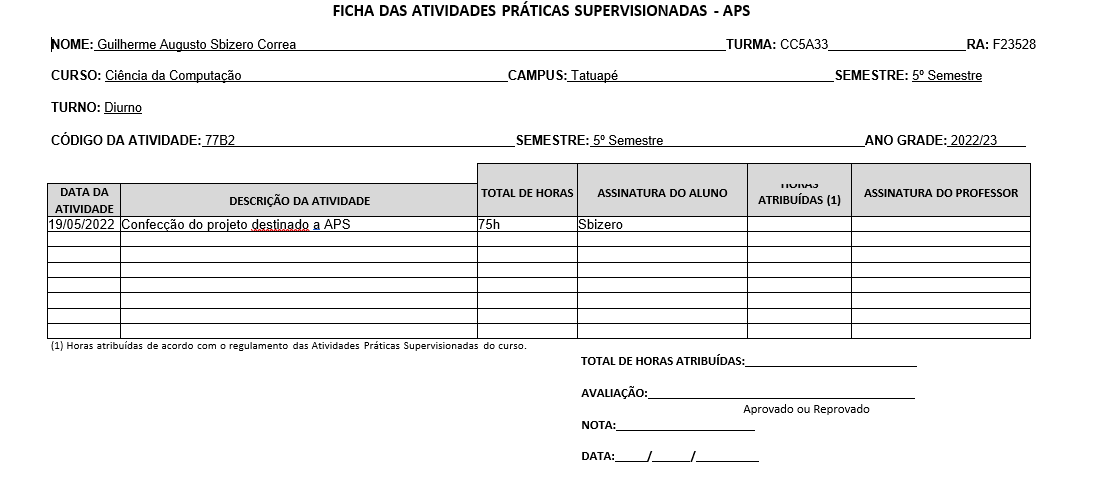
STALLINGS, William. Data and computer communications. 8a. ed. Upper Saddle River, NJ : Prentice Hall, 2006.

STALLINGS, William. Redes e sistemas de comunicação de dados. 5ª. ed. Rio de Janeiro: Elsevier, 2005."

**9 Ficha de Atividade Prática Supervisionada**

****

****

****

**Link para repositório da APS no Google Drive e no GitHub**

**Google Drive:** https://drive.google.com/drive/folders/1y0JGdIWIkx1jXWLdQyzD5Hhpvb13oR0U?usp=sharing

**GitHub:** [Guilherme-Sbizero/APS-5-Semestre: Repositório com os arquivos e programa da aps de 5º semestre de Ciência da Computação (github.com)](https://github.com/Guilherme-Sbizero/APS-5-Semestre)