## **Servidor FTP**

### **Conectarse al servidor y logearse**

//Conexión al servidor

servFTP=”localhost”;

String usuario..

String passwd...

FTPClient cliente = new FTPClient();

cliente.connect(servFTP);

cliente.enterLocalPassiveMode();

System.out.println("Conectando al sevidor: "+servFTP);

boolean logeado = cliente.login(usuario, password);

if (logeado) {

System.out.println("Login exitoso...");

}else {

System.out.println("Error en el login...");

cliente.disconnect();

System.exit(1);

}

### **Mirar ficheros del directorio actual**

SE HACE CONEXION Y LOGIN

print(“Directorio actual”: +cliente.printWprkingDirectory());

FTP[]File files=clientes.listFiles();

for(int i=0;i<files.length;i++){

pritnt(files[i]

LOGOUT Y DESCONEXION

### **Mirar ficheros de un directorio**

String directorio=”/PSP;

if(cliente.changeWorkingDirectory(directorio)){

print(“Directorio actual”: +cliente.printWprkingDirectory());

else{

error

}

FTP[]File files=clientes.listFiles();

for(int i=0;i<files.length;i++){

pritnt(files[i]

### **Subir ficheros al servidor**

#### **Con interfaz**

public static void addArchivo(String servFTP, String usuario, String password) {

try {

//Conexión al servidor

FTPClient cliente = new FTPClient();

cliente.connect(servFTP);

cliente.enterLocalPassiveMode();

System.out.println("Conectando al sevidor: "+servFTP);

boolean logeado = cliente.login(usuario, password);

if (logeado) {

System.out.println("Login exitoso...");

}else {

System.out.println("Error en el login...");

cliente.disconnect();

System.exit(1);

}

//Nos movemos al directorio del usuario

cliente.changeWorkingDirectory("/"+usuario);

//Selección del archivo y subida al servidor FTP

JFileChooser fileChooser = new JFileChooser();

int result = fileChooser.showOpenDialog(null);

if (result == JFileChooser.APPROVE\_OPTION) {

File selectedFile = fileChooser.getSelectedFile();

System.out.println("Archivo: " + selectedFile.getAbsolutePath());

String archivo = selectedFile.getAbsolutePath();

BufferedInputStream in = new BufferedInputStream(new FileInputStream(archivo));

if (cliente.storeFile(selectedFile.getName(), in)) {

JOptionPane.showMessageDialog(

null,

selectedFile.getName()+"=> Subido correctamente",

"Info",

JOptionPane.INFORMATION\_MESSAGE

);

}else {

JOptionPane.showMessageDialog(

null,

selectedFile.getName()+"=> Error al subir archivo",

"ERROR",

JOptionPane.ERROR\_MESSAGE

);

}

in.close();

} else {

System.out.println("Error, no se ha seleccionado ningun archivo.");

}

//Se muestra el contenido del directorio actual

System.out.println("Directorio actual: "+cliente.printWorkingDirectory());

FTPFile[] directorios = cliente.listFiles();

for (int i = 0; i < directorios.length; i++) {

System.out.println("\t =>"+directorios[i].getName());

}

//Desconexión

boolean logout = cliente.logout();

if (logout) {

System.out.println("Logout correcto...");

} else {

System.out.println("Logout incorrecto...");

}

cliente.disconnect();

System.out.println("Desconectado\n");

} catch (IOException e){

e.printStackTrace();

}

#### **Sin interfaz**

import org.apache.commons.net.ftp.FTP;

import org.apache.commons.net.ftp.FTPClient;

import java.io.\*;

public class SubirFichero {

public static void main(String[] args) {

FTPClient cliente = new FTPClient();

String servidor = "localhost"; // Cambiar por la dirección del servidor FTP

String usuario = "usuario1"; // Cambiar por el usuario FTP

String pasw = "usuario1"; // Cambiar por la contraseña FTP

try {

System.out.println("Conectándose a " + servidor + "...");

// Conectar al servidor FTP

cliente.connect(servidor);

cliente.enterLocalPassiveMode();

// Autenticación

boolean login = cliente.login(usuario, pasw);

if (!login) {

System.out.println("Error en la autenticación.");

return;

}

System.out.println("Login correcto");

// 📌 Directorio donde se subirá el archivo

String direc = "NUEVODIREC";

// Si el directorio no existe, lo crea

if (!cliente.changeWorkingDirectory(direc)) {

String directorio = "NUEVODIREC";

if (cliente.makeDirectory(directorio)) {

System.out.println("Directorio creado: " + directorio);

cliente.changeWorkingDirectory(directorio);

} else {

System.out.println("No se ha podido crear el directorio.");

System.exit(0);

}

}

System.out.println("Directorio actual: " + cliente.printWorkingDirectory());

// 📌 Subida del archivo

String archivoLocal = "D:/CLASE/EJEMPLO.pdf"; // Ruta del archivo a subir

File archivo = new File(archivoLocal);

if (!archivo.exists()) {

System.out.println("El archivo no existe en la ruta especificada.");

return;

}

try (InputStream in = new BufferedInputStream(new FileInputStream(archivo))) {

cliente.setFileType(FTP.BINARY\_FILE\_TYPE); // Usar modo binario para evitar corrupción

if (cliente.storeFile("EJEMPLO.pdf", in)) {

System.out.println("Archivo subido correctamente.");

} else {

System.out.println("Error al subir el archivo.");

}

}

// Cerrar sesión y desconectar

cliente.logout();

cliente.disconnect();

System.out.println("Desconectado del servidor FTP.");

} catch (IOException ioe) {

ioe.printStackTrace();

}

}

}

### **Descargar ficheros del servidor**

#### **Con interfaz**

// Cliente FTP para gestionar la conexión y transferencia de archivos

private static FTPClient ftpClient = new FTPClient();

public static void main(String[] args) throws IOException {

// Crear ventana principal de la aplicación

JFrame frame = new JFrame("DESCARGAR FICHEROS");

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

frame.setSize(600, 400);

frame.setLayout(new BorderLayout());

// Panel superior para ingresar usuario y contraseña

JPanel loginPanel = new JPanel();

loginPanel.setLayout(new GridLayout(3, 2));

JLabel userLabel = new JLabel("Usuario:");

JTextField userField = new JTextField("uno"); // Campo para ingresar usuario (predefinido "uno")

JLabel passLabel = new JLabel("Contraseña:");

JPasswordField passField = new JPasswordField("uno"); // Campo para ingresar contraseña (predefinida "uno")

JButton connectButton = new JButton("Conectar");

loginPanel.add(userLabel);

loginPanel.add(userField);

loginPanel.add(passLabel);

loginPanel.add(passField);

loginPanel.add(new JLabel()); // Espaciador vacío

loginPanel.add(connectButton);

frame.add(loginPanel, BorderLayout.NORTH);

// Lista para mostrar los archivos del servidor y botones de interacción

DefaultListModel<String> fileListModel = new DefaultListModel<>();

JList<String> fileList = new JList<>(fileListModel);

JScrollPane scrollPane = new JScrollPane(fileList);

JButton downloadButton = new JButton("Descargar");

JButton exitButton = new JButton("Salir");

JPanel buttonPanel = new JPanel();

buttonPanel.add(downloadButton);

buttonPanel.add(exitButton);

frame.add(scrollPane, BorderLayout.CENTER); // Lista en el centro

frame.add(buttonPanel, BorderLayout.SOUTH); // Botones en la parte inferior

// Acción para conectarse al servidor FTP

connectButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String user = userField.getText();

String password = new String(passField.getPassword());

try {

// Conectar al servidor FTP

ftpClient.connect("localhost", 21);

boolean login = ftpClient.login(user, password);

ftpClient.changeWorkingDirectory("/home/usuario");

if (login) {

JOptionPane.showMessageDialog(frame, "Conexión exitosa.");

ftpClient.enterLocalPassiveMode(); // Configurar modo pasivo

// Listar archivos del servidor

fileListModel.clear();

for (String file : ftpClient.listNames()) {

fileListModel.addElement(file);

}

} else {

JOptionPane.showMessageDialog(frame, "Usuario o contraseña incorrectos.", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (IOException ex) {

JOptionPane.showMessageDialog(frame, "Error al conectar: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

});

// Acción para descargar un archivo seleccionado

downloadButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String selectedFile = fileList.getSelectedValue();

if (selectedFile == null) {

JOptionPane.showMessageDialog(frame, "Seleccione un archivo para descargar.", "Error", JOptionPane.WARNING\_MESSAGE);

return; // Terminar si no se selecciona un archivo

}

// Permitir al usuario seleccionar el directorio de destino

JFileChooser fileChooser = new JFileChooser();

fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES\_ONLY);

int result = fileChooser.showSaveDialog(frame);

if (result == JFileChooser.APPROVE\_OPTION) {

File directory = fileChooser.getSelectedFile();

File localFile = new File(directory, selectedFile); // Ruta de destino local

try (FileOutputStream fos = new FileOutputStream(localFile)) {

boolean success = ftpClient.retrieveFile(selectedFile, fos); // Descargar archivo

if (success) {

JOptionPane.showMessageDialog(frame, "Archivo descargado correctamente en: " + localFile.getAbsolutePath());

} else {

JOptionPane.showMessageDialog(frame, "Error al descargar el archivo.", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (IOException ex) {

JOptionPane.showMessageDialog(frame, "Error al guardar el archivo: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

});

// Acción para salir de la aplicación

exitButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

if (ftpClient.isConnected()) {

ftpClient.logout(); // Cerrar sesión

ftpClient.disconnect(); // Desconectar del servidor

}

} catch (IOException ex) {

ex.printStackTrace(); // Imprimir detalles de cualquier error

}

frame.dispose(); // Cerrar la ventana principal

}

});

frame.setVisible(true); // Mostrar la ventana

}

}

#### **Sin interfaz**

### **Obtener ficheros directorio actual**

#### **Con interfaz**

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.IOException;

import javax.swing.\*;

import org.apache.commons.net.ftp.\*;

public class ClienteFTP {

private static FTPClient ftpClient = new FTPClient(); // Cliente FTP

private static DefaultListModel<String> fileListModel = new DefaultListModel<>();

public static void main(String[] args) {

// Crear ventana principal

JFrame frame = new JFrame("LISTAR ARCHIVOS DEL FTP");

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

frame.setSize(600, 400);

frame.setLayout(new BorderLayout());

// Panel superior (Usuario y contraseña)

JPanel loginPanel = new JPanel(new GridLayout(3, 2));

JLabel userLabel = new JLabel("Usuario:");

JTextField userField = new JTextField("uno");

JLabel passLabel = new JLabel("Contraseña:");

JPasswordField passField = new JPasswordField("uno");

JButton connectButton = new JButton("Conectar");

loginPanel.add(userLabel);

loginPanel.add(userField);

loginPanel.add(passLabel);

loginPanel.add(passField);

loginPanel.add(new JLabel()); // Espaciador vacío

loginPanel.add(connectButton);

frame.add(loginPanel, BorderLayout.NORTH);

// Lista de archivos en el FTP

JList<String> fileList = new JList<>(fileListModel);

JScrollPane scrollPane = new JScrollPane(fileList);

frame.add(scrollPane, BorderLayout.CENTER);

// Botones de interacción

JButton refreshButton = new JButton("Actualizar Lista");

JButton exitButton = new JButton("Salir");

JPanel buttonPanel = new JPanel();

buttonPanel.add(refreshButton);

buttonPanel.add(exitButton);

frame.add(buttonPanel, BorderLayout.SOUTH);

// Acción para conectarse al servidor FTP

connectButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String user = userField.getText();

String password = new String(passField.getPassword());

try {

ftpClient.connect("localhost", 21);

boolean login = ftpClient.login(user, password);

ftpClient.enterLocalPassiveMode();

if (login) {

JOptionPane.showMessageDialog(frame, "Conexión exitosa.");

fileListModel.clear(); // Limpiar lista antes de actualizar

String[] archivos = ftpClient.listNames();

if (archivos != null) {

for (String archivo : archivos) {

fileListModel.addElement(archivo);

}

} else {

JOptionPane.showMessageDialog(frame, "No se encontraron archivos en el directorio actual.");

}

} else {

JOptionPane.showMessageDialog(frame, "Usuario o contraseña incorrectos.", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (IOException ex) {

JOptionPane.showMessageDialog(frame, "Error al conectar: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

});

// Acción para actualizar la lista de archivos

refreshButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

fileListModel.clear();

String[] archivos = ftpClient.listNames();

if (archivos != null) {

for (String archivo : archivos) {

fileListModel.addElement(archivo);

}

} else {

JOptionPane.showMessageDialog(frame, "No se encontraron archivos en el directorio actual.");

}

} catch (IOException ex) {

JOptionPane.showMessageDialog(frame, "Error al obtener la lista de archivos: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

});

// Acción para salir de la aplicación

exitButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

if (ftpClient.isConnected()) {

ftpClient.logout();

ftpClient.disconnect();

}

} catch (IOException ex) {

ex.printStackTrace();

}

frame.dispose();

}

});

frame.setVisible(true);

}

}

#### **Sin interfaz**

## **Comunicación servidor SMTP**

### **Conectarse al servidor y Enviar correo electrónico**

#### **Para 1**

AuthenticatingSMTPClient cliente = new AuthenticatingSMTPClient();

Scanner scanner = new Scanner(System.in);

System.out.print("Introduce el servidor SMTP: ");

String server = scanner.nextLine();

System.out.print("Usuario: ");

String username = scanner.nextLine();

System.out.print("Password: ");

String password = scanner.nextLine();

System.out.print("Puerto: ");

int puerto = Integer.parseInt(scanner.nextLine());

System.out.print("Introduce el remitente: ");

String remitente = scanner.nextLine();

try {

int respuesta;

cliente.connect(server, puerto);

System.out.println("2 - " + cliente.getReplyString());

KeyManagerFactory kmf = KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());

kmf.init(null, null);

KeyManager km = kmf.getKeyManagers()[0];

cliente.setKeyManager(km);

respuesta = cliente.getReplyCode();

if (!SMTPReply.isPositiveCompletion(respuesta)) {

cliente.disconnect();

System.out.println("Se ha producido un error al enviar el servidor SMTP");

return;

}

cliente.ehlo(server);

System.out.println("2 - " + cliente.getReplyString());

if (cliente.execTLS()) {

System.out.println("3 - " + cliente.getReplyString());

if (cliente.auth(AuthenticatingSMTPClient.AUTH\_METHOD.PLAIN, username, password)) {

System.out.println("4 - " + cliente.getReplyString());

System.out.print("Introduce el destinatario: ");

String destino1 = scanner.nextLine();

System.out.print("Introduce el asunto: ");

String asunto = scanner.nextLine();

System.out.print("Introduce el mensaje: ");

String mensaje = scanner.nextLine();

SimpleSMTPHeader cabecera = new SimpleSMTPHeader(remitente, destino1, "Asunto: " + asunto);

cliente.setSender(remitente);

cliente.addRecipient(destino1);

System.out.println("5 - " + cliente.getReplyString());

Writer writer = cliente.sendMessageData();

if (writer != null) {

writer.write(cabecera.toString());

writer.write(mensaje);

writer.close();

System.out.println("6 - " + cliente.getReplyString());

boolean exito = cliente.completePendingCommand();

System.out.println("7 - " + cliente.getReplyString());

if (exito) {

System.out.println("Mensaje enviado correctamente");

} else {

System.out.println("Error al enviar el servidor SMTP");

}

} else {

System.out.println("Fallo al enviar la data");

}

} else {

System.out.println("Usuario no valido");

}

} else {

System.out.println("Error en TLS");

}

cliente.disconnect();

} catch (Exception e) {

e.printStackTrace();

}

try {

cliente.disconnect();

}catch (Exception e) {

e.printStackTrace();

}

System.out.println("Proceso finalizado");

System.out.println(0);

}

#### **Con interfaz**

import org.apache.commons.net.smtp.AuthenticatingSMTPClient;

import org.apache.commons.net.smtp.SimpleSMTPHeader;

import javax.net.ssl.KeyManager;

import javax.net.ssl.KeyManagerFactory;

import javax.swing.\*;

import java.awt.\*;

import java.io.Writer;

public class ClienteSMTPGUI {

private JFrame frame;

private JTextField serverField, portField, userField, senderField, recipientField, subjectField;

private JPasswordField passwordField;

private JTextArea messageArea;

private JRadioButton tlsYes, tlsNo;

private JButton sendButton;

public ClienteSMTPGUI() {

frame = new JFrame("Cliente SMTP");

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

frame.setSize(600, 500);

frame.setLayout(new GridLayout(10, 2));

// Campos para ingresar los datos SMTP

frame.add(new JLabel("Servidor SMTP:"));

serverField = new JTextField();

frame.add(serverField);

frame.add(new JLabel("Puerto:"));

portField = new JTextField("587");

frame.add(portField);

frame.add(new JLabel("Usuario:"));

userField = new JTextField();

frame.add(userField);

frame.add(new JLabel("Contraseña:"));

passwordField = new JPasswordField();

frame.add(passwordField);

frame.add(new JLabel("Remitente:"));

senderField = new JTextField();

frame.add(senderField);

frame.add(new JLabel("Destinatario:"));

recipientField = new JTextField();

frame.add(recipientField);

frame.add(new JLabel("Asunto:"));

subjectField = new JTextField();

frame.add(subjectField);

frame.add(new JLabel("Mensaje:"));

messageArea = new JTextArea(5, 20);

frame.add(new JScrollPane(messageArea));

// Botones de selección para TLS

frame.add(new JLabel("Usar TLS:"));

JPanel tlsPanel = new JPanel();

tlsYes = new JRadioButton("Sí", true);

tlsNo = new JRadioButton("No");

ButtonGroup tlsGroup = new ButtonGroup();

tlsGroup.add(tlsYes);

tlsGroup.add(tlsNo);

tlsPanel.add(tlsYes);

tlsPanel.add(tlsNo);

frame.add(tlsPanel);

// Botón de enviar

sendButton = new JButton("Enviar");

frame.add(new JLabel()); // Espaciador

frame.add(sendButton);

// Acción del botón enviar (sin lambda)

sendButton.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent e) {

enviarCorreo();

}

});

frame.setVisible(true);

}

// Método para conectar y enviar correo

private void enviarCorreo() {

String servidor = serverField.getText();

int puerto = Integer.parseInt(portField.getText());

String usuario = userField.getText();

String clave = new String(passwordField.getPassword());

String remitente = senderField.getText();

String destinatario = recipientField.getText();

String asunto = subjectField.getText();

String mensaje = messageArea.getText();

boolean usarTLS = tlsYes.isSelected();

AuthenticatingSMTPClient cliente = new AuthenticatingSMTPClient();

try {

cliente.connect(servidor, puerto);

JOptionPane.showMessageDialog(frame, "Conectado al servidor SMTP.");

KeyManagerFactory kmf = KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());

kmf.init(null, null);

KeyManager km = kmf.getKeyManagers()[0];

cliente.setKeyManager(km);

cliente.ehlo(servidor);

if (usarTLS && !cliente.execTLS()) {

JOptionPane.showMessageDialog(frame, "Error activando TLS.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

if (cliente.auth(AuthenticatingSMTPClient.AUTH\_METHOD.PLAIN, usuario, clave)) {

SimpleSMTPHeader cabecera = new SimpleSMTPHeader(remitente, destinatario, asunto);

cliente.setSender(remitente);

cliente.addRecipient(destinatario);

Writer writer = cliente.sendMessageData();

if (writer != null) {

writer.write(cabecera.toString());

writer.write(mensaje);

writer.close();

if (cliente.completePendingCommand()) {

JOptionPane.showMessageDialog(frame, "Mensaje enviado correctamente.");

} else {

JOptionPane.showMessageDialog(frame, "Error enviando el mensaje.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

} else {

JOptionPane.showMessageDialog(frame, "Error de autenticación.", "Error", JOptionPane.ERROR\_MESSAGE);

}

cliente.disconnect();

} catch (Exception e) {

JOptionPane.showMessageDialog(frame, "Error: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

public static void main(String[] args) {

new ClienteSMTPGUI();

}

}

#### **Para varios**

import org.apache.commons.net.smtp.AuthenticatingSMTPClient;

import org.apache.commons.net.smtp.SimpleSMTPHeader;

import org.apache.commons.net.smtp.SMTPReply;

import javax.net.ssl.KeyManager;

import javax.net.ssl.KeyManagerFactory;

import java.io.Writer;

import java.security.Security;

import java.util.Scanner;

public class EnviarCorreoSMTP {

public static void main(String[] args) {

AuthenticatingSMTPClient cliente = new AuthenticatingSMTPClient();

Scanner scanner = new Scanner(System.in);

System.out.print("Introduce el servidor SMTP: ");

String server = scanner.nextLine();

System.out.print("Usuario: ");

String username = scanner.nextLine();

System.out.print("Password: ");

String password = scanner.nextLine();

System.out.print("Puerto: ");

int puerto = Integer.parseInt(scanner.nextLine());

System.out.print("Introduce el remitente: ");

String remitente = scanner.nextLine();

try {

int respuesta;

cliente.connect(server, puerto);

System.out.println("2 - " + cliente.getReplyString());

KeyManagerFactory kmf = KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());

kmf.init(null, null);

KeyManager km = kmf.getKeyManagers()[0];

cliente.setKeyManager(km);

respuesta = cliente.getReplyCode();

if (!SMTPReply.isPositiveCompletion(respuesta)) {

cliente.disconnect();

System.out.println("Se ha producido un error al conectar con el servidor SMTP");

return;

}

cliente.ehlo(server);

System.out.println("2 - " + cliente.getReplyString());

if (cliente.execTLS()) {

System.out.println("3 - " + cliente.getReplyString());

if (cliente.auth(AuthenticatingSMTPClient.AUTH\_METHOD.PLAIN, username, password)) {

System.out.println("4 - " + cliente.getReplyString());

System.out.print("Introduce los destinatarios (separados por comas): ");

String destinatariosInput = scanner.nextLine();

String[] destinatarios = destinatariosInput.split(",");

System.out.print("Introduce el asunto: ");

String asunto = scanner.nextLine();

System.out.print("Introduce el mensaje: ");

String mensaje = scanner.nextLine();

// Enviar correo a cada destinatario

for (String destino : destinatarios) {

destino = destino.trim(); // Limpiar espacios

if (!destino.isEmpty()) {

System.out.println("Enviando a: " + destino);

SimpleSMTPHeader cabecera = new SimpleSMTPHeader(remitente, destino, "Asunto: " + asunto);

//establece el remitente

cliente.setSender(remitente);

cliente.addRecipient(destino);

System.out.println("5 - " + cliente.getReplyString());

Writer writer = cliente.sendMessageData();

if (writer != null) {

writer.write(cabecera.toString());

writer.write(mensaje);

writer.close();

System.out.println("6 - " + cliente.getReplyString());

boolean exito = cliente.completePendingCommand();

System.out.println("7 - " + cliente.getReplyString());

if (exito) {

System.out.println("Mensaje enviado correctamente a: " + destino);

} else {

System.out.println("Error al enviar a: " + destino);

}

} else {

System.out.println("Fallo al enviar la data a: " + destino);

}

}

}

} else {

System.out.println("Usuario no válido");

}

} else {

System.out.println("Error en TLS");

}

cliente.disconnect();

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

if (cliente.isConnected()) {

cliente.disconnect();

}

} catch (Exception e) {

e.printStackTrace();

}

}

System.out.println("Proceso finalizado");

}

}

### **Acceder a mensajes del servidor**

import org.apache.commons.net.pop3.POP3Client;

import org.apache.commons.net.pop3.POP3MessageInfo;

public class Actividad5 {

public static void main(String[] args) {

// Crear un objeto Scanner para leer la entrada del usuario

Scanner entrada = new Scanner(System.in);

// Solicitar al usuario los datos de conexión

System.out.print("Servidor POP3: ");

String servidorPOP = entrada.next();

System.out.print("Puerto: ");

int puertoPOP = entrada.nextInt();

System.out.print("Usuario: ");

String usuario = entrada.next();

System.out.print("Contraseña: ");

String clave = entrada.next();

// Crear cliente POP3 para conectar con el servidor

POP3Client clientePOP = new POP3Client();

try {

// Intentar conectar con el servidor POP3

clientePOP.connect(servidorPOP, puertoPOP);

System.out.println("Conexión establecida: " + clientePOP.getReplyString());

// Intentar autenticar con el usuario y la contraseña proporcionados

if (!clientePOP.login(usuario, clave)) {

System.err.println("Error: Autenticación fallida");

clientePOP.disconnect(); // Desconectar si la autenticación falla

return;

}

System.out.println("Autenticación correcta");

// Obtener la lista de mensajes en la bandeja de entrada

POP3MessageInfo[] mensajes = clientePOP.listMessages();

if (mensajes == null || mensajes.length == 0) {

System.out.println("No hay correos en la bandeja de entrada");

} else {

System.out.println("Mensajes en la bandeja: " + mensajes.length);

// Iterar sobre los mensajes y leer su contenido

for (POP3MessageInfo mensaje : mensajes) {

System.out.println("Leyendo mensaje " + mensaje.number);

Reader reader = clientePOP.retrieveMessage(mensaje.number); // Obtener el mensaje

if (reader != null) {

try (BufferedReader bufferedReader = new BufferedReader(reader)) {

String linea;

while ((linea = bufferedReader.readLine()) != null) {

System.out.println(linea); // Mostrar el contenido del mensaje

}

}

} else {

System.out.println("No se pudo recuperar el mensaje " + mensaje.number);

}

}

}

} catch (IOException e) {

System.out.println("Error al conectar con el servidor");

e.printStackTrace();

} finally {

// Asegurarse de cerrar la sesión y desconectar del servidor al finalizar

try {

clientePOP.logout(); // Realizar logout del servidor POP3

clientePOP.disconnect(); // Desconectar del servidor

} catch (IOException e) {

e.printStackTrace();

}

}

System.out.println("Proceso finalizado");

}

}

## **FTP con Sockets e Hilos**

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

public class Servidor {

static final int PUERTO = 44441;

static ServerSocket servidor;

public static void main(String[] args) throws IOException {

JFileChooser fileChooser = new JFileChooser();

fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES\_ONLY);

fileChooser.setDialogTitle("Selecciona el directorio raíz del servidor");

int returnVal = fileChooser.showDialog(null, "Seleccionar");

if (returnVal != JFileChooser.APPROVE\_OPTION) {

System.out.println("Debe seleccionar un directorio.");

System.exit(1);

}

File directorioBase = fileChooser.getSelectedFile();

System.out.println("Directorio del servidor: " + directorioBase.getAbsolutePath());

servidor = new ServerSocket(PUERTO);

System.out.println("Servidor iniciado en el puerto " + PUERTO);

while (true) {

try {

Socket cliente = servidor.accept();

System.out.println("Cliente conectado: " + cliente.getInetAddress());

new HiloServidor(cliente, directorioBase).start();

} catch (IOException e) {

System.out.println("Error: " + e.getMessage());

}

}

}

}

class HiloServidor extends Thread {

private Socket socket;

private File directorioActual;

private ObjectOutputStream outObjeto;

private ObjectInputStream inObjeto;

public HiloServidor(Socket s, File directorio) throws IOException {

socket = s;

directorioActual = directorio;

outObjeto = new ObjectOutputStream(socket.getOutputStream());

inObjeto = new ObjectInputStream(socket.getInputStream());

}

public void run() {

try {

while (true) {

Object peticion = inObjeto.readObject();

if (peticion instanceof String) {

String comando = (String) peticion;

if (comando.equals("LISTAR")) {

outObjeto.writeObject(new EstructuraFicheros(directorioActual));

} else if (comando.startsWith("CAMBIAR\_DIR")) {

String nuevoDir = comando.substring(12);

File nuevoDirectorio = new File(directorioActual, nuevoDir);

if (nuevoDirectorio.isDirectory()) {

directorioActual = nuevoDirectorio;

outObjeto.writeObject(new EstructuraFicheros(directorioActual));

} else {

outObjeto.writeObject(null);

}

}

} else if (peticion instanceof PideFichero) {

PideFichero pedido = (PideFichero) peticion;

File fichero = new File(directorioActual, pedido.getNombreFichero());

if (fichero.exists()) {

FileInputStream fileIn = new FileInputStream(fichero);

byte[] contenido = fileIn.readAllBytes();

fileIn.close();

outObjeto.writeObject(new ObtieneFichero(contenido));

}

} else if (peticion instanceof EnviaFichero) {

EnviaFichero archivo = (EnviaFichero) peticion;

File fichero = new File(directorioActual, archivo.getNombre());

FileOutputStream fileOut = new FileOutputStream(fichero);

fileOut.write(archivo.getContenidoFichero());

fileOut.close();

outObjeto.writeObject(new EstructuraFicheros(directorioActual));

}

}

} catch (Exception e) {

System.out.println("Cliente desconectado.");

}

}

}

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

public class Cliente {

private Socket socket;

private ObjectOutputStream outObjeto;

private ObjectInputStream inObjeto;

public Cliente() {

try {

socket = new Socket("localhost", 44441);

outObjeto = new ObjectOutputStream(socket.getOutputStream());

inObjeto = new ObjectInputStream(socket.getInputStream());

} catch (IOException e) {

System.out.println("Error de conexión: " + e.getMessage());

}

}

public void listarArchivos() {

try {

outObjeto.writeObject("LISTAR");

EstructuraFicheros estructura = (EstructuraFicheros) inObjeto.readObject();

System.out.println("Archivos en el directorio:");

for (EstructuraFicheros f : estructura.getLista()) {

System.out.println(f);

}

} catch (Exception e) {

System.out.println("Error al listar archivos.");

}

}

public void cambiarDirectorio(String nombreDir) {

try {

outObjeto.writeObject("CAMBIAR\_DIR " + nombreDir);

EstructuraFicheros estructura = (EstructuraFicheros) inObjeto.readObject();

if (estructura != null) {

System.out.println("Directorio cambiado a: " + nombreDir);

} else {

System.out.println("No se pudo cambiar de directorio.");

}

} catch (Exception e) {

System.out.println("Error al cambiar de directorio.");

}

}

public void descargarArchivo(String nombreFichero) {

try {

outObjeto.writeObject(new PideFichero(nombreFichero));

ObtieneFichero archivo = (ObtieneFichero) inObjeto.readObject();

File file = new File(nombreFichero);

FileOutputStream fos = new FileOutputStream(file);

fos.write(archivo.getContenidoFichero());

fos.close();

System.out.println("Archivo descargado: " + nombreFichero);

} catch (Exception e) {

System.out.println("Error al descargar archivo.");

}

}

public void subirArchivo(String rutaFichero) {

try {

File file = new File(rutaFichero);

FileInputStream fis = new FileInputStream(file);

byte[] contenido = fis.readAllBytes();

fis.close();

outObjeto.writeObject(new EnviaFichero(contenido, file.getName()));

System.out.println("Archivo subido: " + file.getName());

} catch (Exception e) {

System.out.println("Error al subir archivo.");

}

}

public static void main(String[] args) {

Cliente cliente = new Cliente();

cliente.listarArchivos();

cliente.cambiarDirectorio("SubCarpeta");

cliente.descargarArchivo("Ejemplo.txt");

cliente.subirArchivo("C:/ruta/local/ArchivoSubir.txt");

}

}

import java.io.\*;

public class EstructuraFicheros implements Serializable {

private String nombre;

private boolean esDirectorio;

private EstructuraFicheros[] lista;

public EstructuraFicheros(File dir) {

this.nombre = dir.getName();

this.esDirectorio = dir.isDirectory();

File[] archivos = dir.listFiles();

if (archivos != null) {

lista = new EstructuraFicheros[archivos.length];

for (int i = 0; i < archivos.length; i++) {

lista[i] = new EstructuraFicheros(archivos[i]);

}

}

}

public EstructuraFicheros[] getLista() {

return lista;

}

public String toString() {

return esDirectorio ? "[DIR] " + nombre : nombre;

}

}

**Representa un directorio y sus archivos**:

**Ejemplo de usar TLS O NO EJ2 PG 240**

import org.apache.commons.net.smtp.AuthenticatingSMTPClient;

import org.apache.commons.net.smtp.SimpleSMTPHeader;

import java.io.Writer;

import java.util.Scanner;

public class ClienteSMTP {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

AuthenticatingSMTPClient cliente = new AuthenticatingSMTPClient();

System.out.print("Servidor SMTP: ");

String servidor = scanner.nextLine();

System.out.print("Puerto SMTP: ");

int puerto = Integer.parseInt(scanner.nextLine());

System.out.print("Usuario: ");

String usuario = scanner.nextLine();

System.out.print("Contraseña: ");

String clave = scanner.nextLine();

System.out.print("Correo remitente: ");

String remitente = scanner.nextLine();

System.out.print("Correo destinatario: ");

String destinatario = scanner.nextLine();

System.out.print("Asunto: ");

String asunto = scanner.nextLine();

System.out.print("Mensaje: ");

String mensaje = scanner.nextLine();

System.out.print("¿Usar TLS? (si/no): ");

boolean usarTLS = scanner.nextLine().equalsIgnoreCase("si");

try {

// Conectar al servidor

cliente.connect(servidor, puerto);

System.out.println(cliente.getReplyString());

cliente.ehlo(servidor);

System.out.println(cliente.getReplyString());

if (usarTLS) {

if (cliente.execTLS()) {

System.out.println("TLS activado.");

} else {

System.out.println("Error activando TLS.");

return;

}

}

// Autenticación

if (cliente.auth(AuthenticatingSMTPClient.AUTH\_METHOD.PLAIN, usuario, clave)) {

System.out.println("Autenticación correcta.");

SimpleSMTPHeader cabecera = new SimpleSMTPHeader(remitente, destinatario, asunto);

cliente.setSender(remitente);

cliente.addRecipient(destinatario);

Writer writer = cliente.sendMessageData();

if (writer != null) {

writer.write(cabecera.toString());

writer.write(mensaje);

writer.close();

if (cliente.completePendingCommand()) {

System.out.println("Mensaje enviado correctamente.");

} else {

System.out.println("Error enviando el mensaje.");

}

}

} else {

System.out.println("Error de autenticación.");

}

cliente.disconnect();

} catch (Exception e) {

e.printStackTrace();

}

}

}

COM INTERFAZ:

// Botones para elegir TLS o no frame.add(new JLabel("Usar TLS:"));

JPanel tlsPanel = new JPanel();

tlsYes = new JRadioButton("Sí", true); tlsNo = new JRadioButton("No");

ButtonGroup tlsGroup = new ButtonGroup(); tlsGroup.add(tlsYes);

tlsGroup.add(tlsNo); tlsPanel.add(tlsYes); tlsPanel.add(tlsNo);

frame.add(tlsPanel); // Botón de enviar sendButton = new JButton("Enviar");

frame.add(new JLabel()); // Espaciador frame.add(sendButton);

FALTA EL METODO DEL BOTON PARA ENVIAR EL CORRREO

**Método mostrar consola de los mensajes(especie de log)--> addProtocolCommandListener()**

Preguntar chat gpt **ProtocolCommandListener** y **PrintCommandListener pg 190**

Lista para almacenar y mostrar lista de ficheros y carpetas de un directorio en interfaz= JList

JList--> JTextField,JButton,JScrollPane

Almacenar lista de ficheros del servidor= FTP[]File

Listener para selecciona elemento de la lista --> LIstSelectionListener **pg 195**

**EJERCICIO TIPO TEMA 4(combina sockets e hilos)**

**Implementa un servidor FTP en Java que permita a múltiples clientes autenticarse, listar directorios, subir y descargar archivos.**

**CODIGO SIN INTERFAZ**

**import java.io.\*;**

**import java.net.\*;**

**import java.util.\*;**

**import org.apache.commons.net.ftp.\*;**

**public class ServidorFTP {**

**private static final int PUERTO = 2121;**

**private static final String USUARIO = "admin";**

**private static final String PASSWORD = "1234";**

**private static final String DIRECTORIO\_BASE = "ftp\_files";**

**public static void main(String[] args) {**

**File dir = new File(DIRECTORIO\_BASE);**

**if (!dir.exists()) dir.mkdirs(); // Crear directorio base si no existe**

**try (ServerSocket serverSocket = new ServerSocket(PUERTO)) {**

**System.out.println("Servidor FTP iniciado en el puerto " + PUERTO);**

**while (true) {**

**Socket cliente = serverSocket.accept();**

**new HiloCliente(cliente).start();**

**}**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**static class HiloCliente extends Thread {**

**private Socket socket;**

**private BufferedReader in;**

**private PrintWriter out;**

**private FTPClient ftpClient;**

**public HiloCliente(Socket socket) {**

**this.socket = socket;**

**}**

**public void run() {**

**try {**

**in = new BufferedReader(new InputStreamReader(socket.getInputStream()));**

**out = new PrintWriter(socket.getOutputStream(), true);**

**// Autenticación**

**out.println("Ingrese usuario:");**

**String user = in.readLine();**

**out.println("Ingrese contraseña:");**

**String pass = in.readLine();**

**if (!user.equals(USUARIO) || !pass.equals(PASSWORD)) {**

**out.println("Error: Autenticación fallida.");**

**socket.close();**

**return;**

**}**

**out.println("Autenticado con éxito. Bienvenido al servidor FTP.");**

**while (true) {**

**out.println("Comandos: LIST | UPLOAD <nombre> | DOWNLOAD <nombre> | EXIT");**

**String comando = in.readLine();**

**if (comando.equals("LIST")) {**

**File folder = new File(DIRECTORIO\_BASE);**

**String[] archivos = folder.list();**

**if (archivos != null) {**

**out.println(String.join(",", archivos));**

**} else {**

**out.println("No hay archivos.");**

**}**

**} else if (comando.startsWith("UPLOAD ")) {**

**String nombreArchivo = comando.split(" ")[1];**

**File file = new File(DIRECTORIO\_BASE, nombreArchivo);**

**FileOutputStream fos = new FileOutputStream(file);**

**int data;**

**while ((data = socket.getInputStream().read()) != -1) {**

**fos.write(data);**

**}**

**fos.close();**

**out.println("Archivo " + nombreArchivo + " subido con éxito.");**

**} else if (comando.startsWith("DOWNLOAD ")) {**

**String nombreArchivo = comando.split(" ")[1];**

**File file = new File(DIRECTORIO\_BASE, nombreArchivo);**

**if (!file.exists()) {**

**out.println("El archivo no existe.");**

**continue;**

**}**

**FileInputStream fis = new FileInputStream(file);**

**int data;**

**while ((data = fis.read()) != -1) {**

**socket.getOutputStream().write(data);**

**}**

**fis.close();**

**out.println("Archivo enviado.");**

**} else if (comando.equals("EXIT")) {**

**out.println("Desconectando...");**

**socket.close();**

**break;**

**} else {**

**out.println("Comando no reconocido.");**

**}**

**}**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**}**

**CODIGO CON INTERFAZ**

**import javax.swing.\*;**

**import java.io.\*;**

**import java.net.\*;**

**public class ClienteFTP\_GUI {**

**private static final String SERVIDOR = "localhost";**

**private static final int PUERTO = 2121;**

**private Socket socket;**

**private BufferedReader in;**

**private PrintWriter out;**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(() -> new ClienteFTP\_GUI().iniciar());**

**}**

**public void iniciar() {**

**try {**

**socket = new Socket(SERVIDOR, PUERTO);**

**in = new BufferedReader(new InputStreamReader(socket.getInputStream()));**

**out = new PrintWriter(socket.getOutputStream(), true);**

**String respuesta;**

**while ((respuesta = in.readLine()) != null) {**

**if (respuesta.startsWith("Ingrese usuario:")) {**

**String user = JOptionPane.showInputDialog("Usuario:");**

**out.println(user);**

**} else if (respuesta.startsWith("Ingrese contraseña:")) {**

**String pass = JOptionPane.showInputDialog("Contraseña:");**

**out.println(pass);**

**} else if (respuesta.equals("Autenticado con éxito. Bienvenido al servidor FTP.")) {**

**JOptionPane.showMessageDialog(null, respuesta);**

**mostrarInterfaz();**

**break;**

**} else {**

**JOptionPane.showMessageDialog(null, respuesta);**

**}**

**}**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**private void mostrarInterfaz() {**

**JFrame frame = new JFrame("Cliente FTP");**

**JButton listarBtn = new JButton("Listar archivos");**

**JButton subirBtn = new JButton("Subir archivo");**

**JButton descargarBtn = new JButton("Descargar archivo");**

**listarBtn.addActionListener(e -> {**

**out.println("LIST");**

**try {**

**String archivos = in.readLine();**

**JOptionPane.showMessageDialog(frame, "Archivos disponibles:\n" + archivos);**

**} catch (IOException ex) {**

**ex.printStackTrace();**

**}**

**});**

**subirBtn.addActionListener(e -> {**

**JFileChooser fileChooser = new JFileChooser();**

**if (fileChooser.showOpenDialog(frame) == JFileChooser.APPROVE\_OPTION) {**

**out.println("UPLOAD " + fileChooser.getSelectedFile().getName());**

**JOptionPane.showMessageDialog(frame, "Archivo subido.");**

**}**

**});**

**frame.setLayout(new java.awt.FlowLayout());**

**frame.add(listarBtn);**

**frame.add(subirBtn);**

**frame.add(descargarBtn);**

**frame.setSize(300, 200);**

**frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**frame.setVisible(true);**

**}**

**}**