|  |  |  |
| --- | --- | --- |
|  | **INSTITUTO POLITECNICO NACIONAL**  Escuela Superior de Cómputo |  |

Alumnos:

Guerra Vargas Irving Cristobal

Pantoja Flores Pablo

Unidad de Aprendizaje: Aplicaciones para Comunicaciones en Red

Profesor: Axel Ernesto Moreno Cervantes

**PRACTICA 1: EscomDrive**

Grupo: 3CM7

Contenido

[Introducción 3](#_Toc516597762)

[Desarrollo 3](#_Toc516597763)

[Códigos 5](#_Toc516597764)

[Conclusiones 16](#_Toc516597765)

# Introducción

Una de las funciones más utilizadas en las aplicaciones actuales es la posibilidad de enviar archivos a través de la red.

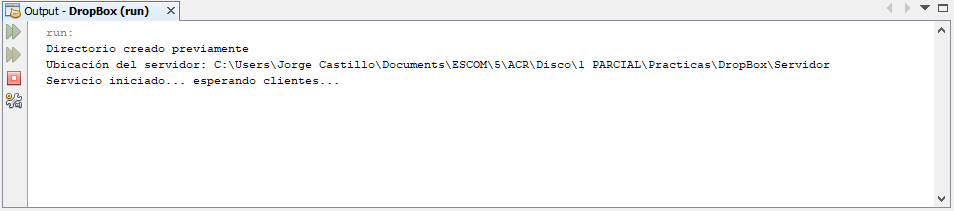
La problemática a la que nos enfrentamos frecuentemente en este tipo de aplicaciones es que existen veces en donde el tamaño del archivo suele ser un impedimento, ya sea porque es demasiado grande o porque la aplicación no va dirigida directamente al envío de archivos.

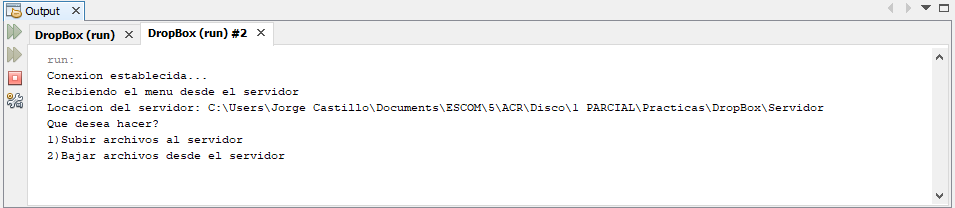
En esta práctica se desarrollará una aplicación que nos permitirá enviar múltiples archivos, independientemente del tamaño de estos y de la cantidad que el usuario seleccione utilizando un algoritmo recursivo.

# Desarrollo

Básicamente para el desarrollo de esta práctica se utilizaron los ejemplos vistos en clase para el envío de un archivo unitario, solamente se amplió el envío a más de un archivo (los seleccionados por el usuario) y enviando los mismos de forma recursiva de tal forma que la estructura en el sistema de archivos original es la misma que aparece en el servidor, lo cual implica la creación de directorios y la obtención de cada ruta de los diferentes archivos.

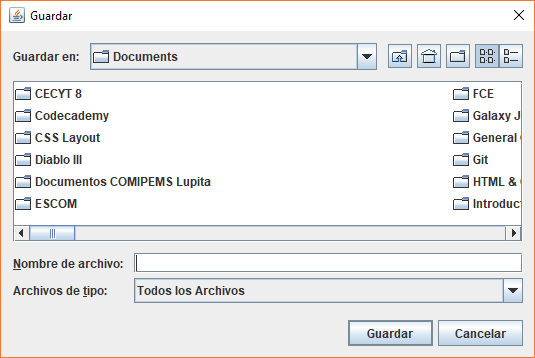
A continuación, se muestra el funcionamiento de esta:



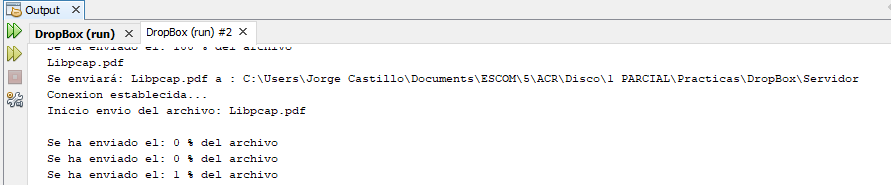


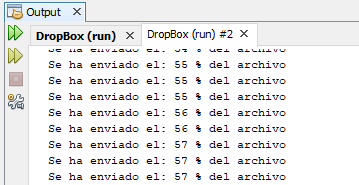
Como se observa, el usuario puede elegir entre subir varios archivos al servidor o bajar archivos que ya se encuentren en éste.

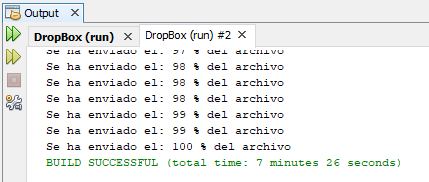
Si elegimos subir directorios al servidor, la aplicación muestra un JFileChooser para que se seleccionen los archivos y directorios.



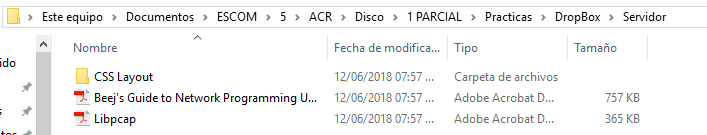
Una vez que se da clic en Guardar comienza el envío al servidor.







Dentro del servidor podemos ver que los archivos se envían satisfactoriamente.



# Códigos

**Servidor.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ventanas;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.util.zip.ZipEntry;

import java.util.zip.ZipOutputStream;

import static javax.swing.JOptionPane.showMessageDialog;

import static jdk.nashorn.internal.objects.ArrayBufferView.buffer;

/\*\*

\*

\* @author irving-mac

\*/

public class Servidor extends javax.swing.JFrame {

int puerto = 1234;

String host = "127.0.0.1";

String listaFiles = "";

ServerSocket s;

Socket cl;

int i = 0;

//Rutas WIN

String cloud = "F:\\ESCOM\\7\\Redes2\\Practica 1\\FilesServer\\src\\cloud";

String pathLocal = "F:\\ESCOM\\7\\Redes2\\Practica 1\\EscomDrive\\src\\local\\";

String containsDownload = "FilesServer\\src\\cloud";

//Rutas MAC

//String cloud = "/Volumes/BLUE/ESCOM/7/Redes2/Practica 1/FilesServer/src/cloud";

//String pathLocal = "/Volumes/BLUE/ESCOM/7/Redes2/Practica 1/EscomDrive/src/local/";

//String containsDownload = "FilesServer/src/cloud";

public Servidor() {

initComponents();

this.setLocationRelativeTo(null);

startService();

esperaFileOrFolder();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents

private void initComponents() {

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 400, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 300, Short.MAX\_VALUE)

);

pack();

}// </editor-fold>//GEN-END:initComponents

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Servidor.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Servidor.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Servidor.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Servidor.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new Servidor().setVisible(true);

}

});

}

private void esperaFileOrFolder() {

try{

for(;;){

//Acepta al cliente

cl = s.accept();

System.out.println("[ OK ] Cliente conectado desde: "+ cl.getInetAddress()+":"+ cl.getPort());

//Leemos la entrada

DataInputStream dis = new DataInputStream(cl.getInputStream());

String nombre = (String) dis.readUTF();

//Si el nombre contiene la palabra "cloud" Significa que nos pide un archivo

if(nombre.contains(containsDownload)){

//Nos pide un archivo o folder

if(nombre.contains(".")){

System.out.println("[ OK ] Transfieriendo archivo...");

File f = new File(nombre);

String name = f.getName();

long tam = f.length();

String path = f.getAbsolutePath();

System.out.println("[ OK ] Enviando archivo: " + name + " que mide " + tam + " bytes\n");

DataOutputStream dos = new DataOutputStream (cl.getOutputStream());

DataInputStream disFile = new DataInputStream(new FileInputStream(path));

//Enviando el archvio

dos.writeUTF(name); //Envia el nombre

dos.flush();

dos.writeLong(tam); //Envia el tamaño

byte[] b =new byte[1500];

long enviados = 0;

int porciento = 0, n=0;

while(enviados < tam){

n = disFile.read(b);

dos.write(b, 0, n);

dos.flush(); //se envian

enviados+=n;

porciento = (int)((enviados\*100)/tam);

System.out.println("[ SEND ] Trasmitido el " + porciento + "%");

}

disFile.close();

dos.close();

cl.close();

System.out.println("[ OK ] Archvio enviado con exito");

dis.close();

cl.close();

}else{

//Creamos el rar

System.out.println("[ OK ] Transfieriendo carpeta en rar...");

//NOS PIDEN UN ZIP

String[] sect = nombre.split("\\\\");

String justName = sect[sect.length-1];

FileOutputStream fos = new FileOutputStream(pathLocal+"\\"+justName+".zip");

ZipOutputStream zos = new ZipOutputStream(fos);

byte[] b =new byte[1500];

addFilesRar(nombre,zos,b);//Carpeta que necesitamos comprimir

zos.close();

dis.close();

cl.close();

}

}else{

//Nos pide almacenar un archivo

System.out.println("[ OK ] Recibiendo archivo...");

//en el nombre se recibe el path entero.

//1 Eliminaremos el path del user, para dejar las posibles carpetas.

nombre = nombre.replace(pathLocal, ""); //Aqui ya solo tiene la ultima ruta.

//Eliminaremos el nombre del archivo. Y crearemos ese directorio.

String[] sect = nombre.split("\\\\");

String justName = sect[sect.length-1];

System.out.println(justName);

String directoryToCreate = nombre.replace(justName, "");

System.out.println(directoryToCreate);

System.out.println(directoryToCreate);

File theDir = new File(cloud+"\\"+directoryToCreate);

// if the directory does not exist, create it

if (!theDir.exists()) {

System.out.println("[ OK ] Creando directorio: " + theDir.getName());

boolean result = false;

try{

theDir.mkdirs();

result = true;

}

catch(SecurityException se){

System.out.println("[ FAIL ] Error al crear el directorio...");

}

if(result) {

System.out.println("[ OK ] Directorio creado");

}

}

long tam = (long) dis.readLong();

DataOutputStream dos = new DataOutputStream (new FileOutputStream(cloud+"\\"+directoryToCreate+"\\"+justName));

byte [] b = new byte [1500];

long recibidos = 0;

int porciento\_recibido = 0, n=0;

while (recibidos < tam){

n = dis.read(b);

dos.write(b, 0, n);

dos.flush();

recibidos +=n;

porciento\_recibido = (int)((recibidos\*100)/tam);

System.out.println("[ OK ] Recibido el " + porciento\_recibido + "%");

}

dos.close();

dis.close();

cl.close();

System.out.println("[ OK ] Archivo recibido");

}

}

} catch(Exception e){

e.printStackTrace();

}

}

private void startService() {

try{

s = new ServerSocket(puerto);

s.setReuseAddress(true);

System.out.println("[ OK ] Servicio Iniciado, esperando cliente...");

} catch(Exception e){

e.printStackTrace();

}

}

private void addFilesRar(String nombre, ZipOutputStream zos, byte[] b) {

File dir = new File(nombre);

File[] files = dir.listFiles();

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

if(files[i].isFile()){

try {

System.out.println("[ OK ] Añadiendo archivo a rar: " + files[i].getName());

FileInputStream fis = new FileInputStream(files[i]);

zos.putNextEntry(new ZipEntry(files[i].getName()));

int length;

while ((length = fis.read(b)) > 0) {

zos.write(b, 0, length);

}

zos.closeEntry();

fis.close();

} catch (IOException ioe) {

System.out.println("[ FAIL ] Error al crear el zip...");

}

}else if(files[i].isDirectory()){

addFilesRar(files[i].getAbsolutePath(), zos, b);

}

}

}

// Variables declaration - do not modify//GEN-BEGIN:variables

// End of variables declaration//GEN-END:variables

}

**Interfaz.java (Cliente)**

**package ventanas;**

**import code.DragListener;**

**import java.awt.dnd.DropTarget;**

**import java.io.DataInputStream;**

**import java.io.DataOutputStream;**

**import java.io.File;**

**import java.io.FileInputStream;**

**import java.io.FileOutputStream;**

**import java.io.IOException;**

**import java.net.ServerSocket;**

**import java.net.Socket;**

**import java.nio.file.Files;**

**import java.util.logging.Level;**

**import java.util.logging.Logger;**

**import java.util.zip.ZipInputStream;**

**import java.util.zip.ZipOutputStream;**

**import javafx.scene.control.SelectionModel;**

**import javax.swing.JFileChooser;**

**import static javax.swing.JOptionPane.showMessageDialog;**

**import javax.swing.JScrollPane;**

**import javax.swing.JTree;**

**import javax.swing.event.TreeSelectionEvent;**

**import javax.swing.event.TreeSelectionListener;**

**import javax.swing.tree.DefaultMutableTreeNode;**

**import javax.swing.tree.DefaultTreeModel;**

**public class Interfaz extends javax.swing.JFrame {**

**//Variables para las conexiones**

**int puerto = 1234;**

**String host="127.0.0.1";**

**//Variable auxiliar que nos dice si se subira o descargara una carpeta.**

**boolean folderUpload = false;**

**boolean folderDownload = false;**

**//Variable para visualizar rutas locales en el arbol**

**private DefaultMutableTreeNode rootLocal;**

**private DefaultTreeModel treeModelLocal;**

**private JTree treeLocal;**

**private String fileLocalToUpload = "";**

**//Variable para visualizar rutas cloud en el arbol**

**private DefaultMutableTreeNode rootCloud;**

**private DefaultTreeModel treeModelCloud;**

**private JTree treeCloud;**

**private String fileCloudToDownload = "";**

**//Rutas WIN**

**//String cloud = "F:\\ESCOM\\7\\Redes2\\Practica 1\\FilesServer\\src\\cloud";**

**//String local = "F:\\ESCOM\\7\\Redes2\\Practica 1\\EscomDrive\\src\\local";**

**//String completePathCloud = "F:\\ESCOM\\7\\Redes2\\Practica 1\\FilesServer\\src\\";**

**//String completePathLocal = "F:\\ESCOM\\7\\Redes2\\Practica 1\\EscomDrive\\src\\";**

**//Rutas MAC**

**String cloud = "/Volumes/BLUE/ESCOM/7/Redes2/Practica 1/FilesServer/src/cloud";**

**String local = "/Volumes/BLUE/ESCOM/7/Redes2/Practica 1/EscomDrive/src/local";**

**String completePathCloud = "/Volumes/BLUE/ESCOM/7/Redes2/Practica 1/FilesServer/src/";**

**String completePathLocal = "/Volumes/BLUE/ESCOM/7/Redes2/Practica 1/EscomDrive/src/";**

**//Socket**

**Socket cl;**

**/\*\***

**\* Creates new form Interfaz**

**\*/**

**int x,y;**

**public Interfaz() {**

**initComponents();**

**this.setLocationRelativeTo(null);**

**connectToDragDrop(); //Activa hacer drag para subir archvios desde PC**

**//Empezaremos cargando su local y cloud**

**loadLocalRepository();**

**loadCloudRepository();**

**}**

**private void connectToDragDrop(){**

**DragListener d = new DragListener();**

**new DropTarget(this.addFiles,d);**

**}**

**/\*\***

**\* This method is called from within the constructor to initialize the form.**

**\* WARNING: Do NOT modify this code. The content of this method is always**

**\* regenerated by the Form Editor.**

**\*/**

**@SuppressWarnings("unchecked")**

**// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents**

**private void initComponents() {**

**jSplitPane1 = new javax.swing.JSplitPane();**

**jSplitPane2 = new javax.swing.JSplitPane();**

**btnSalir = new javax.swing.JButton();**

**btnSeleccionarArchivo = new javax.swing.JButton();**

**addFiles = new javax.swing.JLabel();**

**ventanaLocal = new javax.swing.JInternalFrame();**

**ventanaCloud = new javax.swing.JInternalFrame();**

**btnUploadLocal = new javax.swing.JButton();**

**progressBar = new javax.swing.JProgressBar();**

**btnDownloadCloud = new javax.swing.JButton();**

**FondoServidor = new javax.swing.JLabel();**

**setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);**

**setUndecorated(true);**

**getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());**

**btnSalir.setText("Salir");**

**btnSalir.addActionListener(new java.awt.event.ActionListener() {**

**public void actionPerformed(java.awt.event.ActionEvent evt) {**

**btnSalirActionPerformed(evt);**

**}**

**});**

**getContentPane().add(btnSalir, new org.netbeans.lib.awtextra.AbsoluteConstraints(750, 490, 70, 30));**

**btnSeleccionarArchivo.setText("Seleccionar Archivo...");**

**btnSeleccionarArchivo.setToolTipText("");**

**btnSeleccionarArchivo.addActionListener(new java.awt.event.ActionListener() {**

**public void actionPerformed(java.awt.event.ActionEvent evt) {**

**btnSeleccionarArchivoActionPerformed(evt);**

**}**

**});**

**getContentPane().add(btnSeleccionarArchivo, new org.netbeans.lib.awtextra.AbsoluteConstraints(70, 310, 240, 30));**

**addFiles.setIcon(new javax.swing.ImageIcon(getClass().getResource("/imagenes/add.png"))); // NOI18N**

**getContentPane().add(addFiles, new org.netbeans.lib.awtextra.AbsoluteConstraints(120, 370, 130, 130));**

**ventanaLocal.setVisible(true);**

**getContentPane().add(ventanaLocal, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 20, 380, 240));**

**ventanaCloud.setVisible(true);**

**getContentPane().add(ventanaCloud, new org.netbeans.lib.awtextra.AbsoluteConstraints(410, 20, 400, 240));**

**btnUploadLocal.setText("Subir archivo seleccionado");**

**btnUploadLocal.addActionListener(new java.awt.event.ActionListener() {**

**public void actionPerformed(java.awt.event.ActionEvent evt) {**

**btnUploadLocalActionPerformed(evt);**

**}**

**});**

**getContentPane().add(btnUploadLocal, new org.netbeans.lib.awtextra.AbsoluteConstraints(30, 270, 310, 30));**

**getContentPane().add(progressBar, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 530, 830, -1));**

**btnDownloadCloud.setText("Descargar archvio seleccionado");**

**btnDownloadCloud.addActionListener(new java.awt.event.ActionListener() {**

**public void actionPerformed(java.awt.event.ActionEvent evt) {**

**btnDownloadCloudActionPerformed(evt);**

**}**

**});**

**getContentPane().add(btnDownloadCloud, new org.netbeans.lib.awtextra.AbsoluteConstraints(440, 270, 350, 30));**

**FondoServidor.setIcon(new javax.swing.ImageIcon(getClass().getResource("/imagenes/fondo.png"))); // NOI18N**

**FondoServidor.setText("Fondo");**

**FondoServidor.addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {**

**public void mouseDragged(java.awt.event.MouseEvent evt) {**

**FondoServidorMouseDragged(evt);**

**}**

**});**

**FondoServidor.addMouseListener(new java.awt.event.MouseAdapter() {**

**public void mousePressed(java.awt.event.MouseEvent evt) {**

**FondoServidorMousePressed(evt);**

**}**

**});**

**getContentPane().add(FondoServidor, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 830, 550));**

**pack();**

**}// </editor-fold>//GEN-END:initComponents**

**private void btnSalirActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnSalirActionPerformed**

**System.exit(0);**

**}//GEN-LAST:event\_btnSalirActionPerformed**

**private void FondoServidorMousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_FondoServidorMousePressed**

**x = evt.getX();**

**y = evt.getY();**

**}//GEN-LAST:event\_FondoServidorMousePressed**

**private void FondoServidorMouseDragged(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_FondoServidorMouseDragged**

**this.setLocation(this.getLocation().x + evt.getX() -x,this.getLocation().y + evt.getY() -y);**

**}//GEN-LAST:event\_FondoServidorMouseDragged**

**private void btnSeleccionarArchivoActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnSeleccionarArchivoActionPerformed**

**JFileChooser jf = new JFileChooser();**

**//jf.setMultiSelectionEnabled(true);//seleccionar uno o mas**

**int r = jf.showOpenDialog(null);**

**jf.requestFocus();**

**if (r == JFileChooser.APPROVE\_OPTION){**

**File f = jf.getSelectedFile(); //me permite escoger un archivo**

**//jf.getSelectedFiles();**

**String name = f.getName();**

**long tam = f.length();**

**String path = f.getAbsolutePath();**

**uploadFileFromBrowser(name,tam,path);**

**}**

**}//GEN-LAST:event\_btnSeleccionarArchivoActionPerformed**

**private void btnUploadLocalActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnUploadLocalActionPerformed**

**if (fileLocalToUpload == "" || fileLocalToUpload == null){**

**showMessageDialog(null, "Selecciona un archivo");**

**}else{**

**if(folderUpload == false){ //Se subira un file**

**File f = new File(fileLocalToUpload);**

**String name = f.getName();**

**long tam = f.length();**

**String path = f.getAbsolutePath();**

**uploadFileFromBrowser(name,tam,path);**

**}else{ //Se subira una carpeta entera**

**uploadFolderAndFiles(fileLocalToUpload);**

**showMessageDialog(null, "Carpeta subida con exito");**

**}**

**}**

**}//GEN-LAST:event\_btnUploadLocalActionPerformed**

**private void btnDownloadCloudActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnDownloadCloudActionPerformed**

**if(fileCloudToDownload == "" || fileCloudToDownload == null){**

**showMessageDialog(null, "Seleccione un archivo a descargar");**

**}else{**

**//Pediremos al server que nos devuelva ese archivo o file**

**if(folderDownload == false){ //Pedimos un file**

**connectWithServer();**

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

**try {**

**System.out.println("[ OK ] Pidiendo archivo: " + fileCloudToDownload);**

**DataOutputStream dos = new DataOutputStream (cl.getOutputStream());**

**//Enviando el archvio**

**dos.writeUTF(fileCloudToDownload); //Enviamos el file que queremos**

**dos.flush();**

**//Ya enviado, esperamos la respuesta**

**DataInputStream dis = new DataInputStream(cl.getInputStream());**

**//Recibimos el file.**

**String nombre = (String) dis.readUTF();**

**long tam = (long) dis.readLong();**

**DataOutputStream dosFile = new DataOutputStream (new FileOutputStream(local+"/"+nombre));**

**byte [] b = new byte [1500];**

**long recibidos = 0;**

**int porciento\_recibido = 0, n=0;**

**while (recibidos < tam){**

**n = dis.read(b);**

**dosFile.write(b, 0, n);**

**dosFile.flush();**

**recibidos +=n;**

**porciento\_recibido = (int)((recibidos\*100)/tam);**

**System.out.println("[ OK ] Recibido el " + porciento\_recibido + "%");**

**progressBar.setValue(porciento\_recibido);**

**progressBar.setStringPainted(true);**

**}**

**dos.close();**

**dosFile.close();**

**dis.close();**

**cl.close();**

**System.out.println("[ OK ] Archivo recibido");**

**showMessageDialog(null, "Archivo recibido con exito");**

**} catch (IOException ex) {**

**showMessageDialog(null, "Ocurrio un problema al subir el archvio");**

**}**

**}**

**}else{ //Descargaremos el folder en rar**

**connectWithServer();**

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

**//Se pide un zip**

**try {**

**System.out.println("[ OK ] Pidiendo zip: " + fileCloudToDownload);**

**DataOutputStream dos = new DataOutputStream (cl.getOutputStream());**

**//Enviando el archvio**

**dos.writeUTF(fileCloudToDownload); //Enviamos el file que queremos**

**dos.flush();**

**dos.close();**

**cl.close();**

**System.out.println("[ OK ] Zip recibido");**

**showMessageDialog(null, "Zip recibido con exito");**

**} catch (IOException ex) {**

**showMessageDialog(null, "Ocurrio un problema al descargar el zip");**

**}**

**}**

**}**

**ReloadLocalRepository();**

**}**

**}//GEN-LAST:event\_btnDownloadCloudActionPerformed**

**/\*\***

**\* @param args the command line arguments**

**\*/**

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

**/\* Set the Nimbus look and feel \*/**

**//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">**

**/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.**

**\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html**

**\*/**

**try {**

**for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {**

**if ("Nimbus".equals(info.getName())) {**

**javax.swing.UIManager.setLookAndFeel(info.getClassName());**

**break;**

**}**

**}**

**} catch (ClassNotFoundException ex) {**

**java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);**

**} catch (InstantiationException ex) {**

**java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);**

**} catch (IllegalAccessException ex) {**

**java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);**

**} catch (javax.swing.UnsupportedLookAndFeelException ex) {**

**java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);**

**}**

**//</editor-fold>**

**/\* Create and display the form \*/**

**java.awt.EventQueue.invokeLater(new Runnable() {**

**public void run() {**

**new Interfaz().setVisible(true);**

**}**

**});**

**}**

**// Variables declaration - do not modify//GEN-BEGIN:variables**

**private javax.swing.JLabel FondoServidor;**

**private javax.swing.JLabel addFiles;**

**private javax.swing.JButton btnDownloadCloud;**

**private javax.swing.JButton btnSalir;**

**private javax.swing.JButton btnSeleccionarArchivo;**

**private javax.swing.JButton btnUploadLocal;**

**private javax.swing.JSplitPane jSplitPane1;**

**private javax.swing.JSplitPane jSplitPane2;**

**private javax.swing.JProgressBar progressBar;**

**private javax.swing.JInternalFrame ventanaCloud;**

**private javax.swing.JInternalFrame ventanaLocal;**

**// End of variables declaration//GEN-END:variables**

**private void loadLocalRepository() {**

**File fileRoot = new File(local);**

**rootLocal = new DefaultMutableTreeNode(new FileNode(fileRoot));**

**treeModelLocal = new DefaultTreeModel(rootLocal);**

**treeLocal = new JTree(treeModelLocal);**

**treeLocal.setShowsRootHandles(true);**

**JScrollPane scrollPane = new JScrollPane(treeLocal);**

**ventanaLocal.add(scrollPane);**

**ventanaLocal.setSize(400, 240);**

**ventanaLocal.setVisible(true);**

**CreateChildNodes ccn = new CreateChildNodes(fileRoot, rootLocal);**

**new Thread(ccn).start();**

**treeLocal.addTreeSelectionListener(new TreeSelectionListener() {**

**@Override**

**public void valueChanged(TreeSelectionEvent e) {**

**fileLocalToUpload = "";**

**Object[] paths = treeLocal.getSelectionPath().getPath();**

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

**fileLocalToUpload += paths[i];**

**if (i+1 <paths.length ) {**

**fileLocalToUpload += File.separator;**

**}**

**}**

**if(fileLocalToUpload.contains(".")){ //Si contiene un punto es un archivo**

**folderUpload = false;**

**btnUploadLocal.setText("Subir archivo seleccionado");**

**}else{**

**folderUpload = true;**

**btnUploadLocal.setText("Subir carpeta seleccionada");**

**}**

**fileLocalToUpload = completePathLocal+fileLocalToUpload;**

**//System.out.println(fileLocalToUpload);**

**}**

**});**

**}**

**private void ReloadLocalRepository() {**

**File fileRoot = new File(local);**

**rootLocal = new DefaultMutableTreeNode(new FileNode(fileRoot));**

**treeModelLocal = new DefaultTreeModel(rootLocal);**

**treeLocal = new JTree(treeModelLocal);**

**treeLocal.setShowsRootHandles(true);**

**JScrollPane scrollPane = new JScrollPane(treeLocal);**

**ventanaLocal.getContentPane().removeAll();**

**ventanaLocal.add(scrollPane);**

**ventanaLocal.setSize(400, 240);**

**ventanaLocal.setVisible(true);**

**CreateChildNodes ccn = new CreateChildNodes(fileRoot, rootLocal);**

**new Thread(ccn).start();**

**treeLocal.addTreeSelectionListener(new TreeSelectionListener() {**

**@Override**

**public void valueChanged(TreeSelectionEvent e) {**

**fileLocalToUpload = "";**

**Object[] paths = treeLocal.getSelectionPath().getPath();**

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

**fileLocalToUpload += paths[i];**

**if (i+1 <paths.length ) {**

**fileLocalToUpload += File.separator;**

**}**

**}**

**if(fileLocalToUpload.contains(".")){ //Si contiene un punto es un archivo**

**folderUpload = false;**

**btnUploadLocal.setText("Subir archivo seleccionado");**

**}else{**

**folderUpload = true;**

**btnUploadLocal.setText("Subir carpeta seleccionada");**

**}**

**fileLocalToUpload = completePathLocal+fileLocalToUpload;**

**//System.out.println(fileLocalToUpload);**

**}**

**});**

**}**

**private void loadCloudRepository() {**

**File fileRoot = new File(cloud);**

**rootCloud = new DefaultMutableTreeNode(new FileNode(fileRoot));**

**treeModelCloud = new DefaultTreeModel(rootCloud);**

**treeCloud = new JTree(treeModelCloud);**

**treeCloud.setShowsRootHandles(true);**

**JScrollPane scrollPane = new JScrollPane(treeCloud);**

**ventanaCloud.add(scrollPane);**

**ventanaCloud.setSize(400, 240);**

**ventanaCloud.setVisible(true);**

**CreateChildNodes ccn = new CreateChildNodes(fileRoot, rootCloud);**

**new Thread(ccn).start();**

**treeCloud.addTreeSelectionListener(new TreeSelectionListener() {**

**@Override**

**public void valueChanged(TreeSelectionEvent e) {**

**fileCloudToDownload = "";**

**Object[] paths = treeCloud.getSelectionPath().getPath();**

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

**fileCloudToDownload += paths[i];**

**if (i+1 <paths.length ) {**

**fileCloudToDownload += File.separator;**

**}**

**}**

**fileCloudToDownload = completePathCloud+fileCloudToDownload;**

**if(fileCloudToDownload.contains(".")){ //Si contiene un punto es un archivo**

**folderDownload = false;**

**btnDownloadCloud.setText("Descargar archivo seleccionado");**

**}else{**

**folderDownload = true;**

**btnDownloadCloud.setText("Descargar carpeta seleccionada");**

**}**

**System.out.println(fileCloudToDownload);**

**}**

**});**

**}**

**private void ReloadCloudRepository() {**

**File fileRoot = new File(cloud);**

**rootCloud = new DefaultMutableTreeNode(new FileNode(fileRoot));**

**treeModelCloud = new DefaultTreeModel(rootCloud);**

**treeCloud = new JTree(treeModelCloud);**

**treeCloud.setShowsRootHandles(true);**

**JScrollPane scrollPane = new JScrollPane(treeCloud);**

**ventanaCloud.getContentPane().removeAll();**

**ventanaCloud.add(scrollPane);**

**ventanaCloud.setSize(400, 240);**

**ventanaCloud.setVisible(true);**

**CreateChildNodes ccn = new CreateChildNodes(fileRoot, rootCloud);**

**new Thread(ccn).start();**

**}**

**private void uploadFile(String name, long tam, String path) {**

**connectWithServer();**

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

**try {**

**System.out.println("[ OK ] Enviando archivo: " + path + " que mide " + tam + " bytes\n");**

**DataOutputStream dos = new DataOutputStream (cl.getOutputStream());**

**DataInputStream dis = new DataInputStream(new FileInputStream(path));**

**//Enviando el archvio**

**dos.writeUTF(path); //Envia el nombre**

**dos.flush();**

**dos.writeLong(tam); //Envia el tamaño**

**byte[] b =new byte[1500];**

**long enviados = 0;**

**int porciento = 0, n=0;**

**while(enviados < tam){**

**n = dis.read(b);**

**dos.write(b, 0, n);**

**dos.flush(); //se envian**

**enviados+=n;**

**porciento = (int)((enviados\*100)/tam);**

**System.out.println("[ SEND ] Trasmitido el " + porciento + "%");**

**progressBar.setValue(porciento);**

**progressBar.setStringPainted(true);**

**}**

**dis.close();**

**dos.close();**

**cl.close();**

**System.out.println("[ OK ] Archvio enviado con exito");**

**if(folderUpload == false){**

**showMessageDialog(null, "Archivo enviado con exito");**

**}**

**} catch (IOException ex) {**

**showMessageDialog(null, "Ocurrio un problema al subir el archvio");**

**}**

**}**

**ReloadCloudRepository();**

**}**

**private void uploadFileFromBrowser(String name, long tam, String path) {**

**connectWithServer();**

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

**try {**

**System.out.println("[ OK ] Enviando archivo: " + path + " que mide " + tam + " bytes\n");**

**DataOutputStream dos = new DataOutputStream (cl.getOutputStream());**

**DataInputStream dis = new DataInputStream(new FileInputStream(path));**

**//Enviando el archvio**

**dos.writeUTF(name); //Envia el nombre**

**dos.flush();**

**dos.writeLong(tam); //Envia el tamaño**

**byte[] b =new byte[1500];**

**long enviados = 0;**

**int porciento = 0, n=0;**

**while(enviados < tam){**

**n = dis.read(b);**

**dos.write(b, 0, n);**

**dos.flush(); //se envian**

**enviados+=n;**

**porciento = (int)((enviados\*100)/tam);**

**System.out.println("[ SEND ] Trasmitido el " + porciento + "%");**

**progressBar.setValue(porciento);**

**progressBar.setStringPainted(true);**

**}**

**dis.close();**

**dos.close();**

**cl.close();**

**System.out.println("[ OK ] Archvio enviado con exito");**

**if(folderUpload == false){**

**showMessageDialog(null, "Archivo enviado con exito");**

**}**

**} catch (IOException ex) {**

**showMessageDialog(null, "Ocurrio un problema al subir el archvio");**

**}**

**}**

**ReloadCloudRepository();**

**}**

**private void uploadFolderAndFiles(String path) {**

**File folder = new File(path);**

**File[] listOfFiles = folder.listFiles();**

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

**if (listOfFiles[i].isFile()) {**

**File f = new File(listOfFiles[i].getAbsolutePath());**

**String name = f.getName();**

**long tam = f.length();**

**String pathFile = f.getAbsolutePath();**

**uploadFile(name,tam,pathFile);**

**} else if (listOfFiles[i].isDirectory()) {**

**uploadFolderAndFiles(listOfFiles[i].getAbsolutePath());**

**}**

**}**

**}**

**private void connectWithServer() {**

**//Nos conectamos con els ervidor local.**

**try{**

**cl = new Socket (host, puerto); //socket bloquante**

**System.out.println("[ OK ] Conectado!");**

**}catch(Exception e){**

**showMessageDialog(null, "Ocurrio un problema con el servidor");**

**}**

**}**

**public class CreateChildNodes implements Runnable {**

**private DefaultMutableTreeNode root;**

**private File fileRoot;**

**public CreateChildNodes(File fileRoot,**

**DefaultMutableTreeNode root) {**

**this.fileRoot = fileRoot;**

**this.root = root;**

**}**

**@Override**

**public void run() {**

**createChildren(fileRoot, root);**

**}**

**private void createChildren(File fileRoot,**

**DefaultMutableTreeNode node) {**

**File[] files = fileRoot.listFiles();**

**if (files == null) return;**

**for (File file : files) {**

**DefaultMutableTreeNode childNode =**

**new DefaultMutableTreeNode(new FileNode(file));**

**node.add(childNode);**

**if (file.isDirectory()) {**

**createChildren(file, childNode);**

**}**

**}**

**}**

**}**

**public class FileNode {**

**private File file;**

**public FileNode(File file) {**

**this.file = file;**

**}**

**@Override**

**public String toString() {**

**String name = file.getName();**

**if (name.equals("")) {**

**return file.getAbsolutePath();**

**} else {**

**return name;**

**}**

**}**

**}**

**}**

# Conclusiones

En esta práctica implementamos una aplicación que envía múltiples archivos a través de la red. Esto nos sirvió para darnos una primera idea de cómo funciona este proceso de transferencia.

También nos dimos cuenta de que los archivos deben de ser enviados en orden y que puede que se pierda un pedazo de estos y es por ello que debemos verificar si llegaron en orden y completos.