**ACTIVIDAD 4**

**Enunciado**

Analiza un programa que hace las funciones de cliente FTP básico en base al código Java incluido más adelante.

Como servidor FTP se usará *proFTPd*, el cual se instalará en una máquina virtual con Ubuntu, la cual tendrá configurada una de sus interfaces de red en *modo puente* para posibilitar la conexión con el cliente FTP de Java.

El usuario y contraseña con los que acceder al servidor son *dinux* en ambos casos. La dirección IP del servidor variará según la persona que realice la actividad, aunque en el código de ese programa se utiliza la IP 192.168.1.135.

El programa permitirá subir, descargar y eliminar ficheros así como crear y eliminar directorios o carpetas en un servidor FTP.

En cuanto al **diseño del programa:**

* Se ha usado el IDE Eclipse, complementado con la biblioteca gráfica Swing.
* Se ha añadido código en el main, initialize y constructor.
* En la parte superior de la pantalla hay 3 cajas de texto para mostrar la identificación del servidor, el usuario conectado y el directorio raíz del servidor.
* En la parte central de pantalla hay una Lista que muestra la lista de ficheros y carpetas del directorio actual. Estos se almacenan primero en el JList y luego se visualizan.
* En la parte inferior de la pantalla se muestran los mensajes que van surgiendo según se navega por las carpetas. Al hacer clic en un directorio se muestra su nombre y el número de elementos que tiene. En el otro mensaje se muestra el nombre del directorio actual. Al hacer clic en un fichero se muestra su nombre y directorio donde está.
* En la parte derecha de la pantalla hay 6 botones para subir, descargar y eliminar ficheros; crear y eliminar carpetas, y finalizar el programa.
* La consola va mostrando los mensajes que se van originando en la comunicación con el servidor

En cuanto a la **implementación de la lista**:

* Se hace usando el método *llenarLista()*.
* Para distinguir los ficheros de los directorios, a estos últimos se les va a poner por delante un “(DIR)” seguido de un espacio en blanco.
* Al hacer clic en el directorio se visualiza automáticamente su contenido.
* El nombre del directorio será el primer elemento de la lista JList. Al hacer clic de nuevo en él, se visualizará el contenido del directorio padre.
* Para saber cuándo se selecciona un elemento de la lista hay que implementar ListSelectionListener y desarrollar el método *valueChanged()*, que recibe un objeto ListSelectionEvent.
* Para saber si el usuario ha seleccionado un elemento, se usa el método *getValueIsAdjusting(),* que recibe el valor true si el usuario ha pulsado sobre él.

En cuanto a la **implementación de los botones** se ha seguido el siguiente orden:

* Botón Salir
* Botón Subir Fichero
* Botón Descargar Fichero
* Botón Eliminar Fichero
* Botón Crear Carpeta
* Botón Eliminar Carpeta

En cuanto a la **implementación de los métodos** asociados a ciertos botones,se ha seguido el siguiente orden:

* Método SubirFichero()
* Método DescargarFichero()
* Método BorrarFichero()

El **código del programa** Java es el siguiente:

**import** java.awt.EventQueue;

**import** java.awt.Font;

**import** javax.swing.JFrame;

**import** javax.swing.DefaultListModel;

**import** javax.swing.JButton;

**import** javax.swing.JFileChooser;

**import** java.awt.Color;

**import** java.awt.event.ActionListener;

**import** java.io.BufferedInputStream;

**import** java.io.BufferedOutputStream;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.awt.event.ActionEvent;

**import** javax.swing.JTextField;

**import** org.apache.commons.net.PrintCommandListener;

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

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

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

**import** javax.swing.JList;

**import** javax.swing.JOptionPane;

**import** javax.swing.JScrollPane;

**import** javax.swing.event.ListSelectionEvent;

**import** javax.swing.event.ListSelectionListener;

**public** **class** ClienteFTPBasico **extends** JFrame {

**private** **static** **final** **long** ***serialVersionUID***=1L;

**private** JFrame frame;

**static** JTextField *cab*;

**static** JTextField *cab3*;

**static** JTextField *cab2*;

**static** JList *listaDirec*;

// Datos del servidor FTP

**static** FTPClient *cliente* = **new** FTPClient();

String servidor = "192.168.1.135";

String user = "dinux";

String pasw = "dinux";

**boolean** login;

**static** String *direcInicial* = "/";

//Para saber el directorio y fichero seleccionado

**static** String *direcSelec* = *direcInicial*;

**static** String *ficheroSelec* = "";

**private** JTextField campo;

**private** JTextField campo2;

// Lanzar la aplicación

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

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

**public** **void** run() {

**try** {

ClienteFTPBasico window = **new** ClienteFTPBasico();

window.frame.setVisible(**true**);

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

// Crear la aplicación

**public** ClienteFTPBasico() **throws** IOException {

**super** ("CLIENTE BASICO FTP");

initialize();

// Preparar campos de pantalla

campo.setText("<<ARBOL DE DIRECTORIOS CONSTRUIDO >>");

*cab*.setText("Servidor FTP: "+servidor);

*cab2*.setText("Usuario: " + user);

}

// Inicializar los contenidos del frame

**private** **void** initialize() **throws** IOException {

// Mostrar en consola los mensajes devueltos por el servidor

*cliente*.addProtocolCommandListener(**new** PrintCommandListener (**new** PrintWriter (System.***out***)));

System.***out***.println("Hola " + servidor);

*cliente*.connect(servidor); // Conexión al servidor

System.***out***.println("Hola ");

login = *cliente*.login(user, pasw);

// Establecer el directorio de trabajo actual

*cliente*.changeWorkingDirectory(*direcInicial*);

// Obtener ficheros y directorios del directorio actual

FTPFile[] files = *cliente*.listFiles();

frame = **new** JFrame();

frame.setBounds(100, 100, 781, 404);

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

frame.getContentPane().setLayout(**null**);

JButton botonCargar = **new** JButton("Suvir Fichero");

botonCargar.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

}

});

botonCargar.setBounds(632, 155, 143, 29);

frame.getContentPane().add(botonCargar);

JButton botonDescargar = **new** JButton("Descargar Fichero");

botonDescargar.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

}

});

botonDescargar.setBounds(632, 186, 143, 23);

frame.getContentPane().add(botonDescargar);

JButton botonBorrar = **new** JButton("Eliminar Ficheró");

botonBorrar.setBounds(632, 207, 143, 29);

frame.getContentPane().add(botonBorrar);

JButton botonCreaDir = **new** JButton("Crear Carpeta");

botonCreaDir.setBounds(632, 232, 143, 29);

frame.getContentPane().add(botonCreaDir);

JButton botonDelDir = **new** JButton("Eliminar Carpeta");

botonDelDir.setBounds(632, 260, 143, 29);

frame.getContentPane().add(botonDelDir);

JButton botonSalir = **new** JButton("Salir");

botonSalir.setBounds(632, 285, 143, 29);

frame.getContentPane().add(botonSalir);

*cab* = **new** JTextField();

*cab*.setBounds(28, 6, 363, 28);

frame.getContentPane().add(*cab*);

*cab*.setColumns(10);

*cab3* = **new** JTextField();

*cab3*.setBounds(28, 46, 363, 28);

frame.getContentPane().add(*cab3*);

*cab3*.setColumns(10);

*cab2* = **new** JTextField();

*cab2*.setBounds(403, 6, 256, 28);

frame.getContentPane().add(*cab2*);

*cab2*.setColumns(10);

*cab3*.setText("DIRECTORIO RAÍZ: "+ *direcInicial*);

campo = **new** JTextField();

campo.setColumns(10);

campo.setBounds(28, 271, 397, 28);

frame.getContentPane().add(campo);

campo2 = **new** JTextField();

campo2.setColumns(10);

campo2.setBounds(28, 299, 397, 28);

frame.getContentPane().add(campo2);

*listaDirec* = **new** JList();

*listaDirec*.setBounds(0, 134, 57, -51);

frame.getContentPane().add(*listaDirec*);

JScrollPane barraDesplazamiento = **new** JScrollPane(*listaDirec*);

barraDesplazamiento.setBounds(35, 102, 500, 155);

frame.getContentPane().add(barraDesplazamiento);

frame.getContentPane().setLayout(**null**);

// Construir la lista de ficheros y directorios del diretorio de trabajo actual

*llenarLista*(files, *direcInicial*);

// Implementar el clic sobre la lista

*listaDirec*.addListSelectionListener(**new** ListSelectionListener() {

@Override

**public** **void** valueChanged(ListSelectionEvent lse) {

// **TODO** Auto-generated method stub

**if** (lse.getValueIsAdjusting()) {

*ficheroSelec*="";

// Elemento que se ha seleccionado de la lista

String fic=*listaDirec*.getSelectedValue().toString();

**if** (*listaDirec*.getSelectedIndex() == 0) {

// Hacer clic en el primer elemento del JList

// Comprobar si es el directorio inicial

**try** {

*cliente*.changeToParentDirectory();

*direcSelec*=*cliente*.printWorkingDirectory();

FTPFile[] ff2=**null**;

// directorio de trabajo actual = directorio padre

*cliente*.changeWorkingDirectory(*direcSelec*);

// Obtener ficheros y directorios

ff2 = *cliente*.listFiles();

campo.setText("");

// Llenar la lista con fichero del directorio padre

*llenarLista*(ff2, *direcSelec*);

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

// No se hace click en el primer elemento del JLIST

// Puede ser un fichero o un directorio

**else** {

**if** (fic.substring(0,6).equals("(DIR) ")) {

// SE TRATA DE UN DIRECTORIO

**try** {

fic=fic.substring(6);

String direcSelec2="";

**if** (*direcSelec*.equals("/"))

direcSelec2=*direcSelec*+fic;

**else**

direcSelec2=*direcSelec* + "/" + fic;

FTPFile[] ff2=**null**;

*cliente*.changeWorkingDirectory(direcSelec2);

ff2 = *cliente*.listFiles();

campo.setText("DIRECTORIO: "+fic+", "+ ff2.length +" elementos");

// directorio actual

*direcSelec* = direcSelec2;

// Llenar la lista con los datos del directorio

*llenarLista* (ff2, *direcSelec*);

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**else** {

// SE TRATA DE UN FICHERO

*ficheroSelec* = *direcSelec*;

**if** (*direcSelec*.equals("/"))

*ficheroSelec* += fic;

**else**

*ficheroSelec* += "/"+fic;

campo.setText("FICHERO seleccionado:" + *ficheroSelec* );

*ficheroSelec* = fic; // Me quedo con el nombre

}// fin else fichero

}//fin else fichero directorio

campo2.setText("DIRECTORIO ACTUAL:" + *direcSelec*);

} // fin if inicial

}

}); //////////// fin click lista

// Implementar el botón Salir

botonSalir.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

**try** {

*cliente*.disconnect();

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

System.*exit*(0);

}} );

// Implementar el botón Subir Fichero

botonCargar.addActionListener(**new** ActionListener () {

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

JFileChooser f;

File file;

f = **new** JFileChooser();

//////// Solo se pueden seleccioner ficheros

f.setFileSelectionMode(JFileChooser.***FILES\_ONLY***);

// Titulo de la ventana

f.setDialogTitle("Selecciona el fichero a SUBIR al serbidor");

// Mostrar la ventana

**int** returnVal = f.showDialog(f, "Cargar");

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

// Fichero seleccionado

file = f.getSelectedFile();

// Nombre completo del fichero

String archivo = file.getAbsolutePath();

// Solo nombre del fichero

String nombreArchivo = file.getName();

**try** {

SubirFichero(archivo, nombreArchivo);

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

}

});

// Implementar el botón Descargar Fichero

botonDescargar.addActionListener(**new** ActionListener () {

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

String directorio=*direcSelec*;

**if** (!*direcSelec*.equals("/"))

directorio = directorio + "/";

**if** (!*ficheroSelec*.equals("")) {

**try** {

DescargarFichero(directorio + *ficheroSelec*, *ficheroSelec*);

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

}

});

//Implementar el botón Eliminar Fichero

botonBorrar.addActionListener(**new** ActionListener () {

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

String directorio=*direcSelec*;

**if** (!*direcSelec*.equals("/"))

directorio = directorio + "/";

**if** (!*ficheroSelec*.equals(""))

BorrarFichero(directorio+*ficheroSelec*, *ficheroSelec*);

}

});

//Implementar el botón Crear Carpeta

botonCreaDir.addActionListener(**new** ActionListener () {

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

String nombreCarpeta = JOptionPane.*showInputDialog*(**null**, "Introduce el numero del directorio","carpeta");

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

String directorio=*direcSelec*;

**if** (!*direcSelec*.equals("/")) directorio = directorio + "/";

// Nombre del directorio a crear

directorio +=nombreCarpeta.trim();

**try** {

**if** (*cliente*.makeDirectory(directorio)) {

String m=nombreCarpeta.trim() + " => Se ha creado korrektamente...";

JOptionPane.*showConfirmDialog*(**null**, m);

campo.setText(m);

// Directorio de trabajo actual

*cliente*.changeWorkingDirectory(*direcSelec*);

FTPFile[] ff2 = **null**;

// Obtener ficheros del directorio actual

ff2 = *cliente*.listFiles();

// Llenar la lista

*llenarLista*(ff2, *direcSelec*);

}

**else**

JOptionPane.*showConfirmDialog*(**null**, nombreCarpeta.trim()+" No se ha podido crear");

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

}

});

// Implementar el botón Eliminar Carpeta

botonDelDir.addActionListener(**new** ActionListener () {

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

String nombreCarpeta = JOptionPane.*showInputDialog*(**null**,"Introduce el nombre del directorio a eliminar","carpeta");

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

String directorio=*direcSelec*;

**if** (!*direcSelec*.equals("/"))

directorio=directorio+"/";

// Nombre del directorio a eliminar

directorio+=nombreCarpeta.trim();

**try** {

**if** (*cliente*.removeDirectory(directorio)) {

String m=nombreCarpeta.trim()+" => Se ha eliminado correctamente....";

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

campo.setText(m);

*cliente*.changeWorkingDirectory(*direcSelec*);

FTPFile[] ff2 = **null**;

// Obtener ficheros del directorio actual

ff2 = *cliente*.listFiles();

// Llenar la lista

*llenarLista*(ff2, *direcSelec*);

}

**else**

JOptionPane.*showConfirmDialog*(**null**, nombreCarpeta.trim()+" No se ha podido crear");

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

}

});

}

// Llenar la lista

**private** **static** **void** llenarLista(FTPFile[] files, String direc2) {

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

// Crear un objeto DefaultListModel

DefaultListModel modeloLista = **new** DefaultListModel();

modeloLista=**new** DefaultListModel();

// Definir propiedades para la lista, color y tipo de fuente

*listaDirec*.setForeground(Color.***blue***);

Font fuente=**new** Font("Courier",Font.***PLAIN***, 12);

*listaDirec*.setFont(fuente);

// Eliminar los elementos de la lista

*listaDirec*.removeAll();

**try** {

*cliente*.changeWorkingDirectory(direc2);

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

// Añadir el directorio de trabajo al listmodel, primer elemento

modeloLista.addElement(direc2);

// Recoger el array con los ficheros y directorios

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

**if** (!(files[i].getName()).equals(".") && !(files[i].getName()).equals(".. ")) {

// Saltar los directorios . y ..

// Obtener el nombre del fichero o directorio

String f = files[i].getName();

// Si es directorio añadir al nombre DIR

**if** (files[i].isDirectory()) f = "(DIR) " +f;

// Añadir el nombre del fichero o directorio al listmodel

modeloLista.addElement(f);

} // fin if

} // fin for

*listaDirec*.setModel(modeloLista);

}

////////// Subir Fichero

**private** **boolean** SubirFichero (String archivo, String soloNombre) **throws** IOException {

*cliente*.setFileType(FTP.***BINARY\_FILE\_TYPE***);

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

**boolean** ok=**false**;

// Directorio de trabajo actual

*cliente*.changeWorkingDirectory(*direcSelec*);

**if** (*cliente*.storeFile(soloNombre, in)) {

String s = " " + soloNombre + " =>Subido correctamente";

campo.setText(s);

campo2.setText("Se va a actualizar el árbol de directorios....");

FTPFile[] ff2=**null**;

// Obtener ficheros del directorio actual

ff2=*cliente*.listFiles();

// Llenar la lista con los ficheros del directorio actual

*llenarLista*(ff2,*direcSelec*);

ok=**true**;

}

**else** {

campo.setText("No se ha podido subir ...." + soloNombre);

}

**return** ok;

}

////////// Descargar Fichero

**private** **void** DescargarFichero (String NombreCompleto, String nombreFichero) **throws** IOException {

File file;

String archivoyCarpetaDestino = "";

String carpetaDestino = "";

JFileChooser f = **new** JFileChooser();

// Solo se pueden seleccionar directorios

f.setFileSelectionMode(JFileChooser.***DIRECTORIES\_ONLY***);

// Titulo de la ventana

f.setDialogTitle("Selecciona el directorio donde descargar el fichero");

**int** returnVal = f.showDialog(**null**, "Descargar");

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

// Obtener carpeta de destino OJO!!!!!!

// Esto da problemas por lo menos en MAC

// Con ficheros traducidos no funciona bien

// Coge el destino: /Users/aitor/Desktop/Escriotorio

// Se puede poner el path absoluto para que no de problemas

//

//file = f.getSelectedFile();

file = f.getCurrentDirectory();

carpetaDestino = file.getAbsolutePath().toString();

System.***out***.println("Nombre completo"+carpetaDestino);

// Construir el nombre completo que se creará en nuestro disco

archivoyCarpetaDestino = carpetaDestino + File.***separator*** + nombreFichero;

*cliente*.setFileType(FTP.***BINARY\_FILE\_TYPE***);

BufferedOutputStream out= **new** BufferedOutputStream (**new** FileOutputStream(archivoyCarpetaDestino));

**if** (*cliente*.retrieveFile(NombreCompleto, out))

JOptionPane.*showMessageDialog*(**null**, nombreFichero + " => Se ha descaergado correctamente");

**else**

JOptionPane.*showMessageDialog*(**null**, nombreFichero + " => No se ha podido descargar");

out.close();

}

}

////////// Borrar Fichero

**private** **void** BorrarFichero(String NombreCompleto, String nombreFichero) {

// Pedir confirmación del fichero a borrar

**int** seleccion = JOptionPane.*showConfirmDialog*(**null**,"¿Desea borrar el fichero seleccionado?");

**if** (seleccion == JOptionPane.***OK\_OPTION***) {

// Directorio de trabajo actual

**try** {

**if** (*cliente*.deleteFile(NombreCompleto)){

String m=nombreFichero+ "=> Eliminado correctamente";

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

campo.setText(m);

*cliente*.changeWorkingDirectory(*direcSelec*);

FTPFile[] ff2 = **null**;

// Obtener ficheros del directorio actual

ff2=*cliente*.listFiles();

// Llenar la lista

*llenarLista*(ff2,*direcSelec*);

}

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

}

**Solución**

**ACCIONES** – Realiza las siguientes acciones:

1. Averigua el nombre del método y la clase que se han usado en el programa para mostrar en consola los mensajes que se van originando en la comunicación con el servidor
2. Indica dónde y qué has modificado en el programa para que al arrancarlo, los diferentes botones de la parte derecha se vean en su totalidad

1. Indica a qué mensajes de la aplicación les has tenido que corregir faltas de ortografía y cuál ha sido el cambio que has hecho en cada corrección

1. Indica a qué botones de la aplicación les has tenido que corregir faltas de ortografía y cuál ha sido el cambio que has hecho en cada corrección

1. Indica el nombre de la variable utilizada en el programa para controlar el nombre del fichero seleccionado por el usuario

**EVIDENCIAS –** Incluye a continuación evidencias de lo siguiente:

1. Cómo se ha configurado el servidor proFTPd para que cuando se entre al mismo con el usuario dinux, éste se conecte por defecto a la carpeta /home/ACTIVIDAD4APELLIDO, donde APELLIDO es el primer apellido en mayúsculas de la persona que ha realizado la actividad.
2. Imagen de la aplicación justo después de haber creado dentro de la carpeta ACTIVIDAD4APELLIDO del servidor, unas carpetas de nombre DM2, PROS y CIFP
3. Imagen de la aplicación justo después de haber subido a la carpeta CIFP del servidor unos archivos de nombre Java.jpg y Eclipse.jpg, en este mismo orden
4. Imagen de la aplicación justo antes de confirmar que se va a eliminar el fichero de nombre Eclipse.jpg de la carpeta CIFP
5. Imagen de la aplicación en la que justo se ha introducido el nombre del directorio/carpeta que se quiere eliminar (la carpeta CIFP)

**REQUISITOS**

1. Entregar cada actividad en un archivo comprimido que contenga los ficheros \*.java empleados más este mismo documento con el apartado Solución (ACCIONES y EVIDENCIAS) rellenado.
2. El nombre del documento que contenga el apartado Solución rellenado será el mismo que el del enunciado, añadiendo al final del mismo el primer apellido de la persona que realiza la actividad. Por ejemplo: 4\_ACTIVIDADES 4\_V2(creación cliente FTP)\_Perez
3. Los archivos comprimidos seguirán la siguiente nomenclatura:

*UD4\_Actividadn\_apellido*

donde:

*n*: valdrá 4, representando el número de la actividad

*apellido*: será el primer apellido del alumno

**CRITERIOS DE CORRECCIÓN**

1. Cumplimiento de los requisitos (20%).
2. Evidencias de la solución (40%).
3. Acciones de la solución (40%).