

4.1.- Servicios en red y protocolos estándares de nivel de aplicación

En el tema anterior se ha estudiado aplicaciones servidor-cliente que implementan un protocolo de nivel de aplicación sobre un protocolo de nivel de transporte, actuando uno de los programas como servidor de ese protocolo y el otro como cliente.

Existen muchos protocolos estándares de nivel de aplicación, basados en los protocolos de transporte UDP y TCP, que proporcionan servicios diversos a las aplicaciones. La mayoría de los protocolos más utilizados de nivel de aplicación funcionan sobre TCP. Los que funcionan sobre UDP normalmente solo realizan transferencias de pequeñas cantidades de datos, en las que prima la rapidez sobre la fiabilidad. Un ejemplo de esto es el protocolo DNS , en el que la aplicación de cliente envía un nombre de dominio y recibe una dirección IP asociada. Los protocolos que transfieren una mayor cantidad de datos en secuencia, como , por ejemplo , ficheros o páginas web ,utilizan TCP como protocolo de transporte.

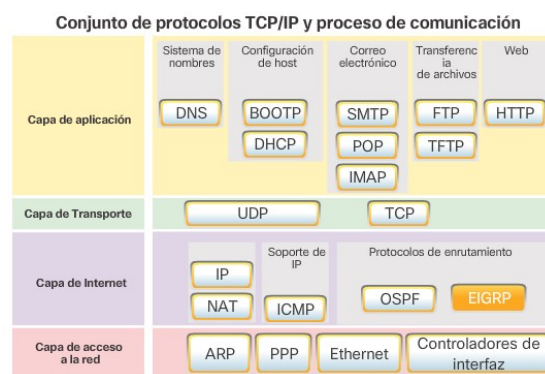
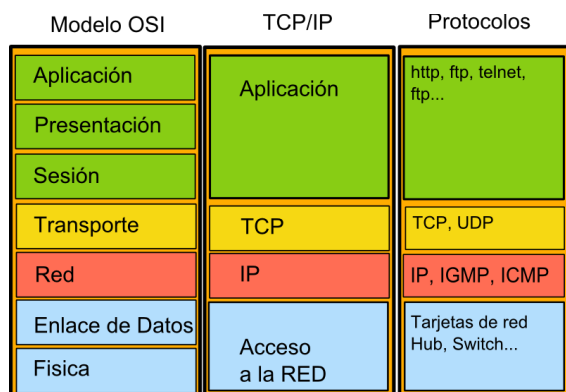
Sobre el protocolo de transporte TCP se puede utilizar el protocolo estándar TLS/SSL , que permite que los datos transmitidos por TCP vayan encriptados. Se han creado variantes de los protocolos estándares que utilizan TLS/SSL sobre TCP. Utilizan puertos distintos que la variante que no utilizan TLS/SSL.

Por ejemplo, el protocolo estándar HTTP funciona sobre TCP y utiliza el puerto 80. HTTPS es una variante de HTTP que utiliza TLS/SSL sobre TCP , y utiliza el puerto 443.

4.2.- Protocolos estándar de comunicación en red

El modelo TCP/IP esta compuesto por cuatro capas o niveles . La capa de aplicación define las aplicaciones de red y los servicios de Internet estándar que puede utilizar un usuario. Estos servicios utilizan una capa de transporte para enviar y recibir datos. Existen varios protocolos de capa de aplicación

- Conexión remota : Telnet
- Correo electrónico: SMTP
- Acceso a ficheros remotos: FTP, NFS , TFTP
- Resolución de nombres de ordenadores: DNS , WINS
- World Wide Web: HTTP.



Capas TCP/IP	Capas y protocolos TCP/IP	
Aplicación	SMTP, Telnet, FTP, HTTP	NFS, SNMP, DNS
Transporte	TCP	UDP
Internet	IP	
Acceso a la red	ARP, RARP	
Física	Sin especificar	

Nota: Todas las aplicaciones que implementan TCP/ IP se basan en el modelo cliente-servidor.

TELNET (Telecommunication Network): Emulación de terminal; permite a un usuario acceder a una maquina remota y manejarla como si estuviese en ella. Es el sistema empleado para solucionar fallos de maquinas remotas o para realizar consultas a distancia. Su principal problema es la seguridad ya que los nombres de usuario y contraseñas viajan por la red como texto plano.

SMTP (Simple Mail Transfer Protocol): Protocolo simple de transferencia de correo electrónico; Este estándar especifica el formato exacto de los mensajes que un cliente en una máquina debe enviar al servidor en otra.

FTP: (File Transfer Protocol): Protocolo de transferencia de ficheros; es un servicio confiable orientado a la conexión que se utiliza para transferir ficheros de una máquina a otra a través de internet.

TFTP(Trivial File Transfer Protocol): Protocolo trivial de transferencia de ficheros; es un protocolo de transferencia muy simple semejante a una versión básica de FTP. Fue definido para aplicaciones que no necesitan tanta interacción entre cliente y servidor.

NFS (Network File System): Sistema de ficheros de red que permite a los usuarios el acceso en línea a ficheros que se encuentran en sistemas remotos, de esta forma el usuario accede a un fichero como si este fuera un fichero local.

SNMP (Simple Network Management Protocol): Protocolo simple de administración de red, es un protocolo utilizado para intercambiar información de gestión entre los dispositivos de una red. Permite a los administradores de red monitoriarla, controlarla y supervisar el funcionamiento.

DNS (Domain Name System) Sistema de nombres de dominio, es un sistema que usa servidores distribuidos a lo largo de la red para resolver el nombre de un host IP en una dirección IP.

4.3.- Comunicación con un servidor FTP

Es una herramienta para el intercambio de ficheros entre distintos ordenadores y la forma habitual de publicar en internet.

Para transferir ficheros a través de FTP, un ordenador debe ser el cliente FTP y el otro el servidor FTP. El cliente envía comandos al servidor (subir, bajar o borrar ficheros, crear directorio) y el servidor lo lleva a cabo.

Hay dos tipos de acceso a través de FTP

1. **Acceso anónimo:** cuando la conexión con la máquina servidora la realiza un usuario sin autentificar y sin ningún tipo de privilegio en el servidor. En este caso al usuario es recluido a un directorio público donde solo se le permite descargar ficheros.
2. **Acceso autorizado:** el usuario que realiza la conexión con la máquina servidora está registrado y tiene ciertos privilegios en el servidor. En este caso, y una vez autenticado, el usuario es dirigido a su directorio personal donde puede subir y bajar ficheros.

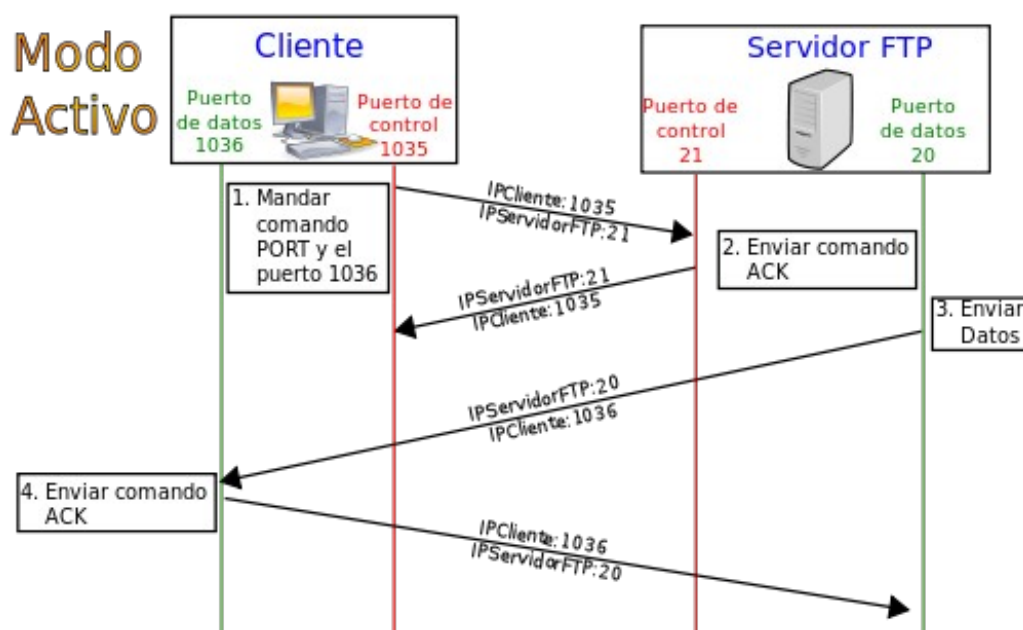
FTP usa dos conexiones distintas TCP, una conexión de control y otra de transferencia de datos.

La primera se encarga de iniciar y mantener la comunicación entre el cliente y el servidor, la segunda se encarga de enviar datos entre cliente y servidor, esta existe solo cuando hay datos a transmitir.

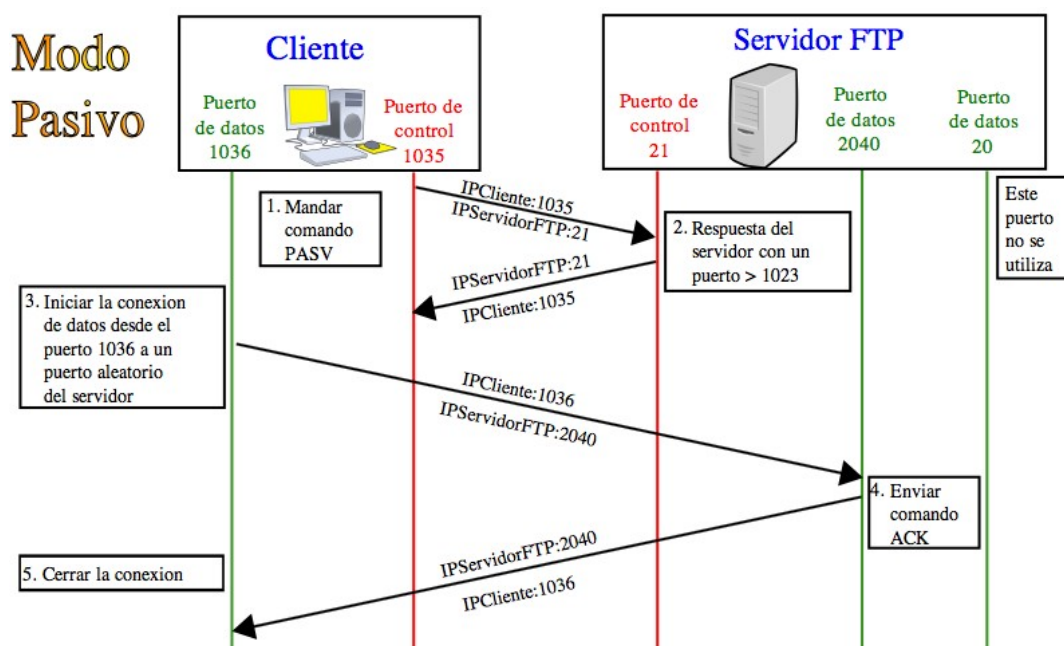
Cuando un cliente se conecta a un servidor FTP, el cliente emplea un puerto aleatorio, sin embargo, el servidor usa el **puerto 21**. Este puerto (control port) es el puerto de comandos por el que se transfieren las ordenes. Para la transferencia de datos no se utilizan los mismos puertos, el cliente obtiene un nuevo puerto y el servidor usa normalmente el puerto 20

En este proceso de comunicación cliente-servidor, el cliente puede actuar de dos modos diferentes:

- a) **Modo activo o modo estándar.** En este modo el servidor siempre crea el canal de datos en su puerto 20, mientras que en el lado del cliente el canal de datos se asocia a un puerto aleatorio mayor que el 1024.



- 1) El cliente envía un comando PORT al puerto 21 del servidor (puerto de comandos) y le indica el puerto para el canal de datos (en este caso el 1036).
 - 2) Seguidamente el servidor le envía al cliente un ACK (recibido) al puerto de comandos del cliente (1035)
 - 3) El servidor inicia la conexión del puerto local de datos (el 20) con el puerto de datos indicado anteriormente (1036)
 - 4) El cliente envía un ACK al servidor.
- b) **Modo pasivo o PASV**, en este caso el cliente envía comandos tipo PASV , por lo que el servidor FTP le indicará por el canal de control el puerto al que debe conectarse el cliente.



- 1) En el primer paso, el cliente manda un comando PASV al puerto de comandos del servidor (el 21).
- 2) En el paso 2, el servidor responde indicando al cliente el puerto por el que escuchará la conexión de datos, en este caso el puerto 2040.
- 3) En el paso 3 , el cliente inicia la conexión de datos desde el puerto de datos del cliente, al puerto de datos del servidor (el 2040)
- 4) Por último, el servidor envía un ACK al puerto de datos del cliente.

NOTA: El modo activo tiene un problema de seguridad y es que se abre una conexión para datos a la maquina cliente desde fuera para dentro, y se el cliente está conectada a una red insegura como Internet puede ser atacado fácilmente. Normalmente, el uso de cortafuegos instalados en el equipo rechazará dichas conexiones; por este motivo, se desarrollo el modo pasivo.

4.3.1.- JAVA para comunicar con un servidor FTP

Existen librerías Java que nos permiten crear programas cliente para comunicar con un servidor FTP. “Apache Commons Net” proporciona una librería de componentes que nos permite implementar el lado cliente de muchos protocolos básicos de Internet. La librería incluye soporte para protocolos como FTP, SMTP, Telnet , FFTP, etc.

¿Como podemos acceder desde un programa cliente Java , a un servidor FTP?

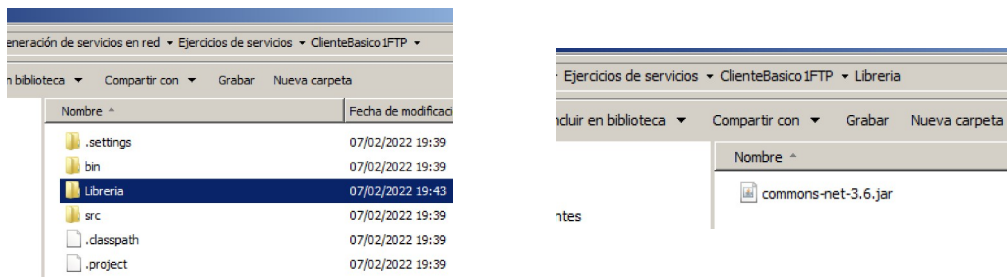
Será necesaria la librería **commons-net-x.y.jar** que la podemos descargar desde:

https://commons.apache.org/proper/commons-net/download_net.cgi

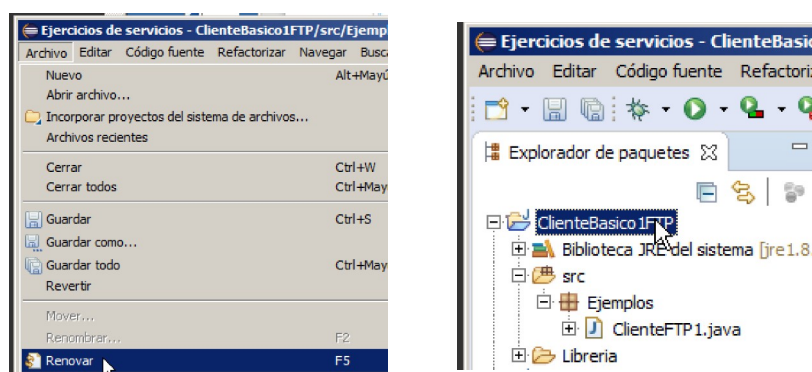
La clase **FTPClient** encapsula toda la funcionalidad necesaria para almacenar y recuperar ficheros de un servidor FTP. Esta clase se encarga de todos los detalles de bajo nivel de la interacción con un servidor FTP. Para utilizarla primero debemos de realizar la conexión al servidor con el método “connect()” , comprobar la respuesta para ver si ha sucedido algún error, realizar las operaciones de transferencia y cuando finalice el proceso, cerrar la conexión usando el método **disconnect()** .

4.3.1.1.-Añadir la librería a nuestro proyecto en Eclipse

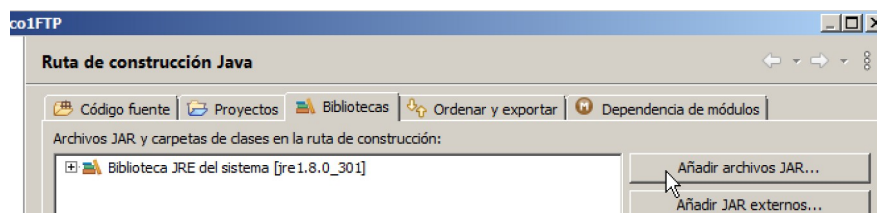
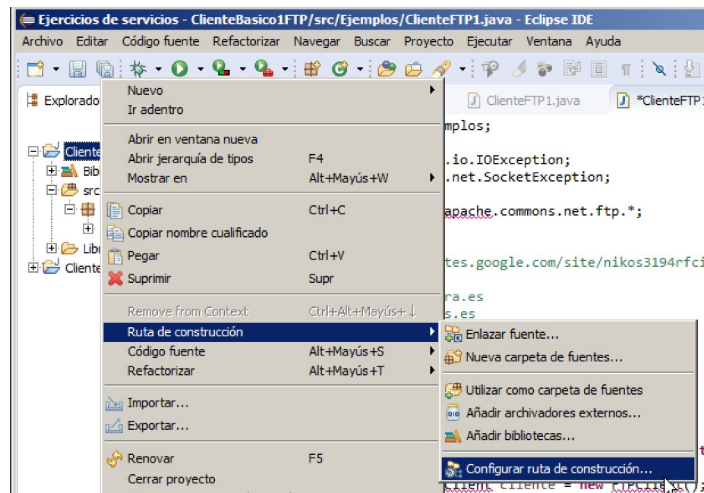
Nos vamos a la carpeta donde está el proyecto y creamos un directorio llamado “Libreria”. Guardamos nuestro fichero .jar de la librería en dicha carpeta



En eclipse refrescamos el proyecto para que aparezca la carpeta librerías

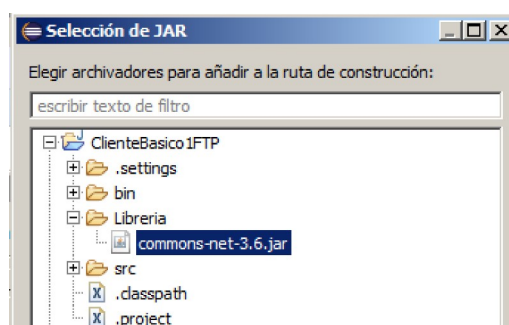


Nos vamos al nombre del proyecto, clic derecho → Ruta de construcción → Configurar ruta de construcción.

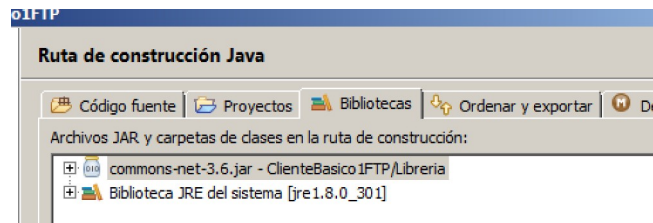


En la pestaña biblioteca, hacemos clic en “Añadir archivos JAR”

Buscamos dentro de la carpeta librería que creamos , el fichero de librería a añadir.

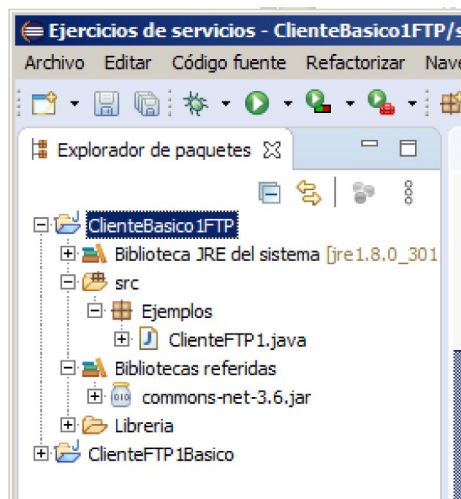


Nos tiene que aparecer como que se ha añadido



Hacemos clic en aplicar y cerrar.

En eclipse nos tiene que aparecer nuestra librerías



4.3.1.2.- Ejemplo de conexión a un servidor FTP (ftp.rediris.es). Se comprueba si se ha realizado correctamente la conexión y se cierra.

```
package Ejemplos;
```

```
import java.io.IOException;
```

```
import java.net.SocketException;
```

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

```
/*  
https://sites.google.com/site/nikos3194rfcindex/home/old-school/ftp  
ftp.uv.es  
ftp.unavarra.es  
ftp.rediris.es  
ftp.uma.es  
ftp.udc.es  
ftp.dit.upm.es
```

```
ftp.freenet.de

*/
public class ClienteFTP1 {
    public static void main(String[] args) throws SocketException, IOException {

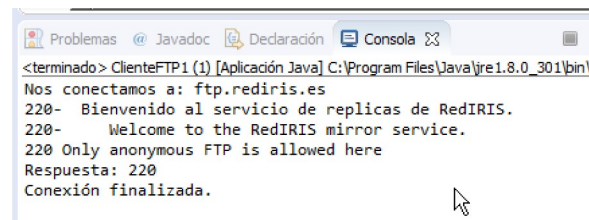
        FTPClient cliente = new FTPClient();
        String servFTP = "ftp.rediris.es"; // servidor FTP
        System.out.println("Nos conectamos a: " + servFTP);
        cliente.connect(servFTP);

        // respuesta del servidor FTP
        System.out.print(cliente.getReplyString());
        // código de respuesta
        int respuesta = cliente.getReplyCode();

        System.out.println("Respuesta: "+respuesta);

        // comprobación del código de respuesta
        if (!FTPReply.isPositiveCompletion(respuesta)) {
            cliente.disconnect();
            System.out.println("Conexión rechazada: " + respuesta);
            System.exit(0);
        }
        // desconexión del servidor FTP
        cliente.disconnect();
        System.out.println("Conexión finalizada.");
    }
}
```

SALIDA



```
<terminado> ClienteFTP1 (1) [Aplicación Java] C:\Program Files\Java\jre1.8.0_301\bin\
Nos conectamos a: ftp.rediris.es
220- Bienvenido al servicio de replicas de RedIRIS.
220- Welcome to the RedIRIS mirror service.
220 Only anonymous FTP is allowed here
Respuesta: 220
Conexión finalizada.
```

La clase **FTPReply** almacena un conjunto de constantes para códigos de respuesta FTP. Para interpretar el significado de los códigos se puede consultar (<https://www.ietf.org/rfc/rfc959.txt>)

El método **isPositiveCompletion(int respuesta)** devuelve true si un código de respuesta ha terminado positivamente. El código 220 significa que el servicio está preparado.

El protocolo FTP usa un esquema de código de respuesta donde cada uno de sus dígitos tiene un significado especial. Son número de tres dígitos en ASCII, el primer dígito indica si la respuesta es buena , mala o incompleta:

- **1yz** : Respuesta preliminar positiva, el servidor inició la acción solicitada
- **2yz**: Respuesta de terminación positiva, el servidor terminó con éxito la acción solicitada.
-

Algunos métodos de la clase FTP.

MÉTODOS	FUNCIÓN
void connect (String host)	Abre la conexión con el servidor FTP indicado en host
int getReplyCode()	Devuelve el valor entero del código de respuesta de la última respuesta FTP
String getReplyString()	Devuelve el texto completo de la respuesta del servidor FTP

Métodos de la clase FTPClient (derivada de FTP), estos métodos se usarán para logearnos al servidor FTP , subir, bajar y eliminar ficheros , movernos de un directorio a otro, etc. Muchos de estos métodos devuelven un valor booleano, true si el método tuvo éxito y false en caso contrario.

MÉTODOS	FUNCIÓN
void disconnect ()	Cierra la conexión con el servidor FTP y restaura los parámetros de conexión a los valores predeterminados.
boolean login (String user , String passwd)	Inicia sesión en el servidor FTP usando el nombre de usuario y la contraseña proporcionados. Devuelve true si se inicia la sesión con éxito, si no devuelve false
boolean logout()	Sale del servidor FTP
void enterLocalActiveMode()	Se establece el modo de conexión de datos actual en modo activo.
void enterLocalPassiveMode()	Se establece el modo de conexión de datos actual en modo pasivo
String printWorkingDirectory ()	Devuelve el nombre de ruta del directorio de trabajo actual
FTPFile [] listFiles()	Obtiene una lista de ficheros del directorio actual como un array de objetos FTPFile
FTPFile [] listFiles (String path)	Obtiene una lista de ficheros del directorio indicado en path
String [] listNames ()	Obtiene una lista de ficheros del directorio actual como un array de cadenas
FTPFile [] listDirectories ()	Obtiene la lista de los directorios que se encuentran en el directorio de trabajo actual
FTPFile [] listDirectories (String parent)	Obtiene la lista de los directorios que se encuentran en el directorio de especificado en parent
boolean change WorkingDirectory (String pathname)	Cambia el directorio de trabajo actual de la sesión FTP al indicado en pathname
boolean changeToParentDirectory ()	Cambia el directorio padre del directorio de trabajo actual
Boolean setFileType (int fileType)	Establece el tipo de fichero a transferir : ASCII_FILE_TYPE (fichero ASCII), BINARY_FILE_TYPE (imagen binaria), etc.
boolean storeFile (String nombre , InputStream local)	Almacena un fichero en el servidor con el nombre indicado tomando como entrada el InputStream, si el fichero existe lo sobrescribe.
boolean retrieveFile (String nombre , Output Stream local)	Recupera un fichero del servidor y lo escribe en el OutputStream dado.

boolean deleteFile (String pathname)	Elimina un fichero en el servidor FTP
boolean rename (String antiguo , String nuevo)	Cambia el nombre de un fichero del servidor FTP
Boolean removeDirectory (String pathname)	Elimina un directorio en el servidor FTP (si está vacío)
Boolean makeDirectory (String pathname)	Crea un nuevo subdirectorio en el servidor FTP en el directorio actual.

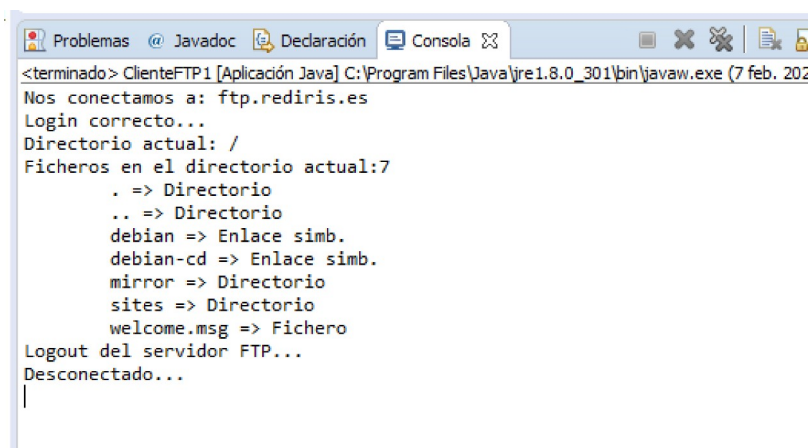
NOTA: Todos los métodos de comunicación con el servidor pueden lanzar “IOException”. El servidor FTP puede optar por cerrar antes de tiempo una conexión si el cliente ha estado inactivo durante más de un tiempo determinado (900 segundos). La clase FTPClient detectará un cierre prematuro de la conexión con el servidor FTP y se puede producir la excepción “FTPConnectionClosedException”

Lo normal es conectar a un servidor FTP con un nombre de usuario y su clave. Para identificarnos usaremos el método login() que devuelve true si la conexión se realiza correctamente. Nos conectamos al servidor en modo pasivo usando el método “enterLocalPassiveMode () “. Para desconectarnos usaremos “logout () “

EJERCICIO 1

Realizar un cliente FTP para conectarnos al servidor FTP “ftp.rediris.es” utilizando un acceso anónimo. Se debe de conectar como usuario = anonymous y clave = anonymous. El cliente debe de mostrar la lista de ficheros del directorio actual. Usar el método “listFiles ()” que devuelve un array de la clase FTPFile con información de los ficheros y directorios encontrados; se recorre el array visualizando el nombre del fichero o directorio y el tipo que puede ser fichero, directorio o enlace simbólico.

La salida que debe mostrar debe se como la siguiente imagen:



```
<terminado> ClienteFTP1 [Aplicación Java] C:\Program Files\Java\jre1.8.0_301\bin\javaw.exe (7 feb. 202
Nos conectamos a: ftp.rediris.es
Login correcto...
Directorio actual: /
Ficheros en el directorio actual:7
. => Directorio
.. => Directorio
debian => Enlace simb.
debian-cd => Enlace simb.
mirror => Directorio
sites => Directorio
welcome.msg => Fichero
Logout del servidor FTP...
Desconectado...
|
```

SOLUCIÓN

package Ejemplos;

```
import java.io.*;
import org.apache.commons.net.ftp.*;
```

```
public class ClienteFTPAnonymous {
    public static void main(String[] args) {
        FTPClient cliente = new FTPClient();
        String servFTP = "ftp.rediris.es";
        System.out.println("Nos conectamos a: " + servFTP);
        String usuario = "anonymous";
        String clave = "anonymous";
        try {
            cliente.connect(servFTP);
            cliente.enterLocalPassiveMode(); //modo pasivo

            boolean login = cliente.login(usuario, clave);
            if (login)
                System.out.println("Login correcto...");
            else {
                System.out.println("Login Incorrecto...");
                cliente.disconnect();
                System.exit(1);
            }
            System.out.println("Directorio actual: "
                               + cliente.printWorkingDirectory());

            FTPFile[] files = cliente.listFiles();
            System.out.println("Ficheros en el directorio actual:"
                               + files.length);
            //array para visualizar el tipo de fichero
            String tipos[] = {"Fichero", "Directorio", "Enlace simb."};

            for (int i = 0; i < files.length; i++) {
                System.out.println("\t" + files[i].getName() + " => "
                                     + tipos[files[i].getType()]);
            }
            boolean logout = cliente.logout();
            if (logout)
                System.out.println("Logout del servidor FTP...");
            else
                System.out.println("Error al hacer Logout...");
            //
            cliente.disconnect();
            System.out.println("Desconectado...");
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }
} // ..
```

NOTA: Para movernos de un directorio a otro usamos el método “changeWorkingDirectory () “. Devuelve true si el directorio existe y false si no existe. Si quiero mostrar el contenido de un directorio, primero nos tenemos que situar en el directorio usando este método.

Ejemplo: Si quiero acceder al directorio “mirror/MySQL/Downloads “ para obtener la lista de ficheros que hay escribiremos lo siguiente:

```
String directorio= “/mirror/MySQL/Downloads/ “;
if (cliente.changeWorkingDirectory (directorio) )
    System.out.println(“Dir Actual: “ + cliente.printWorkingDirectory ( ) );

else
    System.out.println(“NO EXISTE EL DIRECTORIO: “ + directorio);

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

Ejemplo:

Crear un directorio en el directorio actual (tenemos que tener permiso para poder hacerlo) y hacemos que sea el directorio de trabajo actual usando el método “changeWorkingDirectory()”

```
String direc = “NUEVODIREC”;
if (cliente.makeDirectory (direc) ){
    System.out.println(“Directorio creado ....”);
    cliente.changeWorkingDirectory (direc);
}
else
    System.out.println(“NO SE HA PODIDO CREAR EL DIRECTORIO”);
```

La clase FTPFile se utiliza para representar información acerca de los ficheros almacenados en un servidor FTP. Algunos métodos importantes son:

MÉTODOS	FUNCIÓN
String getName()	Devuelve el nombre del fichero.
long getSize()	Devuelve el tamaño del fichero en bytes
int getType ()	Devuelve el tipo del fichero: 0 si es un fichero (FILE_TYPE). 1 si es un directorio (DIRECTORY TIPE) 2 si es un enlace simbólico (SYMBOLIC_LINK_TYPE)
String getUser()	Devuelve el nombre del usuario propietario del fichero
boolean isDirectory()	Devuelve true si el fichero es un directorio
boolean isFile()	Devuelve true si es un fichero
Boolean isSymbolicLink()	Devuelve true si es un enlace simbólico.

EJERCICIO 2 (Realizar este ejercicio)

Conectate a ftp.rediris.es y visualiza los directorios del directorio raíz, entra en cada directorio del directorio raíz mostrando la lista de ficheros y directorio que hay. Prueba en otros servidores FTP que admiten usuarios anónimos como:

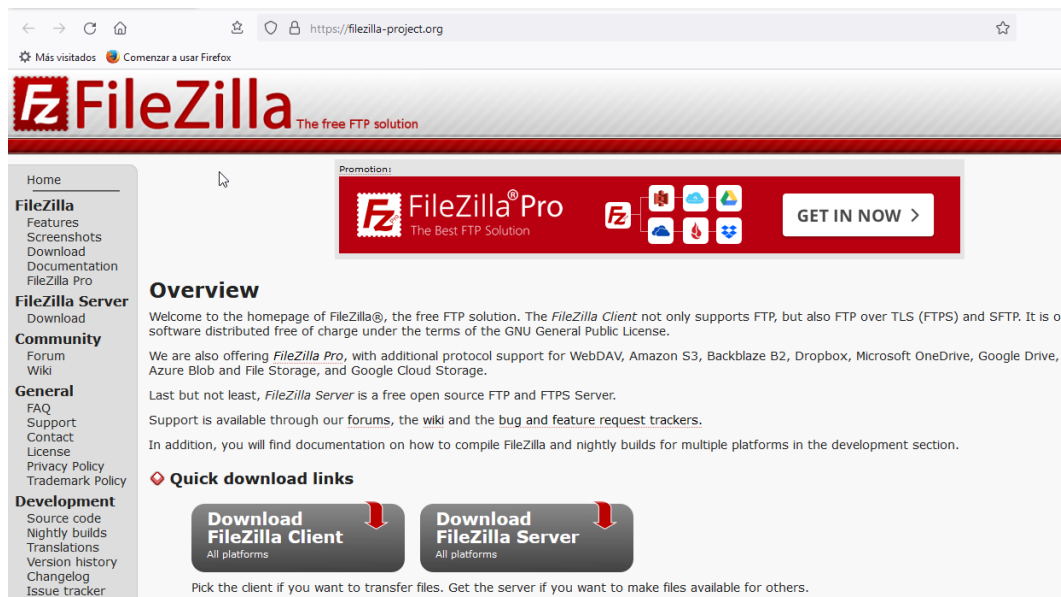
- ftp.uv.es
- ftp.freebsd.org
- ftp.dit.upm.es

EJERCICIO 2.1 (OPCIONAL , la instalación de Filezilla nos sirve para los siguientes ejercicios)

Instala el servidor FTP Filezilla Server en tu maquina local. Crea un directorio en el disco duro de tu equipo para que sirva como servidor FTP. Crea un subdirectorio para cada usuario que crees y copia algunos ficheros en este directorio. Crea varios usuarios y asigna los permisos al directorio al que pueden acceder.

Consulta el documento INSTALACIÓN_FilezillaServer.pdf que hay en la plataforma para poder instalar y configurar el servidor FTP .

Puedes descargar Filezilla Server desde la URL <https://filezilla-project.org/>



Realiza la actividad anterior (Ejercicio 2) conectándote al servidor FTP local. **El nombre del servidor es localhost** y el nombre de usuario y clave , el que hayas creado.

NOTA: Para presentar el ejercicio y ser evaluado positivamente, captura pantalla de los directorios creado en tu disco y pantallazo de las salidas obtenida cuando ejecute la aplicación creada.

4.3.2.- Subir Ficheros al Servidor

Para crea la practica a este apartado, necesitamos tener creada una cuenta en algún servidor FTP que ofrezca el servicio de forma gratuita o bien hacerlo a nivel local con Filezilla Server.

Probar a crear una cuenta gratuita en algunos de los siguientes servidores:

- <http://byethost.com/>
- <https://hostinger.es/>
- <https://www.000webhost.com/>

NOTA: Para subir ficheros al servidor es necesario:

- Un nombre de usuarios.
- La clave del usuario.
- Espacio en el servidor.
- Tener privilegio para subir fichero.

Los pasos a realizar son los siguientes:

1. Conectarnos al servidor.
2. Identificarnos con el usuario y contraseña.
3. Situarnos en el directorio donde vamos a subir el fichero.

Ejemplo:

Suponemos que deseamos subir un fichero en el directorio NUEVODIREC que cuelga del directorio raíz (/).

```
String direc = "/NUEVODIREC";  
cliente.changeWorkingDirectory (direc);
```

A continuación, con el método setFileType() indicamos el tipo de fichero a subir. Este tipo es una constante entera definida en la clase FTP .

Se suele poner BINARY_FILE_TYPE que permite enviar ficheros de cualquier tipo.

```
Cliente.setFileType ( FTP.BINARY_FILE_TYPE);
```

Seguidamente creamos un stream de entrada con los datos del fichero que vamos a subir (supongamos que deseamos subir el fichero EJEMPLO.pdf que lo tenemos en la carpeta [D:/PSP](#)) y se lo pasamos al método storeFile(), en el primer parámetro indicaremos el nombre que tendrá el fichero en el directorio FTP y en el segundo el InputStream, este método devuelve true si el proceso se ha realizado correctamente.

```
String archivo = "D:\\PSP\\EJEMPLO.pdf";  
BufferedInputStream entradabuffer = new (BufferedInputStream(new  
FileInputStream (archivo));
```



```
if (cliente.storeFile("EJEMPLO.pdf", entradabuffer))  
    System.out.println("Fichero subido correctamente...");  
else  
    System.out.println("El fichero no se ha podido subir ...");
```

Por último, será necesario cerrar el flujo de entrada..

EJERCICIO 3 (Tenemos que tener instalado Filezilla Server)

En este ejercicio se subirá un fichero al directorio NUEVODIREC, si el directorio no existe se crea. Al fichero se le da el mismo nombre en el servidor que el que tiene actualmente. El servidor es localhost y el usuario y clave es usuario1 (que previamente lo debimos de configurar en el servidor)

SOLUCIÓN

```
import java.io.*;  
import org.apache.commons.net.ftp.*;  
  
public class SubirFichero {  
    public static void main(String[] args) {  
        FTPClient cliente = new FTPClient();  
  
        String servidor = "localhost";  
        String user = "usuario1";  
        String pasw = "usuario1";  
  
        try {  
            System.out.println("Conectandose a " + servidor);  
            cliente.connect(servidor);  
            boolean login = cliente.login(user, pasw);  
  
            cliente.setFileType(FTP.BINARY_FILE_TYPE);  
            String direc = "/NUEVODIREC";  
            cliente.enterLocalPassiveMode();  
  
            if (login) {  
                System.out.println("Login correcto");  
  
                if (!cliente.changeWorkingDirectory(direc)) {  
                    String directorio = "NUEVODIREC";  
  
                    if (cliente.makeDirectory(directorio)) {  
                        System.out.println("Directorio : " + directorio + " creado ...");  
                        cliente.changeWorkingDirectory(directorio);  
                    }  
                    else {  
                        System.out.println("No se ha podido crear el Directorio");  
                        System.exit(0);  
                    }  
                }  
  
                System.out.println("Directorio actual: " + cliente.printWorkingDirectory());  
            }  
        }  
    }  
}
```

```
String archivo = "D:\\PSP\\EJEMPLO.pdf";
BufferedInputStream in = new BufferedInputStream (new FileInputStream(archivo));

if (cliente.storeFile("EJEMPLO.pdf", in))
    System.out.println("Subido correctamente... ");
else
    System.out.println("No se ha podido subir el fichero... ");

in.close(); // Cerrar flujo
cliente.logout();
cliente.disconnect();
}

} catch (IOException ioe) {
    ioe.printStackTrace();
}

} // main
} // cierra la clase
```

Ejemplo: Renombrar un fichero

Para renombrar un fichero se usa el método **rename(nombreaktigo , nombrenuevo)**.

Este método devuelve true si renombra el fichero con éxito , en caso contrario devuelve false

Supongamos que deseamos renombrar el fichero de nombre EJEMPLO.pdf que se encuentra en la carpeta /NUEVODIREC a EJEMPLO1.pdf

```
String direc = "/NUEVODIREC" ;
cliente.changeWorkingDirectory (direc);

if (cliente.rename ("EJEMPLO.pdt , EJEMPLO1.pdf"))
    System.out.println ("Fichero renombrado . . . ");
else
    System.out.println("No se ha podido renombrar el Fichero . . . ");
```

Ejemplo: Eliminar un fichero

Para eliminar un fichero usamos el método **deleteFile()**.

Este método devuelve true si elimina el fichero y false si no lo elimina.

Para eliminar el fichero EJEMPLO1.pdf , situado en la carpeta /NUEVODIREC se hace de la siguiente forma:

```
String fichero = "/NUEVODIREC/EJEMPLO1.pdf";

if (cliente.deleteFile(fichero) )
    System.out.println (" Fichero eliminado ..... ");
```

```
else (System.out.println ("No se ha podido eliminar el fichero . . . . " );
```

EJERCICIO 4 (Realizar este ejercicio)

Crea un programa Java que permita subir ficheros al servidor FTP local, al directorio raíz del usuario que se conecte. Utiliza la clase **JfileChooser** para seleccionar el fichero de tu disco a subir. Después de realizada la subida se debe mostrar un mensaje indicándolo. A continuación, muestra en consola el contenido del directorio raíz para ver si aparece el fichero subido.

4.3.3- Descargar ficheros del servidor-cliente

Para descargar ficheros del servidor a nuestro disco , usaremos el método:

```
retrieveFile(String remote , OutputStream local)
```

Es necesario saber el directorio desde el que descargaremos el fichero. Este método devuelve true si el proceso se realiza bien y falso en el caso de que se produzca un fallo. También hace falta crear un stream de salida para escribir el fichero en nuestro disco.

Ejemplo:

Supongamos que deseamos descargar el fichero de nombre **texto.txt**, que está en la carpeta del servidor FTP **/NUEVODIREC** / a la carpeta **D\ PSP** con el nombre **textoNuevo.txt**.

```
String direc = "/NUEVODIREC ";  
cliente.changeWorkingDirectory (direc) ;
```

// Creamos el stream de salida para recibir el fichero descargado

```
BufferedOutputStream out = new BufferedOutputStream ( new FileOutputStream ("D:\\  
PSP\\textoNuevo.txt "));  
  
if (cliente.retrieveFile ("texto.txt , out) )  
    System.out.println ("Recuperado el fichero correctamente ....");  
else  
    System.out.println(" No se ha podido descargar el fichero ...");  
  
out.close();
```

EJERCICIO 5: Descarga de fichero (realizar este ejercicio mediante ventanas)

Realizar un programa Java que se conecte al servidor FTP local (localhost) con un nombre de usuario y su clave. Se debe pedir por teclado el nombre de usuario y la clave.

Una vez conectado se mostrará por pantalla la lista de ficheros (solo lo ficheros, los directorios no se mostrarán) del servidor FTP que podemos descargar. Al hacer clic en el fichero se debe mostrar su nombre y si pulsamos el botón "Descargar" , nos debe permitir descargar el fichero en nuestro disco duro. Después de la descarga se mostrará un mensaje indicando si se ha realizado correctamente o no. El botón "Salir" Finaliza la aplicación .

4.3.4.- Creación de un cliente FTP

En este apartado se creará un cliente FTP para poder subir, descargar y eliminar ficheros; y crear y eliminar directorios o carpetas en nuestro sitio FTP . Para probarlo, se usará el servidor FTP Filezilla Server que previamente debemos de tener instalado en nuestra maquina.

Debemos de tener creado en el servidor un usuario de nombre “usuario1” y clave “usuario1”; que tiene permisos sobre la carpeta D:\MiServerFTP\usuario1 del servidor Filezilla Server. Esta carpeta debe de contener ficheros y directorios.

NOTA: Para mostrar en la consola el contenido de los mensajes comando/respuesta que se van originando en la comunicación con el servidor FTP podemos usar el método “addProtocolCommandListener() “ de la clase SocketClient.

Se escribe la siguiente expresión antes de realizar la conexión al servidor:

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

La interfaz ProtocolCommandListener junto con la interfaz PrintCommandListener facilita esta tarea.

Los datos que necesitamos para la conexión al servidor FTP son el nombre del servidor, el nombre del usuario y su clave. En el programa se han usado las siguientes variables para almacenar estos datos y se han asignado los siguientes valores:

- servidor=”localhost”
- user=”usuario1”
- pasw= “usuario1”

La pantalla tiene 5 campos de texto no editables . En la parte superior se muestra el nombre del servidor, el usuario , y el directorio raíz; en la parte inferior se muestra los mensajes que van surgiendo según vamos navegando por las carpetas.

A la derecha se muestra 6 botones que nos permiten , subir, descargar y eliminar ficheros; crear y eliminar carpeta y finalizar la aplicación. En el centro se muestra la lista de ficheros y carpetas del directorio actual. Estos se almacena primero en un Jlist y después se visualizan.

Para distinguir un fichero de un directorio se ha añadido a la izquierda del directorio la palabra (DIR) seguida 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 de forma que al hacer clic de nuevo en él, se volverá a visualizar nuevamente el contenido del directorio padre.

Se usa la variable **direcInicial** para definir el directorio inicial del usuario a partir del cual realizaremos la navegación .

La variable **direcSelec** para saber en todo momento cual es el directorio de trabajo actual.

La variable **ficheroSelec** para saber el último fichero seleccionado (sobre el que hemos hecho clic).

Se inicializa con los siguientes valores:

```
static String direcInicial = “ / “;  
static String direcSelec = direcInicial;  
static String ficheroSelec =””;
```

SOLUCIÓN

```
package Ejemplos;  
import javax.swing.*;  
import javax.swing.event.*;  
import org.apache.commons.net.PrintCommandListener;  
import org.apache.commons.net.ftp.*;  
import java.io.*;  
import java.awt.*;  
import java.awt.event.*;  
  
public class clienteFTPBasico extends JFrame {  
    private static final long serialVersionUID = 1L;  
  
    //campos de cabecera parte superior  
    static JTextField cab = new JTextField();  
    static JTextField cab2 = new JTextField();  
    static JTextField cab3 = new JTextField();  
  
    //campos de mensajes parte inferior  
    private static JTextField campo = new JTextField();  
    private static JTextField campo2 = new JTextField();  
  
    //botones  
    JButton botonCargar = new JButton("Subir fichero");  
    JButton botonDescargar = new JButton("Descargar fichero");  
    JButton botonBorrar = new JButton("Eliminar fichero");  
    JButton botonCreaDir = new JButton("Crear carpeta");  
    JButton botonDelDir = new JButton("Eliminar carpeta");  
    JButton botonSalir = new JButton("Salir");  
  
    //lista para los datos del directorio  
    static JList listaDirec = new JList();  
  
    //contenedor  
    private final Container c = getContentPane();  
  
    //Datos del servidor FTP (Servidor local)  
    static FTPClient cliente = new FTPClient();//cliente FTP  
    String servidor = "localhost";  
    String user = "usuario1";  
    String pasw = "usuario1";  
    boolean login;  
    static String direcInicial = "/";  
  
    //para saber directorio y fichero seleccionado  
    static String direcSelec = direcInicial;  
    static String ficheroSelec =””;
```

/ En el constructor se inicializan las variables , se llena la lista con los nombres de ficheros y directorios del directorio inicial y se pinta la pantalla. El código se muestra a continuación. Se han incluido las líneas más importante como son la conexión del cliente FTP , el establecimiento del directorio de trabajo inicial, la obtención de los ficheros de dicho directorio , el llenado de la lista de ficheros y directorios, y la colocación de la lista en la pantalla. Después se han incluido las acciones de respuesta cuando se pulsa un botón o en un elemento de la lista. */*

```
public clienteFTPBasico() throws IOException {
    super("CLIENTE BÁSICO FTP");
    System.out.println("Conectandose a " + servidor);
    // Para ver los comandos que se originan
    cliente.addProtocolCommandListener(new PrintCommandListener (new PrintWriter (System.out) ));
    cliente.connect(servidor); //Conexión al servidor
    cliente.enterLocalPassiveMode(); //Conexión modo pasivo
    login = cliente.login(user, pasw);
    //Se establece el directorio de trabajo actual
    cliente.changeWorkingDirectory(direcInicial);

    //Obteniendo ficheros y directorios del directorio actual
    FTPFile[] files = cliente.listFiles();
    //Construyendo la lista de ficheros y directorios del directorio de trabajo actual
    llenarLista(files, direcInicial);
    // preparar campos de pantalla
    campo.setBounds(new Rectangle(3, 485, 485, 30));
    campo.setForeground(Color.blue);
    campo.setFont(new Font("Verdana", Font.BOLD, 12));
    campo.setText("<<  ARBOL DE DIRECTORIOS CONSTRUIDO  >>");
    campo2.setBounds(new Rectangle(3, 515, 485, 30));
    campo2.setForeground(Color.blue);
    campo2.setFont(new Font("Verdana", Font.BOLD, 12));
    campo2.setText(" ");

    cab.setBounds(new Rectangle(5, 5, 200, 30));
    cab.setBorder(null);
    cab.setForeground(Color.blue);
    cab.setFont(new Font("Arial", Font.BOLD, 14));
    cab.setText("Servidor FTP: "+servidor);

    cab2.setBounds(new Rectangle(350, 5, 140, 30));
    cab2.setBorder(null);
    cab2.setFont(new Font("Arial", Font.BOLD, 14));
    cab2.setForeground(Color.blue);
    cab2.setText("Usuario: "+user);

    cab3.setBounds(new Rectangle(5, 34, 140, 30));
    cab3.setBorder(null);
    cab3.setFont(new Font("Arial", Font.BOLD, 14));
    cab3.setForeground(Color.blue);
    cab3.setText("DIRECTORIO RAIZ: "+direcInicial);

    botonCargar.setBounds(new Rectangle(350, 100, 140, 30));
    botonDescargar.setBounds(new Rectangle(350, 150, 140, 30));
    botonBorrar.setBounds(new Rectangle(350, 200, 140, 30));
    botonCreaDir.setBounds(new Rectangle(350, 250, 140, 30));
    botonDelDir.setBounds(new Rectangle(350, 300, 140, 30));
```



```
botonSalir.setBounds(new Rectangle(350, 350, 140, 30));
```

/* PREPARACION DE LA LISTA, se configura el tipo de selección para que solo se pueda seleccionar un elemento de la lista*/

```
listaDirec.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
```

//Barra de desplazamiento para la lista

```
JScrollPane barraDesplazamiento = new JScrollPane(listaDirec);  
barraDesplazamiento.setPreferredSize(new Dimension(335, 420));  
barraDesplazamiento.setBounds(new Rectangle(5, 65, 335, 420));  
c.add(barraDesplazamiento);  
c.setLayout(null);
```

// Se añaden el resto de los campos de pantalla

```
c.add(campo); campo.setEditable(false);  
c.add(campo2); campo2.setEditable(false);  
c.add(botonCargar); c.add(botonDescargar); c.add(botonBorrar);  
c.add(botonSalir);c.add(botonCreaDir);c.add(botonDelDir);  
c.add(cab);c.add(cab2);c.add(cab3);  
cab.setEditable(false);cab2.setEditable(false);cab3.setEditable(false);
```

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
setSize(510, 600);  
setVisible(true);
```

//Acciones al pulsar en la lista o en los botones

```
listaDirec.addListSelectionListener(new ListSelectionListener() {  
    public void valueChanged(ListSelectionEvent lse) {  
        if (lse.getValueIsAdjusting()) {  
  
            ficheroSelec="";  
  
            //elemento seleccionado de la lista  
            String fic = listaDirec.getSelectedValue().toString();  
  
            if (listaDirec.getSelectedIndex() == 0) {  
                //Se hace clic en el primer elemento del JList  
                if (!fic.equals(direcInicial)) {  
                    //si no estamos en el directorio inicial, hay que  
                    //subir al directorio padre  
                    try {  
                        cliente.changeToParentDirectory();  
                        direcSelec = cliente.printWorkingDirectory();  
  
                        cliente.changeWorkingDirectory(direcSelec);  
                        FTPFile[] ff2 = cliente.listFiles();  
                        campo.setText("");  
                        //se llena la lista con fich. del directorio padre  
                        llenarLista(ff2, direcSelec);  
                    } catch (IOException e) {e.printStackTrace();}  
                }  
            }  
            //No se hace clic 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");
            direcSelec = direcSelec2;
            llenarLista(ff2, direcSelec);
        } catch (IOException e2) {e2.printStackTrace();}
        } else {
            // SE TRATA DE UN FICHERO
            ficheroSelec = direcSelec;
            if (direcSelec.equals("/"))
                ficheroSelec += fic;
            else
                ficheroSelec += "/" + fic;
            campo.setText("FICHERO seleccionado:" + ficheroSelec);
            ficheroSelec=fic;
        } //fin else
    } //else de fichero o directorio
    campo2.setText("DIRECTORIO ACTUAL: " + direcSelec);
} //fin if inicial
}
}); // fin lista

// --al hacer clic en el botón Salir
botonSalir.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            cliente.disconnect();
        } catch (IOException e1) {
            e1.printStackTrace();
        }
        System.exit(0);
    }
});

//CREAR CARPETA
botonCreaDir.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String nombreCarpeta = JOptionPane.showInputDialog(null,
        "Introduce el nombre del directorio",
        "carpeta");
        if (!(nombreCarpeta==null)) {
            String directorio=direcSelec;
            if (!direcSelec.equals("/"))
                directorio =directorio +"/";
        }
    }
});
```

```
        directorio+=nombreCarpeta.trim() ;

        try {
            if (cliente.makeDirectory(directorio)) {
                String m= nombreCarpeta.trim() + " => Se ha creado
                correctamente ...";
                JOptionPane.showMessageDialog(null, m);
                campo.setText(m);
                cliente.changeWorkingDirectory(direcSelec);
                FTPFile[] ff2 = cliente.listFiles();
                llenarLista(ff2, direcSelec);

            } else
                JOptionPane.showMessageDialog(null,
                nombreCarpeta.trim()+ " => No se ha podido crear ...");

        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
});//..botonCreaDir

botonDelDir.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String nombreCarpeta = JOptionPane.showInputDialog(null,
        "Introduce el nombre del directorio a eliminar",
        "carpeta");
        if (!(nombreCarpeta==null)) {
            String directorio=direcSelec;
            if (!direcSelec.equals("/"))
                directorio =directorio +"/";
            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 = cliente.listFiles();
                    llenarLista(ff2, direcSelec);

                } else

                    JOptionPane.showMessageDialog(null, nombreCarpeta.trim()
                    + " => No se ha podido eliminar ...");

            } catch (IOException e1) {

                e1.printStackTrace();
            }
        }
    }
});
```

```
        }
    }
}); //..botonDelDir

// --al hacer clic en el botón Subir
botonCargar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JFileChooser f = new JFileChooser();
        f.setFileSelectionMode(JFileChooser.FILES_ONLY);
        f.setDialogTitle("Selecciona el Fichero a SUBIR AL SERVIDOR FTP");
        int returnVal = f.showDialog(f, "Cargar");
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = f.getSelectedFile();
            String archivo = file.getAbsolutePath();
            String nombreArchivo = file.getName();
            try {
                SubirFichero(archivo, nombreArchivo);
            } catch (IOException e1) {
                e1.printStackTrace();
            }
        }
    }
}); // Fin boon subir

// --al hacer clic en el botón Descargar
botonDescargar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String directorio=directSelec;
        if (!directSelec.equals("/"))
            directorio =directorio +"/";
        if (!ficheroSelec.equals(""))
        {
            DescargarFichero(directorio + ficheroSelec ,ficheroSelec);
        }
    }
}); // Fin boton descargar

botonBorrar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //System.out.println("borrar");
        String directorio=directSelec;
        if (!directSelec.equals("/"))
            directorio =directorio +"/";
        if (!ficheroSelec.equals(""))
            BorrarFichero(directorio + ficheroSelec ,ficheroSelec);
    }
}); //boton borrar
} //..FIN CONSTRUCTOR

// -----
```

/ Dentro del constructor se hace una llamada al método “llenarLista(files , direcInicial) . Este método se encarga de llenar la lista Jlist con los nombres de ficheros y directorios y mostrarla en pantalla. Recibe como primer parámetro un objeto FTPFile[] con los datos de ficheros del directorio actual y el nombre*

del directorio de trabajo actual (parámetro direc2). Para añadir elementos al Jlist se crea un objeto DefaultListModel. */

```
private static void llenarLista(FTPFile[] files, String direc2) {
    if (files == null) return;
    //Se crea un objeto DefaultListModel
    DefaultListModel modeloLista = new DefaultListModel();
    //Se define propiedades para la lista , color y tipo de fuente
    listaDirec.setForeground(Color.blue);
    Font fuente = new Font("Courier", Font.PLAIN, 12);
    listaDirec.setFont(fuente);
    //Se elimina los elementos de la lista
    listaDirec.removeAll();

    try {
        //Se establece el directorio de trabajo actual
        cliente.changeWorkingDirectory(direc2);
    } catch (IOException e) {
        e.printStackTrace();
    }
    direcSelec = direc2;
    // Se añade el directorio de trabajo al listmodel, primer elemento
    modeloLista.addElement(direc2);
    //Se recorre el array con los ficheros y directorios
    for (int i = 0; i < files.length; i++) {
        if (!(files[i].getName().equals(".")
            && !(files[i].getName().equals("..")))) {
            //Nos saltamos los directorios los directorios y ....
            //Se obtiene el nombre del fichero o directorio
            String f = files[i].getName();
            // Si es directorio se añade al nombre (DIR)
            if (files[i].isDirectory())
                f = "(DIR) " + f;
            //Se añade el nombre del fic o direc al listmodel
            modeloLista.addElement(f);
        } //if
    } // fin for
    try {
        // Se asigna el listmodel al JList, se muestra en pantalla la lista de ficheros y direc
        listaDirec.setModel(modeloLista);
    } catch (NullPointerException n) {
        //Al llegar al último aparece excepcion
        System.out.println("linea 334 - llega al ultimo");
    }
} //Fin llenarLista

private void DescargarFichero(String NombreCompleto, String nombreFichero) {

    String archivoyCarpetaDestino = "";
    String carpetaDestino = "";
    JFileChooser f = new JFileChooser();
    f.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
    f.setDialogTitle("Selecciona el Directorio donde DESCARGAR el fichero");

    int returnVal = f.showDialog(null, "Descargar");
    if (returnVal == JFileChooser.APPROVE_OPTION) {
```

```

File file = f.getSelectedFile();

carpetaDestino = (file.getAbsolutePath()).toString();
//System.out.println("carpeta destino " + carpetaDestino);
archivoyCarpetaDestino = carpetaDestino + File.separator
                        + nombreFichero;

try {
    cliente.setFileType(FTP.BINARY_FILE_TYPE);
    BufferedOutputStream out = null;
    out = new BufferedOutputStream(new FileOutputStream(
        archivoyCarpetaDestino));

    if (cliente.retrieveFile(NombreCompleto, out))
        JOptionPane.showMessageDialog(null, nombreFichero
            + " => Se ha descargado correctamente ...");
    else
        JOptionPane.showMessageDialog(null, nombreFichero
            + " => No se ha podido descargar ...");

    out.close();
} catch (IOException e1) {
    e1.printStackTrace();
}

} // ..DescargarFichero

private void BorrarFichero(String NombreCompleto, String nombreFichero) {
    //pide confirmacion
    int seleccion = JOptionPane.showConfirmDialog(null,
        "¿Desea eliminar el fichero seleccionado?");
    if(seleccion==JOptionPane.OK_OPTION) {
        try {

            if (cliente.deleteFile (NombreCompleto)) {
                String m= nombreFichero + " => Eliminado correctamente... ";
                JOptionPane.showMessageDialog(null, m);
                campo.setText(m);
                cliente.changeWorkingDirectory(direcSelec);
                FTPFile[] ff2 = cliente.listFiles();
                llenarLista(ff2, direcSelec);
            } else
                JOptionPane.showMessageDialog(null, nombreFichero
                    + " => No se ha podido eliminar ...");

        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
} // ..BorrarFichero
// -----
private boolean SubirFichero(String archivo, String nombreArchivo)
    throws IOException {
    System.out.println("Archivo : " +archivo);

```



```
        cliente.setFileType(FTP.BINARY_FILE_TYPE);
        BufferedInputStream in = new BufferedInputStream(new FileInputStream(
            archivo));
        boolean ok = false;
        // System.out.println("Directorio => " +direcSelec);
        cliente.changeWorkingDirectory(direcSelec);
        if (cliente.storeFile(nombreArchivo, in)) {
            String s = " " + nombreArchivo + " => Subido correctamente... ";
            campo.setText(s);
            campo2.setText("Se va a actualizar el árbol de directorios...");
            JOptionPane.showMessageDialog(null, s);
            FTPFile[] ff2 = cliente.listFiles();
            llenarLista(ff2, direcSelec);
            ok = true;
        } else
            campo.setText("No se ha podido subir... " + nombreArchivo);

        return ok;
    } // SubirFichero

    // main-----
    public static void main(String[] args) throws IOException {
        new clienteFTPBasico();
    } // ..FIN main

} // .fin clase
```