

Tema 4. Generación de Servicios en Red.

Objetivos

- Protocolos estándar de comunicación en red a nivel de aplicación (telnet, ftp, http, pop3, smtp, entre otros).
- Librerías de clases y componentes.
- Utilización de objetos predefinidos.
- Establecimiento y finalización de conexiones.
- Transmisión de información.
- Programación de aplicaciones cliente.
- Programación de servidores.
- Implementación de comunicaciones simultáneas.

Contenidos

1.- Conexión Servidor FTP desde Java.

Existen librerías en Java que permiten implementar el lado del cliente de muchos protocolo básicos de Internet como FTP, SMTP, etc...

Necesitamos la librería commons-net.x.y.jar para acceder y trabajar con ficheros y directorios en un Servidor FTP.

Conexión/Desconexión del Servidor.

La clase FTPClient encapsula toda la funcionalidad necesaria para almacenar y recuperar ficheros del Servidor FTP. Debemos conectar con connect() comprobando que no se ha producido ningún error, realizar las operaciones necesarias y desconectar con disconnect().

Ejemplo: Conexión al servidor ftp de rediris.es

```
package clienteftp;
import java.io.IOException;
import java.net.SocketException;
import org.apache.commons.net.ftp.*;
public class ClienteFTP {
    public static void main(String[] args) throws IOException {
        /*
        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
        */
    }
}
```

```

// Creamos un objeto de la clase FTPClient.
FTPClient cliente = new FTPClient();
// Creamos variable con el nombre del servidor.
String servFTP = "ftp.rediris.es"; // servidor FTP
System.out.println("Nos conectamos a: " + servFTP);
// Realizamos la conexión con el servidor
cliente.connect(servFTP); Nos permite conectarnos al servidor

// respuesta del servidor FTP
System.out.print(cliente.getReplyString());
// código de respuesta
int respuesta = cliente.getReplyCode();
// Mostramos el código de respuesta
System.out.println("Respuesta: " + respuesta);

// Comprobamos si la conexión se ha realizado de forma correcta
if (!FTPReply.isPositiveCompletion(respuesta)) {
    // Desconectamos del servidor
    cliente.disconnect();
    System.out.println("Conexión rechazada: " + respuesta);
    System.exit(0);
}
// desconexión del servidor FTP
cliente.disconnect();
System.out.println("Conexión finalizada.");
}
}

```

Salida:

```

run:
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.
BUILD SUCCESSFUL (total time: 0 seconds)

```

La clase `FTPReply` almacena los códigos de respuesta del Servidor FTP, siendo por ejemplo 220 significa que el servicio está preparado. Además, hay que tener en cuenta que los métodos de la clase `FTP` pueden lanzar `IOException` por cierre prematuro de la conexión.

Métodos de la clase `FTP`.

MÉTODOS	MISIÓN
<code>void connect(String host)</code>	Abre la conexión con el servidor FTP indicado en <i>host</i> .
<code>int getReplyCode()</code>	Devuelve el valor entero del código de respuesta de la última respuesta FTP.
<code>String getReplyString()</code>	Devuelve el texto completo de la respuesta del servidor FTP.

MÉTODOS	MISIÓN
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 <i>true</i> si se inicia la sesión con éxito, si no devuelve <i>false</i>
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 <i>path</i> .
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 <i>parent</i> .
boolean changeWorkingDirectory (String pathname)	Cambia el directorio de trabajo actual de la sesión FTP al indicado en <i>pathname</i> .
boolean changeToParentDirectory ()	Cambia al 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, OutputStream local)	Recupera un fichero del servidor y lo escribe en el OutputStream dado.
boolean deleteFile (String pathname)	Elimina un fichero en el servidor FTP.

MÉTODOS	MISIÓN
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.

Lo habitual es conectarnos a un Servidor FTP con usuario y contraseña. Utilizaremos el método `login()` que devuelve `true` si la conexión se realiza correctamente. Por otro lado, el método `listFiles()` devuelve un array de la clase `FTPFile` con la información de los ficheros y directorios encontrados. El método **`changeWorkingDirectory()`** permite movernos a un directorio, si existe devuelve `true` en caso contrario devuelve `false`.

Ejemplo acceso Servidor FTP anonimo:

```
package clienteftp1;
import java.io.IOException;
import java.net.SocketException;
import org.apache.commons.net.ftp.*;
public class ClienteFTP1 {
    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();
        }
    }
}
```

Salida:

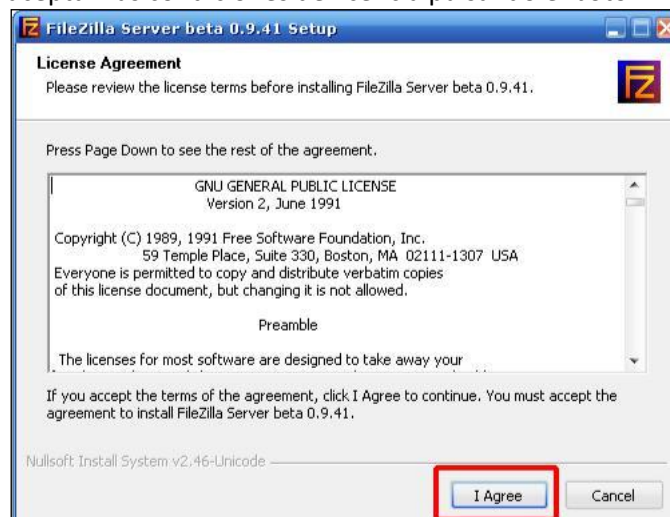
```
run:
Nos conectamos a: ftp.rediris.es
Login correcto...
Directorio actual: /
Ficheros en el directorio actual:7
. => Directorio
.. => Directorio
debian => Enlace simb.
debian-cd => Enlace simb.
mirror => Directorio
sites => Directorio
welcome.msg => Fichero
Logout del servidor FTP...
Desconectado...
BUILD SUCCESSFUL (total time: 1 second)
```

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

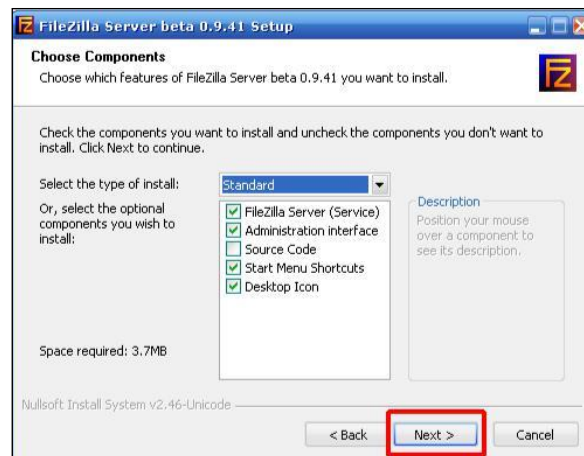
MÉTODOS	MISIÓN
String getName()	Devuelve el nombre del fichero.
long getSize()	Devuelve el tamaño del fichero en bytes.
int getType()	Devuelve el tipo del fichero, 0 si es un fichero (FILE_TYPE), 1 un directorio (DIRECTORY_TYPE) y 2 un enlace simbólico (SYMBOLIC_LINK_TYPE)
String getUser()	Devuelve el nombre del usuario propietario del fichero.
boolean isDirectory()	Devuelve <i>true</i> si el fichero es un directorio.
boolean isFile()	Devuelve <i>true</i> si es un fichero.
boolean isSymbolicLink()	Devuelve <i>true</i> si es un enlace simbólico.

Instalación Filezilla Server.

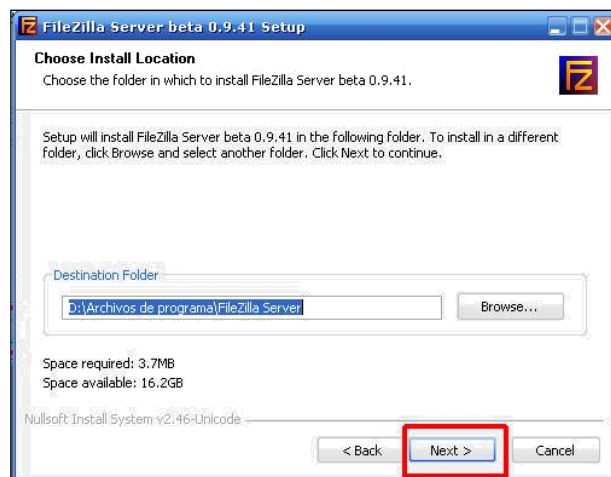
Para instalar Filezilla Server pulsando doble clic sobre el fichero de instalación **FileZilla_Server-0_9_60_2.exe** y se aceptan las condiciones de licencia pulsando el botón *I Agree*:



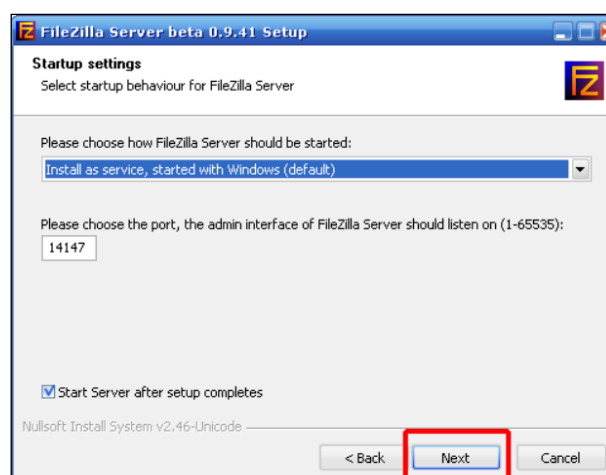
En la siguiente pantalla nos pide el tipo de instalación, dejamos la opción por defecto (*Standard*) y pulsamos *Next*:



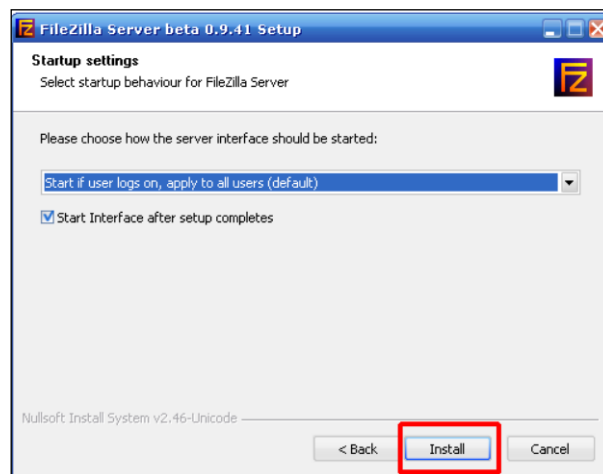
A continuación seleccionamos la carpeta donde se instalará o bien dejamos la opción por defecto. Se pulsa de nuevo *Next*:



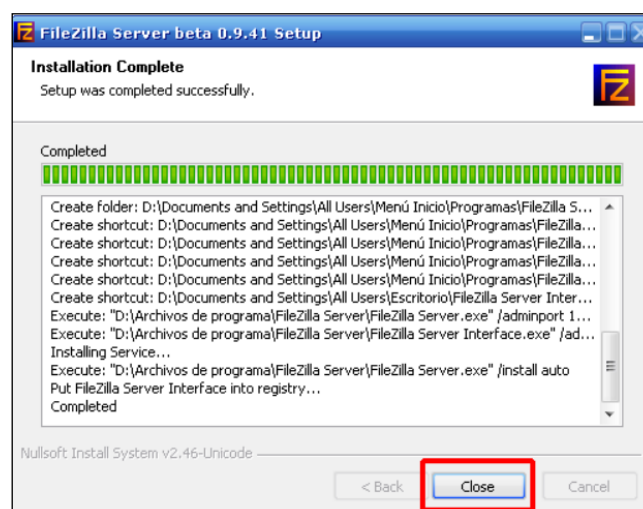
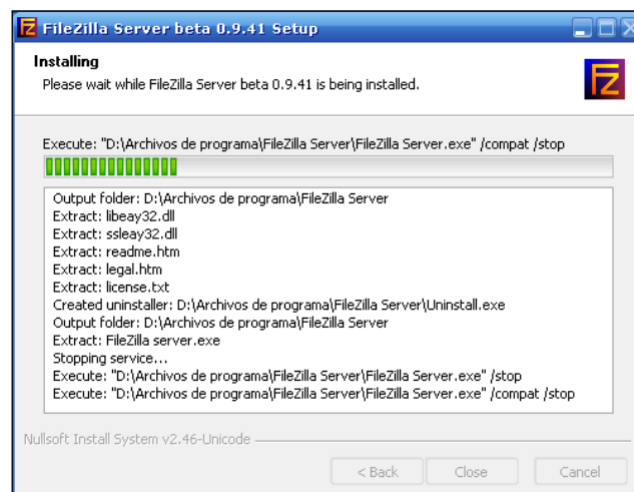
En las siguientes pantallas se definen las opciones de arranque de Filezilla. Se elige la opción de arranque por defecto: Instalación como un servicio de Windows (arranque automático), o se selecciona arranque manual. Se deja el puerto de escucha (14147) o se especifica otro puerto para ejecutar la herramienta de administración.




Por último se indica el modo de arranque de la herramienta de administración. Podemos dejar las opciones por defecto:

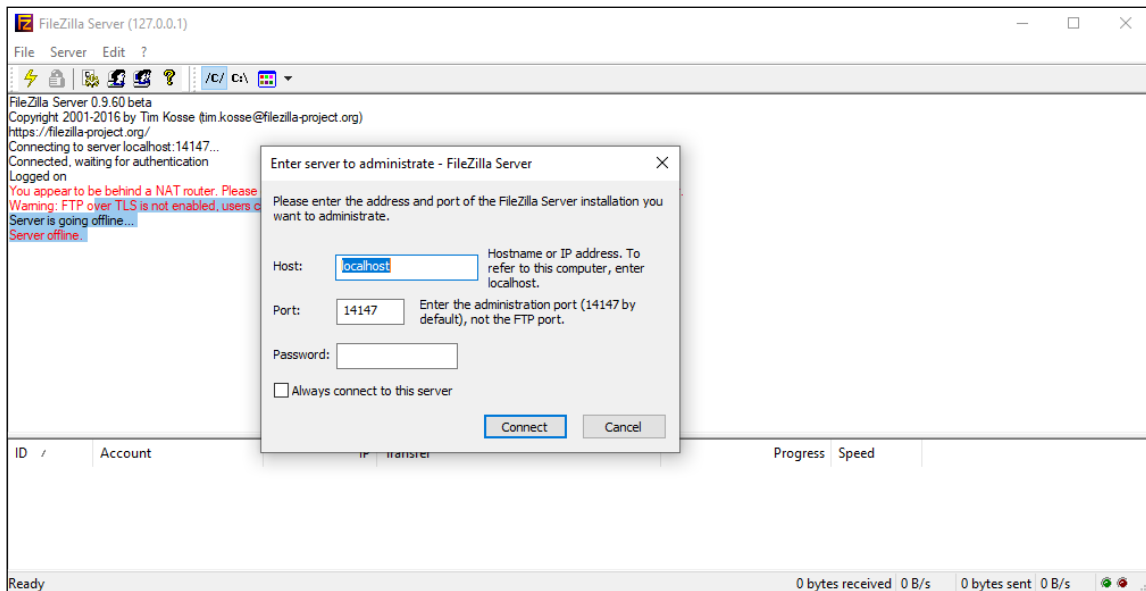



Una vez que hemos pulsado *Install* comienza el proceso de instalación. Cuando termine pulsamos el botón *Close*:

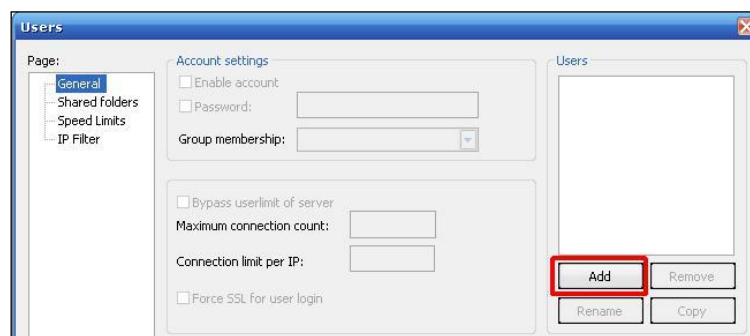


Puesta en marcha y Configuración.

En el escritorio aparece el icono de FileZilla . Hacemos doble clic para que se abra la herramienta de administración. Nos pide la dirección IP (por defecto asume la IP de la máquina local 127.0.0.1), el puerto (por defecto es 14147) y la clave de administración que es nula por defecto, pulsamos el botón *OK*. Se abre la pantalla de configuración:



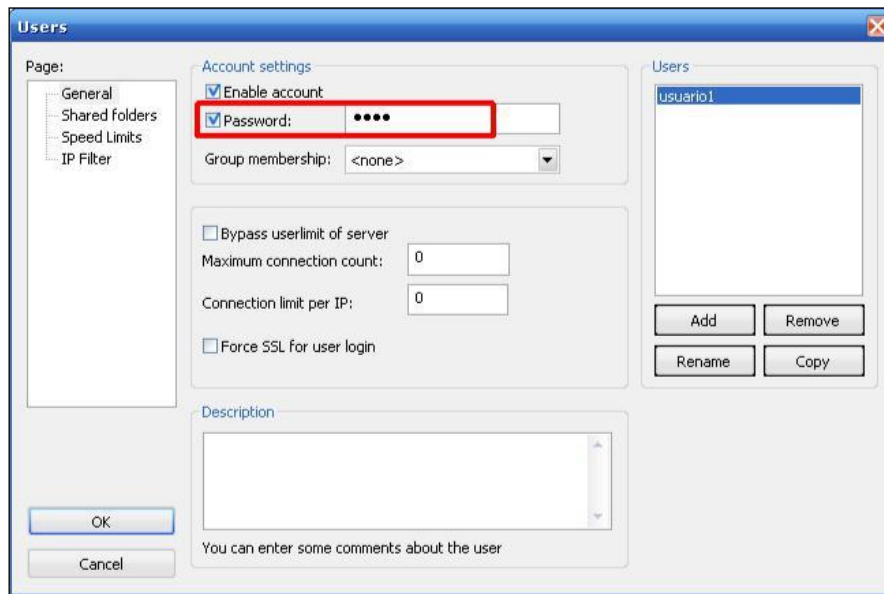
Creamos una carpeta en nuestro disco duro *C:\FTPServer* y pulsamos en la opción de menú *Edit->Users*, o bien hacemos clic en el icono . Se abre la pantalla *Users* para la gestión de usuarios:



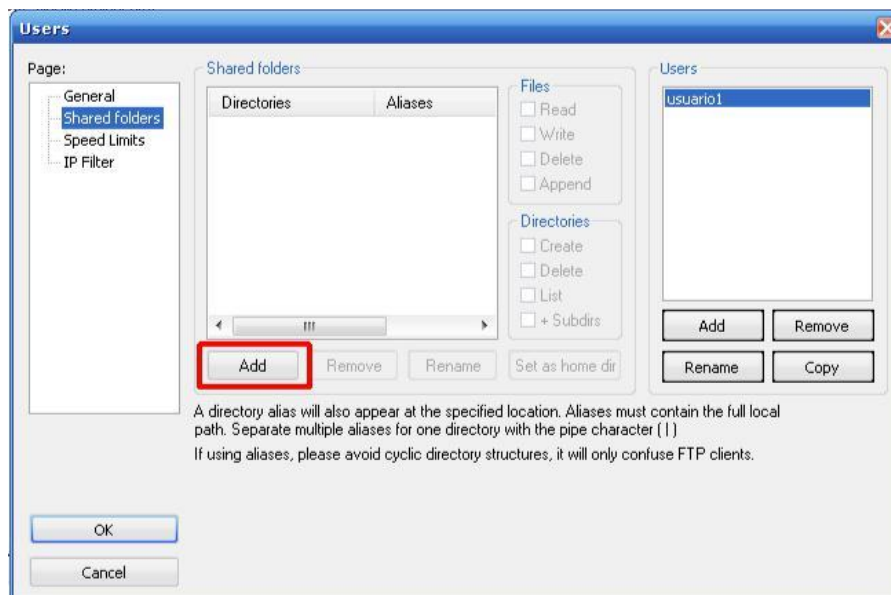
Desde el apartado **General**, pulsamos en el botón *Add*. Escribimos como nombre de usuario y pulsamos *OK*:



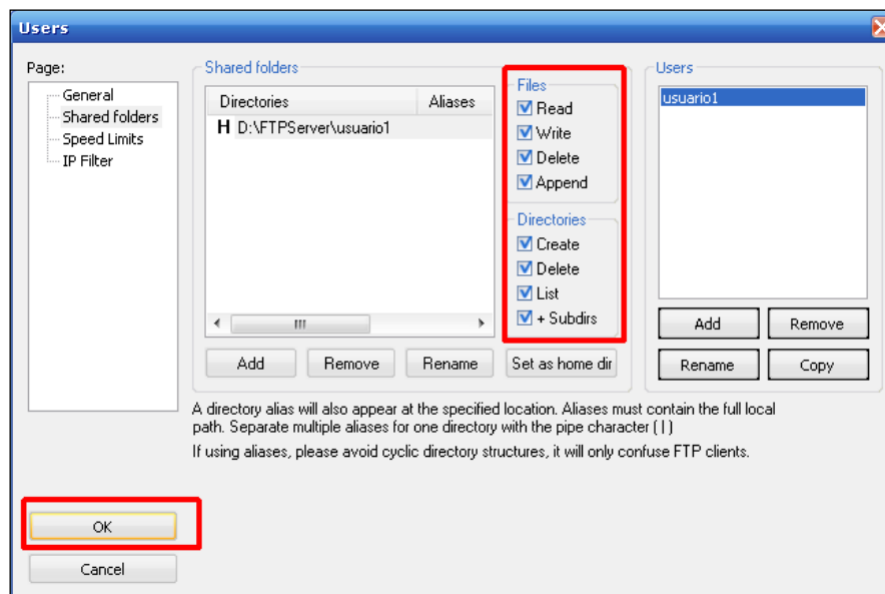
Se visualiza de nuevo la ventana *Users* con el usuario creado y le asignamos una clave desde el apartado *Account settings*:



Desde el apartado ***Shared folders*** indicamos la carpeta dentro del servidor FTP asignada al usuario. Pulsamos el botón **Add**:

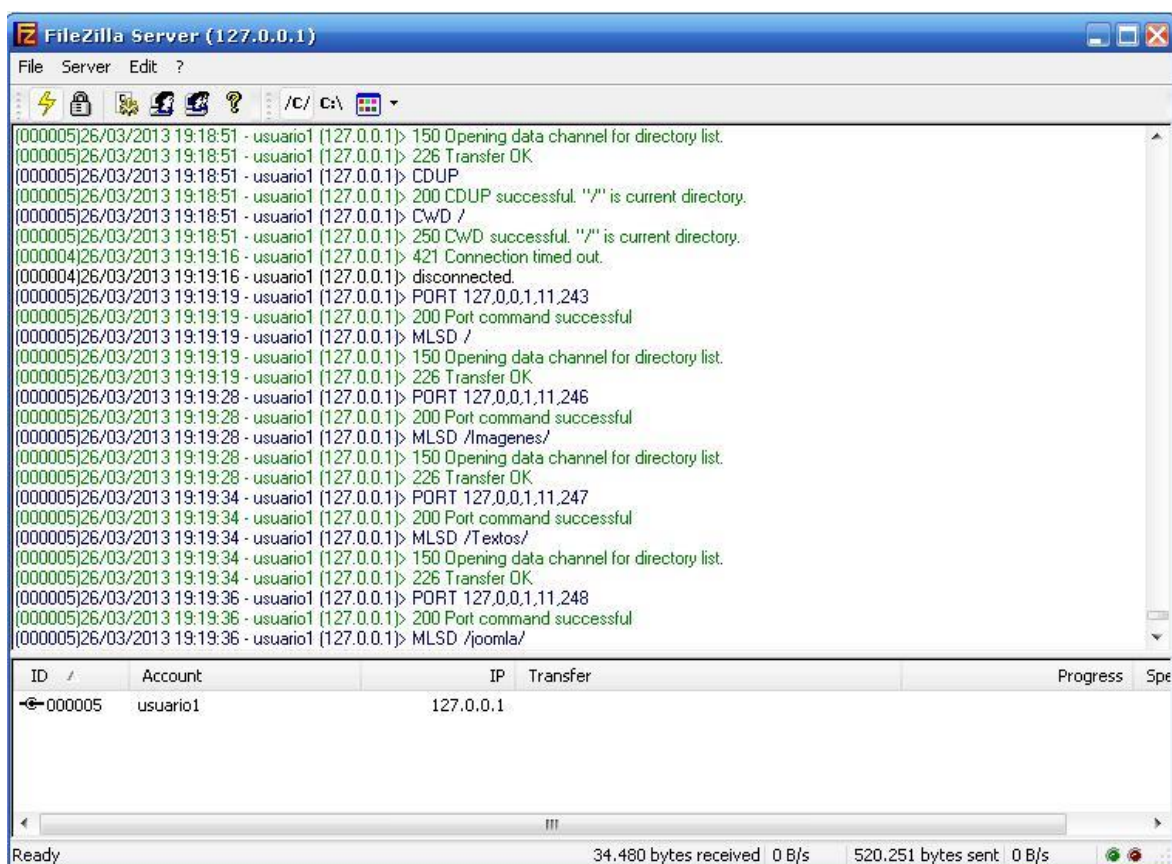


Se elige la carpeta *c:\FTPServer* y se asignan privilegios para que el usuario pueda crear y eliminar ficheros y crear y eliminar directorios. Se pulsa el botón **OK**:



Para probar el servidor FTP escribimos desde el navegador la URL: <ftp://127.0.0.1/>. Nos pedirá nombre de usuario y clave.

Pantalla con el log de las acciones y un usuario conectado.



Subir Ficheros al Servidor.

Para poder acceder al servidor FTP y subir/bajar ficheros debemos de tener acceso con un usuario con clave a su espacio en el servidor y tener privilegios. Los métodos setFileType(TIPO) indica el tipo de fichero a subir, normalmente binario y storeFile(nomfichero, InputStreamAlFichero).

Ejemplo:

```
package subirfichero;
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 = "admin";
        String pasw = "1111";

        try {
            System.out.println("Conectandose a " + servidor);
            cliente.connect(servidor);
            boolean login = cliente.login(user, pasw);

            cliente.setFileType(FTP.BINARY_FILE_TYPE);
            String direc = "/Pdfs";
            cliente.enterLocalPassiveMode();

            if (login) {
                System.out.println("Login correcto");

                if (!cliente.changeWorkingDirectory(direc)) {
                    String directorio = "Pdfs";

                    if (cliente.makeDirectory(directorio)) {
                        System.out.println("Directorio : "
                            + directorio + " creado ...");
                        cliente.changeWorkingDirectory(directorio);
                    } else {
                        System.out.println("No se ha creado Directorio");
                        System.exit(0);
                    }
                }

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

                String archivo = "c:\\Filezilla.pdf";
                BufferedInputStream in =
```

```

        new BufferedInputStream(new FileInputStream(archivo));

        if (cliente.storeFile("Filezilla.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();
    }
}
}

```

Salida:

```

run:
Conectandose a localhost
Login correcto
Directorio actual: /Pdfs
Subido correctamente...
BUILD SUCCESSFUL (total time: 0 seconds)

```

Renombrar fichero.

Para renombrar un fichero utilizaremos el método `rename(nombreantiguo, nombrenuevo)`. Devuelve `true` si renombra el fichero con éxito, en caso contrario devuelve `false`.

Eliminar fichero.

Para eliminar un fichero utilizaremos el método `deleteFile(nombrefichero)`. Devuelve `true` si ha podido eliminar el fichero con éxito, en caso contrario devuelve `false`.

Ejemplo:

```

package renombraryeliminarfchero;

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

public class RenombraryEliminarFchero {

    public static void main(String[] args) {
        FTPClient cliente = new FTPClient();

        String servidor = "localhost";
        String user = "admin";
        String pasw = "1111";
    }
}

```

```

try {
    System.out.println("Conectandose a " + servidor);
    cliente.connect(servidor);
    cliente.enterLocalPassiveMode();
    boolean login = cliente.login(user, pasw);

    cliente.setFileType(FTP.BINARY_FILE_TYPE);
    String direc = "/pdfs";

    if (login) {
        System.out.println("Login correcto");

        if (!cliente.changeWorkingDirectory(direc)) {
            String directorio = "/pdfs";

            if (cliente.makeDirectory(directorio)) {
                System.out.println("Directorio:"+directorio+"creado");
                cliente.changeWorkingDirectory(directorio);
            } else {
                System.out.println("No se ha creado Directorio");
                System.exit(0);
            }
        }

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

        //Renombrar fichero
        direc = "/pdfs";
        cliente.changeWorkingDirectory(direc);
        if (cliente.rename("Filezilla.pdf", "Archivo.pdf")) {
            System.out.println("Fichero renombrado... ");
        } else {
            System.out.println("No he podido renombar el Fichero");
        }

        System.out.println("\nDespues de renombrar el fichero: ");
        VerFicheros(cliente);

        //eliminar fichero
        String fichero = "/pdfs/Archivo.pdf";
        if (cliente.deleteFile(fichero)) {
            System.out.println("Fichero eliminado... ");
        } else {
            System.out.println("No se ha podido eliminar Fichero");
        }

        System.out.println("\nDespues de eliminar el fichero: ");
        VerFicheros(cliente);

        cliente.logout();
    }
}

```

```

        cliente.disconnect();
    }
} catch (IOException ioe) {
    ioe.printStackTrace();
}
} // main

private static void VerFicheros(FTPClient client) throws IOException {

    FTPFile[] files = client.listFiles(client.printWorkingDirectory());
    System.out.println("Ficheros en el directorio actual:" +
files.length);

    String tipos[] = {"Fichero", "Directorio"};

    for (int i = 0; i < files.length; i++) {
        System.out.println(files[i].getName() + " * " +
tipos[files[i].getType()]);
    }
} // VerFicheros
} // ..

```

Salida:

```

run:
Conectandose a localhost
Login correcto
Directorio actual: /pdfs
Fichero renombrado...

Despues de renombrar el fichero:
Ficheros en el directorio actual:1
Archivo.pdf * Fichero
Fichero eliminado...

Despues de eliminar el fichero:
Ficheros en el directorio actual:0
BUILD SUCCESSFUL (total time: 0 seconds)

```

Descargar Ficheros al Servidor.

Para poder acceder al servidor FTP y subir/bajar ficheros debemos de tener acceso con un usuario con clave a su espacio en el servidor y tener privilegios. Los métodos setFileType(TIPO) indica el tipo de fichero a subir, normalmente binario y retrieveFile(nomfichero, OutputStreamAlFichero).

Ejemplo:

```

package descargarfichero;
import java.io.*;
import org.apache.commons.net.ftp.*;
public class DescargarFichero {
    public static void main(String[] args) {

```



```

FTPClient cliente = new FTPClient();

String servidor = "localhost";
String user = "admin";
String pasw = "1111";

try {
    System.out.println("Conectandose a " + servidor);
    cliente.connect(servidor);
    cliente.enterLocalPassiveMode();
    boolean login = cliente.login(user, pasw);

    if (login) {
        System.out.println("Login correcto");

        //descargar fichero
        String direc = "/pdfs";
        cliente.changeWorkingDirectory(direc);

        //stream de salida para recibir el fichero descargado
        BufferedOutputStream out = new BufferedOutputStream(
            new
FileOutputStream("c:\\descargas\\Fichero.pdf"));

        if (cliente.retrieveFile("Fichero.pdf", out)) {
            System.out.println("Recuperado correctamente... ");
        } else {
            System.out.println("No se ha podido descargar... ");
        }

        out.close();

        cliente.logout();
        cliente.disconnect();
    }
} catch (IOException ioe) {
    ioe.printStackTrace();
}
}
}

```

Salida:

```

run:
Conectandose a localhost
Login correcto
Recuperado correctamente...
BUILD SUCCESSFUL (total time: 0 seconds)

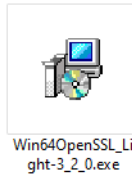
```

2.- Comunicación con Servidor SMTP.

SMTP (Simple Mail Transfer Protocol) es el protocolo estándar de internet para el intercambio de correos. Nos comunicamos con el Servidor SMTP por el puerto 465 mediante comandos.

Vamos a ver como comunicarnos con el Servidor SMTP y enviar mensajes mediante la aplicación cliente OpenSSL, por ejemplo OpenSSL Light.

Para instalar OpenSSL debemos descargar el fichero de la web <https://slproweb.com/products/Win32OpenSSL.html> y descargar la versión EXE para Windows.



Posteriormente lo instalamos pulsando doble clic sobre el ejecutable. La aplicación quedará instalada en C:\Program Files\OpenSSL-Win64\bin.

Envío de Correo desde OpenSSL.

Para enviar correo mediante OpenSSL debemos abrir en la carpeta donde se ha instalado OpenSSL una línea de comandos con:

```
cmd
```

Ejecutamos la conexión mediante openssl al servidor smtp por el puerto 465:

```
openssl s_client -connect smtp.ionos.es:465 -ign_eof
```

El parámetro -ign_eof evita el cierre de la conexión automático.

Una vez dentro del servidor ejecutamos las sentencias que permiten enviar el correo:

```
EHLO smtp.ionos.es
```

```
AUTH LOGIN
```

```
YWx1bW5vQGZvcmlhdmFsLmNvbQ==
```

```
UGFzcy4yMDE5
```

```
MAIL FROM: alumno@formaval.com
```

```
RCPT TO: mario@formaval.com
```

```
DATA
```

```
From: "Alumno" <alumno@formaval.com>
```

```
To: "Mario" <mario@formaval.com>
```

```
Subject: Mi asunto
```

```
Texto del mensaje.
```

Con tantas líneas como necesites.

.

QUIT

Las sentencias utilizadas realizan las siguientes funciones:

- EHLO permite iniciar el saludo con otro servidor de correo, esto con el fin de saber qué métodos son soportados
- AUTH LOGIN permite autenticarnos en el Servidor SMTP para el envío de correos introduciendo el usuario y la contraseña codificada en Base64.
- MAIL FROM indicamos cuenta correo remitente.
- RCPT TO indicamos cuenta correo destinatario.
- DATA indicamos el contenido del mensaje con From, To, Subject y dejando una línea en blanco introducimos el texto del mensaje. Terminamos el mensaje con una línea con un punto.
- QUIT para salir del Servidor SMTP.

Ejemplo de Código Java que devuelve en Base64 el usuario y la contraseña de conexión:

```
package base64encoding;
import java.util.Base64;
public class Base64Encoding {
    public static void main(String[] args) {
        Base64.Encoder enc = Base64.getEncoder();
        Base64.Decoder dec = Base64.getDecoder();
        String str = "alumno@formaval.com";

        // encode data using BASE64
        String encoded = enc.encodeToString(str.getBytes());
        System.out.println("Usuario: \t" + str);
        System.out.println("Encriptado: \t" + encoded);

        System.out.println("-----");
        str = "Pass.2019";

        // encode data using BASE64
        encoded = enc.encodeToString(str.getBytes());
        System.out.println("Contraseña: \t" + str);
        System.out.println("Encriptada: \t" + encoded);
    }
}
```

Salida de la ejecución del envío de mensaje por medio de telnet:

```
openssl s_client -connect smtp.ionos.es:465 -ign_eof
Connecting to 213.165.67.118
CONNECTED(000001B0)
:      :      :
```

```

:      :      :
Verify return code: 20 (unable to get local issuer certificate)
---
220 kundenserver.de (mreue109) Nemesis SMTP Service ready
EHLO smtp.ionos.es
250-kundenserver.de Hello smtp.ionos.es [178.237.233.136]
250-8BITMIME
250-AUTH LOGIN PLAIN
250 SIZE 140000000
AUTH LOGIN
334 VXN1cm5hbWU6
YWx1bW5vQGZvcmlhdmFsLmNvbQ==
334 UGFzc3dvcmQ6
UGFzcy4yMDE5
235 Authentication succeeded
MAIL FROM: alumno@formaval.com
250 Requested mail action okay, completed
RCPT TO: mario@formaval.com
250 OK
DATA
354 Start mail input; end with <CRLF>.<CRLF>
From: alumno@formaval.com
To: mario@formaval.com
Subject: Prueba
Este es un correo enviado por telnet.
.
250 Requested mail action okay, completed: id=1M20Eq-1rIRPs1Pyg-003uvw
QUIT
221 kundenserver.de Service closing transmission channel
closed

```

Java para comunicar con un Servidor SMTP.

La clase `AuthenticatingSMTPClient` que extiende de `SMTP` dispone de métodos que permiten enviar mensajes a través de un Servidor SMTP. Antes del envío es necesario conectarse y después debemos desconectarnos. Los códigos de respuesta del servidor nos permiten saber si la conexión se ha realizado correctamente.

Ejemplo:

```

package enviarcorreo;
// importamos las clases necesarias para la conexión segura

import java.io.*;
import java.security.*;
import java.security.spec.InvalidKeySpecException;
import org.apache.commons.net.smtp.*;

public class EnviarCorreo {

    public static void main(String[] args) throws IOException,
        NoSuchAlgorithmException, InvalidKeyException, InvalidKeySpecException {
        // Creamos las variables con los datos de conexión
        // Servidor, Puerto, Login, Password, Origen, Destino, Asunto y Mensaje

```

```

String servidor = "smtp.ionos.es";
int puerto = 465;
String login = "alumno@formaval.com";
String password = "Pass.2019";
String origen = "alumno@formaval.com";
String destino = "mario@formaval.com";
String asunto = "Saludos";
String mensaje = "Mensaje de bienvenida!!";

// Creamos instancia de AuthenticatingSMTPClient que permite la conexion
// usando SSL/TLS
AuthenticatingSMTPClient cliente = new AuthenticatingSMTPClient("TLS", true);

// Conectamos al servidor mediante el método connect a través del puerto
cliente.connect(servidor, puerto);

// Se envía el comando EHLO al servidor y muestra respuesta del servidor.
cliente.ehlo(servidor);

// Nos autenticamos con el usuario y la contraseña
cliente.auth(AuthenticatingSMTPClient.AUTH_METHOD.LOGIN, login, password);

// Creamos Cabecera del mensaje que vamos a enviar con el origen, destino y
// asunto
SimpleSMTPHeader header = new SimpleSMTPHeader(origen, destino, asunto);

// Indicamos quién realiza el envío y quién recibe el mensaje
cliente.setSender(origen);
cliente.addRecipient(destino);

// se crea instancia de Writer para el envío del texto del mensaje "DATA"
Writer writer = cliente.sendMessageData();

// Enviamos la cabecera y el mensaje
writer.write(header.toString());
writer.write(mensaje);

// Cerramos el stream
writer.close();

// Mensaje de correo enviado
System.out.println("Correo Enviado!!");

// Nos deslogamos y desconectamos
cliente.logout();
cliente.disconnect();
}
}

```

3.- Comunicación con Servidor POP.

Podemos conectarnos y comunicarnos con el Servidor de Correo Entrante POP por medio de OpenSSL. Abriremos el intérprete de comandos en la carpeta donde se ha instalado OpenSSL con:

cmd

Ejecutamos conexión mediante openssl al servidor correo entrante POP a través del puerto 995:

```
openssl s_client -connect pop.ionos.es:995 -quiet
```

El parámetro -quiet evita el cierre de la conexión automático.

Una vez dentro del servidor ejecutamos las sentencias que permiten enviar el correo:

```
+OK POP server ready H mieue001 1MY6jl-1heSP71N0f-00XEvM
USER alumno@formaval.com
+OK password required for user "alumno@formaval.com"
PASS Pass.2019
+OK mailbox "alumno@formaval.com" has 2 messages (58093 octets) H mieue001
STAT
+OK 2 58093
RETR 2
+OK
:      :      :      :      :
Este mensaje es para el alumno.
Un saludo
Mario
:      :      :      :      :
quit
+OK POP server signing off
```

Las sentencias utilizadas realizan las siguientes funciones:

- USER permite indicar el usuario para la conexión.
- PASS permite indicar el password para la conexión.
- STAT indica el estado del buzón mostrando el número de mensajes y el tamaño.
- RETR n: devuelve el contenido del mensaje n. Por ejemplo: RETR 2
- QUIT para salir y cerrar la conexión con el Servidor POP.

Java para comunicar con un Servidor POP.

La clase POP3SClient implementa el lado del cliente del protocolo POP3 con soporte SSL. Dispone de métodos que permite conectar, consultar y borrar los mensajes del Servidor POP.

MÉTODOS	MISIÓN
boolean deleteMessage(int messageId)	Elimina el mensaje con número <i>messageId</i> del servidor POP3. Devuelve <i>true</i> si la operación se realizó correctamente.
POP3MessageInfo listMessage(int messageId)	Lista el mensaje indicado en el parámetro <i>messageId</i> .
POP3MessageInfo[] listMessages()	Obtiene un array con información de todos los mensajes.

POP3MessageInfo listUniqueIdentifier(int messageId)	Obtiene la lista de un único mensaje.
boolean login(String username, String password)	Inicia sesión en el servidor POP3 enviando el nombre de usuario y la clave. Devuelve <i>true</i> si la operación se realizó correctamente.
boolean logout()	Finaliza la sesión con el servidor POP3. Devuelve <i>true</i> si la operación se realizó correctamente.
Reader retrieveMessage(int messageId)	Recupera el mensaje con número <i>messageId</i> del servidor POP3.
Reader retrieveMessageTop(int messageId, int numLines)	Igual que el anterior pero sólo el número de líneas especificadas en el parámetro <i>numLines</i> . Para recuperar la cabecera del mensaje <i>numLines</i> debe ser 0.

Ejemplo:

```
package recibircorreo;

import java.io.*;
import org.apache.commons.net.pop3.*;

public class RecibirCorreo {

    public static void main(String[] args) throws IOException {
        // Creamos las variables con los datos de conexion
        // servidor, puerto, usuario, password
        String servidor = "pop.ionos.es";
        String usuario = "alumno@formaval.com";
        String password = "Pass.2019";
        int puerto = 995;

        // Creamos instancia POP3Client que permite conexion usando SSL/TLS
        POP3Client cliente = new POP3SClient("TLS", true);

        // Realizamos mediante el método connect la conexion al servidor
        cliente.connect(servidor);
        System.out.println("Conexion al Servidor realizada");

        // Realizamos el login mediante usuario y password
        if (!cliente.login(usuario, password)) {
            System.err.println("Login no realizado");
            cliente.disconnect();
            return;
        }

        // Recuperamos la lista de mensajes
        POP3MessageInfo[] mensajes = cliente.listMessages();

        // Mostramos la cantidad de mensajes
        System.out.println("Numero de Mensajes: " + mensajes.length);

        // Recorremos con un bucle el array de mensajes creando un buffer de
```

```

// comunicación sobre cada mensaje mediante el numero identificador
for(POP3MessageInfo mensaje : mensajes) {
    BufferedReader buffer =
        (BufferedReader) cliente.retrieveMessage(mensaje.number);
    //Mostramos el contenido del mensaje
    mostrarMensaje(buffer, mensaje.number);
}

// Nos deslogamos y desconectamos
cliente.logout();
cliente.disconnect();
}

public static void mostrarMensaje(BufferedReader buffer, int id)
    throws IOException {
    String origen = "";
    String asunto = "";
    String linea;
    int inicio;
    int fin;
    while ((linea = buffer.readLine()) != null) {
        String lower = linea.toLowerCase();
        if (lower.startsWith("from: ")) {
            if (linea.contains("<")) {
                inicio = linea.indexOf("<") + 1;
                fin = linea.indexOf(">");
                origen = linea.substring(inicio, fin).trim();
            } else {
                origen = linea.substring(6).trim();
            }
        } else if (lower.startsWith("subject: ")) {
            asunto = linea.substring(9).trim();
        }
    }

    System.out.println("Id Mensaje: "+Integer.toString(id) +
        "\nRemitente: "+ origen +"\nAsunto: " + asunto
        + "\n-----");
}
}

```