

Módulo Profesional 09:
Programación de servicios y procesos

Actividad UF3

CICLO FORMATIVO DE GRADO SUPERIOR EN

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

MODALIDAD ONLINE

Enrique Verea



Nombre de la actividad

Objetivos

Poner en practica los conocimiento de programación de sockets y conceptos de servicios en red.

Competencias asociadas:

- Sockets.
- Servicios de red.

Metodología

- Preparación individual

Entrega

12/12/23 en PDF

Dedicación estimada

600 minutos

Documentos de referencia

Recursos de la UF3.

Resultados de aprendizaje

- RA1. Programa mecanismos de comunicación en red empleando zócalos y analizando el escenario de ejecución
- RA2. Desarrolla aplicaciones que ofrecen servicios en red, utilizando librerías de clases y aplicando criterios de eficiencia y disponibilidad

Criterios de evaluación

- Criterio 1. Realiza aplicaciones de enlazamiento y establecimiento de conexiones.
- Criterio 2. Utilización de zócalos para la transmisión y recepción de información.
- Criterio 3. Programación de aplicaciones cliente y servidor.
- Criterio 4. Utilización de hilos en la programación de aplicaciones en red.
- Criterio 5. Analiza librerías de clases y componentes, que utiliza objetos predefinidos que permitan la implementación de protocolos estándar de comunicación en red a nivel de aplicación (telnet, ftp, http, pop3, smtp, entre otros).
- Criterio 6. Programa y verifica aplicaciones cliente de protocolos estándar.
- Criterio 7. Programa servidores, que permitan el establecimiento de conexiones, la transmisión de información y la finalización de conexiones.
- Criterio 8. Analiza requerimientos para servidores concurrentes.
- Criterio 9. Implementa comunicaciones simultáneas.
- Criterio 10. Depura y documenta aplicaciones

Desarrollo de la actividad

Bloque1. Sockets

Ejercicio1. [1punto]

Cita cuales son las características más importantes de los protocolos UDP Y TCP, y especifica ejemplos para que se usa cada uno de ellos.

TCP

- Requiere que se establezca una conexión entre servidor y cliente antes de comenzar la comunicación entre ambos
- Hace seguimiento de todos los segmentos que envía
- Garantiza la entrega de todos los segmentos
- Garantiza que todos los segmentos son recibidos en el orden en que han sido enviados
- Ofrece un sistema de comprobación de errores que contribuye a garantizar la entrega de segmentos
- Permite el envío de datos de manera bidireccional
- Es utilizado para comunicaciones que requieran mantener la integridad de datos, como por ejemplo, servicios de e-mail, transferencia de ficheros, etc.

UDP

- No requiere establecer una conexión para el envío de datos, simplemente realiza envíos a un ordenador/cliente objetivo
- No hace seguimiento a los segmentos enviados, ni garantiza su entrega ni orden específico
- Envía datos de manera unidireccional
- Al ser mucho más simple que TCP, es considerablemente más rápido
- Es utilizado para comunicaciones que requieran velocidad de transmisión y en las que la pérdida de algunos datos no sea crítica, como por ejemplo, el streaming de vídeos, música, etc.

Ejercicio2. [1punto]

Busca y explica al menos 4 métodos de la clase socket en Java.

bind(SocketAddress)	Conecta el Socket la dirección local facilitada
connect(SocketAddress)	Conecta el Socket al servidor en la dirección facilitada
getInputStream	Abre un flujo de entrada de datos desde el Socket
getOutputStream	Abre un flujo de salida de datos hacia el Socket
close()	Cierra el Socket y sus flujos de entrada y salida. Una vez que ha sido cerrado, un Socket no puede volver a ser utilizado; uno nuevo debe ser creado para resumir las comunicaciones

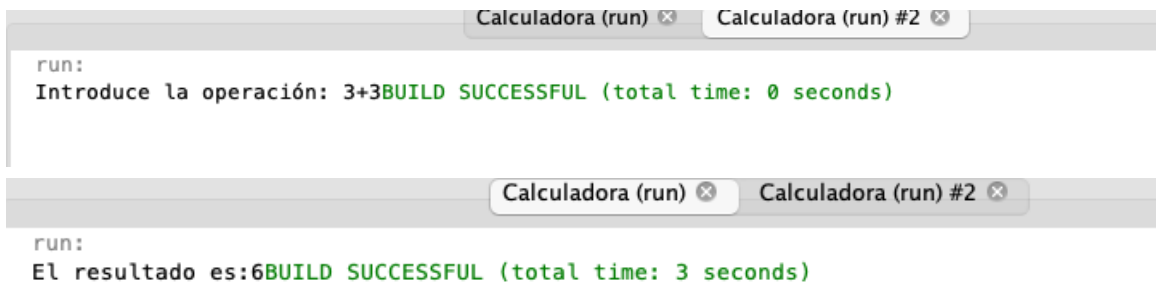
Ejercicio3. [4puntos] +info 1 videoconferencia

Realiza un programa cliente-servidor que haga de calculadora:

- El cliente solicitará la operación a realizar.
- El servidor, extraerá la operación, y los operandos y mostrará el resultado .

Nota: Para simplificar tu programa, esta calculadora solo sumará/ restará y multiplicará números comprendidos entre el 0 y el 9.

Ejemplo:
Cliente: 3+3
Servidor 3+3=6



```

Calculadora (run) x Calculadora (run) #2 x
run:
Introduce la operación: 3+3BUILD SUCCESSFUL (total time: 0 seconds)

Calculadora (run) x Calculadora (run) #2 x
run:
El resultado es:6BUILD SUCCESSFUL (total time: 3 seconds)
  
```

Cilente

Clase Cliente

```
public class Cliente {

    private Socket socket;
    private BufferedReader input;
    private PrintWriter output;

    public String comenzarConexion(String ip, int puerto) throws SocketTimeoutException {
        try {
            socket = new Socket(ip, puerto);

            output = new PrintWriter(socket.getOutputStream(), true);
            input = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            return "Conexión establecida";
        }
        catch (IOException e) {
            if (e instanceof SocketTimeoutException)
                throw (SocketTimeoutException) e;
            return "No se ha podido crear el cliente. Causa: " + e.getMessage();    } }
    public boolean estaConectado() {
        return socket.isConnected();
    }
}

    public String enviarMensaje(String mensaje) {
        try {
            output.println(mensaje);
            return input.readLine();
        }
        catch (IOException e) {
            System.err.println(e.getMessage());
            return "No se ha podido enviar el mensaje. Causa: " + e.getMessage();
        }
    }

    public void cerrarConexion() {
        try {
            output.close();
            input.close();
            socket.close();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Clase Main

```
public class Main {

    public static void main(String[] args) {
        Cliente clienteFtp = new Cliente();

        boolean conectado = clienteFtp.iniciarConexion("demo.wftpservers.com", 21);

        if (!conectado) {
            System.out.println("No se ha podido establecer una conexión. Finalizando programa");
            System.exit(1);
        }

        boolean login = clienteFtp.login("demo", "demo");

        if (!login) {
            System.out.println("No se ha podido iniciar sesión. Finalizando programa");
            System.exit(1);
        }

        System.out.println("Archivos en raíz:");
        listarArchivos(clienteFtp, "/");

        System.out.println("Archivos en /download:");
        listarArchivos(clienteFtp, "/download");

        System.out.println("Descargando archivo 'version.txt'");
        descargarArchivo(clienteFtp, "version.txt", "/download/version.txt");

        System.out.println("Subiendo archivo 'version.txt' a '/upload/version2.txt'");
        subirArchivo(clienteFtp, "version.txt", "/upload/version2.txt");
    }

    private static void listarArchivos(Cliente clienteFtp, String ruta) {
        String respuesta = clienteFtp.listarArchivos(ruta);
        System.out.println(respuesta);
        System.out.println();
    }

    private static void subirArchivo(Cliente clienteFtp, String local, String remoto) {
        String respuesta = clienteFtp.subirArchivo(local, remoto);
        System.out.println(respuesta);
        System.out.println();
    }

    private static void descargarArchivo(Cliente clienteFtp, String local, String remoto) {
        String respuesta = clienteFtp.descargarArchivo(local, remoto);
        System.out.println(respuesta);
        System.out.println();
    }
}
```

Servidor

Clase Servidor

```
public class Servidor {

    private static final String CERRAR_CONEXION = "SALIR";

    private ServerSocket socket;
    private Socket clientSocket;
    private BufferedReader input;
    private PrintWriter output;
    private final Calculadora calculadora = new Calculadora();

    public void escucharEnPuerto(int puerto) throws IOException {
        socket = new ServerSocket(puerto);
        System.out.println("Servidor escuchando en puerto: " + puerto);

        clientSocket = socket.accept(); // bloquea el hilo hasta recibir una solicitud

        output = new PrintWriter(clientSocket.getOutputStream(), true);
        input = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

        gestionarSolicitudes();
    }

    private void gestionarSolicitudes() {
        while (true) {
            try {
                String solicitud = input.readLine(); // bloquea el hilo hasta recibir una nueva línea

                if (solicitud == null)
                    continue;

                if (solicitud.equalsIgnoreCase(CERRAR_CONEXION))
                    break;

                String respuesta = calculadora.realizarOperacion(solicitud);

                output.println(solicitud + " = " + respuesta);
            }
            catch (IOException e) {
                output.print("Error: no se ha podido generar una respuesta");
                e.printStackTrace();
            }
        }

        cerrarConexion();
    }
}
```



```
private void cerrarConexion() {
    try {
        output.print("Conexión finalizada");
        output.close();
        input.close();
        clientSocket.close();
        socket.close();
    }
    catch (IOException e) {
        System.err.println("La conexión se ha podido cerrar correctamente");
        e.printStackTrace();
    }
}
}
```

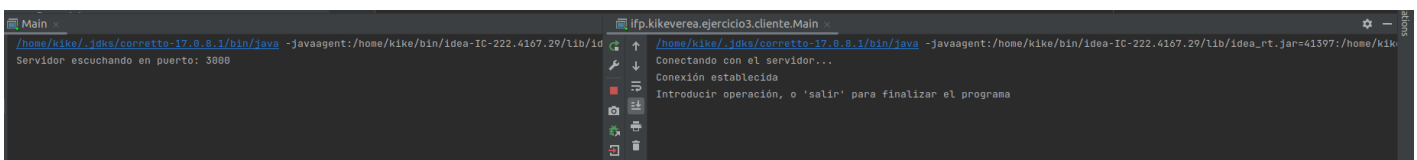
Clase Main

```
public class Main {

    public static void main(String[] args) {
        try {
            Servidor servidor = new Servidor();
            servidor.escucharEnPuerto(3000);

            System.out.println("Servidor desconectado");
        }
        catch (IOException e) {
            System.err.println("No se ha podido iniciar el servidor");
            e.printStackTrace();
        }
    }
}
```

Conexión Inicial



Operaciones

```

/home/kike/.jdk/corretto-17.0.8.1/bin/java -javaagent:/home/kike/bin/idea-IC-222.4167.29/lib/idea_rt.jar=40553:/home/kike/bin/idea-IC-222.4167.29/lib/idea_rt.jar:
Servidor escuchando en puerto: 3000

Conectando con el servidor...
Conexión establecida
Introducir operación, o 'salir' para finalizar el programa
3+3
3+3 = 6
Introducir operación, o 'salir' para finalizar el programa
3-3
3-3 = 0
Introducir operación, o 'salir' para finalizar el programa
2.5*2.5
2.5*2.5 = 6.25
Introducir operación, o 'salir' para finalizar el programa
5/2
5/2 = 2.5
Introducir operación, o 'salir' para finalizar el programa
0/2
0/2 = 0
Introducir operación, o 'salir' para finalizar el programa
5/0
5/0 = Error: División entre cero
Introducir operación, o 'salir' para finalizar el programa

```

Desconexión

```

/home/kike/.jdk/corretto-17.0.8.1/bin/java -javaagent:/home/kike/bin/idea-IC-222.4167.29/lib/idea_rt.jar=40553:/home/kike/bin/idea-IC-222.4167.29/lib/idea_rt.jar:
Servidor escuchando en puerto: 3000
Servidor desconectado
Process finished with exit code 0

Conectando con el servidor...
Conexión establecida
Introducir operación, o 'salir' para finalizar el programa
3+3
3+3 = 6
Introducir operación, o 'salir' para finalizar el programa
3-3
3-3 = 0
Introducir operación, o 'salir' para finalizar el programa
2.5*2.5
2.5*2.5 = 6.25
Introducir operación, o 'salir' para finalizar el programa
5/2
5/2 = 2.5
Introducir operación, o 'salir' para finalizar el programa
0/2
0/2 = 0
Introducir operación, o 'salir' para finalizar el programa
5/0
5/0 = Error: División entre cero
Introducir operación, o 'salir' para finalizar el programa
Programa finalizado
Conexión finalizada
Process finished with exit code 0

```

Bloque2. Servicios

Ejercicio4. [1punto]

Realiza una tabla, donde amplíes la información de los servicios en red vistos en clase, identifica para qué se usa cada servicio, que puerto suele utilizar, que tipo de transferencias realiza y el significado de sus siglas.

Nombre protocolo	Descripción	Puertos	Significado siglas	Tipo Transferencias
FTP	Protocolo de comunicación para la transferencia de ficheros en una arquitectura cliente-servidor	20, 21	File Transfer Protocol	Ficheros
TELNET	Protocolo de red sobre una arquitectura cliente-servidor usado para acceder sistemas remotos	23	Teletype Network	Texto
SMTP	Protocolo de comunicación para transferencia de correos electrónicos	587	Simple Mail Transfer Protocol	Email, a través de comandos de texto
HTTP	Protocolo con un modelo solicitud-respuesta, que pertenece a la capa OSI de aplicación. Es la base de comunicación de internet	80	Hypertext Transfer Protocol	Hipertexto
DNS	Sistema de nomenclatura que traduce nombres de dominios a su correspondiente dirección IP	53	Domain Name System	Texto

Ejercicio5.[3puntos] +info 2 videoconferencia

Realiza un programa en java que realice una conexión FTP al servidor:

<http://demo.wftpserver.com>. A continuación, se detallan el orden de las operaciones que se deben realizar:

1. Conexión al servidor <http://demo.wftpserver.com> con java.
2. Lógate al servidor con java.
3. Listar los archivos de la raíz. con java.
4. Listar los archivos de /download con java.
5. Sube manualmente un archivo.doc
6. Descargar el archivo.doc con java.
7. Sube el archivo.doc a /uploads con el nombre archivo2.doc con java.
8. Desconéctate del servidor con java.

Clase Cliente

```
public class Cliente {

    private final FTPClient cliente = new FTPClient();

    public boolean iniciarConexion(String servidor, int puerto) {
        try {
            cliente.connect(servidor, puerto);
            cliente.enterLocalPassiveMode();
            return FTPReply.isPositiveCompletion(cliente.getReplyCode());
        }
        catch (IOException e) {
            e.printStackTrace();
            return false;
        }
    }

    public boolean login(String usuario, String contrasena) {
        try {
            cliente.login(usuario, contrasena);
            return FTPReply.isPositiveCompletion(cliente.getReplyCode());
        }
        catch (IOException e) {
            e.printStackTrace();
            return false;
        }
    }

    public String listarArchivos(String ruta) {
        try {
            String archivos = Arrays
                .stream(cliente.listFiles(ruta))
                .map(FTPFile::getName)
                .collect(Collectors.joining(", "));

            return FTPReply.isPositiveCompletion(cliente.getReplyCode())
        }
    }
}
```

```

        ? archivos :
        error();
    }
    catch (IOException e) {
        return error(e);
    }
}

public String subirArchivo(String local, String remoto) {
    try (InputStream flujoDesdeLocal = new FileInputStream(local)) {
        cliente.appendFile(remoto, flujoDesdeLocal);

        return FTPReply.isPositiveCompletion(cliente.getReplyCode())
            ? "Archivo " + local + " subido a " + remoto + ""
            : error();
    }
    catch (IOException e) {
        return error(e);
    }
}

public String descargarArchivo(String local, String remoto) {
    try (OutputStream flujoHaciaLocal = new FileOutputStream(local)) {
        cliente.retrieveFile(remoto, flujoHaciaLocal);
        flujoHaciaLocal.flush();

        return FTPReply.isPositiveCompletion(cliente.getReplyCode())
            ? "Archivo " + remoto + " descargado a " + local + ""
            : error();
    }
    catch (IOException e) {
        return error(e);
    }
}

private String error() {
    return "Error " + cliente.getReplyCode() + ": " + cliente.getReplyString();
}

private String error(IOException e) {
    return "Error de Entrada/Salida: " + e.getMessage();
}
}

```

Clase Main

```
public class Main {

    public static void main(String[] args) {
        Cliente clienteFtp = new Cliente();
        boolean conectado = clienteFtp.iniciarConexion("demo.wftpservers.com", 21);

        if (!conectado) {
            System.out.println("No se ha podido establecer una conexión. Finalizando programa");
            System.exit(1);
        }

        boolean login = clienteFtp.login("demo", "demo");

        if (!login) {
            System.out.println("No se ha podido iniciar sesión. Finalizando programa");
            System.exit(1);
        }

        System.out.println("Archivos en raíz:");
        listarArchivos(clienteFtp, "/");

        System.out.println("Archivos en /download:");
        listarArchivos(clienteFtp, "/download");

        System.out.println("Descargando archivo 'version.txt'");
        descargarArchivo(clienteFtp, "version.txt", "/download/version.txt");

        System.out.println("Subiendo archivo 'version.txt' a '/upload/version2.txt'");
        subirArchivo(clienteFtp, "version.txt", "/upload/version2.txt");
    }

    private static void listarArchivos(Cliente clienteFtp, String ruta) {
        String respuesta = clienteFtp.listarArchivos(ruta);
        System.out.println(respuesta);
        System.out.println();
    }

    private static void subirArchivo(Cliente clienteFtp, String local, String remoto) {
        String respuesta = clienteFtp.subirArchivo(local, remoto);
        System.out.println(respuesta);
        System.out.println();
    }

    private static void descargarArchivo(Cliente clienteFtp, String local, String remoto) {
        String respuesta = clienteFtp.descargarArchivo(local, remoto);
        System.out.println(respuesta);
        System.out.println();
    }
}
```

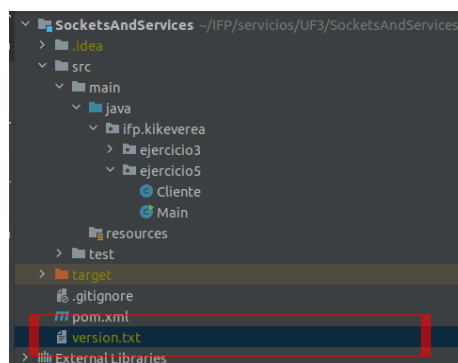
```
/home/kike/.jdk/corretto-17.0.8.1/bin/java -javaagent:/home/kike/bin/idea-IC-222.4167.29/lib/idea_rt.jar=33659:/home/kike/bin/idea-IC-222.4167.29/t
Archivos en raíz:
upload, download

Archivos en /download:
dji_mini2.mp4, Winter.jpg, printable_restaurant_menu.pptx, wftpsrvr-mac-i386.tar.gz, version.txt, wftpsrvr-linux-64bit.tar.gz, Autumn.jpg, Summe

Descargando archivo 'version.txt'
Archivo '/download/version.txt' descargado a 'version.txt'

Subiendo archivo 'version.txt' a '/upload/version2.txt'
Archivo 'version.txt' subido a '/upload/version2.txt'


Process finished with exit code 0
```



```

1 Wing FTP Server v7.2.5 Released: 15/Sep/2023
2 -----
3 Improvement - Increased the maximum of available open handles per SFTP session.
4 Improvement - Added a system option "Disallow OpenSSL session caching".
5 Improvement - Added a system option "Disallow OpenSSL session resumption on renegotiation".
6 Fixed a bug - When existing pictures with the same filename under different subfolders, you will always see the first pi
7 Fixed a bug - When uploading a picture to overwrite the old one, you will still download or see the old picture due to b
8
9
10 Wing FTP Server v7.2.4 Released: 9/Aug/2023
11 -----
12 Fixed a bug - When using Active (PORT) mode in FTPS transfer, the service might crash under some special conditions.
13 Fixed a bug - When adding a Wing Gateway with an existing port (but is not Gateway's), the service might crash.
14 Improvement - When generating or viewing the weblink for a banned file, now it will show the error message "banned file
15 Improvement - Improved the security and process speed for the web admin's session ID.
16 Improvement - Improved the process speed for the web client's session ID.
17

```




**WING FTP
Server**

Upload File




New

Bookmarks

More Actions

 / upload

includes 3 files & 0 folders

	Name	Size	Type	Modified
<input type="checkbox"/>	 version2.txt	105 KB	txt File	2023-12-04 09:04:21
<input type="checkbox"/>	 S_RTIDDP357076_INSURANCE ELIGIBILITY_BCBS of Alabama (Commercial) TERTIARY.pdf	36.3 KB	pdf File	2023-12-04 09:02:53
<input type="checkbox"/>	 Personstanding.jpg	168.1 KB	jpg File	2023-12-04 09:09:08