



***Materia: Aplicaciones para Comunicaciones en Red***

***Nombre del Trabajo:***

***Práctica 1. Servicio de transferencia de archivos***

***Alumno:***

***Peña Atanasio Alberto – 2020630367***

***Martínez Coronel Brayan Yosafat -  
2019630143***

***Maestra: Moreno Cervantes Axel***

***Grupo: 3CM17***

***Fecha de entrega: 15/03/2022***

## Introducción

El envío de archivos a través de la red es una característica importante para la gran mayoría de las aplicaciones que hoy día se utilizan (blogs, redes sociales, mensajería instantánea, Declaración de impuestos, educación en línea, etc.), sin embargo, no todas las aplicaciones disponibles permiten el envío de archivos de gran tamaño (p.e. El correo electrónico no permite enviar archivos de más de 10 o 20 MB).

Esto hace necesario el desarrollo de aplicaciones que permitan transferir archivos sin importar el tamaño de éstos.

En la práctica se implementará un programa que tenga un cliente y un servidor que simulen a Google Drive y un usuario normal, respectivamente. Donde las principales opciones serán:

1. Subir archivos
2. Subir carpeta
3. Eliminar archivos
4. Eliminar carpeta
5. Descargar archivos
6. Descargar carpeta

## Desarrollo de la práctica

### Abrir el proyecto

Para la correcta ejecución del programa se tienen que abrir el proyecto “MenuyAccionesMezclados” con NetBeans, dicho proyecto se encuentra en la misma carpeta donde esta guardo este archivo Word actual, y también ejecutar los archivos seleccionados, que se aprecian en la Figura 1.

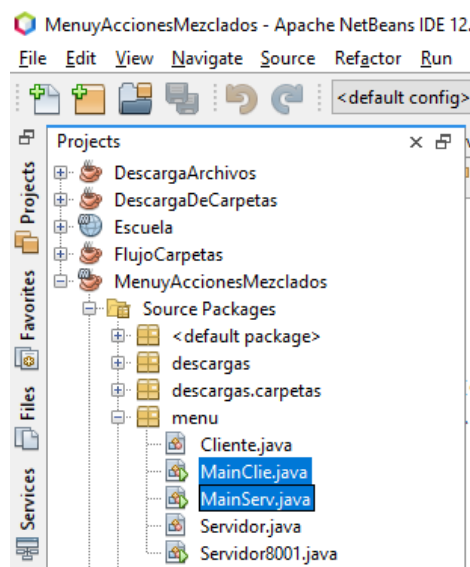
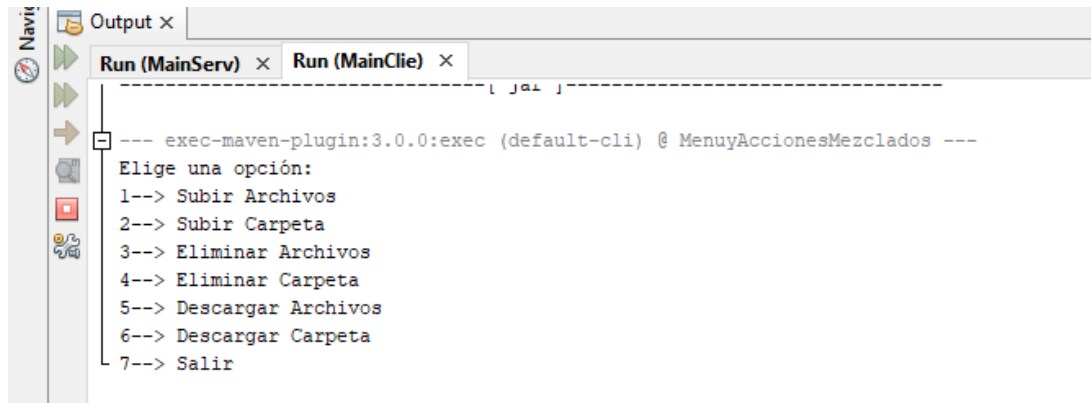


Figura 1

## Menú inicial

Esos 2 archivos, no son más que un menú del lado del cliente y del servidor.

Al ejecutar los 2 archivos aparece el menú. Como se ve en la Figura 2.



```
Output x
Run (MainServ) x Run (MainClie) x
--- exec-maven-plugin:3.0.0:exec (default-cli) @ MenuyAccionesMezclados ---
Elige una opción:
1--> Subir Archivos
2--> Subir Carpeta
3--> Eliminar Archivos
4--> Eliminar Carpeta
5--> Descargar Archivos
6--> Descargar Carpeta
7--> Salir
```

Figura 2

## Opción 1: subir archivos

Si se ingresa el 1, se ejecutarán las funciones de la Figura 3 y 4.

```
switch (choice) {
    case 1:
        CEnvia cl = new CEnvia();
        cl.envioDeCliente(); //Aquí va la de msj1
        cl.cl.close(); //Tenemos que cerrar el socket, para invocar otra función
        System.out.println();
        break;
```

Figura 3

```
if (choice == 1) {
    /*
     * ---> EJECUTANDO EL ENVIO DE
     */
    SRecibe s1 = new SRecibe();
    s1.recibeServidor(); //Aquí
    SRecibe.s.close(); //Cerramos
    s.cl.close();
```

Figura 4

La función `envioDeCliente()` lo que hace es lanzar un `FileChooser` para elegir archivos a subir, conectar el socket y mandar a llamar a la función `mandaArchivo()`, esta última función se ejecutará iterativamente para que se manden los archivos a través de un flujo orientado a Bytes.

Se puede apreciar de mejor manera si abre los archivos seleccionados que aparecen en la figura 5.

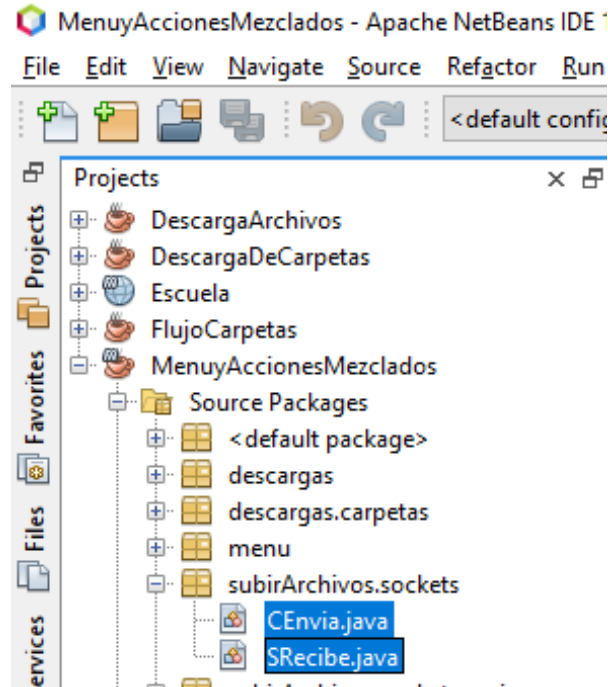


Figura 5

### Opción 2: Subir carpetas

Si se elige la opción 2, entonces ahora se podrán subir carpetas.

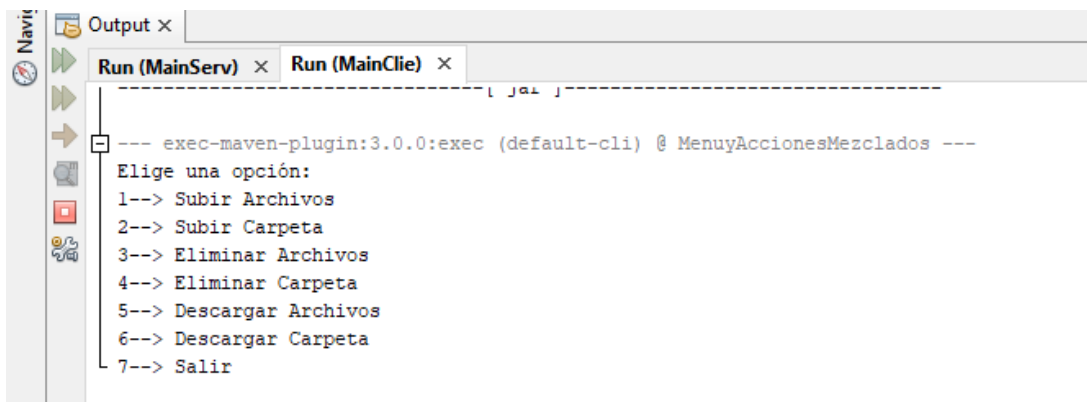


Figura 6

Prácticamente, se sigue la misma lógica que la opción 1, lo único que cambia aquí es que al FileChooser se le configura para que se seleccione sólo una carpeta. Luego, se manda el nombre de la carpeta del lado del cliente y esa carpeta se crea del lado del servidor. Se van recibiendo todos los archivos dentro de la carpeta a través de un envío orientado a Objetos. Para ver la lógica más a detalle, abrir los archivos seleccionados que aparecen en la Figura 7.

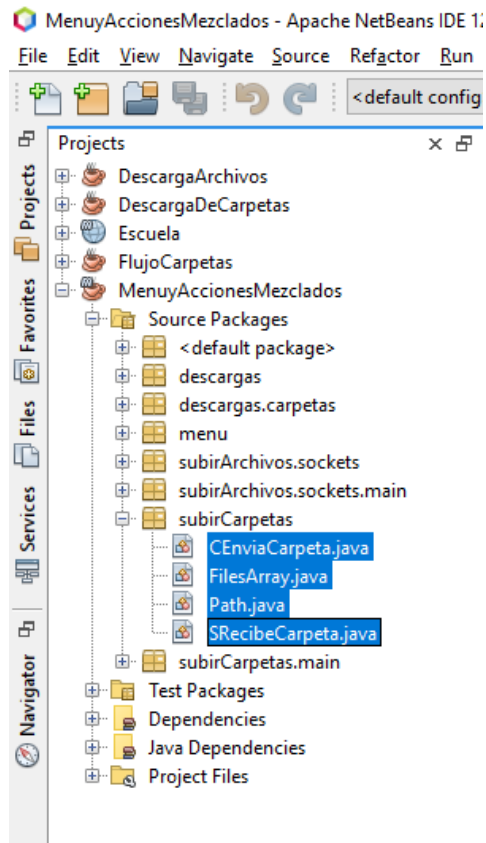


Figura 7

### Opción 3: Eliminar archivos

Si se introduce un 3, se podrán eliminar archivos.

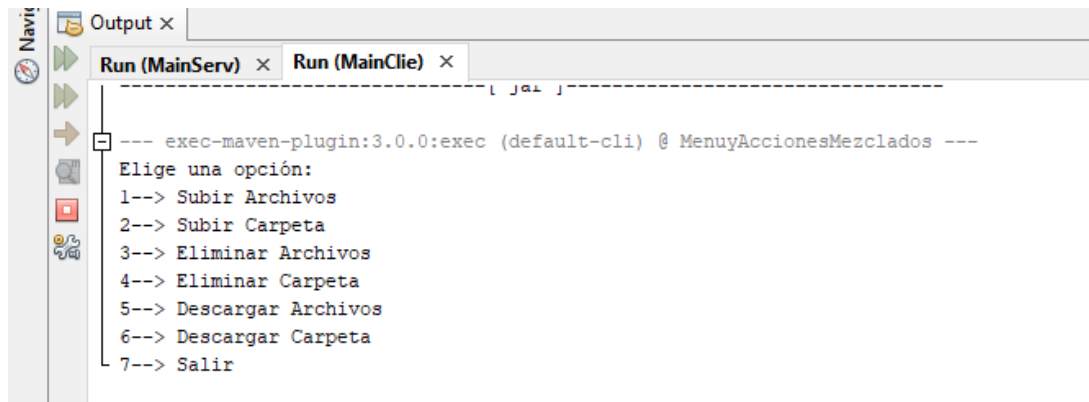


Figura 8

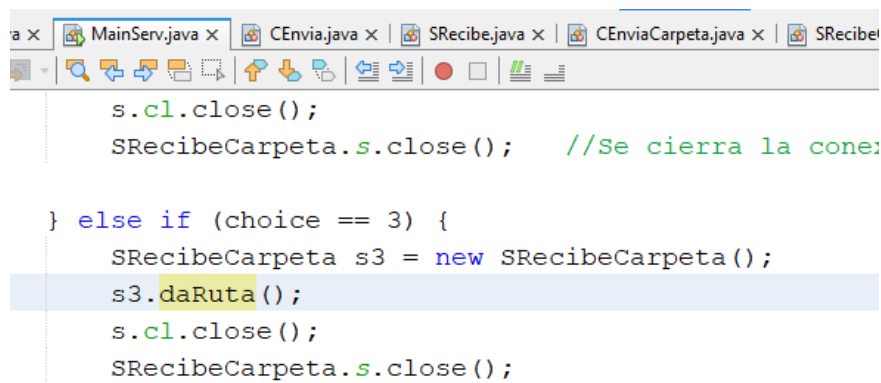
La lógica aquí es lanzar un FileChooser que muestre los archivos que tiene la nube, esto se hace a través de una comunicación de sockets bloqueantes.

Luego el usuario selecciona los archivos que quiera eliminar y dichos nombres de archivos serán recuperados del lado del servidor, para posteriormente ser eliminados.

```
case 3:
    CEnviaCarpeta cEliminaFile = new CEnviaCarpeta();
    System.out.println(cEliminaFile.getRuta());
    String ruta = cEliminaFile.getRuta(); //Obteniendo la ruta
    cEliminaFile.eliminarArchivo(ruta);
    cl.cl.close();
    System.out.println();

    break;
```

Figura 9



```
s.cl.close();
SRecibeCarpeta.s.close(); //Se cierra la conexión

} else if (choice == 3) {
    SRecibeCarpeta s3 = new SRecibeCarpeta();
    s3.daRuta();
    s.cl.close();
    SRecibeCarpeta.s.close();
```

Figura 10

#### Opción 4: Eliminar carpeta

Esta elección permite eliminar carpeta. La lógica aquí es exactamente casi igual a la de la opción 3. A diferencia de que aquí, primero se eliminan los archivos de la carpeta y luego la propia carpeta.

Se lanza un file chooser del lado del cliente y el servidor recupera los nombres de los archivos seleccionados para poder eliminarlos.

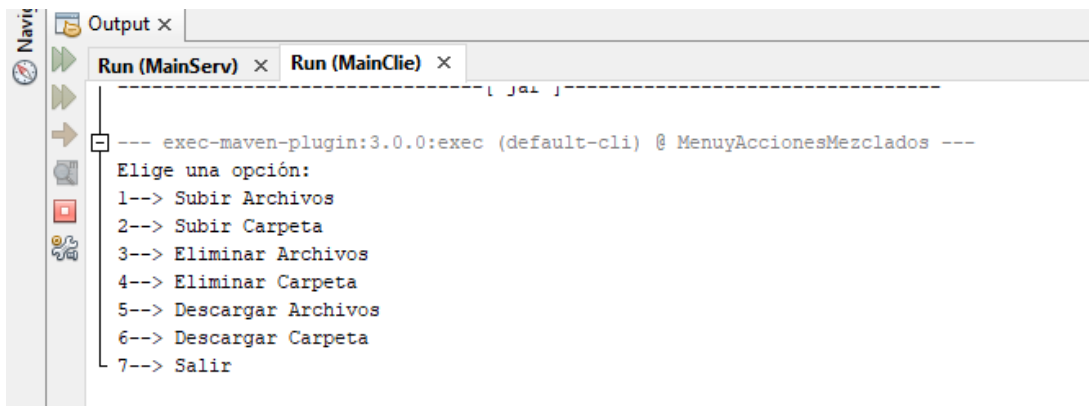


Figura 11

Esto se hace del lado del cliente. Se ve en la figura 12.

```
case 4:
    CEnviaCarpeta cEliminaCarpeta = new CEnviaCarpeta();
    System.out.println(cEliminaCarpeta.getRuta());
    String ruta2 = cEliminaCarpeta.getRuta(); //Obtenien
    cEliminaCarpeta.eliminaCarpeta(ruta2);
    cl.cl.close();
    System.out.println();

    break;
```

Figura 12

Lo que está en la Figura 13 se ejecuta del lado del Servidor.

```
} else if (choice == 4) {
    SRecibeCarpeta s4 = new SRecibeCarpeta();
    s4.daRuta();
    s.cl.close();
    SRecibeCarpeta.s.close();
```

Figura 13

### Opción 5 y 6: Descarga archivos y carpeta

Si se elijen las opciones 5 y 6, prácticamente se hace lo mismo que se hacía en las opciones 1 y 2. Solamente que aquí, quien envía los archivos es el Servidor hacia el cliente.

Lo que se muestra en la figura 14 es lo que se hace del lado del cliente.

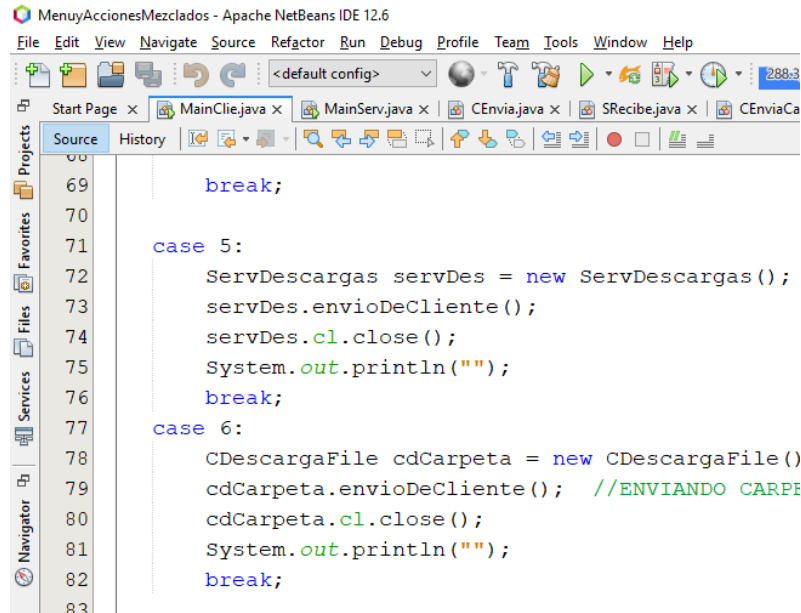


Figura 14

En la figura 15, se muestran las funciones que se mandan a llamar del lado del servidor.

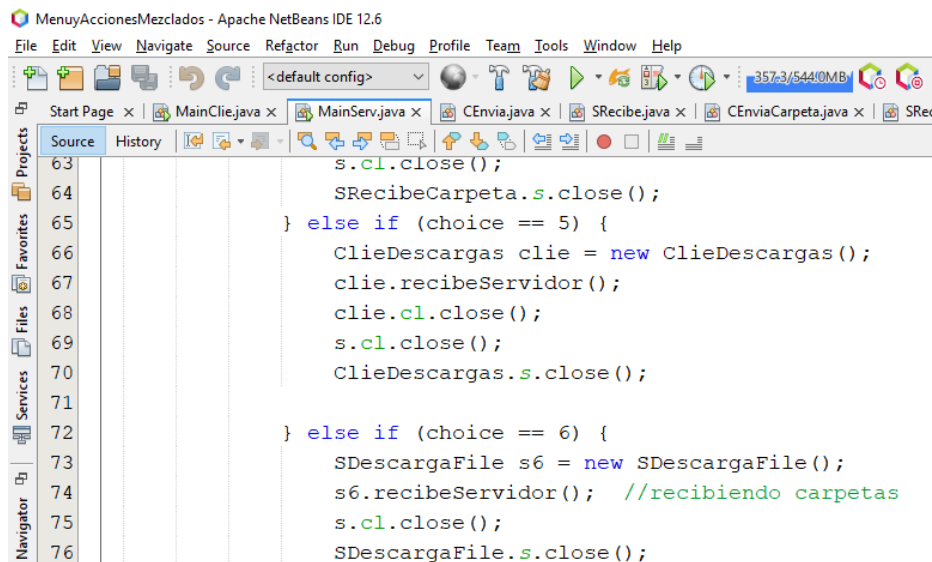


Figura 15



## Cierre

### Conclusión

Esta práctica fue muy interesante y reforzó totalmente los conocimientos que ya se tenían acerca de los Sockets de flujo bloqueantes, Flujos Orientado a Bytes, Flujo orientados a objetos, Sistemas de archivos, lado del Cliente, lado del Servidor, tiempo de conexión del servidor. Además, se comprendieron las clases Socket y SocketServer, las cuales eran totalmente desconocidas.

Creemos que con los conocimientos obtenidos al realizar esta práctica estamos más preparados para los temas siguientes de la materia de Aplicaciones para Comunicaciones en Red.

### Preguntas

**1. ¿Qué tipo de archivos se enviaron más rápido?**

Los de menor peso.

**2. ¿Cuál fue el número máximo de archivos que fue posible enviar a la vez?**

Uno nada más, debido a que son sockets de flujo bloqueantes, sólo se puede enviar uno a la vez.

**3. ¿Cuál fue el tamaño de archivo más grande que se pudo transferir? ¿por qué?**

Se pudo mandar cualquier tamaño del archivo, puesto que los paquetes se estaban enviando en trozos. Los bytes del archivo se mandaban por partes.

**4. ¿Qué es el orden de red?**

El orden en el que el servidor y el cliente envían y reciben los bytes de información en la red, respectivamente.

**5. ¿Por qué razón es importante utilizar el orden de red al enviar los datos a través de un socket?**

Para que no se pierda la información de cada archivo y los bytes de información lleguen en orden.

**6. Si deseáramos enviar archivos de tamaño muy grande, ¿qué cambios sería necesario hacer con respecto a los tipos de datos usados para medir el tamaño de los archivos, así como para leer bloques de datos del archivo?**

Tal vez nada, debido a que se diseñó el programa para que se puedan mandar los paquetes de información, en pedazos y no todos de golpe.