|  |  |  |
| --- | --- | --- |
| Resultado de imagen para logo ipn | Instituto Politécnico Nacional  Escuela Superior de Computo | Resultado de imagen para logo escom |

Aplicaciones para Comunicaciones de Red.

Práctica #1. Servicio de transferencia de archivos.

Grupo: 3CM5

Alumnos:

Cazares Martínez Maximiliano.

Cisneros Rosales Christian Iván.

Profesor:

Moreno Cervantes Axel Ernesto.

# **Introducción.**

Los sockets son mecanismos de comunicación entre procesos que permiten que un proceso envié o reciba información con otro proceso incluso estando en distintas máquinas.

Un socket, haciendo una analogía, es al sistema de comunicación entre ordenadores lo que un buzón o un teléfono es al sistema de comunicación entre personas. Es un punto de comunicación entre dos procesos por el cual se puede enviar o recibir información.

La forma de referenciar un socket por los procesos implicados es mediante un descriptor del mismo tipo que el utilizado para referenciar ficheros. Se podrá realizar redirecciones de los archivos de Entrada-Salida estándar a los sockets y así combinar entre ellos aplicaciones de la red. Todo nuevo proceso creado heredará, por tanto, los descriptores de sockets de su padre.

La comunicación entre procesos a través de sockets se basa en la filosofía Cliente-Servidor. Un proceso en esta comunicación actuará de proceso servidor creando un socket cuyo nombre conocerá el proceso cliente, que podrá "hablar" con el proceso servidor a través de la conexión con dicho socket. El otro proceso actuará como cliente creando un socket sin nombre cuyo descriptor usará para leer o escribir. El enlace entre los dos sockets permite una comunicación bidireccional, característica propia de los sockets y que los diferencia de las tuberías las cuales son canales de comunicación unidireccional entre procesos de una misma máquina.

El mecanismo de comunicación vía sockets tiene los siguientes pasos.

1. El proceso servidor crea un socket con nombre y espera la conexión.
2. El proceso cliente crea un socket sin nombre.
3. El proceso cliente realiza una petición de conexión al socket servidor.
4. El cliente realiza la conexión a través de su socket mientras el proceso servidor mantiene el socket servidor original con nombre.

Normalmente, un servidor se ejecuta en una computadora específica y tiene un socket enlazado a un número de puerto específico. El servidor sólo espera, escuchando en el socket, hasta que el cliente realice una solicitud de conexión.

En el lado del cliente; el cliente conoce el nombre de host de la computadora en la cual se está ejecutando el servidor, y el número de puerto en el cual se encuentra conectado el servidor. Para realizar la solicitud de conexión, el cliente trata de comunicarse con el servidor utilizando la dirección IP de la computadora donde esté se está ejecutando y el número de puerto.

Si todo funciona adecuadamente, el servidor acepta la conexión. Durante la aceptación, el servidor obtiene un socket enlazado a un puerto diferente. Requiere un nuevo socket para poder continuar escuchando en el socket original por otras solicitudes de conexión, mientras atiende las solicitudes del cliente conectado.

En el lado del cliente, si la conexión es aceptada, se crea un socket, que puede ser utilizado para comunicarse con el servidor. El socket en el lado del cliente no se enlaza al número de puerto utilizado para establecer la conexión con el servidor. En lugar de esto, al cliente se le asigna un número de puerto local, en la máquina en la cual se está ejecutando. Ahora el cliente y el servidor pueden comunicarse, escribiendo o leyendo en sus respectivos sockets.

# **Objetivo.**

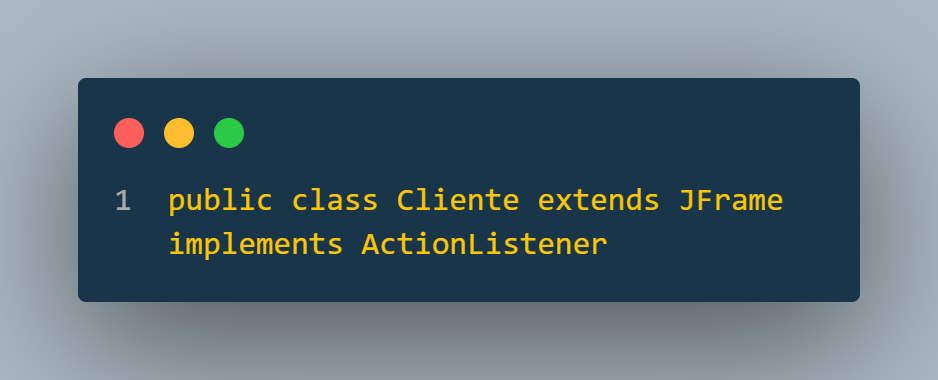
El estudiante implementará una aplicación para el envío de múltiples archivos a través de la red haciendo uso de sockets de flujo.

# **Desarrollo.**

A continuación, se explicará con detalle el código de la práctica. La práctica se desarrolló en java y el proyecto consta de 3 archivos .java; CCliente, Cliente y Servidor. En la clase Cliente está el código de una pequeña interfaz de usuario que posteriormente explicaremos. La clase CCliente tiene el código para él envió y borrado de carpetas y archivos y en la clase Servidor esta todo el código para recibir tanto archivos como carpetas.

## **Clase Cliente.**

Como se mencionó, esta clase despliega una pequeña interfaz de usuario y para eso extiende la clase JFrame e implementa la interface ActionListener.



La interfaz consta de tres botones; Enviar, Borrar y Descargar que son los encargados de realizar las peticiones del cliente.



El botón enviar, crea una carpeta en la carpeta del proyecto llamada CarpetaServidor, en esta dirección es en donde se guardarán todos los archivos y carpetas que mandemos a través del socket.

Cada vez que el botón enviar es presionado, se crea una la carpeta CarpetaServidor y una instancia de la clase CCliente, donde le pasamos como argumento al constructor la dirección de la carpeta anteriormente creada y mandamos a llamar a los métodos seleccionaArchivos y enviarArchivos, los cuales son los encargados de enviar los archivos seleccionados.

El botón borrar, borrara carpetas o archivos de la carpeta CarpetaServidor.

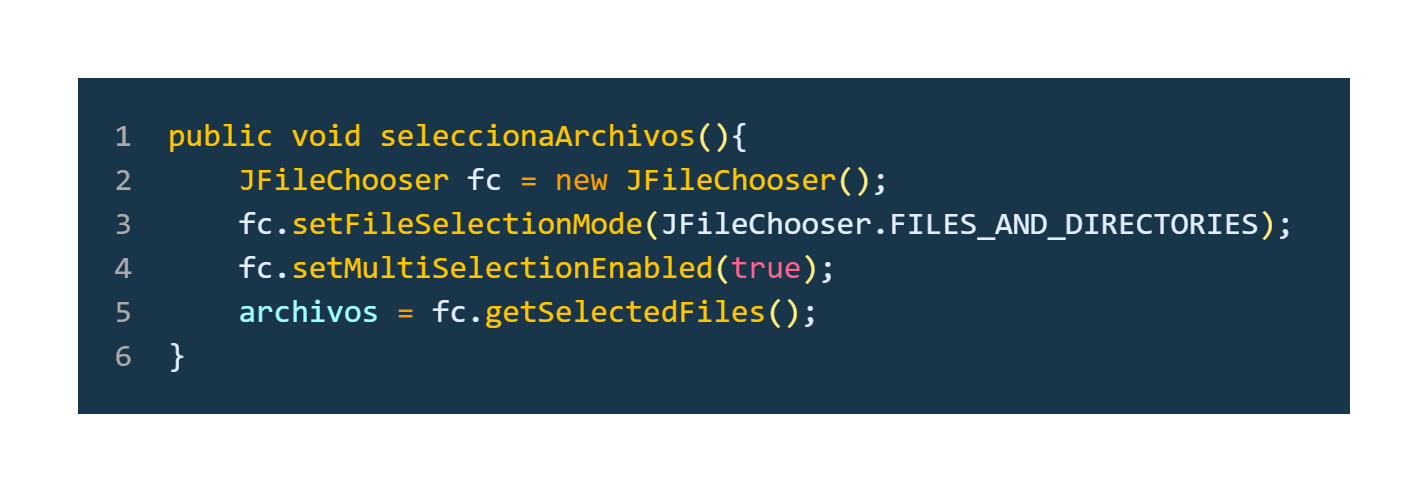
El botón descargar, hará una petición al servidor para que le mande los archivos seleccionados a través del servidor, estos archivos se almacenarán en una carpeta dentro de la carpeta del proyecto llamada Descargas.

Así como el botón enviar crea la carpeta CarpetaServidor, el botón descargar crea la carpeta de Descargas y crea una instancia de tipo CCliente y Servidor para seleccionar los archivos a descargar que le pasara a un método de la clase Servidor junto con la ruta de la carpeta de descargas.



## **Clase CCLiente.**

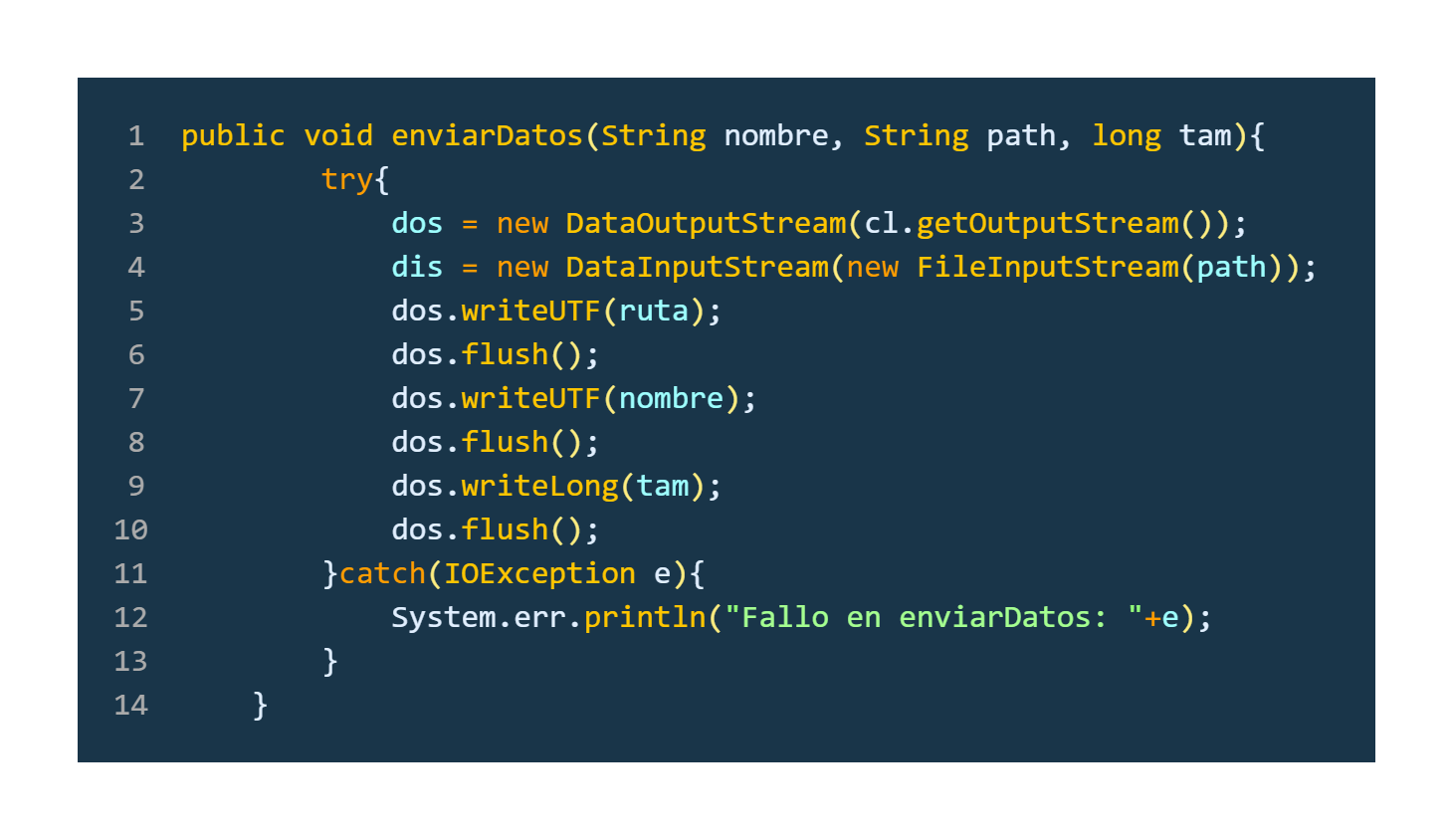
En esta clase hacemos uso diversas funciones para poder enviar los archivos y las carpetas a través del socket. En el método seleccionaArchivos se crea una instancia de tipo JFileChooser que nos permitirá seleccionar tanto archivos como carpetas de nuestra computadora y los almacena en un arreglo de tipo File llamado archivos.

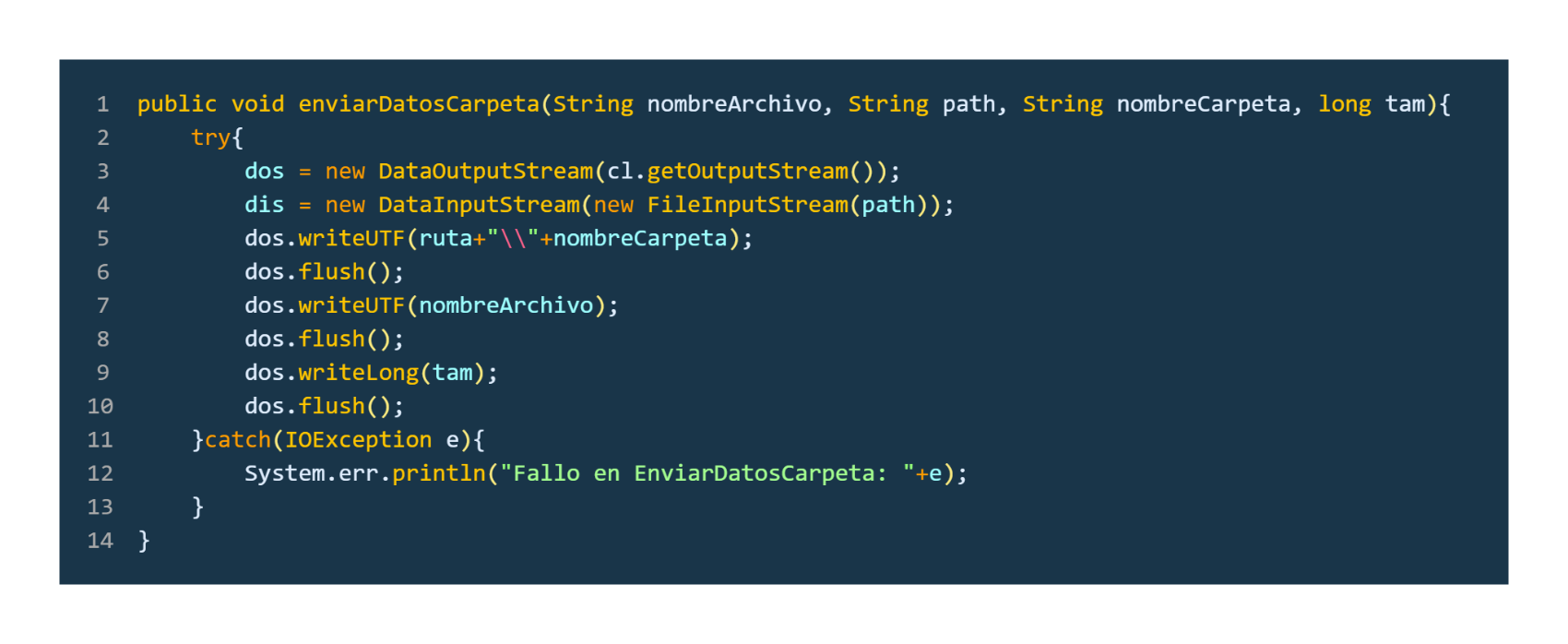


El método enviarArchivo verifica si el fichero seleccionado es una carpeta o un archivo y manda a llamar a los métodos enviarCarpeta y enviarDatos respectivamente.



enviarCarpeta es un método recursivo que manda a través del socket los archivos que se encuentran dentro de la carpeta seleccionada.

enviarDatos y enviarDatosCarpeta hacen uso de flujos de salida para enviar los datos a través del socket.



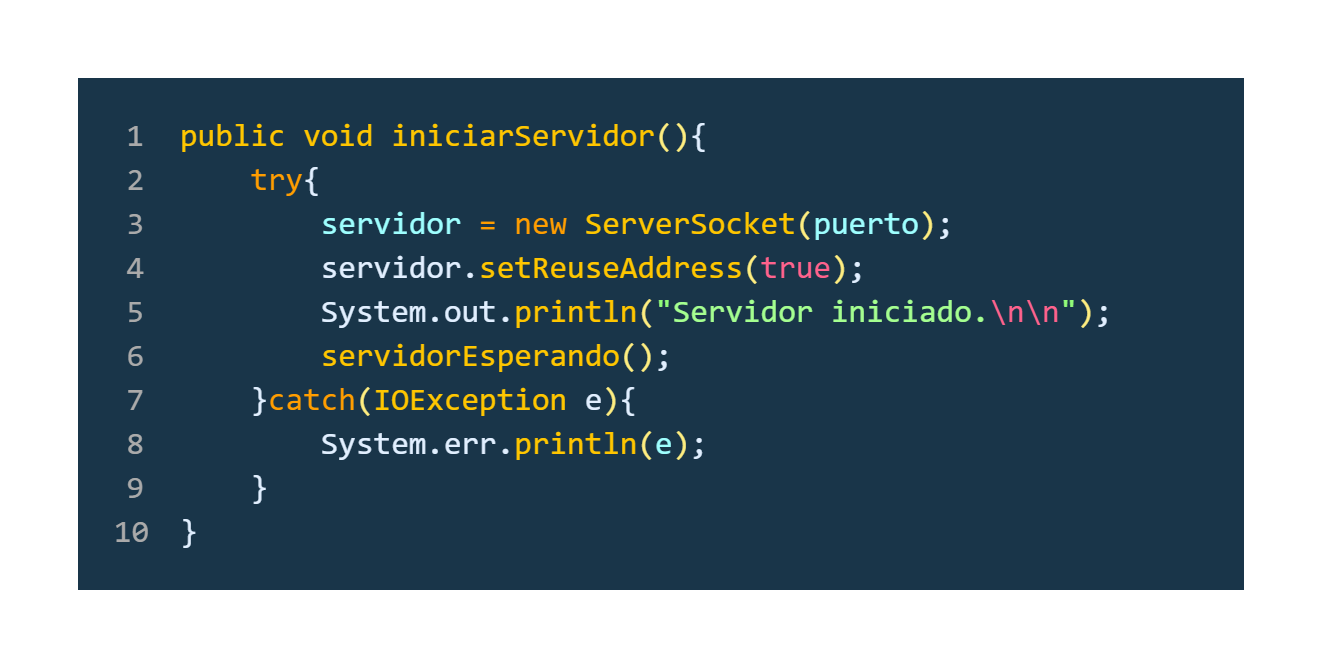
escribirFicheros es un método para verificar el progreso del envió de los datos y es el encargado de escribir los bytes de los datos que se enviarán al servidor.

Borrar es el método que verifica si el fichero a borrar es una carpeta o un archivo, si dicho fichero es una carpeta se manda a llamar al método borrarDirectorio que es un método recursivo que borrara todo lo que en esa carpeta se encuentra.



## **Clase Servidor.**

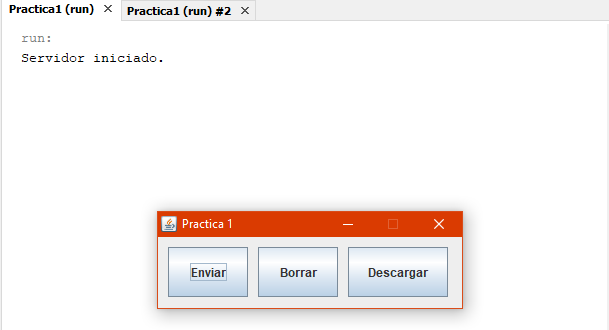
El método iniciarServidor es un método que hace uso de otra función de la misma clase llamada servidorEsperando para iniciar un ciclo infinito donde recibirá los flujos de entrada del socket y escribirá dichos datos en la carpeta correspondiente.





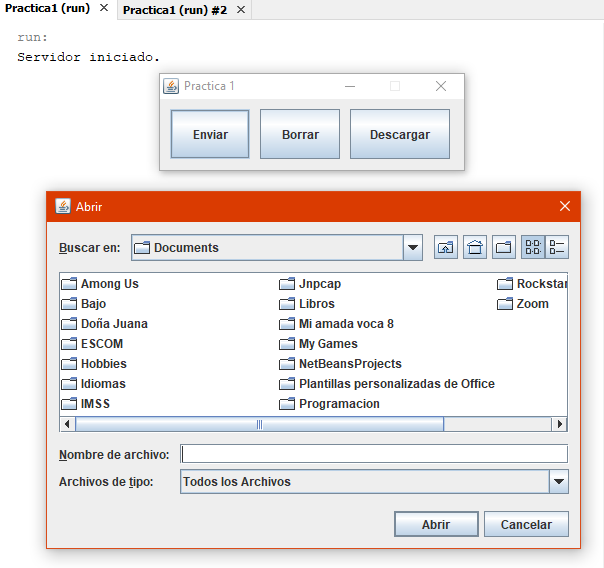
Finalmente, enviarArchivos es un método que recibe un arreglo de tipo File y una String que serán los archivos que el cliente seleccione para mandar a través del socket a la carpeta de Descargas.

# **Capturas de prueba.**

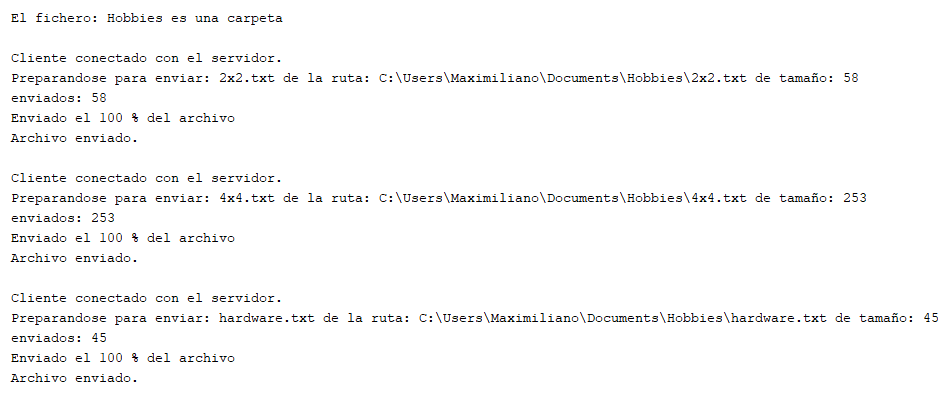


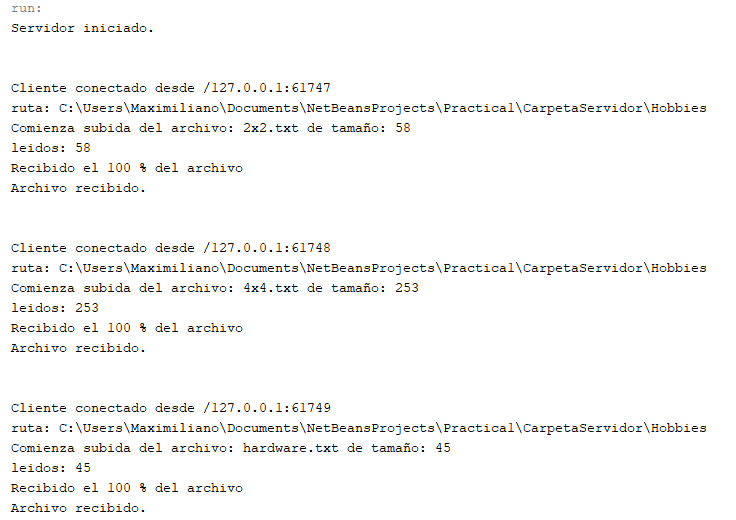
En la siguiente imagen se puede observar la interfaz y la consola de NetBeans ejecutando tanto el servidor como la interfaz del cliente.

A continuación, se muestra un JFileChoseer para elegir los archivos a enviar, borrar o descargar.

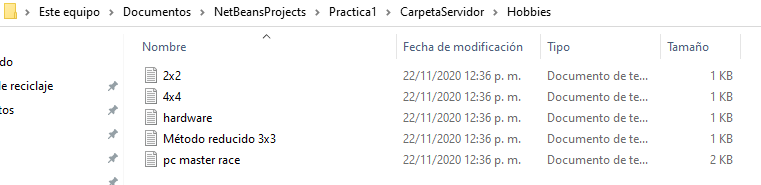


En la consola del lado del cliente se puede observar lo siguiente cuando mandamos una carpeta.



Del lado del servidor, esto se puede observar a continuación.

Verificando en el sistema de archivos podemos observar que efectivamente la carpeta se manda a través del socket.



# **Conclusiones.**

## **Cazares Martínez Maximiliano.**

En muchas organizaciones se requiere transferir grandes volúmenes de archivos a través de la red ya sean archivos de audio, video, fotografías, archivos ejecutables o de cualquier otro tipo de una manera confiable garantizando la entrega de los datos, que estos lleguen en orden y sin duplicados.

Los sockets de flujo bloqueantes pueden mandar archivos a través de la red garantizando el orden de los datos, es por eso que este tipo de sockets es usado en los servicios donde se emplea el modelo de cliente – servidor para el envió de archivos. Este tipo de sockets fue el empleado en la práctica para poder realizar un pequeño servicio de transferencia de archivos, algo parecido a los almacenamientos en la nube, como lo son: Google Drive, Mega, Dropbox, etc. Garantizando el correcto envió de los archivos.

## **Cisneros Rosales Christian Iván.**

Los sockets de flujo bloqueante permiten la comunicación entre procesos, en este caso un servidor y un cliente, permitiéndonos mandar archivos de manera ordenada a través de la red. Los sockets de flujo a diferencia de los sockets de datagrama, está garantizado que lleguen los datos y en orden, mientras que en los sockets de datagrama hay estar mandando mensajes constantemente para verificar que se recibieron los datos, además teniendo en cuenta que los sockets de datagrama no envían de manera ordenada un paquete. Lo más conveniente para este caso fue usar un socket de flujo bloqueante para transmitir una cantidad enorme de archivos conservando el orden de los directorios en caso de que se envíen carpetas.

# **Bibliografía.**

[1] Sockets, Universidad ICESI. [En línea]. Disponible en: https://www.emagister.com/uploads\_user\_home/Comunidad\_Emagister\_5811\_uno.pdf

[2] V. S. Roffé, La intefaz socket, UNNE. [En línea]. Disponible en: http://exa.unne.edu.ar/informatica/SO/Interfaz\_Socket.pdf

[3] Sockets, DAC UPC. [En línea]. Disponible en: http://www3.uji.es/~ochera/curso\_2002\_2003/e52/t\_sockets.pdf