|  |  |  |
| --- | --- | --- |
| **ipn** | **INSTITUTO POLITÉCNICO NACIONAL**  **ESCUELA SUPERIOR DE CÓMPUTO** |  |

**Aplicaciones para Comunicaciones en Red**

**“Manejador de Archivos”**

Abstracto

El abstracto es un reporte preciso de todo el reporte completo. Su función es indicar los contenidos del reporte para que el lector pueda ver si vale la pena leerlo completo o no.

**Por:**

**Cruz Sánchez Sofía Leticia**

**Romero Gamarra Joel Mauricio**

Profesor:

AXEL ERNESTO MORENO CERVANTES

Noviembre 2017

**Índice**

Contenido

[Introducción: 1](#_Toc476313175)

[Análisis Teórico: 1](#_Toc476313176)

[Software (librarias, paquetes, herramientas): 1](#_Toc476313177)

[Procedimiento: 1](#_Toc476313178)

[Resultados 1](#_Toc476313179)

[Discusión: 1](#_Toc476313180)

[Conclusiones: 2](#_Toc476313181)

[Referencias: 2](#_Toc476313182)

[Código 2](#_Toc476313183)

# Introducción:

En esta práctica se realiza un servidor y un cliente, donde el cliente es capaz de enviar al servidor archivos y/o carpetas por medio del lenguaje Java haciendo uso de objetos de tipo JFileChooser, utilizando un socket de flujo unicast transmitiendo de 1, 500 en 1, 500 bytes hasta llegar al final del archivo.

Cuando se hace el envío de una carpeta, debemos de obtener el nombre de esa carpeta y escribirlo como un nuevo directorio, posteriormente abrirla y hacer el copiado de todos los archivos y subcarpetas que haya dentro (para las subcarpetas, la lógica es la misma ya que debemos obtener el nombre y crear un directorio nuevo) con sus correspondientes archivos y directorios dentro.

Podemos darnos cuenta de que un manejador de archivos es algo bastante útil y muy utilizado, por ejemplo cuando queremos enviar algún archivo a la plataforma Moodle (algún archivo, ya que no nos permite enviar carpetas), o por ejemplo cuando queremos compartir con algún compañero un archivo por cualquiera de los servicios como Facebook, WhatsApp, Gmail, etc., debido a que podemos ser capaces de enviar cualquier tipo de archivo ya que la computadora lo ve como bytes y no le estamos haciendo ninguna modificación.

# Análisis Teórico:

Al tratarse de un manejador de archivos, debemos tener alguna forma de seleccionar alguno, para esto nos servirá el objeto JFileChooser de la librería swing, sin embargo, por default este objeto solo permite seleccionar archivos (no directorios) y nosotros necesitamos enviar cualquier tipo de archivo, por lo tanto, únicamente debemos activar una bandera con el método **setFileSelectionMode** y listo.

Sin embargo, debemos realizar la implementación de los métodos Drag y Drop, estos métodos nos permiten como su nombre lo indica, arrastrar algún archivo y/o directorio (porque ya podemos seleccionar ambos) hasta algún “target” y dejarlo ahí para que se empiece a encolar para ser enviado por el cliente hacia el servidor.1

La implementación de estos 2 métodos fue en cierta parte lo más complejo de la práctica debido a que en el equipo teníamos un desconocimiento total sobre ellos, sin embargo, nos dimos cuenta de que esta programación es orientada a eventos, donde los eventos son el click del mouse o más bien el drag, además, debemos de seleccionar nuestro target para poder aceptar los archivos y carpetas seleccionados por el usuario.

En cuanto a la parte de la implementación, cada que se tiene un archivo se copian los bytes tal cual, de 65, 535 en 65, 535 bytes, debido a que no podemos enviar un buffer más grande que eso, en el caso de encontrar una carpeta, primero debemos obtener el nombre del directorio y creamos el directorio en la dirección de destino, y esto lo hacemos recursivamente para que haga las instrucciones que se deben dependiendo si es un directorio o un archivo como cualquier otro, y no importa que sea ya que se abren como bytes sin importar si es una imagen, un video, un archivo de texto, un archivo de Word, etc.

# Software (librarías, paquetes, herramientas):

* Sublime Text 3
* NetBeans

# Resultados

En las Figuras siguientes se muestra el funcionamiento del programa así como la implementación del método Drag & Drop.



Figura 1. Interfaz inicial

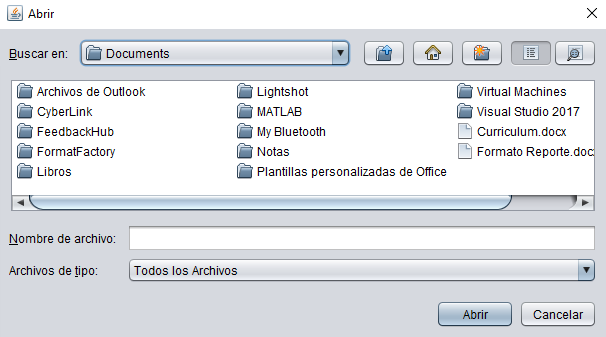


Figura 2. Cuadro para seleccionar archivos

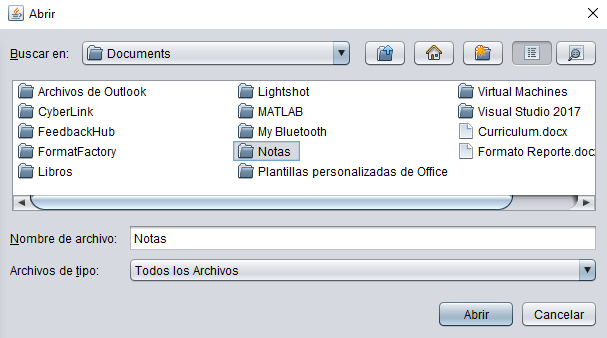


Figura 3. Carpeta seleccionada

Como se puede observar, nos deja seleccionar carpetas para poder enviarlas.

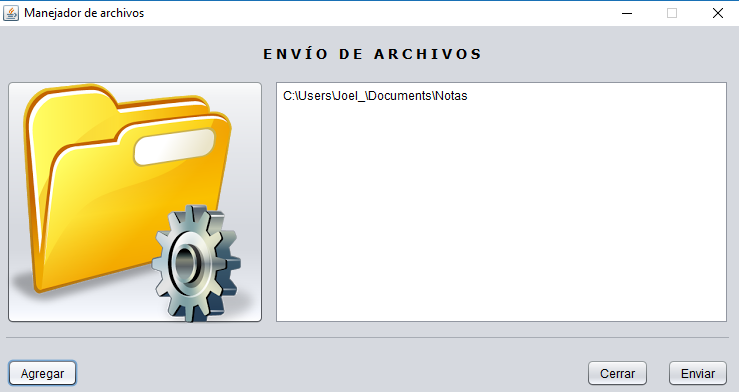


Figura 4. Interfaz encolando archivos

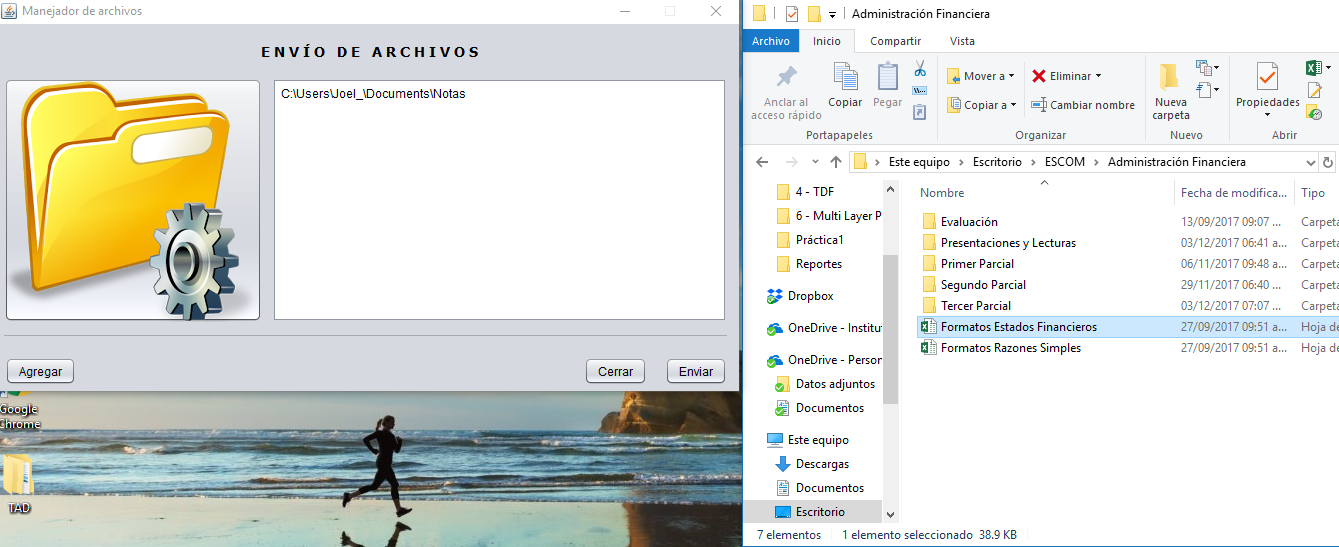


Figura 5. Drag & Drop

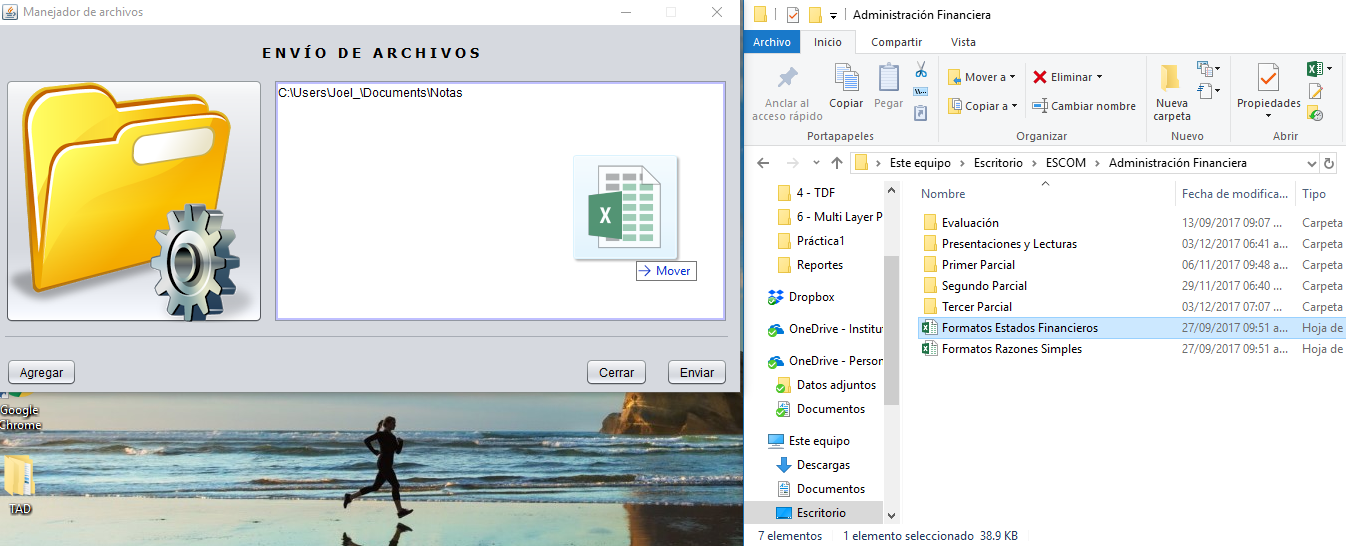


Figura 6. Archivo arrastrado



Figura 7. Archivos que enviar



Figura 8. Archivos enviados

# Discusión:

La sección de discusión tiene 2 objetivos principales:

* Interpretar y explicar los resultados del estudio.
* Explorar la importancia del estudio, encontrando, calificando y explorando la importancia teórica de los resultados.

La discusión es también un espacio en el reporte donde cualquier calificación o reservación que se tiene sobre la investigación debe ser mencionada,

# Conclusiones:

Este programa es muy interesante debido a que es algo con lo que tenemos contacto todos los días, por ejemplo, para hacer algún programa o reporte de práctica entre compañeros porque nos compartimos los archivos por medio de Facebook o alguna otra aplicación.

La parte de la comunicación fue bastante simple debido a que se explicó muy bien dentro del curso en el salón y se resolvieron todas las dudas por parte del profesor, sin duda, una práctica con un nivel de complejidad bastante aceptable y acorde a lo visto hasta el momento.

# Referencias:

**[1]** Java 2s, “Java Swing Tutorial – Java Swing Drag and Drop”, [Online]. Disponible en: <http://www.java2s.com/Tutorials/Java/Java_Swing/1550__Java_Swing_Drag_Drop.htm>.

# Código

**ArchivosCliente.java**

**import** java.io.**\***;

**import** java.net.**\***;

**public** **class** ArchivosCliente

{

**private** **DataOutputStream** buferSalida;

**private** **DataInputStream** buferEntrada;

**private** **Socket** cliente;

**private** **String** host = "127.0.0.1", nombreArch, ruta;

**private** int i = 0, puerto = 9876, porcentaje, n;

**private** long enviados, tamanioArchivo;

**public** ArchivosCliente (**File** [] archivo)

{

**try**

{

**while** (i < archivo.length)

{

nombreArch = (archivo [i]).getName ();

ruta = (archivo [i]).getAbsolutePath ();

tamanioArchivo = (archivo [i]).length ();

**if** ((archivo [i]).isFile ())

envia (archivo [i], nombreArch, ruta, tamanioArchivo, 'a', "");

**else**

envia (archivo [i], nombreArch, ruta, tamanioArchivo, 'd', nombreArch);

i ++;

}

**System**.out.println ("\n\nArchivo enviado correctamente.\n");

buferSalida.close ();

buferEntrada.close ();

cliente.close ();

}**catch** (**Exception** ex)

{

ex.printStackTrace ();

}

}

**public** void envia (**File** archivo, **String** nombreArch, **String** ruta, long tamanioArchivo, char tipo, **String** DirPadre)

{

**try**

{

cliente = **new** **Socket** (host, puerto);

buferSalida = **new** **DataOutputStream** (cliente.getOutputStream ());

buferSalida.writeChar (tipo);

buferSalida.flush ();

buferSalida.writeUTF (nombreArch);

buferSalida.flush ();

buferSalida.writeUTF (DirPadre);

buferSalida.flush ();

**if** (tipo == 'a') //Si se va a copiar un archivo

{

**System**.out.println ("\n\nInicia la transferencia del archivo " + nombreArch);

enviados = 0;

porcentaje = 0;

buferEntrada = **new** **DataInputStream** (**new** **FileInputStream** (ruta));

buferSalida.writeLong (tamanioArchivo);

buferSalida.flush ();

**while** (enviados < tamanioArchivo)

{

byte [] b = **new** byte [1500];

n = buferEntrada.read (b);

buferSalida.write (b, 0, n); //Escribimos desde el byte 0 hasta el byte n

buferSalida.flush ();

enviados += n;

porcentaje = (int)((enviados \* 100) / tamanioArchivo);

**System**.out.print ("\rEnviados: " + porcentaje + " %");

}

}**else**

{

**System**.out.println ("\n\nInicia la transferencia del directorio " + nombreArch);

**File** [] directorio = archivo.listFiles ();

int i = 0;

**while** (i < directorio.length)

{

nombreArch = (directorio [i]).getName ();

ruta = (directorio [i]).getAbsolutePath ();

tamanioArchivo = (directorio [i]).length ();

**if** ((directorio [i]).isFile ())

{

//System.out.println("ENVIANDO COMO RUTA "+ DirPadre);

envia (directorio [i], nombreArch, ruta, tamanioArchivo, 'a', DirPadre);

}**else**

{

//System.out.println("ENVIANDO COMO RUTA "+ DirPadre+"\\"+nombreArch);

envia (directorio [i], nombreArch, ruta, tamanioArchivo, 'd', DirPadre+"\\"+nombreArch);

}

i ++;

}

}

buferSalida.close ();

buferEntrada.close ();

cliente.close ();

}**catch** (**Exception** e)

{

e.printStackTrace ();

}

}

}

**ArchivosServidor.java**

**import** java.net.**\***;

**import** java.io.**\***;

**public** **class** ArchivosServidor

{

**private** **ServerSocket** servidor;

**private** **Socket** cliente;

**private** int puerto = 9876, n = 0, porcentaje = 0;

**private** **DataInputStream** dis;

**private** **DataOutputStream** dos;

**private** **String** nombre,DirPadre;

**private** long tam, r;

**private** char tipo;

**String** rutaS="C:\\Users\\enata\_000\\Documents\\SERVIDOR\\";

**String** carpeta="";

**public** ArchivosServidor ()

{

**try**

{

servidor = **new** **ServerSocket** (puerto);

servidor.setReuseAddress(**true**);

**System**.out.println ("Servicio iniciado... esperando clientes.\n\n");

**for** (;;)

{

cliente = servidor.accept ();

**System**.out.println ("Cliente conectado desde -> " + cliente.getInetAddress () + ":" + cliente.getPort());

recibe (rutaS);

}

}**catch** (**Exception** ex)

{

ex.printStackTrace ();

}

}

**public** void recibe (**String** ruta)

{

r = 0;

**try**

{

dis = **new** **DataInputStream** (cliente.getInputStream ());

tipo = dis.readChar ();

nombre = dis.readUTF ();

DirPadre = dis.readUTF ();

**if** (tipo == 'a')

{

tam = dis.readLong ();

**if**(DirPadre=="")

{

**System**.out.println ("Recibiendo archivo " + nombre +" RUTA DESTINO "+ruta+DirPadre+nombre);

dos = **new** **DataOutputStream** (**new** **FileOutputStream** (ruta+DirPadre+nombre));

}**else**

{

**System**.out.println ("Recibiendo archivo " + nombre +" RUTA DESTINO "+ruta+DirPadre+"\\"+nombre);

dos = **new** **DataOutputStream** (**new** **FileOutputStream** (ruta+DirPadre+"\\"+nombre));

}

**while** (r < tam)

{

byte [] b = **new** byte [1500];

n = dis.read (b);

dos.write (b, 0, n);

dos.flush ();

r += n;

porcentaje = (int) ((r \* 100) / tam);

**System**.out.print ("\rRecibido el " + porcentaje + "%");

}

**System**.out.println ("\nArchivo recibido...\n\n");

}**else**

{

**System**.out.println ("Recibiendo directorio " + nombre + " RUTA DESTINO "+ruta+DirPadre);

boolean bol = **new** **File** (ruta+DirPadre+"\\").mkdir ();

**System**.out.println ("Directorio creado...\n\n");

}

dos.close ();

dis.close ();

cliente.close ();

}**catch** (**Exception** e)

{

e.printStackTrace ();

}

}

**public** **static** void main(**String**[] args)

{

**new** ArchivosServidor ();

}

}