MARTÍNEZ ALVARADO BRYAN ALEXIS

Socket de Envío

```
package sockets;
import java.io.*;
import java.net.Socket;
public class SocketEnvio {
   private final String host;
   private final int port;
   private final int tam_buffer;
   private final boolean algoritmo_nigle;
   public SocketEnvio(String host, int port, int tam_buffer, boolean algoritmo_nigle) {
      this.host = host;
      this.port = port;
      this.tam_buffer = tam_buffer;
      this.algoritmo_nigle = algoritmo_nigle;
   }
   public void enviarArchivo(File file, String destino) throws IOException {
      Socket cl = new Socket(host, port);
      String nombre = file.getName();
      cl.setTcpNoDelay(algoritmo_nigle);
      long tam = file.length();
      long enviados = 0;
      int porcentaje;
      int n;
      String ruta = file.getAbsolutePath();
      System.out.println("Conexion establecida");
      DataOutputStream dos = new DataOutputStream(cl.getOutputStream());
      DataInputStream dis = new DataInputStream(new FileInputStream(ruta));
      // tipo de cliente
      dos.writeInt(1); // cliente tipo 1
```

```
        dos.flush();
        // tam de buffer
        dos.writeInt(tam_buffer);
        dos.flush();
        // bandera de niggle
        dos.writeBoolean(algoritmo_nigle);
        dos.flush();
        // Usamos destino para ir almacenando la ruta del archivo/carpeta
        dos.writeUTF(destino);
        dos.flush();
        // Despues mandamos el nombre del archivo
        dos.writeUTF(nombre);
        dos.flush();
        // Y finalmente su tamaño
        dos.writeLong(tam);
        dos.flush();

        System.out.format("Enviando el archivo: %s...\n", nombre);
        System.out.format("Que esta en la ruta: %s\n", ruta);

        while (enviados < tam) {
           byte[] b = new byte[tam_buffer];
           n = dis.read(b);
           dos.write(b, 0, n);
           dos.flush();
           enviados = enviados + n;
           porcentaje = (int) (enviados * 100 / tam);
           System.out.print("\rSe ha transmitido el: " + porcentaje + "% ...\n");
        }
```

```
        System.out.println("Archivo enviado");
        cl.close();
        dos.close();
        dis.close();
    }
    // el parametro destino nos permite guardar la ubicacion del archivo
    public void enviarCarpetas(File carpeta, String destino) throws IOException {
        System.out.format("Carpeta %s con los archivos:\n", carpeta.getName());
        if (destino.equals("")) destino = carpeta.getName(); // evita que se cree en c:\\
        else destino = destino + "\\" + carpeta.getName(); // concatenar la ruta de los archivos
        for (File file : carpeta.listFiles()) {
            if (file.isDirectory()) enviarCarpetas(file, destino);
            else enviarArchivo(file, destino);
        }
    }
}
```

Cliente

```
package sockets;
import java.io.*;
import java.net.Socket;
public class Cliente {
    public static void main(String[] args) {
        // This is just a client
        StringBuilder LOCAL_PATH = new StringBuilder();
        LOCAL_PATH.append(".");
        LOCAL_PATH.append(File.separator);
        LOCAL_PATH.append("terminal_files");
        LOCAL_PATH.append(File.separator);
```

```
// Server's credentials to connect
String HOST = "201.114.240.145";
Integer PORT = 9999;
try {
   Socket cliente = new Socket(HOST, PORT);
   System.out.println("Cliente: " + cliente.getLocalSocketAddress() + " conectado");
   DataOutputStream dos = new DataOutputStream(cliente.getOutputStream());
   DataInputStream dis = new DataInputStream(cliente.getInputStream());
   dos.writeInt(2);
   dos.flush();
   Integer tam_buffer = dis.readInt();
   Boolean algoritmo_nigle = dis.readBoolean();
   cliente.setTcpNoDelay(algoritmo_nigle);
   Integer numero_archivos = 0;
   numero_archivos= dis.readInt();
   Integer contador = 0;

   while(contador < numero_archivos){
      String nombre_Archivo = dis.readUTF();
      Long longitud_archivo = dis.readLong();
      System.out.println(nombre_Archivo + ":" + longitud_archivo);
      String ruta = LOCAL_PATH + nombre_Archivo;
      System.out.println(ruta);
      DataOutputStream dos_archivos = new DataOutputStream(new FileOutputStream(ruta));
      long recibidos = 0;
      int porcentaje;
      int n;
      while (recibidos < longitud_archivo) {
```

```
                byte[] buffer = new byte[tam_buffer];
                n = dis.read(buffer);
                dos_archivos.write(buffer, 0, n);
                dos_archivos.flush();
                recibidos += n;
                porcentaje = (int) (recibidos * 100 / longitud_archivo);
                System.out.println("\rSe ha recibido el: " + porcentaje + "% ...");
              }
              dos_archivos.close();
              contador += 1;
            }
            dis.close();
            dos.close();
            cliente.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```

Servidor

```
package sockets;
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
public class Servidor {
    private static final int PUERTO = 9999;
    public static void main(String[] args) throws IOException, InterruptedException {
        StringBuilder LOCAL_PATH = new StringBuilder()
```

```
        Integer tam_buffer = 1024; // deafult
        Boolean algoritmo_nigle = Boolean.FALSE; // DEFAULT

        LOCAL_PATH.append(".");
        LOCAL_PATH.append(File.separator);
        LOCAL_PATH.append("middle_files");
        LOCAL_PATH.append(File.separator);

        ServerSocket s = new ServerSocket(PUERTO);
        s.setReuseAddress(true);
        System.out.println("Servicio iniciado...");

        while (true ) {
           System.out.println("Eperando conexion...");
           Socket cl = s.accept();

           System.out.format("Cliente conectado desde: %s:%s\n", cl.getInetAddress(), cl.getPort());
           DataInputStream dis = new DataInputStream(cl.getInputStream());
           DataOutputStream dos = new DataOutputStream(cl.getOutputStream());

           Integer tipo = dis.readInt();

           if(tipo == 1){ // cliente tipo #1 -> incial
              Integer local_buffer = dis.readInt();
              Boolean local_nigle = dis.readBoolean();

              tam_buffer = local_buffer;
              algoritmo_nigle = local_nigle;
```

```
            cl.setTcpNoDelay(local_nigle);

            String ruta = dis.readUTF();
            String nombre = dis.readUTF();

            // recibir archivo
            escribirArchivo(LOCAL_PATH + nombre, dis, local_buffer);

            System.out.println("¡Archivo recibido!\n");
        }else{ // cliente tipo #2 -> final
            System.out.println("Cliente final conectado");

            File carpeta = new File(LOCAL_PATH.toString());
            String[] listado = carpeta.list();
            if (listado == null || listado.length == 0) {
                System.out.println("No hay elementos dentro de la carpeta actual");
            }
            else { // el servidor tiene archivos
                cl.setTcpNoDelay(algoritmo_nigle);

                Integer numero_archivos = listado.length;

                dos.writeInt(tam_buffer);
                dos.flush()
                dos.writeBoolean(algoritmo_nigle);
                dos.flush();
                dos.writeInt(numero_archivos);
                dos.flush()
                for (int i=0; i< listado.length; i++) {
```

```
            File archivo = new File(LOCAL_PATH + listado[i]);
            String nombre_archivo = archivo.getName();
            String ruta_archivo = archivo.getAbsolutePath();
            Long longitud_archivo = archivo.length();
            dos.writeUTF(nombre_archivo);
            dos.flush();

            dos.writeLong(longitud_archivo);
            dos.flush();
            // Vas a escribir al mismo dos
            long enviados = 0;
            int porcentaje;
            int n;
            String ruta = archivo.getAbsolutePath();
            DataInputStream dis_archivo = new DataInputStream(new FileInputStream(ruta));
            while (enviados < longitud_archivo) {
               byte[] b = new byte[1500];
               n = dis_archivo.read(b);
               dos.write(b, 0, n);
               dos.flush();
               enviados = enviados + n;
               porcentaje = (int) (enviados * 100 / longitud_archivo);
               System.out.println("\rSe ha transmitido el: " + porcentaje + "% ...");
            }
            dis_archivo.close();
            Thread.sleep(200);
         }
      }
   }
```

```
            dis.close();
            cl.close();
        }

    }
    private static void escribirArchivo(String nombre, DataInputStream dis, Integer tam_buffer) {
        System.out.format("Escribiendo el archivo: %s\n", nombre);
        try {
            DataOutputStream dos = new DataOutputStream(new FileOutputStream(nombre));
            long recibidos = 0;
            long tam = dis.readLong();
            int n;
            int porcentaje;
            while (recibidos < tam) {
                byte[] buffer = new byte[tam_buffer];
                n = dis.read(buffer);
                dos.write(buffer, 0, n);
                dos.flush();
                recibidos += n;
                porcentaje = (int) (recibidos * 100 / tam);
                System.out.println("\rSe ha recibido el: " + porcentaje + "% ...");
            }
            dos.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

ListTransferHandler

MARTÍNEZ ALVARADO BRYAN ALEXIS

```
package view;

import sockets.SocketEnvio;
import javax.swing.*;
import java.awt.datatransfer.DataFlavor;
import java.awt.datatransfer.UnsupportedFlavorException;
import java.io.File;
import java.io.IOException;
import java.util.List;
public class ListTransferHandler extends TransferHandler {
   private final SocketEnvio socketEnvio; // Socket que usamos en la transferencia
   private int action;
   ListTransferHandler(int action, SocketEnvio socketEnvio) {
      this.action = action;
      this.socketEnvio = socketEnvio;
   }
   public boolean canImport(TransferHandler.TransferSupport support) {
      // Con esto solo se podran arrastar elementos
      if (!support.isDrop()) {
         return false;
      }
      // Para solo poder arrastrar archivos/carpetas
      if (!support.isDataFlavorSupported(DataFlavor.javaFileListFlavor)) {
         System.out.println("NO ES ARCHIVO NI CARPETA");
         return false;
      }

      boolean actionSupported = (action & support.getSourceDropActions()) == action;
      if (actionSupported) {
```

```
            support.setDropAction(action);
            return true;
        }
        return false;
    }

    public boolean importData(TransferHandler.TransferSupport support) {
        // Si no se puede importar el archivo se termina la accion
        if (!canImport(support)) {
            System.out.println("No se soporta la informacion");
            return false;
        }
        // Obtenemos el componente que utiliza drag and drop
        JList jList = (JList) support.getComponent();
        DefaultListModel model = new DefaultListModel();
        jList.setModel(model);
        List<File> archivos = null;
        try {
            // Obtenemos los elmentos arrastrados
            archivos = (List<File>) support.getTransferable()
                    .getTransferData(DataFlavor.javaFileListFlavor);
            // Los mandamos a su respectivo metodo para ser enviados
            for (File file : archivos) {
                model.addElement(file.getName());
                model.removeElement(file.getName());
                // Manda las carpetas recursivamente
                if (file.isDirectory()) socketEnvio.enviarCarpetas(file, "");
                else socketEnvio.enviarArchivo(file, ""); // Manda un solo archivo
                model.removeElement(file.getName());
```

```
                model.addElement(file.getName() + "(enviado)");


            }
        } catch (UnsupportedFlavorException | IOException e) {
            e.printStackTrace();
        }
        return true;
    }
}
```