Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria

Campus Zacatecas

Practica 2: Construcción de un servicio de transferencia de archivos utilizando sockets orientados a conexión

Aplicaciones para comunicaciones de redes

Sandra Mireya Monreal Mendoza

Juan Antonio Ovalle Patiño

1 / Febrero / 2019

Contenido

Introducción

Hoy en día es muy necesario compartir archivos entre dos computadoras o más y esto pasa en cualquier nivel y área (primaria, secundaria, prepa, universidad, papelería, en la oficina, en la tienda, etc.). Aunque hay una forma de compartir archivos, mediante una memoria USB, no todos tienen una y por eso un programa que nos permita compartir archivos puede ser una gran ayuda.

Objetivo

Crear una aplicación que permita la transferencia de archivos de texto mediante la implementación de los sockets de flujo.

Desarrollo

Compartir archivos desde una computadora a otra (inclusive un teléfono inteligente) es un proceso que realizamos de manera cotidiana, ya sea mediante correo electrónico, red social, servicio de mensajería instantánea o chat, etc. Dicho proceso e importante cuando se desea compartir información con otras personas.

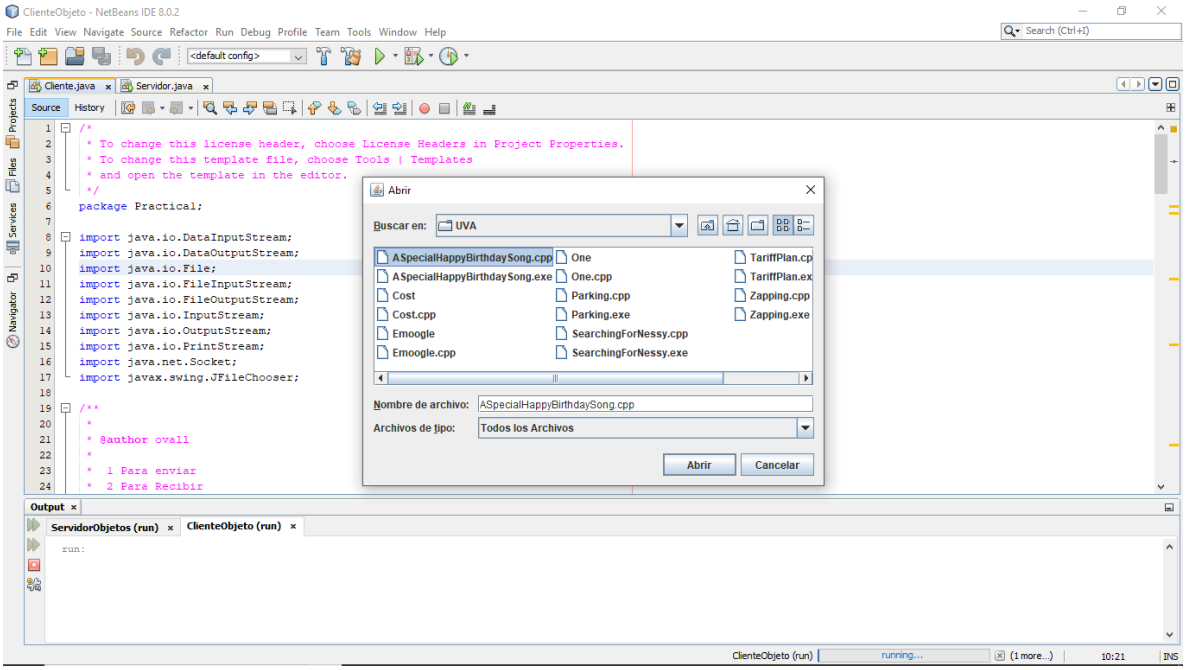
El desarrollo de una aplicación de intercambio de archivos puede ser especialmente útil cuando es necesario que varios funcionarios de diferentes despachos compartan información de forma eficiente y con la seguridad que será dentro de la misma red de la empresa u organización.

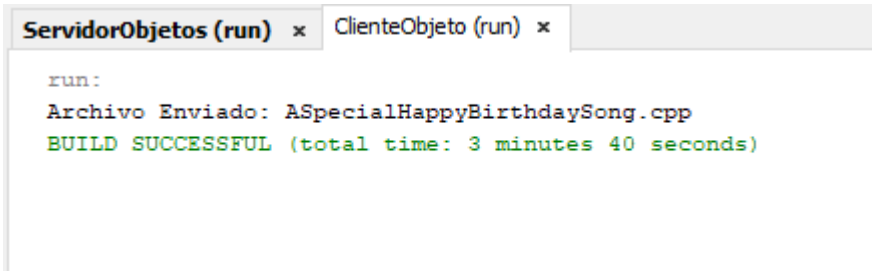
Para esto, se debe realizar una aplicación que permita:

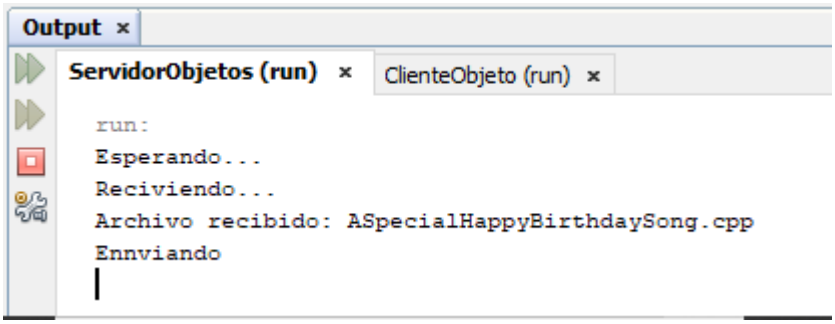
- Compartir archivos entre al menos dos equipos.
- Quien recibe archivos debe mantener el mismo nombre del archivo enviado.
- Crear una carpeta para los archivos compartidos.

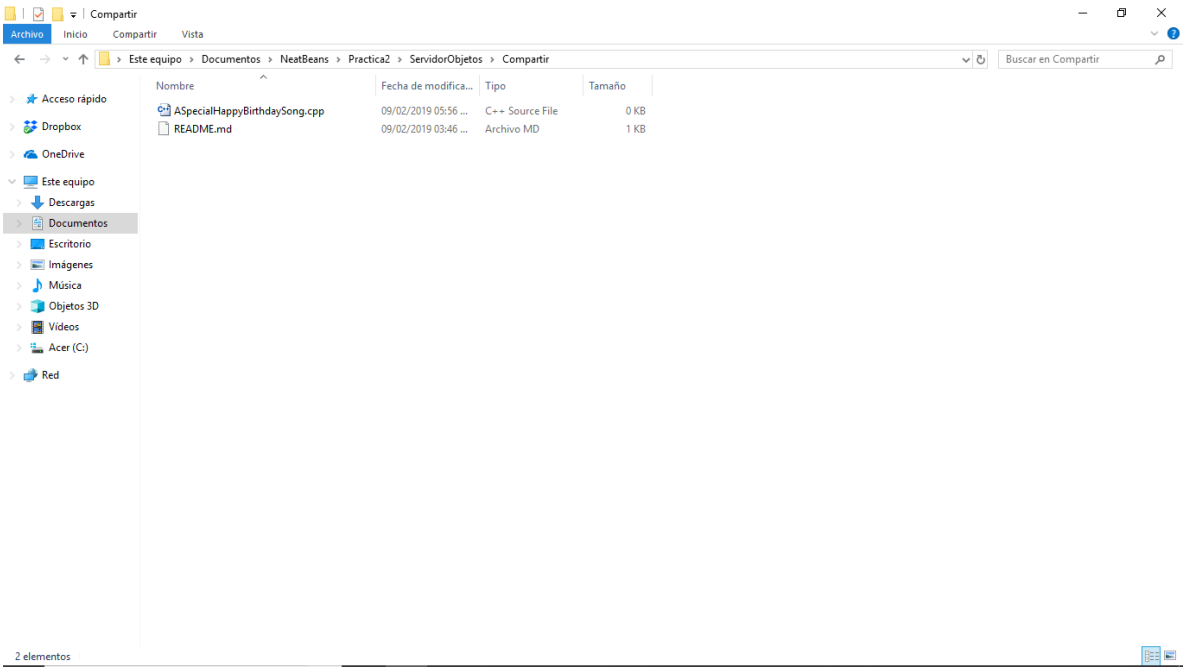
- Preguntar sobre las opciones: enviar archivos, recibir y mostrar un listado de los archivos compartidos.

Resultados









Codigo:

Cliente

```java
public class Cliente {
    private static String ip = "127.0.0.1"; //"192.168.0.18";
    private static String decisionServidor;
    private static String nombreArchivo;
    private static int puerto = 4000;
    private static Socket cliente;
    private static DataOutputStream salida;
    private static DataInputStream entrada;
    private static PrintStream envio;
    private static InputStream llegada;
    private static FileInputStream origen;
    private static FileOutputStream destino;



    public static File openFile(){
        JFileChooser selector = new JFileChooser();
        selector.setFileSelectionMode(JFileChooser.FILES_ONLY);
        int res = selector.showOpenDialog(null);
        if (res == 1 )
            return null;
        File archivo = selector.getSelectedFile();
        return selector.getSelectedFile();
    }

    public static void mandarDecision(String decision){
        try {
            // Creamos el flujo de salida
            salida = new DataOutputStream(cliente.getOutputStream());
            salida.writeUTF(decision);
        } catch (Exception e) {
```

```java
public static void mandarDecision(String decision){
    try {
        // Creamos el flujo de salida
        salida = new DataOutputStream(cliente.getOutputStream());
        salida.writeUTF(decision);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public static void recivirDecision(){
    try {
        System.out.println("Esperando...");
        cliente = new Socket(ip, puerto);
        entrada = new DataInputStream(cliente.getInputStream());
        decisionServidor = entrada.readUTF();
        System.out.println(decisionServidor);
    } catch (Exception e) {
    }
}

public static void enviarA(){
    try {
        OutputStream os =  cliente.getOutputStream();
            envio = new PrintStream(os);
            //Seleccionamos el archivo
            File archivo = openFile();
            // Enviamos el nombre del archivo
            salida.writeUTF(archivo.getName());
            // Creamos flujo de entrada para realizar la lectura del archivo en bytes
            origen = new FileInputStream(archivo.getAbsoluteFile());
            // Creamos un array de tipo byte
```

```
            byte[] buffer = new byte[1024];
            int len = 0;
            while((len = origen.read(buffer)) > 0)
                envio.write(buffer, 0, len);
            System.out.println("Archivo Enviado: "+archivo.getName());
        } catch (Exception e) {
        }
    }

    public static void recivir(){
        try {
            nombreArchivo = entrada.readUTF();
            llegada = cliente.getInputStream();
            //Idicar donde se guarda el archivo
            destino = new FileOutputStream("Compartir/"+nombreArchivo);
            //Creamos el array de bytes para leer los datos del archivo
            byte[] buffer = new byte[1024];
            int len = 0;
            while((len = llegada.read(buffer)) >0)
                destino.write(buffer);
            destino.close();
            llegada.close();
            System.out.println("Archivo recibido: "+nombreArchivo);
        } catch (Exception e) {
        }
    }

    public static void main(String[] args){
        try {
            String decision = "1";
            // Creamos el Socket con la direccion y elpuerto de comunicacion
            cliente = new Socket(ip, puerto);
            //Mandos al servidor la opcion que desa el cliente
            mandarDecision(decision);
            if(decision.equals("1")){
                enviarA();
            }
              recivirDecision();
            //if(decisionServidor.equals("1"))
                    recivir();
            cliente.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Servidor

```
public class Servidor {
    private static int puerto = 4000;
    private static String decisionCliente;
    private static String decision;
    private static String nombreArchivo;
    private static ServerSocket servidor;
    private static Socket cliente;
    private static DataInputStream entrada;
    private static DataOutput salida;
    private static InputStream llegada;
    private static FileOutputStream destino;

    public static void mandarDecision(String decision){
        System.out.println("Ennviando");
        try {
            cliente = servidor.accept();
            //Creamos el flujo de salida
            salida = new DataOutputStream(cliente.getOutputStream());
            salida.writeUTF(decision);
        } catch (Exception e) {
        }
    }

    public static void recivirDecision(){
        try {
            // Creamos flujo de entrada para leer los datos que envia el cliente
            entrada = new DataInputStream(cliente.getInputStream());
            // Obtenemos La decisión del cliente
            decisionCliente = entrada.readUTF();
//              System.out.println(decisionCliente);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```java
public static void recivir(){
    System.out.println("Reciviendo...");
    try {
        nombreArchivo = entrada.readUTF();
        llegada = cliente.getInputStream();
        //Idicar donde se guarda el archivo
        destino = new FileOutputStream("Compartir/"+nombreArchivo);
        // Creamos el array de bytes para leer los datos del archivo
        byte[] buffer = new byte[1024];
        int len = 0;
        while ((len = llegada.read(buffer)) > 0)
            destino.write(buffer);

          destino.close();
          llegada.close();
        System.out.println("Archivo recibido: "+nombreArchivo);

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```java
public static void main(String aegs[]){
    try {
        System.out.println("Esperando...");
        servidor = new ServerSocket(puerto);
        cliente = servidor.accept();
        recivirDecision();
        if(decisionCliente.equals("1")){
            recivir();
        }
        decision = "1";
        mandarDecision(decision);
        destino.close();
        llegada.close();
        cliente.close();
    } catch (Exception e) {
    }
}
```

NOTA

No logramos que el servidor manda un archivo después de haber obtenido el archivo del cliente.

Solo se tiene que el archivo envíe cualquier archivo y lo guarda en la carpeta.

Conclusiones

Los sockets de flujo no pueden ayudar a hacer un programa para poder compartir cualquier archivo que deseemos y así nos podemos ahorrar la molestia de estar conectando y desconectando las memorias USB, per en nuestro caso no logramos hacer que el servidor enviara tal archivo.