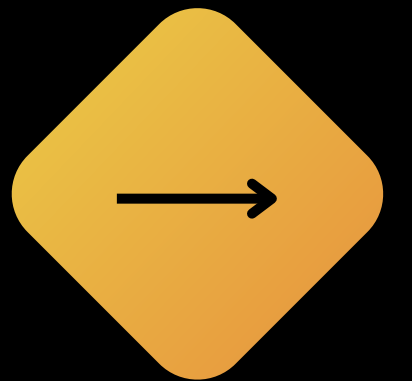


APLICATIVO CLIENTE-SERVIDOR

Presentado por:

David Santiago Reyes Lasso

Luis Fernando Oviedo Dominguez



ELEMENTOS PARA LA CONSTRUCCION DEL APLICATIVO



JAVA

Lenguaje de programación utilizado para el desarrollo del aplicativo.



NETBEANS

Entorno de desarrollo integrado utilizado para el desarrollo del aplicativo.



VIRTUALBOX

Software de virtualización en el cual se instaló Windows con el objetivo de simular la transferencia de archivos.

CONSTRUCCION DEL APLICATIVO

Direccion IP

Servidor

Hilo 1  Subproceso 1

Cliente

Hilo 2  Subproceso 2

Animacion

Hilo 3  Subproceso 3

Al tratar cada funcion como subprocesso se resuelven problemas como:

- Mayor tiempo de respuesta
- Interrupciones por parte de otras funciones
- Ejecucion de procesos al mismo tiempo

Principal

Cliente

Servidor

Clase



CLASE SERVIDOR

java.io.BufferedInputStream
java.io.File
java.io.FileInputStream
java.io.OutputStream
java.text.DecimalFormat

Indispensables para el
tratamiento de archivos

java.net.ServerSocket
java.net.Socket

Invoca libreria Socket para efectuar la
conexion entre dos equipos por medio
de una conexion TCP

java.io.IOException

Control de excepciones

```
int puerto;  
String archivo;
```

```
FileInputStream archEntrada = null;  
BufferedInputStream bufferEntrada = null;  
OutputStream flujoSalida = null;
```

```
ServerSocket servidorSocket = null;  
Socket socket = null;
```

Inicializacion de variables
tanto para el manejo de
archivos como para la
utilizacion de Sockets

CLASE SERVIDOR

```
servidorSocket = new ServerSocket(this.puerto);
```

El servidor recibe el puerto asignado por el usuario y espera respuesta del cliente

```
socket = servidorSocket.accept();
```

Verifica que la direccion IP y el puerto ingresado por el cliente corresponda con la informacion del servidor y acepta la solicitud entrante al socket

```
File archivo = new File(this.archivo);
```

Inicia el proceso de enviar el archivo al socket

```
bufferEntrada = new BufferedInputStream(archEntrada);  
bufferEntrada.read(byteArray, 0, byteArray.length);
```

Copia el archivo en un arreglo de Bytes

```
flujoSalida = socket.getOutputStream();  
flujoSalida.write(byteArray, 0, byteArray.length);  
flujoSalida.flush();
```

Copia el arreglo de Bytes al socket (archivo se envia a traves del socket)

```
bufferEntrada.close();  
socket.close();  
servidorSocket.close();
```

Cierra almacenamiento de buffer y socket

CLASE CLIENTE

java.io.BufferedOutputStream
java.io.FileOutputStream
java.io.InputStream
java.text.DecimalFormat

Indispensables para el
tratamiento de archivos

java.net.Socket

Invoca libreria Socket para efectuar la
conexion entre dos equipos por medio
de una conexion TCP

java.io.IOException

Control de excepciones

```
int port;  
String file;
```

```
FileInputStream archEntrada = null;  
BufferedInputStream bufferEntrada = null;  
OutputStream flujoSalida = null;
```

```
Socket socket = null;
```

Inicializacion de variables
tanto para el manejo de
archivos como para la
utilizacion de Sockets

CLASE CLIENTE

```
socket = new Socket(this.ip, this.port); ----->
```

El cliente recibe el puerto asignado por el usuario y la IP del Servidor, parametros indispensables para efectuar la conexión.

```
fileOutputStream = new FileOutputStream(this.file); ----->
```

Inicia el proceso de recibir el archivo

```
bufferedOutputStream = new BufferedOutputStream(fileOutputStream);  
bytesRead = inputStream.read(byteArray, 0, byteArray.length); ----->
```

Copia el archivo en un arreglo de Bytes

```
bufferedOutputStream.write(byteArray, 0, bytesRead);  
bufferedOutputStream.flush(); ----->
```

Copia el arreglo de Bytes al socket

```
fileOutputStream.close();  
bufferedOutputStream.close(); ----->  
socket.close();
```

Cierra almacenamiento de buffer y socket

CLASE VENTANAP

Librerías

```
import java.awt.Color;
import java.io.File;
import javax.swing.JFileChooser;
import static java.lang.Integer.parseInt;
import javax.swing.JFrame;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.Toolkit;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;
```

Creación de Hilos

```
this.animacion = new Thread(this);
```

} Hilo para manejar la animación del sistema

Establecer Hilos e Iniciarlos

```
public Servidor s;
public Cliente r;
s.start();
r.start();
```

} Se establecen los hilos, Servidor y Cliente respectivamente.

} Se inician los hilos, Servidor y Cliente respectivamente

CLASE VENTANAP

`public void run()`



Método debido a que establece el comportamiento de la animación sobre el panel

`ArchivoServidor.setLocation(x, y);`
`ArchivoCliente.setLocation(x, z);`



permite posicionar la animación con en las coordenadas establecidas para continuar con el desplazamiento

`animacion.stop();`



El hilo animación para cuando llega al limite de las coordenadas establecidas

CLASE DIRECCION IP

```
java.net.InetAddress  
java.net.UnknownHostException
```

Libreria InetAddress obtiene la direccion IP local de la maquina

```
public class DireccionIp {  
    String ResIp;
```

```
    String obtenerIP ( ) throws UnknownHostException {  
        InetAddress ip = InetAddress.getLocalHost();  
        return ip.getHostAddress();  
    }
```

Crea una funcion para obtener la direccion IP del ordenador con el metodo getHostAddress()

```
    public String DirIp() {  
        try {  
            DireccionIp checkIP = new DireccionIp();  
            ResIp= checkIP.obtenerIP();
```

Se crea una funcion para utilizar la direccion IP en otras clases y se valida su existencia

```
        } catch (UnknownHostException ex) {  
            System.err.println(ex);  
        }
```

```
        return ResIp;
```

Retorna la direccion IP local de la maquina

```
    }
```



Gracias