SOCKETS

1- MANDAR MENSAJE DE APAGAR, ENCENDER O REINICIAR:

```
class WorkerSocket extends Thread{

    private Socket client;

    public WorkerSocket(Socket client) {
        this.client=client;
    }

    @Override
    public void run() {
        try(DataInputStream dis=new DataInputStream(client.getInputStream());){
        String mensaje = null;
        do {
            mensaje=dis.readUTF();
            if(mensaje.toLowerCase().equals("encender")) {
                System.out.println("Encendiendo");
            }else if(mensaje.toLowerCase().equals("reiniciar")) {
                System.out.println("Reiniciando");

            }else if(mensaje.toLowerCase().equals("apagar")){
                System.out.println("Apagando");
            }else {
                System.out.println("comando no valido");
            }
        }while(!"salir".equals(mensaje));
        } catch (IOException e){
            System.err.println(e.getMessage());
        }
    }

}

public class ServerSocketBasico {

    public static void main(String[] args) throws IOException{
        try(ServerSocket socket = new ServerSocket(8090)){
            while(true) {
                System.out.println("Escuchando clientes");
                Socket client = socket.accept();
                System.out.println("cliente conectado: "+client.getInetAddress().getHostName());
                new WorkerSocket(client).start();
            }
    }
    }
}
```

```
->apagar
```

```
Escuchando clientes
cliente conectado: 127.0.0.1
Escuchando clientes
Apagando
```

2- Mandar archivo al servidor

```java
class WorkerSocket extends Thread{

    private Socket client;

    public WorkerSocket(Socket client) {
        this.client=client;
    }

    @Override
    public void run() {
         try (DataInputStream dis = new DataInputStream(client.getInputStream())) {
                String fileName = dis.readUTF();

                if (!fileName.isEmpty()) {
                    try (FileOutputStream fos = new FileOutputStream(fileName)) {
                        byte[] buffer = new byte[4096];
                        int bytesRead;
                        while ((bytesRead = dis.read(buffer)) != -1) {
                            fos.write(buffer, 0, bytesRead);
                        }
                    }
                    System.out.println("Archivo recibido y guardado en el servidor: " + fileName);
                    String currentDirectory = System.getProperty("user.dir");
                    System.out.println("Directorio de trabajo actual del servidor: " + currentDirectory);
                }
            } catch (IOException e) {
                System.err.println(e.getMessage());
            }
    }

}

public class ServerSocketHilado{

    public static void main(String[] args) throws IOException{
        try(ServerSocket socket = new ServerSocket(8090)){
            while(true) {
                System.out.println("Escuchando clientes");
                Socket client = socket.accept();
                System.out.println("cliente conectado: "+client.getInetAddress().getHostName());
                new WorkerSocket(client).start();
            }
    }
    }
```

```java
import java.io.*;
import java.net.Socket;
import java.net.UnknownHostException;
public class ClienteHilado{
    public static void main(String[] args) throws UnknownHostException, IOException {
        try (Socket socket = new Socket("localhost", 8090)) {
            DataOutputStream dos = new DataOutputStream(socket.getOutputStream());


            String nombreArchivo = "F:\\tilin.txt";
            File archivo = new File(nombreArchivo);

            if (archivo.exists()) {

                dos.writeUTF(archivo.getName());


                try (FileInputStream fis = new FileInputStream(archivo)) {
                    byte[] buffer = new byte[4096];
                    int bytesRead;
                    while ((bytesRead = fis.read(buffer)) != -1) {
                        dos.write(buffer, 0, bytesRead);
                    }
                }

                System.out.println("Archivo enviado al servidor.");
            } else {
                System.out.println("El archivo no existe.");
            }
        }
    }
```

```
cliente conectado: 127.0.0.1
Escuchando clientes
Archivo recibido y guardado en el servidor: tilin.txt
Directorio de trabajo actual del servidor: C:\Users\Johan\eclipse-workspace\CursoJavaParte1
```

3- Enviar un objeto simulando http

```java
package JavaSeccion16ejercicio3;
import java.io.*;
import java.net.*;
public class ServidorObjeto {
    public static void main(String[] args) {
        try ( ServerSocket serverSocket = new ServerSocket(8090);){

            System.out.println("Servidor en espera de conexiones...");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Cliente conectado desde " + clientSocket.getInetAddress().getHostAddress());

                ObjectInputStream in = new ObjectInputStream(clientSocket.getInputStream());

                ObjectOutputStream out = new ObjectOutputStream(clientSocket.getOutputStream());

                ObjetoHttp httpRecibido = (ObjetoHttp) in.readObject();
                System.out.println("Objeto recibido del cliente: " + httpRecibido.toString());

                ObjetoHttpRespuesta response = new ObjetoHttpRespuesta(httpRecibido.getBody(), httpRecibido.getHeaders(),202);
                out.writeObject(response);
                out.flush();

                in.close();
                out.close();
                clientSocket.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```java
package JavaSeccion16ejercicio3;
import java.io.*;
import java.net.*;
import java.util.HashMap;
import java.util.Map;
public class ClienteObjeto {

    public static void main(String[] args) throws  UnknownHostException, IOException{
        try {
            Socket socket = new Socket("localhost", 8090);


            ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());


            Map<String, String> headers = new HashMap<>();
            headers.put("Content-Type", "text/plain");
            headers.put("Authorization", "Bearer ABC123");

            ObjetoHttp objetoHttp = new ObjetoHttp("Contenido del cuerpo del mensaje", headers);

            out.writeObject(objetoHttp);
            out.flush();


            ObjectInputStream in = new ObjectInputStream(socket.getInputStream());


            Object response = in.readObject();
            System.out.println("Respuesta del servidor: " + response.toString());


            out.close();
            in.close();
            socket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        }
    }
```

```java
package JavaSeccion16ejercicio3;

import java.io.Serializable;

public class ObjetoHttp implements Serializable {
    protected String body;
    protected Map<String,String> headers;

    public ObjetoHttp(String body, Map<String, String> headers) {
        this.body = body;
        this.headers = headers;
    }



    public String getBody() {
        return body;
    }



    public void setBody(String body) {
        this.body = body;
    }



    public Map<String, String> getHeaders() {
        return headers;
    }



    public void setHeaders(Map<String, String> headers) {
        this.headers = headers;
    }



    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("ObjetoHttp {");
        sb.append("body='").append(body).append('\'');
        sb.append(", headers=").append(headers);
        sb.append('}');
        return sb.toString();
    }
}
```

```java
package JavaSeccion16ejercicio3;

import java.util.Map;

public class ObjetoHttpRespuesta extends ObjetoHttp{

    int httpResponse;

    public ObjetoHttpRespuesta(String body, Map<String, String> headers, int httpResponse) {
        super(body, headers);
        this.httpResponse=httpResponse;
    }
    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("ObjetoHttp {");
        sb.append("body='").append(body).append('\'');
        sb.append(", headers=").append(headers);
        sb.append('}');
        sb.append('\n').append(httpResponse);
        return sb.toString();
    }

}
```

```
ServidorObjeto [Java Application] C:\Program Files\Java\jdk1.8.0_202\bin\javaw.exe (24 oct. 2023 22:05:13) [pid: 10396]
Servidor en espera de conexiones...
Cliente conectado desde 127.0.0.1
Objeto recibido del cliente: ObjetoHttp {body='Contenido del cuerpo del mensaje', headers={Authorization=Bearer ABC123, Content-Type=text/plain}}
```

```
Respuesta del servidor: ObjetoHttp {body='Contenido del cuerpo del mensaje', headers={Authorization=Bearer ABC123, Content-Type=text/plain}}
202
```