



SOCKETS UDP

MARIO JIMÉNEZ MARSET

ÍNDICE

1. ENUNCIADO – OBJETIVOS	3
2. DESARROLLO - PROCEDIMIENTOS	3

1. ENUNCIADO – OBJETIVOS

En esta práctica se pedía realizar dos tareas a partir de una aplicación cliente/servidor UDP. Se pedía que el servidor fuese capaz de proporcionar la hora y el día a los clientes que lo soliciten. El cliente espera entonces la respuesta del servidor un tiempo limitado. Si recibe la respuesta, verá la hora y día proporcionados por el servidor. Si no, enviará a la salida estándar un mensaje de error.

Después de conseguir esto, se pedía comparar la hora local con una hora remota; a través de un mensaje, se ponía la diferencia entre ellas.

2. DESARROLLO - PROCEDIMIENTOS

Se incluye el código del cliente y el servidor con comentarios:

Código Servidor:

```
package udpejercicios;
import java.net.*;
import java.text.SimpleDateFormat;
import java.util.*;
import java.io.*;

public class ServidorEj1 {
    public static void main(String[] args) throws InterruptedException, IOException {
        //se crea la fecha del Servidor con la clase Date
        Date dateServidor = new Date();
        //se crea el socket pasándole el número de puerto
        DatagramSocket datagramSocket = new DatagramSocket(5001);
        System.out.println("El servidor esta en funcionamiento");
        //se crea el bufer que almacena la información
        byte[] bufer=new byte[1000];
        //bucle infinito
        while (true) {
            //se crea el DatagramPacket con la información del bufer,
            //llamando al método receive
            DatagramPacket datagramPacketRecibido = new
            DatagramPacket(bufer, bufer.length);
            datagramSocket.receive(datagramPacketRecibido);
            System.out.println("Un cliente ha mandado una petición de
            fecha y hora");
            //se crea la fecha del Cliente
            Date dateCliente = new Date();
            //se crea el formato de fecha
            String formato=new SimpleDateFormat("HH:mm:ss:SS
            dd/MM/yyyy").format(dateCliente);
            String mensaje = new String(formato);
            //se pasa a bytes el mensaje
            bufer = mensaje.getBytes();
            //se crea el paquete y se envía con esta información
            DatagramPacket datagramPacketPaquete = new
            DatagramPacket(bufer, bufer.length, datagramPacketRecibido.getAddress(),
            datagramPacketRecibido.getPort());
            datagramSocket.send(datagramPacketPaquete);
            //se consigue la hora del servidor y el cliente y se imprime la
            //diferencia entre ellos
        }
    }
}
```

```

        long horaServidor = dateServidor.getTime();
        long horaCliente = dateCliente.getTime();
        long diferencia = horaCliente - horaServidor;
        System.out.println("Han pasado " + diferencia / 1000 + "
segundos de tiempo de espera del servidor");
    }
}

```

Código Cliente:

```

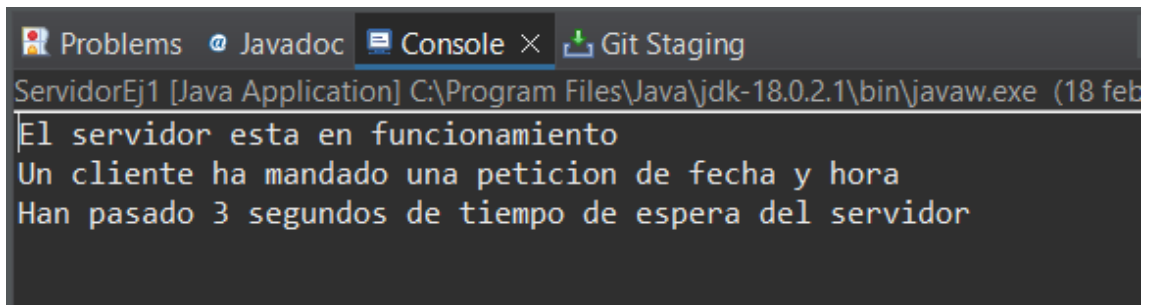
package udpejercicios;
import java.net.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.io.*;

public class ClienteEj1 {
    public static void main(String[] args) throws InterruptedException,
    ParseException {
        //se crea la fechaRemota perteneciente a la clase Date
        Date fechaRemota = new Date();
        try {
            //se crea el bufer que almacena la información recibida
            byte bufer[] = new byte[1000];
            //se crea el socket
            DatagramSocket datagramSocket = new DatagramSocket();
            //se llama al metodo setSoTimeout, el cual permite establecer
            un tiempo de espera límite
            datagramSocket.setSoTimeout(5000);
            //se recoge en una variable 'localhost', el cual es pasado por
            argumentos
            InetAddress ipDestino = InetAddress.getByName(args[0]);
            //se recoge con un DatagramPacket la información
            DatagramPacket datagramPacketRecibido = new
            DatagramPacket(bufer, bufer.length, ipDestino, 5001);
            //se envía esta información
            datagramSocket.send(datagramPacketRecibido);
            datagramPacketRecibido = new DatagramPacket(bufer,
            bufer.length);
            //se recibe la información del servidor
            datagramSocket.receive(datagramPacketRecibido);
            //con un String se recoge la información
            String mensajeRecibido = new
            String(datagramPacketRecibido.getData());
            //se imprime la hora Local y Remota
            System.out.println("Fecha y Hora -Local-: " +
            mensajeRecibido);
            SimpleDateFormat formato = new
            SimpleDateFormat("HH:mm:ss:SS dd/MM/yyyy");
            System.out.println("Fecha y Hora -Remota-: " +
            formato.format(fechaRemota));
            //se parsea
            Date local = formato.parse(mensajeRecibido);
            Date remota = formato.parse(formato.format(fechaRemota));
            //se hace la operación para conseguir la diferencia de tiempos

```

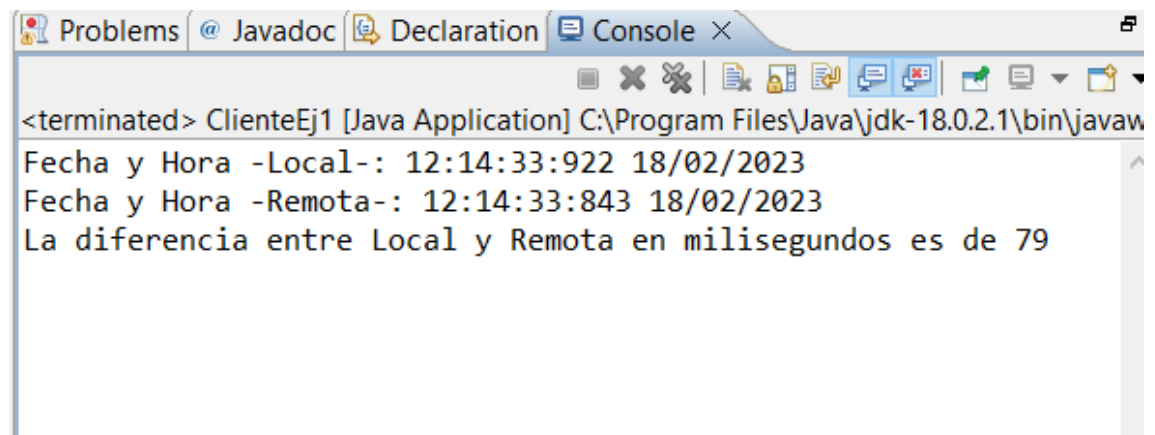
```
        long diferenciaLocalRemota = Math.abs(remota.getTime() -  
local.getTime());  
        long diferenciaMilisegundos =  
        TimeUnit.MILLISECONDS.convert(diferenciaLocalRemota, TimeUnit.MILLISECONDS);  
        //se imprime esta diferencia  
        System.out.println("La diferencia entre Local y Remota en  
milisegundos es de " + diferenciaMilisegundos);  
        //se cierra el socket  
        datagramSocket.close();  
    } catch (SocketException e) {  
        System.out.println("Socket: " + e.getMessage());  
    } catch (IOException e) {  
        System.out.println("IO: " + e.getMessage());  
    }  
    }  
}
```

Resultados:



The screenshot shows the IDE's console window with the following output from the 'ServidorEj1' application:

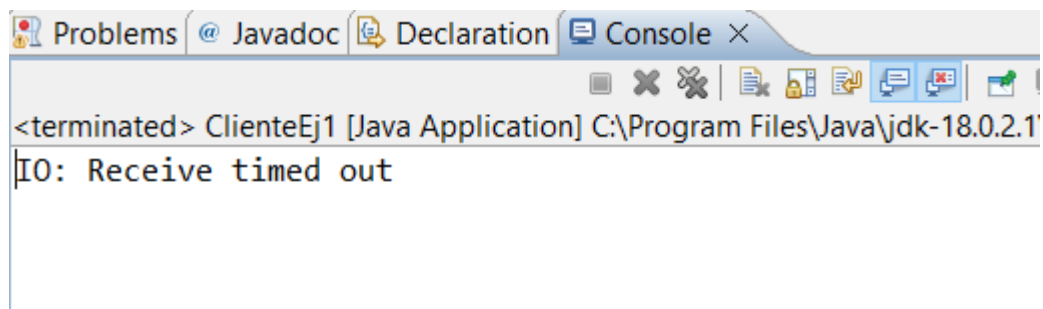
```
El servidor esta en funcionamiento  
Un cliente ha mandado una peticion de fecha y hora  
Han pasado 3 segundos de tiempo de espera del servidor
```



The screenshot shows the IDE's console window with the following output from the 'ClienteEj1' application:

```
<terminated> ClienteEj1 [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw  
Fecha y Hora -Local-: 12:14:33:922 18/02/2023  
Fecha y Hora -Remota-: 12:14:33:843 18/02/2023  
La diferencia entre Local y Remota en milisegundos es de 79
```

Si no fuese bien la conexión, saldría un mensaje de error:



The screenshot shows the IDE's console window with the following error output from the 'ClienteEj1' application:

```
<terminated> ClienteEj1 [Java Application] C:\Program Files\Java\jdk-18.0.2.1  
IO: Receive timed out
```