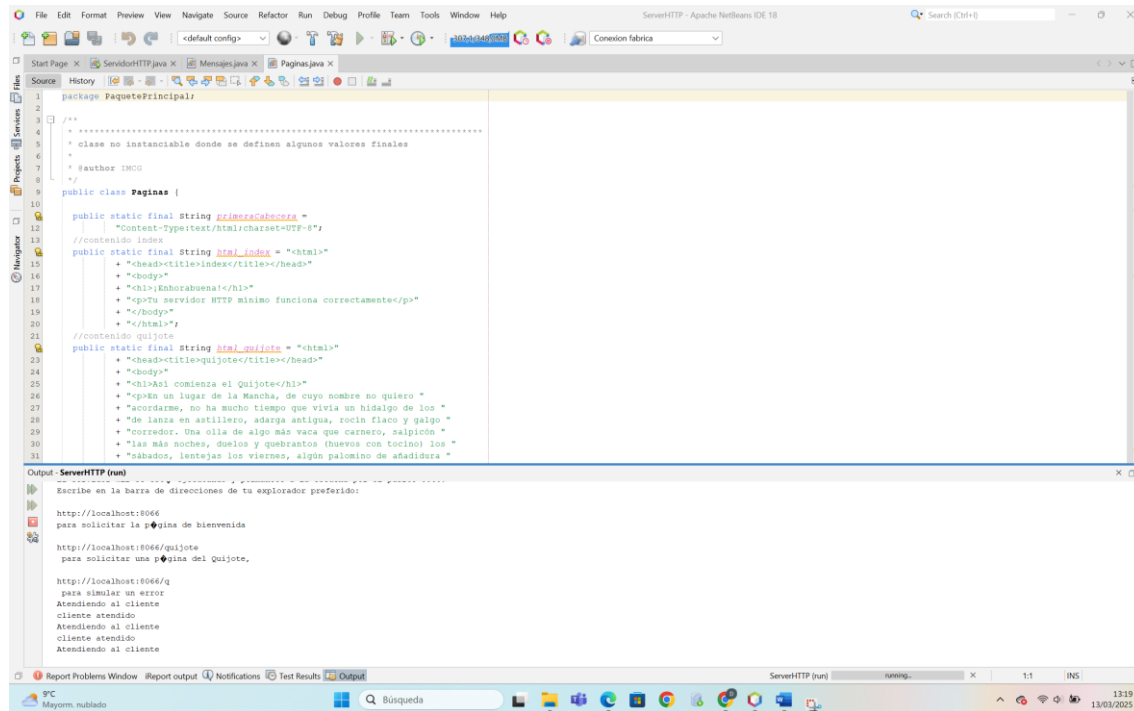


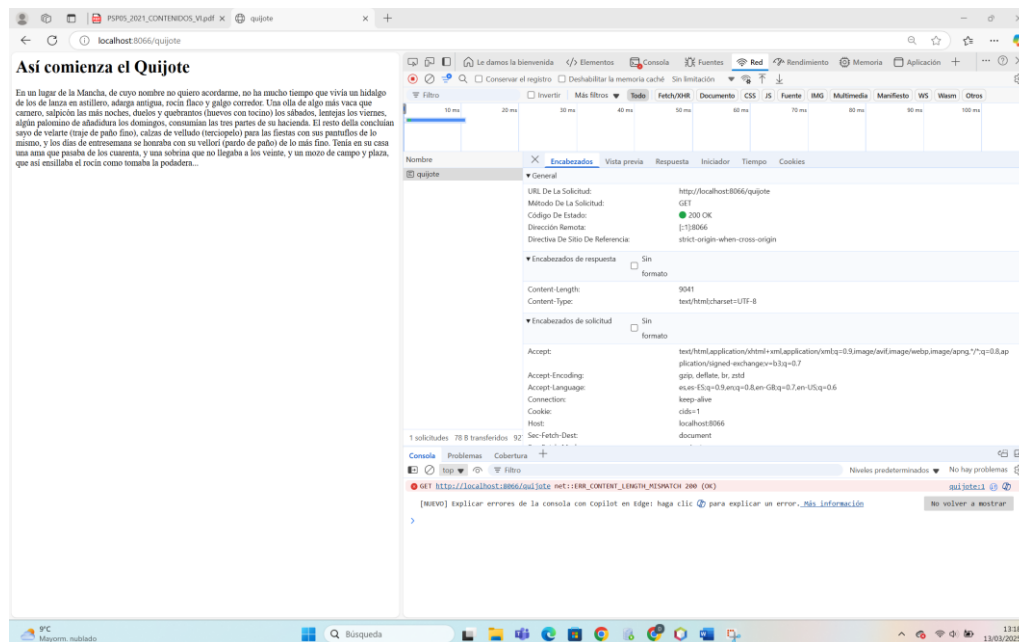
**Enunciado.****Ejercicio 1.**

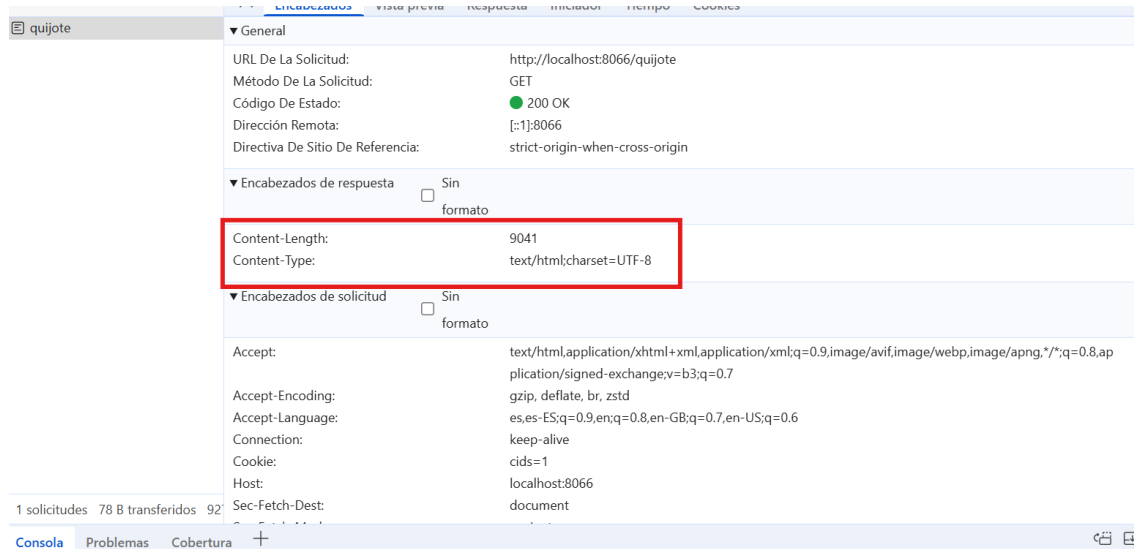
**Modifica el ejemplo del servidor HTTP (Proyecto java ServerHTTP, apartado 5.1 de los contenidos) para que incluya la cabecera Date.**

Como podemos observar en un inicio el código para este HTML no incorpora ninguna cabecera del tipo Date:



Aquí lo podemos ver en la página “quijote” abierta en localhost desde el explorador de Windows:





Para añadir esta cabecera modificamos el código añadiendo un *printWriter* que permite escribir datos de texto en un flujo de salida (en este caso, el socket del cliente). Para ello se ha usado la clase *ZonedDateTime* que representa fecha + hora + zona horaria. Junto con *.now()*, método estático que obtiene la fecha/hora actual. Se indica la zona horaria como *GMT* (Meridiano de Greenwich).

Esta hora se formatea utilizando el *DateTimeFormatter.RFC\_1123\_DATE\_TIME*, formato predefinido equivalente a:

EEE, dd MMM yyyy HH:mm:ss z:

- EEE: Día de la semana corto (ej: "Fri")
- dd: Día del mes (ej: "05")
- MMM: Mes en 3 letras (ej: "Jan")
- yyyy: Año en 4 dígitos
- HH:mm:ss: Hora en formato 24h
- z: Zona horaria (ej: "GMT")

```
if (peticion.startsWith(prefix: "GET")) {  
    //extrae la subcadena entre 'GET' y 'HTTP/1.1'  
    peticion = peticion.substring(beginIndex: 3, endIndex: peticion.lastIndexOf(str: "HTTP"));  
  
    String fecha = ZonedDateTime.now(zone: ZoneId.of(zoneId: "GMT")).format(formatter: DateTimeFormatter.RFC_1123_DATE_TIME);  
    //si corresponde a la página de inicio  
    if (peticion.length() == 0 || peticion.equals(anObject: "/")) {  
        //sirve la página  
        html = Paginas.html_index;  
        printWriter.println(x: Mensajes.lineaInicial_OK);  
        printWriter.println(x: Paginas.primerCabecera);  
        printWriter.println("Content-Length: " + html.length());  
        printWriter.println("Date: " + fecha);  
        printWriter.println(x: "\n");  
        printWriter.println(x: html);  
    } //si corresponde a la página del Quijote  
    else if (peticion.equals(anObject: "/quijote")) {  
        //sirve la página  
        html = Paginas.html_quijote;  
        printWriter.println(x: Mensajes.lineaInicial_OK);  
        printWriter.println(x: Paginas.primerCabecera);  
        printWriter.println("Content-Length: " + html.length());  
        printWriter.println("Date: " + fecha);  
        printWriter.println(x: "\n");  
        printWriter.println(x: html);  
    } //en cualquier otro caso  
    else {  
        //sirve la página  
        html = Paginas.html_noEncontrado;  
        printWriter.println(x: Mensajes.lineaInicial_NotFound);  
        printWriter.println(x: Paginas.primerCabecera);  
        printWriter.println("Content-Length: " + html.length());  
        printWriter.println("Date: " + fecha);  
        printWriter.println(x: "\n");  
        printWriter.println(x: html);  
    }  
}
```

HTTP exige que la cabecera Date esté en GMT, por lo que aparece una hora menos. Si queremos poner hora española habría que añadir GMT +1 en su lugar.

The screenshot shows a web browser window with the address bar displaying 'localhost:8066/quijote'. The page content is titled 'Así comienza el Quijote' and contains a paragraph of text. On the right side, the browser's developer tools are open, showing the 'Encabezados' (Headers) tab. The 'Encabezados de respuesta' (Response Headers) section is expanded, showing the following details:

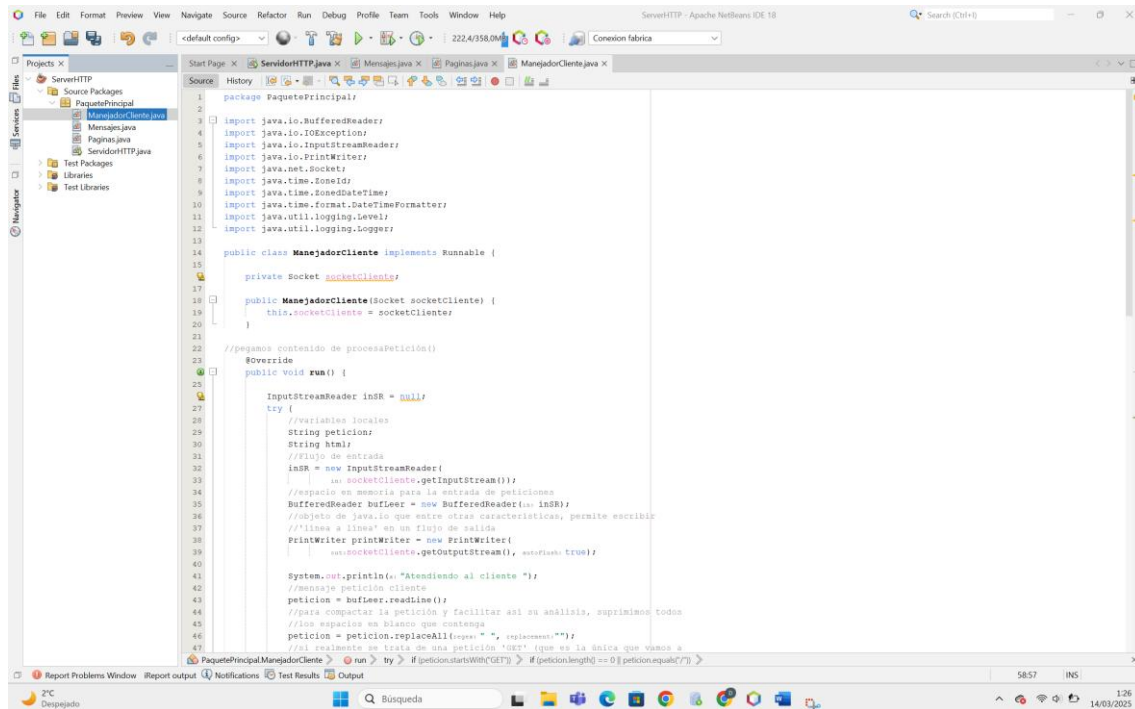
- URL De La Solicitud: http://localhost:8066/quijote
- Método De La Solicitud: GET
- Código De Estado: 200 OK
- Dirección Remota: [::1]:8066
- Directiva De Sitio De Referencia: strict-origin-when-cross-origin
- Encabezados de respuesta: Sin formato
- Content-Length: 904
- Content-Type: text/html; charset=UTF-8
- Date: Thu, 13 Mar 2025 17:38:44 GMT
- Encabezados de solicitud: Sin formato
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng/\*; q=0.8,application/signed-exchange;v=b3;q=0.7
- Accept-Encoding: gzip, deflate, br, zstd
- Accept-Language: es-ES;q=0.9,enq=0.8,en-GB;q=0.7,en-US;q=0.6
- Connection: keep-alive
- Cookies: csls=1
- Host: localhost:8066

The bottom of the screenshot shows the Windows taskbar with the system clock indicating 18:39 on 13/03/2025.

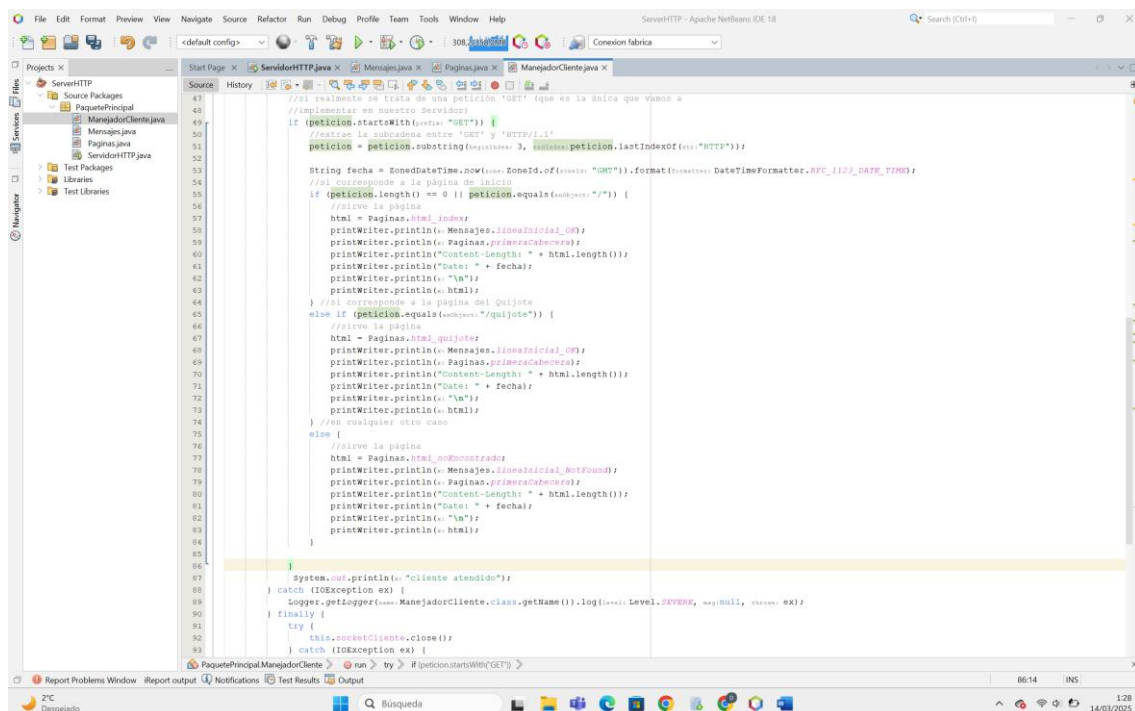
## Ejercicio 2.

Modifica el ejemplo del servidor HTTP (Proyecto java ServerHTTP, apartado 5.1 de los contenidos) para que implemente multihilo, y pueda gestionar la concurrencia de manera eficiente.

Para implementar multihilo en este programa, se ha procedido a crear una nueva clase, la clase `manejadorCliente()`, que implementa `Runnable`. Esto permite que cada instancia maneje una conexión de cliente en un hilo independiente, de forma que se puedan hacer múltiples peticiones al servidor simultáneamente.



```
1 package PaquetePrincipal;
2
3 import java.io.BufferedReader;
4 import java.io.IOException;
5 import java.io.InputStreamReader;
6 import java.io.PrintWriter;
7 import java.net.Socket;
8 import java.time.ZoneId;
9 import java.time.ZonedDateTime;
10 import java.time.format.DateTimeFormatter;
11 import java.util.logging.Level;
12 import java.util.logging.Logger;
13
14 public class ManejadorCliente implements Runnable {
15
16     private Socket socketCliente;
17
18     public ManejadorCliente(Socket socketCliente) {
19         this.socketCliente = socketCliente;
20     }
21
22     //pega el contenido de processaPetición()
23     @Override
24     public void run() {
25
26         InputStreamReader inSR = null;
27         try {
28             //Variables locales
29             String petición;
30             String html;
31             //Flujo de entrada
32             inSR = new InputStreamReader(
33                 socketCliente.getInputStream());
34             //espacio en memoria para la entrada de peticiones
35             BufferedReader bufLeer = new BufferedReader(inSR);
36             //objeto de java.io que entre otras características, permite escribir
37             //Una especie de búfer en un flujo de salida
38             PrintWriter printWriter = new PrintWriter(
39                 socketCliente.getOutputStream(), autoFlush true);
40
41             System.out.println("Atendiendo al cliente ");
42             //Muestra petición cliente
43             petición = bufLeer.readLine();
44             //para compactar la petición y facilitar así su análisis, suprimimos todos
45             //los espacios en blanco que contenga
46             petición = petición.replaceAll("\\s+", "");
47             //si realmente se trata de una petición "GET" (que es la única que vamos a
```



```
48 //si realmente se trata de una petición "GET" (que es la única que vamos a
49 //implementar en nuestro Servidor)
50 if (petición.startsWith("GET")) {
51     //extrae la subcadena entre "GET" y "HTTP/1.1"
52     petición = petición.substring(petición.indexOf("GET") + 1,
53         petición.indexOf("HTTP/1.1"));
54
55     String fecha = ZonedDateTime.now(ZoneId.of("GMT")).format(DateTimeFormatter.RFC_1123_DATE_TIME);
56     //si corresponde a la página de inicio
57     if (petición.length() == 0 || petición.equals("/")) {
58         //serve la página
59         html = Paginas.html_inicio;
60         printWriter.println(Mensajes.lineaInicial_ON);
61         printWriter.println(Paginas.primerCabecera);
62         printWriter.println("Content-Length: " + html.length());
63         printWriter.println("Date: " + fecha);
64         printWriter.println("A");
65         printWriter.println(html);
66     } //si corresponde a la página del Quijote
67     else if (petición.equals("/quijote")) {
68         //serve la página
69         html = Paginas.html_quijote;
70         printWriter.println(Mensajes.lineaInicial_ON);
71         printWriter.println(Paginas.primerCabecera);
72         printWriter.println("Content-Length: " + html.length());
73         printWriter.println("Date: " + fecha);
74         printWriter.println("A");
75         printWriter.println(html);
76     } //en cualquier otro caso
77     else {
78         //serve la página
79         html = Paginas.html_noContenido;
80         printWriter.println(Mensajes.lineaInicial_NotFound);
81         printWriter.println(Paginas.primerCabecera);
82         printWriter.println("Content-Length: " + html.length());
83         printWriter.println("Date: " + fecha);
84         printWriter.println("A");
85         printWriter.println(html);
86     }
87
88     System.out.println("Cliente atendido");
89 } catch (IOException ex) {
90     Logger.getLogger(ManejadorCliente.class.getName()).log(Level.SEVERE, null, ex);
91 } finally {
92     try {
93         this.socketCliente.close();
94     } catch (IOException ex) {
95         //
96     }
97 }
```

Así mismo se ha modificado la clase principal `ServidorHTTP` para hacer que el bucle de aceptación de conexiones, haga que cada cliente se delegue a un nuevo hilo, permitiendo atender a conexiones en paralelo. Así mismo se mostrará por consola la dirección, el host y puerto de cada cliente para comprobar que funciona esta concurrencia:

```
while (true) {  
  
    socCliente = socServidor.accept();  
  
    System.out.println("Dirección de la petición: " +  
socCliente.getInetAddress().getHostAddress() + ":" + socCliente.getPort());  
  
    ManejadorCliente hiloPeticion = new ManejadorCliente(socCliente);  
  
    Thread hilo = new Thread(hiloPeticion);  
  
    hilo.start();  
  
}  
  
}
```

```
class ServidorHTTP {  
    /**  
     * *****  
     * procedimiento principal que asigna a cada petición entrante un socket  
     * cliente, por donde se enviará la respuesta una vez procesada  
     *  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) throws IOException, Exception {  
        //Asociamos al servidor el puerto 8066  
        ServerSocket socServidor = new ServerSocket(port: 8066);  
        imprimeDisponible();  
        Socket socCliente;  
  
        //ante una petición entrante, procesa la petición por el socket cliente  
        //por donde la recibe  
        while (true) {  
            //a la espera de peticiones  
            socCliente = socServidor.accept();  
            System.out.println("Dirección de la petición: " + socCliente.getInetAddress().getHostAddress() + ":" + socCliente.getPort());  
            //atiendo un cliente  
            ManejadorCliente hiloPeticion = new ManejadorCliente(socketCliente: socCliente);  
            Thread hilo = new Thread(task: hiloPeticion);  
            hilo.start();  
        }  
    }  
  
    /**  
     * *****  
     * muestra un mensaje en la Salida que confirma el arranque, y da algunas  
     * indicaciones posteriores  
     */  
    private static void imprimeDisponible() {  
        System.out.println("El Servidor WEB se está ejecutando y permanece a la "  
            + "escucha por el puerto 8066.\nEscribe en la barra de direcciones "  
            + "de tu explorador preferido:\n\nhttp://localhost:8066\npara "  
            + "solicitar la página de bienvenida\n\nhttp://localhost:8066/"  
            + "quijote\n para solicitar una página del Quijote,\n\nhttp://" +  
            + "localhost:8066/q\n para simular un error");  
    }  
}
```

Aquí se muestra como se han ejecutado simultáneamente las 3 páginas mostrando las direcciones correspondientes a cada una:

