Programación de

Servicios y Procesos

CLIENTE-SERVIDOR

Diego González Martínez | 12/01/2023

# Ejercicio 5

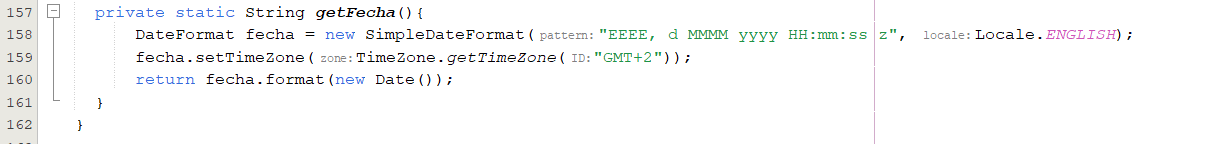
**¿De qué trata el Ejercicio 5?**

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

|  |
| --- |
| 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. |
|  |

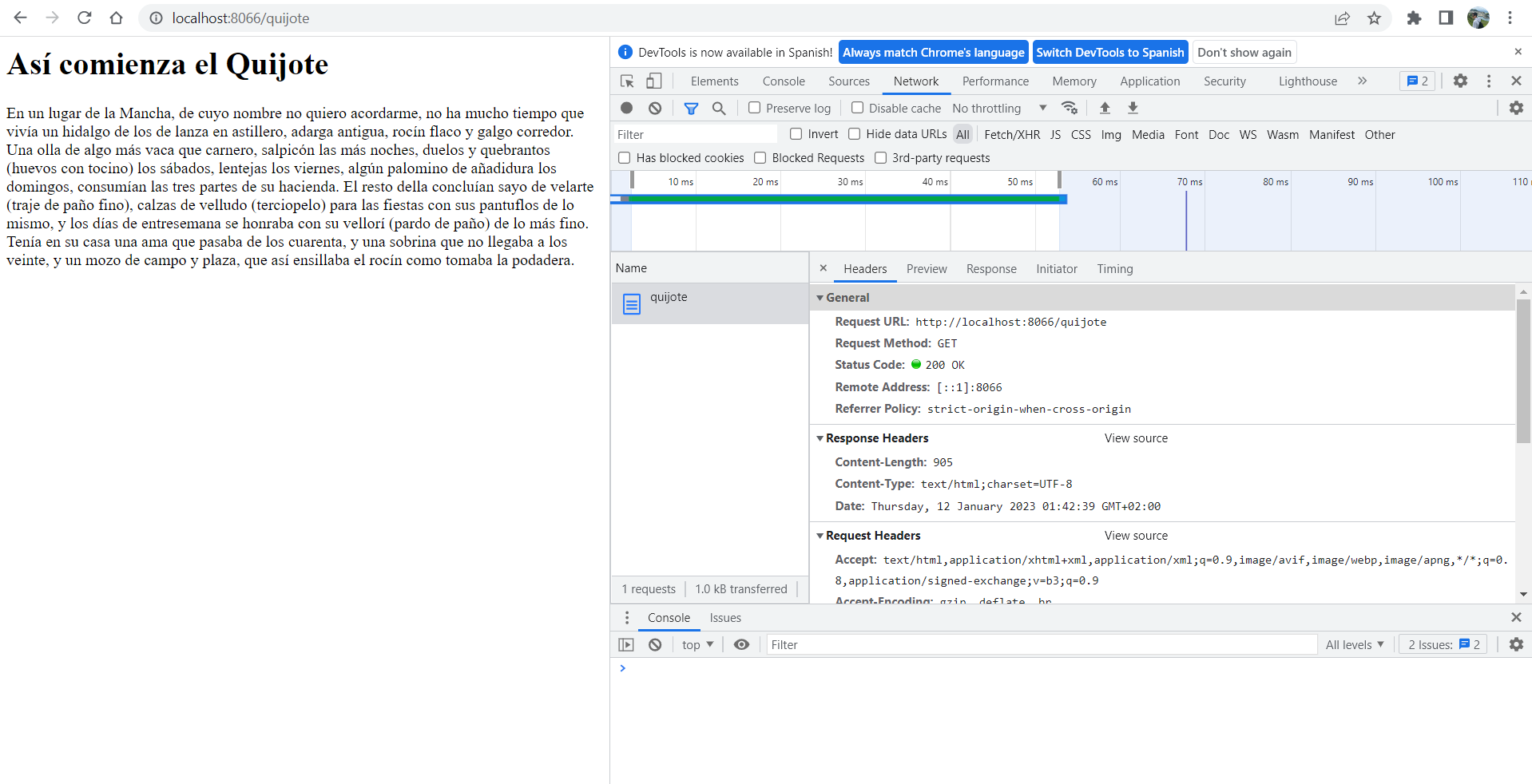
Parte1

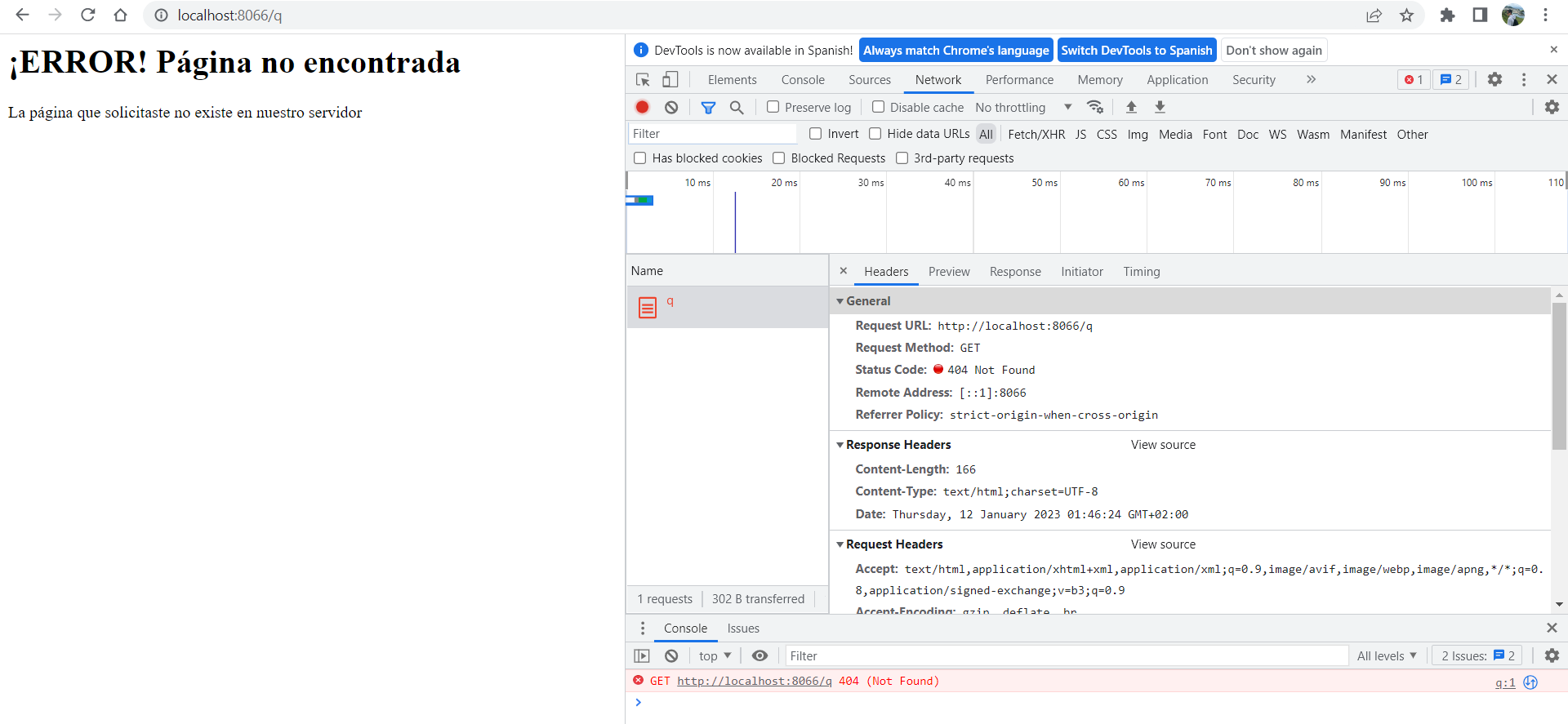
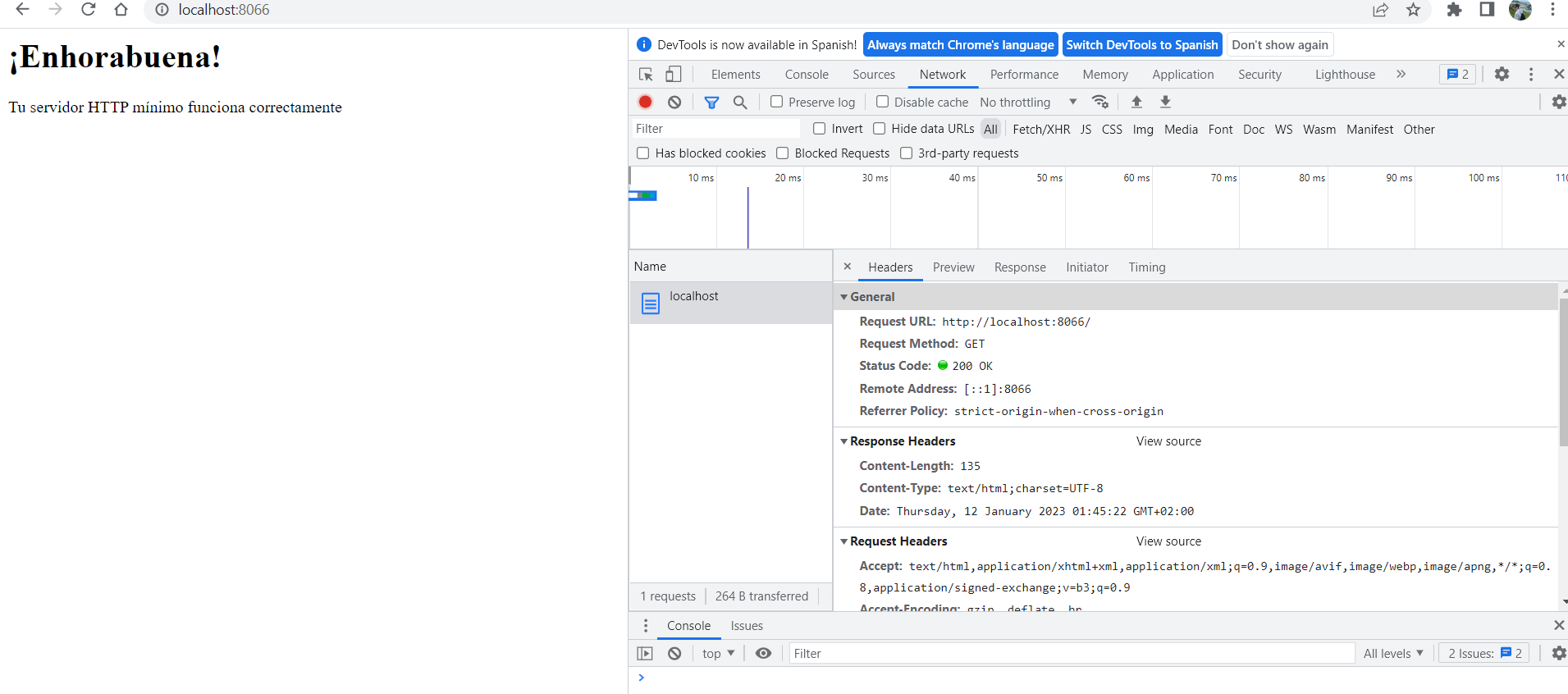
Para realizar este ejercicio tomé como ejemplo el método que devuelve una fecha con el formato adecuado que aparece en la presentación del punto 5.1 y creé el siguiente método:



Lo modifiqué para que aparezca el nombre completo del día de la semana y del mes, pero no he conseguido dar con la configuración correcta de localización para que represente bien las tildes en la fecha devuelta, asi que la dejé ajustada al idioma inglés, tal y como aparece en el ejemplo.

Por último añadí la instrucción necesaria para que imprima la cabecera **Date** en cada una de las tres opciones posibles. A continuación presento las capturas correspondientes a cada una de las tres páginas:

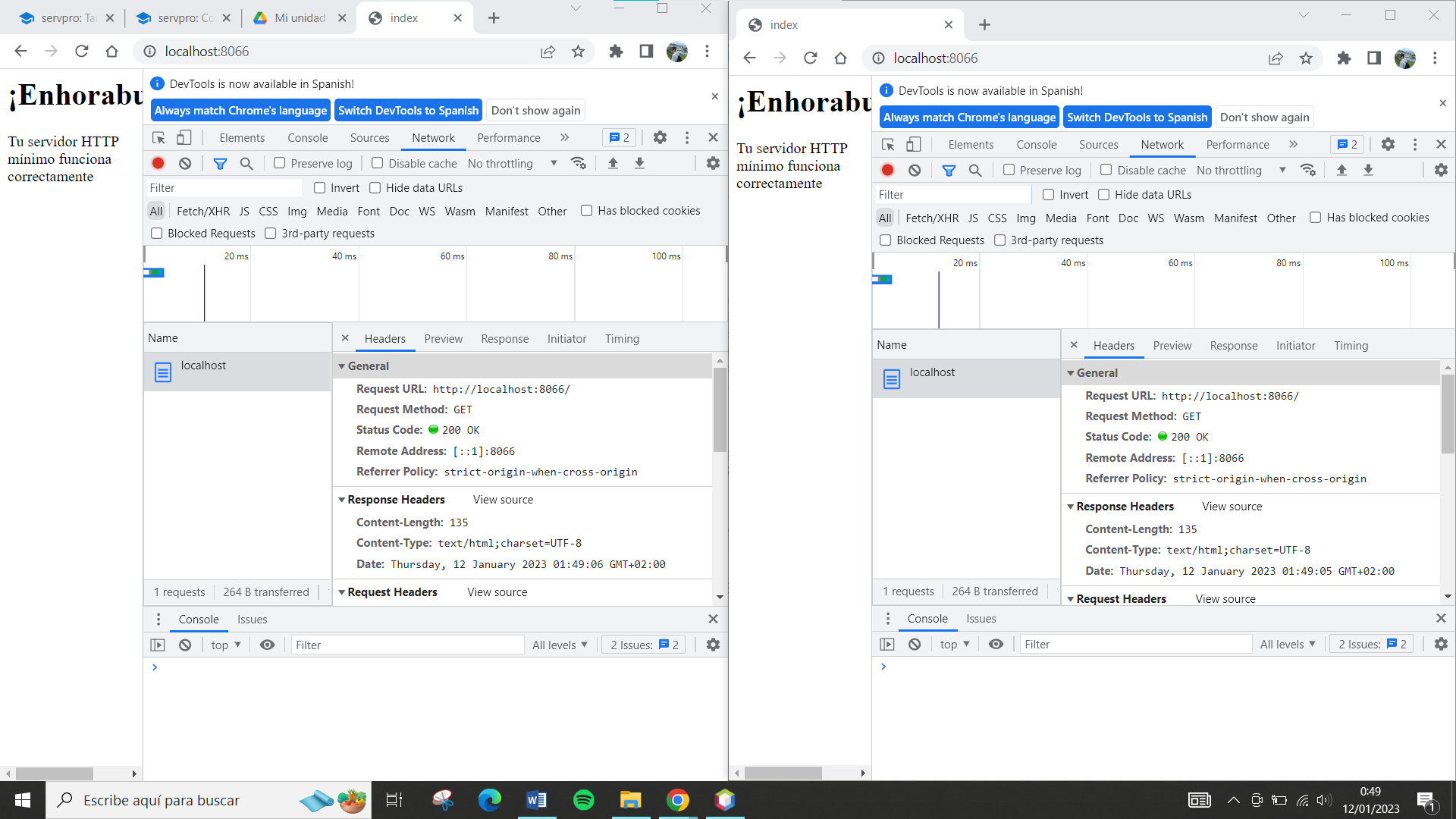




Parte 2:

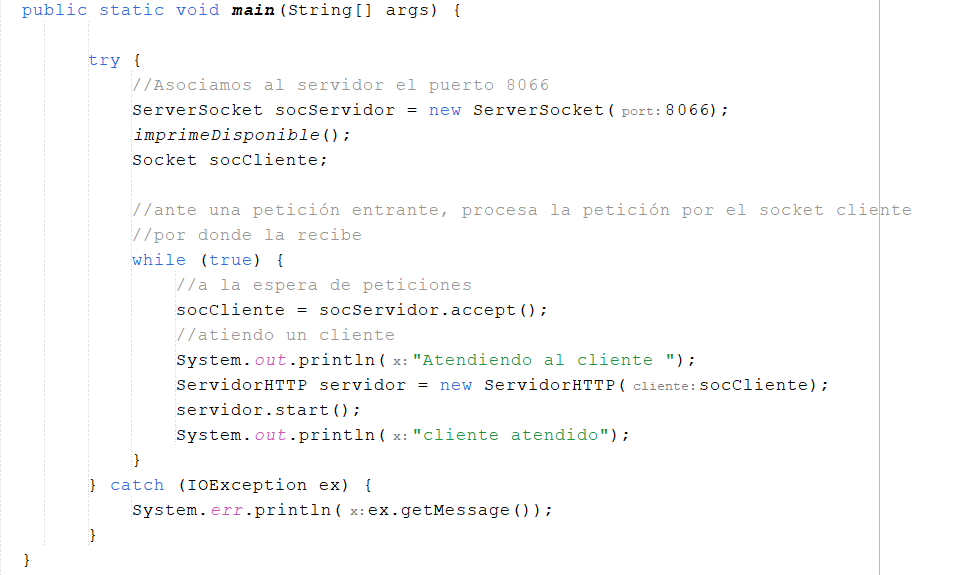
En esta segunda parte he tenido que modificar las instrucciones que fijan el tamaño de la cabezera **Content-Lenght**, poniendo dentro de paréntesis toda la instrucción html.lenght()+1, pues sino el uno final lo tomaba como un **String** y lo añadía al final de la frase como un carácter más, en lugar de sumarlo al valor de la longitud de la variable **html**.

En la siguiente captura se puede apreciar como se ejecutan dos consultas en dos navegadores distintos de forma simultánea:



**¿Cómo se permite que se conecte más de un cliente al servidor?**

La parte del código que implementa multihilo para conectar más de un cliente al servidor es el bucle while en la función main, donde se esperan las peticiones entrantes mediante el método "accept" del objeto ServerSocket. Cada vez que llega una petición, se crea un objeto ServidorHTTP con el socket del cliente y se inicia un nuevo hilo para atender esa petición. El método "run" es el que se ejecuta en el hilo creado y es el que se encarga de procesar la petición y enviar la respuesta.



**Resumen:**

Nuestra clase servidor extiende de Thread, usamos el método start() para iniciar un nuevo hilo cada vez que atiende a una clas; el proceso de petición lo realizamos mediante el método run() de la clase “Servidor HTTP”, también usamos el método run() para cerrar la conexión.

**JAVADOC:**

**Tema 5\TareaPSP05\TareaPSP05\ServerHTTP\nbproject\build-impl.xml:1507**