Práctica 4

Servidor WEB

Integrantes:

- Flores Morales Aldahir Andrés
- Velasco Jiménez Luis Antonio

Materia: Aplicaciones para comunicaciones en red

Grupo: 6CM1

Resumen

En esta práctica, se tuvo que implementar un servidor web, haciendo uso de una **alberca de hilos**, para limitar la cantidad de clientes que se podían conectar al servidor. Hicimos uso los tipos de métodos de peticiones HTTP, mas importantes, como lo son GET, POST, PUT, DELETE y HEAD; con la finalidad de que el cliente pudiera hacer peticiones para obtener, subir, modificar totalmente un recurso, ver los encabezados de respuesta, y borrar elementos en el servidor.

Introducción

Un **pool de hilos** es una técnica utilizada en programación concurrente para administrar y reutilizar un conjunto fijo de hilos en lugar de crear y destruir hilos repetidamente. Esto mejora la eficiencia en aplicaciones donde se realizan muchas tareas concurrentes, ya que la creación y destrucción de hilos conlleva un costo computacional significativo.

¿Cómo funciona un pool de hilos?

1. **Inicialización:** Al iniciar, se crea un número fijo o configurable de hilos en el pool, que permanecen en estado inactivo hasta que se les asigna una tarea.
2. **Asignación de tareas:** Cuando una tarea necesita ejecutarse, se asigna a uno de los hilos disponibles en el pool.
3. **Ejecución de la tarea:** El hilo ejecuta la tarea de manera concurrente. Una vez que la tarea finaliza, el hilo no se destruye, sino que regresa al pool y queda disponible para otras tareas.
4. **Cola de tareas:** Si no hay hilos disponibles, las tareas entrantes se colocan en una cola para esperar hasta que un hilo esté libre.

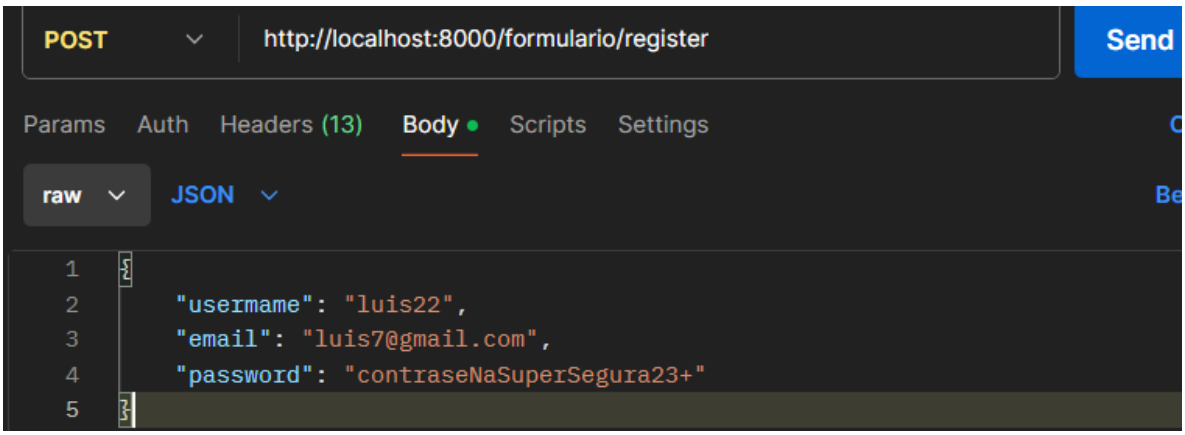
Métodos HTTP

Los **métodos de peticiones HTTP** son acciones que un cliente (como un navegador web o una aplicación) solicita a un servidor. Representan las operaciones que se desean realizar en un recurso determinado identificado por una URL, con el objetivo de lograr algo diferente o inclusive igual con cada método, (con el método GET siempre se obtiene lo mimso).

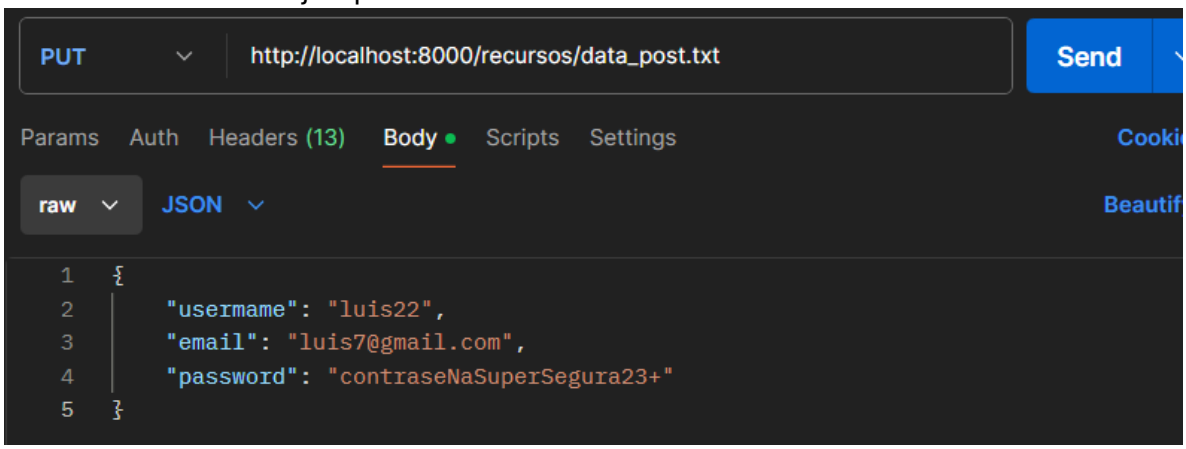
- GET:
  - Se utiliza para solicitar datos o un recurso del servidor.
  - Características:
    - Es idempotente (múltiples peticiones tendrán el mismo resultado).
    - Los datos solicitados se envían en la URL.
    - No modifica el estado del servidor.
    - Ejemplo:

```
Datos: GET /recursos/shaq.jpg HTTP/1.1
```

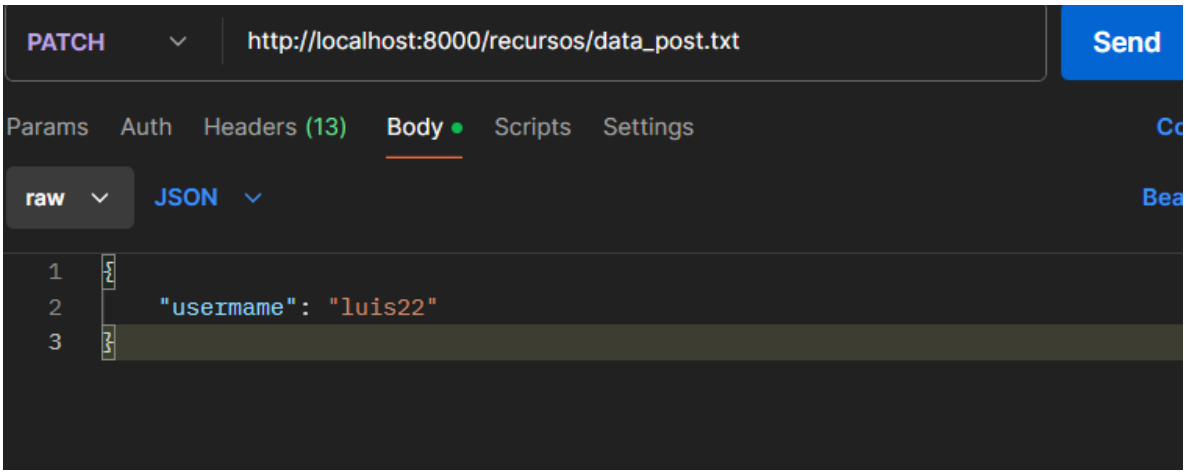
- POST:
  - Se utiliza para enviar datos al servidor, generalmente para crear un nuevo recurso.
  - Características:
    - Puede modificar el estado del servidor.
    - No es idempotente ya que cada petición crea un nuevo recurso en el server.
    - Los datos suelen enviarse en el cuerpo (body) de la petición.
    - Ejemplo: El ejemplo mas sencillo es mediante un formulario, donde los inputs se mandan a través de formato JSON:



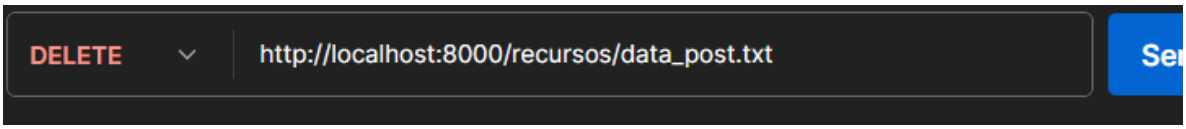
- PUT
  - Se utiliza para actualizar o reemplazar completamente un recurso en el servidor.
  - Características:
    - Es idempotente ya que si buscamos mandar una misma request varias veces para modificar un elemento, el resultado siempre es el mismo.
    - Los datos de envían en el cuerpo (body) de la petición.
    - Ejemplo:



- PATCH
  - Se utiliza para realizar una actualización parcial de un recurso.
  - Características:
    - Es idempotente.
    - Permite modificar solo ciertas propiedades del recurso.
    - Ejemplo:



- DELETE
    - Se utiliza para eliminar un recurso en el servidor.
    - Características:
        - Es idempotente.
        - No necesita cuerpo, ya que solo con proporcionar el recurso a eliminar, es suficiente.
        - Ejemplo:



- HEAD
    - Similar a GET, pero solo solicita los encabezados de la respuesta, sin el cuerpo.
    - Verifica si un recurso está disponible o comprobar su metadata.

Tipos MIME

Los **tipos MIME** (Multipurpose Internet Mail Extensions) son un estándar para identificar el formato de los archivos y datos que se transfieren en internet. Originalmente diseñados para el correo electrónico, ahora se usan ampliamente en HTTP para especificar el tipo de contenido de los recursos web, como documentos HTML, imágenes, videos, archivos JSON, etc.

Cada tipo MIME tiene el formato: ***Tipo / subtipo***

Donde:

- Tipo:
  - Es la categoría general del contenido (por ejemplo, text, image).
- Subtipo:
  - Especifica el formato exacto del contenido (por ejemplo, html, jpeg).

En la siguiente tabla se presentan todos los tipos MIMES:

Extensión	Tipo de documento	Tipo de MIME
.aac	Archivo de audio AAC	audio/aac
.abw	Documento AbiWord	application/x-abiword
.arc	Documento de Archivo (múltiples archivos incrustados)	application/octet-stream
.avi	AVI: Audio Video Intercalado	video/x-msvideo
.azw	Formato eBook Amazon Kindle	application/vnd.amazon.ebook
.bin	Cualquier tipo de datos binarios	application/octet-stream
.bz	Archivo BZip	application/x-bzip
.bz2	Archivo BZip2	application/x-bzip2
.csh	Script C-Shell	application/x-csh
.css	Hojas de estilo (CSS)	text/css
.csv	Valores separados por coma (CSV)	text/csv
.doc	Microsoft Word	application/msword
.epub	Publicación Electrónica (EPUB)	application/epub+zip
.gif	Graphics Interchange Format (GIF)	image/gif
.htm .html	Hipertexto (HTML)	text/html
.ico	Formato Icon	image/x-icon
.ics	Formato iCalendar	text/calendar
.jar	Archivo Java (JAR)	application/java-archive
.jpeg .jpg	Imágenes JPEG	image/jpeg
.js	JavaScript (ECMAScript)	application/javascript
.json	Formato JSON	application/json

Extensión	Tipo de documento	Tipo de MIME
.mid .midi	Interfaz Digital de Instrumentos Musicales (MIDI)	audio/midi
.mpeg	Video MPEG	video/mpeg
.mpkg	Paquete de instalación de Apple	application/vnd.apple.installer+xml
.odp	Documento de presentación de OpenDocument	application/vnd.oasis.opendocument.presentation
.ods	Hoja de Cálculo OpenDocument	application/vnd.oasis.opendocument.spreadsheet
.odt	Documento de texto OpenDocument	application/vnd.oasis.opendocument.text
.oga	Audio OGG	audio/ogg
.ogv	Video OGG	video/ogg
.ogx	OGG	application/ogg
.pdf	Adobe Portable Document Format (PDF)	application/pdf
.ppt	Microsoft PowerPoint	application/vnd.ms-powerpoint
.rar	Archivo RAR	application/x-rar-compressed
.rtf	Formato de Texto Enriquecido (RTF)	application/rtf
.sh	Script Bourne shell	application/x-sh
.svg	Gráficos Vectoriales (SVG)	image/svg+xml
.swf	Small web format (SWF) o Documento Adobe Flash	application/x-shockwave-flash
.tar	Aerchivo Tape (TAR)	application/x-tar
.tif .tiff	Formato de archivo de imagen etiquetado (TIFF)	image/tiff
.ttf	Fuente TrueType	font/ttf

Extensión	Tipo de documento	Tipo de MIME
.vsd	Microsft Visio	application/vnd.visio
.wav	Formato de audio de forma de onda (WAV)	audio/x-wav
.weba	Audio WEBM	audio/webm
.webm	Video WEBM	video/webm
.webp	Imágenes WEBP	image/webp
.woff	Formato de fuente abierta web (WOFF)	font/woff
.woff2	Formato de fuente abierta web (WOFF)	font/woff2
.xhtml	XHTML	application/xhtml+xml
.xls	Microsoft Excel	application/vnd.ms-excel
.xml	XML	application/xml
.xul	XUL	application/vnd.mozilla.xul+xml
.zip	Archivo ZIP	application/zip
.3gp	Contenedor de audio/video 3GPP	video/3gpp audio/3gpp if it doesn't contain video
.3g2	Contenedor de audio/video 3GPP2	video/3gpp2 audio/3gpp2 if it doesn't contain video
.7z	Archivo 7-zip	application/x-7z-compressed

DESARROLLO

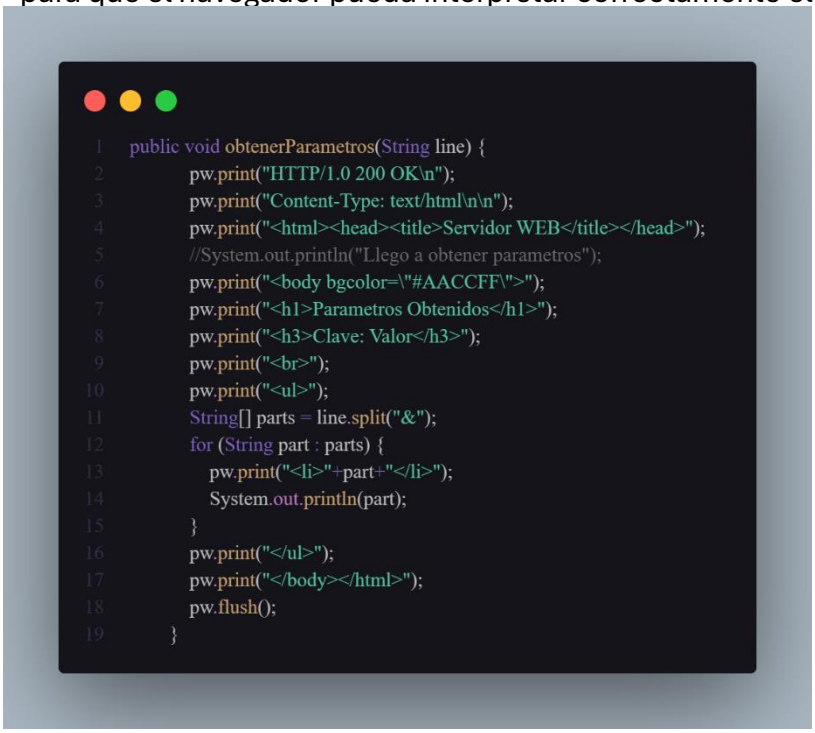
El método donde iniciamos el servidor, junto con la alberca de hilos:

```java
public ServidorWEB() throws Exception {
        System.out.println("Iniciando Servidor.....");
        this.ss = new ServerSocket(PUERTO);
        //creamos una alberca de hilos con 100 hilos
        this.threadPool = Executors.newFixedThreadPool(100);
        System.out.println("Servidor iniciado:---OK");
        System.out.println("Esperando por Cliente....");
        for (;;) {
            Socket accept = ss.accept();
            // Se crea un nuevo hilo con un Runnable
            threadPool.execute(new Manejador(accept));
            //new Thread(new Manejador(accept)).start();
        }
    }
```

Entre las partes más importantes del código se encuentran los métodos:

```java
public String getMimeType(String fileName) {
        if (fileName.endsWith(".html") || fileName.endsWith(".htm")) {
            return "text/html";
        } else if (fileName.endsWith(".pdf")) {
            return "application/pdf";
        } else if (fileName.endsWith(".jpg") || fileName.endsWith(".jpeg"))
{
            return "image/jpeg";
        } else if (fileName.endsWith(".png")) {
            return "image/png";
        } else if (fileName.endsWith(".gif")) {
            return "image/gif";
        } else if (fileName.endsWith(".css")) {
            return "text/css";
        } else if (fileName.endsWith(".js")) {
            return "application/javascript";
        } else if (fileName.endsWith(".txt")) {
            return "text/plain";
        } else if (fileName.endsWith(".ico")) {
            return "image/x-icon";
        } else {
            return "application/octet-stream"; // Tipo MIME por defecto (des
carga de archivos binarios)
        }
    }
```

getMimeType, el cual es usado para obtener el tipo MIME del archivo de la petición, para que el navegador pueda interpretar correctamente el recurso.



```
public void obtenerParametros(String line) {
        pw.print("HTTP/1.0 200 OK\n");
        pw.print("Content-Type: text/html\n\n");
        pw.print("<html><head><title>Servidor WEB</title></head>");
        //System.out.println("Llego a obtener parametros");
        pw.print("<body bgcolor=\"#AACCFF\">");
        pw.print("<h1>Parametros Obtenidos</h1>");
        pw.print("<h3>Clave: Valor</h3>");
        pw.print("<br>");
        pw.print("<ul>");
        String[] parts = line.split("&");
        for (String part : parts) {
           pw.print("<li>"+part+"</li>");
           System.out.println(part);
        }
        pw.print("</ul>");
        pw.print("</body></html>");
        pw.flush();
      }
```

obtenerParametos, ya que, en el GET Y POST, los probamos con un envó de formularios, los datos que vienen en el cuerpo de la petición (POST) o en la URL (GET), serán mostrados en un html.

```
public void getArch(String line) {
        int i;
        int f;
        if (line.toUpperCase().startsWith("GET") || line.toUpperCase().start
sWith("POST") || line.toUpperCase().startsWith("HEAD") || line.toUpperC
ase().startsWith("PUT") || line.toUpperCase().startsWith("DELETE") || lin
e.toUpperCase().startsWith("TRACE") || line.toUpperCase().startsWith("C
ONNECT") || line.toUpperCase().startsWith("PATCH")) {
          i = line.indexOf("/");
          f = line.indexOf(" ", i);
          FileName = line.substring(i + 1, f);
        }
        else {
          FileName = "";
          System.out.println("HTTP/1.0 501 Not Implemented\nhttps://htt
p.cat/status/501");

        }

      }
```

getArch que nos sirve para obtener el nombre de un archivo que venga en la petición.

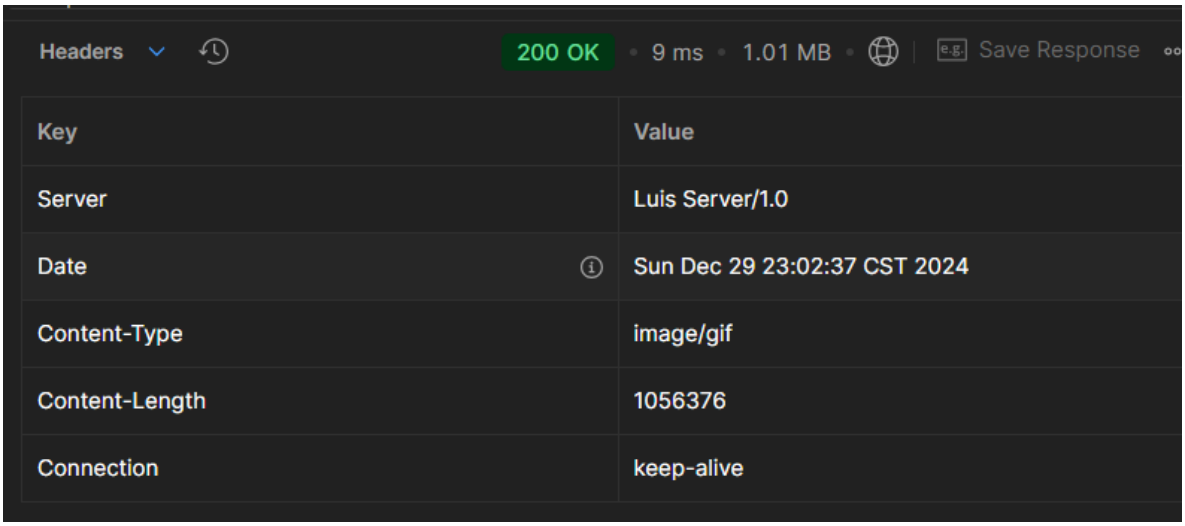
```java
public void SendAA(String fileName, String mimeType, int statusCode) {
        try {
            // Abrir el archivo solicitado
            BufferedInputStream bis2 = new BufferedInputStream(new FileInputStream(fileName));
            byte[] buf = new byte[65535];
            int b_leidos;

            // Obtener el tamaño del archivo
            File file = new File(fileName);
            int tam_archivo = (int) file.length();

            // Crear encabezado HTTP
            StringBuilder sb = new StringBuilder();
            sb.append("HTTP/1.0 " + statusCode  +"\n");
            sb.append("Server: Luis Server/1.0\n");
            sb.append("Date: " + new Date() + "\n");
            sb.append("Content-Type: " + mimeType + "\n");
            sb.append("Content-Length: " + tam_archivo + "\n");
            sb.append("Connection: keep-alive\n");
            sb.append("\n");

            System.out.println("RESPONSE HEADER");
            System.out.println("HTTP/1.0 " + statusCode);
            System.out.println("Server: Luis Server/1.0");
            System.out.println("Date: " + new Date());
            System.out.println("Content-Type: " + mimeType);
            System.out.println("Content-Length: " + tam_archivo);
            System.out.println("Connection: keep-alive");
            System.out.println();  // Línea en blanco para separar los encabezados del cuerpo

            // Enviar encabezado HTTP
            bos.write(sb.toString().getBytes());
            bos.flush();

            // Leer y enviar el contenido del archivo
            while ((b_leidos = bis2.read(buf, 0, buf.length)) != -1) {
                bos.write(buf, 0, b_leidos);
            }
            bos.flush();

            // Cerrar flujo de entrada
            bis2.close();
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
```

SendAA, que recibe el del archivo, el tipo MIME y el statuscode que se le va a mandar en el encabezado de respuesta al cliente.

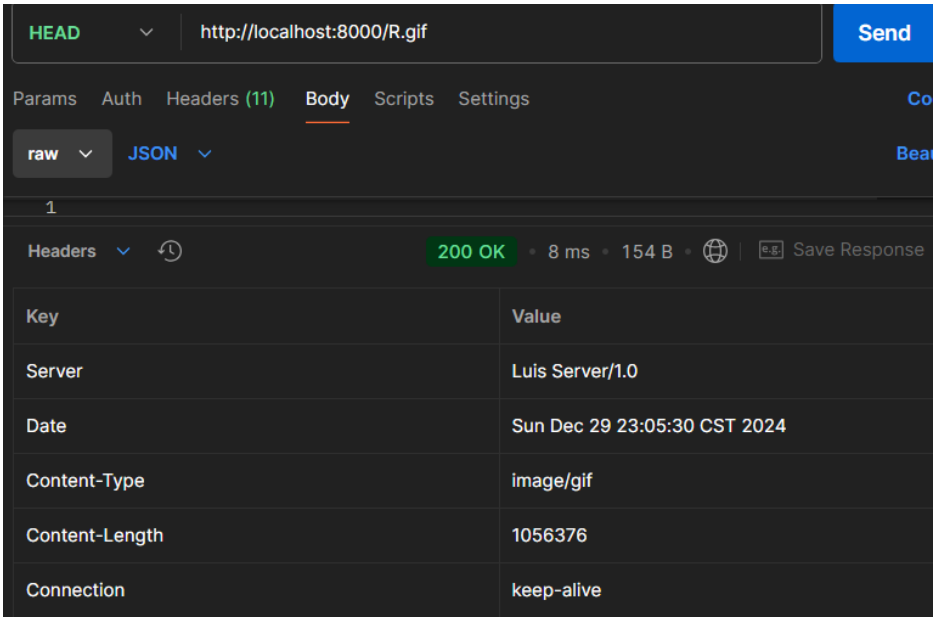
PRUEBAS

Para una petición GET de un recurso:

En el encabezado de respuesta:



Una petición HEAD a este mismo recurso nos devolvería lo mismo:



Una petición DELETE a un recurso

CONCLUSIONES

Velasco Jimenez Luis Antonio: En esta práctica, he podido ver como funciona el pool de hilos, así como su importancia y sus ventajas en la programación concurrente; por otro lado, he podido recordar y sobre todo reforzar el conocimiento de como y para que funciona cada método de las peticiones HTTP, ya que muchas veces, en el mundo de desarrollo se pueden confundir algunas, mas como PUT y PATCH.

Flores Morales Aldahir Andrés: Fuera del tema de hilos, que es con lo que estamos tratando, considero que, en esta práctica, de lo más importante a destacar son los tipos MIME, ya que al principio no teníamos algunos implementados, entonces al querer por ejemplo pedir un recurso con GET, en navegador o nuestro cliente rest (Postman), nos brindaba un formato raro, de datos binarios, ya que este cliente no sabía como interpretar los datos, debido a que no contábamos con un tipo MIME.

REFERENCIAS:

Lista completa de tipos MIME - HTTP | MDN

Métodos de petición HTTP - HTTP | MDN