Telematics Web Server (TWS)
Proyecto: HTTP
ST0255- Telemática
Versión 1.0

1. **Introducción.**
   En este proyecto se explora la aplicación de la capa de aplicación de la arquitectura TCP/IP. Al respecto, se abordará desde el estudio del protocolo HTTP desde una perspectiva de programación en red. Para esto se desarrollará e implementará un servidor web.

   En términos generales la función principal de un servidor web es la entrega de recursos (páginas html, imágenes, archivos de estilos, etc) web a un cliente que lo solicita (p.ej., web browser). Para esto, tanto el cliente como el servidor se comunican a través de un mismo protocolo que es HTTP.

   De esta forma el objetivo final es desarrollar e implementar un servidor web denominado Telematics Web Server – (TWS)) que soporte la versión HTTP/1.1

2. **Objetivo:** Implementar un servidor web que permita procesar peticiones del protocolo HTTP/1.1 y de esta forma entregar los diferentes recursos solicitados por el cliente.

3. **Alcance:**
   - Se requiere que usted implemente un servidor web (TWS) que permita procesar peticiones con la versión de protocolo HTTP/1.1.
   - Para lograr esto se debe emplear la API Berkeley Sockets. Tenga en cuenta que para esto existen diferentes recursos disponibles que puede revisar:
     - Linux Manual Page: socket() .
     - Beej's Guide to Network Programming
     - Ejemplo Client/Server en C.
   - Se requiere que su implementación deba ser capaz de analizar (parsing) tres tipos de métodos a nivel de un HTTPRequest: GET|HEAD|POST|. Para para revisar la estructura de este mensaje, se debe remitir a la especificación del estándar de HTTP/1.1 RFC 2616 (https://datatracker.ietf.org/doc/rfc2616/). Igualmente, de manera complementaria, puede revisar material adicional de apoyo, presentaciones, y referencias (libros) asociados al tema. A continuación, una breve descripción de lo que realizan los métodos.
     - **GET**: permite solicitar un recurso y la respuesta debe incluir los datos.
     - **HEAD**: este método pide una respuesta idéntica a la de una petición GET. La diferencia radica en que no va nada (datos) en el cuerpo de la respuesta (body).

- **POST**: este método se utiliza para enviar una entidad (datos) al recurso identificado en la URI y que se aloja en un servidor. Los datos enviados al servidor van en el cuerpo (body) de la petición.

- Su servidor debe ser capaz de manejar de manera robusta los errores. Para esto, debe soportar los siguientes mensajes de error:
  - **Código de respuesta: 200.** La petición enviada al servidor web fue procesada con éxito y se devolvió la respuesta al cliente.
  - **Código de respuesta: 400**. La petición solicitada por el cliente no pudo ser procesada. Este código debe ser enviado siempre que no se encuentre el recurso, no se entienda la petición, etc.
  - **Código de respuesta: 404.** Esta petición no puede ser procesada por el servidor web dado que no encontró el recurso solicitado. "Page/File Not Found"
- Su servidor debe soportar procesar peticiones de manera concurrente, para esto puede implementar una aproximación Thread Based para el manejo de la concurrencia. Esto quiere decir que puede utilizar hilos para el manejo de múltiples peticiones.
- Su servidor debe implementar el concepto de "logger". Esto con el fin de poder visualizar por la terminal todas las peticiones entrantes a nivel de HTTP, así como la respuesta que se envía a cada cliente.
- Su servidor TWS se debe ejecutar de la siguiente manera:
  - $./server <HTTP PORT> <Log File> <DocumentRootFolder>

  Donde:
    - HTTP PORT es el puerto de la máquina en la cual está corriendo su servidor web TWS.
    - LogFile es el archivo de log que se va a generar con toda la información concerniente a peticiones, errores, info, etc.
    - DocumentRootFolder es la carpeta donde se alojarán los diferentes recursos web
- Para efectos de prueba, se puede utilizar telnet al puerto en que este corriendo su servidor web, script escrito en Python, Postman, wireshark o un browser real.

4. **Casos de Prueba:**
   1. **Despliegue:** Para efectos de la prueba del funcionamiento de su servidor, este debe ser desplegado en la infraestructura de AWS.
   2. **Recursos web a solicitar:**
      i. **Caso 1.** Página web con algunos hipertextos y una imagen.
      ii. **Caso 1.** Página web con algunos hipertextos y múltiples imágenes.
      iii. **Caso 3:** Página web que contiene un solo archivo de aproximadamente un tamaño de 1MB.
      iv. **Caso 4:** Página web que contiene múltiples archivos y que aproximadamente tiene un tamaño de 1MB

5. **Guías para el trabajo:**
    1. **Equipo de trabajo:** El proyecto debe ser desarrollado en grupos de 3 personas como máximo. **NO** debe ser desarrollado de manera individual.
    2. **Lenguaje de Programación:** El desarrollo de su servidor web debe ser en C, C++ o Python.
        i. **Nota:** En caso de ser Python el lenguaje de su elección solo puede utilizar las siguientes librerías: **sys, os, time, socket, Thread**
    3. **Documentación:** La documentación se debe incluir en el repo en un archivo README.md. En este archivo se requiere que usted incluya los detalles de implementación donde como mínimo se esperan las siguientes secciones:
        i. Introducción.
        ii. Desarrollo
        iii. Conclusiones
        iv. Referencias
    4. **Fecha de entrega:** El proyecto se debe culminar el 18 de abril de 2023 hasta las 23:59.
    5. **Mecanismo de entrega:** Por el buzón de Interactiva virtual se debe realizar la entrega.
        i. La entrega debe incluir un enlace a un repositorio en github.
        ii. A partir de esta fecha y hora, no se puede realizar ningún commit al repositorio.

6. **Referencias**

1. https://datatracker.ietf.org/doc/rfc2616/
2. https://www.wireshark.org/
3. https://www.postman.com/
4. https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/
5. https://beej.us/guide/bgnet/

Última actualización: 18 de marzo.