Taller: Servidor HTTP

En este taller se programará un servidor HTTP muy sencillo, con el objetivo de comprender y manipular los distintos elementos de un Request y un Response HTTP. El software **DEBE** programarse en el lenguaje de programación Python. Además, **está prohibido utilizar librerías externas para parsear el protocolo HTTP**. No obstante, sí puede buscar libremente recursos o tutoriales en Internet si así lo requiere.

No olvide descargar/revisar la ***plantilla inicial*** que incluye los casos de prueba automáticos. Para más detalles vea el archivo testing.pdf.

Sobre la entrega

La entrega se debe realizar mediante Aula Virtual. Debe consistir en una carpeta comprimida con todos los archivos necesarios para la correcta ejecución y revisión del taller. La fecha de entrega es el día Domingo 25 de Marzo hasta las 23:55hrs.

Especificaciones

1. Su software debe poder ejecutarse en Linux (Ubuntu Server 16.04) o bien en Windows 10. Usted debe proveer un script iniciarServidor.sh (o iniciarServidor.bat en Windows) que ejecute el servidor HTTP de manera que éste acepte conexiones en la URL http://localhost:9100.

2. Puede asumir que todos los request serán sintácticamente válidos, según los formatos de mensajes visto en clases. Es decir, un request tiene la forma:

```
METHOD path HTTP-VERSION
HEADER-1: VALUE-1
HEADER-2: VALUE-2
HEADER-N: VALUE-N

CONTENT
```

Donde los headers y el contenido son opcionales, es decir, podrían estar o no estar.

3. El servidor debe considerar una carpeta de nombre documentRoot, que es la raíz para obtener recursos, que serán sólo páginas en HTML. Dentro de documentRoot puede haber una cantidad arbitraria de (sub-)carpetas y archivos HTML.

4. Cuando el servidor recibe un request debe funcionar de la siguiente forma:

    - Si existe el archivo documentRoot/path, debe retornar una respuesta con código y mensaje 200 OK, y el contenido debe ser el contenido del archivo encontrado.

    - Si no existe el archivo documentRoot/path, debe retornar una respuesta con código y mensaje 404 Not Found. La respuesta no debe tener contenido.

- En ambos casos, su respuesta debe tener un header de nombre X-ResponseEcho. El valor de este header consiste en la **codificación en formato JSON** de un diccionario con la información del request recibido.

Ejemplos

Considerando el request:

```
GET /myapp/index_3.html HTTP/1.1
Host: localhost:9100
Accept: text/html
Accept-Language: es-Es
```

Y asumiendo una respuesta en el caso que el archivo documentRoot/myapp/index_3.html SÍ EXISTE. La respuesta debe ser:

```
HTTP/1.1 200 OK
X-RequestEcho: {"path": "/myapp/index_3.html", "protocol": "HTTP/1.1",
"method": "GET", "headers": {"Accept": "text/html", "Accept-Language": "es-
ES", "Host": "localhost:9100"}}

<html>
Hola index 3
</html>
```

Importante:

- El texto JSON va en una sola línea, por muy larga que sea.
- Los campos específicos del diccionario son: "path", "protocol", "method", y "headers".
- El campo "headers" es un sub-diccionario en JSON, con cada uno de los headers recibidos en el request; los demás campos son strings.
- No importa el orden de los campos en el texto JSON.
- Puede haber headers "de más", si fueron incluidos por el cliente al momento de hacer las pruebas. Esto se considerará correcto.

Evaluación

Para la evaluación usaremos la técnica de //Behavior-Driven Development//. Para poder verificar si su implementación satisface lo requerido siga la instrucciones archivo README.md.

El total de puntaje es de 6 puntos (+ punto base). Considerando los escenarios de prueba públicos y descargables, los criterios y puntajes son los siguientes:

1. (1 puntos) El script iniciarServidor.sh (o iniciarServidor.bat) existe y funciona correctamente. Al ejecutarlo se levanta el servidor HTTP en el puerto 9100.

2. (1 puntos) Bajo los casos de prueba aplicados, el servidor responde correctamente con los status y mensajes 200 OK o 404 Not Found, según los archivos buscados existan o no en la carpeta documentRoot.

3. (1 punto) El servidor envía correctamente el HTML en la respuesta, en caso de las respuestas con status 200.

4. (2 puntos) El servidor envía correctamente el header X-RequestEcho con los datos que corresponden a los enviados originalmente en el request.

5. (1 punto) El servidor pasa los *casos de prueba privados*, que serán aplicados por el profesor.