Universidad Nacional Autónoma De México
Facultad De Ciencias
Redes de Computadoras, 2019-I

Práctica Extra:
Programación de un cliente de HTTP v1.1

Fecha de entrega:
Viernes 21 de septiembre del 2018.

Integrantes:

Sánchez Morales, Rodrigo Alejandro.

Desarrollo:

1. Investigue para qué se usan los métodos *HEAD*, *GET*, *POST*, *PUT*, *DELETE*.

> *HTTP* define un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado. Aunque estos también pueden ser sustantivos, estos métodos de solicitud a veces son llamados *HTTP verbs*. Cada uno de ellos implementan una semántica diferente, pero algunas características similares son compartidas por un grupo de ellos: ejemplo un request method puede ser *safe*, *idempotent*, o *cacheable*.
>
> 1. **HEAD**: El método *HEAD* pide una respuesta idéntica a la de una petición *GET*, pero sin el cuerpo de la respuesta.
> 2. **GET**: El método *GET* solicita una representación de un recurso específico. Las peticiones que usan el método *GET* sólo deben recuperar datos.
> 3. **POST**: El método *POST* se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.
> 4. **PUT**: El modo *PUT* reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.
> 5. **DELETE**: El método *DELETE* borra un recurso en específico.
>
> https://developer.mozilla.org/es/docs/Web/HTTP/Methods

2. Investigue los diferentes códigos de estado que usa *HTTP*, enliste las categorías.

> Los códigos de estado de respuesta *HTTP* indican si se ha completado satisfactoriamente una solicitud *HTTP* específica. Las respuestas se agrupan en cinco clases: respuestas informativas, respuestas satisfactorias, redirecciones, errores de los clientes y errores de los servidores.
>
> 1. **Respuestas informativas**: *100 Continue* Esta respuesta provisional indica que todo hasta ahora está bien y que el cliente debe continuar con la solicitud o ignorarla si ya está terminada.
> 2. **Respuestas satisfactorias**: *200 OK*. La solicitud ha tenido éxito. El significado de un éxito varía dependiendo del método *HTTP*.
> 3. **Redirecciones**: *301 Moved Permanently*. Este código de respuesta significa que la *URI* del recurso solicitado ha sido cambiado. Probablemente una nueva *URI* sea devuelta en la respuesta.
> 4. **Errores de cliente**: *404 Not Found*. El servidor no pudo encontrar el contenido solicitado. Este código de respuesta es uno de los más famosos dada su alta ocurrencia en la web.
> 5. **Errores de servidor**: *505 HTTP Version Not Supported*. La versión de *HTTP* usada en la petición no está soportada por el servidor.
>
> https://developer.mozilla.org/es/docs/Web/HTTP/Status

3. Investigue el uso del campo *encoding*.

> La cabecera *content-encoding* es usada para comprimir el *media-type*. Cuando está presente, su valor indica qué codificación de contenido adicional ha sido aplicada al cuerpo de la entidad. Permite al cliente saber cómo decodificar para obtener el *media-type* referido por la cabecera *content-type*. Se recomienda comprimir los datos tanto como sea posible y por lo tanto utilizar este campo, pero algunos tipos de recursos, como imágenes *JPEG*, ya están comprimidos. A veces, el uso de compresión adicional no reduce el tamaño de la petición e incluso puede hacer que la petición sea más larga.
>
> https://developer.mozilla.org/es/docs/Web/HTTP/Headers/Content-Encoding

4. Investigue el uso del campo *connection*.

> La cabecera general de conexión controla si la conexión de red permanece o no abierta después de que finaliza la transacción actual. Si el valor enviado es *keep-alive*, la conexión es persistente y no está cerrada, lo que permite realizar peticiones posteriores al mismo servidor.
>
> https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Connection