# **Ejercicio 2:**

# **1**

Un servidor http es un programa que permanece a la escucha de las peticiones de dispositivos clientes por medio de los puertos del ordenador en que se encuentra, haciendo uso de protocolos de transmisión y control de información como tcp, entre otros.

El servidor http mediante protocolo http o (protocolo de transferencia de hipertexto), al recibir una petición la procesa por medio de aplicaciones del lado del servidor y luego envía la respuesta del procesamiento al cliente

# **2**

Los verbos de http son métodos de petición representados en el encabezado de una petición http Por ejemplo GET / HTTP/1.1

A través de estos verbos un dispositivo cliente indica la acción que se desea realizar a un recurso determinado del servidor

Estos recursos suelen ser descriptos por la url en la que se realiza la petición un ejemplo de petición puede ser: DELETE /alumno/david/legajo HTTP/1.1 donde estaría solicitando eliminar el recurso o el legajo de un alumno en un sitio web

De esta forma se puede solicitar el recurso (GET), enviar información para modificarlo (POST), etc.

Los verbos más conocidos son:

GET: realiza solicitud de recursos, las peticiones realizadas con este método solo deben recuperar datos

HEAD: solicita una respuesta idéntica a get, pero sin cuerpo en su respuesta

POST: usada para enviar información a un recurso del servidor, causando un cambio en su estado

PUT: reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.

DELETE: Borra un recurso específico.

CONNECT: Establece un túnel hacia el servidor identificado por el recurso, un ejemplo de uso es pretender realizar una conexión a un dominio mail a través de un protocolo de smtp.

OPTIONS: Usado para describir las opciones de comunicación para el recurso de destino.

TRACE: Realiza una prueba de bucle para testear el retorno de mensaje a lo largo de la ruta al recurso de destino.

PATCH: Utilizado para aplicar modificaciones parciales a un recurso.

# **3**

Una request o petición es un mensaje del tipo http enviado por un cliente al servidor, para pedir el inicio de una acción sobre un recurso del servidor, suele realizar a través de verbos http para indicar al servidor que acción tomar

Una respuesta o response: es la respuesta del servidor a la petición de un cliente, en esta se suele indicar el estado de la acción y el contenido que se esperaba como por ejemplo fragmentos de página web, sí que este fue solicitado

# **4**

queryString o cadena de consulta en una url hace referencia al segmento opcional de la url escrito luego del carácter ? para realizar envió de datos al servidor al momento de realizar una petición

La estructura de la cadena de consulta puede ser de \*nombre=valor separados por el carácter & (ejemplo: var1=val1&var2=val2…)

Luego, estos datos pueden ser recuperados por el servidor mediante método GET.

Un ejemplo completo puede ser https://christianfrancopropiedades.com/venta?PROPIEDAD=CN01

# **5**

ResponseCode o códigos de estado de respuesta HTTP indican el estado de la petición realizada al servidor, cada respuesta se asocia a un rango de variables aceptado, estos son:

Respuestas informativas (100–199),

Respuestas satisfactorias (200–299),

Redirecciones (300–399),

Errores de los clientes (400–499),

Errores de los servidores (500–599).

# **6**

En un método get los datos se envían al servidor mediante una queryString, de forma que estos pueden verse y modificarse si se configura la información de la url

En un método post los datos se envían en segundo plano, ocultos al usuario para que este no los vea ni modifique permitiendo trabajar con información sensible

# **7**

Al acceder a una página el navegador utiliza el verbo GET, para obtener el cuerpo o vista del sitio accedido

# **8**

JSON: es un formato de texto para el intercambio de datos entre lenguajes, su sintaxis es un subconjunto de la notación literal de objetos de javascript definiendo entonces:

Datos: Números, Cadenas, Booleanos, null.

Array: Lista ordenada de cero o más valores de cualquier tipo. separados por comas y encerrado en corchetes. Ejemplo ["juan","pedro","jacinto"]

Objetos: Colecciones de pares de la forma <nombre>:<valor> separados por comas y encerrados en llaves. Donde nombre es una cadena y el valor puede ser de cualquier tipo.

Ejemplo:

{"departamento":8,

"nombredepto":"Ventas",

"director": "juan rodriguez",

"empleados":[

{"nombre":"Pedro",

"apellido":"Fernandez"},

{"nombre":"Jacinto",

"apellido":"Benavente"}

]

}

XML: es un subconjunto del estándar SGML(Estándar Generalised Mark-up Language), xml permite definir lenguajes de marcado, de esta forma definir etiquetas personalizadas para descripción y organización de datos que puede ser leído por cualquier lenguaje

Su sintaxis se compone por:

. ELEMENTOS: Pieza lógica del marcado, se representa con una cadena como dato encerrada entre etiquetas

Los elementos pueden contener atributos.

. INSTRUCCIONES: Ordenes que comienzan por <? Y terminan por ?>, usadas por la aplicación que procesa Ej <?xml-stylesheet type=“text/css” href=“estilo.css”>

. COMENTARIOS: Información que no forma parte del documento. Comienzan por <!-- y terminan por -->.

. DECLARACIONES DE TIPO: Especifican información acerca del documento ej <!DOCTYPE persona SYSTEM “persona.dtd”>, consisten en Secciones llamadas CDATA que se trata de un conjunto de caracteres que no deben ser interpretados por el procesador

Ej <![CDATA[ Aquí se puede meter cualquier carácter, como <, &, >, ... Sin que sean interpretados como marcación]]>

La estructura de un archivo xml se suele dividir en:

En prologo con declaraciones e instrucciones y un cuerpo con elementos que contienen datos en atributos

EJ: (prologo)

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE persona SYSTEM "persona.dtd">

(cuerpo)

<persona>

<nombre>Luis</nombre>

<apellido>Perez</apellido>

</persona>

# **9**

El estándar soap es un pórtalo de acceso a objetos basado en xml, este representa la petición y respuesta de mensajes entre clientes y servidores asi como el estado de esta peticion

Soap libera al cliente de tener que serializar y deserialziar los mensajes xml enviados a servidores (para servidores que utilizan xml como estándar de comunicación), pues soap permite manejar estas acciones mediante el kit de herramientas o bibliotecas soap pertenecientes al lenguaje con el que se trabaja

Para utilizar el estándar se indica en un documento xml el servicio que se desea usar, este documento es conocido como descripción del servicio web, luego soap te permite conocer y llamar a los métodos de los servicios remotos a los que se conecta como si fueran locales.

La desventaja de soap recae en los tiempos de ejecución pues los mensajes xml más detallados y pesados que otros lenguajes

# **10**

El estándar REST Full hace referencia a programas basados en la arquitectura rest, estos funcionan de forma similar a soap, en este estándar se transportan datos mediante protocolo http usando sus verbos para realizar consultas del tipo GET, POST, PUT, DELETE, PATCH, asi como también utiliza los códigos de respuesta 404,200,204,409.

Rest permite transmitir tipos de datos XML, JSON, Binarios (imágenes, documentos), Text,etc. Aunque en la mayoría de los casos se envían datos del tipo JSON.

# **11**

Los header request son los encabezados pertenecientes a los request y respones propios de una solicitud a un servidor http, en estos se almacena información sobre el estado de la petición, la acción a realizar mediante verbos http, la respuesta del servidor, el estado de la coeducación y el tipo de contenido que almacena la respuesta de la solicitud, un ejemplo puede ser

HTTP/1.1 200 OK

Date: Tue, 10 Nov 2015 11:56:29 GMT

Server: Apache/2

X-Powered-By: PHP/5.5.17

P3P: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"

Expires: Mon, 1 Jan 2001 00:00:00 GMT

Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0

Pragma: no-cache

Last-Modified: Tue, 10 Nov 2015 11:56:30 GMT

Vary: Accept-Encoding,User-Agent

Content-Encoding: gzip

Content-Length: 9237

Keep-Alive: timeout=1, max=100

Connection: Keep-Alive

Content-Type: text/html; charset=utf-8

(obtenido de https://www.aprenderaprogramar.com/index.php?option=com\_content&view=article&id=926:cabeceras-http-headers-que-son-y-para-que-sirven-status-authorization-user-agent-referer-cu01208f&catid=83&Itemid=212)

El key Content-type de un header se utiliza para indicar el tipo de contenido que almacena la solicitud http,

En esta key es importante que para las solicitudes de tipo POST, PATCH, PUT, y DELETE, los parámetros que no se incluyen en la URL deben codificarse como JSON con un Content-Type de 'application/json':