Respuestas Teóricas

Ejercicio 2

1. ¿Qué es un servidor HTTP?

Un servidor HTTP tiene como función principal guardar, procesar y enviar recursos a clientes utilizando el protocolo HTTP(Hypertext Transfer Protocol) para comunicarse. HTTP es un protocolo cliente-servidor. El cliente hará una solicitud al servidor mediante HTTP. Cuando la petición llega al servidor, este envía una respuesta también a través de HTTP.

2. ¿Qué son los verbos HTTP? Mencionar los más conocidos

Los verbos HTTP son un conjunto de métodos de petición definidos por el protocolo HTTP que se utilizan para indicar la acción que se desea realizar para un recurso determinado.

GET: Devuelve el recurso solicitado. Debe utilizarse para solicitar un recurso específico y no para cambiarlo en ninguna forma.
HEAD: Hace una petición igual a la de GET, pero la respuesta no tendrá incluida el cuerpo.
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.
PUT: Coloca un recurso especifico. Si el recurso ya existe lo reemplaza, si no existe lo crea.
DELETE: Elimina un recurso especifico.
PATCH: utilizado para realizar modificaciones a un recurso.
CONNECT: Establece un túnel hacia el servidor que contiene el recurso.
OPTIONS: Es utilizado para establecer las opciones de comunicación para el recurso de destino.
TRACE: Realiza una prueba de bucle de retorno de mensaje. Esto permite conocer el path hacia el recurso de destino.

3. ¿Qué es un request y un response en una comunicación HTTP? ¿Qué son los headers?

Requests y responses son los dos tipos de mensajes HTTP que existen.
En una comunicación HTTP un cliente realiza una “request” o solicitud al servidor y el servidor envía una “response” o respuesta.
Una request contiene los siguientes elementos:

- Un verbo HTTP
- El request URI o path del recurso solicitado (sin aclarar el protocolo, el dominio o el puerto TCP)
- La versión del protocolo HTTP
- Headers(opcionales)
- Body(para algunos métodos, como POST

Una responce contiene los siguientes elementos:

- La versión del protocolo HTTP
- Un status code, que indica si la request fue exitosa o no y porque
- Un status messege,
- HTTP headers.
- Un body (opcional para algunos metodos)

Los headers HTTP proveen información adicional requerida acerca del request o del response o acerca del objeto enviado en el body del mensaje. Un header está compuesta por su nombre (no sensible a las mayusculas) seguido de dos puntos ':', y a continuación su valor Hay cuatro tipos de headers: General-header(pueden aplicar a requests o responces), Client Request-header (solo para requests) Server Response-header(solo para responces) y Entity-header (definen meta-información acerca del cuerpo de la entidad o del recurso especificado en el request).

4. ¿Qué es un queryString? (En una url)

Un query string es utilizado para enviar información adicional al servidor en una request. Es opcional incluirlas y solo se usan en el método GET. Está formado por pares de nombre/valor.La querry string se coloca al final de la URL y comienza a partir del primer signo "?". Su sintaxis suele ser pares de "nombre" y "valor" separados por un símbolo "&"

Ejemplo: www.ejemplo.com/index.html?productID=22&color=blue

En este ejemplo se pasan al servidor, mediante la URL, los pares nombre/valor: productID = 22 y color = blue

5. ¿Qué es el responseCode? ¿Qué significado tiene los posibles valores devueltos?

El response code es una parte del mensaje de tipo response. Indica si se realizó o no sactifactoriamente la request y porque.

Suelen dividirse en 5 clases:
Clase 1xx – Informational: indicant que la request fue recibida y se encuentra en procesamiento
Ejemplo: Status code 100 - Continue
Clase 2xx – Success: indican que la request fue correctamente recibida, entendida y aceptada.
Ejemplo: Status code 200 - OK
Clase 3xx – Redirection: Indica que la request fue recibida pero se necesitan más acciones para completarse
Ejemplo: Status code 301 – Moved Permanently
Status code 302 – Moved Temporarily:
Clase 4xx – Client error: Indica que la request contiene sintaxis incorrecta o no puede ser cumplida.
Ejemplo: Status code 403 – Forbidden
Status code 404 – Not Found
Clase 5xx – Server error: indica que el servidor no pudo cumplir la request, aunque esta parecía ser válida.
Ejemplo: Status code 500 – Internal Server Error
Status code 501 – Not Implemented
Status code 503 – Service Unavailable
Status code 550 – Permission denied

6. ¿Cómo se envía data en un Get y cómo en un POST?

En un **GET** se envía data utilizando un queryString que ya se explicó en la pregunta 4. Esto tiene como desventaja que los datos enviados son visibles en la URL y que también el largo del queryString está limitado por el largo máximo de una URL.
En un **POST** se envían también pares de "nombre" y "valor" pero se lo hace dentro del mensaje HTTP que solicita una página al servidor. Esto elimina las dos desventajas mencionadas del método GET.

7. ¿Qué verbo http utiliza el navegador cuando accedemos a una página?

Cuando accedemos a una página, el navegador utiliza el verbo GET.

8. Explicar brevemente qué son las estructuras de datos JSON y XML dando ejemplo de estructuras posibles.

JSON y XML son notaciones que permiten expresar objetos en texto plano. Se utilizan principalmente para el intercambio y almacenamiento de datos ya es que son independientes del lenguaje de programación utilizado. Ambos son autodescriptivos y jerárquicos pero JSON tiene como ventajas que: es más corto, puede utilizar arrays y puede ser parseado por una función standard de javascript(XML en cambio debe ser parseado por un parser XML)

Estructura:

Ejemplo de un objeto expresado en **JSON**:

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

El mismo objeto representado en **XML**:

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

9. Explicar brevemente el estandar SOAP

SOAP o (Simple Access standard protocol) es un standard para el intercambio de data estructurada a través de dos nodos independientemente del lenguaje o plataforma que estos utilicen. Utiliza el formato XML y está construido sobre el protocolo HTTP. SOAP permite a un cliente conectarse fácilmente a servicios remotos e invocar métodos remotos por lo que es muy utilizado en web services y APIs.
Cada mensaje está compuesto por: un envelope(indica que es un mensaje SOAP), un header(opcional, provee información adicional), y un body(el mensaje propiamente dicho a transmitir).

10. Explicar brevemente el estandar REST Full

REpresentational State Transfer o REST es un estilo de arquitectura que utiliza el protocolo HTTP. En una arquitectura REST el REST server simplemente provee acceso a los recursos y el REST client accede y modifica estos recursos. Cada recurso es identificado por una URI. El REST client envía una request por medio de un método como GET, POST, PUT or DELETE. El REST server envía una respuesta que por lo general se encuentra en formato JSON.
Es muy utilizada para la creación de web services.
REST utiliza menos ancho de banda y es más simple y flexible que el standard SOAP, pero es menos robusto.

11. ¿Qué son los headers en un request? ¿Para qué se utilizar el key Content-type en un header?

En un request, los headers contienen información adicional diversa sobre la petición HTTP y sobre el navegador. Estos headers son opcionales. Algunos ejemplos de headers utilizadosen un request son: accept, Accept-Encoding, Cookie, User-Agent, Referer, entre otras.
El key "Content-Type" indica el tipo de media de la representación asociada, por lo que puede ser utilizada tanto por el cliente como por el servidor para identificar el formato de los datos en el mensaje enviado y así lograr que la otra parte interprete correctamente la información.