TEMA 4: HTTP

Módulo

Despliegue de aplicaciones web

para los ciclos

Desarrollo de aplicaciones web

4.1 Introducción a la WWW

La ***WWW*** (World Wide Web - Telaraña mundial) permite el acceso a recursos electrónicos y aplicaciones que se encuentran distribuidos en servidores por todo el mundo. Esos recursos se identifican a través de ***URIs*** o ***URLs***.

La ***WWW*** ha sido desarrollada por el ***CERN*** (Centro Europeo de Investigación Nuclear) en 1989 y es controlada por una comunidad internacional llamada ***W3C*** (World Wide Web Consortium - consorcio de la ***WWW***). Entre otras muchas cosas, la ***W3C*** desarrolla estándares web como ***HTML***, ***XHTML***, ***CSS***, ***XML***, etc.

Un documento de ***hipertexto*** es un documento que contiene enlaces que permiten saltar de unos documentos a otro. Los documentos de hipertexto que están disponibles a través de la ***WWW*** son conocidos como ***páginas web***. Están escritos en lenguajes específicos como ***XHTML***, ***CSS***, ***JavaScript***, ***Flash***, etc e incluyen contenido estático y dinámico.

Un ***sitio web*** es el conjunto de páginas relacionada y accesibles a partir de un mismo dominio ***DNS***.

Una ***aplicación web*** es una aplicación distribuida a la que se accede desde un navegador web.

Los ***navegadores web*** son programas que se utilizan para acceder a la ***WWW***. Aunque son clientes de muchos protocolos, fundamentalmente trabajan como clientes del protocolo ***HTTP***. Ejemplos de navegadores web son: ***Mozilla Firefox***, ***Chrome***, ***Opera***, etc. Para acceder a un recurso con un navegador web, se le proporciona una ***URL*** o ***URI*** y el navegador la localizará en la ***WWW***. Lo primero que tendrá que hacer es usar un ***DNS*** para traducir ese dominio a una ***IP***. Fíjate en la imagen de la página 5.

Los ***servidores web*** son programas que ponen páginas web a disposición de quien las quiera visitar. Atienden peticiones (en el puerto ***80***) bajo el protocolo ***HTTP*** y contestan enviando los recursos solicitados. Ejemplos de servidores web son ***Apache*** e ***IIS7***.

4.1.1 URL y URI

Una **URL** (Uniform Resource Locator, Localizador uniforme de recursos) es el nombre completo de un recurso de la web. Naturalmente ese nombre ha de ser único para cada recurso.

La ***URL*** se compone de cuatro partes:

El ***esquema*** especifica el protocolo que hay que usar para acceder al recurso (***http***, ***https***, ***ftp***, ***file***). Se escribe con minúsculas seguido de ***://*** salvo en el caso de los protocolos ***news*** y ***mailto***, en los que se omiten las barras. Ejemplos de esquemas:

```
http://
ftp://
mailto:
```

La segunda parte de una ***URL*** indica el ***servidor*** donde se aloja el recurso (por ejemplo, ***www.google.com*** o ***iesmurgi.org***)

A continuación pondremos la ***ruta*** o ***path***, que es la carpeta donde se encuentra y finalmente el ***nombre*** del recurso.

esquema	servidor	ruta	nombre
http://	iesjulioverne.es	/help/	ayuda.html
La URL será: **http://iesjulioverne.es/help/ayuda.html**			
file://	/	/help/	ayuda.html
La URL será: **file:///help/ayuda.html**			

La descripción que hemos visto corresponde a un **URL absoluto**. Un **URL relativo** sólo necesita la ruta y el nombre. En este caso el recurso deberá encontrarse en el mismo equipo que el documento desde el que se hace referencia.

Una ***URI*** (Uniform Resource Identifier, identificador uniforme de recursos) permite situarse en una parte concreta de un recurso, por ejemplo, al principio de la página 4 de un documento. Para construir una ***URI*** sólo hace falta añadir a una ***URL*** el símbolo almohadilla (***#***) y el nombre asignado al fragmento de recurso. Ejemplo:

```
http://iesmurgi.org/help/ayuda.html#apendice
```

Naturalmente, dentro del documento ***ayuda.html***, en el sitio adecuado ha sido establecido ***apendice*** como nombre de la sección.

4.2 El protocolo HTTP

Hypertext Transfer Protocol o ***HTTP*** (en español protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la ***World Wide Web (WWW)***. Fue propuesto por ***Tim Berners-Lee*** en los años 80, atendiendo a las necesidades de un sistema global de distribución de información como el ***World Wide Web***. y desarrollado por el ***World Wide Web Consortium*** (***W3C***) y la ***Internet Engineering Task Force*** (***IETF***), colaboración que culminó en 1999 con la publicación de una serie de ***RFC***, el más importante de ellos es el ***RFC 2616*** que especifica la versión 1.1. ***HTTP*** define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse.

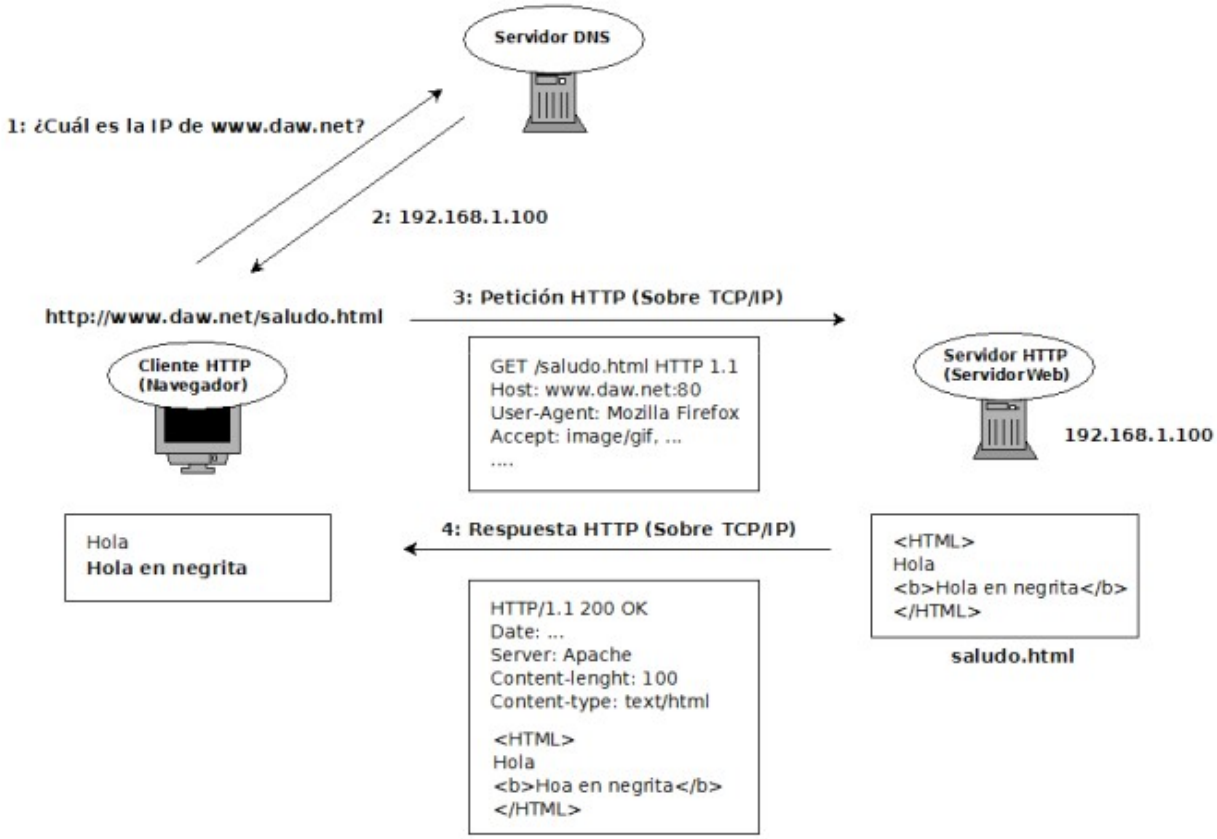
Es un protocolo sigue el esquema petición-respuesta entre un ***navegador web*** y un ***servidor HTTP***. A la información transmitida se la llama recurso y se la identifica mediante un localizador uniforme de recursos (***URL***). Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje (al puerto ***80***) con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento ***HTML***, fichero multimedia o aplicación CGI) es conocido por su ***URL***.

Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

- Un usuario accede a una ***URL***, seleccionando un enlace de un documento ***HTML*** o introduciéndola directamente en la barra de direcciones del cliente Web.

- El cliente Web descodifica la ***URL***, separando sus diferentes partes. Así identifica el protocolo de acceso, el dominio o ***IP*** del servidor, el posible puerto opcional (el valor por defecto es ***80***) y el objeto requerido del servidor.
- Se abre una conexión ***TCP/IP*** con el servidor, llamando al puerto ***TCP*** correspondiente.
  Se realiza la petición. Para ello, se envía el comando necesario (***GET***, ***POST***, ***HEAD***, …), la dirección del objeto requerido (el contenido de la ***URL*** que sigue a la dirección del servidor), la versión del protocolo ***HTTP*** empleada (casi siempre ***HTTP/1.0***) y un conjunto variable de información, que incluye datos sobre las capacidades del browser, datos opcionales para el servidor,…
- El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato ***MIME*** de la información de retorno, seguido de la propia información.
- Se cierra la conexión ***TCP***.



Este proceso se repite en cada acceso al servidor ***HTTP***. Por ejemplo, si se recoge un documento ***HTML*** en cuyo interior están insertadas cuatro imágenes, el proceso anterior se repite cinco veces, una para el documento ***HTML*** y cuatro para las imágenes.

HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener estado. Para esto se usan las ***cookies***, que es información que un servidor puede almacenar en el sistema cliente. Esto le permite a las aplicaciones web instituir la noción de "sesión", y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.

4.2.1 Versiones

HTTP ha pasado por múltiples versiones del protocolo, muchas de las cuales son compatibles con las anteriores. El ***RFC 2145*** describe el uso de los números de versión de ***HTTP***. El cliente le dice al servidor al principio de la petición la versión que usa, y el servidor usa la misma o una anterior en su respuesta.

0.9

Obsoleta. Soporta sólo un comando, ***GET***, y además no especifica el número de versión ***HTTP***. No soporta cabeceras. Como esta versión no soporta ***POST***, el cliente no puede enviarle mucha información al servidor.

HTTP/1.0 (mayo de 1996)

Esta es la primera revisión del protocolo que especifica su versión en las comunicaciones, y todavía se usa ampliamente, sobre todo en servidores proxy.

HTTP/1.1 (junio de 1999)

Versión actual; las conexiones persistentes están activadas por defecto y funcionan bien con los proxies. También permite al cliente enviar múltiples peticiones a la vez por la misma conexión (pipelining) lo que hace posible eliminar el tiempo de Round-Trip delay por cada petición.

HTTP/1.2

Los primeros borradores de 1995 del documento *PEP — an Extension Mechanism for HTTP* (el cuál propone el Protocolo de Extensión de Protocolo, abreviado PEP) los hizo el World Wide Web Consortium y se envió al Internet Engineering Task Force. El PEP inicialmente estaba destinado a convertirse en un rango distintivo de HTTP/1.2.3 En borradores posteriores, sin embargo, se eliminó la referencia a HTTP/1.2. El RFC 2774 (experimental), *HTTP Extension Framework,* incluye en gran medida a PEP. Se publicó en febrero de 2000.

4.2.2 Ejemplo de un diálogo HTTP

INICIAR UNA SESIÓN TELNET DESDE LINUX

Lo normal es que ***Linux*** disponga de un cliente ***telnet*** sin necesidad de instalar nada. Para conectar con el servidor ***telnet*** escribimos:

```
telnet IPservidor
```

o

```
telnet DominioServidor
```

Por ejemplo:

```
telnet 192.168.20.65
```

Después de un momento nos pedirá que nos autentiquemos en el equipo destino:

```
administrador@administrador-TECRA-R950:/etc/apt$ telnet 192.168.43.222
Trying 192.168.43.222...
Connected to 192.168.43.222.
Escape character is '^]'.
Ubuntu
profeserver login: administrador
Password:
Last login: Wed [illegible] 17:15:23 UTC 2018 from administrador-TECRA-R950 on pts/0
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Nov  7 17:34:31 UTC 2018

  System load:  0.0               Processes:             94
  Usage of /:   11.2% of 39.12GB  Users logged in:       1
  Memory usage: 7%                IP address for enp0s3: 192.168.43.222
  Swap usage:   0%


83 packages can be updated.
3 updates are security updates.


administrador@profeserver:~$
```

Desde el equipo cliente (observa el prompt del equipo ***TECRA*** en la primera línea) solicitamos la conexión ***telnet*** con el equipo servidor (***192.168.43.222***). Una vez contactado nos solicita login y password. Finalmente tenemos acceso al servidor (observa el prompt del equipo ***profeserver*** en la última línea).

INICIAR UNA SESIÓN TELNET CON EL SERVIDOR WEB DESDE LINUX

Para iniciar una sesión ***telnet*** con el servidor web lo único a tener en cuenta es que hay que indicar que queremos hacer la conexión con el ***puerto 80***. Recuerda que los servidores web, por defecto, escuchan por el ***puerto 80***. La instrucción sería:

```
telnet IPservidor 80
```

El equipo quedará a la espera durante un pequeño espacio de tiempo (tras un minuto sin usarlos se cortará la sesión ***telnet***) durante el cual podemos hacer peticiones ***HTTP***. Un ejemplo de ***petición y respuesta HTTP*** es la siguiente:

```
administrador@administrador-TECRA-R950:~/Escritorio$ telnet 192.168.43.222 80
Trying 192.168.43.222...
Connected to 192.168.43.222.
Escape character is '^]'.
GET /prueba.html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
  <HEAD>
    <TITLE>
      Prueba de HTML
    </TITLE>
    <META http-equiv="content-type" content="text/html;charset=UTF-8">
  </HEAD>
  <BODY>
    <H1>Prueba de documento HTML</H1>
    <P>Esto es un documento HTML versión 4.01 Strict</P>
  </BODY>
</HTML>
Connection closed by foreign host.
```

Tras iniciar la conexión ***telnet*** hemos hecho una ***petición HTTP***:

```
GET /prueba.html
```

Y obtenemos en texto plano el contenido del fichero solicitado.

4.2.3 Mensajes HTTP de petición

Los mensajes que se intercambian en el protocolo ***HTTP*** son mensajes en texto plano (***ASCII***). Los ***mensajes de petición*** de recursos tienen tres partes:

- Línea inicial: Método, URL y versión.
- Líneas de cabecera (opcional): permiten aportar información adicional sobre la solicitud y/o el cliente (navegador, sistema operativo, etc.). Cada una de estas líneas está formada por un nombre que describe el tipo de encabezado, seguido de dos puntos (:) y el valor del encabezado.
- Cuerpo del mensaje (opcional): deben estar separadas de las líneas precedentes por una línea en blanco y permiten, por ejemplo, que se envíen datos por un comando ***POST*** durante la transmisión de datos al servidor utilizando un formulario.

Así, la solicitud ***HTTP*** posee esta sintaxis (***<crlf>*** es retorno de carro y avance de línea):

```
MÉTODO URL VERSIÓN<crlf>
ENCABEZADO: Valor<crlf>
. . . ENCABEZADO: Valor<crlf>
Línea en blanco <crlf>
CUERPO DE LA SOLICITUD
```

A continuación se encuentra un ejemplo de una solicitud ***HTTP***:

```
GET http://es.kioskea.net HTTP/1.0
Accept: Text/html
If-Modified-Since: Saturday, 15-January-2000 14:37:11 GMT
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)
```

MÉTODOS DE PETICIÓN

El método de petición es la orden que indica la petición del cliente al servidor: ***GET***, ***POST***, ***OPTION***, ***HEAD***, ***PUT***, ***DELETE***, ***TRACE***, ***CONNECT***, ***PATH***.

La versión 0.9 sólo contempla el método ***GET***.

La versión 1.0 sólo contempla los métodos ***GET***, ***POST*** y ***HEAD***.

La versión 1.1 contempla 8 métodos: ***GET***, ***POST***, ***HEAD***, ***PUT***, ***OPTIONS***, ***TRACE***, ***DELETE***, ***CONNECT*** (Sólo obliga a implementar ***GET*** y ***HEAD***, siendo todos los demás opcionales. En cualquier caso, los servidores que implementen alguno de los métodos adicionales, deben atenerse a la especificación de los mismos. Existe también la posibilidad de implementar métodos extendidos, a los que la especificación no pone ningún límite). En caso de que un servidor tenga implementado un método, pero no esté permitido para el recurso que se pide, entonces ha de devolver un código de estado ***405*** (método no permitido). Si lo que ocurre es que no tiene implementado el método entonces devuelve un código ***501*** (no implementado).

GET	Solicita el recurso ubicado en la ***URL*** especificada
	
	Pide una representación del recurso especificado. Por seguridad **no debería** ser usado por aplicaciones que causen efectos ya que transmite información a través de la ***URI*** agregando parámetros a la ***URL***.
	
	Ejemplo:
	`GET /images/logo.png HTTP/1.1`
	
	obtiene un recurso llamado ***logo.png***
	
	Ejemplo con parámetros:
	`/index.php?page=main&lang=es`

HEAD	Solicita el encabezado del recurso ubicado en la ***URL*** especificada Pide una respuesta idéntica a la que correspondería a una petición ***GET***, pero sin el cuerpo de la respuesta. Esto es útil para la recuperación de meta-información escrita en los encabezados de respuesta, sin tener que transportar todo el contenido.
POST	Envía datos al programa ubicado en la ***URL*** especificada Somete los datos a que sean procesados para el recurso identificado. Los datos se incluirán en el cuerpo de la petición. Esto puede resultar en la creación de un nuevo recurso o de las actualizaciones de los recursos existentes o ambas cosas.
PUT	Envía datos a la ***URL*** especificada Sube, carga o realiza un upload de un recurso especificado (archivo), es el camino más eficiente para subir archivos a un servidor, esto es porque en ***POST*** utiliza un **mensaje multiparte** y el mensaje es decodificado por el servidor. En contraste, el método ***PUT*** te permite escribir un archivo en una

	conexión ***socket*** establecida con el servidor. La desventaja del método ***PUT*** es que los servidores de hosting compartido no lo tienen habilitado. Ejemplo: `PUT /path/filename.html HTTP/1.1`
DELETE	Borra el recurso ubicado en la ***URL*** especificada
OPTIONS	Para que el cliente pueda obtener del servidor sus características Devuelve los métodos ***HTTP*** que el servidor soporta para un ***URL*** especifico. Esto puede ser utilizado para comprobar la funcionalidad de un servidor web mediante petición en lugar de un recurso especifico.

El método ***GET*** es el que se usa habitualmente para solicitar un recurso del servidor. Este mensaje se envía cada vez que el usuario introduce una ***URL*** en el navegador o pincha sobre un enlace o envía un formulario ***GET***.

Las peticiones ***GET*** no tienen cuerpo de mensaje por lo que la cantidad de información que se puede enviar en la petición está muy limitada. Además esa información es visible puesto que se incluye como parte de la ***URL***.

El método ***POST*** se usa para solicitar al servidor que acepte información que se envía adjunta en el cuerpo del mensaje. La información no está visible en la ***URL***. Este mensaje se genera cada vez que se envía un formulario ***POST***.

CABECERAS EN UN MENSAJE DE PETICIÓN

Cabeceras de petición	Descripción
Accept	Tipo de contenido aceptado por el navegador (por ejemplo, ***texto/html***). Consulte Tipos de ***MIME***
Accept-Charset	Juego de caracteres que el navegador espera
Accept-Encoding	Codificación de datos que el navegador acepta
Accept-Language	Idioma que el navegador espera (de forma predeterminada, inglés)
Authorization	Identificación del navegador en el servidor
Content-Encoding	Tipo de codificación para el cuerpo de la solicitud
Content-Language	Tipo de idioma en el cuerpo de la solicitud
Content-Length	Extensión del cuerpo de la solicitud
Content-Type	Tipo de contenido del cuerpo de la solicitud (por ejemplo, ***texto/html***). Consulte Tipos de ***MIME***
User-Agent	Cadena con información sobre el cliente, por ejemplo, el nombre y la versión del navegador y el sistema operativo

4.2.4 Mensajes HTTP de respuesta

Los ***mensajes de respuesta*** tienen tres partes:

- Línea inicial: estado, versión, código de estado y texto explicativo.
- Líneas de cabecera (opcional): permiten aportar información adicional sobre la respuesta y/o el servidor. Cada una de estas líneas está compuesta por un nombre que califica el tipo de encabezado, seguido por dos puntos (**:**) y por el valor del encabezado.
- Cuerpo del mensaje (opcional): contiene el documento solicitado.

Por lo tanto, una respuesta ***HTTP*** posee la siguiente sintaxis (**<*crlf*>** significa retorno de carro y avance de línea):

```
VERSIÓN-HTTP CÓDIGO EXPLICACIÓN <crlf>
ENCABEZADO: Valor<crlf>
. . . ENCABEZADO: Valor<crlf>
Línea en blanco <crlf>
CUERPO DE LA RESPUESTA
```

A continuación se encuentra un ejemplo de una respuesta ***HTTP***:

```
HTTP/1.0 200 OK Date: Sat, 15 Jan 2000 14:37:12 GMT Server :
Microsoft-IIS/2.0 Content-Type : text/HTML Content-Length : 1245
Last-Modified : Fri, 14 Jan 2000 08:25:13 GMT
```

CABECEERAS EN UN MENSAJE DE RESPUESTA

Cabeceras de respuesta	Descripción
Content-Encoding	Tipo de codificación para el cuerpo de la respuesta
Content-Language	Tipo de idioma en el cuerpo de la respuesta
Content-Length	Extensión del cuerpo de la respuesta
Content-Type	Tipo de contenido del cuerpo de la respuesta (por ejemplo, ***text/html***). Consulte Tipos de ***MIME***
Date	Fecha en que comienza la transferencia de datos
Expires	Fecha límite de uso de los datos
Forwarded	Utilizado por equipos intermediarios entre el navegador y el servidor
Location	Redireccionamiento a una nueva dirección ***URL*** asociada con el documento
Server	Características del servidor que envió la respuesta

CÓDIGOS DE ESTADO Y ERROR

Los códigos de estado y error los envían los servidores en sus respuestas para informar al cliente de cómo ha ido la petición. Los códigos van acompañados de un texto explicativo. Los códigos tienen 3 cifras, la primera de las cuales será un número del 1 al 5. Esa primera cifra tiene un significado:

- ***100*** a ***199***: Son códigos informativos.
- ***200*** a ***299***: Son códigos de éxito.
- ***300*** a ***399***: Son códigos de redirección.
- ***400*** a ***499***: Son códigos de errores que se producen en el cliente.
- ***500*** a ***599***: Son códigos de errores que se producen en el servidor.

	Código		***Descripción***
1xx Mensajes	1xx		Conexión rechazada
2xx Operación exitosa (Estos códigos indican la correcta ejecución de la transacción)	200	OK	OK
	201	CREATED	Información no oficial
	202	ACCEPTED	
	203	PARTIAL INFORMATION	
	204	NO RESPONSE	Sin Contenido
	205	RESET CONTENT	Contenido para recargar
	206	PARTIAL CONTENT	Contenido parcial
3xx Redireción (Estos códigos indican que el recurso ya no se encuentra en la ubicación especificada)	301	MOVED	Mudado permanentemente
	302	FOUND	Encontrado
	303	METHOD	Vea otros
	304	NOT MODIFIED	No modificado
	305		Utilice un proxy
	307		Redirección temporal
4xx Error por parte del cliente (Estos códigos indican que la solicitud es incorrecta)	400	BAD REQUEST	Solicitud incorrecta
	401	UNAUTHORIZED	No autorizado
	402	PAYMENT REQUIRED	Pago requerido
	403	FORBIDDEN	Prohibido
	404	NOT FOUND	No encontrado
	409		Conflicto
	410		Ya no disponible
	412		Falló precondición
5xx Error del servidor (Estos códigos indican que existe un error interno en el servidor)	500	INTERNAL ERROR	Error interno
	501	NOT IMPLEMENTED	No implementado
	502	BAD GATEWAY	Pasarela incorrecta
	503	SERVICE UNAVAILABLE	Servicio no disponible
	504	GATEWAY TIMEOUT	Tiempo de espera de la pasarela agotado
	505		Versión de HTTP no soportada