Servidores WEB (HTTP)

1. Introducción

WWW

Web o la web, la red o www de World Wide Web, es básicamente un medio de comunicación de texto, gráficos y otros objetos multimedia a través de Internet, es decir, la web es un *sistema de hipertexto* que utiliza Internet como su mecanismo de transporte o desde otro punto de vista, una forma gráfica de explorar Internet.

Características de la Web

Según su propio creador, *Berners-Lee*, la Web es un sistema que presenta las siguientes características:

1. ***Hipermedia***: en la Web podemos manejar ***información multimedia*** y ***navegar*** a través de ella.

2. ***Distribuido***: a diferencia de las antiguas y enormes bases de datos que concentraban la información físicamente en un único lugar, la ***Web es un sistema compuesto por miles de servidores*** localizados en cientos de ciudades del mundo que están interconectadas entre sí.

3. ***Heterogéneo***: por ser un servicio relativamente nuevo, la Web tiene la ventaja de poder reunir ***servicios y protocolos más antiguos*** (como Gopher, los News, FTP, e inclusive el correo electrónico), de tal modo que puede presentar la información desde ***un único programa cliente***.

4. ***Colaborativo***: ésta es la característica que ha dado el mayor empuje a su crecimiento, ya que cualquier persona, en cualquier parte del mundo, puede agregar información a la Web para que luego pueda ser consultada por el resto de los usuarios.

Funcionamiento de la Web

1. El primer paso consiste en ***traducir la parte del nombre del servidor de la URL*** en una ***dirección IP*** usando la base de datos distribuida de Internet conocida como **DNS**. Esta dirección IP es necesaria para contactar con el Servidor Web y poder enviarle paquetes de datos.

2. El siguiente paso es enviar una ***petición HTTP al servidor Web solicitando el recurso***. En el caso de una página web típica, *primero se solicita el texto HTML* y luego es inmediatamente analizado por el navegador, el cual, después, hace *peticiones adicionales para los gráficos y otros ficheros que formen parte de la página*. Las

estadísticas de popularidad de un sitio web normalmente están basadas en el número de páginas vistas o las peticiones de servidor asociadas, o peticiones de fichero, que tienen lugar.

3. Al recibir los ficheros solicitados desde el servidor web, el ***navegador muestra la página tal y como se describe en el código HTML, el CSS y otros lenguajes web***. Al final se incorporan las imágenes y otros recursos para producir la página que ve el usuario en su pantalla.

W3C: estándares Web

W3C son las siglas de World Wide Web Consortium, un consorcio fundado en 1994 para dirigir a la Web hacia su pleno potencial mediante el ***desarrollo de protocolos comunes*** que promuevan su ***evolución*** y aseguren su interoperabilidad.

El consorcio está compuesto por un *grupo de programadores, desarrolladores web, ejecutivos de la industria y usuarios* que ayudan a definir las especificaciones para el desarrollo de la tecnología web.

¿Qué son los Estándares?

> *Un estándar es un conjunto de reglas normalizadas que describen los requisitos que deben ser cumplidos por un producto, proceso o servicio, con el objetivo de establecer un mecanismo base para permitir que distintos elementos hardware o software que lo utilicen, sean compatibles entre sí.*

El ***W3C***, como organización independiente y neutral, ***desarrolla estándares*** relacionados con la ***Web*** también conocidos como **Recomendaciones**, que sirven como referencia para construir una Web accesible, interoperable y eficiente, en la que se puedan desarrollar aplicaciones cada vez más robustas.

Algunos de los ***estándares Web más conocidos*** y ampliamente utilizados son:

4. **HTML** (***HyperText Markup Language***), para definir la estructura de los documentos (páginas webs).
5. **XML** (***eXtensible Markup Language***), que sirve de base para un gran número de tecnologías.
6. **CSS** (***Cascading Style Sheets***), que permite asignar estilos para la representación de los documentos.
7. **XHTML (*XML HyperText Markup Language)***

¿Para qué sirven?

> *La finalidad de los estándares es la creación de una Web universal, accesible, fácil de usar y en la que todo el mundo pueda confiar. Con estas tecnologías abiertas y de uso libre se pretende evitar la fragmentación de la Web y mejorar las infraestructuras para que se pueda evolucionar hacia una Web con la información mejor organizada.*

Nombres y direcciones (URIs y URLs)

La petición de una página web o un objeto se realiza escribiendo en el navegador web su dirección. Básicamente, la estructura de dicha dirección, denominada **URI *(Uniform Resource Identifier)*** o **URL *(Uniform Resource Locator)*** consiste en:

- Una primera sección donde se indica el ***protocolo*** (por ejemplo, ***http***).
- La ***dirección IP del servidor o su nombre de dominio*** (por ejemplo, www.debian.org).
- El ***puerto donde se realiza la petición***, esto es, el puerto donde escucha el servidor (por ejemplo, el ***puerto 80 para http, o 443 para https***).
- Y la ***ruta del documento u objeto*** en el sistema de archivos virtual público del servidor (por ejemplo, ***index.html,*** colocado en el directorio raíz del sistema de archivos virtual).

Todos estos campos van separados por caracteres específicos. Una dirección resultante podría ser:

http://www.debian.org:80/index.html

Normalmente, tanto el servidor como el cliente HTTP pueden asumir valores por defecto:

8. El cliente HTTP asume por defecto:
    a. Como protocolo: *http*.
    b. Como puerto: *80*.
9. El servidor asume por defecto que debe servir, si no se indica ninguno los ficheros catalogados en su configuración como índices (normalmente *index.html, index.htm, index.php*, etc.)

De tal forma que la dirección anterior se podría simplificar (en los navegadores web) al asumir valores por defecto como:

www.debian.org

URIs relativas

Las URIs relativas son ***URIs parciales***, utilizadas para *referirse a un documento desde otro en la misma computadora.*

De esta forma, podemos definir una URI relativa como la ruta que se debe seguir desde la ubicación del documento actual (ruta de directorios) a la ubicación del recurso referido, además del nombre de archivo.

Documento Actual	Documento Destino	URIs relativa
http://servidor.es/doc/uno.php	http://servidor.es/doc/ejemplos.php	**ejemplos.php**
http://servidor.es/doc/uno.php	http://servidor.es/doc/nuevo/dos.php	**nuevo/dos.php**
http://servidor.es/doc/uno.php	http://servidor.es/index.php	**../index.php**
http://servidor.es/doc/uno.php	http://servidor.es/img/foto.jpg	**../img/foto.jpg**

Páginas Web (*estáticas vs dinámicas)* y Sitios Web *(portales Web)*

Páginas web

Una página web es el nombre de un ***documento*** o información electrónica adaptada ***para la World Wide Web*** y que puede ser accedida ***mediante un navegador*** para mostrarse en un monitor de computadora o dispositivo móvil.

Esta información se encuentra generalmente en ***formato HTML o XHTML***, y puede proporcionar navegación a otras páginas web mediante ***enlaces de hipertexto***. Las páginas incluyen otros recursos como hojas de estilo en cascada, guiones (scripts) e imágenes digitales, entre otros.

Las páginas web pueden estar almacenadas (***hosting***) en un equipo local (*sólo disponibles para el cliente instalado en el equipo local*) o en un ***Servidor Web***.

El ***Servidor Web*** puede restringir el acceso a las páginas web, únicamente a redes privadas (***Intranet Corporativa***), o puede ***publicar las páginas en la World Wide Web***.

El acceso a las páginas web es realizado mediante su transferencia desde servidores utilizando el ***protocolo de transferencia de hipertexto (HTTP)***.

Tipos de Páginas Web: estáticas vs dinámicas

El contenido de la página puede ser predeterminado (***«página web estática»)*** o generado al momento de ser visualizada o solicitada al Servidor Web (***«página web dinámica»).***

Las páginas dinámicas que se generan, al ser solicitadas, son creadas por una aplicación (***PHP, ASP, JSP, ..***) en el Servidor Web que alberga las mismas, e incluso podrían necesitar

interactuar con un Sistema Gestor de Bases de Datos (***mysql, Oracle, …***) para generar la página solicitada.

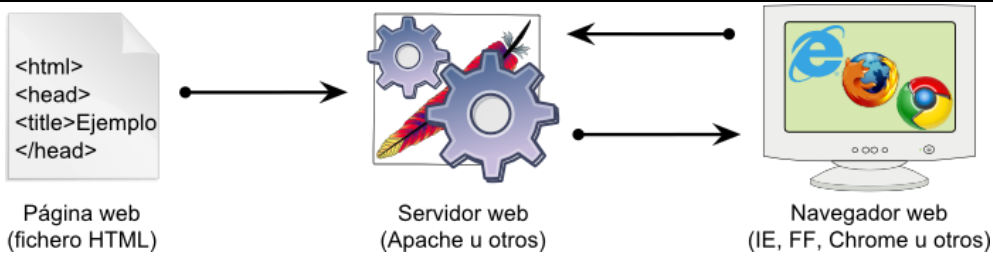


Imagen: Solicitud página web estática

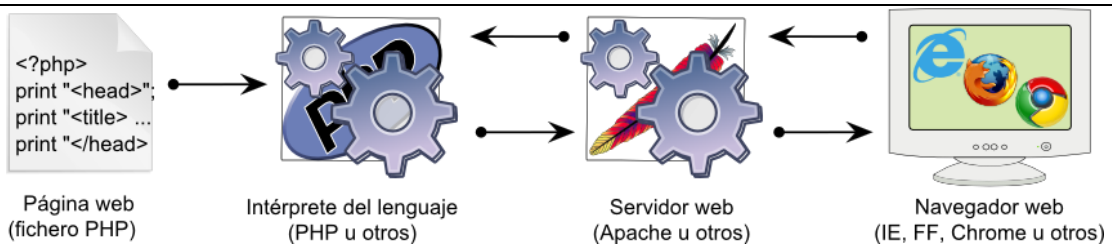


Imagen: Solicitud y generación página web dinámica

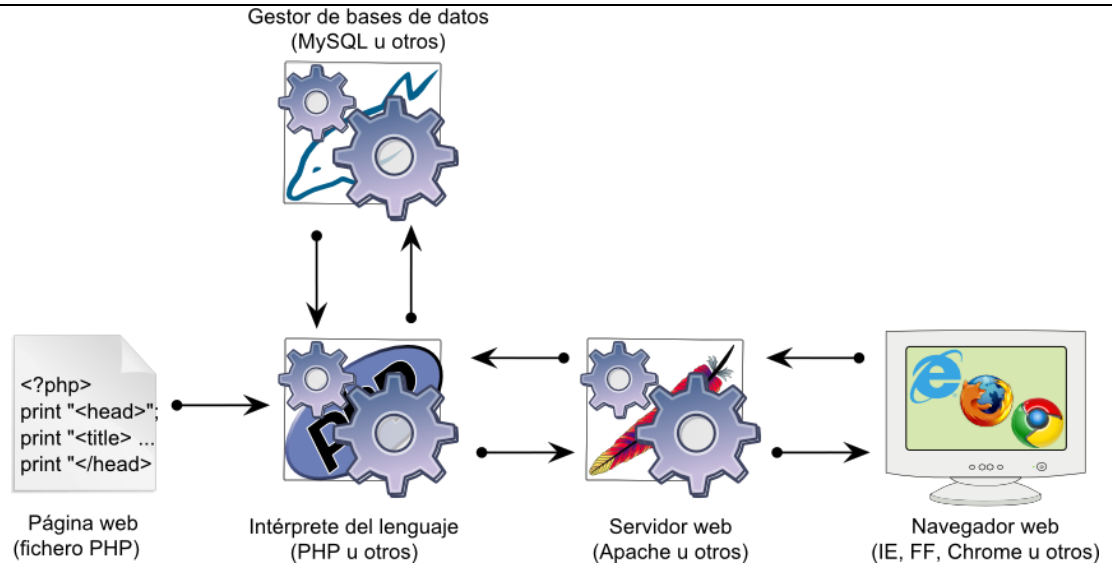


Imagen: Solicitud y generación página web dinámica (con acceso a base de datos)

Sitios Web (*portales Web*)

Un ***sitio web*** es un sitio (localización) en la World Wide Web que c***ontiene documentos (páginas web) organizados jerárquicamente***. Cada documento (página web) contiene texto y o gráficos que aparecen como información digital en la pantalla de un ordenador. Un sitio puede contener una combinación de gráficos, texto, audio, vídeo, y otros materiales dinámicos o estáticos.

Cada ***sitio web tiene una página de inicio*** (en inglés *Home Page*), que es el ***primer documento*** que ve el usuario cuando entra en el sitio web ***poniendo el nombre del dominio*** de ese sitio web en un navegador. El sitio normalmente tiene otros documentos (páginas web) adicionales. Cada sitio pertenece y es gestionado por un individuo, una compañía o una organización.

Una página web es parte de un sitio web y es un único archivo con un nombre de archivo asignado, mientras que un sitio web es un conjunto de archivos llamados páginas web (portal web)

Navegadores WEB

Aplicación que opera a través de Internet, interpretando la información de archivos y sitios web para que el cliente sea capaz de leerla, ya se encuentre ésta alojada en un servidor dentro de la World Wide Web, en un Servidor Local o directamente en el equipo local del cliente.

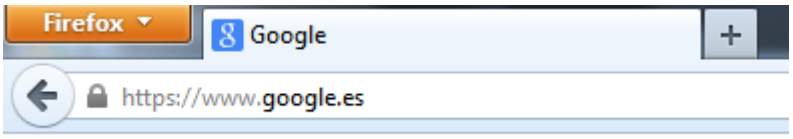
El navegador interpreta el código, HTML, en el que está escrita la página web y lo presenta en pantalla permitiendo al usuario interactuar con su contenido y navegar hacia otros lugares de la red mediante enlaces o hipervínculos.

La funcionalidad básica de un navegador web es permitir la visualización de documentos de texto, posiblemente con recursos multimedia incrustados.

La comunicación entre el servidor web y el navegador se realiza mediante el ***protocolo HTTP***, aunque la mayoría soportan otros protocolos como: ***FTP, Gopher, y HTTPS*** (*versión cifrada de HTTP basada en "Secure Socket Layer" o "Capa de Conexión Segura": SSL)*

La función principal del navegador es descargar documentos HTML y mostrarlos en pantalla. En la actualidad, no solamente descargan este tipo de documentos sino que muestran con el documento sus *imágenes, sonidos e incluso vídeos streaming en diferentes formatos y protocolos.*

Barra de Direcciones: muestra la URL (*Universal Resource Localtion*) del recurso al que se quiere acceder.



Recuerda que hay que ***traducir la parte del nombre del servidor de la URL*** en una ***dirección IP*** utilizando la base de datos distribuida **DNS** para poder contactar con el Servidor Web.

Servidores HTTP (Web)

Caben destacar por su antigüedad y/o utilización, los siguientes ***servidores HTTP*** existentes en la red:

- **NCSA HTTPd**. Fue uno de los ***primeros servidores y además gratuito***. Tenía dos restricciones imponentes:
  - No podía funcionar como servidor de proxy (para la realización de muros de seguridad que eviten intromisiones en su red privada).
  - No disponía a nivel internacional del sistema de transacciones seguras (aunque sí para los EEUU y Canadá).

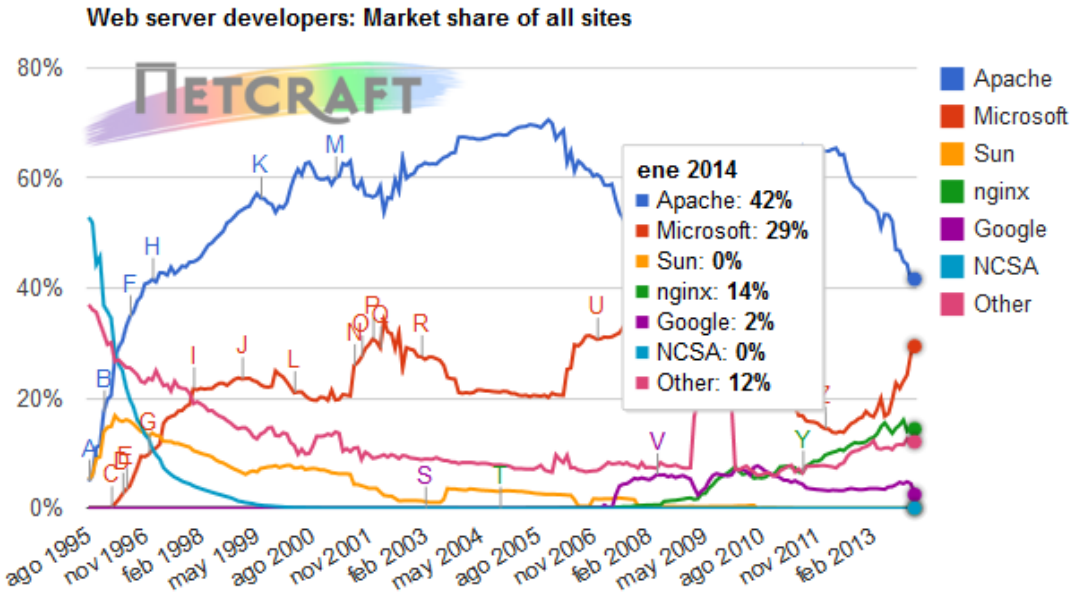
  Éstas son algunas de las razones por las que el proyecto fue abandonado y su sitio oficial recomendó utilizar el ***servidor Apache***.

- **Apache**. Basado inicialmente en su funcionamiento en NCSA, actualmente se encuentra en la versión 2. Su facilidad de configuración y sus prestaciones sobre diversas plataformas lo han convertido en líder en Internet.

- **Internet Information Server (IIS)**. Fue desarrollado por Microsoft para sus sistemas servidores aunque también funciona en sistemas clientes (como Windows 7).

- **Nginx.** Servidor HTTP y proxy caché inverso gratuito de alto rendimiento que está empezando a hacer competencia a Apache debido a su estabilidad, características, configuración simple y su bajo consumo de recursos.

El más utilizado hoy en día en servidores Linux es Apache, y en servidores Windows tanto Apache como IIS.



2. Protocolo HTTP

El **Protocolo de Transferencia de HiperTexto** (***Hypertext Transfer Protocol***) es un sencillo *protocolo Cliente Servidor* que gestiona los intercambios de *información entre los clientes Web y los servidores HTTP*.

Desde el punto de vista de las **<u>comunicaciones</u>**:

- Los elementos software de la arquitectura web (*clientes, servidores, proxies*) utilizan el protocolo HTTP para comunicarse.
- Es un protocolo de la ***capa de aplicación***: por debajo está TCP/IP.

<table>
<tr><td colspan="2">Web</td><td colspan="2">Correo-e</td><td>FTP</td><td>News</td></tr>
<tr><td>HTTP</td><td>POP3</td><td>SMTP</td><td>FTP</td><td>NNTP</td></tr>
<tr><td colspan="5">TCP/IP</td></tr>
<tr><td colspan="5">Red física</td></tr>
</table>

- El Servidor escucha en un puerto de comunicaciones **TCP** (*por defecto, el puerto 80*) y espera las solicitudes de conexión de los clientes web.
- HTTP se basa en sencillas ***operaciones (mensajes) de solicitud/respuesta:***
  - Un ***cliente*** establece una conexión con un servidor y envía un mensaje con los datos de la ***solicitud***.
  - El ***servidor responde*** con un mensaje similar, que contiene el estado de la operación y su posible resultado.
- Es un protocolo que ***no maneja estados***:
  - Hay ausencia de estado tras cada par "petición-respuesta".
  - Tras la respuesta, el servidor cierra inmediatamente la conexión.

Funcionamiento Básico

- El usuario escribe en la ***barra de dirección del navegador*** el recurso al que desea acceder:

  http://www.dioceavila.com/informes/documentos.html

- El ***navegador*** descompone la ***URL en 4 partes***:
  - El protocolo ("*http"*)
  - El nombre del servidor ("*www.dioceavila.com"*)
  - La ruta de acceso al documento ("*/informes/documentos.html*")
  - El puerto (opcional) de la petición: por defecto el puerto 80.

- El ***navegador*** se comunica con el ***Servidor de Nombres (DNS***) para traducir el nombre del Servidor (*www.dioceavila.com)* en una **Dirección IP**, que es utilizada para conectarse con el Servidor.

- Se abre una ***conexión TCP/IP con el servidor***, llamando al ***puerto TCP*** correspondiente.

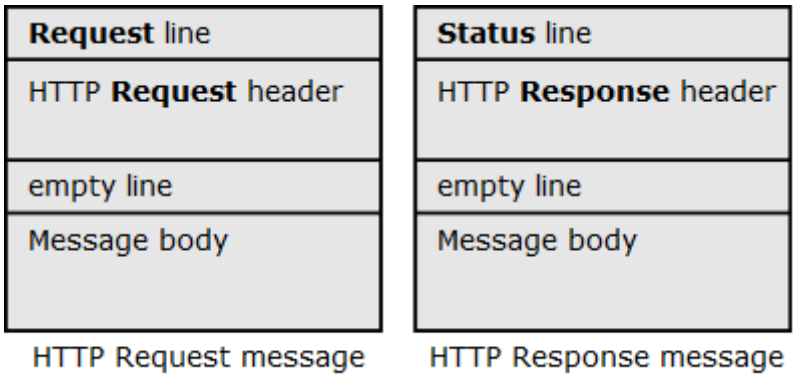
- Se realiza la ***petición***, enviando:
  - El comando necesario: *GET, POST, HEAD*,…
  - La *dirección del objeto* requerido (el contenido de la URL que sigue a la dirección del servidor).
  - La *versión del protocolo HTTP* empleada (casi siempre HTTP/1.0).
  - Y un *conjunto variable de información*, que incluye datos sobre las capacidades del navegador, datos opcionales para el servidor,…

- El ***servidor*** devuelve la ***respuesta*** al cliente, que consiste en:
  - Un *código de estado*.
  - El tipo de dato *MIME* de la información de retorno.
  - La *información* de retorno.

Mensajes HTTP

Los ***mensajes HTTP*** están compuestos por:

1. ***Una línea inicial del mensaje:*** primera línea del mensaje donde se indica qué hacer (*mensaje de petición)* o qué ha ocurrido (*mensaje de respuesta*).
2. ***Cabecera del mensaje:*** bloque de campos terminados por una ***línea en blanco***. Contienen los atributos del mensaje.
3. ***Cuerpo del mensaje:*** Opcional. Su presencia depende de la petición y del resultado.



Ejemplo de Petición	*Ejemplo de Respuesta*

`GET /saludo.html HTTP/1.1`
`Host www.uv.es`
	`HTTP/1.1 200 OK`
	`Date: Sun, 01 May 2003 12:00:01 GMT`
	`Content-Type: text/html`
	`Content-Length: 45`

A. Mensajes HTTP: Línea inicial del mensaje

Si es un mensaje de Petición (*Request*) HTTP

Especifica el recurso que se solicita y qué se precisa de él.

- ***Nombre de método***:
  - ***GET*:** solicita un documento al servidor (*se pueden enviar datos en la URL)*
  - ***HEAD***: Solo solicita las cabeceras HTTP (consultar información sobre el fichero antes de solicitarlo).
  - ***POST***: Similar a GET pero además permite enviar información en el cuerpo del mensaje para su procesado (*la URL debe ser una página dinámica que procese la información enviada*).
  - ***PUT***: almacena el documento enviado en el cuerpo del mensaje.
  - ***DELETE***: elimina el documento referenciado en la URL.
  - ***TRACE***: rastrea los intermediarios por los que pasa la petición.
  - ***OPTIONS***: averigua los métodos HTTP que soporta el servidor para un recurso.
- ***Recurso (URL completa)***
- ***Versión del protocolo HTTP*** (HTTP/x.x)

Ejemplo:

GET /directorio/otro/fichero.html HTTP/1.1

Si es un mensaje de Respuesta (*Response*) HTTP

- ***Versión de HTTP*** (HTTP/x.x)

- ***Código numérico de estado y comentario descriptivo.***
  - ***1xx:*** Respuesta Provisional (mensaje informativo)
  - ***2xx:*** Correcto (el servidor ha procesado la solicitud correctamente)
  - ***3xx:*** Redirección.
  - ***4xx:*** Error de solicitud (*error del cliente)*
  - ***5xx:*** Error del Servidor.

Los más frecuentes son:

200 (Correcto): El Servidor ha procesado la solicitud correctamente. ***401 (No autorizado)***: La solicitud requiere autentificación. ***404 (No se encuentra)***: El servidor no encuentra la página o el recurso solicitado. **503 (Servicio no disponible)**: El Servidor está temporalmente fuera de servicio.

Ejemplo:

HTTP 1.1 200 OK

B. Mensajes HTTP: Cabecera del mensaje

Clasificación de las cabeceras según su función:

- ***De ámbito general.*** Información sobre la sesión de comunicación (cliente y servidor):
  - ***Date*** (*fecha de creación del recurso*)
    - ***Date: Wed, Jan 01 12:37:02 2014 GMT***
  - ***Cache-Control*** (*controla el comportamiento de la caché*)
  - ***Connection: close*** *(la conexión debe cerrarse una vez transmitido el mensaje)*
  - ***Connection: Keep-Alive*** *(no cerrar la conexión)*

- ***Exclusivas de la petición:*** información del cliente (*el cliente envía información adicional al servidor)*
  - ***Accept*** (*tipos MIME aceptados por el navegador)*
  - ***Accept-Charset*** *(preferencias en el conjunto de caracteres)*
  - ***Accept-Encoding*** *(preferencias en la codificación del contenido*)
  - ***Accept-Language*** *(preferencias en el Lenguaje del documento*)
  - ***If-Modified-Since*** *(GET condicional: "si se ha modificado desde")*
  - ***If-Unmodified-Since*** *(GET condicional: "si no se ha modificado desde")*
  - ***If-Match*** *(GET condicional: si coincide con un ETag, para actualizaciones eficientes en la caché)*
  - ***If-None-Match*** *(GET condic.: descarga un recurso si no coincide con un ETag)*
  - ***Max-Forward*** (*límite de cabeceras añadidas en TRACE*)
  - ***Range*** (*rango de bytes, se emplea para GET parciales)*

- ***Host*** *(nombre y puerto del Servidor al que se dirige la petición)*
- ***From*** *(email del autor de la solicitud)*
- ***User-Agent*** *(identificación del programa del cliente: Mozilla, MSIE, ..)*
- ***Referer*** *(URL origen de la petición)*
- ***Authorization*** *(indica las credenciales de acceso: login-password)*
- ***Cookies*** *(nombre y valores de las cookies)*

- ***Exclusivas de la respuesta:*** información del Servidor (*el Servidor envía información adicional al cliente*)
  - ***Location*** *(redirecciona hacia la localización real del recurso)*
  - ***WWW-Authenticate*** *(se solicita autentificación)*
  - ***Etag*** *(etiqueta para cachear)*
  - ***Age*** *(tiempo desde que fue generada la respuesta: para caché)*
  - ***Server*** *(versión del software del servidor: Apache/1.3.12)*
  - ***Retry-After*** *(tiempo de espera antes de solicitar el recurso de nuevo)*
  - ***Accept-Ranges*** *(rangos aceptados del recurso)*
  - ***Set-Cookies*** *(establecer/crear cookies en el cliente)*

- ***De Entidad:*** información sobre el recurso (en mensajes de solicitud y respuesta)
  - ***Allow*** *(métodos permitidos para el recurso)*
  - ***Content-Encoding*** *(codificación/compresión de la entidad: gzip)*
  - ***Content-Length*** *(longitud de la entidad: longitud del cuerpo del recurso)*
  - ***Content-Type*** *(tipo MIME de la entidad: tipo del contenido del recurso)*
  - ***Expires*** *(fecha tope de validez en la caché)*
  - ***Last-Modified*** *(fecha de la última modificación de la entidad/recurso)*
  - ***Content-Language*** *(lenguaje natural de la entidad: idioma del contenido)*
  - ***Content-Location*** *(localización URL alternativa)*

*Las **peticiones** al menos deberían tener la cabecera **User-Agent**, y las respuestas las cabeceras **Server** y **Last-Modified.***

C. Mensajes HTTP: Cuerpo del mensaje

En las **Respuestas:** contiene el recurso pedido o el texto explicando un error.

En las **Peticiones:** contiene datos de usuario o ficheros para subir.

Si existe cuerpo, deben aparecer al menos las ***siguientes cabeceras*** relativas a él:

- ***Content-Type:*** *tipo MIME de los datos (ejemplo: text/html, image/png, ..)*
- ***Content-Length***: número de bytes en el cuerpo.

Ejemplo de Petición (completa):

```
GET /~pdi/test.html HTTP/1.1 <CRLF>
Connection: close <CRLF>
User-Agent: Mozilla/5.0 [en] (X11; I; Linux 2.2.15 i586;
   Nav ...) <CRLF>
Host: www.uv.es <CRLF>
Accept: image/gif, image/x-xbitmap, image/jpeg,
   image/pjpeg, */* <CRLF>
Accept-Encoding: gzip <CRLF>
Accept-Language: es <CRLF>
Accept-Charset: iso-8859-1 <CRLF>
<CRLF>
```

Ejemplo de Respuesta (completa):

```
HTTP/1.1 200 OK <CRLF>
Date: Tue, 23 Jan 2001 12:44:27 GMT <CRLF>
Server: Apache/1.3.9 (Unix) Debian/GNU <CRLF>
Last-Modified: Tue, 23 Jan 2001 12:39:45 GMT <CRLF>
ETag: "19e89f-22-3a6d7b91" <CRLF>
Content-Length: 34 <CRLF>
Content-Type: text/html; charset=iso-8859-1 <CRLF>
<CRLF>
<html>Esto es una prueba</html>
```

Ejemplo de Petición "GET condicional" y Respuesta:

```
GET http://www.uv.es/index.html HTTP/1.1
Host: www.uv.es
If-Modified-Since: Fri, 1 Feb 2004 13:53:40 GMT
```

```
HTTP/1.0 304 Not Modified
Date: Thu, 1 Mar 2004 13:55:13 GMT
Content-Type: text/html
Expires: Fri, 30 Apr 2004 13:55:13 GMT
```

Ejemplo de Petición "GET parcial" y Respuesta:

```
GET /Default.htm HTTP/1.1
Host: www.microsoft.com
Range: bytes=0-80
```

```
HTTP/1.1 206 Partial content
Server: Microsoft-IIS/5.0
Content-Type: text/html
Content-Length: 81
Content-Range: bytes 0-80/19618
<HTML> ....
```

3. Almacenamiento en Caché

Se llama ***caché web*** a la caché que almacena documentos web (páginas, imágenes, etc) para ***reducir el ancho de banda consumido***, ***la carga de los servidores*** y el ***retardo en la descarga***.

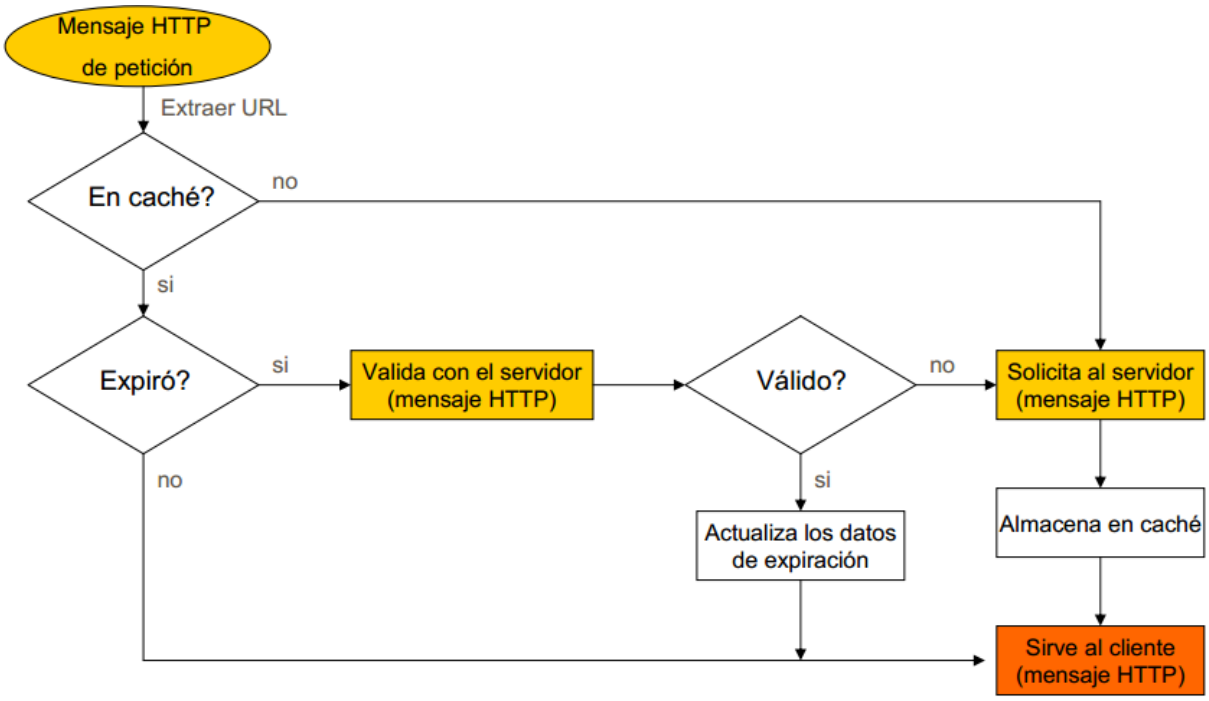
Un caché web almacena copias de los documentos que pasan por él, de forma que subsiguientes peticiones pueden ser respondidas por el propio caché, si se cumplen ciertas condiciones.

Tipos de cachés web

- Las ***cachés*** de ***agente de usuario (User-Agent),*** como las presentes en los navegadores web, son **cachés privados**, que funcionan solo para un único usuario.
- Los intermediarios en la comunicación cliente-servidor también pueden implementar ***cachés compartidos*** (también llamadas ***proxy-cachés directos***) que sirvan páginas a varios usuarios. Los proxy-cachés suelen ser usados por los proveedores de servicios de Internet (ISP), universidades y empresas para ahorrar ancho de banda. La intermediación de estos proxy-cachés difieren de la de los privados en que los clientes no necesitan ser explícitamente configurados para usarlos (***servidor proxy-caché transparente***).
- Las ***cachés pasarela*** (llamadas también ***proxy-cachés inversos*** o aceleradores web) funcionan a cargo del propio servidor web, de forma que los clientes no distinguen unos de otros.

Los intermediarios que funcionan como caché realizan con frecuencia otras tareas, tales como el ***filtrado de contenidos****.*

Funcionamiento



Expiración del documento

- Determina si el tiempo que lleva almacenado el objeto en la caché ha sobrepasado el máximo permitido. Si el objeto guardado en la caché no es lo suficientemente reciente, la caché ***valida*** el recurso con el servidor.

- Cabeceras HTTP involucradas enviadas por el Servidor:
    - **Expires:** fecha de expiración.
    - **Age:** segundos que el objeto lleva almacenado.
    - **Cache-Control: max-age = segundos**
        - Vida máxima del objeto

- La ***caché decide que el recurso ha expirado si:***
    - La edad del recurso es mayor que la vida máxima (***max-age***).
    - La fecha de expiración (***expires***) ha sido sobrepasada.

Validación con el Servidor

- La caché contrasta con el Servidor si el contenido del objeto ha cambiado antes de descargar dicho recurso.

- Utiliza peticiones condicionales:
    - Basadas en la etiqueta (***Etag***) del objeto guardado en caché:
        - **If-Match: etiqueta** (*Etag del objeto en la caché)*
        - **If-None-Match: etiqueta** (*Etag del objeto en la caché)*
    - Basadas en la fecha de la última modificación (***Last-Modified***)
        - **If-Modified-Since: fecha** *(Last-Modified del objeto en cache)*
        - **If-UnModified-Since: fecha** (*Last-Modified del objeto en cache)*

- Un objeto guardado en la caché será actualizado si:
    - La fecha incluida en **If-Modified-Since** es posterior a la fecha de la última modificación del objeto en el servidor origen.
    - La etiqueta incluida en **If-None-Match** no coincide con la del Servidor origen

4. Cookies

Una cookie es ***información enviada desde un servidor de páginas web y almacenada en el disco duro del visitante*** *dentro de ficheros de texto* ***a través del navegador***.

Esta información ***será reenviada de nuevo al servidor en cada petición,*** de forma que el servidor puede identificar o recuperar información sobre el usuario que está accediendo.

Ejemplos de uso

- Guardar información de la sesión.
- Comercio electrónico: "*carrito de la compra*".
- Personalización de páginas (*idiomas*).
- Seguimiento de las visitas a un portal web (*carteles publicitarios*)
- Almacenamiento del login y password del usuario.

Almacenamiento de las cookies

El hecho de que las cookies sean almacenadas en el lado del cliente, libera al servidor de una importante carga: *el cliente devuelve la información al servidor en siguientes peticiones***.**

Existen dos tipos de cookies:

- ***Cookies de sesión***: solo válidas mientras dure la sesión del cliente.
- ***Cookies persistentes***: almacenadas de forma "persistente" en el disco del cliente por parte del propio navegador.

Cookies enviadas por el Servidor

Si el Servidor quiere enviar una cookie al cliente debe incluir la cabecera HTTP: "**Set-Cookie"**. El formato de esta cabecera es:

- **Set-Cookie:**
  - **"nombre = valor;"** (*nombre de la cookie y valor)*
  - **"; expires = fecha"** *(fecha de caducidad de la cookie)*
  - **"; path=camino"** *(ruta del dominio donde la cookie estará disponible)*
  - **"; domain = dominio"** *(dominio al que pertenece la cookie)*
  - **"; secure"** *(sólo se transmite sobre canales seguros: HTTPS)*

Ejemplo:

Set-Cookie: nombreUsuario=Antonio; expires=Mon, 10-Feb-2014 12:35:23 GTM; path=/; domain=dioceavila.com; secure

Cookies enviadas por el Cliente

Cuando un cliente solicita una URL, buscará en su ***lista de cookies*** aquellas que coincidan con el ***dominio y la ruta (path) de la URL***. Estas cookies se deberán incluir dentro de la ***cabecera HTTP "Cookie"*** de la petición, eliminando previamente todas las que hayan caducado (*superado la fecha de expiración*).

El formato de la cabecera HTTP **"Cookie"** del Cliente:

Cookie: nombre1=valor1; nombre2=valor2; …

Gracias a las cookies, un servidor web puede identificar un conjunto pc-navegador-usuario y mostrar la información adecuada a ese conjunto, por ejemplo un carrito de compra que haya creado.

Limitaciones de las Cookies

- Máximo de 300 cookies en el disco: cuando se llegue a la cookie 301, se borra la más antigua.
- Tamaño máximo de 4Kbytes por cookie (nombre y valor).
- 20 cookies como máximo por Servidor o Dominio.
- Ninguna máquina que no coincida con el dominio de la cookie podrá acceder a ella.

Cookies: Seguridad

Las cookies son ***simples ficheros de texto*** almacenados por el navegador, es decir son ***elementos pasivos*** que no pueden emprender ninguna acción con lo cual no podrían infectar el ordenador con ningún tipo de virus.

Tampoco pueden ser usados para extraer datos del ordenador del cliente, solo almacenan la información confidencial que previamente haya sido enviada al Servidor.

Sin embargo:

- Cualquier usuario que tenga acceso a ellas, a través de la red o del equipo local (*nunca a través de internet*) puede extraer sus datos.
- Pueden ser utilizadas por los servidores para hacer un seguimiento oculto de las páginas visitadas por un usuario y crearse un perfil del usuario.

5. Autentificación (*autenticación)* del usuario

La autentificación es el proceso de ***identificar si un cliente es elegible para tener acceso a un recurso***. El protocolo HTTP soporta la **autentificación** como un medio de negociar el acceso a un recurso seguro.

La solicitud inicial de un cliente es normalmente una **solicitud anónima**, que no contiene ninguna información de autentificación. Las aplicaciones de servidor HTTP pueden denegar la solicitud anónima indicando que se requiere la autentificación. La aplicación de servidor envía ***encabezados de la autentificación*** de WWW para indicar los ***esquemas de autentificación soportados***.

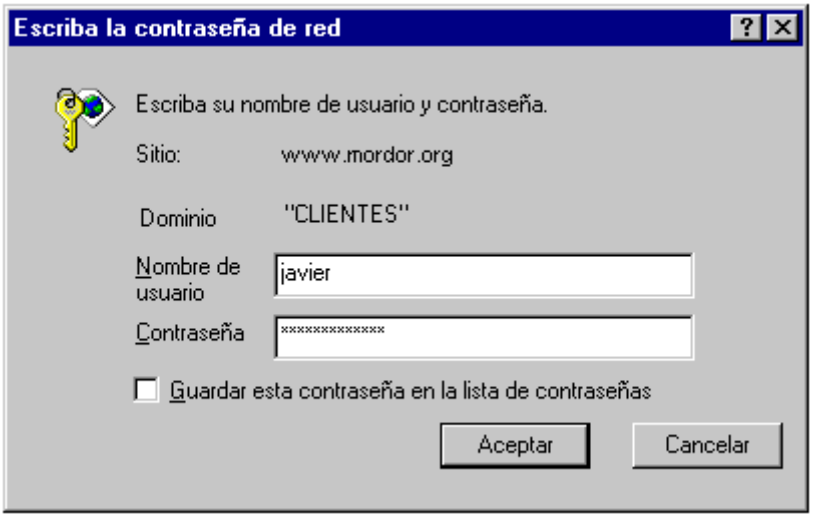
Existen varios **tipos de autentificación:**

- ***Autentificación BÁSICA***: soportado por todos los servidores web y navegadores, así como terminales móviles.
- ***Autentificación mediante DIGEST***: soportada por todos los servidores y por algunos navegadores.
- ***Autentificación https***: es una combinación del protocolo HTTP y protocolos criptográficos.

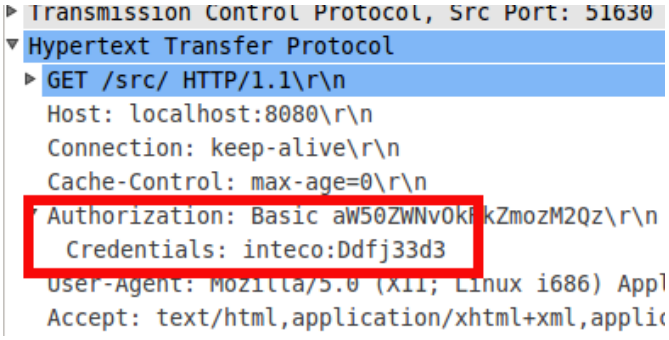
AUTENTIFICACIÓN BÁSICA

Cuando el usuario accede a un recurso del servidor web protegido mediante autentificación básica, tiene lugar el siguiente proceso:

1. El navegador presenta al usuario la ***ventana de autentificación***, para que introduzca su nombre y contraseña.
2. El navegador intenta establecer una conexión con el servidor utilizando esta información.
3. Si el servidor rechaza la información de autentificación, el navegador le presenta nuevamente la ventana al usuario hasta que éste introduce por fin una contraseña válida o cierra la ventana.
4. Cuando el servidor web verifica con éxito los datos de autentificación, se establece la conexión de acceso al recurso protegido.



El gran inconveniente *de este método es que la* ***contraseña viaja en texto "plano"*** *desde el navegador del usuario hasta el servidor, por lo que cualquier atacante con un* ***sniffer*** *podrá interceptarla inmediatamente.*



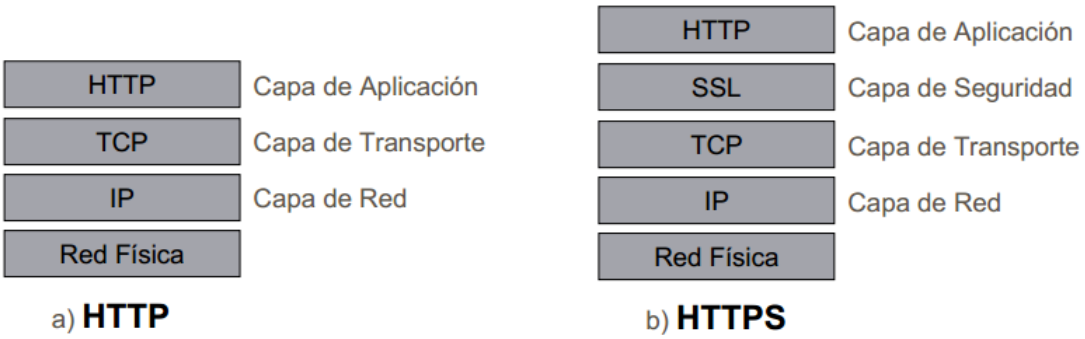
AUTENTIFICACIÓN MEDIANTE DIGEST (RESÚMENES)

Es evidente que el método anterior, no resulta muy adecuado cuando las exigencias de seguridad son elevadas. Para paliar este inconveniente, una alternativa consistiría en enviar un ***resumen criptográfico de la contraseña (un hash, utilizando algoritmos de 128 bits – MD5)*** en vez de la propia contraseña en texto plano.

Esta autentificación para ***evitar ataques de repetición***, que permitirían a un atacante autentificarse reenviando un hash capturado, en el proceso ***se incluye una cadena de texto aleatoria (llamada "nonce") que envía el servidor en la solicitud de autentificación, que se incluye en la generación del hash***. Como el hash enviado cambia a cada petición, sólo es válido una vez.

AUTENTIFICACIÓN HTTPS

HTTPS: protocolo (*las URLS comienzan por* ***https://***) que utiliza ***SSL*** (***Secure Socket Layer)*** para transportar mensajes HTTP por medio del ***puerto 443.***



SSL permite:

- Al cliente SSL, ***autentificar la identidad del servidor SSL.***
    - Utilizando técnicas de criptografía de llave pública, comprueba que el ***certificado del servidor es válido*** y ha sido ***avalado por una autoridad certificadora (CA)***.

- Al servidor SSL, ***autentificar la identidad del cliente SSL*** (utilizando las mismas técnicas).

- Establecer una ***conexión encriptada*** entre ambas máquinas:
    - Encriptación asimétrica RSA (clave pública).
    - Encriptación simétrica.

- Asegura la **integridad** de los mensajes:
        - Además, la información encriptada es protegida con un mecanismo que detecta si ésta fue alterada durante su tránsito por la red.