

# WWW y el protocolo HTTP

## 1. INTRODUCCIÓN

- El **Protocolo de Transferencia de HiperTexto** (*HTTP*) es otro protocolo de la capa de *Aplicación*.
- Permite a los usuarios el acceso a la información hipermedia remota de sistemas conectados a una red TCP/IP.
- La **WWW** (*World Wide Web*) es un servicio de distribución de información que permite acceder a millones de recursos y aplicaciones en equipos identificados por direcciones **URIs** o **URLs**.
- La *WWW* fue desarrollada por el **CERN** (*Centro Europeo de Investigación Nuclear*) en 1989 y actualmente desarrollada por el **W3C** (*World Wide Web Consortium*) que, además, desarrolla estándares como *XHTML*, *CSS* o *XML*.

# WWW y el protocolo HTTP

## 1. INTRODUCCIÓN

- El sistema *HTTP* se basa en el **modelo cliente/servidor** y está formado por los siguientes componentes:
  - **Recursos:** documentos, imágenes, vídeos, aplicaciones, etc., accesibles a través de servidores web y conectados por **hiperenlaces**.
  - **Nombres y direcciones (URIs y URLs):** sistema de nombres basado en cadenas de caracteres que identifican unívocamente y localizan a los recursos en la web.
  - **Clientes Web o HTTP:** aplicaciones que establecen conexiones y dialogan con los servidores web e interpretan y muestran al usuario la información recibida.
  - **Servidores Web o HTTP:** aplicaciones que atienden las peticiones de los clientes y les envían los recursos solicitados.
  - **Protocolo HTTP:** conjunto de normas y reglas en base a las cuales “dialogan” los clientes, los servidores web y los proxies.

# WWW y el protocolo HTTP

## 1. INTRODUCCIÓN

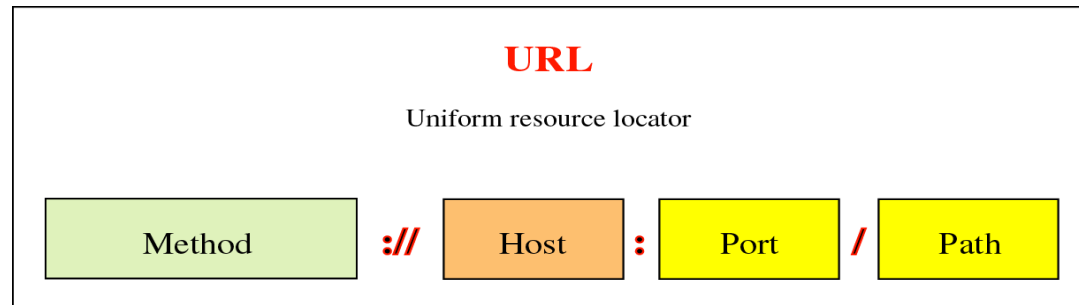
- **Proxies Web o HTTP:** aplicaciones intermedias entre clientes y servidores que actúan de cortafuegos y/o almacenamiento *caché* para aumentar el rendimiento.
- **Tecnologías Web:** utilizadas para desarrollar aplicaciones basadas en la web:
  - CSS
  - XML
  - Ajax
  - Xquery
  - Xpath
  - RDF
  - <http://www.w3c.es/Divulgación/GuiasBreves>
- El puerto bien conocido asociado al servicio es el **TCP/80**, si bien esto es configurable.

**Accesibilidad**  
**CSS**  
Estándares Web  
Independencia de Dispositivo  
Internacionalización  
**Interacción Multimodal**  
Linked Data  
Política de Patentes del W3C  
Privacidad y P3P  
Seguridad  
Servicios Web  
Tecnologías Multimedia  
Tecnologías XML  
**Web Móvil**  
Web Semántica  
**XForms**  
**XHTML**

# WWW y el protocolo HTTP

## 2. URLs

- Para facilitar el acceso a los documentos distribuidos a través del mundo, *HTTP* utiliza el concepto de *localizadores*.
- El ***Localizador Uniforme de Recursos (URL Uniform Resource Locator)*** es un estándar para especificar cualquier tipo de información en Internet.
- Una *URL* define cuatro elementos: *método o protocolo*, *host*, *puerto* y *camino/ruta*.



*http*  
*http*  
*ftp*  
*smb*  
*https*

*[www.google.es](http://www.google.es)*  
*localhost*  
*194.20.10.1*  
*[www.google.es](http://www.google.es)*  
*aula2DAW.com*

*80*  
*5050*  
*8080*  
*8000*

*index.htm*  
*aplic/sghd.htm*  
*apex*

## 3. CLIENTES WEB (Navegadores)

- Existen diversos tipos de clientes *Web*, tanto para sistemas libres como para sistemas propietarios.
  - Internet Explorer
  - Mozilla Firefox
  - Google Chrome
  - Safari
  - Opera
  - Lynx
  - ...
- En la actualidad existe una fuerte competencia entre las empresas que desarrollan navegadores web con el objeto de aumentar su cuota de mercado
  - Amplían su funcionalidad con la instalación de plantillas, idiomas, extensiones y complementos.

## 4. SERVIDORES WEB

- En cuanto a los servidores *WEB* también son tanto para sistemas libres como para sistemas propietarios.
  - Apache HTTP Server
  - IIS
  - Nginx
  - Google sites
  - Lighttpd

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

- Es un protocolo que utiliza TCP como protocolo de transporte y determina los tipos y peticiones que los clientes pueden enviar, así como el formato y estructura de las respuestas.
- También define una estructura de metadatos en forma de cabeceras que se envían tanto en las peticiones como en las respuestas.
- Ha pasado por diversas versiones HTTP/0.9 (obsoleta), HTTP/1.0, HTTP/1.1 (~~versión actual en RFC 2616~~) y HTTP/1.2 (~~experimental~~). HTTP/2 y el futuro HTTP/3

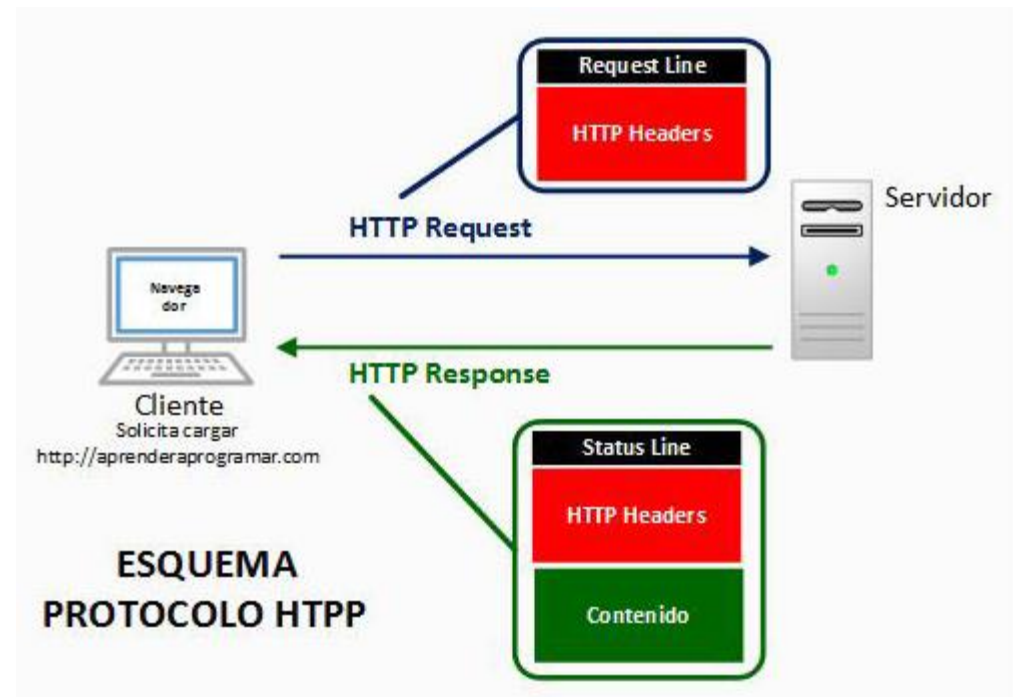
<https://www.somostechies.com/que-es-http2/>

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.1 Mensajes HTTP

- Los mensajes *HTTP* se encuentran en formato *ASCII* (**texto plano**).
- Hay dos tipos de mensajes:
  - Mensajes de petición
  - Mensajes de respuesta.



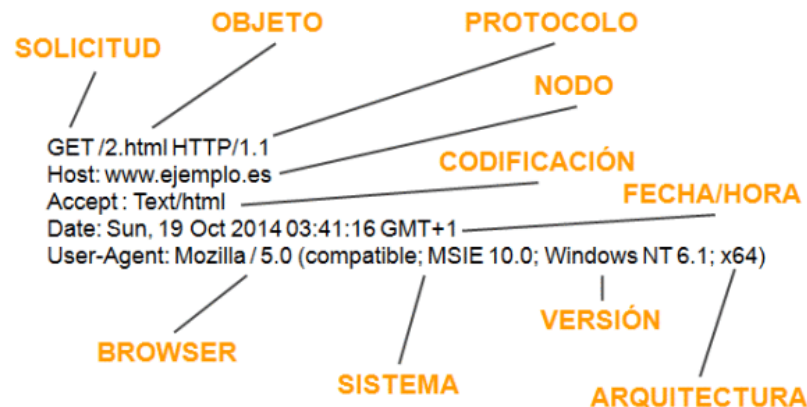


# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.1 Mensajes HTTP. Mensajes de petición.

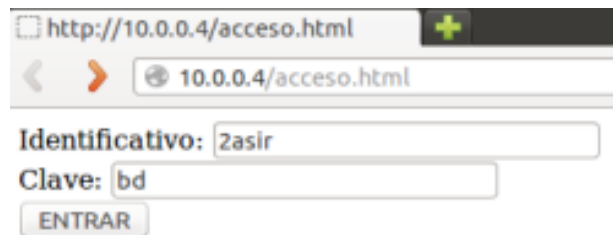
- Están formados por tres partes:
  - Línea inicial de petición: método, ruta relativa, protocolo/versión.
  - Línea(s) de cabecera.
  - Cuerpo del mensaje (opcional). Parámetros o ficheros a enviar al servidor.



# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.1 Mensajes HTTP. Mensajes de petición.



Escriba el texto aquí

Probar desde el sistema `curl http://www.google.es -v`

```
GET /procesar_acceso.php?identificativo=2asir&clave=bd HTTP/1.1
Host: 10.0.0.4
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:25.0)
Gecko/20100101 Firefox/25.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://10.0.0.4/acceso.html
Connection: keep-alive

HTTP/1.1 200 OK
Date: Mon, 18 Nov 2013 22:51:38 GMT
Server: Apache/2.2.22 (Win32) PHP/5.4.3
X-Powered-By: PHP/5.4.3
Content-Length: 124
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

SELECT nombre FROM credenciales
WHERE identificador='2asir' and
... clave='bd'; <br>No AUTORIZADO.
```

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.2 Mensajes HTTP. Mensajes de respuesta.

- Están formados por tres partes:
  - Línea inicial de respuesta (línea de estado): versión HTTP, código de estado y texto explicativo.
  - Línea(s) de cabecera.
  - Cuerpo del mensaje (opcional). Determinado por el tipo de recurso solicitado.

```
HTTP/1.1 200 OK
Date: Tue, 19 Nov 2013 11:00:10 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Sun, 17 Nov 2013 20:33:20 GMT
ETag: "43844-11a-4eb655791616a"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 216
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
.....}....0..g..&{....v.....&.hDI..m.B<..r/v.t>/.
```

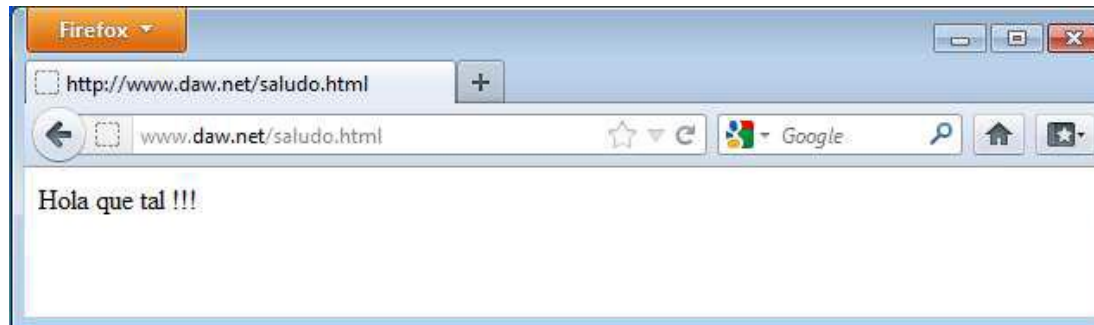
Diagram illustrating the structure of an HTTP response line with labels pointing to its components:

- PROTOCOLO (points to HTTP/1.1)
- CÓDIGO DE ESTADO (points to 200)
- FECHA RESPUESTA (points to OK)
- SERVIDOR / S.O. (points to Apache/2.2.22 (Ubuntu))
- ULT. MODIF. (points to Last-Modified: Mon, 14 Oct 2013 10:06:55 GMT)
- CONT. ASOCIADO (points to ETag: "66091b-b1-4e8b72f36510b")
- TIPO DE RANGO (points to Accept-Ranges: bytes)
- TIPO MIME (points to Content-Type: text/html)
- RESPUESTA SELECCIONADA (points to Content-Length: 177)
- LONGITUD (points to Content-Length: 177)

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.2 Mensajes HTTP. Mensajes de respuesta.



```
GET /saludo.html HTTP/1.1
Host: www.daw.net
User-Agent: Mozilla/5.0 (windows NT 6.1; rv:12.0) Gecko/20100101 Firefox/12.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-es,es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Date: Fri, 27 Apr 2012 07:40:28 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Fri, 27 Apr 2012 07:40:10 GMT
ETag: "20016-20-4bea436cd2425"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 48
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

```
.....(.....IT(MU(I.QPTT....H....j). ...|
```

## 5. PROTOCOLO HTTP

### 5.3 Métodos de petición.

- Especifican la operación que quiere el cliente realizar en el servidor.
  - GET: solicita un documento al servidor. Se pueden enviar datos en la URL.
  - HEAD: similar a GET, pero solo pide las cabeceras HTTP. Se utiliza para comprobar enlaces o para consultar información del fichero (fecha, tamaño, tipo de servidor, ...) antes de solicitarlo.
  - POST: Manda datos al servidor para su procesado.
    - Similar a GET, pero además envía datos en el cuerpo del mensaje.
    - La URL corresponde a un página dinámica que trata los datos enviados.

## 5. PROTOCOLO HTTP

### 5.3 Métodos de petición.

- PUT: Almacena el documento enviado en el cuerpo del mensaje.
- OPTIONS: Averigua los métodos que soporta el servidor.
  - En una caché sólo se guardan las respuestas de las peticiones realizadas con GET y HEAD (POST no)
- DELETE: Elimina el documento referenciado en la URL.
- TRACE: Rastrea los intermediarios por los que pasa la petición.

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

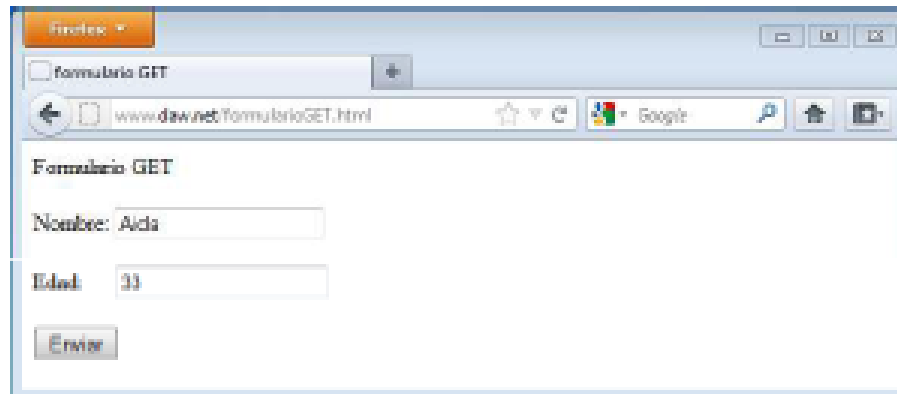
### 5.3 Métodos de petición. GET

- Se emplea para obtener cualquier tipo de información del servidor
- Se invoca normalmente cuando:
  - Se introduce una **URL en el navegador**.
  - Se hace clic sobre un **hiper enlace**.
  - Se envía un **formulario GET**.
- Permite enviar parámetros al servidor en la URI (o URL) conocidos como *Query String*.
  - <http://datosGET.php?nombre=Ana&edad=29>

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.3 Métodos de petición. GET



```
GET /datosGET.php?nombre=Aida&edad=33&Enviar=Enviar HTTP/1.1
Host: www.daw.net
User-Agent: Mozilla/5.0 (windows NT 6.1; rv:12.0) Gecko/20100101 Firefox/12.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-es;es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.daw.net/formularioGET.html
```



## 5. PROTOCOLO HTTP

### 5.3 Métodos de petición. GET

- Las peticiones GET no envían cuerpo de mensaje.
- El tamaño de la información enviada está limitada.
- No se puede usar para subir o realizar otras operaciones que requieran enviar una gran cantidad de datos al servidor.

## 5. PROTOCOLO HTTP

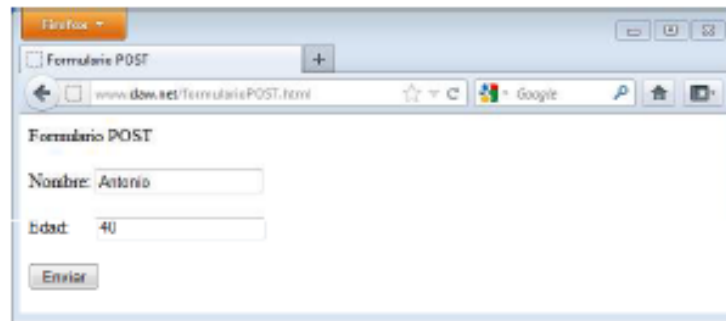
### 5.3 Métodos de petición. POST

- Para solicitar al servidor que acepte información que se envía adjunta en una petición.
- Las peticiones POST envían el cuerpo de mensaje.
- Los parámetros no son visibles por lo tanto en la URL.
- Se invoca normalmente como consecuencia de enviar un formulario POST.

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.3 Métodos de petición. POST



```
POST /datosPOST.php HTTP/1.1
Host: www.daw.net
User-Agent: Mozilla/5.0 (windows NT 6.1; rv:12.0) Gecko/20100101 Firefox/12.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-es,es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.daw.net/formularioPOST.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 36

nombre=Antonio&edad=40&Enviar=Enviar
```

## 5. PROTOCOLO HTTP

### 5.4 Cabeceras

- Parámetros que se envían en cada petición/respuesta HTTP
  - Proporcionan información adicional en formato **Cabecera-valor** y enviadas automáticamente por el navegador o el servidor web.
- Múltiples tipos de cabeceras
  - **Generales** (*Date, Transfer-Encoding, ...*).
  - **De petición (o de cliente)** (*User-Agent, Accept, ...*).
  - **De respuesta (o de servidor)** (*Server, Age, ...*)
  - **De entidad** (*Content-Encoding, Content-Language, Content-Type, ...*)

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.4 Cabeceras

<https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html#sec5.3>

#### ➤ Web

- <http://www.w3.org/Protocols/rfc2616/rfc2616.html>

```
<?php
$headers = apache_request_headers();

foreach ($headers as $header => $value) {
    echo "$header: $value <br />\n";
}
?>
```

```
Host: localhost
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Sec-Fetch-Site: none
Accept-Encoding: gzip, deflate, br
Accept-Language: es-ES,es;q=0.9
```

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.5 Códigos de estado y error

<https://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6>

- Códigos que envían los servidores en las respuestas *HTTP*.

```
HTTP/1.1 404 Not Found
Date: Tue, 19 Nov 2013 11:09:05 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Sun, 17 Nov 2013 21:35:04 GMT
ETag: "46e0e-1a2-4eb6634625fb2;4eb6673542494"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 267
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

```
HTTP/1.1 500 Internal Server Error
Date: Tue, 19 Nov 2013 11:14:17 GMT
Server: Apache/2.2.22 (Ubuntu)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 394
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

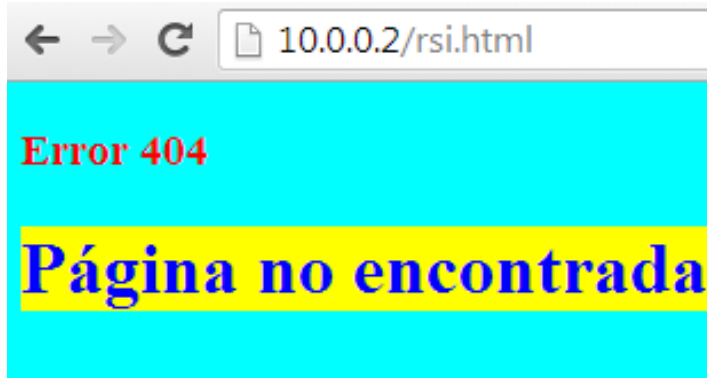
- Informan al cliente de cómo ha sido procesada la petición.
- Se acompañan de un texto explicativo.

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.5 Códigos de estado y error

- Código de 3 dígitos que se clasifican en función del primero.
  - 100 - 199 (Informativo, Informational).
  - 200 - 299 (Éxito, Successful).
  - 300 - 399 (Redirección, Redirection).
  - 400 - 499 (Errores del cliente, Client Error).
  - 500 - 599 (Errores en el servidor, Server Error)



```
HTTP/1.1 404 Not Found
Date: Tue, 19 Nov 2013 11:09:05 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Sun, 17 Nov 2013 21:35:04 GMT
ETag: "46e0e-1a2-4eb6634625fb2;4eb6673542494"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 267
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.6 Cookies

- Una **cookie** es un fragmento de información que envía un servidor web en una respuesta *HTTP* y es almacenada por el navegador.

Ejer-3.php	
Name	Value
<b>Request Cookies</b>	
contador	4
visita	04%2F10%2F2019+10%3A02%3A16
<b>Response Cookies</b>	
contador	5
visita	04%2F10%2F2019+10%3A02%3A18

- El navegador puede enviar la *cookie* en solicitudes posteriores al mismo servidor.



# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.6 Cookies

#### ➤ Cabeceras: Cookies y Set-Cookie

##### ▼ Request Headers [view source](#)

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Accept-Encoding: gzip, deflate, br
Accept-Language: es-ES,es;q=0.9
Cache-Control: max-age=0
Connection: keep-alive
Cookie: visita=04%2F10%2F2019+10%3A02%3A16; contador=4
Host: localhost
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36
```

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.6 Cookies

#### ➤ Cabeceras: Cookies y Set-Cookie

HTTP/1.1 200 OK

Date: Fri, 04 Oct 2019 20:02:18 GMT

Server: Apache/2.4.33 (Win32) OpenSSL/1.1.0h PHP/7.2.6

X-Powered-By: PHP/7.2.6

Set-Cookie: visita=04%2F10%2F2019+10%3A02%3A18; expires=Fri, 04-Oct-2019 21:02:18 GMT; Max-Age=3600

Set-Cookie: contador=5; expires=Fri, 04-Oct-2019 21:02:18 GMT; Max-Age=3600

Content-Length: 616

Keep-Alive: timeout=5, max=100

Connection: Keep-Alive

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

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.7 Sesiones

- *HTTP* es un protocolo “sin estado”.
  - Cada transferencia de datos es independiente de la anterior sin ninguna relación entre ellas. Por cada objeto que se transfiere por la red se realiza una conexión independiente.
- Técnicas para mantener la sesión
  - *Cookies*
  - *URL Rewriting*

<https://josepablosarco.wordpress.com/2009/09/01/http-url-re-writing-modifier/>
  - Campos ocultos en formularios.

`<input type="hidden" id="custId" name="custId" value="3487">`

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

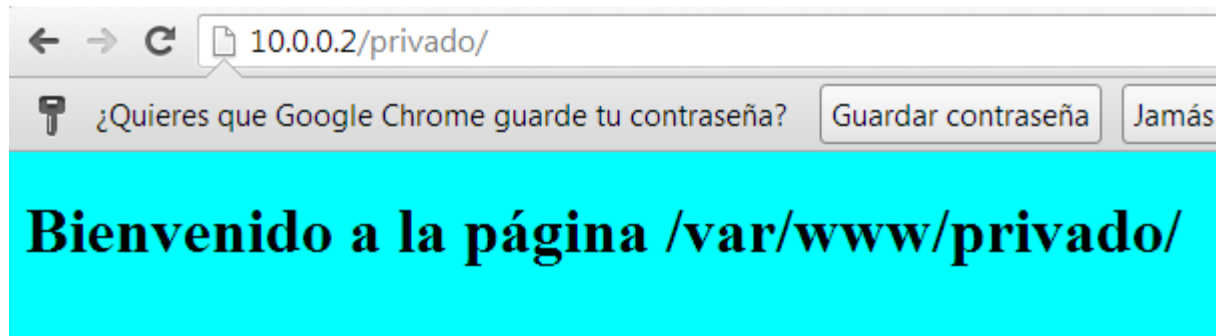
### 5.8 Autenticación

- Mecanismos de autenticación para controlar el acceso a los recursos que ofrece el servidor.
  - **Basic**: el cliente envía un usuario y una clave codificados con el algoritmo *base64*. Método no seguro. Es trivial capturar y obtener la clave encriptada.
  - **Digest**: el cliente envía un usuario y una función hash (resumen) de la clave al servidor utilizando el algoritmo *MD5*. Es más seguro que el método *Basic* pero también es vulnerable a ataques y poco seguro.
- Basados en:
  - Código de estado 401.
  - Cabeceras *WWW-Authenticate* y *Authorization*.

# WWW y el protocolo HTTP

## 5. PROTOCOLO HTTP

### 5.8 Autenticación



```
HTTP/1.1 401 Authorization Required
Date: Tue, 19 Nov 2013 11:22:55 GMT
Server: Apache/2.2.22 (Ubuntu)
www-Authenticate: Basic realm="Acceso restringido"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 337
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
```

## 5. PROTOCOLO HTTP

### 5.9 Almacenamiento en caché

- Muchos servidores web almacenan información que no cambian durante periodos de tiempo muy largos.
- *HTTP* soporta almacenamiento en **caché** para evitar tráfico innecesario y así aumentar el rendimiento.
- Se pueden definir cabeceras que permiten controlar qué se almacena en caché, durante cuánto tiempo, qué no se puede almacenar, si la información se puede cachear en un proxy o no, si hay que actualizar algo en caché porque se ha modificado, etc.
  - *Cache-Control, Last-Modified, Expires, Age, Etag, If-Match, If-Modified-Since, ...*

## 5. PROTOCOLO HTTP

### 5.10 Redireccionamiento

- *HTTP* permite a los servidores y proxies redirigir las peticiones a otras localizaciones.
- Algunas situaciones en las que puede ser útil redireccionar:
  - El contenido de un servidor se ha movido a una URI o URL distinta.
  - Convertir una solicitud POST en una solicitud GET.
  - Redirigir las peticiones a otro proxy.
  - Redirigir peticiones a servlets, JSPs, ASPs, PHP, etc.

## 5. PROTOCOLO HTTP

### 5.11 Conexiones persistentes

- Consiste en que varias peticiones y respuestas sean transferidas usando la misma conexión TCP.
- Su uso reduce el número de conexiones TCP, lo que redundará en menor gasto de CPU/memoria y en una reducción de los tiempos de respuesta.
- Se definen varias cabeceras y temporizadores en servidores y clientes para controlar cuándo hay que cerrar conexiones TCP.
  - *Connection, Content-Length,...*



## 6. ALOJAMIENTO VIRTUAL

- *Web Virtual Hosting* consiste en simular que existen varias máquinas (hosts) con sus respectivos sitios web sobre un solo servidor web, es decir, alojar varios sitios web (p.e., [www.daw1.es](http://www.daw1.es), [www.daw2.es](http://www.daw2.es)) en un mismo servidor.
- El uso de servidores web virtuales permite reducir el número de máquinas físicas necesarias para alojar los millones de sitios web que existen en Internet y al mismo tiempo aprovechar mejor los recursos (uso de CPU, memoria, ...) de los equipos.

HTTP/2 y HTTP/3

- Hay 3 tipos de alojamientos virtuales:
  - Basado en **IPs**.
  - Basado en **nombres**.
  - Basado en **puertos**.

# WWW y el protocolo HTTP

## 6. ALOJAMIENTO VIRTUAL

### 6.1 Basado en IPs

- El servidor tendrá diferentes direcciones IP por cada servidor web virtual.
  - Cada servidor virtual atenderá peticiones en una dirección IP diferente.
  - A efectos de los usuarios es como si existiesen varios servidores web, uno por cada dirección IP.

<http://197.100.200.101> y  
<http://197.100.200.201>

Configuración de  
subinterfaces en Linux

```
# interfaces(5) file used by ifup(8)
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.0.0.2
netmask 255.255.255.0
gateway 10.0.0.1
dns-nameservers 8.8.8.8
# dns-nameservers 192.168.0.1

auto eth0:0
iface eth0:0 inet static
address 10.0.0.101
netmask 255.255.255.0
gateway 10.0.0.1
dns-nameservers 8.8.8.8
```

# WWW y el protocolo HTTP

## 6. ALOJAMIENTO VIRTUAL

### 6.1 Basado en IPs

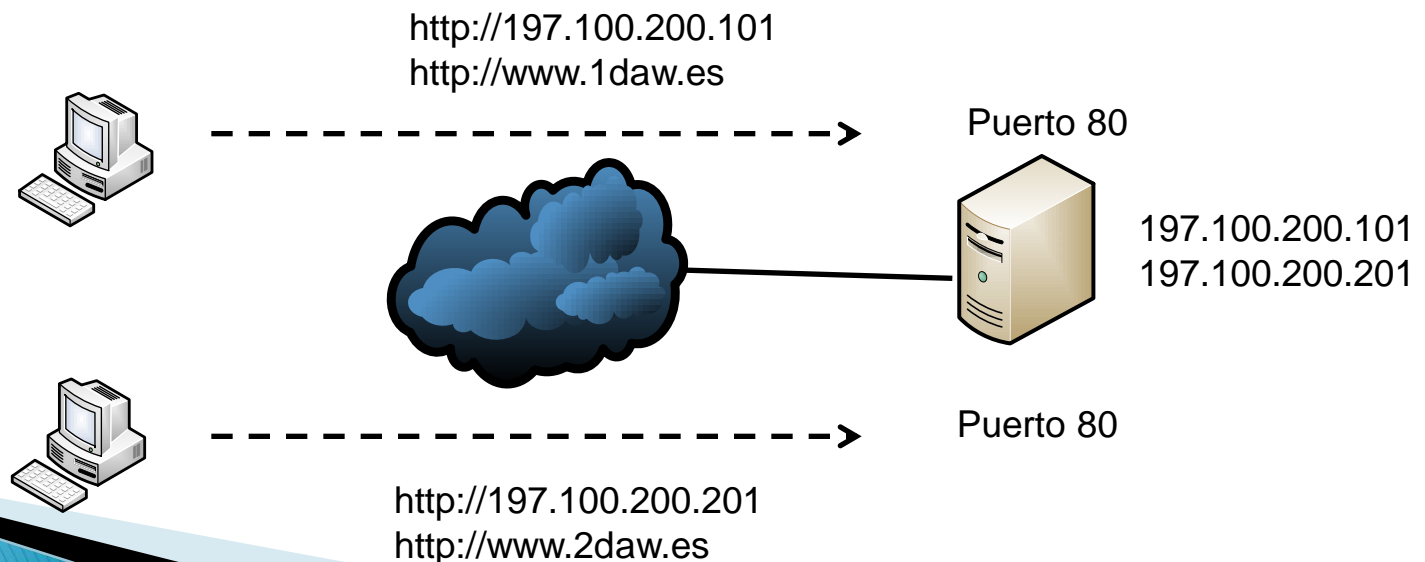
- Si se configura un DNS que tenga los siguientes registros:

*www.1daw.es IN A 192.100.200.101*

*www.2daw.es IN A 192.100.200.201*

- Se podrán realizar las peticiones:

- <http://www.1daw.es> y <http://www.2daw.es>



## 6. ALOJAMIENTO VIRTUAL

### 6.2 Basado en nombres

- Por ejemplo, un equipo con dirección IP 197.100.200.101 y un servidor DNS con los siguientes registros:

*www.1daw.es IN A 197.100.200.101*

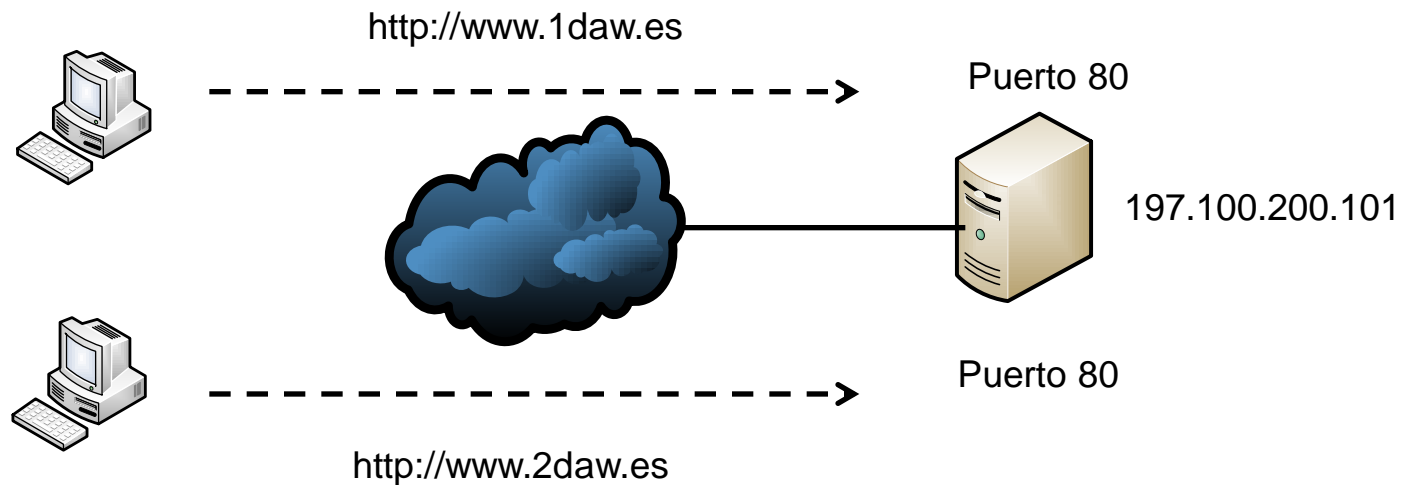
*www.2daw.es IN A 197.100.200.101*

- Se podrán realizar las peticiones:
  - <http://www.1daw.es> y <http://www.2daw.es>

# WWW y el protocolo HTTP

## 6. ALOJAMIENTO VIRTUAL

### 6.2 Basado en nombres



- Es la **forma de alojamiento más utilizada**
  - Al permitir alojar varios dominios en un único equipo y dirección IP, se ahorra tanto en número de equipos como en direcciones IP.
  - Además se simplifica la administración centralizada de los servidores.

# WWW y el protocolo HTTP

## 6. ALOJAMIENTO VIRTUAL

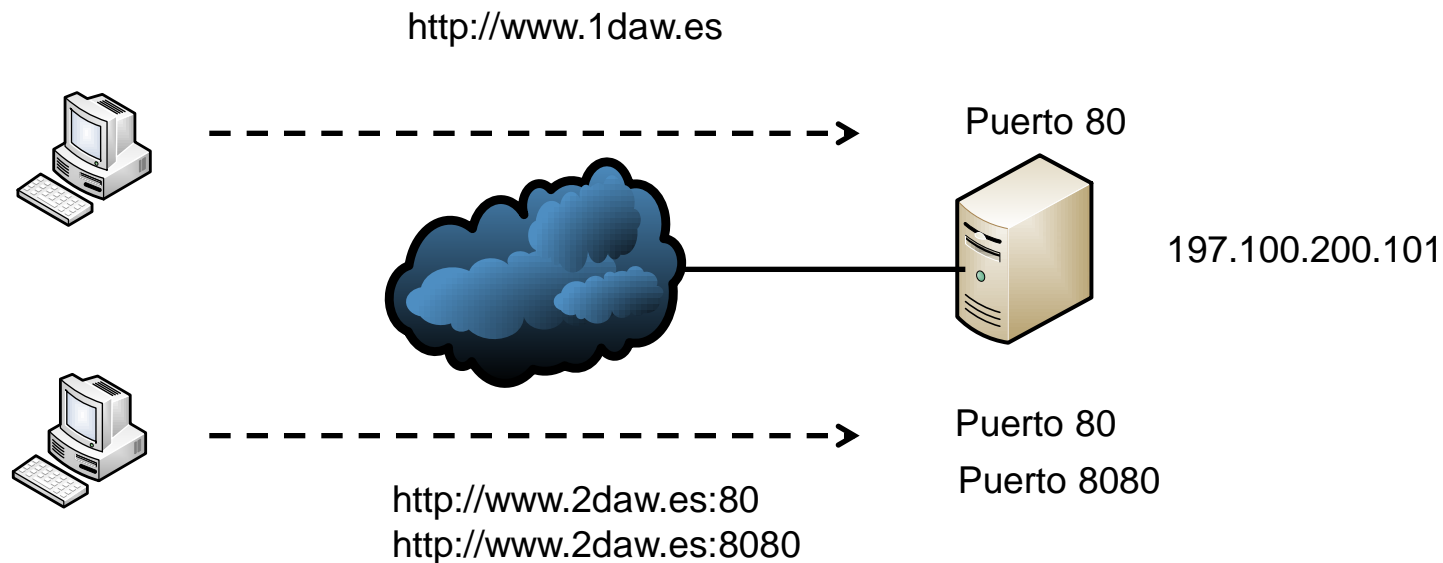
### 6.3 Basado en puertos

- El servidor virtual atiende peticiones en una dirección/IP o nombre de dominio y un puerto diferentes.
- Por ejemplo, un equipo con dirección IP 197.100.200.101 y un servidor DNS con los siguientes registros:  
*www.1daw.es IN A 197.100.200.101*  
*www.2daw.es IN A 197.100.200.101*
- Se configura, además, el servidor web para que atienda las peticiones:
  - <http://www.1daw.es:8080>
  - <http://www.2daw.es:80>
  - <http://www.2daw.es:8080>

# WWW y el protocolo HTTP

## 6. ALOJAMIENTO VIRTUAL

### 6.3 Basado en puertos



## 6. ALOJAMIENTO VIRTUAL

### 6.4 Combinaciones

- En un mismo servidor web se pueden combinar servidores virtuales basados en IPs, en nombres y en puertos.
- El servidor web *Apache* admite la creación de todos los tipos de servidores virtuales.



# WWW y el protocolo HTTP

## 7. SEGURIDAD

- *HTTP* no es un protocolo seguro.
- El intercambio de información se realiza en *texto plano*, por lo que es vulnerable a ataques de *análisis de tráfico de red*.
- No se usan mecanismos para garantizar que los equipos involucrados en la transferencia sean quien dicen ser. Son vulnerables a ataques del tipo *spoofing* (suplantación) y *man-in-the-middle*.
- Existen ataques que se basan en el robo o falsificación de las cookies y/o parámetros enviados en la URL o en el contenido de los mensajes y que permiten al atacante robar la identidad de un usuario y suplantarle en webs: bancos, webmails, redes sociales,

...

## 7. SEGURIDAD

- Actualmente existen una gran cantidad de servicios y aplicaciones basados en la Web: redes sociales, comercio electrónico, banca por internet, wikis, blogs, etc., con una gran repercusión económica y social.
- Por ello, todos los componentes que intervienen en la arquitectura de la Web (protocolos, clientes, servidores, aplicaciones, etc.) son objeto constante de ataques.

# WWW y el protocolo HTTP

## 7. SEGURIDAD. PROTOCOLO HTTPS

- **HTTPS** (*HTTP Secure*) es un protocolo que utiliza **SSL** (*Secure Sockets Layer*) o **TLS** (*Transport Layer Security*) para encapsular mensajes *HTTP*.
- Gracias a la utilización de algoritmos criptográficos y certificados digitales se puede garantizar la confidencialidad y la integridad de la información transmitida, así como la autenticidad de los servidores.
- Los clientes utilizan en la *URL* o *URI* **https://**
- Los servidores web utilizan por defecto el puerto **TCP/443** para escuchar las peticiones de los clientes.

# WWW y el protocolo HTTP

## 7. SEGURIDAD. PROTOCOLO HTTPS

- **HTTPS** (*HTTP Secure*) es un protocolo que utiliza **SSL** (*Secure Sockets Layer*) o **TLS** (*Transport Layer Security*) para encapsular mensajes *HTTP*.

