Grado en Ingeniería de Software

Laboratorio de Bases de Datos y Sistemas Distribuidos

Elena G. Gamella

Servicio de Navegación WEB

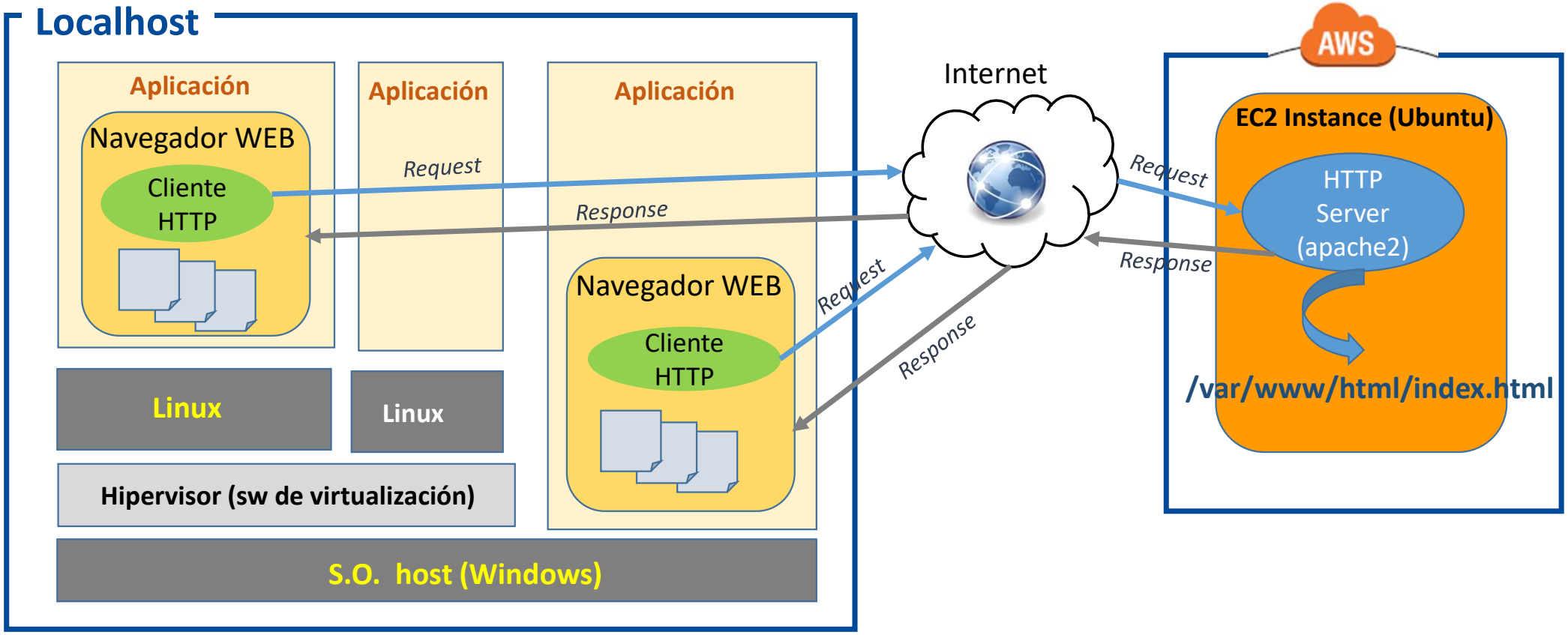
Escenarios de instalación de MariaDB

El objetivo de este documento es entender el servicio de navegación WEB utilizando Apache y PHP para cargar páginas Web estáticas y dinámicas.

Implementaremos el servicio utilizando los recursos de computación en la nube.

Servicio de Navegación WEB utilizando el servicio EC2 de AWS

Instalación Apache (servidor HTTP) en instancia EC2



Servidor de Navegación Web

Pruebas de servidor web (apache2) en instancia EC2 (Ubuntu)

1. Conectar desde nuestra máquina local Linux a nuestra instancia de EC2:
$ ssh -i "fichero_claves.pem" usuario@dns_publico_de_la_máquina

El servidor web “apache 2” viene instalado si usamos AMIs del entorno cloud9Ubuntu. Si no viniera instalado, ir al capítulo siguiente en el que se explica cómo hacer la instalación de apache.

2. Comprobamos el estado del servidor web apache:
$ sudo systemctl status apache2

También podemos probar a conectar desde un navegador para ver que se nos carga la página por defecto. Escribir en un navegador: http://dns_publico_de_nuestra maquina_ec2/

3. Nos movemos al directorio raíz de documentación pública del servidor (desde desde dónde apache sirve los archivos o páginas web).
$ cd /var/www/html

(*) NOTA: apache 2 o httpd (dependiendo de la distribución de linux) *sirve archivos que se guardan en un directorio denominado raíz de documentos de Apache. La raíz de documentos de Apache de Amazon Linux es /var/www/html, que es propiedad del directorio raíz de manera predeterminada.*

Servidor de Navegación Web

4. Crearemos ahí un subdirectorio para alojar nuestras páginas web. No es necesario, se puede hacer para mayor organización de los contenidos. Si no creas el directorio “pruebadir” puedes probar a crear las páginas web directamente en el directorio raíz de documentación del servidor (/var/www/html)

```
$ pwd
/var/www/html
$ sudo mkdir pruebadir
```

5. Cambiamos la propiedad del directorio que hemos creado para que el OWNER sea el usuario "ubuntu“ (nuestro usuario en ec2)

```
$ sudo chown ubuntu pruebadir
```

6. Modificamos los permisos de forma recursiva para añadir al resto de usuarios "others" y sus ficheros/directorios el permiso de "r“ (si no es así no podremos cargar las páginas desde el navegador)

```
$ sudo chmod -R o+r pruebadir
```

7. Cambiamos al directorio creado:

```
$ cd pruebadir
```

8. Creamos un archivo HTML de ejemplo:

```
$ echo "<html><h1>hello from Amazon</h1></html>" > hello.html
```

9. Y nos conectamos desde un navegador (cada uno escribe el dns de su máquina y la ruta completa a la página):

http://dns_publico_de la_máquina_EC2/pruebadir/hello.html

Prueba de PHP para mostrar contenido dinámico

1. Conectar desde nuestra máquina Linux local a la máquina EC2:

```
$ ssh -i "fichero_claves.pem" usuario@dns_publico_de_la_máquina
```

El servidor PHP viene instalado si usamos AMIs del entorno cloud9Ubuntu. Si no viniera instalado, ir al capítulo siguiente en el que se explica cómo hacer la instalación de PHP.

2. En el directorio raíz de apache2: **/var/www/html/** creamos un fichero php de prueba "index.php" con el contenido indicado más abajo, debe tener permisos 644. Para crear el fichero puedes usar el editor vi o (puedes usar nano si lo prefieres)

```
$ cd /var/www/html
$ sudo vi index.php
        <?php
        phpinfo();
        php?>
```

3. Prueba que PHP funciona correctamente. Para ello conecta al servidor desde cualquier navegador. La función *phpinfo ()* nos da los detalles de la versión de php instalada y comprueba que apache y php están funcionando.

http://dns_publico_de la_máquina_EC2/index.php

A partir de aquí estamos en disposición de hacer actividades prácticas con páginas estáticas y dinámicas.

Instalación de apache y PHP

Instalación Servidor Web Apache y PHP

Proceso de Instalación y prueba de servidor web (apache2) en instancia EC2 (Ubuntu)

Conectar desde nuestra máquina local Linux a nuestra instancia de EC2:
$ ssh -i "*fichero_claves.pem*" *usuario@dns_publico_de_la_máquina*

Instalar el servidor web (*)
$ sudo apt-get -y install apache2
$ ps -ef | grep apache

Comprobamos el estado del servidor:
$ sudo systemctl status apache2

También podemos probar a conectar desde un navegador para ver que se nos carga la página por defecto. Escribir en un navegador: http://dns_publico_de_nuestra maquina_ec2/

Cambiamos al directorio raíz de documentación pública del servidor (desde dónde sirve los archivos)
$ cd /var/www/html

(*) NOTA: apache 2 o httpd (dependiendo de la distribución de linux) *sirve archivos que se guardan en un directorio denominado raíz de documentos de Apache. La raíz de documentos de Apache de Amazon Linux es /var/www/html, que es propiedad del directorio raíz de manera predeterminada.*

Instalación Servidor Web Apache y PHP

Instalación y prueba de PHP para mostrar contenido dinámico

Conectar desde nuestra máquina local Linux a nuestra instancia de EC2:
$ ssh -i "*fichero_claves.pem*" *usuario@dns_publico_de_la_máquina*

Instalamos PHP, el módulo PHP para apache2 (servidor web) y para mysql:
$ sudo apt-get install php libapache2-mod-php php-mysql

Rearancamos el servidor Web:
$ sudo systemctl restart apache2

La raíz web de apache2 es el directorio: **/var/www/html/**

Instalación Servidor Web Apache y PHP

Instalación y prueba de PHP para mostrar contenido dinámico

Editamos un fichero php de prueba index.php con el contenido indicado más abajo

$ cd /var/www/html

Nos aseguramos que el fichero tiene permisos 644 y lo editamos.

$ sudo vi index.php

```
<?php
phpinfo();
php?>
```

Y lo probamos conectando al servidor desde cualquier navegador. La función *phpinfo ()* nos da los detalles de la versión de php instalada y comprueba que apache y php están funcionando.

http://dns_publico_de_la_máquina_EC2/index.php

A partir de aquí estamos en disposición de hacer pruebas con tus propias páginas estáticas y dinámicas.

A.1

Repasamos conceptos...
Páginas WEB estáticas y Dinámicas

Páginas Web estáticas

Contienen información que no cambia hasta que el diseñador o programador la modifica manualmente.
Las páginas web estáticas, compuestas únicamente **de HTML y CSS**, y esto se puede realizar de una forma sencilla. Pero una de las grandes limitaciones de las páginas web estáticas es el esfuerzo que se requiere para actualizarlas. Cambiar un solo elemento en una página web estática requiere reconstruir y recargar en el servidor toda la página, o a veces incluso un grupo de páginas web.

Este proceso es demasiado engorroso para una organización que con frecuencia necesita publicar información en tiempo real, tal como eventos, información que se lee de una BBDD, etc. Además, es un proceso vulnerable de cometer fallos y se puede arruinar seriamente la información de la web, o incluso el diseño completo del sitio.

Páginas Web dinámicas

Permiten cambiar fácilmente su contenido en tiempo real sin siquiera tocar el código de la página. **Sin hacer manualmente cambios en la página, la información en la página puede variar**. Esto hace posible mantener el contenido de la página actualizado para que pueda ser modificado o substituido en cualquier momento. El diseño central de la página web puede seguir siendo el mismo, pero los datos presentan cambios constantes.

Para crear una página web dinámica, se debe conocer un método para insertar automáticamente datos en tiempo real en el código HTML que se envía al navegador del cliente. Aquí es donde entran en juego los lenguajes de script, que permiten insertar código de programa dentro de una web, que genera dinámicamente HTML que el navegador del cliente entiende.

Métodos de programación de Páginas Dinámicas

Como el código de programación está insertado en la página web, en alguna parte se debe ejecutar dicho código para producir el HTML dinámico para el nuevo contenido. Hay dos lugares donde se puede ejecutar el código de programa insertado:

- **En el equipo del cliente, después de que el navegador web descarga la página web. Esto se conoce como programación del lado del cliente**.

- **En el servidor web antes de que se envíe la página. Esto se conoce como programación del lado del servidor**.

Almacenamiento de contenido en páginas dinámicas - BBDD

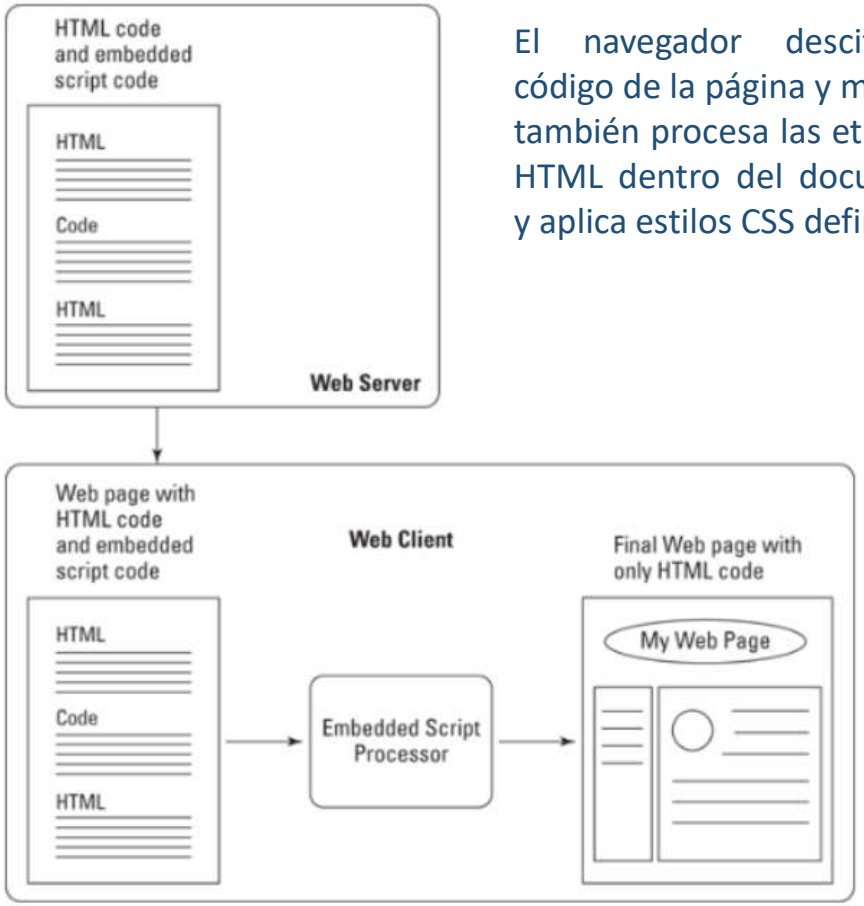
Con aplicaciones web dinámicas, el contenido proviene de algún lugar fuera de la página web. Lo normal es que sea de una base de datos. Las bases de datos son una forma fácil de almacenar y recuperar datos. Son más rápidas que el almacenamiento de datos con archivos estándar y proporcionan mayor nivel de seguridad para proteger sus datos. Al almacenar contenido en una base de datos, también se puede archivar y referenciar fácilmente el contenido anterior y reemplazarlo con contenido nuevo según sea necesario.

Programación del lado del cliente

En la programación del lado del cliente, se inserta el código del programa dentro del código HTML que el servidor envía al navegador del cliente. El navegador debe poder detectar el código del programa incorporado y ejecutarlo, ya sea dentro del navegador o como un programa separado fuera del navegador.

Algunos ejemplos son **JavaScript o JQuery**

Estos lenguajes de script se insertan dentro del código HTML en la página web y se ejecutan dentro del navegador del cliente y pueden utilizar funciones del navegador a las que normalmente no se puede acceder desde HTML estándar.



El navegador descifra el código de la página y mientras también procesa las etiquetas HTML dentro del documento y aplica estilos CSS definidos

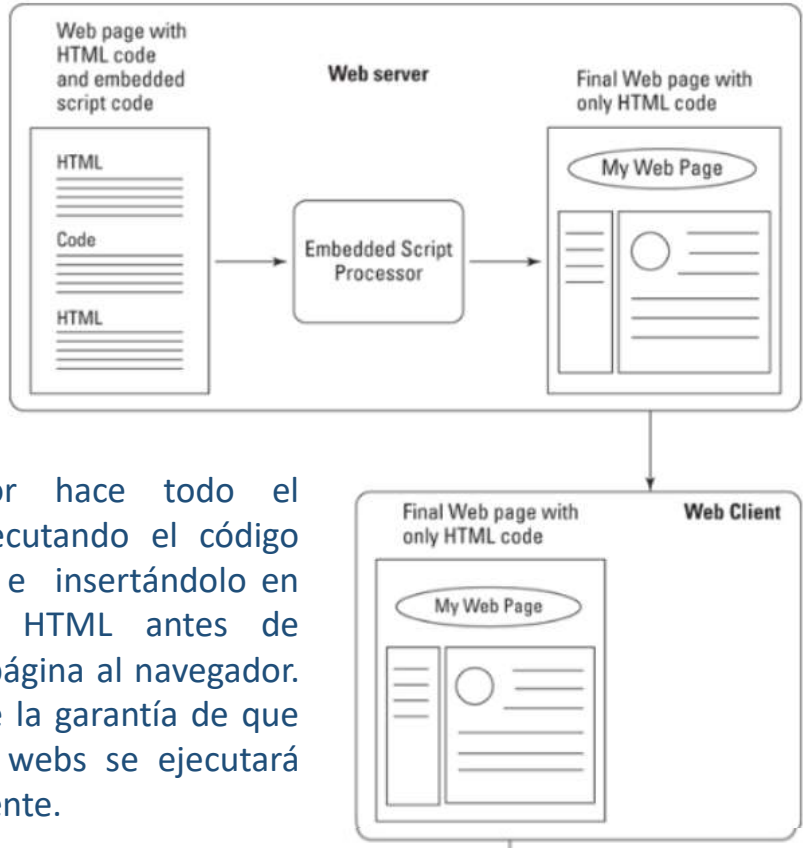
Programación del lado del servidor

Los lenguajes de programación del lado del servidor resuelven el problema ejecutando el código en el servidor. Es decir, **el servidor web interpreta el código de programación insertado antes de enviar la página web al navegador del cliente.**

En el caso de los lenguajes compilados, el servidor web envía el código al compilador y la salida de éste se envía al navegador del cliente como parte del documento HTML

Una vez hecho, el servidor toma el código HTML que el código de programación ha generado y lo inserta directamente en la página web antes de enviarlo al cliente (navegador).

Algunos de los lenguajes del lado del servidor son **Perl y Python** (de tipo CGI script), **JSP** (que necesita compilador de Java en el servidor web), **ASP.NET, Javascript, PHP,** ..



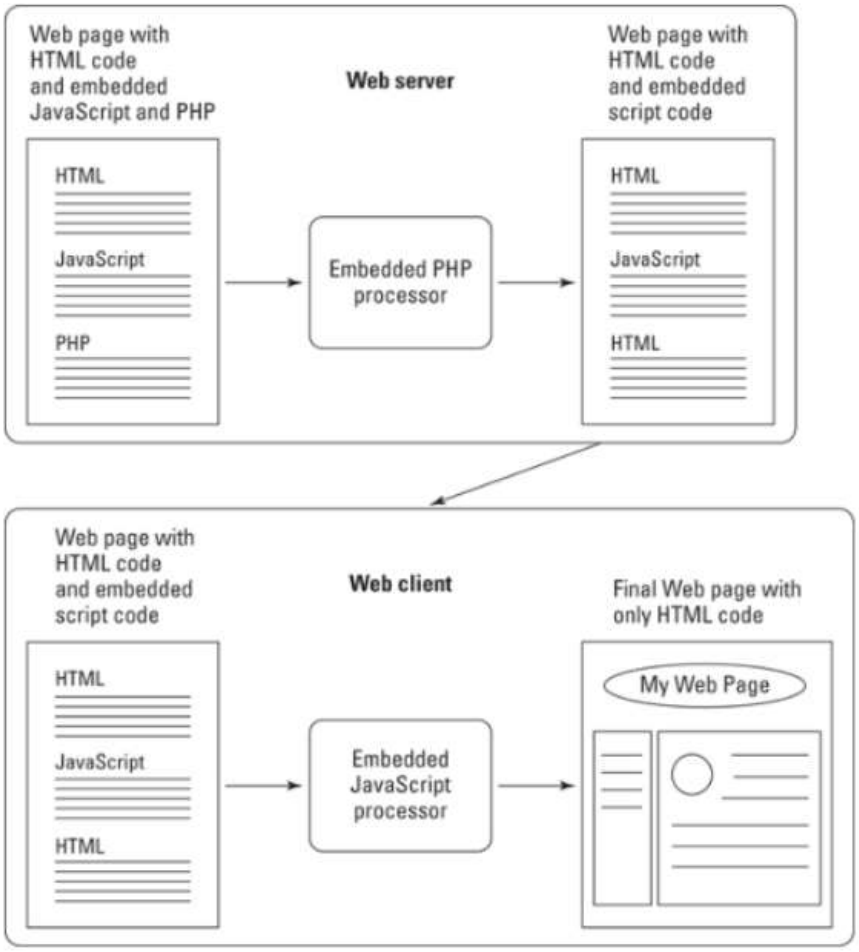
El servidor hace todo el trabajo ejecutando el código encriptado e insertándolo en el código HTML antes de enviar la ´página al navegador. Esto ofrece la garantía de que las página webs se ejecutará correctamente.

Programación mixta: del lado del cliente y del lado del servidor

La programación del lado del cliente y del lado del servidor tienen pros y contras. En lugar de tratar de elegir un método para crear páginas web dinámicas, se pueden utilizar ambos al mismo tiempo.

Se puede insertar fácilmente el código de programación del lado del cliente y del lado del servidor en la misma página web para ejecutar en el servidor, como se muestra a continuación.

Un ejemplo muy común es el de **JavaScript y PHP** por ejemplo para la validación de datos en un formulario HTML. Sólo cuando los datos están bien rellenos (lo controla Javascript) se envían al servidor para el análisis en PHP del resto de validaciones.



Anexo 2

Algunos conocimientos más sobre Servidores WEB y el protocolo HTTP

Servidores WEB: Arquitectura

La arquitectura utilizada es la denominada cliente/servidor, es decir, el equipo cliente hace una solicitud o petición al equipo servidor, y éste atiende dicha solicitud.

Cliente

- En el equipo cliente se ejecuta la aplicación cliente, es decir, el 'navegador o cliente web' que:
  - Sirve de interfaz con el usuario: atiende sus peticiones, muestra los resultados de las consultas y proporciona al usuario un conjunto de herramientas que facilitan su comunicación con el servidor.
  - Se comunica con el servidor web transmitiendo las peticiones de los usuarios.

Servidor

- En el equipo servidor su tarea es:
  - Atender las peticiones recibidas desde los navegadores o clientes web y hacerlo de forma eficiente y segura. Este es el caso de los servidores web seguros que solicitan un nombre de usuario y una contraseña para permitir el acceso sólo a usuarios registrados y por tanto, con permiso para visualizar la página/s.

Servidor HTTP: Apache2

El servidor HTTP Apache2 es un servidor web de software libre desarrollado por la Apache Software Foundation (ASF).

- El producto obtenido de este proyecto es un servidor de código fuente completo, descargable y gratuito.
- La página web del proyecto es www.apache.org.
- Apache2 es robusto y con un ciclo de desarrollo muy rápido gracias a la gran cantidad de colaboradores voluntarios de que dispone.
- Es también un servidor estable, eficiente, extensible y multiplataforma.

Apache proporciona contenidos al cliente web o navegador como:

- **Páginas estáticas**: es el modo más básico y antiguo, pero también es el uso más generalizado que se hace de un servidor web. De esta forma se transfieren archivos HTML, imágenes, etc y no se requiere un servidor muy potente en lo que al hardware se refiere.

- **Páginas dinámicas**: la información que muestran las páginas que sirve Apache cambia continuamente ya que se obtiene a partir de consultas a bases de datos u otras fuentes de datos. Son páginas con contenido dinámico, cambiante.

Protocolo HTTP

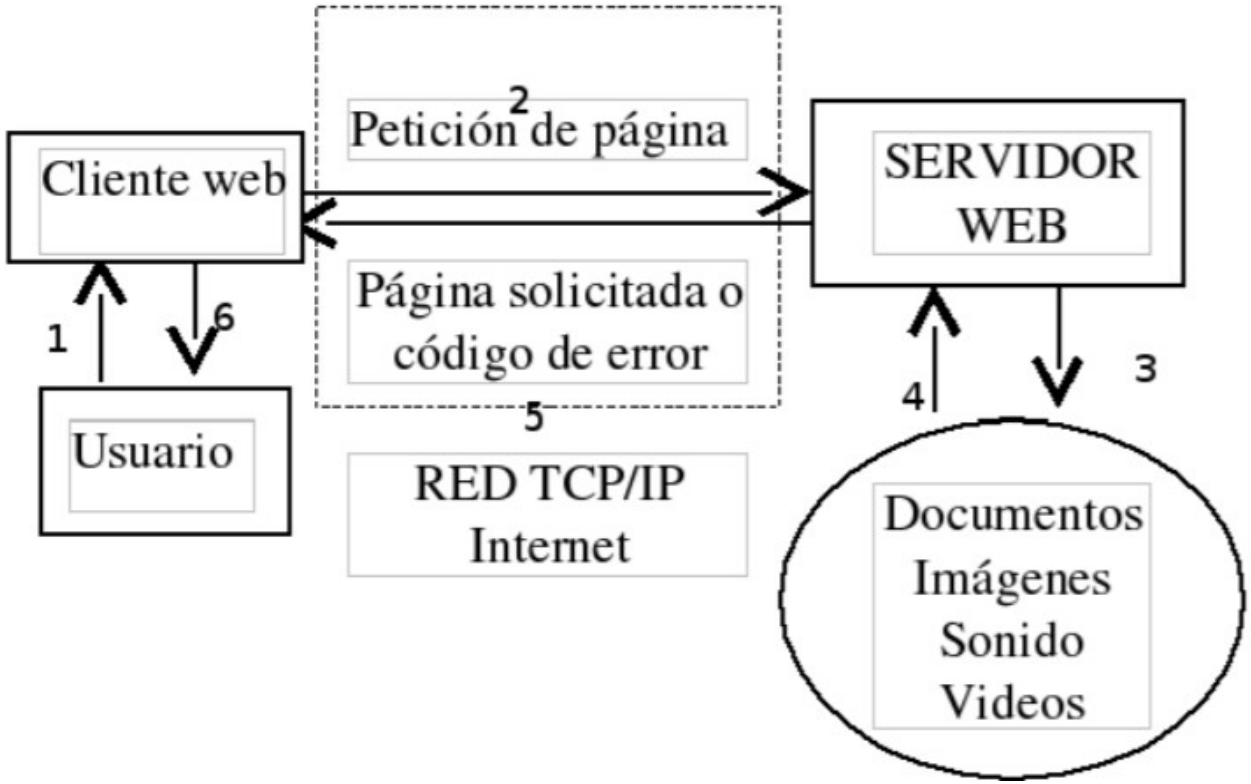
Las siglas HTTP significan Hyper Text Transfer Protocol, Protocolo de Transferencia de HiperTexto. HTTP es el protocolo usado en las transacciones de la web (www).

El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el conjunto de normas mediante las que se envían las peticiones de acceso a una web y la respuesta de esa web.

HTTP se define como un conjunto de normas que permiten la comunicación entre el servidor y los clientes y la transferencia de información (archivos de texto, imágenes, archivos de música, ...) entre ambos. HTTP es un protocolo cliente/servidor cuyo funcionamiento está basado en el envío de mensajes y consiste en que:

- El **usuario especifica** en el cliente web (navegador) **la dirección de la página que quiere consultar** según un
- formato (Ejemplo: http://dirección[:puerto][path]).
- El cliente web decodifica la información de la URL diferenciando el protocolo de acceso, IP o nombre de dominio
- del servidor, puerto,...
- **El cliente web establece una conexión (socket) con el servidor web y solicita la página** (mensaje **request**).
- **El servidor envía dicha página (si no existe envía un código de error) y el cliente web interpreta el código HTML**
- recibido. Mensaje **response** del servidor.
- Se cierra la conexión.

Protocolo HTTP



Protocolo HTTP: conocimientos y algo de historia

HTTP fue inventado por Tim Berners-Lee entre los años 1989-1991, HTTP ha visto muchos cambios, manteniendo la mayor parte de su simplicidad y desarrollando su flexibilidad.

El protocolo HTTP es un protocolo sin estado, es decir, no recuerda nada relativo a conexiones anteriores a la actual. Es decir:

- Si el servidor encuentra el documento HTML solicitado por el cliente web, lo envía.
- Si no existe se envía un código de error.
- En ambos casos, y por ser un protocolo sin estado, al final se libera la conexión.
- Por lo tanto, la conexión sólo tiene la duración correspondiente a la transmisión de la página solicitada.
- Para cada objeto que se transfiere por la red se realiza una conexión independiente.
    - Ejemplo: si el cliente web solicita una página que incorpora varias imágenes se realizan las siguientes conexiones: una para el documento HTML y una por cada una de las imágenes.

Las primeras versiones de HTTP no tenían numeración (la primera fue HTTP/0.9) y eran ciertamente muy limitadas y tanto los navegadores como los servidores, pronto ampliaron el protocolo para que fuera más flexible. ***Aparece la versión HTTP/1.0*** *desarrolladas tras varios años de innovación entre 1991 y 1995. En Noviembre de 1996, para poner fin a estos problemas se publicó un documento informativo que describía las prácticas adecuadas, RFC 1945. Esté documento es la definición del protocolo HTTP/1.0 aunque resulta curioso, que realmente no es un estándar oficial.*

Protocolo HTTP: conocimientos y algo de historia

En la versión HTTP/1.0 el cliente web podía solicitar del servidor sólo tres operaciones:

Métodos:

- **GET** obtener información del servidor, como puede ser pedir una página al hacer un clic sobre un enlace. Es la operación que se ejecuta cada vez que se pulsa sobre un enlace y se accede a una página web.
- **POST** enviar información desde el cliente web al servidor, como, por ejemplo los datos introducidos en un formulario web.
- **HEAD** similar a GET pero sólo se pide la cabecera de la página con información como el tamaño, tipo, fecha de modificación, etc.

Por ejemplo, para un mensaje de petición al servidor la primera línea tiene la estructura: GET /index.html HTTP/1.0 que debe interpretarse como una petición (GET) del archivo index.html que está en el directorio raíz compatible con la versión 1.0 del protocolo HTTP.

Por ejemplo, para un mensaje de respuesta del servidor la primera línea tiene la estructura: HTTP/1.0 200 OK que debe interpretarse como la respuesta de petición servida (código 200) con éxito e indica la versión de HTTP utilizada (1.0). El texto OK está relacionado con el código de error correspondiente. La lista de códigos está disponible en la Unidad 2.

Protocolo HTTP: conocimientos y algo de historia

La versión del protocolo HTTP/1.1 fue la primera versión estandarizada de HTTP, se publicó en 1997, tan solo unos meses después del HTTP/1.0. Según sus creadores el protocolo HTTP/1.1 se puede describir como:

"un protocolo de nivel de aplicación orientado a sistemas distribuidos, para la colaboración e hypermedia. Un protocolo genérico, sin estado, orientado a objetos y que puede ser utilizado para muchas aplicaciones, como servidores de nombres y sistemas de gestión de objetos distribuidos, a través de las extensiones de los métodos de petición. Una característica de este protocolo es la negociación de los tipos y representación de los datos, permitiendo que los sistemas no dependan del tipo de datos que se utilicen"

Algunas de las principales características que añadió son:

- **Conexiones persistentes:** Una conexión podía ser reutilizada, ahorrando así el tiempo de re-abrirla repetidas veces para mostrar los recursos dentro del documento original pedido
- **Pipeline:** permitiendo realizar una segunda petición de datos, antes de que fuera respondida la primera, disminuyendo de este modo la latencia de la comunicación.
- **Varias peticiones simultáneas**: el cliente web puede hacer varias peticiones a través de una única conexión, sin tener que esperar a que el servidor responda a cada una de ellas.
- **Negociación del contenido**: consiste en seleccionar la representación HTTP adecuada cuando se da respuesta a una petición, como por ejemplo, adaptarse a las preferencias del navegador utilizado.

Protocolo HTTP: conocimientos y algo de historia

Aparecen nuevos métodos a los ya existentes de GET, POST y HEAD:

- **TRACE** ver lo que está recibiendo el servidor
- **DELETE** borrar un recurso del servidor
- **PUT** almacenar recursos en el servidor
- **PATCH** aplicar correcciones en un recurso asociado a una URL
- **COPY** copiar recursos identificados por una URL en otro lugar
- **MOVE** mover el recurso identificado por la URL a otro lugar
- **LINK** establecer enlaces entre diferentes recursos
- **UNLINK** quitar enlaces establecidos previamente por LINK
- **OPTIONS** el cliente puede obtener las características del servidor
- **WRAPPED** puede unir varias peticiones y puede protegerlas con un filtrado como pueda ser la encriptación.

HTTP/1.1 fue publicado inicialmente como RFC 2068 en Enero de 1997 *y fue mejorado en dos revisiones: la primera, el documento RFC 2616, publicado en Junio de 1999 y posteriormente en los documentos RFC 7230-RFC 7235 publicados en Junio del 2014, en previsión de la publicación de HTTP/2. Así pues, el protocolo HTTP/1.1 ha sido increíblemente estable durante más de 15 años..*

Protocolo HTTP: conocimientos y algo de historia

La versión del protocolo HTTP/2:

El protocolo HTTP/2, tiene notables diferencias fundamentales respecto a la versión anterior HTTP/1.1

- Es un protocolo binario, en contraposición a estar formado por cadenas de texto, tal y como estaban basados sus protocolos anteriores. Así pues no se puede leer directamente, ni crear manualmente A pesar de este inconveniente, gracias a este cambio es posible utilizar en él técnicas de optimización.

- Es un protocolo multiplexado. Peticiones paralelas pueden hacerse sobre la misma conexión, no está sujeto pues a mantener el orden de los mensajes, ni otras restricciones que tenían los protocolos anteriores HTTP/1.x

- Comprime las cabeceras, ya que estas, normalmente son similares en un grupo de peticiones. Esto elimina la duplicación y retardo en los datos a transmitir.

- Esto permite al servidor almacenar datos en la caché del cliente, previamente a que estos sean pedidos, mediante un mecanismo denominado 'server push'.

HTTP/2.0 fue estandarizado de manera oficial en Mayo de 2015.

Protocolo HTTP: conocimientos y algo de historia

Seguridad en la WEB
Un servidor web que ejecuta el protocolo HTTP no proporciona seguridad de transporte de los datos que envía o recibe. Cuando se conecta a un servidor HTTP utilizando un navegador web, las URL que visita, el contenido de las páginas web que recibe y el contenido (incluidas las contraseñas) de cualquier formulario HTML que envía son visibles a cualquier acceso no autorizado en la ruta de la red.

En este sentido y relacionado con la seguridad, el mayor cambio en el desarrollo de HTTP, fue a finales de 1994. En vez de trasmitir HTTP sobre la capa de TCP/IP, **se creo una capa adicional sobre ésta: SSL**. La versión SSL 1.0 nunca fue publicada fuera de las compañías desarrolladoras, pero el SSL 2.0 y sus sucesoras SSL 3.0 y SSL 3.1 permitieron la creación del comercio electrónico en la Web (e-commerce), encriptando y garantizando la autenticidad de los mensajes intercambiados entre servidor y cliente.

SSL se añadió a la lista de estándares y posteriormente evolucionó hasta ser el protocolo TLS, con versiones 1.0, 1.1 y 1.2, que fueron apareciendo para resolver vulnerabilidades. Actualmente se está desarrollando el protocolo TLS 1.3.

La práctica recomendada para proteger el servidor web es instalar soporte para HTTPS (HTTP seguro), que protege los datos con cifrado SSL/TLS.

U-tad

Calle Playa de Liencres, 2 bis
(entrada por calle Rozabella)
Parque Europa Empresarial
Edificio Madrid
28290 Las Rozas, Madrid

900 373 379 info@u-tad.com

SOLICITA MÁS INFORMACIÓN

CENTRO ADSCRITO A:

Universidad Camilo José Cela

PROYECTO COFINANCIADO POR:

GOBIERNO DE ESPAÑA · MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE