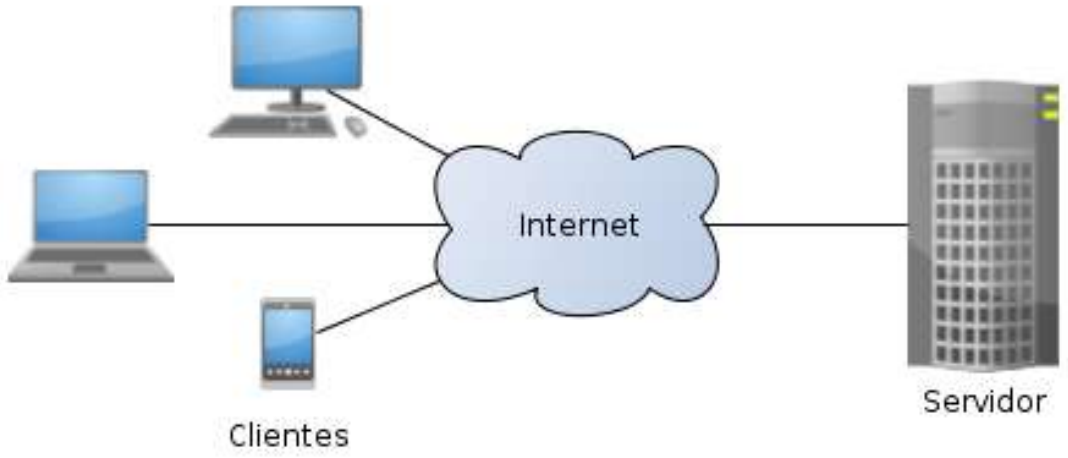
Unidad 1

Introducción a las arquitecturas de entorno servidor

Modelos de ejecución de código en servidores y clientes web

El desarrollo de aplicaciones web se apoya en la arquitectura cliente-servidor. En esta arquitectura, existen dos actores, cliente y servidor.



Cliente Servidor

- El cliente se conecta al servidor para solicitar algun servicio. **Petición* .
- El servidor se encuentra en ejecución de forma ininterrumpida a la espera de que los diferentes clientes realicen una solicitud. **Respuesta* .
- Se suele tratar de obtener el contenido de una página web. También podemos hablar de servicios web donde no se generan páginas web sino contenido en XML o JSON para ser consumido por una aplicación cliente.
- La solicitud que hacen los clientes al servidor se le llama petición (**request**)
- Lo que el servidor devuelve a dicho cliente le llamamos respuesta (**response**).
- Estos términos son los usados por el protocolo HTTP y los encontrarés en cualquier lenguaje DWS.
- También hay que tener en cuenta que esta arquitectura cliente-servidor plantea la posibilidad de numerosos clientes atendidos por un mismo servidor. El servidor será un software multitarea capaz de atender peticiones simultáneas de numerosos clientes.
- Cuando una aplicación o servicio web tiene muchas solicitudes por unidad de tiempo puede ser que un conjunto de servidores o *cluster* desempeñen este servicio en equipo.

Páginas estáticas y dinámicas

- Antes de empezar a hablar de distintas tecnologías para la programación en el entorno servidor, debemos comentar algunas cuestiones:
- Páginas web
- Dinamismo del lado cliente y del lado servidor

Páginas estáticas.

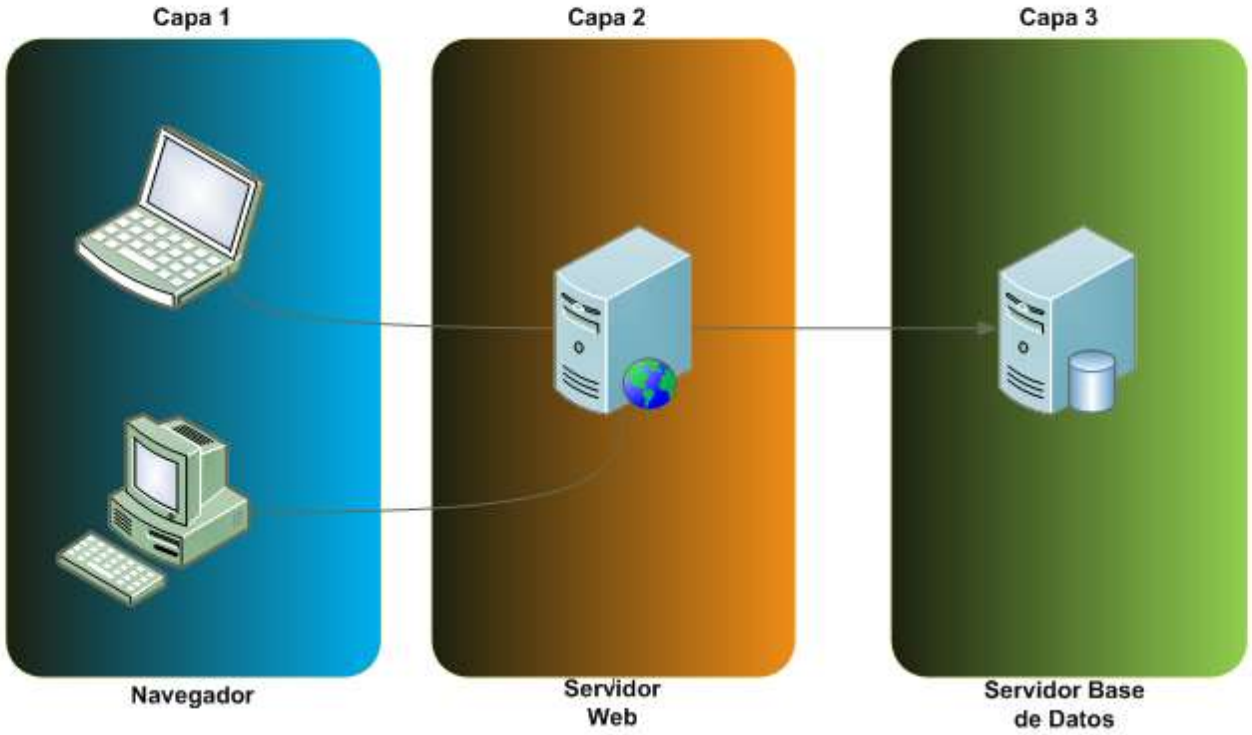
- Decimos que una página es estática cuando está construida enteramente con código HTML y CSS, código escrito directamente en ficheros y que con cambia.
- Para visualizar una página estática sólo es preciso un navegador web. No hace falta ningún otro software.

Páginas dinámicas.

- Decimos que una página es dinámica cuando el contenido y aspecto es cambiante.
- Estos cambios pueden ser fruto de que el servidor modifica el HTML entre peticionees. En este caso hablamos de páginas dinámicas del **lado del servidor**.
- También pueden ocurrir si ejecutamos código en el navegador, javascript. En este caso podemos hablar de páginas dinámicas del **lado del cliente**.
- La tecnología **AJAX**(*Asynchronous JavaScript And XML*) realiza una combinación de ambos dinamisms.
- Permite actualizar el contenido de una página sin recargarla completamente.
- Javascript solicita datos al servidor
- Los datos recibidos son usados para renovar el contenido de la página web.

Desarrollo en capas

- Podemos separar funcionalidades.
- Seguimos trabajando con filosofía cliente-servidor.
- Separamos el servidor en dos máquinas.



- Capa de presentación: Es la capa donde la aplicación se muestra al usuario. Básicamente es la GUI (Graphical User Interface, Interfaz Gráfica de Usuario). En el caso de una aplicación web sería el código HTML que se carga directamente en el navegador web. En cualquier caso, se ejecuta directamente en el equipo del cliente.
- Capa de negocio: Es la capa intermedia donde se lleva a cabo toda la lógica de la aplicación. Siempre se ejecutará en el lado servidor. Esta capa, tras realizar todos los cálculos y/o operaciones sobre los datos, genera el código HTML que será presentado al usuario en la capa siguiente.
- Capa de datos: Es la capa que almacena los datos. Básicamente, en condiciones normales, hace referencia al propio SGBD que es el encargado de almacenar los datos. Dependiendo de la arquitectura de la aplicación, esta capa y la de negocio se pueden encontrar fisicamente en el mismo equipo, aunque también es posible que se tengan que separar por cuestiones de rendimiento. La capa de datos sirve todas la información necesaria a la capa de negocio para llevar a cabo sus operaciones.

Una aplicación web nos encontramos con:

- Navegador web: Mozilla Firefox, Internet Explorer o Google Chrome...
- Servidor: acompañado de un lenguaje de programación web permite desarrollar la parte que ocupa la capa de negocio. Un ejemplo habitual es usar Apache + PHP.
- Base de datos: Cualquier SGBD relacional como MySQL o PostgreSQL o sistemas no relacionales como MongoDB.

Tecnologías de desarrollo servidor

Una vez aclarado esto nos debemos plantear qué tecnología de servidor podemos utilizar:

1. Java EE: Servidor de aplicaciones con JSP y Servlest JAVA.
2. **PHP en conjunción con un servidor web. Tipicamente Apache.**
3. **Node.js. En este caso el servidor es un módulo más de la propia aplicación.**
4. Otras: .Net, CGI, Rubi, Python ...

HTTP

Veamos las cuestiones más notables

- HTTP es el protocolo usado para la transferencia de recursos en la web.
- El cliente o navegador se denomina "User Agent"
- Se basa en transacciones según el esquema petición-respuesta.
- Los recursos se identifican por un localizador único: URL
- Es un protocolo sin **estado**, es decir no guarda información entre conexiones. Esto lo hace flexible y escalable pero es un problema para crear aplicaciones web.
- Http requiere de un extra para *recordarnos* cuando abrimos sesión en una aplicaión. Lo veremos.

Las URL

- El esquema de una URL es: `protocolo://maquina:puerto/camino/fichero`
- Protocolo es **http** o **https**
- Máquina: una IP o un nombre (DNS).
- El puerto suele omitirse por usarse los de defecto: 80 y 443 para http y https respectivamente.
- Camino o ruta relativa al directorio raíz del sitio
- Fichero es el nombre del recurso.

Peticiones HTTP

Las peticiones HTTP siguen la siguiente estructura:

```
GET /index.html HTTP/1.1
Host: www.sitioweb.com:8080
User-Agent: Mozilla/5.0 (Windows;en-GB; rv:1.8.0.11) Gecko/20070312
[Línea en blanco]
```

- Las cabeceras aportan gran información. En el ejemplo vemos el host con el puerto y la identificación del navegador cliente.

- Otro ejemplo con el método POST.

```
POST /en/html/dummy HTTP/1.1
Host: www.explainth.at
User-Agent: Mozilla/5.0 (Windows;en-GB; rv:1.8.0.11) Gecko/20070312 Firefox/1.5.0.11
Accept: text/xml,text/html;q=0.9,text/plain;q=0.8, image/png,*/*;q=0.5
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.explainth.at/en/misc/httpreq.shtml
Content-Type: application/x-www-form-urlencoded
Content-Length: 39

name=MyName&married=not+single&male=yes
```

- La lista completa de cabeceras que se pueden usar la podemos consultar en wikipedia. Se pueden enviar informaciones como:
  - El nombre del navegador
  - El tipo de contenido solicitado y el aceptado (p. ej. página html, o un pdf, ...).
  - El juego de caracteres.
  - El idioma preferido.
  - Si se admite contenido comprimido ...
- El cuerpo del mensaje en las peticiones GET está vacío.
- La lista de métodos es bastante amplia pero de momento sólo nos preocupan los métodos GET y POST.
  - GET pide un recurso
  - POST envía datos al servidor para ser procesados. Los datos se incluyen en el cuerpo del nensaje.

Respuestas HTTP

La respuesta se ajusta al siguiente esquema:

```
Versión-http SP código-estado SP frase-explicación retorno_de_carro
(nombre-cabecera: valor-cabecera ("," valor-cabecera)* CRLF)*
Cuerpo del mensaje
```

Un ejemplo sería

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 673
<html>
<head> <title> Título de nuestra web </title> </head>
<body>
¡Hola mundo!
</body>
</html>
```

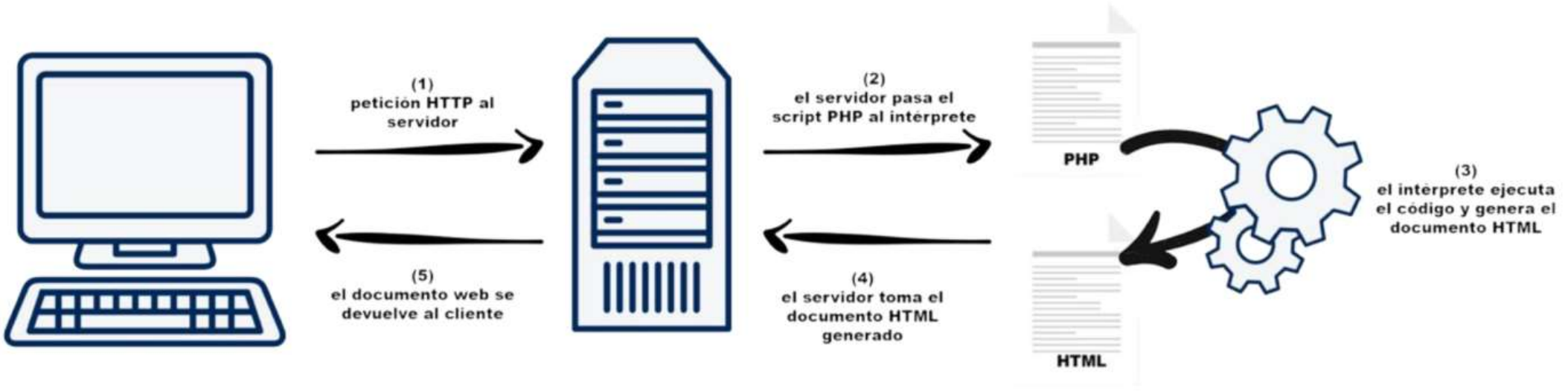
Comprensión de los tipos de código de estado HTTP

Los códigos de estado HTTP se dividen en 5 «tipos». Se trata de agrupaciones de respuestas que tienen significados similares o relacionados. Saber qué son puede ayudarte a determinar rápidamente la sustancia general de un código de estado *antes de que vayas a* buscar su significado específico.

Las cinco clases incluyen:

- **100s:** Códigos informativos que indican que la solicitud iniciada por el navegador continúa.
- **200s:** Los códigos con éxito regresaron cuando la solicitud del navegador fue recibida, entendida y procesada por el servidor.
- **300s:** Códigos de redireccionamiento devueltos cuando un nuevo recurso ha sido sustituido por el recurso solicitado.
- **400s:** Códigos de error del cliente que indican que hubo un problema con la solicitud.
- **500s:** Códigos de error del servidor que indican que la solicitud fue aceptada, pero que un error en el servidor impidió que se cumpliera.

Dentro de cada una de estas tipos, existe una variedad de códigos de servidor y pueden ser devueltos por el servidor. Cada código individual tiene un significado específico y único, que cubriremos en la lista más detallada a continuación.


(1)
petición HTTP al
servidor
(2)
el servidor pasa el
script PHP al intérprete
PHP
(3)
el intérprete ejecuta
el código y genera el
documento HTML
(4)
el servidor toma el
documento HTML
generado
(5)
el documento web se
devuelve al cliente
HTML

Capas

En cambio, las capas lógicas (*layers*) organizan el código respecto a su funcionalidad:

- Presentación
- Negocio / Aplicación / Proceso
- Datos / Persistencia

Como se observa, cada una de las capas se puede implementar con diferentes lenguajes de programación y/o herramientas.