UD1 - Introducción al desarrollo de aplicaciones web

DESARROLLO WEB EN ENTORNO SERVIDOR

Aplicación

¿Qué es una aplicación?

Aplicación

¿Qué es una aplicación?

- Simplemente un programa informático.
- Realiza alguna tarea.
- ¿Ejemplos?

Aplicación

¿Qué es una aplicación?

- Simplemente un programa informático.
- Realiza alguna tarea.
- ¿Ejemplos?

Aplicación web

¿Qué es una aplicación web?

Aplicación web

¿Qué es una aplicación web?

- Simplemente un programa informático.
- Realiza alguna tarea.
- Sigue una arquitectura cliente-servidor.
- ¿Ejemplos?

Aplicación web

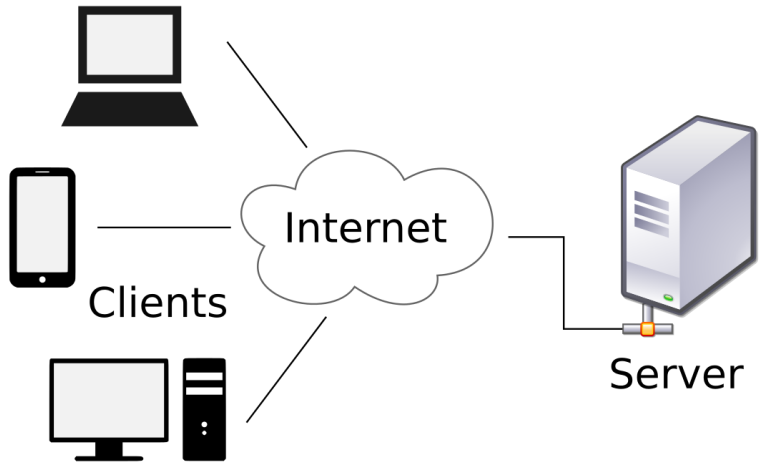
¿Qué es una aplicación web?

- Simplemente un programa informático.
- Realiza alguna tarea.
- Sigue una arquitectura cliente-servidor.
- ¿Ejemplos?

Arquitectura cliente-servidor

Imagina que tienes un programa:

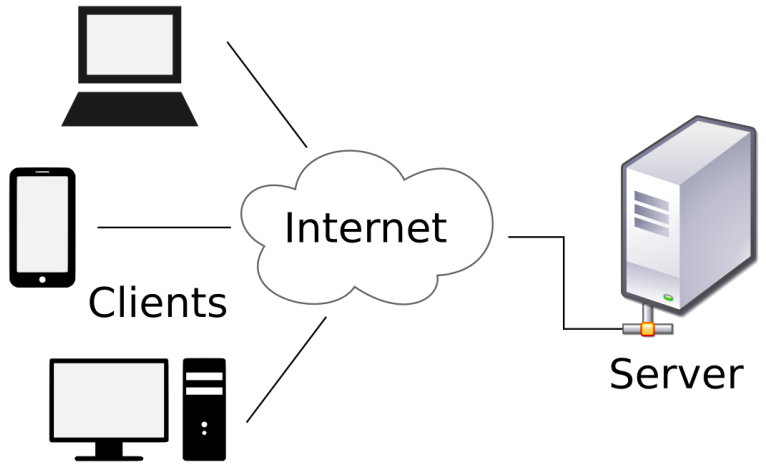
- Lo partimos en 2.



Arquitectura cliente-servidor

Imagina que tienes un programa:

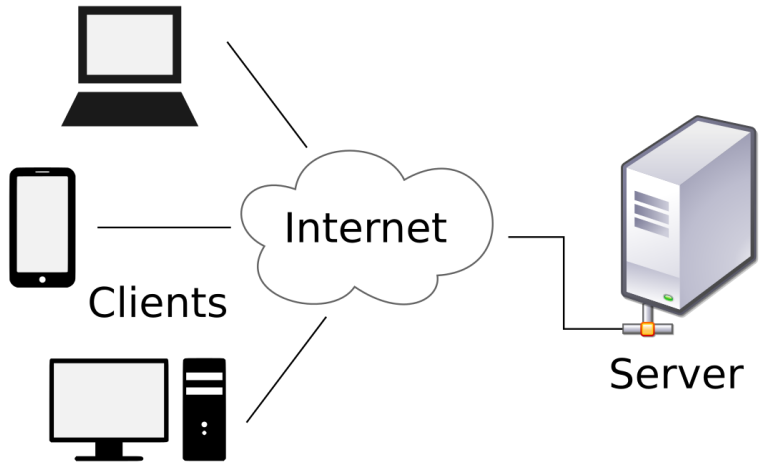
- Lo partimos en 2.
- Nos llevamos una parte al “cliente”.
- Nos llevamos la otra parte al “servidor”.
- Ambos se comunican entre sí por la red.



Arquitectura cliente-servidor

Imagina que tienes un programa:

- Lo partimos en 2.
- Nos llevamos una parte al “cliente”.
- Nos llevamos la otra parte al “servidor”.
- Ambos se comunican entre sí por la red.
- El cliente hace peticiones al servidor.
- El servidor duerme todo el tiempo… salvo para atender las peticiones.



Arquitectura cliente-servidor en una aplicación web

Imagina que tienes un programa (el Office, por ejemplo):

- Lo partimos en 2.
- Nos llevamos una parte al “cliente”. → ¿cómo se llama?
- Nos llevamos la otra parte al “servidor”. → ¿cómo se llama?
- Ambos se comunican entre sí por la red. → ¿con qué protocolo?
- El cliente hace peticiones al servidor. → ¿qué le pide?
- El servidor duerme todo el tiempo… salvo para atender las peticiones. → ¿qué le responde?

Arquitectura cliente-servidor en una aplicación web

Imagina que tienes un programa (el Office, por ejemplo):

- Lo partimos en 2.
- Nos llevamos una parte al "cliente". → ¿cómo se llama? **Navegador**
- Nos llevamos la otra parte al "servidor". → ¿cómo se llama? **Servidor web**
- Ambos se comunican entre sí por la red. → ¿con qué protocolo? **HTTP/S**
- El cliente hace peticiones al servidor. → ¿qué le pide? **La página que quiere**
- El servidor duerme todo el tiempo… salvo para atender las peticiones. → ¿qué le responde? **HTML, CSS y JavaScript (el pobre navegador no entiende otro idioma)**

Arquitectura cliente-servidor en una aplicación web

No te líes durante el curso (y resto de tu vida):

- El cliente y el servidor son programas normales y corrientes, no máquinas.
- Ambos pueden estar ejecutándose dentro de la misma máquina.
- …Pero lo habitual es que estén a cientos de kilómetros entre sí.

Arquitectura cliente-servidor en una aplicación web

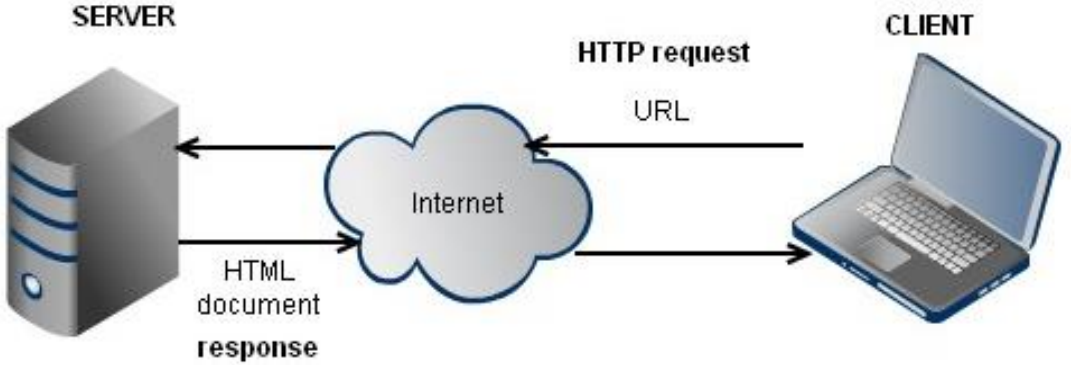
No te líes durante el curso (y resto de tu vida):

- El cliente y el servidor son programas normales y corrientes, no máquinas.
- Ambos pueden estar ejecutándose dentro de la misma máquina.
- …Pero lo habitual es que estén a cientos de kilómetros entre sí.

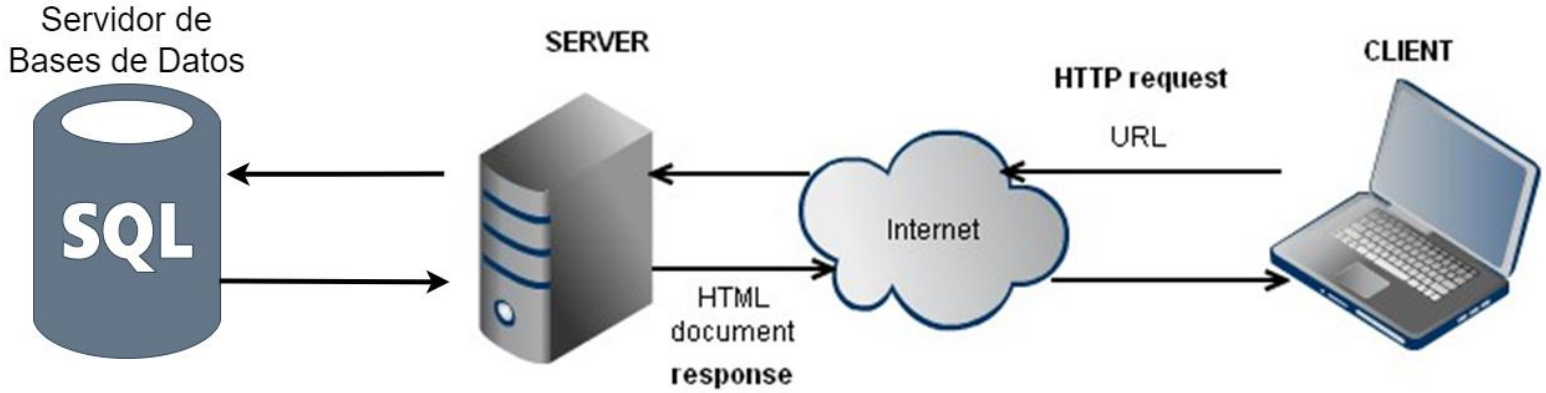
- **OJO**: hay ordenadores que están diseñados específicamente para albergar programas servidores… ¡y a estos también se les llama servidores!

Arquitectura cliente-servidor en una aplicación web: capas

Lo que acabamos de explicar sería un modelo de 2 capas.

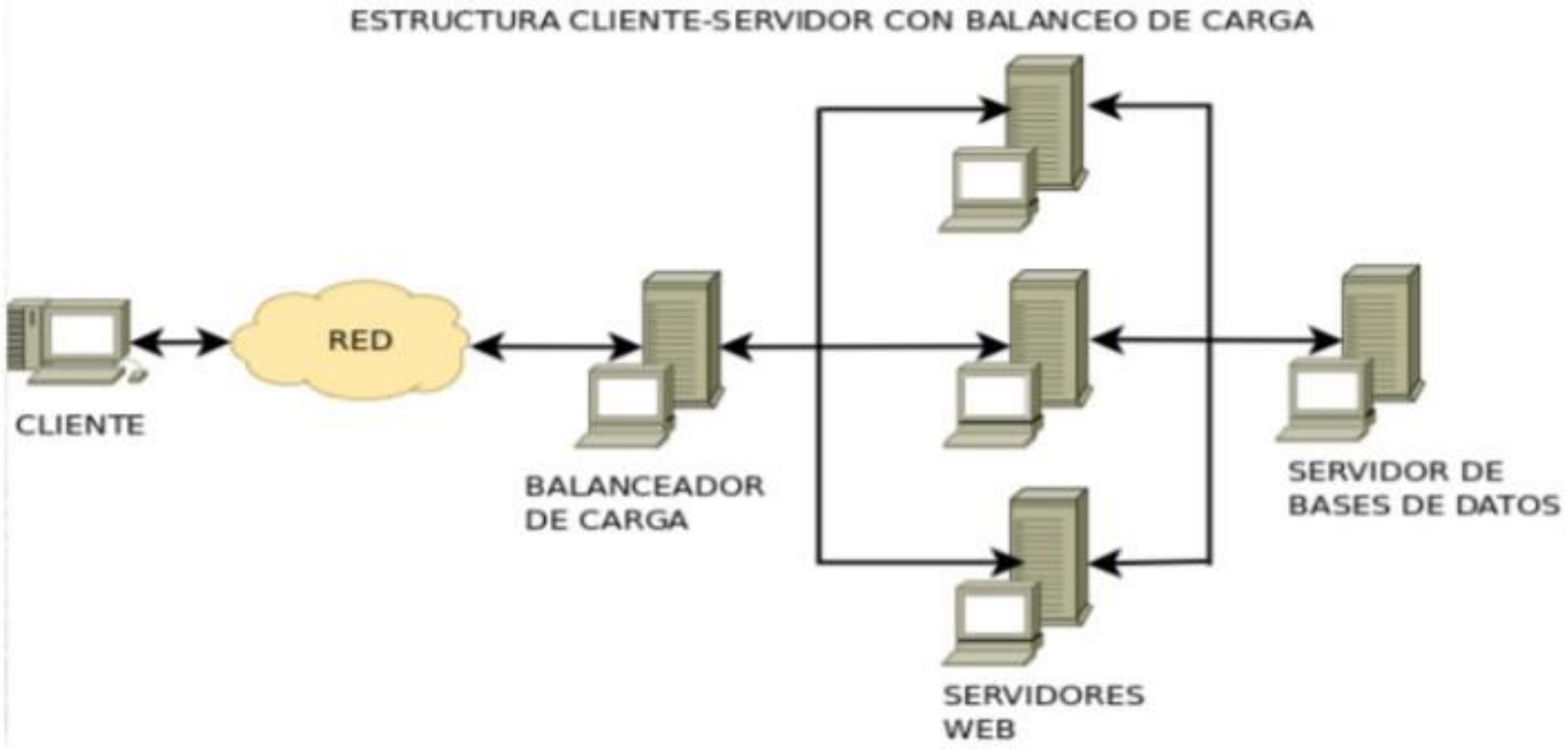


Si usamos un servidor de bases de datos, tendríamos un modelo de 3 capas:



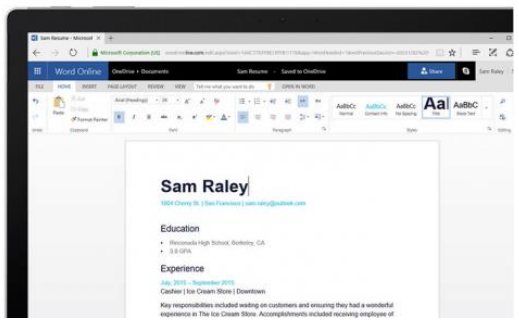
Arquitectura cliente-servidor en una aplicación web: capas

Si usamos más de un servidor web para repartir peticiones, un modelo de balanceo de carga:



Aplicación de escritorio VS Aplicación web

Ventajas y desventajas



Aplicación de escritorio VS Aplicación web

Ventajas

- Más robustas.
- Tiempo de respuesta más rápido.
- Se aprovecha al máximo la capacidad de un dispositivo (tarjeta gráfica, control total de las entradas del usuario, etc.)
- ~~Combinaciones de teclas (ejemplo: CTRL+G para grabar)~~

Desventajas

- Instalación en cada equipo.
- Dependencia del sistema operativo.
- Hay que actualizar cada equipo con nuevas versiones.
- Siempre conectados a internet.

Ventajas

- Desde cualquier lugar.
- Siempre se tiene última versión.
- No hay problemas de incompatibilidad entre versiones, todos con la misma.
- No se instala nada (solo el navegador).
- Cualquier sistema operativo.
- Las copias de seguridad se centralizan.

Desventajas

- Siempre conectados a internet.
- La respuesta puede ser lenta.

Generación de webs

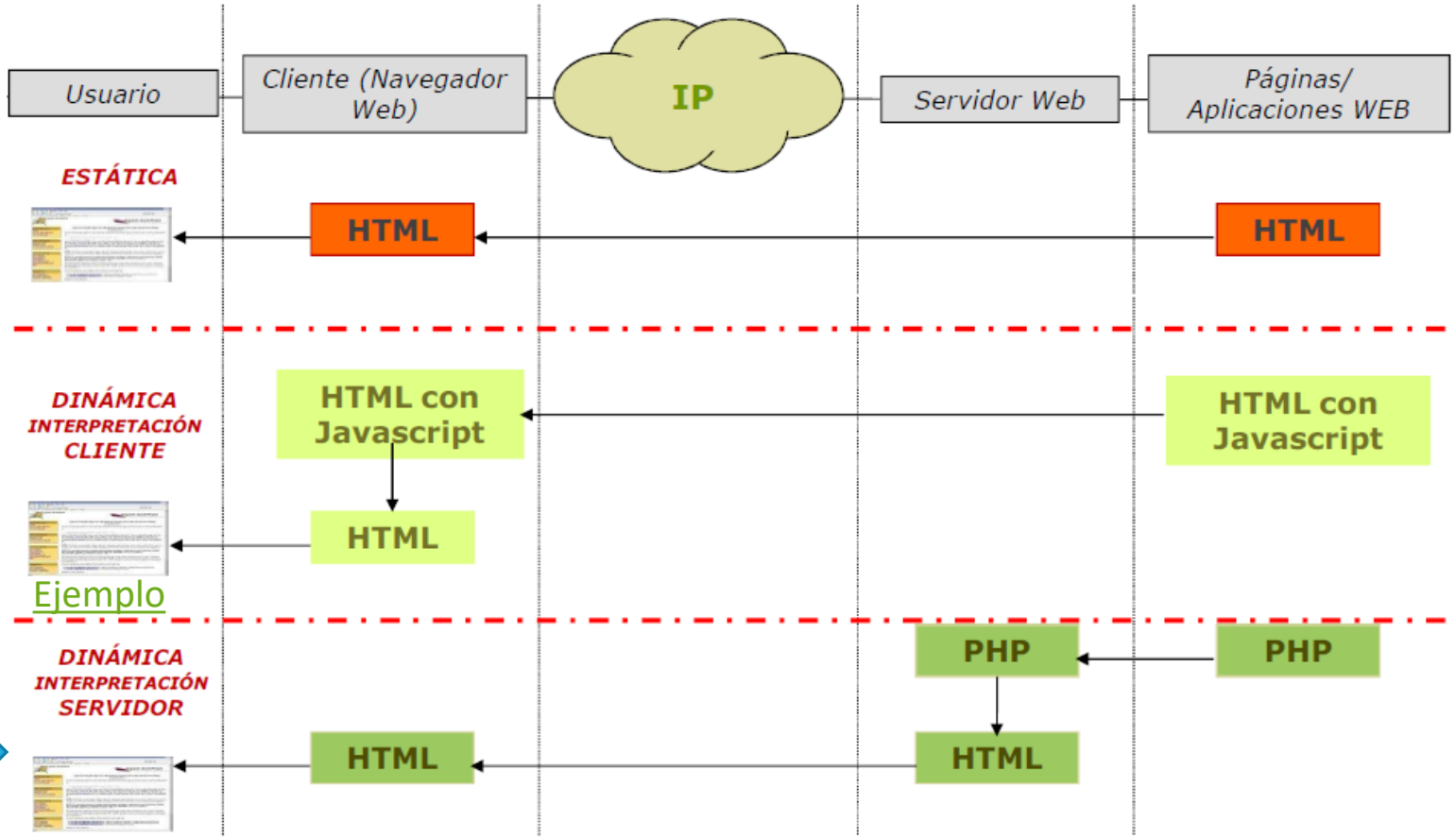
Las reglas del juego son:

- El navegador pide una página.
- El navegador solo entiende HTML, CSS y JavaScript.
- Por lo tanto → el servidor web solo puede responder eso.

Generación de webs

Las reglas del juego son:

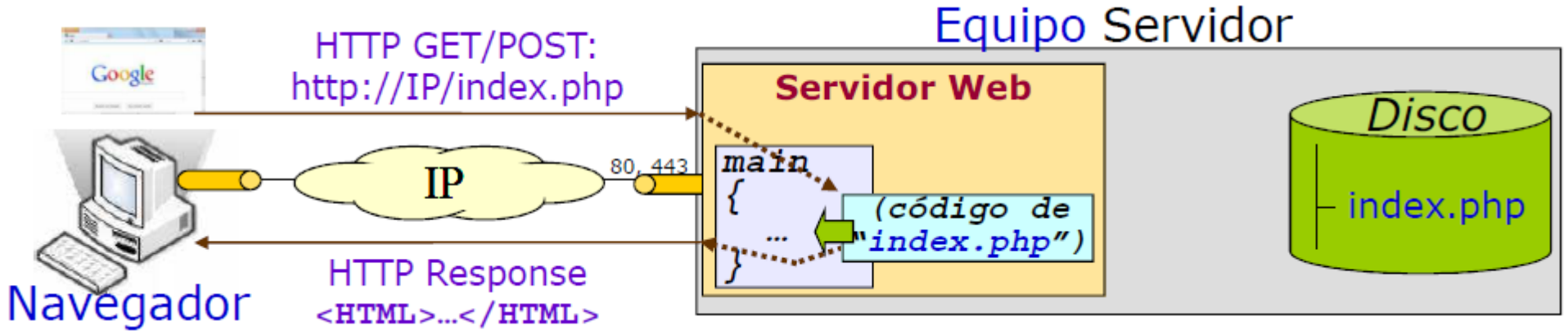
- El navegador pide una página.
- El navegador solo entiende HTML, CSS y JavaScript.
- Por lo tanto → el servidor web solo puede responder eso.



Estamos aquí

¿Y cómo se generan las páginas web?

Nuestro servidor web cuenta con un módulo, generalmente un intérprete, para ejecutar el código:



No es la única manera, el servidor podría pasar la pelota a un segundo programa (CGI).

Lenguajes de cliente

Ya lo hemos dicho:

- HTML: ¿qué hace?
- CSS: ¿qué hace?
- JavaScript: ¿qué hace?

Lenguajes de cliente

Ya lo hemos dicho:

- HTML: describe el contenido.
- CSS: da estilo al contenido.
- JavaScript: hace interactiva la web.

Lenguajes de servidor

Hay muchos lenguajes de entorno servidor, pero estos son los más comunes.

PHP: Es el más extendido en el lado servidor. Se ejecuta como un módulo en el servidor.

ASP.NET: Plataforma de desarrollo web de Microsoft

Java: tú mejor amigo (hasta ahora).

JavaScript: inicialmente usado en el cliente, existen servidores que lo usan.

Lenguajes de servidor

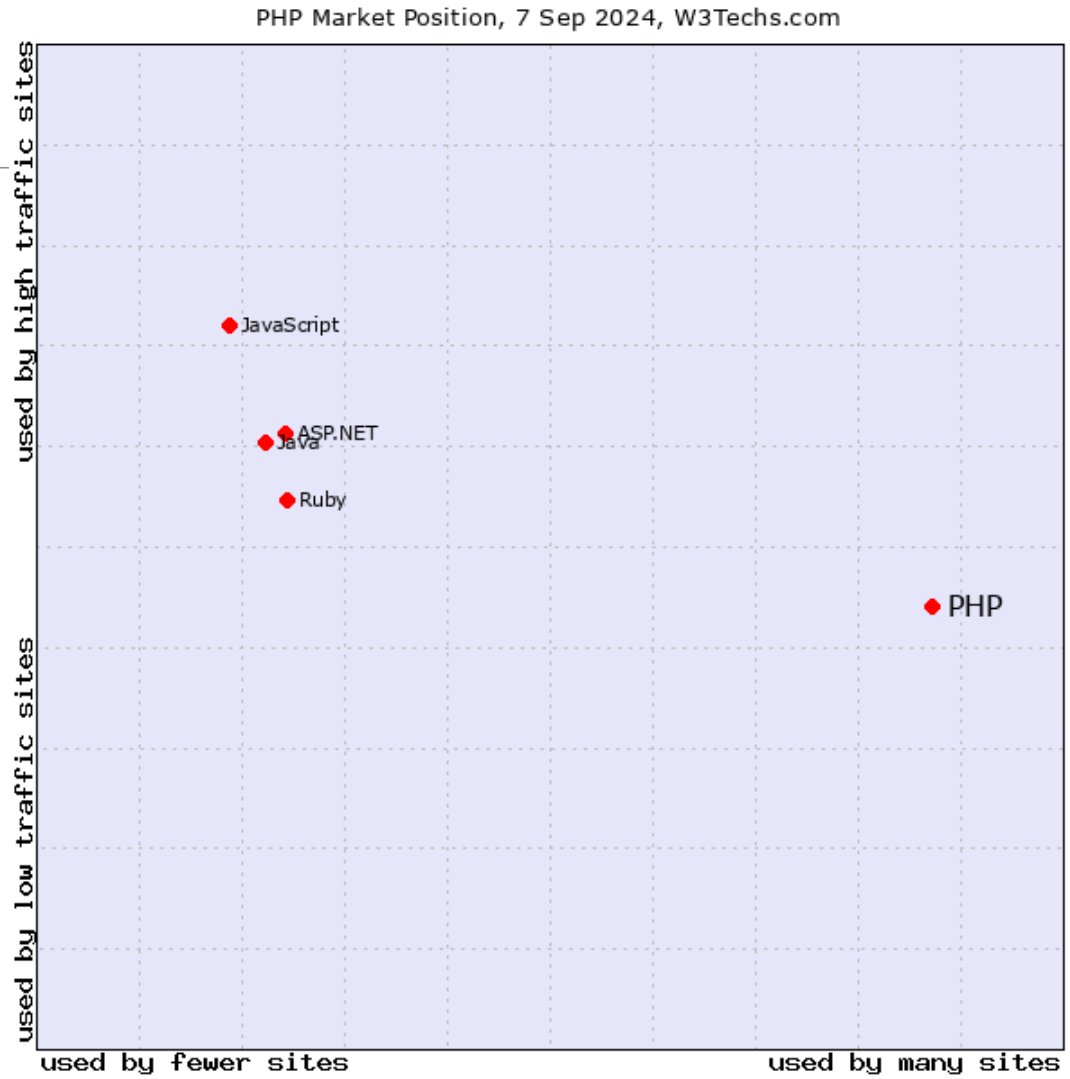
Hay muchos lenguajes de entorno servidor, pero estos son los más comunes.

PHP: Es el más extendido en el lado servidor. Se ejecuta como un módulo en el servidor.

ASP.NET: Plataforma de desarrollo web de Microsoft

Java: tú mejor amigo (hasta ahora).

JavaScript: inicialmente usado en el cliente, existen servidores que lo usan.



Lenguajes de servidor: frameworks

Framework	Lenguaje en el que se basan
Spring	Java Enterprise Edition
Ruby on Rails	Ruby
Django	Python
AngularJS	Javascript
Symfony	PHP

Lenguajes de servidor: frameworks

Los **frameworks** son entornos de trabajo que facilitan el desarrollo de software al programador.

Ofrecen elementos como librerías, paquetes, plantillas, etc. que son más rápidos de usar que el propio lenguaje.

Framework	Lenguaje en el que se basan
Spring	Java Enterprise Edition
Ruby on Rails	Ruby
Django	Python
AngularJS	Javascript
Symfony	PHP

Un poco de cultura general

¿Cómo ha evolucionado la web?

A lo largo del tiempo ha cambiado tanto el modo de usarla como las tecnologías.

1. **Web 1.0:**

¿Cómo ha evolucionado la web?

A lo largo del tiempo ha cambiado tanto el modo de usarla como las tecnologías.

1. **Web 1.0:**
   - Páginas estáticas (el contenido no cambiaba).
   - No se puede considerar que hubiera aplicaciones web.
   - El fin era divulgativo.
   - ¿Ejemplos?

¿Cómo ha evolucionado la web?

2. **Web 2.0 (o web social):**
   -
   -
   -
   -
   -
3. **Web 3.0**:
   -
   -

¿Cómo ha evolucionado la web?

2. **Web 2.0 (o web social):**
   - Término que surge en 2004.
   - El usuario pasa a ser el protagonista, crea y comparte información.
   - Aparecen aplicaciones: RR.SS., los blogs, las wikis, etc.
   - Se trata de *hacer cosas* en vez de *mirar cosas*.
   - ¿Ejemplos?
3. **Web 3.0**:
   -
   -

¿Cómo ha evolucionado la web?

2. **Web 2.0 (o web social):**
   - Término que surge en 2004.
   - El usuario pasa a ser el protagonista, crea y comparte información.
   - Aparecen aplicaciones: RR.SS., los blogs, las wikis, etc.
   - Se trata de *hacer cosas* en vez de *mirar cosas*.
   - ¿Ejemplos?
3. **Web 3.0**:
   - Concepto desarrollado en la actualidad.
   - “Cajón de sastre” donde se meten conceptos como inteligencia artificial y tecnologías relacionadas con las criptomonedas.

¿Cómo ha evolucionado la web?

OJO: los términos web 1.0, web 2.0 y web 3.0 no hacen referencia a versiones, son solo nombres de sus “periodos evolutivos”.

Las tecnologías también han cambiado: