

---

# UD1 – Programación web en el servidor

2º CFGS  
Desarrollo de Aplicaciones Web  
2023-24

# 1.- Características de la programación web

Página web  $\neq$  Aplicación web

# 1.- Características de la programación web

El **contenido de las aplicaciones** web está "programado" en **HTML**.

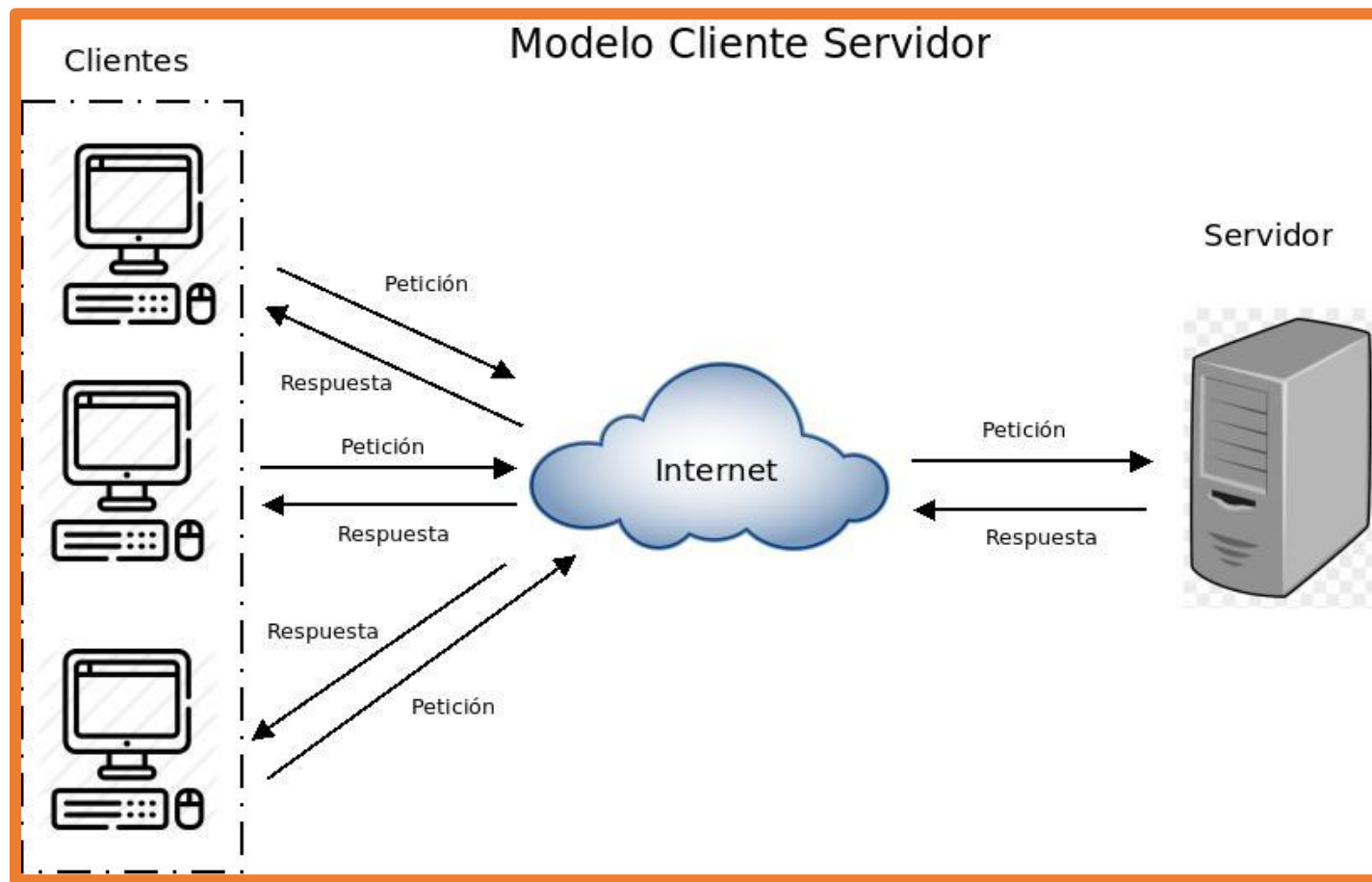
HTML se compone de **etiquetas**.

Cada **etiqueta** tiene un **significado**.

Para mostrar el **estilo** se utilizará uno o varios archivos **CSS**.

## 2.- Modelo Cliente – Servidor

La arquitectura de funcionamiento de las aplicaciones web se basa en el modelo Cliente – Servidor.



### 3.- Características de la programación web

¿Qué pasos son los que suceden cuando se introduce una URL en el navegador y se pulsa "enter" para acceder a ella o cuando se hace clic en un enlace?

1. Petición del recurso al servidor.
2. El servidor busca el recurso en el directorio indicado por la URL.
3. Si se encuentra el recurso lo envía al cliente.
4. El cliente analiza el recurso recibido.
5. Si es necesario se pedirán recursos complementarios (imágenes, CSS, JavaScript...)
6. Se muestra el recurso en la ventana del navegador.

### 3.- Características de la programación web

En una aplicación web, el "programa" termina su ejecución una vez se carga toda la página web en el cliente.

Cada interacción del usuario inicia la ejecución de un "programa" en el servidor.

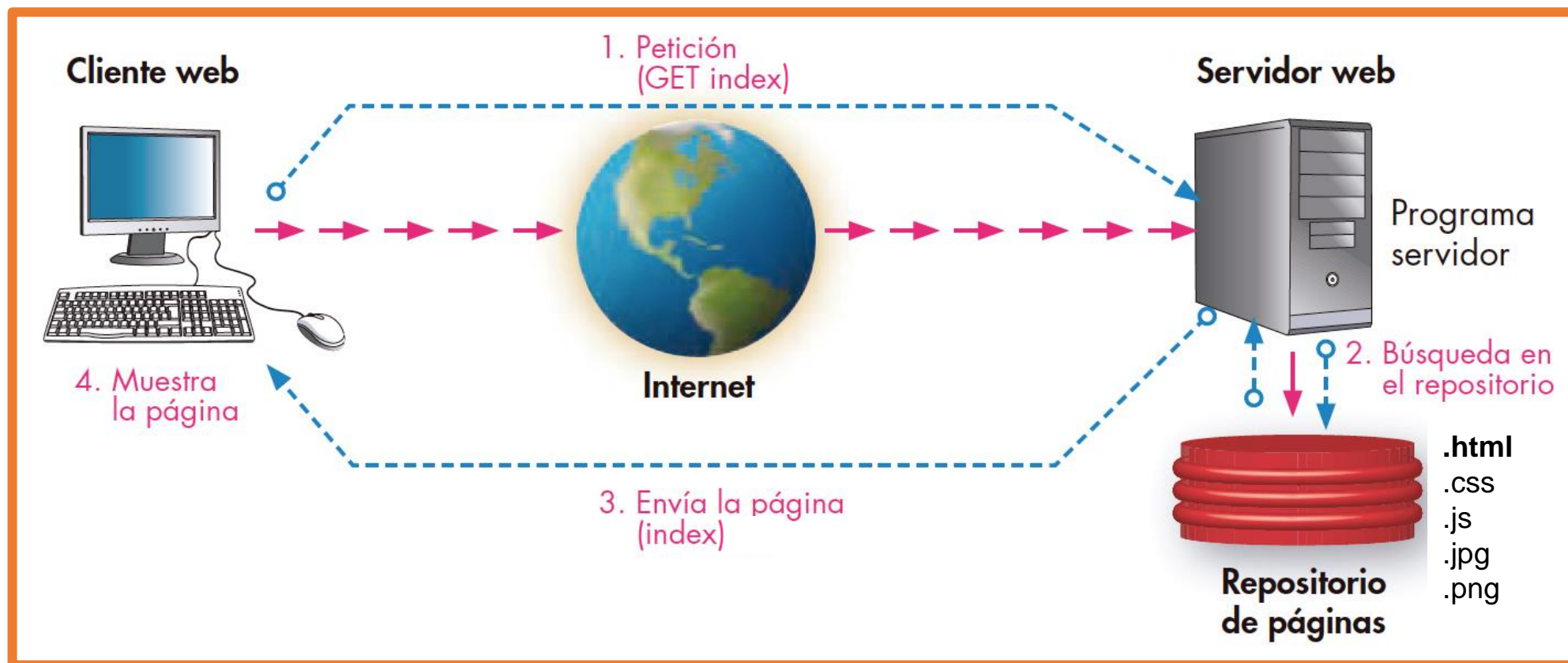
Es importante tener esto en cuenta ya que difiere al comportamiento habitual de las aplicaciones de escritorio y móviles que se mantienen en ejecución hasta que se cierran.

## 4.- Aplicaciones web estáticas

Esquema típico de funcionamiento de una **aplicación web estática**.

Petición URL: `https://iesserpis.com/noticias`

**Recurso final → repositorio/noticias/index.html**



## 4.- Aplicaciones web estáticas

Las páginas web se almacenan en su **forma definitiva**.

Solo varían si el desarrollador **altera** el contenido.

Su utilidad se basa en **mostrar información concreta**.

**Consumen menos recursos.**

La extensión de los archivos es **.html**.

**¿Son útiles hoy en día?**



# Práctica

## **Actividad 1:** El portafolio

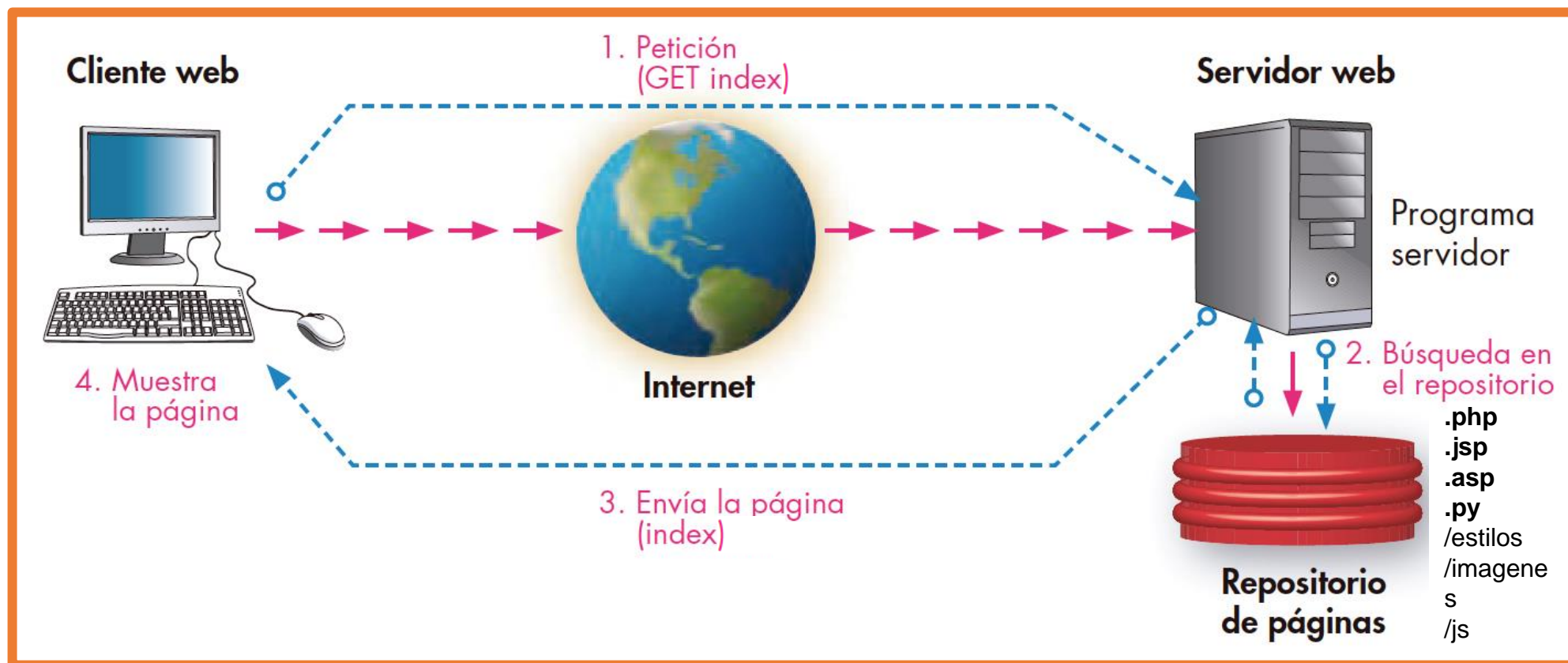
\* Recuerda separar y organizar bien las imágenes y los estilos en sus carpetas correspondientes.

## 5.- Aplicaciones web dinámicas

Esquema típico de funcionamiento de una **aplicación web dinámica**.

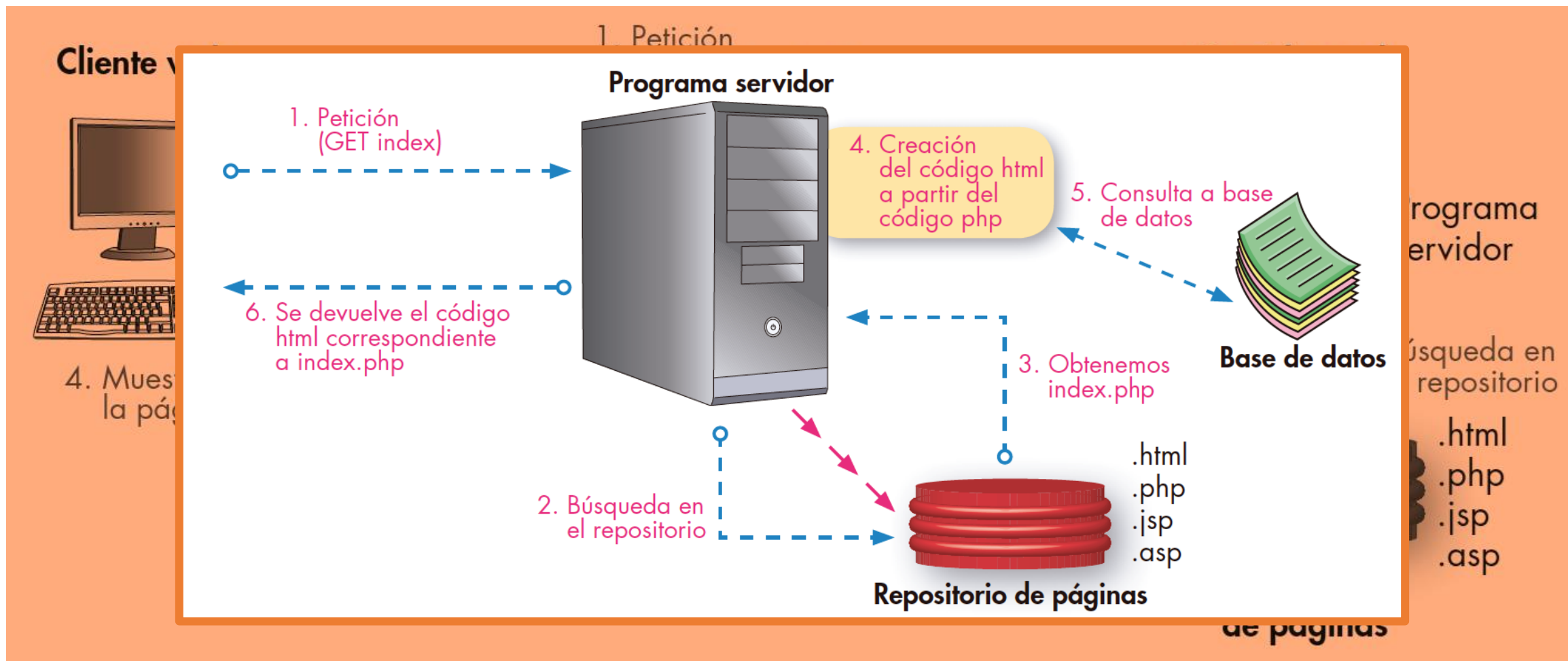
Petición URL: `https://iesserpis.com/noticias`

**Recurso final → repositorio/noticias/index.php**



## 5.- Aplicaciones web dinámicas

Esquema típico de funcionamiento de una **aplicación web dinámica**.



## 5.- Aplicaciones web dinámicas

Su **contenido** cambia dependiendo de **diferentes factores**:

- Día y hora a la que se accede.
- Si se accede con usuario registrado.
- Acciones realizadas previamente.

Como en todas las aplicaciones web el cliente recibe un archivo cuyo **contenido es html**.

En las aplicaciones web dinámicas ese contenido html no se encuentra dentro de un archivo .html si no que se genera a partir de un programa que se ejecuta en el servidor.

## 5.- Aplicaciones web dinámicas

La extensión del archivo dependerá del lenguaje de programación de páginas web dinámicas que entienda el servidor:

- **.php** → PHP
- **.py** → Python
- **.js** → JavaScript (node.js)
- **.jsp** → Java
- **.asp** → C#
- ...

El servidor web debe ser capaz de ejecutar programas en dicho lenguaje de programación.

## 5.- Aplicaciones web dinámicas

### Pasos en el servidor al recibir una petición web de una página dinámica.

Se analiza línea por línea el código del recurso solicitado.

Si es código html se deja igual.

Si es código del lenguaje de programación del servidor lo ejecuta.

La ejecución del lenguaje de programación del servidor suele incluir:

- Acceso a base de datos.

- Acceso a otros archivos.





La ejecución de lenguaje de programación del servidor puede crear o no código html, en el caso de que se cree código html se añadirá en ese punto del documento.

Una vez analizadas todas las líneas de código se envía al cliente el documento generado. este documento **sólo contendrá código html**.







## 6.- Estáticas VS Dinámicas

### Ventajas e inconvenientes estáticas y dinámicas

#### Estáticas:

- No es necesario saber programar. 
- Su contenido nunca varía, los enlaces siempre mostrarán lo mismo. 
- Mejor posicionamiento SEO al tener siempre el mismo contenido. 
- Actualización de manera manual por el desarrollador web. 

#### Dinámicas:

- Más flexibilidad. 
- Mayor dificultad en el desarrollo. 
- Mayor consumo de recursos. 
- Hay que ser cuidadoso para el posicionamiento SEO. 
- Menor velocidad. 
- Mayor coste de mantenimiento de recursos. 

## 6.- Estáticas VS Dinámicas

Hoy en días la mayoría de aplicaciones web contienen partes estáticas y partes dinámicas.

Por ejemplo, las secciones: Contacto, Términos y condiciones, Ubicación...

Las partes estáticas también se generan con archivos que se ejecutan en el servidor (como php) pero que siempre generan el mismo código, de esta manera se pueden aprovechar las ventajas del lenguaje de programación como se verá más adelante.

Esto ocurre porque no todo está almacenado en una base de datos ni se necesita procesar información para mostrar contenido.

**En la unión está la potencia de las aplicaciones web actuales.**



## 7.- Front-end y Back-end

En el mundo del desarrollo web es muy común que aparezcan los siguientes términos:

- **Front-end** o simplemente **front**
- **Back-end** o simplemente **back**.

Estos términos tienen dos significados diferentes según si se refieren a:

- Las **secciones** de una aplicación web.
- Al **desarrollo** de una aplicación web.

## 7.- Front-end y Back-end

Si se refieren a **las secciones de una aplicación web**:

En muchas ocasiones las aplicaciones web dinámicas pueden contener una sección solo disponible para el administrador de la aplicación.

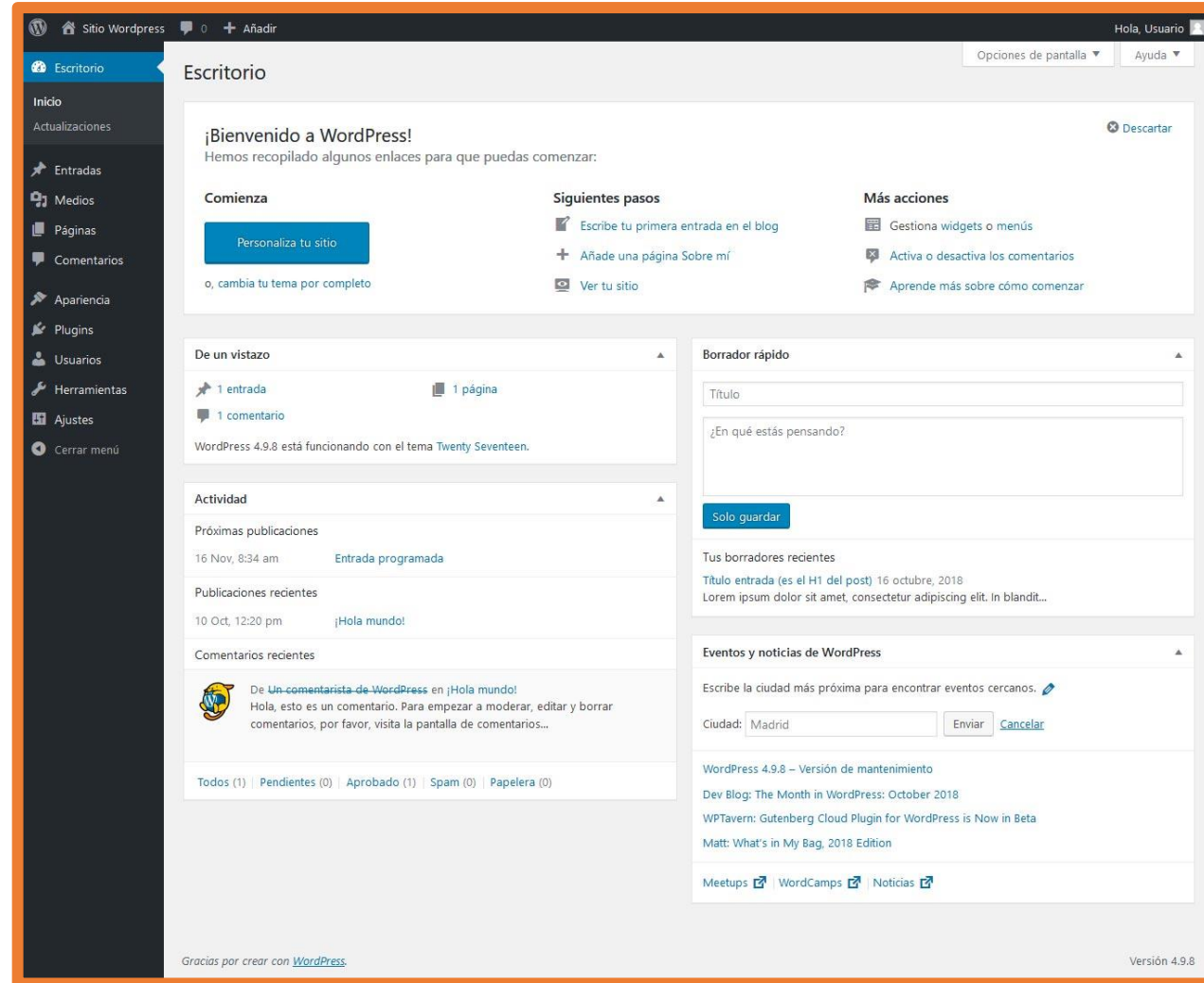
Desde esa sección podrá modificar los contenidos, los estilos, los usuarios... de la aplicación web.

En estos casos la aplicación se conforma de:

- Lo que ven los usuarios: **Front-end**
- Lo que ve el administrador: **Back-end**

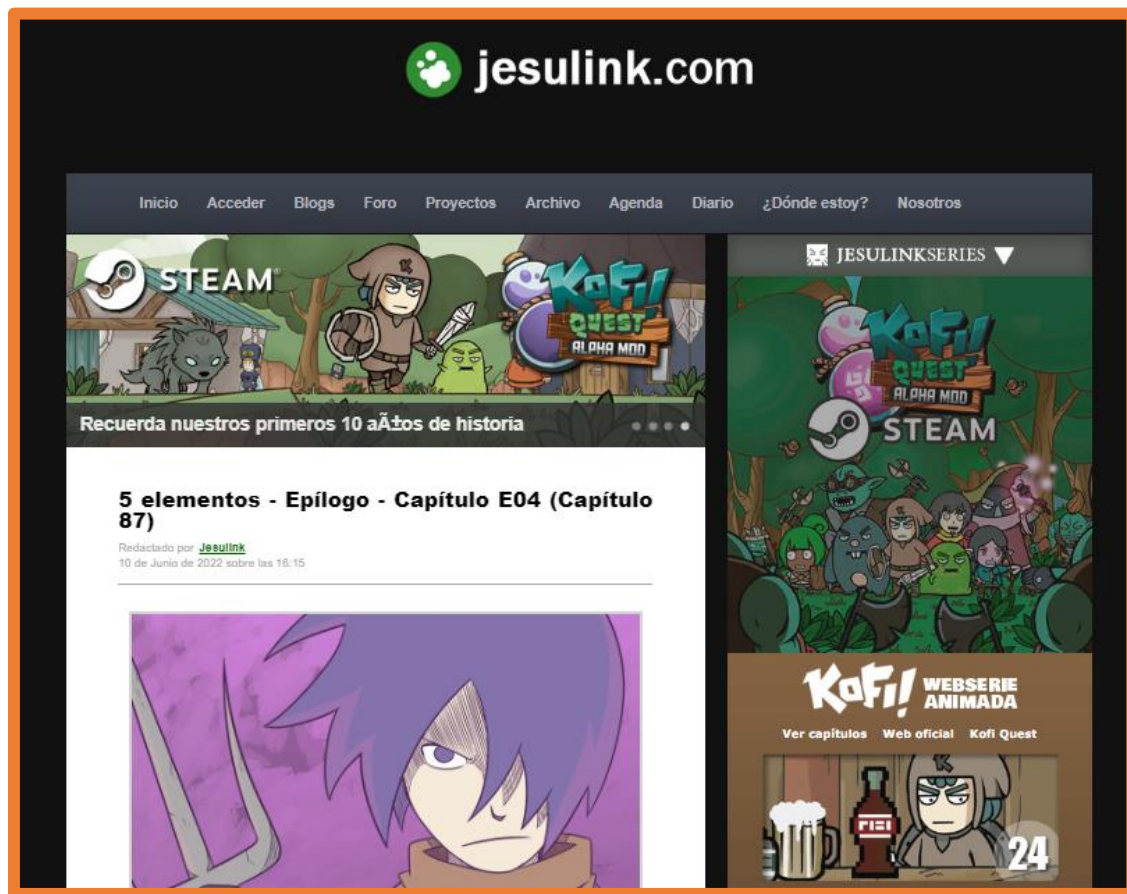
# 7.- Front-end y Back-end

## Back-end de un blog realizado con WordPress.



# 7.- Front-end y Back-end

## Front-end



## Back-end



## 7.- Front-end y Back-end

Si se refieren a **al desarrollo de una aplicación web**:

El **Front-end** es la parte de desarrollo que trabaja toda la parte de la aplicación web que el usuario visualizará, en esta intervienen los estilos, las imágenes, la **interacción**.

El **Back-end** es la parte de desarrollo que se ejecuta en el servidor como toda la gestión de las sesiones o las conexiones a la base de datos.

## 7.- Front-end y Back-end

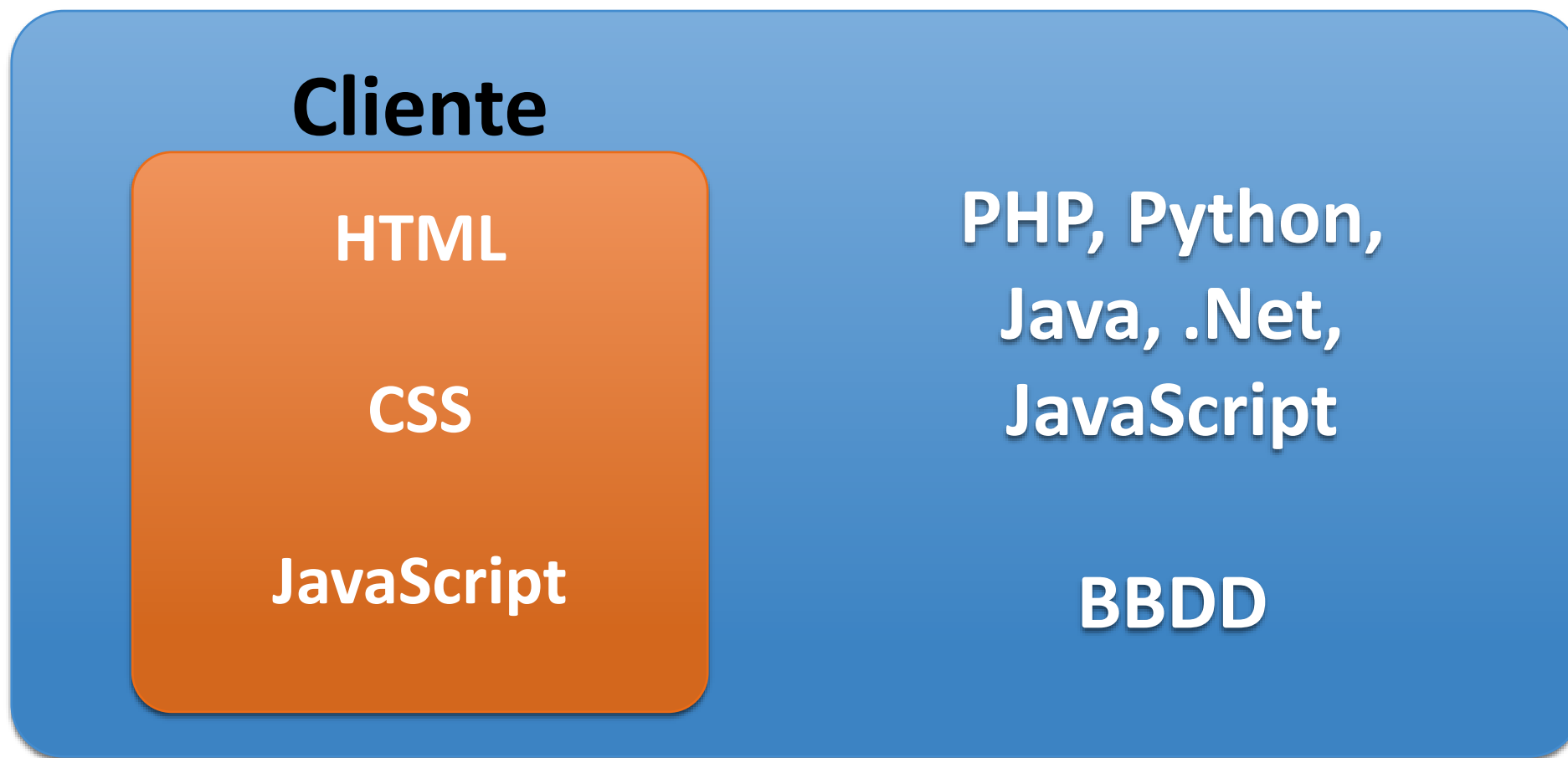
En muchas ofertas de trabajo se piden perfiles determinados:

Perfil	Herramientas	Tecnologías
Front-end	Navegador web IDE	HTML CSS JavaScript
Back-end	Front-end + Servidor web/aplicaciones Base de datos	Front-end + PHP, Python, Java, .Net
Full-stack	Front-end + Back-end	Front-end + Back-end

Además también se suele pedir conocimientos en algún **FrameWork** de front-end y/o back-end.

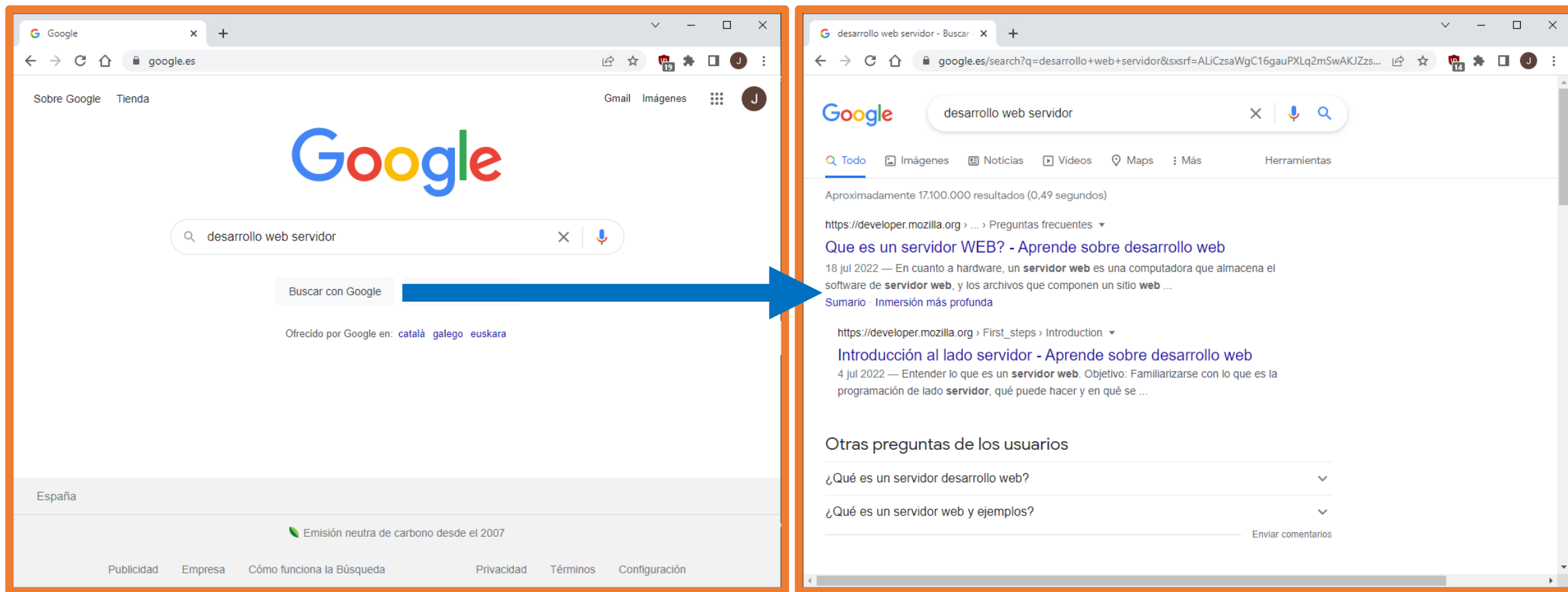
## 8.- Tecnologías para la programación web

### Servidor



# 9.- Aplicaciones web dinámicas: síncronas VS asíncronas

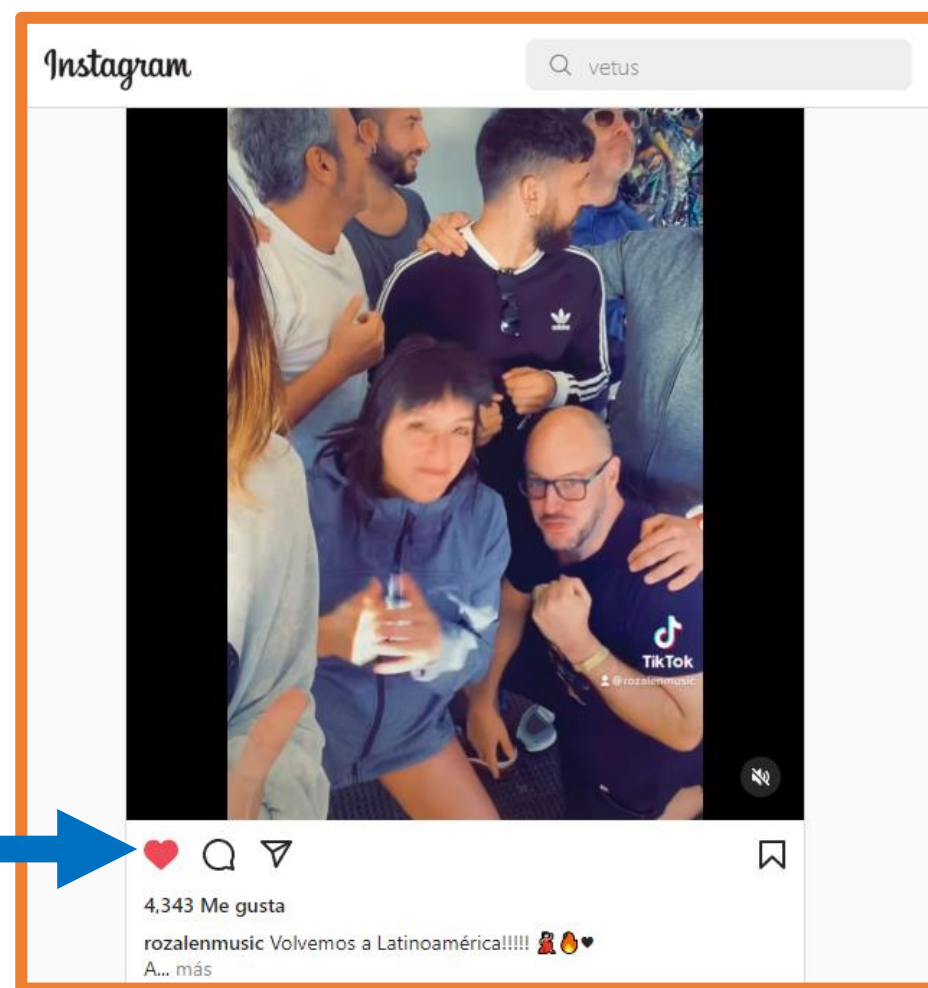
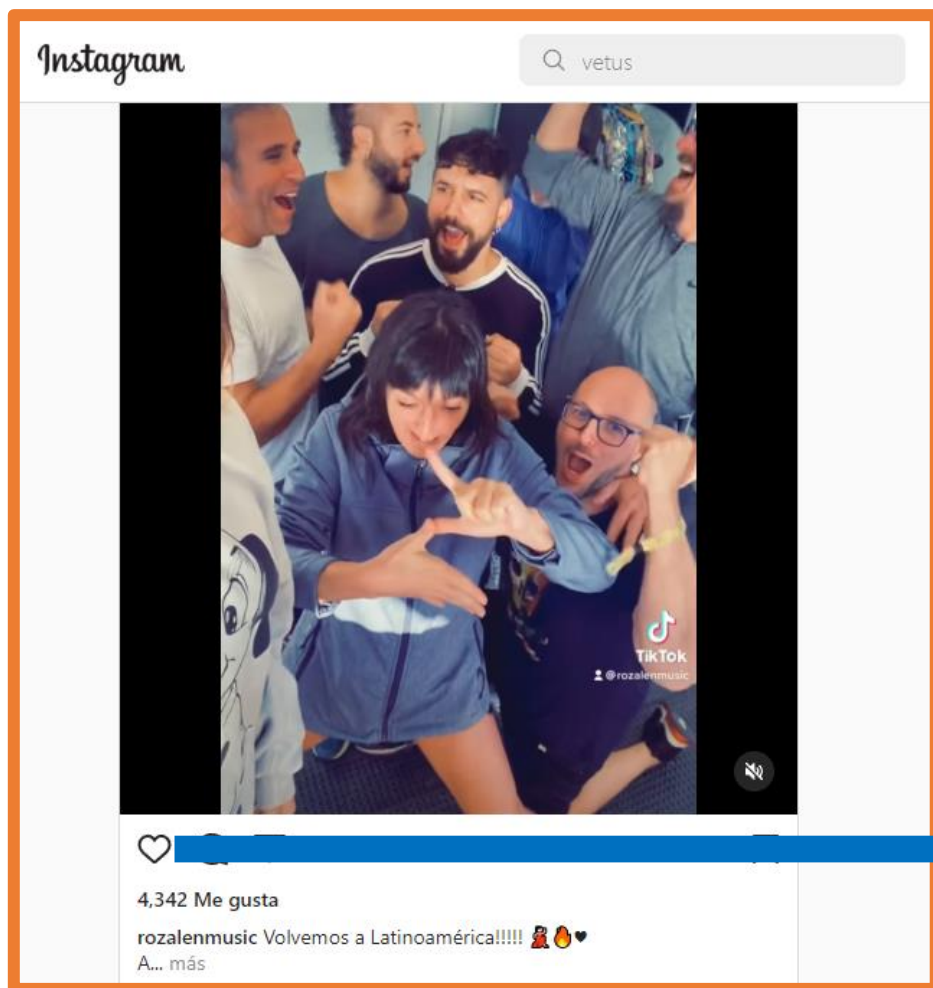
Tradicionalmente las aplicaciones web cambian toda la página cuando se realiza una petición: **Síncronas**.





## 9.- Aplicaciones web dinámicas: síncronas VS asíncronas

Existe otro sistema de funcionamiento que hace el uso de peticiones asíncronas (AJAX) mediante las cuales se permite modificar una parte de la página sin que se produzca una recarga: **Asíncronas**.



## 9.- Aplicaciones web dinámicas: síncronas VS asíncronas

### Asíncronas

Hoy en día el uso de AJAX forma una gran parte del desarrollo de aplicaciones web.

- Me gusta.
- Suscribirse.
- Comentarios.
- Carga automática de contenido.
- ...

## 9.- Aplicaciones web dinámicas: síncronas VS asíncronas

Existe una vertiente en la que **toda la aplicación web funciona con peticiones asíncronas**, son las llamadas **SPA**.

**SPA** (Single Page Application).

En este tipo de aplicaciones web la URL nunca cambia y esto conlleva diferentes problemas como:

- Dificulta la navegabilidad si el usuario usa las flechas del navegador.
- Los buscadores web no saben cuál es el contenido de la aplicación.
- Mal posicionamiento SEO (Search Engine Optimization).

## 10.- Arquitecturas de diseño

La **arquitectura de diseño** es la manera en la que se organizará el código.

En el desarrollo web, como en la mayor parte de desarrollo de software, se usa una **arquitectura en 3 capas**.

- Capa de **presentación**: interpreta las peticiones del usuario y muestra la información.
- Capa de **proceso**: operaciones de la aplicación que generan las páginas.
- Capa de **acceso a datos**: se encarga de almacenar y recuperar los datos.

## 10.- Arquitecturas de diseño

El patrón de diseño más usado para una arquitectura de 3 capas es el llamado **MVC** (Modelo Vista Controlador).

- **Modelo:** capa de acceso a datos.
- **Vista:** capa de presentación.
- **Controlador:** capa de proceso.

MVC se estudiará más adelante con el uso de los **Framework**.

# 11.- Plataformas de desarrollo

Estas son las plataformas comunes para el desarrollo de aplicaciones web dinámicas:

Plataforma	Servidor	Lenguajes de servidor	Comentarios
<b>AMP</b>	Apache	PHP, Perl, Python	Open Source. MySQL o MariaDB. Se puede sustituir Apache por Nginx
<b>JavaEE</b>	Tomcat	Java	JSP y servlets. Existen muchas librerías disponibles.
<b>CGI/Perl</b>	Cualquiera con soporte CGI		CGI permite ejecutar en el servidor web programas en cualquier lenguaje. Lento.
<b>ASP.Net</b>	Microsoft IIS	Visual Basic, C#	Propietario. Incluye IDE.

Existen otras configuraciones en las que se utiliza el servidor web Nginx e incluso otras que usan un servidor web virtual como cuando se usan Frameworks.

## 12.- Decisiones de diseño

¿Qué tamaño tendrá el proyecto?

¿Qué lenguajes de programación conozco? ¿Vale la pena el esfuerzo de aprender uno nuevo?

Herramientas públicas o propietarias.

Coste de soluciones comerciales.

Cantidad de personas del equipo de desarrollo.

¿Tengo ya un servidor web o gestor de bases de datos o puedo elegirlos?

## 13.- Entorno de trabajo para el curso


Para este curso se va a utilizar la siguiente configuración:

- Servidor web: **Apache** (con intérprete de PHP)
- Lenguaje de programación: **PHP**
- Servidor de base de datos: **MariaDB**
- IDE (Integrated Development Enviroment): **Visual Studio Code**.



## 13.- Entorno de trabajo para el curso

Para este curso se va a utilizar la siguiente configuración:

- Servidor web: **Apache** (con intérprete de PHP)
  - Lenguaje de programación: **PHP**
  - Servidor de base de datos: **MariaDB**
  - IDE (Integrated Development Enviroment): **Visual Studio Code.**
- 
- The diagram consists of a large orange curly bracket on the right side of the first three list items. To the right of the bracket is a rectangular box with an orange border containing the text **XAMPP**. This indicates that XAMPP is a package that includes the web server, programming language, and database server components.

# XAMPP

¿Qué es XAMPP?

¿Es necesario?

¿Por qué usar XAMPP?

# 13.- Entorno de trabajo para el curso

## Visual Studio Code.

### Emmet

Cambiar a lang="es".

Generar estructura mínima HTML:     ! + tab

Sintaxis de Emmet. :     div#lista>ul>li\*4

# Hosts virtuales

Las páginas web estáticas, con extensión **.html**, se pueden ejecutar sin disponer de un servidor web.

Es el navegador el que interpreta el código de dichos archivos.

Se ha visto en el primer ejercicio en el que se probaron las etiquetas HTML.

Al hacer doble clic a un archivo **.html** el archivo se abre en el navegador directamente.

# Hosts virtuales

Para poder ejecutar archivos **.php** se necesita que el que ejecute estos archivos sea el servidor web.

A partir de ahora nuestros proyectos se deberán guardar en el **repositorio** de páginas web del **servidor Apache**.

En apache el repositorio de páginas web es el directorio **htdocs**.

En principio en el servidor web sólo tendremos una aplicación web, **el proyecto en el que estemos trabajando en cada momento**. Para esto nos tenemos que asegurar que en el directorio **htdocs** no haya ningún archivo que no sea de nuestro proyecto.

# Hosts virtuales

Gracias a los **hosts virtuales** se puede simular que en un mismo equipo hay diferentes servidores web con sus diferentes repositorios de archivos.

De esta manera los archivos de un proyecto no interfieren en otro proyecto.

Los hosts virtuales se ven detalladamente en el módulo DAW (Despliegue de Aplicaciones Web) por ello en este módulo solo se explicará brevemente como se crean para así poder hacer uso de ellos y sus ventajas desde el principio.

# Hosts virtuales

Para crear un host virtual en XAMPP se debe acceder al archivo de configuración correspondiente:

`C:\xampp\apache\conf\extra\httpd-vhosts.conf`

Dentro de ese archivo se incorporará un bloque por cada host virtual que se desee.

Evidentemente se pueden crear todos los que uno necesite.

# Hosts virtuales

```
<VirtualHost *:80>
```

```
    DocumentRoot "C:/ruta_del_directorio_del_proyecto"
```

```
    ServerName nombre_del_servidor
```

```
    ErrorLog "logs/nombre_del_servidor -error.log"
```

```
    CustomLog "logs/nombre_del_servidor -access.log" common
```

```
</VirtualHost>
```



# Hosts virtuales

# VirtualHost para que siga funcionando localhost

```
<VirtualHost *:80>
    DocumentRoot "C:/xampp/htdocs"
    ServerName localhost
    ErrorLog "logs/localhost-error.log"
    CustomLog "logs/localhost-access.log" common
</VirtualHost>
```

# VirtualHost para las actividades de clase

```
<VirtualHost *:80>
    DocumentRoot "C:/xampp/htdocs/actividades"
    ServerName localhost.actividades
    ErrorLog "logs/actividades-error.log"
    CustomLog "logs/actividades-access.log" common
</VirtualHost>
```

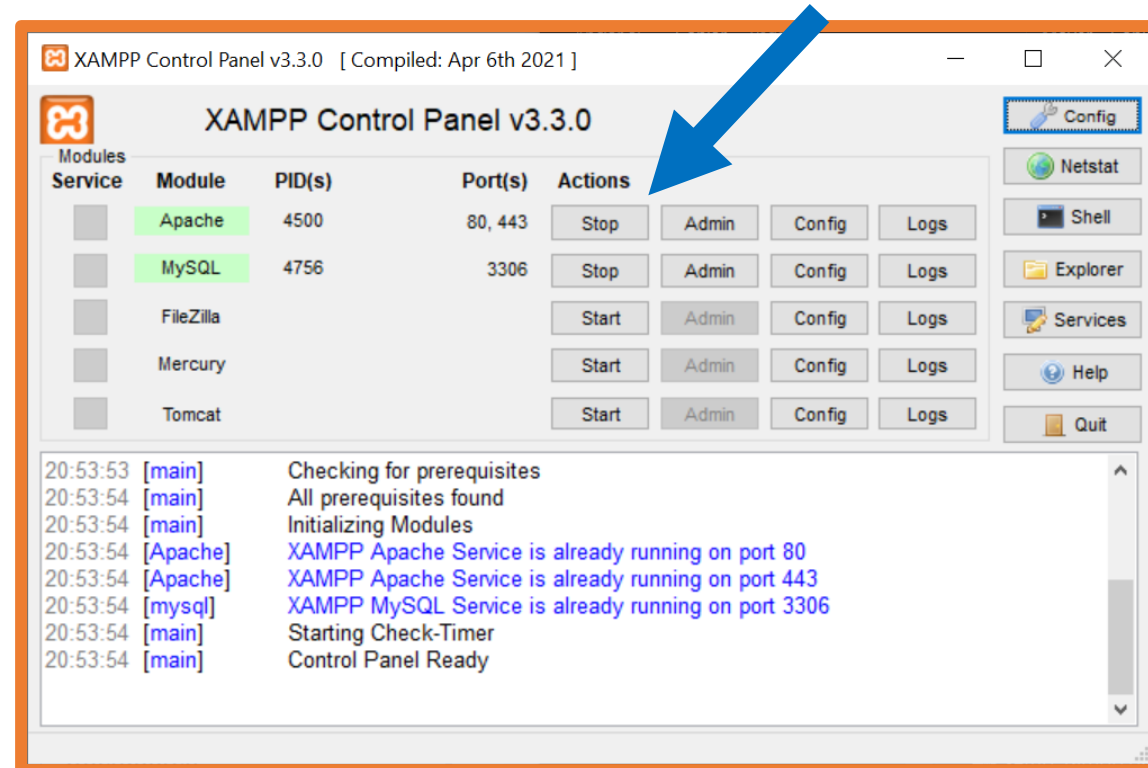
# VirtualHost para el proyecto del trimestre

```
<VirtualHost *:80>
    DocumentRoot "C:/xampp/htdocs/proyecto"
    ServerName localhost.proyecto
    ErrorLog "logs/proyecto-error.log"
    CustomLog "logs/proyecto-access.log" common
</VirtualHost>
```

# Hosts virtuales

Evidentemente se pueden crear todos los que uno necesite.

Tras añadir la configuración de los hosts virtuales se debe reiniciar el servicio Apache.



# Hosts virtuales

El último paso consiste en indicar al servicio DNS del propio ordenador dónde se alojan esos servidores virtuales.

Para ello se debe abrir con permisos de administrador el archivo:

C:\Windows\System32\drivers\etc\hosts

Y añadir en él una línea por cada host virtual configurado:

```
# localhost name resolution is handled within DNS itself.  
#          127.0.0.1          localhost  
#          ::1                localhost  
  
127.0.0.1      localhost  
127.0.0.1      localhost.actividades  
127.0.0.1      localhost.proyecto
```

# Práctica

## Actividad 2:

Preparando el entorno de programación y pruebas