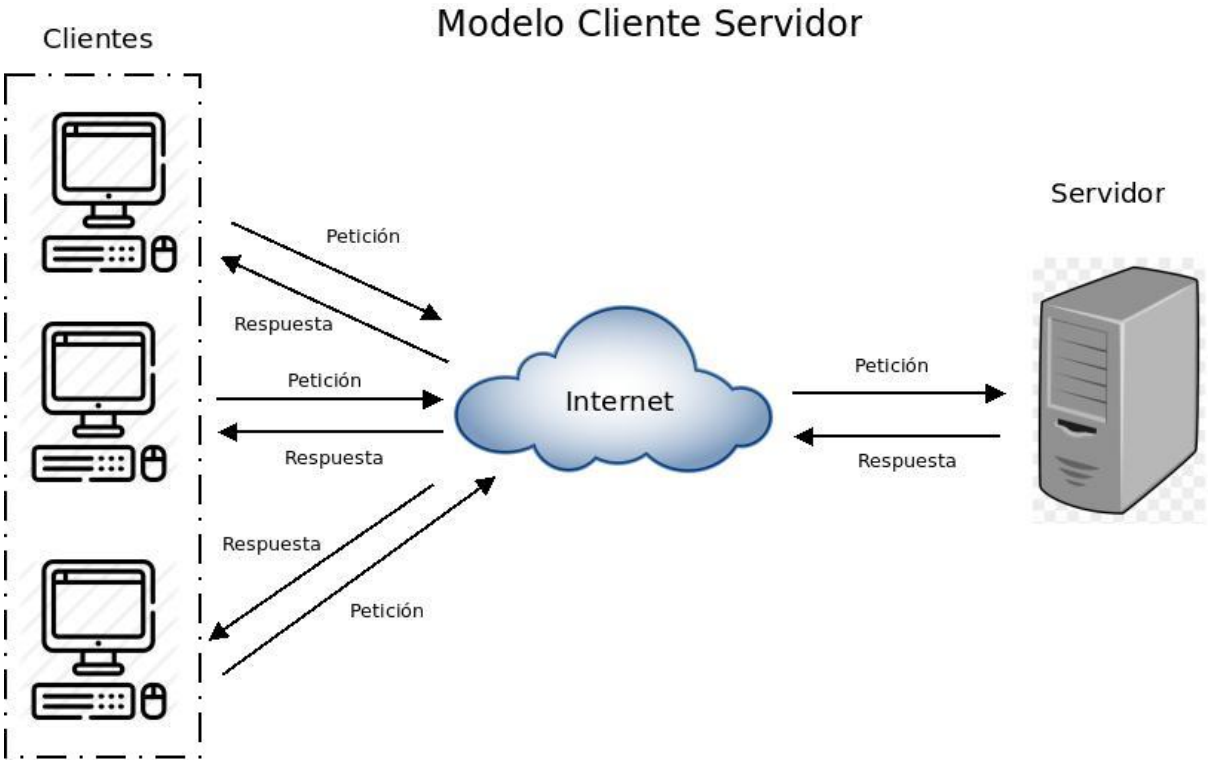
Las arquitecturas web definen la forma en que las páginas de un sitio web están estructuradas y enlazadas entre sí. Las aplicaciones web se basan en el modelo cliente-servidor.

Cliente / Servidor



Arquitectura Cliente Servidor

Uno o varios cliente acceden a un servidor. Las nuevas arquitecturas permiten sustituir el servidor por un balanceador de carga de manera que N servidores den respuesta a M clientes.

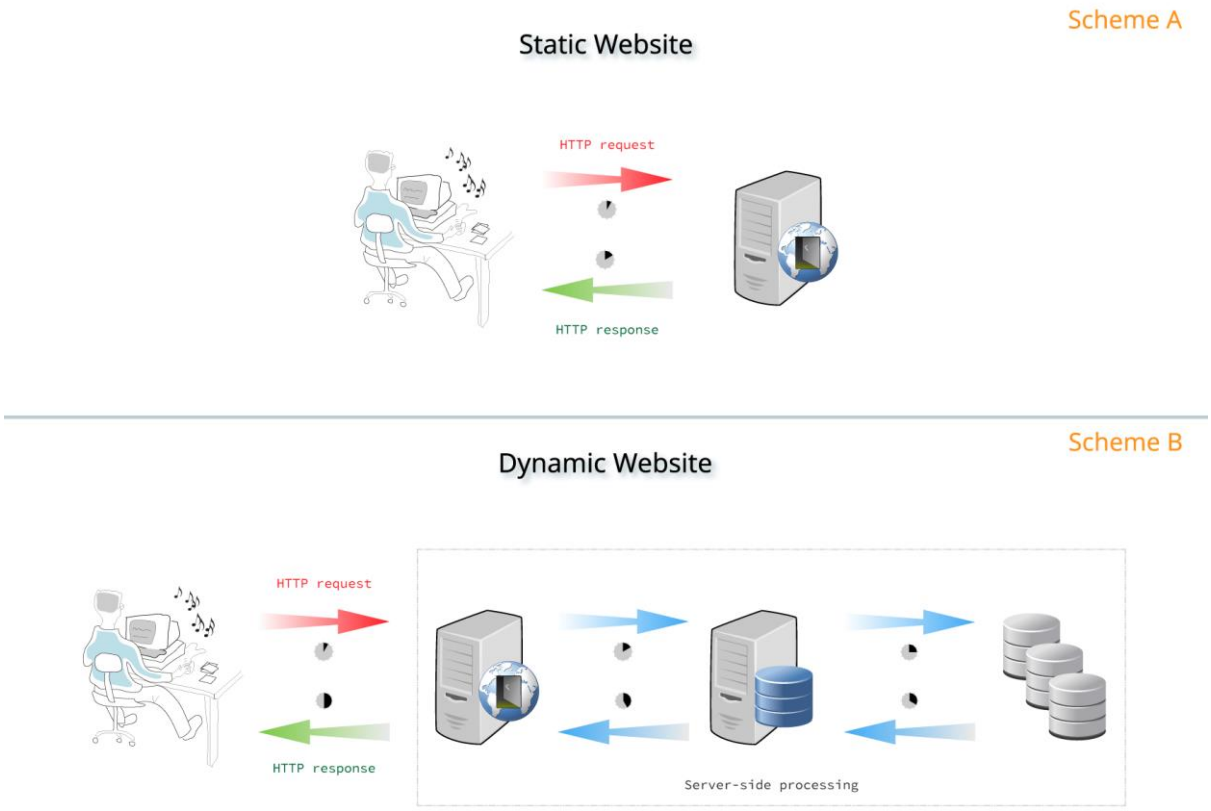
En las aplicaciones web, el cliente es el navegador web.

El cliente hace la petición (*request* normalmente mediante el protocolo GET mediante el puerto 80/443) y el servidor responde (*response*).

Página web dinámica

Si la página web únicamente contiene HTML + CSS se considera una página estática. Para generar una página dinámica, donde el contenido cambia, hoy en día tenemos dos alternativas:

- Utilizar un lenguaje de servidor que genere el contenido, ya sea mediante el acceso a una BD o servicios externos.
- Utilizar servicios REST de terceros invocados desde JS.



Página web estática vs dinámica

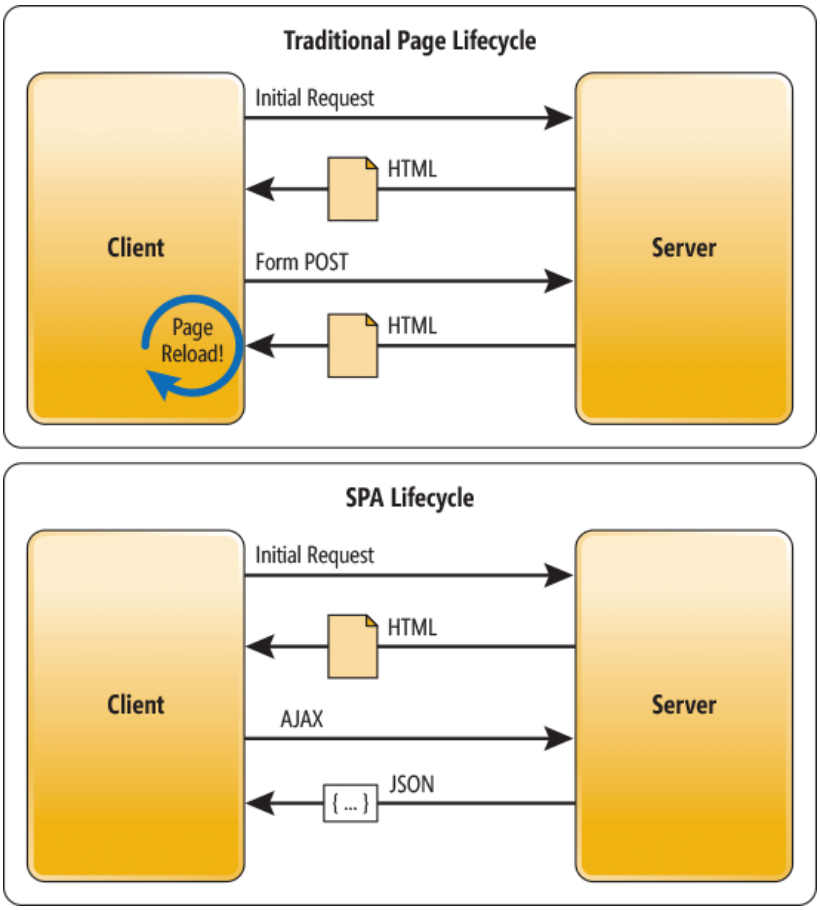
Las tecnologías empleadadas (y los perfiles de desarrollo asociados) para la generación de páginas dinámicas son:

Perfil	**Herramienta**	**Tecnología**
Front-end/Cliente	*Navegador web*	*HTML+CSS+JavaScript*
Back-end/Servidor	*Servidor web+BBDD*	*PHP, Python, Ruby, Java/JSP, -Net/.asp*

Full-stack, perfil que domina tanto el *front-end* como el *back-end*.

Single Page Application

Actualmente, gran parte del desarrollo web está transicionando de una arquitectura web cliente-servidor clásica donde el cliente realiza una llamada al backend, por una arquitectura SPA donde el cliente gana mucho mayor peso y sigue una programación reactiva que accede a servicios remotos REST que realizan las operaciones (comunicándose mediante JSON).



Arquitectura tradicional vs SPA

Arquitectura de 3 capas

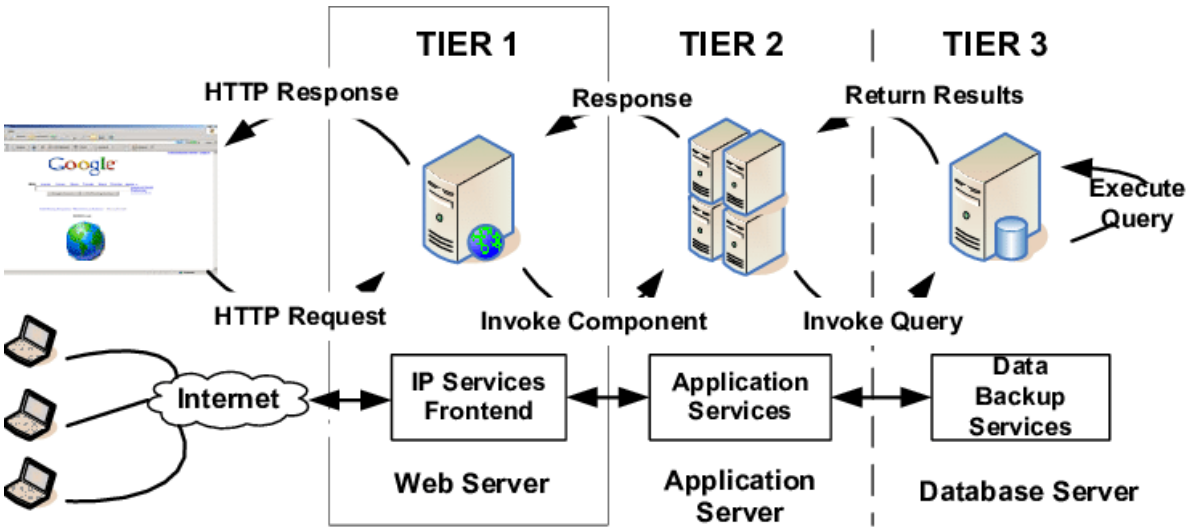
Hay que distinguir entre capas **físicas** (*tier*) y capas **lógicas** (*layer*).

Tier

Capa física de una arquitectura. Supone un nuevo elemento hardware separado físicamente. Las capas físicas más alejadas del cliente están más protegidas, tanto por firewalls como por VPN.

Ejemplo de arquitectura en tres capas físicas (*3 tier*):

- Servidor Web
- Servidor de Aplicaciones
- Servidor de base de datos

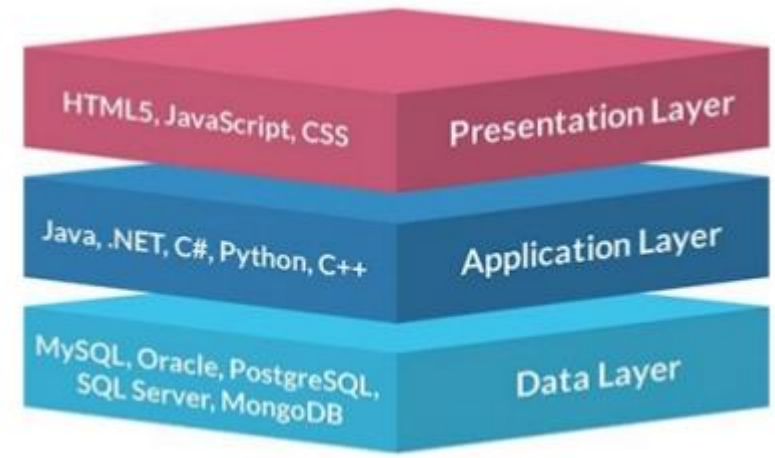


Arquitectura de tres capas físicas

Cluster en tiers

No confundir las capas con la cantidad de servidores. Actualmente se trabaja con arquitecturas con múltiples servidores en una misma capa física mediante un cluster, para ofrecer tolerancia a errores y escalabilidad horizontal.

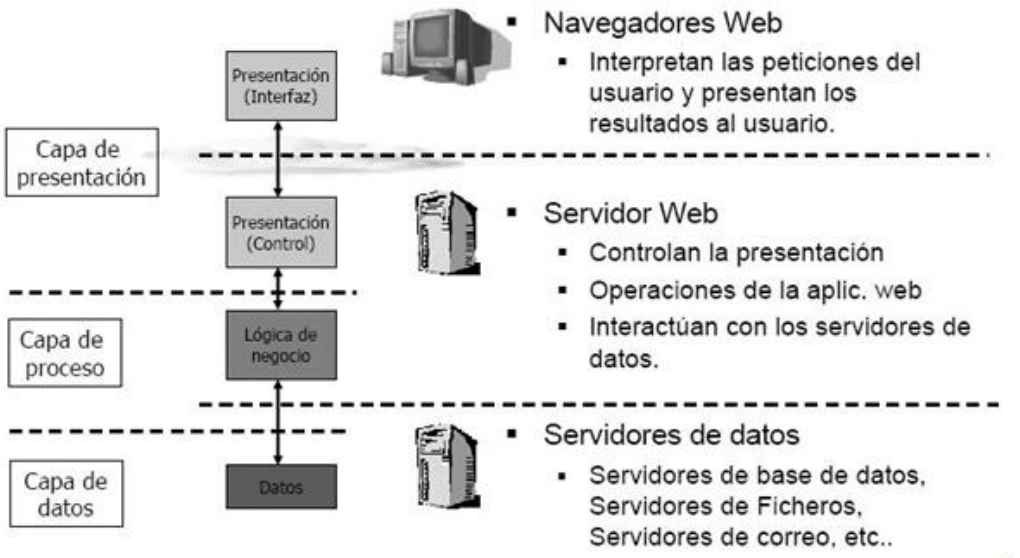
Layer



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

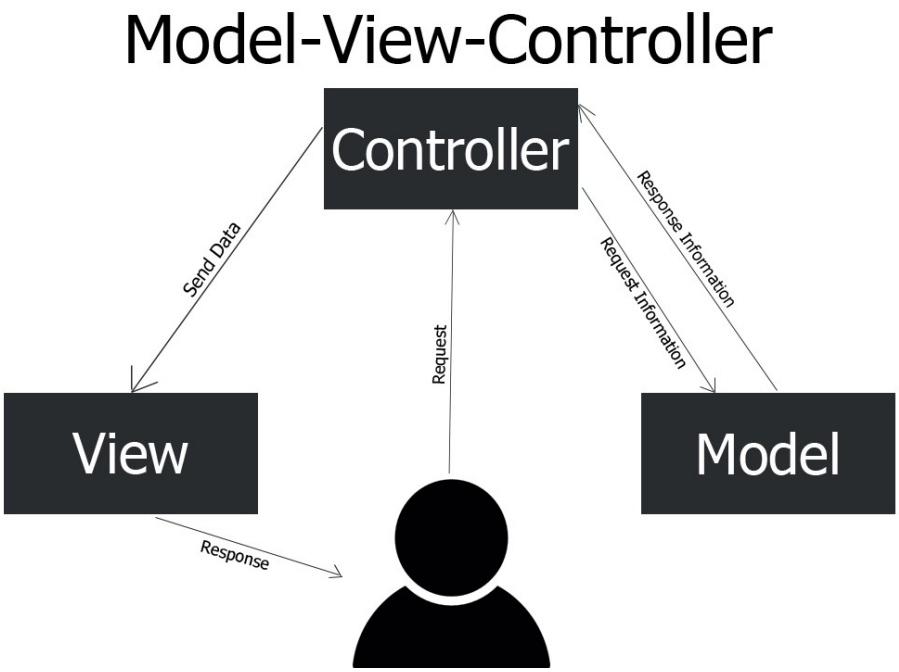
- Presentación
- Negocio / Aplicación / Proceso
- Datos / Persistencia

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



Arquitectura de tres capas físicas en tres lógicas

MVC



Model-View-Controller o Modelo-Vista-Controlador es un modelo de arquitectura que separa los datos y la lógica de negocio respecto a la interfaz de usuario y el componente encargado de gestionar los eventos y las comunicaciones.

Al separar los componentes en elementos conceptuales permite reutilizar el código y mejorar su organización y mantenimiento. Sus elementos son:

- **Modelo**: representa la información y gestiona todos los accesos a ésta, tanto consultas como actualizaciones provenientes, normalmente, de una base de datos. Se accede vía el *controlador*.
- **Controlador**: Responde a las acciones del usuario, y realiza peticiones al modelo para solicitar información. Tras recibir la respuesta del *modelo*, le envía los datos a la *vista*.
- **Vista**: Presenta al usuario de forma visual el *modelo* y los datos preparados por el *controlador*. El usuario interactúa con la *vista* y realiza nuevas peticiones al *controlador*.

Lo estudiaremos en más detalle al profundizar en el uso de los frameworks PHP.

Decisiones de diseño

- ¿Qué tamaño tiene el proyecto?
- ¿Qué lenguajes de programación conozco? ¿Vale la pena el esfuerzo de aprender uno nuevo?
- ¿Voy a usar herramientas de código abierto o herramientas propietarias? ¿Cuál es el coste de utilizar soluciones comerciales?
- ¿Voy a programar la aplicación yo solo o formaré parte de un grupo de programadores?
- ¿Cuento con algún servidor web o gestor de base de datos disponible o puedo decidir libremente utilizar el que crea necesario?
- ¿Qué tipo de licencia voy a aplicar a la aplicación que desarrolle?

Herramientas

Servidor Web

Software que recibe peticiones HTTP (GET, POST, DELETE, ...). Devuelve el recurso solicitado (HTML, CSS, JS, JSON, imágenes, etc).

En la actualidad, Nginx (https://www.nginx.com), creado en 2004, es el servidor web más utilizado, seguido muy de cerca por Apache Web Server (https://httpd.apache.org/), creado en 1995.

Algunas de las diferencias entre ambos son:

- Apache necesita abrir múltiples hilos para atender múltiples peticiones de clientes, mientras que Nginx con un hilo puede atender múltiples peticiones de clientes. Como resultado, consume menos recursos y es capaz de responder más rápidamente.
- Mientras que en alta carga Apache no es capaz de atender múltiples peticiones de forma concurrente y asíncrona, Nginx es todo lo contrario. Está diseñado para atender muchas peticiones de forma asíncrona y

siendo lo más eficiente posible al consumir recursos de RAM y CPU. Esto le permite gestionar mejor los ataques DoS (Denial of Service o Denegación de Servicio).
- Mientras que Apache permite ejecutar PHP dentro del propio servidor web, en Nginx debemos externalizarlo, lo que es mucho más eficiente.
- Nginx es muy eficiente al servir contenido estático (imágenes, PDF, CSS, Javascript, etc.), mientras que Apache es más lento y consume más recursos.
- Apache es un proyecto mucho más maduro, el único durante muchos años, con lo que tiene estandarizadas cosas como la configuración con .htaccess, al que los desarrolladores están acostumbrados.
- Aunque Apache ha mejorado mucho en los últimos años con la aparición de la versión 2.4, sigue estando en desventaja frente a Nginx.
  Comparativa servidores web:
  https://w3techs.com/technologies/history_overview/web_server/ms/q

Servidor de Aplicaciones

- Software que ofrece servicios adicionales a los de un servidor web:
  - Clustering
  - Balanceo de carga
  - Tolerancia a fallos
- *Tomcat* (http://tomcat.apache.org/) es el servidor de aplicaciones *open source* y multiplataforma de referencia para una arquitectura Java.
  - Contiende un contenedor Web Java que interpreta *Servlets* y JSP.

Lenguajes en el servidor

Las aplicaciones que generan las páginas web se programan en alguno de los siguientes lenguajes:

- PHP
- JavaEE: Servlets / JSP
- Python
- ASP.NET → Visual Basic .NET / C#
- Ruby
- ...

PHP

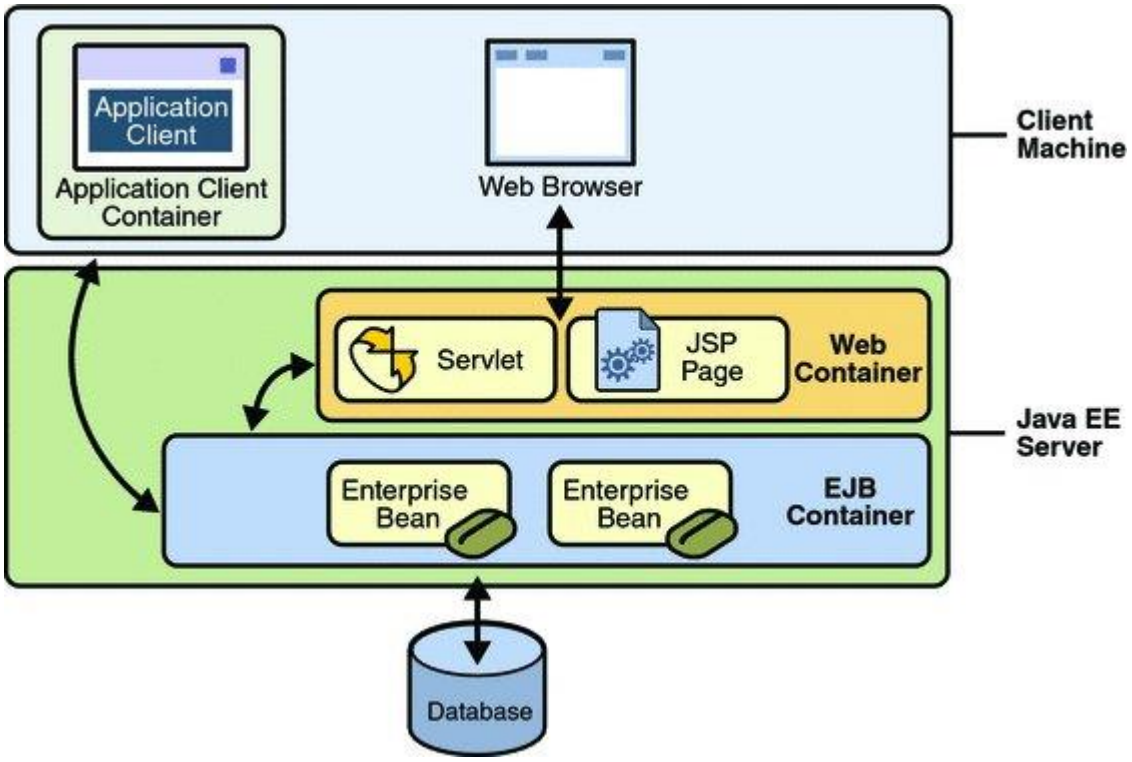
- Lenguaje de propósito general diseñado para el desarrollo de páginas web dinámicas.
- Actualmente en la versión 8. Se recomienda al menos utilizar una versión superior a la 7.0.
- Código embebido en el HTML.
- Instrucciones entre etiquetas <?php y ?>
- Multitud de librerías y frameworks:
    - Laravel, Symfony, Codeigniter, Zend.

Su documentación es bastante completa:
https://www.php.net/manual/es/index.php

Durante las siguientes unidades vamos a estudiar PHP en profundidad.

JavaEE

Java Enterprise Edition es la solución Java para el desarrollo de aplicaciones *enterprise*. Ofrece una arquitectura muy completa y compleja, escalable y tolerante a fallos. Planteada para aplicaciones para grandes sistemas.



Puesta en marcha

Para poder trabajar con un entorno de desarrollo local, hemos de preparar nuestro entorno de desarrollo con las herramientas comentadas. A lo largo del curso vamos a utilizar la versión 8 de PHP.

XAMPP

XAMPP (https://www.apachefriends.org/es/index.html) es una distribución compuesta con el software necesario para desarrollar en entorno servidor. Se compone de las siguientes herramientas en base a sus siglas:

- X para el sistema operativo (de ahí que se conozca también como LAMP o WAMP).
- A para Apache.
- M para MySQL / MariaDB. También incluye phpMyAdmin para la administración de la base de datos desde un interfaz web.
- P para PHP.
- la última P para Perl.

Desde la propia página se puede descargar el archivo ejecutable para el sistema operativo de nuestro ordenador. Se recomienda leer la FAQ de cada sistema operativo con instrucciones para su puesta en marcha.

XAMPP en Windows

Si vas a trabajar con tu propio ordenador, XAMPP es una solución más sencilla que Docker, sobre todo si trabajas con Windows como sistema operativo.

Docker

Docker (https://www.docker.com) es un gestor de contenedores, considerando un contenedor como un método de virtualización del sistema operativo.

El uso de contenedores requiere menos recursos que una máquina virtual, por lo tanto, su lanzamiento y detención son más rápidos que las máquinas virtuales.

Así pues, *Docker* permite crear, probar e implementar aplicaciones rápidamente, a partir de una serie de plantillas que se conocen como imágenes de *Docker*.

Para ello es necesario tener instalado *Docker Desktop* (https://www.docker.com/products/docker-desktop) en nuestros entornos de desarrollo (el cual ya incluye en núcleo de *Docker* y la herramienta *docker compose*).

Plantilla Servidor Web + PHP

Docker se basa en el uso de imágenes para crear contenedores. *Docker Compose* simplifica el trabajo con múltiples contenedores, y por ello, para facilitar el arraque, nos centraremos en *Docker Compose* utilizando una plantilla que únicamente va a contener como servicios Apache/Nginx y PHP.

Para ello, vamos a rellenar el archivo *docker-compose.yaml* con:

Apache y PHP

```
# Services
services:
  # Apache + PHP
  apache_php:
    image: php:8-apache
    # Preparamos un volumen para almacenar nuestro código
    volumes:
      - ./src/:/var/www/html
    expose:
      - 80
    ports:
      - 80:80
```

Dentro de la carpeta que contenga dicho archivo, hemos de crear una carpeta src donde colocaremos nuestro código fuente. Para facilitar la puesta en marcha, tenéis la plantilla de Apache/PHP (plantillaAP2.zip con a2enmod rewrite) disponible para su descarga en Aules.

Cuando estemos listos, lanzaremos el servicio mediante:

```
docker-compose up -d
```

Si queremos ver el contenido de los archivos de log del servicio utilizaremos:

```
docker-compose logs -f
```

Para copiar un archivo desde nuestro sistema al interior del contenedor:

```
docker cp ./miFichero idContenedor:/tmp
```

Y al revés, si queremos consultar un archivo contenido dentro de un contenedor, lo copiaremos a nuestro sistema:

```
docker cp idContenedor:/tmp/archivoAConsultar.txt ./
```

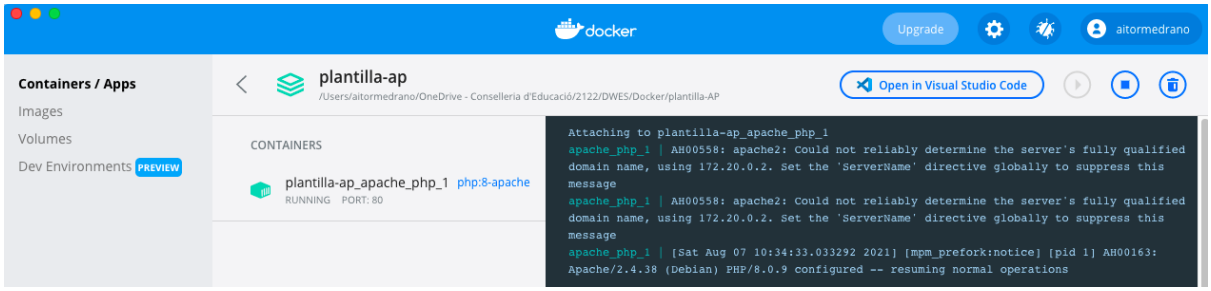
Finalmente, si queremos acceder a un terminal interactivo dentro del contenedor:

```
docker exec -it nombreContenedor bash
```

Otros comandos que podemos llegar a utilizar son:

- docker ps: Ver los contenedores que se estan ejecutando
- docker ps -a: Ver todos los contenedores
- docker start nombreContenedor: Arrancar un contenedor
- docker images: Ver las imágenes que tenemos descargadas

Otra forma más sencilla para lanzar de nuevo los contenedores y gestionarlos una vez creados es utilizar el interfaz gráfico que ofrece *Docker Desktop*:



Arranque de contenedor mediante Docker Desktop

Entorno de desarrollo

En este curso vamos a emplear *Visual Studio Code* (https://code.visualstudio.com) como entorno de desarrollo (IDE).

Existen otras alternativas, siendo PhpStorm la más conocida, pero siendo de pago. Otra posibilidad es utilizar Eclipse, aunque es un entorno bastante pesado.

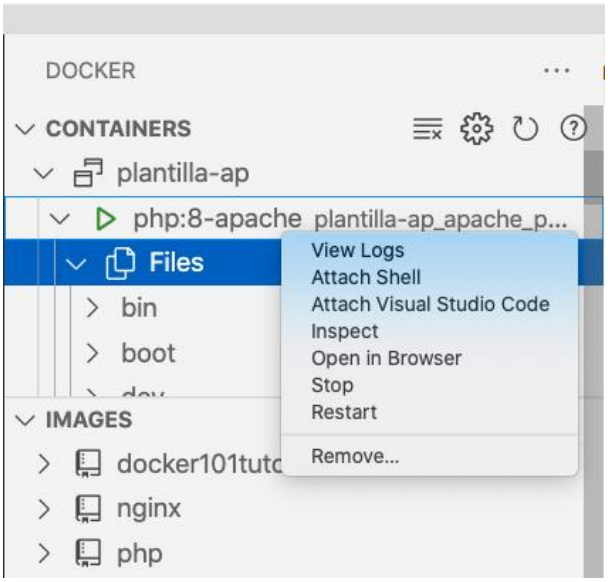
VSCode es un editor de código fuente que se complementa mediante extensiones. Para facilitar el trabajo a lo largo del curso vamos a utilizar las siguientes extensiones:

- PHP Intelephense
- Docker

En esta sesión nos vamos a centrar en *Docker* (más adelante instalaremos nuevas extensiones).

Por ejemplo, si abrimos la extensión de *Docker*, podréis visualizar tanto los contenedores como las imágenes de vuestro sistema. Desde cada

contenedor, mediante clic derecho, podemos iniciar/detener/reiniciar cada contenedor, así como ver su contenido o abrir un terminal dentro del mismo.



Opciones mediante extensión Docker en VSCode

Hola Mundo

Y como no, nuestro primer ejemplo será un *Hola Mundo* en PHP.

Si nombramos el archivo como index.php, al acceder a http://localhost automáticamente cargará el resultado:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hola Mundo</title>
</head>
<body>
    <?php
        echo "Hola Mundo";
    ?>
</body>
</html>
```

Referencias

- Curso DWES Aitor Medrano: https://aitor-medrano.github.io/dwes2122
- Curso de introducción a Docker, por *Sergi García Barea*: https://sergarb1.github.io/CursoIntroduccionADocker/
- Artículo Arquitecturas Web y su evolución
- Raiola Networks: https://raiolanetworks.com/blog/nginx-vs-apache-cual-es-mejor-servidor-web/