UD1- Arquitecturas WEB

??? note "RA1 Selecciona las **arquitecturas** y tecnologías de programación Web en entorno servidor, analizando sus capacidades y características propias"

```
**Criterios de Evaluación**
> *  A Se han caracterizado y diferenciado los modelos de ejecución de código en el servidor y en el
cliente Web.
> *  B Se han reconocido las ventajas que proporciona la generación dinámica de páginas Web y sus
diferencias con la inclusión de sentencias de guiones en el interior de las páginas Web.
> *  C Se han identificado los mecanismos de ejecución de código en los servidores Web.
> *  D Se han reconocido las funcionalidades que aportan los servidores de aplicaciones y su integración
con los servidores Web.
> *  E Se han identificado y caracterizado los principales lenguajes y tecnologías relacionados con la
programación Web en entorno servidor.
> *  F Se han verificado los mecanismos de integración de los lenguajes de marcas con los lenguajes de
programación en entorno servidor.
> *  G Se han reconocido y evaluado las herramientas de programación en entorno servidor.
```

1 Introducción

La web ha evolucionado significativamente desde sus inicios, transformando la manera en que interactuamos en línea. Un breve resumen de su **historia** podría ser:

El nacimiento de la World Wide Web

En 1989, Tim Berners-Lee, un científico del CERN en Suiza, propuso un sistema de hipertexto para compartir información entre investigadores. Este sistema dio lugar a la creación de la **World Wide Web**^1^. En 1991, Berners-Lee lanzó el primer navegador web, llamado WorldWideWeb, y desarrolló HTML, el lenguaje de marcado que se convirtió en la base de la programación web^1^.

HTML y el surgimiento de la programación web

HTML permitió a los desarrolladores crear y organizar contenido en línea utilizando etiquetas y atributos. La rápida adopción de la web llevó a una mayor demanda de contenido en línea y al surgimiento de la programación web^1^.

La Web 2.0

En 2004, el término Web 2.0 fue acuñado para describir la evolución hacia aplicaciones web **interactivas**. Esta fase se caracterizó por la interactividad, la colaboración y la participación de los usuarios en la creación de contenido^1^. JavaScript y AJAX jugaron un papel crucial en esta evolución, permitiendo la actualización dinámica del contenido sin recargar la página completa^1^.

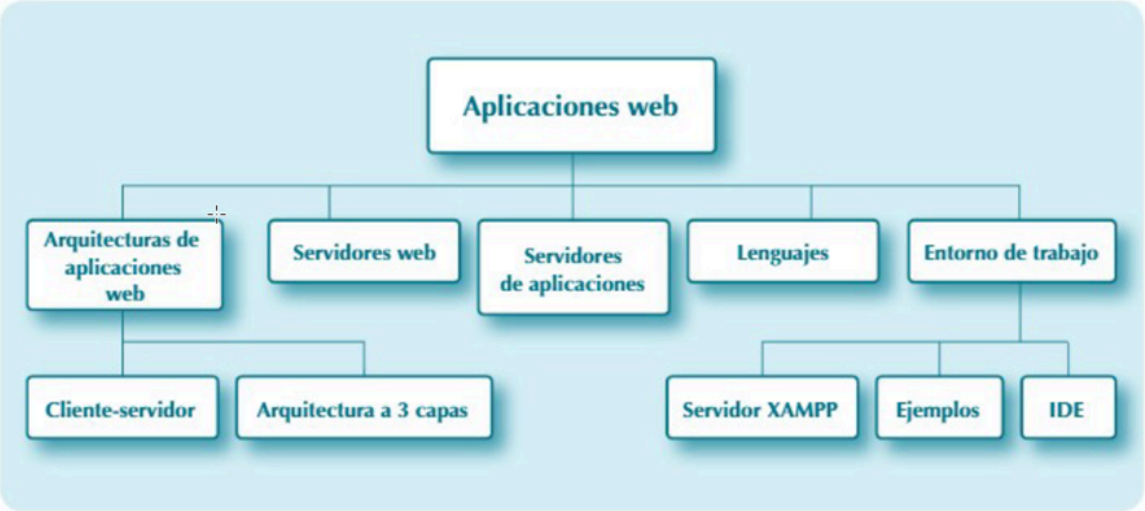
CSS y el diseño web

Con la necesidad de crear páginas web visualmente atractivas, surgió CSS, un lenguaje que permite a los desarrolladores dar estilo a las páginas web. CSS mejoró la experiencia del usuario al permitir diseños más complejos y estilizados^1^.

El presente y el futuro

Hoy en día, la programación web sigue evolucionando con nuevas tecnologías y frameworks que facilitan el desarrollo de aplicaciones web avanzadas.La programación web ha pasado de ser una herramienta básica para compartir información a una plataforma compleja y dinámica que soporta aplicaciones de alta gama.

Comenzaremos esta unidad repasando algunos conceptos importantes en relación a las aplicaciones Web:



1.1 Características de la programación web.

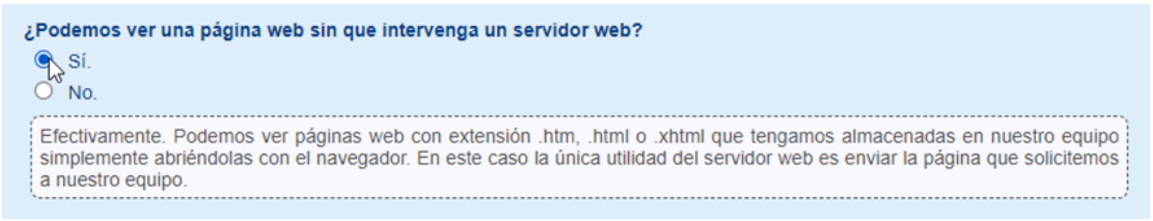
Cuando una página web se descarga a tu ordenador, su contenido define qué se debe mostrar en pantalla. Este contenido está programado en un lenguaje de marcado, formado por etiquetas, que puede ser **HTML** o XHTML. Las etiquetas que componen la página indican el objetivo de cada una de las partes que la componen. Así, dentro de estos lenguajes hay etiquetas para indicar que un texto es un **encabezado**, que forma parte de una **tabla**, o que simplemente es un **párrafo** de texto. Además, si la página está bien estructurada, la información que le indica al navegador el **estilo** con que se debe mostrar cada parte de la página estará almacenado en otro fichero, una hoja de estilos o **CSS**. La hoja de estilos se encuentra indicada en la página web y el navegador la descarga junto a ésta. En ella nos podemos encontrar, por ejemplo, estilos que indican que el encabezado debe ir con tipo de letra Arial y en color rojo, o que los párrafos deben ir alineados a la izquierda. Estos dos ficheros se descargan a tu ordenador desde un servidor web como respuesta a una petición.Los pasos son los siguientes:

Tu ordenador solicita a un servidor web una página con extensión .htm, .html o .xhtml.

El servidor busca esa página en un almacén de páginas (cada una suele ser un fichero).

Si el servidor encuentra esa página, la recupera.

Y por último se la envía al **navegador** para que éste pueda mostrar su contenido. Este es un ejemplo típico de una comunicación **cliente-servidor**. El cliente es el que hace la petición e **inicia** la comunicación, y el servidor es el que recibe la petición y la atiende. En nuestro caso, el **navegador** es el cliente web



W3C

El World Wide Web Consortium **(W3C)** es una organización internacional que desarrolla estándares para la web. Aquí tienes algunas de sus principales utilidades:

1. **Establecimiento de estándares** : El W3C crea y mantiene estándares web como HTML, CSS y XML, asegurando que los desarrolladores tengan un conjunto común de reglas para construir sitios y aplicaciones web.
2. **Interoperabilidad** : Al seguir los estándares del W3C, los desarrolladores pueden crear sitios web que funcionen de manera consistente en diferentes navegadores y dispositivos, mejorando la experiencia del usuario.
3. **Accesibilidad** : El W3C promueve la accesibilidad web a través de iniciativas como las Pautas de Accesibilidad para el Contenido Web (WCAG), ayudando a que las páginas web sean accesibles para personas con discapacidades.
4. **Innovación** : Al desarrollar nuevas tecnologías y estándares, el W3C impulsa la innovación en la web, permitiendo la creación de aplicaciones más avanzadas y funcionales.
5. **Seguridad** : El W3C trabaja en estándares de seguridad para proteger la información y la privacidad de los usuarios en la web.

En resumen, el W3C juega un papel crucial en el desarrollo y la evolución de la web, asegurando que sea accesible, segura y funcional para todos.

Actividad 1

!!! success "Actividad 1"

```
Busca en la web W3C algunos ejemplos de por qué su uso
```

2 Arquitecturas Web

Las arquitecturas web son fundamentales para el diseño y desarrollo de sitios y aplicaciones en línea, ya que determinan cómo se estructuran y organizan.

¿Qué es una Arquitectura Web?

Una **arquitectura web** se refiere a la estructura general y los componentes que interactúan en un sistema web. Esta incluye el diseño de la infraestructura, la comunicación entre servidores y clientes, y la manera en que se gestionan los datos y la lógica de la aplicación.

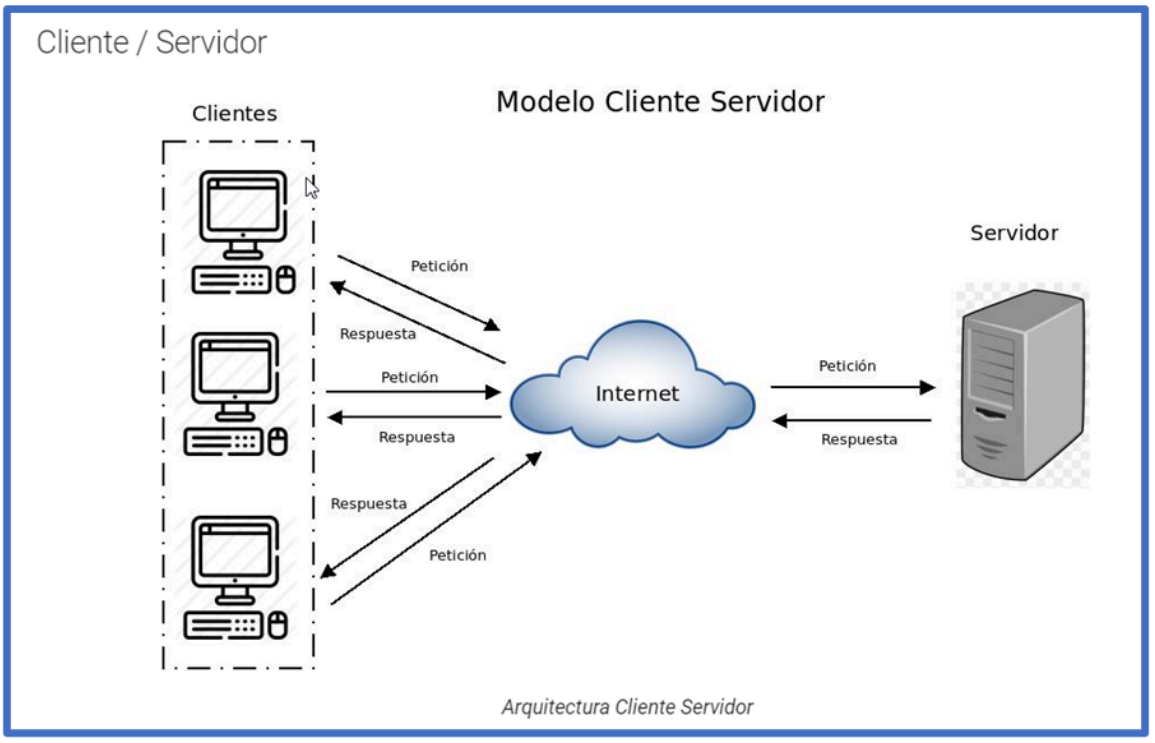
Las arquitecturas web más comunes son:

- **Monolítica**: En esta arquitectura, todos los componentes de la aplicación están integrados en un único código base. Es simple de desarrollar pero puede ser difícil de escalar y mantener.
- **Cliente-Servidor**: El cliente (generalmente un navegador web) interactúa con el servidor que procesa las solicitudes y devuelve respuestas. Es un modelo clásico en la web.
- **Arquitectura de Tres Capas (3-tier)**: Se divide en capa de presentación (front-end), capa lógica (back-end), y capa de datos (base de datos). Esto separa la lógica de negocio y la presentación, facilitando el mantenimiento y la escalabilidad.
- **Microservicios**: Cada función de la aplicación se desarrolla como un servicio independiente que puede escalar y desplegarse por separado. Es popular en sistemas modernos por su flexibilidad y escalabilidad.

2.1 Modelo Cliente Servidor

Una de las arquitecturas de software más utilizadas en informática es la llamada **arquitectura cliente / servidor** . En este tipo de arquitectura hay una aplicación que hace peticiones (el **cliente**) y otra aplicación que está a la espera de recibir esas peticiones y, cuando las recibe, procesarlas y responder (el **servidor**).

Ejemplos de esta arquitectura son los servidores **NTP** (Network Time Protocol), a los cuales se le pide la hora actual para sincronizarse, los servidores **FTP** (File Transfer Protocol), a los que se les puede enviar y pedir archivos, o precisamente los servidores **HTTP** , que son los encargados de servir las páginas web.



Explicación modelo cliente servidor

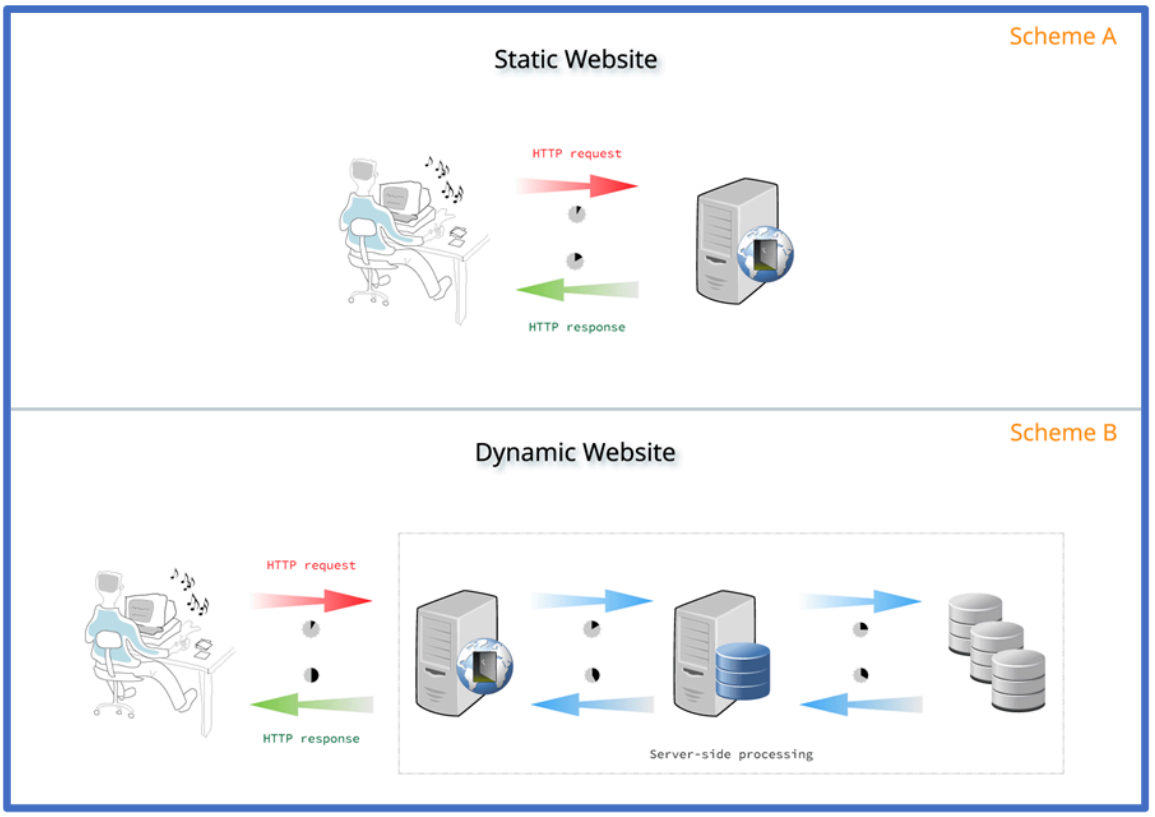
2.2 Páginas Web Estáticas / Dinámicas

Las páginas que viste en el ejemplo anterior se llaman **páginas web estáticas** . Estas páginas se encuentran almacenadas en su **forma definitiva** , tal y como se crearon, y su contenido no varía. Son útiles para mostrar una información concreta, y mostrarán esa misma información cada vez que se carguen. La única forma en que pueden cambiar es si un programador la modifica y actualiza su contenido.

En contraposición a las páginas web estáticas, como ya te imaginarás, existen las **páginas web dinámicas** . Estas páginas, como su nombre indica, se caracterizan porque su contenido cambia en función de diversas variables, como puede ser el navegador que estás usando, el usuario con el que te has identificado, o las acciones que has efectuado con anterioridad.

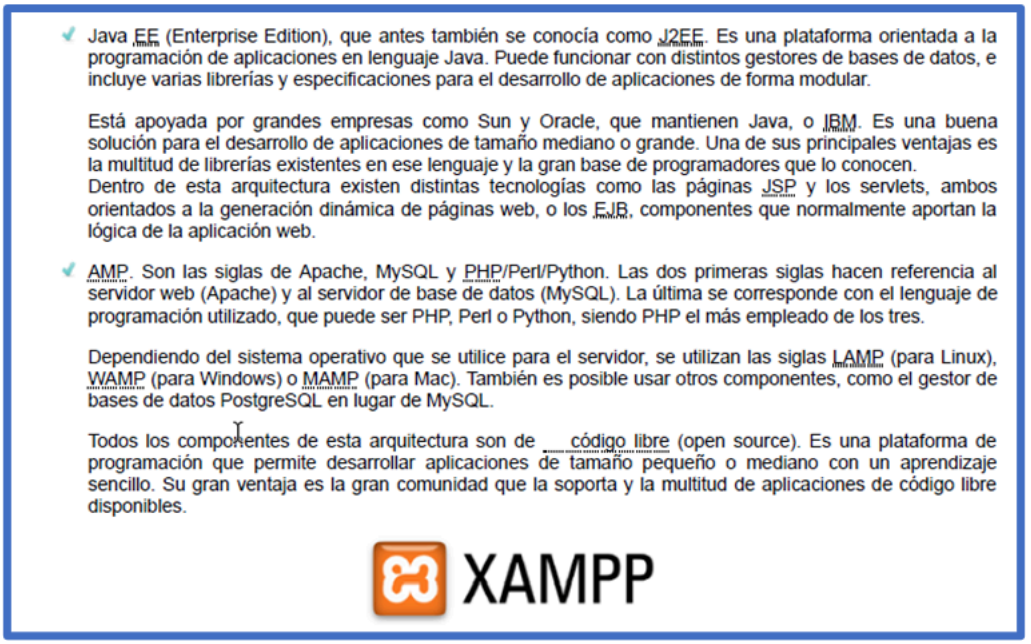
Dentro de las **páginas web dinámicas** , es muy importante distinguir **dos tipos** :

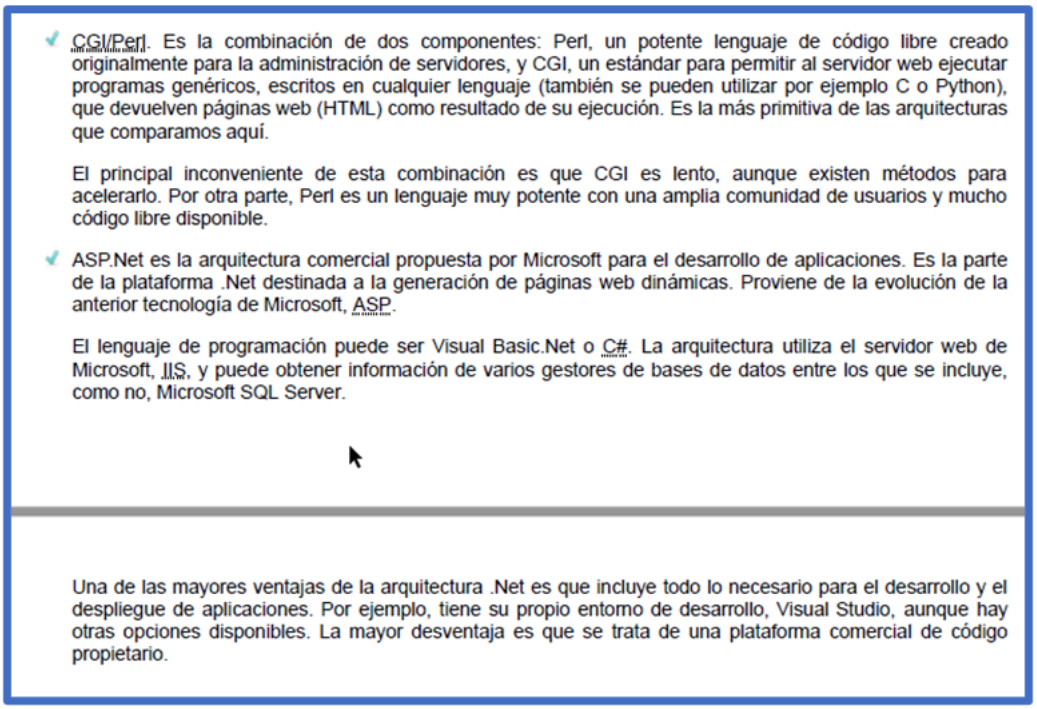
- Aquellas que **incluyen código que ejecuta el navegador**. En estas páginas el código ejecutable, normalmente en **lenguaje JavaScript** , se incluye dentro del HTML (o XHTML) y se descarga junto con la página. Cuando el navegador muestra la página en pantalla, ejecuta el código que la acompaña. Este código puede incorporar múltiples **funcionalidades** que pueden ir desde mostrar animaciones hasta cambiar totalmente la apariencia y el contenido de la página. En este módulo no vamos a ver JavaScript, salvo cuando éste se relaciona con la programación web del lado del servidor.
- Como ya sabes, hay muchas páginas en Internet que **no tienen extensión .htm, .html o .xhtml** . Muchas de estas páginas tienen extensiones como .php, .asp, .jsp, .cgi o .aspx. En éstas, el contenido que se descarga al navegador es similar al de una página web estática: HTML (o XHTML). Lo que cambia es la forma en que se obtiene ese contenido. Al contrario de lo que vimos hasta ahora, esas páginas no están almacenadas en el servidor; más concretamente, el contenido que se almacena no es el mismo que después se envía al navegador. **El HTML de estas páginas se forma como resultado de la ejecución de un programa** , y esa ejecución tiene lugar en el servidor web (aunque no necesariamente por ese mismo servidor).



2.3 Arquitecturas y lenguajes

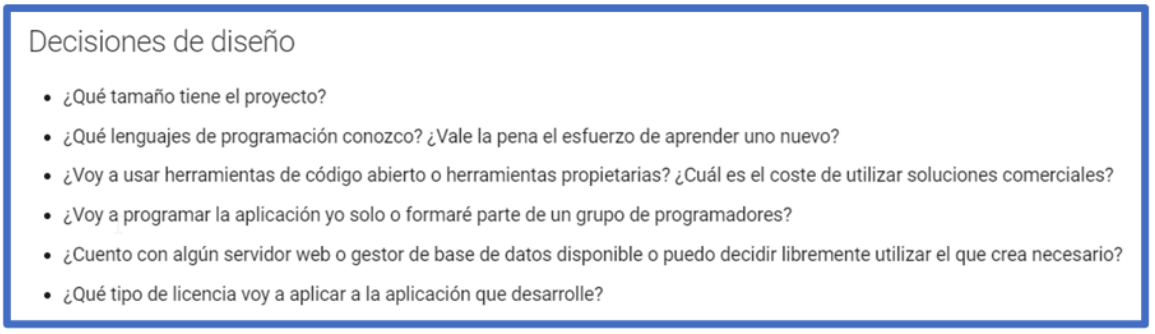
Una de las primeras decisiones que tendrás que tomar al programar una aplicación web es la **arquitectura **que vas a utilizar y la que se adecúa más a tu proyecto por distintas razones. Tenemos un sinfín de alternativas, podemos nombrar algunas de ellas como son:





2.4 Decisiones de diseño

Para tomar la decisión del punto anterior, es muy importante tener en cuenta diferentes aspectos en cuanto a decisiones de diseño, como pueden ser:



3 Arquitectura de 3 capas y modelo MVC

3.1 Ejecución de código en el Servidor y en el Cliente

Como vimos, cuando tu navegador solicita a un servidor web una página, es posible que antes de enviártela haya tenido que ejecutar, por sí mismo o por delegación, algún programa para obtenerla. Ese programa es el que genera, en parte o en su totalidad, la página web que llega a tu equipo. En estos casos, **el código se ejecuta en el entorno del servidor web** .

Además, cuando una página web llega a tu navegador, es también posible que incluya algún programa o fragmentos de código que se deban ejecutar. Ese código, normalmente en lenguaje **JavaScript**, se ejecutará en tu navegador y, además de poder modificar el contenido de la página, también puede llevar a cabo acciones como la animación de textos u objetos de la página o la comprobación de los datos que introduces en un formulario.

Estas dos tecnologías se complementan una con otra. Así, volviendo al ejemplo del correo web:

- el programa que se encarga de obtener tus mensajes y su contenido de una base de datos se ejecuta en el entorno del **servidor** ,
- mientras que tu **navegador** ejecuta, por ejemplo, el código encargado de avisarte cuando quieres enviar un mensaje y te has olvidado de poner un texto en el asunto.

- desde hace unos años existe una técnica de desarrollo web conocida como **AJAX** , que nos posibilita realizar programas en los que el código JavaScript que se ejecuta en el navegador pueda comunicarse con un servidor de Internet para obtener información con la que, por ejemplo, modificar la página web actual.

Actividad 2

!!! note "Actividad 2"

```
Añade a tu repositorio de seguimiento algún **enlace de Youtube** que explique este modelo y haz un
**comentario con tus palabras** al respecto
```

3.2 Arquitectura de 3 capas

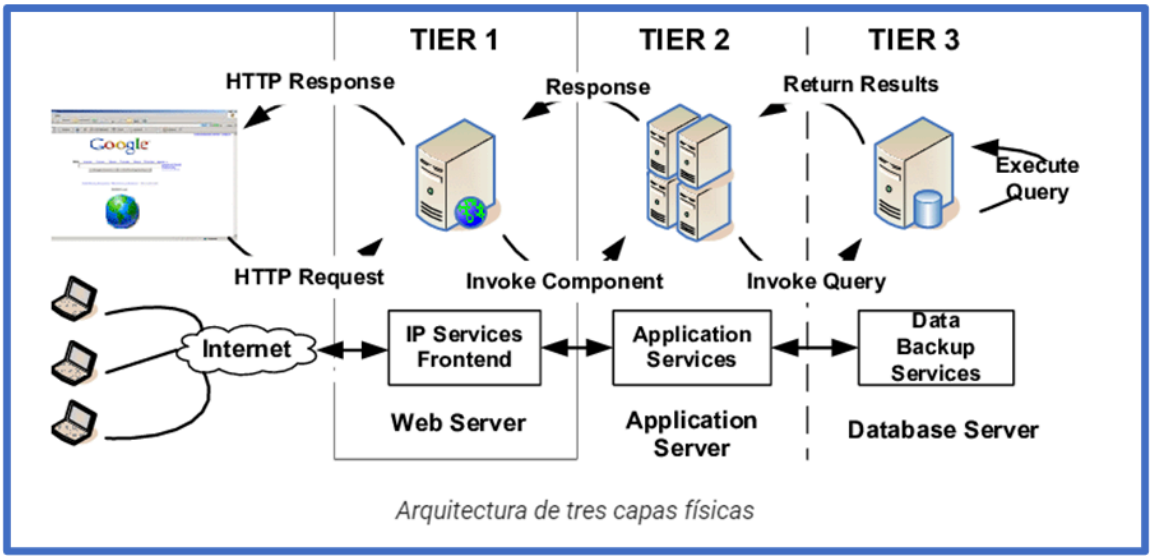
Antes de hablar de cada una de estas capas tenemos que distinguir entre capas **físicas** (*tier*) y capas **lógicas** (*layer*).

Tier (nivel/escalón)

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 bases de datos**



Arquitectura de tres capas físicas

3.3 Cluster en tiers del modelo 3 capas

No debemos 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.

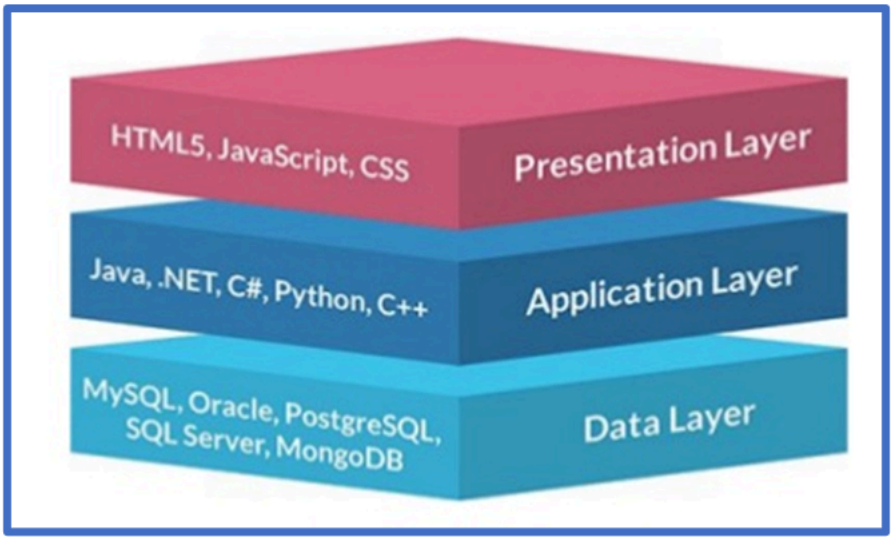
En una aplicación puedes distinguir, de forma general, **funciones de presentación ** (se encarga de dar formato a los datos para presentárselo al usuario final), **lógica ** (utiliza los datos para ejecutar un proceso y obtener un resultado), **persistencia ** (que mantiene los datos almacenados de forma organizada) y **acceso ** (que obtiene e introduce datos en el espacio de almacenamiento).

Cada capa puede ocuparse de una o varias de las funciones anteriores. Por ejemplo, en las aplicaciones de **3capas ** nos podemos encontrar con:

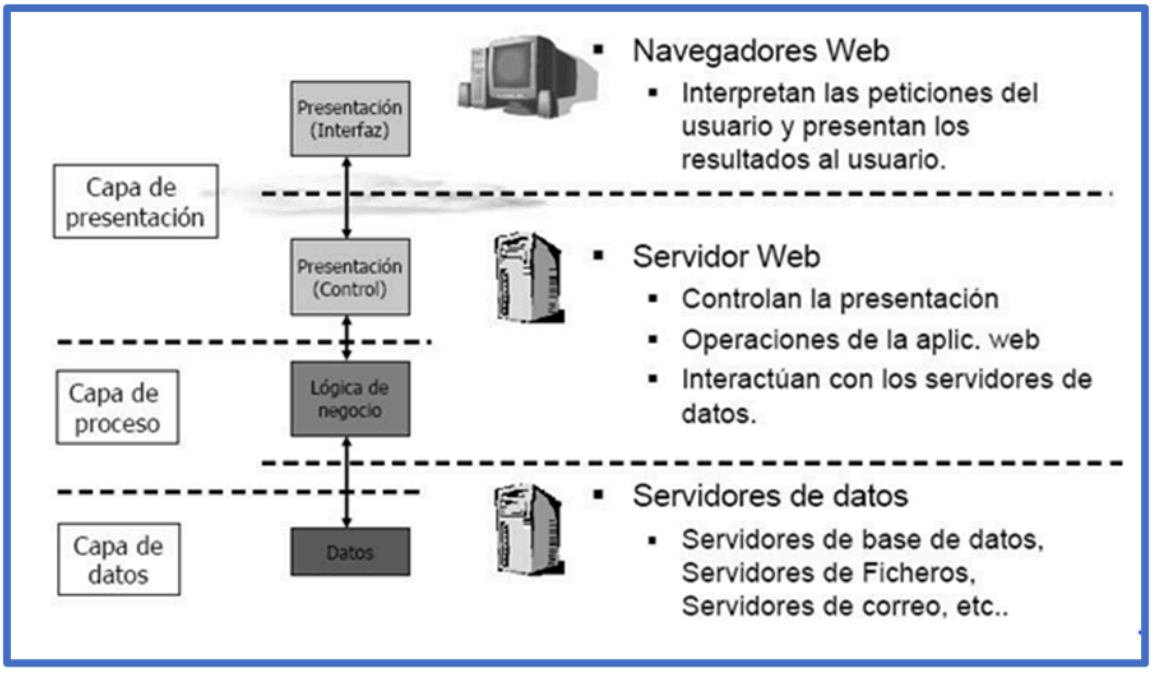
- Una **capa cliente** , que es donde programarás todo lo relacionado con la interface de usuario, esto es, la parte visible de la aplicación con la que interactuará el usuario.
- Una **capa intermedia** donde deberás programar la funcionalidad de tu aplicación.
- Una **capa de acceso a datos** , que se tendrá que encargar de almacenar la información de la aplicación en una base de datos y recuperarla cuando sea necesario.

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

1. **Capa de Presentación** (Front-End): Desarrollada con HTML, CSS, y JavaScript.
2. **Capa Lógica** (Back-End): Implementada en lenguajes como Python o Java.
3. **Capa de Datos**: Uso de bases de datos como MySQL o MongoDB.



Así, en la siguiente figura también se representan los aspectos generales de estas tres capas.



Actividad 3 Capas de presentación, aplicación y datos

Busca alguna imagen en internet de este **modelo de 3 capas** y coméntala brevemente en repositorio, debatiremos en clase las capturas que hemos encontrado.

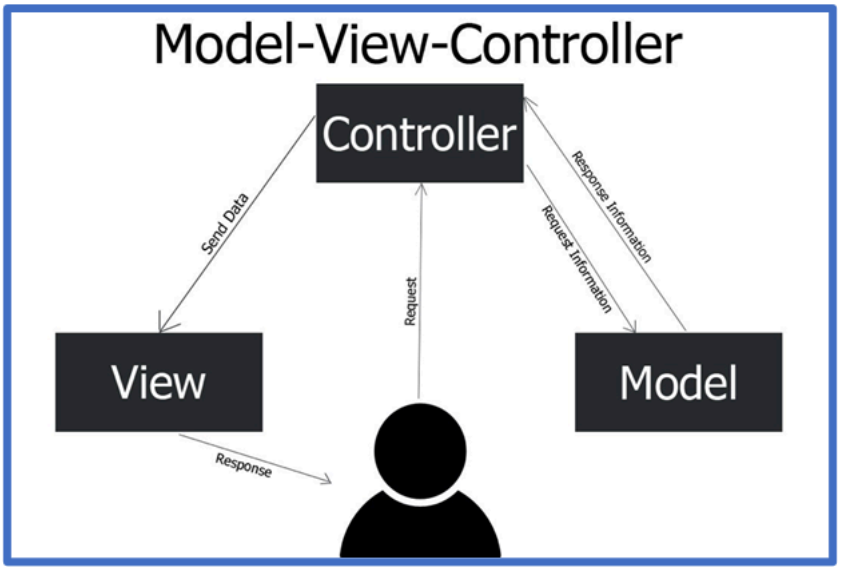
3.4 MVC: Modelo Vista Controlador

Model-View-Controller o Modelo-Vista-** C**ontrolador** 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.

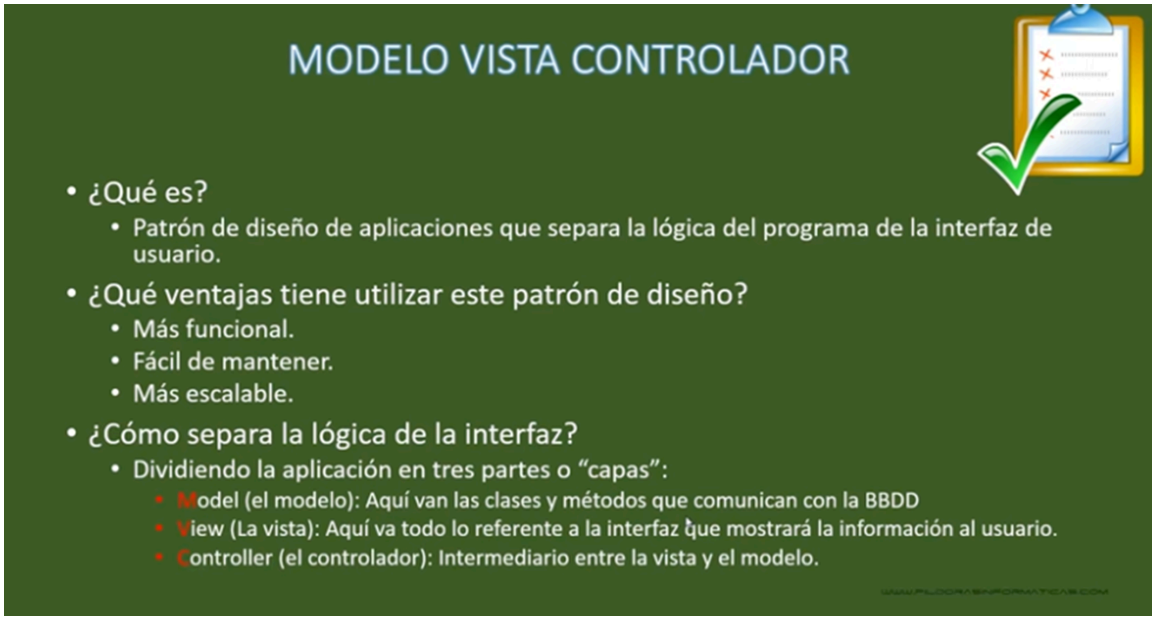
Al separar los componentes en elementos conceptuales permite reutilizar el código y mejorar su organización y mantenimiento. Sus elementos son:

- **Modelo** : representa la información y gestiona todos los accesos a ésta, tanto consultas como actualizaciones provenientes, normalmente, de una base de datos. Se accede via 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**



Características



4 Lenguajes de Programación Más Usados en el Desarrollo Web

Front-End

- **HTML**: Lenguaje de marcado que estructura el contenido de la web.
- **CSS**: Lenguaje que define el estilo y la apariencia de la web.

JavaScript

JavaScript es el lenguaje del lado del cliente más popular. Se ejecuta en el navegador del usuario y se utiliza para crear interacciones dinámicas en la página web. También puede usarse en el backend con tecnologías como Node.js.

Ejemplo Básico en JavaScript

```
document.getElementById("mensaje").innerHTML = "¡Hola desde JavaScript!";
```

Este código busca un elemento con el ID `mensaje` y cambia su contenido a "¡Hola desde JavaScript!".

Back-End

PHP

PHP es un lenguaje de scripting del lado del servidor ampliamente utilizado para el desarrollo web. Fue creado en 1994 y es conocido por su facilidad de uso y su capacidad para generar páginas web dinámicas.

Ejemplo Básico en PHP

```
<?php
echo "Hola, mundo!";
?>
```

Cuando un servidor web ejecuta este código, enviará al cliente un simple texto "Hola, mundo!".

Python

Python es un lenguaje de propósito general que también se utiliza en el desarrollo web, especialmente en el backend con frameworks como Django o Flask.

Ejemplo Básico en Python (con Flask)

```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "¡Hola desde Flask!"

if __name__ == "__main__":
    app.run()
```

Este código crea una pequeña aplicación web que devuelve "¡Hola desde Flask!" cuando se accede a la ruta raíz (`/`).

Java:

Utilizado en grandes aplicaciones empresariales, a menudo con el framework Spring.

Referencias

- https://escuelainformatica.es/blog/historia-de-la-programacion-web
- https://www.arquitecturajava.com/arquitecturas-web-y-su-evolucion/
- "The Architecture of Open Source Applications" - aosabook.org
- "JavaScript Info" - javascript.info
- Client-server model - Wikipedia
- PHP Documentation