DESARROLLO WEB ENTORNO SERVIDOR

Plataformas de programación web en entorno servidor

1.- Características de la programación web.

- 1.1.- Páginas web estáticas y dinámicas.
- 1.2.- Ejecución de código en el servidor y en el cliente.

2.- Tecnologías para programación web del lado del servidor.

- 2.1.- Arquitecturas y plataformas. Selección de la arquitectura de programación.
- 2.2.- Integración con el servidor web.

3.- Lenguajes y plataformas.

- 3.1.- Código embebido en el lenguaje de marcas.
- 3.2.- Lenguajes de programación.
- 3.3.- Herramientas de programación.
- 3.4.- Instalación de Visual Studio Code.
- 3.5.- Instalación de una plataforma XAMPP.
- 3.6.- Integración PHP en Visual Studio Code.

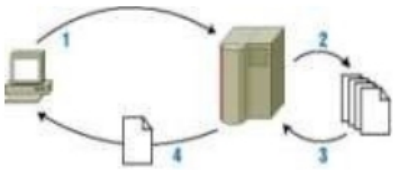
1.- Características de la programación web

Seguro que **ya sabes** exactamente qué es una **página web**, e incluso conozcas cuáles son los pasos que se suceden para que, cuando visitas una web poniendo su dirección en el navegador, la página se descargue a tu equipo y se pueda mostrar. Sin embargo, este procedimiento que puede parecer sencillo, a veces no lo es tanto. Todo depende de cómo se haya hecho esa página.

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** (Abreviatura de "Hoja de estilos en cascada", del inglés Cascading Style Sheet (CSS). Es un lenguaje utilizado para definir las características de presentación de un documento escrito en lenguaje HTML, XHTML o XML). 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. El proceso es el que se refleja en la siguiente figura.



Los pasos son los siguientes:

1. **Tu ordenador solicita a un servidor web una página** con extensión .htm, .html o .xhtml.
2. **El servidor busca esa página** en un almacén de páginas (cada una suele ser un fichero).
3. Si **el servidor encuentra esa página**, la recupera.
4. 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.

1.1.- Páginas web estáticas y dinámicas

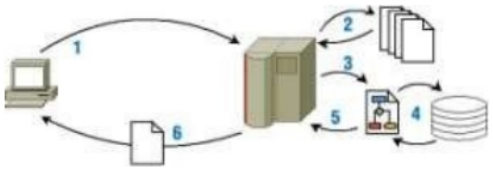
Las páginas que viste en el ejemplo anterior se llaman **páginas web estáticas**. Estas páginas se encuentran almacenadas en su forma definitiva, tal y como se crearon, y su contenido no varía. Son útiles para mostrar una información concreta, y mostrarán esa misma información cada vez que se carguen. 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).

El esquema de funcionamiento de una página web dinámica es el siguiente:

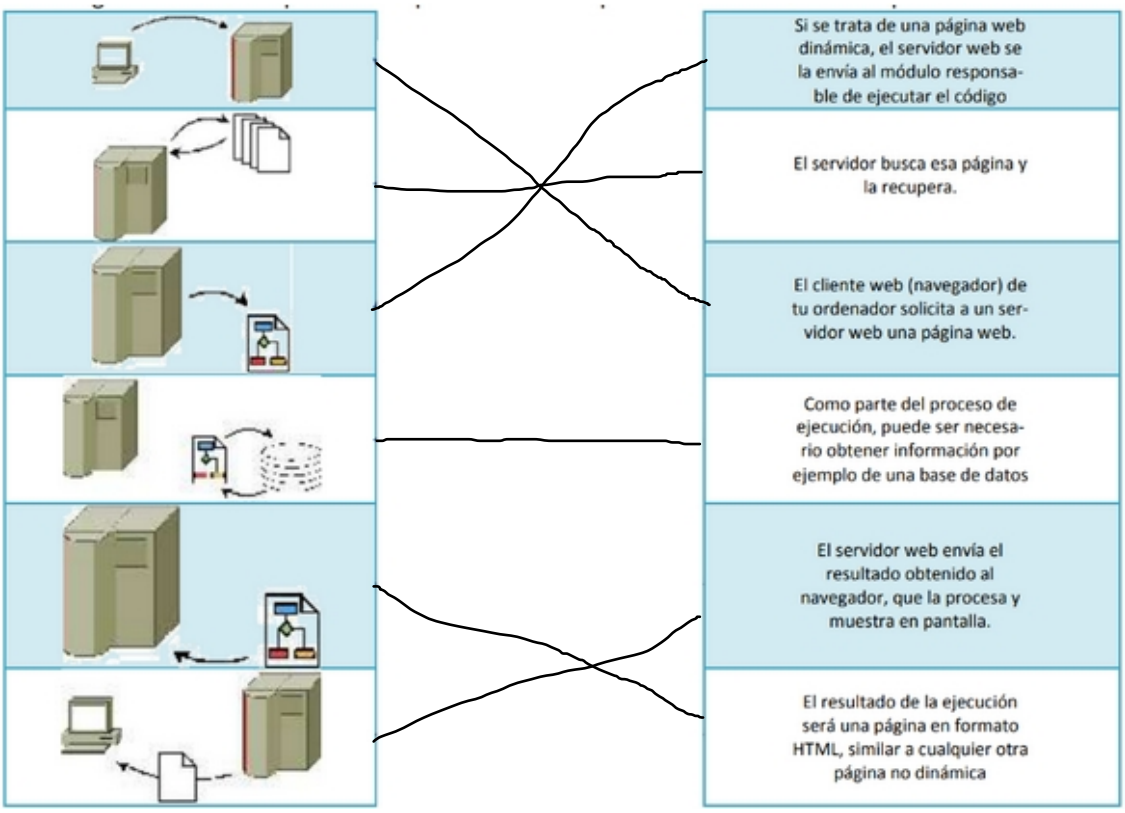


1. **El cliente web** (navegador) de tu ordenador **solicita** a un servidor web una **página web.**
2. El **servidor busca** esa página y la recupera.
3. En el caso de que se trate de una **página web dinámica**, es decir, que su contenido deba ejecutarse para obtener el HTML que se devolverá, el **servidor web** contacta con el módulo responsable de **ejecutar el código** y se lo envía.
4. Como parte del **proceso de ejecución**, puede ser necesario obtener información de algún repositorio (Cualquier almacén de información digital, normalmente una base de datos), como por ejemplo consultar registros almacenados en una base de datos.
5. **El resultado** de la ejecución será una página en **formato HTML**, similar a cualquier otra página web no dinámica.
6. El **servidor web envía el resultado** obtenido al navegador, que la procesa y muestra en pantalla.

Este procedimiento tiene lugar constantemente mientras consultamos páginas web. Por ejemplo, cuando consultas tu correo en Gmail, Hotmail, Yahoo o cualquier otro servicio de correo vía web, lo primero que tienes que hacer es introducir tu nombre de usuario y contraseña. A continuación, lo más habitual es que el servidor te muestre una pantalla con la bandeja de entrada, en la que aparecen los mensajes recibidos en tu cuenta. Esta pantalla es un claro ejemplo de una página web dinámica.

Obviamente, el navegador no envía esa misma página a todos los usuarios, sino que la **genera de forma dinámica** en función de quién sea el usuario que se conecte. Para generarla ejecuta un programa que obtiene los datos de tu usuario (tus contactos, la lista de mensajes recibidos) y con ellos compone la página web que recibes desde el servidor web.

En la siguiente actividad puedes comprobar si te han quedado claros estos conceptos:



Aunque la utilización de páginas web dinámicas te parezca la mejor opción para construir un sitio web, no siempre lo es. Sin lugar a dudas, es la que más potencia y flexibilidad permite, pero las páginas **web estáticas** tienen también **algunas ventajas:**

- **No es necesario saber programar** para crear un sitio que utilice únicamente páginas web estáticas. Simplemente habría que conocer HTML/XHTML y CSS, e incluso esto no sería indispensable: se podría utilizar algún programa de diseño web para generarlas.
- La característica diferenciadora de las páginas web estáticas es que **su contenido nunca varía**, y esto en algunos casos también **puede suponer una ventaja**. Sucede, por ejemplo, cuando quieres almacenar un enlace a un contenido concreto del sitio web: si la página es dinámica, al volver a visitarla utilizando el enlace su contenido puede variar con respecto a cómo estaba con anterioridad. O cuando quieres dar de alta un sitio que has creado en un motor de búsqueda como Google.

Para que Google muestre un sitio web en sus resultados de búsqueda, previamente tiene que indexar su contenido. Es decir, un programa recorre las páginas del sitio consultando su contenido y clasificándolo. Si las páginas se generan de forma dinámica, puede ser que su contenido, en parte o por completo, no sea visible para el buscador y por tanto no quedará indexado. Esto nunca sucedería en un sitio que utilizase páginas web estáticas.

Como ya sabes, para que un servidor web pueda procesar una página web dinámica, necesita **ejecutar un programa**. Esta ejecución la realiza un **módulo concreto**, que puede estar integrado en el servidor o ser independiente.

Además, puede ser necesario **consultar una base de datos** como parte de la ejecución del programa. Es decir, la ejecución de una página web dinámica requiere una serie de recursos del lado del servidor.

Estos recursos deben instalarse y mantenerse.

Las páginas web estáticas sólo necesitan un servidor web que se comunique con tu navegador para enviártela. Y de hecho para ver una página estática almacenada en tu equipo no necesitas siquiera de un servidor web. Son archivos que pueden almacenarse en un soporte de almacenamiento como puede ser un disco óptico o una memoria USB y abrirse desde él directamente con un navegador web.

Pero si decides hacer un sitio web utilizando **páginas estáticas**, ten en cuenta que **tienen limitaciones**. La desventaja más importante ya la comentamos anteriormente: la **actualización** de su contenido debe hacerse de **forma manual** editando la página que almacena el servidor web. Esto implica un mantenimiento que puede ser prohibitivo en sitios web con gran cantidad de contenido.

Las **primeras páginas web** que se crearon en Internet fueron páginas **estáticas**. A esta web compuesta por páginas estáticas se le considera la primera generación. La segunda generación de la web surgió gracias a las páginas web dinámicas.

Centrémonos ahora en las **Aplicaciones web.** Las **aplicaciones web** emplean **páginas web dinámicas** para crear aplicaciones que se **ejecuten en un servidor web** y se muestren en un navegador. Puedes encontrar aplicaciones web para realizar múltiples tareas. Unas de las primeras en aparecer fueron los clientes de correo, que te permiten consultar los mensajes de correo recibidos y enviar los tuyos propios utilizando un navegador.

Hoy en día existen aplicaciones web para multitud de tareas como procesadores de texto, gestión de tareas, o edición y almacenamiento de imágenes. Estas aplicaciones tienen ciertas ventajas e inconvenientes si las comparas con las "***aplicaciones tradicionales***" que se ejecutan sobre el sistema operativo de la propia máquina. Veamos ventajas e inconvenientes.

Ventajas de las aplicaciones web:

- **No es necesario instalarlas en aquellos equipos en que se vayan a utilizar**. Se instalan y se ejecutan solamente en un equipo, en el servidor, y esto es suficiente para que se puedan utilizar de forma simultánea desde muchos equipos.
- Como solo se encuentran instaladas en un equipo, **es muy sencillo gestionarlas** (hacer copias de seguridad de sus datos, corregir errores, actualizarlas).
- **Se pueden utilizar en todos aquellos sistemas que dispongan de un navegador web**, independientemente de sus características (no es necesario un equipo potente) o de su sistema operativo.
- **Se pueden utilizar desde cualquier lugar en el que dispongamos de conexión con el servidor**. En muchos casos esto hace posible que se pueda acceder a las aplicaciones desde sistemas no convencionales, como por ejemplo teléfonos móviles.

Inconvenientes de las aplicaciones web:

- **El interface de usuario de las aplicaciones web es la página que se muestra en el navegador**. Esto **restringe** las características del interface a aquellas de una página web.
- **Dependemos de una conexión con el servidor para poder utilizarlas**. Si nos falla la conexión, no podremos acceder a la aplicación web.
- **La información que se muestra en el navegador debe transmitirse desde el servidor**. Esto hace que cierto tipo de aplicaciones no sean adecuadas para su implementación como aplicación web (por ejemplo, las aplicaciones que manejan contenido multimedia, como las de edición de vídeo).

Hoy en día muchas aplicaciones web utilizan las ventajas que les ofrece la generación de páginas dinámicas. La gran mayoría de su contenido está almacenado en una **base de datos**. Aplicaciones como **Drupal**, **Joomla!** y otras muchas ofrecen dos partes bien diferenciadas:

- **Una parte externa o front-end**, que es el conjunto de páginas que ven la gran mayoría de **usuarios** que las usan (usuarios externos).
- **Una parte interna o back-end**, que es otro conjunto de páginas dinámicas que utilizan las personas que **producen el contenido** y las que **administran** la aplicación web (usuarios internos) para crear contenido, organizarlo, decidir la apariencia externa, etc.

1.2.- 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 (cliente)** 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.

Esta división es así porque el **código que se ejecuta en el cliente** web (en tu navegador) no tiene, o mejor dicho **tradicionalmente no tenía**, **acceso a los datos** que se almacenan en el **servidor**.

Sin embargo, 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.

En este módulo vas a aprender a crear aplicaciones web que se ejecuten en el lado del servidor. El módulo Desarrollo Web en Entorno Cliente, enseña las características de la programación de código que se ejecute en el navegador web.

Muchas de las **aplicaciones web actuales utilizan estas dos tecnologías**: la ejecución de **código en el servidor y en el cliente**. Así, el código que se ejecuta en el servidor genera páginas web que ya incluyen código destinado a su ejecución en el navegador.

2.- Tecnologías para programación web del lado del servidor

Cuando programas una **aplicación**, utilizas un **lenguaje de programación**. Por ejemplo, utilizas el lenguaje Java para crear aplicaciones que se ejecuten en distintos sistemas operativos. Al programar cada aplicación utilizas ciertas herramientas como un entorno de desarrollo o librerías de código. Además, una vez acabado su desarrollo, esa aplicación necesitará ciertos componentes para su ejecución, como por ejemplo una máquina virtual de Java.

En este apartado **vas a aprender las distintas tecnologías** que se pueden utilizar para **programar aplicaciones** que se ejecuten en un **servidor web**, y cómo se relacionan unas con otras. Verás las ventajas e inconvenientes de utilizar cada una, y qué lenguajes de programación deberás aprender para utilizarlas.

Componentes principales con los que debes contar para ejecutar aplicaciones web en un servidor son:

- Un **servidor web** para recibir las peticiones de los clientes web (normalmente navegadores) y enviarles la página que solicitan (una vez generada puesto que hablamos de páginas web dinámicas). El servidor web debe conocer el procedimiento a seguir para generar la página web: qué módulo se encargará de la ejecución del código y cómo se debe comunicar con él.
- El **módulo encargado de ejecutar el código** o programa y generar la página web resultante. Este módulo debe integrarse de alguna forma con el servidor web, y dependerá del lenguaje y tecnología que utilicemos para programar la aplicación web.
- Una **aplicación de base de datos**, que normalmente también será un servidor. Este módulo no es estrictamente necesario pero en la práctica se utiliza en todas las aplicaciones web que utilizan grandes cantidades de datos para almacenarlos.
- El **lenguaje de programación** que utilizarás para desarrollar las aplicaciones.

Además de los componentes a utilizar, también es importante decidir cómo vas a **organizar el código** de la aplicación. Muchas de las arquitecturas que se usan en la programación de aplicaciones web te ayudan a estructurar el código de las aplicaciones en **capas o niveles**.

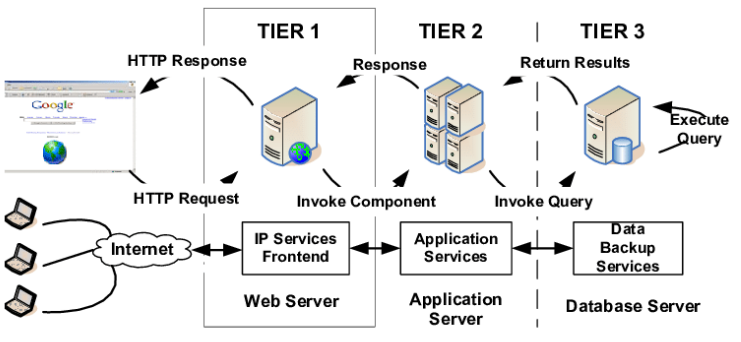
Dividiendo en capas el diseño de una aplicación, se podrán **separar las funciones lógicas/físicas** de la misma, de tal forma que sea posible ejecutar, si es necesario, cada una en un servidor distinto. Por tanto distinguimos entre capas **físicas** (tier) y capas **lógicas** (layer).

Física

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



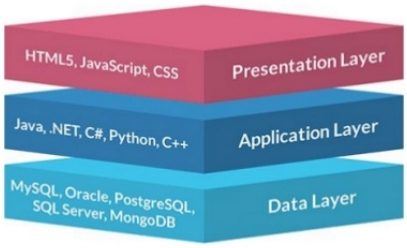
Arquitectura de tres capas físicas

No confundir las capas con la cantidad de servidores.

Lógica

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

- Presentación
- Negocio / Aplicación / Proceso / Lógica de negocio
- Datos / Persistencia

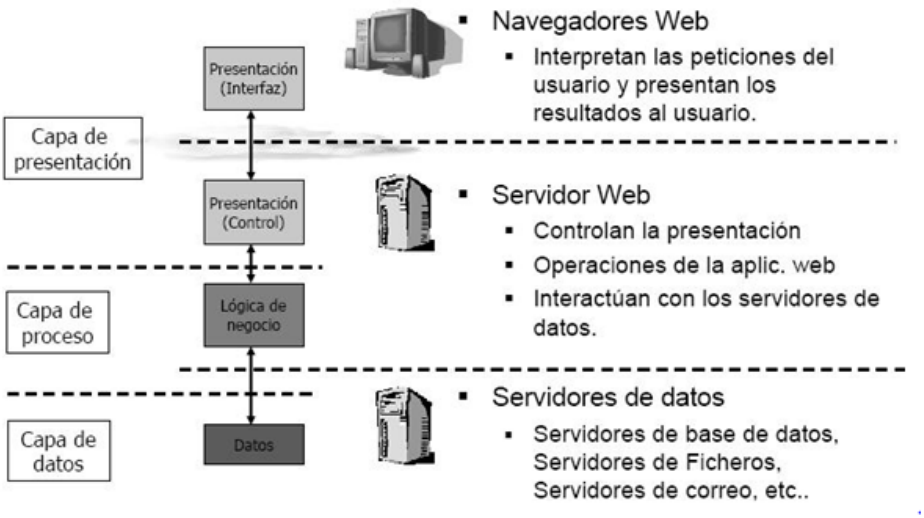


Como se observa, cada una de las capas se puede implementar con diferentes lenguajes de programación y/o herramientas.

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 **3 capas** nos podemos encontrar con:

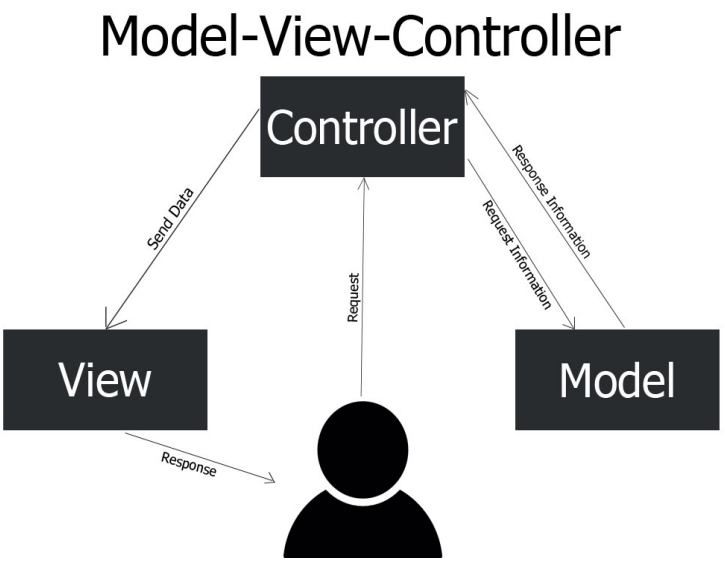
- Una **capa cliente**, que es donde programarás todo lo relacionado con el interface de usuario, esto es, la parte visible de la aplicación con la que interactuará el usuario.
- Una **capa intermedia** donde deberás programar la funcionalidad de tu aplicación.
- Una **capa de acceso a datos**, que se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



Arquitectura de tres capas físicas en tres lógicas

MVC

Model-View-Controller o *Modelo-Vista-Controlador* es un ejemplo de modelo de arquitectura lógica, 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.

2.1.- Arquitecturas y plataformas. Selección de la arquitectura de programación

La primera elección que harás antes de comenzar a programar una aplicación web es la arquitectura que vas a utilizar. Hoy en día, puedes elegir entre:

- **Java EE (Enterprise Edition)**: antes también se conocía como J2EE. Es una plataforma orientada a la programación de aplicaciones en lenguaje Java. Puede funcionar con distintos gestores de bases de datos, e incluye varias librerías y especificaciones para el desarrollo de aplicaciones de forma modular.

  Está apoyada por grandes empresas como Sun y Oracle, que mantienen Java, o IBM. Es una buena solución para el desarrollo de aplicaciones de tamaño mediano o grande. Una de sus principales ventajas es la multitud de librerías existentes en ese lenguaje y la gran base de programadores que lo conocen.

  Dentro de esta arquitectura existen distintas tecnologías como las páginas JSP y los servlets, ambos orientados a la generación dinámica de páginas web, o los EJB (Enterprise JavaBeans), componentes que normalmente aportan la lógica de la aplicación web.

- **AMP:** son las siglas de Apache, MySQL y PHP/Perl/Python. Las dos primeras siglas hacen referencia al servidor web (Apache) y al servidor de base de datos (MySQL). La última se corresponde con el lenguaje de programación utilizado, que puede ser PHP, Perl o Python.

  Dependiendo del sistema operativo que se utilice para el servidor, se utilizan las siglas LAMP (para Linux), WAMP (para Windows) o MAMP (para Mac). También es posible usar otros componentes, como el gestor de bases de datos PostgreSQL en lugar de MySQL.

  Todos los componentes de esta arquitectura son de código libre (open source). Es una plataforma de programación que permite desarrollar aplicaciones de tamaño pequeño o mediano con un aprendizaje sencillo. Su gran ventaja es la gran comunidad que la soporta y la multitud de aplicaciones de código libre disponibles.

  Existen paquetes software que incluyen en una única instalación una plataforma AMP completa. Algunos ni siquiera es necesario instalarlos, e incluso disponen de versiones para distintos sistemas operativos como Linux, Windows o Mac. Uno de los más conocidos es XAMPP.

  https://www.apachefriends.org/es/download.html

- **CGI/Perl:** es la combinación de dos componentes: Perl, un potente lenguaje de código libre creado originalmente para la administración de servidores, y CGI, un estándar para permitir al servidor web ejecutar programas genéricos, escritos en cualquier lenguaje (también se pueden utilizar por ejemplo C o Python), que devuelven páginas web (HTML) como resultado de su ejecución. Es la más primitiva de las arquitecturas que comparamos aquí.

  El principal inconveniente de esta combinación es que CGI es lento, aunque existen métodos para acelerarlo. Por otra parte, Perl es un lenguaje muy potente con una amplia comunidad de usuarios y mucho código libre disponible.

- **ASP.Net:** es la arquitectura comercial propuesta por Microsoft para el desarrollo de aplicaciones. Es la parte de la plataforma .Net destinada a la generación de páginas web dinámicas. Proviene de la evolución de la anterior tecnología de Microsoft, ASP.

  El lenguaje de programación puede ser Visual Basic.Net o C#. La arquitectura utiliza el servidor web de Microsoft, IIS, y puede obtener información de varios gestores de bases de datos entre los que se incluye, como no, Microsoft SQL Server.

  Una de las mayores ventajas de la arquitectura .Net es que incluye todo lo necesario para el desarrollo y el despliegue de aplicaciones. Por ejemplo, tiene su propio entorno de desarrollo, Visual Studio, aunque hay otras opciones disponibles. La mayor desventaja es que se trata de una plataforma comercial de código propietario.

Toma de decisiones: selección de una arquitectura de programación web

Como has visto, hay muchas decisiones que debes tomar antes aún de comenzar el desarrollo de una aplicación web. La arquitectura que utilizarás, el lenguaje de programación, el entorno de desarrollo, el gestor de bases de datos, el servidor web, incluso cómo estructurarás tu aplicación.

Para tomar una decisión correcta, deberás considerar entre otros los siguientes puntos:

- ¿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?

Estudiando las respuestas a éstas y otras preguntas, podrás ver qué arquitecturas se adaptan mejor a tu aplicación y cuáles no son viables.

Cada proyecto tiene sus peculiaridades. Aunque unas arquitecturas son más potentes que otras, no hay una en concreto que podamos considerar mejor que las demás para todos los casos.

2.2.- Integración con el servidor web

Como ya sabes, la comunicación entre un cliente web o navegador y un servidor web se lleva a cabo gracias al protocolo HTTP. ***En el caso de las aplicaciones web, HTTP es el vínculo de unión entre el usuario y la aplicación en sí***. Cualquier introducción de información que realice el usuario se transmite mediante una petición HTTP, y el resultado que obtiene le llega por medio de una respuesta HTTP.

En el **lado del servidor**, estas peticiones son procesadas por el **servidor web** (también llamado *servidor HTTP*). Es por tanto el servidor web el encargado de decidir cómo procesar las peticiones que recibe. Cada una de las arquitecturas que acabamos de ver tiene una forma de integrarse con el servidor web para ejecutar el código de la aplicación. El producto más implantando es Apache Web Server (https://httpd.apache.org/), creado en 1995, software libre y multiplataforma. Utiliza el archivo **.htaccess** para su configuración.

La tecnología CGI es una **importante tecnología** que permite a un cliente, navegador web, solicitar datos de un programa ejecutado en un servidor web. Actualmente aunque también es posible ejecutar código en lenguaje PHP utilizando CGI, los intérpretes PHP no lo suelen utilizar. Existen módulos dinámicos que podemos instalar en el servidor web Apache para que ejecute páginas web dinámicas. El módulo **mod_php** es la forma habitual que se utiliza para ejecutar guiones en **PHP** utilizando plataformas AMP, y su equivalente para el lenguaje Python es **mod_python**.

La arquitectura **Java EE** es más compleja. Para poder ejecutar aplicaciones Java EE en un servidor básicamente tenemos dos opciones: **servidores de aplicaciones**, que implementan todas las tecnologías disponibles en Java EE, y **contenedores de servlets**, que soportan solo parte de la especificación. Dependiendo de la magnitud de nuestra aplicación y de las tecnologías que utilice, tendremos que instalar una solución u otra.

Existen varias implementaciones de ***servidores de aplicaciones Java EE*** certificados. Las dos soluciones comerciales más usadas son IBM Websphere y BEA Weblogic. También existen soluciones de código abierto como JBoss, Geronimo (de la fundación Apache) o *Glassfish*.

Sin embargo, en la mayoría de ocasiones no es necesario utilizar un servidor de aplicaciones completo, especialmente si no utilizamos EJB en nuestras aplicaciones, sino que nos será suficiente un ***contenedor de servlets***. En esta área, destaca *Tomcat*, la implementación por referencia de un contenedor de servlets, que además es de código abierto.

Una vez instalada la solución que hayamos escogido, tenemos que integrarla con el servidor web que utilicemos, de tal forma que reconozca las peticiones destinadas a servlets y páginas JSP y las redirija.

Otra opción es utilizar una única solución para páginas estáticas y páginas dinámicas. Por ejemplo, el contenedor de servlets Tomcat incluye un servidor HTTP propio que puede sustituir a Apache.

La arquitectura **ASP.Net** utiliza el **servidor IIS** de Microsoft, que ya integra soporte en forma de módulos para manejar peticiones de páginas dinámicas ASP y ASP.Net. La utilidad de administración del servidor web incluye funciones de administración de las aplicaciones web instaladas en el mismo.

3.- Lenguajes y plataformas

Una de las diferencias más notables entre un lenguaje de programación web y otro es la manera en que se ejecutan en el servidor web. Describimos aquí dos grupos:

Lenguajes de guiones (interpretados). Son aquellos en los que los programas se ejecutan directamente a partir de su código fuente original. Se almacenan normalmente en un fichero de texto plano. Cuando el servidor web necesita ejecutar código programado en un lenguaje de guiones, le pasa la petición a un intérprete, que procesa las líneas del programa y genera como resultado una página web.

De los lenguajes que estudiaste anteriormente, pertenecen a este grupo Python, PHP, Ruby, ASP (el precursor de ASP.Net).

Lenguajes compilados a código intermedio. Son lenguajes en los que el código fuente original se traduce a un código intermedio, independiente del procesador, antes de ser ejecutado. Es la forma en la que se ejecutan por ejemplo las aplicaciones programadas en Java, y lo que hace que puedan ejecutarse en varias plataformas distintas.

En la programación web, operan de esta forma los lenguajes de las arquitecturas Java EE (servlets y páginas JSP) y ASP.Net.

Cada una de estas formas de ejecución del código por el servidor web tiene sus ventajas e inconvenientes.

- Los lenguajes interpretados tienen la ventaja de que no es necesario traducir el código fuente original para ser ejecutados, lo que aumenta su portabilidad. Si se necesita realizar alguna modificación a un programa, se puede hacer en el momento. Por el contrario el proceso de interpretación ofrece un peor rendimiento que las otras alternativas.
- Los lenguajes compilados a código intermedio ofrecen buen rendimiento y pueden portarse entre distintas plataformas en las que exista una implementación de la arquitectura (como un contenedor de servlets o un servidor de aplicaciones Java EE).

3.1.- Código embebido en el lenguaje de marcas

Cuando la web comenzó a evolucionar desde las páginas web estáticas a las dinámicas, una de las primeras tecnologías que se utilizaron fue la ejecución de código utilizando CGI. Este código se ejecuta y genera como salida el código HTML de una página web. Por tanto, deben contener, mezcladas dentro de su código, sentencias encargadas de generar la página web.

Esta es una de las principales formas de realizar páginas web dinámicas: **integrar las etiquetas HTML en el código de los programas**. Un ejemplo de etiquetas HTML en PHP podría ser:

Para ello se utilizan los **comandos PRINT o ECHO** para imprimir HTML en la página.

```php
<? php

    echo "<html>";
    echo "<title> Etiquetas HTML en PHP </title>";
    echo "<b> Tex to en negrita </b>";
    // El código php aquí
    print "<i> ¡La impresión también funciona! </i>";
    print "<p> Esto es un párrafo para terminar </p>";
    print "</html>"

?>
```

Un enfoque distinto consiste en **integrar el código del programa en medio de las etiquetas HTML de la página web**. De esta forma, el contenido que no varía de la página se puede introducir directamente en HTML, y el lenguaje de programación se utilizará para todo aquello que pueda variar de forma dinámica.

Por ejemplo, puedes incluir dentro de una página HTML un pequeño código en lenguaje PHP que muestre el nombre del servidor:

```php
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Loose//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-loose.dtd">
<html>
<head>
  <title>Prueba PHP</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0" charset="utf-8">
 </head>
<body>
  <p>Pueba de código PHP en HTML </br>
  El servidor es: <?php echo $_SERVER['SERVER_NAME']; ?></p>
</body>
</html>
```

Esta metodología de programación es la que se emplea en los lenguajes ASP, PHP y con las páginas JSP de Java EE.

3.2.- Lenguajes de programación

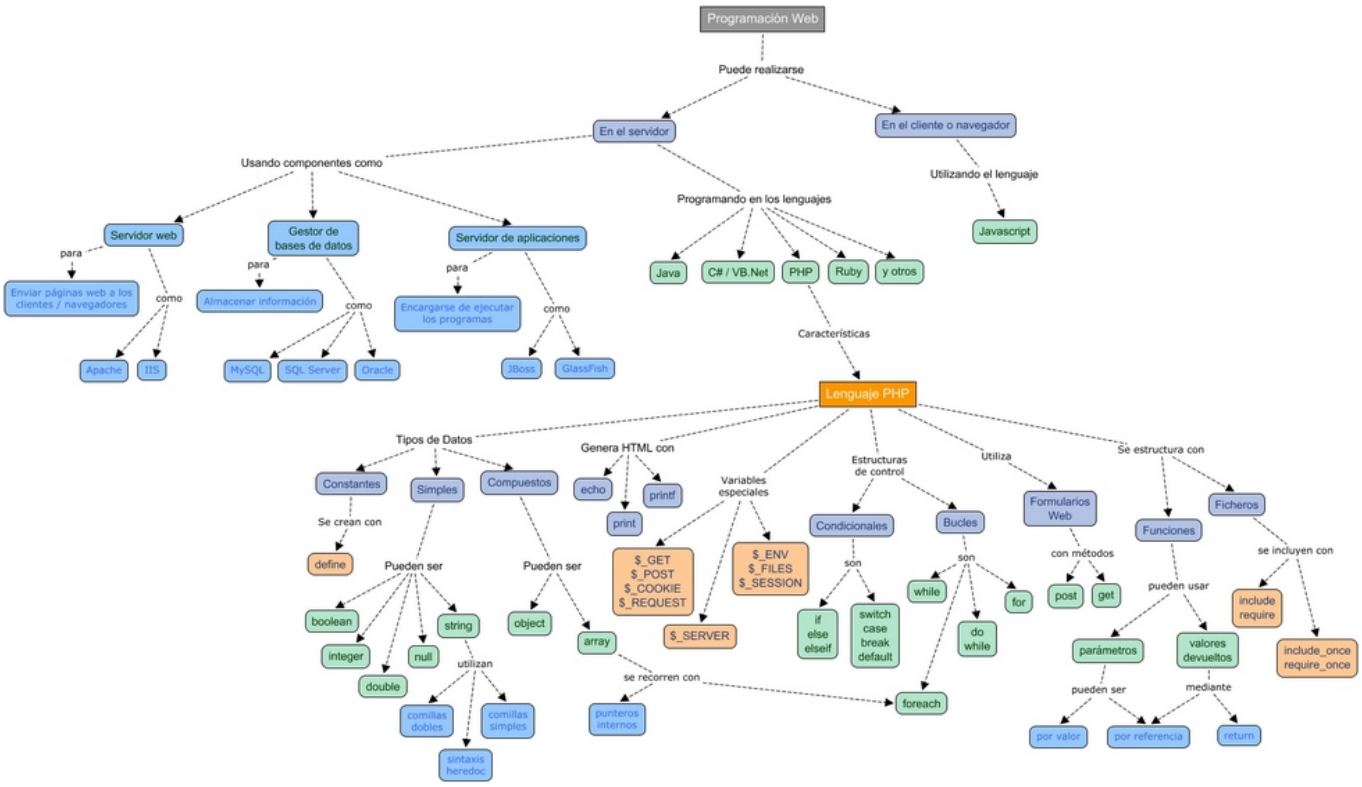
PHP (PHP Hypertex Preprocessor)

- Lenguaje de propósito general diseñado para el desarrollo de páginas web dinámicas.
- En un principio, lenguaje no tipado (no se necesita declarar el tipo de dato ya que el intérprete/compilador lo deduce).
- Actualmente en la versión 8. Se recomienda al menos utilizar una versión superior a la 7.0.
- Código embebido en el HTML.

- PHP funciona prácticamente con cualquier servidor web y Base de Datos existentes aunque lo más habitual es verlo formando lo que se conoce como una arquitectura XAMP (Linux/Windows, Apache, MySQL y PHP).
- Instrucciones entre etiquetas ***<?php*** y ***?>.*** Para generar codigo dentro de PHP, podemos usar la instrucción ***echo.***
- El código se ejecuta por un entorno de ejecución con el que se integra el servidor web (normalmente utilizando Apache con el módulo **mod_php**).
- Multitud de librerías y frameworks: Laravel, Symfony, Codeigniter, Zend.

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

El siguiente mapa mental muestra un resumen de sus elementos:



Elementos del lenguaje PHP

JSP de JavaEE

Java es uno de los lenguajes de programación orientado a objetos más utilizado hoy en día.

El código fuente se escribe en archivos con extensión .java. El compilador genera por cada clase un archivo .class. Para ejecutar una aplicación programada en Java necesitamos tener instalado un entorno de ejecución (JRE). Para crear aplicaciones en Java necesitamos el kit de desarrollo de Java (JDK), que incluye el compilador.

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

Como ya viste, existen básicamente dos tecnologías que te permiten programar páginas web dinámicas utilizando Java EE: **servlets** (clases Java compiladas que contienen instrucciones de salida para generar las etiquetas HTML de las páginas) y **JSP** (páginas web que contienen instrucciones para añadir contenido de forma dinámica).

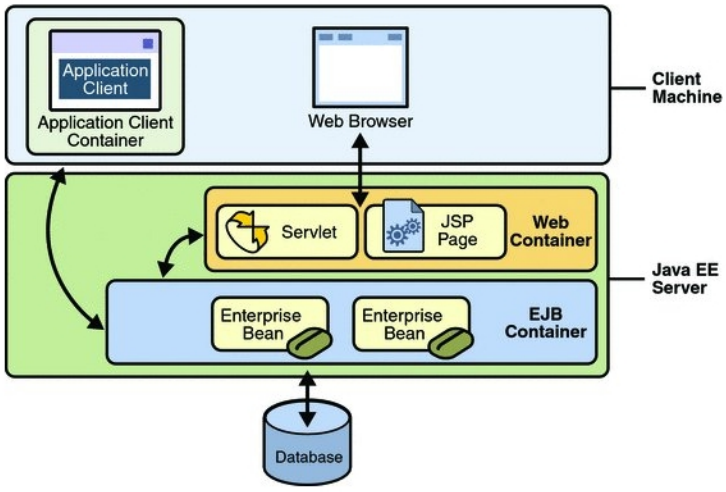
Aunque no es así en todos los casos, la mayoría de implementaciones disponibles para JSP compilan cada página y generan un servlet a partir de la misma la primera vez que se va a ejecutar. Este servlet se almacena para ser usado en futuras peticiones.

Por ejemplo, si quieres calcular la suma de dos números y enviar el resultado al navegador, lo podríamos realizar con una página JSP, incluyendo el código en Java dentro de las etiquetas HTML utilizando los delimitadores <% y %> o utilizando el método println dentro de un servlet.

No hay nada que se pueda hacer con JSP que no pueda hacerse también con servlets. De hecho, como ya viste, las primeras se suelen convertir en servlets para ser ejecutadas.

El problema de utilizar servlets directamente es que, aunque son muy eficientes, son muy tediosos de programar puesto que hay que generar la salida en código HTML con gran cantidad de funciones como `println`. Este problema se resuelve fácilmente utilizando JSP, puesto que aprovecha la eficiencia del código Java, para generar el contenido dinámico, y la lógica de presentación se realiza con HTML normal.

De esta forma estas dos tecnologías se suelen combinar para crear aplicaciones web. Los servlets se encargan de procesar la información y obtener resultados, y las páginas JSP se encargan del interface, incluyendo los resultados obtenidos por los servlets dentro de una página web.



3.3.- Herramientas de programación

A la hora de ponerte a programar una aplicación web, debes tener en cuenta con que herramientas cuentas que te puedan ayudar de una forma u otra a realizar el trabajo. Además de las herramientas que se tengan que utilizar en el servidor una vez que la aplicación se ejecute, como por ejemplo el servidor de aplicaciones o el gestor de bases de datos, de las que ya conoces su objetivo, existen otras que resultan de gran ayuda en el proceso previo, en el desarrollo de la aplicación.

Desde hace tiempo, existen **Entornos Integrados de Desarrollo** (**IDE**) que agrupan en un único programa muchas de estas herramientas. Algunos de estos entornos de desarrollo son específicos de una plataforma o de un lenguaje, como sucede por ejemplo con Visual Studio, el IDE de Microsoft para desarrollar aplicaciones en lenguaje C# o Visual Basic para la plataforma .Net. Otros como Eclipse, NetBeans o Visual Studio Code, permiten personalizar el entorno para trabajar con diferentes lenguajes y plataformas, como Java EE o PHP.

No es imprescindible utilizar un IDE para programar. En muchas ocasiones puedes echar mano de un simple editor de texto para editar el código que necesites. Sin embargo, si tu objetivo es desarrollar una aplicación web, las características que te aporta un entorno de desarrollo son muy convenientes. Entre estas características se encuentran:

- **Resaltado de texto.** Muestra con distinto color o tipo de letra los diferentes elementos del lenguaje: sentencias, variables, comentarios, etc. También genera indentado automático para diferenciar de forma clara los distintos bloques de un programa.
- **Completado automático.** Detecta qué estás escribiendo y cuando es posible te muestra distintas opciones para completar el texto.
- **Navegación en el código.** Permite buscar de forma sencilla elementos dentro del texto, por ejemplo, definiciones de variables.
- **Comprobación de errores al editar.** Reconoce la sintaxis del lenguaje y revisa el código en busca de errores mientras lo escribes.
- **Generación automática de código.** Ciertas estructuras, como la que se utiliza para las clases, se repiten varias veces en un programa. La generación automática de código puede encargarse de crear la estructura básica, para que sólo tengas que rellenarla.
- **Ejecución y depuración.** Esta característica es una de las más útiles. El IDE se puede encargar de ejecutar un programa para poder probar su funcionamiento. Además, cuando algo no funciona, te permite depurarlo con herramientas como la ejecución paso a paso, el establecimiento de puntos de ruptura o la inspección del valor que almacenan las variables.
- **Gestión de versiones.** En conjunción con un sistema de control de versiones, el entorno de desarrollo te puede ayudar a guardar copias del estado del proyecto a lo largo del tiempo, para que si es necesario puedas revertir los cambios realizados.

3.4.- Instalación de Visual Studio Code

En nuestro caso utilizaremos Visual Studio Code:

https://code.visualstudio.com/

3.5.- Instalación de una plataforma XAMPP

Ahora instalaremos XAMPP:

https://www.apachefriends.org/es/index.html

3.6.- Integración PHP en Visual Studio Code

Configuramos en Visual Studio Code la extensión PHP:

https://www.youtube.com/watch?v=BJ0S7CYLVwI