Tema 1: Plataformas de programación web en entorno servidor. Aplicaciones LAMP.

# 1.- Características de la programación web.

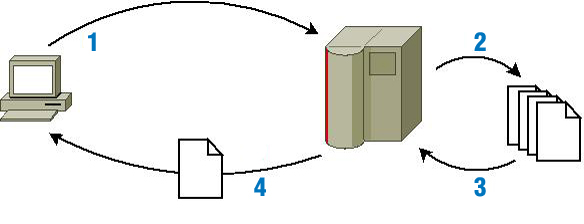
Cuando una **página web se descarga** a tu ordenador, su contenido define qué se debe mostrar en pantalla. Este contenido está programado en un **lenguaje de marcado** (formado por etiquetas), que puede ser **HTML** o **XHTML**.

Dentro de estos lenguajes **hay etiquetas para indicar que un texto es**:

* un **encabezado**: <h1>, <h2>, <h3>…
* **que forma parte de una tabla**: <table>
* **o que simplemente es un párrafo de texto**: <p>.

**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 esta.

Estos dos ficheros se descargan a tu ordenador desde un servidor web como respuesta a una petición. 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.
3. Si **el servidor encuentra esa página**, la recupera.
4. Y por último, **se la envía al navegador** par que este 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.

Interfaz de usuario gráfica

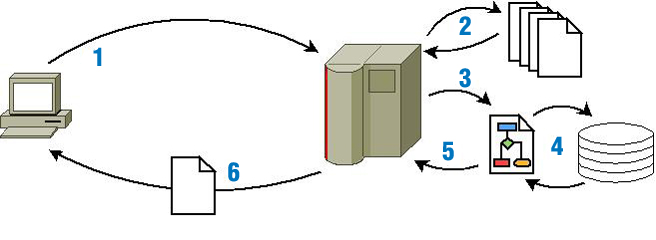
Descripción generada automáticamente con confianza media

## 1.1.- Páginas web estáticas y dinámicas (I).

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**, salvo que el programador la actualice manualmente.

Por ello, existen las **páginas web dinámicas**. El contenido de estas páginas cambia en función de diversas variables. Dentro de las **páginas web dinámicas**, es muy importante distinguir **2 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).
* **Muchas páginas tienen extensiones** .php, .asp, .jsp, .cgi o .aspx. En ellas, el contenido que se descarga al navegador es similar al de una página web estática. **Pero el HTML de estas páginas se crea como resultado de la ejecución de un programa** en dichos lenguajes, la cual tiene lugar en el servidor web (aunque no necesariamente por el mismo que brinda las páginas HTML).



**Pasos**:

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**, 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 o registros almacenados en una Base de datos**.
5. **El resultado** de la ejecución será una página en **formato HTML**.
6. **El servidor web envía el resultado** obtenido al navegador.

Ejemplos de páginas en las que sucede esto: los clientes de correo web, como Gmail, Hotmail (Outlook ahora), Yahoo, etc.

### 1.1.1.- Páginas web estáticas y dinámicas (II).

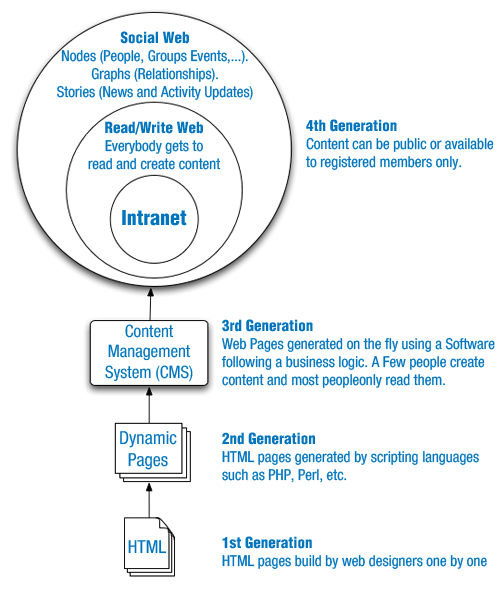
La utilización de páginas web dinámicas no siempre es lo mejor. Es la que más potencia y flexibilidad permite, pero las **páginas web estáticas tienen también algunas ventajas**:

* **Para crear un sitio que utilice únicamente páginas web estáticas, solo hay que conocer HTML/XHTML y CSS**. Incluso ni si quiera haría falta eso, si utilizamos un programa de diseño web para generarlas.
* **Si la página es dinámica, al volver a visitarla utilizando el enlace, su contenido puede variar con respecto a cómo estaba con anterioridad**. Aquí es donde brillan las ventajas de las páginas web estáticas.
  + **Ejemplo**: 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**.

**Pero tiene sus desventajas**:

* La actualización de su contenido debe hacerse de forma manual, lo que puede hacer que el mantenimiento sea prohibitivo en sitios web con gran cantidad de contenido.

Las primeras páginas web que se crearon en Internet, eran **estáticas**. A esta web compuesta por páginas estáticas se le considera la 1ª generación. La 2ª surgió de las páginas web dinámicas.



Por otra parte, tenemos las páginas web dinámicas:

La ejecución de una página web dinámica requiere una serie de recursos del lado del servidor.

* **Para que un servidor web pueda procesar una página web dinámica, necesita ejecutar un programa**.
* Esta ejecución **la realiza un módulo concreto** del servidor web.
* Además, **puede ser necesario consultar una base de datos como parte de la ejecución del programa**.

#### 1.1.1.1.- 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. Unas de las primeras en aparecer fueron los clientes de correo.

Estas aplicaciones **tienen ciertas ventajas e inconvenientes si las comparas con las aplicaciones tradicionales** que se ejecutan sobre el S.O de la propia máquina.

|  |  |
| --- | --- |
| **Ventajas** | * No es necesario instalarlas en los equipos en los que se van a utilizar. Solo hace falta instalarla y ejecutarla en el servidor. * Como se encuentran instaladas en un equipo (el servidor), **es muy sencillo gestionarlas** (copias de seguridad, corrección de errores, actualizaciones…). * **Se pueden utilizar en todos aquellos sistemas que dispongan de un navegador web**. * Se pueden utilizar desde cualquier lugar en el que dispongamos de conexión con el servidor. |
| **Inconvenientes** | * La interface de usuario de las aplicaciones web es la página web que se muestra en el navegador. **Lo que restringe las características de la interface a aquellas de una página web**. * Dependemos de una conexión con el servidor para poder utilizarlas. * **La información que se muestra en el navegador debe transmitirse desde el servidor**. Consecuencia:   + **Cierto tipo de aplicaciones no son adecuadas para su implementación como aplicación web**. Ejemplo: editores de vídeo. |

Hoy día, muchas de las aplicaciones web tienen su contenido almacenado en **bases 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**.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## 1.2.- Ejecución de código en el servidor y en el cliente.

* **En el lado del servidor**: 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**.
* **En el lado del cliente**: cuando una página web llega a tu navegador, es posible que incluya algún programa o fragmento de código que se deban ejecutar, normalmente en el **JavaScript** del navegador. **¿Qué puede hacer este código?**
  + modificar el contenido de la página,
  + 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 la otra**. Esta división es así porque **elcódigo que se ejecuta en el cliente web** **tradicionalmente no tenía acceso a los datos** del servidor.

* **Cuando en tu navegador querías leer un nuevo correo, el código de JavaScript que se ejecutaba en el mismo**, no podía obtener de la base de datos el contenido de ese mensaje. **Lo que se hacía era crear una nueva página en el servidor con la información que se pedía y entregarla al navegador**.
* 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** **pueda comunicarse con un servidor de Internet** para obtener información.
  + Sin salir de una página, poder modificar su contenido en base a la información que se almacena en un servidor de Internet.

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

Así, el código que se ejecuta en el servidor genera páginas web que ya incluyen código destinado a su ejecución en el navegador.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

**Para saber más** sobre los lenguajes del lado del cliente y del lado del servidor: [Lenguajes del lado servidor o cliente (adelat.org)](http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html#:~:text=Javascript%20es%20un%20lenguaje%20de,soporta%20la%20carga%20de%20procesamiento.)

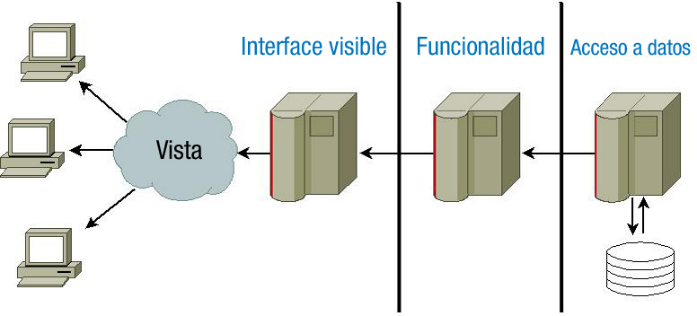
# 2.- Tecnologías para programación web del lado del servidor.

Cuando programas una aplicación, utilizas un lenguaje de programación. **Al programar cada aplicación utilizas ciertas herramientas como un entorno de desarrollo o librerías de código**.

Los **componentes principales con los que debes contar** **para ejecutar aplicaciones web** en un servidor son los siguientes:

* Un **servidor web** para recibir las peticiones de los clientes web y enviarles la página que solicitan.
* El **módulo encargado de ejecutar el código** o programa y generar la página web resultante. El módulo debe integrarse 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**.
* El **lenguaje de programación** **que utilizarás** para desarrollar las aplicaciones.

También es importante decidir cómo vas a **organizar el código**.

**Las arquitecturas** que se usan en la programación de aplicaciones web **te ayudan a estructurar el código de las aplicaciones en** **capas o niveles**.

De esta forma, **se separan las funciones lógicas** de una aplicación, para que pueda ser posible ejecutar cada una en un servidor distinto (en caso de que sea necesario).

**En una aplicación puedes distinguir funciones de**:

* **Presentación**: para dar formato.
* **Lógica**: para ejecutar un proceso y obtener un resultado.
* **Persistencia**: para mantener los datos.
* **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**: en ella se encontrará el interface de usuario.
* Una **capa intermedia**, donde deberás programar la funcionalidad de tu aplicación.
* Una **capa de acceso a datos**: encargada de almacenar la información de la aplicación en una base de datos, y recuperarla cuando sea necesario.

## 2.1.- Arquitecturas y plataformas.

Lo primero que hay que elegir antes de comenzar a programar una app web es la **arquitectura**:

|  |  |
| --- | --- |
| **Tipo** | **Características** |
| JavaEE (Enterprise Edition) (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**. Ya que Java Cuenta con muchas librerías |
| * 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**. |
| En esta arquitectura, **existen distintas tecnologías**:   * **Páginas JSP y Servlets**: ambos orientados a la generación dinámica de páginas web. * **Enterprise JavaBeans (EJB)**: componentes que normalmente aportan la lógica de la aplicación web. |
| AMP | * **Son las siglas** de Apache (servidor web), MySQL (servidor de base de datos) y PHP/Perl/Python (lenguaje de programación utilizado). * **Dependiendo del sistema operativo que se utilice para el servidor, hablamos de**:   + **LAMP**: para Linux (L).   + WAMP: para Windows (W).   + MAMP: para MAC (M). * **También es posible usar otros componentes**, **como** el gestor de bases de datos **PostgreSQL en lugar de MySQL**. |
| **Para saber más**:  Existen paquetes software que incluyen en una única instalación una plataforma AMP completa. Algunos ni si quiera es necesario instalarlos, e incluso son multiplataforma (disponibles tanto para Windows, Mac y Linux).  [XAMPP Installers and Downloads for Apache Friends](https://www.apachefriends.org/es/index.html) |
| CGI/Perl | **Es la combinación de dos componentes**:   * **Perl**: es un potente lenguaje de código libre creado originalmente para la administración de servidores. Tiene una amplia comunidad de usuarios y mucho código libre disponible. * **CGI**: es un estándar para permitir al servidor web ejecutar programas genéricos, escritos en cualquier lenguaje (incluso C o Python). |
|  | **Inconveniente**:CGI **es lento.** Aunque existen métodos para acelerarlo. |
| ASP.Net | * **Es la arquitectura comercial de Microsoft para el desarrollo de aplicaciones**. Se encarga de la generación de páginas web dinámicas. Es una evolución de la anterior tecnología, ASP. * **Lenguajes de programación**: Visual Basic.NET o C#. * **Utiliza el servidor web de Microsoft**, Internet Information Server (IIS). * Puede tener obtener información de **varios gestores de BB.DD**, entre los que se incluye Microsoft SQL Server. * **Desventaja**: plataforma comercial de código propietario. |

### 2.1.1.- Selección de una arquitectura de programación web.

Hay muchas decisiones que debes tomar antes aún de comenzar el desarrollo de una aplicación web.

* ¿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**?
* ¿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?

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## 2.2.- Integración con el servidor web.

La **comunicación** **entre un cliente web** (navegador) **y un servidor web** se lleva a cabo gracias al protocolo HTTP. **HTTP es el vínculo de unión entre el usuario y la aplicación en sí**.

En el lado del servidor, **estas peticiones son procesadas por el servidor web** (también llamado servidor HTTP). **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.

|  |  |  |
| --- | --- | --- |
| **Arquitectura** | **Cómo se integra con el servidor web** | |
| **CGI** | * Es un **protocolo estándar** que existe en muchas plataformas. Lo implementan la gran mayoría de servidores web. * Define qué debe hacer el servidor web para delegar en un programa externo la generación de una página web. * Esos **programas externos** son los **guiones CGI**, independientemente del lenguaje en el que estén programados (aunque se suelen programar en lenguajes de guiones como Perl). | |
| * **Principal problema**: cada vez que se ejecuta un guión CGI, el S.O debe crear un nuevo proceso. Esto implica un mayor consumo de recursos y menor velocidad de ejecución. * Existen algunas **soluciones para acelerarlo como FastCGI**, y también **otros métodos para ejecutar guiones**. Por ejemplo: el **módulo mod\_perl** **para ejecutar** en Apache **guiones programados en Perl**. | |
| * También **es posible ejecutar código PHP utilizando CGI**, pero los intérpretes PHP no se suelen utilizar con este método.   + El módulo **mod\_php** es la forma más habitual que se utiliza para **ejecutar guiones en PHP utilizando** **plataformas AMP**. Con Python sería el módulo **mod\_python**. | |
| **Java EE** | * Es la arquitectura más **compleja**. **Para poder ejecutar aplicaciones Java EE en un servidor, tenemos dos opciones**:   + **Servidores de aplicaciones**, que implementan todas las tecnologías disponibles en Java EE.   + **Contenedores de servlets**, que soportan solo parte de la especificación. * Dependiendo de la magnitud de nuestra aplicación y de las tecnologías que utilice, usaremos una solución u otra. | |
| Hay varias implementaciones de servidores de apps Java EE certificados. | |
| **Comerciales** | **De código abierto** |
| * IBM Websphere * BEA Weblogic | * JBoss. * Geronimo (de la fundación Apache). * Glassfish. |
| **Para saber más**: consultar lista con los servidores de apps Java EE certificados por Sun: [Java EE - Wikiwand](https://www.wikiwand.com/es/Java_EE#/Servidores_de_Aplicaciones_Java_EE_5_certificados) | |
| * Sin embargo, en la mayoría de ocasiones, **no es necesario utilizar un servidor de apps completo**, sobre todo si no utilizamos EJB en nuestras apps. **Nos será suficiente con un contenedor de servlets**. * En esta área, destaca **Tomcat**, implementación por referencia de un contenedor de servlets, que es de código abierto además. * **Una vez instalada la solución** que hayamos escogido, **tenemos que integrarla con el servidor web que utilicemos**. * **Otra opción es utilizar una única solución para páginas estáticas y páginas dinámicas**.   + **Tomcat incluye un servidor HTTP propio** que puede sustituir a Apache. | |
| **ASP.Net** | * Utiliza el servidor IIS de Microsoft, que ya integra soporte en forma de módulos para manejar peticiones de páginas dinámicas de ASP y ASP.Net. | |

# 3.- Lenguajes.

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:

|  |  |
| --- | --- |
| **Lenguajes de guiones (scripting)** | * Son aquellos en los que los programas **se ejecutan directamente a partir de su código fuente original**. * Se almacenan 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**. * **Lenguajes**: Perl, Python, PHP y ASP (precursor de ASP.Net) |
| * **Ventajas**: no es necesario traducir el código fuente original para ser ejecutados, lo que aumenta su portabilidad. * **Desventajas**: el proceso de interpretación tiene peor rendimiento que las otras alternativas. |
| **Lenguajes compilados a código nativo** | * Son aquellos cuyo código fuente se traduce a código binario, dependiente del procesador, antes de ser ejecutado. * El servidor web almacena los programas en su modo binario, que ejecuta directamente cuando se les invoca. * **El método principal** para ejecutar programas binarios desde un servidor web **es CGI**. Esto nos permite ejecutar código programado en cualquier lenguaje. |
| **Ventajas e inconvenientes**:   * Son los de **mayor velocidad de ejecución**, pero tienen **problemas en la integración con el servidor web**. * Son programas de propósito general que **no están pensados para ejecutarse en el entorno de un servidor web**. * **Por cada petición** que se haga al servidor web, se crea un **nuevo proceso**. Además, **no son portables** entre distintas plataformas. |
| **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. **Ejemplo**: Java (Servlets y páginas JSP) y ASP.Net. * En ambas plataformas e implementaciones de Java EE, se utiliza un procedimiento de **compilación JIT** (Just In Time), es decir, el código intermedio se convierte en código binario “al vuelo”. * **Esto acelera la ejecución**, ya que el compilador puede traducir todo o parte del código intermedio a código nativo cuando se invoca un programa. **El código ya traducido a nativo se almacena** para ser utilizado de nuevo cuando sea necesario. |
| **Ventajas**: ofrecen un equilibrio entre las 2 opciones anteriores.   * Su rendimiento es muy bueno. * Puede portarse entre distintas plataformas en las que exista una implementación de la arquitectura (contenedor de servlets o servidor de aplicaciones Java EE). |

## 3.1.- Código embebido en el lenguaje de marcas.

Hay dos formas de embeber el código en el lenguaje de marcas:

|  |  |
| --- | --- |
| **Integrar etiquetas HTML en el código de los programas** | Los guiones CGI son programas estándar que se ejecutan en el S.O, pero que generan como salida el código HTML de una página web. |
| **Los guiones CGI deben contener, mezcladas dentro de su código, sentencias encargadas de generar la página web**. Por ejemplo, si quieres generar una página web utilizando CGI a partir de un guión de sentencias en Linux, tienes que hacer algo como lo siguiente. |
| De esta forma se programan, por ejemplo, los guiones CGI y los servlets. |
| **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 solo 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: |
| Esta metodología de programación es la que **se emplea en los lenguajes ASP, PHP y con las páginas JSP de Java EE**. |

Diferencias entre Java EE y páginas JSP:

* **Java EE**. Son programas Java compilados y almacenados en el contenedor de servlets.
* **Páginas JSP**: contiene código Java embebido en lenguaje HTML. La primera vez que se necesita una página JSP, se convierte a un servlet y éste se guarda para ser utilizado en posteriores llamadas a la misma página

En la arquitectura de ASP.Net

* **Cada página se divide en dos ficheros**: uno contiene las etiquetas HTML y otro el código en el lenguaje de programación utilizado.
* **De esta forma se logra independencia** entre el aspecto de la aplicación y la gestión del contenido dinámico.
* **A partir de esos ficheros se obtiene un código intermedio** (MSIL: Microsoft Intermediate Language), que es lo que se almacena en el servidor.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## 3.2.- Herramientas de programación.

Debes tener en cuenta con qué herramientas cunetas que te puedan ayudar de una forma u otra a realizar el trabajo.

Además de las que ya conoces (servidor de aplicaciones, gestor de bases de datos), **hay 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 IDEs son específicos de una plataforma o de un lenguaje**: Visual Studio, para desarrollar en lenguaje C# o Visual Basic para la plataforma .Net.
* **Otros** como Eclipse o NetBeans **te permiten** personalizar el entorno para **trabajar con diferentes lenguajes y plataformas**, como Java EE o PHP.

No es imprescindible utilizar un IDE para programar, pero si tu objetivo es desarrollar una aplicación web, **las** **características de un entorno de desarrollo son muy convenientes**:

* **Resaltado de texto**: sentencias, variables, comentarios. También genera indentado automático.
* **Completado automático**.
* **Navegación en el código**: permite buscar de forma sencilla elementos dentro del texto, como, por ejemplo, definiciones de variables.
* **Comprobación de errores al editar**: reconoce la sintaxis del lenguaje.
* **Generación automática de código**: puede encargarse de crear la estructura básica, para que solo tengas que rellenarla.
* **Ejecución y depuración**:
  + El IDE se puede encargar de **ejecutar** un programa para poder probar su funcionamiento.
  + **Además, te permite depurarlo** con herramientas como la ejecución paso a paso, el establecimiento de puntos de ruptura o la **inspección del valor**.
* **Gestión de versiones**: en conjunción con un sistema de control de versiones, el IDE te puede ayudar a guardar copias del estado del proyecto a lo largo del tiempo.

**Los dos IDE de código abierto más utilizados en la actualidad son Eclipse y NetBeans**. Aunque en sus orígenes se centraron en la programación en lenguaje Java, **hoy en día admiten directamente o a través de módulos, varios lenguajes** entre los que se incluyen C, C++, PHP, Python y Ruby.

**Ambos** **además ofrecen para la descarga versiones personalizadas** del IDE que pueden ser usadas directamente **para programar en un lenguaje determinado**, sin necesidad de cambiar la configuración o instalar módulos.

### 3.2.1.- Instalación de NetBeans para PHP en Linux.

1. Lo primero que debes **comprobar** es **si tienes instalada la última versión de Java**:

java -version

1. En caso de no tener **Java instalado**, debemos instalarlo antes del IDE.
   1. **Instalar los repositorios**:
      1. Si tienes Ubuntu 10.04:

sudo add-apt-repository “deb http://archive.canonical.com/ lucid partner”

* + 1. Si tienes Ubuntu 10.10:

sudo add-apt-repository “deb http://archive.canonical.com/ maverick partner”

* 1. Actualiza la lista de paquetes e instala Java:

sudo apt-get update

sudo apt-get install sun-java-jdk

1. Mientras instalas Java, aprovecha para descargar NetBeans desde su página oficial: [Welcome to Apache NetBeans](https://netbeans.apache.org/)
2. Para instalar NetBeans:
   1. Debes marcar el fichero como ejecutable:

chmod +x netbeans-7.0beta2-ml-php-linux.sh

* 1. Ejecuta el instalador:

sudo ./netbeans-7.0beta2-ml-php-linux.sh

1. Una vez finalizado, puedes acceder a NetBeans mediante una nueva entrada que se ha generado en el menu.

### 3.2.2.- Instalación de una plataforma LAMP en Ubuntu.

Una vez instalado el IDE, es necesario instalar el resto de la plataforma de desarrollo. **Si vas a programar en PHP, necesitas todos los componentes de una arquitectura LAMP**.

Puedes instalar los componentes uno a uno o utilizar el **comando tasksel** desde la consola. Este comando viene con algunas tareas predefinidas, que nos permiten instalar con un solo comando grupos de aplicaciones.

**Pasos**:

1. **sudo tasksel install lamp-server**
2. Durante la instalación, tendrás que **introducir la contraseña para el usuario administrador (root) de MySQL**.
3. Para **probar el correcto funcionamiento de la plataforma**, puedes crear en el directorio /var/www (que es la carpeta raíz que viene configurada de forma predeterminada de la cual Apache extrae los documentos a publicar) un fichero “prueba.php” con el siguiente contenido:

<?php

phpinfo();

?>

Lo que estás haciendo es **llamar a una función del lenguaje PHP que genera por sí misma una página web completa** con información sobre la instalación de PHP.

1. Si todo va bien, al abrir <http://localhost/prueba.php>, deberías poder ver la página de la que hablábamos antes, dentro de la propia máquina Ubuntu en la que has instalado LAMP).

**Para saber más**: sobre el proceso de instalación de LAMP y configuración de PHP en Ubuntu: [Configuring the PHP Development Environment in Linux Ubuntu (apache.org)](https://netbeans.apache.org/kb/docs/php/configure-php-environment-ubuntu.html)

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## 3.3.- Programación web con Java.

Java es el lenguaje de programación más utilizado hoy en día. Es un lenguaje orientado a objetos. Todo el código que escribas en Java debe pertenecer a una clase. Algunas generalidades:

* **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 el entorno de ejecución (JRE).
* Para crear aplicaciones en Java, necesitamos el kit de desarrollo de Java (JDK), que incluye el compilador.

Existen básicamente d**os tecnologías que te permiten programar páginas web dinámicas utilizando Java EE (Enterprise Edition)**:

* **Servlets**: clases compiladas que contiene instrucciones de salida para generar las etiquetas HTML de las páginas.
* **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.

**Por ejemplo**: si quieres calcular la suma de 2 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 %> de la siguiente manera:

|  |  |
| --- | --- |
| **Incrustando el código Java en el HTML: creando una JSP** |  |
| **Creando un servlet** | Utilizando el método println dentro de un servlet, que obtiene los valores a sumar de otra página que ha hecho la solicitud Http: |

No hay nada que se pueda hacer con JSP que no pueda hacerse también con servlets. **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.

**Con JSP podemos aprovechar la eficiencia del código Java para generar el contenido dinámico, y la lógica de presentación se realiza con HTML normal**.

**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.

## 3.4.- Programación web con PHP.

PHP es un lenguaje de guiones de propósito general, pero diseñado para el desarrollo de páginas web dinámicas utilizando código embebido dentro del lenguaje de marcas. Su sintaxis es muy similar a la de Java.

Los **delimitadores recomendados para incluir código PHP dentro de una página web** son<?php y ?>

* El código **se ejecuta por un entorno de ejecución con el que se integra el servidor web** (con el módulo de Apache **mod\_php**).
* La configuración tanto del servidor web Apache, como de PHP, se realiza por medio de ficheros de configuración.
  + El de Apache es **httpd.conf**,
  + y el de PHP es **php.ini**.

El **php.ini** puede encontrarse en distintas ubicaciones. La función **phpinfo()** **te informa del lugar en el que se encuentra almacenado** el fichero php.ini.

En nuestro caso es en **/etc/php5/apache2/php.ini**.

**Dependiendo de cómo se integre PHP con Apache** **los cambios** que realices en su fichero de configuración **se aplicarán en un momento o en otro**.

* **Si utilizamos mod\_php para ejecutar PHP como un módulo de Apache** (nuestro caso): la configuración PHP se aplicarán tras reiniciar el servidor Apache.
  + **Reiniciar Apache**: dos formas de hacerlo:
    - sudo /etc/init.d/apache2 restart
    - sudo service apache2 restart
* Alternativa… No la han sugerido.

Algunas de las **directivas más utilizadas** que figuran en el fichero **php.ini** son:

* **short\_open\_tags**: Indica si se pueden utilizar en PHP los delimitadores cortos <? y ?>
* **max\_execution\_time**: permite que puedas ajustar el nº máximo de segundos que podrá durar la ejecución de un script PHP.
* **error\_reporting**. Indica **qué** **tipo de errores se mostrarán** en el caso de que se produzcan.
  + Por ejemplo, si haces **error\_reporting = E\_ALL**, te mostrará todos los tipos de errores.
  + Si no quieres que te muestre los avisos pero sí otros tipos de errores, puedes hacer **error\_reporting = E\_ALL & ~E\_NOTICE**.
* **file\_uploads**: Indica si se pueden o no subir ficheros al servidor por HTTP.
* **upload\_max\_filesize**. En caso de que se puedan subir ficheros por HTTP, puedes indicar el límite máximo permitido para el tamaño de cada archivo.
  + Por ejemplo, **upload\_max\_filesize = 1M**.

**Para saber más**: lista completa de las directivas que se pueden utilizar en **php.ini**. [PHP: Descripción de las directivas del núcleo de php.ini - Manual](https://www.php.net/manual/es/ini.core.php)

A medida que vayas escribiendo código en PHP, **será útil que introduzcas en el mismo algunos comentarios**.

* **En** una página web los comentarios al **HTML** **van entre los delimitadores <!—y -->**
* Dentro del código PHP, hay tres formas de poner comentarios:
  + **Comentarios de una línea utilizando //**: igual que en C.
  + **Comentarios de una línea #**: igual que en la sintaxis de los scripts de Linux.
  + **Comentarios de varias líneas**: con /\* para abrir comentario y \*/ para cerrarlo.

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente con confianza media

### 3.4.1.- Variables y tipos de datos en PHP.

* Las variables en PHP siempre **deben comenzar por el signo $**.
* Los nombres de las variables **deben comenzar por letras o por el carácter \_**, y **pueden contener también números**.
* Sin embargo, en PHP no es necesario declarar una variable ni especificarle un tipo (entero, cadena,…) concreto. **Para empezar a usar una variable, simplemente asígnale un valor utilizando el operador =**.
  + **Ejemplo**: $mi\_variable = 7;
* **Dependiendo del valor que se le asigne**, a la variable se le aplica un tipo de datos, que puede cambiar si cambia su contenido. **El tipo de la variable se decide en función del contexto en que se emplee**.
* **Puedes declarar varias variables y asignarles el mismo valor**. Por ejemplo:
  + **$a = $b = 3.1415**

**Texto

Descripción generada automáticamente**

**Los tipos de datos simples en PHP son**:

* Booleano (boolean).
* Entero (integer).
* Real (float).
* Cadena (string).
* null: es un tipo de datos especial, que se usa para indicar que la variable no tiene valor.

**Por ejemplo**:

Texto

Descripción generada automáticamente

Si realizas una operación con variables de distintos tipos, ambas se convierten primero a un tipo común. **Por ejemplo, si sumas un entero con un real, el entero se convierte a real antes de realizar la suma**.

Texto

Descripción generada automáticamente

Estas conversiones de tipo, también se pueden realizar de forma forzada (haciendo casting).

Texto

Descripción generada automáticamente

**Debes conocer**: las conversiones de tipo posibles y los resultados obtenidos con cada una.

#### Anexo I. Conversiones de tipo posibles y resultados obtenidos con cada una.

Imagen que contiene nombre de la empresa

Descripción generada automáticamente

**Los siguientes forzados de tipos están permitidos**:

* (int), (integer) - **forzado a integer**
* (bool), (boolean) - forzado a boolean
* (float), (double), (real) - forzado a float
* (string) - forzado a string
* (array) - forzado a array
* (object) - forzado a object
* (unset) - forzado a NULL (PHP 5)

El forzado (binary) y el soporte del prefijo **b** fueron añadidos en PHP 5.2.1.

**Ejemplos**:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

### 3.4.2.- Expresiones y operadores.

En PHP **se utilizan las expresiones para realizar acciones dentro de un programa**. Todas las expresiones deben contener al menos un operando y un operador.

Diagrama

Descripción generada automáticamente con confianza media

**Los operadores que se pueden usar en PHP** son similares a los de lenguajes como Java:

* **Realizar operaciones aritméticas**:
  + Identidad: +$a
  + Negación: -$a
  + Suma: +
  + Resta: -
  + Multiplicación: \*
  + División: /
  + Módulo: %
  + Exponenciación: \*\*
  + Operador preincremento y predecremento: ++$a; --$a
  + Operador postincremento y postdecremento: $a++; $a—

**Ejemplos**:

Texto

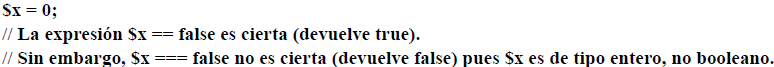
Descripción generada automáticamente

Texto

Descripción generada automáticamente

* **Realizar asignaciones**: además del operador =, existen operadores con los que realizar operaciones y asignaciones en un único paso:
  + += para sumar valores al valor inicial de la variable a la que se refiera.
  + -= lo mismo pero para restar.
* **Comparar operandos**: además de > y >=, en PHP tenemos:
  + **Dos operadores de igualdad**:
    - ==: comprueba si dos variables son iguales en contenido.
    - ===: comprueba si dos variables son iguales en contenido y tipo.
  + **Tres operadores para comprobar diferencia**:
    - <> y !=: comprueba si dos variables son distintas en contenido.
    - ¡==: comprueba si dos variables son distintas en contenido y tipo.

**Ejemplo**:



* **Comparar expresiones booleanas**: tratan a los operandos como variables booleanas (true o false). Existen **operaciones para**:
  + **Y lógico**: and o &&
  + **O lógico**: or o ||
  + **No lógico**: !
  + **O lógico exclusivo** (o esto, o lo otro, pero no ambos): xor
* **Realizar operaciones con los bits que forman un entero**: invertirlos, desplazarlos, etc.

Creo que se refieren a estos:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

* **Operaciones con arrays**:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Debes conocer**: en la documentación PHP se explican en detalle todos los operadores disponibles en el lenguaje y la forma en que se utilizan: [PHP: Operadores - Manual](https://www.php.net/manual/es/language.operators.php)

### 3.4.3.- Ámbito de utilización de las variables.

* En PHP puedes **utilizar variables en cualquier lugar de un programa**.
* Si esa variable aún no existe, la primera vez que se utiliza se reserva espacio para ella.
* Las variables dentro de funciones son locales a la función, por tanto, no son accesibles desde fuera.
* **Por tanto, si aparece una asignación fuera de la función, se le considerará una variable distinta (exactamente igual que en Java)**. Ejemplo:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Si se desea utilizar la variable externa $a, se usa **la palabra global**.

Texto

Descripción generada automáticamente

* Si quieres mantener el valor de una variable local a una función entre distintas llamadas a la función, deberás declarar la variable como estática, mediante **la palabra static**.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

* + Las variables estáticas **deben inicializarse en la misma sentencia en que se declaran como estáticas**.

**Para saber más**: sobre el ámbito de las variables: [PHP: Ámbito de las variables - Manual](https://www.php.net/manual/es/language.variables.scope.php)

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente