

DESARROLLO WEB

ENTORNO SERVIDOR

Desarrollo de Aplicaciones Web

ÍNDICE

1.- Características de la programación web.	- 2 -
1.1.- Páginas web estáticas y dinámicas (I).	- 3 -
1.1.1.- Páginas web estáticas y dinámicas (II).	- 4 -
1.2.- Ejecución de código en el servidor y en el cliente.	- 6 -
2.- Tecnologías para programación web del lado del servidor.	- 8 -
2.1.- Arquitecturas y plataformas.	- 9 -
2.1.1.- Selección de una arquitectura de programación web.	- 10 -
2.2.- Integración con el servidor web.	- 11 -
3.- Lenguajes.	- 13 -
3.1.- Código embebido en el lenguaje de marcas.	- 14 -
3.2.- Herramientas de programación.	- 15 -
3.2.1.- Instalación de NetBeans para PHP en Linux.	- 16 -
3.2.2.- Instalación de una plataforma LAMP en Ubuntu.	- 17 -
3.3.- Programación web con Java.	- 18 -
3.4.- Programación web con PHP.	- 19 -
3.4.1.- Variables y tipos de datos en PHP.	- 20 -
3.4.2.- Expresiones y operadores.	- 21 -
3.4.3.- Ámbito de utilización de las variables.	- 22 -

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.

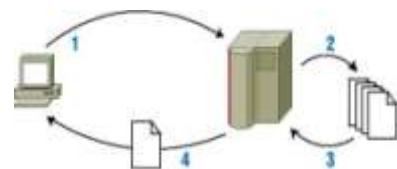
Cuando una **página web se descarga** a tu ordenador, su contenido define qué se debe mostrar en pantalla. Este contenido está programado en un **lenguaje de marcado**, formado por etiquetas, que puede ser **HTML** o **XHTML**. Las etiquetas que componen la página indican el objetivo de cada una de las 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.

¿Podemos ver una página web sin que intervenga un servidor web?

**Sí****No**

Podemos ver páginas web con extensión .htm, .html o .xhtml que tengamos almacenadas en nuestro equipo simplemente abriéndolas con el navegador. En este caso la única utilidad del servidor web es enviar la página que solicitemos a nuestro equipo.

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

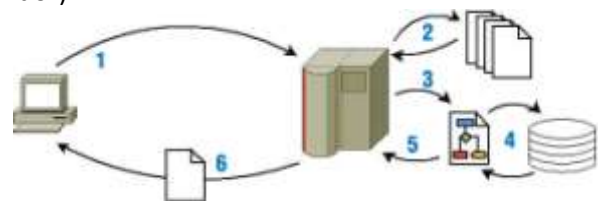
Las páginas que viste en el ejemplo anterior se llaman **páginas web estáticas**. Estas páginas se encuentran almacenadas en su forma definitiva, tal y como se crearon, y su contenido no varía. Son útiles para mostrar una información concreta, y mostrarán esa misma información cada vez que se carguen.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



Pasos:

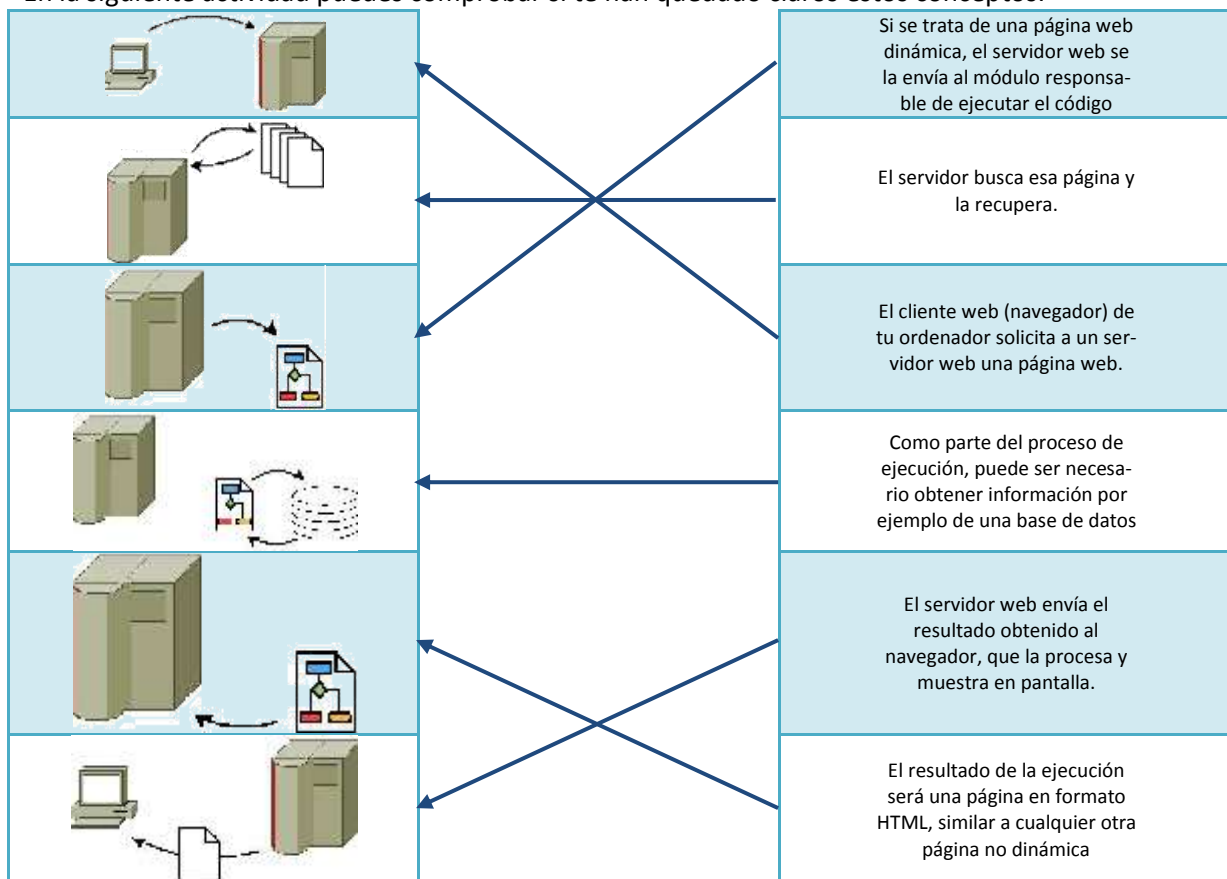
1. **El cliente web** (navegador) de tu ordenador **solicita** a un servidor web una **página web**.
2. El **servidor busca** esa página y la recupera.
3. En el caso de que se trate de una **página web dinámica**, es decir, que su contenido deba ejecutarse para obtener el HTML que se devolverá, el **servidor 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 apare-

cen 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.



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

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 comuniquen 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.

¿Qué tipo de tecnología emplearías para crear una página web personal? ¿Sería necesario utilizar páginas dinámicas? ¿Qué tareas de actualización y mantenimiento tendrías que realizar en cada caso?

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. Tomando como base las web dinámicas, han ido surgiendo otras tecnologías que han hecho evolucionar Internet hasta llegar a lo que ahora conocemos.

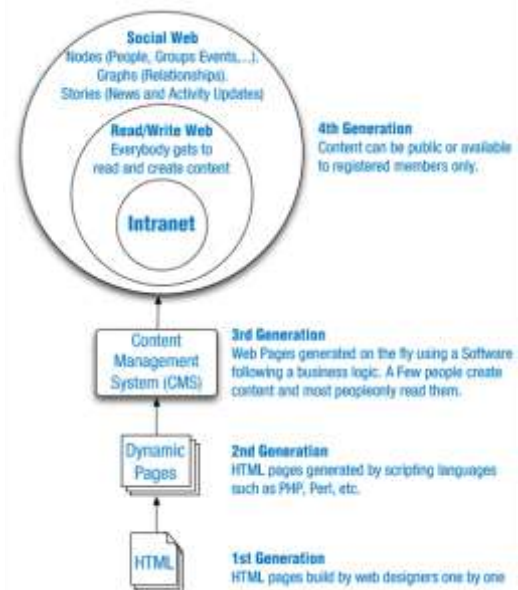
1.1.1.1.- Aplicaciones web.

Las **aplicaciones web** emplean **páginas web dinámicas** para crear aplicaciones que se **ejecuten en un servidor web** y se muestren en un navegador. Puedes encontrar aplicaciones web para realizar múltiples tareas. Unas de las primeras en aparecer fueron las que viste antes, 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.

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.



¿Cuál de las siguientes no es una característica de una aplicación web?

- ☐ Sólo es necesario instalarla una vez.
- ☐ Se crea a partir de páginas web dinámicas.
- ☐ Se puede utilizar en múltiples sistemas.
- ☒ **Sólo necesita un servidor web para ejecutarse.**

Las aplicaciones web emplean páginas web dinámicas. Y como ya vimos, para ejecutar páginas web dinámicas se necesitan una serie de recursos adicionales en el servidor, como un módulo que ejecute el código o un sistema gestor de bases de datos.

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** 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**. Es decir, cuando en tu navegador querías leer un nuevo correo, el código Javascript que se ejecutaba en el mismo no podía obtener de la base de datos el contenido de ese mensaje. La solución era crear una nueva página en el servidor con la información que se pedía y enviarla de nuevo 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 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 nuestro ejemplo, cuando pulsas con el ratón encima de un correo que quieres leer, la página puede contener código Javascript que detecte la acción y, en ese instante, consultar a través de Internet el texto que contiene ese mismo correo y mostrarlo en la misma página, modificando su estructura en caso de que sea necesario. Es decir, 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 vas a aprender a crear aplicaciones web que se ejecuten en el lado del servidor. Otro módulo de este mismo ciclo, 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. Hacia el final de este módulo verás las técnicas que se usan para programar aplicaciones que incorporen esta funcionalidad.

Si quieres verificar que la contraseña introducida en una página web tenga una longitud mínima, ¿dónde sería preferible que se ejecutara el código de comprobación?



En el navegador web.

En el servidor web.

Obviamente; no es necesario enviar la contraseña al servidor web para comprobar su longitud, cuando esa tarea se puede hacer perfectamente 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 bloque **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.

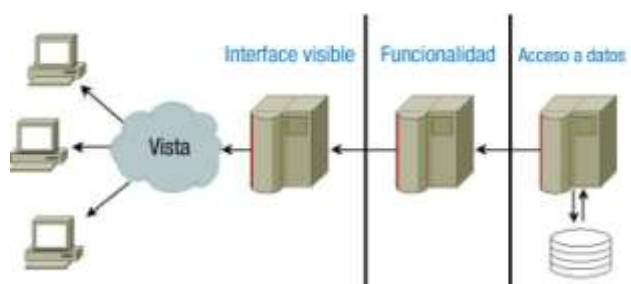
Los **componentes principales** con los que debes contar para ejecutar aplicaciones web en un servidor son los siguientes:

- ✓ Un **servidor web** para recibir las peticiones de los clientes web (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**.

El motivo de dividir en capas el diseño de una aplicación es que se puedan **separar las funciones lógicas** de la misma, de tal forma que sea posible ejecutar cada una en un servidor distinto (en caso de que sea necesario).

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 ob-



tiene 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.

2.1.- Arquitecturas y plataformas.

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), que 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, 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.

<http://www.apachefriends.org/es/xampp.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.

2.1.1.- 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. ¿Crees que vale la pena el esfuerzo de cambiar de arquitectura de desarrollo para cada aplicación? ¿Es necesario en todos los casos?

¿Cuál de estas tecnologías permite la ejecución por el servidor web de programas escritos en cualquier lenguaje?

- ☐ PHP.
- ☐ Java EE.
- ☐ AMP.
- ☒ CGI.

Efectivamente es correcto, CGI es un estándar que permite al servidor web la ejecución de programas genéricos, escritos en cualquier lenguaje.

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.

La tecnología más antigua es CGI. 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 se conocen como guiones CGI, independientemente del lenguaje en el que estén programados (aunque se suelen programar en lenguajes de guiones como Perl).

El principal problema de CGI es que cada vez que se ejecuta un guión CGI, el sistema operativo debe crear un nuevo proceso. Esto implica un mayor consumo de recursos y menor velocidad de ejecución. Existen algunas soluciones que aceleran la ejecución, como FastCGI, y también otros métodos para ejecutar guiones en el entorno de un servidor web, por ejemplo el módulo **mod_perl** para ejecutar en Apache guiones programados en Perl.

Aunque también es posible ejecutar código en lenguaje PHP utilizando CGI, los intérpretes PHP no se suelen utilizar con este método. Al igual que **mod_perl**, existen otros módulos 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.

Puedes consultar una lista con los servidores de aplicaciones Java EE certificados por Sun en la Wikipedia.

http://es.wikipedia.org/wiki/Java_EE#Servidores_de_Aplicaciones_Java_EE_5_certificados

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.

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. Debes distinguir tres grandes grupos:

- ✓ **Lenguajes de guiones** (scripting). Son aquellos en los que los programas se ejecutan directamente a partir de su código fuente (*Conjunto de instrucciones que componen un programa, y que no son ejecutables directamente, sino que deben traducirse utilizando un compilador, intérprete o similar antes de que pueda ser ejecutado por la máquina*) 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 Perl, Python, PHP y ASP (el precursor de ASP.Net).

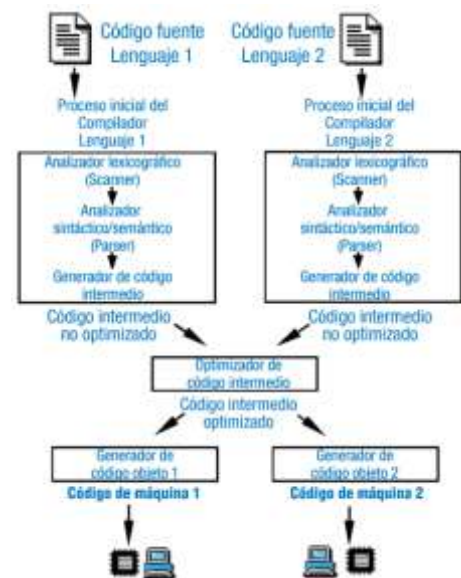
- ✓ **Lenguajes compilados a código nativo** (*Denominación habitual del lenguaje máquina. Código que puede ser ejecutado directamente por el procesador*). Son aquellos en los que el código fuente se traduce a código binario, dependiente del procesador, antes de ser ejecutado. El servidor web almacena 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. Utilizando CGI podemos hacer que el servidor web ejecute código programado en cualquier lenguaje de propósito general como puede ser C.

- ✓ **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.

En la plataforma ASP.Net y en muchas implementaciones de Java EE, se utiliza un procedimiento de compilación JIT. Este término hace referencia a la forma en que se convierte el código intermedio a código binario para ser ejecutado por el procesador. Para acelerar la ejecución, el compilador puede traducir todo o parte del código intermedio a código nativo cuando se invoca a un programa. El código nativo obtenido suele almacenarse para ser utilizado de nuevo cuando sea necesario.



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

- ✓ Los lenguajes de guiones 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 nativo son los de mayor velocidad de ejecución, pero tienen problemas en lo relativo a su integración con el servidor web. Son programas de propósito general que no están pensados para ejecutarse en el entorno de un servidor web. Por ejemplo, no se reutilizan los procesos para atender a varias peticiones: por cada petición que se haga al servidor web, se debe ejecutar un nuevo proceso. Además los programas no son portables entre distintas plataformas.
- ✓ Los lenguajes compilados a código intermedio ofrecen un equilibrio entre las dos opciones anteriores. Su rendimiento es muy bueno 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. Los guiones CGI son programas estándar, que se ejecutan por el sistema operativo, pero que generan como salida el código HTML de una página web. Por tanto, 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:

```
1 #!/bin/sh
2 # Generamos la cabecera HTTP
3 echo "Content-Type: text/html"
4 echo ""
5 # y el contenido de la página web
6 echo "<html>"
7 echo "<head>"
8 echo "<title>Prueba CGI</title>"
9 echo "</head>"
10 echo "<body>"
11 echo "Prueba de guión bash CGI"
12 echo "</body>"
13 echo "</html>"
```

Esta es una de las principales formas de realizar páginas web dinámicas: **integrar las etiquetas HTML en el código de los programas**. Es decir, los programas como el guión anterior incluyen dentro de su código sentencias de salida (echo en el caso anterior) que son las que incluyen el código HTML de la página web que se obtendrá cuando se ejecuten. De esta forma se programan, por ejemplo, los guiones CGI y los servlets.

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:

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
2 <html>
3 <head>
4 <title>Prueba PHP</title>
5 <meta http-equiv="Content-type" content="text/html; charset=utf-8">
6 </head>
7 <body>
8 Prueba de página en PHP. </br>
9 El nombre del servidor es: <?php echo $_SERVER['SERVER_NAME']; ?>
10 </body>
11 </html>
```

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

Los servlets de Java EE se diferencian de las páginas JSP en que los primeros son programas Java compilados y almacenados en el contenedor de servlets. Sin embargo, las páginas JSP contienen código Java embebido en lenguaje HTML y se almacenan de forma individual en el servidor web. 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 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 cierta independencia entre el aspecto de la aplicación y la gestión del contenido dinámico. A partir de esos ficheros se obtiene un código intermedio (MSIL en la terminología de la plataforma) que es lo que almacena el servidor.

La relación entre la forma de ejecución de un lenguaje, y el método para integrarse con las etiquetas HTML de una página web es:

- ☐ Si el lenguaje integra en su código etiquetas HTML, entonces se trata de un lenguaje de guiones.
- ☐ Si las instrucciones del lenguaje se integran dentro de las etiquetas HTML de una página web, entonces se trata de un lenguaje de guiones.
- ☐ Si las instrucciones del lenguaje se integran dentro de las etiquetas HTML de una página web, entonces se trata de un lenguaje compilado.
- ☒ **Es indistinto, no hay una relación directa.**

No existe una relación directa entre la forma en que se ejecuta un lenguaje, y cómo se integra con las etiquetas HTML de una página web.

3.2.- 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 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. 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.

Los dos IDE de código abierto más utilizados en la actualidad son Eclipse y NetBeans. Ambos permiten el desarrollo de aplicaciones informáticas en varios lenguajes de programación. 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.

Vamos a ver cómo instalar el IDE NetBeans en un sistema Linux. Lo primero que debes hacer es comprobar que tienes la última versión de Java, pues tanto NetBeans como Eclipse necesitan de la máquina virtual de Java para ejecutarse. Para ello, desde una consola escribe:

```
java -version
```

En caso de no tener Java, deberemos instalarlo antes del IDE. Si estamos trabajando en Ubuntu, lo primero sería añadir previamente el repositorio necesario. Por ejemplo, si tu versión de Ubuntu es la 10.04:

```
sudo add-apt-repository "deb http://archive.canonical.com/ lucid partner"
```

O si fuera la 10.10:

```
sudo add-apt-repository "deb http://archive.canonical.com/ maverick partner"
```

Una vez añadido el repositorio, actualiza la lista de paquetes e instala Java:

```
sudo apt-get update
sudo apt-get install sun-java-jdk
```

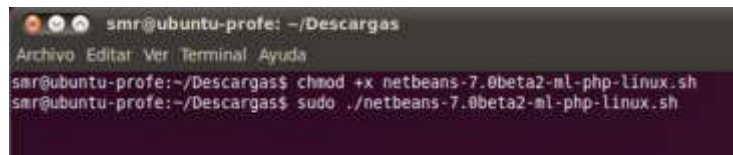
Mientras instalas Java (o en el caso de que ya lo tuvieras instalado), puedes aprovechar para descargar NetBeans desde su página oficial. En la página de descarga puedes escoger el idioma, la plataforma, y el paquete concreto.



Escoge el paquete en español, para sistemas Linux (x86/x64) y lenguaje de programación PHP.

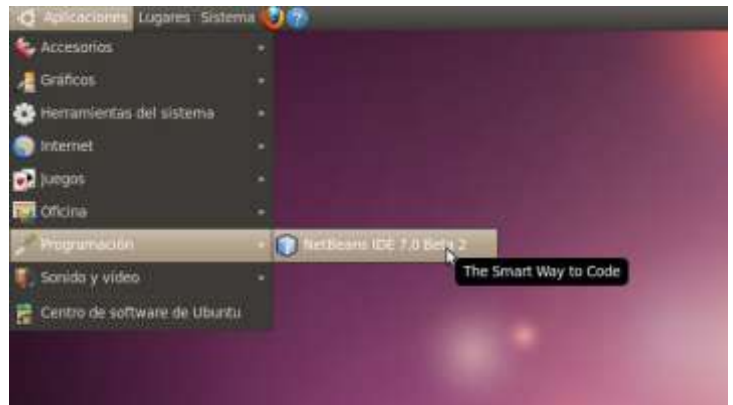
<http://www.netbeans.org/>

Al finalizar la instalación de Java, puedes empezar con la de NetBeans. Debes marcar el archivo como ejecutable y abrirlo desde la línea de comandos.



```
smr@ubuntu-profe: ~/Descargas
Archivo Editar Ver Terminal Ayuda
smr@ubuntu-profe:~/Descargas$ chmod +x netbeans-7.0beta2-ml-php-linux.sh
smr@ubuntu-profe:~/Descargas$ sudo ./netbeans-7.0beta2-ml-php-linux.sh
```

No es necesario modificar ningún parámetro durante el proceso de instalación. Una vez finalizado, puedes acceder a NetBeans mediante una nueva entrada que se ha generado en el menú.



3.2.2.- Instalación de una plataforma LAMP en Ubuntu.

Una vez instalado el entorno de desarrollo, es necesario instalar el resto de la plataforma de desarrollo. Si vas a programar en lenguaje PHP, necesitas todos los componentes de una arquitectura LAMP; recuerda:

- ✓ **L** de **Linux**, el sistema operativo.
- ✓ **A** de **Apache**, el servidor web.
- ✓ **M** de **MySQL**, el gestor de bases de datos.
- ✓ **P** del lenguaje de programación, que puede ser PHP, Perl o Python. Vamos a instalar el más común de estos tres, **PHP**.

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. Entre las tareas que incluye `tasksel` se encuentra `lamp-server`, que incorpora los componentes de una arquitectura LAMP antes mencionados. Para instalar LAMP ejecuta desde una consola:

```
sudo tasksel install lamp-server
```

Durante el proceso de instalación tendrás que introducir la contraseña para el usuario administrador (root) de MySQL.



Con la configuración predeterminada, la carpeta raíz de dónde el servidor web Apache extrae los documentos que va a publicar es `/var/www`. Por tanto, para probar el correcto funcionamiento de la plataforma puedes crear en ese directorio 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. Si todo va bien, al abrir esa página con un navegador web (desde la propia máquina de Ubuntu la URL será <http://localhost/prueba.php>) deberás ver algo como lo



siguiente:

Si surgen problemas durante la instalación de la plataforma de desarrollo en Ubuntu, o una vez instalada no funciona correctamente, en esta página puedes consultar más detalles sobre el proceso de instalación de LAMP.

<http://netbeans.org/kb/docs/php/configure-php-environment-ubuntu.html>

¿Cuál de los siguientes elementos no es necesario para programar aplicaciones web en lenguaje PHP?

- ☐ Un servidor web.
- ☐ Un sistema operativo.
- ☐ El lenguaje de programación PHP.
- ☒ **Un entorno integrado de desarrollo.**

Efectivamente es correcto, no es imprescindible el uso de un entorno integrado de desarrollo (IDE) para programar, aunque sí muy útil.

3.3.- Programación web con Java.

Java es el lenguaje de programación más utilizado hoy en día. Es un lenguaje orientado a objetos, basado en la sintaxis de C y C++ y eliminando algunas características de éstos que daban lugar a errores de programación, como los punteros. Todo el código que escribas en Java debe pertenecer a una clase.



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.

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 `%>` de la siguiente manera:

O también utilizando el método `println` dentro de un servlet como el siguiente, que obtiene los valores a sumar de otra página:

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.

```

1 public class Page {
2     public Page(int p, String t) {
3         this.p = p;
4         this.t = t;
5     }
6 }
7
8 public class Page2 {
9     public Page2(int p, String t) {
10         this.p = p;
11         this.t = t;
12     }
13 }
14
15 public class Page3 {
16     public Page3(int p, String t) {
17         this.p = p;
18         this.t = t;
19     }
20 }
21
22 public class Page4 {
23     public Page4(int p, String t) {
24         this.p = p;
25         this.t = t;
26     }
27 }
28
29 public class Page5 {
30     public Page5(int p, String t) {
31         this.p = p;
32         this.t = t;
33     }
34 }
35
36 public class Page6 {
37     public Page6(int p, String t) {
38         this.p = p;
39         this.t = t;
40     }
41 }
42
43 public class Page7 {
44     public Page7(int p, String t) {
45         this.p = p;
46         this.t = t;
47     }
48 }
49
50 public class Page8 {
51     public Page8(int p, String t) {
52         this.p = p;
53         this.t = t;
54     }
55 }
56
57 public class Page9 {
58     public Page9(int p, String t) {
59         this.p = p;
60         this.t = t;
61     }
62 }
63
64 public class Page10 {
65     public Page10(int p, String t) {
66         this.p = p;
67         this.t = t;
68     }
69 }
70
71 public class Page11 {
72     public Page11(int p, String t) {
73         this.p = p;
74         this.t = t;
75     }
76 }
77
78 public class Page12 {
79     public Page12(int p, String t) {
80         this.p = p;
81         this.t = t;
82     }
83 }
84
85 public class Page13 {
86     public Page13(int p, String t) {
87         this.p = p;
88         this.t = t;
89     }
90 }
91
92 public class Page14 {
93     public Page14(int p, String t) {
94         this.p = p;
95         this.t = t;
96     }
97 }
98
99 public class Page15 {
100     public Page15(int p, String t) {
101         this.p = p;
102         this.t = t;
103     }
104 }
105
106 public class Page16 {
107     public Page16(int p, String t) {
108         this.p = p;
109         this.t = t;
110     }
111 }
112
113 public class Page17 {
114     public Page17(int p, String t) {
115         this.p = p;
116         this.t = t;
117     }
118 }
119
120 public class Page18 {
121     public Page18(int p, String t) {
122         this.p = p;
123         this.t = t;
124     }
125 }
126
127 public class Page19 {
128     public Page19(int p, String t) {
129         this.p = p;
130         this.t = t;
131     }
132 }
133
134 public class Page20 {
135     public Page20(int p, String t) {
136         this.p = p;
137         this.t = t;
138     }
139 }
140
141 public class Page21 {
142     public Page21(int p, String t) {
143         this.p = p;
144         this.t = t;
145     }
146 }
147
148 public class Page22 {
149     public Page22(int p, String t) {
150         this.p = p;
151         this.t = t;
152     }
153 }
154
155 public class Page23 {
156     public Page23(int p, String t) {
157         this.p = p;
158         this.t = t;
159     }
160 }
161
162 public class Page24 {
163     public Page24(int p, String t) {
164         this.p = p;
165         this.t = t;
166     }
167 }
168
169 public class Page25 {
170     public Page25(int p, String t) {
171         this.p = p;
172         this.t = t;
173     }
174 }
175
176 public class Page26 {
177     public Page26(int p, String t) {
178         this.p = p;
179         this.t = t;
180     }
181 }
182
183 public class Page27 {
184     public Page27(int p, String t) {
185         this.p = p;
186         this.t = t;
187     }
188 }
189
190 public class Page28 {
191     public Page28(int p, String t) {
192         this.p = p;
193         this.t = t;
194     }
195 }
196
197 public class Page29 {
198     public Page29(int p, String t) {
199         this.p = p;
200         this.t = t;
201     }
202 }
203
204 public class Page30 {
205     public Page30(int p, String t) {
206         this.p = p;
207         this.t = t;
208     }
209 }
210
211 public class Page31 {
212     public Page31(int p, String t) {
213         this.p = p;
214         this.t = t;
215     }
216 }
217
218 public class Page32 {
219     public Page32(int p, String t) {
220         this.p = p;
221         this.t = t;
222     }
223 }
224
225 public class Page33 {
226     public Page33(int p, String t) {
227         this.p = p;
228         this.t = t;
229     }
230 }
231
232 public class Page34 {
233     public Page34(int p, String t) {
234         this.p = p;
235         this.t = t;
236     }
237 }
238
239 public class Page35 {
240     public Page35(int p, String t) {
241         this.p = p;
242         this.t = t;
243     }
244 }
245
246 public class Page36 {
247     public Page36(int p, String t) {
248         this.p = p;
249         this.t = t;
250     }
251 }
252
253 public class Page37 {
254     public Page37(int p, String t) {
255         this.p = p;
256         this.t = t;
257     }
258 }
259
260 public class Page38 {
261     public Page38(int p, String t) {
262         this.p = p;
263         this.t = t;
264     }
265 }
266
267 public class Page39 {
268     public Page39(int p, String t) {
269         this.p = p;
270         this.t = t;
271     }
272 }
273
274 public class Page40 {
275     public Page40(int p, String t) {
276         this.p = p;
277         this.t = t;
278     }
279 }
280
281 public class Page41 {
282     public Page41(int p, String t) {
283         this.p = p;
284         this.t = t;
285     }
286 }
287
288 public class Page42 {
289     public Page42(int p, String t) {
290         this.p = p;
291         this.t = t;
292     }
293 }
294
295 public class Page43 {
296     public Page43(int p, String t) {
297         this.p = p;
298         this.t = t;
299     }
300 }
301
302 public class Page44 {
303     public Page44(int p, String t) {
304         this.p = p;
305         this.t = t;
306     }
307 }
308
309 public class Page45 {
310     public Page45(int p, String t) {
311         this.p = p;
312         this.t = t;
313     }
314 }
315
316 public class Page46 {
317     public Page46(int p, String t) {
318         this.p = p;
319         this.t = t;
320     }
321 }
322
323 public class Page47 {
324     public Page47(int p, String t) {
325         this.p = p;
326         this.t = t;
327     }
328 }
329
330 public class Page48 {
331     public Page48(int p, String t) {
332         this.p = p;
333         this.t = t;
334     }
335 }
336
337 public class Page49 {
338     public Page49(int p, String t) {
339         this.p = p;
340         this.t = t;
341     }
342 }
343
344 public class Page50 {
345     public Page50(int p, String t) {
346         this.p = p;
347         this.t = t;
348     }
349 }
350
351 public class Page51 {
352     public Page51(int p, String t) {
353         this.p = p;
354         this.t = t;
355     }
356 }
357
358 public class Page52 {
359     public Page52(int p, String t) {
360         this.p = p;
361         this.t = t;
362     }
363 }
364
365 public class Page53 {
366     public Page53(int p, String t) {
367         this.p = p;
368         this.t = t;
369     }
370 }
371
372 public class Page54 {
373     public Page54(int p, String t) {
374         this.p = p;
375         this.t = t;
376     }
377 }
378
379 public class Page55 {
380     public Page55(int p, String t) {
381         this.p = p;
382         this.t = t;
383     }
384 }
385
386 public class Page56 {
387     public Page56(int p, String t) {
388         this.p = p;
389         this.t = t;
390     }
391 }
392
393 public class Page57 {
394     public Page57(int p, String t) {
395         this.p = p;
396         this.t = t;
397     }
398 }
399
400 public class Page58 {
401     public Page58(int p, String t) {
402         this.p = p;
403         this.t = t;
404     }
405 }
406
407 public class Page59 {
408     public Page59(int p, String t) {
409         this.p = p;
410         this.t = t;
411     }
412 }
413
414 public class Page60 {
415     public Page60(int p, String t) {
416         this.p = p;
417         this.t = t;
418     }
419 }
420
421 public class Page61 {
422     public Page61(int p, String t) {
423         this.p = p;
424         this.t = t;
425     }
426 }
427
428 public class Page62 {
429     public Page62(int p, String t) {
430         this.p = p;
431         this.t = t;
432     }
433 }
434
435 public class Page63 {
436     public Page63(int p, String t) {
437         this.p = p;
438         this.t = t;
439     }
440 }
441
442 public class Page64 {
443     public Page64(int p, String t) {
444         this.p = p;
445         this.t = t;
446     }
447 }
448
449 public class Page65 {
450     public Page65(int p, String t) {
451         this.p = p;
452         this.t = t;
453     }
454 }
455
456 public class Page66 {
457     public Page66(int p, String t) {
458         this.p = p;
459         this.t = t;
460     }
461 }
462
463 public class Page67 {
464     public Page67(int p, String t) {
465         this.p = p;
466         this.t = t;
467     }
468 }
469
470 public class Page68 {
471     public Page68(int p, String t) {
472         this.p = p;
473         this.t = t;
474     }
475 }
476
477 public class Page69 {
478     public Page69(int p, String t) {
479         this.p = p;
480         this.t = t;
481     }
482 }
483
484 public class Page70 {
48
```

[illegible]

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.

3.4.- 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 está basada en la de C / C++, y por lo tanto es muy similar a la de Java. Aunque lo puedes hacer de otras formas, 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 (normalmente utilizando Apache con el módulo `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`. Este fichero, `php.ini`, puede encontrarse en distintas ubicaciones. La función `phpinfo()` que ejecutaste antes te informa, entre otras muchas cosas, del lugar en que se encuentra almacenado el fichero `php.ini` en tu ordenador. 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 como es nuestro caso, utilizamos `mod_php` para ejecutar PHP como un módulo de Apache, las opciones de configuración de PHP se aplicarán cada vez que se reinicie Apache. Por tanto, no te olvides de hacerlo cada vez que hagas cambios en `php.ini`. Por ejemplo: `sudo /etc/init.d/apache2 restart` o también `sudo service apache2 restart`.

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 `?>`. Es preferible no usarlos, pues puede causarnos problemas si utilizamos páginas con XML. Para prohibir la utilización de estos delimitadores con PHP le asignamos a esta directiva el valor Off.
- ✓ **max_execution_time**. Permite que puedas ajustar el número máximo de segundos que podrá durar la ejecución de un script PHP. Evita que el servidor se bloquee si se produce algún error en un script.
- ✓ **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`.

Iremos viendo otras directivas a lo largo del módulo cuando las vayamos necesitando. Realiza la siguiente actividad para comprobar si recuerdas las que acabamos de ver.

Teclea el nombre de las directivas de configuración de PHP	
Indica si se pueden o no subir ficheros al servidor por HTTP.	<code>file_uploads</code>
Permite ajustar los segundos que podrá durar la ejecución de un script.	<code>max_execution_time</code>
Indica si se pueden utilizar en PHP los delimitadores cortos <code><?</code> y <code>?></code> .	<code>short_open_tags</code>
Indica qué tipo de errores se mostrarán en el caso de que se produzcan.	<code>error_reporting</code>

En la documentación de PHP se incluye una lista completa de las directivas que se pueden utilizar en `php.ini`.

<http://php.net/manual/es/ini.core.php>

A medida que vayas escribiendo código en PHP, será útil que introduzcas en el mismo algunos comentarios que ayuden a revisarlo cuando lo necesites. 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 `//`. Son comentarios al estilo del lenguaje C. Cuando una línea comienza por los símbolos `//`, toda ella se considera que contiene un comentario, hasta la siguiente línea.
- ✓ Comentarios de una línea utilizando `#`. Son similares a los anteriores, pero utilizando la sintaxis de los scripts de Linux.
- ✓ Comentarios de varias líneas. También iguales a los del lenguaje C. Cuando en una línea aparecen los caracteres `/*`, se considera que ahí comienza un comentario. El comentario puede extenderse varias líneas, y acabará cuando escribas la combinación de caracteres opuesta: `*/`.

Recuerda que cuando pongas comentarios al código PHP, éstos no figurarán en ningún caso en la página web que se envía al navegador (justo al contrario de lo que sucede con los comentarios a las etiquetas HTML).

Parámetro	Relación	Finalidad
<code>file_uploads</code>	1	1. Indica si se pueden subir ficheros al servidor web.
<code>max_execution_time</code>	4	2. Indica qué tipo de delimitadores se pueden usar para el código PHP.
<code>upload_max_filesize</code>	3	3. Limita el tamaño máximo de los ficheros que se suben al servidor.
<code>short_open_tags</code>	2	4. Limita el tiempo máximo de ejecución de un guión PHP.

3.4.1.- Variables y tipos de datos en PHP.

Como en todos los lenguajes de programación, en PHP puedes crear variables para almacenar valores. 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, al contrario que en muchos otros lenguajes, 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 `=`.

```
$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. Esto es, el tipo de la variable se decide en función del contexto en que se emplee.

```
// Al asignarle el valor 7, la variable es de tipo "entero"
$mi_variable = 7;
// Si le cambiamos el contenido
$mi_variable = "siete";
// La variable puede cambiar de tipo
// En este caso pasa a ser de tipo "cadena"
```

Los tipos de datos simples en PHP son:

- ✓ **booleano** (`boolean`). Sus posibles valores son `true` y `false`. Además, cualquier número entero se considera como `true`, salvo el 0 que es `false`.

- ✓ **entero** (`integer`). Cualquier número sin decimales. Se pueden representar en formato decimal, octal (comenzando por un 0), o hexadecimal (comenzando por 0x).
- ✓ **real** (`float`). Cualquier número con decimales. Se pueden representar también en notación científica.
- ✓ **cadena** (`string`). Conjuntos de caracteres delimitados por comillas simples o dobles.
- ✓ **null**. Es un tipo de datos especial, que se usa para indicar que la variable no tiene valor.

Por ejemplo:

```
$mi_booleano = false;
$mi_entero = 0x2A;
$mi_real = 7.3e-1;
$mi_cadena = "texto";
$mi_variable = Null;
```

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:

```
$mi_entero = 3;
$mi_real = 2.3;
$resultado = $mi_entero + $mi_real;
// La variable $resultado es de tipo real
```

Estas conversiones de tipo, que en el ejemplo anterior se lleva a cabo de forma automática, también se pueden realizar de forma forzada:

```
$mi_entero = 3;
$mi_real = 2.3;
$resultado = $mi_entero + (int) $mi_real;
// La variable $mi_real se convierte a entero (valor 2) antes de sumarse.
// La variable $resultado es de tipo entero (valor 5)
```

En la documentación de PHP se especifican las conversiones de tipo posibles y los resultados obtenidos con cada una:

<http://www.php.net/manual/es/language.types.type-juggling.php#language.types.typecasting>

3.4.2.- Expresiones y operadores.

Como en muchos otros lenguajes, 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. Por ejemplo:

```
$mi_variable = 7;
$a = $b + $c;
$valor++;
$x += 5;
```

Los operadores que puedes usar en PHP son similares a los de muchos otros lenguajes como Java. Entre ellos tenemos operadores para:

- ✓ Realizar operaciones aritméticas: negación, suma, resta, multiplicación, división y módulo. Entre éstos se incluyen operadores de pre y post incremento y decremento, `++` y `--`.

Estos operadores incrementan o decrementan el valor del operando al que se aplican. Si se utilizan junto a una expresión de asignación, modifican el operando antes o después de la asignación en función de su posición (antes o después) con respecto al operando. Por ejemplo:

```
$a = 5;
$b = ++$a;
// Antes se le suma uno a $a (pasa a tener valor 6)
// y después se asigna a $b (que también acaba con valor 6).
$a = 5;
$b = $a++;
```



```
// Antes se asigna a $b el valor de $a (5).
// y después se le suma uno a $a (pasa a tener valor 6)
```

- ✓ Realizar asignaciones. Además del operador `=`, existen operadores con los que realizar operaciones y asignaciones en un único paso (`+=`, `-=`,...).
- ✓ Comparar operandos. Además de los que nos podemos encontrar en otros lenguajes (`<`, `>`,...), en PHP tenemos dos operadores para comprobar igualdad (`==`, `===`) y tres para comprobar diferencia (`<>`, `!=` y `!==`).

Los operadores `<>` y `!=` son equivalentes. Comparan los valores de los operandos.

El operador `===` devuelve verdadero (`true`) sólo si los operandos son del mismo tipo y además tienen el mismo valor. El operador `!==` devuelve verdadero (`true`) si los valores de los operandos son distintos o bien si éstos no son del mismo tipo. Por ejemplo:

```
$x = 0;
// La expresión $x == false es cierta (devuelve true).
// Sin embargo, $x === false no es cierta (devuelve false) pues $x es de tipo entero, no booleano.
```

- ✓ Comparar expresiones booleanas. Tratan a los operandos como variables booleanas (`true` o `false`). Existen operadores para realizar un Y lógico (operadores `and` o `&&`), O lógico (operadores `or` o `||`), No lógico (operador `!`) y O lógico exclusivo (operador `xor`).
- ✓ Realizar operaciones con los bits que forman un entero: invertirlos, desplazarlos, etc.

En la documentación de PHP se explican en detalle todos los operadores disponibles en el lenguaje y la forma en que se utilizan:

<http://www.php.net/manual/es/language.operators.php>

3.4.3.- Ámbito de utilización de las variables.

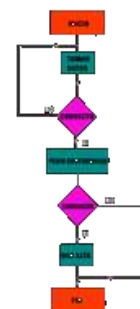
En PHP puedes utilizar variables en cualquier lugar de un programa. Si esa variable aún no existe, la primera vez que se utiliza se reserva espacio para ella. En ese momento, dependiendo del lugar del código en que aparezca, se decide desde qué partes del programa se podrá utilizar esa variable. A esto se le llama visibilidad de la variable.

Si la variable aparece por primera vez dentro de una función, se dice que esa variable es local a la función. Si aparece una asignación fuera de la función, se le considerará una variable distinta. Por ejemplo:

```
$a = 1;
function prueba()
{
    // Dentro de la función no se tiene acceso a la variable $a anterior
    $b = $a;
    // Por tanto, la variable $a usada en la asignación anterior es una variable nueva
    // que no tiene valor asignado (su valor es null)
}
```

Si en la función anterior quisieras utilizar la variable `$a` externa, podrías hacerlo utilizando la palabra `global`. De esta forma le dices a PHP que no cree una nueva variable local, sino que utilice la ya existente.

```
$a = 1;
function prueba()
{
    global $a;
    $b = $a;
    // En este caso se le asigna a $b el valor 1
}
```



Las variables locales a una función desaparecen cuando acaba la función y su valor se pierde. Si quisieras mantener el valor de una variable local entre distintas llamadas a la función, deberás declarar la variable como estática utilizando la palabra `static`.

```
function contador()  
{  
    static $a=0;  
    $a++;  
    // Cada vez que se ejecuta la función, se incrementa el valor de $a  
}
```

Las variables estáticas deben inicializarse en la misma sentencia en que se declaran como estáticas. De esta forma, se inicializan sólo la primera vez que se llama a la función.

Puedes consultar más ejemplos sobre los ámbitos de las variables en la documentación de PHP.

<http://www.php.net/manual/es/language.variables.scope.php>

Si hacemos `$a=1`, ¿cuál de las siguientes comparaciones es verdadera?

- ☐ `"1" === $a`
- ☐ `$a == false`
- ☐ `a$++ == 2`
- ☒ `--$a == false`

Antes de hacer la comparación, se decrementa `$a`, con lo cual su valor pasa a ser cero. Y en PHP, cero es equivalente a `false`.