

# Plataformas de programación web en entorno servidor. Características del lenguaje PHP.

## Caso práctico



En la empresa **BK Programación** están a punto de hacerse cargo de un importante proyecto. **Ada**, la directora, va a comprometerse con un cliente y amigo, **Esteban**, que necesita ayuda con un problema concreto.

### **una pequeña empresa**

cámaras, televisores, material informático, etc. Con el tiempo, esa pequeña empresa ha crecido. El número de ventas ha aumentado, así como el catálogo de productos que ofrece, e incluso ha abierto dos nuevas tiendas en localidades cercanas.

Pero como sucede muchas veces, **al aumentar el negocio** han ido surgiendo ciertas **necesidades**. El proyecto que Esteban le ha propuesto a Ada consiste en **desarrollar una web**. No una página web que explique dónde está la empresa o qué hace. Quiere una web enfocada a dos temas concretos: mejorar la **comunicación con sus clientes**, y que le aporte **información interna** a la empresa sobre su negocio.

Por ejemplo, quiere que los clientes puedan ver desde su casa los productos que vende, el precio de los mismos, o la disponibilidad en una u otra tienda. O que los empleados de la propia empresa puedan ver de forma sencilla el stock que tienen de los productos en las distintas sucursales, para poder decidir mejor qué productos se piden a los distribuidores y en qué cantidad.

Sin embargo, tal y como Ada ya le ha comentado a Esteban, la experiencia de BK Programación en el desarrollo de aplicaciones web es muy reducida. La mayoría de proyectos que han realizado hasta el momento se han centrado en aplicaciones para plataformas Windows y Linux, o para dispositivos móviles. Sólo uno de sus empleados, **Juan**, tiene cierta experiencia con la **programación web**.

Aun así, Esteban confía en que Ada y su empresa lograrán llevar a cabo el proyecto. No tiene prisa por ponerlo en funcionamiento, y sabe que a BK Programación le servirá para ir formando a sus empleados en temas relacionados en el desarrollo de aplicaciones web, por lo que finalmente llegan a un acuerdo.

# 1.- Características de la programación web.

## Caso práctico



El nuevo proyecto al que se enfrenta **BK Programación** requiere que sus **empleados actualicen su formación** en temas relacionados con la **programación web**. **Juan**, el único con cierta experiencia en ese ámbito, ha **propuesto** a sus compañeros la realización de unas **jornadas** en las que explique a los demás los **fundamentos del desarrollo** de aplicaciones para la **web**. De esta forma, aquellos que quieran pasarán a formar parte del equipo que se dedique al nuevo proyecto.

**Todos** se apuntan a la **propuesta** de Juan, **incluyendo a Ada**, la directora.

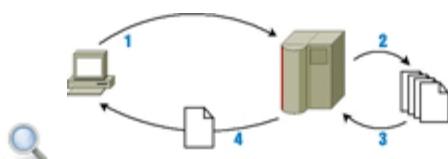
El **primer objetivo** de las jornadas es ver en qué consiste la programación web. Por ejemplo, ver la **diferencia** existente entre hacer una **página web** y **programar** una **aplicación web**.

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**. 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.

## Autoevaluación

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

- Sí.
- No.

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

---

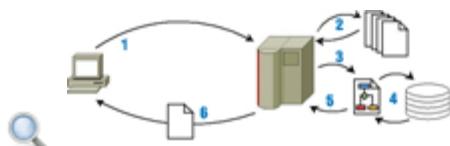
Las páginas que viste en el ejemplo anterior se llaman **páginas web estáticas**. Estas páginas se encuentran almacenadas en su forma definitiva, tal y como se crearon, y su contenido no varía. Son útiles para mostrar una información concreta, y mostrarán esa misma información cada vez que se carguen.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

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 quieras 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 quieras dar de alta un sitio que has creado en un motor de búsqueda como Google.

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

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

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

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

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

## Reflexiona

¿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?

## **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 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:

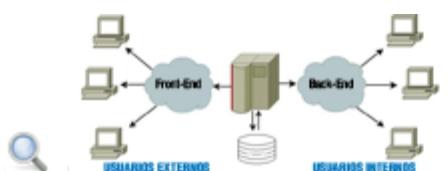
- ✓ **No es necesario instalarlas en aquellos equipos en que se vayan a utilizar.** Se instalan y se ejecutan solamente en un equipo, en el servidor, y esto es suficiente para que se puedan utilizar de forma simultánea desde muchos equipos.
- ✓ Como solo se encuentran instaladas en un equipo, **es muy 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.



# Autoevaluación

**¿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.

## 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 quieras 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 quieras 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.

## Autoevaluación

**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.

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

### Caso práctico



En **BK Programación** han formado un **equipo** que se dedicará al nuevo proyecto. **Juan** será el **jefe del proyecto**, y contará con la **ayuda de María**, que trabaja habitualmente en la instalación y mantenimiento de servidores, y con **Carlos**, un amigo de Juan que pese a **no tener experiencia** en ese tipo de trabajo se siente muy atraído por todo lo relacionado con la programación web.

**Juan ha programado** aplicaciones en lenguajes C, Java y PHP. **María** por su parte **conoce el lenguaje C** y utiliza Perl en su trabajo habitual. **Carlos es autodidacta** y tiene nociones de programación en lenguaje Pascal. **Todos** conocen bien el lenguaje **HTML**.

En la primera reunión, los tres ponen en común los objetivos generales del trabajo y deciden estudiar las distintas opciones que tienen para lograr el objetivo.

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.

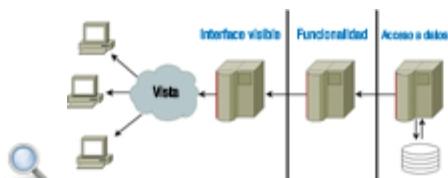
En este bloque **vas a aprender las distintas tecnologías** que se pueden utilizar para **programar aplicaciones** que se ejecuten en un **servidor web**, y cómo se relacionan unas con otras. Verás las ventajas e inconvenientes de utilizar cada una, y qué lenguajes de programación deberás aprender para 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 obtiene e introduce datos en el espacio de almacenamiento). Cada capa puede ocuparse de una o varias de las funciones anteriores. Por ejemplo, en las aplicaciones de **3 capas** nos podemos encontrar con:

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

## 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, siendo PHP el más empleado de los tres.

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.



### Para saber más

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.

[XAMPP](#)

- ✓ 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.

## 2.2.- Integración con el servidor web.

---

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 (o HTTPS si utiliza el protocolo seguro).

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.

## Para saber más

Puedes consultar una lista con los servidores de aplicaciones Java EE certificados por Sun en la Wikipedia, y las implementaciones de software compatibles con JEE 6

[Servidores de aplicaciones Java EE certificados por Sun](#)

[Implementaciones de software compatibles con JEE 6](#)

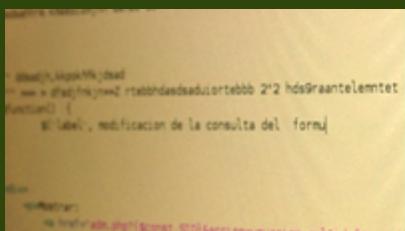
Sin embargo, en la mayoría de ocasiones no es necesario utilizar un servidor de aplicaciones completo, especialmente si no utilizamos EJB en nuestras aplicaciones, sino que nos será suficiente un contenedor de servlets. En esta área, destaca Tomcat, la implementación por referencia de un contenedor de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

### Caso práctico



Tras analizar distintas arquitecturas de programación web, Juan y su equipo comprueban que una de las decisiones más importantes antes de ponerse manos a la obra es el lenguaje de programación que utilizarán en el desarrollo.

Además de la sintaxis propia de cada lenguaje, la elección de uno u otro afecta a la complejidad del proyecto, a las herramientas que tendrán que utilizar, e incluso a la forma en que tendrán que programar la aplicación web.

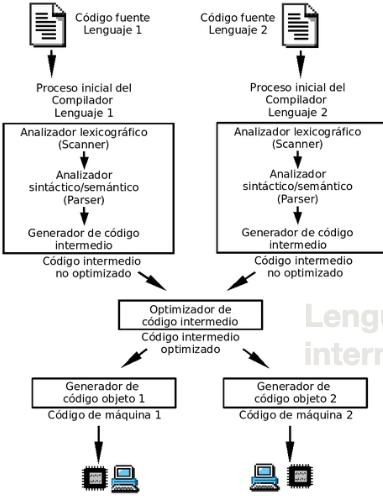
Juan propone utilizar el lenguaje PHP, que ya conoce, pero no sin antes estudiar las ventajas y desventajas que les supondría con respecto a la utilización de otras alternativas.

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 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**. 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.



todo principal para ejecutar mas binarios desde un servidor web es CGI. Utilizando CGI podemos hacer que el servidor web ejecute código programado en un lenguaje de propósito general como puede ser C.

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:

  
A screenshot of a web browser window. The address bar shows 'http://www.google.es/search?hl=es&q=Prueba+CGI'. The search results page displays several links, with the first one being 'Prueba CGI - Wikipedia, la encyclopédie libre'. A magnifying glass icon is positioned over this link.

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.

# Autoevaluación

**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.

## **3.2.- Herramientas de programación.**

---



Junior Libby

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.

## Debes conocer

Aunque en otros módulos como Entornos de Desarrollo o Programación ya se ve el proceso de instalación de NetBeans al final de esta unidad tienes un Anexo con el proceso de **Instalación de NetBeans para Linux (Anexo XIII)**.

Otra herramienta que necesitarás instalar es una plataforma XAMP, del mismo modo al final de esta unidad dispones del **Anexo XIV** que te muestra la **Instalación de una plataforma LAMP en Linux**.

## 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 quieras 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.

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.**

---



[Imonk72](#)

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.



## Para saber más

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

[Lista completa de las directivas](#)

## Autoevaluación

**Relaciona cada parámetro de la configuración de PHP con su finalidad:**

### Ejercicio de relacionar

Parámetro	Relación	Finalidad
file_uploads	0	1. Indica si se pueden subir ficheros al servidor web.
max_execution_time	0	2. Indica qué tipo de delimitadores se pueden usar para el código PHP.
upload_max_filesize	0	3. Limita el tamaño máximo de los ficheros que se suben al servidor.
short_open_tag	0	4. Limita el tiempo máximo de ejecución de un guión PHP.

**Enviar**

## 4.- Características del lenguaje PHP.

### Caso práctico



Stockbyte. CD-DVD Num. CD109.

El nuevo proyecto va a ponerse en marcha. En **BK Programación** las tres personas asignadas comienzan los preparativos. **Juan, el jefe de proyecto**, está elaborando un calendario para intentar definir las distintas fases del desarrollo. **María**, en el tiempo que le puede dedicar, se ha puesto a refrescar sus conocimientos de **programación en PHP**. **Carlos** es el que más trabajo tiene por delante, sabe que si quiere aportar algo, debe aprender a programar aplicaciones web, y pronto.

**Carlos le pide ayuda a Juan**, que le orienta sobre los conceptos fundamentales del lenguaje y le ofrece su ayuda en todo lo que le sea posible. Sabe que es importante adquirir unos conocimientos sólidos antes de comenzar el desarrollo, para no cometer errores al principio que después sea complicado solucionar.

**Con la ayuda de María**, ponen en funcionamiento un **servidor de aplicaciones** dentro de la empresa. De momento lo utilizarán para ir haciendo pruebas, pero dentro de poco será la plataforma sobre la que programarán la nueva aplicación.

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 aparezcan 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).

## **4.1.- Elementos del lenguaje PHP.**

---

Ya sabes cómo se integran las etiquetas HTML con el código del lenguaje, utilizando los delimitadores **<?php** y **?>**.

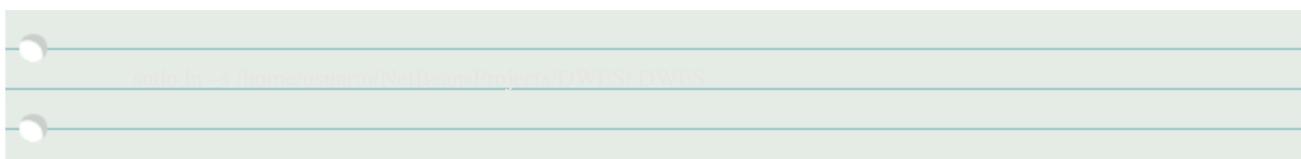
Ahora debes aprender a utilizar otros elementos del lenguaje que te permitan crear programas completos en PHP. Los programas escritos en PHP, además de encontrarse estructurados normalmente en varias páginas (ya veremos más adelante cómo se pueden comunicar datos de unas páginas a otras), suelen incluir en una misma página varios bloques de código. Cada bloque de código debe ir entre delimitadores, y en caso de que genere alguna salida, ésta se introduce en el código HTML en el mismo punto en el que figuran las instrucciones en PHP.

Por ejemplo, en las siguientes líneas tenemos dos bloques de código en PHP (delimitados por las etiquetas :

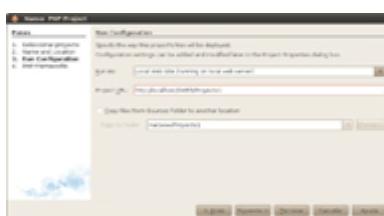
```
<body>  
<?php $a=1; ?>  
  
<p>Página de prueba</p>  
  
<?php $b=$a; ?>  
  
...  
...
```

Ahora empezarás a crear tus propios programas en PHP. Para ello vas a usar el IDE NetBeans. Deberías organizar tus programas en proyectos, y almacenarlos en una estructura en árbol, colgando todos por ejemplo de /home/usuario/NetBeansProjectos/DWES. Para crear un proyecto nuevo vete a **Archivo – Proyecto Nuevo** y selecciona **PHP Application**.

En la siguiente pantalla de configuración del proyecto, debes indicar la ruta que se usará para acceder al mismo desde un navegador. Es decir, tienes que hacer que el servidor web Apache pueda acceder a la ruta anterior en la que vas a almacenar tus proyectos. Esto puedes hacerlo, por ejemplo, creando un  enlace simbólico en la raíz del servidor web (/var/www):



De esta forma, si creas un proyecto nuevo en la ruta /home/usuario/NetBeansProjectos/DWES/Proyecto1, la URL que tendrás que poner en la siguiente pantalla de configuración será <http://localhost/DWES/Proyecto1>.



Elaboración Propia

## 4.1.1.- 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 asignale 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.



Piotr Siedlecki

// Al asignarle el valor 7, la variable es de tipo “entero”

```
$mi_variable = 7;
```

// Si le cambiamos el contenido

```
$mi_variable = “siete”;
```

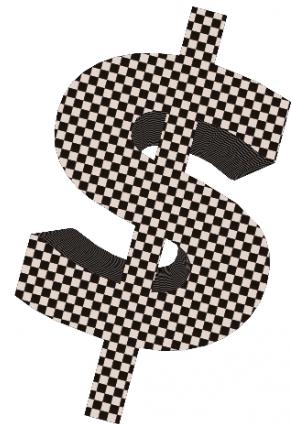
// La variable puede cambiar de tipo

// 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)
```

## **Debes conocer**

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

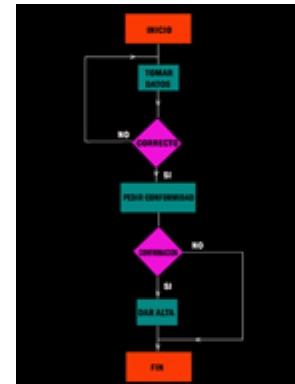
[Conversiones de tipos posibles y los resultados obtenidos](#)

## 4.1.2.- 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.

## Para saber más

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

[Ámbito de las variables](#)

# Autoevaluación

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

- "1" === \$a
- \$a == false
- \$a++ == 2
- \$a == false

## 4.1.3.- Variables especiales de PHP.

En la unidad anterior ya aprendiste qué eran y cómo se utilizaban las variables globales. PHP incluye unas cuantas variables internas predefinidas que pueden usarse desde cualquier ámbito, por lo que reciben el nombre de **variables superglobales**. Ni siquiera es necesario que uses **global** para acceder a ellas.

Cada una de estas variables es un array que contiene un conjunto de valores (en esta unidad veremos más adelante cómo se pueden utilizar los arrays). Las variables superglobales disponibles en PHP son las siguientes:



Elaboración Propia

**`$_SERVER`**. Contiene información sobre el entorno del servidor web y de ejecución. Entre la información que nos ofrece esta variable, tenemos:

### Principales valores de la variable `$_SERVER`

Valor	Contenido
<code>\$_SERVER['PHP_SELF']</code>	guión que se está ejecutando actualmente.
<code>\$_SERVER['SERVER_ADDR']</code>	dirección IP del servidor web.
<code>\$_SERVER['SERVER_NAME']</code>	nombre del servidor web.
<code>\$_SERVER['DOCUMENT_ROOT']</code>	directorio raíz bajo el que se ejecuta el guión actual.
<code>\$_SERVER['REMOTE_ADDR']</code>	dirección IP desde la que el usuario está viendo la página.
<code>\$_SERVER['REQUEST_METHOD']</code>	método utilizado para acceder a la página ('GET', 'HEAD', 'POST' o 'PUT')

### Debes conocer

En la documentación de PHP puedes consultar toda la información que ofrece `$_SERVER`:

[Información que ofrece `\$\_SERVER`.](#)

**`$_GET`, `$_POST` y `$_COOKIE`** contienen las variables que se han pasado al guión actual utilizando respectivamente los métodos GET (parámetros en la URL), HTTP POST y Cookies HTTP.

**`$_REQUEST`** junta en uno solo el contenido de los tres arrays anteriores, `$_GET`, `$_POST` y `$_COOKIE`.

**`$_ENV`** contiene las variables que se puedan haber pasado a PHP desde el entorno en que se ejecuta.

**`$_FILES`** contiene los ficheros que se puedan haber subido al servidor utilizando el método POST.

**`$_SESSION`** contiene las variables de sesión disponibles para el guión actual.

En posteriores unidades iremos trabajando con estas variables.

## Para saber más

Conviene tener a mano la información sobre estas variables superglobales disponible en el manual de PHP.

[Información sobre las variables superglobales.](#)

# Autoevaluación

Relaciona cada variable con la información que contiene:

Ejercicio para relacionar.

Parámetro	Relación	Finalidad
<code>\$_SERVER['DOCUMENT_ROOT']</code>	0	1. Variables de entorno disponibles.
<code>\$_ENV</code>	0	2. Guión que se está ejecutando.
<code>\$_SESSION</code>	0	3. Variables de sesión disponibles.
<code>\$_SERVER['PHP_SELF']</code>	0	4. Directorio raíz del guión actual.

Enviar

## 4.1.4.- 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:

$$2 + 2 = 4$$

```
$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.

## Debes conocer

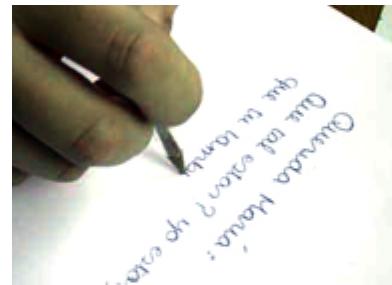
En la documentación de PHP se explican en detalle todos los operadores disponibles en el lenguaje y la forma en que se utilizan:

[Operadores disponibles en el lenguaje](#)

## 4.1.5.- Generación de código HTML.

Existen varias formas incluir contenido en la página web a partir del resultado de la ejecución de código PHP. La forma más sencilla es usando echo, que no devuelve nada (void), y genera como salida el texto de los parámetros que recibe.

```
void echo (string $arg1, ...);
```



Otra posibilidad es print, que funciona de forma similar. La diferencia más importante entre print y echo, es que print sólo puede recibir un parámetro y devuelve siempre 1.

```
int print (string $arg);
```

Tanto print como echo no son realmente funciones, por lo que no es obligatorio que pongas paréntesis cuando las utilices. Por ejemplo, el código del siguiente documento puede hacerse igualmente utilizando echo.

### Consulta el Anexo I: Utilización de la función print en PHP

printf es otra opción para generar una salida desde PHP. Puede recibir varios parámetros, el primero de los cuales es siempre una cadena de texto que indica el formato que se ha de aplicar. Esa cadena debe contener un especificador de conversión por cada uno de los demás parámetros que se le pasen a la función, y en el mismo orden en que figuran en la lista de parámetros. Por ejemplo:

```
<?php  
  
$ciclo="DAW";  
  
$modulo="DWES";  
  
print "<p>";  
  
printf("%s es un módulo de %d curso de %s", $modulo, 2, $ciclo);  
  
print "</p>";  
  
?>
```

Cada especificador de conversión va precedido del carácter % y se compone de las siguientes partes:

- ✓ **signo** (opcional). Indica si se pone signo a los números negativos (por defecto) o también a los positivos (se indica con un signo +).
- ✓ **relleno** (opcional). Indica que carácter se usará para ajustar el tamaño de una cadena. Las opciones son el carácter 0 o el carácter espacio (por defecto se usa el espacio).
- ✓ **alineación** (opcional). Indica qué tipo de alineación se usará para generar la salida: justificación derecha (por defecto) o izquierda (se indica con el carácter -).
- ✓ **ancho** (opcional). Indica el mínimo número de caracteres de salida para un parámetro dado.
- ✓ **precisión** (opcional). Indica el número de dígitos decimales que se mostrarán para un número real. Se escribe como un dígito precedido por un punto.
- ✓ **tipo** (obligatorio). Indica cómo se debe tratar el valor del parámetro correspondiente. En la siguiente tabla puedes ver una lista con todos los especificadores de tipo.

## Especificadores de tipo para las funciones printf y sprintf.

Especificador	Significado
<b>b</b>	el argumento es tratado como un entero y presentado como un número binario.
<b>c</b>	el argumento es tratado como un entero, y presentado como el carácter con dicho valor <u>ASCII</u> .
<b>d</b>	el argumento es tratado como un entero y presentado como un número decimal.
<b>u</b>	el argumento es tratado como un entero y presentado como un número decimal sin signo.
<b>o</b>	el argumento es tratado como un entero y presentado como un número octal.
<b>x</b>	el argumento es tratado como un entero y presentado como un número hexadecimal (con minúsculas).
<b>X</b>	el argumento es tratado como un entero y presentado como un número hexadecimal (con mayúsculas).
<b>f</b>	el argumento es tratado como un doble y presentado como un número de coma flotante (teniendo en cuenta la localidad).
<b>F</b>	el argumento es tratado como un doble y presentado como un número de coma flotante (sin tener en cuenta la localidad).
<b>e</b>	el argumento es presentado en notación científica, utilizando la e minúscula (por ejemplo, 1.2e+3).
<b>E</b>	el argumento es presentado en notación científica, utilizando la e mayúscula (por ejemplo, 1.2E+3).
<b>g</b>	se usa la forma más corta entre %f y %e.
<b>G</b>	se usa la forma más corta entre %f y %E.
<b>s</b>	el argumento es tratado como una cadena y es presentado como tal.
<b>%</b>	se muestra el carácter %. No necesita argumento..

Por ejemplo, al ejecutar la línea siguiente, en la salida el número PI se obtiene con signo y sólo con dos decimales.

```
printf("El número PI vale %+.2f", 3.1416); // +3.14
```

Existe una función similar a printf pero en vez de generar una salida con la cadena obtenida, permite guardarla en una variable: sprintf.

```
$txt_pi = sprintf("El número PI vale %+.2f", 3.1416);
```

## Autoevaluación

**Tenemos una variable real, y queremos mostrarla utilizando un número fijo de decimales, por ejemplo 3. ¿Podemos hacerlo sin utilizar la función printf?**

- No.
- Sí.

## 4.1.6.- Cadenas de texto.

En PHP las cadenas de texto pueden usar tanto comillas simples como comillas dobles. Sin embargo hay una diferencia importante entre usar unas u otras. Cuando se pone una variable dentro de unas comillas dobles, se procesa y se sustituye por su valor. Así, el ejemplo anterior sobre el uso de print también podía haberse puesto de la siguiente forma:

```
<?php  
$modulo="DWES";  
  
print "<p>Módulo: $modulo</p>"  
  
?>
```

La variable \$modulo se reconoce dentro de las comillas dobles, y se sustituye por el valor "DWES" antes de generar la salida. Si esto mismo lo hubieras hecho utilizando comillas simples, no se realizaría sustitución alguna.

Para que PHP distinga correctamente el texto que forma la cadena del nombre de la variable, a veces es necesario rodearla entre llaves.

```
print "<p>Módulo: ${modulo}</p>"
```

Cuando se usan comillas simples, sólo se realizan dos sustituciones dentro de la cadena: cuando se encuentra la secuencia de caracteres \' , se muestra en la salida una comilla simple; y cuando se encuentra la secuencia \\ , se muestra en la salida una barra invertida.

Estas secuencias se conocen como **secuencias de escape**. En las cadenas que usan comillas dobles, además de la secuencia \\ , se pueden usar algunas más, pero no la secuencia \'. En esta tabla puedes ver las secuencias de escape que se pueden utilizar, y cuál es su resultado.

reconoció esta oposición y la amplia con  
inámbica. Van Gogh lo formula en su au  
juego de puntos multicolores se extiende  
como envolviendo el vivo encarnado de  
de París desde la habitación de Vincent  
otro momento dentro de su programa a  
eran garantes de su dinámica pictórica, el c  
constituye la base de su estética según la  
es había captado el mar de casas de la g



idITE=111597.

## Secuencias de escape.

Secuencia	Resultado
\`	se muestra una barra invertida.
\'	se muestra una comilla simple.
\\"	se muestra una comilla doble.
\<code>n	se muestra un avance de línea (LF o 0x0A (10) en ASCII).
\r	se muestra un retorno de carro (CR o 0x0D (13) en ASCII).
\t	se muestra un tabulador horizontal (HT o 0x09 (9) en ASCII). <b>NOTA:</b> HTML convertirá el tabulador en espacio salvo si se mete entre etiquetas y ej: print "prueba prueba";
\v	se muestra un tabulador vertical (VT o 0x0B (11) en ASCII).
\f	se muestra un avance de página (FF o 0x0C (12) en ASCII).
\\$	se muestra un signo del dólar.

En PHP tienes dos operadores exclusivos para trabajar con cadenas de texto. Con el operador de concatenación punto (.) puedes unir las dos cadenas de texto que le pases como operandos. El operador de asignación y concatenación (.=) concatena al argumento del lado izquierdo la cadena del lado derecho.

```
<?php
$a = "Módulo ";
$b = $a . "DWES"; // ahora $b contiene "Módulo DWES"
$a .= "DWES"; // ahora $a también contiene "Módulo DWES"
?>
```



## 4.1.7.- Funciones relacionadas con los tipos de datos(I).

En PHP existen funciones específicas para comprobar y establecer el tipo de datos de una variable, `gettype` obtiene el tipo de la variable que se le pasa como parámetro y devuelve una cadena de texto, que puede ser **array**, **boolean**, **double**, **integer**, **object**, **string**, **null**, **resource** o **unknown**.

También podemos comprobar si la variable es de un tipo concreto utilizando una de las siguientes funciones: **is\_array()**, **is\_bool()**, **is\_float()**, **is\_integer()**, **is\_null()**, **is\_numeric()**, **is\_object()**, **is\_resource()**, **is\_scalar()** e **is\_string()**. Devuelven true si la variable es del tipo indicado.

Análogamente, para establecer el tipo de una variable utilizamos la función **settype** pasándole como parámetros la variable a convertir, y una de las siguientes cadenas: **boolean**, **integer**, **float**, **string**, **array**, **object** o **null**. La función **settype** devuelve true si la conversión se realizó correctamente, o false en caso contrario.

```
<?php  
  
$a = $b = "3.1416"; // asignamos a las dos variables la misma cadena de texto  
  
settype($b, "float"); // y cambiamos $b a tipo float  
  
print "$a vale $a y es de tipo ".gettype($a);  
  
print "<br />";  
  
print "$b vale $b y es de tipo ".gettype($b);  
  
?>
```

El resultado del código anterior es:

\$a vale 3.1416 y es de tipo string

\$b vale 3.1416 y es de tipo double



Si lo único que te interesa es saber si una variable está definida y no es **null**, puedes usar la función **isset**. La función **unset** destruye la variable o variables que se le pasa como parámetro.

```
<?php  
$a = "3.1416";  
  
if (isset($a)) // la variable $a está definida  
  
unset($a); //ahora ya no está definida  
  
?>
```

Es importante no confundir el que una variable esté definida o valga **null**, con que se considere como vacía debido al valor que contenga. Esto último es lo que nos indica la función **empty**.

#### Función empty.

Existe también en PHP una función, **define**, con la que puedes definir constantes, esto es, identificadores a los que se les asigna un valor que no cambia durante la ejecución del programa.

```
bool <strong>define</strong> ( string $identificador , mixed $valor [, bool $case_insensitive = false ] );
```

Los identificadores no van precedidos por el signo "\$" y suelen escribirse en mayúsculas, aunque existe un tercer parámetro opcional, que si vale true hace que se reconozca el identificador independientemente de si está escrito en mayúsculas o en minúsculas.

```
<?php  
  
define ("PI", 3.1416, true);  
  
print "El valor de PI es ".pi; //El identificador se reconoce tanto por PI como por pi  
  
?>
```

Sólo se permiten los siguientes tipos de valores para las constantes: **integer**, **float**, **string**, **boolean** y **null**.

# Autoevaluación

¿Qué se muestra en pantalla al ejecutar el siguiente código?

```
$a = "-3.1416";  
  
printf("La variable \"\$a\" vale %+.2f", $a);
```

- La variable '\$a' vale -3.14.
- La variable '\$a' vale +3.14.
- La variable '\"\$a\"' vale -3.14.
- La variable '\"\$a\"' vale +3.14.

## 4.1.8.- Funciones relacionadas con los tipos de datos (II).



En PHP no existe un tipo de datos específico para trabajar con fechas y horas. La información de fecha y hora se almacena internamente como un número entero. Sin embargo, dentro de las funciones de PHP tienes a tu disposición unas cuantas para trabajar con ese tipo de datos.

Una de las más útiles es quizás la función **date**, que te permite obtener una cadena de texto a partir de una fecha y hora, con el formato que tú elijas. La función recibe dos parámetros, la descripción del formato y el número entero que identifica la fecha, y devuelve una cadena de texto formateada.

```
string date (string $formato [, int $fechahora]);
```

El formato lo debes componer utilizando como base una serie de caracteres de los que figuran en la siguiente tabla.

### Función date: caracteres de formato para fechas y horas.

Carácter	Resultado
<b>d</b>	día del mes con dos dígitos.
<b>j</b>	día del mes con uno o dos dígitos ( sin ceros iniciales ).
<b>z</b>	día del año, comenzando por el cero ( 0 = 1 de enero ).
<b>N</b>	día de la semana ( 1 = lunes, ..., 7 = domingo ).
<b>w</b>	día de la semana ( 0 = domingo, ..., 6 = sábado ).
<b>I</b>	texto del día de la semana, en inglés ( Monday, ..., Sunday ).
<b>D</b>	texto del día de la semana, solo tres letras, en inglés ( Mon, ..., Sun ).
<b>W</b>	número de la semana del año.
<b>m</b>	número del mes con dos dígitos.
<b>n</b>	número del mes con uno o dos dígitos ( sin ceros iniciales ).

<b>t</b>	número de días que tiene el mes.
<b>F</b>	texto del día del mes, en inglés ( January, ..., December ).
<b>M</b>	texto del día del mes, solo tres letras, en inglés ( Jan, ..., Dec ).
<b>Y</b>	número del año.
<b>y</b>	dos últimos dígitos del número del año.
<b>L</b>	1 si el año es bisiesto, 0 si no lo es.
<b>h</b>	formato de 12 horas, siempre con dos dígitos.
<b>H</b>	formato de 24 horas, siempre con dos dígitos.
<b>g</b>	formato de 12 horas, con uno o dos dígitos ( sin ceros iniciales ).
<b>G</b>	formato de 24 horas, con uno o dos dígitos ( sin ceros iniciales ).
<b>i</b>	minutos, siempre con dos dígitos.
<b>s</b>	segundos, siempre con dos dígitos.
<b>u</b>	microsegundos.
<b>a</b>	am o pm, en minúsculas.
<b>A</b>	AM o PM, en mayúsculas.
<b>r</b>	fecha entera con formato <a href="#">RFC 2822</a> .

Además, el segundo parámetro es opcional. Si no se indica, se utilizará la hora actual para crear la cadena de texto.

Para que el sistema pueda darte información sobre tu fecha y hora, debes indicarle tu zona horaria. Puedes hacerlo con la función **date\_default\_timezone\_set**. Para establecer la zona horaria en España peninsular debes indicar:

```
date_default_timezone_set('Europe/Madrid');
```

## Para saber más

En la documentación de PHP puedes consultar las distintas zonas horarias que se pueden indicar.

[Distintas zonas horarias soportadas en PHP](#)

Si utilizas alguna función de fecha y hora sin haber establecido previamente tu zona horaria, lo más probable es que recibas un error o mensaje de advertencia de PHP indicándolo.

Otras funciones como getdate devuelven un array con información sobre la fecha y hora actual.

## Debes conocer

En la documentación de PHP puedes consultar todas las funciones para gestionar fechas y horas:

[Funciones para gestionar fechas y horas.](#)

Intenta resolver la siguiente actividad, relativa a las funciones de PHP que acabas de ver.



(110 KB)

Para realizar la actividad descarga el paquete, descomprímelo y ejecuta la página .html (Funciona en Ubuntu con Firefox y Flash 11.2)

## 4.2.- Estructuras de control.

### Caso práctico

**¡Bien! Ya están claros los fundamentos del lenguaje.**

Pero con lo visto hasta el momento, sólo es posible hacer programas muy sencillos. Para poder empezar a programar, **Carlos** sabe qué debe estudiar a continuación. Una de las partes más importantes de cualquier lenguaje es la que permite tomar decisiones, es decir, las sentencias que se pueden usar para indicar bajo qué condiciones se debe ejecutar una instrucción o un bloque de instrucciones. Y como no, también las sentencias para repetir la ejecución de ciertas líneas de código.

Cuando domine esas estructuras, podrá empezar a probar todo lo que lleva aprendido.



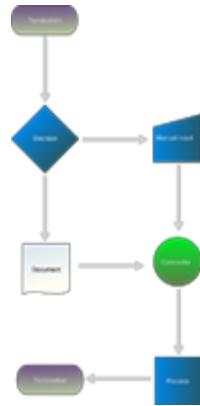
Stockbyte. CD-DVD Num. CD165.

**En PHP los guiones se construyen en base a sentencias.** Utilizando llaves, puedes agrupar las sentencias en conjuntos, que se comportan como si fueran una única sentencia.

Para definir el flujo de un programa en PHP, al igual que en la mayoría de lenguajes de programación, hay sentencias para dos tipos de **estructuras de control**: **sentencias condicionales**, que permiten definir las condiciones bajo las que debe ejecutarse una sentencia o un bloque de sentencias; y **sentencias de bucle**, con las que puedes definir si una sentencia o conjunto de sentencias se repite o no, y bajo qué condiciones.

Además, en PHP puedes usar también (aunque no es recomendable) la sentencia goto, que te permite saltar directamente a otro punto del programa que indiques mediante una etiqueta.

```
<?php  
  
$a = 1;  
  
goto salto;  
  
$a++; //esta sentencia no se ejecuta  
  
salto:  
  
echo $a; // el valor obtenido es 1  
  
?>
```



Elaboración Propia

## 4.2.1.- Condicionales.

---



La sentencia **if** permite definir una expresión para ejecutar o no la sentencia o conjunto de sentencias siguiente. Si la expresión se evalúa a true (verdadero), la sentencia se ejecuta. Si se evalúa a false (falso), no se ejecutará.

Cuando el resultado de la expresión sea false, puedes utilizar **else** para indicar una sentencia o grupo de sentencias a ejecutar en ese caso. Otra alternativa a **else** es utilizar **elseif** y escribir una nueva expresión que comenzará un nuevo condicional.

```
<?php  
  
if ($a < $b)  
  
print "a es menor que b";  
  
elseif ($a > $b)  
  
print "a es mayor que b";  
  
else  
  
print "a es igual a b";  
  
?>
```

Cuando, como sucede en el ejemplo, la sentencia if elseif o **else** actúe sobre una única sentencia, no será necesario usar llaves. Tendrás que usar llaves para formar un conjunto de sentencias siempre que quieras que el condicional actúe sobre más de una sentencia.

**switch.** La sentencia switch es similar a enlazar varias sentencias **if** comparando una misma variable con diferentes valores. Cada valor va en una sentencia **case**. Cuando se encuentra una coincidencia, comienzan a ejecutarse las sentencias siguientes hasta que acaba el bloque **switch**, o hasta que se encuentra una sentencia **break**. Si no existe coincidencia con el valor de ningún **case**, se ejecutan las sentencias del bloque **default**, en caso de que exista.

```
<?php
```

```
switch ($a) {
```

```
case 0:
```

```
print "a vale 0";
```

```
break;
```

```
case 1:
```

```
print "a vale 1";
```

```
break;
```

```
default:
```

```
print "a no vale 0 ni 1";
```

```
}
```

```
?>
```

## Ejercicio resuelto

**Haz una página web que muestre la fecha actual en castellano, incluyendo el día de la semana, con un formato similar al siguiente: "Miércoles, 14 de Octubre de 2015".**

[Mostrar retroalimentación](#)

## Autoevaluación

**¿Siempre se puede sustituir una sentencia switch por otra sentencia o sentencias if?**

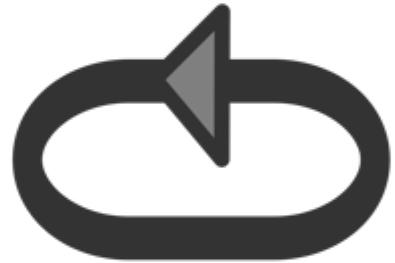
- No.
- Sí.

## 4.2.2.- Bucles.

---

- ✓ **while:** Usando **while** puedes definir un bucle que se ejecuta mientras se cumpla una expresión. La expresión se evalúa antes de comenzar cada ejecución del bucle.

```
<?php  
$a = 1;  
  
while ($a < 8)  
  
    $a += 3;  
  
    print $a; // el valor obtenido es 10  
  
?>
```



 Anonima

- ✓ **do / while:** Es un bucle similar al anterior, pero la expresión se evalúa al final, con lo cual se asegura que la sentencia o conjunto de sentencias del bucle se ejecutan al menos una vez.

```
<?php  
$a = 5;  
  
do  
  
    $a -= 3;  
  
    while ($a > 10);  
  
    print $a; // el bucle se ejecuta una sola vez, con lo que el valor obtenido es 2  
  
?>
```

- ✓ **for:** Son los bucles más complejos de PHP. Al igual que los del lenguaje C, se componen de tres expresiones:

```
for (expr1; expr2; expr3)
```

sentencia o conjunto de sentencias;

La primera expresión, expr1, se ejecuta solo una vez al comienzo del bucle.

La segunda expresión, expr2, se evalúa para saber si se debe ejecutar o no la sentencia o conjunto de sentencias. Si el resultado es false, el bucle termina.

Si el resultado es true, se ejecutan las sentencias y al finalizar se ejecuta la tercera expresión, expr3, y se vuelve a evaluar expr2 para decidir si se vuelve a ejecutar o no el bucle.

```
<?php  
  
for ($a = 5; $a<10; $a+=3) {  
  
    print $a; // Se muestran los valores 5 y 8  
  
    print "<br />";  
  
}  
  
?>
```

Puedes anidar cualquiera de los bucles anteriores en varios niveles. También puedes usar las sentencias break, para salir del bucle, y continue, para omitir la ejecución de las sentencias restantes y volver a la comprobación de la expresión respectivamente.

---

## Para saber más

En el siguiente videotutorial puedes ver con ejemplos la utilización de bucles en PHP.

**Videotutorial sobre la utilización de bucles en PHP.**

[Resumen textual alternativo](#)

## Autoevaluación

**Si quieres mostrar una cadena de texto letra a letra, y no sabes si está vacía, ¿qué tipo de bucle emplearías, while o do-while?**

- while.
- do-while.

## 4.3.- Funciones.

### Caso práctico



Stockbyte. CD-DVD Num. CD303.

Juan observa con agrado los **progresos que va haciendo Carlos** en su aprendizaje del lenguaje PHP. Con la ilusión que está poniendo, se integrará sin problemas en el nuevo proyecto. Cuantos más puedan colaborar, mejor.

Tras lo que ya ha visto, le recomienda que aprenda a **crear y utilizar funciones**. Sabe que no sólo es muy importante saber usarlas, sino también conocer todas las que hay disponibles en el lenguaje, o al menos, saber cómo buscarlas. En un lenguaje abierto como PHP, si sabes utilizar el código que ya hay programado puedes ahorrarte una gran parte del trabajo.

Cuando quieras repetir la ejecución de un bloque de código, puedes utilizar un bucle. Las **funciones** tienen una utilidad similar: **nos permiten asociar una etiqueta** (el nombre de la función) **con un bloque de código** a ejecutar. Además, al usar funciones estamos ayudando a estructurar mejor el código. Como ya sabes, las funciones permiten crear variables locales que no serán visibles fuera del cuerpo de las mismas.

Como programador puedes aprovecharte de la gran cantidad de funciones disponibles en PHP. De éstas, muchas están incluidas en el núcleo de PHP y se pueden usar directamente. Otras muchas se encuentran disponibles en forma de extensiones, y se pueden incorporar al lenguaje cuando se necesitan.

Con la distribución de PHP se incluyen varias extensiones. Para poder usar las funciones de una extensión, tienes que asegurarte de activarla mediante el uso de una directiva **extensión** en el fichero php.ini. Muchas otras extensiones no se incluyen con PHP y antes de poder utilizarlas tienes que descargarlas.

Para obtener extensiones para el lenguaje PHP puedes utilizar [PECL](#). PECL es un repositorio de extensiones para PHP. Junto con PHP se incluye un **comando pecl** que puedes utilizar para instalar extensiones de forma sencilla:

```
pecl install nombre_extensión
```

## Para saber más

En el manual de PHP tienes más información sobre PECL.

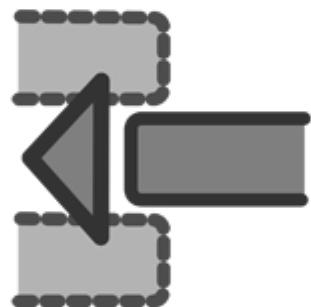
[Información sobre PECL.](#)

## 4.3.1.- Inclusión de ficheros externos.

Conforme vayan creciendo los programas que hagas, verás que resulta trabajoso encontrar la información que buscas dentro del código. En ocasiones resulta útil agrupar ciertos grupos de funciones o bloques de código, y ponerlos en un fichero aparte. Posteriormente, puedes hacer referencia a esos ficheros para que PHP incluya su contenido como parte del programa actual.

Para incorporar a tu programa contenido de un archivo externo, tienes varias posibilidades:

- ✓ **include:** Evalúa el contenido del fichero que se indica y lo incluye como parte del fichero actual, en el mismo punto en que se realiza la llamada. La ubicación del fichero puede especificarse utilizando una ruta absoluta, pero lo más usual es con una ruta relativa. En este caso, se toma como base la ruta que se especifica en la directiva include\_path del fichero php.ini. Si no se encuentra en esa ubicación, se buscará también en el directorio del guion actual, y en el directorio de ejecución.



Anonima

### Ejercicio resuelto

Te presentamos un ejemplo de utilización de `include`.

[Mostrar retroalimentación](#)

Cuando se comienza a evaluar el contenido del fichero externo, se abandona de forma automática el modo PHP y su contenido se trata en principio como etiquetas HTML. Por este motivo, es necesario marcar el inicio del código PHP que contendrá nuestro archivo externo utilizando el delimitador `<?php`

**IMPORTANTE:** Sin embargo no es recomendable cerrar un fichero para funciones (o en general los llamados desde `include`) con un `?>` puesto que puede confundir el cerrar y luego a la vuelta desde el fichero original desde dónde se llamó no volver a abrir.

También es recomendable que todos los ficheros de texto estén codificados en UTF-8 y que acaben con una línea en blanco, esto último evita confusiones en GIT cuando se modifican.

- ✓ **include\_once:** Si por equivocación incluyes más de una vez un mismo fichero, lo normal es que obtengas algún tipo de error (por ejemplo, al repetir una definición de una función). include\_once funciona exactamente igual que include, pero solo incluye aquellos ficheros que aún no se hayan incluido.
- ✓ **require:** Si el fichero que queremos incluir no se encuentra, include da un aviso y continua la ejecución del guión. La diferencia más importante al usar require es que en ese caso, cuando no se puede incluir el fichero, se detiene la ejecución del guión.
- ✓ **require\_once.** Es la combinación de las dos anteriores. Asegura la inclusión del fichero indicado solo una vez, y genera un error si no se puede llevar a cabo.

Muchos programadores utilizan la doble extensión .inc.php para aquellos ficheros en lenguaje PHP cuyo destino es ser incluidos dentro de otros, y nunca han de ejecutarse por sí mismos.

## Autoevaluación

**¿Puedes utilizar include o require para incluir el mismo encabezado HTML en varias páginas?**

- Sí.
- No.

## 4.3.2.- Ejecución y creación de funciones.

Ya sabes que para hacer una llamada a una función, basta con poner su nombre y unos paréntesis.

```
<?php
```

```
    phpinfo();
```

```
?>
```



 PUBLIC DOMAIN mothinator.

Para crear tus propias funciones, deberás usar la palabra function.

```
<?php
```

```
function precio_con_iva() {  
    global $precio;  
  
    $precio_iva = $precio * 1.21;  
  
    print "El precio con IVA es ".$precio_iva;  
  
}  
  
$precio = 10;  
  
precio_con_iva();  
  
?>
```

En PHP no es necesario que definas una función antes de utilizarla, excepto cuando está condicionalmente definida como se muestra en el siguiente ejemplo:



## 4.3.3.- Argumentos.

---

En el ejemplo anterior en la función usabas una variable global, lo cual no es una buena práctica. Siempre es mejor utilizar argumentos o parámetros al hacer la llamada. Además, en lugar de mostrar el resultado en pantalla o guardar el resultado en una variable global, las funciones pueden devolver un valor usando la sentencia return. Cuando en una función se encuentra una sentencia return, termina su procesamiento y devuelve el valor que se indica.

Puedes reescribir la función anterior de la siguiente forma:



<?php

 Benjamin Pavie / ben

```
function precio_con_iva($precio) {  
  
    return $precio * 1.21;  
  
}  
  
$precio = 10;  
  
$precio_iva = precio_con_iva($precio);  
  
print "El precio con IVA es ".$precio_iva  
  
?>
```

Los argumentos se indican en la definición de la función como una lista de variables separada por comas. No se indica el tipo de cada argumento, al igual que no se indica si la función va a devolver o no un valor (si una función no tiene una sentencia return, devuelve null al finalizar su procesamiento).

Al definir la función, puedes indicar valores por defecto para los argumentos, de forma que cuando hagas una llamada a la función puedes no indicar el valor de un argumento; en este caso se toma el valor por defecto indicado.

```
<?php

function precio_con_iva($precio, $iva=0.21) {

    return $precio * (1 + $iva);

}

$precio = 10;

$precio_iva = precio_con_iva($precio);

print "El precio con IVA es ".$precio_iva

?>
```

Puede haber valores por defecto definidos para varios argumentos, pero en la lista de argumentos de la función todos ellos deben estar a la derecha de cualquier otro argumento sin valor por defecto.

En los ejemplos anteriores los argumentos se pasaban **por valor**. Esto es, cualquier cambio que se haga dentro de la función a los valores de los argumento no se reflejará fuera de la función. Si quieres que esto ocurra debes definir el parámetro para que su valor se pase **por referencia**, añadiendo el símbolo & antes de su nombre.

```
<?php

function precio_con_iva(&$precio, $iva=0.18) {

    $precio *= (1 + $iva);

}

$precio = 10;
precio_con_iva($precio);

print "El precio con IVA es ".$precio

?>
```

# Autoevaluación

¿Está bien definida una función con el siguiente encabezado?

|                       |
|-----------------------|
| <input type="radio"/> |

- Sí.
- No.

## Debes conocer

A la hora de pasar argumentos por referencia, no existe ningún signo de referencia en una llamada a una función - sólo en la definición de la función. Las definiciones de funciones por sí solas son suficientes para pasar correctamente el argumento por referencia.

A partir de PHP **5.3.0**, se obtendrá una advertencia diciendo que "call-time pass-by-reference" (pasar por referencia en tiempo de llamada) está obsoleto cuando se use & en foo(&\$a);.

A partir de PHP **5.4.0**, el paso por referencia en tiempo de llamada ha sido eliminado, por lo que su uso emitirá un **error fatal**.

[Más info sobre paso por referencia](#)

## Ejercicio resuelto

Anteriormente hiciste un ejercicio que mostraba la fecha actual en castellano. Con el mismo objetivo (puedes utilizar el código ya hecho), crea una función que devuelva una cadena de texto con la fecha en castellano, e introduce la función en un fichero externo. Después crea una página en PHP que incluya ese fichero y utilice la función para mostrar en pantalla la fecha obtenida.

[Mostrar retroalimentación](#)

## 4.4.- Tipos de datos compuestos.

### Caso práctico



Photodisc CD-DVD Num. V07

**Es muy raro el programa que utilice solo tipos simples**, y más en PHP. **Carlos ya ha asumido que tendrá que utilizar arrays** para casi cualquier código que haga. La información del servidor, los datos que introduce el usuario, o las cadenas de texto. Gran parte de las variables que se usan en un programa están en forma de array.

Por tanto, el siguiente paso está claro: **hay que dominar los arrays**. Crearlos, utilizarlos, recorrerlos... Y como acaba de aprender, antes de ponerse a programar le echará un vistazo a las funciones que ya existen para manejarlos. Si las puede aprovechar en sus programas, ¡mejor!

Un tipo de datos compuesto es aquel que te permite almacenar más de un valor. En PHP puedes utilizar dos tipos de datos compuestos: el **array** y el **objeto**. Los objetos los veremos más adelante; vamos a empezar con los arrays.

Un **array** es un tipo de datos que nos permite almacenar varios valores. Cada miembro del **array** se almacena en una posición a la que se hace referencia utilizando un valor clave. Las claves pueden ser numéricas o asociativas.

// array numérico

```
$modulos1 = array(0 => "Programación", 1 => "Bases de datos", ..., 9 => "Desarrollo web en entorno servidor");
```

// array asociativo

```
$modulos2 = array("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "Desarrollo web en entorno servidor");
```

# Debes conocer

En PHP existe la función print\_r, que nos muestra todo el contenido del array que le pasamos. Es muy útil para tareas de depuración.

[Función print\\_r](#)

Para hacer referencia a los elementos almacenados en un array, tienes que utilizar el valor clave entre corchetes:

```
$modulos1 [9]
```

```
$modulos2 ["DWES"]
```

Los arrays anteriores son vectores, esto es, arrays unidimensionales. En PHP puedes crear también arrays de varias dimensiones almacenando otro array en cada uno de los elementos de un array.

```
// array bidimensional  
  
$ciclos = array(  
  
    "DAW" => array ("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "Desarrollo web en entorno servidor"),  
  
    "DAM" => array ("PR" => "Programación", "BD" => "Bases de datos", ..., "PMDM" => "Programación multimedia y de dispositivos r  
  
);
```

Para hacer referencia a los elementos almacenados en un **array multidimensional**, debes indicar las claves para cada una de las dimensiones:

```
$ciclos ["DAW"] ["DWES"]
```

En PHP no es necesario que indiques el tamaño del array antes de crearlo. Ni siquiera es necesario indicar que una variable concreta es de tipo array. Simplemente puedes comenzar a asignarle valores:



## 4.4.1.- Recorrer arrays (I).

Las **cadenas de texto** o **strings** se pueden tratar como arrays en los que se almacena una letra en cada posición, siendo 0 el índice correspondiente a la primera letra, 1 el de la segunda, etc.

```
// cadena de texto  
  
$modulo = "Desarrollo web en entorno servidor";  
  
// $modulo[3] == "a";
```



 PUBLIC DOMAIN [Anonima](#)

Para recorrer los elementos de un array, en PHP puedes usar un bucle específico: foreach. Utiliza una variable temporal para asignarle en cada iteración el valor de cada uno de los elementos del array. Puedes usarlo de dos formas. Recorriendo sólo los elementos:

```
$modulos = array("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "Desarrollo web en entorno servidor");  
  
foreach ($modulos as $modulo) {  
  
    print "Módulo: ".$modulo. "<br />"  
  
}
```

O recorriendo los elementos y sus valores clave de forma simultánea:

```
$modulos = array("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "Desarrollo web en entorno servidor");  
  
foreach ($modulos as $codigo => $modulo) {  
  
    print "El código del módulo ".$modulo." es ".$codigo."  
    <br />"  
}
```

# Ejercicio resuelto

**Haz una página PHP que utilice foreach para mostrar todos los valores del array `$_SERVER` en una tabla con dos columnas. La primera columna debe contener el nombre de la variable, y la segunda su valor.**

[Mostrar retroalimentación](#)

## Autoevaluación

**La inicialización del array en el código siguiente, ¿generará algún error?**

```
$a[0] = 0;
```

```
$a[1] = "uno";
```

```
$a["tres"] = 3;
```

- Sí.
- No.

## 4.4.2.- Recorrer arrays (II).

Pero en PHP también **hay otra forma de recorrer los valores de un array**. Cada array mantiene un **puntero** interno, que se puede utilizar con este fin. Utilizando funciones específicas, podemos avanzar, retroceder o inicializar el puntero, así como recuperar los valores del elemento (o de la pareja clave / elemento) al que apunta el puntero en cada momento. Algunas de estas funciones son:



### Funciones para recorrer arrays.



PUBLIC DOMAIN

Anonima

| Función        | Resultado   |
|----------------|---|
| <b>reset</b>   | Sitúa el puntero interno al comienzo del array.   |
| <b>next</b>    | Avanza el puntero interno una posición.   |
| <b>prev</b>    | Mueve el puntero interno una posición hacia atrás.  |
| <b>end</b>     | Sitúa el puntero interno al final del array.  |
| <b>current</b> | Devuelve el elemento de la posición actual.   |
| <b>key</b>     | Devuelve la clave de la posición actual.  |
| <b>each</b>    | Devuelve un array con la clave y el elemento de la posición actual. Además, avanza el puntero interno una posición. |

Las funciones **reset**, **next**, **prev** y **end**, además de mover el puntero interno devuelven, al igual que **current**, el valor del nuevo elemento en que se posiciona. Si al mover el puntero te sales de los límites del array (por ejemplo, si ya estás en el último elemento y haces un **next**), cualquiera de ellas devuelve **false**. Sin embargo, al comprobar este valor devuelto no serás capaz de distinguir si te has salido de los límites del array, o si estás en una posición válida del array que contiene el valor "false".

La función **key** devuelve **null** si el puntero interno está fuera de los límites del array.

La función each devuelve un array con cuatro elementos. Los elementos **0** y '**key**' almacenan el valor de la clave en la posición actual del puntero interno. Los elementos **1** y '**value**' devuelven el valor almacenado.

Si el puntero interno del array se ha pasado de los límites del array, la función each devuelve false, por lo que la puedes usar para crear un bucle que recorra el array de la siguiente forma:

```
while ($modulo = each($modulos)) {  
  
    print "El código del módulo ".$modulo[1]." es ".$modulo[0]."<br />"  
  
}
```

## Ejercicio resuelto

**Haz una página PHP que utilice estas funciones para crear una tabla como la del ejercicio anterior.**

[Mostrar retroalimentación](#)

## 4.4.3.- Funciones relacionadas con los tipos de datos compuestos.

---

Además de asignando valores directamente, **la función array permite crear un array con una sola línea de código**, tal y como vimos anteriormente. Esta función recibe un conjunto de parámetros, y crea un array a partir de los valores que se le pasan. Si en los parámetros no se indica el valor de la clave, crea un array numérico (con base 0). Si no se le pasa ningún parámetro, crea un array vacío.



```
$a = array(); // array vacío

$modulos = array("Programación", "Bases de datos", ..., "Desarrollo web en entorno servidor"); // array numérico
```

Una vez definido un array puedes añadir nuevos elementos (no definiendo el índice, o utilizando un índice nuevo) y modificar los ya existentes (utilizando el índice del elemento a modificar). También se pueden eliminar elementos de un array utilizando la función unset.

En el caso de los arrays numéricos, eliminar un elemento significa que las claves del mismo ya no estarán consecutivas.

```
unset ($modulos [0]);

// El primer elemento pasa a ser $modulos [1] == "Bases de datos";
```

La función array\_values recibe un array como parámetro, y devuelve un nuevo array con los mismos elementos y con índices numéricos consecutivos con base 0.

Para comprobar si una variable es de tipo array, utiliza la función is\_array. Para obtener el número de elementos que contiene un array, tienes la función count.

Si quieras buscar un elemento concreto dentro de un array, puedes utilizar la función in\_array. Recibe como parámetros el elemento a buscar y la variable de tipo array en la que buscar, y devuelve true si encontró el elemento o false en caso contrario.

```
$modulos = array("Programación", "Bases de datos", "Desarrollo web en entorno servidor");  
  
$modulo = "Bases de datos";  
  
if (in_array($modulo, $modulos)) printf "Existe el módulo de nombre ".$modulo;
```

Otra posibilidad es la función `array_search`, que recibe los mismos parámetros pero devuelve la clave correspondiente al elemento, o false si no lo encuentra.

Y si lo que quieras buscar es una clave en un array, tienes la función `array_key_exists`, que devuelve true o false.

## Para saber más

En realidad en PHP hay muchas funciones para gestionar arrays. Puedes consultar una lista completa en el manual online de PHP.

[Lista completa.](#)

## Autoevaluación

**¿Se puede usar el siguiente código para recorrer un array \$a cualquiera?**

```
while ($variable = $current($a))  
{  
    ...  
    next($a);  
}
```

- No.
- Sí.



## 4.5.- Formularios web.

### Caso práctico

**Carlos** está viendo que el esfuerzo que le dedica al **aprendizaje del nuevo lenguaje** empieza a dar sus frutos. Hace unos días casi no sabía ni que existía PHP, y ahora ya es capaz de realizar programas sencillos por sí mismo.

Para poder avanzar aún más, sabe cuál ha de ser su siguiente paso: **obtener y utilizar información de un usuario**. De esta forma, los programas que haga no serán lineales, sino que tendrán un comportamiento u otro en función de los datos que aporte el usuario.



Stockbyte. CD-DVD Num. CDV43

Como le ha comentado **Juan**, para obtener información de un usuario, en PHP se utilizan los **formularios HTML**. ¡A por ellos!

La forma natural para hacer llegar a la aplicación web los datos del usuario desde un navegador, es utilizar **formularios HTML**.

Los formularios HTML van encerrados siempre entre las etiquetas <<strong>FORM</strong>> <<strong>FORM</strong>>. Dentro de un formulario se incluyen los elementos sobre los que puede actuar el usuario, principalmente usando las etiquetas <INPUT>, <SELECT>, <TEXTAREA> y <BUTTON>.

El atributo action del elemento FORM indica la página a la que se le enviarán los datos del formulario. En nuestro caso se tratará de un guión PHP.

Por su parte, el atributo method especifica el método usado para enviar la información. Este atributo puede tener dos valores:

- ✓ get: con este método los datos del formulario se agregan al URI utilizando un signo de interrogación "?" como separador.
- ✓ post: con este método los datos se incluyen en el cuerpo del formulario y se envían utilizando el protocolo HTTP.

Como vamos a ver, los datos se recogerán de distinta forma dependiendo de cómo se envíen.

## Para saber más

Para no tener problemas al programar en PHP, debes conocer el lenguaje HTML, concretamente los detalles relativos a la creación de formularios web. Puedes consultar esta información por ejemplo en el curso sobre HTML de aulaClic:

[Curso sobre HTML de aulaClic.](#)

## Ejercicio resuelto

**Crea un formulario HTML para introducir el nombre del alumno y el módulo que cursa, a escoger entre “Desarrollo Web en Entorno Servidor” y “Desarrollo Web en Entorno Cliente”. Envía el resultado a la página “procesa.php”, que será la encargada de procesar los datos.**

[Mostrar retroalimentación](#)

Fíjate que si en un formulario web tienes que enviar alguna variable en la que sea posible almacenar más de un valor, como es el caso de las **casillas de verificación** en el ejemplo anterior (se pueden marcar varias a la vez), tendrás que ponerle **corchetes** al nombre de la variable para indicar que se trata de un **array**.

El atributo action del formulario, se puede dejar vacío y hace el mismo efecto que ,es decir, se envía a la misma página desde la que se ha llamado al formulario.

Antiguamente (antes de HTML5) también se hacía con action="#"

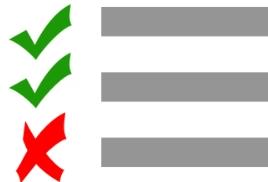
Ahora, incluso si no se pone action, por defecto también te envía a la misma página desde la que se ha llamado al formulario.

[Sending form data \(https://developer.mozilla.org\)](#)

## 4.5.1.- Procesamiento de la información devuelta por un formulario web.

---

En el ejemplo anterior creaste un **formulario en una página HTML** que recogía datos del usuario y los enviaba a una página PHP para que los procesara. Como usaste el método POST, los datos se pueden recoger utilizando la variable `$_POST`. Si simplemente los quisieras mostrar por pantalla, éste podría ser el código de "procesa.php":



**Consulta el Anexo VII con el Código de "procesa.php"**

Si por el contrario hubieras usado el método GET, el código necesario para procesar los datos sería similar; simplemente haría falta cambiar la variable `$_POST` por `$_GET`.



**Consulta el Anexo VIII con el Código necesario para procesar los datos**

En cualquiera de los dos casos podrías haber usado `$_REQUEST` sustituyendo respectivamente a `$_POST` y a `$_GET`.

**Consulta el Anexo IX Usando `$_REQUEST` para sustituir respectivamente a `$_POST` y a `$_GET`**

Siempre que sea posible, es preferible **validar los datos que se introducen en el navegador antes de enviarlos**. Para ello deberás usar código en **lenguaje Javascript**.

Si por algún motivo hay datos que se tengan que validar en el servidor, por ejemplo, porque necesites comprobar que los datos de un usuario no existan ya en la base de datos antes de introducirlos, será necesario hacerlo con código PHP en la página que figura en el atributo action del formulario.

En este caso, una posibilidad que deberás tener en cuenta es usar la misma página que muestra el formulario como destino de los datos. Si tras comprobar los datos éstos son correctos, se reenvía a otra página. Si son incorrectos, se rellenan los datos correctos en el formulario y se indican cuáles son incorrectos y por qué.

Para hacerlo de este modo, tienes que comprobar si la página recibe datos (hay que mostrarlos y no generar el formulario), o si no recibe datos (hay que mostrar el formulario). Esto se puede hacer utilizando la función isset con una variable de las que se deben recibir (por ejemplo, poniéndole un nombre al botón de enviar y comprobando sobre él). En el siguiente código de ejemplo se muestra cómo hacerlo.

**Consulta el Anexo X que muestra como Procesar datos en la misma página que el formulario**

Fíjate en la forma de englobar el formulario dentro de una sentencia else para que sólo se genere si no se reciben datos en la página. Además, para enviar los datos a la misma página que contiene el formulario puedes usar `$_SERVER['PHP_SELF']` para obtener su nombre; esto hace que no se produzca un error aunque la página se cambie de nombre.

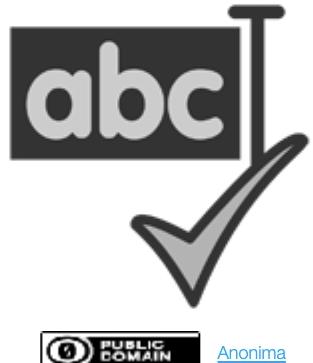
## 4.5.2.- Generación de formularios web en PHP.

---

Vamos a volver sobre el ejemplo anterior, revisando los datos que se obtienen antes de mostrarlos. Concretamente, tienes que comprobar que el nombre no esté vacío, y que se haya seleccionado como mínimo uno de los módulos.

Además, en el caso de que falte algún dato, deberás generar el formulario rellenando aquellos datos que el usuario haya introducido correctamente.

Lo primero que tienes que hacer es la validación de los datos. En el ejemplo propuesto será algo así como:



Anonima

A large, light blue grid of horizontal lines, resembling a sheet of lined notebook paper. There are 15 lines in total, with a vertical line on the left side for hanging the paper. The lines are evenly spaced and extend across the width of the page.

Para que el usuario no pierda, después de enviar el formulario, los datos correctamente introducidos, utiliza el atributo value en las entradas de texto (utilizamos la función isset para comprobar que la variable que queremos mostrar existe, para que no nos muestre un mensaje de error si es la primera vez que se carga la página y todavía no ha enviado nada el formulario):

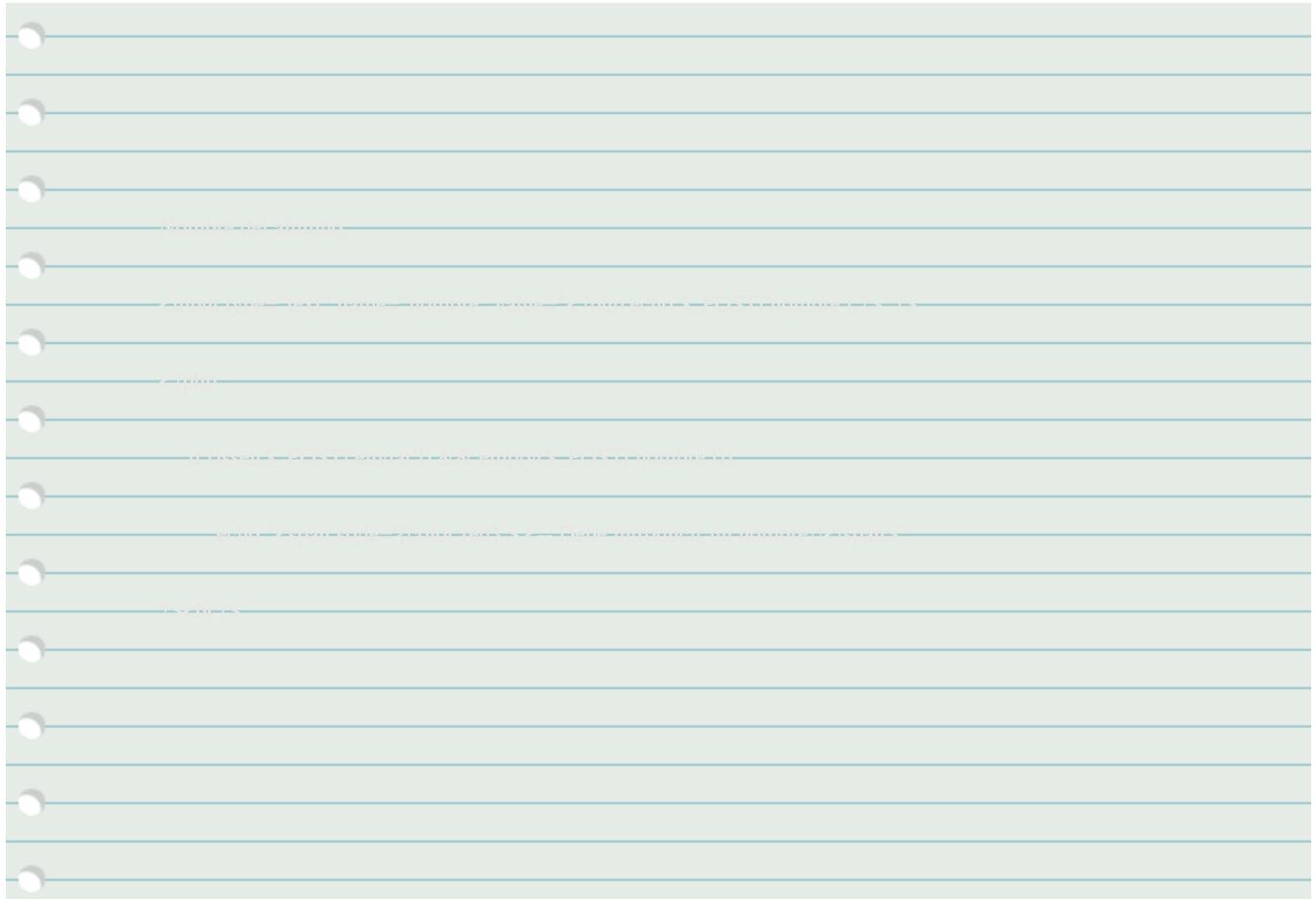
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 

Y el atributo checked en las casillas de verificación:

- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 

Fíjate en el uso de la función `in_array` para buscar un elemento en un array.

Para indicar al usuario los datos que no ha rellenado (o que ha rellenado de forma incorrecta), deberás comprobar si es la primera vez que se visualiza el formulario, o si ya se ha enviado. Se puede hacer por ejemplo de la siguiente forma:



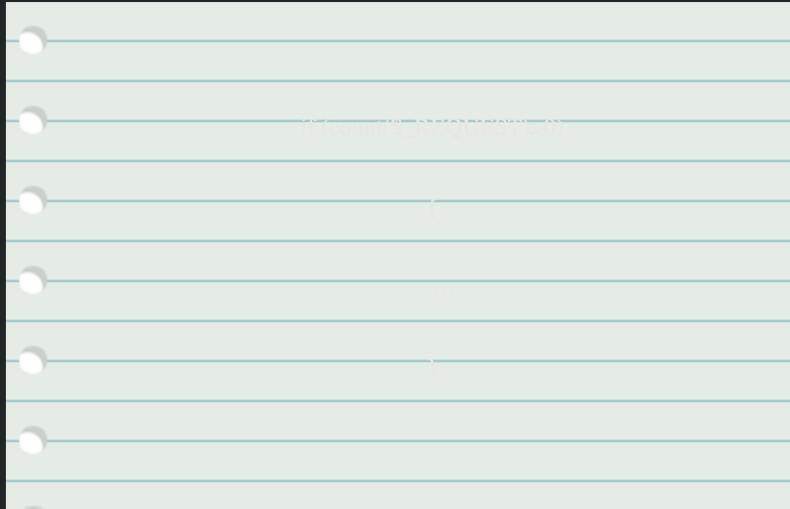
Revisa el documento con el ejemplo completo, fijándote en las partes que hemos comentado anteriormente.

**Consulta el Anexo XI que contiene el Documento con el ejemplo anterior completo**

Una forma de enviar información de una página PHP a otra, es incluyéndola en campos ocultos dentro de un formulario.

# Autoevaluación

¿Serviría el siguiente código para comprobar si se han recibido los datos de un formulario?



- Sí.
- No.

## Ejercicio resuelto

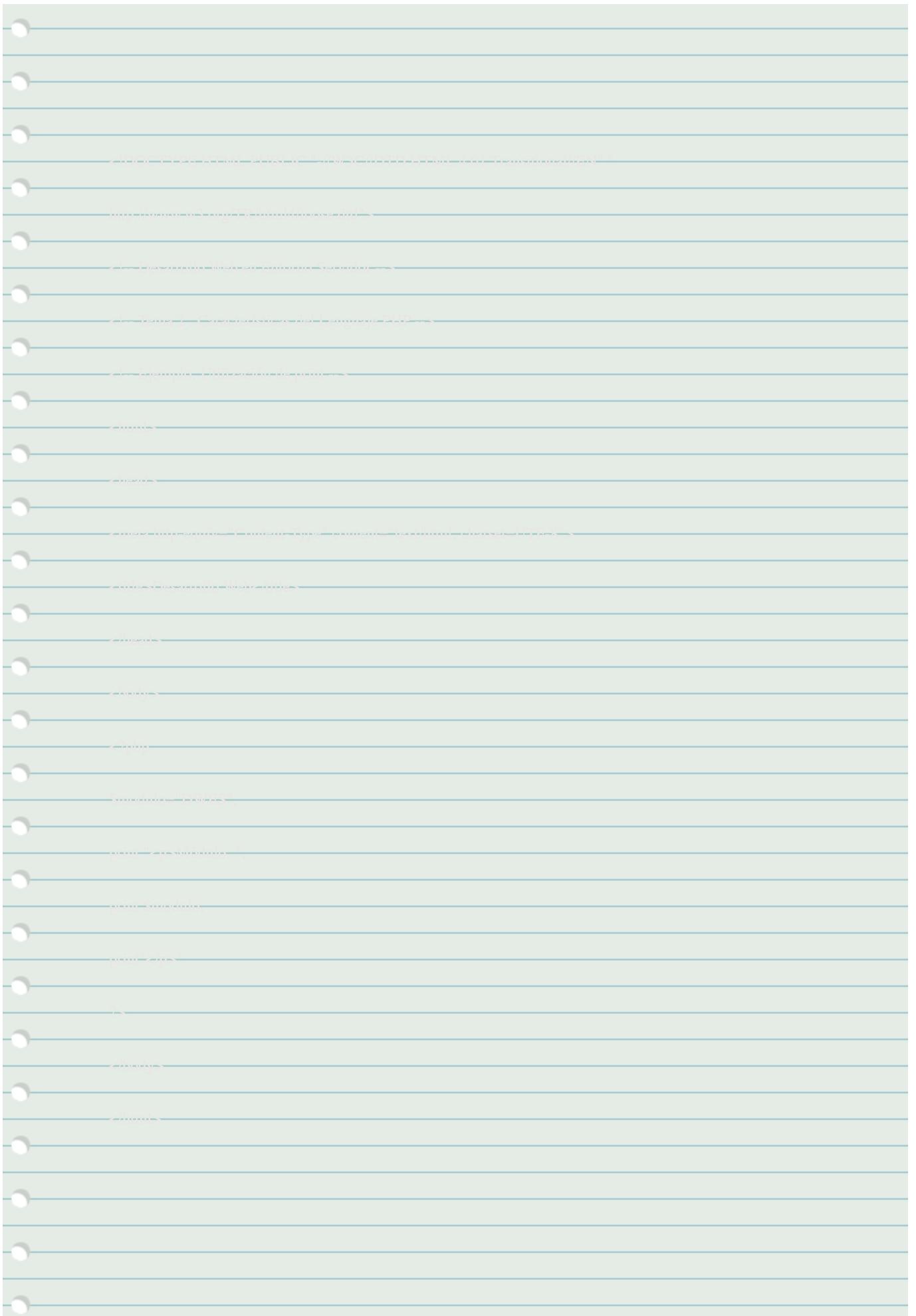
Modifica el ejercicio que mostraba la fecha en castellano, para que obtenga lo mismo a partir de un día, mes y año introducido por el usuario. Antes de mostrar la fecha, se debe comprobar que es correcta. Utilizar la misma página PHP para el formulario de introducción de datos y para mostrar la fecha obtenida en castellano.

Consultar las funciones `checkdate` y `mktime` en el manual de PHP.

[Mostrar retroalimentación](#)

## **Anexo I.- Utilización de la función print en PHP.**

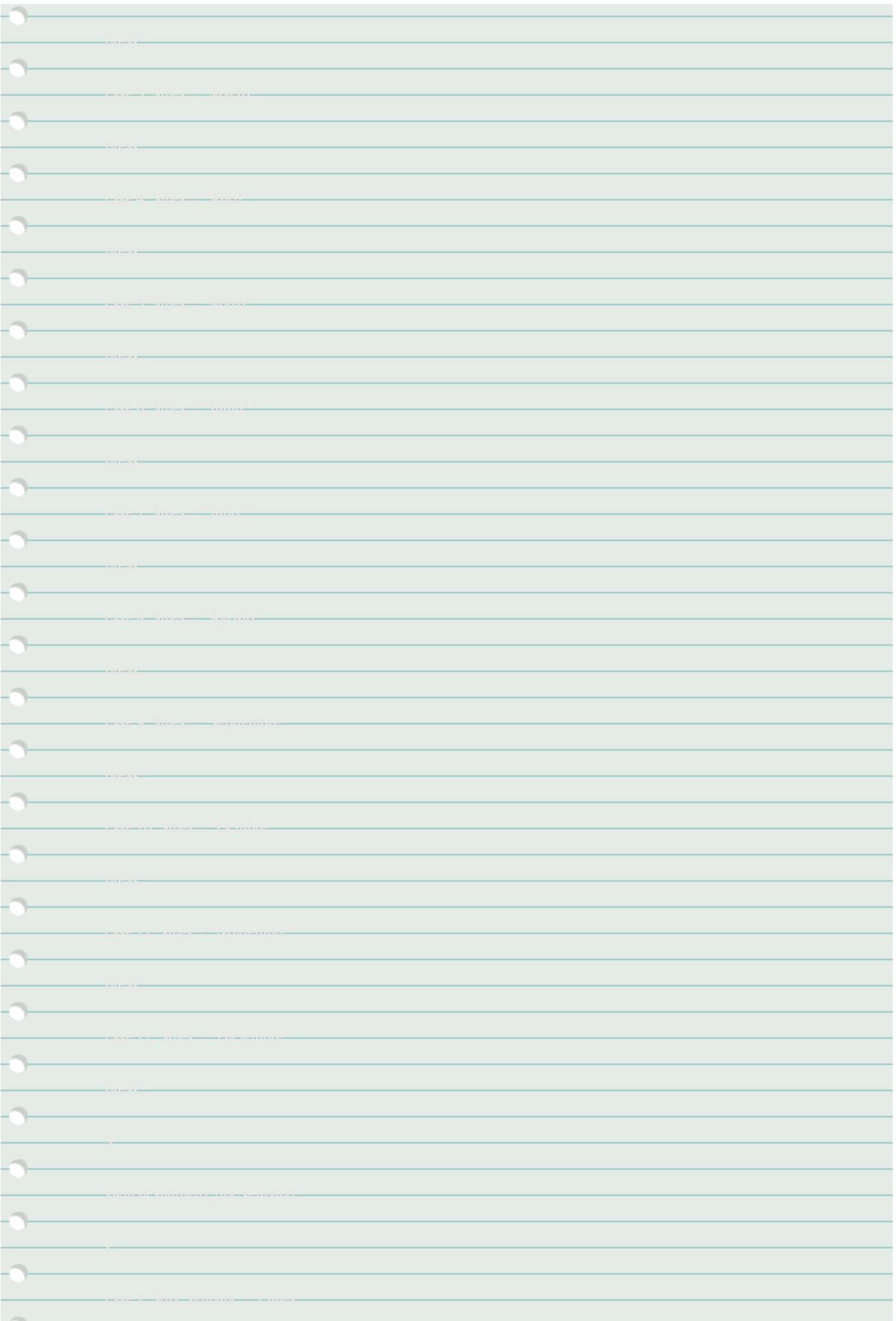
---

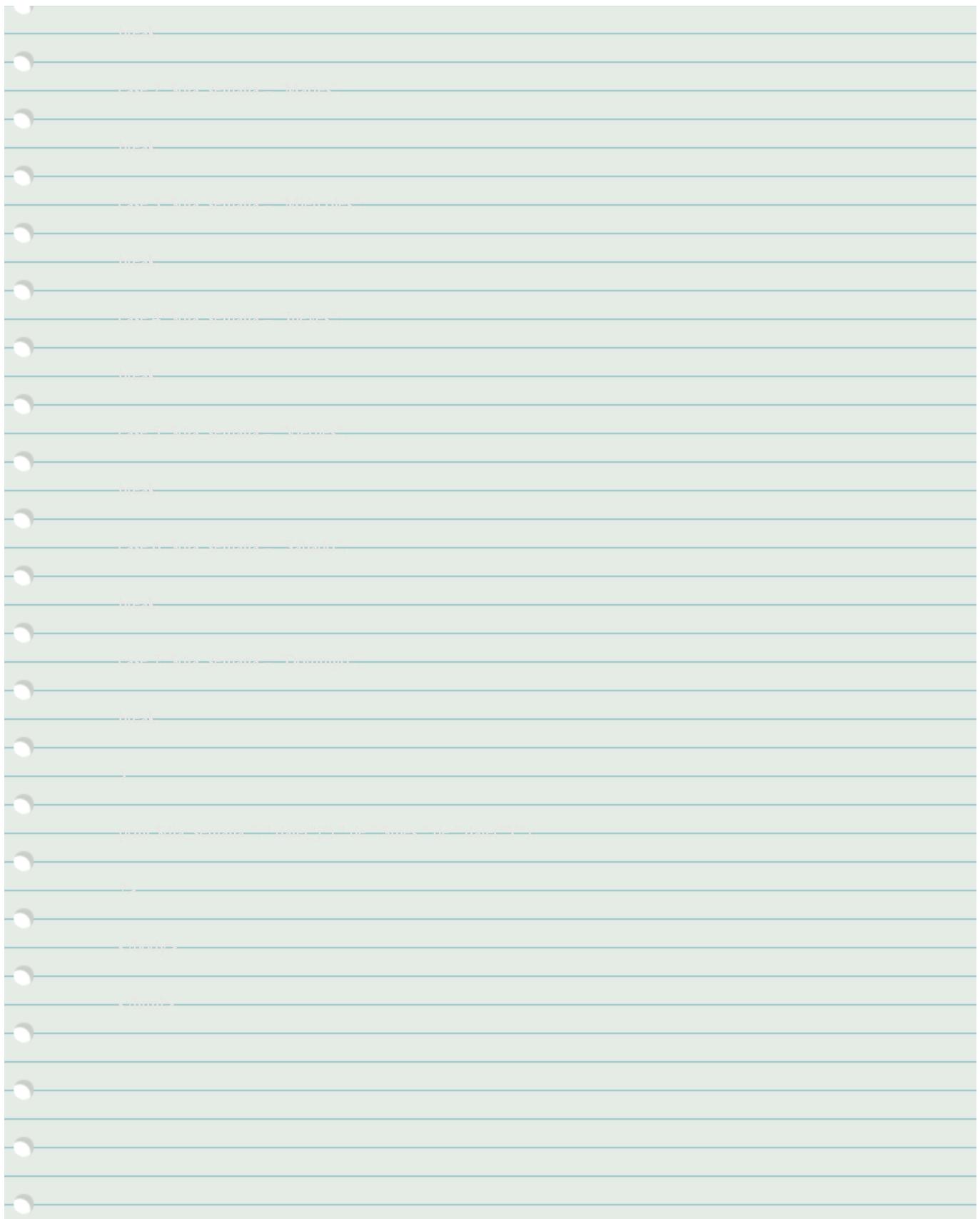


## Anexo II.- Solución propuesta I.

---

This image shows a single sheet of white paper with horizontal blue ruling lines and vertical grey margin lines on the left side. There are 22 horizontal lines in total, with a slightly larger gap at the top and bottom. The vertical lines create a margin on the left side. The paper is otherwise blank, with no handwriting or other markings.





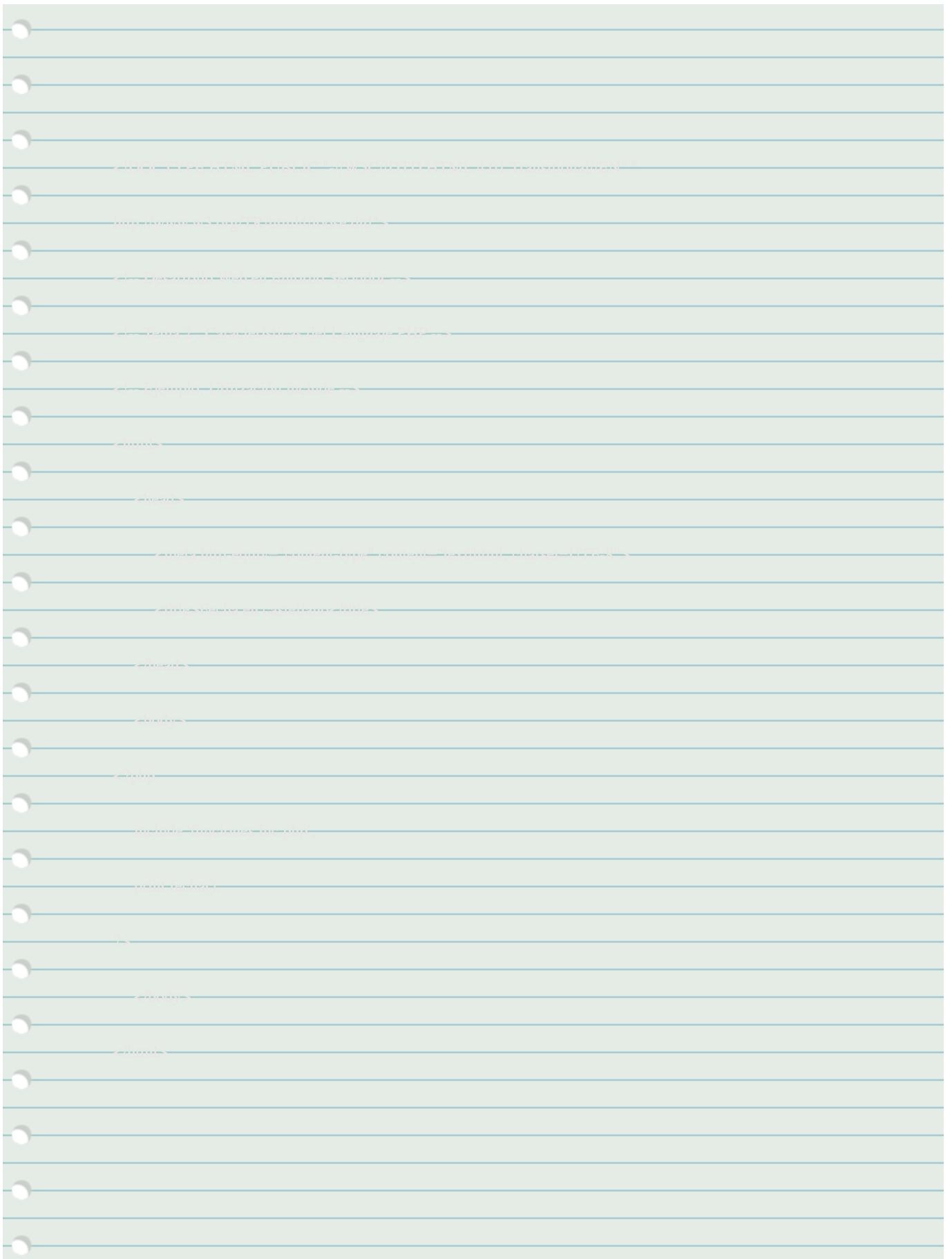
## Para saber más

Otra forma más sencilla sería usando la función **strftime**:

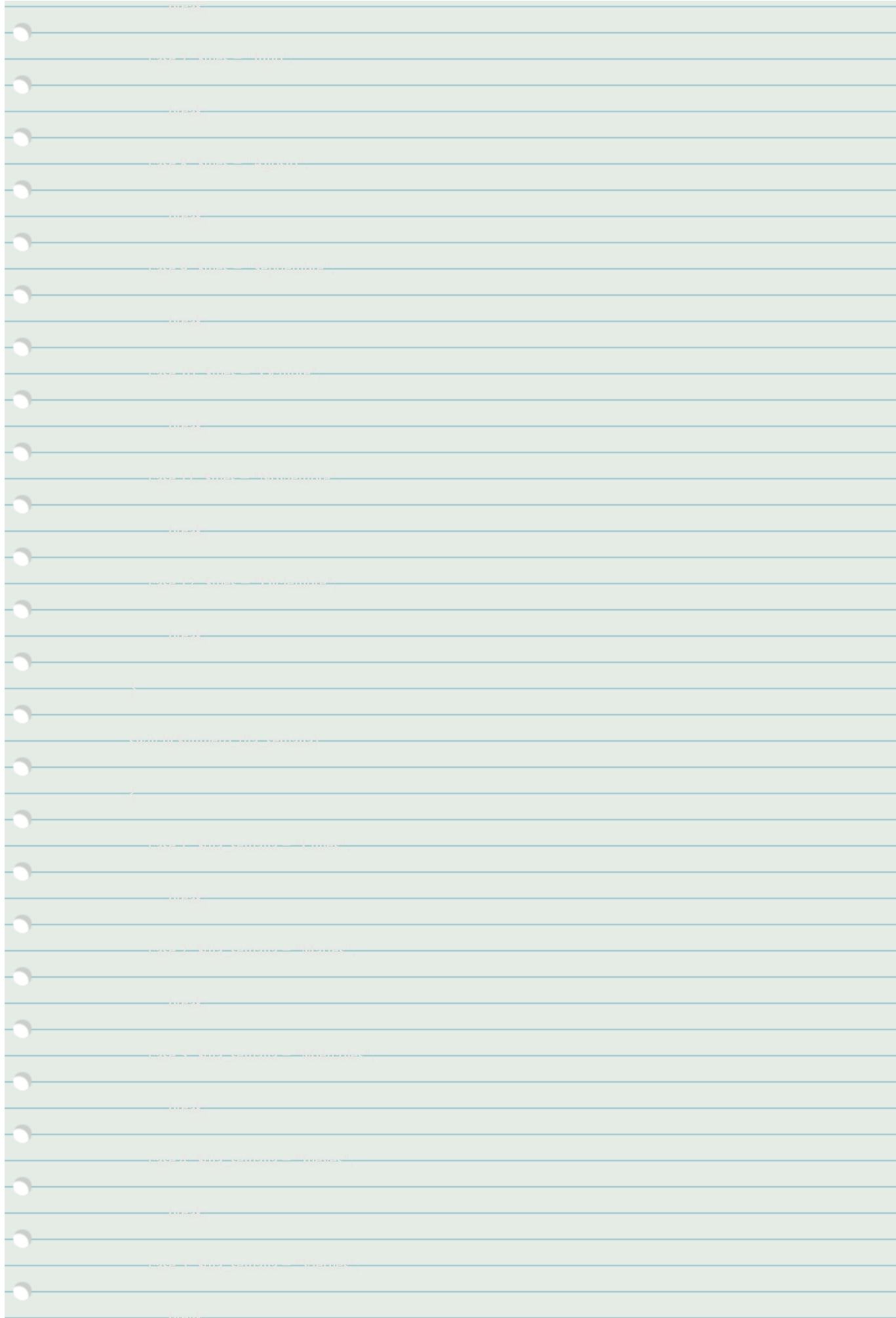
```
setlocale(LC_ALL,"es_ES.UTF-8");echo strftime("%A, %d de %B del %Y");
```

## **Anexo III.- Solución propuesta II.**

---









## Anexo IV.- Solución propuesta III.

---

This image shows a single sheet of white paper with horizontal blue ruling lines and vertical grey margin lines on the left side. There are 22 horizontal lines in total, with a slightly larger gap at the top and bottom. The vertical lines create a margin on the left side. The paper is otherwise blank, with no handwriting or other markings.



## Anexo V.- Solución propuesta IV.

---

http://www.w3.org/2001/XMLSchema#

<!-- Desarrollo Web en Entorno Servidor -->

<!-- Desarrollo de Aplicaciones de Escritorio -->

<!-- Desarrollo de Aplicaciones de Móvil -->

Alumnos

Asociados

<!-- Desarrollo de Aplicaciones de Escritorio -->

Centros Educativos

Centros de Investigación

<!-- Desarrollo de Aplicaciones de Escritorio -->

Centros de Información

Centros Comunitarios

Centros

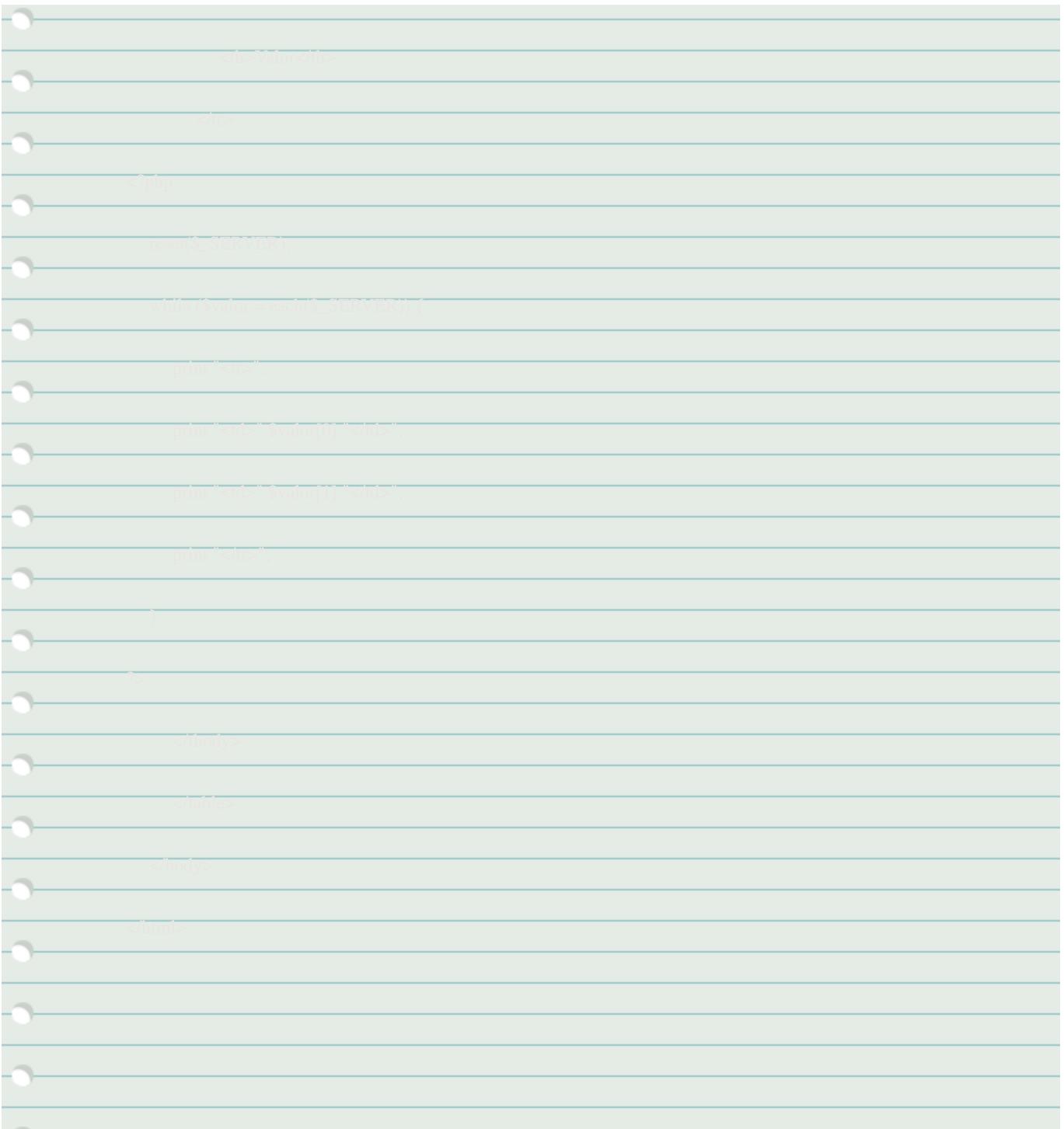
Centro

Centro

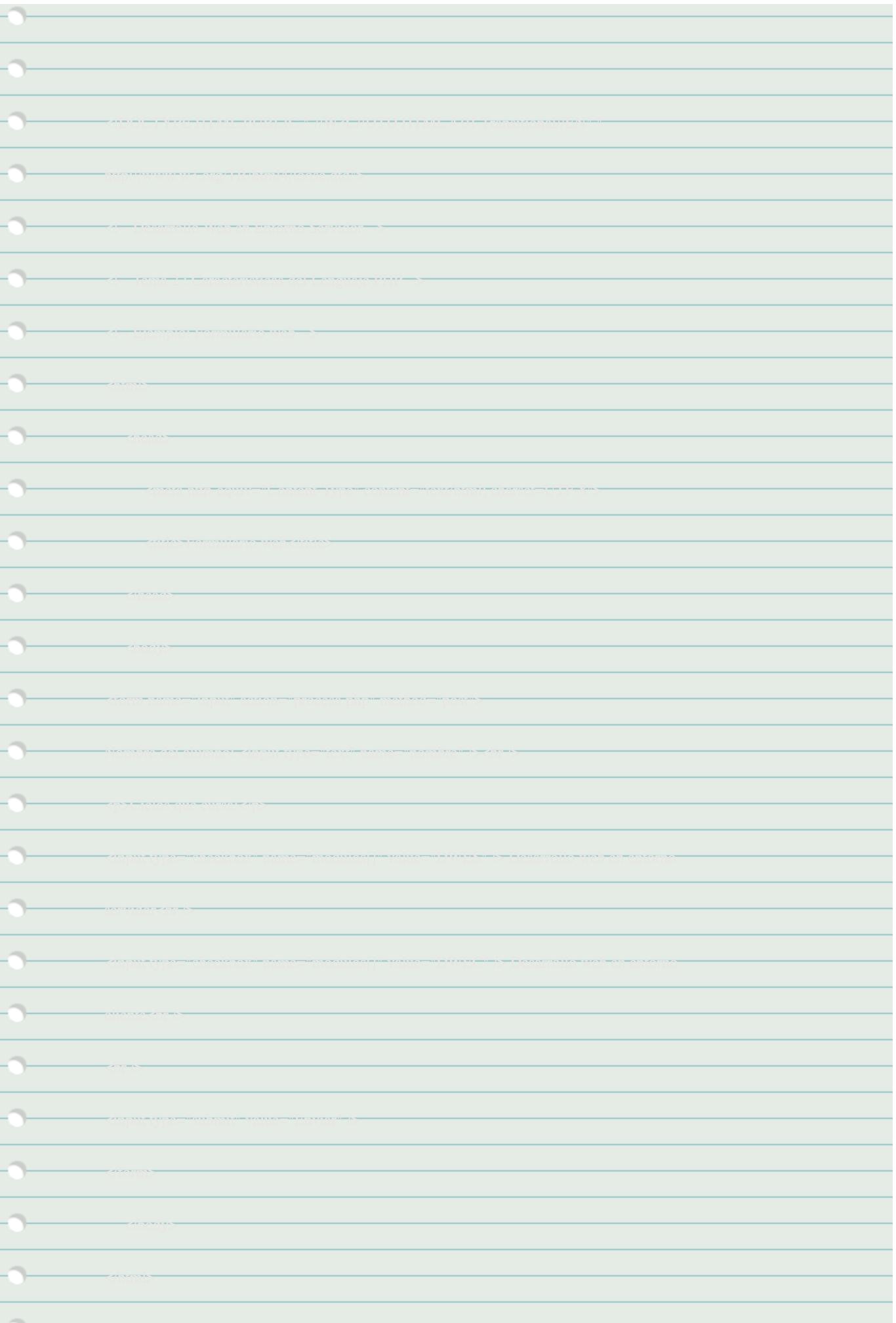
Centro

Centro

Centro



## Anexo VI.- Solución propuesta V.





## Anexo VII.- Código de "procesa.php".

---

A blank sheet of white paper with horizontal blue ruling lines and a vertical grey margin line on the left side. There are 21 lines in total, starting with a top line and ending with a bottom line. The margin line is located approximately one-third of the way from the left edge.

```
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0 Transitional//EN">
```

```
http://www.w3.org/1999/xhtml4/strict.dtd"
```

```
<!-- Desarrollo Web en Entorno Servidor -->
```

```
<!-- Tema 2: Características del Lenguaje PHP -->
```

```
<!-- Ejemplo: Proyecto.php -->
```

```
<html>
```

```
    <head>
```

```
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
```

```
    <title>Desarrollo Web</title>
```

```
    <body>
```

```
        <h1>
```

```
        Nombre = $_POST['Nombre']
```

```
        $modulos = $_POST['modulos'];
```

```
        print "Nombre: $Nombre<br/>";
```

```
        foreach ($modulos as $modulo) {
```

```
            print "Modulo: $modulo<br/>";
```

```
        }
```

```
    </h1>
```

```
    </body>
```

```
</html>
```



## **Anexo VIII.- Código necesario para procesar los datos.**

---



## **Anexo IX.- Ejemplo formulario web utilizando request.**

---



# Anexo X.- Procesar datos en la misma página que el formulario.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<!-- Desarrollo Web en Entorno Servidor -->

<!-- Tema 2 : Características del Lenguaje PHP -->

<!-- Ejemplo: Procesar datos en la misma página que el formulario -->

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Desarrollo Web</title>

</head>

<body>

<?php

if (isset($_POST['enviar'])) {

$nombre = $_POST['nombre'];

$modulos = $_POST['modulos'];

print "Nombre: ".$nombre."<br />";

foreach ($modulos as $modulo) {

print "Modulo: ".$modulo."<br />";

}

} else {

}
```



# Anexo XI.- Ejemplo de validación.

---

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<!-- Desarrollo Web en Entorno Servidor -->

<!-- Tema 2 : Características del Lenguaje PHP -->

<!-- Ejemplo: Validar datos en la misma página que el formulario -->

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Desarrollo Web</title>

</head>

<body>

<?php

if (!empty($_POST['modulos']) && !empty($_POST['nombre'])) {

    $nombre = $_POST['nombre'];

    $modulos = $_POST['modulos'];

    print "Nombre: ".$nombre."<br />";

    foreach ($modulos as $modulo) {

        print "Modulo: ".$modulo."<br />";

    }

} else {

?>
```





# Anexo XII.- Solución propuesta VI.

---

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<!-- Desarrollo Web en Entorno Servidor -->

<!-- Tema 2 : Características del Lenguaje PHP -->

<!-- Ejemplo: Mostrar fecha completa a partir de día, mes y año introducidos -->

<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title>Fecha completa a partir de día, mes y año</title>
</head>
<body>
<?php
date_default_timezone_set('Europe/Madrid');

if (!empty($_POST['dia']) && !empty($_POST['mes']) && !empty($_POST['ano'])) {
    if (checkdate($_POST['mes'],$_POST['dia'],$_POST['ano']))
        {
            $fecha = mktime(0,0,0,$_POST['mes'],$_POST['dia'],$_POST['ano']);
            $numero_dia_semana = date("N", $fecha);
            switch($_POST['mes'])
            {
                case 1: $mes = "Enero";
                break;
            }
        }
}

```







# Anexo XIII.- 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 14.04:



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



Para versiones anteriores:

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
```

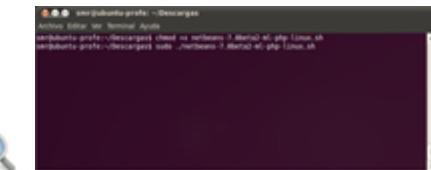
Ubuntu 13.04 y 12.10 ya no están siendo soportados por Canonical: no habrá actualizaciones de seguridad y no permiten subir nuevos paquetes a los PPAs de Launchpad.

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. Es recomendable que descargues la última versión disponible.

[NetBeans](#)



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.



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.



# Anexo XIV.- 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**.

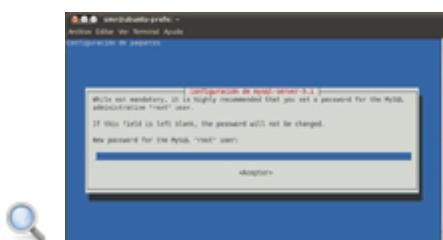
Puedes instalar los componentes uno a uno, o utilizar el comando tasksel desde la consola. Este comando viene con algunas tareas predefinidas, que nos permiten instalar con un solo comando grupos de aplicaciones. Entre las tareas que incluye tasksel se encuentra lamp-server, que incorpora los componentes de una arquitectura LAMP antes mencionados. Para instalar LAMP ejecuta desde una consola:

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

Si no te funciona el comando tasksel seguramente se debe a que no lo tengas instalado, por lo que antes deberás instalar la herramienta:

```
sudo apt-get install tasksel
```

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();  
?>
```



# Para saber más

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.

## Proceso de instalación de LAMP

# Autoevaluación

**¿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.

# Condiciones y términos de uso de los materiales

Materiales desarrollados inicialmente por el Ministerio de Educación, Cultura y Deporte y actualizados por el profesorado de la Junta de Andalucía bajo licencia Creative Commons BY-NC-SA.



MINISTERIO  
DE EDUCACIÓN  
Y FORMACIÓN PROFESIONAL



Antes de cualquier uso leer detenidamente el siguiente [Aviso legal](#)

# Historial de actualizaciones

<b>Versión: 02.01.00</b>	<b>Fecha de actualización:</b> 27/11/19	<b>Autoría:</b> Jesús Manuel Marín Navarro
<p><b>Ubicación:</b> Toda la unidad <b>Mejora (tipo 2):</b> Cambiar el formato de los ejemplos de código usando resaltado de código por colores y quitando la libretilla. De paso tabularlo bien y repasarlo</p>		
<b>Versión: 02.00.00</b>	<b>Fecha de actualización:</b> 01/07/16	<b>Autoría:</b> Julio Gómez López
<p><b>Ubicación:</b> No especificada. <b>Mejora (tipo 1):</b> Cambios realizados por Sonia Amate <b>Ubicación:</b> Toda la unidad <b>Mejora (tipo 3):</b> Se unifican las unidades 1 y 2. <b>Ubicación:</b> Añadir contenidos <b>Mejora (tipo 2):</b> Ampliar la unidad y modificar su nombre. La ampliación consistiría en añadir parte de la unidad 2 a modo de introducción a PHP, para que la tarea1 incluya una sencilla aplicación PHP (menús, formularios básicos y sentencias de control simples). Se ha llegado a esta conclusión debido a la dificultad tan grande que encuentran los alumnos en la tarea2, ya que vienen de una tarea1 que sólo les pide instalaciones (muy sencilla) a una tarea2 que les solicita un aplicación completa.</p>		
<b>Versión: 01.01.01</b>	<b>Fecha de actualización:</b> 30/09/14	<b>Autoría:</b> Sonia Amate Garrido
<p>Corrección de erratas y enlaces que no funcionan. Adición de actividades multimedia que se habían perdido. Actualización de los contenidos a las nuevas versiones de SW: jdk8, Netbeans 8.0.1</p>		
<b>Versión: 01.00.00</b>	<b>Fecha de actualización:</b> 20/04/14	
<p>Versión inicial de los materiales.</p>		