

Tema 1. Selección de arquitecturas y herramientas de programación

DESARROLLO EN ENTORNO SERVIDOR 2º DAW

MARÍA CRIADO DOMÍNGUEZ

Índice

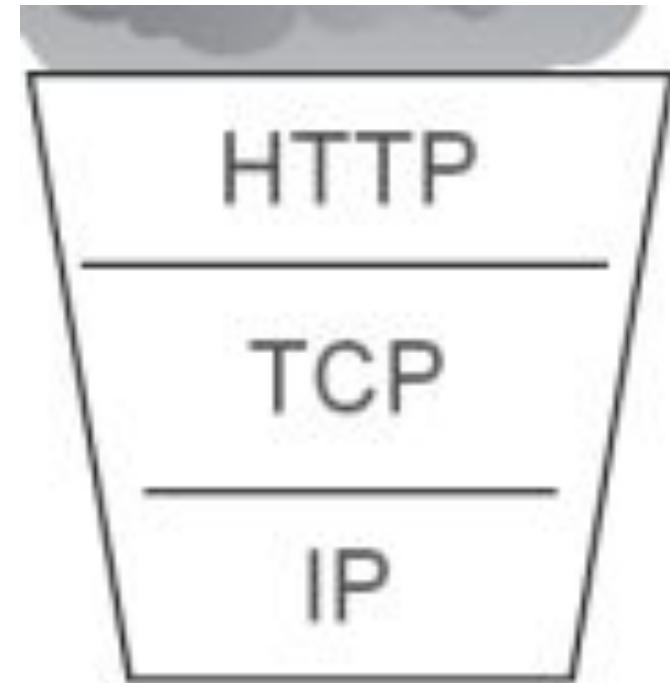
- Modelos de programación en entornos cliente / servidor.
- Generación dinámica de páginas web.
- Tecnologías de programación en entorno servidor.
- Integración con los servidores web.
- Herramientas de programación.

Modelos de programación en entornos cliente/servidor

- La World Wide Web (o la Web, como se conoce comúnmente) representa un universo de información accesible globalmente a través de internet. Está formada por un conjunto de recursos interconectados que conforman el conocimiento humano actual.
- El funcionamiento de la Web es posible debido a la coexistencia de una serie de componentes software y hardware. Estos elementos abarcan desde los componentes físicos de internet (hubs, repetidores, puentes, pasarelas, encaminadores, etc.) y los protocolos de comunicaciones (TCP, IP, HTTP, FTP, SMTP, etc.) hasta la utilización del sistema de nombres de dominio (DNS) para la búsqueda y recuperación de recursos o la utilización de software específico para proveer y consumir dichos recursos.

Modelos de programación en entornos cliente/servidor

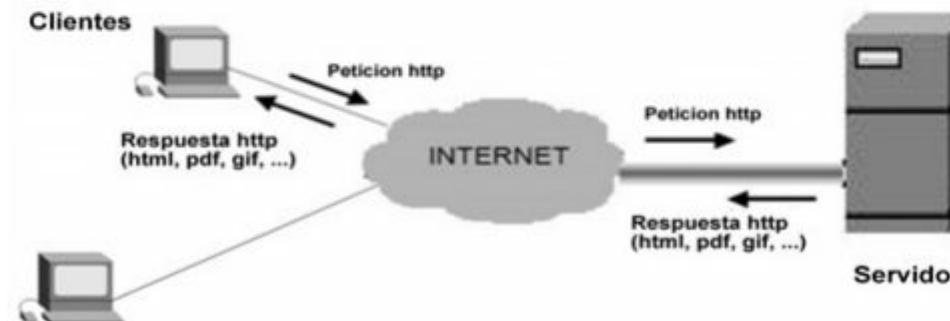
- El desarrollo en entornos web debe tener en cuenta la distribución de los elementos y la función que tiene cada uno de ellos. La configuración arquitectónica más habitual se basa en el modelo denominado Cliente/ Servidor
- Basado en la idea de servicio, en el que el cliente es un componente consumidor de servicios y el servidor es un proceso proveedor de servicios.
- Además, esta relación está robustamente cimentada en el intercambio de mensajes como el único elemento de acoplamiento entre ambos.



Modelos de programación en entornos cliente/servidor

- El agente que solicita la información se denomina **cliente**, mientras que el componente software que responde a esa solicitud es el que se conoce como **servidor**. En un proceso habitual el cliente es el que inicia el intercambio de información, solicitando datos al servidor, que responde enviando uno o más flujos de datos al cliente.
- Además de la transferencia de datos real, este intercambio puede requerir información adicional, como la autenticación del usuario o la identificación del archivo de datos que vayamos a transferir

Modelo Cliente/Servidor



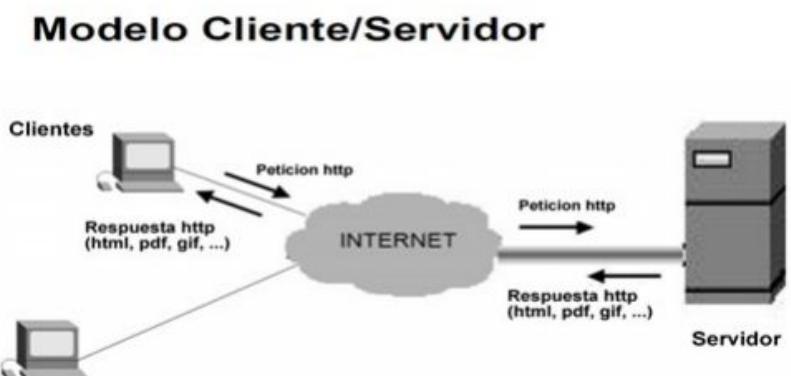
Modelos de programación en entornos cliente/servidor

- Los pasos son los siguientes:

1. Tu ordenador solicita a un servidor web una página con extensión .htm, .html, .xhtml, .php, .asp, .jsp
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.



Generación dinámica de páginas web

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.

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.

Generación dinámica de páginas web



Generación dinámica de páginas web

Páginas web dinámicas.

Cliente

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.

Generación dinámica de páginas web

Páginas web dinámicas.

Servidor

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).

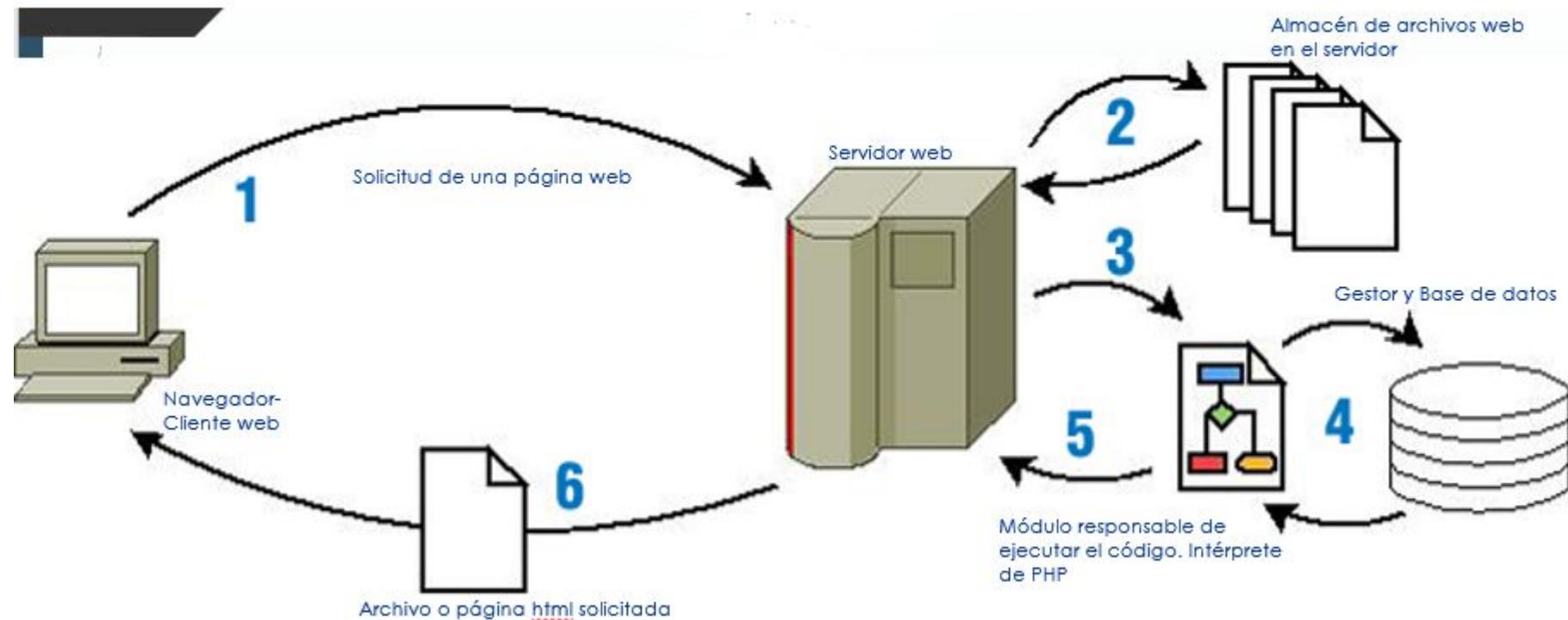
Generación dinámica de páginas web

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

1. El cliente web (navegador) de tu ordenador solicita a un servidor web una página web.
2. El servidor busca esa página y la recupera.
3. En el caso de que se trate de una página web dinámica, es decir, que su contenido deba ejecutarse para obtener el HTML que se devolverá, el servidor web contacta con el módulo responsable de ejecutar el código y se lo envía.
4. Como parte del proceso de ejecución, puede ser necesario obtener información de algún repositorio, 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.

Generación dinámica de páginas web

Páginas web dinámicas. El esquema de funcionamiento de una página web dinámica.



Generación dinámica de páginas web

Páginas web estáticas vs dinámicas.

Las primeras páginas web que se crearon en Internet fueron páginas estáticas. A esta web compuesta por páginas estáticas se le considera la primera generación. La segunda generación de la web surgió gracias a las páginas web dinámicas. Tomando como base las web dinámicas, han ido surgiendo otras tecnologías que han hecho evolucionar Internet hasta llegar a lo que ahora conocemos.

Las páginas web estáticas sólo necesitan un navegador para interpretarlas. 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.

Generación dinámica de páginas web

Páginas web estáticas vs dinámicas.

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.

Generación dinámica de páginas web

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.

Generación dinámica de páginas web

Aplicaciones web 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 móviles

Generación dinámica de páginas web

Aplicaciones web 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)

Generación dinámica de páginas web

Ejecución de código

- 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.

Generación dinámica de páginas web

Ejecución de código

- Estas dos tecnologías se complementan una con otra.
- Esta división es así porque el código que se ejecuta en el cliente web (en el navegador) no tiene, o mejor dicho tradicionalmente no tenía, acceso a los datos que se almacenan en el servidor.
- 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

Tecnologías para programación web del lado del servidor.

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

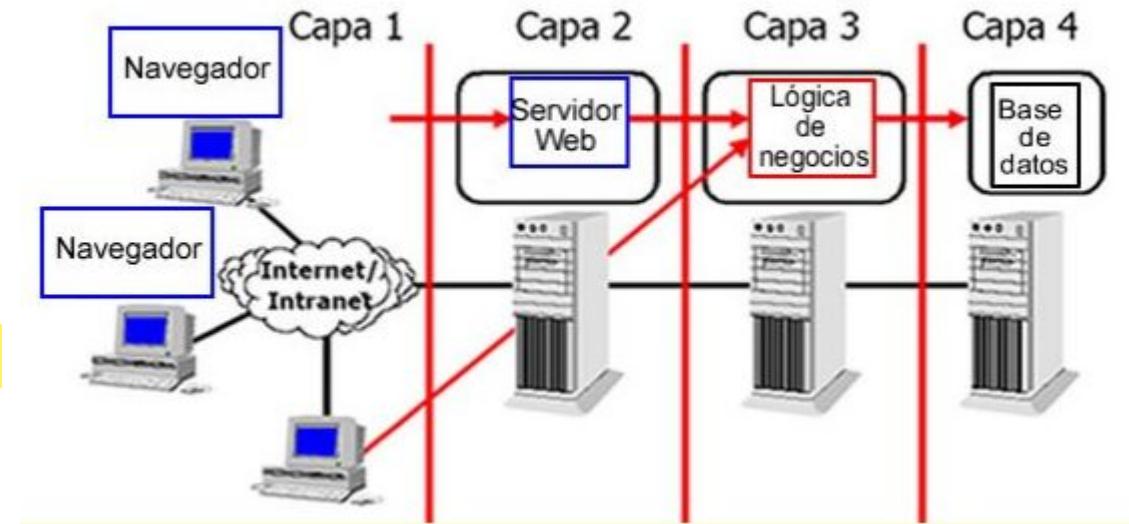
- Un servidor web para recibir las peticiones de los clientes web (normalmente navegadores) y enviarles la página que solicitan (una vez generada puesto que hablamos de páginas web dinámicas).
- El servidor web debe conocer el procedimiento a seguir para generar la página web: qué módulo se encargará 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.

Tecnologías para programación web del lado del servidor.

- 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

Tecnologías para programación web del lado del servidor.

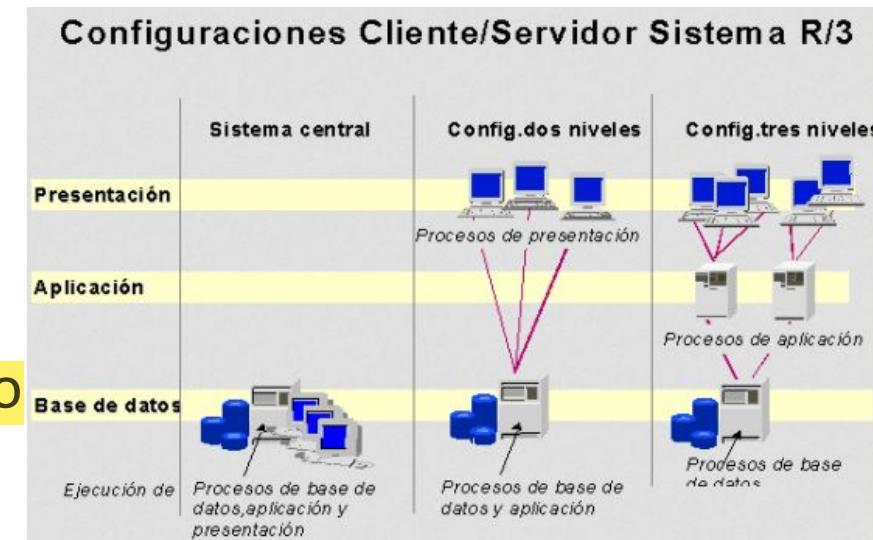
- En una aplicación puedes distinguir, de forma general:
 - Funciones de presentación (se encarga de dar formato a los datos para presentárselo al usuario final)
 - Lógica (utiliza los datos para ejecutar un proceso y obtener un resultado),
 - Persistencia (que mantiene los datos almacenados de forma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



Tecnologías para programación web del lado del servidor.

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.

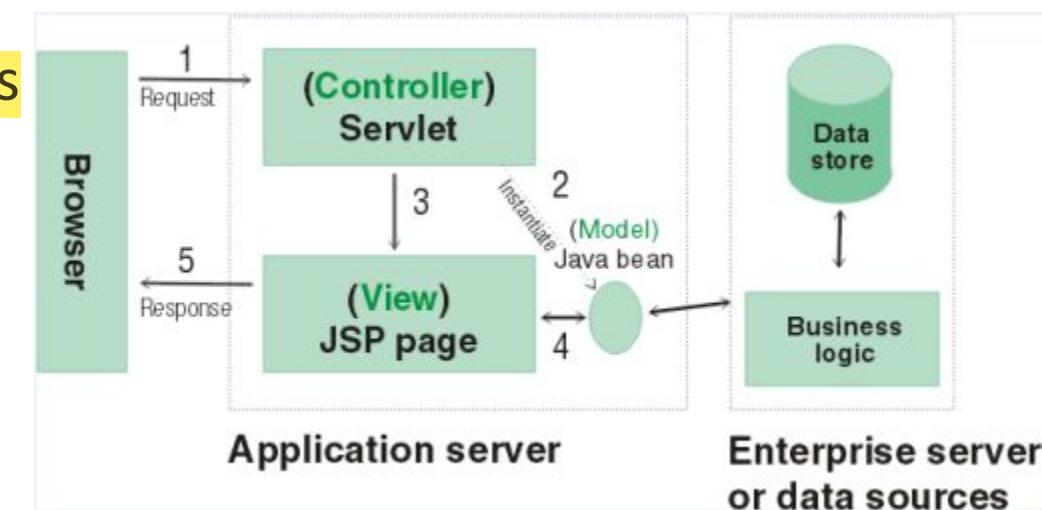


Tecnologías para programación en el servidor.

Arquitectura y Plataformas

Java EE

- 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.



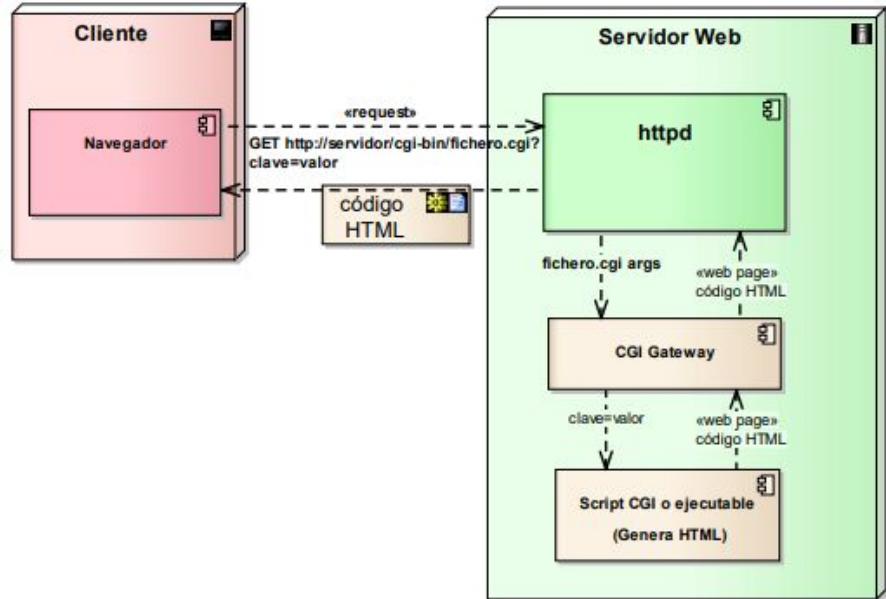
Tecnologías para programación en el servidor.

Arquitectura y Plataformas

AMP

- Son las siglas de Apache, MySQL y PHP/Perl/Python. Las dos primeras siglas hacen referencia al servidor web (Apache) y al servidor de base de datos (MySQL). La última se corresponde con el lenguaje de programación utilizado, que puede ser PHP, Perl o Python,
- Dependiendo del sistema operativo que se utilice para el servidor, se utilizan las siglas LAMP (para Linux), WAMP (para Windows) o MAMP (para Mac).
- 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

Tecnologías para programación en el servidor. Arquitectura y Plataformas



CGI

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.

Tecnologías para programación en el servidor. Arquitectura y Plataformas



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.

Tecnologías para programación en el servidor.

Arquitectura y Plataformas

Selección de una arquitectura de programación web

Como has visto, hay muchas decisiones que debes tomar antes aún de comenzar el desarrollo de una aplicación web:

- La arquitectura
- El lenguaje de programación
- El entorno de desarrollo
- El gestor de bases de datos
- El servidor web
- La estructurara de la aplicación

Tecnologías para programación en el servidor.

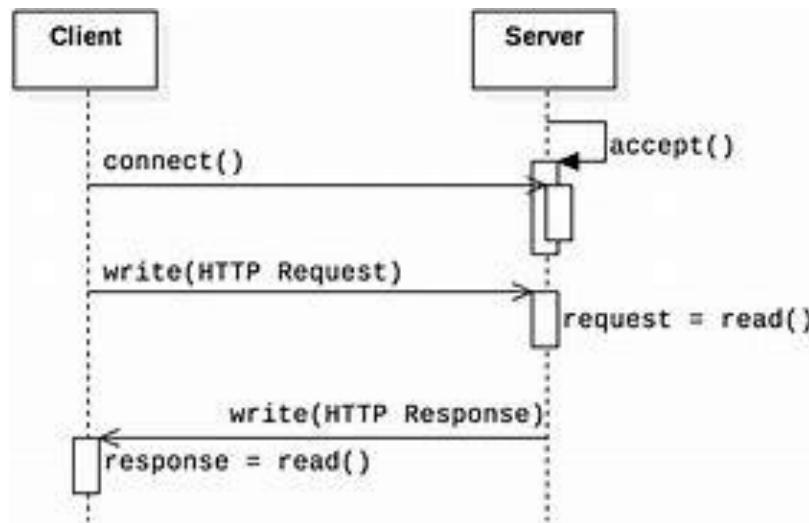
Arquitectura y Plataformas

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

- ¿Qué tamaño tiene el proyecto?
- ¿Qué lenguajes de programación conozco? ¿Vale la pena el esfuerzo de aprender uno nuevo?
- ¿Voy a usar herramientas de código abierto o herramientas propietarias? ¿Cuál es el coste de utilizar soluciones comerciales?
- ¿Voy a programar la aplicación yo solo o formaré parte de un grupo de programadores?
- ¿Cuento con algún servidor web o gestor de base de datos disponible o puedo decidir libremente utilizar el que crea necesario?
- ¿Qué tipo de licencia voy a aplicar a la aplicación que desarrolle?

Tecnologías para programación en el servidor.

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**.
- 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.

Tecnologías para programación en el servidor.

Integración con el servidor web

- 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.

Tecnologías para programación en el servidor.

Integración con el servidor web

- 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.
- En la mayoría de ocasiones no es necesario utilizar un servidor de aplicaciones completo, 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

Tecnologías para programación en el servidor.

Integración con el servidor web

- 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.

Lenguajes

Una de las **diferencias más notables** entre un **lenguaje de programación web** y otro **es la manera en que se ejecutan en el servidor web**. Debes distinguir tres grandes grupos:

- Lenguajes de guiones (scripting).
- Lenguajes compilados a código nativo
- Lenguajes compilados a código intermedio.

Lenguajes

Lenguajes de guiones (scripting)

- Son aquellos en los que los programas se ejecutan directamente a partir de su código fuente (Conjunto de instrucciones que componen un programa, y que no son ejecutables directamente, sino que deben traducirse utilizando un compilador, intérprete o similar antes de que pueda ser ejecutado por la máquina) original.
- Se almacenan normalmente en un fichero de texto plano. Cuando el servidor web necesita ejecutar código programado en un lenguaje de guiones, le pasa la petición a un intérprete, que procesa las líneas del programa y genera como resultado una página web.
- De los lenguajes que estudiaste anteriormente, pertenecen a este grupo Perl, Python, PHP y ASP.

Lenguajes

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.

Lenguajes

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

Lenguajes

Ventajas e inconvenientes

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

Lenguajes

Lenguajes compilados a código intermedio

- Son lenguajes en los que el código fuente original se traduce a un código intermedio, independiente del procesador, antes de ser ejecutado
- Es la forma en la que se ejecutan por ejemplo las aplicaciones programadas en Java, y lo que hace que puedan ejecutarse en varias plataformas distintas.
- En la programación web, operan de esta forma los lenguajes de las arquitecturas Java EE (servlets y páginas JSP)

Lenguajes

Ventajas e inconvenientes

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)

Código embebido en html

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.

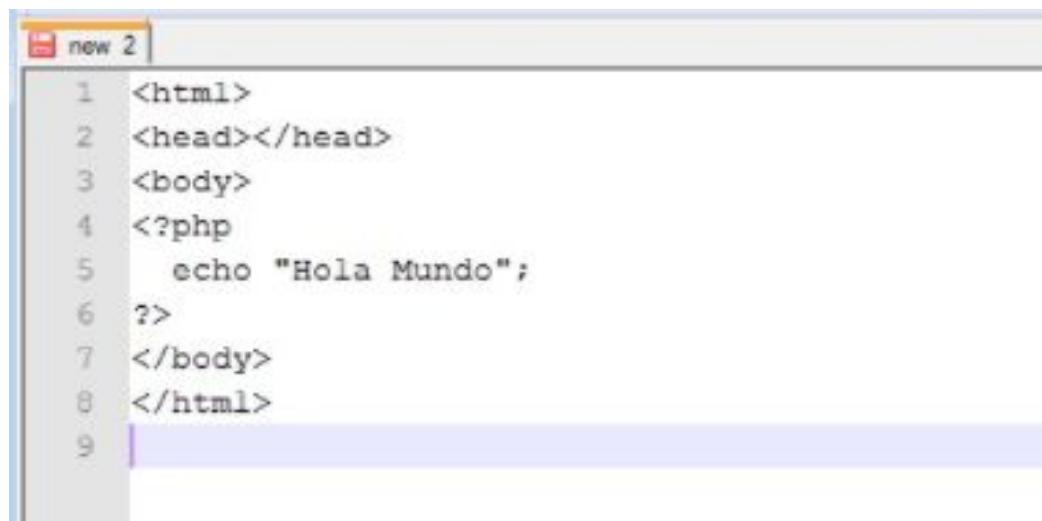
```
echo "Content-Type: text/html"
echo
echo

echo "<html><head></head>"
echo "<body>"
echo "Parameters are:<br />"
user=`echo $QUERY_STRING | cut -d"&" -f 1 | cut -d"=" -f 2`
pass=`echo $QUERY_STRING | cut -d"&" -f 2 | cut -d"=" -f 2`

echo $user $pass
echo "</body></html>"
```

Código embebido en html

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



The image shows a screenshot of a code editor window titled "new 2". The code editor displays the following PHP code:

```
1 <html>
2 <head></head>
3 <body>
4 <?php
5   echo "Hola Mundo";
6 ?>
7 </body>
8 </html>
9
```

The code consists of an HTML document structure with a single PHP echo statement that outputs the string "Hola Mundo". The code editor interface includes a toolbar at the top and a status bar at the bottom.

Herramientas de programación

- 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.

Preparación del entorno

Visual Studio Code

<https://code.visualstudio.com/>

Ubuntu server

<https://ubuntu.com/download/server>

Php

<https://www.php.net/downloads>

Apache

<https://httpd.apache.org/>

Tema 2. Inserción de código en páginas Web

DESARROLLO EN ENTORNO SERVIDOR 2º DAW

MARÍA CRIADO DOMÍNGUEZ

Índice

1. PHP
2. Lenguajes embebidos en HTML
3. Obtención del lenguaje de marcas a mostrar en el cliente.
4. Etiquetas para inserción de código.
5. Bloques de código.
6. Tipos de datos. Conversiones entre tipos de datos. Variables. Ámbito de utilización de las variables
7. Constantes

Elemento del lenguaje PHP

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

PHP está enfocado principalmente a la programación de scripts del lado del servidor, por lo que se puede hacer cualquier cosa que pueda hacer otro programa CGI, como recopilar datos de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies.

<https://www.php.net/manual/es/intro-whatis.php>

Elemento del lenguaje PHP

PHP puede emplearse en todos los sistemas operativos principales, incluyendo Linux, muchas variantes de Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, macOS y probablemente otros más. PHP admite la mayoría de servidores web de hoy en día, incluyendo Apache, IIS, y muchos otros. Esto incluye cualquier servidor web que pueda utilizar el binario de PHP FastCGI, como lighttpd y nginx. PHP funciona tanto como módulo como procesador de CGI.

De modo que con PHP, se tiene la libertad de elegir el sistema operativo y el servidor web. Además, se tiene la posibilidad de utilizar programación por procedimientos o programación orientada a objetos (POO), o una mezcla de ambas.

Elemento del lenguaje PHP

Con PHP no se está limitado a generar HTML. Entre las capacidades de PHP se incluyen la creación de imágenes, ficheros PDF ...

También se puede generar fácilmente cualquier tipo de texto, como HTML y cualquier otro tipo de fichero XML. PHP puede autogenerar estos ficheros y guardarlos en el sistema de ficheros.

Una de las características más potentes y destacables de PHP es su soporte para un amplio abanico de bases de datos. Escribir una página web con acceso a una base de datos es increíblemente simple utilizando una de las extensiones específicas de bases de datos o utilizar una capa de abstracción como PDO, o conectarse a cualquier base de datos que admita el estándar de Conexión Abierta a Bases de Datos por medio de la extensión ODBC.

Elemento del lenguaje PHP

- Como PHP se ejecuta del lado del servidor sólo puede tener acceso a los datos del propio servidor.
- No puede acceder a los recursos del cliente
- No puede saber qué hora es en el cliente
- No puede acceder a los archivos del cliente
 - Salvo la excepción de las Cookies

Elemento del lenguaje PHP

Fichero php.ini

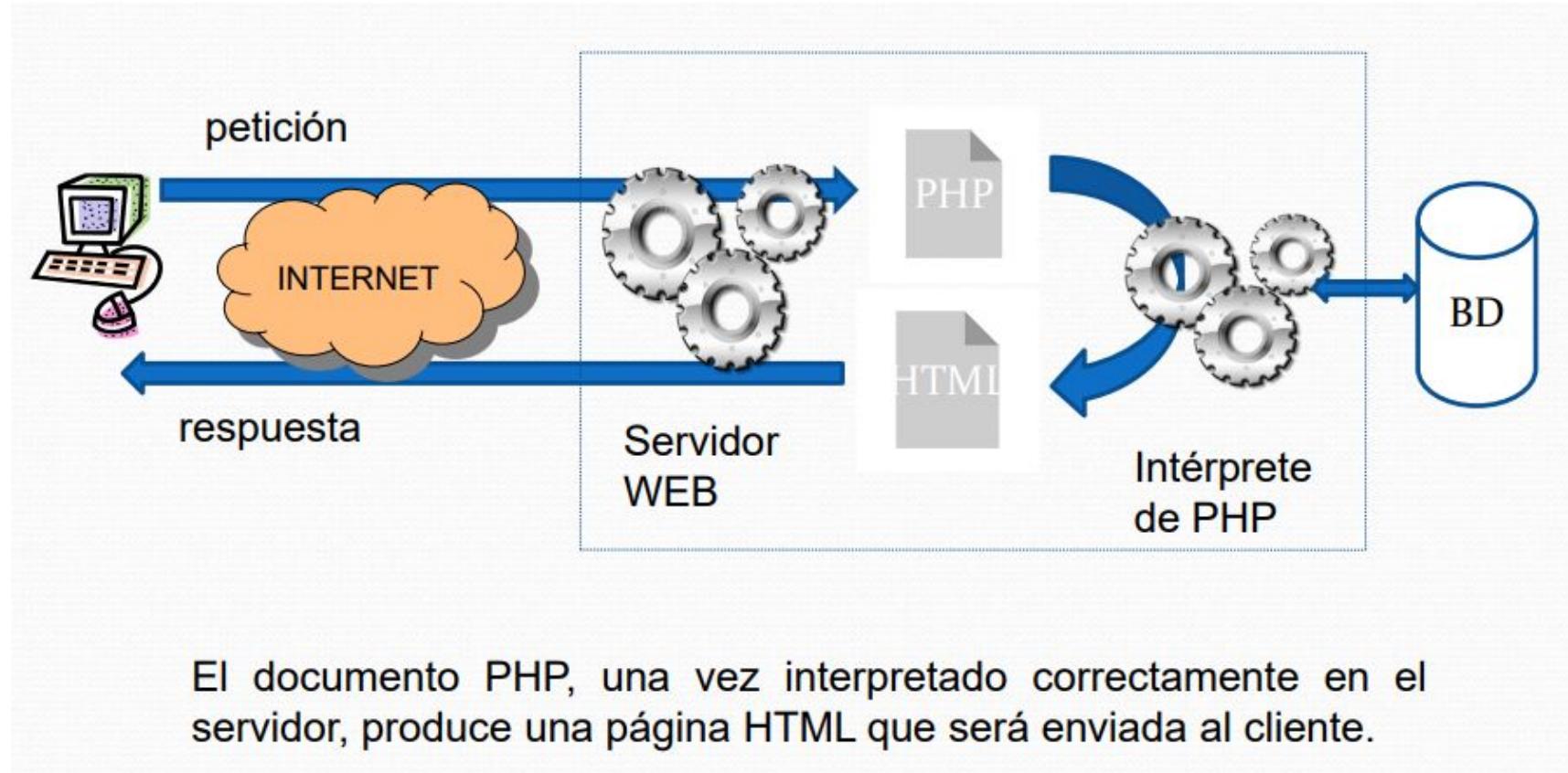
- Indica una serie de valores que determinan el comportamiento del intérprete PHP.
- Se encuentra en el directorio raíz de PHP
- Las instrucciones del fichero se denominan directivas
 - [PHP: Listado de directivas de php.ini – Manual](#)
- Las directivas están formadas por una pareja de clave y valor.
- Las directivas que comienzan por ; están comentadas y son ignoradas por el motor del intérprete.
- El fichero php.ini se lee cada vez que se arranca el servidor web.

Lenguajes embebidos en HTML

Los programas escritos en PHP, además 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.

Lenguajes embebidos en HTML



Etiquetas para inserción de código

- **Comentarios**

PHP admite comentarios al estilo de 'C', 'C++' y de consola de Unix (estilo de Perl).

```
// Esto es un comentario al estilo de c++ de una sola  
línea
```

```
/* Esto es un comentario multilínea  
y otra línea de comentarios */
```

```
# Esto es un comentario al estilo de consola de una s  
ola línea
```

Etiquetas para inserción de código

Hay dos tipos de etiquetas para delimitar bloques de código PHP:

- 1^a Forma. Para servir documentos HTML, XHTML o XML:

```
<?php  
    Instrucciones PHP
```

```
?>
```

- 2^a Formato corto:

```
<?  
    Instrucciones PHP  
?>
```

Etiquetas para inserción de código

Sólo la primera forma asegura portabilidad.

- Para usar el formato corto debemos activar en el fichero php.ini la directiva:

```
short_open_tag = on
```

Etiquetas para inserción de código

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, ...);
```

```
echo "Hola mundo";
```

```
echo "Hola ", "mundo";
```

Etiquetas para inserción de código

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);
```

```
print "Hola mundo";
```

```
print "Hola " . "mundo";
```

Etiquetas para inserción de código

Generación de código HTML.

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.

```
printf(string $format, mixed $args = ?, mixed $... = ?
```

```
printf("%d", "17,999")
```

```
printf("%f", "17,999")
```

```
printf("%s", "17,999")
```

Etiquetas para inserción de código

Generación de código HTML.

Esta función muestra información estructurada sobre una o más expresiones incluyendo su tipo y valor. Las matrices y los objetos son explorados recursivamente con valores sangrados para mostrar su estructura.

```
var_dump("maria", 3.14);
```

Etiquetas para inserción de código

- Ejemplo
 - Hola Mundo

Etiquetas para inserción de código

Generación de código HTML

Uso de \n para generar código HTML legible

Código PHP

```
print ("<P>Párrafo 1</P>\n");
print ("<P>Párrafo 2</P>\n");
```

Código HTML

```
<P>Párrafo 1</P>
<P>Párrafo 2</P>
```

Salida

```
Párrafo 1
Párrafo 2
```

Etiquetas para inserción de código

Heredoc

- Cuando tenemos necesidad de escribir largos bloques de código HTML, incluso con variables intercaladas, podemos usar la construcción heredoc que nos permite escribir grandes cantidades de texto, sin necesidad de escapar caracteres en su interior.
- También podemos almacenarlo dentro de una variable

```
<<< TEXT
```

```
    Todo el texto que queremos <p>"con comillas" </p> o las  
variables... todo hasta
```

```
TEXT;
```

Variables

- Como en todos los lenguajes de programación, en PHP puedes crear variables para almacenar valores. Las variables en PHP siempre deben comenzar por el signo \$.
- Los nombres de las variables deben comenzar por letras o por el carácter _, y pueden contener también números. Sin embargo, al contrario que en muchos otros lenguajes, en PHP no es necesario declarar una variable ni especificarle un tipo (entero, cadena,...) concreto.
- Para empezar a usar una variable, simplemente asígnale un valor utilizando el operador =.

```
$mi_variable = 7;
```

Variables

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.
- Objetos. Variable de una clase

Variables

Funciones con variables

`gettype()`. Devuelve el tipo de la variable

`settype()` - Establece el tipo de una variable

`is_array()` - Comprueba si una variable es un array

`is_bool()` - Comprueba si una variable es de tipo booleano

`is_float()` - Comprueba si el tipo de una variable es float

`is_int()` - Comprueba si el tipo de una variable es integer

`is_null()` - Comprueba si una variable es null

`is_numeric()` - Comprueba si una variable es un número o un string numérico

`is_object()` - Comprueba si una variable es un objeto

`is_string()` - Comprueba si una variable es de tipo string

Variables

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
```

Variables

La conversión automática que realiza PHP no siempre es lo que queremos.

PHP permite otras conversiones implícitas de tipos :

- (int) : Fuerza la conversión a entero
- (real), (double), (float): Fuerza la conversión a coma flotante.
- (string): Fuerza la conversión a cadena de caracteres.
- (array): Fuerza la conversión a matriz
- (object): Fuerza la conversión a un objeto.

Variables

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)
```

Variables

Variable de variables

Se pueden crear nombres de variables dinámicamente anteponiendo **\$\$** a una variable.

La variable de variable toma su nombre del valor de otra variable previamente declarada.

```
<?php
    $var = "uno";
    $$var = "dos";
    print ($var); // produce el texto: "uno"
    print ($uno); // produce el texto: "dos"
    print ($$var); // produce el texto: "dos"
```

Variables por referencia

La variable no contiene un valor sino la dirección de otra variable.

En PHP las variables se pasan por referencia al precederlas del símbolo &.

El signo & indica que se está almacenando la dirección de la variable y no su contenido.

```
$ref = &$var;
```

Variab les Especiales

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 .

- `$_SERVER`. Contiene información sobre el entorno del servidor web y de ejecución.
- `$_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.

Variables En la URL

Un mecanismo práctico aunque no muy seguro de intercambio de información entre una página y otra consiste en pasar las variables a través de un sufijo en la URL de la página llamada.

`http://www.mipagina.es/pagina.php?nombre='maría'`

El programa PHP recibe estas variables dentro de las matrices superglobales `$_REQUEST` o `$_GET`,

`GET ['nombre']`

Utilizar la función `urlencode()` cuando los valores de las variables en la URL contienen caracteres especiales.

Variables

Ámbito

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 aparece una asignación fuera de la función, se le considerará una variable distinta.

```
$a = 1;  
  
function prueba() { //no se tiene acceso a la variable $a  
    $b = $a; // no tiene valor asignado (su valor es null)  
}
```

Variables

Ámbito

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  
}
```

Variables

Ámbito

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.

Constantes

- Una constante es un identificador de un dato que no cambia de valor durante toda la ejecución de un programa.
- Las constantes no se asignan con el operador =, sino con la función define :

```
define(nombre_constante_entre_comillas, dato_constante);  
define ("PI", 3.1416)
```

- No llevan \$ delante
- La función defined("PI") devuelve TRUE si existe la constante.
- Son siempre globales por defecto.
- Sólo se pueden definir constantes de los tipos escalares (boolean, integer, double, string)
- No se puede cambiar su valor ,se generará un error

Constantes predefinidas

Dependen de las extensiones que se hayan cargado en el servidor, aunque hay constantes predefinidas que siempre están presentes :

- `PHP_VERSION`: Indica la versión de PHP que se está utilizando.
- `PHP_OS`: Nombre del sistema operativo que ejecuta PHP.
- `TRUE`
- `FALSE`
- `E_ERROR`: Indica los errores de interpretación que no se pueden recuperar.
- `E_PARSE`: Indica errores de sintaxis que no se pueden recuperar.
- `E_ALL`: Representa a todas las constantes que empiezan por `E_`.

Inclusión de ficheros externos

La inclusión de ficheros externos se consigue con:

- `include()`
- `require()`

Ambos incluyen y evalúan el fichero especificado

Diferencia: en caso de error `include()` produce un warning y `require()` un error fatal

Se usará `require()` si al producirse un error debe interrumpirse la carga de la página

- `require_once()`
- `Include_once()`

Tipos de datos especiales

Fechas

- Con frecuencia cuando creamos webs o apps tenemos que trabajar con fechas y calendarios. Por ejemplo, en la página web de un hotel o un restaurante es posible que trabajemos con fechas y horas de comienzo de reserva, de fin de reserva, etc. PHP dispone de funciones nativas para facilitar el trabajo con fechas, horas y tiempos.
- El manejo del “tiempo” es un aspecto controvertido en la programación.
- La primera dificultad y más obvia es que existen cientos de dispositivos electrónicos y no todos manejan la misma fecha.

Tipos de datos especiales

Fechas

Función time()

- La función `time()` devuelve la hora GMT actual medida como el número de segundos desde el 1 de enero de 1970 00:00:00 GMT (hora del meridiano de Greenwich) obtenidos a partir de la hora actual local del servidor.
- Cuando hablamos de “hora actual del servidor” nos referimos a una hora local, que es distinta según el país donde nos encontramos.
- La función `time()` devuelve un valor numérico entero largo, por ejemplo 1335169779.

Tipos de datos especiales

Fechas

Función date_default_timezone

`date_default_timezone_get()`

- informa de la zona horaria establecida en el servidor en el que se ejecuta el script

`date_default_timezone_set('Europe/Madrid');`

- Establece la zona horaria donde se encuentra el servidor
- Ya tiene en cuenta horario de verano e invierno

Tipos de datos especiales

Fechas

Función date

Para transformar ese número en una fecha “entendible por las personas” usamos la función date, cuya sintaxis general es:

```
date ("formato de salida", valorTimeValido)
```

<https://www.php.net/manual/es/function.date.php>

Tipos de datos especiales

Fechas

Función strtotime

- Esta función espera que se proporcione una cadena que contenga un formato de fecha en Inglés US
- Intentará convertir ese formato a marca de tiempo Unix (el número de segundos desde el 1 de Enero del 1970 00:00:00 UTC)
- Si no se le proporciona el parámetro now, tomará como fecha actual la que tengamos establecida en el script o en caso contrario la del servidor.

```
strtotime ( string $time [, int $now = time() ] ) : int
```

Tipos de datos especiales

Fechas

Función strptime

Las fechas en los formatos m/d/y o d-m-y no son ambiguas al observar el separador entre los distintos componentes:

- si el separador es una barra (/), se asume el formato norteamericano m/d/y; mientras que si el separador es un guion (-) se asume el formato europeo d-m-y.
- Si el año se proporciona en un formato de dos dígitos y el separador es un guion (-), la cadena de la fecha se analiza como y-m-d.
- Para evitar esta ambigüedad potencial es mejor usar fechas ISO 8601 (YYYY-MM-DD)

Tipos de datos especiales

Fechas

Función mktime

Obtener la marca de tiempo Unix de una fecha

```
mktime(  
    int $hour = date("H"),  
    int $minute = date("i"),  
    int $second = date("s"),  
    int $month = date("n"),  
    int $day = date("j"),  
    int $year = date("Y"),  
    int $is_dst = -1  
)
```

Tipos de datos especiales

Fechas

Función getdate

Devuelve un array asociativo que contiene la información de la fecha de timestamp, o el momento local actual si no se da timestamp.

```
getdate(int $timestamp = time())
```

```
Array
(
    [seconds] => 40
    [minutes] => 58
    [hours] => 21
    [mday] => 17
    [wday] => 2
    [mon] => 6
    [year] => 2003
    [yday] => 167
    [weekday] => Tuesday
    [month] => June
    [0] => 1055901520
)
```

Tipos de datos especiales

Fechas

DateTime y DateTimeInterval

Este método es un estilo orientado a objetos para obtener la diferencia entre dos fechas, este es también el más fácil ya que no requiere el cálculo manual de las fechas y recomendado ya que es de la más reciente versión de PHP.

```
$firstDate = new DateTime ("2019-01-01") ;  
$secondDate = new DateTime ("2020-03-04") ;  
$intvl = $firstDate->diff ($secondDate) ;
```

Tema 3. Programación basada en lenguajes de marcas con código embebido

DESARROLLO EN ENTORNO SERVIDOR

2º DAW

MARÍA CRIADO DOMÍNGUEZ

Índice

1. Operadores
2. Tomas de decisión.
3. Bucles.
4. Arrays.
5. Funciones.
6. Paso de parámetros. Devolución de valores.
7. Recuperación y utilización de información proveniente del cliente Web.
8. Interacción con el usuario: formularios.
9. Procesamiento de la información introducida en un formulario.

Operadores

- Una expresión es una combinación de operadores, variables, constantes, y funciones que está formada correctamente, es decir, es válida sintácticamente, y que tiene sentido, o lo que es igual, es válida semánticamente.
- Toda expresión produce un valor al ser procesada.
- La mayor parte del código que realicemos en PHP van a ser expresiones.

Operadores

Un operador es un elemento, palabra o símbolo, que al aplicarlo sobre otros elementos, los operandos, proporciona un valor.

Según el tipo de operación que realizan, los operadores más utilizados se clasifican en:

- **Aritméticos:** son los operadores empleados en las operaciones aritméticas: suma, resta,...
- **Asignación:** se utilizan para almacenar un dato dentro de una variable.
- **De bit:** permiten evaluar y manipular bits determinados dentro de un entero.
- **Comparación:** como su nombre indica, comparan dos elementos.
- **Lógicos:** el resultado obtenido de estos operadores se valora a true o false, Cierto o Falso.

Operadores Aritméticos

Operadores

Suma

Resta

Multiplicación

División

Módulo

Exponenciación

Negación

Representación

$\$x + \y

$\$x - \y

$\$x * \y

$\$x / \y

$\$x \% \y (resto de la división)

$\$x ** \y

$-\$x$

Operadores

Comparación

Ejemplo	Nombre	Resultado
<code>\$a == \$b</code>	Igual	TRUE si <code>\$a</code> es igual a <code>\$b</code> después de la manipulación de tipos.
<code>\$a === \$b</code>	Idéntico	TRUE si <code>\$a</code> es igual a <code>\$b</code> , y son del mismo tipo.
<code>\$a != \$b</code>	Diferente	TRUE si <code>\$a</code> no es igual a <code>\$b</code> después de la manipulación de tipos.
<code>\$a <> \$b</code>	Diferente	TRUE si <code>\$a</code> no es igual a <code>\$b</code> después de la manipulación de tipos.
<code>\$a !== \$b</code>	No idéntico	TRUE si <code>\$a</code> no es igual a <code>\$b</code> , o si no son del mismo tipo.
<code>\$a < \$b</code>	Menor que	TRUE si <code>\$a</code> es estrictamente menor que <code>\$b</code> .
<code>\$a > \$b</code>	Mayor que	TRUE si <code>\$a</code> es estrictamente mayor que <code>\$b</code> .
<code>\$a <= \$b</code>	Menor o igual que	TRUE si <code>\$a</code> es menor o igual que <code>\$b</code> .
<code>\$a >= \$b</code>	Mayor o igual que	TRUE si <code>\$a</code> es mayor o igual que <code>\$b</code> .
<code>\$a <=> \$b</code>	Nave espacial	Un integer menor que, igual a, o mayor que cero cuando <code>\$a</code> es respectivamente menor que, igual a, o mayor que <code>\$b</code> . Disponible a partir de PHP 7.

Operadores

Comparación

Si se compara un número con un string o la comparación implica strings numéricos, entonces cada string es convertido en un número y la comparación se realiza numéricamente.

PHP 7 introduce un nuevo tipo de operador, que se puede utilizar para comparar expresiones llamado nave espacial cuyo resultado es:

`$a <=> $b` evalúa a:

0 si `$a == $b`

-1 si `$a < $b`

1 Si `$a > $b`

Operadores

Asignación

Operador	Ejemplo	Equivalencia
=	\$a=\$b;	\$a toma el valor de \$b
+=	\$a += \$b;	\$a = \$a + \$b
-=	\$a -= \$b	\$a = \$a - \$b
*=	\$a *= \$b;	\$a = \$a * \$b;
/=	\$a /= \$b;	\$a = \$a / \$b;
%=	\$a %= \$b;	\$a = \$a % \$b;
.=	\$a .= \$b;	\$a = \$a . \$b;

Operadores

Asignación

Los operadores de incremento y decremento sólo afectan a números y strings, sin afectar a arrays, objects o resources.

Decrementar un valor NULL no tiene efecto, pero si se incrementa se obtiene 1.
Incrementar o decrementar booleanos no tiene efecto.

Operador	Efecto
<code>++\$x</code>	Incrementa \$x en 1 y devuelve \$x
<code>\$x++</code>	Retorna \$x y luego incrementa \$x en 1
<code>--\$x</code>	Decrementa \$x en 1 y devuelve \$x
<code>\$x--</code>	Retorna \$x y luego decrementa \$x en 1

Operadores Lógicos

Ejemplo	Nombre	Resultado
<code>\$a and \$b</code>	And (y)	TRUE si tanto <code>\$a</code> como <code>\$b</code> son TRUE .
<code>\$a or \$b</code>	Or (o inclusivo)	TRUE si cualquiera de <code>\$a</code> o <code>\$b</code> es TRUE .
<code>\$a xor \$b</code>	Xor (o exclusivo)	TRUE si <code>\$a</code> o <code>\$b</code> es TRUE , pero no ambos.
<code>! \$a</code>	Not (no)	TRUE si <code>\$a</code> no es TRUE .
<code>\$a && \$b</code>	And (y)	TRUE si tanto <code>\$a</code> como <code>\$b</code> son TRUE .
<code>\$a \$b</code>	Or (o inclusivo)	TRUE si cualquiera de <code>\$a</code> o <code>\$b</code> es TRUE .

Operadores Bit

Ejemplo	Nombre	Resultado
<code>\$a & \$b</code>	And (y)	Los bits que están activos en ambos <code>\$a</code> y <code>\$b</code> son activados.
<code>\$a \$b</code>	Or (o inclusivo)	Los bits que están activos ya sea en <code>\$a</code> o en <code>\$b</code> son activados.
<code>\$a ^ \$b</code>	Xor (o exclusivo)	Los bits que están activos en <code>\$a</code> o en <code>\$b</code> , pero no en ambos, son activados.
<code>~ \$a</code>	Not (no)	Los bits que están activos en <code>\$a</code> son desactivados, y viceversa.
<code>\$a << \$b</code>	Shift left(desplazamiento a izquierda)	Desplaza los bits de <code>\$a</code> , <code>\$b</code> pasos a la izquierda (cada paso quiere decir "multiplicar por dos").
<code>\$a >> \$b</code>	Shift right (desplazamiento a derecha)	Desplaza los bits de <code>\$a</code> , <code>\$b</code> pasos a la derecha (cada paso quiere decir "dividir por dos").

Tomas de decisión

- 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;
 - **sentencias de bucle**, con las que puedes definir si una sentencia o conjunto de sentencias se repite o no, y bajo qué condiciones.

Tomas de decisión

- 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.
 - 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.

Tomas de decisión 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.

```
<?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";
?>
```

Tomas de decisión

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";
    }
?>
```

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
?>
```

Bucles

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
?>
```

Bucles

for

Son los bucles más complejos de PHP. Al igual que los del lenguaje C, se componen de tres expresiones

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

Bucles

Break/Continue

- Se utilizan para controlar la ejecución dentro de las sentencias en las que se encuentra el programa.
- Para salir de una estructura de control condicional o iterativa puede usarse BREAK o CONTINUE.
- Ambas se emplean de una forma análoga, con la diferencia de que mientras BREAK finaliza totalmente la ejecución del bucle; CONTINUE hace que se salte a la siguiente iteración.

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.

En PHP, hay tres tipos de arrays:

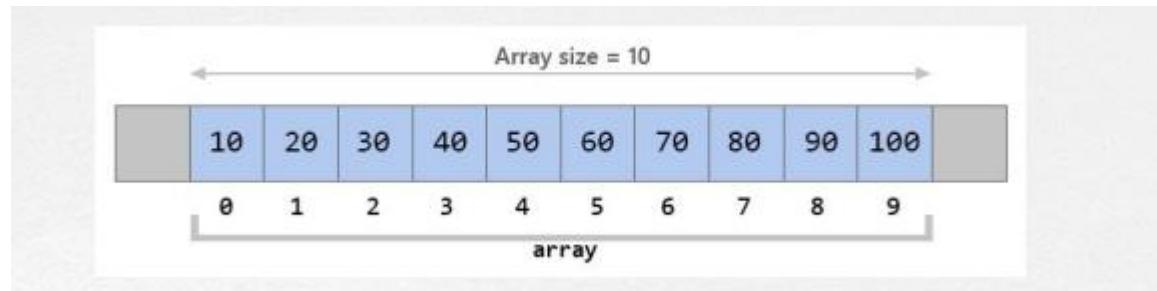
- Arrays numéricos: arrays con un índice numérico
- Arrays asociativos: arrays con claves con nombre
- Arrays multidimensionales: los arrays de varias dimensiones son arrays que contienen uno o más arrays en sus elementos. La dimensión de un array indica la cantidad de índices que necesita para seleccionar un elemento.

Arrays

Arrays numéricos

- Creación
 - Constructor array()
 - Sintaxis corta con corchete []

```
<?php  
$coches = array("Volvo", "BMW", "Toyota");  
echo "Las marcas de coches son:" . $coches[0] . ", "  
     . $coches[1] . " y " . $coches[2] . ". ";  
?>
```



Arrays

Arrays numéricos

Recorrer

```
for ($i=0; $i < count($dias); $i++) {  
    echo "<p>".$dias[$i]."</p>";  
}  
  
foreach($dias as $d)  
    echo "<p>".$d."</p>";
```

Arrays

Arrays asociativos o indexados

- Arrays cuyos keys son Strings personalizados
- Los Strings son CaseSensitive

```
<?php  
  
    $edad = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");  
  
    echo "Peter tiene " . $edad['Peter'] . " años.";  
  
?>
```

Arrays

Arrays asociativos o indexados

Recorrer

```
foreach ($edad as $d)  
    echo "<p>".$d."</p>";  
  
foreach ($edad as $variable => $valor) {  
    print "<p>".$variable." ->";  
    print $valor."</p>";  
}
```

Arrays

Arrays multidimensional

```
<?php  
  
$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 móviles")  
);  
?>
```

Arrays

Arrays multidimensional

Recorrer

```
foreach ($ciclos as $ciclo => $array) {  
    print "<p>".$ciclo."</p>";  
    foreach ($array as $inicial => $nombre) {  
        print "<p>".$inicial."> ->";  
        print $nombre."</p>";  
    }  
}
```

Arrays

Arrays multidimensional

Recorrer

```
foreach ($ciclos as $ciclo => $array) {  
    print "<p>".$ciclo."</p>";  
    foreach ($array as $inicial => $nombre) {  
        print "<p>".$inicial."> ->";  
        print $nombre."</p>";  
    }  
}
```

Arrays

Recorrer

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.

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

Arrays

Recorrer

- 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.

Arrays

Funciones

- 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
- 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 quieres buscar un elemento concreto dentro de un array, puedes utilizar la función in_array. Devuelve true si encontró el elemento o false en caso contrario.
- 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 quieres buscar es un clave en un array, tienes la función array_key_exists, que devuelve true o false.

<https://www.php.net/manual/es/ref.array.php>

Funciones

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.

Funciones

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`

Funciones

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

```
function precio_con_iva() {  
    global $precio;  
    $precio_iva = $precio * 1.18;  
    echo "El precio con IVA es ".$precio_iva;  
}  
  
$precio = 10;  
precio_con_iva();
```

Funciones

Argumentos

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.

```
function precio_con_iva($precio) {  
    return $precio * 1.18;  
}  
  
$precio = 10;  
$precio_iva = precio_con_iva($precio);  
print "El precio con IVA es ".$precio_iva
```

Funciones

Argumentos

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.

```
function precio_con_iva($precio, $iva=0.18) {  
    return $precio * (1 + $iva);  
}
```

Funciones

Argumentos

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.

```
function precio_con_iva(&$precio, $iva=0.18) {  
    $precio *= (1 + $iva);  
}  
$precio = 10;  
print "El precio con IVA es ".$precio
```

Formularios

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 <FORM> </FORM>. 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

Formularios

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.

```
<form name="input" action="procesa.php" method="post">
```

Formularios

- 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.

Formularios

Cadenas

<https://www.php.net/manual/es/ref.strings.php>

- [crypt](#) – Hash de cadenas de un sólo sentido
- [chr](#) – Devuelve un carácter específico
- [explode](#) – Divide un string en varios string
- [implode](#) – Une elementos de un array en un string
- [md5](#) – Calcula el 'hash' md5 de un string
- [money_format](#) – Da formato a un número como un string de moneda
- [number_format](#) – Formatear un número con los millares agrupados
- [sha1](#) – Calcula el 'hash' sha1 de un string
- [str_contains](#): determina si una cadena contiene una subcadena determinada
- [str_replace](#) – Reemplaza todas las apariciones del string buscado con el string de reemplazo
- [str_split](#) – Convierte un string en un array
- [strlen](#) – Obtiene la longitud de un string
- [trim](#) – Elimina espacio en blanco (u otro tipo de caracteres) del inicio y el final de la cadena

Formularios

Expresiones regulares

Una expresión regular nos permite compararla con textos mayores con el fin de verificar que estos contienen el patrón definido en dicha expresión, su uso es muy amplio sobre todo en operaciones e validación de datos, búsqueda de texto...

```
$patron = '/maría/';  
preg_match($patron, 'maría es mi profe favorita')
```

<https://www.php.net/manual/es/ref.pcre.php>

Formularios

Expresiones regulares

Modificadores

- g (global): Se aplicará a toda la cadena
- i (insensible): Ignora las mayúsculas y minúsculas
- m (multilinea): seguirá buscando correspondencias a pesar de haber alcanzado el final de la línea

Formularios

Expresiones regulares

Metacaracteres

- ? . : El punto se interpreta como cualquier carácter, es decir, busca cualquier carácter sin incluir los saltos de línea.
- ? \: indica que el siguiente carácter es un carácter especial y no debe ser interpretado como literal
- ? []: La función de los corchetes es representar "clases de caracteres", es decir, agrupar caracteres en grupos o clases. Dentro de los corchetes es posible utilizar el guion "-" para especificar rangos de caracteres.

Formularios

Expresiones regulares

Metacaracteres

- | : Sirve para indicar una de varias opciones.
- \$: Representa el final de la cadena de caracteres o el final de la línea.
- ^: Éste carácter tiene una doble funcionalidad, en función de si utiliza individualmente y si se utiliza en conjunto con otros caracteres especiales.
 - Su funcionalidad como carácter individual: el carácter ^ representa el inicio de la cadena
 - Cuando se utiliza en conjunto con los corchetes, permite encontrar cualquier carácter que NO se encuentre dentro del grupo indicado

Formularios

Expresiones regulares

Metacaracteres

- (): De forma similar que los corchetes, los paréntesis sirven para agrupar caracteres, sin embargo, existen varias diferencias fundamentales entre los grupos establecidos por medio de corchetes y los grupos establecidos por paréntesis:
 - Los caracteres especiales conservan su significado dentro de los paréntesis.
 - Los grupos establecidos con paréntesis establecen una "etiqueta" o "punto de referencia" para el motor de búsqueda, que puede ser utilizada posteriormente.
 - Utilizados en conjunto con la barra | permiten hacer búsquedas opcionales.

Formularios

Expresiones regulares

Metacaracteres

- ?: El signo de interrogación especifica que una parte de la búsqueda es opcional.
- * : El asterisco sirve para encontrar algo que se encuentra repetido 0 o más veces.
- {}: Comúnmente las llaves son caracteres literales cuando se utilizan por separado en una expresión regular. Para que adquieran su función de metacaracteres es necesario que encierren uno o varios números separados por coma y que estén colocados a la derecha de otra expresión regular
 - {n} indica el numero de repeticiones del caracter que lo precede
 - {n,m} Las minimas y maximas

Formularios

Expresiones regulares

Metacaracteres

- +: Se utiliza para encontrar una cadena que se encuentre repetida una o más veces.
- grupos anónimos: Los grupos anónimos se establecen cada vez que se encierra una expresión regular en paréntesis, y almacenará la referencia

Ficheros

Funciones con archivos

- Abrir
- Leer
- Escribir
- Cerrar

Ficheros

Abrir un archivo

Los archivos en PHP se abren con la función fopen(), que requiere dos parámetros: el archivo que se quiere abrir y el modo en el que abrir el archivo. La función devuelve un puntero en el archivo si es satisfactoria o cero si no lo es. Los archivos se abren para realizar operaciones de lectura o escritura.

Si no es posible abrir el archivo, devuelve cero, por eso es frecuente utilizar esta función en una condición:

```
if (! $fp = fopen ("miarchivo.txt", "r")) {  
    echo "No se ha podido abrir el archivo";  
}
```

Ficheros

Modos de acceso existentes para fopen son:

Modo	Descripción
r	Apertura para lectura. Puntero al principio del archivo
r+	Apertura para lectura y escritura. Puntero al principio del archivo
w	Apertura para escritura. Puntero al principio del archivo y lo sobreescribe. Si no existe se intenta crear.
w+	Apertura para lectura y escritura. Puntero al principio del archivo y lo sobreescribe. Si no existe se intenta crear.
a	Apertura para escritura. Puntero al final del archivo. Si no existe se intenta crear.
a+	Apertura para lectura y escritura. Puntero al final del archivo. Si no existe se intenta crear.
x	Creación y apertura para sólo escritura. Puntero al principio del archivo. Si el archivo ya existe dará error E_WARNING. Si no existe se intenta crear.
x+	Creación y apertura para lectura y escritura. Mismo comportamiento que x.
c	Apertura para escritura. Si no existe se crea. Si existe no se sobreescribe ni da ningún error. Puntero al principio del archivo.
c+	Apertura para lectura y escritura. Mismo comportamiento que C.

Ficheros

Escritura:

`fwrite()` escribe el contenido de `string` al flujo de archivo apuntado por `handle`.

```
fwrite(resource $handle, string $string, int $length) : int
```

Leer:

`fread()` lee hasta `length` bytes desde el puntero al fichero referenciado por `handle`. La lectura termina tan pronto como se encuentre una de las siguientes condiciones:

- `length` bytes han sido leídos
- EOF (fin de fichero) es alcanzado

```
fread(resource $handle, int $length) : string
```

Ficheros XML

Instalar el modulo

```
apt-get install php7.2-xml
```

Este paquete proporciona los módulos DOM, SimpleXML, WDDX, XML y XSL para PHP.

Reiniciar el servicio de apache

Ficheros XML

Leer un xml

- Leerlo con [simplexml_load_file\(\)](#).
- simplexml_load_file – Interpreta un fichero XML en un objeto

```
$xml = simplexml_load_file('test.xml');
```

SimpleXMLElement

<https://www.php.net/manual/es/class.simplexmlelement.php>

Ficheros XML

Escribir un xml

Clase DOMDocument <https://www.php.net/manual/es/class.domdocument.php>

Representa un documento HTML o XML en su totalidad; sirve como raíz del árbol de documento.

```
$XML = new DOMDocument("1.0", "utf-8"); //Creacion del objeto  
DOMDocument  
  
$XML->formatOutput = true; //Para que salga bien formateado  
//La primera etiqueta será el nombre de raiz  
$raiz = $XML->appendChild($XML->createElement("RaizXML"));  
$XML->save("./departamento.xml")
```

Ficheros XML

Modificar un xml

Leer el documento, transformarlo en dom y guardarlo

```
$xml = simplexml_load_file("./ficheroXML.xml");
$xmlDom = dom_import_simplexml($xml)->ownerDocument;
$xmlDom->save("./ficheroXML.xml");
```

Tema 4. Acceso a datos

DESARROLLO EN ENTORNO SERVIDOR

2º DAW

MARÍA CRIADO DOMÍNGUEZ

Índice

1. Instalar y configurar mysql
2. Utilización de bases de datos MySQL en PHP
3. Mysqli
4. DBO

Instalar y configurar mysql

MySQL es un sistema gestor de bases de datos (SGBD) relacionales. Es un programa de código abierto que se ofrece bajo licencia GNU GPL

MySQL se emplea en múltiples aplicaciones web, ligado en la mayor parte de los casos al lenguaje PHP y al servidor web Apache. Utiliza SQL para la gestión, consulta y modificación de la información almacenada. Soporta la mayor parte de las características de ANSI SQL

En Linux, la instalación de MySQL se divide básicamente en dos paquetes que puedes instalar:

- mysql-server. Es el servidor en sí. Necesitas instalar este paquete para gestionar las bases de datos y permitir conexiones desde el equipo local o a través de la red.
- mysql-client. Es el cliente desde donde se hace la conexión con el servidor

Instalar y configurar mysql

Una vez instalado, puedes gestionar la ejecución del servicio de la misma forma que cualquier otro servicio del sistema:

```
systemctl status mysql
```

En una instalación típica, el usuario root no tiene por defecto contraseña de acceso al servidor. Es importante asignarle una por razones de seguridad:

```
mysqladmin -u root password nueva-contraseña
```

El servidor se ejecuta por defecto en el Puerto TCP 3306.

Utilización de bases de datos MySQL en PHP

Las acciones sobre las bases de datos como:

- Establecer conexiones.
- Ejecutar sentencias SQL.
- Obtener los registros afectados o devueltos por una sentencia SQL.
- Emplear transacciones.
- Ejecutar procedimientos almacenados.
- Gestionar los errores que se produzcan durante la conexión o en el establecimiento de la misma.

MySQLi

Esta extensión que viene incluida con PHP a partir de la versión 5. Ofrece un interface de programación dual, pudiendo accederse a las funcionalidades de la extensión utilizando objetos o funciones de forma indiferente.

```
// utilizando constructores y métodos de la programación orientada a objetos  
$conexion = new mysqli('localhost', 'usuario', 'contraseña', 'base_de_datos');  
print conexion->server_info;  
  
// utilizando llamadas a funciones  
$conexion = mysqli_connect('localhost', 'usuario', 'contraseña', 'base_de_datos');  
print mysqli_get_server_info($conexion);
```

MySQLi

Las opciones de configuración de PHP se almacenan en el fichero php.ini. En este fichero hay una sección específica para las opciones de configuración propias de cada extensión. Entre las opciones que puedes configurar para la extensión MySQLi están:

- `mysqli.allow_persistent`. Permite crear conexiones persistentes.
- `mysqli.default_port`. Número de puerto TCP predeterminado a utilizar cuando se conecta al servidor de base de datos.
- `mysqli.reconnect`. Indica si se debe volver a conectar automáticamente en caso de que se pierda la conexión.
- `mysqli.default_host`. Host predeterminado a usar cuando se conecta al servidor de base de datos.
- `mysqli.default_user`. Nombre de usuario predeterminado a usar cuando se conecta al servidor de base de datos.
- `mysqli.default_pw`. Contraseña predeterminada a usar cuando se conecta al servidor de base de datos.

MySQLi

Para poder comunicarte desde un programa PHP con un servidor MySQL, el primer paso es establecer una conexión. Toda comunicación posterior que tenga lugar, se hará utilizando esa conexión.

Si utilizas la extensión MySQLi, establecer una conexión con el servidor significa crear una instancia de la clase `mysqli`. El constructor de la clase puede recibir seis parámetros, todos opcionales, aunque lo más habitual es utilizar los cuatro primeros:

- El nombre o dirección IP del servidor MySQL al que te quieras conectar.
- Un nombre de usuario con permisos para establecer la conexión.
- La contraseña del usuario.
- El nombre de la base de datos a la que conectarse.
- El número del puerto en que se ejecuta el servidor MySQL.
- El socket o la tubería con nombre (named pipe) a usar.

MySQLi

Conexión

```
$dwes = new mysqli('localhost', 'dwes', 'abc123.', 'dwes');  
$dwes = mysqli_connect('localhost', 'dwes', 'abc123.', 'dwes');
```

Es importante verificar que la conexión se ha establecido correctamente. Para comprobar el error, en caso de que se produzca, puedes usar las siguientes propiedades (o funciones equivalentes) de la clase mysqli:

- connect_errno (o la función mysqli_connect_errno) devuelve el número de error o null si no se produce ningún error.
- connect_error (o la función mysqli_connect_error) devuelve el mensaje de error o null si no se produce ningún error.

MySQLi

Conexión

Si una vez establecida la conexión, quieres cambiar la base de datos puedes usar el método `select_db` (o la función `mysqli_select_db` de forma equivalente) para indicar el nombre de la nueva.

```
$dwes->select_db('otra_bd');
```

Una vez finalizadas las tareas con la base de datos, utiliza el método `close` (o la función `mysqli_close`) para cerrar la conexión con la base de datos y liberar los recursos que utiliza.

```
$dwes->close();
```

MySQLi

Ejecutar consultas

En el caso de ejecutar una sentencia SQL que sí devuelva datos (como un SELECT), éstos se devuelven en forma de un objeto resultado (de la clase mysqli_result).

```
$resultadoConsulta = $miDB->query('consultaSQL');

$resultadoConsulta = $miDB->query('consultaSQL', MYSQLI_USE_RESULT);
• MYSQLI_USE_RESULT //Recupera la información a medida que la usamos
• MYSQLI_STORE_RESULT //Recupera toda la información al ejecutar la consulta
(defecto)

$miDB->affected_rows;

$resultadoConsulta->free(); //Libera el espacio ocupado tras la
consulta
```

MySQLi

Obtención de resultados

```
$registroArray = $resultadoConsulta->fetch_array();
```

Obtiene un registro en el array (asociativo y numérico por defecto) y avanza el puntero por el conjunto de registros

```
$registroObjeto = $resultadoConsulta->fetch_object();
```

Obtiene un registro en el objeto (asociativo) y avanza el puntero por el conjunto de registros

Ambos métodos devuelven null cuando terminan de recorrer el conjunto de resultados.

MySQLi

Consultas preparadas (no devuelve resultados)

```
$consulta = $miDB->stmt_init(); //Devuelve un objeto de la clase mysqli_stmt
```

```
$consulta->prepare('ConsultaPreparada VALUES (?, ?, ...);
```

Preparamos la consulta pendiente de dar valor a las ?

```
$consulta->bind_param('cadenaformato[i,d,s,b]', $var1, $var2,...);
```

Sustituye las interrogaciones por el valor de las variables que pasamos en bind_param
i – entero
d – real doble precisión
s – cadena de texto
b – BLOB – binario

```
$consulta->execute(); //Ejecuta la consulta
```

```
$consulta->close();
```

```
$miDB->close();
```

MySQLi

Consultas preparadas (con resultados)

```
$consulta = $miDB->stmt_init(); //Devuelve un objeto de la clase mysqli_stmt  
$consulta->prepare('SELECT campo1, campo2,... FROM...');  
$consulta->execute(); //Ejecuta la consulta  
$consulta->bind_result($var1, $var2,...);  
while ($consulta->fetch())  
    {Tratamiento de $var1, $var2,...}  
$consulta->close();  
$miDB->close();
```

MySQLi

Transacciones

```
$miDB->autocommit(false);
//Deshabilita el modo transaccional automático

$resultadoConsulta1 = $miDB->query('delete from usuarios where 1');

//si todo ha ido bien

$miDB->commit(); //Confirma los cambios y los consolida

// sino

$miDB->rollback(); //Revierte o deshace los cambios
```

PDO

Utilizar otro servidor como almacenamiento que no sea MySql, deberás adoptar una capa de abstracción para el acceso a los datos.

Existen varias alternativas como ODBC, pero sin duda la opción más recomendable en la actualidad es PDO.

El objetivo es que si llegado el momento necesitas cambiar el servidor de base de datos, las modificaciones que debas realizar en tu código sean mínimas. Incluso es posible desarrollar aplicaciones preparadas para utilizar un almacenamiento u otro según se indique en el momento de la ejecución.

PDO

La extensión PDO debe utilizar un driver o controlador específico para el tipo de base de datos que se utilice. Para consultar los controladores disponibles en tu instalación de PHP, puedes utilizar la información que proporciona la función `phpinfo`.

PDO se basa en las características de orientación a objetos de PHP pero, al contrario que la extensión MySQLi, no ofrece un interface de programación dual. Para acceder a las funcionalidades de la extensión tienes que emplear los objetos que ofrece, con sus métodos y propiedades. No existen funciones alternativas.

[PHP: Controladores de PDO – Manual](#)

PDO

Establecimiento de conexión

```
$miDB = new PDO ('DSN', 'nobreusuario', 'password' );
```

Instanciamos un objeto PDO y establecemos la conexión

Construcción de la cadena PDO: (ej. 'mysql:host=localhost; dbname=midb')

host – nombre o dirección IP del servidor

port – número de puerto en el que escucha el servidor

dbname – nombre de la base de datos

```
$miDB->getAttribute (...) ; //Obtiene el valor de un atributo
```

```
$miDB->setAttribute (...) ; //Establece el valor de un atributo
```

```
unset ($miDB) ;
```

PDO

Consultas

En el caso de las consultas de acción, como INSERT, DELETE o UPDATE, el método exec devuelve el número de registros afectados.

```
$numRegistros = $miDB->exec('consultaSQLDeActualizacion');
```

Si la consulta genera un conjunto de datos, como es el caso de SELECT, debes utilizar el método query, que devuelve un objeto de la clase PDOStatement.

```
$resultadoConsulta = $miDB->query('consultaSQLDeSeleccion');
```

PDO

Obtención y utilización de conjuntos de resultados

Por defecto, el método `fetch` genera y devuelve, a partir de cada registro, un array con claves numéricas y asociativas. Para cambiar su comportamiento, admite un parámetro opcional que puede tomar uno de los siguientes valores:

- `PDO::FETCH_ASSOC`. Devuelve solo un array asociativo.
- `PDO::FETCH_NUM`. Devuelve solo un array con claves numéricas.
- `PDO::FETCH_BOTH`. Devuelve un array con claves numéricas y asociativas. Es el comportamiento por defecto.
- `PDO::FETCH_OBJ`. Devuelve un objeto cuyas propiedades se corresponden con los campos del registro
- `PDO::FETCH_LAZY`. Devuelve tanto el objeto como el array con clave dual anterior.
- `PDO::FETCH_BOUND`. Devuelve `true` y asigna los valores del registro a variables, según se indique con el método `bindColumn`.

PDO

Obtención y utilización de conjuntos de resultados

```
$resultadoConsulta = $miDB->query('consultaSQL');
```

```
$registroArray = $resultadoConsulta->fetch();
```

Obtiene un registro en el array (asociativo y numérico por defecto) y avanza el puntero por el conjunto de registros

```
$registroObjeto = $resultadoConsulta->fetch(PDO::FETCH_OBJ);
```

Obtiene un registro en el objeto (asociativo) y avanza el puntero por el conjunto de registros
Ambos métodos devuelven null cuando terminan de recorrer el conjunto de resultados.

PDO

Consultas preparadas. Forma 1

```
$consulta      =      $miDB->prepare ('ConsultaPreparada      VALUES  
( ?, ?, ... ) ;
```

Devuelve un objeto de la clase PDOStatement

Preparamos la consulta pendiente de dar valor a las ?

```
$consulta->bindParam(1, $var1);
```

```
$consulta->bindParam(2, $var2);
```

Sustituye las interrogaciones por el valor de las variables que pasamos en bind_param

```
$consulta->execute(); Ejecuta la consulta
```

PDO

Consultas preparadas

```
$consulta = $miDB->prepare('ConsultaPreparada    VALUES  
(:param1, :param2,...) );  
  
$parametros = array(  
    ":param1"=>"nombrecampo1",  
    ":param2"=>"nombrecampo2");  
  
$consulta->execute($parametros);
```

PDO

Consultas preparadas

```
$consulta = $miDB->prepare('ConsultaPreparada    VALUES    (:param1,  
:param2,...)');
```

Devuelve un objeto de la clase PDOStatement

Preparamos la consulta poniendo nombre a los campos precedidos de :

```
$consulta->bindParam(:param1, $var1);
```

```
$consulta->bindParam(:param2, $var2);
```

Sustituye las :param por el valor de las variables que pasamos en bind_param

```
$consulta->execute(); Ejecuta la consulta
```

PDO

Transacciones

Por defecto PDO trabaja en modo "autocommit", esto es, confirma de forma automática cada sentencia que ejecuta el servidor. Para trabajar con transacciones, PDO incorpora tres métodos:

- `beginTransaction`. Deshabilita el modo "autocommit" y comienza una nueva transacción, que finalizará cuando ejecutes uno de los dos métodos siguientes.
- `commit`. Confirma la transacción actual.
- `rollback`. Revierte los cambios llevados a cabo en la transacción actual.

PDO

Transacciones

```
$miDB->beginTransaction(); //Deshabilita el modo autocommit  
  
$resultadoConsulta1 = $miDB->exec('consultaSQL1');  
  
$resultadoConsulta2 = $miDB->exec('consultaSQL2');  
  
$miDB->commit(); //Confirma los cambios y los consolida  
  
$miDB->rollback(); Revierte o deshace los cambios
```

Tema 5. Desarrollo de aplicaciones Web utilizando código embebido

DESARROLLO EN ENTORNO SERVIDOR

2º DAW

MARÍA CRIADO DOMÍNGUEZ

Índice

- Autenticación de usuarios. Mecanismos.
- Sesiones.
- Cookies. Aplicaciones prácticas. Las cookies y el navegador.
- Seguridad: usuarios, perfiles, roles.

Autenticación de usuarios

Para identificar a los usuarios que visitan un sitio web, se pueden utilizar distintos métodos como el DNI digital o certificados digitales de usuario (documento digital que contiene información acerca del usuario como el nombre o la dirección. Esa información está firmada por otra entidad, llamada entidad certificadora, que debe ser de confianza y garantiza que la información que contiene es cierta), pero el más extendido es solicitar al usuario cierta información que solo él conoce: la combinación de un nombre de usuario y una contraseña.

Autentificación de usuarios

La información de autentificación (nombre y contraseña de los usuarios) se envía en texto plano desde el navegador hasta el servidor web. Esta práctica es altamente insegura y nunca debe usarse sin un protocolo como HTTPS que permita cifrar las comunicaciones con el servidor web.

Autentificación de usuarios

Mecanismos de autentificación

El protocolo HTTP ofrece un método sencillo para autenticar a los usuarios. El proceso es el siguiente:

- El servidor web debe proveer algún método para definir los usuarios que se utilizarán y cómo se pueden autenticar. Además, se tendrán que definir los recursos a los que se restringe el acceso y qué lista de control de acceso (ACL - lista de permisos sobre un objeto (fichero, directorio, etc.), que indica qué usuarios pueden utilizar el objeto y las acciones concretas que pueden realizar con el mismo (lectura, escritura, borrado, etc.)) se aplica a cada uno.
- Cuando un usuario no autenticado intenta acceder a un recurso restringido, el servidor web responde con un error de "Acceso no autorizado" (código 401).

Autentificación de usuarios

Mecanismos de autentificación

- El navegador recibe el error y abre una ventana para solicitar al usuario que se autentifique mediante su nombre y contraseña.
- La información de autentificación del usuario se envía al servidor, que la verifica y decide si permite o no el acceso al recurso solicitado. Esta información se mantiene en el navegador para utilizarse en posteriores peticiones a ese servidor.

Autentificación de usuarios

Mecanismos de autentificación

En el servidor web Apache, existe una utilidad en línea de comandos, htpasswd, que permite almacenar en un fichero una lista de usuarios y sus respectivas contraseñas. La información relativa a las contraseñas se almacena cifrada; así, es conveniente crear este fichero en un lugar no accesible por los usuarios del servidor web.

Ir a ruta /etc/apache2/

sudo htpasswd -c users dwes

-c solo con el primer usuario

el fichero se creara en la ruta /etc/apache2/users, que en principio no es accesible vía web.

Autentificación de usuarios

Mecanismos de autentificación

Para indicarle al servidor Apache qué recursos tienen acceso restringido, una opción es crear un fichero .htaccess en el directorio en que se encuentren, con las siguientes directivas:

```
AuthName "Contenido restringido"  
AuthType Basic AuthUserFile /etc/apache2/users  
require valid-user
```

Cambiar la configuración de apache2.conf

```
AllowOverride All
```

Autentificación de usuarios

Mecanismos de autentificación II

```
header('WWW-Authenticate: Basic Realm="Contenido restringido"');
```

Desde PHP puedes acceder a la información de autentificación HTTP que ha introducido el usuario utilizando el array superglobal `$_SERVER`. Valor Contenido

- `$_SERVER['PHP_AUTH_USER']` Nombre de usuario que se ha introducido.
- `$_SERVER['PHP_AUTH_PW']` Contraseña introducida.
- `$_SERVER['AUTH_TYPE']` Método HTTP usado para autenticar. Puede ser Basic o Digest.

Si no introduces un usuario/contraseña válidos, el navegador te mostrará el error 401.

Además, en PHP puedes usar la función `header` para forzar a que el servidor envíe un error de "Acceso no autorizado" (código 401). De esta forma no es necesario utilizar ficheros `.htaccess` para indicarle a Apache qué recursos están restringidos.

Sesiones

- Una sesión es un mecanismo de programación de las tecnologías de web scripting que permite conservar información sobre un usuario al pasar de una página a otra.
- A diferencia de una cookie, los datos asociados a una sesión se almacenan en el servidor y nunca en el cliente.
- En el servidor web están almacenados todos los datos de la sesión y se accede a ellos cada vez que se pasa de página gracias al identificador almacenado en la cookie.

Sesiones

Directiva	Significado
<code>session.use_cookies</code>	Indica si se deben usar cookies (1) o propagación en la URL (0) para almacenar el SID.
<code>session.use_only_cookies</code>	Se debe activar (1) cuando utilizas cookies para almacenar los SID, y además no quieras que se reconozcan los SID que se puedan pasar como parte de la URL (este método se puede usar para usurpar el identificador de otro usuario).
<code>session.save_handler</code>	Se utiliza para indicar a PHP cómo debe almacenar los datos de la sesión del usuario. Existen cuatro opciones: en ficheros (<code>files</code>), en memoria (<code>mm</code>), en una base de datos SQLite (<code>sqlite</code>) o utilizando para ello funciones que debe definir el programador (<code>user</code>). El valor por defecto (<code>files</code>) funcionará sin problemas en la mayoría de los casos.
<code>session.name</code>	Determina el nombre de la cookie que se utilizará para guardar el SID. Su valor por defecto es <code>PHPSESSID</code> .
<code>session.auto_start</code>	Su valor por defecto es 0, y en este caso deberás usar la función <code>session start</code> para gestionar el inicio de las sesiones. Si usas sesiones en el sitio web, puede ser buena idea cambiar su valor a 1 para que PHP active de forma automática el manejo de sesiones.
<code>session.cookie_lifetime</code>	Si utilizas la URL para propagar el SID, éste se perderá cuando cierres tu navegador. Sin embargo, si utilizas cookies, el SID se mantendrá mientras no se destruya la cookie. En su valor por defecto (0), las cookies se destruyen cuando se cierra el navegador. Si quieras que se mantenga el SID durante más tiempo, debes indicar en esta directiva ese tiempo en segundos.
<code>session.gc_maxlifetime</code>	Indica el tiempo en segundos que se debe mantener activa la sesión, aunque no haya ninguna actividad por parte del usuario. Su valor por defecto es 1440. Es decir, pasados 24 minutos desde la última actividad por parte del usuario, se cierra su sesión automáticamente.

Sesiones

Inicio de sesión

1. Ejecutando la función `session_start()` para indicar a PHP que inicie una nueva sesión o reanude la anterior. Todas las páginas que necesiten utilizar la información almacenada en la sesión, deberán ejecutar la función `session_start()`.
2. Si has activado la directiva `session.auto_start` en la configuración de PHP, la sesión comenzará automáticamente en cuanto un usuario se conecte a tu sitio web.
 - En caso de que ese usuario ya haya abierto una sesión con anterioridad, y ésta no se haya eliminado, en lugar de abrir una nueva sesión se reanudará la anterior.
 - Si has activado la directiva `session.auto_start` en la configuración de PHP, la sesión comenzará automáticamente en cuanto un usuario se conecte a tu sitio web.
 - En caso de que ese usuario ya haya abierto una sesión con anterioridad, y ésta no se haya eliminado, en lugar de abrir una nueva sesión se reanudará la anterior.

Sesiones

session_id(): asigna un identificador a una sesión SID (se recomienda que sea corto y que contenga sólo caracteres alfanuméricos). Si no asignamos un identificador, se asignará uno automáticamente. En caso de ser usada, debe estar antes de session_start().

session_name(): asigna/retorna el nombre a una sesión (se recomienda que sea corto y que contenga sólo caracteres alfanuméricos). Si no indicamos un nombre, se usará por defecto el definido en el archivo de configuración de PHP (por defecto es PHPSESSID).

IMPORTANTE: en caso de usarse deberá estar definida en cada archivo .php, y situada antes de session_start().

SID Identificador de sesión

- Cada usuario distinto de un sitio web tiene su propia información de sesión. Para distinguir una sesión de otra se usan los identificadores de sesión (SID).
- Un SID es un atributo que se asigna a cada uno de los visitantes de un sitio web y lo identifica. Es único para cada usuario.
- De esta forma, si el servidor web conoce el SID de un usuario, puede relacionarlo con toda la información que posee sobre él, que se mantiene en la sesión del usuario.

Variable superglobal `$_SESSION`

Mientras la sesión permanece abierta, puedes utilizar la variable superglobal `$_SESSION` para añadir información a la sesión del usuario, o para acceder a la información almacenada en la sesión.

Cerrar una sesión

Cerrar una sesión supone:

- Destruir la matriz `$_SESSION`
- Borrar el identificador de la sesión (SID).

Puede cerrarse:

- El usuario: puede cerrar la sesión simplemente cerrando el navegador.
- Un script: puede cerrar la sesión mediante la función `session_destroy()`.
- El servidor: puede cerrar la sesión cuando ha pasado el tiempo indicado en segundos en la directiva `session.gc_maxlifetime`.

Cookies

Una cookie es un fichero de texto que un sitio web guarda en el entorno del usuario del navegador. Su uso más típico es el almacenamiento de las preferencias del usuario (por ejemplo, el idioma en que se deben mostrar las páginas), para que no tenga que volver a indicarlas la próxima vez que visite el sitio.

Según el protocolo HTTP, las cookies no pueden ser más grandes de 4096 Bytes (4KB).

También hay un límite en el número total de cookies en el disco duro del cliente. Suele ser de unas 300 cookies. Cuando se llega a este número, una cookie antigua se elimina antes de crear la nueva.

Cookies

Los servidores web sólo pueden acceder a cookies establecidas por su propio dominio.

Este dominio es establecido por el navegador cuando el servidor crea una nueva cookie , y sólo puede ser un dominio o subdominio del servidor.

Según su duración:

- **Cookies de sesión**

Las cookies de sesión son cookies cortas que caducan al cerrar una página web, por ejemplo, al cerrar la pestaña o el navegador/app.

- **Cookies persistentes**

Opuestas a las cookies de sesión, las cookies persistentes son aquellas que siguen almacenadas en el ordenador durante más tiempo incluso después de la sesión (pueden ser horas, meses o años).

Cookies

Según su finalidad:

- Cookies de técnicas
- Cookies personalización
- Las cookies de análisis
- Las cookies publicitarias
- Las cookies de redes sociales y otros plugings

Cookies

En PHP, para almacenar una cookie en el navegador del usuario, puedes utilizar la función setcookie. El único parámetro obligatorio que tienes que usar es el nombre de la cookie, pero admite varios parámetros más opcionales. <http://es.php.net/manual/es/function.setcookie.php>

Por ejemplo, si quieras almacenar en una cookie el nombre de usuario que se transmitió en las credenciales HTTP puedes hacer:

```
setcookie("nombre", "maría", time()+3600);
```

Los dos primeros parámetros son el nombre de la cookie y su valor. El tercero es la fecha de caducidad de la misma (una hora desde el momento en que se ejecute). En caso de no figurar este parámetro, la cookie se eliminará cuando se cierre el navegador. Ten en cuenta que también se pueden aplicar restricciones a las páginas del sitio que pueden acceder a una cookie en función de la ruta.

Cookies

Por ejemplo, si quieras almacenar en una cookie el nombre de usuario que se transmitió en las credenciales HTTP (es solo un ejemplo, no es en absoluto aconsejable almacenar información relativa a la seguridad en las cookies), puedes hacer:

```
setcookie("nombre", "maría", time()+3600);
```

Los dos primeros parámetros son el nombre de la cookie y su valor. El tercero es la fecha de caducidad de la misma (una hora desde el momento en que se ejecute). En caso de no figurar este parámetro, la cookie se eliminará cuando se cierre el navegador. Ten en cuenta que también se pueden aplicar restricciones a las páginas del sitio que pueden acceder a una cookie en función de la ruta.

Cookies

Las cookies se transmiten entre el navegador y el servidor web de la misma forma que las credenciales que acabas de ver; utilizando los encabezados del protocolo HTTP. Por ello, las sentencias setcookie deben enviarse antes de que el navegador muestre información alguna en pantalla.

El proceso de recuperación de la información que almacena una cookie es muy simple. Cuando accedes a un sitio web, el navegador le envía de forma automática todo el contenido de las cookies que almacene relativas a ese sitio en concreto.

Desde PHP puedes acceder a esta información por medio del array `$_COOKIE`.

Cookies

Es importante tener en cuenta al trabajar con cookies es el orden en que se realiza el envío y la creación de cookies, así como su disponibilidad en `$_COOKIES`:

- Cuando una página pide al navegador que cree una cookie, el valor de la cookie no está disponible en `$_COOKIE` en esa página en ese momento.
- El valor de la cookie estará disponible en `$_COOKIE` en las páginas en llamadas posteriores, cuando el navegador las pida al servidor y envíe el valor de la cookie en la petición.

Cookies

Para borrar una cookie, simplemente se debe volver a crear la cookie con un tiempo de expiración anterior al presente.

```
setcookie("nombre", "Pepito Conejo", time() - 1);
```

Si solamente queremos borrar el valor almacenado en la cookie sin borrar la propia cookie, simplemente se debe volver a crear la cookie, sin indicarle el valor a almacenar:

```
// Esta cookie no se borra, pero no guardará ningún valor.  
setcookie("nombre");
```

Tema 6. Generación dinámica de páginas

DESARROLLO EN ENTORNO SERVIDOR

2º DAW

MARÍA CRIADO DOMÍNGUEZ

Objetos

Características

- **Herencia.** Es el proceso de crear una clase a partir de otra, heredando su comportamiento y características y pudiendo redefinirlos.
- **Abstracción.** Hace referencia a que cada clase oculta en su interior las peculiaridades de su implementación, y presenta al exterior una serie de métodos (interface) cuyo comportamiento está bien definido. Visto desde el exterior, cada objeto es un ente abstracto que realiza un trabajo.
- **Polimorfismo.** Un mismo método puede tener comportamientos distintos en función del objeto con que se utilice.
- **Encapsulación.** En la POO se juntan en un mismo lugar los datos y el código que los manipula.

Objetos

Ventajas

- **Modularidad.** La POO permite dividir los programas en partes o módulos más pequeños, que son independientes unos de otros pero pueden comunicarse entre ellos.
- **Extensibilidad.** Si se desean añadir nuevas características a una aplicación, la POO facilita esta tarea de dos formas: añadiendo nuevos métodos al código, o creando nuevos objetos que extiendan el comportamiento de los ya existentes.
- **Mantenimiento.** Los programas desarrollados utilizando POO son más sencillos de mantener, debido a la modularidad antes comentada. También ayuda seguir ciertas convenciones al escribirlos, por ejemplo, escribir cada clase en un fichero propio. No debe haber dos clases en un mismo fichero, ni otro código aparte del propio de la clase.

Objetos

La estructura de los objetos se define en las clases. En ellas se escribe el código que define el comportamiento de los objetos y se indican los miembros que formarán parte de los objetos de dicha clase. Entre los miembros de una clase puede haber:

- **Métodos.** Son los miembros de la clase que contienen el código de la misma. Un método es como una función. Puede recibir parámetros y devolver valores.
- **Atributos o propiedades.** Almacenan información acerca del estado del objeto al que pertenecen (y por tanto, su valor puede ser distinto para cada uno de los objetos de la misma clase).

Objetos

Nivel de acceso

- **public** los atributos o métodos public pueden utilizarse directamente por los objetos de la clase en cualquier sitio donde se instancie el objeto.
- **private** los atributos o métodos solo pueden ser accedidos y modificados por los métodos definidos en la clase, no directamente por los objetos de la clase.
- **protected** atributos o métodos (“privados”) visibles en las subclases (heredados) getAtributo() método que nos permite acceder a un atributo privado setAtributo() método que nos permite establecer un atributo privado \$this. referencia al objeto que hace la llamada

Objetos

Ejemplo

```
class Producto {  
    private $codigo;  
    public $nombre;  
    public $PVP;  
  
    public function muestra() {  
        print "" . $this->codigo . "";  
    }  
}  
$p = new Producto();
```

Objetos

- Métodos mágicos que, si se implementan, son utilizados por defecto
- `__set()`
- `__get()`
- `__construct()`
- `__destruct()`
- `__call()`
- `__clone()`
- `__toString()`

Objetos

Estáticos

- `const constantes` de clase static atributos y métodos estáticos – atributos y métodos de clase (public static o private static)
 - `::` operador de resolución e ámbito (-> para objetos `::` para clases)
 - `self::` clase actual (`this->` para objetos `self::` para clases)
- Utilizaremos `self::` para acceder a las constantes de clase y a los atributos y métodos estáticos.
- Utilizaremos `Clase::metodoEstatico()` para acceder a los métodos estáticos y públicos de una clase.

Objetos

Operador instanceof: if (\$p instanceof Producto)

Funciones útiles para el manejo de objetos y clases:

- get_class devuelve el nombre de la clase del objeto
- class_exists devuelve true si la clase está definida
- get_declared_classes devuelve un array con los nombres de las clases definidas
- class_alias crea un alias para una clase
- get_class_methods devuelve un array con los nombres de los métodos accesibles
- method_exists devuelve true si existe el método (sea accesible o no)
- get_class_vars devuelve un array con los nombres de los atributos accesibles
- get_object_vars devuelve un array con los nombres de los métodos accesible
- property_exists devuelve true si existe el atributo (sea accesible o no)

Objetos

Comparación de objetos == si son de la misma clase y tienen el mismo valor en sus atributos
(pero son dos objetos distintos)

Comparación de objetos === si son el mismo objetos, dos referencias al mismo objeto Alias – el mismo objeto con varios nombres – referencias al mismo objeto

Serialización – almacenamiento de un objeto en un formato no orientado a objeto

Unserialize – recuperar un objeto de su almacenamiento no orientado a objeto

Serialización automática – cuando guardamos objetos en la sesión

Serialización explícita – para guardar objetos en una base de datos relacional

Objetos

Herencia

Mecanismo de la POO que nos permite definir nuevas clases en base a otra ya existente.

```
class subclase extends superclase { }
```

Los nuevos objetos de la subclase son también objetos de la superclase.

Utilizaremos la visibilidad `protected` en la superclase con aquellos atributos o métodos privados que queremos que se hereden en la subclase. Los atributos o métodos privados no se heredan en la subclase.

Polimorfismo: Podemos redefinir los métodos `protected` creando otro método en la subclase con el mismo nombre.

Funciones asociadas a la herencia:

- `get_parent_class`: Devuelve el nombre de la clase padre del objeto o clase que se le pasa.
- `is_subclass_of`: Devuelve `true` si el objeto o clase del primer parámetro tiene como clase base la que se indica en el segundo parámetro.

Objetos

Herencia

Sintaxis de la herencia

- La herencia en PHP se consigue indicando la cláusula `extends` seguida de la clase que proporciona la herencia.
- Se heredan todos los métodos públicos y protegidos de la clase padre. A menos que una clase sobrescriba esos métodos, mantendrán su funcionalidad original.
- Se pueden sobreescribir métodos heredados para adecuarlos al comportamiento de la clase hija.
- Se utiliza la cláusula `parent::` para acceder a los miembros de la clase padre desde una clase heredada.
- El método constructor de la clase hija se debe redefinir para ajustarlo a las variables miembro de dicha clase.

Objetos

Interface

- Son declaraciones de funcionalidades que tienen que cubrir las clases que implementan la interfaz.
- En una interfaz se definen un juego de funciones, es decir, se declaran una serie de métodos o funciones sin especificar ningún código fuente asociado.
- Las clases que implementan la interfaz serán las encargadas de proporcionar un código a los métodos que contiene esa interfaz.
- Para definir una interfaz se utiliza la palabra clave interface.
- Para implementar una interfaz se utiliza la palabra clave implements.
- Se pueden implementar varias interfaces en una misma clase => herencia múltiple

Objetos Interface

Un interfaz es como una clase vacía que solo contiene declaraciones de métodos.

Se definen utilizando la palabra interface (sin la palabra class).

```
interface nombreinterface {}
```

Se utilizan en las clases que están obligadas a implementar estos métodos (entre otros).

La clase que implementa una interfaz o varias la añade en su definición con la palabra implements seguida de las interfaces que implementa.

```
class Nombrededeclase implements nombreinterface {}
```

Todos los métodos que se declaren en un interfaz deben ser públicos. Además de métodos, las interfaces podrán contener constantes pero no atributos. Un interfaz es como un contrato que la clase debe cumplir.

Funciones para interfaces:

get_declared_interfaces devuelve un array con el nombre de los interfaces declarados

interface_exists true si existe el interfaz

Objetos

Clase abstracta

- Son clases que no se pueden instanciar
- Sirven de base para otra clases
- Son como “plantillas”
- Se declaran con la palabra clave abstract

```
abstract class miClaseBase {  
    abstract protected function getItem();  
    abstract protected function quantity($qty);  
    public function listItem() {  
        $result = '<p>' . $this->getItem() . '</p>';  
        return $result;  
    }  
}
```

MVC

El patrón Modelo Vista Controlador MVC divide el código en 3 capas:

El proceso de desarrollo llamado Model View Controller (MVC), que es un estándar de programación de aplicaciones, utilizado tanto para hacer sitios web como programas tradicionales.

MVC (Model-View-Controller) es un patrón de diseño de software en torno a la interconexión de los tres tipos de componentes principales en un lenguaje de programación como PHP, a menudo con un fuerte enfoque en la programación orientada a objetos (POO).

MVC

El modelo

En el modelo mantendremos encapsulada la complejidad de nuestra base de datos y simplemente crearemos funciones para recibir, insertar, actualizar o borrar información de nuestras tablas.

Al mantenerse todas las llamadas a la base de datos en un mismo código, desde otras partes del programa podremos invocar las funciones que necesitemos del modelo y éste se encargará de procesarlas.

En el modelo nos podrán preocupar cosas como el tipo de base de datos con la que trabajamos, o las tablas y sus relaciones, pero desde las otras partes del programa simplemente llamaremos a las funciones del modelo sin importarnos qué tiene que hace éste para conseguir realizar las acciones invocadas.

El modelo representa las estructuras de datos, normalmente, en clases que contendrán métodos que le ayudarán a recuperar, insertar y actualizar información en la base de datos.

MVC

La vista

La vista es donde se encontrarán todos los elementos de la interfaz de usuario de una aplicación, esta puede contener código HTML, hojas de estilo CSS y archivos Javascript.

Cualquier cosa que el usuario pueda ver, es guardado en la vista, y algunas veces lo que ve el usuario en un momento es la combinación de varias vistas en la misma petición.

Una vista normalmente será una página web, pero una vista también puede ser un fragmento de página como un encabezado o pie de página.

MVC

El controlador

El controlador es el componente encargado de conectar el modelo con la vista.

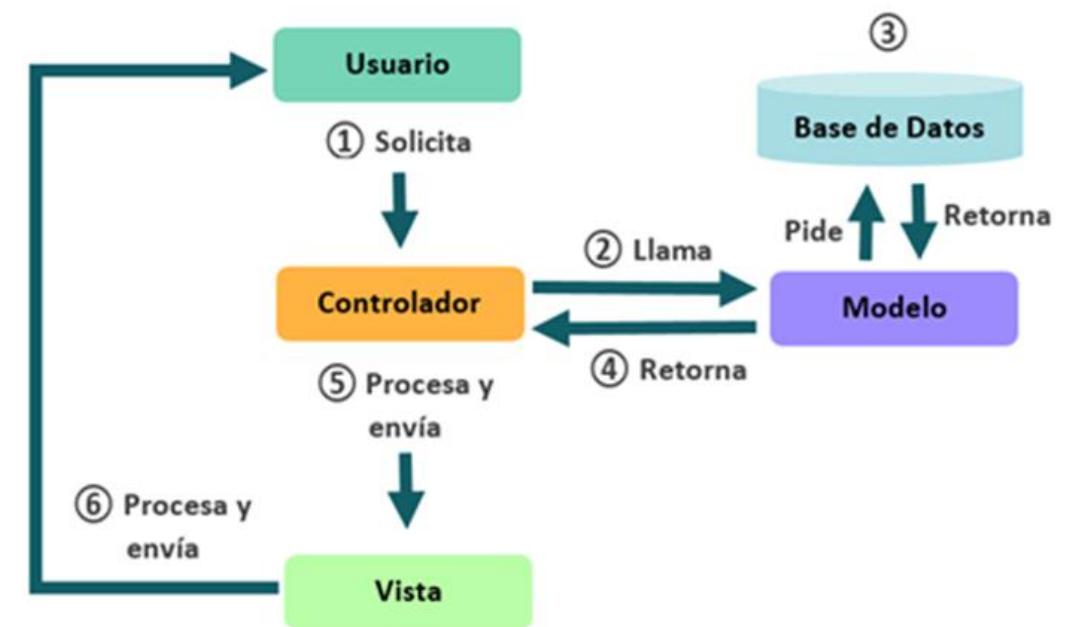
Los controladores son el primer punto de entrada en estos componentes, ya que la primera solicitud se pasa a un controlador.

En el controlador guardamos la lógica de nuestras páginas y realizamos todas las acciones que sean necesarias para generarlas, ayudados del modelo o la vista.

MVC

Ciclo

1. El usuario realiza una petición. El controlador captura la petición del usuario.
2. El controlador llama al modelo.
3. El modelo interactúa con la base de datos, y retorna la información al controlador.
4. El controlador recibe la información, la procesa y la envía a la vista.
5. La vista procesa la información recibida y la entrega de una manera visualmente entendible al usuario



MVC

Ventajas

VS

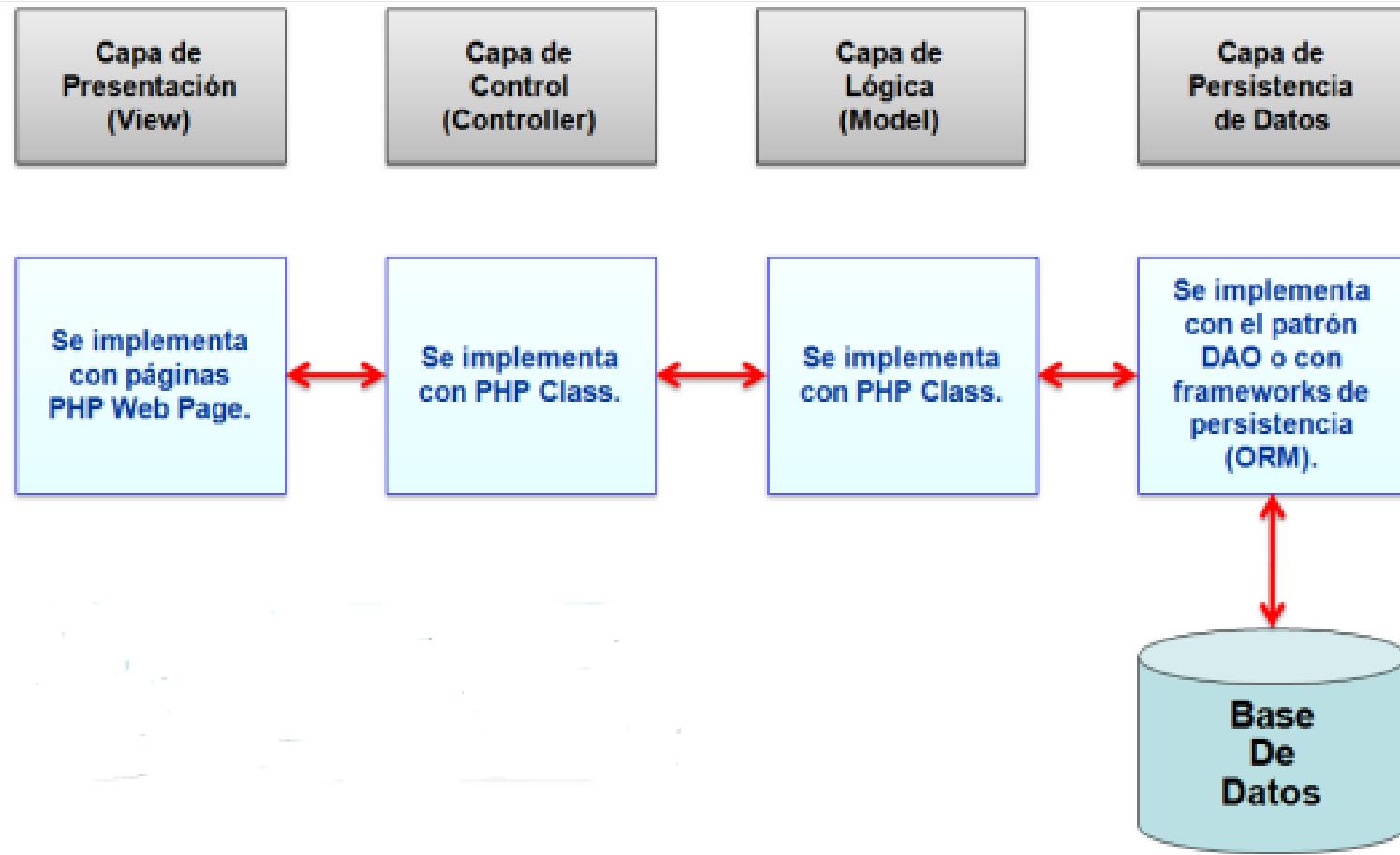
Inconvenientes

1. La separación del *Modelo* y la *Vista*, lo cual logra separar los datos, de su representación visual.
2. Facilita el manejo de errores.
3. Permite que el sistema sea escalable (aumentar sus componentes o funciones sin comprometer la operabilidad y la calidad).
4. Es posible agregar múltiples representaciones de los datos.

1. La cantidad de archivos que se deben mantener incrementa considerablemente.
2. La curva de aprendizaje es más alta que utilizando otros modelos.
3. Su separación en capas, aumenta la complejidad del sistema.

MVC

El controlador



Tema 7. Servicios web

DESARROLLO EN ENTORNO SERVIDOR

2º DAW

MARÍA CRIADO DOMÍNGUEZ

Índice

- Servicio web
- Características
- Tipos
- Api REST
 - Generar servicio
 - Consumir servicio

Servicio web

Utilizamos los servicios web cuando las aplicaciones que desarrollamos tienen que compartir información con otras aplicaciones web.

- Queremos compartir esta información sin dar acceso a la misma base de datos a dos o más aplicaciones. Solo una aplicación accede a la base de datos.
- Queremos compartir una lógica de negocio que hemos programado en una aplicación para que pueda ser utilizada (sin necesidad de volver a codificarla) en otra aplicación. Reutilización de código desarrollado en otra aplicación.
- Queremos controlar las modificaciones que otras aplicaciones realizan sobre nuestra base de datos. Validación de las modificaciones en nuestros datos.

Servicio web

Partes

- El servidor puede ofrecer un punto de acceso a la información que quiere compartir. Controla y facilita el acceso a otras aplicaciones.
- Los clientes no necesitan conocer la estructura de almacenamiento, únicamente el punto de acceso a la información que les interesa.

Alternativas

- **SOAP.** Protocolo que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML (Cliente-Servidor)
- **WSDL.** Web Services Description Language, es un formato del Extensible Markup Language (XML) que se utiliza para describir servicios web
- **REST.** La Transferencia de Estado Representacional (Representational State Transfer) o REST es una arquitectura que, haciendo uso del protocolo HTTP, proporciona una API que utiliza cada uno de sus métodos (GET, POST, PUT, DELETE, etcétera) para poder realizar diferentes operaciones entre la aplicación que ofrece el servicio web y el cliente.

API

Es una forma de describir la manera en que los programas o los sitios webs intercambian datos.

El formato de intercambio de datos normalmente es **JSON** o **XML**.

Una API especifica cómo los componentes de software deben interactuar entre sí.

Es un conjunto de protocolos y rutinas, y sus respuestas generalmente se devuelven como datos JSON o XML.

API

El cliente no necesita saber qué procedimiento se llama en el servidor.

En su lugar, utiliza un conjunto de comandos (llamados “verbos”) que están integrados en HTTP y cuando el comando llega al otro extremo, depende del sistema receptor saber qué hacer con él.

Por ejemplo, el verbo HTTP que se usa para recuperar datos es “GET”.

El sistema cliente (generalmente llamado “*consumidor*”) y el sistema servidor (el “*proveedor*”) son tan independientes entre sí, que pueden usar diferentes lenguajes (**Java, Python, Ruby, PHP, etc.**) para su implementación general.

Arquitectura REST

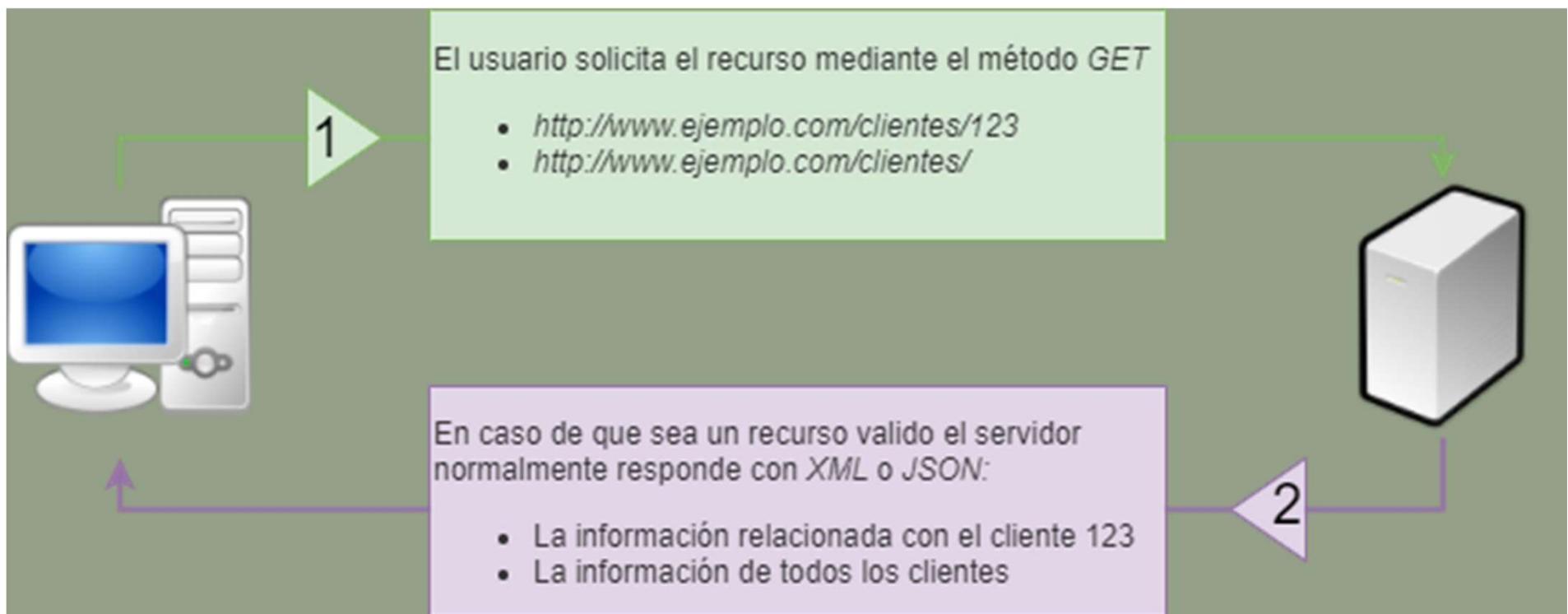
- Protocolo cliente/servidor sin estado: cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla. Esto es, nada de variables de sesión en el servidor.
- REST pide a los desarrolladores que utilicen el protocolo HTTP de forma explícita y de una forma que sea coherente con la definición del protocolo.
- Este principio básico del diseño de REST establece una correlación individual entre las operaciones de crear, leer, actualizar y borrar (CRUD) y los métodos HTTP.

Arquitectura REST

Operación	Método HTTP	URI	Parámetros	Resultado
Listar	GET	/{recurso}	No aplica	Lista del tipo de recurso, archivo JSON
Crear	POST	/{recurso}	Archivo JSON	Se crea un nuevo recurso
Leer	GET	/{recurso}/{recurso_id}	No aplica	Recurso en función al id
Actualizar	PUT	/{recurso}/{recurso_id}	Archivo JSON	Se actualiza el recurso
Borrar	DELETE	/{recurso}/{recurso_id}	No aplica	Se elimina el recurso en función al id

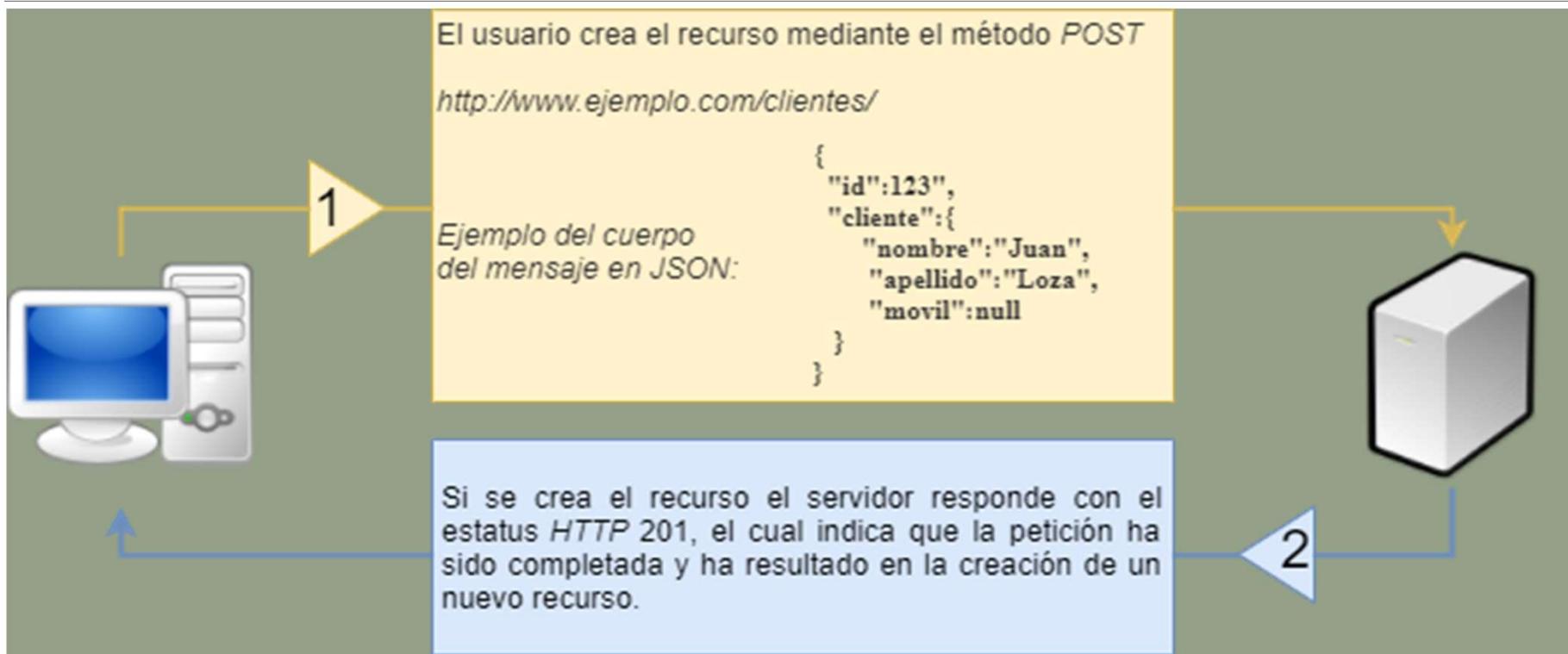
Arquitectura REST

GET



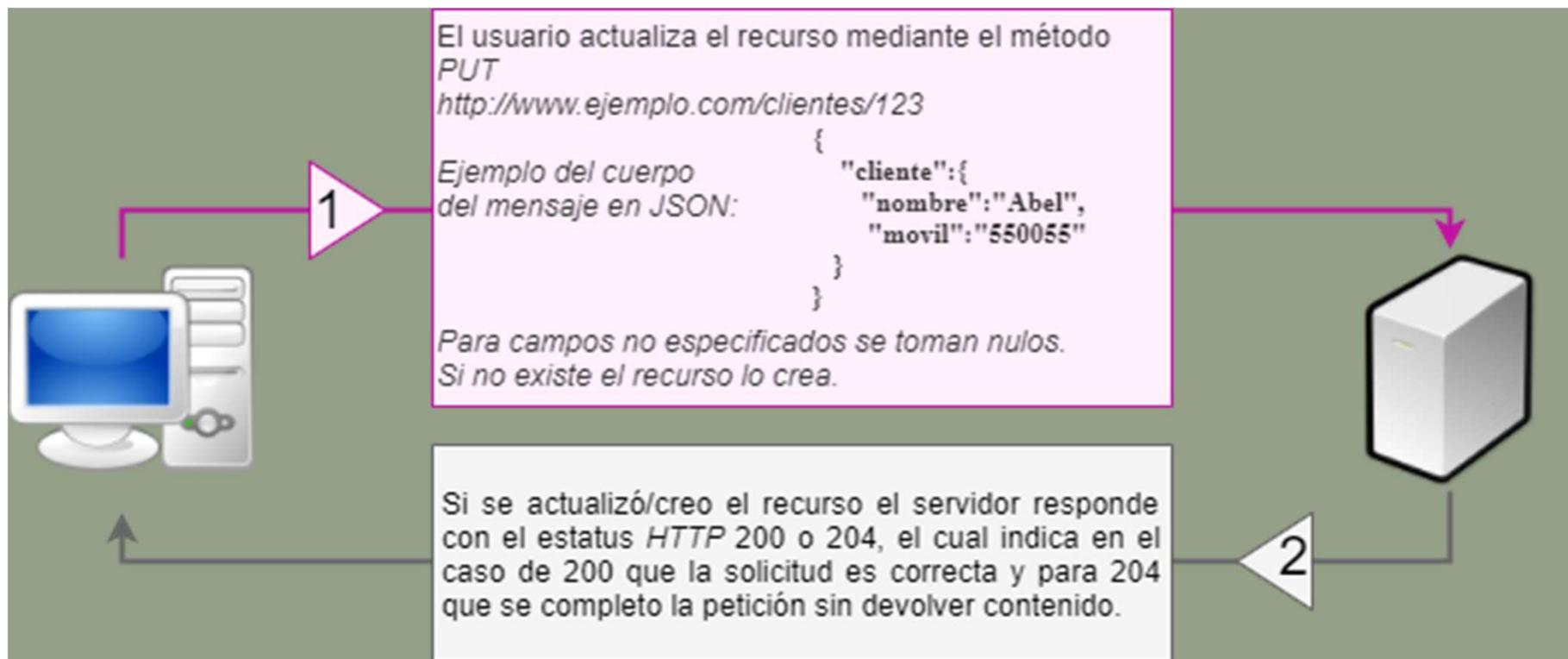
Arquitectura REST

POST



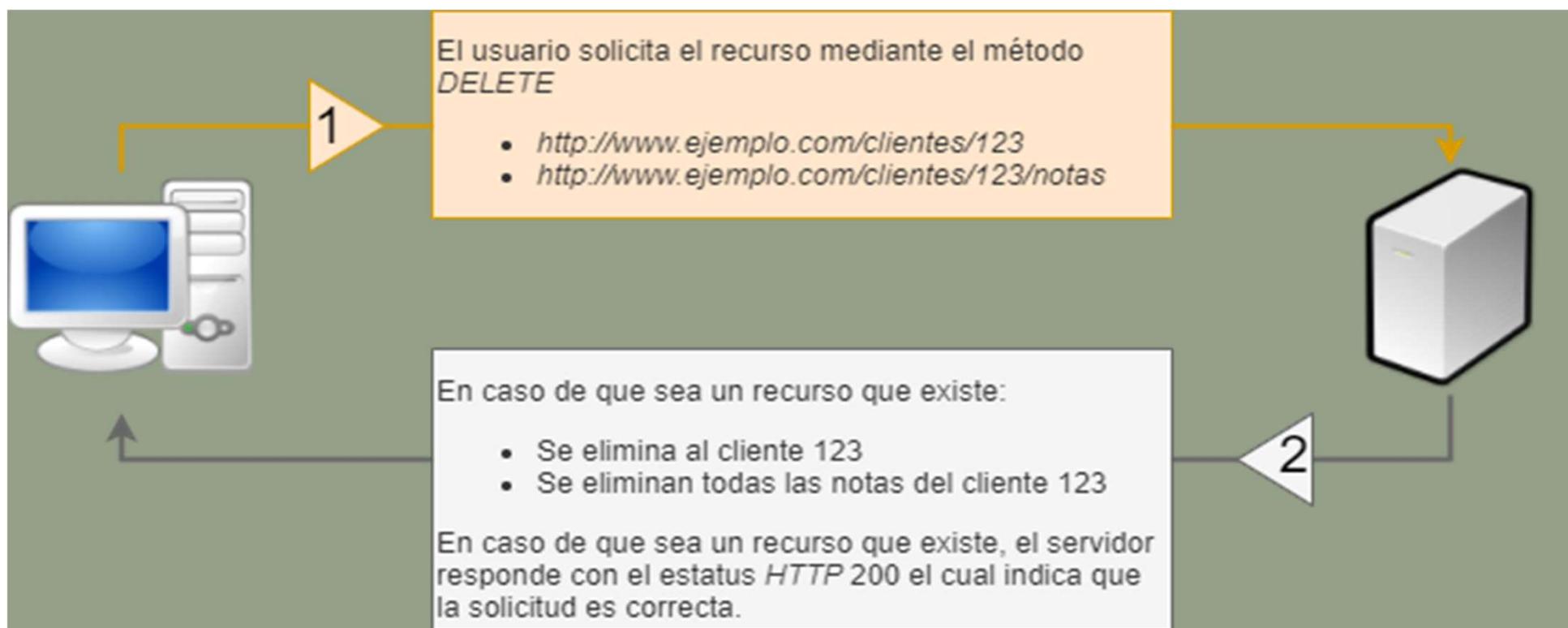
Arquitectura REST

PUT



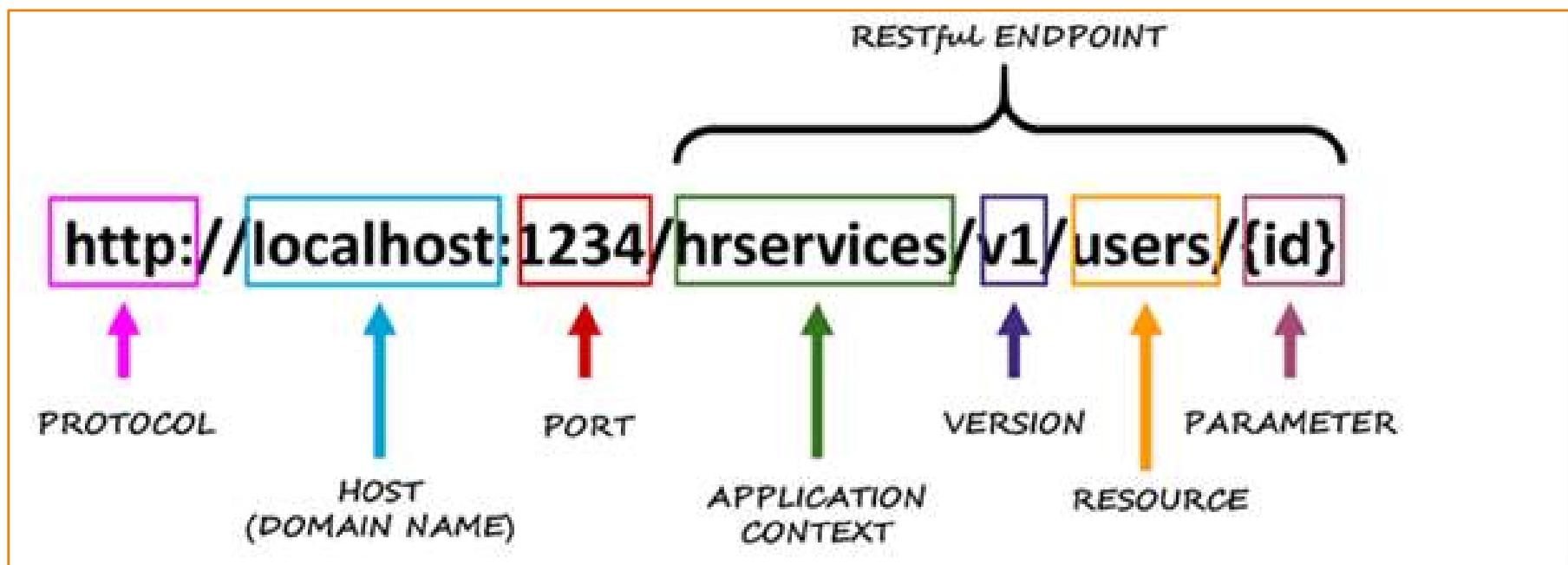
Arquitectura REST

DELETE



Arquitectura REST

Endpoint



Arquitectura REST

Endpoint

Existen varias reglas básicas para ponerle nombre a la URI de un recurso:

- Los nombres de URI no deben implicar una acción, por lo tanto debe evitarse usar **verbos** en ellos.
- Deben ser únicas, no debemos tener más de una URI para identificar un mismo recurso.
- Deben ser independiente de formato.
- Deben mantener una jerarquía lógica.
- Los **filtrados de información de un recurso no se hacen en la URI**, sino con una cadena de consulta (string query).

Arquitectura REST

Endpoint

Existen varias reglas básicas para ponerle nombre a la URI de un recurso:

- Los nombres de URI no deben implicar una acción, por lo tanto debe evitarse usar **verbos** en ellos.
- Deben ser únicas, no debemos tener más de una URI para identificar un mismo recurso.
- Deben ser independiente de formato.
- Deben mantener una jerarquía lógica.
- Los **filtrados de información de un recurso no se hacen en la URI**, sino con una cadena de consulta (string query).

Arquitectura REST Endpoint

- La URI `/facturas/orden/desc/fecha-desde/2007/pagina/2` sería incorrecta ya que el recurso de listado de facturas sería el mismo pero utilizariamos una URI distinta para filtrarlo, ordenarlo o paginarlo.

- La URI correcta en este caso sería:
`/facturas?fecha-desde=2007&orden=DESC&página=2`

Arquitectura REST

Resultado

RESTful exige que sea enviado un código de estado HTTP informando del resultado de la petición.

Para enviar la respuesta en PHP debemos utilizar un header, con el código correspondiente, ejemplo:

Header ("HTTP/1.1 **200 OK**");

- 201 Recurso creado correctamente
- 400 Petición errónea. Esto puede estar causado por varias acciones del usuario, como proveer un JSON no válido en el cuerpo de la petición, proveyendo parámetros de acción no válidos, etc.
- 404 No encontrado El recurso solicitado no existe.
- 405 Método no permitido El método HTTP de la solicitud no se permite en el recurso.
- 406 Not Acceptable. Indica que el servidor no puede producir una respuesta que coincida con la lista de valores aceptables definidos en las cabeceras del cliente, por ejemplo, pide xml y el recurso genera JSON.
- 500 Error interno del servidor Se produjo un error interno del servidor al procesar la solicitud. Por ejemplo, al conectar con la Base de Datos.

CORS

CORS son las siglas de "Cross-origin resource sharing" y es básicamente una restricción de acceso a recursos que están localizados en otros dominios.

Tu página o aplicación puede estar en <http://midominio.com> y tu servidor de API (PHP) puede estar en <http://otrodominio.com>

Para evitar que la barrera de CORS nos afecte, el servidor tiene que emitir unas cabeceras HTTP en la respuesta.

Estas cabeceras le dicen al navegador que ciertos recursos sí que van a estar disponibles desde otros dominios distintos del habitual.

Así que desde PHP es tan sencillo como escribir unas pocas líneas de código para el envío de esas cabeceras y nuestra API estará disponible para el acceso desde otros dominios.

Las cabeceras son las siguientes.

```
header('Access-Control-Allow-Origin: *');  
header("Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept");  
header('Access-Control-Allow-Methods: GET, POST, PUT, OPTIONS, DELETE');
```