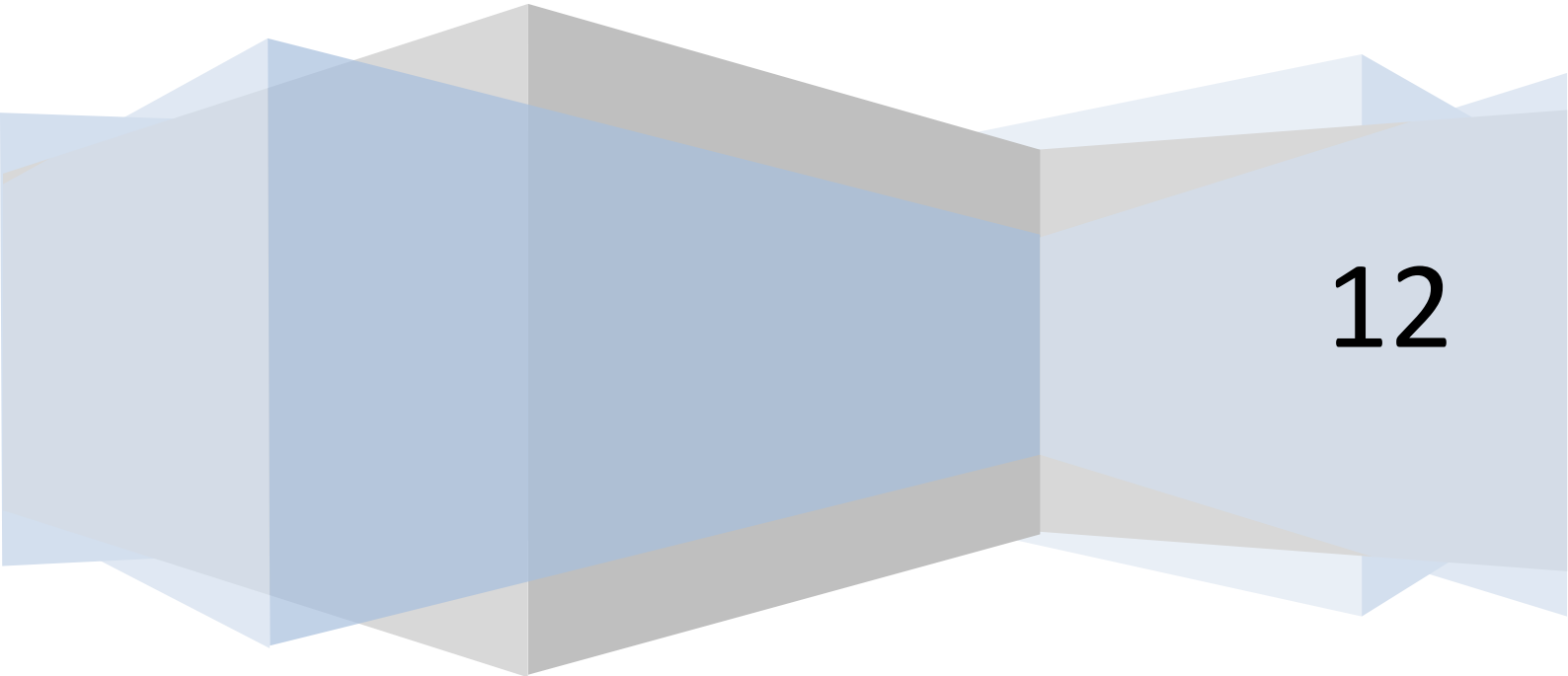


DESARROLLO WEB ENTORNO SERVIDOR

Desarrollo de Aplicaciones Web

José Luis Comesaña



12

ÍNDICE

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

Plataformas de programación web en entorno servidor. Aplicaciones LAMP.

Caso práctico

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

Esteban fundó hace ahora tres años una pequeña empresa dedicada a la venta de material tecnológico: cámaras, televisores, material informático, etc. Con el tiempo, esa pequeña empresa ha crecido. El número de ventas ha aumentado, así como el catálogo de productos que ofrece, e incluso ha abierto dos nuevas tiendas en localidades cercanas.

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

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

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

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

1.- Características de la programación web.

Caso práctico

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

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

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

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

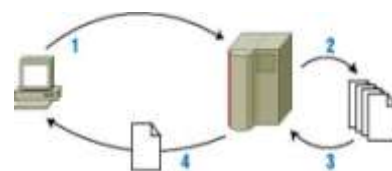
Cuando una **página web se descarga** a tu ordenador, su contenido define qué se debe mostrar en pantalla. Este contenido está programado en un **lenguaje de marcado**, formado por etiquetas, que puede ser **HTML** o **XHTML**. Las etiquetas que componen la página indican el objetivo de cada una de las partes que la componen. Así, dentro de estos lenguajes hay etiquetas para indicar que un texto es un encabezado, que forma parte de una tabla, o que simplemente es un párrafo de texto.

Además, si la página está bien estructurada, la información que le indica al navegador el **estilo** con que se debe **mostrar cada parte de la página** estará almacenado en otro fichero, una **hoja de estilos** o **CSS** (Abreviatura de "Hoja de estilos en cascada", del inglés *Cascading Style Sheet (CSS)*). Es un lenguaje utilizado para definir las características de presentación de un documento escrito en lenguaje HTML, XHTML o XML). La hoja de estilos se encuentra indicada en la página web y el navegador la descarga junto a ésta. En ella nos podemos encontrar, por ejemplo, estilos que indican que el encabezado debe ir con tipo de letra Arial y en color rojo, o que los párrafos deben ir alineados a la izquierda.

Estos dos ficheros se descargan a tu ordenador desde un servidor web como respuesta a una petición. El proceso es el que se refleja en la siguiente figura.

Los pasos son los siguientes:

1. **Tu ordenador solicita a un servidor web una página** con extensión .htm, .html o .xhtml.
2. **El servidor busca esa página** en un almacén de páginas (cada una suele ser un fichero).
3. Si **el servidor encuentra esa página**, la recupera.
4. Y por último **se la envía al navegador** para que éste pueda mostrar su contenido.



Este es un ejemplo típico de una **comunicación cliente-servidor**. El cliente es el que hace la petición e inicia la comunicación, y el servidor es el que recibe la petición y la atiende. En nuestro caso, el navegador es el cliente web.

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



Sí



No

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

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

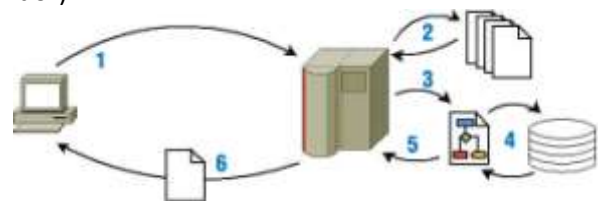
Las páginas que viste en el ejemplo anterior se llaman **páginas web estáticas**. Estas páginas se encuentran almacenadas en su forma definitiva, tal y como se crearon, y su contenido no varía. Son útiles para mostrar una información concreta, y mostrarán esa misma información cada vez que se carguen. La única forma en que pueden cambiar es si un programador la modifica y actualiza su contenido.

En contraposición a las páginas web estáticas, como ya te imaginarás, existen las **páginas web dinámicas**. Estas páginas, como su nombre indica, se caracterizan porque su contenido cambia en función de diversas variables, como puede ser el navegador que estás usando, el usuario con el que te has identificado, o las acciones que has efectuado con anterioridad.

Dentro de las **páginas web dinámicas**, es muy importante distinguir **dos tipos**:

- ✓ Aquellas que **incluyen código que ejecuta el navegador**. En estas páginas el código ejecutable, normalmente en **lenguaje JavaScript**, se incluye dentro del HTML (o XHTML) y se descarga junto con la página. Cuando el navegador muestra la página en pantalla, ejecuta el código que la acompaña. Este código puede incorporar múltiples funcionalidades que pueden ir desde mostrar animaciones hasta cambiar totalmente la apariencia y el contenido de la página. En este módulo no vamos a ver JavaScript, salvo cuando éste se relaciona con la programación web del lado del servidor.
- ✓ Como ya sabes, hay muchas páginas en Internet que no tienen extensión .htm, .html o .xhtml. Muchas de estas páginas tienen extensiones como .php, .asp, .jsp, .cgi o .aspx. En éstas, el contenido que se descarga al navegador es similar al de una página web estática: HTML (o XHTML). Lo que cambia es la forma en que se obtiene ese contenido. Al contrario de lo que vimos hasta ahora, esas páginas no están almacenadas en el servidor; más concretamente, el contenido que se almacena no es el mismo que después se envía al navegador. **El HTML de estas páginas se forma como resultado de la ejecución de un programa**, y esa ejecución tiene lugar en el servidor web (aunque no necesariamente por ese mismo servidor).

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



Pasos:

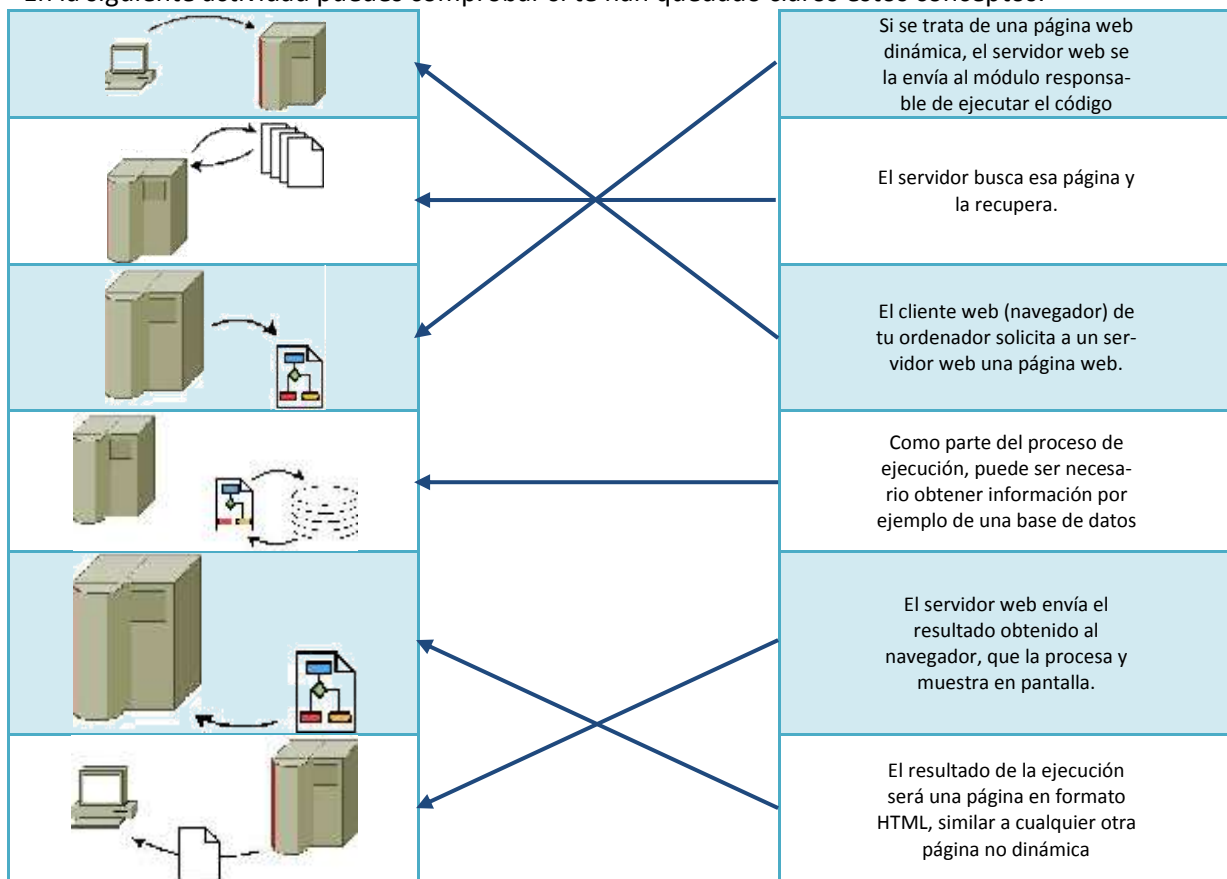
1. **El cliente web** (navegador) de tu ordenador **solicita** a un servidor web una **página web**.
2. El **servidor busca** esa página y la recupera.
3. En el caso de que se trate de una **página web dinámica**, es decir, que su contenido deba ejecutarse para obtener el HTML que se devolverá, el **servidor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

Este procedimiento tiene lugar constantemente mientras consultamos páginas web. Por ejemplo, cuando consultas tu correo en GMail, HotMail, Yahoo o cualquier otro servicio de correo vía web, lo primero que tienes que hacer es introducir tu nombre de usuario y contraseña. A continuación, lo más habitual es que el servidor te muestre una pantalla con la bandeja de entrada, en la que apare-

cen los mensajes recibidos en tu cuenta. Esta pantalla es un claro ejemplo de una página web dinámica.

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

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



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

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

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

Para que Google muestre un sitio web en sus resultados de búsqueda, previamente tiene que indexar su contenido. Es decir, un programa recorre las páginas del sitio consultando su contenido y clasificándolo. Si las páginas se generan de forma dinámica, puede ser que su contenido, en parte o

por completo, no sea visible para el buscador y por tanto no quedará indexado. Esto nunca sucedería en un sitio que utilizase páginas web estáticas.

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

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

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

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

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

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

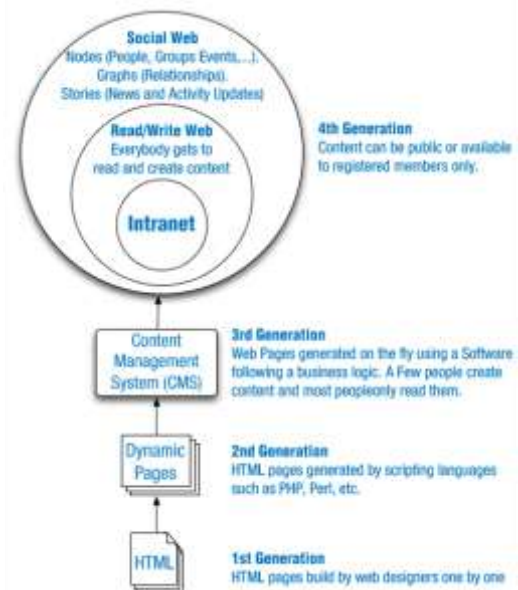
1.1.1.1.- Aplicaciones web.

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

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

Ventajas de las aplicaciones web:

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



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

Inconvenientes de las aplicaciones web:

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

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

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



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

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

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

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

Como vimos, cuando tu navegador solicita a un servidor web una página, es posible que antes de enviártela haya tenido que ejecutar, por sí mismo o por delegación, algún programa para obtenerla. Ese programa es el que genera, en parte o en su totalidad, la página web que llega a tu equipo. En estos casos, **el código se ejecuta en el entorno del servidor web**.

Además, cuando una página web llega a tu navegador, es también posible que incluya algún programa o fragmentos de código que se deban ejecutar. Ese código, normalmente en lenguaje **JavaScript**, **se ejecutará en tu navegador** y, además de poder modificar el contenido de la página, también puede llevar a cabo acciones como la animación de textos u objetos de la página o la comprobación de los datos que introduces en un formulario.

Estas dos tecnologías se complementan una con otra. Así, volviendo al ejemplo del correo web, el programa que se encarga de obtener tus mensajes y su contenido de una base de datos se ejecuta en el entorno del servidor, mientras que tu navegador ejecuta, por ejemplo, el código encargado de avisarte cuando quieres enviar un mensaje y te has olvidado de poner un texto en el asunto.



Esta división es así porque el **código que se ejecuta en el cliente web** (en tu navegador) no tiene, o mejor dicho **tradicionalmente no tenía, acceso a los datos** que se almacenan en el **servidor**. Es decir, cuando en tu navegador querías leer un nuevo correo, el código Javascript que se ejecutaba en el mismo no podía obtener de la base de datos el contenido de ese mensaje. La solución era crear una nueva página en el servidor con la información que se pedía y enviarla de nuevo al navegador.

Sin embargo, desde hace unos años existe una **técnica de desarrollo web conocida como AJAX**, que nos posibilita realizar programas en los que el código JavaScript que se ejecuta en el navegador pueda comunicarse con un servidor de Internet para obtener información con la que, por ejemplo, modificar la página web actual.

En nuestro ejemplo, cuando pulsas con el ratón encima de un correo que quieres leer, la página puede contener código Javascript que detecte la acción y, en ese instante, consultar a través de Internet el texto que contiene ese mismo correo y mostrarlo en la misma página, modificando su estructura en caso de que sea necesario. Es decir, sin salir de una página poder modificar su contenido en base a la información que se almacena en un servidor de Internet.

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

Muchas de las **aplicaciones web actuales utilizan estas dos tecnologías**: la ejecución de **código en el servidor y en el cliente**. Así, el código que se ejecuta en el servidor genera páginas web que ya incluyen código destinado a su ejecución en el navegador. Hacia el final de este módulo verás las técnicas que se usan para programar aplicaciones que incorporen esta funcionalidad.

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



En el navegador web.

En el servidor web.

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

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

Caso práctico

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

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

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

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

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

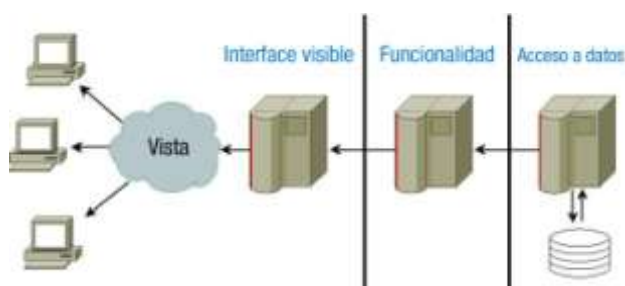
Los **componentes principales** con los que debes contar para ejecutar aplicaciones web en un servidor son los siguientes:

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

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

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

En una aplicación puedes distinguir, de forma general, **funciones de presentación** (se encarga de dar formato a los datos para presentárselo al usuario final), **lógica** (utiliza los datos para ejecutar un proceso y obtener un resultado), **persistencia** (que mantiene los datos almacenados de forma organizada) y **acceso** (que ob-



tiene e introduce datos en el espacio de almacenamiento).

Cada capa puede ocuparse de una o varias de las funciones anteriores. Por ejemplo, en las aplicaciones de **3 capas** nos podemos encontrar con:

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

2.1.- Arquitecturas y plataformas.

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

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

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

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

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

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

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



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

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

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

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

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

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

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

2.1.1.- Selección de una arquitectura de programación web.

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

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

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

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

Cada proyecto tiene sus peculiaridades. Aunque unas arquitecturas son más potentes que otras, no hay una en concreto que podamos considerar mejor que las demás para todos los casos. ¿Crees que vale la pena el esfuerzo de cambiar de arquitectura de desarrollo para cada aplicación? ¿Es necesario en todos los casos?

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

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

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

2.2.- Integración con el servidor web.

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

En el lado del servidor, estas peticiones son procesadas por el servidor web (también llamado servidor HTTP). Es por tanto el servidor web el encargado de decidir cómo procesar las peticiones que recibe. Cada una de las arquitecturas que acabamos de ver tiene una forma de integrarse con el servidor web para ejecutar el código de la aplicación.

La tecnología más antigua es CGI. CGI es un protocolo estándar que existe en muchas plataformas. Lo implementan la gran mayoría de servidores web. Define qué debe hacer el servidor web para delegar en un programa externo la generación de una página web. Esos programas externos se conocen como guiones CGI, independientemente del lenguaje en el que estén programados (aunque se suelen programar en lenguajes de guiones como Perl).

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

Aunque también es posible ejecutar código en lenguaje PHP utilizando CGI, los intérpretes PHP no se suelen utilizar con este método. Al igual que **mod_perl**, existen otros módulos que podemos instalar en el servidor web Apache para que ejecute páginas web dinámicas. El módulo **mod_php** es la forma habitual que se utiliza para ejecutar guiones en PHP utilizando plataformas AMP, y su equivalente para el lenguaje Python es **mod_python**.

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



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

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

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

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

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

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

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

3.- Lenguajes.

Caso práctico

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

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

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

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

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

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

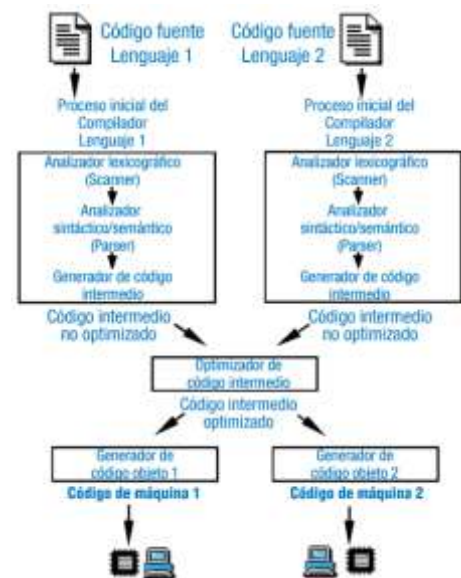
- ✓ **Lenguajes compilados a código nativo** (Denominación habitual del lenguaje máquina. Código que puede ser ejecutado directamente por el procesador). Son aquellos en los que el código fuente se traduce a código binario, dependiente del procesador, antes de ser ejecutado. El servidor web almacena los programas en su modo binario, que ejecuta directamente cuando se les invoca.

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

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

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

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



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

- ✓ Los lenguajes de guiones tienen la ventaja de que no es necesario traducir el código fuente original para ser ejecutados, lo que aumenta su portabilidad. Si se necesita realizar alguna modificación a un programa, se puede hacer en el momento. Por el contrario el proceso de interpretación ofrece un peor rendimiento que las otras alternativas.
- ✓ Los lenguajes compilados a código nativo son los de mayor velocidad de ejecución, pero tienen problemas en lo relativo a su integración con el servidor web. Son programas de propósito general que no están pensados para ejecutarse en el entorno de un servidor web. Por ejemplo, no se reutilizan los procesos para atender a varias peticiones: por cada petición que se haga al servidor web, se debe ejecutar un nuevo proceso. Además los programas no son portables entre distintas plataformas.
- ✓ Los lenguajes compilados a código intermedio ofrecen un equilibrio entre las dos opciones anteriores. Su rendimiento es muy bueno y pueden portarse entre distintas plataformas en las que exista una implementación de la arquitectura (como un contenedor de servlets o un servidor de aplicaciones Java EE).

3.1.- Código embebido en el lenguaje de marcas.

Cuando la web comenzó a evolucionar desde las páginas web estáticas a las dinámicas, una de las primeras tecnologías que se utilizaron fue la ejecución de código utilizando CGI. Los guiones CGI son programas estándar, que se ejecutan por el sistema operativo, pero que generan como salida el código HTML de una página web. Por tanto, los guiones CGI deben contener, mezcladas dentro de su código, sentencias encargadas de generar la página web.

Por ejemplo, si quieres generar una página web utilizando CGI a partir de un guión de sentencias en Linux, tienes que hacer algo como lo siguiente:

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

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

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

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

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

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

Los servlets de Java EE se diferencian de las páginas JSP en que los primeros son programas Java compilados y almacenados en el contenedor de servlets. Sin embargo, las páginas JSP contienen código Java embebido en lenguaje HTML y se almacenan de forma individual en el servidor web. La primera vez que se necesita una página JSP, se convierte a un servlet y éste se guarda para ser utilizado en posteriores llamadas a la misma página.

En la arquitectura ASP.Net, cada página se divide en dos ficheros: uno contiene las etiquetas HTML y otro el código en el lenguaje de programación utilizado. De esta forma se logra cierta independencia entre el aspecto de la aplicación y la gestión del contenido dinámico. A partir de esos ficheros se obtiene un código intermedio (MSIL en la terminología de la plataforma) que es lo que almacena el servidor.

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

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

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

3.2.- Herramientas de programación.

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

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

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

- ✓ **Resaltado de texto.** Muestra con distinto color o tipo de letra los diferentes elementos del lenguaje: sentencias, variables, comentarios, etc. También genera indentado automático para diferenciar de forma clara los distintos bloques de un programa.
- ✓ **Completado automático.** Detecta qué estás escribiendo y cuando es posible te muestra distintas opciones para completar el texto.
- ✓ **Navegación en el código.** Permite buscar de forma sencilla elementos dentro del texto, por ejemplo, definiciones de variables.
- ✓ **Comprobación de errores al editar.** Reconoce la sintaxis del lenguaje y revisa el código en busca de errores mientras lo escribes.

- ✓ **Generación automática de código.** Ciertas estructuras, como la que se utiliza para las clases, se repiten varias veces en un programa. La generación automática de código puede encargarse de crear la estructura básica, para que sólo tengas que rellenarla.
- ✓ **Ejecución y depuración.** Esta característica es una de las más útiles. El IDE se puede encargar de ejecutar un programa para poder probar su funcionamiento. Además, cuando algo no funciona, te permite depurarlo con herramientas como la ejecución paso a paso, el establecimiento de puntos de ruptura o la inspección del valor que almacenan las variables.
- ✓ **Gestión de versiones.** En conjunción con un sistema de control de versiones, el entorno de desarrollo te puede ayudar a guardar copias del estado del proyecto a lo largo del tiempo, para que si es necesario puedas revertir los cambios realizados.

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



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

3.2.1.- Instalación de NetBeans para PHP en Linux.

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

```
java -version
```

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

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

O si fuera la 10.10:

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

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

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

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



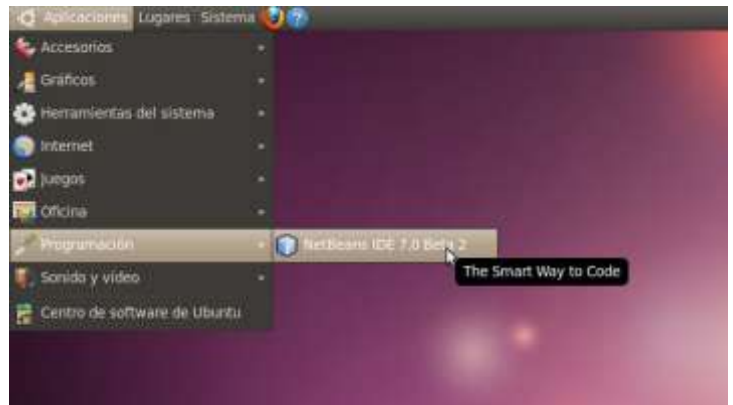
Escoge el paquete en español, para sistemas Linux (x86/x64) y lenguaje de programación PHP.

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

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

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

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



3.2.2.- Instalación de una plataforma LAMP en Ubuntu.

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

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

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

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

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



Con la configuración predeterminada, la carpeta raíz de dónde el servidor web Apache extrae los documentos que va a publicar es `/var/www`. Por tanto, para probar el correcto funcionamiento de la plataforma puedes crear en ese directorio un fichero `prueba.php` con el siguiente contenido:

```
<?php
    phpinfo();
?>
```

Lo que estás haciendo es llamar a una función del lenguaje PHP que genera por sí misma una página web completa con información sobre la instalación de PHP. Si todo va bien, al abrir esa página con un navegador web (desde la propia máquina de Ubuntu la URL será `http://localhost/prueba.php`) deberás ver algo como lo



siguiente:

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

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

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

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

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

3.3.- Programación web con Java.

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



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

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

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

Por ejemplo, si quieres calcular la suma de dos números y enviar el resultado al navegador, lo podríamos realizar con una página JSP, incluyendo el código en Java dentro de las etiquetas HTML utilizando los delimitadores `<%` y `%>` de la siguiente manera:

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

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

[illegible][illegible]

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

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

3.4.- Programación web con PHP.

PHP es un lenguaje de guiones de propósito general, pero diseñado para el desarrollo de páginas web dinámicas utilizando código embebido dentro del lenguaje de marcas. Su sintaxis está basada en la de C / C++, y por lo tanto es muy similar a la de Java. Aunque lo puedes hacer de otras formas, los delimitadores recomendados para incluir código PHP dentro de una página web son `<?php` y `?>`.



El código se ejecuta por un entorno de ejecución con el que se integra el servidor web (normalmente utilizando Apache con el módulo `mod_php`). La configuración tanto del servidor web Apache, como de PHP, se realiza por medio de ficheros de configuración. El de Apache es `httpd.conf`, y el de PHP es `php.ini`. Este fichero, `php.ini`, puede encontrarse en distintas ubicaciones. La función `phpinfo()` que ejecutaste antes te informa, entre otras muchas cosas, del lugar en que se encuentra almacenado el fichero `php.ini` en tu ordenador. En nuestro caso es en `/etc/php5/apache2/php.ini`.

Dependiendo de cómo se integre PHP con Apache, los cambios que realices en su fichero de configuración se aplicarán en un momento o en otro. Si como es nuestro caso, utilizamos `mod_php` para ejecutar PHP como un módulo de Apache, las opciones de configuración de PHP se aplicarán cada vez que se reinicie Apache. Por tanto, no te olvides de hacerlo cada vez que hagas cambios en `php.ini`. Por ejemplo: `sudo /etc/init.d/apache2 restart` o también `sudo service apache2 restart`.

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

- ✓ **short_open_tags**. Indica si se pueden utilizar en PHP los delimitadores cortos `<?` y `?>`. Es preferible no usarlos, pues puede causarnos problemas si utilizamos páginas con XML. Para prohibir la utilización de estos delimitadores con PHP le asignamos a esta directiva el valor Off.
- ✓ **max_execution_time**. Permite que puedas ajustar el número máximo de segundos que podrá durar la ejecución de un script PHP. Evita que el servidor se bloquee si se produce algún error en un script.
- ✓ **error_reporting**. Indica qué tipo de errores se mostrarán en el caso de que se produzcan. Por ejemplo, si haces `error_reporting = E_ALL`, te mostrará todos los tipos de errores. Si no quieres que te muestre los avisos pero sí otros tipos de errores, puedes hacer `error_reporting = E_ALL & ~E_NOTICE`.
- ✓ **file_uploads**. Indica si se pueden o no subir ficheros al servidor por HTTP.
- ✓ **upload_max_filesize**. En caso de que se puedan subir ficheros por HTTP, puedes indicar el límite máximo permitido para el tamaño de cada archivo. Por ejemplo, `upload_max_filesize = 1M`.

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

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

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

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

A medida que vayas escribiendo código en PHP, será útil que introduzcas en el mismo algunos comentarios que ayuden a revisarlo cuando lo necesites. En una página web los comentarios al HTML van entre los delimitadores `<!--` y `-->`. Dentro del código PHP, hay tres formas de poner comentarios:

- ✓ Comentarios de una línea utilizando `//`. Son comentarios al estilo del lenguaje C. Cuando una línea comienza por los símbolos `//`, toda ella se considera que contiene un comentario, hasta la siguiente línea.
- ✓ Comentarios de una línea utilizando `#`. Son similares a los anteriores, pero utilizando la sintaxis de los scripts de Linux.
- ✓ Comentarios de varias líneas. También iguales a los del lenguaje C. Cuando en una línea aparecen los caracteres `/*`, se considera que ahí comienza un comentario. El comentario puede extenderse varias líneas, y acabará cuando escribas la combinación de caracteres opuesta: `*/`.

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

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

3.4.1.- Variables y tipos de datos en PHP.

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

```
$mi_variable = 7;
```

Dependiendo del valor que se le asigne, a la variable se le aplica un tipo de datos, que puede cambiar si cambia su contenido. Esto es, el tipo de la variable se decide en función del contexto en que se emplee.

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

Los tipos de datos simples en PHP son:

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

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

Por ejemplo:

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

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

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

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

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

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

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

3.4.2.- Expresiones y operadores.

Como en muchos otros lenguajes, en PHP se utilizan las expresiones para realizar acciones dentro de un programa. Todas las expresiones deben contener al menos un operando y un operador. Por ejemplo:

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

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

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

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

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

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

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

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

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

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

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

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

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

3.4.3.- Ámbito de utilización de las variables.

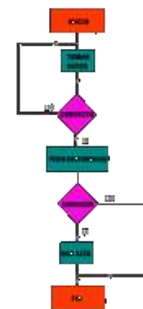
En PHP puedes utilizar variables en cualquier lugar de un programa. Si esa variable aún no existe, la primera vez que se utiliza se reserva espacio para ella. En ese momento, dependiendo del lugar del código en que aparezca, se decide desde qué partes del programa se podrá utilizar esa variable. A esto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

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

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

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

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



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

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

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

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

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

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

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

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

TEMA 2

Contenido

1.- Elementos del lenguaje PHP.....	1
1.1.- Generación de código HTML.....	2
Utilización de la función print en PHP	2
Especificadores de tipo para las funciones <code>printf</code> y <code>sprintf</code>	3
1.2.- Cadenas de texto.....	4
Secuencias de escape.....	4
1.3.- Funciones relacionadas con los tipos de datos(I).	5
empty	5
1.3.1.- Funciones relacionadas con los tipos de datos (II).	6
Función date: caracteres de formato para fechas y horas.	7
1.4.- Variables especiales de PHP.	8
Principales valores de la variable \$_SERVER	8
2.- Estructuras de control.....	10
2.1.- Condicionales.....	10
2.2.- Bucles.....	12
3.- Funciones.....	14
3.1.- Inclusión de ficheros externos.	14
3.2.- Ejecución y creación de funciones.	15
3.3.- Argumentos.....	16
4.- Tipos de datos compuestos.....	19
4.1.- Recorrer arrays (I).....	20
4.1.1.- Recorrer arrays (II).	21
4.2.- Funciones relacionadas con los tipos de datos compuestos.....	22
5.- Formularios web.	24
5.1.- Procesamiento de la información devuelta por un formulario web.	25
Código de "procesa.php"	25
Código necesario para procesar los datos.....	25
Ejemplo formulario web utilizando request.....	26
Procesar datos en la misma página que el formulario.....	26
5.2.- Generación de formularios web en PHP.	27
Ejemplo de validación	28

Características del lenguaje PHP.

Caso práctico

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

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

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

1.- Elementos del lenguaje PHP.

Caso práctico

Carlos comienza su aprendizaje del nuevo lenguaje. Conforme va avanzando, comprueba que muchos de los conceptos que aprende son similares a lo que ya conocía. Otros, sin embargo, son muy distintos y los tiene que practicar para entender bien su manejo.

Aunque es consciente que al principio va a cometer fallos, se pone como meta utilizar lo que va aprendiendo para hacer pequeños programas que pueda volver a usar en el futuro. Y si consigue hacer algo que pueda tener alguna utilidad para la nueva aplicación, ¡mejor!

En la unidad anterior, aprendiste a preparar un entorno para programar en PHP. Además también viste algunos de los elementos que se usan en el lenguaje, como las variables y tipos de datos, comentarios, operadores y expresiones.

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

En esta unidad aprenderás a utilizar otros elementos del lenguaje que te permitan crear programas completos en PHP. 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.

Por ejemplo, en las siguientes líneas tenemos dos bloques de código en PHP:

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

Aunque no se utilice el valor de las variables, en el segundo bloque de código la variable `$a` mantiene el valor 1 que se le ha asignado anteriormente.

En esta unidad empezarás a crear tus propios programas en PHP. Para ello vas a usar el IDE NetBeans, que instalaste anteriormente. Deberías organizar tus programas en proyectos, y almacenarlos en una estructura en árbol, colgando todos por ejemplo de

/home/usuario/NetBeansProyectos/DWES. Para crear un proyecto nuevo vete a **Archivo – Proyecto Nuevo** y selecciona **PHP Application**.

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

```
sudo ln -s /home/usuario/NetBeansProyectos/DWES/ DWES
```

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



1.1.- Generación de código HTML.

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

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

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

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

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

Utilización de la función print en PHP

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Utilización de print -->
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Desarrollo Web</title>
  </head>
  <body>
    <?php
      $modulo="DWES";
      print "<p>Módulo: ";
      print $modulo;
      print"</p>"
    ?>
  </body>
</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. Por ejemplo:

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

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

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

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

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

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

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

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

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

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



Sí



No

Efectivamente. A partir de la variable real, y utilizando la función `sprintf`, podemos obtener una cadena con el texto exacto que queremos mostrar, haciendo `sprintf("%.3f", $a)`

1.2.- Cadenas de texto.

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

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

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

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

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

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

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

Secuencias de escape.

Secuencia	Resultado
<code>\\</code>	muestra una barra invertida.
<code>\'</code>	muestra una comilla simple.
<code>\"</code>	muestra una comilla doble.
<code>\n</code>	muestra un avance de línea (LF o 0x0A (10) en ASCII).
<code>\r</code>	muestra un retorno de carro (CR o 0x0D (13) en ASCII).
<code>\t</code>	muestra un tabulador horizontal (HT o 0x09 (9) en ASCII).
<code>\v</code>	muestra un tabulador vertical (VT o 0x0B (11) en ASCII).
<code>\f</code>	muestra un avance de página (FF o 0x0C (12) en ASCII).
<code>\\$</code>	muestra un signo de dólar.

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

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

En PHP tienes otra alternativa para crear cadenas: la sintaxis **heredoc**. Consiste en poner el operador `<<<` seguido de un identificador de tu elección, y a continuación y empezando en la línea siguiente la cadena de texto, sin utilizar comillas. La cadena finaliza cuando escribes ese mismo identificador en una nueva línea. Esta línea de cierre no debe llevar más caracteres, ni siquiera espacios o sangría, salvo quizás un punto y coma después del identificador.

```
<?php
    $a = <<<MICADENA
    Desarrollo de Aplicaciones Web<br />
    Desarrollo Web en Entorno Servidor
    MICADENA;
    print $a;
?>
```

El texto se procesa de igual forma que si fuera una cadena entre comillas dobles, sustituyendo variables y secuencias de escape. Si no quisieras que se realizara ninguna sustitución, debes poner el identificador de apertura entre comillas simples.

```
$a = <<<'MICADENA'
...
MICADENA;
```

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

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

También podemos comprobar si la variable es de un tipo concreto utilizando una de las siguientes funciones: `is_array()`, `is_bool()`, `is_float()`, `is_integer()`, `is_null()`, `is_numeric()`, `is_object()`, `is_resource()`, `is_scalar()` e `is_string()`. Devuelven `true` si la variable es del tipo indicado.

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

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

El resultado del código anterior es:

```
$a vale 3.1416 y es de tipo string
$b vale 3.1416 y es de tipo double
```

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

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

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

empty

Determina si una variable está vacía. Tiene la sintaxis: `bool empty($var)`

Una variable se considera vacía si no existe si su valor es igual a `FALSE`. No genera una advertencia si la variable no existe.

`empty()` sólo comprueba variables ya que cualquier otra cosa producirá un error de análisis. En otras palabras, lo siguiente no funcionará: `empty(trim($name))`. Use en su lugar `trim($name) == false`.

No se genera una advertencia si la variable no existe. Esto significa que `empty()` es esencialmente el equivalente conciso de `!isset($var) || $var == false`.

Devuelve `FALSE` si `var` existe y tiene un valor no vacío, distinto de cero. De otro modo devuelve `TRUE`.

Las siguientes expresiones son consideradas como vacías:

- ✓ `""` (una cadena vacía)
- ✓ `0` (0 como un integer)
- ✓ `0` (0 como un float)
- ✓ `"0"` (0 como un string)
- ✓ `NULL`
- ✓ `FALSE`
- ✓ `array()` (un array vacío)
- ✓ `$var;` (una variable declarada, pero sin un valor)

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

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

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

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

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

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

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

- ☐ La variable '\$a' vale -3.14.
- ☐ La variable '\$a' vale +3.14.
- ☒ La variable '\$a' vale -3.14.
- ☐ La variable '\$a' vale +3.14.

Efectivamente. Dentro de comillas dobles no se sustituye la secuencia de escape \'. Además, el signo + en un especificador de conversión indica que se visualice el signo del número aunque sea positivo, pero el signo nunca se cambia y siempre se muestra si el número es negativo.

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

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

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

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

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

Función date: caracteres de formato para fechas y horas.

Carácter	Resultado
d	día del mes con dos dígitos.
j	día del mes con uno o dos dígitos (sin ceros iniciales).
z	día del año, comenzando por el cero (0 = 1 de enero).
N	día de la semana (1 = lunes, ..., 7 = domingo).
w	día de la semana (0 = domingo, ..., 6 = sábado).
l	texto del día de la semana, en inglés (Monday, ..., Sunday).
D	texto del día de la semana, solo tres letras, en inglés (Mon, ..., Sun).
W	número de la semana del año.
m	número del mes con dos dígitos.
n	número del mes con uno o dos dígitos (sin ceros iniciales).
t	número de días que tiene el mes.
F	texto del día del mes, en inglés (January, ..., December).
M	texto del día del mes, solo tres letras, en inglés (Jan, ..., Dec).
Y	número del año.
y	dos últimos dígitos del número del año.
L	1 si el año es bisiesto, 0 si no lo es.
h	formato de 12 horas, siempre con dos dígitos.
H	formato de 24 horas, siempre con dos dígitos.
g	formato de 12 horas, con uno o dos dígitos (sin ceros iniciales).
G	formato de 24 horas, con uno o dos dígitos (sin ceros iniciales).
i	minutos, siempre con dos dígitos.
s	segundos, siempre con dos dígitos.
u	microsegundos.
a	am o pm, en minúsculas.
A	AM o PM, en mayúsculas.
r	fecha entera con formato RFC 2822.

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

Para que el sistema pueda darte información sobre tu fecha y hora, debes indicarle tu zona horaria. Puedes hacerlo con la función `date_default_timezone_set`. Para establecer la zona horaria en España peninsular debes indicar:

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

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

<http://es2.php.net/manual/es/timezones.php>

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

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

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

<http://es2.php.net/manual/es/ref.datetime.php>

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

Se utiliza para definir constantes		define
Muestra una cadena con formato		printf
Indica si una variable está definida y su valor no es null		isset
Establece el tipo de una variable		settype
Obtiene una cadena de texto a partir de una fecha/hora		date
Indica si una variable es de tipo cadena de texto (string)		is_string
Obtiene un array con información sobre la fecha y hora actual		getdate
establece la zona horaria		date_default_timezone_set

1.4.- Variables especiales de PHP.

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

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

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

Principales valores de la variable `$_SERVER`

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

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

<http://es.php.net/manual/es/reserved.variables.server.php>

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

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

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

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

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

En posteriores unidades iremos trabajando con estas variables.

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

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

Relaciona cada variable con la información que contiene:

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

4
1
3
2

2.- Estructuras de control.

Caso práctico

¡Bien! Ya están claros los fundamentos del lenguaje.

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

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

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

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



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

2.1.- Condicionales.

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

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

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

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

switch. La sentencia **switch** es similar a enlazar varias sentencias **if** comparando una misma variable con diferentes valores. Cada valor va en una sentencia **case**. Cuando se encuentra una coincidencia, comienzan a ejecutarse las sentencias siguientes hasta que acaba el bloque **switch**, o hasta que se

encuentra una sentencia `break`. Si no existe coincidencia con el valor de ningún `case`, se ejecutan las sentencias del bloque `default`, en caso de que exista.

```
<?php
switch ($a) {
    case 0:
        print "a vale 0";
        break;
    case 1:
        print "a vale 1";
        break;
    default:
        print "a no vale 0 ni 1";
}
?>
```

Haz una página web que muestre la fecha actual en castellano, incluyendo el día de la semana, con un formato similar al siguiente: "Miércoles, 13 de Abril de 2011".

Compara la solución propuesta con la que has obtenido.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Mostrar fecha en castellano -->
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Fecha en castellano</title>
</head>
<body>
    <?php
        date_default_timezone_set('Europe/Madrid');
        $numero_mes = date("m");
        $numero_dia_semana = date("N");
        switch($numero_mes){
            case 1: $mes = "Enero";
                break;
            case 2: $mes = "Febrero";
                break;
            case 3: $mes = "Marzo";
                break;
            case 4: $mes = "Abril";
                break;
            case 5: $mes = "Mayo";
                break;
            case 6: $mes = "Junio";
                break;
            case 7: $mes = "Julio";
                break;
            case 8: $mes = "Agosto";
                break;
            case 9: $mes = "Septiembre";
                break;
            case 10: $mes = "Octubre";
                break;
            case 11: $mes = "Noviembre";
                break;
            case 12: $mes = "Diciembre";
                break;
        }
        switch($numero_dia_semana){
            case 1: $dia_semana = "Lunes";
                break;
            case 2: $dia_semana = "Martes";
                break;
            case 3: $dia_semana = "Miércoles";
                break;
            case 4: $dia_semana = "Jueves";
                break;
            case 5: $dia_semana = "Viernes";
                break;
            case 6: $dia_semana = "Sábado";
                break;
        }
    >
```

```

        case 7: $dia_semana = "Domingo";
            break;
        }
        print $dia_semana.", ".date("j")." de ".$mes." de ".date("Y");
    ?>
</body>
</html>

```

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



Sí



No

Efectivamente. Una sentencia `switch` se puede sustituir siempre por una o varias sentencias `if` en las que las condiciones de cada `if` serán comparar el valor de la variable con cada una de las constantes que figuran en las sentencias `case`

2.2.- Bucles.

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



```

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

```

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

```

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

```

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

```

for (expr1; expr2; expr3)
    sentencia o conjunto de sentencias;

```

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

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

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

```

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

```

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

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

http://www.youtube.com/watch?feature=player_embedded&v=VjqJwSy_BPQ

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



while



do-while

Efectivamente. Como no sabemos si está o no vacía, antes de imprimir algún carácter hay que comprobar si existe, incluido el primero. Con un bucle `do-while` se intentaría mostrar una letra antes de comprobar si existe.

3.- Funciones.

Caso práctico

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

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

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

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

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

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

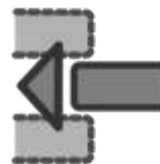
```
pecl install nombre_extensión
```

En el manual de PHP tienes más información sobre PECL.

<http://es2.php.net/manual/es/install.pecl.php>

3.1.- Inclusión de ficheros externos.

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



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

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

Te presentamos un ejemplo de utilización de `include`.

definiciones.php

```
<?php
    $modulo = 'DWES';
    $ciclo = 'DAW';
?>
```

programa.php

```
<?php
    print "Módulo $modulo del ciclo $ciclo<br />"; //Solo muestra "Modulo del ciclo"
    include 'definiciones.php';
    print " Módulo $modulo del ciclo $ciclo<br />"; // muestra "Modulo DWES del ciclo DAW"
?>
```

Cuando se comienza a evaluar el contenido del fichero externo, se abandona de forma automática el modo PHP y su contenido se trata en principio como etiquetas HTML. Por este motivo, es necesario delimitar el código PHP que contenga nuestro archivo externo utilizando dentro del mismo los delimitadores `<?php` y `?>`.

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

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

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



Sí



No

Efectivamente. Como ya viste, el contenido de los ficheros externos se trata como HTML a no ser que figuren los delimitadores `<?php` y `?>`. No es necesario incluir código PHP en un fichero externo para poder utilizar `include` o `require` con él

3.2.- Ejecución y creación de funciones.

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

```
<?php
    phpinfo();
?>
```

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

```
<?php
    function precio_con_iva() {
        global $precio;
        $precio_iva = $precio * 1.18;
        print "El precio con IVA es ".$precio_iva;
    }
    $precio = 10;
    precio_con_iva();
?>
```

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

```
<?php
    $iva = true;
    $precio = 10;
    precio_con_iva(); // Da error, pues aquí aún no está definida la función
    if ($iva) {
        function precio_con_iva() {
```

```

    global $precio;
    $precio_iva = $precio * 1.18;
    print "El precio con IVA es ".$precio_iva;
}
}
precio_con_iva(); // Aquí ya no da error
?>

```

Cuando una función está definida de una forma condicional sus definiciones deben ser procesadas antes de ser llamadas. Por tanto, la definición de la función debe estar antes de cualquier llamada.

3.3.- Argumentos.

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



Puedes reescribir la función anterior de la siguiente forma:

```

<?php
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
?>

```

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

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

```

<?php
function precio_con_iva($precio, $iva=0.18) {
    return $precio * (1 + $iva);
}
$precio = 10;
$precio_iva = precio_con_iva($precio);
print "El precio con IVA es ".$precio_iva
?>

```

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

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

```

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

```

¿Está bien definida una función con el siguiente encabezado?

```
function precio_final (&$precio, $iva=0.18, $aplicar_iva)
```



Sí



No

La función no puede tener esos argumentos, pues los opcionales (\$iva) deben estar a la derecha de cualquier otro que no tenga valor por defecto(\$aplicar_iva).

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

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Utilización include -->
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Fecha en castellano</title>
  </head>
  <body>
    <?php
      include 'funciones.inc.php';
      print fecha();
    ?>
  </body>
</html>
```

```
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Utilización include -->
<!-- Fichero: funciones.inc.php -->
<?php
  // Función que devuelve un texto con la fecha actual en castellano
  function fecha(){
    date_default_timezone_set('Europe/Madrid');
    $numero_mes = date("m");
    $numero_dia_semana = date("N");
    switch($numero_mes){
      case 1: $mes = "Enero";
        break;
      case 2: $mes = "Febrero";
        break;
      case 3: $mes = "Marzo";
        break;
      case 4: $mes = "Abril";
        break;
      case 5: $mes = "Mayo";
        break;
      case 6: $mes = "Junio";
        break;
      case 7: $mes = "Julio";
        break;
      case 8: $mes = "Agosto";
        break;
      case 9: $mes = "Septiembre";
        break;
      case 10: $mes = "Octubre";
        break;
      case 11: $mes = "Noviembre";
        break;
      case 12: $mes = "Diciembre";
        break;
    }
    switch($numero_dia_semana){
      case 1: $dia_semana = "Lunes";
        break;
```

```
        case 2: $dia_semana = "Martes";  
            break;  
        case 3: $dia_semana = "Miércoles";  
            break;  
        case 4: $dia_semana = "Jueves";  
            break;  
        case 5: $dia_semana = "Viernes";  
            break;  
        case 6: $dia_semana = "Sábado";  
            break;  
        case 7: $dia_semana = "Domingo";  
            break;  
    }  
    return $dia_semana.", ".date("j")." de ".$mes." de ".date("Y");  
}  
?>
```

4.- Tipos de datos compuestos.

Caso práctico

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

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

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

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

```
// array numérico
$modulos1 = array(0 => "Programación", 1 => "Bases de datos", ..., 9 => "Desarrollo web en
entorno servidor");
// array asociativo
$modulos2 = array("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "Desarrollo
web en entorno servidor");
```

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

<http://es.php.net/manual/es/function.print-r.php>

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

```
$modulos1 [9]
$modulos2 ["DWES"]
```

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

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

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

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

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

```
// array numérico
$modulos1 [0] = "Programación";
$modulos1 [1] = "Bases de datos";
...
$modulos1 [9] = "Desarrollo web en entorno servidor";
// array asociativo
$modulos2 ["PR"] = "Programación";
$modulos2 ["BD"] = "Bases de datos";
...
$modulos2 ["DWES"] = "Desarrollo web en entorno servidor";
```

Ni siquiera es necesario que especifiques el valor de la clave. Si la omites, el array se irá llenando a partir de la última clave numérica existente, o de la posición 0 si no existe ninguna:

```
$modulos1 [ ] = "Programación";
$modulos1 [ ] = "Bases de datos";
...
$modulos1 [ ] = "Desarrollo web en entorno servidor";
```

4.1.- Recorrer arrays (I).

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

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

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

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

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

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

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

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Tabla con los valores del array $_SERVER utilizando foreach -->
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Tabla</title>
    <style type="text/css">
        td, th {border: 1px solid grey; padding: 4px;}
        th {text-align:center;}
        table {border: 1px solid black;}
    </style>
</head>
<body>
    <table>
        <tbody>
            <tr>
                <th>Variable</th>
                <th>Valor</th>
            </tr>
            <?php
                foreach ($_SERVER as $variable => $valor) {
                    print "<tr>";
                    print "<td>".$variable."</td>";
                    print "<td>".$valor."</td>";
                    print "</tr>";
                }
            <?>
        </tbody>
    </table>
</body>
</html>
```

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

```
$a[0] = 0;
$a[1] = "uno";
$a["tres"] = 3;
```



Sí



No

No es necesario que la clave de un array sea siempre numérica o texto. Pueden mezclarse ambos tipos de valores.

4.1.1.- Recorrer arrays (II).

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

Funciones para recorrer arrays.

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.

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

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

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

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

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

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

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Tabla con los valores del array $_SERVER utilizando función each -->
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Tabla</title>
```

```

<style type="text/css">
    td, th {border: 1px solid grey; padding: 4px;}
    th {text-align:center;}
    table {border: 1px solid black;}
</style>
</head>
<body>
    <table>
        <tbody>
            <tr>
                <th>Variable</th>
                <th>Valor</th>
            </tr>
            <?php
                reset($_SERVER);
                while ($valor = each($_SERVER)) {
                    print "<tr>";
                    print "<td>".$_SERVER[0]."</td>";
                    print "<td>".$_SERVER[1]."</td>";
                    print "</tr>";
                }
            ?>
        </tbody>
    </table>
</body>
</html>

```

4.2.- Funciones relacionadas con los tipos de datos compuestos.

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

```

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

```

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

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

```

unset ($modulos [0]);
// El primer elemento pasa a ser $modulos [1] == "Bases de datos";

```

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

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

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

```

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

```

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

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

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

<http://es.php.net/manual/es/ref.array.php>

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

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



Sí



No

El bucle anterior funcionará bien sólo en aquellos arrays en los que estemos seguros de que no existe ningún valor "false".

5.- Formularios web.

Caso práctico

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

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

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

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

Los formularios HTML van encerrados siempre entre las etiquetas `<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.

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

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

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

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

http://www.aulaClic.es/html/t_8_1.htm

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

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Formulario web -->
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Formulario web</title>
  </head>
  <body>
    <form name="input" action="procesa.php" method="post">
      Nombre del alumno: <input type="text" name="nombre" /><br />
      <p>Ciclos que cursa:</p>
      <input type="checkbox" name="ciclos[]" value="DWES" /> Desarrollo web en entorno
        servidor<br />
      <input type="checkbox" name="ciclos[]" value="DWECE" /> Desarrollo web en entorno
        cliente<br />
    <br />
  </body>
</html>
```

```

        <input type="submit" value="Enviar" />
    </form>
</body>
</html>

```

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

5.1.- Procesamiento de la información devuelta por un formulario web.

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

Código de "procesa.php"

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Procesar datos post -->
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Desarrollo Web</title>
</head>
<body>
    <?php
        $nombre = $_POST['nombre'];
        $modulos = $_POST['modulos'];
        print "Nombre: ".$nombre."<br />";
        foreach ($modulos as $modulo) {
            print "Modulo: ".$modulo."<br />";
        }
    ?>
</body>
</html>

```

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

Código necesario para procesar los datos

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Procesar datos get -->
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Desarrollo Web</title>
</head>
<body>
    <?php
        $nombre = $_GET['nombre'];
        $modulos = $_GET['modulos'];
        print "Nombre: ".$nombre."<br />";
        foreach ($modulos as $modulo) {
            print "Modulo: ".$modulo."<br />";
        }
    ?>
</body>
</html>

```

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

Ejemplo formulario web utilizando request

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Procesar datos request -->
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Desarrollo Web</title>
  </head>
  <body>
    <?php
      $nombre = $_REQUEST['nombre'];
      $modulos = $_REQUEST['modulos'];
      print "Nombre: ".$nombre."<br />";
      foreach ($modulos as $modulo) {
        print "Modulo: ".$modulo."<br />";
      }
    <?>
  </body>
</html>
```

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

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

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

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

Procesar datos en la misma página que el formulario

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Procesar datos en la misma página que el formulario -->
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Desarrollo Web</title>
  </head>
  <body>
    <?php
      if (isset($_POST['enviar'])) {
        $nombre = $_POST['nombre'];
        $modulos = $_POST['modulos'];
        print "Nombre: ".$nombre."<br />";
        foreach ($modulos as $modulo) {
```

```

        print "Modulo: ".$modulo."<br />";
    }
} else {
?>
<form name="input" action="<?php echo $_SERVER['PHP_SELF'];?>" method="post">
    Nombre del alumno: <input type="text" name="nombre" /><br />
    <p>Módulos que cursa:</p>
    <input type="checkbox" name="modulos[]" value="DWES" />
    Desarrollo web en entorno servidor<br />
    <input type="checkbox" name="modulos[]" value="DWEK" />
    Desarrollo web en entorno cliente<br />
    <br />
    <input type="submit" value="Enviar" name="enviar"/>
</form>
<?php
}
?>
</body>
</html>

```

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

5.2.- Generación de formularios web en PHP.

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

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

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

```

if (!empty($_POST['modulos']) && !empty($_POST['nombre'])) {
    // Aquí se incluye el código a ejecutar cuando los datos son correctos
}
else {
    // Aquí generamos el formulario, indicando los datos incorrectos
    // y rellenando los valores correctamente introducidos
}

```

Para que el usuario no pierda, después de enviar el formulario, los datos correctamente introducidos, utiliza el atributo **value** en las entradas de texto:

```

Nombre del alumno:
<input type="text" name="nombre" value="<?php echo $_POST['nombre'];?>" />
Y el atributo checked en las casillas de verificación:
<input type="checkbox" name="modulos[]" value="DWES"
<?php
    if (in_array("DWES", $_POST['modulos']))
        echo 'checked="checked"';
?>
/>

```

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

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

```

Nombre del alumno:
<input type="text" name="nombre" value="<?php echo $_POST['nombre'];?>" />
<?php
    if (isset($_POST['enviar']) && empty($_POST['nombre']))
        echo "<span style='{color:red}'> <-- Debe introducir un nombre!!</span>"
?><br />

```

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

Ejemplo de validación

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Validar datos en la misma página que el formulario -->
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Desarrollo Web</title>
</head>
<body>
<?php
    if (!empty($_POST['modulos']) && !empty($_POST['nombre'])) {
        $nombre = $_POST['nombre'];
        $modulos = $_POST['modulos'];
        print "Nombre: ".$nombre."<br />";
        foreach ($modulos as $modulo) {
            print "Modulo: ".$modulo."<br />";
        }
    } else {
        ?>
<form name="input" action="<?php echo $ SERVER['PHP SELF'];?>" method="post">
    Nombre del alumno:
    <input type="text" name="nombre" value="<?php echo $_POST['nombre'];?>" />
    <?php
        if (isset($_POST['enviar']) && empty($_POST['nombre']))
            echo "<span style='color:red'> &lt;-- Debe introducir un nombre!!</span>"
    ?><br />
    <p>Módulos que cursa:
    <?php
        if (isset($_POST['enviar']) && empty($_POST['modulos']))
            echo "<span style='color:red'> &lt;-- Debe escoger al menos uno!!</span>"
    ?>
    </p>
    <input type="checkbox" name="modulos[]" value="DWES"
    <?php
        if (isset($_POST['modulos']) && in_array("DWES", $_POST['modulos']))
            echo 'checked="checked"';
    ?>
    />
    Desarrollo web en entorno servidor
    <br />
    <input type="checkbox" name="modulos[]" value="DWEC"
    <?php
        if (isset($_POST['modulos']) && in_array("DWEC", $_POST['modulos']))
            echo 'checked="checked"';
    ?>
    />
    Desarrollo web en entorno cliente<br />
    <br />
    <input type="submit" value="Enviar" name="enviar"/>
</form>
<?php
    }
    ?>
</body>
</html>
```

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

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

```
if (count($_REQUEST)>0)
{
    ...
}
```



Sí



No

De esta forma se comprueba si se ha enviado algún formulario. Habría que comprobar también si los datos que se han enviado son correctos.

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

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

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 2 : Características del Lenguaje PHP -->
<!-- Ejemplo: Mostrar fecha completa a partir de día, mes y año introducidos -->
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Fecha completa a partir de día, mes y año</title>
  </head>
  <body>
    <?php
      date_default_timezone_set('Europe/Madrid');
      if (!empty($_POST['dia']) && !empty($_POST['mes']) && !empty($_POST['ano'])) {
        if (checkdate($_POST['mes'], $_POST['dia'], $_POST['ano'])) {
          $fecha = mktime(0,0,0,$_POST['mes'],$_POST['dia'],$_POST['ano']);
          $numero_dia_semana = date("N", $fecha);
          switch($_POST['mes']){
            case 1: $mes = "Enero";
              break;
            case 2: $mes = "Febrero";
              break;
            case 3: $mes = "Marzo";
              break;
            case 4: $mes = "Abril";
              break;
            case 5: $mes = "Mayo";
              break;
            case 6: $mes = "Junio";
              break;
            case 7: $mes = "Julio";
              break;
            case 8: $mes = "Agosto";
              break;
            case 9: $mes = "Septiembre";
              break;
            case 10: $mes = "Octubre";
              break;
            case 11: $mes = "Noviembre";
              break;
            case 12: $mes = "Diciembre";
              break;
          }
          switch($numero_dia_semana){
            case 1: $dia_semana = "Lunes";
              break;
            case 2: $dia_semana = "Martes";
              break;
            case 3: $dia_semana = "Miércoles";
              break;
            case 4: $dia_semana = "Jueves";
              break;
            case 5: $dia_semana = "Viernes";
              break;
            case 6: $dia_semana = "Sábado";
              break;
            case 7: $dia_semana = "Domingo";
              break;
          }
          print $dia_semana.", ".date("j", $fecha)." de ".$mes." de ".date("Y", $fecha);
        }else
          print "<span style='color:red'>La fecha introducida no es correcta!!</span>";
      }
    <?>
    <form name="input" action="<?php echo $_SERVER['PHP_SELF'];?>" method="post">
      Día:
      <input type="text" name="dia" value="<?php echo $_POST['dia'];?>" />
    <?php
```

```
        if (isset($_POST['enviar']) && empty($_POST['dia']))
            echo "<span style='color:red'> &lt;-- Debe introducir un día!!</span>"
    ?><br />
    Mes:
    <input type="text" name="mes" value="<?php echo $_POST['mes'];?>" />
    <?php
        if (isset($_POST['enviar']) && empty($_POST['mes']))
            echo "<span style='color:red'> &lt;-- Debe introducir un mes!!</span>"
    ?><br />
    Año:
    <input type="text" name="ano" value="<?php echo $_POST['ano'];?>" />
    <?php
        if (isset($_POST['enviar']) && empty($_POST['ano']))
            echo "<span style='color:red'> &lt;-- Debe introducir un año!!</span>"
    ?><br />
    <input type="submit" value="Enviar" name="enviar"/>
</form>
</body>
</html>
```

TEMA 3

Contenido

1.- Acceso a bases de datos desde PHP.....	1
Características básicas de la utilización de objetos en PHP	2
Las clases: class.....	2
Utilizar la clase	3
La variable \$this	3
Constructores	3
2.- MySQL.....	5
2.1.- Instalación y configuración.	5
2.2.- Herramientas de administración.	6
2.2.1.- mysql y mysqladmin.....	7
2.2.2.- phpMyAdmin.....	9
3.- Utilización de bases de datos MySQL en PHP.....	15
3.1.- Extensión MySQLi.....	15
3.1.1.- Establecimiento de conexiones.	16
3.1.2.- Ejecución de consultas.....	17
3.1.3.- Transacciones.....	18
3.1.4.- Obtención y utilización de conjuntos de resultados.	20
3.1.5.- Consultas preparadas.	22
3.2.- PHP Data Objects (PDO).	25
3.2.1.- Establecimiento de conexiones.....	26
3.2.2.- Ejecución de consultas.....	27
3.2.3.- Obtención y utilización de conjuntos de resultados.	28
3.2.4.- Consultas preparadas.	30
4.- Errores y manejo de excepciones.....	33
4.1.- Excepciones.....	34

Trabajar con bases de datos en PHP.

Caso práctico

Una de las tareas prioritarias que tienen que abordar en el nuevo proyecto de BK Programación es el almacenamiento de la información que utilizará la aplicación web, y el método de acceso que se utilizará para manejarla desde PHP.

En una reunión de trabajo, **Esteban** les informa que para la gestión de la empresa están utilizando una aplicación de código libre que almacena los datos en un servidor MySQL. Afortunadamente, este servidor es el más utilizado en la programación con lenguaje PHP, por lo que no tendrán problemas en integrar la nueva aplicación web con la ya existente. Solo necesitan conocer la estructura de los datos que se almacenan, y ver qué métodos puede usar para manejar la información.

1.- Acceso a bases de datos desde PHP.

Caso práctico

Carlos es nuevo en el mundo de la programación web. Además, apenas ha trabajado con bases de datos, por lo que se asombra de la gran diversidad de opciones que existen en PHP para trabajar con datos almacenados en servidores de distintos tipos.

Algunos de los gestores sobre los que lee mientras revisa la documentación de PHP los conoce, otros simplemente le suenan, pero hay muchos de los que ni siquiera conocía su existencia. Sabe que debe centrarse en el servidor MySQL, que es el que usarán para desarrollar la aplicación, pero aun así el volumen de información disponible es tan grande que le cuesta decidirse por dónde empezar.

Una de las aplicaciones más frecuentes de PHP es generar un interface web para acceder y gestionar la información almacenada en una base de datos. Usando PHP podemos mostrar en una página web información extraída de la base de datos, o enviar sentencias al gestor de la base de datos para que elimine o actualice algunos registros.

PHP soporta más de 15 sistemas gestores de bases de datos: SQLite, Oracle, SQL Server, PostgreSQL, IBM DB2, MySQL, etc. Hasta la versión 5 de PHP, el acceso a las bases de datos se hacía principalmente utilizando extensiones específicas para cada sistema gestor de base de datos (extensiones nativas). Es decir, que si queríamos acceder a una base de datos de PostgreSQL, deberíamos instalar y utilizar la extensión de ese gestor en concreto. Las funciones y objetos a utilizar eran distintos para cada extensión.

A partir de la versión 5 de PHP se introdujo en el lenguaje una extensión para acceder de una forma común a distintos sistemas gestores: PDO. La gran ventaja de PDO está clara: podemos seguir utilizando una misma sintaxis aunque cambiemos el motor de nuestra base de datos. Por el contrario, en algunas ocasiones preferiremos seguir usando extensiones nativas en nuestros programas. Mientras PDO ofrece un conjunto común de funciones, las extensiones nativas normalmente ofrecen más potencia (acceso a funciones específicas de cada gestor de base de datos) y en algunos casos también mayor velocidad.

De los distintos SGBD existentes, vas a aprender a utilizar MySQL. MySQL es un gestor de bases de datos relacionales de código abierto bajo licencia GNU GPL. Es el gestor de bases de datos más empleado con el lenguaje PHP. Como ya vimos, es la letra "M" que figura en los acrónimos AMP y XAMPP.

En esta unidad vas a ver cómo acceder desde PHP a bases de datos MySQL utilizando tanto PDO como la extensión nativa MySQLi. Previamente verás una pequeña introducción al manejo de MySQL, aunque para el seguimiento de esta unidad se supone que conoces el lenguaje SQL utilizado en la gestión de bases de datos relacionales.

Además, para el acceso a las funcionalidades de ambas extensiones deberás utilizar objetos. Aunque más adelante verás todas las características que nos ofrece PHP para crear programas orientados a objetos, debemos suponer también en este punto un cierto conocimiento de programación orientada a objetos. Básicamente, debes saber cómo crear y utilizar objetos.

En PHP se utiliza la palabra `new` para crear un nuevo objeto instanciando una clase:

```
$a = new A();
```

Y para acceder a los miembros de un objeto, debes utilizar el operador flecha `->`:

```
$a->fecha();
```

Características básicas de la utilización de objetos en PHP

La programación orientada a objetos es una metodología de programación avanzada y bastante extendida, en la que los sistemas se modelan creando clases, que son un conjunto de datos y funcionalidades. Las clases son definiciones, a partir de las que se crean objetos. Los objetos son ejemplares de una clase determinada y como tal, disponen de los datos y funcionalidades definidos en la clase.

La programación orientada a objetos permite concebir los programas de una manera bastante intuitiva y cercana a la realidad. La tendencia es que un mayor número de lenguajes de programación adopten la programación orientada a objetos como paradigma para modelizar los sistemas. Prueba de ello es la nueva versión de PHP (5), que implanta la programación de objetos como metodología de desarrollo. También Microsoft ha dado un vuelco hacia la programación orientada a objetos, ya que .NET dispone de varios lenguajes para programar y todos orientados a objetos.

Así pues, la programación orientada a objetos es un tema de gran interés, pues es muy utilizada y cada vez resulta más esencial para poder desarrollar en casi cualquier lenguaje moderno. En este artículo vamos a ver algunas nociones sobre la programación orientada a objetos en PHP. Aunque es un tema bastante amplio, novedoso para muchos y en un principio, difícil de asimilar, vamos a tratar de explicar la sintaxis básica de PHP para utilizar objetos, sin meternos en mucha teoría de programación orientada a objetos en general.

Las clases: *class*

Una clase es un conjunto de variables, llamados atributos, y funciones, llamadas métodos, que trabajan sobre esas variables. Las clases son, al fin y al cabo, una definición: una especificación de propiedades y funcionalidades de elementos que van a participar en nuestros programas.

Por ejemplo, la clase "Caja" tendría como atributos características como las dimensiones, color, contenido y cosas semejantes. Las funciones o métodos que podríamos incorporar a la clase "caja" son las funcionalidades que deseamos que realice la caja, como `introduce()`, `muestra contenido()`, `comprueba si cabe()`, `vaciate()`...

Las clases en PHP se definen de la siguiente manera:

```
<?
class Caja{
    var $alto;
    var $ancho;
    var $largo;
    var $contenido;
    var $color;

    function introduce($cosa){
        $this->contenido = $cosa;
    }
}
```

```
function muestra_contenido(){  
    echo $this->contenido;  
}  
}  
?>
```

En este ejemplo se ha creado la clase Caja, indicando como atributos el ancho, alto y largo de la caja, así como el color y el contenido. Se han creado, para empezar, un par de métodos, uno para introducir un elemento en la caja y otro para mostrar el contenido.

Si nos fijamos, los atributos se definen declarando unas variables al principio de la clase. Los métodos se definen declarando funciones dentro de la clase. La variable `$this`, utilizada dentro de los métodos la explicaremos un poco más abajo.

Utilizar la clase

Las clases solamente son definiciones. Si queremos utilizar la clase tenemos que crear un ejemplar de dicha clase, lo que corrientemente se le llama instanciar un objeto de una clase.

```
$micaja = new Caja;
```

Con esto hemos creado, o mejor dicho, instanciado, un objeto de la clase Caja llamado `$micaja`.

```
$micaja->introduce("algo");  
$micaja->muestra_contenido();
```

Con estas dos sentencias estamos introduciendo "`algo`" en la caja y luego estamos mostrando ese contenido en el texto de la página. Nos fijamos que los métodos de un objeto se llaman utilizando el código "`->`".

```
nombre_del_objeto->nombre_de_metodo()
```

Para acceder a los atributos de una clase también se accede con el código "`->`". De esta forma:

```
nombre_del_objeto->nombre_del_atributo
```

La variable `$this`

Dentro de un método, la variable `$this` hace referencia al objeto sobre el que invocamos el método. En la invocación `$micaja->introduce("algo")` se está llamando al método `introduce` sobre el objeto `$micaja`. Cuando se está ejecutando ese método, se vuelca el valor que recibe por parámetro en el atributo `contenido`. En ese caso `$this->contenido` hace referencia al atributo `contenido` del objeto `$micaja`, que es sobre el que se invocaba el método.

Constructores

Los constructores son funciones, o métodos, que se encargan de realizar las tareas de inicialización de los objetos al ser instanciados. Es decir, cuando se crean los objetos a partir de las clases, se llama a un constructor que se encarga de inicializar los atributos del objeto y realizar cualquier otra tarea de inicialización que sea necesaria.

No es obligatorio disponer de un constructor, pero resultan muy útiles y su uso es muy habitual. En el ejemplo de la caja, que comentábamos anteriormente, lo normal sería inicializar las variables como color o las relacionadas con las dimensiones y, además, indicar que el contenido de la caja está vacío. Si no hay un constructor no se inicializan ninguno de los atributos de los objetos.

El constructor se define dentro de la propia clase, como si fuera otro método. El único detalle es que el constructor debe tener el mismo nombre que la clase. Atentos a PHP, que diferencia entre mayúsculas y minúsculas.

Para la clase Caja definida anteriormente, se podría declarar este constructor:

```
function Caja($alto=1,$ancho=1,$largo=1,$color="negro") {  
    $this->alto=$alto;  
    $this->ancho=$ancho;  
    $this->largo=$largo;  
    $this->color=$color;  
    $this->contenido="";  
}
```

En este constructor recibimos por parámetro todos los atributos que hay que definir en una caja. Es muy útil definir unos valores por defecto en los parámetros que recibe el constructor, igualando el parámetro a un valor dentro de la declaración de parámetros de la función constructora, pues así, aunque se llame al constructor sin proporcionar parámetros, se inicializará con los valores por defecto que se hayan definido.

Es importante señalar que en los constructores no se tienen por qué recibir todos los valores para inicializar el objeto. Hay algunos valores que pueden inicializarse a vacío o a cualquier otro valor fijo, como en este caso el contenido de la caja, que inicialmente hemos supuesto que estará vacía.

2.- MySQL.

Caso práctico

Juan y Carlos deciden comenzar revisando el servidor que van a utilizar, MySQL. Aunque van a utilizar un servidor que ya está en funcionamiento, deben comprender sus capacidades y las herramientas de las que disponen para poder gestionar tanto el servidor como los datos que almacena.

María conoce bien MySQL y les orienta sobre los pasos necesarios para instalarlo y configurarlo. Con su ayuda y con el permiso de **Esteban**, hacen una copia a algunos de los datos que necesitan, y los replican en un servidor local para poder trabajar con ellos. Por supuesto, se aseguran de no utilizar para las pruebas información sensible como la de los clientes o proveedores, que pueda ocasionarles problema legales.



MySQL es un sistema gestor de bases de datos (SGBD) relacionales. Es un programa de código abierto que se ofrece bajo licencia GNU GPL, aunque también ofrece una licencia comercial en caso de que quieras utilizarlo para desarrollar aplicaciones de código propietario. En las últimas versiones (a partir de la 5.1), se ofrecen, de hecho, varios productos distintos: uno de código libre (Community Edition), y otro u otros comerciales (Standard Edition, Enterprise Edition).

Incorpora múltiples motores de almacenamiento, cada uno con características propias: unos son más veloces, otros, aportan mayor seguridad o mejores capacidades de búsqueda. Cuando crees una base de datos, puedes elegir el motor en función de las características propias de la aplicación. Si no lo cambias, el motor que se utiliza por defecto se llama **MyISAM**, que es muy rápido pero a cambio no contempla integridad referencial (*característica de las bases de datos que permite crear relaciones válidas entre dos registros de la misma o de diferentes tablas, y definir las operaciones necesarias para mantener la validez de las relaciones cuando se borra o modifica alguno de los registros*) ni tablas transaccionales (*conjunto de operaciones sobre los datos que se han de realizar de forma conjunta, una sola vez, e independientemente del resto de manipulaciones sobre los datos. Toda transacción debe cumplir cuatro propiedades: atomicidad, consistencia, aislamiento y permanencia*). El motor **InnoDB** es un poco más lento pero sí soporta tanto integridad referencial como tablas transaccionales.

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 99 (*revisión del estándar ANSI SQL del año 1999, que agrega a la revisión anterior (SQL2 o SQL 92) disparadores, expresiones regulares, y algunas características de orientación a objetos*), y añade además algunas extensiones propias.

En las siguientes secciones darás un rápido repaso a lo que debes saber sobre la instalación, configuración y las herramientas de administración de MySQL. Si necesitas ampliar información, puedes consultar el manual en línea de MySQL.

<http://dev.mysql.com/doc/refman/5.0/es/index.html>

¿A qué hacen referencia las siglas PDO?



A un motor de almacenamiento utilizado por MySQL.



A una extensión de PHP que permite acceder a varios gestores de bases de datos.

los motores de almacenamiento de los que hablamos son MyISAM e InnoDB. Aunque no son los únicos que se pueden utilizar con MySQL sí son los más comunes.

2.1.- Instalación y configuración.

En la primera unidad ya viste cómo podías instalar en un único paso una plataforma **LAMP** para desarrollar aplicaciones web en **Ubuntu**. En Linux, la instalación de MySQL se divide básicamente en dos paquetes que puedes instalar de forma individual según tus necesidades:

- ✓ **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**. Son los programas cliente, necesarios para conectarse a un servidor MySQL. Solo necesitas instalarlos en aquel o aquellos equipos que se vayan a conectar al SGBD (en nuestro caso, las conexiones se realizarán normalmente desde el mismo equipo en el que se ejecuta el servidor).

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

```
sudo service mysql status // también start, stop, restart
```

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. Esto lo debes tener en cuenta para permitir el acceso a través del cortafuegos en configuraciones en red.

El fichero de configuración del servidor MySQL se llama **my.cnf** y se encuentra alojado en **/etc/mysql**. Su contenido se divide en secciones. Las opciones que contiene cada una de las secciones afectan al comportamiento de un módulo concreto. Entre las secciones disponibles destacan:

- ✓ **[client]**. Sus parámetros influyen sobre los distintos clientes que se conectan al servidor MySQL.
- ✓ **[mysqld]**. Contiene opciones relativas a la ejecución del servidor.

Además del fichero global de configuración, las opciones de ejecución se pueden aplicar por línea de comandos y obtener de otros orígenes distintos. Puedes consultar más información en el manual en línea de MySQL.

<http://dev.mysql.com/doc/refman/5.0/es/index.html>

Entre los parámetros que puedes configurar en el fichero my.cnf tienes:

- ✓ **port**. Indica el puerto TCP en el que escuchará el servidor y con el que se establecerán las conexiones.
- ✓ **user**. Nombre del usuario que se utilizará para ejecutar el servidor.
- ✓ **datadir**. Directorio del servidor en el que se almacenarán las bases de datos.

En la documentación de MySQL tienes también información sobre todas las opciones de configuración disponibles para ajustar su funcionamiento.

<http://dev.mysql.com/doc/refman/5.0/es/index.html>

2.2.- Herramientas de administración.

Existen muchas herramientas que permiten establecer una conexión con un servidor MySQL para realizar tareas de administración. Algunas herramientas se ejecutan en la línea de comandos, otras presentan un interface gráfico basado en web o propio del sistema operativo en que se ejecuten. Unas se incluyen con el propio servidor, y otras es necesario obtenerlas e instalarlas de forma independiente. Las hay que están orientadas a algún propósito concreto y también que permiten realizar varias funciones de administración.

Con el servidor MySQL se incluyen algunas herramientas de administración en línea de comandos, entre las que debes conocer:

- ✓ **mysql**. Permite conectarse a un servidor MySQL para ejecutar sentencias SQL.
- ✓ **mysqladmin**. Es un cliente específico para tareas de administración.
- ✓ **mysqlshow**. Muestra información sobre bases de datos y tablas.

En la documentación de MySQL tienes información sobre las distintas utilidades que incorpora.

<http://dev.mysql.com/doc/refman/5.0/es/client-side-scripts.html>

Estas herramientas comparten unas cuantas opciones relativas al establecimiento de la conexión con el servidor. Muchas de estas opciones tienen también una forma abreviada:

- ✓ `--user=nombre_usuario` (`-u nombre_usuario`). Indica un nombre de usuario con permisos para establecer la conexión. Si no se especifica se usará el nombre de usuario actual del sistema operativo.
- ✓ `--password=contraseña` (`-pcontraseña`). Contraseña asociada al nombre de usuario anterior. Si se utiliza la opción abreviada, debe figurar justo a continuación de la letra p, sin espacios intermedios. Si es necesario introducir una contraseña y no se indica ninguna, se pedirá para establecer la conexión.
- ✓ `--host=equipo_servidor` (`-h equipo_servidor`). Nombre del equipo con el que se establecerá la conexión. Si no se indica nada, se usará `"localhost"`.

Por ejemplo, para establecer una conexión al servidor local con la herramienta `mysql`, podemos hacer:

```
mysql -u root -p
```

Conviene no indicar nunca la contraseña en la misma línea de comandos. En caso de que la cuenta esté convenientemente protegida por una contraseña, es mejor utilizar solo la opción `-p` como en el ejemplo anterior. De esta forma, la herramienta solicita la introducción de la contraseña y ésta no queda almacenada en ningún registro como puede ser el historial de comandos del sistema.

De entre el resto de herramientas de administración independientes que podemos utilizar con MySQL, podemos destacar dos:

- ✓ **MySQL Workbench** es una herramienta genérica con interface gráfico nativo que permite administrar tanto el servidor como las bases de datos que éste gestiona. Ha sido desarrollada por los creadores de MySQL y se ofrece en dos ediciones, una de ellas de código abierto bajo licencia GPL.
<http://dev.mysql.com/doc/workbench/en/index.html>
- ✓ **phpMyAdmin** es una aplicación web muy popular para la administración de servidores MySQL. Presenta un interface web de administración programado en PHP bajo licencia GPL. Su objetivo principal es la administración de las bases de datos y la gestión de la información que maneja el servidor.

<http://www.phpmyadmin.net/>

Relaciona cada herramienta de administración con el tipo de interface que utiliza:

Herramienta.	Relación.	Tipo de interface.
MySQL Workbench.	3	1. Línea de comandos.
mysql.	1	2. Web.
phpMyAdmin.	2	3. Nativo.
mysqladmin.	4	4. Línea de comandos.

2.2.1.- *mysql y mysqladmin.*

La forma más habitual de utilizar la herramienta `mysql` es en modo interactivo. Una vez te conectas al servidor MySQL, te presenta una línea de órdenes. En esa línea de órdenes puedes introducir sentencias SQL, que se ejecutarán sobre la base de datos seleccionada, y algunos comandos

especiales. Las sentencias SQL deben terminar en el carácter ";". Entre los comandos especiales que puedes usar están:

- ✓ **connect**. Establece una conexión con un servidor MySQL.
- ✓ **use**. Permite seleccionar una base de datos.
- ✓ **exit** o **quit**. Termina la sesión interactiva con mysql.
- ✓ **help**. Muestra una pantalla de ayuda con la lista de comandos disponibles.

Por ejemplo, si cuando estás utilizando la herramienta quieres seleccionar la base de datos "dwes", debes hacer:

```
mysql> use dwes
```

Las sentencias SQL que teclees a partir de ese instante se ejecutarán sobre la base de datos "dwes".

Para comprobar la sintaxis de las sentencias SQL admitidas por MySQL, puedes consultar la documentación en línea.

<http://dev.mysql.com/doc/refman/5.0/es/sql-syntax.html>

También puedes usar mysql en modo de procesamiento por lotes (*ejecución de un conjunto de tareas repetitivas o de forma consecutiva, sin la supervisión directa del usuario*), para ejecutar sobre un servidor MySQL todas las sentencias almacenadas en un fichero de texto (normalmente con extensión .sql). Por ejemplo:

```
mysql -u root -pabc123. < crear_bd_dwes.sql
```

Utiliza las sentencias SQL que contiene el siguiente fichero para crear la estructura de la base de datos "dwes" en tu instalación de MySQL.

```
-- Creamos la base de datos
CREATE DATABASE 'dwes' DEFAULT CHARACTER SET utf8 COLLATE utf8_spanish_ci;
USE 'dwes';

-- Creamos las tablas
CREATE TABLE 'dwes'.'tienda' (
  'cod' INT NOT NULL AUTO INCREMENT PRIMARY KEY ,
  'nombre' VARCHAR( 100 ) NOT NULL ,
  'tlf' VARCHAR( 13 ) NULL
) ENGINE = INNODB;

CREATE TABLE 'dwes'.'producto' (
  'cod' VARCHAR( 12 ) NOT NULL ,
  'nombre' VARCHAR( 200 ) NULL ,
  'nombre_corto' VARCHAR( 50 ) NOT NULL ,
  'descripcion' TEXT NULL ,
  'PVP' DECIMAL( 10, 2 ) NOT NULL ,
  'familia' VARCHAR( 6 ) NOT NULL ,
  PRIMARY KEY ( 'cod' ) ,
  INDEX ( 'familia' ) ,
  UNIQUE ( 'nombre_corto' )
) ENGINE = INNODB;

CREATE TABLE 'dwes'.'familia' (
  'cod' VARCHAR( 6 ) NOT NULL ,
  'nombre' VARCHAR( 200 ) NOT NULL ,
  PRIMARY KEY ( 'cod' )
) ENGINE = INNODB;

CREATE TABLE 'dwes'.'stock' (
  'producto' VARCHAR( 12 ) NOT NULL ,
  'tienda' INT NOT NULL ,
  'unidades' INT NOT NULL ,
  PRIMARY KEY ( 'producto' , 'tienda' )
) ENGINE = INNODB;

-- Creamos las claves foráneas
ALTER TABLE 'producto'
ADD CONSTRAINT 'producto_ibfk_1'
FOREIGN KEY ( 'familia' ) REFERENCES 'familia' ( 'cod' )
ON UPDATE CASCADE;
```

```
ALTER TABLE `stock`
ADD CONSTRAINT `stock_ibfk_2`
FOREIGN KEY (`tienda`) REFERENCES `tienda` (`cod`)
ON UPDATE CASCADE,
ADD CONSTRAINT `stock_ibfk_1`
FOREIGN KEY (`producto`) REFERENCES `producto` (`cod`)
ON UPDATE CASCADE;

CREATE USER `dwes`
IDENTIFIED BY 'abc123.';

GRANT ALL ON `dwes`.*
TO `dwes`;
```

Utilizando la herramienta mysql ejecuta:

```
mysql -u root -p < dwes.sql
```

mysqladmin es una herramienta no interactiva orientada a tareas de administración del propio servidor. Las tareas concretas de administración a llevar a cabo, se indican mediante parámetros en la línea de comandos. Entre las tareas que puedes llevar a cabo con esta utilidad se encuentran:

- ✓ crear y eliminar bases de datos.
- ✓ mostrar la configuración y el estado del servidor.
- ✓ cambiar contraseñas.
- ✓ detener un servidor.

Por ejemplo, si quieres mostrar información sobre el estado actual del servidor local, puedes utilizar el comando **status**:

```
mysqladmin -u root -pabc123. status
```

En la documentación de MySQL tienes información sobre todos los comandos que admite **mysqladmin** y su significado.

<http://dev.mysql.com/doc/refman/5.0/es/mysqladmin.html>

Si quieres saber si en una tabla de una base de datos existe o no un registro, ¿qué herramienta en línea de comandos puedes usar?

- ☐ mysqladmin.
- ☒ **mysql.**

La herramienta **mysqladmin** la puedes utilizar para realizar tareas administrativas, pero no para ejecutar consultas sobre el contenido de las bases de datos

2.2.2.- phpMyAdmin.

Al contrario que las dos herramientas anteriores, **phpMyAdmin** no se instala con el servidor MySQL. Debes instalarlo de forma individual, en el caso de Ubuntu utilizando el gestor de paquetes:

```
sudo apt-get install phpmyadmin
```

El proceso de instalación es sencillo. Simplemente te pregunta por el servidor web a utilizar (escoger apache2), y después debes dejar que configure una nueva base de datos propia en el servidor. Una vez instalada la aplicación, podrás acceder vía web con un navegador utilizando la URL "<http://localhost/phpmyadmin/>".



Para poder entrar, debes indicar un nombre de usuario y contraseña válidos. Si realizaste el ejercicio anterior, se habrá creado en tu servidor un **usuario** "**dwes**" con **contraseña** "**abc123.**" con **permisos**

para la base de datos "`dwes`". Si utilizas ese usuario para entrar en la aplicación, ésta te permitirá gestionar la base de datos "`dwes`".

El interface de la aplicación se compone de un panel de navegación a la izquierda, donde se muestran las bases de datos, y un panel principal con un menú en la parte superior y una serie de acciones e información en la parte central. Si seleccionas la base de datos "`dwes`", la información en pantalla cambia.



Utilizando los menús de la parte superior, puedes:

- ✓ Ver y modificar la **estructura** de la base de datos.
- ✓ Ejecutar sentencias **SQL**.
- ✓ **Buscar** información en toda la base de datos o en parte de la misma.
- ✓ **Generar una consulta** utilizando un asistente.
- ✓ **Exportar e importar** información, tanto de la estructura como de los datos.
- ✓ **Diseñar** las relaciones existentes entre las tablas.
- ✓ Otras **operaciones**, como hacer una copia de la base de datos.

Si seleccionas una tabla en lugar de la base de datos, podrás efectuar a ese nivel operaciones similares a las anteriores. En la siguiente presentación sobre `phpMyAdmin` tienes información sobre el manejo básico de la aplicación.

En la página web de la aplicación tienes documentación sobre su configuración y utilización.
http://www.phpmyadmin.net/localized_docs/es/Documentation.html

Utiliza `phpMyAdmin` para ejecutar las consultas del siguiente fichero, que rellenan con datos las tablas de la base de datos "`dwes`". Esta información la utilizaremos en los próximos ejercicios.

```
USE `dwes`;

INSERT INTO `tienda` (`cod`, `nombre`, `tlf`) VALUES
(1, 'CENTRAL', '600100100'),
(2, 'SUCURSAL1', '600100200'),
(3, 'SUCURSAL2', NULL);

INSERT INTO `familia` (`cod`, `nombre`) VALUES
('CAMARA', 'Cámaras digitales'),
('CONSOL', 'Consolas'),
('EBOOK', 'Libros electrónicos'),
('IMPRES', 'Impresoras'),
('MEMFLA', 'Memorias flash'),
('MP3', 'Reproductores MP3'),
('MULTIF', 'Equipos multifunción'),
('NETBOK', 'Netbooks'),
('ORDENA', 'Ordenadores'),
('PORTAT', 'Ordenadores portátiles'),
('ROUTER', 'Routers'),
('SAI', 'Sistemas de alimentación ininterrumpida'),
('SOFTWA', 'Software'),
('TV', 'Televisores'),
('VIDEOC', 'Videocámaras');

INSERT INTO `producto` (`cod`, `nombre`, `nombre_corto`, `descripcion`, `PVP`, `familia`) VALUES
('3DSNG', NULL, 'Nintendo 3DS negro', 'Consola portátil de Nintendo que permitirá disfrutar de efectos 3D sin necesidad de gafas especiales, e incluirá retrocompatibilidad con el software de DS y de DSi.', '270.00', 'CONSOL'),
('ACERAX3950', NULL, 'Acer AX3950 I5-650 4GB 1TB W7HP', 'Características:\n\nSistema Operativo : Windows® 7 Home Premium Original\n\nProcesador / Chipset:\nNúmero de Ranuras PCI: 1\n\nFabricante de Procesador: Intel\n\nTipo de Procesador: Core i5\n\nModelo de Procesador: i5-650\n\nNúcleo de Procesador: Dual-core\n\nVelocidad de Procesador: 3,20 GHz\n\nCaché: 4 MB\n\nVelocidad de Bus: No aplicable\n\nVelocidad HyperTransport: No aplicable\n\nInterconexión QuickPathNo aplicable\n\nProcesamiento de 64 bits: Sí\n\nHyper-Threading: Sí\n\nFabricante de Chipset: Intel\n\nModelo de Chipset: H57 Express\n\nMemoria:\nMemoria Estándar: 4 GB\n\nMemoria Máxima: 8 GB\n\nTecnología de la
```

```

Memoria: DDR3 SDRAM\r\nEstándar de Memoria: DDR3-1333/PC3-10600\r\nNúmero de Ranuras de
Memoria (Total): 4\r\nLector de tarjeta memoria: Sí\r\nSoporte de Tarjeta de Memoria: Tarjeta
CompactFlash (CF)\r\nSoporte de Tarjeta de Memoria: MultiMediaCard (MMC)\r\nSoporte de Tarjeta
de Memoria: Micro Drive\r\nSoporte de Tarjeta de Memoria: Memory Stick PRO\r\nSoporte de
Tarjeta de Memoria: Memory Stick\r\nSoporte de Tarjeta de Memoria: CF+\r\nSoporte de Tarjeta
de Memoria: Tarjeta Secure Digital (SD)\r\n\r\nStorage\r\nCapacidad Total del Disco Duro: 1
TB\r\nRPM de Disco Duro: 5400\r\nTipo de Unidad Óptica: Grabadora DVD\r\nCompatibilidad de
Dispositivo Óptico: DVD-RAM/±R/±RW\r\nCompatibilidad de Medios de Doble Capa: Sí', '410.00',
'ORDENA'),
('ARCLPMP32GBN', NULL, 'Archos Clipper MP3 2GB negro', 'Características:\r\n\r\nAlmacenamiento
Interno Disponible en 2 GB*\r\nCompatibilidad Windows o Mac y Linux (con soporte para
almacenamiento masivo)\r\nInterfaz para ordenador USB 2.0 de alta velocidad\r\nBatería 2 11
horas música\r\nReproducción Música 3 MP3\r\nMedidas Dimensiones: 52mm x 27mm x 12mm, Peso: 14
Gr', '26.70', 'MP3'),
('BRAVIA2BX400', NULL, 'Sony Bravia 32IN FULLHD KDL-32BX400', 'Características:\r\n\r\nFull
HD: Vea deportes películas y juegos con magníficos detalles en alta resolución gracias a la
resolución 1920x1080.\r\n\r\nHDMI@: 4 entradas (3 en la parte posterior, 1 en el
lateral)\r\n\r\nUSB Media Player: Disfrute de películas, fotos y música en el
televisor.\r\n\r\nSintonizador de TV HD MPEG-4 AVC integrado: olvídense del codificador y
acceda a servicios de TV que incluyen canales HD con el sintonizador DVB-T y DVB-C integrado
con decodificador MPEG4 AVC (dependiendo del país y sólo con operadores
compatibles)\r\n\r\nSensor de luz: ajusta automáticamente el brillo según el nivel de la
iluminación ambiental para que pueda disfrutar de una calidad de imagen óptima sin consumo
innecesario de energía.\r\n\r\nBRAVIA Sync: controle su sistema de ocio doméstico entero con
un mismo mando a distancia universal que le permite reproducir contenidos o ajustar la
configuración de los dispositivos compatibles con un solo botón.\r\n\r\nBRAVIA ENGINE 2:
experimente colores y detalles de imagen increíblemente nítidos y definidos. \r\n\r\nLive
Colour™: seleccione entre cuatro modos: desactivado, bajo, medio y alto, para ajustar el color
y obtener imágenes vivas y una calidad óptima. \r\n\r\n24p True Cinema™: reproduzca una
auténtica experiencia cinematográfica y disfrute de películas exactamente como el director las
concibió a 24 fotogramas por segundo.', '356.90', 'TV'),
('EEEPC1005PXD', NULL, 'Asus EEEPC 1005PXD N455 1 250 BL', 'Características:\r\n\r\nProcesador:
1660 MHz, N455, Intel Atom, 0.5 MB. \r\n\r\nMemoria: 1024 MB, 2 GB, DDR3, SO-DIMM, 1 x 1024 MB.
\r\n\r\nAccionamiento de disco: 2.5 ", 250 GB, 5400 RPM, \r\n\r\nSerial ATA, Serial ATA II, 250 GB.
\r\n\r\nMedios de almacenaje: MMC, SD, SDHC. \r\n\r\nExhibición: 10.1 ", 1024 x 600 Píxeles, LCD TFT.
\r\n\r\nCámara fotográfica: 0.3 MP. \r\n\r\nRed: 802.11 b/g/n, 10, 100 Mbit/s, \r\n\r\nFast Ethernet.
\r\n\r\nAudio: HD. \r\n\r\nSistema operativo/software: Windows 7 Starter. \r\n\r\nColor: Blanco.
\r\n\r\nControl de energía: 8 MB/s, Litio-Ion, 6 piezas, 2200 mAh, 48 W. \r\n\r\nPeso y dimensiones:
1270 g, 178 mm, 262 mm, 25.9 mm, 36.5 mm', '245.40', 'NETBOK'),
('HPMIN1103120', NULL, 'HP Mini 110-3120 10.1LED N455 1GB 250GB W7S negro',
'Características:\r\n\r\nSistema operativo instalado \r\n\r\nWindows® 7 Starter original 32 bits
\r\n\r\n\r\nProcesador \r\n\r\nProcesador Intel® Atom™ N4551,66 GHz, Cache de nivel 2, 512 KB
\r\n\r\n\r\nChipset NM10 Intel® + ICH8m \r\n\r\n\r\nMemoria \r\n\r\nDDR2 de 1 GB (1 x 1024 MB) \r\n\r\nMemoria
máxima \r\n\r\nAdmite un máximo de 2 GB de memoria DDR2 \r\n\r\n\r\nRanuras de memoria \r\n\r\n1 ranura de
memoria accesible de usuario \r\n\r\n\r\nUnidades internas \r\n\r\nDisco duro SATA de 250 GB (5400
rpm) \r\n\r\n\r\nGráficos \r\n\r\nTamaño de pantalla (diagonal) \r\n\r\nPantalla WSVGA LED HP
Antirreflejos de 25,6 cm (10,1") en diagonal \r\n\r\n\r\nResolución de la pantalla \r\n\r\n1024 x 600
', '270.00', 'NETBOK'),
('IXUS115HS AZ', NULL, 'Canon Ixus 115HS azul', 'Características:\r\n\r\nHS System (12,1 MP)
\r\n\r\nZoom 4x, 28 mm. IS Óptico \r\n\r\nCuerpo metálico estilizado \r\n\r\nPantalla LCD PureColor II G
de 7,6 cm (3,0") \r\n\r\nFull HD. IS Dinámico. HDMI \r\n\r\nModo Smart Auto (32 escenas) ', '196.70',
'CÁMARA'),
('KSTDT101G2', NULL, 'Kingston DataTraveler 16GB DT101G2 USB2.0 negro',
'Características:\r\n\r\nCapacidades - 16GB\r\n\r\nDimensiones - 2.19" x 0.68" x 0.36" (55.65mm x
17.3mm x 9.05mm)\r\n\r\nTemperatura de Operación - 0° hasta 60° C / 32° hasta 140°
F\r\n\r\nTemperatura de Almacenamiento - -20° hasta 85° C / -4° hasta 185° F\r\n\r\nSimple - Solo debe
conectarlo a un puerto USB y está listo para ser utilizado\r\n\r\nPractico - Su diseño sin tapa
giratorio, protege el conector USB; sin tapa que perder\r\n\r\nGarantizado - Cinco años de
garantía', '19.20', 'MEMFLA'),
('KSTDTG332GBR', NULL, 'Kingston DataTraveler G3 32GB rojo', 'Características:\r\n\r\n\r\nTipo de
producto Unidad flash USB\r\n\r\nCapacidad almacenamiento 32GB\r\n\r\nAnchura 58.3 mm\r\n\r\nProfundidad
23.6 mm\r\n\r\nAltura 9.0 mm\r\n\r\nPeso 12 g\r\n\r\nColor incluido RED\r\n\r\nTipo de interfaz USB', '40.00',
'MEMFLA'),
('KSTMSDHC8GB', NULL, 'Kingston MicroSDHC 8GB', 'Kingston tarjeta de memoria flash 8 GB
microSDHC\r\n\r\nÍndice de velocidad Class 4\r\n\r\nCapacidad almacenamiento 8 GB\r\n\r\nFactor de
forma MicroSDHC\r\n\r\nAdaptador de memoria incluido Adaptador microSDHC a SD\r\n\r\nGarantía del
fabricante Garantía limitada de por vida', '10.20', 'MEMFLA'),
('LEGRIAFS306', NULL, 'Canon Legria FS306 plata', 'Características:\r\n\r\n\r\nGrabación en
tarjeta de memoria SD/SDHC \r\n\r\nLa cámara de vídeo digital de Canon más pequeña nunca vista
\r\n\r\nInstantánea de Vídeo (Video Snapshot) \r\n\r\nZoom Avanzado de 41x \r\n\r\nGrabación Dual (Dual
Shot) \r\n\r\nEstabilizador de la Imagen con Modo Dinámico \r\n\r\nPre grabación (Pre REC) \r\n\r\nSistema
16:9 de alta resolución realmente panorámico \r\n\r\nBatería inteligente y Carga Rápida
\r\n\r\nCompatible con grabador de DVD DW-100 \r\n\r\nSISTEMA DE VÍDEO\r\n\r\nSoporte de grabación:
Tarjeta de memoria extraíble (SD/SDHC)\r\n\r\nTiempo máximo de grabación: Variable, dependiendo
del tamaño de la tarjeta de memoria.\r\n\r\nTarjeta SDHC de 32 GB: 20 horas 50 minutos', '175.00',
'VIDEOC'),
('LGM237WDP', NULL, 'LG TDT HD 23 M237WDP-PC FULL HD',
'Características:\r\n\r\n\r\nGeneral\r\n\r\nTamaño (pulgadas): 23\r\n\r\nPantalla LCD: Sí\r\n\r\nFormato:

```

```
69\r\nResolución: 1920 x 1080\r\nFull HD: Si\r\nnBrillo (cd/m2): 300\r\nnRatio Contraste: 50.000:1\r\nnTiempo Respuesta (ms): 5\r\nnÁngulo Visión (°): 170\r\nnNúmero Colores (Millones): 16.7\r\nn\r\nnTV\r\nnTDT: TDT HD\r\nnConexiones\r\nnD-Sub: Si\r\nnDVI-D: Si\r\nnHDMI: Si\r\nEuroconector: Si\r\nSalida auriculares: Si\r\nEntrada audio: Si\r\nUSB Servicio: Si\r\nRS-232C Servicio: Si\r\nPCMCIA: Si\r\nSalida óptico: Si', '186.00', 'TV'), ('LJPROP1102W', NULL, 'HP Laserjet Pro Wifi P1102W', 'Impesora laserjet P1102W es facil de instalar y usar, ademas de que te ayudara a ahorrar energia y recursos.\r\nOlviadte de los cables y disfura de la libertad que te proporcina su tecnologia WIFI, imprime facilmente desde cualquier de tu oficina.\r\n\r\nFormato máximo aceptado A4 A2 No\r\nA3 NoA4 Si\r\nA5 SiA6 Si\r\nB5 SiB6 Si\r\nSobres C5 (162 x 229 mm) SiSobres C6 (114 x 162 mm) Si', '99.90', 'IMPRES'), ('OPTIOLS1100', NULL, 'Pentax Optio LS1100', 'La LS1100 con funda de transporte y tarjeta de memoria de 2GB incluidas \r\nes la compacta digital que te llevarás a todas partes.\r\nEsta cámara diseñada por Pentax incorpora un sensor CCD de 14,1 megapíxeles y un objetivo gran angular de 28 mm.\r\n', '104.80', 'CAMARA'), ('PAYPRE62GB', NULL, 'Lector ebooks Papyre6 con SD2GB + 500 ebooks', 'Marca Papyre \r\nModelo Papyre 6.1 \r\nUso Lector de libros electrónicos \r\nTecnología e-ink (tinta electrónica, Vizplez) \r\nCPU Samsung Am9 200MHz \r\nMemoria - Interna: 512MB \r\nExterna: SD/SDHC (hasta 32GB) \r\nFormatos PDF, RTF, TXT, DOC, HTML, MP3, CHM, ZIP, FB2, Formatos de imagen \r\nPantalla 6" (600x800px), blanco y negro, 4 niveles de grises ', '205.50', 'EBOOK'), ('PBELLI810323', NULL, 'Packard Bell I8103 23 I3-550 4G 640GB NVIDIAIG210', 'Características:\r\n\r\nCPU CHIPSET\r\n\r\nProcesador : Ci3-550\r\nNorthBridge : Intel H57\r\n\r\nMEMORIA\r\nMemoria Rma : Ddr3 4096 MB\r\n\r\nDISPOSITIVOS DE ALMACENAMIENTO\r\nDisco Duro: 640Gb 7200 rpm\r\nÓptico : Slot Load siper multi Dvdrw\r\nLector de Tarjetas: 4 in 1 (XD, SD, HC, MS, MS PRO, MMC)\r\n\r\nndispositivos gráficos\r\nMonitor: 23 fHD\r\nTarjeta Gráfica: Nvidia G210M D3 512Mb\r\nMemoria Máxima: Hasta 1918Mb\r\n\r\nAUDIO\r\nAudio Out: 5.1 Audio Out\r\nAudio In: 1 jack\r\nHeasphone in: 1x jack\r\nAltavoces: Stereo\r\n\r\nACCESORIOS\r\nTeclado y ratón inalámbrico\r\nMando a distancia: EMEA Win7 WMC\r\n\r\nCOMUNICACIONES\r\nWireless: 802.11 b/g/n mini card \r\nTarjeta de Red: 10/100/1000 Mbps\r\nBluetooth: Bluethoot\r\nWebcam: 1mpixel Hd (1280x720)\r\nTv tuner: mCARD/SW/ DVB-T\r\n\r\nMONITOR\r\nTamaño: 23"\r\ncontraste: 1000:1\r\nTiempo de respuesta: 5MS\r\nResolución: 1920 X 1080\r\n\r\nPUERTOS E/S\r\nUsb 2.0 : 6\r\nMini Pc-e : 2\r\nEsata: 1\r\n\r\nSISTEMA OPERATIVO\r\nO.S: Microsoft Windows 7 Premium', '761.80', 'ORDENA'), ('PIXMAIP4850', NULL, 'Canon Pixma IP4850', 'Características:\r\n\r\nTipo: chorro de tinta cartuchos independientes\r\nConexión: Hi-Speed USB\r\nPuerto de impresión directa desde camaras\r\nResolución máxima: 9600x2400 ppp\r\nVelocidad impresión: 11 ipm (negro) / 9.3 ipm (color)\r\nTamaño máximo papel: A4\r\nBandeja entrada: 150 hojas\r\nDimensiones: 43.1 cm x 29.7 cm x 15.3 cm', '97.30', 'IMPRES'), ('PIXMAMP252', NULL, 'Canon Pixma MP252', 'Características:\r\n\r\nFunciones: Impresora, Escáner , Copiadora\r\nConexión: USB 2.0\r\nDimensiones:444 x 331 x 155 mm\r\nPeso: 5,8 Kg\r\n\r\nIMPRESORA\r\nResolución máxima: 4800 x 1200 ppp\r\nVelocidad de impresión:\r\nNegro/color: 7,0 ipm / 4,8 ipm\r\nTamaño máximo papel: A4\r\n\r\nCARTUCHOS\r\nNegro: PG-510 / PG-512\r\nColor: CL-511 / CL-513\r\n\r\nESCANER\r\nResolución máxima: 600 x 1200 ppp (digital: 19200 x 19200)\r\nProfundidad de color: 48/24 bits\r\nArea máxima de escaneado: A4\r\n\r\nCOPIA\r\nTiempo salida 1ª copia: aprox 39 seg.', '41.60', 'MULTIF'), ('PS3320GB', NULL, 'PS3 con disco duro de 320GB', 'Este Pack Incluye:\r\n- La consola Playstation 3 Slim Negra 320GB\r\n- El juego Killzone 3\r\n', '380.00', 'CONSOL'), ('PWSHA3100PT', NULL, 'Canon Powershot A3100 plata', 'La cámara PowerShot A3100 IS, inteligente y compacta, presenta la calidad de imagen de Canon en un cuerpo\r\ncompacto y ligero para capturar fotografías sin esfuerzo; es tan fácil como apuntar y disparar.\r\nCaracterísticas:\r\nnl2,1 MP \r\nZoom óptico 4x con IS \r\nPantalla LCD de 6,7 cm (2,7") ', '101.40', 'CAMARA'), ('SMSCGLX3175', NULL, 'Samsung CLX3175', 'Características:\r\n\r\nFunción: Impresión color, copidora, escáner\r\nImpresión \r\nVelocidad (Mono)Hasta 16 ppm en A4 (17 ppm en Carta)\r\n\r\nVelocidad (Color)Hasta 4 ppm en A4 (4 ppm en Carta)\r\n\r\nSalida de la Primer Página (Mono)Menos de 14 segundos (Desde el Modo Listo)\r\n\r\nResoluciónHasta 2400 x 600 dpi de salida efectiva\r\n\r\nSalida de la Primer Página (Color)Menos de 26 segundos (Dese el Modo Listo)\r\n\r\nDuplexManual\r\nEmulaciónSPL-C (Lenguaje de color de impresión SAMSUNG)\r\n\r\nCopiado \r\nSalida de la Primer Página (Mono)18 segundos\r\n\r\nMulticopiadol ~ 99\r\n\r\nZoom25 ~ 400 %\r\n\r\nFunciones de CopiadoCopia ID, Clonar Copia, Copia N-UP, Copiar Poster\r\n\r\nResoluciónTexto, Texto / Foto, Modo Revista: hasta 600 x 600 ppp, Modo Foto: Hasta 1200 x 1200 ppp\r\n\r\nVelocidad (Mono)Hasta 17 ppm en Carta (16 ppm en A4)\r\n\r\nVelocidad (Color)Hasta 4 ppm en Carta (4 ppm en A4 )\r\n\r\nSalida de la Primer Página (Color)45 segundos\r\n\r\nEscaneo \r\nCompatibilidadNorma TWAIN, Norma WIA (Windows2003 / XP / Vista)\r\n\r\nMétodoEscáner plano color\r\n\r\nResolución (Óptica)1200 x 1200 dpi\r\n\r\nResolución (Mejorada)4800 x 4800 dpi\r\n\r\nEscaneo a Escanear a USB / Carpeta', '190.00', 'MULTIF'), ('SMSN150101LD', NULL, 'Samsung N150 10.1LED N450 1GB 250GB BAT6 BT W7 R', 'Características:\r\n\r\nSistema Operativo Genuine Windows® 7 Starter \r\n\r\nProcesador Intel® ATOM Processor N450 (1.66GHz, 667MHz, 512KB) \r\n\r\nChipset Intel® NM10\r\n\r\nMemoria del Sistema 1GB (DDR2 / 1GB x 1) Ranura de Memoria 1 x SODIMM \r\n\r\nPantalla LCD 10.1" WSVGA (1024 x 600), Non-Gloss, LED Back Light Gráficos \r\n\r\nProcesador Gráfico Intel® GMA 3150 DVMT \r\nMemoria Gráfica Shared Memory (Int. Graphip) \r\n\r\nMultimedia \r\nSonido HD (High Definition) Audio \r\nCaracterísticas de Sonido SRS 3D Sound Effect \r\nAltavoces 3W Stereo Speakers (1.5W x 2) \r\nCámara Integrada Web Camera \r\n\r\nAlmacenamiento \r\nDisco duro 250GB SATA (5400 rpm S-ATA) \r\n\r\nConectividad\r\nWiired Ethernet LAN (RJ45) 10/100 LAN \r\nWireless LAN 802.11 b/g/N\r\n\r\nBluetooth Bluetooth 3.0 High Speed \r\n\r\nI/O Port \r\nVGA \r\nHeadphone-out \r\nMic-in\r\nInternal Mic\r\nUSB (Chargeable USB included) 3 x USB
```

```

2.0 \r\nMulti Card Slot 4-in-1 (SD, SDHC, SDXC, MMC)\r\nDC-in (Power Port)\r\n\r\nTipo de
Teclado 84 keys \r\nTouch Pad, Touch Screen Touch Pad (Scroll Scope, Flat Type)
\r\n\r\nSeguridad\r\nRecovery Samsung Recovery Solution \r\nVirus McAfee Virus Scan (trial
version) \r\nSeguridad BIOS Boot Up Password / HDD Password \r\nBloqueo Kensington Lock Port
\r\n\r\nBatería \r\nAdaptador 40 Watt Batería \r\n6 Cell Dimensiones ', '260.60', 'NETBOK'),
('SMSSMXC200PB', NULL, 'Samsung SMX-C200PB EDC ZOOM 10X', 'Características:\r\n\r\nSensor de
Imagen Tipo 1 / 6" 800K pixel CCD\r\n\r\nLente Zoom Óptico 10 x optico\r\n\r\nCaracterísticas
Grabación Vídeo Estabilizador de Imagen Hiper estabilizador de imagen digital\r\n\r\nInterfaz
Tarjeta de Memoria Ranura de Tarjeta SDHC / SD', '127.20', 'VIDEOC'),
('STYLUSSX515W', NULL, 'Epson Stylus SX515W', 'Características:\r\n\r\nResolución máxima5760 x
1440 DPI\r\nVelocidad de la impresión\r\nVelocidad de impresión (negro, calidad normal, A4)36
ppm\r\nVelocidad de impresión (color, calidad normal, A4)36 ppm\r\n\r\nTecnología de la
impresión\r\nTecnología de impresión inyección de tinta\r\nNúmero de cartuchos de impresión4
piezas\r\nCabeza de impresoraMicro Piezo\r\n\r\nExploración\r\nResolución máxima
de escaneado2400 x 2400 DPI\r\nEscaner color: si\r\nTipo de digitalización Escáner
plano\r\nEscaneaer integrado: si\r\nTecnología de exploración CIS\r\nWLAN, conexión: si',
'77.50', 'MULTIF'),
('TSSD16GBC10J', NULL, 'Toshiba SD16GB Class10 Jewel Case', 'Características:\r\n\r\nDensidad:
16 GB\r\nPINs de conexión: 9 pins\r\nInterfaz: Tarjeta de memoria SD standard
compatible\r\nVelocidad de Escritura: 20 MBytes/s* \r\nVelocidad de Lectura: 20
MBytes/s*\r\nDimensiones: 32.0 mm (L) x 24.0 mm (W) x 2.1 mm (H)\r\nPeso: 2g\r\nTemperatura: -
25°C a +85°C (Recomendada)\r\nHumedad: 30% to 80% RH (sin condensación)', '32.60', 'MEMFLA'),
('ZENMP48GB300', NULL, 'Creative Zen MP4 8GB Style 300', 'Características:\r\n\r\n8 GB de
capacidad\r\nAutonomía: 32 horas con archivos MP3 a 128 kbps\r\nPantalla TFT de 1,8 pulgadas y
64.000 colores\r\nFormatos de audio compatibles: MP3, WMA (DRM9), formato Audible
4\r\n\r\nFormatos de foto compatibles: JPEG (BMP, TIFF, GIF y PNG\r\n\r\nFormatos de vídeo
compatibles: AVI transcodificado (Motion JPEG)\r\n\r\nEcuilizador de 5 bandas con 8
preajustes\r\n\r\nMicrófono integrado para grabar voz\r\n\r\nAltavoz y radio FM incorporada', '58.90',
'MP3');

```

```

INSERT INTO `stock` (`producto`, `tienda`, `unidades`) VALUES
('3DSNG', 1, 2),
('3DSNG', 2, 1),
('ACERAX3950', 1, 1),
('ARCLPMP32GBN', 2, 1),
('ARCLPMP32GBN', 3, 2),
('BRAVIA2BX400', 3, 1),
('EEEEPC1005PXD', 1, 2),
('EEEEPC1005PXD', 2, 1),
('HPMIN1103120', 2, 1),
('HPMIN1103120', 3, 2),
('IXUS115HSAZ', 2, 2),
('KSTDT101G2', 3, 1),
('KSTDTG332GBR', 2, 2),
('KSTMSDHC8GB', 1, 1),
('KSTMSDHC8GB', 2, 2),
('KSTMSDHC8GB', 3, 2),
('LEGRIAFS306', 2, 1),
('LGM237WDP', 1, 1),
('LJPROP1102W', 2, 2),
('OPTIOLS1100', 1, 3),
('OPTIOLS1100', 2, 1),
('PAPYRE62GB', 1, 2),
('PAPYRE62GB', 3, 1),
('PBELLI810323', 2, 1),
('PIXMAIP4850', 2, 1),
('PIXMAIP4850', 3, 2),
('PIXMAMP252', 2, 1),
('PS3320GB', 1, 1),
('PWSHTA3100PT', 2, 2),
('PWSHTA3100PT', 3, 2),
('SMSGCLX3175', 2, 1),
('SMSN150101LD', 3, 1),
('SMSSMXC200PB', 2, 1),
('STYLUSSX515W', 1, 1),
('TSSD16GBC10J', 3, 2),
('ZENMP48GB300', 1, 3),
('ZENMP48GB300', 2, 2),
('ZENMP48GB300', 3, 2);

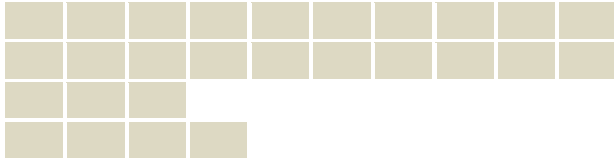
```

Selecciona en el panel de la izquierda la base de datos, y utilizando el menú SQL pega todo el texto del fichero (las consultas a ejecutar) y pulsa el botón "Continuar" para ejecutarlas.

Intenta resolver la siguiente actividad relativa a las bases de datos MySQL.



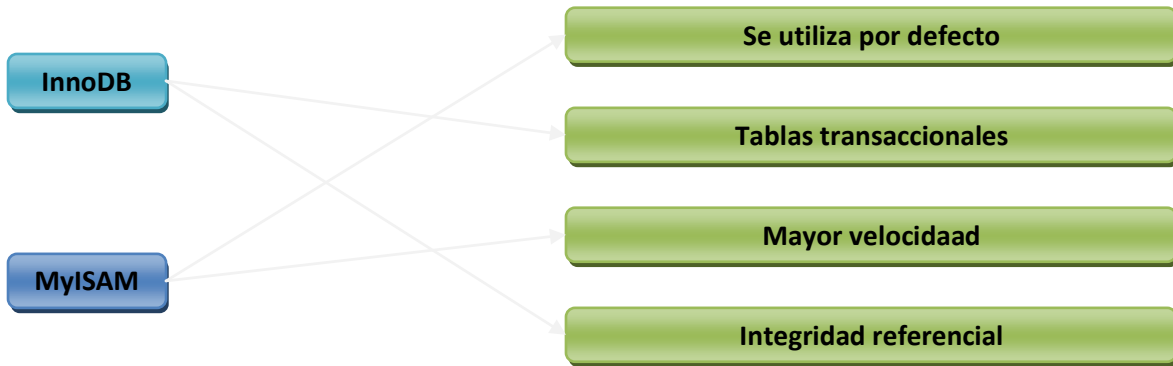
1. Permite ejecutar sentencias SQL.
2. Muestra información sobre bases de datos y tablas.



3. En línea de comandos, específica para tareas de administración.
4. Programada en lenguaje PHP.
5. Comando de mysql para seleccionar una base de datos.
6. Usuario administrador de MySQL, por defecto.

1 - MySQL 2 - mysqlshow 3 - mysqladmin 4 - phpmyadmin 5 - use 6 - root

Relaciona los motores de almacenamiento de MySQL con sus características principales



3.- Utilización de bases de datos MySQL en PHP.

Caso práctico

Entre **María, Juan y Carlos**, han creado una pequeña base de datos con cuatro tablas y unas decenas de registros que usarán en las pruebas de la nueva aplicación web.

Juan, que ha tenido cierta experiencia programando aplicaciones en PHP, se da cuenta que el lenguaje ha evolucionado mucho en los últimos tiempos. Y uno de los aspectos que más ha evolucionado es precisamente el que concierne al acceso a bases de datos MySQL.

En las aplicaciones que había realizado hace ya algunos años, siempre había utilizado la misma extensión. Y ahora, por lo que ha estado viendo, existen otras maneras más eficientes o más genéricas de llevar a cabo esa tarea.

Para estar seguro, **busca consejo en algunos programadores amigos** y llega a una conclusión: **tendrá que escoger entre una extensión nativa, MySQLi, y PDO**. Revisa la documentación sobre ambas y realiza un pequeño estudio comparativo. Además, diseña unas pruebas para llevar a cabo con la **ayuda de Carlos** y poder tomar una decisión. Siempre es mejor asegurarse antes de empezar, aunque eso implique alargar algo más los plazos.

Como ya viste, existen dos formas de comunicarse con una base de datos desde PHP: utilizar una extensión nativa programada para un SGBD concreto, o utilizar una extensión que soporte varios tipos de bases de datos. Tradicionalmente las conexiones se establecían utilizando la extensión nativa **mysql**. Esta extensión se mantiene en la actualidad para dar soporte a las aplicaciones ya existentes que la utilizan, pero no se recomienda utilizarla para desarrollar nuevos programas. Lo más habitual es elegir entre **MySQLi** (extensión nativa) y **PDO**.

Con cualquiera de ambas extensiones, podrás realizar 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.

PDO y MySQLi (y también la antigua extensión mysql) utilizan un **driver de bajo nivel** para comunicarse con el servidor MySQL. Hasta hace poco el único driver disponible para realizar esta función era **libmysql**, que no estaba optimizado para ser utilizado desde PHP. A partir de la versión 5.3, PHP viene preparado para utilizar también un nuevo driver mejorado para realizar esta función, el Driver Nativo de MySQL, **mysqlnd**.

3.1.- Extensión MySQLi.

Esta extensión se desarrolló para aprovechar las ventajas que ofrecen las versiones 4.1.3 y posteriores de MySQL, y 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. Por ejemplo, para establecer una conexión con un servidor MySQL y consultar su versión, podemos utilizar cualquiera de las siguientes formas:



```
// 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);
```

En ambos casos, la variable `$conexion` es de tipo objeto. La utilización de los métodos y propiedades que aporta la clase **mysqli** normalmente produce un código más corto y legible que si utilizas llamadas a funciones.

Toda la información relativa a la instalación y utilización de la extensión, incluyendo las funciones y métodos propios de la extensión, se puede consultar en el manual de PHP.

<http://es.php.net/manual/es/book.mysql.php>

Entre las mejoras que aporta a la antigua extensión mysql, figuran:

- ✓ Interface orientado a objetos.
- ✓ Soporte para transacciones.
- ✓ Soporte para consultas preparadas.
- ✓ Mejores opciones de depuración.

Como ya viste en la primera unidad, 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.

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

<http://es2.php.net/manual/es/mysqli.configuration.php>

¿Qué interface o interfaces de programación admite la extensión MySQLi?



Orientado a objetos únicamente.



Dos interfaces de programación: procedimental y orientado a objetos.

aunque el interface orientado a objetos es una mejora sobre la antigua extensión mysql, MySQLi mantiene también de forma paralela un interface procedimental.

3.1.1.- Establecimiento de conexiones.

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 quieres 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.



Si utilizas el constructor de la clase, para conectarte a la base de datos "dwes" puedes hacer:

```
// utilizando el constructor de la clase
$dwes = new mysqli('localhost', 'dwes', 'abc123.', 'dwes');
```

Aunque también tienes la opción de primero crear la instancia, y después utilizar el método `connect` para establecer la conexión con el servidor:

```
// utilizando el método connect
$dwes = new mysqli();
$dwes->connect('localhost', 'dwes', 'abc123.', 'dwes');
```

Por el contrario, utilizando el interface procedimental de la extensión:

```
// utilizando llamadas a funciones
$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.

Por ejemplo, el siguiente código comprueba el establecimiento de una conexión con la base de datos "dwes" y finaliza la ejecución si se produce algún error:

```
@ $dwes = new mysqli('localhost', 'dwes', 'abc123.', 'dwes');
$error = $dwes->connect_errno;
if ($error != null) {
    echo "<p>Error $error conectando a la base de datos: $dwes->connect_error</p>";
    exit();
}
```

En PHP, como veremos posteriormente con más detalle, puedes anteponer a cualquier expresión el operador de control de errores `@` para que se ignore cualquier posible error que pueda producirse al ejecutarla.

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

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.

```
// utilizando el método connect
$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();
```

3.1.2.- Ejecución de consultas.

La forma más inmediata de ejecutar una consulta, si utilizas esta extensión, es el método `query`, equivalente a la función `mysqli_query`. Si se ejecuta una consulta de acción que no devuelve datos (como una sentencia SQL de tipo `UPDATE`, `INSERT` o `DELETE`), la llamada devuelve `true` si se ejecuta correctamente o `false` en caso contrario. El número de registros afectados se puede obtener con la propiedad `affected_rows` (o con la función `mysqli_affected_rows`).



```
@ $dwes = new mysqli('localhost', 'dwes', 'abc123.', 'dwes');
$error = $dwes->connect_errno;
if ($error == null) {
    $resultado = $dwes->query('DELETE FROM stock WHERE unidades=0');
    if ($resultado) {
```

```
print "<p>Se han borrado $dwes->affected_rows registros.</p>";
}
$dwes->close();
}
```

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`). En el punto siguiente verás cómo se pueden manejar los resultados obtenidos.

El método `query` tiene un parámetro opcional que afecta a cómo se obtienen internamente los resultados, pero no a la forma de utilizarlos posteriormente. En la opción por defecto, `MYSQLI_STORE_RESULT`, los resultados se recuperan todos juntos de la base de datos y se almacenan de forma local. Si cambiamos esta opción por el valor `MYSQLI_USE_RESULT`, los datos se van recuperando del servidor según se vayan necesitando.

```
$resultado = $dwes->query('SELECT producto, unidades FROM stock', MYSQLI_USE_RESULT);
```

Otra forma que puedes utilizar para ejecutar una consulta es el método `real_query` (o la función `mysqli_real_query`), que siempre devuelve `true` o `false` según se haya ejecutado correctamente o no. Si la consulta devuelve un conjunto de resultados, se podrán recuperar de forma completa utilizando el método `store_result`, o según vaya siendo necesario gracias al método `use_result`.

<http://es.php.net/manual/es/mysqli.real-query.php>

Es importante tener en cuenta que los resultados obtenidos se almacenarán en memoria mientras los estés usando. Cuando ya no los necesites, los puedes liberar con el método `free` de la clase `mysqli_result` (o con la función `mysqli_free_result`):

```
$resultado->free();
```

De las dos opciones que admite el método `query`, `MYSQLI_STORE_RESULT` y `MYSQLI_USE_RESULT`, ¿qué opción será recomendable utilizar para ejecutar una consulta que devuelva una enorme cantidad de datos?



`MYSQLI_STORE_RESULT`.



`MYSQLI_USE_RESULT`.

Con esta opción se van obteniendo los datos del servidor a medida que se vayan necesitando. Si utilizaras la otra opción, los datos tendrían que transferirse todos juntos al ejecutar la consulta

3.1.3.- Transacciones.

Como ya comentamos, si necesitas utilizar transacciones deberás asegurarte de que estén soportadas por el motor de almacenamiento que gestiona tus tablas en MySQL. Si utilizas X, por defecto cada consulta individual se incluye dentro de su propia transacción. Puedes gestionar este comportamiento con el método `autocommit` (función `mysqli_autocommit`).

```
$dwes->autocommit(false); // deshabilitamos el modo transaccional automático
```

Al deshabilitar las transacciones automáticas, las siguientes operaciones sobre la base de datos iniciarán una transacción que deberás finalizar utilizando:

- ✓ `commit` (o la función `mysqli_commit`). Realizar una operación "`commit`" de la transacción actual, devolviendo `true` si se ha realizado correctamente o `false` en caso contrario.
- ✓ `rollback` (o la función `mysqli_rollback`). Realizar una operación "`rollback`" de la transacción actual, devolviendo `true` si se ha realizado correctamente o `false` en caso contrario.

```
...
$dwes->query('DELETE FROM stock WHERE unidades=0'); // Inicia una transacción
$dwes->query('UPDATE stock SET unidades=3 WHERE producto="STYLUSSX515W"');
...
```

```
$dwes->commit(); // Confirma los cambios
```

Una vez finalizada esa transacción, comenzará otra de forma automática.

Según la información que figura en la tabla stock de la base de datos dwes, la tienda 1 (CENTRAL) tiene 2 unidades del producto de código 3DSNG y la tienda 3 (SUCURSAL2) ninguno. Suponiendo que los datos son esos (no hace falta que los compruebes en el código), utiliza una transacción para mover una unidad de ese producto de la tienda 1 a la tienda 3.

Deberás hacer una consulta de actualización (para poner unidades=1 en la tienda 1) y otra de inserción (pues no existe ningún registro previo para la tienda 3). Observa el código de la solución. Comprueba que se ejecuta bien solo la primera vez, pues en ejecuciones posteriores ya no es posible insertar la misma fila en la tabla.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 3 : Trabajar con bases de datos en PHP -->
<!-- Ejercicio: Transacción con MySQLi -->
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Ejercicio: Transacción con MySQLi</title>
  </head>
  <body>
    <?php
      @ $dwes = new mysqli("localhost", "dwes", "abc123.", "dwes", 3316);
      $error = $dwes->connect_errno;
      if ($error != null) {
        print "<p>Se ha producido el error: $dwes->connect error.</p>";
        exit();
      }
      // Definimos una variable para comprobar la ejecución de las consultas
      $todo bien = true;
      // Iniciamos la transacción
      $dwes->autocommit(false);
      $sql = 'UPDATE stock SET unidades=1 WHERE producto="3DSNG" AND tienda=1';
      if ($dwes->query($sql) != true) $todo bien = false;
      $sql = 'INSERT INTO `stock` (`producto`, `tienda`, `unidades`) VALUES ("3DSNG", 3,
1)';
      if ($dwes->query($sql) != true) $todo bien = false;
      // Si todo fue bien, confirmamos los cambios
      // y en caso contrario los deshacemos
      if ($todo bien == true) {
        $dwes->commit();
        print "<p>Los cambios se han realizado correctamente.</p>";
      } else {
        $dwes->rollback();
        print "<p>No se han podido realizar los cambios.</p>";
      }
      $dwes->close();
      unset($dwes);
    ?>
  </body>
</html>
```

En el modo de gestión de transacciones que se utiliza por defecto, ¿es posible revertir los cambios que se aplican al ejecutar una consulta de acción?



No



Sí

el modo por defecto indica que se realice un "commit" automático por cada consulta, con lo cual es imposible revertir los cambios que se han realizado.

3.1.4.- Obtención y utilización de conjuntos de resultados.

Ya sabes que al ejecutar una consulta que devuelve datos obtienes un objeto de la clase `mysqli_result`. Esta clase sigue los criterios de ofrecer un interface de programación dual, es decir, una función por cada método con la misma funcionalidad que éste.

Para trabajar con los datos obtenidos del servidor, tienes varias posibilidades:

`fetch_array` (función `mysqli_fetch_array`). Obtiene un registro completo del conjunto de resultados y lo almacena en un array. Por defecto el array contiene tanto claves numéricas como asociativas.

Por ejemplo, para acceder al primer campo devuelto, podemos utilizar como clave el número 0 o su nombre indistintamente.

```
$resultado = $dws->query('SELECT producto, unidades FROM stock WHERE unidades<2');
$stock = $resultado->fetch_array(); // Obtenemos el primer registro
$producto = $stock['producto']; // O también $stock[0];
$unidades = $stock['unidades']; // O también $stock[1];
print "<p>Producto $producto: $unidades unidades.</p>";
```

Este comportamiento por defecto se puede modificar utilizando un parámetro opcional, que puede tomar los siguientes valores:

- ✓ `MYSQLI_NUM`. Devuelve un array con claves numéricas.
- ✓ `MYSQLI_ASSOC`. Devuelve un array asociativo.
- ✓ `MYSQLI_BOTH`. Es el comportamiento por defecto, en el que devuelve un array con claves numéricas y asociativas.

`fetch_assoc` (función `mysqli_fetch_assoc`). Idéntico a `fetch_array` pasando como parámetro `MYSQLI_ASSOC`.

`fetch_row` (función `mysqli_fetch_row`). Idéntico a `fetch_array` pasando como parámetro `MYSQLI_NUM`.

`fetch_object` (función `mysqli_fetch_object`). Similar a los métodos anteriores, pero devuelve un objeto en lugar de un array. Las propiedades del objeto devuelto se corresponden con cada uno de los campos del registro.

Para recorrer todos los registros de un array, puedes hacer un bucle teniendo en cuenta que cualquiera de los métodos o funciones anteriores devolverá `null` cuando no haya más registros en el conjunto de resultados.

```
$resultado = $dws->query('SELECT producto, unidades FROM stock WHERE unidades<2');
$stock = $resultado->fetch_object();
while ($stock != null) {
    print "<p>Producto $stock->producto: $stock->unidades unidades.</p>";
    $stock = $resultado->fetch_object();
}
```

En el manual de PHP tienes más información sobre los métodos y propiedades de la clase `mysqli_result`.

<http://es.php.net/manual/es/class.mysqli-result.php>

Crea una página web en la que se muestre el stock existente de un determinado producto en cada una de las tiendas. Para seleccionar el producto concreto utiliza un cuadro de selección dentro de un formulario en esa misma página. Puedes usar como base los siguientes ficheros.

Plantilla.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
```

```

<title>Plantilla para Ejercicios Tema 3</title>
<link href="dwes.css" rel="stylesheet" type="text/css">
</head>

<body>

<div id="encabezado">
  <h1>Ejercicio: </h1>
  <form id="form_seleccion" action="<?php echo $_SERVER['PHP_SELF'];?>" method="post">
  </form>
</div>

<div id="contenido">
  <h2>Contenido</h2>
</div>

<div id="pie">
</div>
</body>
</html>

```

dwes.css

```

h1 {margin-bottom:0;}
#encabezado {background-color:#ddf0a4;}
#contenido {background-color:#EEEEEE;height:600px;}
#pie {background-color:#ddf0a4;color:#ff0000;height:30px;}

```

Revisa la solución propuesta y verifica que los resultados que obtuviste son correctos.

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 3 : Trabajar con bases de datos en PHP -->
<!-- Ejemplo: Conjuntos de datos con MySQLi -->
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Ejercicio Tema 3: Conjuntos de resultados en MySQLi</title>
    <link href="dwes.css" rel="stylesheet" type="text/css">
  </head>
  <body>
    <div id="encabezado">
      <h1>Ejercicio: Conjuntos de resultados en MySQLi</h1>
      <form id="form_seleccion" action="." method="post">
        <span>Producto: </span>
        <select name="producto">
          <?php
            if (isset($_POST['producto'])) $producto = $_POST['producto'];
            // Rellenamos el desplegable con los datos de todos los productos
            @ $dwes = new mysqli("localhost", "dwes", "abc123.", "dwes");
            $error = $dwes->connect_errno;
            if ($error == null) {
              $sql = "SELECT cod, nombre_corto FROM producto";
              $resultado = $dwes->query($sql);
              if($resultado) {
                $row = $resultado->fetch_assoc();
                while ($row != null) {
                  echo "<option value='{$row['cod']}'>";
                  // Si se recibió un código de producto lo seleccionamos
                  // en el desplegable usando selected='true'
                  if (isset($producto) && $producto == $row['cod'])
                    echo " selected='true'";
                  echo ">{$row['nombre_corto']}</option>";
                  $row = $resultado->fetch_assoc();
                }
                $resultado->close();
              }
            } else {
              $mensaje = $dwes->connect_error;
            }
          ?>
        </select>
        <input type="submit" value="Mostrar stock" name="enviar"/>
      </form>
    </div>
    <div id="contenido">

```

```

<h2>Stock del producto en las tiendas:</h2>
<?php
// Si se recibió un código de producto y no se produjo ningún error
// mostramos el stock de ese producto en las distintas tiendas
if ($error == null && isset($producto)) {$sql = <<<SQL
SELECT tienda.nombre, stock.unidades
FROM tienda INNER JOIN stock ON tienda.cod=stock.tienda
WHERE stock.producto='$producto'
SQL;

$resultado = $dwes->query($sql);
if($resultado) {
    $row = $resultado->fetch_assoc();
    while ($row != null) {
        echo "<p>Tienda ${row['nombre']}: ${row['unidades']} unidades.</p>";
        $row = $resultado->fetch_assoc();
    }
    $resultado->close();
}
}
?>
</div>
<div id="pie">
<?php
// Si se produjo algún error se muestra en el pie
if ($error != null) echo "<p>Se ha producido un error! $mensaje</p>";
else {
    $dwes->close();
    unset($dwes);
}
?>
</div>
</body>
</html>

```

3.1.5.- Consultas preparadas.

Cada vez que se envía una consulta al servidor, éste debe analizarla antes de ejecutarla. Algunas sentencias SQL, como las que insertan valores en una tabla, deben repetirse de forma habitual en un programa. Para acelerar este proceso, MySQL admite consultas preparadas. Estas consultas se almacenan en el servidor listas para ser ejecutadas cuando sea necesario.

Para trabajar con consultas preparadas con la extensión MySQLi de PHP, debes utilizar la clase **mysqli_stmt**. Utilizando el método **stmt_init** de la clase **mysqli** (o la función **mysqli_stmt_init**) obtienes un objeto de dicha clase.

```

$dwes = new mysqli('localhost', 'dwes', 'abc123.', 'dwes');
$consulta = $dwes->stmt_init();

```

Los pasos que debes seguir para ejecutar una consulta preparada son:

- ✓ Preparar la consulta en el servidor MySQL utilizando el método **prepare** (función **mysqli_stmt_prepare**).
- ✓ Ejecutar la consulta, tantas veces como sea necesario, con el método **execute** (función **mysqli_stmt_execute**).
- ✓ Una vez que ya no se necesita más, se debe ejecutar el método **close** (función **mysqli_stmt_close**).

Por ejemplo, para preparar y ejecutar una consulta que inserta un nuevo registro en la tabla familia:

```

$consulta = $dwes->stmt_init();
$consulta->prepare('INSERT INTO familia (cod, nombre) VALUES ("TABLET", "Tablet PC")');
$consulta->execute();
$consulta->close();
$dwes->close();

```

El problema que ya habrás observado, es que de poco sirve preparar una consulta de inserción de datos como la anterior, si los valores que inserta son siempre los mismos. Por este motivo las

consultas preparadas admiten parámetros. Para preparar una consulta con parámetros, en lugar de poner los valores debes indicar con un signo de interrogación su posición dentro de la sentencia SQL.

```
$consulta->prepare('INSERT INTO familia (cod, nombre) VALUES (?, ?)');
```

Y antes de ejecutar la consulta tienes que utilizar el método `bind_param` (o la función `mysqli_stmt_bind_param`) para sustituir cada parámetro por su valor. El primer parámetro del método `bind_param` es una cadena de texto en la que cada carácter indica el tipo de un parámetro, según la siguiente tabla.

Caracteres indicativos del tipo de los parámetros en una consulta preparada.	
Carácter.	Tipo del parámetro.
I.	Número entero.
D.	Número real (doble precisión).
S.	Cadena de texto.
B.	Contenido en formato binario (BLOB).

En el caso anterior, si almacenas los valores a insertar en sendas variables, puedes hacer:

```
$consulta = $dwes->stmt_init();
$consulta->prepare('INSERT INTO familia (cod, nombre) VALUES (?, ?)');
$cod_producto = "TABLET";
$nombre_producto = "Tablet PC";
$consulta->bind_param('ss', $cod_producto, $nombre_producto);
$consulta->execute();
$consulta->close();
$dwes->close();
```

Cuando uses `bind_param` para enlazar los parámetros de una consulta preparada con sus respectivos valores, deberás usar siempre variables como en el ejemplo anterior. Si intentas utilizar literales, por ejemplo:

```
$consulta->bind_param('ss', 'TABLET', 'Tablet PC'); // Genera un error
```

Obtendrás un error. El motivo es que los parámetros del método `bind_param` se pasan por referencia. Aprenderás a usar paso de parámetros por referencia en una unidad posterior.

El método `bind_param` permite tener una consulta preparada en el servidor MySQL y ejecutarla tantas veces como quieras cambiando ciertos valores cada vez. Además, en el caso de las consultas que devuelven valores, se puede utilizar el método `bind_result` (función `mysqli_stmt_bind_result`) para asignar a variables los campos que se obtienen tras la ejecución. Utilizando el método `fetch` (`mysqli_stmt_fetch`) se recorren los registros devueltos. Observa el siguiente código:

```
$consulta = $dwes->stmt_init();
$consulta->prepare('SELECT producto, unidades FROM stock WHERE unidades<2');
$consulta->execute();
$consulta->bind_result($producto, $unidades);
while($consulta->fetch()) {
    print "<p>Producto $producto: $unidades unidades.</p>";
}
$consulta->close();
$dwes->close();
```

En el manual de PHP tienes más información sobre consultas preparadas y la clase `mysqli_stmt`.

<http://es.php.net/manual/es/class.mysqli-stmt.php>

A partir de la página web obtenida en el ejercicio anterior, añade la opción de modificar el número de unidades del producto en cada una de las tiendas. Utiliza una consulta preparada para la actualización de registros en la tabla stock. No es necesario tener en cuenta las tareas de inserción (no existían unidades anteriormente) y borrado (si el número final de unidades es cero).

En esta ocasión es necesario crear un nuevo formulario en la página, en la sección donde se muestra el número de unidades por tienda. Cuando se envía ese formulario, hay que preparar la consulta y ejecutarla una vez por cada registro de la tabla stock (una vez por cada tienda en la que exista stock de ese producto). Revisa el código de la solución y pruébalo.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 3 : Trabajar con bases de datos en PHP -->
<!-- Ejemplo: Consultas preparadas con MySQLi -->
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Ejercicio Tema 3: Consultas preparadas en MySQLi</title>
    <link href="dwes.css" rel="stylesheet" type="text/css">
  </head>
  <body>
    <?php
      if (isset($_POST['producto'])) $_producto = $_POST['producto'];
      @ $dwes = new mysqli("localhost", "dwes", "abc123.", "dwes");
      $error = $dwes->connect_errno;
      if ($error == null) {
        // Comprobamos si tenemos que actualizar los valores
        if (isset($_POST['actualiz'])) {
          // Preparamos la consulta
          $tienda = $_POST['tienda'];
          $unidades = $_POST['unidades'];
          $consulta = $dwes->stmt_init();
          $sql = "UPDATE stock SET unidades=? WHERE tienda=? AND
producto='$_producto'";
          $consulta->prepare($sql);
          // La ejecutamos dentro de un bucle, tantas veces como tiendas haya
          for($i=0;$i<count($tienda);$i++) {
            $consulta->bind_param('ii', $unidades[$i], $tienda[$i]);
            $consulta->execute();
          }
          $mensaje = "Se han actualizado los datos.";
          $consulta->close();
        }
        // Si no se ha podido establecer la conexión, generamos un mensaje de error
        else $mensaje = $dwes->connect_error;
      }
    <?php
    <div id="encabezado">
      <h1>Ejercicio: Consultas preparadas con MySQLi</h1>
      <form id="form_seleccion" action="." method="post">
        <span>Producto: </span>
        <select name="producto">
          <?php
            // Rellenamos el desplegable con los datos de todos los productos
            if ($error == null) {
              $sql = "SELECT cod, nombre_corto FROM producto";
              $resultado = $dwes->query($sql);
              if ($resultado) {
                $row = $resultado->fetch_assoc();
                while ($row != null) {echo "<option value='{$row['cod']}'>";
                // Si se recibió un código de producto lo seleccionamos
                // en el desplegable usando selected='true'
                if (isset($_producto) && $_producto == $row['cod'])
                  echo " selected='true'";
                echo ">{$row['nombre_corto']}</option>";
                $row = $resultado->fetch_assoc();
                }
                $resultado->close();
              }
            }
          <?php
        </select>
        <input type="submit" value="Mostrar stock" name="enviar"/>
      </form>
    </div>
    <div id="contenido">
      <h2>Stock del producto en las tiendas:</h2>
    <?php
```

```

// Si se recibió un código de producto y no se produjo ningún error
// mostramos el stock de ese producto en las distintas tiendas
if ($error == null && isset($producto)) {
    // Ahora necesitamos también el código de tienda
    $sql = <<<SQL
    SELECT tienda.cod, tienda.nombre, stock.unidades
    FROM tienda INNER JOIN stock ON tienda.cod=stock.tienda
    WHERE stock.producto='$producto'
SQL;

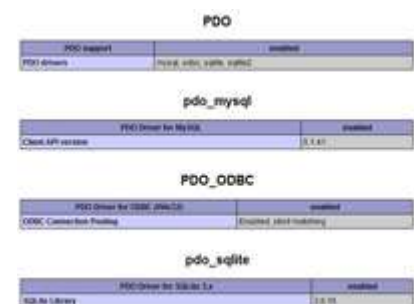
    $resultado = $dwes->query($sql);
    if($resultado) {
        // Creamos un formulario con los valores obtenidos
        echo '<form id="form actualiz" action="." method="post">';
        $row = $resultado->fetch_assoc();
        while ($row != null) {
            // Metemos ocultos el código de producto y los de las tiendas
            echo "<input type='hidden' name='producto' value='$producto' />";
            echo "<input type='hidden' name='tienda[]' value='". $row['cod']. "' />";
            echo "<p>Tienda ${row['nombre']}: ";
            // El número de unidades ahora va en un cuadro de texto
            echo "<input type='text' name='unidades[]' size='4' ";
            echo "value='". $row['unidades']. "' /> unidades.</p>";
            $row = $resultado->fetch_assoc();
        }
        $resultado->close();
        echo "<input type='submit' value='Actualizar' name='actualiz' />";
        echo "</form>";
    }
}
?>
</div>
<div id="pie">
    <?php
    // Si se produjo algún error se muestra en el pie
    if ($error != null) echo "<p>Se ha producido un error! $mensaje</p>";
    else {
        echo $mensaje;
        $dwes->close();
        unset($dwes);
    }
    ?>
</div>
</body>
</html>

```

3.2.- PHP Data Objects (PDO).

Si vas a programar una aplicación que utilice como sistema gestor de bases de datos MySQL, la extensión MySQLi que acabas de ver es una buena opción. Ofrece acceso a todas las características del motor de base de datos, a la vez que reduce los tiempos de espera en la ejecución de sentencias.

Sin embargo, **si en el futuro tienes que cambiar el SGBD** por otro distinto, tendrás que volver a programar gran parte del código de la misma. Por eso, antes de comenzar el desarrollo, es muy importante revisar las características específicas del proyecto. En el caso de que exista la posibilidad, presente o futura, de utilizar otro servidor como almacenamiento, 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, pero éste no es el objetivo principal de PDO. PDO no abstrae de forma completa el sistema gestor que se utiliza. Por ejemplo, no modifica las sentencias SQL para adaptarlas a las características específicas de cada servidor. Si esto fuera necesario, habría que programar una capa de abstracción completa.

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.

3.2.1.- Establecimiento de conexiones.

Para establecer una conexión con una base de datos utilizando PDO, debes instanciar un objeto de la clase PDO pasándole los siguientes parámetros (solo el primero es obligatorio):

- ✓ Origen de datos (DSN). Es una cadena de texto que indica qué controlador se va a utilizar y a continuación, separadas por el carácter dos puntos, los parámetros específicos necesarios por el controlador, como por ejemplo el nombre o dirección IP del servidor y el nombre de la base de datos.
- ✓ Nombre de usuario con permisos para establecer la conexión.
- ✓ Contraseña del usuario.
- ✓ Opciones de conexión, almacenadas en forma de array.

Por ejemplo, podemos establecer una conexión con la base de datos 'dwes' creada anteriormente de la siguiente forma:

```
$dwes = new PDO('mysql:host=localhost;dbname=dwes', 'dwes', 'abc123.');
```

Si como en el ejemplo, se utiliza el controlador para MySQL, los parámetros específicos para utilizar en la cadena DSN (separadas unas de otras por el carácter punto y coma) a continuación del prefijo **mysql:** son los siguientes:

- ✓ `host`. Nombre o dirección IP del servidor.
- ✓ `port`. Número de puerto TCP en el que escucha el servidor.
- ✓ `dbname`. Nombre de la base de datos.
- ✓ `unix_socket`. Socket de MySQL en sistemas Unix.

Si quisieras indicar al servidor MySQL utilice codificación UTF-8 para los datos que se transmitan, puedes usar una opción específica de la conexión:

```
$opciones = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8");  
$dwes = new PDO('mysql:host=localhost;dbname=dwes', 'dwes', 'abc123.', $opciones);
```

En el manual de PHP puedes consultar más información sobre los controladores existentes, los parámetros de las cadenas DSN y las opciones de conexión particulares de cada uno.

<http://es.php.net/manual/es/pdo.drivers.php>

Una vez establecida la conexión, puedes utilizar el método `getAttribute` para obtener información del estado de la conexión y `setAttribute` para modificar algunos parámetros que afectan a la misma. Por ejemplo, para obtener la versión del servidor puedes hacer:

```
$version = $dwes->getAttribute(PDO::ATTR_SERVER_VERSION);  
print "Versión: $version";
```

Y si quieres por ejemplo que te devuelva todos los nombres de columnas en mayúsculas:

```
$version = $dwes->setAttribute(PDO::ATTR_CASE, PDO::CASE_UPPER);
```

En el manual de PHP, las páginas de las funciones `getAttribute` y `setAttribute` te permiten consultar los posibles parámetros que se aplican a cada una.

<http://es.php.net/manual/es/pdo.getattribute.php>
<http://es.php.net/manual/es/pdo.setattribute.php>

Para establecer una conexión con MySQL utilizando PDO, ¿dónde se puede indicar el número de puerto TCP?



En la cadena DSN que indica el origen de datos.



En el array en que figuran las opciones específicas de conexión con el servidor.

Se incluye como parte de la cadena DSN utilizando el parámetro "port"

3.2.2.- Ejecución de consultas.

Para ejecutar una consulta SQL utilizando PDO, debes diferenciar aquellas sentencias SQL que no devuelven como resultado un conjunto de datos, de aquellas otras que sí lo devuelven.

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

```
$registros = $dwes->exec('DELETE FROM stock WHERE unidades=0');
print "<p>Se han borrado $registros registros.</p>";
```

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`.

```
$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
$resultado = $dwes->query("SELECT producto, unidades FROM stock");
```

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.

Una vez ejecutado un `commit` o un `rollback`, se volverá al modo de confirmación automática.

```
$ok = true;
$dwes->beginTransaction();
if($dwes->exec('DELETE ...') == 0) $ok = false;
if($dwes->exec('UPDATE ...') == 0) $ok = false;
...
if ($ok) $dwes->commit(); // Si todo fue bien confirma los cambios
else $dwes->rollback(); // y si no, los revierte
```

Ten en cuenta que no todos los motores soportan transacciones. Tal es el caso, como ya viste, del motor MyISAM de MySQL. En este caso concreto, PDO ejecutará el método `beginTransaction` sin errores, pero naturalmente no será capaz de revertir los cambios si fuera necesario ejecutar un `rollback`.

De una forma similar al anterior ejercicio de transacciones, utiliza PDO para repartir entre las tiendas las tres unidades que figuran en stock del producto con código PAPYRE62GB.

En esta ocasión, para comprobar si los cambios se hacen correctamente en la base de datos y confirmamos la transacción, se revisa el número de registros afectados por la ejecución de las consultas. Revisa el código de la página y comprueba que la segunda vez que intentas ejecutarlo no actualizará los datos, tal y como sucedía en el ejercicio equivalente de la extensión MySQLi.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 3 : Trabajar con bases de datos en PHP -->
<!-- Ejercicio: Transacción con PDO -->
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Ejercicio: Transacción con PDO</title>
  </head>
  <body>
    <?php
      $dwes = new PDO('mysql:host=localhost;dbname=dwes', 'dwes', 'abc123. ');
      // Definimos una variable para comprobar la ejecución de las consultas
      $todo_bien = true;
      // Iniciamos la transacción
      $dwes->beginTransaction();
      $sql = 'UPDATE stock SET unidades=1 WHERE producto="PAPYRE62GB" AND tienda=1';
      if ($dwes->exec($sql) == 0) $todo_bien = false;
      $sql = 'INSERT INTO `stock` (`producto`, `tienda`, `unidades`) VALUES
        ("PAPYRE62GB", 2, 1)';
      if ($dwes->exec($sql) == 0) $todo_bien = false;
      // Si todo fue bien, confirmamos los cambios
      // y en caso contrario los deshacemos
      if ($todo_bien == true) {
        $dwes->commit();
        print "<p>Los cambios se han realizado correctamente.</p>";
      } else {
        $dwes->rollback();
        print "<p>No se han podido realizar los cambios.</p>";
      }
      unset($dwes);
    ?>
  </body>
</html>
```

Si programas tu aplicación correctamente utilizando "beginTransaction" antes de realizar un cambio, ¿siempre será posible revertirlo utilizando "rollback"?



Sí



No

depende de si el motor de almacenamiento que estás utilizando soporta o no transacciones.

3.2.3.- Obtención y utilización de conjuntos de resultados.

Al igual que con la extensión MySQLi, en PDO tienes varias posibilidades para tratar con el conjunto de resultados devuelto por el método `query`. La más utilizada es el método `fetch` de la clase `PDOStatement`. Este método devuelve un registro del conjunto de resultados, o `false` si ya no quedan registros por recorrer.

```
$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
$resultado = $dwes->query("SELECT producto, unidades FROM stock");
while ($registro = $resultado->fetch()) {
    echo "Producto ".$registro['producto'].": ".$registro['unidades']."<br />";
}
```

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.

```
$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
$resultado = $dwes->query("SELECT producto, unidades FROM stock");
```

```
while ($registro = $resultado->fetch(PDO::FETCH_OBJ)) {
    echo "Producto ".$registro->producto." : ".$registro->unidades."<br />";
}
```

✓ **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**. Este método debe ser llamado una vez por cada columna, indicando en cada llamada el número de columna (empezando en 1) y la variable a asignar.

```
$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
$resultado = $dwes->query("SELECT producto, unidades FROM stock");
$resultado->bindColumn(1, $producto);
$resultado->bindColumn(2, $unidades);
while ($registro = $resultado->fetch(PDO::FETCH_OBJ)) {
    echo "Producto ".$producto." : ".$unidades."<br />";
}
```

Modifica la página web que muestra el stock de un producto en las distintas tiendas, obtenida en un ejercicio anterior utilizando MySQLi, para que use PDO. Omite la gestión de errores, que veremos en el último punto de este tema.

Comprueba en la solución propuesta los cambios realizados respecto a la solución del ejercicio equivalente que utilizaba MySQLi.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 3 : Trabajar con bases de datos en PHP -->
<!-- Ejemplo: Conjuntos de datos con PDO -->
<html>
    <head>
        <meta http-equiv="content-type" content="text/html; charset=UTF-8">
        <title>Ejercicio Tema 3: Conjuntos de resultados en PDO</title>
        <link href="dwes.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <div id="encabezado">
            <h1>Ejercicio: Conjuntos de resultados en PDO</h1>
            <form id="form_seleccion" action="." method="post">
                <span>Producto: </span>
                <select name="producto">
                    <?php
                        if (isset($ POST['producto'])) $producto = $ POST['producto'];
                        // Rellenamos el desplegable con los datos de todos los productos
                        $dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
                        $sql = "SELECT cod, nombre_corto FROM producto";
                        $resultado = $dwes->query($sql);
                        if($resultado) {
                            $row = $resultado->fetch();
                            while ($row != null) {
                                echo "<option value='{$row['cod']}'>";
                                // Si se recibió un código de producto lo seleccionamos
                                // en el desplegable usando selected='true'
                                if (isset($producto) && $producto == $row['cod'])
                                    echo " selected='true'";
                                echo ">{$row['nombre_corto']}</option>";
                                $row = $resultado->fetch();
                            }
                        }
                    <?>
                </select>
                <input type="submit" value="Mostrar stock" name="enviar"/>
            </form>
        </div>
        <div id="contenido">
            <h2>Stock del producto en las tiendas:</h2>
            <?php
                // Si se recibió un código de producto
                // mostramos el stock de ese producto en las distintas tiendas
                if (isset($producto)) {
                    $sql = <<<SQL
                        SELECT tienda.nombre, stock.unidades
                        FROM tienda INNER JOIN stock ON tienda.cod=stock.tienda
                        WHERE stock.producto='{$producto}'
                    SQL;
                    $resultado = $dwes->query($sql);
```

```

        if($resultado) {
            $row = $resultado->fetch();
            while ($row != null) {
                echo "<p>Tienda {$row['nombre']}: {$row['unidades']} unidades.</p>";
                $row = $resultado->fetch();
            }
        }
    }
    ?>
</div>
<div id="pie">
    <?php
        unset($dwes);
    ?>
</div>
</body>
</html>

```

¿Cuál es el comportamiento por defecto del método "fetch"?



Devuelve un array con claves numéricas y asociativas.



Devuelve un array asociativo.

Es similar a utilizar el parámetro "PDO::FETCH_BOTH"

3.2.4.- Consultas preparadas.

Al igual que con MySQLi, también utilizando PDO podemos preparar consultas parametrizadas en el servidor para ejecutarlas de forma repetida. El procedimiento es similar e incluso los métodos a ejecutar tienen prácticamente los mismos nombres.

Para preparar la consulta en el servidor MySQL, deberás utilizar el método `prepare` de la clase PDO. Este método devuelve un objeto de la clase `PDOStatement`. Los parámetros se pueden marcar utilizando signos de interrogación como en el caso anterior.

```

$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
$consulta = $dwes->prepare('INSERT INTO familia (cod, nombre) VALUES (?, ?)');

```

O también utilizando parámetros con nombre, precediéndolos por el símbolo de dos puntos.

```

$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
$consulta = $dwes->prepare('INSERT INTO familia (cod, nombre) VALUES (:cod, :nombre)');

```

Antes de ejecutar la consulta hay que asignar un valor a los parámetros utilizando el método `bindParam` de la clase `PDOStatement`. Si utilizas signos de interrogación para marcar los parámetros, el procedimiento es equivalente al método `bindColumn` que acabamos de ver.

```

$cod_producto = "TABLET";
$nombre_producto = "Tablet PC";
$consulta->bindParam(1, $cod_producto);
$consulta->bindParam(2, $nombre_producto);

```

Si utilizas parámetros con nombre, debes indicar ese nombre en la llamada a `bindParam`.

```

$consulta->bindParam(":cod", $cod_producto);
$consulta->bindParam(":nombre", $nombre_producto);

```

Tal y como sucedía con la extensión MySQLi, cuando uses `bindParam` para asignar los parámetros de una consulta preparada, deberás usar siempre variables como en el ejemplo anterior.

Una vez preparada la consulta y enlazados los parámetros con sus valores, se ejecuta la consulta utilizando el método `execute`.

```

$consulta->execute();

```

Alternativamente, es posible asignar los valores de los parámetros en el momento de ejecutar la consulta, utilizando un array (asociativo o con claves numéricas dependiendo de la forma en que hayas indicado los parámetros) en la llamada a `execute`.

```
$parametros = array(":cod" => "TABLET", ":nombre" => "Tablet PC");
$consulta->execute($parametros);
```

Puedes consultar la información sobre la utilización en PDO de consultas preparadas y la clase PDOStatement en el manual de PHP.

<http://es.php.net/manual/es/class.pdostatement.php>

Modifica el ejercicio sobre consultas preparadas que realizaste con la extensión MySQLi, el que modificaba el número de unidades de un producto en las distintas tiendas, para que utilice ahora la extensión PDO.

Como puedes comprobar, para obtener la solución se puede aprovechar la mayoría del código existente en el ejercicio anterior.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 3 : Trabajar con bases de datos en PHP -->
<!-- Ejemplo: Consultas preparadas con PDO -->
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Ejercicio Tema 3: Consultas preparadas en PDO</title>
    <link href="dwes.css" rel="stylesheet" type="text/css">
  </head>
  <body>
    <?php
      if (isset($_POST['producto'])) $producto = $_POST['producto'];
      $dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
      // Comprobamos si tenemos que actualizar los valores
      if (isset($_POST['actualiz'])) {
        // Preparamos la consulta
        $tienda = $_POST['tienda'];
        $unidades = $_POST['unidades'];
        $sql = "UPDATE stock SET unidades=:unidades WHERE tienda=:tienda";
        $sql .= " AND producto='$producto'";
        $consulta=$dwes->prepare($sql);
        // La ejecutamos dentro de un bucle, tantas veces como tiendas haya
        for($i=0;$i<count($tienda);$i++) {
          $consulta->bindParam(":unidades", $unidades[$i]);
          $consulta->bindParam(":tienda", $tienda[$i]);
          $consulta->execute();
        }
        $mensaje = "Se han actualizado los datos.";
      }
    ?>
    <div id="encabezado">
      <h1>Ejercicio: Consultas preparadas en PDO</h1>
      <form id="form_seleccion" action="." method="post">
        <span>Producto: </span>
        <select name="producto">
          <?php
            // Rellenamos el desplegable con los datos de todos los productos
            $sql = "SELECT cod, nombre_corto FROM producto";
            $resultado = $dwes->query($sql);
            if($resultado) {
              $row = $resultado->fetch();
              while ($row != null) {
                echo "<option value='{$row['cod']}'>";
                // Si se recibió un código de producto lo seleccionamos
                // en el desplegable usando selected='true'
                if (isset($producto) && $producto == $row['cod'])
                  echo " selected='true'";
                echo ">{$row['nombre_corto']}</option>";
                $row = $resultado->fetch();
              }
            }
          ?>
        </select>
```

```
        <input type="submit" value="Mostrar stock" name="enviar"/>
    </form>
</div>
<div id="contenido">
    <h2>Stock del producto en las tiendas:</h2>
    <?php
        // Si se recibió un código de producto y no se produjo ningún error
        // mostramos el stock de ese producto en las distintas tiendas
        if (isset($producto)) {
            // Ahora necesitamos también el código de tienda
            $sql = <<<SQL
                SELECT tienda.cod, tienda.nombre, stock.unidades
                FROM tienda INNER JOIN stock ON tienda.cod=stock.tienda
                WHERE stock.producto='$producto'
SQL;

            $resultado = $dwes->query($sql);
            if($resultado) {
                // Creamos un formulario con los valores obtenidos
                echo '<form id="form_actualiz" action="." method="post">';
                $row = $resultado->fetch();
                while ($row != null) {
                    // Metemos ocultos el código de producto y los de las tiendas
                    echo "<input type='hidden' name='producto' value='$producto' />";
                    echo "<input type='hidden' name='tienda[]' value='". $row['cod']. "' />";
                    echo "<p>Tienda ${row['nombre']}: ";
                    // El número de unidades ahora va en un cuadro de texto
                    echo "<input type='text' name='unidades[]' size='4' ";
                    echo "value='". $row['unidades']. "' /> unidades.</p>";
                    $row = $resultado->fetch();
                }
                echo "<input type='submit' value='Actualizar' name='actualiz' />";
                echo "</form>";
            }
        }
    ?>
</div>
<div id="pie">
    <?php
        echo $mensaje;
        unset($dwes);
    ?>
</div>
</body>
</html>
```

4.- Errores y manejo de excepciones.

Caso práctico

En sus primeros pasos en la programación en lenguaje PHP, **Carlos se ha encontrado frecuentemente con pequeños errores en el código**. En la mayoría de las ocasiones los errores estaban causados por fallos de programación. Pero en las recientes pruebas llevadas a cabo con bases de datos, se ha dado cuenta de que **en algunas ocasiones se pueden producir fallos ajenos al programa**.

Por ejemplo, puede obtener un error porque **no esté disponible el servidor de bases de datos** con el que se ha de conectar la aplicación, o porque **se acaba de borrar de la base de datos un registro al que estaba accediendo**. En éstos, y muchos otros, casos es necesario que la aplicación que desarrollen se comporte de forma sólida y coherente. Es necesario estudiar a fondo las **posibilidades que ofrece PHP para la gestión de errores**.

A buen seguro que, conforme has ido resolviendo ejercicios o simplemente probando código, te has encontrado con errores de programación. Algunos son reconocidos por el entorno de desarrollo (NetBeans), y puedes corregirlos antes de ejecutar. Otros aparecen en el navegador en forma de mensaje de error al ejecutar el guión.

PHP define una clasificación de los errores que se pueden producir en la ejecución de un programa y ofrece métodos para ajustar el tratamiento de los mismos. Para hacer referencia a cada uno de los niveles de error, PHP define una serie de constantes. Cada nivel se identifica por una constante. Por ejemplo, la constante `E_NOTICE` hace referencia a avisos que pueden indicar un error al ejecutar el guión, y la constante `E_ERROR` engloba errores fatales que provocan que se interrumpa forzosamente la ejecución.

La lista completa de constantes la puedes consultar en el manual de PHP, donde también se describe el tipo de errores que representa.

<http://es.php.net/manual/es/errorfunc.constants.php>

La configuración inicial de cómo se va a tratar cada error según su nivel se realiza en `php.ini` el fichero de configuración de PHP. Entre los principales parámetros que puedes ajustar están:

- ✓ `error_reporting`. Indica qué tipos de errores se notificarán. Su valor se forma utilizando los operadores a nivel de bit para combinar las constantes anteriores. Su valor predeterminado es `E_ALL & ~E_NOTICE` que indica que se notifiquen todos los errores (`E_ALL`) salvo los avisos en tiempo de ejecución (`E_NOTICE`).
- ✓ `display_errors`. En su valor por defecto (`on`), hace que los mensajes se envíen a la salida estándar (y por lo tanto se muestren en el navegador). Se debe desactivar (`off`) en los servidores que no se usan para desarrollo sino para producción.

Existen otros parámetros que podemos utilizar en `php.ini` para ajustar el comportamiento de PHP cuando se produce un error.

<http://es.php.net/manual/es/errorfunc.configuration.php>

Desde código, puedes usar la función `error_reporting` con las constantes anteriores para establecer el nivel de notificación en un momento determinado. Por ejemplo, si en algún lugar de tu código figura una división en la que exista la posibilidad de que el divisor sea cero, cuando esto ocurra obtendrás un mensaje de error en el navegador. Para evitarlo, puedes desactivar la notificación de errores de nivel `E_WARNING` antes de la división y restaurarla a su valor normal a continuación:

```
error_reporting(E_ALL & ~E_NOTICE & ~E_WARNING);  
$resultado = $dividendo / $divisor;  
error_reporting(E_ALL & ~E_NOTICE);
```

Al usar la función `error_reporting` solo controlas qué tipo de errores va a notificar PHP. A veces puede ser suficiente, pero para obtener más control sobre el proceso existe también la posibilidad de reemplazar la gestión de los mismos por la que tú definas. Es decir, puedes programar una función para que sea la que ejecuta PHP cada vez que se produce un error. El nombre de esa función se indica utilizando `set_error_handler` y debe tener como mínimo dos parámetros obligatorios (el nivel del error y el mensaje descriptivo) y hasta otros tres opcionales con información adicional sobre el error (el nombre del fichero en que se produce, el número de línea, y un volcado del estado de las variables en ese momento).

```
set_error_handler("miGestorDeErrores");
$resultado = $dividendo / $divisor;
restore_error_handler();

function miGestorDeErrores($nivel, $mensaje)
{
    switch($nivel) {
        case E_WARNING:
            echo "Error de tipo WARNING: $mensaje.<br />";
            break;
        default:
            echo "Error de tipo no especificado: $mensaje.<br />";
    }
}
```

La función `restore_error_handler` restaura el manejador de errores original de PHP (más concretamente, el que se estaba usando antes de la llamada a `set_error_handler`).

4.1.- Excepciones.

A partir de la versión 5 se introdujo en PHP un modelo de excepciones similar al existente en otros lenguajes de programación:

- ✓ El código susceptible de producir algún error se introduce en un bloque `try`.
- ✓ Cuando se produce algún error, se lanza una excepción utilizando la instrucción `throw`.
- ✓ Después del bloque `try` debe haber como mínimo un bloque `catch` encargado de procesar el error.
- ✓ Si una vez acabado el bloque `try` no se ha lanzado ninguna excepción, se continúa con la ejecución en la línea siguiente al bloque o bloques `catch`.

Por ejemplo, para lanzar una excepción cuando se produce una división por cero podrías hacer:

```
try {
    if ($divisor == 0)
        throw new Exception("División por cero.");
    $resultado = $dividendo / $divisor;
}
catch (Exception $e) {
    echo "Se ha producido el siguiente error: ".$e->getMessage();
}
```

PHP ofrece una clase base `Exception` para utilizar como manejador de excepciones. Para lanzar una excepción no es necesario indicar ningún parámetro, aunque de forma opcional se puede pasar un mensaje de error (como en el ejemplo anterior) y también un código de error.

Entre los métodos que puedes usar con los objetos de la clase `Exception` están:

- ✓ `getMessage`. Devuelve el mensaje, en caso de que se haya puesto alguno.
- ✓ `getCode`. Devuelve el código de error si existe.

Las funciones internas de PHP y muchas extensiones como MySQLi usan el sistema de errores visto anteriormente. Solo las extensiones más modernas orientadas a objetos, como es el caso de PDO, utilizan este modelo de excepciones. En este caso, lo más común es que la extensión defina sus propios manejadores de errores heredando de la clase `Exception` (veremos cómo utilizar la herencia en una unidad posterior).

Utilizando la clase `ErrorException` es posible traducir los errores a excepciones.

<http://es.php.net/manual/es/class.errorexception.php>

Concretamente, la clase **PDO** permite definir la fórmula que usará cuando se produzca un error, utilizando el atributo `PDO::ATTR_ERRMODE`. Las posibilidades son:

- ✓ `PDO::ERRMODE_SILENT`. No se hace nada cuando ocurre un error. Es el comportamiento por defecto.
- ✓ `PDO::ERRMODE_WARNING`. Genera un error de tipo `E_WARNING` cuando se produce un error.
- ✓ `PDO::ERRMODE_EXCEPTION`. Cuando se produce un error lanza una excepción utilizando el manejador propio `PDOException`.

Es decir, que si quieres utilizar excepciones con la extensión **PDO**, debes configurar la conexión haciendo:

```
$dwes->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

Por ejemplo, el siguiente código:

```
$dwes = new PDO("mysql:host=localhost; dbname=dwes", "dwes", "abc123.");
$dwes->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
try {
    $sql = "SELECT * FROM stox";
    $result = $dwes->query($sql);
    ...
}
catch (PDOException $p) {
    echo "Error ". $p->getMessage(). "<br />";
}
```

Captura la excepción que lanza PDO debido a que la tabla no existe. El bloque `catch` muestra el siguiente mensaje:

```
Error SQLSTATE[42S02]: Base table or view not found: 1146 Table 'dwes.stox' doesn't exist
```

Agrega control de excepciones para controlar los posibles errores de conexión que se puedan producir en el último ejercicio, mostrando en la parte inferior de la pantalla los errores que se produzcan.

Revisa en la solución propuesta los cambios realizados para utilizar excepciones en el establecimiento de la conexión

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 3 : Trabajar con bases de datos en PHP -->
<!-- Ejemplo: Utilización de excepciones en PDO -->
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title>Ejercicio Tema 3: Excepciones en PDO</title>
<link href="dwes.css" rel="stylesheet" type="text/css">
</head>
<body>
<?php
    if (isset($_POST['producto'])) $producto = $_POST['producto'];
    try {
        $dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
        $dwes->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }
    catch (PDOException $e) {
        $error = $e->getCode();
        $mensaje = $e->getMessage();
    }
    if (!isset($error)) {
        // Comprobamos si tenemos que actualizar los valores
        if (isset($_POST['actualiz'])) {
            // Preparamos la consulta
            $tienda = $_POST['tienda'];
```

```

        $unidades = $_POST['unidades'];
        $sql = "UPDATE stock SET unidades=:unidades WHERE tienda=:tienda";
        $sql .= " AND producto='$_producto'";
        $consulta=$dwes->prepare($sql);
        // La ejecutamos dentro de un bucle, tantas veces como tiendas haya
        for($i=0;$i<count($tienda);$i++) {
            $consulta->bindParam(":unidades", $unidades[$i]);
            $consulta->bindParam(":tienda", $tienda[$i]);
            $consulta->execute();
        }
        $mensaje = "Se han actualizado los datos.";
    }
}
?>
<div id="encabezado">
    <h1>Ejercicio: Utilización de excepciones en PDO</h1>
    <form id="form_seleccion" action="." method="post">
        <span>Producto: </span>
        <select name="producto">
            <?php
                if (!isset($error)) {
                    // Rellenamos el desplegable con los datos de todos los productos
                    $sql = "SELECT cod, nombre corto FROM producto";
                    $resultado = $dwes->query($sql);
                    if($resultado) {
                        $row = $resultado->fetch();
                        while ($row != null) {
                            echo "<option value='{$row['cod']}'>";
                            // Si se recibió un código de producto lo seleccionamos
                            // en el desplegable usando selected='true'
                            if (isset($producto) && $producto == $row['cod'])
                                echo " selected='true'";
                            echo ">{$row['nombre corto']}</option>";
                            $row = $resultado->fetch();
                        }
                    }
                }
            <?php
            </select>
            <input type="submit" value="Mostrar stock" name="enviar"/>
        </form>
    </div>
    <div id="contenido">
        <h2>Stock del producto en las tiendas:</h2>
        <?php
            // Si se recibió un código de producto y no se produjo ningún error
            // mostramos el stock de ese producto en las distintas tiendas
            if (!isset($error) && isset($producto)) {
                // Ahora necesitamos también el código de tienda
                $sql = <<<SQL
                    SELECT tienda.cod, tienda.nombre, stock.unidades
                    FROM tienda INNER JOIN stock ON tienda.cod=stock.tienda
                    WHERE stock.producto='$_producto'
SQL;

                $resultado = $dwes->query($sql);
                if($resultado) {
                    // Creamos un formulario con los valores obtenidos
                    echo '<form id="form_actualiz" action="." method="post">';
                    $row = $resultado->fetch();
                    while ($row != null) {
                        // Metemos ocultos el código de producto y los de las tiendas
                        echo "<input type='hidden' name='producto' value='$_producto'/>";
                        echo "<input type='hidden' name='tienda[]' value='".$row['cod']."'>";
                        echo "<p>Tienda {$row['nombre']}: ";
                        // El número de unidades ahora va en un cuadro de texto
                        echo "<input type='text' name='unidades[]' size='4' ";
                        echo "value='".$row['unidades']."'> unidades.</p>";
                        $row = $resultado->fetch();
                    }
                    echo "<input type='submit' value='Actualizar' name='actualiz'/>";
                    echo "</form>";
                }
            }
        <?php
    </div>
    <div id="pie">
        <?php

```

```

        if (isset($error)) echo "<p>Se ha producido un error! $mensaje</p>";
        else {
            echo $mensaje;
            unset($dwes);
        }
    }
    ?>
</div>
</body>
</html>

```

¿Cuántos bloques "`catch`" se han de utilizar después de un bloque "`try`"?



Uno.



Uno o más.

se ejecutará el bloque "`catch`" cuyo manejador coincida que se ha lanzado mediante "`throw`". Esto sólo tiene sentido si se utilizan distintos manejadores extendiendo el base que incluye PHP, "`Exception`".

Intenta resolver la siguiente actividad relativa a las extensiones para manejo bases de datos MySQL y al control de errores.

Extensión mysqli	
Si utilizas mysqli para establecer una conexión, ¿qué método usaras para utilizar otra base de datos?	<code>select_db</code>
¿Cuál es la forma procedimental para establecer conexiones utilizando mysqli?	<code>mysqli_result</code>
En mysqli, ¿de qué clase es el objeto que devuelve una llamada al método query?	<code>mysqli_connect</code>
En InnoDB por defecto cada consulta se incluye dentro de su propia transacción. ¿Cuál es el nombre de la función de mysqli que permite cambiar este comportamiento?	<code>mysqli_autocommit</code>
¿Qué clase debes utilizar si quieres ejecutar consultas preparadas en mysqli?	<code>mysqli_stmt</code>
En mysqli, ¿cuál debe ser el valor del parámetro del método fetch_array para que devuelva un array asociativo?	<code>MYSQLI_ASSOC</code>

PDO	
¿Qué valor debemos pasarle al método fetch para que devuelva solamente un array numérico?	<code>PDO::FETCH_NUM</code>
¿Qué método se utiliza en PDO para obtener información del estado de la conexión?	<code>getAttribute</code>
¿Qué método se utiliza en PDO para indicar que se quiere comenzar una nueva transacción?	<code>beginTransaction</code>
¿Qué método se utiliza en PDO para ejecutar una consulta de tipo DELETE?	<code>exec</code>

Errores y manejo de excepciones	
Instrucción que debe seguir a un bloque try.	<code>catch</code>
Nombre de la función de PHP que permite establecer un manejador de errores propio.	<code>set_error_handler</code>
Clase base para manejar excepciones en PHP.	<code>exception</code>
Parámetro de php.ini que indica qué errores se notificarán.	<code>error_reporting</code>

TEMA 4

Contenido

1.- Autenticación de usuarios y control de acceso.....	1
1.1.- Mecanismos de autenticación (I).....	2
1.1.1.- Mecanismos de autenticación (II).....	3
1.2.- Incorporación de métodos de autenticación a una aplicación web.....	5
2.- Cookies.....	7
3.- Manejo de sesiones.....	10
3.1.- Configuración.....	11
3.2.- Inicio y fin de una sesión.....	12
3.3.- Gestión de la información de la sesión (I).....	14
3.3.1.- Gestión de la información de la sesión (II).....	16
3.3.2.- Gestión de la información de la sesión (III).....	19
3.3.3.- Gestión de la información de la sesión (IV).....	21
4.- Herramientas para depuración de código.....	24
4.1.- Instalación de herramientas de depuración.....	25
4.2.- Depuración de código en PHP.....	26

Desarrollo de aplicaciones web con PHP.

Caso práctico

Va siendo hora de comenzar a dar pasos firmes en el desarrollo del nuevo proyecto, y en BK Programación ultiman los preparativos y se toman las últimas decisiones para establecer el método que seguirán en su desarrollo.

Mientras, **Carlos** revisa los conocimientos adquiridos y decide hacer una lista con todo lo aprendido hasta el momento y otra con todo lo que aún no sabe hacer. El saldo final es bastante positivo: ya sabe utilizar variables, hacer llamadas a funciones, estructurar el código, utilizar formularios web, trabajar con bases de datos... ¡Incluso ha diseñado una pequeña página para gestionar su colección de comics!

Entre lo que no sabe hacer todavía hay algo que le llama la atención: no sabe cómo mantener los datos entre las distintas páginas de una aplicación. Es decir, la única forma que conoce es utilizando un formulario web..., o bases de datos. Pero tiene que haber otras maneras más adecuadas. Algún modo sencillo de mantener información del usuario dentro de la aplicación web.

De nuevo tendrá que pedir consejo a **Juan**.

1.- Autenticación de usuarios y control de acceso.

Caso práctico

Tal y como **Esteban** les ha explicado, el objetivo principal del proyecto no es crear una página web pública, con información sobre la empresa. Necesitan una aplicación web con un objetivo más específico: permitir a clientes y a empleados conocer información sobre los productos de la empresa.

Juan sabe que con estas condiciones, uno de los puntos fundamentales con los que deberá tratar en el nuevo proyecto es el control de acceso a la aplicación web. Todos los usuarios que accedan habrán de identificarse para poder acceder a las páginas del sitio web. Además, en función de si el usuario es un cliente o un empleado, habrá que darle acceso a una o a otra información.

Juan nunca ha programado sitios web con autenticación de los usuarios. Además, se encuentra terminando otro proyecto, y no dispone de demasiado tiempo. Por este motivo, le pide a **Carlos** que se documente sobre el tema para poder decidir el camino a tomar.

Muchas veces es importante verificar la identidad de los dos extremos de una comunicación. En el caso de una comunicación web, existen métodos para identificar tanto al servidor en el que se aloja el sitio web, como al usuario del navegador que se encuentra en el otro extremo.

Los sitios web que necesitan emplear identificación del servidor, como las tiendas o los bancos, utilizan el protocolo HTTPS. Este protocolo requiere de un certificado válido, firmado por una autoridad confiable, que es verificado por el navegador cuando se accede al sitio web. Además, HTTPS utiliza métodos de cifrado para crear un canal seguro entre el navegador y el servidor, de tal forma que no se pueda interceptar la información que se transmite por el mismo.

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.

En la unidad anterior aprendiste a utilizar aplicaciones web para gestionar información almacenada en bases de datos. En la mayoría de los casos es importante implantar en este tipo de aplicaciones web, las que acceden a bases de datos, algún mecanismo de control de acceso que

obligue al usuario a identificarse. Una vez identificado, se puede limitar el uso que puede hacer de la información.

Así, puede haber sitios web en los que los usuarios autenticados pueden utilizar sólo una parte de la información (como los bancos, que permiten a sus clientes acceder únicamente a la información relativa a sus cuentas). Otros sitios web necesitan separar en grupos a los usuarios autenticados, de tal forma que la información a la que accede un usuario depende del grupo en que éste se encuentre. Por ejemplo, una aplicación de gestión de una empresa puede tener un grupo de usuarios a los que permite visualizar la información, y otro grupo de usuarios que, además de visualizar la información, también la pueden modificar.

Debes distinguir la autenticación de los usuarios y el control de acceso, de la utilización de mecanismos para asegurar las comunicaciones entre el usuario del navegador y el servidor web. Aunque ambos aspectos suelen ir unidos, son independientes.

En los ejemplos de esta unidad, la información de autenticación (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.** Sin embargo, la configuración de servidores web que permitan usar el protocolo HTTPS para cifrar la información que reciben y transmiten no forma parte de los contenidos de este módulo. Por este motivo, durante esta unidad utilizaremos únicamente el protocolo no seguro HTTP.

1.1.- Mecanismos de autenticación (I).

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).
- ✓ El navegador recibe el error y abre una ventana para solicitar al usuario que se autentique mediante su nombre y contraseña.
- ✓ La información de autenticación 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.



En el servidor web Apache, el que has estado usando en anteriores unidades, 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; aun así, es conveniente crear este fichero en un lugar no accesible por los usuarios del servidor web.

<http://httpd.apache.org/docs/2.0/es/howto/auth.html>

Por ejemplo, para crear el fichero de usuario y añadirle el usuario "dwes", puedes hacer:

```
sudo htpasswd -c users dwes
```

e introducir la contraseña correspondiente a ese usuario.

La opción `-c` indica que se debe crear el fichero, por lo que solo deberás usarla cuando introduzcas el primer usuario y contraseña. Fíjate que en el ejemplo anterior, el fichero se crea en la ruta `/etc/apache2/users`, que en principio no es accesible vía web.

```

sml@ubuntu-profe: /etc/apache2
Archivo Editar Ver Terminal Ayuda
sml@ubuntu-profe:/etc/apache2$ sudo htpasswd -c users dwes
New password:
Re-type new password:
Adding password for user dwes
sml@ubuntu-profe:/etc/apache2$

```

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

```

El significado de cada una de las directivas anteriores es el siguiente:

Directiva	Significado
AuthName	Nombre de dominio que se usará en la autenticación. Si el cliente se autentifica correctamente, esa misma información de autenticación se utilizará automáticamente en el resto de las páginas del mismo dominio.
AuthType	Método de autenticación que se usará. Además del método Basic, Apache también permite utilizar el método Digest.
AuthUserFile	Ruta al archivo de credenciales que has creado con htpasswd.
Require	Permite indicar que sólo puedan acceder algunos usuarios o grupos de usuarios concretos. Si indicamos "valid-user", podrán acceder todos los usuarios que se autentifiquen correctamente.

Además tendrás que asegurarte de que en la configuración de Apache se utiliza la directiva `AllowOverride` para que se aplique correctamente la configuración que figura en los ficheros `.htaccess`.

<http://httpd.apache.org/docs/2.0/es/mod/core.html#allowoverride>

La sentencia `sudo htpasswd -c users admin` añade un nuevo usuario con nombre "admin" al fichero "users" que hemos creado anteriormente.



Verdadero



Falso

Al incluir la opción `-c` lo que hacemos es crear un nuevo fichero, con lo cual eliminamos el contenido anterior del mismo.

1.1.1.- Mecanismos de autenticación (II).

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

Valor	Contenido
<code>\$_SERVER['PHP_AUTH_USER']</code>	Nombre de usuario que se ha introducido.
<code>\$_SERVER['PHP_AUTH_PW']</code>	Contraseña introducida.
<code>\$_SERVER['AUTH_TYPE']</code>	Método HTTP usado para autenticar. Puede ser Basic o Digest.

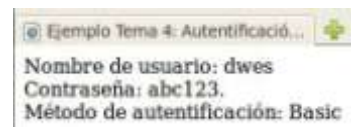
Es decir, que si creas una página web que muestre los valores de estas variables, y preparas el servidor web para utilizar autenticación HTTP, cuando accedas a esa página con el usuario "dwes" obtendrás algo como lo siguiente:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 4 : Desarrollo de aplicaciones web con PHP -->
<!-- Ejemplo: Autenticación HTTP -->
<html>

```

```
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title>Ejemplo Tema 4: Autenticación HTTP</title>
<link href="dwes.css" rel="stylesheet" type="text/css">
</head>
<body>
<?php
echo "Nombre de usuario: ".$_SERVER['PHP_AUTH_USER']."<br />";
echo "Contraseña: ".$_SERVER['PHP_AUTH_PW']."<br />";
echo "Método de autenticación: ".$_SERVER['AUTH_TYPE']."<br />";
?>
</body>
</html>
```



Ejemplo Tema 4: Autenticación...
Nombre de usuario: dwes
Contraseña: abc123.
Método de autenticación: Basic

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. En su lugar, puedes añadir las siguientes líneas en tus páginas PHP:

```
<?php
if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic Realm="Contenido restringido"');
    header('HTTP/1.0 401 Unauthorized');
    echo "Usuario no reconocido!";
    exit;
}
?>
```

La función `header` envía encabezados HTTP (bloque de datos que forma parte del protocolo HTTP y se envía antes de la información propia que se transmite. Permite especificar códigos de estado, acciones requeridas al servidor, o el tipo de información que se transmite), pero debe utilizarse antes de que se muestre nada por pantalla. En caso contrario, obtendrás un error.

<http://es.php.net/manual/es/function.header.php>

Con el código anterior, la página envía un error 401, lo que fuerza al navegador a solicitar las credenciales de acceso (nombre de usuario y contraseña). Si se introducen, se ejecuta el resto de la página y se muestra su contenido. En este caso, **habría que añadir algún código para comprobar que el nombre de usuario y contraseña son válidos**, tal y como veremos a continuación. Si se pulsa el botón "Cancelar", se muestra el mensaje de error que se indica.

Modifica la página anterior utilizando la función `header` para que solicite las credenciales al usuario.

Tendrás que crear una página similar a la anterior, y añadir el código para forzar el error 401 antes de cualquier otro.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 4 : Desarrollo de aplicaciones web con PHP -->
<!-- Ejemplo: Función header para autenticación HTTP -->
<?php
if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic Realm="Contenido restringido"');
    header('HTTP/1.0 401 Unauthorized');
    echo "Usuario no reconocido!";
    exit;
}
?>
<html>
<head>
```

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title>Ejercicio: Función header para autenticación HTTP</title>
<link href="dwes.css" rel="stylesheet" type="text/css">
</head>
<body>
<?php
echo "Nombre de usuario: ".$_SERVER['PHP_AUTH_USER']."<br />";
echo "Contraseña: ".$_SERVER['PHP_AUTH_PW']."<br />";
?>
</body>
</html>
```

1.2.- Incorporación de métodos de autenticación a una aplicación web.

Si utilizas la función `header` para forzar al navegador a solicitar credenciales HTTP, el usuario introducirá un nombre y una contraseña. Pero el servidor no verificará esta información; deberás ser tú quien provea un método para comprobar que las credenciales que ha introducido el usuario son correctas.

El método más simple es incluir en el código PHP de tu página las sentencias necesarias para comparar los datos introducidos con otros datos fijos. Por ejemplo, para permitir el acceso a un usuario "dwes" con contraseña "abc123.", puedes hacer:

```
<?php
if ($_SERVER['PHP_AUTH_USER'] != 'dwes' || $_SERVER['PHP_AUTH_PW'] != 'abc123.') {
    header('WWW-Authenticate: Basic Realm="Contenido restringido"');
    header('HTTP/1.0 401 Unauthorized');
    echo "Usuario no reconocido!";
    exit;
}
?>
```

Recuerda que el código PHP no se envía al navegador, por lo que la información de autenticación que introduzcas en el código no será visible por el usuario. Sin embargo, esto hará tu código menos portable. Si necesitas modificar el nombre de usuario o la contraseña, tendrás que hacer modificaciones en el código. Además, no podrás permitir al usuario introducir su propia contraseña. Una solución mejor es utilizar un almacenamiento externo para los nombres de usuario y sus contraseñas. Para esto podrías emplear un fichero de texto, o mejor aún, una base de datos. La información de autenticación podrá estar aislada en su propia base de datos, o compartir espacio de almacenamiento con los datos que utilice tu aplicación web.

Si quieres almacenar la información de los usuarios en la base de datos "dwes", tienes que crear una nueva tabla en su estructura. Para ello, revisa y ejecuta estas sentencias SQL.

```
-- Seleccionamos la base de datos
USE dwes;

-- Creamos la tabla
CREATE TABLE usuarios (
    usuario VARCHAR(20) NOT NULL PRIMARY KEY,
    contrasena VARCHAR(32) NOT NULL
) ENGINE = INNODB;

-- Creamos el usuario dwes
INSERT INTO usuarios (usuario, contrasena) VALUES
('dwes', 'e8dc8ccd5e5f9e3a54f07350ce8a2d3d');
```

Aunque se podrían almacenar las contraseñas en texto plano, es mejor hacerlo en formato encriptado. En el ejemplo anterior, para el usuario "dwes" se almacena el hash MD5 (*método para generar un resumen de un texto o un documento, de tal forma que a partir del resumen obtenido no es posible recuperar el texto original, ni hallar otro texto a partir del cual se obtenga el mismo resumen. Se llama hash al resumen obtenido al aplicar una función hash. Una de las funciones hash más extendidas es MD5, que genera 128 bits como resumen (normalmente se representa mediante una cadena de texto de 28 caracteres o mediante 32 dígitos hexadecimales)*) correspondiente a la contraseña "abc123.". En PHP puedes usar la función `md5` para calcular el hash MD5 de una cadena de texto.

<http://es.php.net/manual/es/function.md5.php>

Utiliza la extensión MySQLi para modificar el ejercicio anterior, de tal forma que las credenciales del usuario se comprueben con la información de la nueva tabla "usuarios" creada en la base de datos "dwes". Si no existe el usuario, o la contraseña es incorrecta, vuelve a pedir las credenciales al usuario.

Revisa la solución propuesta. Fíjate que se debe usar la función md5 para comprobar la contraseña. Si introduces un usuario o contraseña incorrectos, el comportamiento depende del navegador que utilices; algunos te pedirán las credenciales de forma indefinida, y otros un número limitado de veces.

```
<?php
// Si el usuario aún no se ha autenticado, pedimos las credenciales
if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="Contenido restringido"');
    header("HTTP/1.0 401 Unauthorized");
    exit;
}
// Si ya ha enviado las credenciales, las comprobamos con la base de datos
else {
    // Conectamos a la base de datos
    @ $dwes = new mysqli("localhost", "dwes", "abc123.", "dwes");
    $error = $dwes->connect_errno;
    // Si se estableció la conexión
    if ($error == null) {
        // Ejecutamos la consulta para comprobar si existe
        // esa combinación de usuario y contraseña
        $sql = "SELECT usuario FROM usuarios
                WHERE usuario='$_SERVER[PHP_AUTH_USER]' AND
                contrasena=md5('$_SERVER[PHP_AUTH_PW]')";
        $resultado = $dwes->query($sql);
        // Si no existe, se vuelven a pedir las credenciales
        if ($resultado->num_rows == 0) {
            header('WWW-Authenticate: Basic realm="Contenido restringido"');
            header("HTTP/1.0 401 Unauthorized");
        }
        exit;
    }
    $resultado->close();
    $dwes->close();
}
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 4 : Desarrollo de aplicaciones web con PHP -->
<!-- Ejemplo: Utilización de MySQL para autenticación HTTP -->
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title>Ejemplo Tema 4: Utilización de MySQL para autenticación HTTP</title>
<link href="dwes.css" rel="stylesheet" type="text/css">
</head>
<body>
<?php
echo "Nombre de usuario: " . $_SERVER['PHP_AUTH_USER'] . "<br />";
echo "Hash de la contraseña: " . md5($_SERVER['PHP_AUTH_PW']) . "<br />";
?>
</body>
</html>
```

Las dos posibilidades que hemos visto para solicitar al usuario que se autentique vía HTTP son la creación del fichero .htaccess, y la utilización de la función header de PHP. ¿Cuál de esas dos formas será preferible si quieres que los privilegios de acceso a tu aplicación varíen en función del día de la semana (por ejemplo, unos usuarios que puedan acceder de lunes a viernes y otros distintos el fin de semana)?



El fichero .htaccess



La función header

Si utilizas la función header para enviar los encabezados HTTP, debes escribir tú el código que decide si el usuario se ha autenticado correctamente y se le permite acceder o no. Por tanto, puedes incluir las reglas adicionales que quieras, por ejemplo, darle o no acceso en función del día de la semana.

2.- Cookies.

Caso práctico

Una vez resuelto el tema de la autenticación, el siguiente objetivo de **Carlos** es aprender a gestionar las cookies. **Juan** le ha explicado qué son y cómo funcionan, y seguramente las tengan que utilizar en la aplicación que están desarrollando.

Sabe que muchos de los sitios web que visita las utilizan, porque ha probado a desactivarlas en su navegador, y le han empezado a aparecer mensajes pidiéndole que las vuelva a activar. Se ha propuesto averiguar para qué las utilizan, y cómo las podrán gestionar desde PHP.

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 indicárselas la próxima vez que visite el sitio.

Si utilizas Firefox como navegador, puedes instalar la extensión Web Developer para obtener información sobre las páginas web que visitas. Entre sus características te permite consultar y editar las cookies almacenadas en el mismo.

<https://addons.mozilla.org/es/firefox/addon/web-developer/>

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 quieres 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_usuario", $_SERVER['PHP_AUTH_USER'], 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.

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`.

Siempre que utilices cookies en una aplicación web, debes tener en cuenta que en última instancia su disponibilidad está controlada por el cliente. Por ejemplo, algunos usuarios deshabilitan las cookies en el navegador porque piensan que la información que almacenan puede suponer un potencial problema de seguridad. O la información que almacenan puede llegar a perderse porque el usuario decide formatear el equipo o simplemente eliminarlas de su sistema.

Si una vez almacenada una cookie en el navegador quieres eliminarla antes de que expire, puedes utilizar la misma función `setcookie` pero indicando una fecha de caducidad anterior a la actual.

Sobre el mismo ejercicio anterior, almacena en una cookie el último instante en que el usuario visitó la página. Si es su primera visita, muestra un mensaje de bienvenida. En caso contrario, muestra la fecha y hora de su anterior visita.

Revisa la solución propuesta. Deberás utilizar la función `setcookie` para guardar el instante de su anterior visita y mostrar su contenido utilizando el array `$_COOKIE`.

```
<?php
// Si el usuario aún no se ha autenticado, pedimos las credenciales
if( !isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="Contenido restringido"');
    header("HTTP/1.0 401 Unauthorized");
    exit;
}
// Si ya ha enviado las credenciales, las comprobamos con la base de datos
else {
    // Conectamos a la base de datos
    @ $dwes = new mysqli("localhost", "dwes", "abc123.", "dwes");
    $error = $dwes->connect_errno;
    // Si se estableció la conexión
    if ($error == null) {
        date_default_timezone_set('Europe/Madrid');
        // Ejecutamos la consulta para comprobar si existe
        // esa combinación de usuario y contraseña
        $sql = "SELECT usuario FROM usuarios
        WHERE usuario='{$_SERVER['PHP_AUTH_USER']}' AND
        contraseña=md5('{$_SERVER['PHP_AUTH_PW']}')";
        $resultado = $dwes->query($sql);
        // Si no existe, se vuelven a pedir las credenciales
        if($resultado->num rows == 0) {
            header('WWW-Authenticate: Basic realm="Contenido restringido"');
            header("HTTP/1.0 401 Unauthorized");
            exit;
        } else {
            if (isset($_COOKIE['ultimo login'])) {
                $ultimo_login = $_COOKIE['ultimo login'];
            }
            setcookie("ultimo_login", time(), time()+3600);
        }
        $resultado->close();
        $dwes->close();
    }
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 4 : Desarrollo de aplicaciones web con PHP -->
<!-- Ejemplo: Cookies en autenticación HTTP -->
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title>Ejemplo Tema 4: Cookies en autenticación HTTP</title>
<link href="dwes.css" rel="stylesheet" type="text/css">
</head>
<body>
<?php
    if ($error == null) {
        echo "Nombre de usuario: ".$SERVER['PHP AUTH USER']."<br />";
        echo "Hash de la contraseña: ".md5($SERVER['PHP AUTH PW'])."<br />";
        if (isset($ultimo_login))
            echo "Ultimo login: " . date("d/m/y \a \l\ a\s H:i", $ultimo_login);
        else
            echo "Bienvenido. Esta es su primera visita.";
    }
    else
        echo "Se ha producido el error $error.<br />";
?>
</body>
</html>
```

¿Cuál es la duración por defecto de una cookie si no se indica la fecha de caducidad, como en la siguiente llamada a la función setcookie?

```
setcookie("idioma", "español");
```



Hasta que se cierre el navegador del usuario.



1 hora.

Cuando se acaba la sesión del usuario, al cerrar el navegador, se borra la cookie.

3.- Manejo de sesiones.

Caso práctico

Carlos ha aprendido a utilizar las cookies, y cuáles son sus limitaciones. Está claro que les serán de utilidad, pero también que no cubren las necesidades de una aplicación web. Al viajar en las cabeceras HTTP, la información que pueden mantener está muy limitada.

*Aunque sabe que aún le queda mucho por aprender, **Juan** le ha comentado que con los conocimientos que acaba de adquirir ya tiene una buena base para dar los primeros pasos programando en PHP. Dice que sólo le falta aprender cómo funcionan las sesiones para empezar a desarrollar aplicaciones web completas por su cuenta.*

*Animado por estos comentarios, **Carlos** decide dar un paso más. Veamos cómo funcionan las sesiones en PHP.*

Como acabas de ver, una forma para guardar información particular de cada usuario es utilizar cookies. Sin embargo, existen diversos problemas asociados a las cookies, como el número de ellas que admite el navegador, o su tamaño máximo. Para solventar estos inconvenientes, existen las sesiones. El término sesión hace referencia al conjunto de información relativa a un usuario concreto. Esta información puede ser tan simple como el nombre del propio usuario, o más compleja, como los artículos que ha depositado en la cesta de compra de una tienda online.

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. 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. Esa información se almacena en el servidor web, generalmente en ficheros aunque también se pueden utilizar otros mecanismos de almacenamiento como bases de datos.

Como ya habrás supuesto, la cuestión ahora es: ¿y dónde se almacena ese SID, el identificador de la sesión, que es único para cada usuario? Pues existen dos maneras de mantener el SID entre las páginas de un sitio web que visita el usuario:

- ✓ Utilizando cookies, tal y como ya viste.
- ✓ Propagando el SID en un parámetro de la URL. El SID se añade como una parte más de la URL, de la forma:

<http://www.misitioweb.com/tienda/listado.php&PHPSESSID=34534fg4ffg34ty>

En el ejemplo anterior, el SID es el valor del parámetro `PHPSESSID`.

Ninguna de las dos maneras es perfecta. Ya sabes los problemas que tiene la utilización de cookies. Pese a ello, es el mejor método y el más utilizado. Propagar el SID como parte de la URL conlleva mayores desventajas, como la imposibilidad de mantener el SID entre distintas sesiones, o el hecho de que compartir la URL con otra persona implica compartir también el identificador de sesión.

La buena noticia, es que el proceso de manejo de sesiones en PHP está automatizado en gran medida. Cuando un usuario visita un sitio web, no es necesario programar un procedimiento para ver si existe un SID previo y cargar los datos asociados con el mismo. Tampoco tienes que utilizar la función `setcookie` si quieres almacenar los SID en cookies, o ir pasando el SID entre las páginas web de tu sitio si te decides por propagarlo. Todo esto PHP lo hace automáticamente.

A la información que se almacena en la sesión de un usuario también se le conoce como cookies del lado del servidor (server side cookies). Debes tener en cuenta que aunque esta información no viaja entre el cliente y el servidor, sí lo hace el SID, bien como parte de la URL o en un encabezado HTTP si se guarda en una cookie. En ambos casos, esto plantea un posible problema de seguridad. El SID puede ser conseguido por otra persona, y a partir

del mismo obtener la información de la sesión del usuario. La manera más segura de utilizar sesiones es almacenando los SID en cookies y utilizar HTTPS para encriptar la información que se transmite entre el servidor web y el cliente.

3.1.- Configuración.

Por defecto, PHP incluye soporte de sesiones incorporado. Sin embargo, antes de utilizar sesiones en tu sitio web, debes configurar correctamente PHP utilizando las siguientes directivas en el fichero `php.ini` según corresponda:

Directiva	Significado
session.use_cookies	Indica si se deben usar cookies (1) o propagación en la URL (0) para almacenar el SID.
session.use_only_cookies	Se debe activar (1) cuando utilizas cookies para almacenar los SID, y además no quieres 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).
session.save_handler	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>mem</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.
session.name	Determina el nombre de la cookie que se utilizará para guardar el SID. Su valor por defecto es <code>PHPSESSID</code> .
session.auto_start	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.
session.cookie_lifetime	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 quieres que se mantenga el SID durante más tiempo, debes indicar en esta directiva ese tiempo en segundos.
session.gc_maxlifetime	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.

La función `phpinfo`, de la que ya hablamos con anterioridad, te ofrece información sobre la configuración actual de las directivas de sesión.

En la documentación de PHP tienes información sobre éstas y otras directivas que permiten configurar el manejo de sesiones.
<http://es.php.net/manual/es/session.configuration.php>

SESSION		
Session Support	enabled	
Registered name handlers	file user sqlite	
Registered serializer handlers	php php_serialize	
Directive	Local Value	Master Value
session.auto_start	Off	Off
session.bug_compat_42	On	On
session.bug_compat_were	On	On
session.cache_expire	180	180
session.cache_limiter	nocache	nocache
session.cookie_domain	no value	no value
session.cookie_httponly	Off	Off
session.cookie_lifetime	0	0
session.cookie_path	/	/
session.cookie_secure	Off	Off
session.cookie_samesite	no value	no value
session.cookie_length	0	0
session.gc_destroy	100	100
session.gc_maxlifetime	1440	1440
session.gc_probability	1	1
session.hash_bits_per_character	5	5
session.hash_function	0	0
session.name	PHPSESSID	PHPSESSID
session.referer_check	no value	no value
session.save_handler	file	file
session.save_path	/tmp/PHPSESSID	/tmp/PHPSESSID
session.serialize_handler	php	php
session.use_cookies	On	On
session.use_only_cookies	Off	Off
session.use_only_sid	0	0

Si la información del usuario que quieres almacenar incluye contenido privado como una contraseña, ¿qué utilizarías, cookies o la sesión del usuario?



La sesión del usuario.



Cookies.

La sesión del usuario se almacena y se procesa en el servidor web, por lo que no es necesario transmitirla hasta el ordenador del usuario, lo que aumenta su seguridad.

3.2.- Inicio y fin de una sesión.

El inicio de una sesión puede tener lugar de dos formas. 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 esta no se haya eliminado, en lugar de abrir una nueva sesión se reanudará la anterior. Para ello se utilizará el SID anterior, que estará almacenado en una cookie (recuerda que si usas propagación del SID, no podrás restaurar sesiones anteriores; el SID figura en la URL y se pierde cuando cierras el navegador).

Si por el contrario, decides no utilizar el inicio automático de sesiones, deberás ejecutar la función `session_start` para indicar a PHP que inicie una nueva sesión o reanude la anterior. Aunque anteriormente esta función devolvía siempre `true`, a partir de la versión 5.3.0 de PHP su comportamiento es más coherente: devuelve `false` en caso de no poder iniciar o restaurar la sesión.

Como el inicio de sesión requiere utilizar cookies, y éstas se transmiten en los encabezados HTTP, debes tener en cuenta que para poder iniciar una sesión utilizando `session_start`, tendrás que hacer las llamadas a esta función antes de que la página web muestre información en el navegador.

Además, todas las páginas que necesiten utilizar la información almacenada en la sesión, deberán ejecutar la función `session_start`.

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. Por ejemplo, para contar el número de veces que el usuario visita la página, puedes hacer:

```
<?php
// Iniciamos la sesión o recuperamos la anterior sesión existente
session_start();
// Comprobamos si la variable ya existe
if (isset($_SESSION['visitas']))
    $_SESSION['visitas']++;
else
    $_SESSION['visitas'] = 0;
?>
```

Si en lugar del número de visitas, quisieras almacenar el instante en que se produce cada una, la variable de sesión "visitas" deberá ser un array y por tanto tendrás que cambiar el código anterior por:

```
<?php
// Iniciamos la sesión o recuperamos la anterior sesión existente
session_start();
// En cada visita añadimos un valor al array "visitas"
$_SESSION['visitas'][] = mktime();
?>
```

Aunque como ya viste, puedes configurar PHP para que elimine de forma automática los datos de una sesión pasado cierto tiempo, en ocasiones puede ser necesario cerrarla de forma manual en un momento determinado. Por ejemplo, si utilizas sesiones para recordar la información de autenticación, deberás darle al usuario del sitio web la posibilidad de cerrar la sesión cuando lo crea conveniente.

En PHP tienes dos funciones para eliminar la información almacenada en la sesión:

- ✓ `session_unset`. Elimina las variables almacenadas en la sesión actual, pero no elimina la información de la sesión del dispositivo de almacenamiento usado.
- ✓ `session_destroy`. Elimina completamente la información de la sesión del dispositivo de almacenamiento.

Crea una página similar a la anterior, almacenando en la sesión de usuario los instantes de todas sus últimas visitas. Si es su primera visita, muestra un mensaje de bienvenida. En caso contrario, muestra la fecha y hora de todas sus visitas anteriores. Añade un botón a la página que permita borrar el registro de visitas.

Utiliza también una variable de sesión para comprobar si el usuario se ha autenticado correctamente. De esta forma no hará falta comprobar las credenciales con la base de datos constantemente.

Revisa la solución propuesta. Acuérdate de que debes hacer una llamada a la función `session_start()` antes de utilizar la variable superglobal `$_SESSION` para acceder a la información de la sesión.

```
<?php
// Si el usuario aún no se ha autenticado, pedimos las credenciales
if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="Contenido restringido"');
    header("HTTP/1.0 401 Unauthorized");
    exit;
}
// Vamos a guardar el usuario en una variable de sesión
// si no existe, aún no se ha autenticado
session_start();
if (!isset($_SESSION['usuario'])) {
    // Conectamos a la base de datos
    @ $dwes = new mysqli("localhost", "dwes", "abc123.", "dwes");
    $error = $dwes->connect_errno;
    // Si se estableció la conexión
    if ($error == null) {
        // Ejecutamos la consulta para comprobar si existe
        // esa combinación de usuario y contraseña
        $sql = "SELECT usuario FROM usuarios
            WHERE usuario='{$_SERVER['PHP_AUTH_USER']}' AND
            contraseña=md5('{$_SERVER['PHP_AUTH_PW']}')";
        $resultado = $dwes->query($sql);
        // Si no existe, se vuelven a pedir las credenciales
        if($resultado->num_rows == 0) {
            header('WWW-Authenticate: Basic realm="Contenido restringido"');
            header("HTTP/1.0 401 Unauthorized");
            exit;
        }else{
            $_SESSION['usuario'] = $_SERVER['PHP_AUTH_USER'];
            $resultado->close();
            $dwes->close();
        }
    }
}
// Si ya está autenticado
else {
    // Comprobamos si se ha enviado el formulario de limpiar el registro
    if (isset($_POST['limpiar']))
        unset($_SESSION['visita']);
    else
        $_SESSION['visita'][] = time();
}
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 4 : Desarrollo de aplicaciones web con PHP -->
<!-- Ejemplo: Cookies en autenticación HTTP -->
<html>
    <head>
        <meta http-equiv="content-type" content="text/html; charset=UTF-8">
        <title>Ejemplo Tema 4: Cookies en autenticación HTTP</title><link href="dwes.css"
rel="stylesheet" type="text/css">
```

```

</head>
<body>
  <?php
    if ($error == null) {
      echo "Nombre de usuario: ".$SERVER['PHP_AUTH_USER']."<br />";
      echo "Hash de la contraseña: ".md5($_SERVER['PHP_AUTH_PW'])."<br />";
      if (count($_SESSION['visita']) == 0)
        echo "Bienvenido. Esta es su primera visita.";
      else {
        date_default_timezone_set('Europe/Madrid');
        foreach($_SESSION['visita'] as $v)
          echo date("d/m/y \a \l\ a\s H:i", $v) . "<br />";
      }
    }
    <form id='vaciar' action='<?php echo $_SERVER['PHP_SELF'];?>' method='post'>
      <input type='submit' name='limpiar' value='Limpiar registro' />
    </form>
    <?php
      }
    }else
      echo "Se ha producido el error $error.<br />";
    ?>
  </body>
</html>

```

Si usamos el inicio de sesión automático, la sesión de un usuario se inicia en cuanto se autentifica correctamente en el servidor web.



Verdadero



Falso

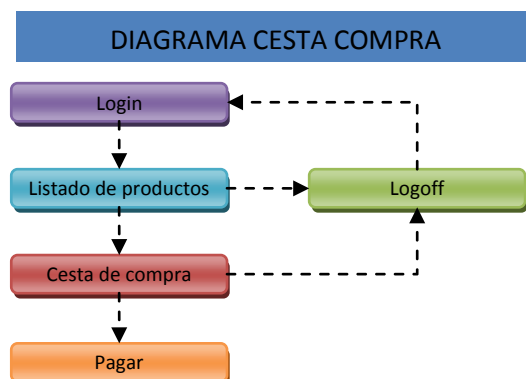
Con el inicio de sesión automático, la sesión del usuario se inicia en cuanto se conecta al sitio web, esté o no autenticado. No tiene nada que ver una cosa con otra. Un usuario no autenticado puede tener una sesión de la misma forma que otro que sí se haya autenticado correctamente

3.3.- Gestión de la información de la sesión (I).

En este punto vas a ver paso a paso un ejemplo de utilización de sesiones para almacenar la información del usuario. Utilizarás la base de datos "dwes", creada anteriormente, para crear un prototipo de una tienda web dedicada a la venta de productos de informática.

Las páginas de que constará tu tienda online son las siguientes:

- ✓ **Login** ([login.php](#)). Su función es autenticar al usuario de la aplicación web. Todos los usuarios de la aplicación deberán autenticarse utilizando esta página antes de poder acceder al resto de páginas.
- ✓ **Listado de productos** ([productos.php](#)). Presenta un listado de los productos de la tienda, y permite al usuario seleccionar aquellos que va a comprar.
- ✓ **Cesta de compra** ([cesta.php](#)). Muestra un resumen de los productos escogidos por el usuario para su compra y da acceso a la página de pago.
- ✓ **Pagar** ([pagar.php](#)). Una vez confirmada la compra, la última página debería ser la que permitiera al usuario escoger el método de pago y la forma de envío. En este ejemplo no la vas a implementar como tal. Simplemente mostrará un mensaje de tipo "Gracias por su compra" y ofrecerá un enlace para comenzar una nueva compra.
- ✓ **Logoff** ([logoff.php](#)). Esta página desconecta al usuario de la aplicación y redirige al usuario de forma automática a la pantalla de autenticación. No muestra ninguna información en pantalla, por lo que no es visible para el usuario.



Recuerda poner a las páginas los nombres que aquí figuran, almacenando todas en la misma carpeta. Si cambias algún nombre o mueves alguna página de lugar, los enlaces internos no funcionarán.

Aunque el aspecto de la aplicación no es importante para nuestro objetivo, utilizaremos una hoja de estilos, `tienda.css`, común a todas las páginas y una imagen, `cesta.png`, para ofrecer un interface más amigable.

```
.error {
  font-family: Verdana, Arial, sans-serif;
  font-size: 0.7em;
  color: #900;
  background-color : #ffff00;
}

.divisor {
  clear:both;
  height:0;
  font-size: 1px;
  line-height: 0px;
}

#login fieldset {
  position: absolute;
  left: 50%;
  top: 50%;
  width: 230px;
  margin-left: -115px;
  height: 160px;
  margin-top: -80px;
  padding:10px;
  border:1px solid #ccc;
  background-color: #eee;
}

#login legend {
  font-family : Arial, sans-serif;
  font-size: 1.3em;
  font-weight:bold;
  color:#333;
}

#login .campo {
  margin-top:8px;
  margin-bottom: 10px;
}

#login label {
  font-family : Arial, sans-serif;
  font-size:0.8em;
  font-weight: bold;
}

#login input[type="text"], #login
input[type="password"] {
  font-family : Arial, Verdana, sans-serif;
  font-size: 0.8em;
  line-height:140%;
  color : #000;
  padding : 3px;
  border : 1px solid #999;
  height:18px;
  width:220px;
}

#login input[type="submit"] {
  width:100px;
  height:30px;
  padding-left:0px;
}

.pagproductos body, .pagcesta body {
  font: 100% Verdana, Arial, Helvetica,
  sans-serif;
  background: #666;
```

```
margin: 0;
padding: 0;
text-align: center;
color: #000000;
}

#contenedor {
  width: 90%;
  background: #fff;
  margin: 0 auto;
  border: 1px solid #000;
  text-align: left;
}

#encabezado {
  padding: 0 10px;
  border-top-width: thin;
  border-right-width: thin;
  border-bottom-width: thin;
  border-left-width: thin;
  border-top-style: solid;
  border-right-style: solid;
  border-bottom-style: solid;
  border-left-style: solid;
  background-color: #9cf;
}

#encabezado h1 {
  margin: 0;
  padding: 10px 0;
}

.pagproductos #cesta {
  float: right;
  width: 12em;
  background-color: #588;
}

.pagproductos #cesta h3 {
  margin-left: 10px;
  margin-right: 10px;
}

.pagproductos #cesta p {
  margin-left: 10px;
  margin-right: 10px;
  font-size: 10px;
}

.pagproductos #cesta input[type="submit"] {
  margin-left: 10px;
  margin-right: 10px;
  margin-bottom: 3px;
  width:100px;
  height:30px;
  padding-left:0px;
}

.pagproductos #productos {
  margin: 0 10em 0 10px;
}

.pagcesta #productos {
  margin: 10px;
  font-size: 12px;
}

.pagproductos #productos input,
.pagproductos #productos p {
  font-size: 10px;
}
```

```
#productos .codigo {  
    width: 20%;  
    float: left;  
}  
  
#productos .nombre {  
    width: 60%;  
    float: left;  
}  
  
#productos .precio {  
    width: 20%;  
    text-align: right;  
    font-weight: bold;
```

```
}  
  
#pie {  
    padding: 0 10px;  
    background-color: #99ccff;  
    border-top-width: thin;  
    border-right-width: thin;  
    border-bottom-width: thin;  
    border-left-width: thin;  
    border-top-style: solid;  
    border-right-style: solid;  
    border-bottom-style: solid;  
    border-left-style: solid;  
}
```



cesta.png

Antes de comenzar ten en cuenta que la aplicación que vas a desarrollar no es completamente funcional. Además de no desarrollar la página con la información de pago, habrá algunas opciones que no tendrás en cuenta para simplificar el código. Por ejemplo:

- ✓ No tendrás en cuenta la posibilidad de que el usuario compre varias unidades de un mismo producto.
- ✓ Una vez añadido un producto a la cesta de compra, no se podrá retirar de la misma. La única posibilidad será vaciar toda la cesta y comenzar de nuevo añadiendo productos.
- ✓ No se mostrarán imágenes de los productos, ni será posible ver el total de la compra hasta que ésta haya finalizado.
- ✓ Se muestran todos los productos en una única página. Sería preferible filtrarlos por familia y mostrarlos en varias páginas, limitando a 10 o 20 productos el número máximo de cada página.

Aunque reduzcamos en este ejemplo la funcionalidad de la tienda, te animamos a que una vez finalizado el mismo, añadas por tu cuenta todas aquellas opciones que quieras. Recuerda que la mejor forma de aprender programación es... ¡programando!

3.3.1.- Gestión de la información de la sesión (II).

La primera página que vas a programar es la de autenticación del usuario (`login.php`). Para variar, utilizarás las capacidades de manejo de sesiones de PHP para almacenar la identificación de los usuarios. Además, utilizaremos la información de la tabla "`usuarios`" en la base de datos "`dwes`", accediendo mediante PDO en lugar de MySQLi.

Un formulario web con el título "Login". Contiene dos campos de entrada: "Usuario:" y "Contraseña:". Debajo de los campos hay un botón que dice "Enviar".

Vas a crear en la página un formulario con dos campos, uno de tipo text para el usuario, y otro de tipo password para la contraseña. Al pulsar el botón Enviar, el formulario se enviará a esta misma página, donde se compararán las credenciales proporcionadas por el usuario con las almacenadas en la base de datos. Si los datos son correctos, se iniciará una nueva sesión y se almacenará en ella el nombre del usuario que se acaba de conectar.

Vamos por pasos. El código HTML para crear el formulario, que irá dentro del cuerpo de la página (entre las etiquetas `<body>`) será el siguiente:

```
<div id='login'>  
<form action='login.php' method='post'>  
<fieldset >  
    <legend>Login</legend>
```

```

<div><span class='error'><?php echo $error; ?></span></div>
<div class='campo'>
    <label for='usuario' >Usuario:</label><br/>
    <input type='text' name='usuario' id='usuario' maxlength="50" /><br/>
</div>
<div class='campo'>
    <label for='password' >Contraseña:</label><br/>
    <input type='password' name='password' id='password' maxlength="50" /><br/>
</div>
<div class='campo'>
    <input type='submit' name='enviar' value='Enviar' />
</div>
</fieldset>
</form>
</div>

```

Fíjate que existe un espacio para poner los posibles mensajes de error que se produzcan, como la falta de algún dato necesario, o un error de credenciales erróneas.

El código PHP que debe figurar al comienzo de esta misma página (antes de que se muestre cualquier texto), se encargará de:

Comprobar que se han introducido tanto el nombre de usuario como la contraseña.

```

if (isset($_POST['enviar'])) {
    $usuario = $_POST['usuario'];
    $password = $_POST['password'];

    if (empty($usuario) || empty($password))
        $error = "Debes introducir un nombre de usuario y una contraseña";
    else {

```

Conectarse a la base de datos.

```

try {
    $opc = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8");
    $dsn = "mysql:host=localhost;dbname=dwes";
    $dwes = new PDO($dsn, "dwes", "abc123.", $opc);
}
catch (PDOException $e) {
    die("Error: " . $e->getMessage());
}

```

Comprobar las credenciales.

```

$sql = "SELECT usuario FROM usuarios WHERE usuario='$usuario' " .
    "AND contrasena='" . md5($password) . "'";

if($resultado = $dwes->query($sql)) {
    $fila = $resultado->fetch();
    if ($fila != null) {

```

Iniciar la sesión y almacenar en la variable de sesión `$_SESSION['usuario']` el nombre de usuario.

```

session_start();
$_SESSION['usuario']=$usuario;

```

Redirigir a la página del listado de productos.

```

header("Location: productos.php");

```

Revisa el código completo de la página login.php y comprueba su funcionamiento antes de seguir con las demás.

```

<?php
// Comprobamos si ya se ha enviado el formulario
if (isset($_POST['enviar'])) {
    $usuario = $_POST['usuario'];
    $password = $_POST['password'];

    if (empty($usuario) || empty($password))
        $error = "Debes introducir un nombre de usuario y una contraseña";
    else {
        // Comprobamos las credenciales con la base de datos
        // Conectamos a la base de datos
        try {

```

```

        $opc = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8");
        $dsn = "mysql:host=localhost;dbname=dwes";
        $dwes = new PDO($dsn, "dwes", "abc123.", $opc);
    }
    catch (PDOException $e) {
        die("Error: " . $e->getMessage());
    }

    // Ejecutamos la consulta para comprobar las credenciales
    $sql = "SELECT usuario FROM usuarios " .
    "WHERE usuario='$usuario' " .
    "AND contrasena='" . md5($password) . "'";

    if($resultado = $dwes->query($sql)) {
        $fila = $resultado->fetch();
        if ($fila != null) {
            session start();
            $_SESSION['usuario']=$usuario;
            header("Location: productos.php");
        }
        else {
            // Si las credenciales no son válidas, se vuelven a pedir
            $error = "Usuario o contraseña no válidos!";
        }
        unset($resultado);
    }
    unset($dwes);
}

}

?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 4 : Desarrollo de aplicaciones web con PHP -->
<!-- Ejemplo Tienda Web: login.php -->
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Ejemplo Tema 4: Login Tienda Web</title>
    <link href="tienda.css" rel="stylesheet" type="text/css">
</head>

<body>
    <div id='login'>
        <form action='login.php' method='post'>
            <fieldset >
                <legend>Login</legend>
                <div><span class='error'><?php echo $error; ?></span></div>
                <div class='campo'>
                    <label for='usuario' >Usuario:</label><br/>
                    <input type='text' name='usuario' id='usuario' maxlength="50" /><br/>
                </div>
                <div class='campo'>
                    <label for='password' >Contraseña:</label><br/>
                    <input type='password' name='password' id='password' maxlength="50" /><br/>
                </div>

                <div class='campo'>
                    <input type='submit' name='enviar' value='Enviar' />
                </div>
            </fieldset>
        </form>
    </div>
</body>
</html>

```

En el código anterior, la sesión del usuario se inicia solo si proporciona un nombre de usuario y contraseña correctos. ¿Se podría haber iniciado la sesión al principio del código, aunque el usuario no proporcione credenciales?



Sí



No

Sí, se puede iniciar la sesión de igual forma, aunque si el usuario no se autentifica no tendríamos ninguna información que incluir dentro de la misma.

3.3.2.- Gestión de la información de la sesión (III).

Cuando un usuario proporciona unas credenciales de inicio de sesión correctas (recuerda que tú ya habías añadido el usuario "dwes" con contraseña "abc123."), se le redirige de forma automática a la página del listado de productos (`productos.php`) para que pueda empezar a hacer la compra. Esa es la página que vas a programar a continuación.



```
<?php
// Recuperamos la información de la sesión
session_start();

// Y comprobamos que el usuario se haya autenticado
if (!isset($_SESSION['usuario'])) {
    die("Error - debe <a href='login.php'>identificarse</a>.<br />");
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 4 : Desarrollo de aplicaciones web con PHP -->
<!-- Ejemplo Tienda Web: productos.php -->
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Ejemplo Tema 4: Listado de Productos</title>
    <link href="tienda.css" rel="stylesheet" type="text/css">
</head>

<body class="pagproductos">

<div id="contenedor">
    <div id="encabezado">
        <h1>Listado de productos</h1>
    </div>
    <div id="cesta">
        <h3> Cesta</h3>
        <hr />
    </div>
<?php
// Comprobamos si se ha enviado el formulario de vaciar la cesta
if (isset($_POST['vaciar'])) {
    unset($_SESSION['cesta']);
}

// Comprobamos si se ha enviado el formulario de añadir
if (isset($_POST['enviar'])) {
    // Creamos un array con los datos del nuevo producto
    $producto['nombre'] = $_POST['nombre'];
    $producto['precio'] = $_POST['precio'];
    // y lo añadimos
    $_SESSION['cesta'][$_POST['producto']] = $producto;
}

// Si la cesta está vacía, mostramos un mensaje
$cesta_vacia = true;
if (count($_SESSION['cesta'])==0) {
    print "<p>Cesta vacía</p>";
}

// Si no está vacía, mostrar su contenido
else {
    foreach ($_SESSION['cesta'] as $codigo => $producto)
        print "<p>$codigo</p>";
    $cesta_vacia = false;
}
```

```

    }
?>
    <form id='vaciar' action='productos.php' method='post'>
        <input type='submit' name='vaciar' value='Vaciar Cesta'
            <?php if ($cesta vacia) print "disabled='true'"; ?>
        />
    </form>
    <form id='comprar' action='cesta.php' method='post'>
        <input type='submit' name='comprar' value='Comprar'
            <?php if ($cesta vacia) print "disabled='true'"; ?>
        />
    </form>
</div>
<div id="productos">
<?php
    try {
        $opc = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8");
        $dsn = "mysql:host=localhost;dbname=dwes";
        $dwes = new PDO($dsn, "dwes", "abc123.", $opc);
    }
    catch (PDOException $e) {
        $error = $e->getCode();
        $mensaje = $e->getMessage();
    }

    if (!isset($error)) {
        $sql = <<<SQL
SELECT cod, nombre_corto, PVP
FROM producto
SQL;
        $resultado = $dwes->query($sql);
        if($resultado) {
            // Creamos un formulario por cada producto obtenido
            $row = $resultado->fetch();
            while ($row != null) {
                echo "<p><form id='{$row['cod']}' action='productos.php' method='post'>";
                // Metemos ocultos los datos de los productos
                echo "<input type='hidden' name='producto' value='{$row['cod']}' />";
                echo "<input type='hidden' name='nombre' value='{$row['nombre corto']}' />";
                echo "<input type='hidden' name='precio' value='{$row['PVP']}' />";
                echo "<input type='submit' name='enviar' value='Añadir' />";
                echo " {$row['nombre_corto']}: ";
                echo $row['PVP']." euros.";
                echo "</form>";
                echo "</p>";
                $row = $resultado->fetch();
            }
        }
    }
?>

</div>
<br class="divisor" />
<div id="pie">
    <form action='logout.php' method='post'>
        <input type='submit' name='desconectar' value='Desconectar usuario <?php echo
$_SESSION['usuario']; ?>' />
    </form>
<?php
    if (isset($error)) {
        print "<p class='error'>Error $error: $mensaje</p>";
    }
?>
</div>
</div>
</body>
</html>

```

La página se divide en varias zonas, cada una definida por una etiqueta **< div>** en el código HTML:

- ✓ **encabezado.** Contiene únicamente el título de la página.
- ✓ **productos.** Contiene el listado de todos los productos tal y como figuran en la base de datos. Cada producto figura en una línea (nombre y precio).
Se crea un formulario por cada producto, con un botón "Añadir" que envía a esta misma página los datos código, nombre y precio del producto. Cuando se abre la página, se comprueba si se ha

enviado este formulario, y si fuera así se añade un elemento al array asociativo `$_SESSION['cesta']` con los datos del nuevo producto.

```
// Comprobamos si se ha enviado el formulario de añadir
if (isset($_POST['enviar'])) {
    // Creamos un array con los datos del nuevo producto
    $producto['nombre'] = $_POST['nombre'];
    $producto['precio'] = $_POST['precio'];
    // y lo añadimos
    $_SESSION['cesta'][$_POST['producto']] = $producto;
}
```

El array `$_SESSION['cesta']` es la variable de sesión en la que guardaremos los datos de todos los productos que va a comprar el usuario. Fíjate que los datos del nuevo producto se incluyen a su vez como un array con dos elementos, el precio y el nombre.

- ✓ **cesta.** Muestra el código de los productos que se van añadiendo a la cesta.

```
// Si la cesta está vacía, mostramos un mensaje
$cesta_vacia = true;
if (count($_SESSION['cesta'])==0) {
    print "<p>Cesta vacía</p>";
}

// Si no está vacía, mostrar su contenido
else {
    foreach ($_SESSION['cesta'] as $codigo => $producto)
        print "<p>$codigo</p>";
    $cesta_vacia = false;
}
```

Contiene dos formularios. Uno para vaciar la cesta (botón "Vaciar Cesta"), dirigido a esta misma página, y otro para realizar la compra (botón "Comprar"), que dirige a la página `cesta.php`. Para vaciar la cesta, se incluye en la página el siguiente código:

```
// Comprobamos si se ha enviado el formulario de vaciar la cesta
if (isset($_POST['vaciar'])) {
    unset($_SESSION['cesta']);
}
```

- ✓ **pie.** Contiene un botón para desconectar al usuario actual. Llama a la página `logout.php`, que borra la sesión actual.

Además, tanto en esta página como en todas las demás, es necesario comprobar la variable de sesión `$_SESSION['usuario']` para verificar que el usuario se ha autenticado correctamente. Para ello, debes incluir el siguiente código al inicio de la página.

```
<?php
// Recuperamos la información de la sesión
session_start();

// Y comprobamos que el usuario se haya autenticado
if (!isset($_SESSION['usuario'])) {
    die("Error - debe <a href='login.php'>identificarse</a>.<br />");
}
?>
```

Si el usuario no se ha autenticado, se muestra un mensaje de error junto con un enlace a la página `login.php`.

3.3.3.- Gestión de la información de la sesión (IV).

Si desde la página del listado de productos, el usuario pulsa sobre el botón "Comprar", se le dirige a la página de la Cesta de la compra (`cesta.php`), en la que se le muestra un resumen de los productos que ha

Cesta de la compra			
ITEM1100010	Samsung 3010 16.1LED 2040 1GB 200GB BATA BT WTR		288.00
ITEM1000008	Exigore MicroCNC 908		10.20
ITEM1100002	Casco Ios 11500 anti		196.70
Precio total: 497.90			
Comprar			
Desconectar usuario actual			

seleccionado junto al importe total de los mismos.

```
<?php
// Recuperamos la información de la sesión
session_start();

// Y comprobamos que el usuario se haya autenticado
if (!isset($_SESSION['usuario'])) {
    die("Error - debe <a href='login.php'>identificarse</a>.<br />");
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 4 : Desarrollo de aplicaciones web con PHP -->
<!-- Ejemplo Tienda Web: cesta.php -->
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Ejemplo Tema 4: Cesta de la Compra</title>
    <link href="tienda.css" rel="stylesheet" type="text/css">
</head>

<body class="pagcesta">

<div id="contenedor">
    <div id="encabezado">
        <h1>Cesta de la compra</h1>
    </div>
    <div id="productos">
<?php
$total = 0;
foreach($_SESSION['cesta'] as $codigo => $producto) {
    echo "<p><span class='codigo'>$codigo</span>";
    echo "<span class='nombre'>${producto['nombre']}</span>";
    echo "<span class='precio'>${producto['precio']}</span></p>";
    $total += $producto['precio'];
}
?>

    <hr />
    <p><span class='pagar'>Precio total: <?php print $total; ?> €</span></p>
    <form action='pagar.php' method='post'>
        <p>
            <span class='pagar'>
                <input type='submit' name='pagar' value='Pagar' />
            </span>
        </p>
    </form>
</div>
<br class="divisor" />
<div id="pie">
    <form action='logout.php' method='post'>
        <input type='submit' name='desconectar' value='Desconectar' usuario <?php echo
$SESSION['usuario']; ?>' />
    </form>
</div>
</div>
</body>
</html>
```

En la página figuran dos formularios que simplemente redirigen a otras páginas. El que contiene el botón "Pagar", que redirige a la página `pagar.php`, que en nuestro caso lo único que debe hacer es eliminar la sesión del usuario. Y el que contiene el botón de desconexión, que es similar al que figuraba en la página `productos.php`, y dirige a la página `logout.php`, que cierra la sesión del usuario.

```
<?php
// Recuperamos la información de la sesión
session_start();
unset($_SESSION['cesta']);

die("Gracias por su compra.<br />Quiere <a href='productos.php'>comenzar de nuevo</a>?");
?>
```

Los datos que figuran en la página se obtienen todos de la información almacenada en la sesión del usuario. No es necesario establecer conexiones con la base de datos. El código que sirve para mostrar el listado de los productos seleccionados es el siguiente:

```
<?php
$total = 0;
foreach($_SESSION['cesta'] as $codigo => $producto) {
    echo "<p><span class='codigo'>$codigo</span>";
    echo "<span class='nombre'>${producto['nombre']}</span>";
    echo "<span class='precio'>${producto['precio']}</span></p>";
    $total += $producto['precio'];
}
?>

<hr />
<p><span class='pagar'>Precio total: <?php print $total; ?> €</span></p>
```

Recuerda que al principio de esta página, también será necesario verificar la variable de sesión `$SESSION['usuario']` para comprobar que el usuario se ha autenticado correctamente.

Tanto desde esta página como desde la página del listado de productos, se le ofrece al usuario la posibilidad de cerrar la sesión. Para ello se le dirige a la página `logoff.php`, que no muestra nada en pantalla. Su código es el siguiente:

```
<?php
// Recuperamos la información de la sesión
session_start();

// Y la eliminamos
session_unset();
header("Location: login.php");
?>
```

Tras eliminar la sesión, redirige al usuario a la página de autenticación donde puede iniciar una nueva sesión con el mismo o con otro usuario distinto.

En las páginas anteriores, `productos.php` y `cesta.php`, se ha incluido un botón para desconectar al usuario cerrando su sesión. ¿Se podría utilizar un enlace a la página `logoff.php` en lugar de crear un botón dentro de un formulario?

**Sí****No**

La única función del botón es abrir la página `logoff.php`. No necesitamos enviar nada a ésta página, por lo que se podría haber logrado lo mismo con un simple enlace.

4.- Herramientas para depuración de código.

Caso práctico

Con lo que ha aprendido, **Carlos** se ha aventurado por su cuenta en la programación de una aplicación web sencilla. Ahora ya sabe cómo y dónde almacenar la información que genera cada página, para poderla utilizar más adelante. Ha cogido la página que había creado para catalogar su colección de comics, y ha decidido convertirla en una aplicación de verdad. Empieza a estructurar el código que tenía y decide programar algunas páginas más y crear un menú para moverse entre ellas. Y entonces... comienzan a surgir los problemas: páginas que no muestran lo que se suponía que deberían mostrar, errores inexplicables cuando se intenta conectar a la base de datos, listados que algunas veces funcionan y otras no,...

Tras muchas horas intentando encontrar y solucionar los problemas que van apareciendo, busca en Internet y descubre que existen algunas utilidades que pueden ayudarle a arreglar los errores del código. No se trata de que no vaya a cometer fallos nunca más, sino de que cuando los cometa tenga una forma más cómoda y rápida de resolverlos.

Carlos está descubriendo por su cuenta cómo funciona la depuración de código en lenguaje PHP.

Seguramente en la aplicación de tienda online que acabas de programar te hayas encontrado con algunos problemas. Algún código que no funcionaba, problemas de conexión a la base de datos, o páginas que no mostraban lo que deberían.

Con lo que has visto hasta ahora ya puedes empezar a desarrollar algunas aplicaciones web sencillas en lenguaje PHP. Sin embargo, cuando empiezan a surgir problemas tienes que ingeniártelas para poder descubrir qué es lo que está fallando. Puedes poner trozos de código en las páginas que desarrollas para mostrar información interna de la aplicación, que muestren en pantalla la secuencia en que se ejecutan las líneas de código, o el contenido de las variables que usas. Para ello puedes utilizar `echo`, `print`, `print_r` o `var_dump`.

En la documentación de PHP puedes consultar información sobre la función `var_dump`.
<http://php.net/manual/es/function.var-dump.php>

Por ejemplo, si quieres ver el contenido de la variable superglobal `$_SERVER`, puedes introducir una línea como:

```
<?php print_r($_SERVER); ?>
```

Este método de depuración es muy tedioso y requiere hacer cambios constantes en el código de la aplicación. También existe la posibilidad de que una vez encontrado el fallo, te olvides de eliminar de tus páginas algunas de las líneas creadas a propósito para la depuración, con los consiguientes problemas.

Si quieres conocer más sobre los principios de la depuración, y la utilización de comandos como `echo` o `print_r` para depurar tu código, puedes consultar este artículo de php-hispano.
<http://www.php-hispano.net/articulos/debug-en-php.html>

Si ya has programado anteriormente en otros lenguajes, seguramente conoces algunas herramientas de depuración específicas que se integran con el entorno de desarrollo, permitiéndote por ejemplo detener la ejecución cuando se llega a cierta línea y ver el contenido de las variables en ese momento. Al usar estas herramientas minimizas o eliminas por completo la necesidad de introducir líneas específicas de depuración en tus programas y agilizas el procedimiento de búsqueda de errores.

De las herramientas disponibles que posibilitan la depuración en lenguaje PHP, en esta unidad aprenderás a configurar y utilizar la extensión Xdebug. Es una extensión de código libre, bajo licencia

propia (The Xdebug License, basada en la licencia de PHP) que se integra perfectamente con el IDE que estás usando, NetBeans.

Si la depuración está activada, Xdebug controla la ejecución de los guiones en PHP. Puede pausar, reanudar y detener los programas en cualquier momento. Cuando el programa está pausado, Xdebug puede obtener información del estado de ejecución, incluyendo el valor de las variables (puede incluso cambiar su valor).

Xdebug es un servidor que recibe instrucciones de un cliente, que en nuestro caso será NetBeans. De esta forma, no es necesario que el entorno de desarrollo esté en la misma máquina que el servidor PHP con Xdebug.

4.1.- Instalación de herramientas de depuración.

Concretamente, vas a aprender a preparar un sistema Ubuntu para depurar aplicaciones en lenguaje PHP utilizando Xdebug y el IDE Netbeans. Los pasos que verás a continuación los puedes seguir también utilizando el tutorial multimedia que figura al final de esta unidad.

Aunque existen diversas formas para realizar la instalación de Xdebug. Probablemente la más sencilla es a través del repositorio de extensiones PECL, del que ya hablamos en la unidad 2. Necesitas por tanto asegurarte de que PECL está disponible en tu instalación de PHP. Para instalarlo desde los repositorios de Ubuntu, ejecuta desde una consola:

```
sudo apt-get install php-pear
```

Es conveniente mantener actualizados los repositorios ejecutando `sudo apt-get update` antes de instalar paquetes.

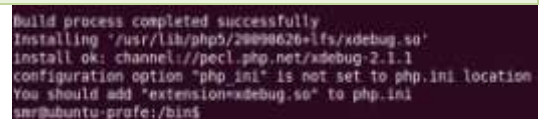
Además, para poder compilar las extensiones que descargues con PECL, debes instalar el paquete con las herramientas de desarrollo `php5-dev`. Incluye dos programas necesarios para instalar Xdebug: `phpize` y `php-config`.

```
sudo apt-get install php5-dev
```

Si todo fue bien, ya tienes el sistema preparado para instalar Xdebug.

```
sudo pecl install xdebug
```

Al finalizar la instalación verás un mensaje como el siguiente:



```
Build process completed successfully
Installing '/usr/lib/php5/20090626+lfs/xdebug.so'
install ok: channel://pecl.php.net/xdebug-2.1.1
configuration option 'php_ini' is not set to php.ini location
You should add 'extension=xdebug.so' to php.ini
usr@ubuntu-profe:/bin$
```

En esa pantalla figura (tras "`Installing`") la ruta en que se ha instalado la extensión (archivo `xdebug.so`). Además te indica que debes modificar el archivo `php.ini` para activarla. Abre por tanto el archivo `php.ini` de tu instalación de PHP, y añade las siguientes líneas al final.

```
[xdebug]
zend_extension="/usr/lib/php5/20090626+lfs/xdebug.so"
; Opciones de configuración
xdebug.remote_enable=on
xdebug.remote_handler=dbgp
xdebug.remote_host=localhost
xdebug.remote_port=9000
```

Ajusta la ruta anterior al archivo `xdebug.so` para que sea la misma que has obtenido en tu sistema tras instalar la extensión.

Por último, recuerda reiniciar el servidor web siempre que modifiques el archivo `php.ini` para aplicar los cambios.

```
sudo /etc/init.d/apache2 restart
```

Si necesitas instalar Xdebug en sistemas Windows, puedes encontrar información al respecto en Internet. En la página web de la extensión Xdebug tienes información en lenguaje inglés.
<http://xdebug.org/docs/install>

4.2.- Depuración de código en PHP.

Entre las características de depuración que incorpora Xdebug destacan:

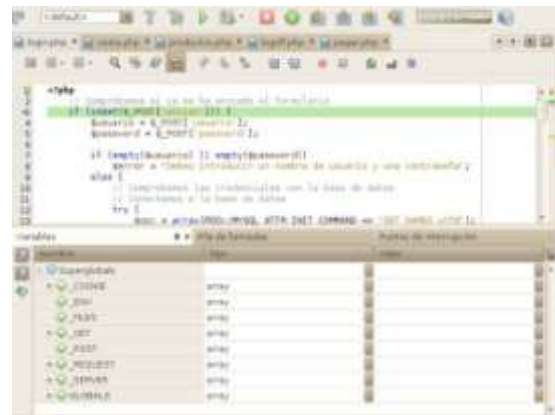
- ✓ Creación de registros con las llamadas a funciones que se produzcan, incluyendo parámetros y valores devueltos.
- ✓ Creación de registros de optimización, que permitan analizar el rendimiento de los programas.
- ✓ Depuración remota.

Vamos a centrarnos en la depuración remota utilizando NetBeans como cliente.

Utilizando los menús, botones, o la combinación de teclas adecuada, puedes iniciar la depuración del proyecto actual (Ctrl+F5, comenzando por el archivo definido en la configuración de ejecución del mismo) o directamente la del archivo que tengas abierto en el editor (Ctrl+May+F5).



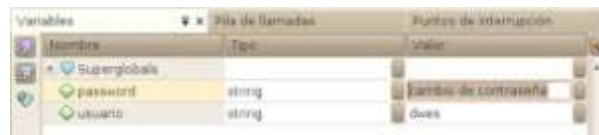
Una vez iniciada la depuración, se abre la aplicación en el navegador y se modifica la apariencia del editor. En las barras de herramientas y de menús, puedes controlar la ejecución con las opciones habituales (continuar, detener, paso a paso, etc.). En la parte inferior se muestran tres zonas nuevas con información sobre las variables que haya definidas en ese instante, la pila de llamadas y los puntos de interrupción.



En cualquier momento puedes añadir o eliminar un punto de interrupción en una línea de código simplemente pulsando con el ratón en la zona izquierda del editor, sobre los números de las líneas.



También puedes modificar el valor de las variables editando directamente sobre la zona en que se muestran.



Mientras estás depurando una aplicación, la ejecución se detendrá antes de ejecutar la primera línea de cada una de las páginas que se vayan abriendo. Éste es el comportamiento por defecto en NetBeans. En las opciones del entorno (Herramientas – Opciones – PHP) puedes modificarlo para que



se detenga solo en los puntos de ruptura que hayas indicado.

Al instalar la extensión Xdebug en PHP, se instala automáticamente el cliente para NetBeans, que permite gestionar la ejecución del código.



No



Sí

No es necesario instalar el cliente de NetBeans para Xdebug, ya viene incluido con la instalación por defecto de NetBeans

TEMA 5

Contenido

1.- Características de orientación a objetos en PHP	1
1.1.- Características de orientación a objetos en PHP.	2
1.2.- Creación de clases.	3
1.3.- Utilización de objetos.	7
1.4.- Mecanismos de mantenimiento del estado.	9
1.5.- Herencia.	10
1.6.- Interfaces.	13
1.7.- Ejemplo de POO en PHP.	15
2.- Programación en capas.	19
2.1.- Separación de la lógica de negocio.	20
2.2.- Generación del interface de usuario.	26
Anexo I - Patrón MVC Modelo Vista Controlador en PHP	28
Las capas de la arquitectura MVC.....	28
Programación simple y llana	28
Un script simple	28
Separando la presentación.....	29
La parte del controlador, en <code>index.php</code>	29
La parte de la vista, en <code>vista.php</code>	29
Separando la manipulación de los datos.....	30
La parte del modelo, en <code>modelo.php</code>	30
La parte del controlador, modificada, en <code>index.php</code>	30
Separación en capas más allá del MVC.....	31
Abstracción de la base de datos	31
Los elementos de la vista	32
Acciones y controlador frontal.....	32
Orientación a objetos.....	32

Programación Orientada a Objetos en PHP.

Caso práctico

Carlos empieza a darse cuenta de que, con lo que lleva aprendido de PHP, ya es capaz de hacer bastantes cosas. Ha acabado de programar la aplicación web para gestionar su colección de comics, y está satisfecho con el resultado obtenido. De vez en cuando ha tenido que echar mano de la documentación del lenguaje, para buscar información sobre cómo hacer algo, o los parámetros que requiere cierta función, pero siempre ha podido solucionarlo por sí mismo.

Sin embargo, cuando revisa el código de su aplicación, se da cuenta de que está muy desorganizado. Cada vez que necesita hacer algún cambio, o introducir un añadido, tiene que rebuscar entre las páginas para encontrar el código a reprogramar.

Y si eso le pasa con su pequeña aplicación, no se imagina lo que sucederá cuando tenga que programar una aplicación más compleja. —¡Tiene que haber algún modo de obtener un código más limpio y estructurado!

1.- Características de orientación a objetos en PHP

Caso práctico

Carlos comenta a **Juan** su problema, y éste le dice que seguramente gran parte del problema se arregle si utiliza programación orientada a objetos en sus aplicaciones. Le explica por encima de qué se trata y cuáles son sus ventajas, pero no puede ayudarlo mucho más. La última vez que programó en PHP, intentó utilizar objetos en su aplicación, pero las características de orientación a objetos del lenguaje dejaban mucho que desear, por lo que acabó haciéndola como siempre.

Sin embargo, por lo que ha oído, parece ser que en las últimas versiones de PHP eso ha cambiado considerablemente. El lenguaje evoluciona, y él lleva bastante tiempo sin utilizarlo, así que tendrá que actualizarse.

Mientras tanto, **Carlos** ya se ha puesto un nuevo reto: debe aprender a utilizar objetos en PHP. Y cuando tenga cierta soltura, lo primero que hará es modificar el código de su aplicación de comics.

La programación orientada a objetos (POO, o OOP en lenguaje inglés), es una metodología de programación basada en objetos. Un objeto es una estructura que contiene datos y el código que los maneja.

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).

A la creación de un objeto basado en una clase se le llama **instanciar una clase** y al objeto obtenido también se le conoce como **instancia de esa clase**.

Los pilares fundamentales de la POO son:

- ✓ **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.

Las ventajas más importantes que aporta la programación orientada a objetos son:

- ✓ **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.

En este módulo no se pretende realizar un estudio profundo de las ventajas y características de la POO, sino simplemente recordar conceptos que ya deberías haber asimilado con anterioridad. Si tienes dudas sobre algo de lo que acabamos de repasar, puedes consultar este tutorial de la web desarrolloweb.com.

<http://www.desarrolloweb.com/articulos/499.php>

1.1.- Características de orientación a objetos en PHP.

Seguramente todo, o la mayoría de lo que acabas de ver, ya lo conocías, y es incluso probable que sepas utilizar algún lenguaje de programación orientado a objetos, así que vamos a ver directamente las peculiaridades propias de PHP en lo que hace referencia a la POO.

Como ya has visto en las unidades anteriores, especialmente con las extensiones para utilizar bases de datos, con PHP puedes utilizar dos estilos de programación: estructurada y orientada a objetos.

```
// utilizando programación estructurada
$dws = mysqli_connect('localhost', 'dws', 'abc123.', 'dws');
// utilizando POO
$dws = new mysqli();
$dws->connect('localhost', 'dws', 'abc123.', 'dws');
```

Sin embargo, el lenguaje PHP original no se diseñó con características de orientación a objetos. Sólo a partir de la versión 3, se empezaron a introducir algunos rasgos de POO en el lenguaje. Esto se potenció en la versión 4, aunque todavía de forma muy rudimentaria. Por ejemplo, en PHP4:

- ✓ Los objetos se pasan siempre por valor, no por referencia.
- ✓ No se puede definir el nivel de acceso para los miembros de la clase. Todos son públicos.
- ✓ No existen los interfaces.
- ✓ No existen métodos destructores.

En la versión actual, PHP5, se ha reescrito y potenciado el soporte de orientación a objetos del lenguaje, ampliando sus características y mejorando su rendimiento y su funcionamiento general. Aunque iremos detallando y explicando cada una posteriormente con detenimiento, las características de POO que soporta PHP5 incluyen:

- ✓ Métodos estáticos.
- ✓ Métodos constructores y destructores.
- ✓ Herencia.
- ✓ Interfaces.
- ✓ Clases abstractas.

Entre las características que no incluye PHP5, y que puedes conocer de otros lenguajes de programación, están:

- ✓ Herencia múltiple.
- ✓ Sobrecarga de métodos (tener varios métodos con el mismo nombre, pero con funcionalidades distintas. Los métodos sobrecargados deben tener distintos parámetros. Cuando se llama al método, se utiliza una u otra versión en función de los parámetros con que se realice la llamada (pudiendo distinguir por su número y por su tipo, según el lenguaje de programación utilizado)) (incluidos los métodos constructores).
- ✓ Sobrecarga de operadores (similar a la sobrecarga de métodos. Se pueden definir varias funcionalidades a un mismo operador, y se utilizará una u otra en función del tipo de los operandos que se usen en cada instante).

Antes de PHP5, el comportamiento cuando se pasaba una variable a una función era siempre el mismo, independientemente de si la variable fuera un objeto o de cualquier otro tipo: siempre se creaba una nueva variable copiando los valores de la original.



Verdadero.



Falso.

Solo a partir de PHP5, cuando se pasa un objeto como parámetro a una función, se hace por referencia y no por valor.

1.2.- Creación de clases.

La declaración de una clase en PHP se hace utilizando la palabra `class`. A continuación y entre llaves, deben figurar los miembros de la clase. Conviene hacerlo de forma ordenada, primero las propiedades o atributos, y después los métodos, cada uno con su código respectivo.

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

Como comentábamos antes, es preferible que cada clase figure en su propio fichero (`producto.php`). Además, muchos programadores prefieren utilizar para las clases nombres que comiencen por letra mayúscula, para, de esta forma, distinguirlos de los objetos y otras variables.

Una vez definida la clase, podemos usar la palabra **new** para instanciar objetos de la siguiente forma:

```
$p = new Producto();
```

Para que la línea anterior se ejecute sin error, previamente debemos haber declarado la clase. Para ello, en ese mismo fichero tendrás que incluir la clase poniendo algo como:

```
require_once('producto.php');
```

Los atributos de una clase son similares a las variables de PHP. Es posible indicar un valor en la declaración de la clase. En este caso, todos los objetos que se instancien a partir de esa clase, partirán con ese valor por defecto en el atributo.

Para acceder desde un objeto a sus atributos o a los métodos de la clase, debes utilizar el **operador flecha** (fíjate que sólo se pone el símbolo `$` delante del nombre del objeto):

```
$p->nombre = 'Samsung Galaxy S';  
$p->muestra();
```

Cuando se declara un atributo, se debe indicar su nivel de acceso. Los principales niveles son:

- ✓ `public`. Los atributos declarados como `public` pueden utilizarse directamente por los objetos de la clase. Es el caso del atributo `$nombre` anterior.
- ✓ `private`. Los atributos declarados como `private` sólo pueden ser accedidos y modificados por los métodos definidos en la clase, no directamente por los objetos de la misma. Es el caso del atributo `$codigo`.

En PHP4 no se podía definir nivel de acceso para los atributos de una clase, por lo que todos se precedían de la palabra `var`.

```
var $nombre;
```

Hoy en día, aunque aún es aceptado por PHP5, no se recomienda su uso. Si encuentras algún código que lo utilice, ten en cuenta que tiene el mismo efecto que `public`.

Uno de los motivos para crear atributos privados es que su valor forma parte de la información interna del objeto y no debe formar parte de su interface. Otro motivo es mantener cierto control sobre sus posibles valores.

Por ejemplo, no quieres que se pueda cambiar libremente el valor del código de un producto. O necesitas conocer cuál será el nuevo valor antes de asignarlo. En estos casos, se suelen definir esos atributos como privados y además se crean dentro de la clase métodos para permitirnos obtener y/o modificar los valores de esos atributos. Por ejemplo:

```
private $codigo;
public function setCodigo($nuevo_codigo) {
    if (noExisteCodigo($nuevo_codigo)) {
        $this->codigo = $nuevo_codigo;
        return true;
    }
    return false;
}
public function getCodigo() { return $this->codigo; }
```

Aunque no es obligatorio, el nombre del método que nos permite obtener el valor de un atributo suele empezar por `get`, y el que nos permite modificarlo por `set`.

En PHP5 se introdujeron los llamados métodos mágicos, entre ellos `__set` y `__get`. Si se declaran estos dos métodos en una clase, PHP los invoca automáticamente cuando desde un objeto se intenta usar un atributo no existente o no accesible. Por ejemplo, el código siguiente simula que la clase `Producto` tiene cualquier atributo que queramos usar.

```
class Producto {
    private $atributos = array();

    public function __get($atributo) {
        return $this->atributos[$atributo];
    }
    public function set($atributo, $valor) {
        $this->atributos[$atributo] = $valor;
    }
}
```

En la documentación de PHP tienes más información sobre los métodos mágicos.

<http://es.php.net/manual/es/language.oop5.magic.php>

En lugar de programar un método `set` para modificar el valor de los atributos privados en que sea necesario, puedo utilizar el método mágico `__set`.



Verdadero.



Falso.

Sí, pero tendrías que comprobar el nombre del atributo usado y asignar el valor al adecuado

Cuando desde un objeto se invoca un método de la clase, a éste se le pasa siempre una referencia al objeto que hizo la llamada. Esta referencia se almacena en la variable `$this`. Se utiliza, por ejemplo, en el código anterior para tener acceso a los atributos privados del objeto (que sólo son accesibles desde los métodos de la clase).

```
print "<p>" . $this->codigo . "</p>";
```

Una referencia es una forma de utilizar distintos nombres de variables para acceder al mismo contenido. En los puntos siguientes aprenderás a crearlas y a utilizarlas.

<http://es.php.net/manual/es/language.references.php>

Además de métodos y propiedades, en una clase también se pueden definir **constantes**, utilizando la palabra `const`. Es importante que no confundas los atributos con las constantes. Son conceptos distintos: las constantes no pueden cambiar su valor (obviamente, de ahí su nombre), no usan el carácter `$` y, además, su valor va siempre entre comillas y está asociado a la clase, es decir, no existe una copia del mismo en cada objeto. Por tanto, para acceder a las constantes de una clase, se debe utilizar el nombre de la clase y el operador `::`, llamado **operador de resolución de ámbito** (que se utiliza para acceder a los elementos de una clase).

```
class DB {  
    const USUARIO = 'dwes';  
    ...  
}  
echo DB::USUARIO;
```

Es importante resaltar que no es necesario que exista ningún objeto de una clase para poder acceder al valor de las constantes que defina. Además, sus nombres suelen escribirse en mayúsculas.

Tampoco se deben confundir las constantes con los miembros estáticos de una clase. En PHP5, una clase puede tener atributos o métodos estáticos, también llamados a veces atributos o métodos de clase. Se definen utilizando la palabra clave `static`.

```
class Producto {  
    private static $num_productos = 0;  
    public static function nuevoProducto() {  
        self::$num_productos++;  
    }  
    ...  
}
```

Los atributos y métodos estáticos no pueden ser llamados desde un objeto de la clase utilizando el operador `->`. Si el método o atributo es público, deberá accederse utilizando el nombre de la clase y el operador de resolución de ámbito.

```
Producto::nuevoProducto();
```

Si es privado, como el atributo `$num_productos` en el ejemplo anterior, sólo se podrá acceder a él desde los métodos de la propia clase, utilizando la palabra `self`. De la misma forma que `$this` hace referencia al objeto actual, `self` hace referencia a la clase actual.

```
self::$num_productos ++;
```

Los atributos estáticos de una clase se utilizan para guardar información general sobre la misma, como puede ser el número de objetos que se han instanciado. Sólo existe un valor del atributo, que se almacena a nivel de clase.

Los métodos estáticos suelen realizar alguna tarea específica o devolver un objeto concreto. Por ejemplo, las clases matemáticas suelen tener métodos estáticos para realizar logaritmos o raíces cuadradas. No tiene sentido crear un objeto si lo único que queremos es realizar una operación matemática.

Los métodos estáticos se llaman desde la clase. No es posible llamarlos desde un objeto y por tanto, no podemos usar `$this` dentro de un método estático.

Como ya viste, para instanciar objetos de una clase se utiliza `new`:

```
$p = new Producto();
```

En PHP5 puedes definir en las clases métodos constructores, que se ejecutan cuando se crea el objeto. El constructor de una clase debe llamarse `__construct`. Se pueden utilizar, por ejemplo, para asignar valores a atributos.

```
class Producto {  
    private static $num productos = 0;  
    private $codigo;  
  
    public function __construct() {  
        self::$num_productos++;  
    }  
    ...  
}
```

El constructor de una clase puede llamar a otros métodos o tener parámetros, en cuyo caso deberán pasarse cuando se crea el objeto. Sin embargo, sólo puede haber un método constructor en cada clase.

```
class Producto {  
    private static $num productos = 0;  
    private $codigo;  
  
    public function __construct($codigo) {  
        $this->$codigo = $codigo;  
        self::$num_productos++;  
    }  
    ...  
}  
$p = new Producto('GALAXYS');
```

Por ejemplo, si como en este ejemplo, definimos un constructor en el que haya que pasar el código, siempre que instancias un nuevo objeto de esa clase tendrás que indicar su código.

Una de las posibilidades de los métodos mágicos de PHP5 es utilizar `__call` para capturar llamadas a métodos que no estén implementados en la clase. Entonces, en función del nombre del método y del número de parámetros que se pasen, se podrían realizar unas acciones u otras.

<http://es.php.net/manual/es/language.oop5.overloading.php#language.oop5.overloading.methods>

También es posible definir un método destructor, que debe llamarse `__destruct` y permite definir acciones que se ejecutarán cuando se elimine el objeto.

```
class Producto {  
    private static $num productos = 0;  
    private $codigo;  
  
    public function __construct($codigo) {  
        $this->$codigo = $codigo;  
        self::$num_productos++;  
    }  
  
    public function __destruct() {  
        self::$num_productos--;  
    }  
    ...  
}  
$p = new Producto('GALAXYS');
```

Los métodos constructores también existen en PHP4, pero en lugar de llamarse `construct`, se deben llamar del mismo modo que la clase. Los métodos destructores son nuevos en PHP5; no existían en versiones anteriores del lenguaje.

¿Cuál es la utilidad del operador de resolución de ámbito ::?



Nos permite hacer referencia a la clase del objeto actual.



Se utiliza para acceder a los elementos de una clase, como constantes y miembros estáticos.

Si; y por tanto debe usarse precedido por el nombre de una clase, o por una referencia a una clase como self.

1.3.- Utilización de objetos.

Ya sabes cómo instanciar un objeto utilizando `new`, y cómo acceder a sus métodos y atributos públicos con el operador flecha:

```
$p = new Producto();  
$p->nombre = 'Samsung Galaxy S';  
$p->muestra();
```

Una vez creado un objeto, puedes utilizar el operador `instanceof` para comprobar si es o no una instancia de una clase determinada.

```
if ($p instanceof Producto) {  
    ...  
}
```

Además, en PHP5 se incluyen una serie de funciones útiles para el desarrollo de aplicaciones utilizando POO.

Funciones de utilidad para objetos y clases en PHP5		
Función	Ejemplo	Significado
get_class	<pre>echo "La clase es: " . get_class(\$p);</pre>	Devuelve el nombre de la clase del objeto.
class_exists	<pre>if (class_exists('Producto') { \$p = new Producto(); ... }</pre>	Devuelve true si la clase está definida o false en caso contrario.
get_declared_classes	<pre>print_r(get_declared_classes());</pre>	Devuelve un array con los nombres de las clases definidas.
class_alias	<pre>class_alias('Producto', 'Articulo'); \$p = new Articulo();</pre>	Crea un alias para una clase.
get_class_methods	<pre>print_r(get_class_methods('Producto'));</pre>	Devuelve un array con los nombres de los métodos de una clase que son accesibles desde dónde se hace la llamada.
method_exists	<pre>if (method_exists('Producto', 'vende'){ ... }</pre>	Devuelve true si existe el método en el objeto o la clase que se indica, o false en caso contrario, independientemente de si es accesible o no.
get_class_vars	<pre>print_r(get_class_vars('Producto'));</pre>	Devuelve un array con los nombres de los atributos de una clase que son accesibles desde dónde se hace la llamada.
get_object_vars	<pre>print_r(get_object_vars(\$p));</pre>	Devuelve un array con los nombres de los métodos de un objeto que son accesibles desde dónde se hace la llamada.
property_exists	<pre>if (property_exists('Producto', 'codigo') { ... }</pre>	Devuelve true si existe el atributo en el objeto o la clase que se indica, o false en caso contrario, independientemente de si es accesible o no.

Desde PHP5, puedes indicar en las funciones y métodos de qué clase deben ser los objetos que se pasen como parámetros. Para ello, debes especificar el tipo antes del parámetro.

```
public function vendeProducto(Producto $p) {  
    ...  
}
```

Si cuando se realiza la llamada, el parámetro no es del tipo adecuado, se produce un error que podrías capturar. Además, ten en cuenta que sólo funciona con objetos (y a partir de PHP5.1 también con arrays).

Una característica de la POO que debes tener muy en cuenta es qué sucede con los objetos cuando los pasas a una función, o simplemente cuando ejecutas un código como el siguiente:

```
$p = new Producto();  
$p->nombre = 'Samsung Galaxy S';  
$a = $p;
```

En PHP4, la última línea del código anterior crea un nuevo objeto con los mismos valores del original, de la misma forma que se copia cualquier otro tipo de variable. Si después de hacer la copia se modifica, por ejemplo, el atributo 'nombre' de uno de los objetos, el otro objeto no se vería modificado.

Sin embargo, en PHP5 este comportamiento varía. El código anterior simplemente crearía un **nuevo identificador del mismo objeto**. Esto es, en cuanto se utilice uno cualquiera de los identificadores para cambiar el valor de algún atributo, este cambio se vería también reflejado al acceder utilizando el otro identificador. Recuerda que, aunque haya dos o más identificadores del mismo objeto, en realidad todos se refieren a la única copia que se almacena del mismo.

Para crear nuevos identificadores en PHP5 a un objeto ya existente, se utiliza el operador `=`. Sin embargo, como ya sabes, este operador aplicado a variables de otros tipos, crea una copia de la misma. En PHP puedes crear referencias a variables (como números enteros o cadenas de texto), utilizando el operador `&`:

```
$a = 'Samsung Galaxy S';  
$b = &$a;
```

En el ejemplo anterior, `$b` es una referencia a la variable `$a`. Cuando se cambia el valor de una de ellas, este cambio se refleja en la otra.

Las referencias se pueden utilizar para pasarlas como parámetros a las funciones. Si utilizamos el operador `&` junto al parámetro, en lugar de pasar una copia de la variable, se pasa una referencia a la misma.

```
function suma(&$v) {  
    $v++;  
}  
$a = 3;  
suma ($a);  
echo $a; // Muestra 4
```

De esta forma, dentro de la función se puede modificar el contenido de la variable que se pasa, no el de una copia.

<http://es.php.net/manual/es/language.references.php>

Por tanto, a partir de PHP5 no puedes copiar un objeto utilizando el operador `=`. Si necesitas copiar un objeto, debes utilizar `clone`. Al utilizar `clone` sobre un objeto existente, se crea una copia de todos los atributos del mismo en un nuevo objeto.

```
$p = new Producto();  
$p->nombre = 'Samsung Galaxy S';  
$a = clone($p);
```

Además, existe una forma sencilla de personalizar la copia para cada clase particular. Por ejemplo, puede suceder que quieras copiar todos los atributos menos alguno. En nuestro ejemplo, al menos el código de cada producto debe ser distinto y, por tanto, quizás no tenga sentido copiarlo al crear un

nuevo objeto. Si éste fuera el caso, puedes crear un método de nombre `__clone` en la clase. Este método se llamará automáticamente después de copiar todos los atributos en el nuevo objeto.

```
class Producto {  
    ...  
    public function __clone($atributo) {  
        $this->codigo = nuevo_codigo();  
    }  
    ...  
}
```

¿Cuál es el nombre de la función que se utiliza para hacer una copia de un objeto?



`clone.`



`__clone.`

Además, cuando utilizas la función `clone`, si la clase tiene definido un método de nombre `__clone`, se llama automáticamente.

A veces tienes dos objetos y quieres saber su relación exacta. Para eso, en PHP5 puedes utilizar los operadores `==` y `===`.

Si utilizas el operador de comparación `==`, comparas los valores de los atributos de los objetos. Por tanto dos objetos serán iguales si son instancias de la misma clase y, además, sus atributos tienen los mismos valores.

```
$p = new Producto();  
$p->nombre = 'Samsung Galaxy S';  
$a = clone($p);  
// El resultado de comparar $a == $p da verdadero  
// pues $a y $p son dos copias idénticas
```

Sin embargo, si utilizas el operador `===`, el resultado de la comparación será `true` sólo cuando las dos variables sean referencias al mismo objeto.

```
$p = new Producto();  
$p->nombre = 'Samsung Galaxy S';  
$a = clone($p);  
// El resultado de comparar $a === $p da falso  
// pues $a y $p no hacen referencia al mismo objeto  
$a = &$p;  
// Ahora el resultado de comparar $a === $p da verdadero  
// pues $a y $p son referencias al mismo objeto.
```

Los operadores de comparación `==` y `===` no funcionan de la misma manera que acabas de ver en PHP4. Si utilizas objetos en tus programas, es recomendable que te asegures de la versión del intérprete que los va a ejecutar, utilizando por ejemplo la función `phpversion`.

<http://es.php.net/manual/es/function.phpversion.php>

1.4.- Mecanismos de mantenimiento del estado.

En la unidad anterior aprendiste a usar la sesión del usuario para almacenar el estado de las variables, y poder recuperarlo cuando sea necesario. El proceso es muy sencillo; se utiliza el array superglobal `$_SESSION`, añadiendo nuevos elementos para ir guardando la información en la sesión.

El procedimiento para almacenar objetos es similar, pero hay una diferencia importante. Todas las variables almacenan su información en memoria de una forma u otra según su tipo. Los objetos, sin embargo, no tienen un único tipo. Cada objeto tendrá unos atributos u otros en función de su clase. Por tanto, para almacenar los objetos en la sesión del usuario, hace falta convertirlos a un formato estándar. Este proceso se llama serialización.

En PHP, para serializar un objeto se utiliza la función `serialize`. El resultado obtenido es un `string` que contiene un flujo de bytes, en el que se encuentran definidos todos los valores del objeto.

```
$p = new Producto();  
$a = serialize($p);
```

Esta cadena se puede almacenar en cualquier parte, como puede ser la sesión del usuario, o una base de datos. A partir de ella, es posible reconstruir el objeto original utilizando la función `unserialize`.

```
$p = unserialize($a);
```

Las funciones `serialize` y `unserialize` se utilizan mucho con objetos, pero sirven para convertir en una cadena cualquier tipo de dato, excepto el tipo `resource`. Cuando se aplican a un objeto, convierten y recuperan toda la información del mismo, incluyendo sus atributos privados. La única información que no se puede mantener utilizando estas funciones es la que contienen los atributos estáticos de las clases.

Si simplemente queremos almacenar un objeto en la sesión del usuario, deberíamos hacer por tanto:

```
session_start();  
$_SESSION['producto'] = serialize($p);
```

Pero en PHP esto aún es más fácil. Los objetos que se añadan a la sesión del usuario son serializados automáticamente. Por tanto, no es necesario usar `serialize` ni `unserialize`.

```
session_start();  
$_SESSION['producto'] = $p;
```

Para poder deserializar un objeto, debe estar definida su clase. Al igual que antes, si lo recuperamos de la información almacenada en la sesión del usuario, no será necesario utilizar la función `unserialize`.

```
session_start();  
$p = $_SESSION['producto'];
```

Como ya viste en el tema anterior, el mantenimiento de los datos en la sesión del usuario no es perfecta; tiene sus limitaciones. Si fuera necesario, es posible almacenar esta información en una base de datos. Para ello tendrás que usar las funciones `serialize` y `unserialize`, pues en este caso PHP ya no realiza la serialización automática.

En PHP además tienes la opción de personalizar el proceso de serialización y deserialización de un objeto, utilizando los métodos mágicos `__sleep` y `__wakeup`. Si en la clase está definido un método con nombre `__sleep`, se ejecuta antes de serializar un objeto. Igualmente, si existe un método `__wakeup`, se ejecuta con cualquier llamada a la función `unserialize`.

<http://www.php.net/manual/es/language.oop5.magic.php#language.oop5.magic.sleep>

Si serializas un objeto utilizando `serialize`, ¿puedes almacenarlo en una base de datos MySQL?



Verdadero.



Falso.

No solo puedes, sino que además es la forma correcta de hacerlo. `serialize` convierte el objeto en una cadena, que se puede almacenar donde sea necesario.

1.5.- Herencia.

La herencia es un mecanismo de la POO que nos permite definir nuevas clases en base a otra ya existente. Las nuevas clases que heredan también se conocen con el nombre de **subclases**. La clase de la que heredan se llama **clase base** o **superclase**.

Por ejemplo, en nuestra tienda web vamos a tener productos de distintos tipos. En principio hemos creado para manejarlos una clase llamada `Producto`, con algunos atributos y un método que genera una salida personalizada en formato HTML del código.

```
class Producto {  
    public $codigo;  
    public $nombre;
```

```

    public $nombre_corto;
    public $PVP;

    public function muestra() {
        print "<p>" . $this->codigo . "</p>";
    }
}

```

Esta clase es muy útil si la única información que tenemos de los distintos productos es la que se muestra arriba. Pero, si quieres personalizar la información que vas a tratar de cada tipo de producto (y almacenar, por ejemplo para los televisores, las pulgadas que tienen o su tecnología de fabricación), puedes crear nuevas clases que hereden de `Producto`. Por ejemplo, `TV`, `Ordenador`, `Movil`.

```

class TV extends Producto {
    public $pulgadas;
    public $tecnologia;
}

```

Como puedes ver, para definir una clase que herede de otra, simplemente tienes que utilizar la palabra `extends` indicando la superclase. Los nuevos objetos que se instancien a partir de la subclase son también objetos de la clase base; se puede comprobar utilizando el operador `instanceof`.

```

$t = new TV();
if ($t instanceof Producto) {
    // Este código se ejecuta pues la condición es cierta
...
}

```

Antes viste algunas funciones útiles para programar utilizando objetos y clases. Las de la siguiente tabla están además relacionadas con la herencia.

Funciones de utilidad en la herencia en PHP5		
Función	Ejemplo	Significado
get_parent_class	<pre> echo "La clase padre es: " . get_parent_class(\$t); </pre>	Devuelve el nombre de la clase padre del objeto o la clase que se indica.
is_subclass_of	<pre> if (is_subclass_of(\$t, 'Producto')) { ... } </pre>	Devuelve true si el objeto o la clase del primer parámetro, tiene como clase base a la que se indica en el segundo parámetro, o false en caso contrario.

La nueva clase hereda todos los atributos y métodos públicos de la clase base, pero no los privados. Si quieres crear en la clase base un método no visible al exterior (como los privados) que se herede a las subclases, debes utilizar la palabra `protected` en lugar de `private`. Además, puedes redefinir el comportamiento de los métodos existentes en la clase base, simplemente creando en la subclase un nuevo método con el mismo nombre.

```

class TV extends Producto {
    public $pulgadas;
    public $tecnologia;

    public function muestra() {
        print "<p>" . $this->pulgadas . " pulgadas</p>";
    }
}

```

Existe una forma de evitar que las clases heredadas puedan redefinir el comportamiento de los métodos existentes en la superclase: utilizar la palabra `final`. Si en nuestro ejemplo hubiéramos hecho:

```

class Producto {
    public $codigo;
    public $nombre;
    public $nombre_corto;
    public $PVP;

    public final function muestra() {
        print "<p>" . $this->codigo . "</p>";
    }
}

```

En este caso el método `muestra` no podría redefinirse en la clase `TV`.

Incluso se puede declarar una clase utilizando `final`. En este caso no se podrían crear clases heredadas utilizándola como base.

```
final class Producto {  
    ...  
}
```

Opuestamente al modificador `final`, existe también `abstract`. Se utiliza de la misma forma, tanto con métodos como con clases completas, pero en lugar de prohibir la herencia, obliga a que se herede. Es decir, una clase con el modificador `abstract` no puede tener objetos que la instancien, pero sí podrá utilizarse de clase base y sus subclasses sí podrán utilizarse para instanciar objetos.

```
abstract class Producto {  
    ...  
}
```

Y un método en el que se indique `abstract`, debe ser redefinido obligatoriamente por las subclasses, y no podrá contener código.

```
class Producto {  
    ...  
    abstract public function muestra();  
}
```

Obviamente, no se puede declarar una clase como `abstract` y `final` simultáneamente. `abstract` obliga a que se herede para que se pueda utilizar, mientras que `final` indica que no se podrá heredar.

La función `is_subclass_of` recibe como primer parámetro:



Un objeto.



Un objeto o una clase.

Efectivamente, puedes pasarle un objeto o el nombre de una clase como primer parámetro, y te dirá si es o no una subclase del nombre de clase que le pases como segundo parámetro.

Vamos a hacer una pequeña modificación en nuestra clase `Producto`. Para facilitar la creación de nuevos objetos, crearemos un constructor al que se le pasará un array con los valores de los atributos del nuevo producto.

```
class Producto {  
    public $codigo;  
    public $nombre;  
    public $nombre_corto;  
    public $PVP;  
  
    public function muestra() {  
        print "<p>" . $this->codigo . "</p>";  
    }  
  
    public function construct($row) {  
        $this->codigo = $row['cod'];  
        $this->nombre = $row['nombre'];  
        $this->nombre_corto = $row['nombre_corto'];  
        $this->PVP = $row['PVP'];  
    }  
}
```

¿Qué pasa ahora con la clase `TV`, qué hereda de `Producto`? Cuando crees un nuevo objeto de esa clase, ¿se llamará al constructor de `Producto`? ¿Puedes crear un nuevo constructor específico para `TV` que redefina el comportamiento de la clase base?

Empezando por esta última pregunta, obviamente puedes definir un nuevo constructor para las clases heredadas que redefinan el comportamiento del que existe en la clase base, tal y como harías

con cualquier otro método. Y dependiendo de si programas o no el constructor en la clase heredada, se llamará o no automáticamente al constructor de la clase base.

En PHP5, si la clase heredada no tiene constructor propio, se llamará automáticamente al constructor de la clase base (si existe). Sin embargo, si la clase heredada define su propio constructor, deberás ser tú el que realice la llamada al constructor de la clase base si lo consideras necesario, utilizando para ello la palabra `parent` y el operador de resolución de ámbito.

```
class TV extends Producto {
    public $pulgadas;
    public $tecnologia;

    public function muestra() {
        print "<p>" . $this->pulgadas . " pulgadas</p>";
    }

    public function __construct($row) {
        parent::construct($row);
        $this->pulgadas = $row['pulgadas'];
        $this->tecnologia = $row['tecnologia'];
    }
}
```

Ya viste con anterioridad cómo se utilizaba la palabra clave `self` para tener acceso a la clase actual. La palabra `parent` es similar. Al utilizar `parent` haces referencia a la clase base de la actual, tal y como aparece tras `extends`.

Si una subclase no tiene método constructor, y su clase base sí lo tiene, cuando se instancie un nuevo objeto de la subclase:



Se llamará automáticamente al constructor de la clase base.



No se llamará automáticamente al constructor de la clase base.

Fíjate que esta llamada automática solo ocurre cuando la clase heredada no tiene constructor; en otro caso tendrás que hacer tú la llamada manualmente.

1.6.- Interfaces.

Un interface es como una clase vacía que solamente contiene declaraciones de métodos. Se definen utilizando la palabra `interface`.

Por ejemplo, antes viste que podías crear nuevas clases heredadas de `Producto`, como `TV` o `Ordenador`. También viste que en las subclases podías redefinir el comportamiento del método `muestra` para que generara una salida en HTML diferente para cada tipo de producto.

Si quieres asegurarte de que todos los tipos de productos tengan un método `muestra`, puedes crear un interface como el siguiente.

```
interface iMuestra {
    public function muestra();
}
```

Y cuando creas las subclases deberás indicar con la palabra `implements` que tienen que implementar los métodos declarados en este interface.

```
class TV extends Producto implements iMuestra {
    ...
    public function muestra() {
        print "<p>" . $this->pulgadas . " pulgadas</p>";
    }
    ...
}
```

Todos los métodos que se declaren en un interface deben ser públicos. Además de métodos, los interfaces podrán contener constantes pero no atributos.

Un interface es como un contrato que la clase debe cumplir. Al implementar todos los métodos declarados en el interface se asegura la interoperabilidad entre clases. Si sabes que una clase implementa un interface determinado, sabes qué nombre tienen sus métodos, qué parámetros les debes pasar y, probablemente, podrás averiguar fácilmente con qué objetivo han sido escritos.

Por ejemplo, en la librería de PHP está definido el interface `Countable`.

```
Countable {  
    abstract public int count ( void )  
}
```

Si creas una clase para la cesta de la compra en la tienda web, podrías implementar este interface para contar los productos que figuran en la misma.

Antes aprendiste que en PHP5 una clase sólo puede heredar de otra. En PHP5 no existe la herencia múltiple. Sin embargo, sí es posible crear clases que implementen varios interfaces, simplemente separando la lista de interfaces por comas después de la palabra `implements`.

```
class TV extends Producto implements iMuestra, Countable {  
    ...  
}
```

La única restricción es que los nombres de los métodos que se deban implementar en los distintos interfaces no coincidan. Es decir, en nuestro ejemplo, el interface `iMuestra` no podría contener un método `count`, pues éste ya está declarado en `Countable`.

En PHP5 también se pueden crear nuevos interfaces heredando de otros ya existentes. Se hace de la misma forma que con las clases, utilizando la palabra `extends`.

Una de las dudas más comunes en POO, es qué solución adoptar en algunas situaciones: interfaces o clases abstractas. Ambas permiten definir reglas para las clases que los implementen o hereden respectivamente. Y ninguna permite instanciar objetos. Las diferencias principales entre ambas opciones son:

- ✓ En las clases abstractas, los métodos pueden contener código. Si van a existir varias subclases con un comportamiento común, se podría programar en los métodos de la clase abstracta. Si se opta por un interface, habría que repetir el código en todas las clases que lo implemente.
- ✓ Las clases abstractas pueden contener atributos, y los interfaces no.
- ✓ No se puede crear una clase que herede de dos clases abstractas, pero sí se puede crear una clase que implemente varios interfaces.

Por ejemplo, en la tienda online va a haber dos tipos de usuarios: clientes y empleados. Si necesitas crear en tu aplicación objetos de tipo `Usuario` (por ejemplo, para manejar de forma conjunta a los clientes y a los empleados), tendrías que crear una clase no abstracta con ese nombre, de la que heredarían `Cliente` y `Empleado`.

```
class Usuario {  
    ...  
}  
class Cliente extends Usuario {  
    ...  
}  
class Empleado extends Usuario {  
    ...  
}
```

Pero si no fuera así, tendrías que decidir si crearías o no `Usuario`, y si lo harías como una clase abstracta o como un interface.

Si por ejemplo, quisieras definir en un único sitio los atributos comunes a `Cliente` y a `Empleado`, deberías crear una clase abstracta `Usuario` de la que hereden.

```
abstract class Usuario {  
    public $dni;
```

```
protected $nombre;
...
}
```

Pero esto no podrías hacerlo si ya tienes planificada alguna relación de herencia para una de estas dos clases.

Para finalizar con los interfaces, a la lista de funciones de PHP relacionadas con la POO puedes añadir las siguientes.

Funciones de utilidad para interfaces en PHP5		
Función	Ejemplo	Significado
get_declared_interfaces	<pre>print_r (get_declared_interfaces());</pre>	Devuelve un array con los nombres de los interfaces declarados.
interface_exists	<pre>if (interface_exists('iMuestra') { ... }</pre>	Devuelve true si existe el interface que se indica, o false en caso contrario.

Si en tu código utilizas un interface, y quieres crear uno nuevo basándote en él:



Puedes utilizar la herencia para crear el nuevo constructor extendiendo al primero.

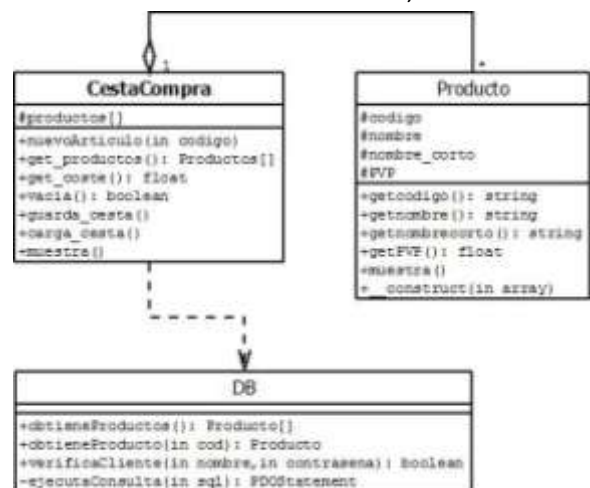
No puedes hacerlo, pues no se puede utilizar herencia con los interfaces; solo con las clases.

Sí; se hace utilizando extends, de la misma forma que con las clases

1.7.- Ejemplo de POO en PHP.

Es hora de llevar a la práctica lo que has aprendido. Vamos a aplicar los principios de la POO a la aplicación de tienda web con la que trabajamos en la unidad anterior. Concretamente, vamos a crear en un subdirectorio `include` las siguientes clases:

- ✓ **DB.** Va a ser la clase encargada de interactuar con la base de datos.
- ✓ **Producto.** Las instancias de esta clase representan los productos que se venden en la tienda.
- ✓ **CestaCompra.** Con esta clase vas a gestionar los productos que escoge el cliente de la tienda para comprar.



Obviamente en una tienda web real, la lista de clases que deberíamos crear sería mucho más amplia y debería haberse estudiado el diagrama resultante. Veamos estas tres clases una a una.

Las instancias de la clase `Producto` van a almacenar la siguiente información de cada producto: `código`, `nombre`, `nombre_corto` y `PVP`. Cada valor se almacenará en un atributo de tipo `protected`, para limitar el acceso a su contenido y, a la vez, permitir su herencia (en caso de que en el futuro creamos clases heredadas). Además, en nuestra aplicación no será necesario cambiar sus valores, pero sí acceder a ellos, por lo que se creará un método de tipo `get` para cada uno.

También necesitamos un constructor. En nuestro caso, para facilitar la creación de objetos de la clase `Producto` a partir del contenido de la base de datos, le pasaremos como parámetro un array obtenido de una fila de la base de datos.

Por último, vamos a crear también un método para mostrar el código del producto. La clase `Producto` quedará por tanto como sigue:

```
class Producto {
    protected $codigo;
    protected $nombre;
```

```

protected $nombre_corto;
protected $PVP;

public function getcodigo() {return $this->codigo; }
public function getnombre() {return $this->nombre; }
public function getnombrecorto() {return $this->nombre_corto; }
public function getPVP() {return $this->PVP; }

public function muestra() {print "<p>" . $this->codigo . "</p>";}

public function construct($row) {
    $this->codigo = $row['cod'];
    $this->nombre = $row['nombre'];
    $this->nombre_corto = $row['nombre_corto'];
    $this->PVP = $row['PVP'];
}
}

```

La clase **DB** no necesita almacenar ninguna información; simplemente deberá contener métodos para realizar acciones sobre la base de datos. Por tanto, vamos a definir en la misma únicamente métodos estáticos. No hará falta instanciar ningún objeto de esta clase.

Los métodos son los siguientes:

- ✓ **obtieneProductos**. Devuelve un array con todos los productos de la base de datos.
- ✓ **obtieneProducto(\$codigo)**. Devuelve el producto que coincide con el código que se indica.
- ✓ **verificaCliente(\$nombre, \$contrasena)**. Devuelve **true** o **false**, según sean correctas o no las credenciales que se proporcionen.

```

<?php
require_once('Producto.php');

class DB {
    protected static function ejecutaConsulta($sql) {
        $opc = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8");
        $dsn = "mysql:host=localhost;dbname=dwes";
        $usuario = 'dwes';
        $contrasena = 'abc123.';

        $dwes = new PDO($dsn, $usuario, $contrasena, $opc);
        $resultado = null;
        if (isset($dwes)) $resultado = $dwes->query($sql);
        return $resultado;
    }

    public static function obtieneProductos() {
        $sql = "SELECT cod, nombre_corto, nombre, PVP FROM producto;";
        $resultado = self::ejecutaConsulta ($sql);
        $productos = array();

        if($resultado) {
            // Añadimos un elemento por cada producto obtenido
            $row = $resultado->fetch();
            while ($row != null) {
                $productos[] = new Producto($row);
                $row = $resultado->fetch();
            }
        }

        return $productos;
    }

    public static function obtieneProducto($codigo) {
        $sql = "SELECT cod, nombre_corto, nombre, PVP FROM producto";
        $sql .= " WHERE cod='" . $codigo . "'";
        $resultado = self::ejecutaConsulta ($sql);
        $producto = null;

        if(isset($resultado)) {
            $row = $resultado->fetch();
            $producto = new Producto($row);
        }

        return $producto;
    }
}

```

```

    }

    public static function verificaCliente($nombre, $contrasena) {
        $sql = "SELECT usuario FROM usuarios ";
        $sql .= "WHERE usuario='$nombre' ";
        $sql .= "AND contrasena='" . md5($contrasena) . "'";
        $resultado = self::ejecutaConsulta ($sql);
        $verificado = false;

        if(isset($resultado)) {
            $fila = $resultado->fetch();
            if($fila !== false) $verificado=true;
        }
        return $verificado;
    }
}

?>

```

Utilizaremos de apoyo un método `protected`, `ejecutaConsulta`, que será el que realmente ejecute las consultas sobre la base de datos.

Por último, la clase `CestaCompra` debe almacenar un array con los productos que figuran en la cesta. Ese array lo crearemos como `protected` para controlar el acceso a su contenido, pero necesitamos un método `get_productos` para devolverlo.

Además, hemos implementado los siguientes métodos en la misma.

- ✓ `nuevo_articulo($codigo)`. Introduce en la cesta el artículo indicado por su código.
- ✓ `get_productos`. Devuelve un array con todos los productos de la base de datos.
- ✓ `get_coste`. Devuelve el coste de los productos que figuran en la cesta.
- ✓ `vacia`. Devuelve `true` o `false`, según la cesta esté o no vacía.
- ✓ `obtieneProductos`. Devuelve un array con todos los productos de la base de datos.

También necesitamos dos funciones para guardar la cesta en la sesión del usuario, y para recuperarla. Y programaremos otra más para mostrar el contenido de la cesta en formato HTML.

- ✓ `guarda_cesta`. Guarda la cesta en la sesión del usuario.
- ✓ `carga_cesta`. Recupera el contenido de la cesta de la sesión del usuario.
- ✓ `muestra`. Genera una salida en formato HTML con el contenido de la cesta.

```

<?php
require_once('DB.php');

class CestaCompra {
    protected $productos = array();

    // Introduce un nuevo artículo en la cesta de la compra
    public function nuevo_articulo($codigo) {
        $producto = DB::obtieneProducto($codigo);
        $this->productos[] = $producto;
    }

    // Obtiene los artículos en la cesta
    public function get_productos() { return $this->productos; }

    // Obtiene el coste total de los artículos en la cesta
    public function get_coste() {
        $coste = 0;
        foreach($this->productos as $p) $coste += $p->getPVP();
        return $coste;
    }

    // Devuelve true si la cesta está vacía
    public function vacia() {
        if(count($this->productos) == 0) return true;
        return false;
    }

    // Guarda la cesta de la compra en la sesión del usuario

```

```

public function guarda_cesta() { $_SESSION['cesta'] = $this; }

// Recupera la cesta de la compra almacenada en la sesión del usuario
public static function carga_cesta() {
    if (!isset($_SESSION['cesta'])) return new CestaCompra();
    else return ($_SESSION['cesta']);
}

// Muestra el HTML de la cesta de la compra, con todos los productos
public function muestra() {
    // Si la cesta está vacía, mostramos un mensaje
    if (count($this->productos)==0) print "<p>Cesta vacía</p>";
    // y si no está vacía, mostramos su contenido
    else foreach ($this->productos as $producto) $producto->muestra();
}
}
?>

```

El resto de ficheros que componen la tienda web tendremos que reescribirlos para que utilicen las clases que acabas de definir. En general, el resultado obtenido es mucho más claro y conciso. Veamos algunos ejemplos:

En `login.php`, para autentificar al usuario basta con hacer:

```

if (DB::verificaCliente($_POST['usuario'], $_POST['password'])) {
    ...
}

```

En `productos.php`, el código para vaciar o añadir un nuevo producto a la cesta de la compra del usuario será:

```

// Recuperamos la cesta de la compra
$cesta = CestaCompra::carga_cesta();
// Comprobamos si se ha enviado el formulario de vaciar la cesta
if (isset($_POST['vaciar'])) {
    unset($_SESSION['cesta']);
    $cesta = new CestaCompra();
}
// Comprobamos si se quiere añadir un producto a la cesta
if (isset($_POST['enviar'])) {
    $cesta->nuevo_articulo($_POST['cod']);
    $cesta->guarda_cesta();
}

```

Para tener el código más organizado, hemos creado una función para mostrar el listado de todos los productos:

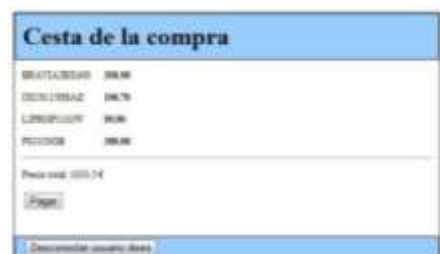
```

function creaFormularioProductos(){
    $productos = DB::obtieneProductos();
    foreach ($productos as $p) {
        // Creamos el formulario en HTML para cada producto
        ...
    }
}

```

Prueba a instalar y ejecutar la aplicación de tienda online resultante. Revisa el código y comprueba que entiendes su funcionamiento.

[Aplicación de tienda online resultante.](#)



La clase DB tiene todos sus métodos estáticos. No tiene sentido por tanto crear ningún objeto de esa clase, y podría haberse implementado igualmente como un interface.



Verdadero.



Falso.

Efectivamente, no se podría, pues no se pueden programar métodos en los interfaces.

2.- Programación en capas.

Caso práctico

Después de unas semanas, **Carlos** ha conseguido reprogramar su aplicación de catalogación utilizando orientación a objetos. Le ha costado bastante esfuerzo, pero reconoce que el resultado merece la pena. El código resultante es mucho más limpio, y cada clase de las que ha creado tiene un cometido concreto y bien definido.

Ahora es mucho más fácil hacer cambios en el código. Definitivamente, tendrán que utilizar orientación a objetos cuando empiecen el nuevo proyecto. Mientras tanto, se propone volver sobre una asignatura pendiente: mejorar el aspecto de sus páginas. Y aunque conoce el lenguaje HTML, y sabe utilizar las hojas de estilo, decide pedirle consejo a un amigo suyo que se dedica al diseño de webs.

Su amigo trabaja en otra empresa y su función es diseñar el aspecto de los sitios web que crean. Cuando ve la aplicación de **Carlos**, queda muy impresionado por lo que ha avanzado en poco tiempo. Tras examinarla, le indica que tal y como está es muy difícil cambiar su aspecto. Tiene el código HTML distribuido en diversos ficheros, y entremezclado con el código PHP.

Le comenta que en su empresa utilizan mecanismos de separación del código y le anima a que los pruebe. Si lo hace, él se ofrece a darle un diseño más profesional a su aplicación. —¡Manos a la obra!

En el ejemplo anterior, hemos programado una aplicación web sencilla utilizando programación orientada a objetos. Sin embargo, si observaste el resultado obtenido, habrás visto como en muchas ocasiones se mezcla el código propio de la lógica de la aplicación, con el código necesario para crear el interface web que se presenta a los usuarios.

Por ejemplo, tanto la clase `CestaCompra` como la clase `Producto`, cuyo objetivo debería ser implementar la lógica de la aplicación, tienen un método llamado `muestra` destinado a generar etiquetas HTML. E inversamente, en algunas páginas que deberían simplemente generar HTML, puedes encontrar código que forma parte de la lógica de la aplicación. Por ejemplo, en la página `productos.php`, el código para incluir un nuevo producto en la cesta de la compra se encuentra mezclado con las etiquetas HTML.

Existen varios métodos que permiten separar la lógica de presentación (en nuestro caso, la que genera las etiquetas HTML) de la lógica de negocio, donde se implementa la lógica propia de cada aplicación. El más extendido es el patrón de diseño Modelo – Vista – Controlador (MVC). Este patrón pretende dividir el código en tres partes, dedicando cada una a una función definida y diferenciada de las otras.

- ✓ **Modelo.** Es el encargado de manejar los datos propios de la aplicación. Debe proveer mecanismos para obtener y modificar la información del mismo. Si la aplicación utiliza algún tipo de almacenamiento para su información (como un SGBD), tendrá que encargarse de almacenarla y recuperarla.
- ✓ **Vista.** Es la parte del modelo que se encarga de la interacción con el usuario. En esta parte se encuentra el código necesario para generar el interface de usuario (en nuestro caso en HTML), según la información obtenida del modelo.
- ✓ **Controlador.** En este módulo se decide qué se ha de hacer, en función de las acciones del usuario con su interface. Con esta información, interactúa con el modelo para indicarle las acciones a realizar y, según el resultado obtenido, envía a la vista las instrucciones necesarias para generar el nuevo interface.

La gran ventaja de este patrón de programación es que genera código muy estructurado, fácil de comprender y de mantener. En la web puedes encontrar algunos ejemplos de implementación del modelo MVC en PHP. Échale un vistazo al siguiente artículo sobre MVC en PHP.

Anexo I - Patrón MVC Modelo Vista Controlador en PHP

Aunque puedes programar utilizando MVC por tu cuenta, es más habitual utilizar el patrón MVC en conjunción con un framework o marco de desarrollo. Existen numerosos frameworks disponibles en PHP, muchos de los cuales incluyen soporte para MVC. En esta unidad no profundizaremos en la utilización de un framework específico, pero existen numerosos recursos con información en Internet, incluyendo la página www.phpwebframeworks.com, dedicada exclusivamente a ellos.

PHP Framework	PHP4	PHP5	MVC	Multiple DB's	ORM	DB Objects	Templa tes	Caching	Valida tion	Ajax	Auth Module	Modules	EDP
Akelos	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Ash.MVC		✓	✓			✓	✓		✓		✓	✓	
CakePHP	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	
CodeIgniter	✓	✓	✓	✓		✓	✓	✓	✓				
DIY		✓	✓		✓	✓	✓	✓		✓			
eZ Components		✓		✓		✓	✓	✓	✓				
Fusebox	✓	✓	✓	✓				✓		✓		✓	
PHP on TRAX		✓	✓	✓	✓	✓			✓	✓		✓	
PHPDev Shell		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	
PhpOpen biz		✓	✓	✓	✓	✓	✓		✓	✓	✓		
Prado		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
QPHP	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓
Seagull	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Symfony		✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	
WACT	✓	✓	✓	✓		✓	✓		✓			✓	
WASP		✓	✓			✓	✓		✓	✓	✓	✓	
Yii		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Zend		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
ZooP	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓		

2.1.- Separación de la lógica de negocio.

Otros mecanismos disponibles en PHP, menos complejos que la utilización del patrón MVC, y que también permiten la separación de la lógica de presentación y la lógica de negocio, son los llamados motores de plantillas (template engines).

Un **motor de plantillas web** es una aplicación que genera una página web a partir de un fichero con la información de presentación (denominado plantilla o template, que viene a ser similar a la vista en el patrón MVC) y otro con la lógica interna de la aplicación (similar al modelo de MVC). De esta forma, es sencillo dividir el trabajo de programación de una aplicación web en dos perfiles: un programador, que debe conocer el lenguaje de programación en el que se implementará la lógica de la aplicación (en nuestro caso PHP), y un diseñador, que se encargará de elaborar las plantillas, (en el caso de la web básicamente en HTML, aunque como veremos la lógica de presentación que se incorpore utilizará un lenguaje propio).

En PHP existen varios motores de plantillas con diferentes características. Quizás el más conocido es Smarty, de código abierto y disponible bajo licencia LGPL.

<http://www.smarty.net/>

Entre las características de Smarty cabe destacar:

- ✓ Permite la inclusión en las plantillas de una lógica de presentación compleja.
- ✓ Acelera la generación de la página web resultante. Uno de los problemas de los motores de plantillas es que su utilización influye negativamente en el rendimiento. Smarty convierte

internamente las plantillas a guiones PHP equivalentes, y posibilita el almacenamiento del resultado obtenido en memoria temporal.

- ✓ Al ser usado por una amplia comunidad de desarrolladores, existen multitud de ejemplos y foros para la resolución de los problemas que te vayas encontrando al utilizarlo.

Para instalar Smarty, debes descargar la última versión desde su sitio web, y seguir los siguientes pasos:

1. Copiar los archivos de la librería de Smarty (el directorio `libs` que obtienes tras descomprimir el fichero que has descargado) en tu sistema; por ejemplo en la carpeta `/usr/share/smarty`.

```
smr@ubuntu-profe: ~/Descargas
Archivo Editar Ver Terminal Ayuda
smr@ubuntu-profe:~/Descargas$ unzip -q Smarty-3.0.8.zip
smr@ubuntu-profe:~/Descargas$ sudo mkdir /usr/share/smarty/
smr@ubuntu-profe:~/Descargas$ sudo cp -r Smarty-3.0.8/libs/* /usr/share/smarty/
smr@ubuntu-profe:~/Descargas$ ls /usr/share/smarty
debug.tpl  plugins  Smarty.class.php  sysplugins
smr@ubuntu-profe:~/Descargas$
```

2. Modificar el fichero `php.ini` para que se incluya en la variable `include_path` de PHP la ruta en la que acabas de instalar Smarty, y reiniciar Apache para aplicar los cambios.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Paths and Directories ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; UNIX: "/path1:/path2"
; include_path = ".:/usr/share/php"

; Windows: "\\path1;\path2"
; include_path = ".;c:\php\includes"

; PHP's default setting for include_path is ".:/path/to/php/pear"
; http://php.net/include-path
include_path = ".:/usr/share/php:/usr/share/pear:/usr/share/smarty"
```

3. Crear la estructura de directorios necesaria para Smarty. Concretamente debes crear cuatro directorios, con nombres `templates`, `templates_c`, `configs` y `cache`. Es conveniente que estén ubicados en un lugar no accesible por el servidor web, y en una ubicación distinta para cada aplicación web que programes. Por ejemplo, puedes crear en la raíz de tu servidor una carpeta llamada `smarty`, y bajo ella otra con el nombre de cada aplicación (por ejemplo `stienda`), que será la que contendrá las carpetas.

Descargamos Smarty desde su [página oficial](http://www.smarty.net/download)



Existen múltiples versiones, pero descargaremos la última versión estable

Latest Stable Release

- Smarty 3.0.8 ([tar.gz](#)) ([zip](#)) June 3rd, 2011

Nosotros descargaremos la última versión estable, en formato zip.

Latest Unstable Release

- Smarty 3.1 RC1 ([tar.gz](#)) ([zip](#)) June 27th, 2011

Previous Releases

- Smarty 3.0.7 ([tar.gz](#)) ([zip](#)) February 11th, 2011
- Smarty 3.0.6 ([tar.gz](#)) ([zip](#)) December 13th, 2010
- Smarty 3.0.5 ([tar.gz](#)) ([zip](#)) November 20th, 2010
- Smarty 3.0.4 ([tar.gz](#)) ([zip](#)) November 13th, 2010
- Smarty 3.0.3 ([tar.gz](#)) ([zip](#)) November 13th, 2010
- Smarty 3.0.2 ([tar.gz](#)) ([zip](#)) November 12th, 2010
- Smarty 3.0.1 ([tar.gz](#)) ([zip](#)) November 12th, 2010
- Smarty 3.0.0 ([tar.gz](#)) ([zip](#)) November 11th, 2010
- Smarty 3.0rc4 ([tar.gz](#)) ([zip](#)) October 5th, 2010
- Smarty 3.0rc3 ([tar.gz](#)) ([zip](#)) July 14th, 2010
- Smarty 3.0rc2 ([tar.gz](#)) ([zip](#)) June 14th, 2010
- Smarty 3.0rc1 ([tar.gz](#)) ([zip](#)) April 29th 2010
- Smarty 3.0b8 ([tar.gz](#)) ([zip](#)) March 5th, 2010
- Smarty 3.0b7 ([tar.gz](#)) ([zip](#)) Jan 17th, 2010

Smarty 2 Releases

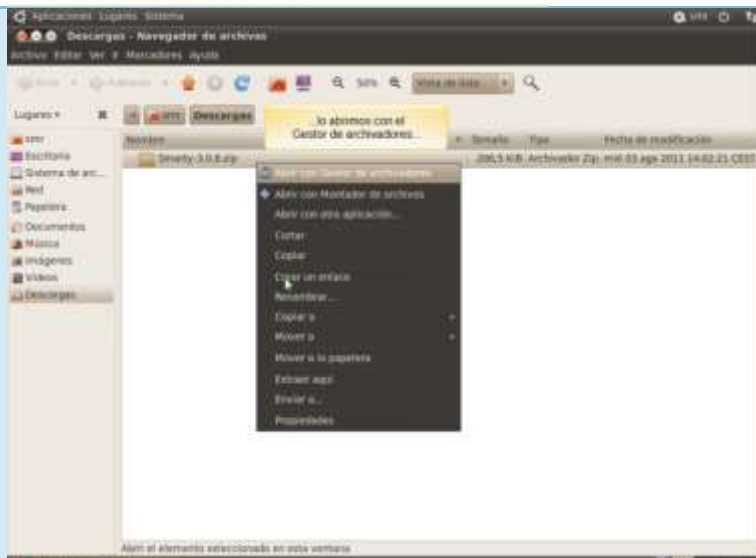
- Smarty 2.6.26 ([tar.gz](#)) ([zip](#)) June 18th, 2009

Abrimos ahora un nuevo terminal y tecleamos:

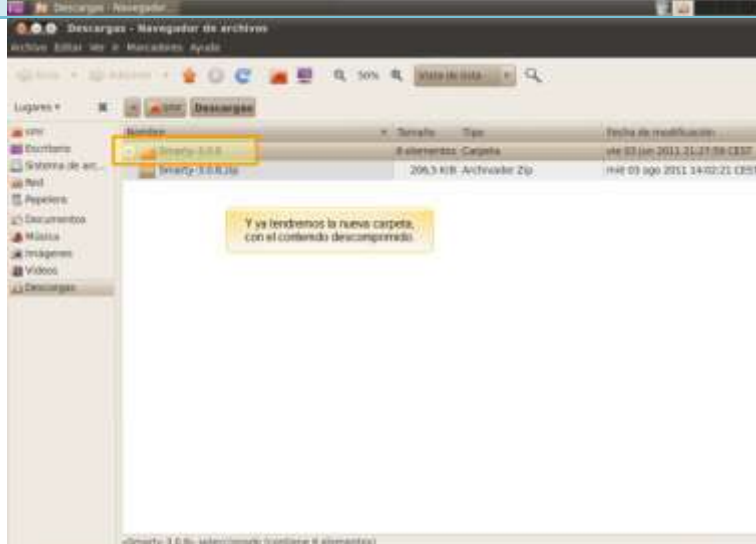
```
gksudo nautilus
```

para abrir un navegador de archivos con permisos de superusuario.

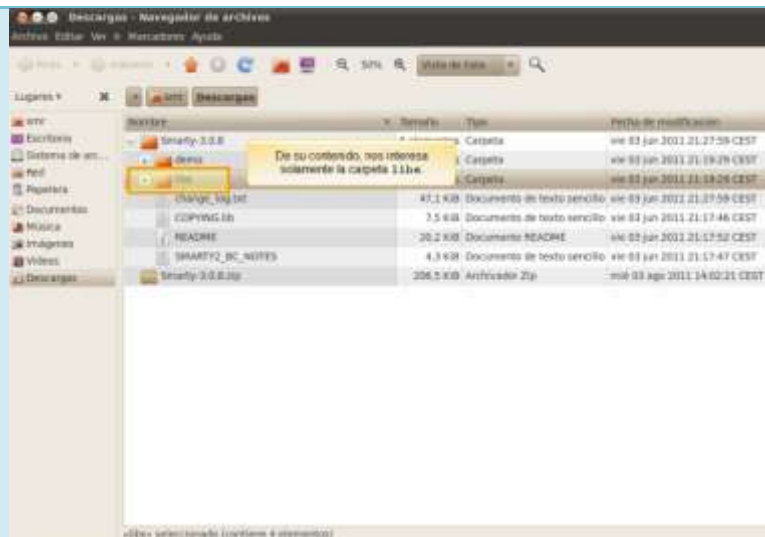
Nos desplazamos hasta donde se encuentra el archivo descargado, y lo abrimos con el *Gestor de archivadores*



Extraemos todos los archivos que contiene el fichero comprimido, y tendremos en el mismo lugar de la descarga la nueva carpeta con los archivos descomprimidos.



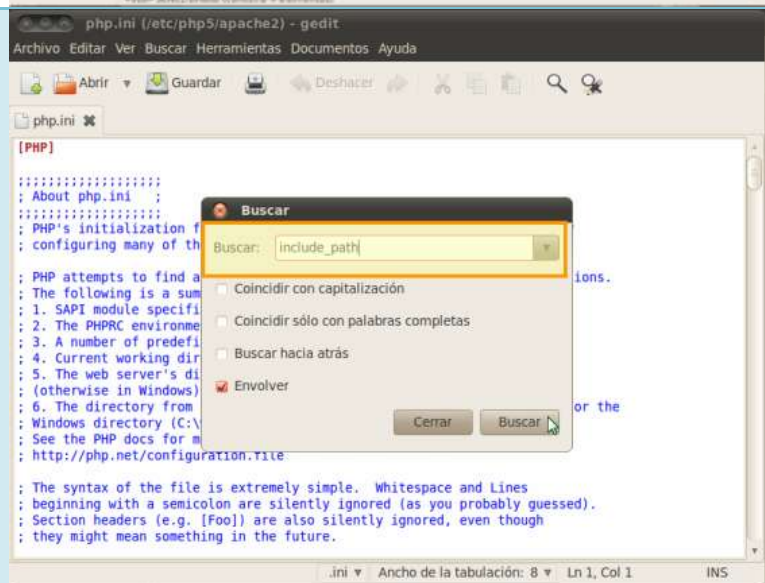
De su contenido nos interesa la carpeta `libs`, la cual copiaremos a `/usr/share`



Una vez copiada, la renombraremos como smarty



Ahora, para hacerlo visible a php tendremos que editar `php.ini` que se encuentra en `/etc/php5/apache2` y buscamos la cadena `include_path`



Y le añadimos la ruta de Smarty

```
*php.ini (/etc/php5/apache2) - gedit
Archivo Editar Ver Buscar Herramientas Documentos Ayuda

;default_charset = "iso-8859-1"

; Always populate the $HTTP_RAW_POST_DATA variable. PHP's default behavior is
; to disable this feature.
; http://php.net/always-populate-raw-post-data
;always_populate_raw_post_data = On

;;;;;;;;;;;;;;;;;;;;;;;;;
; Paths and Directories ;
;;;;;;;;;;;;;;;;;;;;;;;;;

; UNIX: "/path1:/path2"
;include_path = ".:/usr/share/php"

; Windows: "\path1;\path2"
;include_path = ".;c:\php\includes"

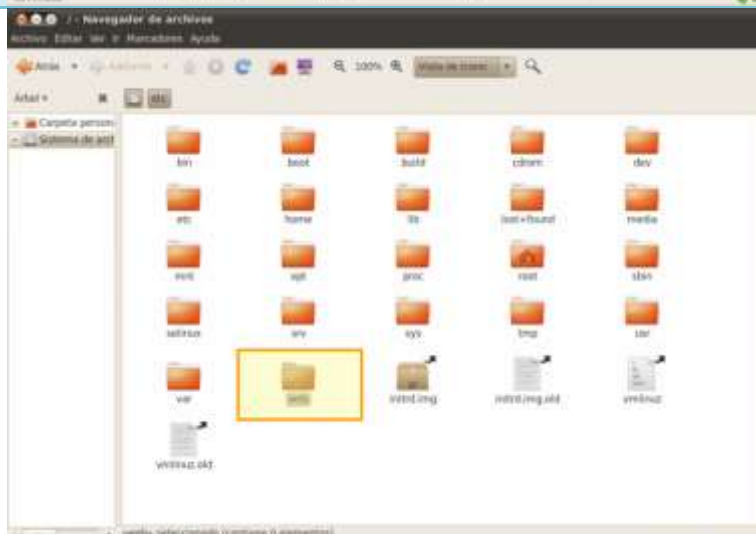
; PHP's default setting for include_path is ".:/path/to/php/pear"
; http://php.net/include_path
include_path = ".:/usr/share/php:/usr/share/pear:/usr/share/smarty"

; The root of the PHP pages, used only if nonempty.
; if PHP was not compiled with FORCE_REDIRECT, you SHOULD set doc root
; if you are running php as a CGI under any web server (other than IIS)
; see documentation for security issues. The alternate is to use the
; cgi.force_redirect configuration below
```

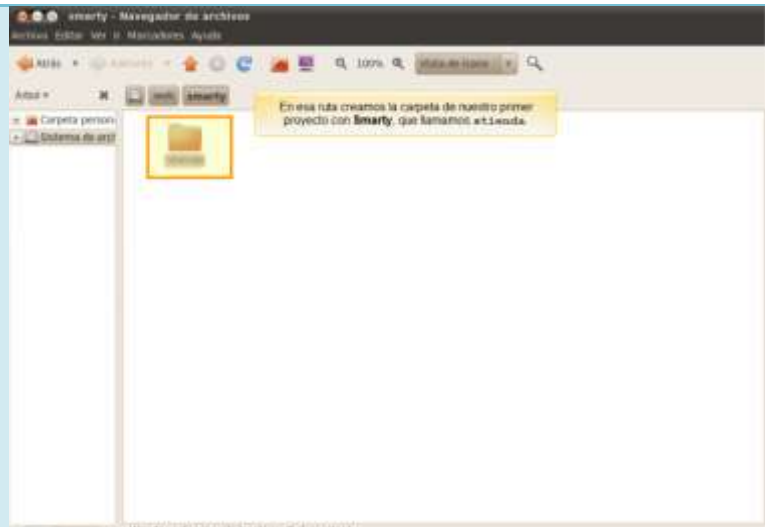
Tras guardar los cambios, abrimos de nuevo una terminal y recargamos el servidor apache: `sudo /etc/init.d/apache2 reload`. Ahora abrimos un navegador para comprobar que se ha cargado correctamente, para lo que usaremos la función `phpinfo()`



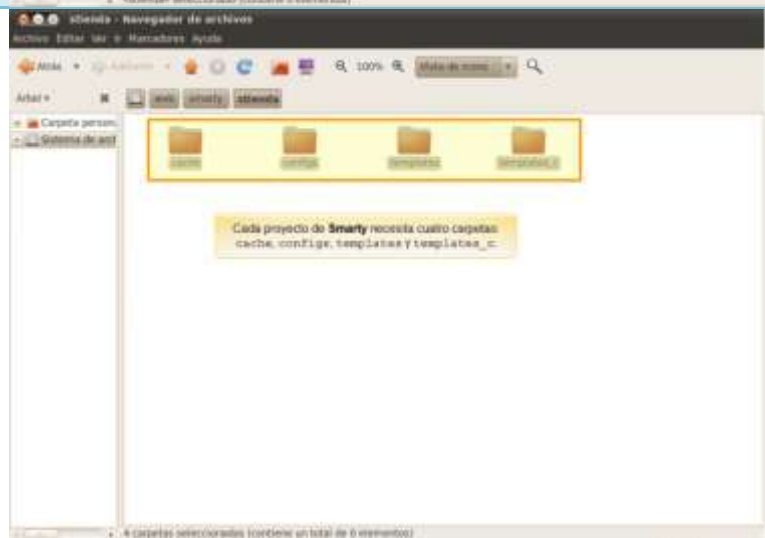
Por último, hemos de crear las carpetas necesarias para Smarty. Vamos a `/etc` y creamos una carpeta denominada `web` (por ejemplo)



Dentro de la carpeta recién creada, añadiremos otra a la que llamaremos `smarty` y que será la que recoja todos nuestros proyectos con Smarty, por ejemplo, uno al que denominaremos `stienda`



Cada proyecto Smarty tendrá al menos 4 carpetas: `cache`, `configs`, `templates` y `templates_c`



Y ya tendríamos Smarty instalado. Sólo falta empezar a utilizarlo en nuestras aplicaciones web

Al utilizar Smarty, las plantillas que obtienes no precisan ningún tipo de programación; sólo el contenido normal de una página web.



Verdadero



Falso.

Al utilizar Smarty logras separar la lógica de negocio de la información relativa a la presentación; pero en muchas ocasiones, ésta última sigue necesitando algún código, aunque no necesariamente en lenguaje PHP

Para utilizar Smarty, simplemente tienes que añadir a tus páginas PHP el fichero `Smarty.class.php`, que es donde está declarada la clase `Smarty`. Después debes instanciar un nuevo objeto de esa clase, y configurar la ruta a cada uno de los directorios que acabas de crear.

```
require_once('Smarty.class.php');
$smarty = new Smarty;
$smarty->template_dir = '/web/smarty/stienda/templates/';
$smarty->compile_dir = '/web/smarty/stienda/templates_c/';
$smarty->config_dir = '/web/smarty/stienda/configs/';
$smarty->cache_dir = '/web/smarty/stienda/cache/';
```

Cuando quieras que algún valor o variable obtenido en tus páginas, esté disponible para mostrarlo en las páginas web a través de una plantilla, tienes que usar el método `assign`, indicando el nombre del identificador. Puedes utilizar `assign` con variables de cualquier tipo, incluyendo arrays asociativos y objetos.

```
$smarty->assign('usuario', $_SESSION['usuario']);
$smarty->assign('productoscesta', $cesta->get_productos());
$smarty->assign('coste', $cesta->get_coste());
```

Y una vez que hayas preparado los identificadores que se usarán en la plantilla, deberás mostrarla utilizando el método `display`.

```
$smarty->display('cesta.tpl');
```

Así, por ejemplo, al utilizar Smarty en la página `productos.php` de la tienda online, puedes obtener algo como:

```
require_once('include/DB.php');
require_once('include/CestaCompra.php');
require_once('Smarty.class.php');
// Recuperamos la información de la sesión
session_start();
// Y comprobamos que el usuario se haya autenticado
if (!isset($SESSION['usuario']))
    die("Error - debe <a href='login.php'>identificarse</a>.<br />");
// Recuperamos la cesta de la compra
$cesta = CestaCompra::carga_cesta();
// Cargamos la librería de Smarty
$smarty = new Smarty;
$smarty->template_dir = '/web/smarty/stienda/templates/';
$smarty->compile_dir = '/web/smarty/stienda/templates_c/';
$smarty->config_dir = '/web/smarty/stienda/configs/';
$smarty->cache_dir = '/web/smarty/stienda/cache/';
// Comprobamos si se ha enviado el formulario de vaciar la cesta
if (isset($_POST['vaciar'])) {
    unset($_SESSION['cesta']);
    $cesta = new CestaCompra();
}
// Comprobamos si se quiere añadir un producto a la cesta
if (isset($_POST['enviar'])) {
    $cesta->nuevo_articulo($_POST['cod']);
    $cesta->guarda_cesta();
}
// Ponemos a disposición de la plantilla los datos necesarios
$smarty->assign('usuario', $_SESSION['usuario']);
$smarty->assign('productos', DB::obtieneProductos());
$smarty->assign('productoscesta', $cesta->get_productos());
// Mostramos la plantilla
$smarty->display('productos.tpl');
```

2.2.- Generación del interface de usuario.

Las plantillas de Smarty son ficheros con extensión `tpl`, en los que puedes incluir prácticamente cualquier contenido propio de una página web. Además, en el medio intercalarás delimitadores para indicar la inclusión de datos y de lógica propia de la presentación.

Los delimitadores de Smarty son llaves. De los distintos elementos que puedes incluir entre las llaves están:

- ✓ **Comentarios.** Van encerrados entre asteriscos.

```
{* Este es un comentario de plantilla en Smarty *}
```

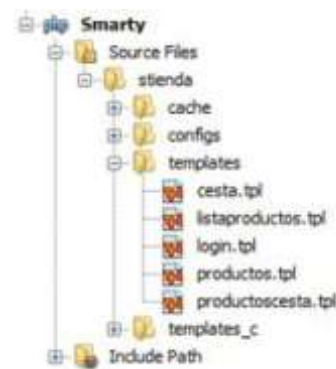
- ✓ **Variables.** Se incluye simplemente su nombre, precedido por el símbolo `$`. También se pueden especificar modificadores, separándolos de la variable por una barra vertical. Existen varios modificadores para, por ejemplo, dar formato a una fecha (`date format`) o mostrar un contenido predeterminado si la variable está vacía (`default`).

```
{ $producto->codigo }
```

- ✓ **Estructura de procesamiento condicional:** `if`, `elseif`, `else`. Permite usar condiciones, de forma similar a PHP, para decidir si se procesa o no cierto contenido.

```
{if empty($productoscesta)}
    <p>Cesta vacía</p>
{else}
    ...
{/if}
```

- ✓ **Bucles:** `foreach`. Son muy útiles para mostrar varios elementos, por ejemplo en una tabla. Deberás indicar al menos con `from` el array en el que están los elementos, y con `item` la variable a la que se le irán asignado los elementos en cada iteración.



```
{foreach from=$productoscesta item=producto}
  <p>{$producto->codigo}</p>
{/foreach}
```

- ✓ **Inclusión de otras plantillas.** Smarty permite descomponer una plantilla compleja en trozos más pequeños y almacenarlos en otras plantillas, que se incluirán en la actual utilizando la sentencia `include`.

```
<div id="cesta">
  {include file="productoscesta.tpl"}
</div>
<div id="productos">
  {include file="listaproductos.tpl"}
</div>
```

Las que acabas de ver son una pequeña parte de las funcionalidades que ofrece Smarty. En su página web, puedes acceder a la documentación completa. En el momento de escribir estas líneas, la última versión disponible en español de la documentación era la correspondiente a la versión 2.

<http://www.smarty.net/docsv2/es/>

A partir del código obtenido utilizando POO para la tienda web, aplica el motor de plantillas Smarty para dividir la lógica de presentación de la lógica de negocio. Utiliza como apoyo la documentación disponible en Internet. Cuando acabes, puedes comparar lo que has obtenido con la solución propuesta.

Solución propuesta.

Las plantillas que crees en Smarty es preferible alojarlas:



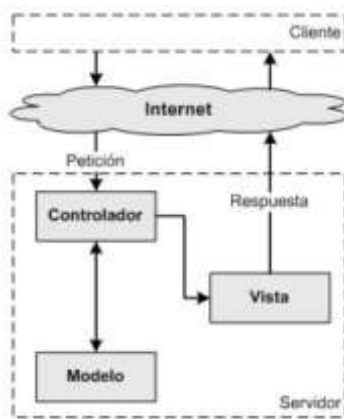
En un lugar no accesible por el servidor web.



En un lugar accesible por el servidor web.

En realidad es PHP el que debe acceder a ellas, no el servidor web.

Anexo I - Patrón MVC Modelo Vista Controlador en PHP



El patrón clásico del diseño web conocido como arquitectura MVC, está formado por tres niveles:

1. El **modelo** representa la información con la que trabaja la aplicación, es decir, su **lógica de negocio**.
2. La **vista** transforma el modelo en una página web que permite al usuario interactuar con ella.
3. El **controlador** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La **arquitectura MVC** separa la **lógica de negocio** (el **modelo**) y la **presentación** (la **vista**) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación

debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una **vista** nueva para cada dispositivo; manteniendo el **controlador** y el **modelo** original. El **controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo** utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El **modelo se encarga de la abstracción de la lógica relacionada con los datos**, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación.

Las capas de la arquitectura MVC

Para poder entender las **ventajas de utilizar el patrón MVC**, se va a transformar una aplicación simple realizada con PHP en una aplicación que sigue la **arquitectura MVC**. Un buen ejemplo para ilustrar esta explicación es el de mostrar una lista con las últimas entradas o artículos de un blog como este mismo <http://arleytriana.blogspot.com>.

Programación simple y llana

Utilizando solamente PHP normal y corriente, el script necesario para mostrar los artículos almacenados en una base de datos se muestra a continuación:

Un script simple

```

<?php
// Conectar con la base de datos y seleccionarla
$conexion = mysql_connect('localhost', 'miusuario', 'micontrasena');
mysql_select_db('blog_db', $conexion);
// Ejecutar la consulta SQL
$resultado = mysql_query('SELECT fecha, titulo FROM articulo',$conexion);
?>
<html>
<head>
<title>Listado de Artículos</title>
</head>
<body>
<h1>Listado de Artículos</h1>
<table>
<tr><th>Fecha</th><th>Titulo</th></tr>
<?php
// Mostrar los resultados con HTML
while ($fila = mysql_fetch_array($resultado, MYSQL_ASSOC)){
    echo "<tr>\n";
    printf("<td> %s </td>\n", $fila['fecha']);
    printf("<td> %s </td>\n", $fila['titulo']);
    echo "</tr>\n";
}
?>
</table>
</body>
</html>
<?php
// Cerrar la conexion
  
```

```
mysql_close($conexion);  
?>
```

El script anterior es fácil de escribir y rápido de ejecutar, pero muy difícil de mantener y actualizar.

Los principales problemas del código anterior son:

- ✓ No existe protección frente a errores (¿qué ocurre si falla la conexión con la base de datos?).
- ✓ El código HTML y el código PHP están mezclados en el mismo archivo e incluso en algunas partes están entrelazados.
- ✓ El código solo funciona si la base de datos es MySQL.

Separando la presentación

Las llamadas a `echo` y `printf` del listado anterior dificultan la lectura del código. De hecho, modificar el código HTML del script anterior para mejorar la presentación es un muy complicado debido a cómo está programado. Así que el código va a ser dividido en dos partes. En primer lugar, el código PHP puro con toda la **lógica de negocio** se incluye en el **script del controlador**, como se muestra a continuación.

La parte del controlador, en `index.php`

```
<?php  
// Conectar con la base de datos y seleccionarla  
$conexion = mysql_connect('localhost', 'miusuario', 'micontrasena');  
mysql_select_db('blog_db', $conexion);  
// Ejecutar la consulta SQL  
$resultado = mysql_query('SELECT fecha, titulo FROM articulo', $conexion);  
// Crear el array de elementos para la capa de la vista  
$articulos = array();  
while ($fila = mysql_fetch_array($resultado, MYSQL_ASSOC)) {  
    $articulos[] = $fila;  
}  
// Cerrar la conexión  
mysql_close($conexion);  
// Incluir la lógica de la vista  
require('vista.php');  
?>
```

El código HTML, que contiene cierto código PHP a modo de plantilla, se almacena en el script de la vista, como se muestra a continuación.

La parte de la vista, en `vista.php`

```
<html>  
  <head>  
    <title>Listado de Artículos</title>  
  </head>  
  <body>  
    <h1>Listado de Artículos</h1>  
    <table>  
      <tr><th>Fecha</th><th>Título</th></tr>  
      <?php foreach ($articulos as $articulo): ?>  
        <tr>  
          <td><?php echo $articulo['fecha'] ?></td>  
          <td><?php echo $articulo['titulo'] ?></td>  
        </tr>  
      <?php endforeach; ?>  
    </table>  
  </body>  
</html>
```

Una buena regla general para determinar si la parte de la vista está suficientemente limpia de código es que debería contener una cantidad mínima de código PHP, la suficiente como para que un diseñador HTML sin conocimientos de PHP pueda entenderla. Las instrucciones más comunes en la parte de la vista suelen ser `echo`, `if/else`, `foreach/endforeach` y poco más. Además, no se deben incluir instrucciones PHP que generen etiquetas HTML.

Toda la lógica se ha centralizado en el script del controlador, que solamente contiene código PHP y ningún tipo de HTML. De hecho, y como puedes imaginar, el mismo controlador se puede reutilizar para otros tipos de presentaciones completamente diferentes, como por ejemplo un archivo PDF o una estructura de tipo XML.

Separando la manipulación de los datos

La mayor parte del script del controlador se encarga de la manipulación de los datos. Pero, ¿qué ocurre si se necesita la lista de entradas del blog para otro controlador, por ejemplo uno que se dedica a generar el canal RSS de las entradas del blog? ¿Y si se quieren centralizar todas las consultas a la base de datos en un único sitio para evitar duplicidades?

¿Qué ocurre si cambia el modelo de datos y la tabla artículo pasa a llamarse artículo_blog? ¿Y si se quiere cambiar a [PostgreSQL](#) en vez de [MySQL](#)? Para poder hacer todo esto, es imprescindible eliminar del controlador todo el código que se encarga de la manipulación de los datos y ponerlo en otro script, llamado el modelo, tal y como se muestra a continuación.

La parte del modelo, en `modelo.php`

```
<?php
function getTodosLosArticulos(){
    // Conectar con la base de datos y seleccionarla
    $conexion = mysql_connect('localhost', 'miusuario', 'micontrasena');
    mysql_select_db('blog_db', $conexion);
    // Ejecutar la consulta SQL
    $resultado = mysql_query('SELECT fecha, titulo FROM articulo',$conexion);
    // Crear el array de elementos para la capa de la vista
    $articulos = array();
    while ($fila = mysql_fetch_array($resultado, MYSQL_ASSOC)){
        $articulos[] = $fila;
    }
    // Cerrar la conexión
    mysql_close($conexion);
    return $articulos;
}
?>
```

El controlador modificado se puede ver aquí.

La parte del controlador, modificada, en `index.php`

```
<?php
// Incluir la lógica del modelo
require once('modelo.php');
// Obtener la lista de artículos
$articulos = getTodosLosArticulos();
// Incluir la lógica de la vista
require('vista.php');
?>
```

Ahora el controlador es mucho más fácil de leer. Su única tarea es la de **obtener los datos del modelo y pasárselos a la vista**. En las aplicaciones más complejas, **el controlador** se encarga además de **procesar las peticiones, las sesiones de los usuarios, la autenticación**, etc. El uso de nombres apropiados para las funciones del modelo hace que sea innecesario añadir comentarios al código del controlador.

El script del **modelo solamente se encarga del acceso a los datos** y puede ser reorganizado a tal efecto. Todos los parámetros que no dependen de la **capa de datos**(como por ejemplo los parámetros de la petición del usuario) se deben obtener a través del controlador y por tanto, no se puede acceder a ellos directamente desde el modelo. Las funciones del modelo se pueden reutilizar fácilmente en otros controladores.

Separación en capas más allá del MVC

El principio más importante de la arquitectura MVC es la separación del código del programa en **tres capas**, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo, el código de **la presentación** en la vista y la lógica de la aplicación en el controlador.

La programación se puede simplificar si se utilizan otros patrones de diseño. De esta forma, las **capas del modelo, la vista y el controlador** se pueden subdividir en más **capas**.

Abstracción de la base de datos

La capa del modelo se puede dividir en la **capa de acceso a los datos** y en la **capa de abstracción de la base de datos**. De esta forma, las funciones que acceden a los datos no utilizan sentencias ni consultas que dependen de una base de datos, sino que utilizan otras funciones para realizar las consultas. Así, si se cambia de sistema gestor de bases de datos, solamente es necesario actualizar la **capa de abstracción de la base de datos**.

La parte del modelo correspondiente a la abstracción de la base de datos: muestra una capa de acceso a datos específica para MySQL.

```
<?php
function crear_conexion($servidor, $usuario, $contrasena){
    return mysql_connect($servidor, $usuario, $contrasena);
}
function cerrar_conexion($conexion){
    mysql_close($conexion);
}
function consulta_base_de_datos($consulta, $base_datos, $conexion){
    mysql_select_db($base_datos, $conexion);
    return mysql_query($consulta, $conexion);
}
function obtener_resultados($resultado){
    return mysql_fetch_array($resultado, MYSQL_ASSOC);
}
?>
```

La parte del modelo correspondiente al acceso a los datos: muestra una capa sencilla de abstracción de la base de datos.

```
<?php
function getTodosLosArticulos(){
    // Conectar con la base de datos
    $conexion = crear_conexion('localhost', 'miusuario', 'micontrasena');
    // Ejecutar la consulta SQL
    $resultado = consulta_base_de_datos('SELECT fecha, titulo FROM articulo', 'blog_db',
    $conexion);
    // Crear el array de elementos para la capa de la vista
    $articulos = array();
    while ($fila = obtener_resultados($resultado)){
        $articulos[] = $fila;
    }
    // Cerrar la conexión
    cerrar_conexion($conexion);
    return $articulos;
}
?>
```

Como se puede comprobar, **la capa de acceso a datos** no contiene funciones dependientes de ningún sistema gestor de bases de datos, por lo que es independiente de la base de datos utilizada. Además, las funciones creadas en la capa de abstracción de la base de datos se pueden reutilizar en otras funciones del modelo que necesiten acceder a la base de datos.

NOTA

Estos últimos dos ejemplos no son completos, y todavía hace falta añadir algo de código para tener una completa abstracción de la base de datos (**abstraer el código SQL** mediante un **constructor de**

consultas independiente de la base de datos, añadir todas las funciones a una clase, etc.) El propósito de este artículo no es mostrar cómo se puede escribir todo ese código.

Los elementos de la vista

La **capa de la vista** también puede aprovechar la **separación de código**. Las páginas web suelen contener elementos que se muestran de forma idéntica a lo largo de toda la aplicación: cabeceras de la página, el layout genérico, el pie de página y la navegación global. Normalmente sólo cambia el interior de la página. Por este motivo, la vista se separa en un **layout** y en una plantilla. Normalmente, el layout es global en toda la aplicación o al menos en un grupo de páginas. La plantilla sólo se encarga de visualizar las variables definidas en el controlador. Para que estos componentes interactúen entre sí correctamente, es necesario añadir cierto código. Siguiendo estos principios, la parte de la vista del scrip inicial se puede separar en tres partes.

La parte de la plantilla de la vista, en `miplantilla.php`

```
<h1>Listado de Artículos</h1>
<table>
  <tr><th>Fecha</th><th>Título</th></tr>
  <?php foreach ($articulos as $articulo): ?>
    <tr>
      <td><?php echo $articulo['fecha'] ?></td>
      <td><?php echo $articulo['titulo'] ?></td>
    </tr>
  <?php endforeach; ?>
</table>
```

La parte de la lógica de la vista

```
<?php
  $titulo = 'Listado de Artículos';
  $contenido = include('miplantilla.php');
?>
```

La parte del layout de la vista

```
<html>
  <head>
    <title><?php echo $titulo ?></title>
  </head>
  <body>
    <?php echo $contenido ?>
  </body>
</html>
```

Acciones y controlador frontal

En el ejemplo anterior, el **controlador** no se encargaba de realizar muchas tareas, pero en las aplicaciones web reales el controlador suele tener mucho trabajo. Una parte importante de su trabajo es común a todos los controladores de la aplicación. Entre las tareas comunes se encuentran el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras tareas similares. Por este motivo, el controlador normalmente se divide en un **controlador frontal**, que es único para cada aplicación, y las acciones, que incluyen el código específico del controlador de cada página.

Una de las principales ventajas de utilizar un **controlador frontal** es que **ofrece un punto de entrada único para toda la aplicación**. Así, en caso de que sea necesario impedir el acceso a la aplicación, solamente es necesario editar el script correspondiente al *controlador frontal*. Si la aplicación no dispone de controlador frontal, se debería modificar cada uno de los controladores.

Orientación a objetos

Los ejemplos anteriores utilizan la **programación procedimental**. Las posibilidades que ofrecen los lenguajes de programación modernos para trabajar con objetos permiten simplificar la programación, ya que los objetos pueden *encapsular la lógica, heredar métodos y atributos* entre

diferentes objetos y proporcionan una serie de convenciones claras sobre la forma de nombrar a los objetos.

La implementación de una arquitectura MVC en un lenguaje de programación que no está orientado a objetos puede encontrarse con problemas de *namespaces* y *código duplicado*, dificultando la lectura del código de la aplicación.

La orientación a objetos permite a los *desarrolladores* trabajar con *objetos de la vista*, **objetos del controlador** y **clases del modelo**, transformando las funciones de los ejemplos anteriores en métodos. Se trata de un requisito obligatorio para las arquitecturas de tipo MVC.

TEMA 6

Contenido

1.- Servicios web	- 2 -
1.1.- Características.	- 3 -
1.2.- Intercambio de información: SOAP.....	- 4 -
1.3.- Intercambio de información: SOAP (II).....	- 5 -
1.4.- Descripción del servicio: WSDL.	- 6 -
1.5.- Descripción del servicio: WSDL (II).	- 7 -
1.6.- Descripción del servicio: WSDL (III).	- 9 -
1.7.- Descripción del servicio: WSDL (IV).	- 10 -
1.8.- Descripción del servicio: WSDL (V).	- 11 -
1.9.- Descripción del servicio: WSDL (VI).	- 12 -
2.- Extensión PHP5 SOAP.....	- 14 -
2.1.- Utilización de un servicio web.	- 14 -
2.2.- Utilización de un servicio web (II).	- 16 -
2.3.- Utilización de un servicio web (III).	- 16 -
Ejercicio resuelto	- 17 -
2.4.- Utilización de un servicio web (IV).	- 18 -
2.5.- Creación de un servicio web.	- 19 -
2.6.- Creación de un servicio web (II).	- 20 -
2.7.- Creación de un servicio web (III).	- 21 -
2.8.- Creación de un servicio web (IV).	- 22 -
Anexo I - CurrencyConvertor.wsdl.....	- 24 -
Anexo II - Fichero con código PHP	- 30 -
Anexo III - globalweather.wsdl	- 34 -

Servicios web.

Caso práctico

En BK Programación, Juan ha estado diseñando la aplicación web que deben desarrollar. El trabajo le está sirviendo para actualizar sus conocimientos en el lenguaje PHP. Y al mismo tiempo, se está dando cuenta de que cuanto más conoce, más herramientas puede utilizar para programar.

Uno de los detalles que le preocupa del nuevo proyecto, es la posibilidad de reutilizar en el futuro parte del código que se genere. Por ejemplo, si se crea una función para ver las unidades almacenadas, en las tiendas, de un producto concreto, sería bueno que esa función se pudiera usar no sólo desde la propia aplicación web, sino también desde cualquier otra aplicación que pueda necesitar esa información.

Lo ha estado hablando con Esteban, y antes de avanzar más han decidido tomarse un tiempo para evaluar las distintas posibilidades con las que cuentan al respecto. Seguramente retrase un tiempo el proyecto, pero a cambio la aplicación que obtendrán resultará más abierta y su información podrá aprovecharse de forma sencilla cuando sea necesario.

1.- Servicios web

Caso práctico

De su trabajo anterior con la programación de aplicaciones, Juan conoce algunos mecanismos que podrían aplicarse en este caso. En uno de sus primeros proyectos utilizó una técnica llamada RPC, que le permitía a un programa ejecutar de forma remota funciones que se encontraban en otro equipo y obtener su resultado.

Pero sabe que ahora existen otros métodos más potentes de funcionamiento similar: los servicios web. Aunque nunca los ha utilizado, tiene una idea general sobre su funcionamiento. Tiene que profundizar sobre ellos para ver si se adaptan a lo que busca y conocer de forma más precisa cómo se pueden integrar con una aplicación web programada en lenguaje PHP.

En ocasiones, las aplicaciones que desarrolles necesitarán compartir información con otras aplicaciones.

Sin ir más lejos, cojamos la aplicación de tienda web que estuvimos utilizando en los ejemplos y ejercicios del módulo. La información que se almacena sobre los productos incluye su código, nombre, descripción, PVP, etc. Seguramente los proveedores a los que se compran los artículos, manejen la misma o parecida información. Y quizás puedas aprovechar esa información para tu propia aplicación.

O puede ser que, una vez que esté finalizada y funcionando, quieras programar una nueva aplicación (y no necesariamente una aplicación web) que la complemente para, por ejemplo, procesar la información sobre los pedidos realizados.

Para compartir la información que gestiona tu aplicación, normalmente es suficiente con dar acceso a la base de datos en que se almacena. Pero ésta generalmente no es una buena idea. Cuantas más aplicaciones utilicen los mismos datos, más posibilidades hay de que se generen errores en los mismos. Además, existen otros inconvenientes:

- ✓ Si ya tienes una aplicación funcionando, ya has programado la lógica de negocio correspondiente, y ésta no se podrá aprovechar en otras aplicaciones si utilizan directamente la información almacenada en la base de datos.
- ✓ Si quieres poner la base de datos a disposición de terceros, éstos necesitarán conocer su estructura. Y al dar acceso directo a los datos, será complicado mantener el control sobre las modificaciones que se produzcan en los mismos.

Por otro lado, gran parte de la información que gestionan las aplicaciones web ya está disponible para que otros la utilicen (dejando a un lado las consideraciones relacionadas con el control de

acceso). Por ejemplo, si alguien quiere conocer el precio de un producto en la tienda web, basta con buscar ese producto en la página en que se listan todos los productos. Pero, para que esa misma información (el precio de un producto) la pueda obtener un programa, éste tendría que contemplar un procedimiento para buscar el producto concreto dentro de las etiquetas HTML de la página y extraer su precio.

Para facilitar esta tarea existen los servicios web. Un servicio web es un método que permite que dos equipos intercambien información a través de una red informática. Al utilizar servicios web, el servidor puede ofrecer un punto de acceso a la información que quiere compartir. De esta forma controla y facilita el acceso a la misma por parte de otras aplicaciones.

Los clientes del servicio, por su parte, no necesitan conocer la estructura interna de almacenamiento. En lugar de tener que programar un mecanismo para localizar la información, tienen un punto de acceso directo a la que les interesa.

Volviendo al ejemplo de nuestra tienda, si quisiéramos aprovechar la información de que disponen nuestros proveedores, éstos tendrían que ofrecer un servicio web que nos permitiese recuperarla. Por ejemplo, enviándonos el código de un producto, podríamos obtener su nombre, descripción, precio, etc. Inversamente, si quisiéramos facilitar la obtención de datos de nuestra tienda por parte de otras aplicaciones, podríamos programar y ofrecer un servicio web de forma que, por ejemplo, devolviese el listado de pedidos del cliente que se requiera.

1.1.- Características.

Existen numerosos protocolos que permiten la comunicación entre ordenadores a través de una red: FTP, HTTP, SMTP, POP3, TELNET, etc. En todos estos protocolos se definen un servidor y un cliente. El servidor es la máquina que está esperando conexiones (escuchando) por parte de un cliente. El cliente es la máquina que inicia la comunicación. Cada uno de estos protocolos tiene asignado además un puerto (TCP o UDP) concreto, que será el que utilicen normalmente los equipos servidores.

Cada uno de los protocolos que hemos nombrado ha sido creado para un fin específico: FTP para transferencia de archivos, HTTP para páginas web, SMTP y POP3 para correo electrónico y TELNET para acceso remoto. No han sido diseñados para transportar peticiones de información genéricas entre aplicaciones, como solicitar el PVP de un producto. Sin embargo, ya desde hace tiempo existen otras soluciones para este tipo de problemas. Una de las más populares es RPC.

El protocolo RPC se creó para permitir a un sistema acceder de forma remota a funciones o procedimientos que se encuentren en otro sistema. El cliente se conecta con el servidor, y le indica qué función debe ejecutar. El servidor la ejecuta y le devuelve el resultado obtenido. Así, por ejemplo, podemos crear en el servidor RPC una función que reciba un código de producto y devuelva su PVP.

RPC usa su propio puerto, pero normalmente solo a modo de directorio. Los clientes se conectan a él para obtener el puerto real del servicio que les interesa. Este puerto no es fijo; se asigna de forma dinámica.

Los servicios web se crearon para permitir el intercambio de información al igual que RPC, pero sobre la base del protocolo HTTP (de ahí el término web). En lugar de definir su propio protocolo para transportar las peticiones de información, utilizan HTTP para este fin. La respuesta obtenida no será una página web, sino la información que se solicitó. De esta forma pueden funcionar sobre cualquier servidor web; y, lo que es aún más importante, utilizando el puerto 80 reservado para este protocolo. Por tanto, cualquier ordenador que pueda consultar una página web, podrá también

solicitar información de un servicio web. Si existe algún cortafuegos en la red, tratará la petición de información igual que lo haría con la solicitud de una página web.



Existen al menos dos cuestiones que debería resolver un servicio web para poder funcionar correctamente:

- ✓ Cómo se transmite la información. Si se va a usar HTTP para las peticiones y las respuestas, el cliente y el servidor tendrán que ponerse de acuerdo en la forma de enviar unas y otras. Es decir, ¿cómo hace el cliente para indicar que quiere conocer el PVP del artículo con código X?, y también, ¿cómo envía el servidor la respuesta obtenida?
- ✓ Cómo se publican las funciones a las que se puede acceder en un servidor determinado. Este punto es opcional, pero muy útil. Es decir, el cliente puede saber que la función del servidor que tiene que utilizar se llama `getPVPArticulo`, y que debe recibir como parámetro el código del artículo. Pero si no lo sabe, sería útil que hubiera un mecanismo donde pudiera consultar las funciones que existen en el servidor y cómo se utiliza cada una.

Cada uno de los métodos que podemos utilizar hoy en día para crear un servicio web responde a estas preguntas de formas distintas. Para la primera cuestión, nosotros veremos el protocolo SOAP, que utiliza el lenguaje XML para intercambiar información. En cuanto a la segunda cuestión, la resolveremos con un lenguaje llamado WSDL, que también está basado en XML y fue creado para describir servicios web, es decir, indicar cómo se debe acceder a un servicio y utilizarlo.

Relaciona las siglas con aquello a que hacen referencia:

Siglas	Relación	Significado
SOAP.	3	1. Protocolo para transmitir páginas web.
HTTP.	1	2. Protocolo para ejecutar código de forma remota.
RPC.	2	3. Protocolo para intercambiar información en un servicio web.
WSDL.	4	4. Lenguaje para describir servicios web.

1.2.- Intercambio de información: SOAP.

SOAP es un protocolo que indica cómo deben ser los mensajes que se intercambien el servidor y el cliente, cómo deben procesarse éstos, y cómo se relacionan con el protocolo que se utiliza para transportarlos de un extremo a otro de la comunicación (en el caso de los servicios web, este protocolo será HTTP).

Aunque nosotros vamos a utilizar HTTP para transmitir la información, SOAP no requiere el uso de un protocolo concreto para transmitir la información. SOAP se limita a definir las reglas que rigen los mensajes que se deben intercambiar el cliente y el servidor. Cómo se envíen esos mensajes no es relevante desde el punto de vista de SOAP. En lugar de utilizar HTTP para transmitirlos, se podrían utilizar, por ejemplo, correos electrónicos (claro que en este caso ya no sería un servicio web).

El nombre SOAP surgió como acrónimo de **S**imple **O**bject **A**ccess **P**rotocol, pero, a partir de la versión 1.2 del protocolo, el nombre SOAP ya no se refiere a nada en concreto.

Al igual que su antecesor, XML-RPC, SOAP utiliza XML para componer los mensajes que se transmiten entre el cliente (que genera una petición) y el servidor (que envía una respuesta) del servicio web.



Veamos un ejemplo. Si implementamos un servicio web para informar sobre el precio de los artículos que se venden en la tienda web, una petición de información para el artículo con código 'KSTMSDHC8GB' podría ser de la siguiente forma:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns1=""
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
>
  <soap:Body>
    <ns1:getPVP>
      <param0 xsi:type="xsd:string">KSTMSDHC8GB</param0>
    </ns1:getPVP>
  </soap:Body>
</soap:Envelope>
```

Y su respectiva respuesta:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns1=""
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
>
  <soap:Body>
    <ns1:getPVPResponse>
      <return xsi:type="xsd:string">10.20</return>
    </ns1:getPVPResponse>
  </soap:Body>
</soap:Envelope>
```

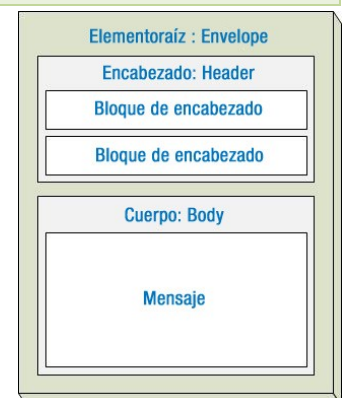
1.3.- Intercambio de información: SOAP (II).

En un mensaje SOAP, como mínimo debe figurar un elemento **Envelope**, que es lo que identifica al documento XML como un mensaje SOAP, y donde se deben declarar al menos los siguientes espacios de nombres:

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

El espacio de nombres que se utilice para el elemento **Envelope** indica la versión del protocolo SOAP utilizado. En la versión 1.1 (la del ejemplo anterior), el espacio de nombres es <http://schemas.xmlsoap.org/soap/envelope/>. En la versión 1.2 se debe utilizar <http://www.w3.org/2003/05/soap-envelope>.

Al cambiar la versión de SOAP, también se deben cambiar los espacios de nombres relativos al estilo de codificación. En la versión 1.1, se debe utilizar <http://schemas.xmlsoap.org/soap/encoding/>, y en la versión 1.2 <http://www.w3.org/2003/05/soap-encoding>.



Como primer miembro del elemento **Envelope**, puede haber de forma opcional un elemento **Header**. Si existe, puede contener varios elementos con información adicional sobre cómo procesar el mensaje SOAP. A continuación debe figurar obligatoriamente un elemento **Body**, que es dónde se incluye, dependiendo del tipo de mensaje, la petición o la respuesta.

Sería muy complejo programar un servicio web que procesase el XML recibido en cada petición SOAP, y generase el XML relativo a cada respuesta correspondiente. Existen mecanismos de ayuda que nos evitan tener que tratar con las interioridades del protocolo SOAP.

De las implementaciones de SOAP que podemos usar con PHP, cabe destacar tres: **NuSOAP**, **PEAR::SOAP** y **PHP5 SOAP**. Las tres nos permiten crear tanto un cliente como un servidor SOAP, pero existen algunas características que las diferencian:

- ✓ **PHP5 SOAP** es la implementación de SOAP que se incluye con PHP a partir de la versión 5 del lenguaje. En versiones anteriores se tenía que recurrir a otras opciones para trabajar con SOAP. Es una extensión nativa (escrita en lenguaje C) y por tanto más rápida que las otras posibilidades. Como veremos más adelante, su gran inconveniente es que no permite la generación automática del documento WSDL una vez programado el servidor SOAP correspondiente.
- ✓ **NuSOAP** es un conjunto de clases programadas en PHP que ofrecen muchas funcionalidades para utilizar SOAP. Al contrario que PHP5 SOAP, funcionan también con PHP4, y además permite generar automáticamente el documento WSDL correspondiente a un servicio web.
- ✓ **PEAR::SOAP** es un paquete PEAR que permite utilizar SOAP con PHP a partir de su versión 4. Al igual que NuSOAP, también está programado en PHP.

Debido a una coincidencia en el nombre de las clases, NuSOAP es incompatible con PHP5 SOAP. Ambas incluyen una clase de nombre SoapClient. Si quieres programar con NuSOAP en PHP5, es recomendable cambiar el nombre de esta clase o utilizar la alternativa NuSOAP for PHP5, que utiliza en su lugar el nombre SOAPClientNuSOAP.

<https://code.google.com/p/nusoap-for-php5/>

Más adelante aprenderás a crear y utilizar servicios web desde PHP5 con PHP5 SOAP.

El elemento Envelope debe figurar como raíz en un mensaje SOAP, y obligatoriamente deberá contener un elemento:



Header.



Body.

Efectivamente. El elemento Header es opcional, pero el elemento Body debe figurar obligatoriamente dentro del elemento Envelope.

1.4.- Descripción del servicio: WSDL.

Una vez que hayas creado un servicio web, puedes programar el correspondiente cliente y comenzar a utilizarlo. Como el servicio lo has creado tú, sabrás cómo acceder a él: en qué URL está accesible, qué parámetros recibe, cuál es la funcionalidad que aporta, y qué valores devuelve. Sin embargo, si lo que quieres es que el servicio web sea accesible a aplicaciones desarrolladas por otros programadores, deberás indicarles cómo usarlo, es decir, crear un documento WSDL que describa el servicio.

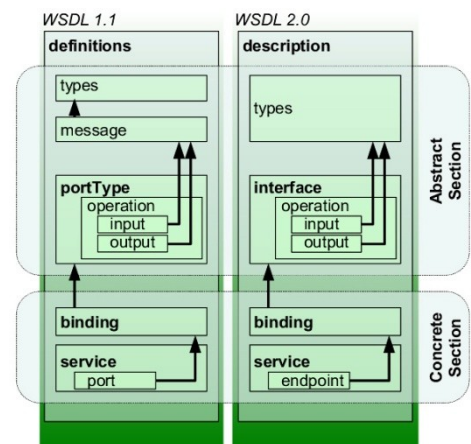
WSDL es un lenguaje basado en XML que utiliza unas reglas determinadas para generar el documento de descripción de un servicio web. Una vez generado, ese documento se suele poner a disposición de los posibles usuarios del servicio (normalmente se accede al documento WSDL añadiendo **?wsdl** a la URL del servicio).

El espacio de nombres de un documento WSDL es <http://schemas.xmlsoap.org/wsdl/>, aunque en un documento WSDL se suelen utilizar también otros espacios de nombres. La estructura de un documento WSDL es la siguiente:

```
<definitions
  name="..."
  targetNamespace="http://..."
  xmlns:tns="http://..."
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  ...
>
  <types>
    ...
  </types>
  <message>
    ...
  </message>
  <portType>
    ...
  </portType>
  <binding>
    ...
  </binding>
  <service>
    ...
  </service>
</definitions>
```

El objetivo de cada una de las secciones del documento es el siguiente:

- ✓ **types**. Incluye las definiciones de los tipos de datos que se usan en el servicio.
- ✓ **message**. Define conjuntos de datos, como la lista de parámetros que recibe una función o los valores que devuelve.
- ✓ **portType**. Cada **portType** es un grupo de funciones que implementa el servicio web. Cada función se define dentro de su **portType** como una operación (**operation**).
- ✓ **binding**. Define cómo va a transmitirse la información de cada **portType**.
- ✓ **service**. Contiene una lista de elementos de tipo **port**. Cada **port** indica dónde (en qué URL) se puede acceder al servicio web.



Las secciones anteriores son las correspondientes a la versión 1.1 de WSDL. En la versión 2.0, los conceptos y la nomenclatura cambia ligeramente; por ejemplo, lo que en 1.1 es un **portType** se denomina **interface** en la versión 2.0.

En los siguientes puntos veremos paso a paso cómo crear cada una de las secciones de un documento WSDL.

1.5.- Descripción del servicio: WSDL (II).

Existen servicios web sencillos a los que puedes pasar como parámetro un número o una cadena de texto (por ejemplo, las siglas de una moneda, **USD**), y te devuelven también un dato de un tipo simple, como un número decimal (la tasa de conversión actual). E igualmente existen también servicios web más elaborados, que pueden requerir o devolver un array de elementos, o incluso objetos.

Para crear y utilizar estos servicios, deberás definir los tipos de elementos que se transmiten: de qué tipo son los valores del array, o qué miembros poseen los objetos que maneja. La definición de tipos en WSDL se realiza utilizando la etiqueta **types**. Veamos un ejemplo:

```

<types>
  <xsd:schema targetNamespace="http://localhost/dwes/ut6">
    <xsd:complexType name="direccion">
      <xsd:all>
        <xsd:element name="ciudad" type="xsd:string"/>
        <xsd:element name="calle" type="xsd:string"/>
        <xsd:element name="numero" type="xsd:string"/>
        <xsd:element name="piso" type="xsd:string"/>
        <xsd:element name="CP" type="xsd:string"/>
      </xsd:all>
    </xsd:complexType>
    <xsd:complexType name="usuario">
      <xsd:all>
        <xsd:element name="id" type="xsd:int"/>
        <xsd:element name="nombre" type="xsd:string"/>
        <xsd:element name="direccion" type="tns:direccion"/>
        <xsd:element name="email" type="xsd:string"/>
      </xsd:all>
    </xsd:complexType>
  </xsd:schema>
</types>

```

En el código anterior, se definen dos tipos de datos usando XML Schema: **direccion** y **usuario**. De hecho, los tipos **direccion** y **usuario** son la forma en que se definen en WSDL las clases para transmitir la información de sus objetos.



En WSDL, las clases se definen utilizando los tipos complejos de XML Schema. Al utilizar **all** dentro del tipo complejo, estamos indicando que la clase contiene esos miembros, aunque no necesariamente en el orden que se indica (si en lugar de **all** hubiésemos utilizado **sequence**, el orden de los miembros de la clase debería ser el mismo que figura en el documento).

Obviamente, los métodos de la clase forman parte de la lógica de la aplicación y no se definen en el documento WSDL.

Aunque en WSDL se puede usar cualquier lenguaje para definir los tipos de datos, es aconsejable usar XML Schema, indicándolo dentro de la etiqueta **types** e incluyendo el espacio de nombres correspondiente en el elemento **definitions**.

```

<definitions
  name="WSDLusuario"
  targetNamespace="http://localhost/dwes/ut6"
  xmlns:tns="http://localhost/dwes/ut6"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  ...
>

```

El otro tipo de datos que necesitaremos definir en los documentos WSDL son los **arrays**. Para definir un array, no existe en el XML Schema un tipo base adecuado que podamos usar. En su lugar, se utiliza el tipo **Array** definido en el esquema **encoding** de SOAP. Por ejemplo, podríamos añadir un tipo array de usuarios al documento anterior haciendo:

```
<xsd:complexType name="ArrayOfusuario">
  <xsd:complexContent>
    <xsd:restriction base="soapenc:Array">
      <xsd:attribute
        ref="soapenc:arrayType" arrayType="tns:usuario[]"
      />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

Al definir un array en WSDL, se debe tener en cuenta que:

- ✓ El atributo **arrayType** se utiliza para indicar qué elementos contendrá el array.
- ✓ Se debe añadir al documento el espacio de nombres **SOAP encoding**:

```
<definitions
  name="WSDLusuario"
  targetNamespace="http://localhost/dwes/ut6"
  xmlns:tns="http://localhost/dwes/ut6"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  ...
>
```

- ✓ El nombre del array debería ser **ArrayOfXXX**, dónde **XXX** es el nombre del tipo de elementos que contiene el array.

En muchas ocasiones no será necesario definir tipos propios, y por tanto en el documento WSDL no habrá sección **types**; será suficiente con utilizar alguno de los tipos propios de XML Schema, como **xsd:string**, **xsd:float** o **xsd:boolean**.

En la sección **types** de un documento WSDL, se deben definir:



Todos los tipos de elementos que se usen en el servicio web.



Los tipos de elementos compuestos que se usen en el servicio web, como los objetos y arrays.

Al usar como base los tipos propios de XML Schema, solo será necesario definir aquellos tipos compuestos que se utilicen en el servicio web.

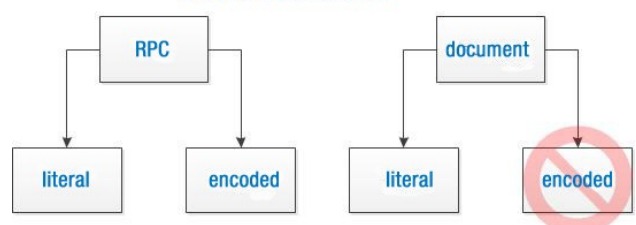
1.6.- Descripción del servicio: WSDL (III).

Ahora que ya sabes cómo definir los tipos de datos que se usan en un servicio web, el siguiente paso es indicar cómo se agrupan esos tipos para formar los parámetros de entrada y de salida. Veámoslo con un ejemplo. Siguiendo con los usuarios que acabamos de definir, podríamos crear en el servicio web una función **getUsuario** para dar acceso a los datos de un usuario. Como parámetro de entrada de esa función vamos a pedir el **id** del usuario, y como valor de salida se obtendrá un objeto **usuario**. Por tanto, debemos definir los siguientes mensajes:

```
<message name="getUsuarioRequest">
  <part name="id" type="xsd:int"/>
</message>
<message name="getUsuarioResponse">
  <part name="getUsuarioReturn" type="tns:usuario"/>
</message>
```

Como ves, normalmente por cada función del servicio web se crea un mensaje para los parámetros de entrada, y otro para los de salida. Dentro de cada mensaje, se incluirán tantos elementos **part** como sea necesario. Cada mensaje contendrá un atributo **name** que

Estilos de enlazado



debe ser único para todos los elementos de este tipo. Además, es aconsejable que el nombre del mensaje con los parámetros de entrada acabe en **Request**, y el correspondiente a los parámetros de salida en **Response**.

En un documento WSDL podemos especificar dos estilos de enlazado: **document** o **RPC**. La selección que hagamos influirá en cómo se transmitan los mensajes dentro de las peticiones y respuestas SOAP. Por ejemplo, un mensaje SOAP con estilo **document** podría ser:

```
<SOAP-ENV:Body>
  <producto>
    <codigo>KSTMSDHC8GB</codigo>
  </producto>
</SOAP-ENV:Body>
```

Y un mensaje con estilo **RPC** sería por ejemplo:

```
<SOAP-ENV:Body>
  <ns1:getPVP>
    <param0 xsi:type="xsd:string">KSTMSDHC8GB</param0>
  </ns1:getPVP>
</SOAP-ENV:Body>
```

El estilo de enlazado **RPC** está más orientado a sistemas de petición y respuesta que el **document** (más orientado a la transmisión de documentos en formato XML). En este estilo de enlazado, cada elemento **message** de WSDL debe contener un elemento **part** por cada parámetro (de entrada o de salida), y dentro de éste indicar el tipo de datos del parámetro mediante un atributo **type**, como se muestra en el ejemplo anterior.

Además, cada estilo de enlazado puede ser de tipo **encoded** o **literal** (aunque en realidad la combinación **document/encoded** no se utiliza). Al indicar **encoded**, estamos diciendo que vamos a usar un conjunto de reglas de codificación, como las que se incluyen en el propio protocolo SOAP (espacio de nombres <http://schemas.xmlsoap.org/soap/encoding/>), para convertir en XML los parámetros de las peticiones y respuestas.

El ejemplo anterior de **RPC** es en realidad **RPC/encoded**. Un ejemplo de un mensaje SOAP con estilo **RPC/literal** sería:

```
<SOAP-ENV:Body>
  <ns1:getPVP>
    <param0>KSTMSDHC8GB</param0>
  </ns1:getPVP>
</SOAP-ENV:Body>
```

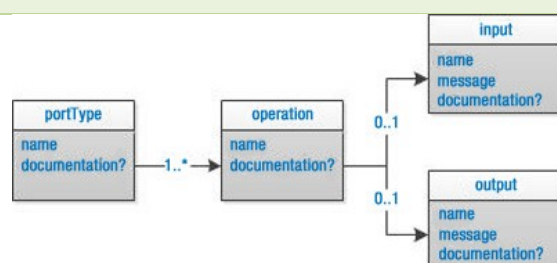
En lo sucesivo, trabajaremos únicamente con estilo de enlazado **RPC/encoded**.

1.7.- Descripción del servicio: WSDL (IV).

Las funciones que creas en un servicio web, se conocen con el nombre de **operaciones** en un documento WSDL. En lugar de definir las una a una, es necesario agruparlas en lo que en WSDL se llama **portType**. Un **portType** contiene una lista de funciones, indicando para cada función (**operation**) la lista de parámetros de entrada y de salida que le corresponden. Por ejemplo:

```
<portType name="usuarioPortType">
  <operation name="getUsuario">
    <input message="tns:getUsuarioRequest"/>
    <output message="tns:getUsuarioResponse"/>
  </operation>
</portType>
```

A no ser que estés generando un servicio web bastante complejo, el documento WSDL contendrá un único **portType**. Podrías necesitar dividir las funciones del servicio en distintos **portType** para, por ejemplo,



utilizar un estilo de enlazado distinto para las funciones de cada grupo.

Cada `portType` debe contener un atributo `name` con el nombre (único para todos los elementos `portType`). Cada elemento `operation` también debe contener un atributo `name`, que se corresponderá con el nombre de la función que se ofrece. Además, en función del tipo de operación de que se trate, contendrá:

- ✓ Un elemento `input` para indicar funciones que no devuelven valor (su objetivo es sólo enviar un mensaje al servidor).
- ✓ Un elemento `input` y otro `output`, en este orden, para el caso más habitual: funciones que reciben algún parámetro, se ejecutan, y devuelven un resultado.

Es posible (pero muy extraño) encontrarse funciones a la inversa: sólo con un parámetro `output` (el servidor envía una notificación al cliente) o con los parámetros `output` e `input` por ese orden (el servidor le pide al cliente alguna información). Por tanto, al definir una función (un elemento `operation`) se debe tener cuidado con el orden de los elementos `input` y `output`.

Normalmente, los elementos `input` y `output` contendrán un atributo `message` para hacer referencia a un mensaje definido anteriormente.

En un documento WSDL, cada una de las funciones que implementa el servicio se refleja en un elemento de tipo:

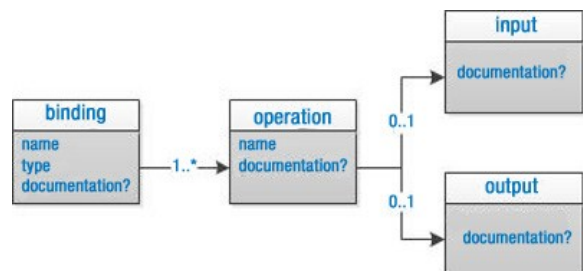
- `operation`.
- `portType`.

Efectivamente, pues el elemento `portType` solo es una forma de agrupar funciones (elementos `operation`), y únicamente se utiliza en caso de servicios complejos.

1.8.- Descripción del servicio: WSDL (V).

El siguiente elemento de un documento WSDL es `binding`. Antes comentábamos que existían distintos estilos de enlazado, que influían en cómo se debían crear los mensajes. En el elemento `binding` es dónde debes indicar que el estilo de enlazado de tu documento sea `RPC/encoded`.

Aunque es posible crear documentos WSDL con varios elementos `binding`, la mayoría contendrán solo uno (si no fuera así, sus atributos `name` deberán ser distintos). En él, para cada una de las funciones (`operation`) del `portType` que acabamos de crear, se deberá indicar cómo se codifica y transmite la información.



Para el `portType` anterior, podemos crear un elemento `binding` como el siguiente:

```
<binding name="usuarioBinding" type="tns:usuarioPortType">
  <soap:binding
    style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"
  />
  <operation name="getUsuario">
    <soap:operation
      soapAction="http://localhost/dwes/ut6/getUsuario.php?getUsuario"
    />
    <input>
      <soap:body
        use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://localhost/dwes/ut6"
      />
    />
    </input>
    <output>
      <soap:body
        use="encoded"
      />
    />
  />
</binding>
```

```

        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://localhost/dwes/ut6"
    />
</output>
</operation>
</binding>

```

Fíjate que el atributo `type` hace referencia al `portType` creado anteriormente. El siguiente elemento indica el tipo de codificación (`RPC`) y, mediante la URL correspondiente, el protocolo de transporte a utilizar (`HTTP`). Obviamente, deberás añadir el correspondiente espacio de nombres al elemento raíz:

```

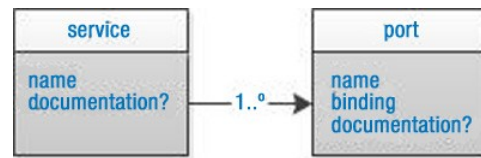
<definitions
  name="WSDLusuario"
  targetNamespace="http://localhost/dwes/ut6"
  xmlns:tns="http://localhost/dwes/ut6"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  ...
>

```

El elemento `soap:operation` debe contener un atributo `soapAction` con la URL para esa función (`operation`) en particular. Dentro de él habrá normalmente un elemento `input` y otro `output` (los mismos que en la `operation` correspondiente). En ellos, mediante los atributos del elemento `soap:body`, se indica el estilo concreto de enlazado (`encoded` con su `encodingStyle` correspondiente).

1.9.- Descripción del servicio: WSDL (VI).

Por último, falta definir el elemento `service`. Normalmente sólo encontraremos un elemento `service` en cada documento WSDL. En él, se hará referencia al `binding` anterior utilizando un elemento `port`, y se indicará la URL en la que se puede acceder al servicio.



Por ejemplo:

```

<service name="usuario">
  <port name="usuarioPort" binding="tns:usuarioBinding">
    <soap:address
      location="http://localhost/dwes/ut6/getUsuario.php"
    />
  </port>
</service>

```

Para finalizar, veamos cómo quedaría el documento WSDL correspondiente a un servicio con una única función encargada de devolver el PVP de un producto de la tienda web a partir de su código.

```

<?xml version="1.0" encoding="UTF-8"?>
<definitions
  name="WSDLgetPVP"
  targetNamespace="http://localhost/dwes/ut6"
  xmlns:tns="http://localhost/dwes/ut6"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
>
  <message name="getPVPRequest">
    <part name="codigo" type="xsd:string"/>
  </message>
  <message name="getPVPResponse">
    <part name="PVP" element="xsd:float"/>
  </message>
  <portType name="getPVPPortType">
    <operation name="getPVP">
      <input message="tns:getPVPRequest"/>
      <output message="tns:getPVPResponse"/>
    </operation>
  </portType>

```

```

    </operation>
  </portType>
  <binding name="getPVPBinding" type="tns:getPVPPortType">
    <soap:binding
style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"
/>
    <operation name="getPVP">
      <soap:operation
        soapAction="http://localhost/dwes/ut6/getPVP.php?getPVP"
      />
      <input>
        <soap:body
namespace="http://localhost/dwes/ut6"
use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
/>
      </input>
      <output>
        <soap:body
namespace="http://localhost/dwes/ut6"
use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
/>
      </output>
    </operation>
  </binding>
  <service name="getPVPService">
    <port name="getPVPPort" binding="tns:getPVPBinding ">
      <soap:address location="http://localhost/dwes/ut6/getPVP.php"/>
    </port>
  </service>
</definitions>

```

Relaciona los elementos que componen un documento WSDL, con la parte del servicio web a que hacen referencia:

Elemento	Relación	Hace referencia a
types	4	1. Conjunto de datos, como las listas de parámetros de las funciones.
definitions	3	2. Caracterización del servicio, que incluye la URL de acceso.
message	1	3. Elemento raíz de un documento WSDL.
port	2	4. Definición de los tipos de elementos usados en el servicio.

2.- Extensión PHP5 SOAP.

Caso práctico

Juan ya tiene claro cuál es la solución que ha estado buscando: los servicios web. Ha estado leyendo sobre ellos y realizando algunas pruebas de funcionamiento, y se ajustan de forma precisa a lo que necesita.

Además ha visto que tiene distintas posibilidades para utilizar servicios web desde PHP y ha investigado sobre las ventajas e inconvenientes de cada una. Y en este punto es en dónde tiene más dudas. No ha encontrado una solución perfecta que le permita aprovechar de forma sencilla todas las posibilidades de los servicios web. De las opciones que existen en la actualidad, ha decidido probar con PHP5 SOAP.

Se reúne con Carlos y le pone al día sobre todo lo que ha ido averiguando en los últimos días. Deben probar el funcionamiento de la extensión PHP5 SOAP, antes de utilizarla en el nuevo proyecto. Para ello, diseñan dos servicios web sencillos, y deciden implementarlos entre ambos. Cada uno se encargará de programar un servidor y un cliente, y al final de la prueba compartirán la experiencia adquirida para tomar una decisión definitiva.

Como ya comentamos, de las posibilidades que tenemos para utilizar SOAP en PHP vamos a aprender a utilizar la extensión que viene incluida con el lenguaje a partir de su versión 5: PHP5 SOAP.

soap

Soap Client	enabled	
Soap Server	enabled	
Directive	Local Value	Master Value
soap.wsdl_cache	1	1
soap.wsdl_cache_dir	/tmp	/tmp
soap.wsdl_cache_enabled	1	1
soap.wsdl_cache_limit	5	5
soap.wsdl_cache_ttl	86400	86400

Gracias a PHP5 SOAP, puedes utilizar y crear de forma sencilla servicios web en tus aplicaciones. En el momento de escribir este texto, es compatible con las versiones SOAP 1.1 y SOAP 1.2, así como con WSDL 1.1, aunque no permite la generación automática del documento WSDL a partir del servicio web programado.

Para poder usar la extensión, deberás comprobar si ya se encuentra disponible (por ejemplo, consultando la salida obtenida por la función `phpinfo()`):

Normalmente la extensión SOAP formará parte de PHP5 (se habrá compilado con el ejecutable) y podrás utilizarla directamente. Si no fuera así, deberás instalarla y habilitarla en el fichero `php.ini`.

Las dos clases principales que deberás utilizar en tus aplicaciones son `SoapClient` y `SoapServer`. La primera te permitirá comunicarte con un servicio web, y con la segunda podrás crear tus propios servicios.

2.1.- Utilización de un servicio web.

Vamos a comenzar viendo cómo crear en PHP una aplicación que se comunique con un servicio web para obtener información. Por ejemplo, imagínate que has finalizado la aplicación de tienda web y lleva un tiempo funcionando. Un día la empresa necesita comenzar a vender en el extranjero, y quiere dar la posibilidad de mostrar los precios de los productos en dólares.

Lo primero que necesitas es conocer la tasa de conversión entre euros y dólares. Y para tener esa información lo más actualizada posible, decides buscar un servicio web que te la ofrezca en tiempo real. Por ejemplo, el disponible en WEbserviceX.NET.

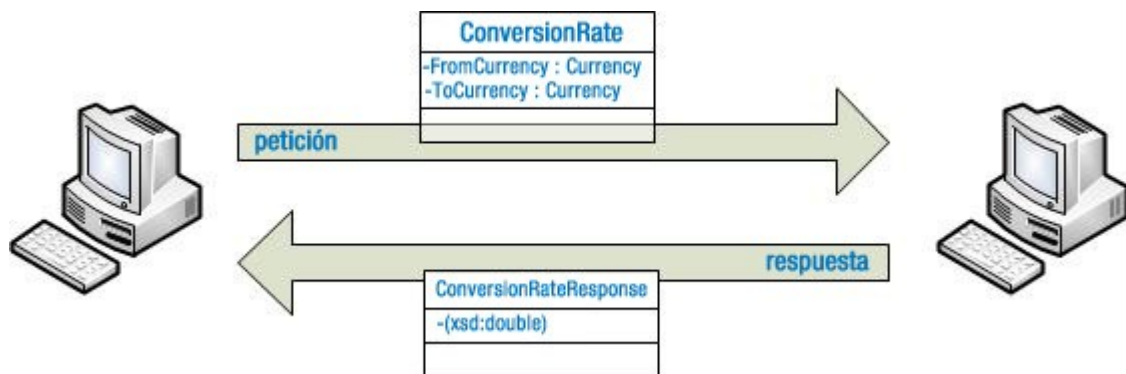
<http://www.webserviceX.net/ws/WSDetails.aspx?CATID=2&WSID=10>

Para crear un cliente del servicio, deberás conocer los detalles del mismo (como mínimo, los parámetros de entrada y salida que debes usar, y cuál es la URL del servicio) y emplear en tu código la clase `SoapClient`. Para averiguar los detalles del servicio, puedes consultar el documento WSDL del servicio, disponible en la dirección <http://www.webserviceX.net/CurrencyConvertor.asmx?WSDL>.

<http://www.webserviceX.net/CurrencyConvertor.asmx?WSDL>

En el documento WSDL obtenido, puedes observar que:

Anexo I - CurrencyConvertor.wSDL



El alias del espacio de nombres correspondiente al XML Schema que utiliza el documento es `s`.

```
<wsdl:definitions
  ...
  xmlns:s="http://www.w3.org/2001/XMLSchema"
  ...
>
```

El tipo `Currency` debe ser un `string` de tres caracteres de los que se listan en el documento, correspondiente a las siglas de una divisa.

```
<s:simpleType name="Currency">
  <s:restriction base="s:string">
    <s:enumeration value="AFA" />
    ...
  </s:restriction>
</s:simpleType>
```

El tipo `ConversionRate` es una secuencia de dos elementos `Currency`.

```
<s:element name="ConversionRate">
<s:complexType>
  <s:sequence>
    <s:element
      minOccurs="1" maxOccurs="1"
      name="FromCurrency" type="tns:Currency"
    />
    <s:element
      minOccurs="1" maxOccurs="1"
      name="ToCurrency" type="tns:Currency"
    />
  </s:sequence>
</s:complexType>
</s:element>
```

El tipo `ConversionRateResponse` es un `double`.

```
<s:element name="ConversionRateResponse">
<s:complexType>
  <s:sequence>
    <s:element
      minOccurs="1" maxOccurs="1"
      name="ConversionRateResult" type="s:double"
    />
  </s:sequence>
</s:complexType>
</s:element>
```

El principal inconveniente de la extensión PHP5 SOAP es que:



No ofrece un interface de programación orientado a objetos.



Una vez que has programado un servicio web, no permite generar de forma automática el documento WSDL.

Efectivamente, aunque existen librerías disponibles en varios lenguajes de programación que permiten la generación automática del documento WSDL a partir de un servicio web, no es el caso de PHP5 SOAP.

2.2.- Utilización de un servicio web (II).

El estilo de enlazado es `document/literal` (recuerda que nosotros vimos el `RPC/encoded` solamente), por lo que los elementos de tipo `message` tienen un formato distinto. Sin embargo, en base a su contenido (fíjate en los elementos que terminan en `soap`) se puede deducir también que:

- ✓ El nombre de la función a la que debes llamar es `ConversionRate`.

```
<wsdl:operation name="ConversionRate">
```

- ✓ Como parámetro de entrada le tienes que pasar un elemento de tipo `ConversionRate` (dos `string`), y devolverá un elemento `ConversionRateResponse` (un `double`).

```
<wsdl:message name="ConversionRateSoapIn">
  <wsdl:part name="parameters" element="tns:ConversionRate" />
</wsdl:message>
<wsdl:message name="ConversionRateSoapOut">
  <wsdl:part name="parameters" element="tns:ConversionRateResponse" />
</wsdl:message>
...
<wsdl:portType name="CurrencyConvertorSoap">
  <wsdl:operation name="ConversionRate">
    <wsdl:input message="tns:ConversionRateSoapIn" />
    <wsdl:output message="tns:ConversionRateSoapOut" />
  </wsdl:operation>
</wsdl:portType>
```

- ✓ La URL para acceder al servicio es <http://www.webservicex.net/CurrencyConvertor.asmx>.

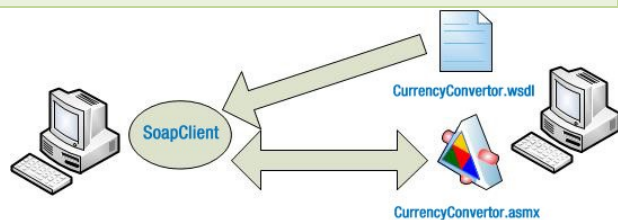
```
<wsdl:service name="CurrencyConvertor">
  <wsdl:port name="CurrencyConvertorSoap" binding="tns:CurrencyConvertorSoap">
    <soap:address location="http://www.webservicex.net/CurrencyConvertor.asmx" />
  </wsdl:port>
</wsdl:service>
```

Con la información anterior, para utilizar el servicio desde PHP creas un nuevo objeto de la clase `SoapClient`. Como el servicio tiene un documento WSDL asociado, en el constructor le indicas dónde se encuentra:

```
$cliente = new SoapClient(
    "http://www.webservicex.net/CurrencyConvertor.asmx?WSDL"
);
```

Y para realizar la llamada a la función `ConversionRate`, incluyes los parámetros en un array:

```
$parametros = array("FromCurrency" => "EUR",
    "ToCurrency" => "USD");
$tasa = $cliente->ConversionRate($parametros);
```



La llamada devuelve un objeto de una clase predefinida en PHP llamada `stdClass`. Para utilizar el valor devuelto, puedes hacer:

```
print("Resultado: ".$tasa->ConversionRateResult);
```

2.3.- Utilización de un servicio web (III).

El constructor de la clase `SoapClient` puede usarse de dos formas: indicando un documento WSDL, como en el caso anterior, o sin indicarlo. En el primer caso, la extensión SOAP examina la definición del servicio y establece las opciones adecuadas para la comunicación, con lo cual el código necesario para utilizar un servicio es bastante simple.

En el segundo caso, si no indicas en el constructor un documento WSDL (bien porque no existe, o porque necesitas configurar manualmente alguna opción), el primer parámetro debe ser `null`, y las opciones para comunicarse con el servicio las tendrás que establecer en un array que se pasa como segundo parámetro.

Si el servicio dispone del correspondiente documento WSDL, en muchos lenguajes de programación existen utilidades que facilitan aún más el desarrollo de aplicaciones que lo utilicen. Nosotros vamos a ver la herramienta **wsdl2php**.

<http://www.urдалen.no/wsdl2php/index.php>



Se trata de un guión escrito en lenguaje PHP que examina un documento WSDL y genera un fichero PHP específico para comunicarse con el servicio web correspondiente. Su uso es muy sencillo: se ejecuta pasándole como parámetro la URL en que se encuentra el documento WSDL, y como resultado genera un fichero con código PHP.

Anexo II - Fichero con código PHP

```
smr@ubuntu-profe: ~/Descargas
Archivo Editar Ver Terminal Ayuda
smr@ubuntu-profe:~/Descargas$ sudo pear install wsdl2php-0.2.1-pear.tgz
PHP Deprecated: Comments starting with '#' are deprecated in /etc/php5/cli/conf
.d/mcrypt.ini on line 1 in Unknown on line 0
install ok: channel://pear.php.net/wsdl2php-0.2.1
smr@ubuntu-profe:~/Descargas$ wsdl2php http://www.webservicex.net/CurrencyConver
tor.asmx?WSDL
PHP Deprecated: Comments starting with '#' are deprecated in /etc/php5/cli/conf
.d/mcrypt.ini on line 1 in Unknown on line 0
Analyzing WSDL.....done
Generating code...done
Writing CurrencyConvertor.php...done
smr@ubuntu-profe:~/Descargas$
```

En el ejemplo anterior, para utilizar el servicio con las clases generadas automáticamente, debes hacer:

```
// Necesitas utilizar el fichero generado por wsdl2php
require_once('CurrencyConvertor.php');

// Creas los parámetros (una instancia de la clase ConversionRate)
$p = new ConversionRate();
$p->FromCurrency = "EUR";
$p->ToCurrency = "USD";

// Creas una instancia de la clase CurrencyConvertor
$ccliente = new CurrencyConvertor();
// Y llamas al método ConversionRate
$r = $ccliente->ConversionRate($p);
```

En esta ocasión, el objeto obtenido al utilizar el servicio web es de la clase **ConversionRateResponse**. Podemos ver paso a paso el procedimiento anterior en el siguiente vídeo.

http://www.youtube.com/watch?feature=player_embedded&v=Lf3kjc_xErc

Ejercicio resuelto

Inspecciona el servicio disponible en la URL <http://www.webservicex.com/globalweather.asmx>, que ofrece información meteorológica sobre distintas ciudades de todo el mundo. A partir de su documento WSDL, utiliza la herramienta **wsdl2php** y, partiendo de las clases que ésta genera, crea el código PHP necesario para mostrar las ciudades españolas de las que ofrece información, y la predicción meteorológica para la ciudad de Santiago de Compostela.

<http://www.webservicex.com/globalweather.asmx>

Anexo III - globalweather.wsdl

RESPUESTA:

Tras ejecutar **wsdl2php** pasándole como parámetro la URL <http://www.webservicex.com/globalweather.asmx?wsdl>, correspondiente al documento WSDL del servicio web, obtendrás un fichero **GlobalWeather.php** con las clases

correspondientes al servicio. Para ejecutar la consulta al servicio y mostrar la información que se pide, puedes ejecutar un código como el que se propone en la solución al ejercicio.

```
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 6 : Servicios web -->
<!-- Ejercicio: Programación de un cliente para el servicio GlobalWeather -->
<?php
    require_once('GlobalWeather.php');

    //Creamos un cliente para llamar a esa URL.
    $tiempo = new GlobalWeather();
    //Llamamos a la operación suma (tenemos que saber nosotros que existe)
    $pais = new GetCitiesByCountry();
    $pais->CountryName="SPAIN";
    $ciudades = $tiempo->GetCitiesByCountry($pais);
    print_r($ciudades);

    $ciudad = new GetWeather();
    $ciudad->CountryName="SPAIN";
    $ciudad->CityName = "Santiago / Labacolla";

    $tiempociudad = $tiempo->GetWeather($ciudad);
    print_r($tiempociudad);

?>
```

Al utilizar la clase SoapClient para comunicarte con un servicio web:



La herramienta **wsdl2php** no es necesaria para obtener las opciones de configuración del mismo a partir del documento WSDL.



Si el servicio dispone de una descripción en formato WSDL, puedes utilizar la herramienta **wsdl2php** para no especificar a mano las opciones del mismo en la llamada al constructor de la clase **SoapClient**.

*Correcta. El objetivo de la herramienta **wsdl2php** no es simplificar la configuración del cliente, sino obtener una estructura de clases que facilite la utilización del servicio.*

2.4.- Utilización de un servicio web (IV).

Si estás usando un documento WSDL para acceder al servicio web, la clase **SoapClient** implementa dos métodos que muestran parte de la información que contiene; concretamente, los tipos de datos definidos por el servicio, y las funciones que ofrece. Para conocer esta información, una vez creado el objeto, debes utilizar los métodos **__getTypes** y **__getFunctions** respectivamente.

```
$cliente = new CurrencyConvertor();
print_r($cliente->__getTypes());
print_r($cliente->__getFunctions());
```

El resultado obtenido es:

```
Array (
    [0] => struct ConversionRate {
        Currency FromCurrency;
        Currency ToCurrency;
    }
    [1] => string Currency
    [2] => struct ConversionRateResponse {
        double ConversionRateResult;
    }
)
Array (
    [0] => ConversionRateResponse ConversionRate
        (ConversionRate $parameters)
    [1] => ConversionRateResponse ConversionRate
        (ConversionRate $parameters)
)
```

SoapClient
+__construct() : SoapClient +__doRequest() : string +__getFunctions() : array +getLastRequest() : string +__getLastRequestHeaders() : string +__getLastResponse() : string +__getLastResponseHeaders() : string +__getTypes() : array +__setCookie() : void +__setLocation() : string +__setSoapHeaders() : bool +__soapCall() : mixed

Donde la función **ConversionRate** aparece duplicada, dado que el servicio web ofrece dos versiones de la misma: una para SOAP 1.1 y otra para SOAP 1.2.

La extensión PHP5 SOAP también incluye opciones de depuración muy útiles para averiguar qué está pasando cuando la conexión al servicio web no funciona como debería. Para habilitarlas, cuando hagas la llamada al constructor de la clase `SoapClient`, debes utilizar la opción `trace` en el array de opciones del segundo parámetro.

```
$cliente = new SoapClient(
    "http://www.webservicex.net/CurrencyConvertor.asmx?WSDL",
    array('trace'=>true)
);
```

Existen bastantes opciones que se pueden utilizar con el constructor `SoapClient`, pero si el servicio web dispone de un documento WSDL, normalmente no necesitarás utilizar ninguna. En caso contrario deberás definir al menos las opciones `location` (la URL en la que se encuentra el servicio) y `uri` (su espacio de nombres).

<http://www.php.net/manual/es/soapclient.soapclient.php>

Una vez activada la depuración, podrás utilizar los siguientes métodos para revisar los últimos mensajes SOAP enviados y recibidos.

Método	Significado
<code>getLastRequest</code>	Devuelve el XML correspondiente a la última petición enviada.
<code>getLastRequestHeaders</code>	Devuelve el XML correspondiente a los encabezados de la última petición enviada.
<code>getLastResponse</code>	Devuelve el XML correspondiente a la última respuesta recibida.
<code>getLastResponseHeaders</code>	Devuelve el XML correspondiente a los encabezados de la última respuesta recibida.

2.5.- Creación de un servicio web.

En PHP5 SOAP, para crear un servicio web, debes utilizar la clase `SoapServer`. Veamos un ejemplo sencillo:

```
function suma($a,$b){ return $a+$b; }
function resta($a,$b){ return $a-$b; }

$url="http://localhost/dwes/ut6";
$server = new SoapServer(null,array('uri'=>$url));
$server->addFunction("suma");
$server->addFunction("resta");
$server->handle();
```

SoapServer
<pre>+__construct() : SoapServer +__construct() : void +addFunction() : void +addSoapHeader() : void +fault() : void +getFunctions() : array +handle() : array +handle() : void +setClass() : void +setObject() : void +setPersistence() : void</pre>

El código anterior crea un servicio web con dos funciones: `suma` y `resta`. Cada función recibe dos parámetros y devuelve un valor. Para consumir este servicio, necesitas escribir el siguiente código:

```
$url="http://localhost/dwes/ut6/servicio.php";
$uri="http://localhost/dwes/ut6";
$cliente = new SoapClient(null,array('location'=>$url,'uri'=>$uri));

$suma = $cliente->suma(2,3);
$resta = $cliente->resta(2,3);
print("La suma es ".$suma);
print("<br />La resta es ".$resta);
```

El servicio que has creado no incluye un documento WSDL para describir sus funciones. Sabes que existen los métodos `suma` y `resta`, y los parámetros que debes utilizar con ellos, porque conoces el código interno del servicio. Un usuario que no tuviera esta información, no sabría cómo consumir el servicio.

Al igual que sucedía con `SoapClient` al programar un cliente, cuando utilizas `SoapServer` puedes crear un servicio sin documento WSDL asociado (como en el caso anterior), o indicar el documento WSDL correspondiente al servicio; pero antes deberás haberlo creado.

El primer parámetro del constructor indica la ubicación del WSDL correspondiente. El segundo parámetro es una colección de opciones de configuración del servicio. Si existe el primer parámetro, ya no hace falta más información. PHP5 SOAP utiliza la información del documento WSDL para ejecutar el servicio. Si, como en el ejemplo, no existe WSDL, deberás indicar en el segundo parámetro al menos la opción **uri**, con el espacio de nombres destino del servicio.

El constructor `SoapServer` permite indicar, además de **uri**, otras opciones en el segundo parámetro. Por ejemplo, la opción `soap_version` indica si se va a usar SOAP 1.1 o SOAP 1.2.
<http://www.php.net/manual/es/soapserver.soapserver.php>

Además, en el código anterior utilizamos los métodos `addFunction` y `handle`. El primero se encarga de publicar en el servicio la función que se le pase como parámetro. El método `handle` es el encargado de procesar las peticiones, recogiendo los datos que se reciban utilizando POST por HTTP.

Al utilizar la clase SoapServer para crear un servicio web:



Si no creas y le asocias un documento WSDL, deberás indicar las opciones del mismo en la llamada al constructor `SoapServer`.

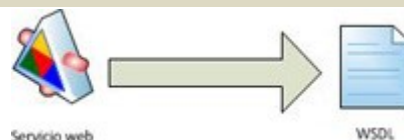


Debes indicar la ubicación del documento WSDL de descripción del servicio.

Si no se especifica como primer parámetro un documento WSDL, el segundo parámetro que recibe el constructor debe contener las opciones de configuración del servicio.

2.6.- Creación de un servicio web (II).

Para crear un documento WSDL de descripción del servicio, tendrás que seguir los pasos vistos anteriormente.



Al programar un servicio web, es importante cambiar en el fichero `php.ini` la directiva `soap.wsdl_cache_enabled` a `0`. En caso contrario, con su valor por defecto (`1`) los cambios que realices en los ficheros WSDL no tendrán efecto de forma inmediata.

El elemento raíz del documento será:

```
<definitions
  name="WSDLCalcula"
  targetNamespace="http://localhost/dwes/ut6"
  xmlns:tns="http://localhost/dwes/ut6"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
>
```

En este caso no necesitas definir ningún tipo nuevo, por lo que no tendrás sección `types`. Los elementos `message` necesarios para la suma (los de la resta son similares) serán:

```
<message name="sumaRequest">
  <part name="a" type="xsd:float"/>
  <part name="b" type="xsd:float"/>
</message>
<message name="sumaResponse">
  <part name="resultado" type="xsd:float"/>
</message>
```

El `portType` (no se incluye el `operation` de la resta, que es equivalente):

```
<portType name="CalculaPortType">
  <operation name="suma">
    <input message="tns:sumaRequest"/>
    <output message="tns:sumaResponse"/>
  </operation>
```

```
</portType>
```

Suponiendo que el servicio web está en el fichero `calcula.php`, la parte de la operación `suma` correspondiente al `binding` será:

```
<binding name="CalculaBinding" type="tns:CalculaPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="suma">
    <soap:operation
      soapAction="http://localhost/dwes/ut6/calcula.php?method=suma"
    />
    <input>
      <soap:body
        namespace="http://localhost/dwes/ut6"
        use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      />
    />
    <output>
      <soap:body
        namespace="http://localhost/dwes/ut6"
        use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      />
    />
  />
</binding>
```

Y para finalizar, el elemento `service`:

```
<service name="Calcula">
  <port name="CalculaPort" binding="tns:CalculaBinding">
    <soap:address location="http://localhost/dwes/ut6/calcula.php"/>
  </port>
</service>
```

2.7.- Creación de un servicio web (III).

En vez de utilizar funciones para la lógica interna del servicio web, como la suma y la resta del ejemplo anterior, es aconsejable definir una clase que implemente los métodos que queramos publicar en el servicio.



```
class Calcula {
  public function suma($a, $b){ return $a+$b; }
  public function resta($a, $b){ return $a-$b; }
}
```

Al hacerlo de esta forma, en lugar de añadir una a una las funciones, podemos añadir la clase completa al servidor utilizando el método `setClass` de `SoapServer`.

```
require_once('Calcula.php');

$server = new SoapServer(null, array('uri'=>''));
$server->setClass('Calcula');
$server->handle();
```

En lugar de una clase, también es posible indicar un objeto para procesar las peticiones SOAP utilizando el método `setObject` de la clase `SoapServer`.

Aunque como ya sabes, PHP5 SOAP no genera el documento WSDL de forma automática para los servicios que crees, existen algunos mecanismos que nos permiten generarlo, aunque siempre es aconsejable revisar los resultados obtenidos antes de publicarlos. Una de las formas más sencillas es utilizar la librería `WSDLDocument`.

<https://code.google.com/p/wsdlDocument/>

Esta librería revisa los comentarios que hayas añadido al código de la clase que quieras a publicar (debe ser una clase, no funciones aisladas), y genera como salida el documento WSDL correspondiente.

Para que funcione correctamente, es necesario que los comentarios de las clases sigan un formato específico: el mismo que utiliza la herramienta de documentación `PHPDocumentor`.

`PHPDocumentor` es una herramienta de código libre para generación automática de documentación, similar a Javadoc (para el lenguaje Java). Si comentamos el código de nuestras aplicaciones siguiendo unas normas, `PHPDocumentor` es capaz de generar, a partir de los comentarios que introduzcamos en el código mientras programamos, documentación en diversos formatos (HTML, PDF, XML).

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

Los comentarios se deben ir introduciendo en el código distribuidos en bloques, y utilizando ciertas marcas específicas como `@param` para indicar un parámetro y `@return` para indicar el valor devuelto por una función. Por ejemplo, la clase `Calcula` comentada según estas normas quedaría:

<http://www.vivalared.com/phpdocumentor-como-generar-una-documentacion-desde-php>

```
/**
 * Clase Calcula
 *
 * Desarrollo Web en Entorno Servidor
 * Tema 6: Servicios web
 * Ejemplo: Documentación para generación
 *          automática del documento WSDL
 * @author Víctor Lourido
 */

class Calcula {
    /**
     * Suma dos números y devuelve el resultado
     *
     * @param float $a
     * @param float $b
     * @return float
     */
    public function suma($a, $b){
        return $a+$b;
    }

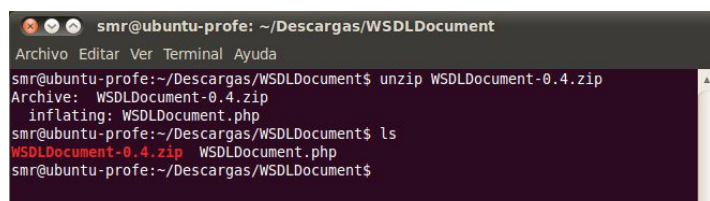
    /**
     * Resta dos números y devuelve el resultado
     *
     * @param float $a
     * @param float $b
     * @return float
     */
    public function resta($a, $b){
        return $a-$b;
    }
}
```

2.8.- Creación de un servicio web (IV).

No es necesario instalar `WSDLDocument`, basta con descargarlo en tu equipo y descomprimir el archivo. Su contenido es un único fichero con código en PHP.

Para generar el documento WSDL a partir de la clase `Calcula` anterior, debes crear un nuevo fichero con el siguiente código:

```
require_once("Calcula.php");
// Ruta a WSDLDocument
require_once("WSDLDocument.php");
```



```
smr@ubuntu-profe: ~/Descargas/WSDLDocument
Archivo Editar Ver Terminal Ayuda
smr@ubuntu-profe:~/Descargas/WSDLDocument$ unzip WSDLDocument-0.4.zip
Archive:  WSDLDocument-0.4.zip
  inflating: WSDLDocument.php
smr@ubuntu-profe:~/Descargas/WSDLDocument$ ls
WSDLDocument-0.4.zip  WSDLDocument.php
smr@ubuntu-profe:~/Descargas/WSDLDocument$
```

```
$wsdl = new WSDLDocument (
    "Calcula",
    "http://localhost/dwes/ut6/servicio.php",
    "http://localhost/dwes/ut6"
);
echo $wsdl->saveXml();
```

Es decir, crear un nuevo objeto de la clase `WSDLDocument`, e indicar como parámetros:

- ✓ El nombre de la clase que gestionará las peticiones al servicio.
- ✓ La URL en que se ofrece el servicio.
- ✓ El espacio de nombres destino.

El método `saveXML` obtiene como salida el documento WSDL de descripción del servicio. Revisalo, pues posiblemente tengas que realizar algunos cambios (por ejemplo, pasar el formato de codificación a UTF-8, o cambiar el nombre de alguna de las clases que contiene y de su constructor respectivo).

Cuando esté listo, publícalo con tu servicio. Para ello, copia el fichero obtenido en una ruta accesible vía web (por ejemplo, en la misma ruta en la que se encuentre la clase que gestiona el servicio), e indica la URL en que se encuentra cuando instancias la clase `SoapServer`.

```
$server = new SoapServer("http://localhost/dwes/ut6/calcula.wsdl");
```

Si añades a la URL del servicio el parámetro POST `wsdl`, verás el fichero de descripción del servicio. En nuestro caso la URL sería `http://localhost/dwes/ut6/calcula.php?wsdl`.

En el siguiente vídeo puedes comprobar todo el proceso de creación de un servicio web, utilización de `WSDLDocument` para obtener su documento WSDL de descripción, y la publicación del mismo.

http://www.youtube.com/watch?feature=player_embedded&v=Er2rhLfNV4Q

Relaciona los términos siguientes con el concepto a que hacen referencia:

Elemento	Relación	Hace referencia a
<code>wsdl2php</code>	4	1. Herramienta para generación de documentación, a partir de un programa PHP adecuadamente documentado.
<code>WSDLDocument</code>	3	2. Método de la clase <code>WSDLDocument</code> que genera el documento WSDL.
<code>PHPDocumentor</code>	1	3. Herramienta para construir un documento WSDL, a partir de un servicio web programado como una clase PHP adecuadamente documentada.
<code>saveXml</code>	2	4. Herramienta para construir clases PHP a partir de un documento WSDL.

Anexo I - CurrencyConvertor.wsdl

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://www.webserviceX.NET/"
  xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  targetNamespace="http://www.webserviceX.NET/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://www.webserviceX.NET/">
      <s:element name="ConversionRate">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="FromCurrency" type="tns:Currency" />
            <s:element minOccurs="1" maxOccurs="1" name="ToCurrency" type="tns:Currency" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:simpleType name="Currency">
        <s:restriction base="s:string">
          <s:enumeration value="AFA" />
          <s:enumeration value="ALL" />
          <s:enumeration value="DZD" />
          <s:enumeration value="ARS" />
          <s:enumeration value="AWG" />
          <s:enumeration value="AUD" />
          <s:enumeration value="BSD" />
          <s:enumeration value="BHD" />
          <s:enumeration value="BDT" />
          <s:enumeration value="BBD" />
          <s:enumeration value="BZD" />
          <s:enumeration value="BMD" />
          <s:enumeration value="BTN" />
          <s:enumeration value="BOB" />
          <s:enumeration value="BWP" />
          <s:enumeration value="BRL" />
          <s:enumeration value="GBP" />
          <s:enumeration value="BND" />
          <s:enumeration value="BIF" />
          <s:enumeration value="XOF" />
          <s:enumeration value="XAF" />
          <s:enumeration value="KHR" />
          <s:enumeration value="CAD" />
          <s:enumeration value="CVE" />
          <s:enumeration value="KYD" />
          <s:enumeration value="CLP" />
          <s:enumeration value="CNY" />
          <s:enumeration value="COP" />
          <s:enumeration value="KMF" />
          <s:enumeration value="CRC" />
          <s:enumeration value="HRK" />
          <s:enumeration value="CUP" />
          <s:enumeration value="CYP" />
          <s:enumeration value="CZK" />
          <s:enumeration value="DKK" />
          <s:enumeration value="DJF" />
          <s:enumeration value="DOP" />
          <s:enumeration value="XCD" />
          <s:enumeration value="EGP" />
          <s:enumeration value="SVC" />
          <s:enumeration value="EEK" />
          <s:enumeration value="ETB" />
          <s:enumeration value="EUR" />
          <s:enumeration value="FKP" />
          <s:enumeration value="GMD" />
          <s:enumeration value="GHC" />
          <s:enumeration value="GIP" />
          <s:enumeration value="XAU" />
          <s:enumeration value="GTQ" />
          <s:enumeration value="GNF" />
          <s:enumeration value="GYD" />
          <s:enumeration value="HTG" />
          <s:enumeration value="HNL" />
        </s:restriction>
      </s:simpleType>
    </s:schema>
  </wsdl:types>

```

```
<s:enumeration value="HKD" />
<s:enumeration value="HUF" />
<s:enumeration value="ISK" />
<s:enumeration value="INR" />
<s:enumeration value="IDR" />
<s:enumeration value="IQD" />
<s:enumeration value="ILS" />
<s:enumeration value="JMD" />
<s:enumeration value="JPY" />
<s:enumeration value="JOD" />
<s:enumeration value="KZT" />
<s:enumeration value="KES" />
<s:enumeration value="KRW" />
<s:enumeration value="KWD" />
<s:enumeration value="LAK" />
<s:enumeration value="LVL" />
<s:enumeration value="LBP" />
<s:enumeration value="LSL" />
<s:enumeration value="LRD" />
<s:enumeration value="LYD" />
<s:enumeration value="LTL" />
<s:enumeration value="MOP" />
<s:enumeration value="MKD" />
<s:enumeration value="MGF" />
<s:enumeration value="MWK" />
<s:enumeration value="MYR" />
<s:enumeration value="MVR" />
<s:enumeration value="MTL" />
<s:enumeration value="MRO" />
<s:enumeration value="MUR" />
<s:enumeration value="MXN" />
<s:enumeration value="MDL" />
<s:enumeration value="MNT" />
<s:enumeration value="MAD" />
<s:enumeration value="MZM" />
<s:enumeration value="MMK" />
<s:enumeration value="NAD" />
<s:enumeration value="NPR" />
<s:enumeration value="ANG" />
<s:enumeration value="NZD" />
<s:enumeration value="NIO" />
<s:enumeration value="NGN" />
<s:enumeration value="KPW" />
<s:enumeration value="NOK" />
<s:enumeration value="OMR" />
<s:enumeration value="XPF" />
<s:enumeration value="PKR" />
<s:enumeration value="XPD" />
<s:enumeration value="PAB" />
<s:enumeration value="PGK" />
<s:enumeration value="PYG" />
<s:enumeration value="PEN" />
<s:enumeration value="PHP" />
<s:enumeration value="XPT" />
<s:enumeration value="PLN" />
<s:enumeration value="QAR" />
<s:enumeration value="ROL" />
<s:enumeration value="RUB" />
<s:enumeration value="WST" />
<s:enumeration value="STD" />
<s:enumeration value="SAR" />
<s:enumeration value="SCR" />
<s:enumeration value="SLL" />
<s:enumeration value="XAG" />
<s:enumeration value="SGD" />
<s:enumeration value="SKK" />
<s:enumeration value="SIT" />
<s:enumeration value="SBD" />
<s:enumeration value="SOS" />
<s:enumeration value="ZAR" />
<s:enumeration value="LKR" />
<s:enumeration value="SHP" />
<s:enumeration value="SDD" />
<s:enumeration value="SRG" />
<s:enumeration value="SZL" />
<s:enumeration value="SEK" />
<s:enumeration value="CHF" />
<s:enumeration value="SYP" />
```

```

        <s:enumeration value="TWD" />
        <s:enumeration value="TZS" />
        <s:enumeration value="THB" />
        <s:enumeration value="TOP" />
        <s:enumeration value="TTD" />
        <s:enumeration value="TND" />
        <s:enumeration value="TRL" />
        <s:enumeration value="USD" />
        <s:enumeration value="AED" />
        <s:enumeration value="UGX" />
        <s:enumeration value="UAH" />
        <s:enumeration value="UYU" />
        <s:enumeration value="VUV" />
        <s:enumeration value="VEB" />
        <s:enumeration value="VND" />
        <s:enumeration value="YER" />
        <s:enumeration value="YUM" />
        <s:enumeration value="ZMK" />
        <s:enumeration value="ZWD" />
        <s:enumeration value="TRY" />
    </s:restriction>
</s:simpleType>
<s:element name="ConversionRateResponse">
    <s:complexType>
        <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="ConversionRateResult" type="s:double" />
        </s:sequence>
    </s:complexType>
</s:element>
<s:element name="double" type="s:double" />
</s:schema>
</wsdl:types>
<wsdl:message name="ConversionRateSoapIn">
    <wsdl:part name="parameters" element="tns:ConversionRate" />
</wsdl:message>
<wsdl:message name="ConversionRateSoapOut">
    <wsdl:part name="parameters" element="tns:ConversionRateResponse" />
</wsdl:message>
<wsdl:message name="ConversionRateHttpGetIn">
    <wsdl:part name="FromCurrency" type="s:string" />
    <wsdl:part name="ToCurrency" type="s:string" />
</wsdl:message>
<wsdl:message name="ConversionRateHttpGetOut">
    <wsdl:part name="Body" element="tns:double" />
</wsdl:message>
<wsdl:message name="ConversionRateHttpPostIn">
    <wsdl:part name="FromCurrency" type="s:string" />
    <wsdl:part name="ToCurrency" type="s:string" />
</wsdl:message>
<wsdl:message name="ConversionRateHttpPostOut">
    <wsdl:part name="Body" element="tns:double" />
</wsdl:message>
<wsdl:portType name="CurrencyConvertorSoap">
    <wsdl:operation name="ConversionRate">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">&lt;br&gt;&lt;b&gt;Get
conversion
rate
from
one
currency
to
another
currency
&lt;br&gt;&lt;p&gt;&lt;b&gt;&lt;font
color='#000080'
size='1'
face='Verdana'&gt;&lt;u&gt;Differenct
currency
Code
and
Names
around
the
world&lt;/u&gt;&lt;/font&gt;&lt;/b&gt;&lt;/p&gt;&lt;blockquote&gt;&lt;p&gt;&lt;font
face='Verdana'
size='1'&gt;AFA-Afghanistan
Afghani&lt;br&gt;ALL-Albanian
Lek&lt;br&gt;DZD-
Algerian
Dinar&lt;br&gt;ARS-Argentine
Peso&lt;br&gt;AWG-Aruba
Florin&lt;br&gt;AUD-Australian
Dollar&lt;br&gt;BSD-Bahamian
Dollar&lt;br&gt;BHD-Bahraini
Dinar&lt;br&gt;BDT-Bangladesh
Taka&lt;br&gt;BBD-Barbados
Dollar&lt;br&gt;BZD-Belize
Dollar&lt;br&gt;BMD-Bermuda
Dollar&lt;br&gt;BTN-Bhutan
Ngultrum&lt;br&gt;BOB-Bolivian
Boliviano&lt;br&gt;BWP-Botswana
Pula&lt;br&gt;BRL-Brazilian
Real&lt;br&gt;GBP-British
Pound&lt;br&gt;BND-Brunei
Dollar&lt;br&gt;BIF-Burundi
Franc&lt;br&gt;XOF-CFA
Franc
(BCEAO)&lt;br&gt;XAF-CFA
Franc
(BEAC)&lt;br&gt;KHR-Cambodia
Riel&lt;br&gt;CAD-Canadian
Dollar&lt;br&gt;CVE-Cape
Verde
Escudo&lt;br&gt;KYD-Cayman
Islands
Dollar&lt;br&gt;CLP-Chilean
Peso&lt;br&gt;CNY-Chinese
Yuan&lt;br&gt;COP-Colombian
Peso&lt;br&gt;KMF-Comoros
Franc&lt;br&gt;CRC-Costa
Rica
Colon&lt;br&gt;HRK-Croatian
Kuna&lt;br&gt;CUP-Cuban
Peso&lt;br&gt;CYP-Cyprus
Pound&lt;br&gt;CZK-Czech
Koruna&lt;br&gt;DKK-Danish
Krone&lt;br&gt;DJF-Dijibouti
Franc&lt;br&gt;DOP-Dominican
Peso&lt;br&gt;XCD-East
Caribbean
Dollar&lt;br&gt;EGP-Egyptian
Pound&lt;br&gt;SVC-El
Salvador
Colon&lt;br&gt;EEK-Estonian
Kroon&lt;br&gt;ETB-Ethiopian
Birr&lt;br&gt;EUR-Euro&lt;br&gt;FKP-Falkland
Islands
Pound&lt;br&gt;GMD-Gambian
Dalasi&lt;br&gt;GHC-Ghanian
Cedi&lt;br&gt;GIP-Gibraltar
Pound&lt;br&gt;XAU-Gold
Ounces&lt;br&gt;GTQ-Guatemala
Quetzal&lt;br&gt;GNF-Guinea
Franc&lt;br&gt;GYD-Guyana

```

- 27 -

```

Rupiah<!-->XPD-Palladium Ounces<!-->PAB-Panama Balboa<!-->PGK-Papua New Guinea
Kina<!-->PYG-Paraguayan Guarani<!-->PEN-Peruvian Nuevo Sol<!-->PHP-Philippine
Peso<!-->XPT-Platinum Ounces<!-->PLN-Polish Zloty<!-->QAR-Qatar
Rial<!-->ROL-Romanian Leu<!-->RUB-Russian Rouble<!-->LKR-Sri Lanka
Tala<!-->STD-Sao Tome Dobra<!-->SAR-Saudi Arabian Riyal<!-->SCR-Seychelles
Rupiah<!-->SLL-Sierra Leone Leone<!-->XAG-Silver Ounces<!-->SGD-Singapore
Dollar<!-->SKK-Slovak Koruna<!-->SIT-Slovenian Tolar<!-->SBD-Solomon Islands
Dollar<!-->SOS-Somali Shilling<!-->ZAR-South African Rand<!-->LKR-Sri Lanka
Rupiah<!-->SHP-St Helena Pound<!-->SDD-Sudanese Dinar<!-->SRG-Surinam
Guilder<!-->SZL-Swaziland Lilageni<!-->SEK-Swedish Krona<!-->TRY-Turkey
Lira<!-->CHF-Swiss Franc<!-->SYP-Syrian Pound<!-->TWD-Taiwan
Dollar<!-->TZS-Tanzanian Shilling<!-->THB-Thai Baht<!-->TOP-Tonga
Pa'anga<!-->TTD-Trinidad& Tobago Dollar<!-->TND-Tunisian
Dinar<!-->TRL-Turkish Lira<!-->USD-U.S. Dollar<!-->AED-UAE Dirham<!-->UGX-
Ugandan Shilling<!-->UAH-Ukraine Hryvnia<!-->UYU-Uruguayan New Peso<!-->VUV-
Vanuatu Vatu<!-->VEB-Venezuelan Bolivar<!-->VND-Vietnam Dong<!-->YER-Yemen
Riyal<!-->YUM-Yugoslav Dinar<!-->ZMK-Zambian Kwacha<!-->ZWD-Zimbabwe
</font></p></blockquote></wsdl:documentation>
<wsdl:input message="tns:ConversionRateHttpGetIn" />
<wsdl:output message="tns:ConversionRateHttpGetOut" />
</wsdl:operation>
</wsdl:portType>
<wsdl:portType name="CurrencyConvertorHttpGetPost">
<wsdl:operation name="ConversionRate">
<wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"><!-->Get
conversion rate from one currency to another currency
<!--><!-->font color='#000080' size='1'
face='Verdana'<!--><!-->Different currency Code and Names around the
world<!--><!-->font<!--><!-->font<!--><!-->font
face='Verdana' size='1'<!-->AFA-Afghanistan Afghani<!-->ALL-Albanian Lek<!-->DZD-
Algerian Dinar<!-->ARS-Argentine Peso<!-->AWG-Aruba Florin<!-->AUD-Australian
Dollar<!-->BSD-Bahamian Dollar<!-->BHD-Bahraini Dinar<!-->BDT-Bangladesh
Taka<!-->BBD-Barbados Dollar<!-->BZD-Belize Dollar<!-->BMD-Bermuda
Dollar<!-->BTN-Bhutan Ngultrum<!-->BOB-Bolivian Boliviano<!-->BWP-Botswana
Pula<!-->BRL-Brazilian Real<!-->GBP-British Pound<!-->BND-Brunei
Dollar<!-->BIF-Burundi Franc<!-->XOF-CFA Franc (BCEAO)<!-->XAF-CFA Franc
(BEAC)<!-->KHR-Cambodia Riel<!-->CAD-Canadian Dollar<!-->CVE-Cape Verde
Escudo<!-->KYD-Cayman Islands Dollar<!-->CLP-Chilean Peso<!-->CNY-Chinese
Yuan<!-->COP-Colombian Peso<!-->KMF-Comoros Franc<!-->CRC-Costa Rica
Colon<!-->HRK-Croatian Kuna<!-->CUP-Cuban Peso<!-->CYP-Cyprus
Pound<!-->CZK-Czech Koruna<!-->DKK-Danish Krone<!-->DJF-Djibouti
Franc<!-->DOP-Dominican Peso<!-->XCD-East Caribbean Dollar<!-->EGP-Egyptian
Pound<!-->SVC-El Salvador Colon<!-->EEK-Estonian Kroon<!-->ETB-Ethiopian
Birr<!-->EUR-Euro<!-->FKP-Falkland Islands Pound<!-->GMD-Gambian
Dalasi<!-->GHC-Ghanian Cedi<!-->GIP-Gibraltar Pound<!-->XAU-Gold
Ounces<!-->GTQ-Guatemala Quetzal<!-->GNF-Guinea Franc<!-->GYD-Guyana
Dollar<!-->HTG-Haiti Gourde<!-->HNL-Honduras Lempira<!-->HKD-Hong Kong
Dollar<!-->HUF-Hungarian Forint<!-->ISK-Iceland Krona<!-->INR-Indian
Rupiah<!-->IDR-Indonesian Rupiah<!-->IQD-Iraqi Dinar<!-->ILS-Israeli
Shekel<!-->JMD-Jamaican Dollar<!-->JPY-Japanese Yen<!-->JOD-Jordanian
Dinar<!-->KZT-Kazakhstan Tenge<!-->KES-Kenyan Shilling<!-->KRW-Korean
Won<!-->KWD-Kuwaiti Dinar<!-->LAK-Lao Kip<!-->LVL-Latvian Lat<!-->LBP-
Lebanese Pound<!-->LSL-Lesotho Loti<!-->LRD-Liberian Dollar<!-->LYD-Libyan
Dinar<!-->LTL-Lithuanian Lita<!-->MOP-Macau Pataca<!-->MKD-Macedonian
Denar<!-->MGF-Malagasy Franc<!-->MWK-Malawi Kwacha<!-->MYR-Malaysian
Ringgit<!-->MVR-Maldives Rufiyaa<!-->MTL-Maltese Lira<!-->MRO-Mauritania
Ougulya<!-->MUR-Mauritius Rupee<!-->MXN-Mexican Peso<!-->MDL-Moldovan
Leu<!-->MNT-Mongolian Tugrik<!-->MAD-Moroccan Dirham<!-->MZM-Mozambique
Metical<!-->MMK-Myanmar Kyat<!-->NAD-Namibian Dollar<!-->NPR-Nepalese
Rupiah<!-->ANG-Neth Antilles Guilder<!-->NZD-New Zealand Dollar<!-->NIO-
Nicaragua Cordoba<!-->NGN-Nigerian Naira<!-->KPW-North Korean Won<!-->NOK-
Norwegian Krone<!-->OMR-Omani Rial<!-->XPF-Pacific Franc<!-->PKR-Pakistani
Rupiah<!-->XPD-Palladium Ounces<!-->PAB-Panama Balboa<!-->PGK-Papua New Guinea
Kina<!-->PYG-Paraguayan Guarani<!-->PEN-Peruvian Nuevo Sol<!-->PHP-Philippine
Peso<!-->XPT-Platinum Ounces<!-->PLN-Polish Zloty<!-->QAR-Qatar
Rial<!-->ROL-Romanian Leu<!-->RUB-Russian Rouble<!-->LKR-Sri Lanka
Tala<!-->STD-Sao Tome Dobra<!-->SAR-Saudi Arabian Riyal<!-->SCR-Seychelles
Rupiah<!-->SLL-Sierra Leone Leone<!-->XAG-Silver Ounces<!-->SGD-Singapore
Dollar<!-->SKK-Slovak Koruna<!-->SIT-Slovenian Tolar<!-->SBD-Solomon Islands
Dollar<!-->SOS-Somali Shilling<!-->ZAR-South African Rand<!-->LKR-Sri Lanka
Rupiah<!-->SHP-St Helena Pound<!-->SDD-Sudanese Dinar<!-->SRG-Surinam
Guilder<!-->SZL-Swaziland Lilageni<!-->SEK-Swedish Krona<!-->TRY-Turkey
Lira<!-->CHF-Swiss Franc<!-->SYP-Syrian Pound<!-->TWD-Taiwan
Dollar<!-->TZS-Tanzanian Shilling<!-->THB-Thai Baht<!-->TOP-Tonga
Pa'anga<!-->TTD-Trinidad& Tobago Dollar<!-->TND-Tunisian
Dinar<!-->TRL-Turkish Lira<!-->USD-U.S. Dollar<!-->AED-UAE Dirham<!-->UGX-
Ugandan Shilling<!-->UAH-Ukraine Hryvnia<!-->UYU-Uruguayan New Peso<!-->VUV-
Vanuatu Vatu<!-->VEB-Venezuelan Bolivar<!-->VND-Vietnam Dong<!-->YER-Yemen
-->

```

```

Riyal<br>YUM-Yugoslav      Dinar<br>ZMK-Zambian      Kwacha<br>ZWD-Zimbabwe
Dollar<br>
<wsdl:documentation>
  <wsdl:input message="tns:ConversionRateHttpPostIn" />
  <wsdl:output message="tns:ConversionRateHttpPostOut" />
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="CurrencyConvertorSoap" type="tns:CurrencyConvertorSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="ConversionRate">
    <soap:operation soapAction="http://www.webserviceX.NET/ConversionRate" style="document"
  />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="CurrencyConvertorSoap12" type="tns:CurrencyConvertorSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="ConversionRate">
    <soap12:operation
      soapAction="http://www.webserviceX.NET/ConversionRate"
      style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="CurrencyConvertorHttpGet" type="tns:CurrencyConvertorHttpGet">
  <http:binding verb="GET" />
  <wsdl:operation name="ConversionRate">
    <http:operation location="/ConversionRate" />
    <wsdl:input>
      <http:urlEncoded />
    </wsdl:input>
    <wsdl:output>
      <mime:mimeXml part="Body" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="CurrencyConvertorHttpPost" type="tns:CurrencyConvertorHttpPost">
  <http:binding verb="POST" />
  <wsdl:operation name="ConversionRate">
    <http:operation location="/ConversionRate" />
    <wsdl:input>
      <mime:content type="application/x-www-form-urlencoded" />
    </wsdl:input>
    <wsdl:output>
      <mime:mimeXml part="Body" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service name="CurrencyConvertor">
  <wsdl:port name="CurrencyConvertorSoap" binding="tns:CurrencyConvertorSoap">
    <soap:address location="http://www.webservices.net/CurrencyConvertor.asmx" />
  </wsdl:port>
  <wsdl:port name="CurrencyConvertorSoap12" binding="tns:CurrencyConvertorSoap12">
    <soap12:address location="http://www.webservices.net/CurrencyConvertor.asmx" />
  </wsdl:port>
  <wsdl:port name="CurrencyConvertorHttpGet" binding="tns:CurrencyConvertorHttpGet">
    <http:address location="http://www.webservices.net/CurrencyConvertor.asmx" />
  </wsdl:port>
  <wsdl:port name="CurrencyConvertorHttpPost" binding="tns:CurrencyConvertorHttpPost">
    <http:address location="http://www.webservices.net/CurrencyConvertor.asmx" />
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

Anexo II - Fichero con código PHP

```
<?php
class ConversionRate {
    public $FromCurrency; // Currency
    public $ToCurrency; // Currency
}

class Currency {
    const AFA = 'AFA';
    const ALL = 'ALL';
    const DZD = 'DZD';
    const ARS = 'ARS';
    const AWG = 'AWG';
    const AUD = 'AUD';
    const BSD = 'BSD';
    const BHD = 'BHD';
    const BDT = 'BDT';
    const BBD = 'BBB';
    const BZD = 'BZD';
    const BMD = 'BMD';
    const BTN = 'BTN';
    const BOB = 'BOB';
    const BWP = 'BWP';
    const BRL = 'BRL';
    const GBP = 'GBP';
    const BND = 'BND';
    const BIF = 'BIF';
    const XOF = 'XOF';
    const XAF = 'XAF';
    const KHR = 'KHR';
    const CAD = 'CAD';
    const CVE = 'CVE';
    const KYD = 'KYD';
    const CLP = 'CLP';
    const CNY = 'CNY';
    const COP = 'COP';
    const KMF = 'KMF';
    const CRC = 'CRC';
    const HRK = 'HRK';
    const CUP = 'CUP';
    const CYP = 'CYP';
    const CZK = 'CZK';
    const DKK = 'DKK';
    const DJF = 'DJF';
    const DOP = 'DOP';
    const XCD = 'XCD';
    const EGP = 'EGP';
    const SVC = 'SVC';
    const EEK = 'EEK';
    const ETB = 'ETB';
    const EUR = 'EUR';
    const FKP = 'FKP';
    const GMD = 'GMD';
    const GHC = 'GHC';
    const GIP = 'GIP';
    const XAU = 'XAU';
    const GTQ = 'GTQ';
    const GNF = 'GNF';
    const GYD = 'GYD';
    const HTG = 'HTG';
    const HNL = 'HNL';
    const HKD = 'HKD';
    const HUF = 'HUF';
    const ISK = 'ISK';
    const INR = 'INR';
    const IDR = 'IDR';
    const IQD = 'IQD';
    const ILS = 'ILS';
    const JMD = 'JMD';
    const JPY = 'JPY';
    const JOD = 'JOD';
    const KZT = 'KZT';
    const KES = 'KES';
    const KRW = 'KRW';
    const KWD = 'KWD';
```

```
const LAK = 'LAK';
const LVL = 'LVL';
const LBP = 'LBP';
const LSL = 'LSL';
const LRD = 'LRD';
const LYD = 'LYD';
const LTL = 'LTL';
const MOP = 'MOP';
const MKD = 'MKD';
const MGF = 'MGF';
const MWK = 'MWK';
const MYR = 'MYR';
const MVR = 'MVR';
const MTL = 'MTL';
const MRO = 'MRO';
const MUR = 'MUR';
const MXN = 'MXN';
const MDL = 'MDL';
const MNT = 'MNT';
const MAD = 'MAD';
const MZM = 'MZM';
const MMK = 'MMK';
const NAD = 'NAD';
const NPR = 'NPR';
const ANG = 'ANG';
const NZD = 'NZD';
const NIO = 'NIO';
const NGN = 'NGN';
const KPW = 'KPW';
const NOK = 'NOK';
const OMR = 'OMR';
const XPF = 'XPF';
const PKR = 'PKR';
const XPD = 'XPD';
const PAB = 'PAB';
const PGK = 'PGK';
const PYG = 'PYG';
const PEN = 'PEN';
const PHP = 'PHP';
const XPT = 'XPT';
const PLN = 'PLN';
const QAR = 'QAR';
const ROL = 'ROL';
const RUB = 'RUB';
const WST = 'WST';
const STD = 'STD';
const SAR = 'SAR';
const SCR = 'SCR';
const SLL = 'SLL';
const XAG = 'XAG';
const SGD = 'SGD';
const SKK = 'SKK';
const SIT = 'SIT';
const SBD = 'SBD';
const SOS = 'SOS';
const ZAR = 'ZAR';
const LKR = 'LKR';
const SHP = 'SHP';
const SDD = 'SDD';
const SRG = 'SRG';
const SZL = 'SZL';
const SEK = 'SEK';
const CHF = 'CHF';
const SYP = 'SYP';
const TWD = 'TWD';
const TZS = 'TZS';
const THB = 'THB';
const TOP = 'TOP';
const TTD = 'TTD';
const TND = 'TND';
const TRL = 'TRL';
const USD = 'USD';
const AED = 'AED';
const UGX = 'UGX';
const UAH = 'UAH';
const UYU = 'UYU';
const VUV = 'VUV';
const VEB = 'VEB';
```

```

const VND = 'VND';
const YER = 'YER';
const YUM = 'YUM';
const ZMK = 'ZMK';
const ZWD = 'ZWD';
const _TRY = 'TRY';
}

class ConversionRateResponse {
    public $ConversionRateResult; // double
}

/**
 * CurrencyConvertor class
 *
 *
 *
 * @author      {author}
 * @copyright   {copyright}
 * @package     {package}
 */
class CurrencyConvertor extends SoapClient {

    private static $classmap = array(
        'ConversionRate' => 'ConversionRate',
        'Currency' => 'Currency',
        'ConversionRateResponse' => 'ConversionRateResponse',
    );

    public function CurrencyConvertor($wsdl = "http://www.webservices.net/CurrencyConvertor.asmx?WSDL", $options = array()) {
        foreach(self::$classmap as $key => $value) {
            if(!isset($options['classmap'][$key])) {
                $options['classmap'][$key] = $value;
            }
        }
        parent::__construct($wsdl, $options);
    }

    /**
     * <br><b>Get conversion rate from one currency to another currency <b><br><p><b><font
     * color='#000080'
     * size='1' face='Verdana'><u>Different currency Code and Names around the
     * world</u></font></b></p><blockquote><p><font
     * face='Verdana' size='1'>AFA-Afghanistan Afghani<br>ALL-Albanian Lek<br>DZD-Algerian
     * Dinar<br>ARS-Argentine
     * Peso<br>AWG-Aruba Florin<br>AUD-Australian Dollar<br>BSD-Bahamian Dollar<br>BHD-Bahraini
     * Dinar<br>BDT-Bangladesh Taka<br>BBD-Barbados Dollar<br>BZD-Belize Dollar<br>BMD-Bermuda
     * Dollar<br>BTN-Bhutan Ngultrum<br>BOB-Bolivian Boliviano<br>BWP-Botswana Pula<br>BRL-
     * Brazilian
     * Real<br>GBP-British Pound<br>BND-Brunei Dollar<br>BIF-Burundi Franc<br>XOF-CFA Franc
     * (BCEAO)<br>XAF-CFA
     * Franc (BEAC)<br>KHR-Cambodia Riel<br>CAD-Canadian Dollar<br>CVE-Cape Verde Escudo<br>KYD-
     * Cayman
     * Islands Dollar<br>CLP-Chilean Peso<br>CNY-Chinese Yuan<br>COP-Colombian Peso<br>KMF-
     * Comoros
     * Franc<br>CRC-Costa Rica Colon<br>HRK-Croatian Kuna<br>CUP-Cuban Peso<br>CYP-Cyprus
     * Pound<br>CZK-Czech
     * Koruna<br>DKK-Danish Krone<br>DJF-Djibouti Franc<br>DOP-Dominican Peso<br>XCD-East
     * Caribbean
     * Dollar<br>EGP-Egyptian Pound<br>SVC-El Salvador Colon<br>EEK-Estonian Kroon<br>ETB-
     * Ethiopian
     * Birr<br>EUR-Euro<br>FKP-Falkland Islands Pound<br>GMD-Gambian Dalasi<br>GHC-Ghanian
     * Cedi<br>GIP-Gibraltar
     * Pound<br>XAU-Gold Ounces<br>GTQ-Guatemala Quetzal<br>GNF-Guinea Franc<br>GYD-Guyana
     * Dollar<br>HTG-Haiti
     * Gourde<br>HNL-Honduras Lempira<br>HKD-Hong Kong Dollar<br>HUF-Hungarian Forint<br>ISK-
     * Iceland
     * Krona<br>INR-Indian Rupee<br>IDR-Indonesian Rupiah<br>IQD-Iraqi Dinar<br>ILS-Israeli
     * Shekel<br>JMD-Jamaican
     * Dollar<br>JPY-Japanese Yen<br>JOD-Jordanian Dinar<br>KZT-Kazakhstan Tenge<br>KES-Kenyan
     * Shilling<br>KRW-Korean Won<br>KWD-Kuwaiti Dinar<br>LAK-Lao Kip<br>LVL-Latvian Lat<br>LBP-
     * Lebanese
     * Pound<br>LSL-Lesotho Loti<br>LRD-Liberian Dollar<br>LYD-Libyan Dinar<br>LTL-Lithuanian
     * Lita<br>MOP-Macau Pataca<br>MKD-Macedonian Denar<br>MGF-Malagasy Franc<br>MWK-Malawi
     * Kwacha<br>MYR-Malaysian

```

```

* Ringgit<br>MVR-Maldives  Rufiyaa<br>MTL-Maltese  Lira<br>MRO-Mauritania  Ougulya<br>MUR-
Mauritius
* Rupee<br>MXN-Mexican Peso<br>MDL-Moldovan Leu<br>MNT-Mongolian Tugrik<br>MAD-Moroccan
* Dirham<br>MZM-Mozambique  Metical<br>MMK-Myanmar  Kyat<br>NAD-Namibian  Dollar<br>NPR-
Nepalese
* Rupee<br>ANG-Neth  Antilles  Guilder<br>NZD-New  Zealand  Dollar<br>NIO-Nicaragua
Cordoba<br>NGN-Nigerian
* Naira<br>KPW-North Korean Won<br>NOK-Norwegian Krone<br>OMR-Omani Rial<br>XPF-Pacific
* Franc<br>PKR-Pakistani Rupee<br>XPD-Palladium Ounces<br>PAB-Panama Balboa<br>PGK-Papua
* New Guinea Kina<br>PYG-Paraguayan Guarani<br>PEN-Peruvian Nuevo Sol<br>PHP-Philippine
* Peso<br>XPT-Platinum  Ounces<br>PLN-Polish  Zloty<br>QAR-Qatar  Rial<br>ROL-Romanian
Leu<br>RUB-Russian
* Rouble<br>WST-Samoa  Tala<br>STD-Sao  Tome  Dobra<br>SAR-Saudi  Arabian  Riyal<br>SCR-
Seychelles
* Rupee<br>SLL-Sierra Leone  Leone<br>XAG-Silver  Ounces<br>SGD-Singapore  Dollar<br>SKK-
Slovak
* Koruna<br>SIT-Slovenian  Tolar<br>SBD-Solomon  Islands  Dollar<br>SOS-Somali
Shilling<br>ZAR-South
* African Rand<br>LKR-Sri Lanka Rupee<br>SHP-St Helena Pound<br>SDD-Sudanese Dinar<br>SRG-
Surinam
* Guilder<br>SZL-Swaziland Lilageni<br>SEK-Swedish Krona<br>TRY-Turkey Lira<br>CHF-Swiss
* Franc<br>SYP-Syrian  Pound<br>TWD-Taiwan  Dollar<br>TZS-Tanzanian  Shilling<br>THB-Thai
Baht<br>TOP-Tonga
* Pa'anga<br>TTD-Trinidad&Tobago  Dollar<br>TND-Tunisian  Dinar<br>TRL-Turkish
Lira<br>USD-U.S.
* Dollar<br>AED-UAE Dirham<br>UGX-Ugandan Shilling<br>UAH-Ukraine Hryvnia<br>UYU-Uruguayan
* New Peso<br>VUV-Vanuatu Vatu<br>VEB-Venezuelan Bolivar<br>VND-Vietnam Dong<br>YER-Yemen
* Riyal<br>YUM-Yugoslav  Dinar<br>ZMK-Zambian  Kwacha<br>ZWD-Zimbabwe
Dollar</font></p></blockquote>
*
*
* @param ConversionRate $parameters
* @return ConversionRateResponse
*/
public function ConversionRate(ConversionRate $parameters) {
    return $this-> soapCall('ConversionRate', array($parameters), array(
        'uri' => 'http://www.webserviceX.NET/',
        'soapaction' => ''
    ));
}

}

?>

```

Anexo III - globalweather.wsdl

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:tns="http://www.webserviceX.NET"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    targetNamespace="http://www.webserviceX.NET"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://www.webserviceX.NET">
      <s:element name="GetWeather">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="CityName" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="CountryName" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetWeatherResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="GetWeatherResult" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetCitiesByCountry">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="CountryName" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetCitiesByCountryResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="GetCitiesByCountryResult"
              type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="string" nillable="true" type="s:string" />
    </s:schema>
  </wsdl:types>
  <wsdl:message name="GetWeatherSoapIn">
    <wsdl:part name="parameters" element="tns:GetWeather" />
  </wsdl:message>
  <wsdl:message name="GetWeatherSoapOut">
    <wsdl:part name="parameters" element="tns:GetWeatherResponse" />
  </wsdl:message>
  <wsdl:message name="GetCitiesByCountrySoapIn">
    <wsdl:part name="parameters" element="tns:GetCitiesByCountry" />
  </wsdl:message>
  <wsdl:message name="GetCitiesByCountrySoapOut">
    <wsdl:part name="parameters" element="tns:GetCitiesByCountryResponse" />
  </wsdl:message>
  <wsdl:message name="GetWeatherHttpGetIn">
    <wsdl:part name="CityName" type="s:string" />
    <wsdl:part name="CountryName" type="s:string" />
  </wsdl:message>
  <wsdl:message name="GetWeatherHttpGetOut">
    <wsdl:part name="Body" element="tns:string" />
  </wsdl:message>
  <wsdl:message name="GetCitiesByCountryHttpGetIn">
    <wsdl:part name="CountryName" type="s:string" />
  </wsdl:message>
  <wsdl:message name="GetCitiesByCountryHttpGetOut">
    <wsdl:part name="Body" element="tns:string" />
  </wsdl:message>
  <wsdl:message name="GetWeatherHttpPostIn">
    <wsdl:part name="CityName" type="s:string" />
    <wsdl:part name="CountryName" type="s:string" />
  </wsdl:message>
  <wsdl:message name="GetWeatherHttpPostOut">
```

```

    <wsdl:part name="Body" element="tns:string" />
  </wsdl:message>
  <wsdl:message name="GetCitiesByCountryHttpPostIn">
    <wsdl:part name="CountryName" type="s:string" />
  </wsdl:message>
  <wsdl:message name="GetCitiesByCountryHttpPostOut">
    <wsdl:part name="Body" element="tns:string" />
  </wsdl:message>
  <wsdl:portType name="GlobalWeatherSoap">
    <wsdl:operation name="GetWeather">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get weather report for
all major cities around the world.</wsdl:documentation>
      <wsdl:input message="tns:GetWeatherSoapIn" />
      <wsdl:output message="tns:GetWeatherSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="GetCitiesByCountry">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get all major cities
by country name(full / part).</wsdl:documentation>
      <wsdl:input message="tns:GetCitiesByCountrySoapIn" />
      <wsdl:output message="tns:GetCitiesByCountrySoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="GlobalWeatherHttpGet">
    <wsdl:operation name="GetWeather">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get weather report for
all major cities around the world.</wsdl:documentation>
      <wsdl:input message="tns:GetWeatherHttpGetIn" />
      <wsdl:output message="tns:GetWeatherHttpGetOut" />
    </wsdl:operation>
    <wsdl:operation name="GetCitiesByCountry">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get all major cities
by country name(full / part).</wsdl:documentation>
      <wsdl:input message="tns:GetCitiesByCountryHttpGetIn" />
      <wsdl:output message="tns:GetCitiesByCountryHttpGetOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="GlobalWeatherHttpPost">
    <wsdl:operation name="GetWeather">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get weather report for
all major cities around the world.</wsdl:documentation>
      <wsdl:input message="tns:GetWeatherHttpPostIn" />
      <wsdl:output message="tns:GetWeatherHttpPostOut" />
    </wsdl:operation>
    <wsdl:operation name="GetCitiesByCountry">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get all major cities
by country name(full / part).</wsdl:documentation>
      <wsdl:input message="tns:GetCitiesByCountryHttpPostIn" />
      <wsdl:output message="tns:GetCitiesByCountryHttpPostOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="GlobalWeatherSoap" type="tns:GlobalWeatherSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetWeather">
      <soap:operation soapAction="http://www.webserviceX.NET/GetWeather" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetCitiesByCountry">
      <soap:operation
style="document" soapAction="http://www.webserviceX.NET/GetCitiesByCountry" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="GlobalWeatherSoap12" type="tns:GlobalWeatherSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetWeather">
      <soap12:operation soapAction="http://www.webserviceX.NET/GetWeather" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>

```

```

        </wsdl:input>
        <wsdl:output>
            <soap12:body use="literal" />
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetCitiesByCountry">
        <soap12:operation
            soapAction="http://www.webserviceX.NET/GetCitiesByCountry"
style="document" />
        <wsdl:input>
            <soap12:body use="literal" />
        </wsdl:input>
        <wsdl:output>
            <soap12:body use="literal" />
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="GlobalWeatherHttpGet" type="tns:GlobalWeatherHttpGet">
    <http:binding verb="GET" />
    <wsdl:operation name="GetWeather">
        <http:operation location="/GetWeather" />
        <wsdl:input>
            <http:urlEncoded />
        </wsdl:input>
        <wsdl:output>
            <mime:mimeXml part="Body" />
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetCitiesByCountry">
        <http:operation location="/GetCitiesByCountry" />
        <wsdl:input>
            <http:urlEncoded />
        </wsdl:input>
        <wsdl:output>
            <mime:mimeXml part="Body" />
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="GlobalWeatherHttpPost" type="tns:GlobalWeatherHttpPost">
    <http:binding verb="POST" />
    <wsdl:operation name="GetWeather">
        <http:operation location="/GetWeather" />
        <wsdl:input>
            <mime:content type="application/x-www-form-urlencoded" />
        </wsdl:input>
        <wsdl:output>
            <mime:mimeXml part="Body" />
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetCitiesByCountry">
        <http:operation location="/GetCitiesByCountry" />
        <wsdl:input>
            <mime:content type="application/x-www-form-urlencoded" />
        </wsdl:input>
        <wsdl:output>
            <mime:mimeXml part="Body" />
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:service name="GlobalWeather">
    <wsdl:port name="GlobalWeatherSoap" binding="tns:GlobalWeatherSoap">
        <soap:address location="http://www.webservices.com/globalweather.asmx" />
    </wsdl:port>
    <wsdl:port name="GlobalWeatherSoap12" binding="tns:GlobalWeatherSoap12">
        <soap12:address location="http://www.webservices.com/globalweather.asmx" />
    </wsdl:port>
    <wsdl:port name="GlobalWeatherHttpGet" binding="tns:GlobalWeatherHttpGet">
        <http:address location="http://www.webservices.com/globalweather.asmx" />
    </wsdl:port>
    <wsdl:port name="GlobalWeatherHttpPost" binding="tns:GlobalWeatherHttpPost">
        <http:address location="http://www.webservices.com/globalweather.asmx" />
    </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

TEMA 7

Contenido

1.- Programación del cliente web.....	2
1.1.- Página web dinámicas.	3
1.2.- El lenguaje JavaScript.	4
1.3.- Comunicación asíncrona con el servidor web: AJAX.....	6
2.- PHP y JavaScript.	8
2.1.- Aplicaciones web con PHP y JavaScript.	9
2.2.- Aplicaciones web con PHP y JavaScript (II).....	10
form.php	11
validar.php	11
estilos.css	12
2.3.- Aplicaciones web con PHP y JavaScript (III).....	13
form.php	13
validar.php	14
validar.js	15
3.- Utilización de AJAX con PHP.....	16
3.1.- Xajax.	17
3.2.- Xajax (II).	18
3.3.- Xajax (III).	19
3.4.- Xajax (IV).	20
form.php	20
validar.js	22
estilos.css	22
login.php	23
login.php	24
valida.php.....	25
validar.js	26
3.5.- JQuery4PHP.....	26
3.6.- JQuery4PHP (II).	27
3.7.- JQuery4PHP (III).	28
form.php	29
validar.php	30
estilos.css	30
3.8.- JQuery4PHP (IV).	31
3.9.- JQuery4PHP (V).	31
form.php	32
estilos.css	34

Aplicaciones web dinámicas. PHP y JavaScript

Caso práctico

Juan y Carlos tienen definida la estructura de la aplicación web que deben desarrollar. Han llevado a cabo algunas pruebas de programación, y están contentos con los resultados obtenidos. Antes de seguir avanzando, deciden reunir al equipo de BK Programación para mostrarles los progresos realizados y el plan de desarrollo previsto.

Durante la presentación de la aplicación web, todo el equipo se muestra entusiasmado con el aspecto que está tomando el proyecto. **Ada**, la directora, les felicita por el trabajo que han llevado a cabo hasta el momento. **Ana**, que tiene experiencia como diseñadora gráfica, se ofrece a ayudarles con el aspecto visual del interfaz web. Pero de todas las opiniones, hay una que les llama la atención.

María, que se encarga del mantenimiento de servidores y sitios web, les pregunta si han tenido en cuenta la posibilidad de **integrar en su aplicación algún tipo de código cliente**. Les comenta que muchas aplicaciones actuales lo utilizan dentro de su estructura para muy diversas funciones.

Juan y Carlos se miran, y saben que ambos están pensando lo mismo. Habrá que hacer un último esfuerzo e informarse sobre el tema. Si deciden que resulta interesante, aún están a tiempo de incorporarlo en su proyecto.

1.- Programación del cliente web

Caso práctico

El primer paso que deciden dar **Juan y Carlos**, siempre contando con el asesoramiento de **María**, es informarse sobre las tecnologías de programación web actuales. Conocen de oídas el lenguaje JavaScript, y saben cuáles son los fundamentos de la tecnología AJAX, pero no tienen claras las ventajas e inconvenientes que les puede suponer su incorporación al proyecto en el que están trabajando.

Y lo más importante; si finalmente consideran que puede ser ventajoso introducir en su aplicación web algún tipo de programación que se ejecute en el navegador... ¿cómo se puede integrar ésta con la programación del servidor web? ¿Pueden coexistir, y aún más, integrarse ambos lenguajes de alguna forma?

Siguiendo su método de trabajo, deciden buscar información por separado y compartirla pasados unos días.

Cuando comenzaste con el presente módulo, uno de los primeros conceptos que aprendiste es a diferenciar entre la ejecución de código en el servidor web y la ejecución de código en el navegador o cliente web.

Todo lo que has aprendido hasta el momento se ha centrado en la ejecución de código en el servidor web utilizando el lenguaje PHP. La otra parte importante de una aplicación web, la programación de código que se ejecute en el navegador, no forma parte de los contenidos de este módulo. Tiene su propio módulo dedicado, **Desarrollo Web en Entorno Cliente**.

Muchas de las aplicaciones web que existen en la actualidad tienen esos dos componentes: una parte de la aplicación, generalmente la que contiene la lógica de negocio, se ejecuta en el servidor; y otra parte de la aplicación, de menor peso, se ejecuta en el cliente. Existen incluso cierto tipo de aplicaciones web, como Google Docs, en las que gran parte de las funcionalidades que ofrecen se implementan utilizando programación del cliente web.

En la presente unidad vas a aprender cómo integrar estos dos componentes de una misma aplicación web: el código PHP que se ejecutará en el servidor, con el código que se enviará al cliente para que éste lo ejecute.

La programación de guiones para su ejecución en un cliente web es similar a lo que ya conoces sobre PHP, salvo que en este caso el código completo del guión llega al navegador junto con las etiquetas HTML, y es éste el encargado de procesarlo.

Así como el código PHP se marcaba utilizando los delimitadores `<?PHP` y `?>`, en HTML existe una etiqueta que se utiliza para integrar el código ejecutable por el navegador junto al resto de etiquetas. Se trata de la etiqueta `<script>`, que puede indicar tanto la localización del código en un fichero externo, como simplemente delimitar unas líneas de código dentro del propio fichero HTML.

```
// Inclusión de código en el documento HTML
<script type="text/javascript">
// Código que ejecuta el navegador
</script>

// Inclusión de código en un fichero externo
<script type="text/javascript" src="codigo.js"></script>
```

Cuando el código se incluye en el propio documento HTML, se suele encerrar en una sección `<CDATA` para no encontrar errores de validación en documentos XHTML. Estos errores aparecerían si dentro del código del guión hubiera algún carácter especial como `<` o `>`.

Sin embargo, al utilizar una sección `<CDATA` (forma de marcar una parte de un documento XML (XHTML en nuestro caso) para que no sea interpretada al procesar el documento. Las siglas se refieren a que esa parte del documento contiene sólo "Character Data"), puede suceder que cuando se procese la página como documento HTML, algún navegador no reconozca éstas secciones. Es por este motivo que el texto `<CDATA` que identifica estas secciones suele también ponerse dentro de un comentario (por ejemplo, utilizando `//` o `/* */`).

```
<script type="text/javascript">
/* <![CDATA[ */
// Código que ejecuta el navegador
/* ]]> */
</script>
```

¿Qué etiqueta HTML se usa para marcar el código que ejecutará el navegador?



CDATA.



script.

La etiqueta script puede incluir por sí misma el código que ejecutará el navegador, o indicar el fichero en que se encuentra.

1.1.- Página web dinámicas.

La inclusión de código en páginas web para su ejecución por parte del navegador tiene ciertas limitaciones:

- ✓ Cuando ejecutas código PHP en un servidor, es normalmente el programador el que tiene el control sobre el entorno de ejecución. Al cliente llegan únicamente etiquetas en lenguaje HTML o XHTML. Sin embargo, cuando programas código para que se ejecute en un cliente web, no tienes siquiera la certeza de que el navegador del usuario soporte la ejecución del código que recibe. Existen ciertos sistemas, como dispositivos móviles o navegadores integrados en hardware específico, que no permiten la ejecución de código de cliente.
- ✓ El código que se ejecuta en el navegador está normalmente limitado a ser ejecutado en un entorno controlado, que no permite, por ejemplo, la lectura o escritura de ficheros en el ordenador del usuario. De esta forma se restringen los efectos negativos que pueda causar un guión y se favorece la confianza del usuario en este tipo de código.

Pese a estas limitaciones, la ejecución de código en el navegador encaja perfectamente con cierto tipo de tareas como:

- ✓ Comprobar y/o procesar los datos que introduce el usuario en los formularios, como paso previo a su envío al servidor web.

- ✓ Gestionar diferentes marcos y/o ventanas del navegador.
- ✓ Modificar de forma dinámica los elementos que componen la página web, ajustando sus propiedades o estilos en respuesta a la interacción del usuario.

El código JavaScript de una página se puede ejecutar en respuesta a eventos generados por el navegador. Por ejemplo, utilizando el evento `onsubmit` podemos llamar a una función `validar_email` para validar una dirección de correo introducida por el usuario cuando se intenta enviar el formulario:

```
<form action="usuario.php" method="get" name="datos_usuario"
onsubmit="return validar_email()">
<input type="text" id="email" />
</form>
```

Para la función que realiza la validación básica de una dirección de email puedes utilizar, por ejemplo, el siguiente código:



```
function validar_email() {
    valor = document.getElementById("email").value;
    pos_arroba = valor.indexOf("@");
    pos_punto = valor.lastIndexOf(".");
    if (pos_arroba < 1 || pos_punto < pos_arroba+2
|| pos_punto+2>=valor.length) {
        alert('Dirección de correo no válida.');
```

Las páginas web que se aprovechan de las capacidades de ejecución de código en el cliente para cambiar su apariencia, o su funcionamiento, se conocen como páginas web dinámicas. Se llama **HTML dinámico** (DHTML) al conjunto de técnicas que emplean HTML, el modelo de objetos del documento web modelo de objetos del documento web (*DOM: referencia a un conjunto de objetos para representar un documento HTML y XHTML, que pueden ser usados para acceder mediante programación a las distintas partes del mismo*), hojas de estilo CSS y lenguaje ejecutado en el navegador para crear sitios webs dinámicos.

1.2.- El lenguaje JavaScript.

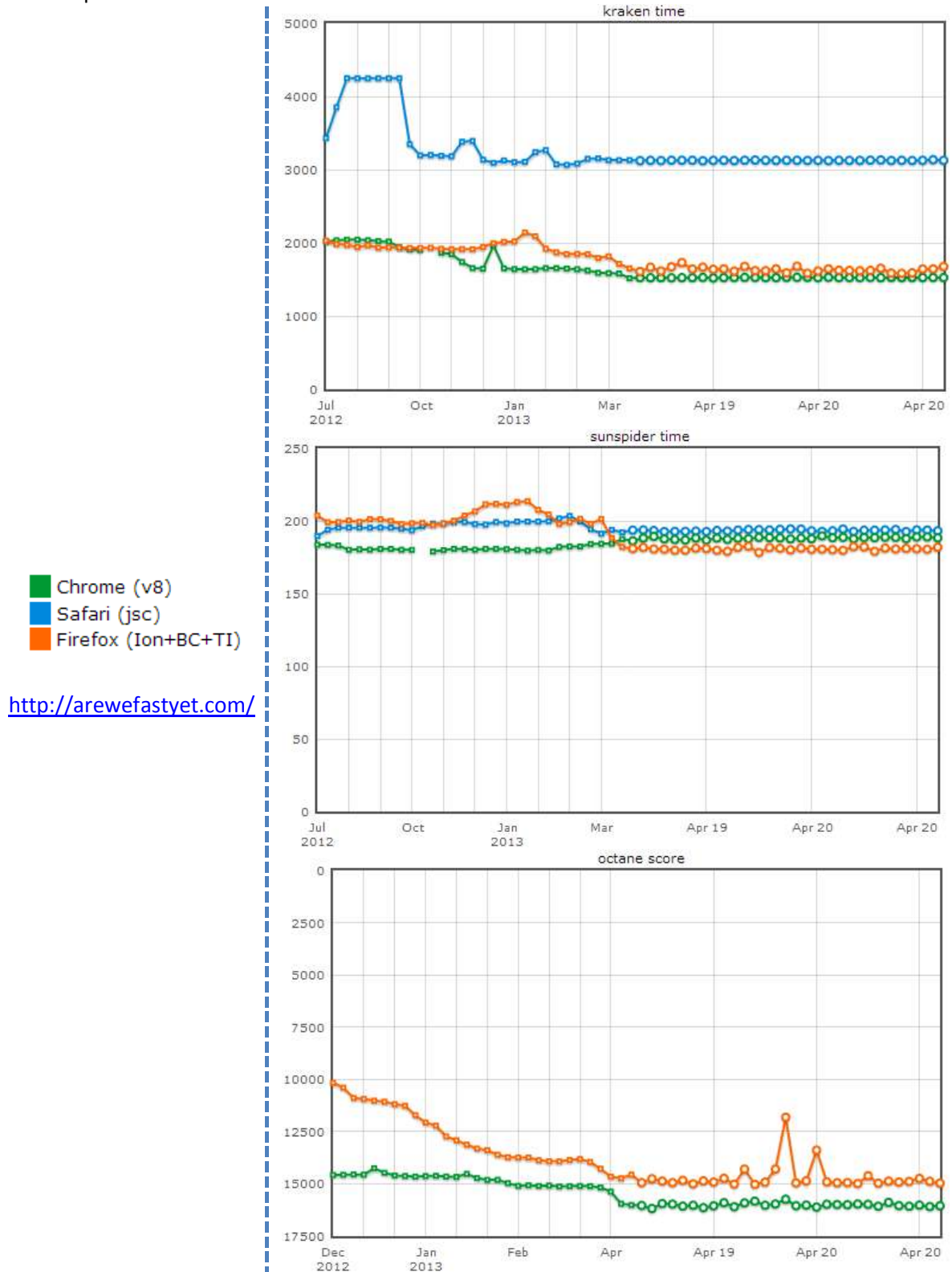
El lenguaje de guiones que se utiliza mayoritariamente hoy en día para la programación de clientes web es JavaScript. Su sintaxis está basada en la del lenguaje C, parecida a la que conocemos del lenguaje PHP. Aunque su utilización principal es incorporarlo a páginas web, también puedes encontrar **JavaScript** en otros lugares como en documentos PDF, o para definir la funcionalidad de extensiones de escritorio o de algunas aplicaciones (*widgets (aplicación, generalmente pequeña, que se ejecuta sobre otra aplicación denominada motor de widgets, y con frecuencia orientada a incorporar alguna funcionalidad adicional o a mejorar el aspecto visual de otra aplicación o del escritorio del sistema operativo del usuario)*).

Si bien, la gran mayoría de navegadores web soportan código en lenguaje JavaScript, debes tener en cuenta que:

- ✓ La ejecución de JavaScript en el navegador puede haber sido deshabilitada por el usuario.
- ✓ La implementación de JavaScript puede variar de un navegador a otro. Lo mismo sucede con el interface de programación que usa JavaScript para acceder a la estructura de las páginas web: el DOM. Por este motivo, es conveniente que verifiques la funcionalidad del código en diversos navegadores antes de publicarlo como parte de tu sitio web.

Se conoce como motor JavaScript a la parte del navegador encargada de interpretar y ejecutar el código JavaScript que forma parte de las páginas web. Los motores JavaScript que se incluyen en los navegadores han experimentado una importante mejora de rendimiento en los últimos tiempos.

Existen pruebas específicas destinadas a medir la velocidad de ejecución de distintos motores JavaScript:



Aunque no vamos a aprender en esta unidad a programar en JavaScript, deberías saber cómo depurar el código que vamos a utilizar. Es conveniente que manejes un depurador para cada navegador que utilices, pero para esta unidad llegará con la **extensión** Firebug para el navegador Firefox.

<https://addons.mozilla.org/es-es/firefox/addon/firebug/>



¿Cuál es una de las principales desventajas de la programación del cliente web?



Que no es posible asegurar que el navegador vaya a ejecutar el código.



Que el navegador puede no ser capaz de mostrar correctamente la página al confundir el código con las etiquetas HTML / XHTML.

La ejecución de código JavaScript puede estar deshabilitada por el usuario, o el navegador puede no soportar la ejecución de código.

1.3.- Comunicación asíncrona con el servidor web: AJAX.



Una de las principales causas de la evolución de JavaScript en los últimos tiempos es, sin duda, la tecnología **AJAX**. Como ya vimos en la primera unidad, el término AJAX hace referencia a la posibilidad de una página web de establecer una comunicación con un servidor web y recibir una respuesta sin necesidad de que el navegador recargue la página.

AJAX utiliza el objeto `XMLHttpRequest`, creado originariamente por Microsoft como parte de su librería MSXML, y que hoy en día se ha incorporado de forma nativa a todos los navegadores actuales.

Pese al nombre del objeto, y a que la letra X de las siglas AJAX hace referencia a XML, la información que se transmite de forma asíncrona entre el navegador y el servidor web no es necesario que se encuentre en formato XML.

Entre las tareas que puedes llevar a cabo gracias a AJAX están:

- ✓ Actualizar el contenido de una página web sin necesidad de recargarla.
- ✓ Pedir y recibir información desde un servidor web manteniendo la página cargada en el navegador.
- ✓ Enviar información de la página a un servidor web en segundo plano.

Dependiendo del navegador del usuario, y de si utiliza una versión antigua o moderna, tendrás que usar un método u otro para crear el objeto `XMLHttpRequest`:

```
// Distintas formas para crear el objeto
// XMLHttpRequest según el navegador
xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
xmlhttp = new XMLHttpRequest();
```

Afortunadamente, muchas de las librerías JavaScript de las que hablábamos antes soportan también AJAX, y utilizan un código adaptado según el navegador del usuario. Si utilizas una de estas librerías podrás ahorrarte muchos quebraderos de cabeza al programar. Por ejemplo, si utilizas jQuery podrías utilizar AJAX para enviar en segundo plano el email validado en el ejemplo anterior. Simplemente tendrías que incluir en el HTML la librería jQuery.

```
// Utilizamos la versión de jQuery disponible en las CDN de Google
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js" type="text/javascript"></script>
```

Y tras el código de validación, debes ejecutar la función **ajax** incluida en la librería jQuery:

```
function validar_email() {
    valor = document.getElementById("email").value;

    // Aquí iría el código de validación

    $.ajax({
        type: "POST", url: "email.php", data: "email=" + valor,
        statusCode: {
            404: function() { alert('Página no encontrada'); }
        },
        success: function(result) { alert( "Resultado: " + result ); }
    });
    return false;
}
```

```
}
```

La función anterior envía la dirección de email introducida mediante POST a la página `email.php`, y muestra un mensaje con el resultado obtenido.

2.- PHP y JavaScript.

Caso práctico

Pasados unos días, **Juan y Carlos** se vuelven a reunir y deciden que en el diseño de su aplicación existen muchos lugares en los que podrían aprovechar las capacidades que ofrece la programación del navegador web, concretamente el lenguaje JavaScript.

Sin embargo, ambos siguen sin tener claro cómo pueden integrar el código del cliente web en una aplicación programada en lenguaje PHP. Animados por **María**, que les orienta sobre el rumbo que deben tomar, preparan una serie de pruebas de programación que les puedan orientar sobre el tema.

Si sabes programar aplicaciones que se ejecuten en un servidor web (con un lenguaje como PHP) y en el navegador del usuario (con JavaScript/jQuery), tienes en tu mano las herramientas necesarias para construir aplicaciones web completas. Sin embargo, es necesario que antes de comenzar tengas claras las funcionalidades que soporta cada una de estas tecnologías de desarrollo, y cómo puedes hacer para utilizar ambas a la vez.

Llevándolo a los extremos, podrías hacer aplicaciones en PHP que utilicen programación del cliente simplemente para tareas sencillas como verificar campos en los formularios antes de enviarlos. O, por el contrario, sería también posible programar aplicaciones completas que se ejecuten en el navegador del usuario, dejando el lenguaje del servidor para proporcionar ciertas funcionalidades como almacenamiento en bases de datos.

Una alternativa no es necesariamente mejor que la otra. Es necesario analizar de forma independiente la lógica de cada aplicación, de manera que no se malgasten los recursos del servidor realizando tareas que podrían trasladarse al cliente web. En ocasiones también es necesario comprobar los tiempos de carga de las páginas y el tamaño de las mismas. Puede ser preferible utilizar AJAX para, por ejemplo, enviar nuevos registros al servidor, si el esfuerzo que invertimos redundaría en un interfaz de usuario más ágil y usable. En cualquier caso, la consistencia y robustez de la aplicación no debe verse afectada.

Si decides unir en una aplicación programación del cliente web con programación del servidor web, habrá ocasiones en que necesites comunicar ambos lenguajes. En el ejemplo anterior ya has comprobado cómo puedes hacer para pasar un valor o una variable JavaScript desde el navegador web a un guión en PHP: enviándolo como parámetro **POST** o **GET** en una petición de nueva página, bien sea al cargarla en el navegador o utilizando AJAX en una comunicación asíncrona:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Desarrollo Web</title>
  </head>
  <body>
    <script type="text/javascript">
      <?php
        // Creamos en la página un código JavaScript que
        // utiliza la variable PHP "$email"
        $email = "alumno@educacion.es";
        print 'window.open("email.php?email='.urlencode($email).'");';
      ?>
    </script>
  </body>
</html>
```

En ambos casos deberás asegurarte de que las cadenas que insertes en las direcciones URL no incluyan caracteres inválidos. Para evitarlo, en PHP puedes usar la función `urlencode`, y en JavaScript `encodeURIComponent`.

¿Cómo es posible obtener en PHP el contenido de una variable JavaScript?



En una petición de nueva página, como parámetro POST o GET.



Enviando desde el servidor web una solicitud al navegador web.

Además, ten en cuenta que cualquier contenido que forme parte de una URL debería procesarse previamente utilizando la función `JavaScript encodeURIComponent`.

2.1.- Aplicaciones web con PHP y JavaScript.

Con lo que has visto hasta el momento, seguramente, ya te has hecho una idea de qué tareas son las que con más frecuencia hacen uso de JavaScript. Y si hay una que destaca sobre el resto es, sin duda, la validación de formularios, que vamos a utilizar en los ejemplos que se desarrollan a lo largo de la presente unidad.

Como ya sabes, el mecanismo que utilizan las aplicaciones web para permitir al usuario la entrada de información es el formulario HTML. Los datos que se introducen en un formulario web se envían al servidor utilizando bien el método `POST`, bien el método `GET`. Muchos de los campos de los formularios tienen, normalmente, restricciones sobre el contenido que se debe rellenar. La validación de un formulario web consiste en comprobar que el contenido introducido en todos los componentes del mismo cumpla estas restricciones.

Si no utilizas código ejecutable en el navegador, la única forma de validar un formulario consiste en enviarlo al servidor web para comprobar si existen errores en los datos. En caso de que así sea, habrá que volver a enviar el formulario al navegador del usuario mostrando las advertencias oportunas.

Obviamente, si utilizas JavaScript en tus aplicaciones, la validación se puede realizar en el cliente web. De esta forma el proceso es mucho más rápido. No es necesario enviar la información al servidor hasta que se haya comprobado que no existen errores de validación.

Sin embargo, aunque parecen claras las ventajas de la validación de formularios en el cliente web, hay ocasiones en las que ésta no es posible. Ya vimos que no siempre podrás asegurar que el navegador que utiliza el usuario tiene capacidad para ejecutar código JavaScript, o incluso puede suceder que se haya deshabilitado por motivos de seguridad.

En los casos que no sea posible asegurar la capacidad de ejecución de código de los clientes, la solución óptima sería utilizar un escenario dual: diseñar las aplicaciones suponiendo que es posible ejecutar JavaScript en los clientes, y al mismo tiempo prever la posibilidad de que no sea así. Por ejemplo, en el caso de la validación del formulario, podrías crear el código JavaScript de validación y, si en algún cliente este código no se puede ejecutar, preparar un código PHP similar que realice la validación en el servidor.

Por ejemplo, supongamos que queremos validar los datos que se introducen en el siguiente formulario web:

Un formulario web con el título "Introducción de datos". Contiene cuatro campos de texto etiquetados como "Nombre:", "Contraseña:", "Repite la contraseña:" y "Email:". Debajo de los campos hay un botón etiquetado como "Enviar".

Si realizas la validación en PHP, podrías generar junto con la página web el texto con las advertencias de validación. Y si el formulario lo quieres validar también utilizando JavaScript, tendrás que crear los mismos textos o similares. Una posibilidad para no repetir el código que introduce esos textos en el cliente y en el servidor, es introducir los textos de validación en las etiquetas HTML de la página web, y utilizar estilos para mostrarlos, o no, según sea oportuno.

Por ejemplo, para realizar la validación del formulario web anterior, puedes crear los siguientes textos en HTML (asociados al nombre, las contraseñas y la dirección de correo respectivamente):

```
<span class='error'>Debe tener más de 3 caracteres.</span>
<span class='error'>Debe ser mayor de 5 caracteres o no coinciden.</span>
<span class='error'>La dirección de email no es válida.</span>
```

El código PHP y JavaScript deberá ocultar cada uno de los textos cuando la validación de su elemento respectivo sea correcta. Por ejemplo, si el nombre tiene más de tres letras, validará correctamente y no se deberá mostrar el primer mensaje.

2.2.- Aplicaciones web con PHP y JavaScript (II).

Para realizar la validación del formulario en el servidor web utilizando PHP, necesitarás utilizar las siguientes funciones o similares:

```
<?php
function validarNombre($nombre){
    if(strlen($nombre) < 4) return false;
    return true;
}

function validarEmail($email){
    return preg_match("/^[a-z0-9]+([ \\. -][a-z0-9]+)*@([a-z0-9]+([ \\. -][a-z0-9]+)*)+\.[a-z]{2,}$/i", $email);
}

function validarPasswords($pass1, $pass2) {
    return $pass1 == $pass2 && strlen($pass1) > 5;
}

function validar($nombre, $email, $pass1, $pass2) {
    return validarNombre($nombre) && validarEmail($email)
    && validarPasswords($pass1, $pass2);
}
?>
```

Fíjate en el uso de la función `preg_match` y de expresiones regulares (*expresión regular o patrón es una cadena de texto compuesta por un conjunto de caracteres, algunos de los cuales tienen significado especial, que permite definir una serie de reglas con las que comprobar la validez, o no, de otras cadenas de texto*) para validar la dirección de correo.

Función `preg_match`

<http://es2.php.net/manual/es/function.preg-match.php>

Ten en cuenta que las barras invertidas (`\`) tienen un significado especial dentro de una cadena de PHP (*sirven para escapar el siguiente carácter, por ejemplo por si queremos mostrar unas comillas*); por ese motivo, la doble barra `\\` se convierte en una barra simple `\` cuando se interpreta la expresión regular.

En ocasiones es importante saber construir expresiones regulares en PHP para realizar comparación de cadenas de texto con patrones. Tienes más información en el siguiente enlace.

Expresiones regulares en PHP

<http://www.desarrolloweb.com/manuales/expresiones-regulares.html>

Con las funciones anteriores, puedes crear código en PHP que oculte los mensajes de validación cuando no sean necesarios:

```
<span id='errorNombre' for='nombre' class='<?php if(!isset($_POST['enviar'])) ||
validarNombre($_POST['nombre'])) echo "oculto "; ?>error'>Debe tener más de 3
caracteres.</span>
<span id='errorPassword' for='password' class='<?php if(!isset($_POST['enviar'])) ||
validarPasswords($_POST['password1'], $_POST['password2'])) echo "oculto "; ?>error'>Debe ser
mayor de 5 caracteres o no coinciden.</span>
<span id='errorEmail' for='email' class='<?php if(!isset($_POST['enviar'])) ||
validarEmail($_POST['email'])) echo "oculto "; ?>error'>La dirección de email no es
válida.</span>
```

El código anterior aplica la clase `oculto` a los textos de validación que no sea necesario mostrar. En la hoja de estilos correspondientes, deberás definir para esa clase un estilo como `display:none` o `visibility: hidden`.

¿Qué significa la siguiente expresión regular: `\.[a-z]{2,}`?



Una barra seguida por un carácter cualquier, una letra entre la a y la z, y un número mayor o igual a 2.



Un punto seguido de 2 o más letras minúsculas.

form.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 7 : Aplicaciones web dinámicas: PHP y Javascript -->
<!-- Ejemplo Validación formulario con PHP: form.php -->
<?php require_once("validar.php"); ?>
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <title>Ejemplo Tema 7: Validación formulario</title>
  <link rel="stylesheet" href="estilos.css" type="text/css" />
</head>

<body>
  <div id='form'>
    <form id='datos' action='form.php' method='post'>
      <fieldset >
        <legend>Introducción de datos</legend>
        <div class='campo'>
          <label for='nombre' >Nombre:</label>
          <input type='text' name='nombre' id='nombre' maxlength="50" value="<?php echo
$_POST['nombre'] ?>" /><br />
          <span id='errorNombre' for='nombre' class='<?php if(!isset($_POST['enviar']) ||
validarNombre($_POST['nombre'])) echo "oculto "; ?>error'>Debe tener más de 3
caracteres.</span>
        </div>
        <div class='campo'>
          <label for='password1' >Contraseña:</label>
          <input type='password' name='password1' id='password1' maxlength="50" value="<?php
echo $_POST['password1'] ?>" /><br />
          <span id='errorPassword' for='password' class='<?php if(!isset($_POST['enviar'])
|| validarPasswords($_POST['password1'], $_POST['password2'])) echo "oculto "; ?>error'>Debe
ser mayor de 5 caracteres o no coinciden.</span>
        </div>
        <div class='campo'>
          <label for='password2' >Repita la contraseña:</label>
          <input type='password' name='password2' id='password2' maxlength="50" value="<?php
echo $_POST['password1'] ?>" />
        </div>
        <div class='campo'>
          <label for='email' >Email:</label>
          <input type='text' name='email' id='email' maxlength="50" value="<?php echo
$_POST['email'] ?>" /><br />
          <span id='errorEmail' for='email' class='<?php if(!isset($_POST['enviar']) ||
validarEmail($_POST['email'])) echo "oculto "; ?>error'>La dirección de email no es
válida.</span>
        </div>

        <div class='campo'>
          <input type='submit' name='enviar' value='Enviar' />
        </div>
      </fieldset>
    </form>
  </div>
</body>
</html>
```

validar.php

```
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 7 : Aplicaciones web dinámicas: PHP y Javascript -->
<!-- Ejemplo Validación formulario: validar.php -->
<?php
```

```
function validarNombre($nombre){
    if(strlen($nombre) < 4) return false;
    return true;
}

function validarEmail($email){
    return preg_match("/^[a-z0-9]+([_\\.-][a-z0-9]+)*@[a-z0-9]+(\\.[a-z]{2,})$/i", $email);
}

function validarPasswords($pass1, $pass2) {
    return $pass1 == $pass2 && strlen($pass1) > 5;
}

?>
```

estilos.css

```
#form fieldset {
    position: absolute;
    left: 50%;
    top: 50%;
    width: 340px;
    margin-left: -170px;
    height: 330px;
    margin-top: -150px;
    padding: 10px;
    border: 1px solid #ccc;
    background-color: #eee;
}

legend, h3 {
    font-family : Arial, sans-serif;
    font-size: 1.3em;
    font-weight: bold;
    color: #333;
}

#form .campo {
    margin-top: 8px;
    margin-bottom: 10px;
    margin-left: 10px;
}

#form label {
    font-family : Arial, sans-serif;
    font-size: 0.8em;
    font-weight: bold;
}

#form input[type="text"], #form input[type="password"] {
    font-family : Arial, Verdana, sans-serif;
    font-size: 0.8em;
    line-height: 140%;
    color : #000;
    padding : 3px;
    border : 1px solid #999;
    height: 18px;
    width: 280px;
}

#form input[type="submit"] {
    width: 100px;
    height: 30px;
    padding-left: 0px;
}

.error{
    font-size: 10px;
    text-decoration: underline;
    background: #ffdddd;
    color: #ee2211;
}

.oculto {
    display: none;
}
```

Introducción de datos

Nombre: Alonso

Contraseña:

Debe ser mayor de 7 caracteres y no coincidir.

Repita la contraseña:

Email:

La contraseña no está en español.

Enviar

2.3.- Aplicaciones web con PHP y JavaScript (III).

Partiendo de la página PHP anterior, que ya incluye código para validar el formulario web en el servidor, vas a ver cómo puedes hacer para incorporarle código en JavaScript que realice la misma validación en el cliente. De esta forma, si el navegador del usuario soporta JavaScript, se reducirá el procesamiento del servidor y la transferencia de información entre éste y el cliente.

Crearás todo el código necesario en un archivo externo que llamaremos `validar.js`. Como vamos a utilizar la librería jQuery, tendrás que añadir las dos líneas siguientes a la página anterior:

```
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
<script type="text/javascript" src="validar.js"></script>
```

En el código JavaScript habrá que definir unas funciones de validación similares a las programadas anteriormente en PHP:

```
function validarNombre(){
    if(nombre.val().length < 4) {
        errorNombre.removeClass("oculto");
        return false;
    }
    errorNombre.addClass("oculto");
    return true;
}

function validarEmail(){
    ...
}

function validarPasswords(){
    ...
}

function validar(){
    return validarNombre() & validarEmail() & validarPasswords();
}
```

En JavaScript usamos la función `match` para validar las direcciones de email. Las expresiones regulares que admite esta función no son exactamente iguales a las que viste anteriormente para la función `preg_match` de PHP.



jQuery
New Wave JavaScript

Las funciones usan los métodos de jQuery `addClass` y `removeClass` para incorporar, y quitar, respectivamente, los textos de validación a la clase `oculto`, lo mismo que hacía el código PHP anterior. Para seleccionar los elementos a ocultar, puedes utilizar también jQuery. Por ejemplo, para seleccionar el elemento de la página web con identificador `nombre`, se pone:

```
var nombre = $("#nombre");
```

Como no quieres que el formulario se envíe realmente, captura el evento `submit` del formulario forzando a que se valide con la función de JavaScript. Si la validación es correcta (`return true;`), el formulario finalmente se envía.

```
$("#datos").submit(function(){
    if(validar()) return true;
    else return false;
});
```

El código final del ejemplo realiza la validación tanto en el cliente (si éste soporta JavaScript) como en el servidor web (si no soporta JavaScript). Puedes probar a deshabilitar la ejecución de JavaScript en el navegador para comprobar que el formulario se sigue validando.

form.php

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

```

<!-- Tema 7 : Aplicaciones web dinámicas: PHP y Javascript -->
<!-- Ejemplo Validación formulario con PHP y JavaScript: form.php -->
<?php require_once("validar.php"); ?>
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Ejemplo Tema 7: Validación formulario</title>
    <link rel="stylesheet" href="estilos.css" type="text/css" />
</head>

<body>
    <div id='form'>
        <form id='datos' action='form.php' method='post'>
            <fieldset >
                <legend>Introducción de datos</legend>
                <div class='campo'>
                    <label for='nombre' >Nombre:</label>
                    <input type='text' name='nombre' id='nombre' maxlength="50" value="<?php echo
$_POST['nombre'] ?>" /><br />
                    <span id='errorNombre' for='nombre' class='<?php if(!isset($_POST['enviar'])) ||
validarNombre($_POST['nombre']) echo "oculto "; ?>error'>Debe tener más de 3
caracteres.</span>
                </div>
                <div class='campo'>
                    <label for='password1' >Contraseña:</label>
                    <input type='password' name='password1' id='password1' maxlength="50" value="<?php
echo $_POST['password1'] ?>" /><br />
                    <span id='errorPassword' for='password' class='<?php if(!isset($_POST['enviar']))
|| validarPasswords($_POST['password1'], $_POST['password2']) echo "oculto "; ?>error'>Debe
ser mayor de 5 caracteres o no coinciden.</span>
                </div>
                <div class='campo'>
                    <label for='password2' >Repita la contraseña:</label>
                    <input type='password' name='password2' id='password2' maxlength="50" value="<?php
echo $_POST['password1'] ?>" />
                </div>
                <div class='campo'>
                    <label for='email' >Email:</label>
                    <input type='text' name='email' id='email' maxlength="50" value="<?php echo
$_POST['email'] ?>" /><br />
                    <span id='errorEmail' for='email' class='<?php if(!isset($_POST['enviar'])) ||
validarEmail($_POST['email']) echo "oculto "; ?>error'>La dirección de email no es
válida.</span>
                </div>

                <div class='campo'>
                    <input type='submit' name='enviar' value='Enviar' />
                </div>
            </fieldset>
        </form>
    </div>
    <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
    <script type="text/javascript" src="validar.js"></script>
</body>
</html>

```

validar.php

```

<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 7 : Aplicaciones web dinámicas: PHP y Javascript -->
<!-- Ejemplo Validación formulario: validar.php -->
<?php
function validarNombre($nombre){
    if(strlen($nombre) < 4) return false;
    return true;
}

function validarEmail($email){
    return preg_match("/^[a-z0-9]+([_\\.\-][a-z0-9]+)*@([a-z0-9]+([_\\.\-][a-z0-9]+)*)+\\.([a-z]{2,})$/i", $email);
}

function validarPasswords($pass1, $pass2) {
    return $pass1 == $pass2 && strlen($pass1) > 5;
}

```

```
?>
```

validar.js

```
$(document).ready(function() {  
    function validarNombre() {  
        if(nombre.val().length < 4) {  
            errorNombre.removeClass("oculto");  
            return false;  
        }  
        errorNombre.addClass("oculto");  
        return true;  
    }  
  
    function validarEmail() {  
        if(!email.val().match("^[a-zA-Z0-9]+[a-zA-Z0-9_-]+@[a-zA-Z0-9]+[a-zA-Z0-9.-]+[a-zA-Z0-9]+.[a-z]{2,4}$")) {  
            errorEmail.removeClass("oculto");  
            return false;  
        }  
        errorEmail.addClass("oculto");  
        return true;  
    }  
  
    function validarPasswords() {  
        if(password1.val().length < 6 || password1.val() != password2.val()) {  
            errorPassword.removeClass("oculto");  
            return false;  
        }  
        errorPassword.addClass("oculto");  
        return true;  
    }  
  
    function validar() {  
        return validarNombre() & validarEmail() & validarPasswords();  
    }  
  
    var nombre = $("#nombre");  
    var password1 = $("#password1");  
    var password2 = $("#password2");  
    var email = $("#email");  
    var errorNombre = $("#errorNombre");  
    var errorPassword = $("#errorPassword");  
    var errorEmail = $("#errorEmail");  
  
    $("#datos").submit(function() {  
        if(validar()) return true;  
        else return false;  
    });  
});
```

3.- Utilización de AJAX con PHP.

Caso práctico

En los últimos días **Juan** y **Carlos** han logrado grandes avances. Tienen claras las capacidades que les ofrece la programación del cliente web. Han profundizado en la tecnología AJAX. Y saben, incluso, de qué manera pueden integrar en una misma aplicación ambos tipos de programación: la programación del servidor web en lenguaje PHP, y la programación del cliente web en lenguaje JavaScript.

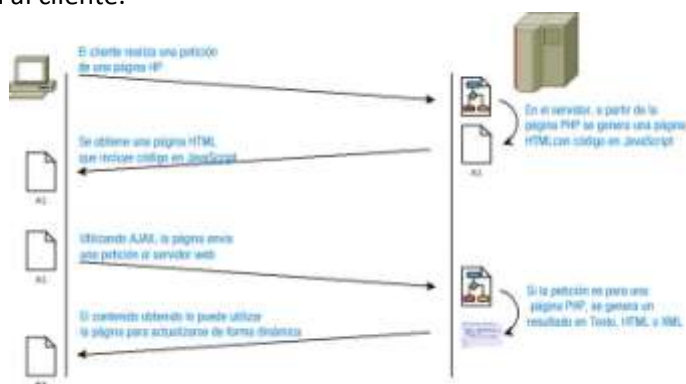
Sólo les falta un detalle: las herramientas. Saben lo que tienen que hacer, pero antes de ponerse manos a la obra, deben decidir de qué forma hacerlo. **María** les ha informado de que existen multitud de librerías y herramientas que se pueden utilizar para agilizar la programación de aplicaciones en lenguaje JavaScript. Su último paso será decidir qué librerías y herramientas utilizarán para facilitar la programación de la aplicación web, tomando como punto de partida que sería preferible utilizar un único lenguaje en todo el proyecto para evitar la fragmentación del código de la misma.

Como ya sabes, la tecnología AJAX se utiliza desde el cliente web para permitir comunicaciones asíncronas con el servidor web, sin necesidad de recargar la página web que se muestra en el navegador. Se basa en la utilización de código en lenguaje JavaScript. Por tanto, te estarás preguntando, ¿qué tiene que ver la tecnología AJAX con el lenguaje PHP?

Acabamos de ver cómo se pueden crear aplicaciones web que utilicen de forma simultánea la programación del cliente web (con JavaScript) y la programación del servidor web (con PHP). En el procedimiento seguido en el ejemplo anterior, ambas tecnologías coexistían en una misma página, pero su programación era independiente. El código PHP se ejecutaba en el servidor, y en la misma página se incluían también guiones en lenguaje JavaScript que se ejecutan en el navegador. En nuestro ejemplo solamente existía una relación: si el navegador permitía la ejecución de código JavaScript, se deshabilitaba el envío del formulario y la validación se realizaba en el cliente. En este caso no se llegaba a enviar la página al servidor y no se ejecutaba por tanto el código PHP de validación.

Si vas a usar aplicaciones que utilicen ambos lenguajes, es preferible tener un mecanismo mejor para integrarlos. Afortunadamente existen librerías para PHP que te permiten aprovechar las capacidades de JavaScript utilizando casi exclusivamente código en lenguaje PHP. Estas librerías definen una serie de objetos que puedes utilizar en el código de servidor, y que generan de forma automática código JavaScript en las páginas web que se envían al cliente.

La mayoría de estas librerías añaden a las aplicaciones web funcionalidades de la tecnología AJAX. Esto es: permiten crear páginas PHP que, tras ejecutarse en el servidor, producen páginas web que incorporan código JavaScript con funcionalidades AJAX. El mecanismo de funcionamiento lo puedes observar en el siguiente diagrama.



Muchas de estas librerías suelen apoyarse en librerías JavaScript como jQuery para la ejecución de código en el cliente. En Internet puedes encontrar información sobre librerías PHP con soporte para AJAX.

Librerías PHP con soporte para AJAX

http://ajaxpatterns.org/PHP_Ajax_Frameworks

A continuación vas a aprender a utilizar dos de estas librerías: **Xajax** y **jQuery4PHP**.

¿Qué es jQuery?



Una librería de programación para PHP.

Una librería de programación para JavaScript.

La librería para PHP que nos permite utilizar la funcionalidad de jQuery se llama jQuery4PHP.

3.1.- Xajax.

Xajax es una librería PHP de código abierto que permite generar aplicaciones web con tecnología AJAX. Facilita la utilización desde el cliente de funciones existentes en el servidor. Al utilizar los objetos AJAX en el código PHP, las páginas HTML que se obtienen incorporan el código JavaScript necesario para realizar las llamadas al servidor mediante AJAX. En la página web del proyecto tienes información disponible sobre su utilización.

Página web del proyecto

<http://www.xajax-project.org/en/home/>

Para poder utilizar AJAX, descárgate la última versión de la librería desde la sección de descargas de su página web. De las carpetas que contienen los ficheros comprimidos, necesitas el contenido de **xajax_core** y **xajax_js**. Cópialas a una ruta de tu servidor web en la que sean accesibles por tus aplicaciones web.

Sección de descargas

<http://www.xajax-project.org/en/download/>

En las **páginas PHP en que quieras utilizar AJAX**, deberás incluir la librería escribiendo el siguiente código:

```
require_once("xajax_core/xajax.inc.php");
```

Asegúrate de que la ruta a la librería sea la correcta. Lo siguiente es crear un objeto de la clase **xajax**, indicando como parámetro el script PHP que contiene las funciones a las que se podrán realizar llamadas mediante AJAX. Puedes incluir estas funciones en una página aparte o en la misma página PHP, y en este caso no será necesario indicar ningún parámetro:

Clase xajax

<http://www.xajax-project.org/en/docs-tutorials/api-docs/>

```
$xajax = new xajax();
```

AJAX necesita incluir en la página web que se envía al navegador su propio código JavaScript. Para ello, tienes que incluir en tu código PHP la siguiente llamada al método **printJavaScript** del objeto **\$xajax**:

```
$xajax->printJavaScript();
```

En caso necesario, también deberás configurar la ruta de acceso a la carpeta **xajax_js** que contiene el código JavaScript de la librería (usa tu propia ruta como segundo parámetro):

```
$xajax->configure('javascript URI','./libs/');
```

Existen otras opciones de configuración de Xajax. Por ejemplo, cuando las cosas no funcionan como deberían, es muy interesante la opción que activa los mensajes de depuración:

```
$xajax->configure('debug',true);
```



Y por último, tienes que utilizar el método **register** para registrar cada una de las funciones PHP del servidor que estarán disponibles para ser ejecutadas de forma asíncrona desde el navegador:

```
$xajax->register(XAJAX_FUNCTION,"funcion1");
$xajax->register(XAJAX_FUNCTION,"funcion2");
...
```

En el **guión PHP en que defines las funciones** (si no es la misma página que la anterior), tienes que incluir la librería, crear el objeto y registrar las funciones de la misma forma que hiciste antes:

```
require_once("xajax_core/xajax.inc.php");
$xajax = new xajax();
$xajax->register(XAJAX_FUNCTION,"funcion1");
$xajax->register(XAJAX_FUNCTION,"funcion2");
```

Al registrar una función web, crea automáticamente una función JavaScript en el documento HTML con su mismo nombre prefijado por "**xajax_**". En el caso anterior, se crearán las funciones **xajax_funcion1** y **xajax_funcion2**.

Además deberás utilizar el método **processRequest**, que es el encargado de procesar las llamadas que reciba la página.

```
$xajax->processRequest();
```

Es importante tener en cuenta que la llamada a **processRequest** debe realizarse antes de que el guión PHP genere ningún tipo de salida.

En cada una de las funciones que defines, podrás instanciar y utilizar un objeto de la clase **xajaxResponse** para devolver al navegador los comandos resultado del procesamiento:

Clase xajaxResponse

<http://www.xajax-project.org/en/docs-tutorials/api-docs/xajax-core/xajaxresponse-inc-php/xajaxresponse/>

```
function funcion1($a){
    $respuesta = new xajaxResponse();
    ...
    return $respuesta;
}
```

3.2.- Xajax (II).

Vamos a ver un ejemplo de programación que utilice la librería xajax. Partiendo del formulario web que utilizaste anteriormente, veremos el código xajax necesario para utilizar AJAX en su validación. La idea es crear una función PHP, que llamaremos **validarFormulario**, que reciba los datos del formulario y realice su validación. La página web, al pulsar el botón de enviar del formulario, utilizará AJAX para llamar a esta función y mostrar los errores de validación que devuelva.

En este ejemplo, todo el código va a compartir la misma página, incluyendo las funciones que se ejecutarán mediante AJAX. El primer paso es incluir la librería xajax, y definir la función de validación, que aprovechará el código de validación PHP que creaste antes:

```
require_once("xajax_core/xajax.inc.php");

function validarFormulario($valores) {
    $respuesta = new xajaxResponse();
    $error = false;

    if (!validarNombre($valores['nombre'])) {
        $respuesta->assign("errorNombre", "innerHTML",
        "El nombre debe tener más de 3 caracteres.");
        $error = true;
    }
    else $respuesta->clear("errorNombre", "innerHTML");

    // Validamos también las contraseñas y el email
    ...
}
```

```

if (!$error) $respuesta->alert("Todo correcto.");

$respuesta->assign("enviar","value","Enviar");
$respuesta->assign("enviar","disabled",false);
return $respuesta;
}

```

Como ves, si el nombre no valida, se utiliza el método `assign` de `xajaxResponse` para asignar al elemento de `id=errorNombre` (un `span` que ya está creado) el mensaje de error correspondiente. Si el nombre es válido, se vacía el contenido de ese elemento utilizando `clear` (por si se estuviera mostrando algún mensaje de error anterior). Este procedimiento tendrás que realizarlo para los tres elementos que debes validar en el formulario.

Si no se produce ningún error de validación, se muestra un mensaje de información de que todo está correcto. Además, se utiliza `assign` para volver a habilitar el botón de envío del formulario, y restaurar su etiqueta a `Enviar` (ahora vemos en qué otro lugar se cambia).

Como ya viste anteriormente, antes de crear el contenido HTML de la página, deberás incluir las siguientes sentencias PHP:

```

$xajax = new xajax();
$xajax->register(XAJAX_FUNCTION,"validarFormulario");
$xajax->processRequest();

```

¿En qué páginas debes crear un objeto de la clase xajax?



En las páginas que vayan a utilizar AJAX para realizar peticiones a otras páginas PHP del servidor



En las páginas que vayan a utilizar AJAX para realizar peticiones a otras páginas PHP del servidor, y en las páginas del servidor que vayan a recibir dichas peticiones.

Ambas páginas necesitan instanciar un objeto de la clase xajax.

3.3.- Xajax (III).

Para ejecutar el código de validación cuando se envíe el formulario deberás crear una función JavaScript y asignarla al evento `onsubmit` del formulario.

```

<form id='datos' action="javascript:void(null);" onsubmit="enviarFormulario();">

```

La función JavaScript `enviarFormulario` utilizará la función `xajax_validarFormulario` que ha sido creada automáticamente por Xajax al registrar la función correspondiente. Además se encargará de deshabilitar el botón de envío y de cambiar el texto que muestra:

```

xajax.$('enviar').disabled=true;
xajax.$('enviar').value="Un momento...";
xajax_validarFormulario (xajax.getFormValues("datos"));

```

Este método de llamar a una función registrada se conoce como asíncrono. Una vez realizada la llamada a la función, el procesamiento del formulario continúa sin esperar a recibir una respuesta. Es por este motivo que se deshabilita el botón de enviar el formulario. Cuando se recibe la respuesta de la función, se producen en el formulario los cambios que se indican en la misma.

Existe en Xajax otro método de realizar llamadas síncronas a una función registrada: el método JavaScript `xajax.request`:

```

respuesta = xajax.request({xjxfun:"validarFormulario"}, {mode:'synchronous', parameters:
["valor1", "valor2", ...]})

```

La función PHP a la que se realiza la llamada, debe recibir tantos parámetros como se pasan. Para indicar el valor que se devuelve, puede usar el método `setReturnValue()` de la clase `xajaxResponse`.

```
$respuesta = new xajaxResponse();
...
$respuesta->setReturnValue("valorDevuelto");
return $respuesta;
```

Fíjate que para realizar estas acciones se utiliza el objeto JavaScript xajax. Su método `getFormValues` se encarga de obtener los datos de un formulario.

Clase JavaScript xajax

<http://www.xajax-project.org/en/docs-tutorials/api-docs/xajax-js/xajax-core-uncompressed-js/xajax/>

Para acceder a los elementos de una página HTML mediante JavaScript mediante su atributo id, se puede usar:

```
objUsuario = document.getElementById("usuario");
```

Y si quieres conocer su valor:

```
valorUsuario = objUsuario.value;
```

Si utilizas Firebug puedes comprobar que los parámetros se envían mediante POST.



Es importante que tengas clara la estructura en clases de la librería Xajax, y sepas diferenciar entre las clases PHP y las clases JavaScript de la misma. Las principales clases son las que has visto en el ejemplo anterior: `xajax` y `xajaxResponse` en cuanto a clases PHP, y `xajax` en cuanto a JavaScript.

3.4.- Xajax (IV).

Esta función, que se encargará de validar el formulario, puedes crearla en el mismo fichero PHP o en un fichero aparte. Deberás incluir también el código necesario para Xajax:

```
<head>
...
<?php $xajax->printJavascript(); ?>
<script type="text/javascript" src="validar.js"></script>
</head>
```

Puedes comprobar el código completo de la página, incluyendo el código JavaScript y los estilos:

form.php

```
<?php
/**
 * Desarrollo Web en Entorno Servidor
 * Tema 7 : Aplicaciones web dinámicas: PHP y Javascript
 * Ejemplo Validación formulario con Xajax: form.php
 */

// Incluimos la librería Xajax
require_once("xajax_core/xajax.inc.php");

// Creamos las funciones de validación, que van a ser llamadas
// desde JavaScript

function validarNombre($nombre){
    if(strlen($nombre) < 4) return false;
    return true;
}

function validarEmail($email){
```

```

        return      ereg('^[a-zA-Z0-9]+[a-zA-Z0-9_-]+@[a-zA-Z0-9]+[a-zA-Z0-9.-]+[a-zA-Z0-9]+\.[a-
z]{2,4}$', $email);
    }

    function validarPasswords($pass1, $pass2) {
        return $pass1 == $pass2 && strlen($pass1) > 5;
    }

    function validarFormulario($valores) {
        $respuesta = new xajaxResponse();
        $error = false;

        if (!validarNombre($valores['nombre'])) {
            $respuesta->assign("errorNombre", "innerHTML", "El nombre debe tener más de 3
caracteres.");
            $error = true;
        }
        else $respuesta->clear("errorNombre", "innerHTML");

        if (!validarPasswords($valores['password1'], $valores['password2'])) {
            $respuesta->assign("errorPassword", "innerHTML", "La contraseña debe ser mayor de 5
caracteres o no coinciden.");
            $error = true;
        }
        else $respuesta->clear("errorPassword", "innerHTML");

        if (!validarEmail($valores['email'])) {
            $respuesta->assign("errorEmail", "innerHTML", "La dirección de email no es válida.");
            $error = true;
        }
        else $respuesta->clear("errorEmail", "innerHTML");

        if (!$error) $respuesta->alert("Todo correcto.");

        $respuesta->assign("enviar","value","Enviar");
        $respuesta->assign("enviar","disabled",false);

        return $respuesta;
    }

    // Creamos el objeto xajax
    $xajax = new xajax();

    // Registramos la función que vamos a llamar desde JavaScript
    $xajax->register(XAJAX_FUNCTION,"validarFormulario");
    // Y configuramos la ruta en que se encuentra la carpeta xajax_js
    $xajax->configure('javascript URI','./');

    // El método processRequest procesa las peticiones que llegan a la página
    // Debe ser llamado antes del código HTML
    $xajax->processRequest();
    ?>
<!DOCTYPE      html      PUBLIC      "-//W3C//DTD      HTML      4.01      Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Ejemplo Tema 7: Validación formulario con Xajax</title>
    <link rel="stylesheet" href="estilos.css" type="text/css" />
    <?php
    // Le indicamos a Xajax que incluya el código JavaScript necesario
    $xajax->printJavascript();
    ?>
    <script type="text/javascript" src="validar.js"></script>
</head>

<body>
    <div id='form'>
        <!-- Cuando se vaya a enviar el formulario ejecutamos
        una función en JavaScript, que realiza la llamada a PHP -->
        <form id='datos' action="javascript:void(null);" onsubmit="enviarFormulario();">
        <fieldset >
            <legend>Introducción de datos</legend>
            <div class='campo'>
                <label for='nombre' >Nombre:</label><br />
                <input type='text' name='nombre' id='nombre' maxlength="50" /><br />
                <span id="errorNombre" class="error" for="nombre"></span>
            </div>

```

```
<div class='campo'>
  <label for='password1' >Contraseña:</label><br />
  <input type='password' name='password1' id='password1' maxlength="50" />
  <span id="errorPassword" class="error" for="password"></span>
</div>
<div class='campo'>
  <label for='password2' >Repita la contraseña:</label><br />
  <input type='password' name='password2' id='password2' maxlength="50" />
</div>
<div class='campo'>
  <label for='email' >Email:</label><br />
  <input type='text' name='email' id='email' maxlength="50" />
  <span id="errorEmail" class="error" for="email"></span>
</div>

<div class='campo'>
  <input type='submit' id='enviar' name='enviar' value='Enviar' />
</div>
</fieldset>
</form>
</div>
</body>
</html>
```

validar.js

```
function enviarFormulario() {
  // Se cambia el botón de Enviar y se deshabilita
  // hasta que llegue la respuesta
  xajax.$('enviar').disabled=true;
  xajax.$('enviar').value="Un momento...";

  // Aquí se hace la llamada a la función registrada de PHP
  xajax_validarFormulario (xajax.getFormValues("datos"));

  return false;
}
```

estilos.css

```
#form fieldset {
  position: absolute;
  left: 50%;
  top: 50%;
  width: 310px;
  margin-left: -170px;
  height: 330px;
  margin-top: -150px;
  padding:10px;
  border:1px solid #ccc;
  background-color: #eee;
}

legend, h3 {
  font-family : Arial, sans-serif;
  font-size: 1.3em;
  font-weight:bold;
  color:#333;
}

#form .campo {
  margin-top:8px;
  margin-bottom: 10px;
  margin-left: 10px;
}

#form label {
  font-family : Arial, sans-serif;
  font-size:0.8em;
  font-weight: bold;
}

#form input[type="text"], #form input[type="password"] {
  font-family : Arial, Verdana, sans-serif;
  font-size: 0.8em;
  line-height:140%;
  color : #000;
  padding : 3px;
```

```

border : 1px solid #999;
height:20px;
width:280px;
}

#form input[type="submit"] {
width:100px;
height:30px;
padding-left:0px;
}

#form .error{
font-size:10px;
text-decoration: underline;
background: #ffdddd;
color: #ee2211;
}

.oculto {
display: none;
}

```

Asegúrate de incluir la librería y ajustar las rutas en el código. Si tienes problemas, puedes utilizar Firebug para comprobar que los cambios que realiza Xajax en el HTML son los adecuados.



Utilizando Xajax, ¿cómo debes hacer para que la página web espere por la respuesta de una petición AJAX?



Llamando a la función JavaScript que crea Xajax cuando registras una función del servidor (su nombre comienza por xajax_).



Mediante el método request, indicando como parámetro mode:'synchronous'.

La clase JavaScript xajax implementa este método que permite detener la ejecución de código en el navegador (modo síncrono) hasta que se resuelva la petición AJAX.

Partiendo de la aplicación de tienda online que programaste en unidades anteriores, utiliza Xajax para cambiar el mecanismo de login. Se trata de crear una función en PHP de nombre `validarLogin`, que reciba como parámetros un nombre y contraseña, y que compruebe estas credenciales con la base de datos y devuelva `false` si no son correctas o `true` si son válidas. En este caso, deberá encargarse, también, de almacenar el nombre del usuario en una variable de sesión.

login.php

```

<?php
require_once('include/DB.php');

// Comprobamos si ya se ha enviado el formulario
if (isset($_POST['enviar'])) {

    if (empty($_POST['usuario']) || empty($_POST['password']))
        $error = "Debes introducir un nombre de usuario y una contraseña";
    else {
        // Comprobamos las credenciales con la base de datos
        if (DB::verificaCliente($_POST['usuario'], $_POST['password'])) {
            session_start();
            $_SESSION['usuario']=$_POST['usuario'];

```

```

        header("Location: productos.php");
    }
    else {
        // Si las credenciales no son válidas, se vuelven a pedir
        $error = "Usuario o contraseña no válidos!";
    }
}
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 7 : Aplicaciones web dinámicas: PHP y Javascript -->
<!-- Ejercicio: Formulario de Login con Xajax: login.php -->
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Ejemplo Tema 5: Login Tienda Web</title>
    <link href="tienda.css" rel="stylesheet" type="text/css">
</head>

<body>
    <div id='login'>
        <form action='login.php' method='post'>
            <fieldset >
                <legend>Login</legend>
                <div><span class='error'><?php echo $error; ?></span></div>
                <div class='campo'>
                    <label for='usuario' >Usuario:</label><br/>
                    <input type='text' name='usuario' id='usuario' maxlength="50" /><br/>
                </div>
                <div class='campo'>
                    <label for='password' >Contraseña:</label><br/>
                    <input type='password' name='password' id='password' maxlength="50" /><br/>
                </div>

                <div class='campo'>
                    <input type='submit' name='enviar' value='Enviar' />
                </div>
            </fieldset>
        </form>
    </div>
</body>
</html>

```

Modifica la página **login.php** para que, utilizando Xajax, haga una llamada a la función `validarLogin` cuando se pulsa el botón `Enviar` del formulario. Si la función devuelve `false`, mostrará un mensaje que indique que las credenciales son incorrectas. En caso contrario, cargará la página `productos.php`. Deberás utilizar `xajax.request` para realizar una llamada síncrona a la función. Revisa el código que se propone como solución al ejercicio.

login.php

```

<?php
/**
 * Desarrollo Web en Entorno Servidor
 * Tema 7 : Aplicaciones web dinámicas: PHP y Javascript
 * Ejemplo Validación formulario con Xajax: form.php
 */

// Incluimos la librería Xajax
require_once('include/xajax_core/xajax.inc.php');

// Creamos el objeto xajax
$xajax = new xajax('include/valida.php');

// Registramos la función que vamos a llamar desde JavaScript
$xajax->register(XAJAX_FUNCTION, "validarLogin");

// Y configuramos la ruta en que se encuentra la carpeta xajax_js
$xajax->configure('javascript URI', './include/');
?>

```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 7 : Aplicaciones web dinámicas: PHP y Javascript -->
<!-- Ejercicio: Formulario de Login con Xajax: login.php -->
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <title>Ejemplo Tema 7: Login Tienda Web utilizando Xajax</title>
  <link href="tienda.css" rel="stylesheet" type="text/css">
  <!-- Incluimos el código JavaScript necesario -->
  <?php $xajax->printJavascript(); ?>
  <script type="text/javascript" src="validar.js"></script>
</head>

<body>
  <div id='login'>
    <!-- Cuando se vaya a enviar el formulario ejecutamos
          una función en JavaScript, que realiza la llamada a PHP -->
    <form id='datos' action='productos.php' method='post' onsubmit='return
enviarFormulario();'>
      <fieldset>
        <legend>Login</legend>
        <div><span class='error'><?php echo $error; ?></span></div>
        <div class='campo'>
          <label for='usuario' >Usuario:</label><br/>
          <input type='text' name='usuario' id='usuario' maxlength="50" /><br/>
        </div>
        <div class='campo'>
          <label for='password' >Contraseña:</label><br/>
          <input type='password' name='password' id='password' maxlength="50" /><br/>
        </div>

        <div class='campo'>
          <input type='submit' name='enviar' value='Enviar' />
        </div>
      </fieldset>
    </form>
  </div>
</body>
</html>

```

valida.php

```

<?php
/**
 * Desarrollo Web en Entorno Servidor
 * Tema 7 : Aplicaciones web dinámicas: PHP y Javascript
 * Ejercicio: Formulario de Login con Xajax: verifica.php
 */

// Incluimos la librería Xajax
require_once('xajax_core/xajax.inc.php');
require_once('DB.php');

// Creamos el objeto xajax
$xajax = new xajax();

// Registramos la función que vamos a llamar desde JavaScript
$xajax->register(XAJAX_FUNCTION,"validarLogin");

// El método processRequest procesa las peticiones que llegan a la página
// Debe ser llamado antes del código HTML
$xajax->processRequest();

// Validamos el nombre y contraseña enviados
function validarLogin($usuario, $password) {
  $respuesta = new xajaxResponse();

  if (empty($usuario) || empty($password))
    // $error = "Debes introducir un nombre de usuario y una contraseña";
    $respuesta->setReturnValue(false);
  else {
    // Comprobamos las credenciales con la base de datos
    if (DB::verificaCliente($usuario, $password)) {
      session_start();
      $_SESSION['usuario']=$usuario;
      $respuesta->setReturnValue(true);
    }
  }
}

```

```
    }
    else {
        // Si las credenciales no son válidas
        $respuesta->setReturnValue(false);
    }
}

return $respuesta;
}

?>
```

validar.js

```
function enviarFormulario() {
    var usuario = document.getElementById("usuario").value;
    var password = document.getElementById("password").value;

    // Aquí se hace la llamada a la función registrada de PHP
    var respuesta = xajax.request({xjxfun:"validarLogin"}, {mode:'synchronous', parameters:
[usuario, password]});
    if (respuesta==false) alert("Nombre de usuario y/o contraseña no válidos.");
    return respuesta;
}
```

3.5.- JQuery4PHP.



La otra librería con la que vas a trabajar se llama JQuery4PHP. Su objetivo es proporcionar un interface de programación en PHP que aproveche las capacidades de la librería de JavaScript jQuery. Es de código abierto, disponible bajo licencias MIT y GPLv2, y funciona únicamente con las últimas versiones de PHP (a partir de PHP5).

JQuery4PHP

<http://jquery4php.sourceforge.net/>

Al igual que con XAJAX, para poder utilizarla en tus páginas simplemente descárgate la última versión desde la página de descargas, y extrae del fichero comprimido la carpeta **YepSua**. Cópiala en un lugar accesible por el servidor web y tus aplicaciones.

Página de descargas

<http://sourceforge.net/projects/jquery4php/files/>

En las **páginas PHP en que quieras utilizar JQuery4PHP**, deberás incluir la librería escribiendo el siguiente código:

```
<?php
require_once ("YepSua/Labs/RIA/jquery4PHP/YsJQueryAutoloader.php");
YsJQueryAutoloader::register();
?>
```

Asegúrate de que la ruta a la librería sea la correcta.

El código anterior ejecuta un método estático de la clase **YsJQueryAutoloader** que se encarga de incluir todos los ficheros necesarios para la librería en el código de tu página.

Para acceder a las capacidades de JQuery4PHP, deberás utilizar en tu código PHP la clase **YsJQuery**.

Antes viste que con XAJAX necesitabas ejecutar en el servidor el método **printJavascript**, para que incluyese su propio código JavaScript en la página que se envía al cliente. La librería JQuery4PHP se apoya completamente en el código JavaScript de la librería jQuery. Por tanto, al escribir el código HTML de la página deberás asegurarte de que se incluye dicha librería, poniendo una línea como la siguiente:

```
<head>
// Utilizamos la versión de jQuery disponible en las CDN de Google
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
...
</head>
```

En la documentación de la librería, y en los ejemplos que vas a programar a continuación, se utiliza una sintaxis peculiar de programación propia de PHP. Por ejemplo, comprueba el siguiente código:

```
<?php
$jq = YsJQuery::newInstance();
$jq->onClick();
$jq->in('#enviar');
$jq->execute('alert("Has pulsado el botón.")');
$jq->write();
?>
```

Como ves, se crea un nuevo objeto a partir del método estático `newInstance`, y a continuación se ejecutan ciertos métodos implementados en la clase `YsJQuery`. Estos métodos definen código JavaScript asociado al evento `onClick` del botón con id `enviar`. El código que se ejecutará cuando se pulse el botón se incluye dentro del método `execute`, y muestra un mensaje en pantalla. El último método (`write`) es el encargado de generar el código JavaScript en la página.

Ese mismo código se puede escribir de la forma siguiente, que es la que utilizaremos a continuación.

```
<?php
echo
YsJQuery::newInstance()
->onClick()
->in('#enviar')
->execute('alert("Has pulsado el botón.")')
?>
```

Fíjate que no se asigna nombre al objeto que se crea, pues no es necesario nombrarlo si los métodos se ejecutan justo a continuación de su instanciación. Además, se ha sustituido la llamada al método `write` por un comando `echo` al comienzo.

3.6.- JQuery4PHP (II).

Para comenzar a ver la librería, utilizaremos una versión simplificada del formulario de introducción de datos con el que has estado trabajando. El objetivo es el mismo, validar los datos introducidos por el usuario; pero en lugar de mostrar los errores de validación integrados en la página web, vas a emplear la función `alert` de JavaScript.



La página `form.php` es similar a la que has utilizado anteriormente, añadiéndole código para:

- ✓ Incluir y registrar la librería:

```
require_once("YepSua/Labs/RIA/jQuery4PHP/YsJQueryAutoloader.php");
YsJQueryAutoloader::register();
```

- ✓ Incluir también la librería de JavaScript jQuery:

```
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
```

Además, hay que capturar el evento `onClick` del formulario, e indicarle que envíe los datos del mismo a una página PHP de validación.

```
<?php
echo
```

```

YsjQuery::newInstance()
->onClick()
->in("#enviar")
->execute(
    YsjQuery::getJSON(
        "validar.php",
        YsjQuery::toArray()->in('#datos input'),
        new YsJsFunction('
            if(msg.errorNombre) alert(msg.errorNombre);
            if(msg.errorPassword) alert(msg.errorPassword);
            if(msg.errorEmail) alert(msg.errorEmail);', 'msg'
        )
    )
);
?>

```

En el código anterior, primero se asocia al evento `onClick` del botón `enviar` al código que se encuentra en la llamada a `execute`. Este código llama al guión `validar.php`, enviando como parámetros los datos del formulario (los que se han introducido en etiquetas de tipo `input`) convertidos a array y, una vez recibida la respuesta, ejecuta una función JavaScript. En esta función se usa la función `alert` de JavaScript para mostrar los errores de validación obtenidos.

Fíjate que para comunicarse con el servidor, se utiliza el método `getJSON`. Este método utiliza notación JSON (*formato de intercambio de información más sencillo de procesar que XML (especialmente al utilizar el lenguaje JavaScript)*) para transmitir la información con el servidor.

¿Cuál es la función del método `write` de la clase `YsjQuery`?



Generar el código JavaScript necesario para que la página lleve a cabo las funciones que se han definido.



Mostrar un texto en la página web generada, de forma similar al `echo` o `print` de PHP.

Recuerda que no es necesario llamar a este método si empleas una construcción comenzando por `echo`, como en los ejemplos anteriores.

3.7.- JQuery4PHP (III).

Al ejecutar la página PHP que has programado, puedes observar que se genera el siguiente código JavaScript en la misma cuando se envía al navegador:

```

<script type="text/javascript" language="javascript">
/*  */
jQuery('#enviar').click(function(){
    jQuery.getJSON('validar.php',
        jQuery('#datos input').toArray(),
        function(msg){
            if(msg.errorNombre) alert(msg.errorNombre);
            if(msg.errorPassword) alert(msg.errorPassword);
            if(msg.errorEmail) alert(msg.errorEmail);
        }
    )
})
/* ]]&gt; */
&lt;/script&gt;
</pre>
</div>
<div data-bbox="91 744 862 776" data-label="Text">
<p>Como ves, todo el código creado utilizando la librería <code>JQuery4PHP</code> se traduce en llamadas a la librería <code>jQuery</code> de JavaScript.</p>
</div>
<div data-bbox="91 792 862 824" data-label="Text">
<p>En la página PHP de validación, <code>validar.php</code>, puedes utilizar las mismas funciones de ejemplos anteriores y definir una nueva de nombre <code>validarFormulario</code>:</p>
</div>
<div data-bbox="91 824 506 910" data-label="Text">
<pre>
function validarFormulario($valores) {
    $respuesta = array();
    if (!validarNombre($valores['nombre']))
        $respuesta['errorNombre'] =
            "El nombre debe tener más de 3 caracteres.";

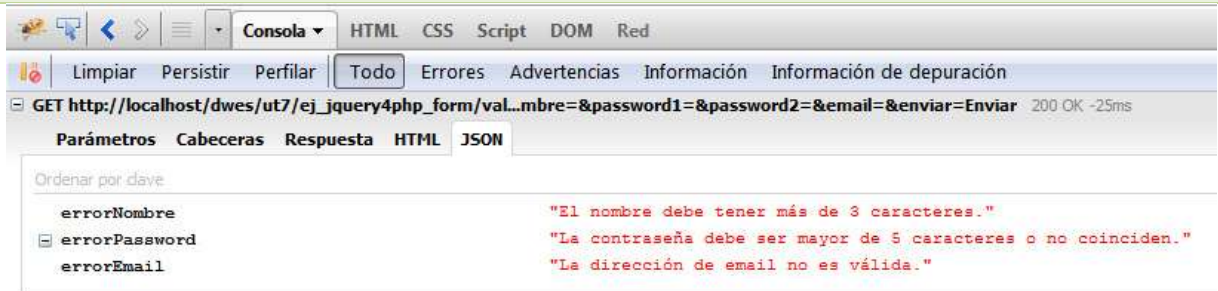
    // Validamos también las contraseñas y el email
    ...
}
</pre>
</div>
<div data-bbox="91 938 135 955" data-label="Page-Footer">- 28 -</div>
```

```

    return $respuesta;
}

echo json_encode(validarFormulario($_REQUEST));

```



Utilizaremos la función `json_encode` de PHP para devolver los errores de validación con notación JSON. Revisa el código obtenido y comprueba su funcionamiento.

form.php

```

<?php
/**
 * Desarrollo Web en Entorno Servidor
 * Tema 7 : Aplicaciones web dinámicas: PHP y Javascript
 * Ejemplo Validación formulario con jQuery4PHP: form.php
 */

// Incluimos la librería jQuery4PHP
include_once('../lib/YepSua/Labs/RIA/jquery4PHP/YsjQueryAutoloader.php');
YsjQueryAutoloader::register();
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <title>Ejemplo Tema 7: Validación formulario con jQuery4PHP</title>
  <link rel="stylesheet" href="estilos.css" type="text/css" />
  <!-- Incluimos la librería de JavaScript jQuery -->
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js" type="text/javascript"></script>
</head>

<body>
  <div id='form'>
    <form id='datos' action="javascript:void(null);">
      <fieldset>
        <legend>Introducción de datos</legend>
        <div class='campo'>
          <label for='nombre'>Nombre:</label>
          <input type='text' name='nombre' id='nombre' maxlength="50" />
        </div>
        <div class='campo'>
          <label for='password1'>Contraseña:</label><br />
          <input type='password' name='password1' id='password1' maxlength="50" />
        </div>
        <div class='campo'>
          <label for='password2'>Repita la contraseña:</label><br />
          <input type='password' name='password2' id='password2' maxlength="50" />
        </div>
        <div class='campo'>
          <label for='email'>Email:</label><br />
          <input type='text' name='email' id='email' maxlength="50" />
        </div>

        <div class='campo'>
          <input type='submit' id='enviar' name='enviar' value='Enviar' />
        </div>
      </fieldset>
    </form>
  </div>

<?php
echo
YsjQuery::newInstance()

```

```

->onClick()
->in("#enviar")
->execute(
    YsJQuery::getJSON(
        "validar.php",
        YsJQuery::toArray()->in('#datos input'),
        new YsJsFunction('
            if(msg.errorNombre) alert(msg.errorNombre);
            if(msg.errorPassword) alert(msg.errorPassword);
            if(msg.errorEmail) alert(msg.errorEmail);', 'msg'
        )
    )
);
?>
</body>
</html>

```

validar.php

```

<?php
// Creamos las funciones de validación, que van a ser llamadas
// desde JavaScript

function validarNombre($nombre){
    if(strlen($nombre) < 4) return false;
    return true;
}

function validarEmail($email){
    return preg_match("/^[a-z0-9]+([_\\.-][a-z0-9]+)*@[a-z0-9]+(\\.-[a-z0-9]+)*+\\.([a-z]{2,})$/i", $email);
}

function validarPasswords($pass1, $pass2) {
    return $pass1 == $pass2 && strlen($pass1) > 5;
}

function validarFormulario($valores) {
    $respuesta = array();
    if (!validarNombre($valores['nombre']))
        $respuesta['errorNombre'] = "El nombre debe tener más de 3 caracteres.";

    if (!validarPasswords($valores['password1'], $valores['password2']))
        $respuesta['errorPassword'] = "La contraseña debe ser mayor de 5 caracteres o no coinciden.";

    if (!validarEmail($valores['email']))
        $respuesta['errorEmail'] = "La dirección de email no es válida.";

    return $respuesta;
}

echo json_encode(validarFormulario($_REQUEST));
?>

```

estilos.css

```

#form fieldset {
    position: absolute;
    left: 50%;
    top: 50%;
    width: 340px;
    margin-left: -170px;
    height: 300px;
    margin-top: -150px;
    padding: 10px;
    border: 1px solid #ccc;
    background-color: #eee;
}

legend, h3 {
    font-family : Arial, sans-serif;
    font-size: 1.3em;
    font-weight: bold;
    color: #333;
}

#form .campo {

```

```

margin-top:8px;
margin-bottom: 10px;
margin-left: 10px;
}

#form label {
font-family : Arial, sans-serif;
font-size:0.8em;
font-weight: bold;
}

#form input[type="text"], #form input[type="password"] {
font-family : Arial, Verdana, sans-serif;
font-size: 0.8em;
line-height:140%;
color : #000;
padding : 3px;
border : 1px solid #999;
height:20px;
width:280px;
}

#form input[type="submit"] {
width:100px;
height:30px;
padding-left:0px;
}

.oculto {
display:none;
}

```

3.8.- JQuery4PHP (IV).

Una de las características que cabe destacar de jQuery4PHP es su extensibilidad. Existen varias extensiones que se integran con la librería y permiten realizar de forma sencilla tareas adicionales a las que soporta el núcleo de la misma.

Por ejemplo, si en la página anterior quisiéramos integrar en etiquetas los mensajes de validación, el código necesario sería más complejo. De hecho, ya has visto que para mostrar los mensajes de alerta has tenido que utilizar código JavaScript mezclado con el código PHP.

Vamos a ver cómo podemos utilizar la **extensión JqValidate** para realizar de forma mucho más sencilla y eficaz la validación del formulario anterior.

Extensión JqValidate

<http://jquery4php.sourceforge.net/index.php?section=plugins&module=jqValidate&method=about>

Para poder usar esta extensión en tus páginas has de:

- ✓ Indicar a la librería jQuery4PHP que vas a usar la extensión:

```
YsJQuery::usePlugin(YsJQueryConstant::PLUGIN_JQVALIDATE);
```
- ✓ Incluir el código JavaScript necesario por la extensión, que en este caso concreto se corresponde con la **extensión Validate de jQuery**:

```
<script src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.8.1/jquery.validate.min.js" type="text/javascript"></script>
```
- ✓ Cargar los mensajes de validación en idioma español (de no hacerlo, se mostrarán en inglés):

```
echo YsJQueryAssets::loadScripts('jq4php-showcase/showcase/jquery4php-assets/js/plugins/bassistance/validate/localization/messages_es.js')->execute();
```

Como siempre, revisa las rutas del código anterior para ajustarlas a las de tu sistema.

3.9.- JQuery4PHP (V).

Al utilizar la extensión **JqValidate** para validar los datos introducidos en un formulario, el código de validación que se usará utiliza la extensión **jQuery.Validate**. No será necesario que programes los

algoritmos de validación, sino que indiques qué reglas se deberán aplicar a cada uno de los campos del formulario. Esto se hace utilizando el método `rules`.

Para indicar las reglas, se pasa como parámetro un array, que contendrá tantos elementos como campos a validar. Para cada uno de estos campos se crea un nuevo array, que contendrá las reglas de dicho campo. Cada una de las reglas se compone en base a los distintos métodos de validación que incorpora la extensión de JavaScript `jQuery.Validate`.

Métodos de validación

http://docs.jquery.com/Plugins/Validation#List_of_built-in_Validation_methods

El código PHP necesario para validar nuestro formulario será:

```
<?php
    echo
    YsJQuery::newInstance()
        ->onClick()
        ->in("#enviar")
        ->execute(
            YsJQValidate::build()->in('#datos')
            -> rules(array(
'nombre' => array('required' => true, 'minlength' => 4),
'email' => array('required' => true, 'email' => true),
'password1' => array('required' => true,
'minlength' => 6, 'equalTo' => '#password2')
)
)
);
?>
```

Fíjate que la asociación del código con el evento `onClick` del botón se sigue haciendo como antes.

Si revisas el código JavaScript que se genera en la página HTML, observarás lo siguiente:

```
<script type="text/javascript" language="javascript">
/*  */
jQuery('#enviar').click(function(){
jQuery('#datos').validate({"rules": {
"nombre": {"required": true,"minlength": 4},
"email": {"required": true,"email": true},
"password1": {"required": true,"minlength": 6,
"equalTo": "#password2"}}})
})
/* ]]&gt; */
&lt;/script&gt;</pre></div><div data-bbox="90 629 862 709" data-label="Text"><p>Como ves, todo se sigue traduciendo en llamadas a la librería jQuery de JavaScript. Además, una de las grandes ventajas de este método es que una vez definidas las reglas de validación en PHP, todo el código que se ejecuta para verificarlas es JavaScript. Si el navegador soporta la ejecución de código en lenguaje JavaScript, no es necesario establecer ningún tipo de tráfico de validación con el servidor web.</p></div><div data-bbox="90 725 540 742" data-label="Text"><p>Revisa el código obtenido y comprueba su funcionamiento.</p></div><div data-bbox="90 741 292 758" data-label="Text"><p><a href="#">Código obtenido</a> (2.00 KB)</p></div><div data-bbox="90 771 862 807" data-label="Text"><p><b>Al emplear la extensión JqValidate de jQuery4PHP, ¿qué código JavaScript deberás incluir en tus páginas?</b></p></div><div data-bbox="102 807 138 826" data-label="Image"><input checked="" type="radio"/></div><div data-bbox="151 807 649 825" data-label="Text"><p>El correspondiente a la librería jQuery y a su extensión Validate.</p></div><div data-bbox="102 828 138 847" data-label="Image"><input type="radio"/></div><div data-bbox="151 827 534 846" data-label="Text"><p>Únicamente el correspondiente a la librería jQuery</p></div><div data-bbox="98 846 473 861" data-label="Text"><p><i>JqValidate utiliza en la parte cliente la extensión Validate de jQuery.</i></p></div><div data-bbox="90 871 173 889" data-label="Section-Header"><h3>form.php</h3></div><div data-bbox="90 888 137 910" data-label="Text"><pre>&lt;?php
/**</pre></div><div data-bbox="90 937 135 955" data-label="Page-Footer"><p>- 32 -</p></div>
```

```

* Desarrollo Web en Entorno Servidor
* Tema 7 : Aplicaciones web dinámicas: PHP y Javascript
* Ejemplo Validación formulario con jQuery4PHP y JqValidate: form.php
*/

// Incluimos la librería jQuery4PHP
include_once('../lib/YepSua/Labs/RIA/jQuery4PHP/YsjQueryAutoloader.php');
YsjQueryAutoloader::register();
// Y el plugin de validación
YsjQuery::usePlugin(YsjQueryConstant::PLUGIN_JQVALIDATE);
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <title>Ejemplo Tema 7: Validación formulario con jQuery4PHP</title>
  <link rel="stylesheet" href="estilos.css" type="text/css" />
  <!-- Incluimos la librería de JavaScript jQuery -->
  <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
  <!-- Y también la de validación -->
  <script type="text/javascript"
src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.8.1/jquery.validate.min.js"></script>
</head>

<body>
<?php
// Cargamos los mensajes de validación en castellano
echo YsjQueryAssets::loadScripts('../jq4php-showcase/showcase/jquery4php-
assets/js/plugins/bassistance/validate/localization/messages_es.js')->execute();
?>
  <div id='form'>
  <form id='datos' action="javascript:void(null);">
  <fieldset >
    <legend>Introducción de datos</legend>
    <div class='campo'>
      <label for='nombre' >Nombre:</label><br />
      <input type='text' name='nombre' id='nombre' maxlength="50" />
    </div>
    <div class='campo'>
      <label for='password1' >Contraseña:</label><br />
      <input type='password' name='password1' id='password1' maxlength="50" />
    </div>
    <div class='campo'>
      <label for='password2' >Repita la contraseña:</label><br />
      <input type='password' name='password2' id='password2' maxlength="50" />
    </div>
    <div class='campo'>
      <label for='email' >Email:</label><br />
      <input type='text' name='email' id='email' maxlength="50" />
    </div>

    <div class='campo'>
      <input type='submit' id='enviar' name='enviar' value='Enviar' />
    </div>
  </fieldset>
</form>
</div>
<?php
echo
YsjQuery::newInstance()
->onClick()
->in("#enviar")
->execute(
  YsJQValidate::build()->in('#datos')
  ->_rules(
    array('nombre' => array('required' => true, 'minlength' => 4),
      'email' => array('required' => true, 'email' => true),
      'password1' => array('required' => true, 'minlength' => 6, 'equalTo' =>
'#password2')
    )
  );
?>
</body>
</html>

```

estilos.css

```
#form fieldset {
  position: absolute;
  left: 50%;
  top: 50%;
  width: 310px;
  margin-left: -170px;
  height: 320px;
  margin-top: -150px;
  padding: 10px;
  border: 1px solid #ccc;
  background-color: #eee;
}

legend, h3 {
  font-family : Arial, sans-serif;
  font-size: 1.3em;
  font-weight: bold;
  color: #333;
}

#form .campo {
  margin-top: 8px;
  margin-bottom: 10px;
  margin-left: 10px;
}

#form label {
  font-family : Arial, sans-serif;
  font-size: 0.8em;
  font-weight: bold;
}

#form input[type="text"], #form input[type="password"] {
  font-family : Arial, Verdana, sans-serif;
  font-size: 0.8em;
  line-height: 140%;
  color : #000;
  padding : 3px;
  border : 1px solid #999;
  height: 20px;
  width: 280px;
}

#form input[type="submit"] {
  width: 100px;
  height: 30px;
  padding-left: 0px;
}

#form .error{
  list-style: square;
  font-size: 10px;
  color: #e46c6d;
}
```

TEMA 8

CONTENIDO

1.- Reutilización de código e información.	2
2.- Características de las aplicaciones web híbridas.	4
3.- Utilización de repositorios de información.	5
3.1.- OAuth2.	6
3.2.- JSON y XML.	9
4.- Creación de aplicaciones web híbridas.	11
4.1.- Yahoo! PlaceFinder.	12
4.1.1- Yahoo! PlaceFinder (II).	13
4.2.- Google Geocoding.	14
4.3.- Aplicación web híbrida de geocodificación.	16
form.php	16
ubicar.js	18
estilos.css	19
4.4.- Google Directions.	20
4.5.- Google Tasks.	21
4.5.1- Google Tasks (II).	22
4.6.- Aplicación web híbrida de gestión de repartos.	23
4.6.1.- Aplicación web híbrida de gestión de repartos (II).	24
4.6.2.- Aplicación web híbrida de gestión de repartos (III).	24
4.6.3.- Aplicación web híbrida de gestión de repartos (IV).	25
repartos.php	26
ajaxmaps.php	29
código.js	30
estilos.css	32

Aplicaciones web híbridas.

Caso práctico

Después de semanas de trabajo, **Carlos** da por finalizado el desarrollo de la aplicación en la que ha estado trabajando. Ha tenido que reescribir el código en varias ocasiones, cambiar la apariencia de algunas pantallas del interfaz, y retocar el esquema de la base de datos que habían diseñado en un principio, pero finalmente parece que el trabajo ha dado sus frutos.

Habla con **Juan** y entre los dos revisan el resultado. **Juan** está muy contento con lo que ve, y le propone hablar con **Ada**, para mostrarle la aplicación web. Aunque la directora ha seguido el progreso de la misma, en gran parte ha sido en las conversaciones que ha mantenido con **Juan**, y hace ya tiempo que no le informan directamente sobre los últimos avances.

1.- Reutilización de código e información.

Caso práctico

Ada revisa la aplicación web y les propone retocar un par de detalles, sobre todo relativos a la apariencia. Con los años de experiencia que tiene en el desarrollo de aplicaciones, les ofrece también algunos consejos sobre la usabilidad de los interfaces, y le indica que es muy importante seguir algunas normas básicas que garanticen la accesibilidad de la aplicación.

Por último, tienen que hablar con su amigo **Esteban** para concretar los detalles de la implantación de la aplicación en las dependencias del cliente. Parece que están a punto de finalizar el proyecto.

La unidad 6 de este módulo tenía por título "Servicios Web". En ella aprendiste a crear y utilizar servicios web, empleando una arquitectura orientada a servicios (SOA). Los principales temas que practicaste en esa unidad son:

- ✓ A utilizar el protocolo SOAP para comunicarte con un servicio web. Con ayuda de la clase `SoapClient`, intercambiabas peticiones y respuestas en formato SOAP con un servicio web existente.
- ✓ A crear tu propio servicio web. Mediante la clase `SoapServer` podías publicar tus propias funciones para que fueran accesibles como servicio web mediante SOAP.
- ✓ A procesar los documentos WSDL de descripción de los servicios web, y a crear los documentos WSDL de descripción de tus propios servicios.

Los servicios web permiten a tus aplicaciones comunicarse con otras utilizando la web (el protocolo HTTP) como medio de transmisión. Sin embargo, los mecanismos que has utilizado hasta ahora no son la única forma de implementar y utilizar servicios web. Desde hace un tiempo han ido apareciendo servicios web que utilizan arquitecturas basadas en **REST**.

REST hace referencia a un conjunto de normas que se pueden utilizar para establecer mecanismos de comunicación con servicios web. Concretamente, un servicio web implementado mediante REST debería al menos:

- ✓ Utilizar una estructura de URIs para acceder a los recursos gestionables mediante el servicio web:

```
/articulo/KSTDTG332GBR  
/tienda/CENTRAL
```

- ✓ Usar los distintos métodos HTTP (El protocolo HTTP 1.1 define 9 métodos que puede usar el cliente para identificar el tipo de acción que desea realizar. El método más conocido y utilizado es GET, que indica que se quiere obtener una representación de un recurso del servidor. Los 8 métodos restantes son HEAD, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT y PATCH) para las peticiones. Por ejemplo, se podría utilizar el método HTTP `GET` para obtener información de un artículo:

```
GET /articulo/KSTDTG332GBR
```

Y el método HTTP `DELETE` para borrarlo:

```
DELETE /articulo/KSTDTG332GBR
```

- ✓ No almacenar información sobre el estado: todas las peticiones se tratarán de forma independiente y deben incluir toda la información necesaria para poder atenderla.

- ✓ Utilizar XML o JSON en sus comunicaciones (o incluso ambos).

REST no es un estándar, pero muchos servicios web actuales se implementan utilizando arquitecturas de tipo REST. Existen en Internet varios documentos sobre REST y ejemplos de su utilización que conviene revisar.

REST y Servicios Web.

<http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf>

Serie de artículos sobre REST.

<http://eamodeorubio.wordpress.com/category/webservices/>

En la presente unidad crearás aplicaciones que utilicen diversos servicios web.

2.- Características de las aplicaciones web híbridas.

Caso práctico

En poco tiempo, **Carlos** modifica los detalles que había apreciado **Ada**. Está contento con el resultado obtenido, pero con la experiencia que ha adquirido en programación web durante su desarrollo, sabe que habría algunos detalles que se podrían añadir a la aplicación y que influirían positivamente en la experiencia del usuario.

Ada le llama y le comenta que la próxima semana **Esteban** vendrá a BK Programación a conocer la aplicación. **Carlos** guarda una copia de la aplicación en su estado actual, y decide tomarse esa semana para intentar mejorarla en algunos aspectos. Hace una lista de los detalles que le podría añadir, y al revisarla se da cuenta de que otras aplicaciones que conoce ya los implementan; pero en muchos casos, no son desarrollos propios sino que se basan en servicios ofrecidos por terceros: mapas de Google, imágenes de Flickr, ... ¿Podrá aprovechar algún servicio existente e integrarlo en su propia aplicación?

Una aplicación web híbrida, también conocida por su nombre en inglés (*mashup*), se caracteriza por combinar datos y/o funcionalidades procedentes de diversos orígenes para formar un nuevo tipo de aplicación o servicio.

Los tipos de fuentes de información más habituales que se utilizan en una aplicación web híbrida son:

- ✓ Información proveniente de servicios web, disponible mediante diversos protocolos y estructurada utilizando formatos de intercambio como JSON o XML.

En ocasiones el proveedor del servicio ofrece también un interface de programación (API) para facilitar el acceso a los datos. Es el caso de las API de compañías como Google, Yahoo!, Flickr, Microsoft o Amazon.

En otras ocasiones los datos se ofrecen de forma pública utilizando protocolos de redifusión web (también conocido como *sindicación web* es una manera sencilla de poner ciertos contenidos de un sitio web a disposición de otros) como **RSS** o **Atom**, y puede ser necesario procesarlos para extraer la información necesaria.

Redifusión web.

http://es.wikipedia.org/wiki/Redifusi%C3%B3n_web

- ✓ Información generada y gestionada por el propietario de la aplicación web híbrida, como pueden ser datos internos de una empresa.

De forma menos habitual, podemos encontrarnos aplicaciones web que utilicen técnicas de ingeniería inversa para extraer los datos que se muestran en algunos sitios web, como puede ser el caso de los precios de los productos en las tiendas web. Estas técnicas se conocen por su nombre en inglés: **web scraping**.

Por ejemplo, podrías montar una aplicación web híbrida que utilice la API de Google Maps, e información de ubicación geográfica de las franquicias de una empresa para mostrar la localización de las tiendas en un mapa.

En esta unidad, vas a programar una aplicación web híbrida para la tienda web con la que has venido trabajando. En este caso se trata de facilitar la gestión de los envíos de las compras.

Las siglas REST hacen referencia a un estándar que se utiliza en la implementación de servicios web.



Verdadero.



Falso.

Aunque el enunciado pueda parecer correcto, REST no es un estándar sino un conjunto de normas.

3.- Utilización de repositorios de información.

Caso práctico

Carlos se informa sobre los servicios web que pueden serle útiles, y decide añadir a la aplicación una funcionalidad que no tiene: un servicio de gestión de envíos para los productos que se vendan. En realidad nadie ha solicitado esa funcionalidad, pero cree que si es capaz de programarla la empresa de **Esteban** la llegará a utilizar. En ocasiones los clientes no piden una característica simplemente porque desconocen que es posible realizarla. Y cuando comenzó este proyecto en particular, no había nadie en BK Programación con la experiencia suficiente como para orientar correctamente al cliente.

Se pone manos a la obra. Si en la semana que tiene le da tiempo a finalizarla, se la mostrará a **Esteban**. Y si no le da tiempo, echará mano de la versión anterior. De cualquier modo, no es tiempo perdido. Este proyecto le está sirviendo para adquirir experiencia que a buen seguro aprovechará en el futuro inmediato.

Cuando utilices servicios de terceros para desarrollar una aplicación web híbrida, deberás tener en cuenta que en ocasiones existen condiciones y límites al uso que puedes hacer del mismo.

La mayoría de las grandes compañías que proveen servicios web al usuario, como Google o Yahoo!, requieren un registro previo y ofrecen unas condiciones para su uso gratuito que dependen del servicio al que necesites acceder. Algunos de estos servicios ofrecen una versión adicional de pago con mejores condiciones.

Por ejemplo, actualmente Google ofrece los siguientes límites de acceso gratuito para los siguientes servicios:

- ✓ **Google Tasks:** 5.000 consultas diarias.
- ✓ **Google Maps:** 25.000 consultas diarias.
- ✓ **Google Custom Search:** 100 consultas diarias.

En muchas ocasiones, el proveedor del servicio (aunque también puede ser un tercero) ofrece además librerías que facilitan la utilización del servicio desde un lenguaje de programación determinado y ejemplos de utilización del mismo. Por ejemplo, si quieres utilizar la API de *Google Tasks* existen librerías de programación para los lenguajes Java, Python, PHP y para la plataforma .Net de Microsoft.

Para que se pueda verificar la utilización que hace cada usuario de un servicio determinado, es necesario incluir dentro del código que interactúa con el mismo una clave personal que le identifica en el sistema. Por ejemplo, si quieres utilizar la API de *Google Books*, necesitarás indicar tu código de desarrollador al hacer una consulta de forma similar a:

```
$client->setDeveloperKey('la clave de desarrollador va aquí');
```

De la misma forma, si quieres acceder al servicio de geocodificación (*Asignación de coordenadas geográficas como latitud / longitud a una dirección*) *PlaceFinder* de Yahoo!, tendrás que indicar en la consulta un identificador de aplicación.

Existen ciertos servicios web que nos permiten acceder a información privada que almacenan de sus usuarios. Por ejemplo, la API de *Google Calendar* posibilita gestionar la información que cada usuario mantiene en sus calendarios personales. Si vas a usar un servicio de este tipo (como *Google Tasks*), necesitarás que tu aplicación solicite permiso a los propietarios de la información antes de poder acceder a la misma. Para ello muchos proveedores de servicios web utilizan un protocolo llamado **OAuth** (la versión actual es la 2.0).



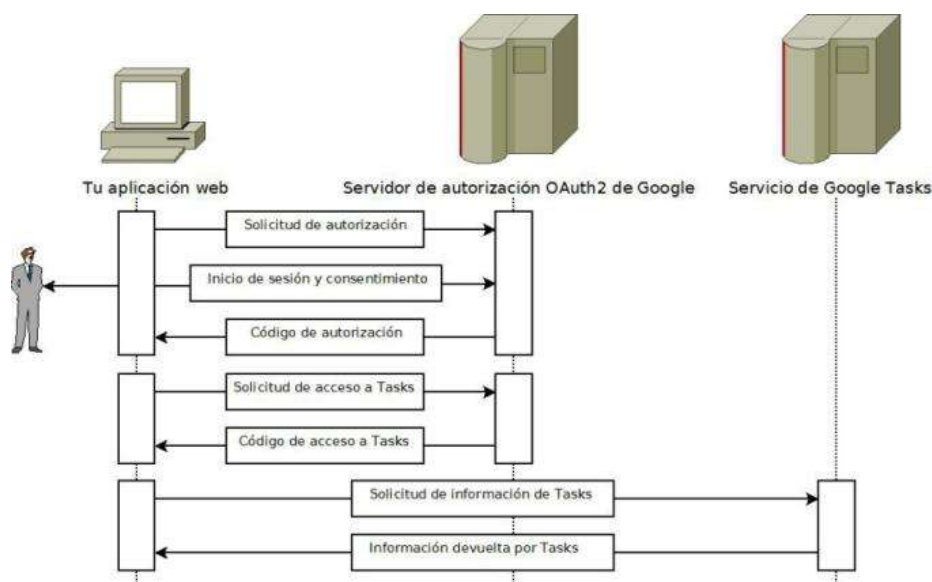
3.1.- OAuth2.

El protocolo estándar de autorización **OAuth2**, permite a una aplicación externa obtener acceso a información de carácter privado a través de un servicio web. Para ello establece un acuerdo de acceso a la misma entre la aplicación externa, el servicio web y el propietario de los datos a los que se solicita el acceso.

Por ejemplo, si una aplicación "X" solicita a Google acceso a los calendarios del usuario "dwes", Google pedirá a "dwes" permiso indicándole qué aplicación es la que solicita el acceso y a qué información. Si el usuario "dwes" otorga permiso, la aplicación "X" podrá acceder a los datos que solicitó a través del servicio de Google.

En uno de los videotutoriales anteriores, en el que creaste los identificadores necesarios para utilizar las API de Yahoo!, vimos un ejemplo de utilización de OAuth. Al utilizar una cuenta de Google para registrarnos como desarrolladores, Yahoo! necesitaba utilizar los servicios de Google para acceder a parte de nuestra información. Para ello, remitió una solicitud a Google, y éste a su vez nos mostró una pantalla informando de que Yahoo! solicitaba información nuestra y pidiéndonos permiso para darle acceso.

OAuth2 funciona de forma similar pero ligeramente distinta dependiendo de quién solicite acceso a la información. En nuestro caso supondremos que el solicitante será siempre una aplicación web. Veamos por ejemplo qué sucede cuando nuestra aplicación necesita acceder a información personal del usuario a través del servicio de Google Tasks. En este caso, los pasos que se seguirán son los siguientes:



- ✓ La aplicación web se comunica con el servidor de autorización OAuth2, indicando la información a que quiere acceder y el tipo de acceso a la misma.
- ✓ El servidor de autorización OAuth2 requiere al usuario de la aplicación web a que inicie sesión con su cuenta de Google (si aún no lo ha hecho), y le redirige a una página en la que le pide su consentimiento para otorgar acceso a su información privada.

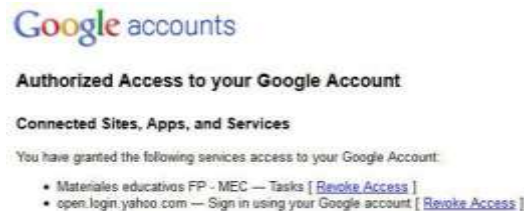
Materiales educativos FP - MEC está solicitando permiso para:

- ▶ Administra tus tareas.
- ▶ Realizar estas operaciones cuando no esté utilizando la aplicación

Formación Profesional

Materiales educativos FP - MEC
[Más información](#)

- ✓ Si el usuario da su consentimiento, el servidor de autorización OAuth2 devuelve a la aplicación web un código de autorización.
- ✓ Antes de poder acceder a la información privada del usuario, la aplicación web debe intercambiar ese código de autorización por otro código de acceso.
- ✓ Utilizando el código de acceso, la aplicación puede utilizar el servicio de Google Tasks para gestionar la información privada del usuario, dentro de los límites de acceso que se han otorgado.
- ✓ Los códigos de acceso tienen un tiempo de vida limitado. Cuando caducan, la aplicación ha de comunicarse de nuevo con el servidor de autorización OAuth2 para obtener un código de refresco.



http://www.youtube.com/watch?feature=player_embedded&v=ssrhy1pAlak

Utilizar OAuth2 con Google

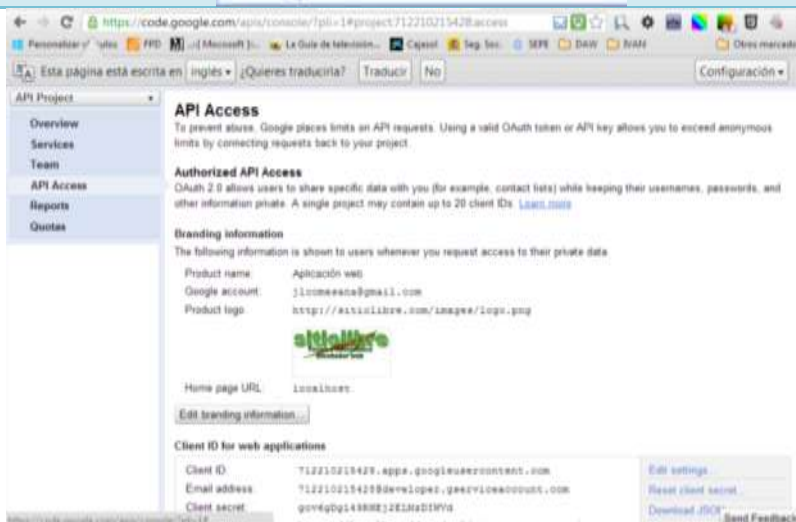
Iniciamos sesión en Google con nuestra cuenta de gmail y accedemos a <http://code.google.com/apis/console>

Seleccionamos la opción API Access y creamos un identificador de cliente OAuth2 para nuestra aplicación web

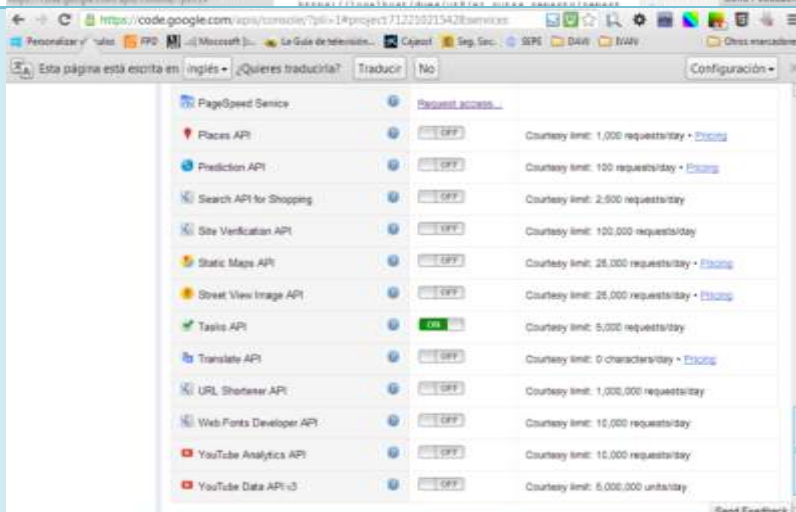
Rellenamos la información relativa a nuestra aplicación y pulsamos sobre Create Client ID.



Es importante ajustar correctamente las URIs de redirección, porque es a dónde podrá dirigirse el navegador tras el consentimiento del usuario. Siempre podremos cambiar los datos pulsando sobre *Edit setting*



Uno de los servicios que requiere OAuth2 es Google Tasks. Puedes activarlo para crear una aplicación que lo utilice, para lo que seleccionaremos la opción *Services* del menú de Google Apis, y pulsaremos sobre el botón OFF para activarlo, lo cual tiene efecto tras aceptar los términos de uso. Y ya podrás crear tu aplicación web utilizando OAuth2 con los servicios de Google Task



Existe una extensión PHP para programar las partes cliente y servidor del protocolo OAuth. **Extensión OAuth.** <http://es.php.net/manual/es/book.oauth.php>

Al utilizar OAuth2, cuando tu aplicación solicite información privada de un usuario, deberá acreditar su autorización utilizando:



Un código de autorización.



Un código de acceso.

Antes de acceder a información privada de un usuario, tu aplicación deberá estar en posesión de un código de autorización, pero tendrá que utilizarlo para conseguir un código de acceso, que será el que finalmente utilice

3.2.- JSON y XML.

Muchas de las operaciones que se llevan a cabo cuando utilizas un servicio web implican la obtención de cierta información por parte del mismo. La información obtenida puede ser bastante sencilla o tener cierto grado de complejidad. Por ejemplo, cuando utilizas un servicio de geocodificación para averiguar las coordenadas concretas de una dirección, obtienes básicamente esas coordenadas. Pero cuando utilizas un servicio como **Google Directions** para averiguar la ruta a seguir entre dos puntos, la respuesta que obtienes es una ruta compuesta por una serie de indicaciones a seguir para llegar al destino.

Los formatos más utilizados por los servicios web para dar formato a esas respuestas son dos: **JSON** y **XML**. En algunos casos tendrás que adaptar tu código al tipo de respuesta que ofrezca el servicio. Otros servicios permiten que escojas el tipo de respuesta que prefieras. Veamos cómo se pueden procesar de forma sencilla desde PHP mensajes en ambos formatos.

A partir de la versión 5.2 de PHP, se incluye por defecto la extensión JSON. Su funcionamiento es muy sencillo. Incorpora dos funciones para tratar con cadenas de texto en notación JSON:

Extensión JSON. <http://es.php.net/manual/es/book.json.php>

- ✓ **json_decode**. Decodifica una cadena de texto JSON y la transforma en un objeto PHP (opcionalmente también se podría convertir en un array).

```
<?php
$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';
var_dump(json_decode($json));
var_dump(json_decode($json, true));
?>
```

Salida:

```
object(stdClass) [1]
  public 'a' => int 1
  public 'b' => int 2
  public 'c' => int 3
  public 'd' => int 4
  public 'e' => int 5
array
  'a' => int 1
  'b' => int 2
  'c' => int 3
  'd' => int 4
  'e' => int 5
```

- ✓ **json_encode**. Función inversa a la anterior. Devuelve una cadena de texto en notación JSON a partir del contenido de una variable PHP.

```
<?php
$arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
echo json_encode($arr);
?>
```

Salida:

```
{"a":1,"b":2,"c":3,"d":4,"e":5}
```

Así como las opciones para trabajar con JSON desde PHP están bien definidas, para utilizar XML hay más variedad de herramientas para escoger. En PHP4 podías utilizar dos formas para procesar documentos en formato XML: **DOM** y **SAX**. La extensión **SimpleXML**, habilitada por defecto a partir de PHP 5.1.2, facilita la tarea de extraer información de un documento XML. Vamos a ver su funcionamiento.

Extensión SimpleXML. <http://es.php.net/manual/es/book.simplexml.php>

La extensión convierte un documento XML en un objeto de la clase **SimpleXMLElement**. Puedes cargar el documento:

- ✓ desde una cadena de texto, utilizando la función **simple_load_string**.

```
$xml = simplexml_load_string($cadena);
```

- ✓ desde un fichero, utilizando la función `simplexml_load_file`. Puedes utilizar una dirección URI como origen, por ejemplo:

```
$xml = simplexml_load_file('http://localhost/dwes/ut8/config.xml');
```

Los nodos y atributos del documento XML se convierten en atributos. Los nodos pasan a ser arrays que contienen a su vez como elementos los atributos y subnodos del documento XML. Por ejemplo, para acceder al resumen (`summary`) de una ruta en formato XML obtenida a partir del servicio **Google Directions**, podrías utilizar:

```
$ruta = simplexml_load_file('ruta al servicio');  
$ruta->route[0]->summary
```

Ruta de Google Directions.

<http://code.google.com/intl/es-ES/apis/maps/documentation/directions/#XML>

Es importante que conozcas cómo procesar documentos XML para extraer información de los mismos. En el manual de PHP puedes encontrar información sobre la utilización de la extensión SimpleXML.

Extensión SimpleXML.

<http://www.php.net/manual/es/book.simplexml.php>

4.- Creación de aplicaciones web híbridas.

Caso práctico

Pasa la semana y **Carlos** consigue tener a punto la nueva versión de la aplicación. Para la presentación, pone las dos versiones en funcionamiento, y no le comenta a nadie los nuevos cambios. Si hay algún problema con las últimas modificaciones, echará mano de la versión anterior. Cuando en BK Programación muestran a **Esteban** la aplicación, éste queda asombrado por el resultado. El nuevo servicio de gestión de envíos sorprende a todos positivamente, especialmente a **Ada** y a **Juan**, que empiezan a darse cuenta de las nuevas capacidades que ha adquirido **Carlos** en las últimas semanas. **Esteban** comenta que muy posiblemente les sea útil, especialmente al poder consultar la información de los envíos desde un dispositivo móvil. Pasarán un par de meses probando la aplicación, y a continuación plantea la posibilidad de tener una nueva reunión para hablar de posibilidades de ampliación y de otros proyectos. Ha quedado muy contento con el trabajo de **Carlos** y quiere seguir contando con él próximamente.

Para crear aplicaciones web híbridas, necesitarás que tu aplicación web acceda a diversos servicios para obtener información. En muchas ocasiones esos servicios estarán accesibles mediante HTTP. En otras ocasiones, especialmente cuando accedas a información privada de un usuario, necesitarás utilizar un protocolo seguro como HTTPS.

Tanto si vas a generar tu propio código para acceder a un servicio web, como si utilizas una API ya programada, es posible que necesites utilizar la librería **cURL**.

Manual de PHP.

<http://es.php.net/manual/es/book.curl.php>

cURL es una librería (y también una herramienta para utilizar desde la línea de comandos), que permite utilizar de forma sencilla varios protocolos de comunicaciones para transmitir información. Soporta, entre otros, los protocolos HTTP, HTTPS, FTP, TFTP, TELNET, LDAP, SMTP, POP3 e IMAP. Es importante que te asegures de que tu instalación de PHP incluye dicha librería. Si no la tienes activada en tu instalación de PHP, en Ubuntu puedes instalarla ejecutando:

```
sudo apt-get install php5-curl"
```

```
smr@ubuntu-profe:~$ sudo apt-get install php5-curl
[sudo] password for smr:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes extras:
  libapache2-mod-php5 php5-cli php5-common php5-gd php5-mysql
Paquetes sugeridos:
  php5-suhosin
Se instalarán los siguientes paquetes NUEVOS:
  php5-curl
Se actualizarán los siguientes paquetes:
  libapache2-mod-php5 php5-cli php5-common php5-gd php5-mysql
5 actualizados, 1 se instalarán, 0 para eliminar y 204 no actualizados.
Necesito descargar 6325kB de archivos.
Se utilizarán 119kB de espacio de disco adicional después de esta operación.
¿Desea continuar [S/n]? █
```

Recuerda reiniciar apache tras la instalación, y comprueba que la ejecución de `phpinfo()` muestra algo como lo siguiente.

curl	
curl support	enabled
CURL Information	7.15.1
Age	5
Features	
AsynchDNS	no
Debug	no
GSS-Negotiate	no
IDN	no
IPv6	no
Language	no
NTLM	no
SPNEGO	no
SSL	no
SSPI	no
libSASL	no
libz	no
libzstd	no
Protocols	ftp, http, https, imap, irc, ldap, ldaps, mqtt, nntp, pop3, smtp, telnet, tftp
Proxy	no
SSL Version	OpenSSL 1.0.1
zlib Version	1.2.3

En Windows la instalación necesita que añadas otras dos librerías:

`libeay32.dll` y `ssleay32.dll`.

En el manual de PHP y en otras páginas web tienes más información sobre la instalación de cURL en este sistema operativo.

Instalación de cURL en Windows.

<http://dy3g0.wordpress.com/2008/12/09/activar-curl-en-windows/>

Aunque la instalación y utilización de cURL en sistemas Linux/Ubuntu es inmediata, no sucede lo mismo con los sistemas Windows. En Windows la librería cURL se incluye en la instalación estándar de PHP, y no incluye una lista de autoridades de certificación.

Esta lista de autoridades de certificación es imprescindible para establecer conexiones seguras utilizando el protocolo HTTPS. Los certificados de los sitios web seguros deben estar firmados por una autoridad de certificación confiable, y su firma se debe poder comprobar. Si no tenemos una lista de autoridades de certificación confiables, no se podrá realizar esta comprobación y obtendremos un mensaje como:

```
CURL Error 60: SSL certificate problem, verify that the CA cert is OK.
```

Para solventar este problema, desde la web de cURL puedes descargar la lista adaptada de autoridades de certificación que incluye Mozilla en su navegador Firefox. Al utilizar la librería cURL desde tu código, tendrás que indicarle que utilice esa lista de autoridades de certificación, añadiendo una línea como la siguiente:

```
curl_setopt($ch, CURLOPT_CAINFO, 'ruta a la lista');
```

Lista de autoridades de certificación.

<http://curl.haxx.se/docs/caextract.html>

Si utilizas Windows para los ejemplos de aplicaciones web híbridas de este tema, tendrás que modificar el código de la API de Google para no tener problemas de certificados con cURL.

La librería cURL debe utilizar una lista válida de autoridades de certificación:



Para poder acceder a servidores web utilizando HTTPS.

Para poder acceder a servidores web utilizando HTTP.

Para poder certificar la validez del certificado digital que envía el servidor cuando se utiliza HTTPS, cURL necesita una lista válida de autoridades de certificación

4.1.- Yahoo! PlaceFinder.

Existen muchas fuentes de datos disponibles en Internet que puedes utilizar para construir una aplicación web híbrida. En esta unidad vamos a centrarnos en la información accesible a través de servicios web, concretamente en los que ofrecen las empresas Google y Yahoo!.



Comenzaremos creando una aplicación web híbrida sencilla que haga uso de los servicios de geocodificación que ofrecen ambos. El de Yahoo! se llama PlaceFinder. Puedes encontrar toda la información necesaria sobre su utilización en la web de desarrollo de Yahoo!.

Yahoo! PlaceFinder.

<http://developer.yahoo.com/geo/placefinder/>

El servicio **Yahoo! PlaceFinder** se basa en REST y los datos relativos a la petición se envían mediante parámetros **GET**. Es necesario indicar como mínimo un parámetro de localización.

La forma más sencilla de indicar una localización es utilizando el parámetro `location` (o su equivalente `q`).

Existe otro tipo de parámetros, los de control, que permiten indicar otra información no directamente relacionada con la localización. Es obligatorio el parámetro `appid` para indicar el ID de tu aplicación web. Otros parámetros de control son:

Parámetro de control	Significado
appid	ID de la aplicación que utiliza el servicio. Es obligatorio.
locale	Código del lenguaje y país. Por defecto "en_US". En nuestro caso deberíamos utilizar "es_ES".
count	Número máximo de respuestas que se devolverán. Por defecto 10.
flags	Cadena de caracteres que especifica qué datos se obtendrán. Por ejemplo: J – Indica que la información se devuelva en formato JSON (por defecto se utiliza XML). P – Indica que la información se devuelva en formato PHP serializado. G – Devuelve información global, no específica de los Estados Unidos.
gflags	Cadena de caracteres que especifica cómo se realizará la búsqueda. Por ejemplo: L – Buscar sólo en el país que se indica. R – Realizar una búsqueda inversa: obtener la dirección a partir de una latitud y longitud.

En la documentación del servicio tienes una lista completa de los parámetros de localización y de control que puedes emplear.

[Parámetros de localización.](#)

<http://developer.yahoo.com/geo/placefinder/guide/requests.html>

Por ejemplo, una petición simple podría tener la siguiente forma:

http://where.yahooapis.com/geocode?location=Plaza+de+la+Peregrina,Pontevedra,Spain&locale=es_ES&flags=G&count=1&appid=tuID

Recuerda incluir en las peticiones a los servicios de Google y Yahoo!, tus propios identificadores (los que te han asignado al registrarte) allí donde sea necesario.

4.1.1- Yahoo! PlaceFinder (II).

Las respuestas obtenidas al utilizar el servicio, contienen los siguientes elementos (indiferentemente de si se indica XML, JSON o PHP serializado):



Elemento	Significado
ResultSet	Elemento de nivel superior que contiene al resto.
Error	Código de error (0 si no se ha producido ningún error). En la documentación del servicio se incluye una tabla con los posibles códigos de error y su significado.
ErrorMessage	Mensaje de error. "No error" o "Sin errores" si no se ha producido ninguno.
Locale	Código del lenguaje utilizado en la respuesta.
Quality	Estimación de la calidad de la respuesta obtenida.
Found	Número de resultado obtenidos.
Result	Cada uno de los resultados obtenidos.

En la documentación del servicio tienes toda la información sobre los elementos que conforman los resultados obtenidos.

[Formato de las respuestas de PlaceFinder.](#)

<http://developer.yahoo.com/geo/placefinder/guide/responses.html>

Por ejemplo, como respuesta a la petición anterior podíamos obtener el siguiente documento en formato XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<ResultSet version="1.0">
  <Error>0</Error>
  <ErrorMessage>Sin Errores</ErrorMessage>
  <Locale>es_ES</Locale>
  <Quality>87</Quality>
  <Found>1</Found>
  <Result>
    <quality>72</quality>
    <latitude>42.430283</latitude>
    <longitude>-8.643625</longitude>
    ...
  </Result>
</ResultSet>
```

Si quisieras efectuar una consulta inversa (obtener una dirección a partir de unas coordenadas de latitud y longitud), podrías realizar una consulta utilizando `gflags=R`, como:

```
http://where.yahooapis.com/geocode?location=42.430283,-
8.643625&flags=G&locale=es_ES&gflags=R&appid=tuID
```

Y obtendrías:

```
<?xml version="1.0" encoding="UTF-8"?>
<ResultSet version="1.0">
  <Error>0</Error>
  <ErrorMessage>Sin errores</ErrorMessage>
  <Locale>es_ES</Locale>
  <Quality>99</Quality>
  <Found>1</Found>
  <Result>
    <quality>99</quality>
    <latitude>42.430283</latitude>
    <longitude>-8.643625</longitude>
    ...
    <line1>praza da Peregrina, 9</line1>
    <line2>36001 Pontevedra</line2>
    <line3></line3>
    <line4>España</line4>
    ...
  </Result>
</ResultSet>
```

Relaciona las palabras con su significado relativo al servicio Yahoo! PlaceFinder:

Palabra	Relación	Significado
<code>appid</code>	3	1. Número de resultados que incluye la respuesta.
<code>Found</code>	1	2. Parámetro que se utiliza en una petición para indicar el número máximo de respuestas que se devolverán.
<code>count</code>	2	3. Parámetro que se utiliza en una petición para indicar el identificador de la aplicación.
<code>Quality</code>	4	4. Estimación de la calidad de la respuesta obtenida.

En la pregunta figuran tanto parámetros que se utilizan al hacer las peticiones, como elementos que se incluyen en las respuestas.

4.2.- Google Geocoding.

El servicio de Google Geocoding, que forma parte de los servicios web de Google Maps, es muy similar a PlaceFinder de Yahoo!. También se basa en peticiones REST, indicando a continuación el tipo de respuesta que queramos obtener:



Google Geocoding.

<https://developers.google.com/maps/documentation/geocoding/>

Google Geocoding respuestas en formato JSON (recomendado por Google).

<http://maps.google.com/maps/api/geocode/json>

Google Geocoding para respuestas en formato XML

<http://maps.google.com/maps/api/geocode/xml>

Como parte de la petición, puedes utilizar entre otros los siguientes parámetros:

Parámetro	Significado
address	Dirección de la que quieres obtener las coordenadas. Es obligatorio, salvo para peticiones inversas.
latlng	Coordenadas a partir de las cuales quieres obtener una dirección. Es obligatorio solo para peticiones inversas.
language	Idioma en el que se devuelven los resultados. Nosotros utilizaremos "es". Es opcional.
sensor	Indica si la solicitud proviene de un dispositivo con sensor de localización. Sus posibles valores son true y false. Es obligatorio.

Para usar el servicio Google Geocoding no necesitas indicar tu ID de desarrollador registrado en Google. Tienes más información sobre el mismo en su página de documentación.

Documentación de Google Geocoding.

<http://code.google.com/intl/es-ES/apis/maps/documentation/geocoding/>

Así, por ejemplo, las solicitudes siguientes son equivalentes a las que realizaste anteriormente al servicio Yahoo! PlaceFinder, pero devolviendo información en formato JSON en lugar de XML:

```
http://maps.google.com/maps/api/geocode/json?address=plaza+de+la+peregrina,pontevedra,spain&language=es&sensor=false
http://maps.google.com/maps/api/geocode/json?latlng=42.430283,-8.643625&language=es&sensor=false
```

La respuesta obtenida a la primera petición es:

```
{
  "results" : [
    {
      "address_components" : [
        ...
      ],
      "formatted_address" : "Plaza de la Peregrina, 36001 Pontevedra, España",
      "geometry" : {
        ...
        "location" : {
          "lat" : 42.43080090,
          "lng" : -8.644160399999999
        },
        ...
      },
      "types" : [ "route" ]
    }
  ],
  "status" : "OK"
}
```

En este caso, la información que nos interesa es la que contiene el elemento `location`. En la petición inversa, para obtener la dirección podremos usar los elementos `formatted_address` o `address_components`, que nos ofrecen la misma información de forma compacta o desglosada respectivamente.

De nuevo puedes recurrir a la documentación del servicio para obtener información sobre cada uno de los elementos que forman la respuesta.

El servicio Google Geocoding devuelve la información en un formato u otro:



Dependiendo de la URL que se utilice en la petición.



En función de un parámetro GET que se debe utilizar en la petición.

La parte final de la URL puede ser json o xml en función del formato en que necesites la respuesta.

4.3.- Aplicación web híbrida de geocodificación.

Vas a crear una aplicación sencilla que utilice los dos servicios web que acabas de ver de Google y Yahoo!. Se trata simplemente de ver cómo se pueden utilizar desde PHP. Para ello crearemos un interfaz como el siguiente:

El formulario está dividido en dos zonas. En la superior, el usuario podrá introducir unas coordenadas y en la inferior los datos de una dirección. Se trata de utilizar los dos botones del formulario para realizar consultas a ambos servicios de geocodificación, de tal forma que:

- ✓ Al utilizar el botón "Ver dirección con Yahoo!", se realiza una consulta inversa al servicio **PlaceFinder** y se cubren los datos de la zona inferior del formulario con la respuesta obtenida.
- ✓ Al pulsar sobre el botón "Ver coordenadas con Google" se lanza una petición a **Google Geocoding** y se cubren las coordenadas de la parte superior del formulario con las que se reciben.

Para realizar las llamadas mediante Ajax utilizarás la librería Xajax, vista en la unidad anterior. La función que se encarga de realizar la llamada a Yahoo! PlaceFinder incluye el siguiente código:

```
$respuesta = new xajaxResponse();
$search =
'http://where.yahooapis.com/geocode?location='.$coordenadas['latitud'].'+ '.$coordenadas['longitud'].'&flags=G&locale=es_ES&gflags=R&appid=tuID';

$xml = simplexml_load_file($search);
$respuesta->assign("calle", "value", (string) $xml->Result[0]->street . " " . $xml->Result[0]->house);
...
return $respuesta;
```

Y la que utiliza Google Geocoding para obtener las coordenadas de una dirección (también en formato XML):

```
$respuesta = new xajaxResponse();
$search =
'http://maps.google.com/maps/api/geocode/xml?address='.$coordenadas['calle'].'+ '.$coordenadas['ciudad'].'+ '.$coordenadas['pais'].'&language=es&sensor=false';
$xml = simplexml_load_file($search);

$respuesta->assign("latitud", "value", (string) $xml->result[0]->geometry->location->lat);
$respuesta->assign("longitud", "value", (string) $xml->result[0]->geometry->location->lng);
```

Examina el código completo de la aplicación que se incluye en el siguiente fichero. Asegúrate de ajustar la ruta a la librería Xajax y de configurar tu propio ID de aplicación para el acceso a Yahoo! PlaceFinder.

form.php

```
<?php
/**
 * Desarrollo Web en Entorno Servidor
 * Tema 8 : Aplicaciones web híbridadas
```

```

* Ejemplo geocodificación: form.php
*/

// Incluimos la librería Xajax
require_once("../../libs/xajax_core/xajax.inc.php");

// Creamos las funciones que van a ser llamadas
// desde JavaScript

function ubicaryahoo($coordenadas){
    $respuesta = new xajaxResponse();
    // Hacemos una búsqueda inversa (gflags=R), esto es
    // obtenemos una dirección a partir de unas coordenadas
    // Indicamos también búsqueda global (flags=G)
    // y localización española para el lenguaje (locale=es_ES)
    $search =
'http://where.yahooapis.com/geocode?location='.$coordenadas['latitud'].'+ '.$coordenadas['longitud'].'&flags=G&locale=es_ES&gflags=R&appid=tuID';
    $xml = simplexml_load_file($search);

    $respuesta->assign("calle", "value", (string) $xml->Result[0]->street . " " . $xml->Result[0]->house);
    $respuesta->assign("ciudad", "value", (string) $xml->Result[0]->level3 . " " . $xml->Result[0]->level2 . " " . $xml->Result[0]->level1);
    $respuesta->assign("pais", "value", (string) $xml->Result[0]->level0);
    $respuesta->assign("cp", "value", (string) $xml->Result[0]->uzip);

    $respuesta->assign("enviarcoordenadas","value","Ver dirección");
    $respuesta->assign("enviarcoordenadas","disabled",false);
    $respuesta->assign("enviardireccion","disabled",false);
    return $respuesta;
}

function ubicargoogle($coordenadas){
    $respuesta = new xajaxResponse();
    // Indicamos idioma español (language=es) y
    // que no estamos usando un sensor de localización (sensor=false)
    $search =
'http://maps.google.com/maps/api/geocode/xml?address='.$coordenadas['calle'].'+ '.$coordenadas['ciudad'].'+ '.$coordenadas['pais'].'&language=es&sensor=false';
    $xml = simplexml_load_file($search);

    $respuesta->assign("latitud", "value", (string) $xml->result[0]->geometry->location->lat);
    $respuesta->assign("longitud", "value", (string) $xml->result[0]->geometry->location->lng);

    $respuesta->assign("enviardireccion","value","Ver coordenadas");
    $respuesta->assign("enviarcoordenadas","disabled",false);
    $respuesta->assign("enviardireccion","disabled",false);
    return $respuesta;
}

// Creamos el objeto xajax
$xajax = new xajax();

// Registramos las funciones que vamos a llamar desde JavaScript
$xajax->register(XAJAX_FUNCTION,"ubicaryahoo");
$xajax->register(XAJAX_FUNCTION,"ubicargoogle");

// Y configuramos la ruta en que se encuentra la carpeta xajax_js
$xajax->configure('javascript URI','../../libs/');

// El método processRequest procesa las peticiones que llegan a la página
// Debe ser llamado antes del código HTML
$xajax->processRequest();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title>Ejemplo Tema 8: Geolocation con jQuery4PHP</title>
<link rel="stylesheet" href="estilos.css" type="text/css" />
<?php
// Le indicamos a Xajax que incluya el código JavaScript necesario
$xajax->printJavascript();
?>
<script type="text/javascript" src="ubicar.js"></script>

```

```

<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
<script type="text/javascript">
  // <![CDATA[
    $(document).ready(function() {
      navigator.geolocation.getCurrentPosition(
        function(posicion){
          $("#latitud").val(posicion.coords.latitude);
          $("#longitud").val(posicion.coords.longitude);
        }
      );
    });
  // ]]>
</script>
</head>

<body>
  <div id='form'>
    <form id='datos' action="javascript:void(null);">
      <fieldset>
        <legend>Servicios de geocodificación</legend>
        <div class='campo'>
          <label for='latitud' >Latitud:</label>
          <input type='text' name='latitud' id='latitud' />
        </div>
        <div class='campo'>
          <label for='longitud' >Longitud:</label>
          <input type='text' name='longitud' id='longitud' />
        </div>

        <div class='campo'>
          <input type='submit' id='enviarcoordenadas' name='enviar' value='Ver dirección con
Yahoo!' onclick="enviarCoordenadas();" />
          <input type='submit' id='enviardireccion' name='enviar' value='Ver coordenadas con
Google' onclick="enviarDireccion();" />
        </div>
        <div class='campo'>
          <label for='calle' >Calle:</label>
          <input type='text' name='calle' id='calle' />
        </div>
        <div class='campo'>
          <label for='ciudad' >Ciudad:</label><br />
          <input type='text' name='ciudad' id='ciudad' />
        </div>
        <div class='campo'>
          <label for='pais' >País:</label><br />
          <input type='text' name='pais' id='pais' />
        </div>
        <div class='campo'>
          <label for='cp' >CP:</label><br />
          <input type='text' name='cp' id='cp' maxlength="5" />
        </div>
      </fieldset>
    </form>
  </div>
</body>
</html>

```

ubicar.js

```

function enviarCoordenadas() {
  // Se cambia el botón de Enviar y se deshabilita
  // hasta que llegue la respuesta
  xajax.$('enviarcoordenadas').disabled=true;
  xajax.$('enviardireccion').disabled=true;
  xajax.$('enviarcoordenadas').value="Un momento...";

  // Aquí se hace la llamada a la función registrada de PHP
  xajax_ubicaryahoo (xajax.getFormValues("datos"));

  return false;
}

function enviarDireccion() {
  // Se cambia el botón de Enviar y se deshabilita
  // hasta que llegue la respuesta
  xajax.$('enviarcoordenadas').disabled=true;
  xajax.$('enviardireccion').disabled=true;

```

```
xajax.$('enviardiireccion').value="Un momento...";

// Aquí se hace la llamada a la función registrada de PHP
xajax_ubicargoogle (xajax.getFormValues("datos"));

return false;
}
```

estilos.css

```
#form {
    position: absolute;
    left: 50%;
    top: 50%;
    width: 340px;
    margin-left: -170px;
    height: 450px;
    margin-top: -210px;
    padding: 10px;
    border: 1px solid #ccc;
    background-color: #eee;
}

legend, h3 {
    font-family : Arial, sans-serif;
    font-size: 1.3em;
    font-weight: bold;
    color: #333;
}

#form .campo {
    margin-top: 8px;
    margin-bottom: 10px;
    margin-left: 10px;
}

#form label {
    font-family : Arial, sans-serif;
    font-size: 0.8em;
    font-weight: bold;
}

#form input[type="text"] {
    font-family : Arial, Verdana, sans-serif;
    font-size: 0.8em;
    line-height: 140%;
    color : #000;
    padding : 3px;
    border : 1px solid #999;
    height: 20px;
    width: 280px;
}

#form input[type="submit"] {
    width: 200px;
    height: 30px;
    padding-left: 0px;
}

.oculto {
    display: none;
}
```

HTML5 incluye entre sus nuevas características una API de geolocalización, que permite a las aplicaciones web utilizar código JavaScript para obtener las coordenadas en que se encuentra el usuario. La aplicación del código anterior utiliza esa funcionalidad. Para conocer más detalles sobre el funcionamiento de la geolocalización en HTML5 y ver cómo se puede integrar con Google Maps, puedes consultar la siguiente página web.

Geolocalización con HTML5. <http://blog.atruiweb.com/html5/geolocalizacion-con-html5>

4.4.- Google Directions.

Google Directions es un servicio web de Google, que al igual que Google Geocoding también forma parte de Google Maps, y cuya principal utilidad es el cálculo de rutas para llegar desde una ubicación origen a otra ubicación destino. Las rutas pueden incluir además puntos intermedios (hitos), y tanto ellos como el origen o el destino pueden especificarse mediante direcciones reconocibles por el usuario o mediante coordenadas (latitud y longitud).



Documentación de Google Directions.

<http://code.google.com/intl/es-ES/apis/maps/documentation/directions/>

Google Directions respuestas en formato JSON (recomendado por Google).

<http://maps.google.com/maps/api/directions/json>

Google Directions para respuestas en formato XML.

<http://maps.google.com/maps/api/directions/xml>

Los únicos parámetros que deben figurar obligatoriamente en una petición son:

- ✓ El punto origen (`origin`).
- ✓ El punto destino (`destination`).
- ✓ La indicación de si se utiliza o no un dispositivo de localización (`sensor`).

Entre los parámetros opcionales están:

- ✓ El idioma en que se devuelven los resultados (`language`).
- ✓ El medio de transporte para el cálculo de las rutas (`mode`).
- ✓ Los puntos intermedios o hitos (`waypoints`). Se deben separar unos de otros utilizando una barra vertical "|". Opcionalmente se puede incluir "`optimize:true`" como primer argumento, lo que provocará que se reordenen los puntos intermedios para optimizar la ruta. En este caso, en el elemento `waypoint_order` de la respuesta se mostrará el orden resultante de optimizar la ruta.

Por ejemplo, una consulta a Google Directions podría tener la siguiente forma:

```
http://maps.google.com/maps/api/directions/json?origin=42.430283,-8.643625&destination=42.402497,-8.812001&language=es&sensor=false
```

La respuesta obtenida en formato JSON mostrará las indicaciones necesarias para llegar desde el origen al destino. El código necesario en PHP para utilizar el servicio será:

```
// Con una URL como la anterior almacenada en la variable $url:
$json = file_get_contents($url);
$respuesta = json_decode($json);
```

Una vez decodificado el formato JSON obtenido, el acceso a la información se realiza utilizando los elementos del objeto `$respuesta`. Revisa la documentación del servicio para ver cómo se estructuran las respuestas. Por ejemplo:

```
$resumen = $respuesta->routes[0]->summary;
```

Se utiliza la función `file_get_contents` de PHP para almacenar en una variable la respuesta obtenida del servicio Google Directions.

Función `file_get_contents`. <http://es.php.net/manual/es/function.file-get-contents.php>

Cuando crees una aplicación que utilice servicios proporcionados por terceros, deberás tener en cuenta siempre sus **licencias de uso**. Por ejemplo, los resultados obtenidos al usar tanto Google

Directions como Google Geocoding han de acompañarse obligatoriamente de los resultados de visualización de un mapa de Google. Además, en algunos casos las respuestas incluyen advertencias e información sobre derechos de autor que deberán mostrarse a los usuarios.

En una petición al servicio Google Directions, deben figurar obligatoriamente los parámetros:



origin, destination y sensor.

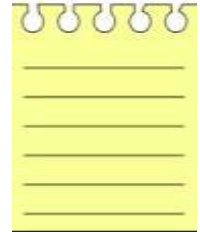


origin, destination, mode y sensor.

El parámetro mode indica el medio de transporte a utilizar pero es opcional (se supone automóvil por defecto)

4.5.- Google Tasks.

Los servicios que has utilizado hasta ahora recibían peticiones por parte del usuario (en nuestro caso una aplicación web) y generaban, y devolvían, respuestas a las mismas en formatos JSON o XML. El servicio Google Tasks necesita, además, una autorización para devolver información privada de un usuario. Para obtener esa autorización, nuestra aplicación deberá usar el protocolo OAuth2.



El servicio de Google Tasks nos permite gestionar las tareas personales del usuario.

Servicio de Google Tasks.

<http://code.google.com/intl/es-ES/apis/tasks/>

Básicamente existen dos tipos de recursos:

- ✓ **Listas de tareas.** Una de ellas es la lista de tareas por defecto; existe siempre y no se puede eliminar.
- ✓ **Tareas.** Es cada uno de los elementos que contiene una lista de tareas. Puede contener información como el título de la tarea, notas, o fechas.

Hay dos formas de utilizar el servicio:

- ✓ Mediante llamadas REST directamente, al igual que hicimos cuando utilizamos los servicios anteriores.
- ✓ Mediante una librería cliente, disponible para múltiples lenguajes (entre ellos PHP).

En la documentación del servicio tienes información y ejemplos sobre la utilización del servicio desde cualquiera de estos dos métodos. Para poder utilizar la librería cliente deberás, en primer lugar, descargarla en su versión para PHP.

Librerías cliente para Google Tasks.

<http://code.google.com/intl/es-ES/apis/tasks/libraries.html>

Una vez descomprimida y ubicada en una ruta accesible al servidor web, deberás incluirla en las aplicaciones que la utilicen. Existen un fichero común, y otro específico según el tipo de servicio al que necesites acceder. Por ejemplo, para poder utilizar el servicio Google Tasks tendrás que añadir las siguientes líneas en tu código PHP:

```
session_start();
require_once 'google-api-php-client/src/apiClient.php';
require_once 'google-api-php-client/src/contrib/apiTasksService.php';
```

Asegúrate de ajustar correctamente la ruta a la librería.

Será necesario crear dos objetos; uno de tipo `apiClient`, que utilizará OAuth2 para gestionar las autorizaciones de acceso a los servicios.

```
// Creamos el objeto de la API de Google
$cliente = new apiClient();

// Y lo configuramos con los nuestros identificadores
$cliente->setClientId('Aquí irá tu identificador de cliente');
$cliente->setClientSecret('Aquí tu clave secreta');
```

```
$cliente->setRedirectUri('http://localhost/dwes/ut8/tareas.php');
$cliente->setDeveloperKey('Aquí irá tu clave de la API de Google');
```

Es importante señalar que la URI de redirección debe ser válida y estar dada de alta como tal en la configuración de la aplicación web, tal y como se mostró en el videotutorial sobre OAuth2 (por ejemplo, puede ser la misma dirección de la página que estamos programando).

También necesitamos crear otro objeto del tipo específico según el servicio al que accedamos, en nuestro caso `apiTasksService`.

```
// Creamos un objeto para manejar las listas y sus tareas
$apitareas = new apiTasksService($cliente);
```

4.5.1- Google Tasks (II).

El primer paso para acceder al servicio es autenticarse utilizando el método `authenticate` de la clase `apiClient`. La clave de acceso obtenida se debe almacenar utilizando la función `setAccessToken`. Puede almacenarse en una variable de sesión para futuras llamadas al servicio.

```
if (isset($_SESSION['clave_acceso'])) {
    $cliente->setAccessToken($_SESSION['clave_acceso']);
} else {
    $cliente->setAccessToken($cliente->authenticate());
    $_SESSION['clave_acceso'] = $cliente->getAccessToken();
}
```



Una vez autenticado, puedes emplear el objeto de la clase `apiTasksService` para gestionar las listas de tareas y las tareas del usuario.

- ✓ Para crear una nueva lista de tareas:

```
$nuevalista = new TaskList();
$nuevalista->setTitle('titulo_lista');
$apitareas->tasklists->insert($nuevalista);
```

- ✓ Para crear una nueva tarea y agregarla a una lista de tareas:

```
$nuevatarea = new Task();
$nuevatarea->setTitle('titulo_tarea');
$nuevatarea->setNotes('notas_tarea');
$apitareas->tasks->insert('id_lista_tareas', $nuevatarea);
```

- ✓ Para eliminar una lista de tareas:

```
$apitareas->tasklists->delete('id_lista_tareas');
```

- ✓ Para eliminar una tarea de una lista:

```
$apitareas->tasks->delete('id_lista_tareas', 'id_lista');
```

- ✓ Para recorrer las listas de tareas y sus tareas:

```
// Para recorrer las listas de tareas
$listas = $apitareas->tasklists->listTasklists();
foreach ($listas['items'] as $lista) {
    // Para recorrer las tareas de cada lista
    $tareas = $apitareas->tasks->listTasks($lista['id']);
    foreach ($tareas['items'] as $tarea) {
        ...
    }
}
```

Para que la librería funcione correctamente en sistemas Windows, deberás solucionar el problema de cURL con las autoridades de certificación. Una vez descargado el fichero de autoridades tal y como se mencionó anteriormente, deberás indicar que se utilice. Para ello, en el fichero `src/io/apiCurlIO.php`, tendrás que modificar la función `makeRequest` añadiendo la siguiente línea después del conjunto de llamadas a `curl_setopt`:

```
curl_setopt($ch, CURLOPT_CAINFO, 'ruta a la lista');
```

Para utilizar el servicio Google Tasks desde PHP, puedes emplear:

La API que ofrece Google.



La API que ofrece Google o llamadas REST.

*Puedes utilizar uno u otro método, pero la API que ofrece Google es el más sencillo de ambos.***4.6.- Aplicación web híbrida de gestión de repartos.**

Vamos a ver cómo puedes crear una aplicación web híbrida que utilice los servicios que acabas de ver, utilizando como punto de partida la aplicación web con la que has estado trabajando en unidades anteriores. El supuesto del que se parte es el siguiente:

Como la zona de influencia de la tienda aún es pequeña, y pensando también en mantener la relación con los clientes, se ha decidido hacer reparto directo de los productos que se compran en la tienda online. Para ello se ha pensado en crear una aplicación web híbrida con las siguientes características:

- ✓ Se utilizará la API del servicio de tareas de Google (Google Tasks) para almacenar como listas de tareas la información de los repartos. De esta forma la información podrá ser consultada desde cualquier lugar utilizando un dispositivo con conexión a Internet. Cada lista de tareas se corresponde en la aplicación con una lista de reparto, y cada una de sus tareas con un envío. Para diferenciar una lista de otra, se le pone como parte del título la fecha del día en que se hará el reparto.
- ✓ Para cada producto que se reparte se creará una tarea en la lista correspondiente. Esa tarea almacenará la dirección de envío y sus coordenadas. Para obtenerlas, y para mostrar su ubicación en un mapa, en el momento en que se introduzca la dirección se utilizará el servicio de geocodificación de Google (Google Geocoding).
- ✓ Para optimizar la ruta que se ha de recorrer, se utilizará Google Directions. La idea es reorganizar de forma automática el orden de los productos que se van a repartir cada día de forma que se minimice la distancia recorrida.

La apariencia de la aplicación será:

Ejemplo Tema 8: Rutas de reparto

Crear Nueva Lista de Reparto:

Repartos 21/12/2012 (Ver lista de tareas)

- HP Laserjet Pro P1102W - Avenida de Betamar 25, O Grove, Pontevedra (42.4951299, -8.8603783) (Ver mapa)
- Sony Bravia KDL-32BX400 - Avenida de Las Carolinas 36, Vilagarcía de Arousa, Pontevedra (42.5941793, -8.7582143) (Ver mapa)
- Packard Bell 19103 - Rúa de Calvo Sotelo 3, Pontevedra (42.4228354, -8.6385278) (Ver mapa)

Repartos 22/12/2012 (Ver lista de tareas)

- Acer AC950 - Calle de la Rúa 5, Cambado, Pontevedra (42.4334362, -8.7054620) (Ver mapa)
- HP Mini 110-3120 - Avenida de Cambados 20, Meis, Pontevedra (42.5165285, -8.6988174) (Ver mapa)
- Canon Legria FS306 - Calle del Rego 2, Cambados, Pontevedra (42.5132317, -8.8150704) (Ver mapa)
- Epson Stylus SX315W - Rúa Crucero 37, Vilalonga, Pontevedra (42.4429686, -8.8298182) (Ver mapa)

Repartos 23/12/2012 (Ver lista de tareas)

- Dell Optiplex 360 - Marqués de Riestra 16, Pontevedra (42.4301398, -8.6464481) (Ver mapa)

Datos del nuevo envío

Dirección:

Latitud:

Longitud:

Título:

Cuando se cree una nueva tarea (un nuevo envío), se pedirá la dirección y se mostrará una pantalla como la siguiente para que el usuario complete los datos necesarios.

Se utiliza también Google Maps para mostrar en una nueva ventana el mapa correspondiente a las coordenadas de envío de los productos.

4.6.1.- Aplicación web híbrida de gestión de repartos (II).

Para gestionar (crear y eliminar) las listas de tareas y sus tareas, puedes utilizar parámetros **GET** y recargar la misma página. Si está presente un parámetro **'accion'**, se realizará un procesamiento determinado. Por ejemplo:

```
switch ($_GET['accion']) {
    case 'nuevalista':
        if (!empty($_GET['nuevotitulo'])) {
            // Crear una nueva lista de reparto
            try {
                $nuevalista = new TaskList();
                $nuevalista->setTitle($_GET['nuevotitulo']);
                $apitareas->tasklists->insert($nuevalista);
            } catch (Exception $e) {
                $error="Error al crear un nuevo reparto.";
            }
        }
    }
}
```

Para abrir una ventana que muestre ciertas coordenadas en un mapa, puedes utilizar una función Javascript como la siguiente:

```
function abrirMaps(coordenadas) {
    var url = "http://maps.google.com/maps?hl=es&t=h&z=17&output=embed&ll=";

    if(!coordenadas)
        // Cogemos las coordenadas del diálogo
        url+=$("#latitud").val()+"", "+$("#longitud").val();
    else
        // Si hay coordenadas, las usamos
        url+=coordenadas;
    window.open(url, 'nuevaventana', 'width=425,height=350');
}
```



Para obtener las coordenadas de una dirección concreta, una solución es utilizar Xajax para llamar a Google Geocoding, de forma similar a como ya hiciste en la aplicación web de geocodificación anterior.

4.6.2.- Aplicación web híbrida de gestión de repartos (III).

Un caso especial es el método a utilizar para optimizar las distintas entregas de un pedido (las tareas de una lista). Una solución es realizar el proceso en dos pasos. En primer lugar puedes utilizar Xajax para llamar al servicio Google Directions y obtener el orden óptimo de entrega. Al pulsar en el botón **'Ordenar'** de una lista, se ejecuta la siguiente función JavaScript:

```
function ordenarReparto(idReparto) {
    // Utilizamos jQuery para obtener una lista con las coordenadas
    // de los puntos intermedios que debemos ordenar
    var paradas = $('#'+idReparto+' li').map(function() {
        return this.title;
    }).get().join('|');

    // Modo síncrono (esperamos a obtener una respuesta)
    var respuesta = xajax.request({xjxfun:"ordenarReparto"}, {mode:'synchronous', parameters:
[paradas]});
    if (respuesta) {
        // Cogemos la URL base del documento actual
        var url = document.location.href.replace(/\?.*/,'');
        // Añadimos el código de la lista de reparto
        url += '?accion=ordenarEnvios&reparto='+idReparto;
        // Y las nuevas posiciones que deben ocupar los envíos
        for(var r in respuesta) url += '&pos[]='+respuesta[r];

        window.location = url;
    }
}
```

Esta función ejecuta mediante AJAX el siguiente código PHP, que obtiene el orden óptimo de reparto:

```
function ordenarReparto($coordenadasPuntosIntermedios){
    // Coordenadas del "almacén" (punto de origen y destino)
    // Se podría añadir código para permitir al usuario indicarlo
    $coordenadasOrigen = "42.402497,-8.812001";
    $url =
    'http://maps.google.es/maps/api/directions/json?origin='.$coordenadasOrigen.'&destination='.$c
    oordenadasOrigen;

    // Se añaden los puntos intermedios, indicando que optimice
    $url .= '&waypoints=optimize:true|';
    $url .= $coordenadasPuntosIntermedios.'&sensor=false';

    // Como el resultado es JSON, lo procesamos de la siguiente forma
    $json = file_get_contents($url);
    $respuesta = json_decode($json);
    $orden = $respuesta->routes[0]->waypoint_order;

    // Y devolvemos ese array
    $respuesta = new xajaxResponse();
    $respuesta->setReturnValue($orden);
    return $respuesta;
}
```

El array obtenido se envía en parámetros **GET** a la misma página, que lo debe procesar para ordenar las tareas según indica.

Para obtener una ruta optimizada utilizando el servicio Google Directions, debes utilizar `optimize:true` en el parámetro `waypoints`, y al procesar la respuesta recibida:



Revisar el orden de los elementos `step` recibidos.



Revisar el orden que contiene el elemento `waypoint_order`.

El elemento `waypoint_order` contiene el orden óptimo de los puntos intermedios que tu indicas al realizar la petición

4.6.3.- Aplicación web híbrida de gestión de repartos (IV).

Una vez obtenido el array que indica el orden óptimo de los puntos intermedios, se vuelve a cargar la página y en ese momento se deberán procesar los parámetros **GET** recibidos que indican cómo reordenar las tareas de la lista.

```
switch ($ _GET['accion']) {
case 'ordenarEnvios':
    if (!empty($ _GET['reparto']) && !empty($ _GET['pos'])) {
        // Reordenar las tareas de envío según el orden que se recibe
        try {
            // Obtenemos todas las tareas de la lista de reparto
            $tareas = $apitareas->tasks->listTasks($ _GET['reparto']);

            // Y las movemos según se indica en el array 'pos'
            // 'pos' indica qué posición debe tener cada tarea
            // $pos[0] = 3 significa que la 1ª tarea (la de índice 0)
            // debe ponerse en la 4ª posición (la de índice 3)

            // Lo convertimos en el array 'orden' que contiene
            // las tareas que debe haber en cada posición de la lista
            // $orden[3] = 0 significa que en la 4ª posición
            // debemos poner la 1ª tarea
            $orden = array_flip($ _GET['pos']);

            // Recorremos el array en orden inverso
            // En cada paso ponemos una tarea en la primera posición
            for($i=count($orden)-1; $i>=0; $i--){
                $apitareas->tasks->move($ _GET['reparto'], $tareas['items'][$orden[$i]]['id']);
            }
        } catch (Exception $e) {
            $error="Se ha producido un error al intentar ordenar los envíos del reparto.";
        }
    }
    break;
```

Intenta completar por ti mismo la programación de la aplicación web. Puedes descargar y revisar el código de la solución completa en el siguiente enlace. Recuerda que debes ajustar el código para indicar las rutas correctas a las librerías de Xajax y de Google, y teclear en el mismo tus propias claves de uso de los servicios web de Google y la ruta de redirección, que debe estar registrada.

repartos.php

```
<?php
/**
 * Desarrollo Web en Entorno Servidor
 * Tema 8 : Aplicaciones web híbridas
 * Ejemplo Rutas de reparto: repartos.php
 */

session_start();

// Incluimos la API de Google
require_once '../libs/google-api-php-client/src/apiClient.php';
require_once '../libs/google-api-php-client/src/contrib/apiTasksService.php';

// y la librería Xajax
require_once("../libs/xajax_core/xajax.inc.php");

// Creamos el objeto xajax
$xajax = new xajax('ajaxmaps.php');

// Configuramos la ruta en que se encuentra la carpeta xajax_js
$xajax->configure('javascript URI','../libs/');

// Y registramos las funciones que vamos a llamar desde JavaScript
$xajax->register(XAJAX_FUNCTION,"obtenerCoordenadas");
$xajax->register(XAJAX_FUNCTION,"ordenarReparto");

// Creamos el objeto de la API de Google
$cliente = new apiClient();

// Y lo configuramos con los nuestros identificadores
$cliente->setClientId('Aquí irá tu identificador de cliente');
$cliente->setClientSecret('Aquí tu clave secreta');
$cliente->setRedirectUri('http://localhost/dwes/ut8/ej_rutas_reparto/repartos.php');
$cliente->setDeveloperKey('Aquí irá tu clave de la API de Google');

// Creamos también un objeto para manejar las listas y sus tareas
$apitareas = new apiTasksService($cliente);

// Comprobamos o solicitamos la autorización de acceso
if (isset($_SESSION['clave_acceso'])) {
    $cliente->setAccessToken($_SESSION['clave_acceso']);
} else {
    $cliente->setAccessToken($cliente->authenticate());
    $_SESSION['clave_acceso'] = $cliente->getAccessToken();
}

// Comprobamos si se debe ejecutar alguna acción
if (isset($_GET['accion'])) {
    switch ($_GET['accion']) {
        case 'nuevalista':
            if (!empty($_GET['nuevotitulo'])) {
                // Crear una nueva lista de reparto
                try {
                    $nuevalista = new TaskList();
                    $nuevalista->setTitle($_GET['nuevotitulo']);
                    $apitareas->tasklists->insert($nuevalista);
                }
                catch (Exception $e) {
                    $error="Se ha producido un error al intentar crear un nuevo reparto.";
                }
            }
            break;
        case 'nuevatarea':
            if (!empty($_GET['nuevotitulo']) && !empty($_GET['idreparto']) &&
!empty($_GET['latitud']) && !empty($_GET['longitud'])) {
                // Crear una nueva tarea de envío
                try {
                    $nuevatarea = new Task();
                    $nuevatarea->setTitle($_GET['nuevotitulo']);
```

```

        if (isset($_GET['direccion'])) $nuevatarea->setTitle($_GET['nuevotitulo']." - ".$_GET['direccion']);
        else $nuevatarea->setTitle($_GET['nuevotitulo']);
        $nuevatarea->setNotes($_GET['latitud']." ".$_GET['longitud']);
        // Añadimos la nueva tarea de envío a la lista de reparto
        $apitareas->tasks->insert($_GET['idreparto'], $nuevatarea);
    }
    catch (Exception $e) {
        $error="Se ha producido un error al intentar crear un nuevo envío.";
    }
}
break;
case 'borrarlista':
    if (!empty($_GET['reparto'])) {
        // Borrar una lista de reparto
        try {
            $apitareas->tasklists->delete($_GET['reparto']);
        }
        catch (Exception $e) {
            $error="Se ha producido un error al intentar borrar el reparto.";
        }
    }
    break;
case 'borrartarea':
    if (!empty($_GET['reparto']) && !empty($_GET['envio'])) {
        // Borrar una tarea de envío
        try {
            $apitareas->tasks->delete($_GET['reparto'], $_GET['envio']);
        }
        catch (Exception $e) {
            $error="Se ha producido un error al intentar borrar el envío.";
        }
    }
    break;
case 'ordenarEnvios':
    if (!empty($_GET['reparto']) && !empty($_GET['pos'])) {
        // Reordenar las tareas de envío según el orden que se recibe en el array
'pos'
        try {
            // Primero obtenemos todas las tareas de la lista de reparto
            $tareas = $apitareas->tasks->listTasks($_GET['reparto']);

            // Y después las movemos según la posición recibida en el array 'pos'
            // El array 'pos' indica la posición que debe tener cada tarea de la lista
            // $pos[0] = 3 significa que la 1ª tarea (la de índice 0)
            // debe ponerse en la 4ª posición (la de índice 3)
            //
            // Lo convertimos en el array 'orden' que contiene las tareas que debe
haber
            // en cada posición de la lista
            // $orden[3] = 0 significa que en la 4ª posición debemos poner la 1ª tarea
            $orden = array_flip($_GET['pos']);

            // Recorremos el array en orden inverso, esto es, empezando por la tarea
            // que debería figurar en última posición de la lista
            // En cada paso ponemos una tarea en la primera posición de la lista
            for($i=count($orden)-1; $i>=0; $i--)
                $apitareas->tasks->move($_GET['reparto'],
                $tareas['items'][$orden[$i]]['id']);
        }
        catch (Exception $e) {
            $error="Se ha producido un error al intentar ordenar los envíos del
reparto.";
        }
    }
    break;
}

// Obtenemos el id de la lista de tareas por defecto, para no mostrarla
$listapordefecto = $apitareas->tasklists->get('@default');
$id_defecto = $listapordefecto['id'];
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8" />

```

```

<title>Ejemplo Tema 8: Rutas de reparto</title>
<link href="estilos.css" rel="stylesheet" type="text/css" />
<?php
// Le indicamos a Xajax que incluya el código JavaScript necesario
$xajax->printJavascript();
?>
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
<script type="text/javascript" src="codigo.js"></script>
</head>

<body>
<div id="dialogo">
  <a id="cerrarDialogo" onclick="ocultarDialogo();">x</a>
  <h1>Datos del nuevo envío</h1>
  <form id="formenvio" name="formenvio" action="<?php echo $_SERVER['PHP_SELF'];?>"
method="get">
    <fieldset>
      <div id="datosDireccion">
        <p>
          <label for='direccion' >Dirección:</label>
          <input type='text' size="60" name='direccion' id='direccion' />
        </p>
        <input type='button' id='obtenerCoordenadas' value='Obtener coordenadas'
onclick="getCoordenadas();" /><br />
      </div>
      <div id="datosEnvio">
        <p>
          <label for='latitud' >Latitud:</label>
          <input type='text' size="10" name='latitud' id='latitud' />
        </p>
        <p>
          <label for='longitud' >Longitud:</label>
          <input type='text' size="10" name='longitud' id='longitud' />
        </p>
        <p>
          <label for='nuevotitulo' >Título:</label>
          <input type='text' size="40" name='nuevotitulo' id='titulo' />
        </p>
        <input type='hidden' name='accion' value='nuevatarea' />
        <input type='hidden' name='idreparto' id='idrepartoactual' />
        <input type='submit' id='nuevoEnvio' value='Crear nuevo Envio' />
        <a href="#" onclick="abrirMaps();">Ver en Google Maps</a><br />
      </div>
    </fieldset>
  </form>
</div>
<div id="fondonegro" onclick="ocultarDialogo();"></div>
<div class="contenedor">
  <div class="encabezado">
    <h1>Ejemplo Tema 8: Rutas de reparto</h1>
    <form id="nuevoreparto" action="<?php echo $_SERVER['PHP_SELF'];?>" method="get">
      <fieldset>
        <input type='hidden' name='accion' value='nuevalista' />
        <input type='submit' id='crearnuevotitulo' value='Crear Nueva Lista de Reparto' />
        <label for='nuevotitulo' >con título:</label>
        <input type='text' name='nuevotitulo' id='nuevotitulo' />
      </fieldset>
    </form>
  </div>
  <div class="contenido">
    <?php
    $repartos = $apitareas->tasklists->listTasklists();
    // Para cada lista de reparto
    foreach ($repartos['items'] as $reparto) {
      // Excluyendo la lista por defecto de Google Tasks
      if($reparto['id'] == $id_defecto) continue;

      print '<div id="'. $reparto['id']. '>';
      print '<span class="titulo">'. $reparto['title']. '</span>';
      $idreparto = " ". $reparto['id']. " ";
      print '<span class="accion">(<a href="#"
onclick="ordenarReparto('.$idreparto.')">Ordenar</a></span>';
      print '<span class="accion">(<a href="#" onclick="nuevoEnvio('.$idreparto.')">Nuevo
Envío</a></span>';

```

```

        print '<span class="accion">(<a
href="'.$_SERVER['PHP_SELF'].'?accion=borrarlista&reparto='.$reparto['id'].'">Borrar</a>)</span>';

        print '<ul>';
        // Cogemos de la lista de reparto las tareas de envío
        $envios = $apitareas->tasks->listTasks($reparto['id']);

        // Por si no hay tareas de envío en la lista
        if (!empty($envios['items']))
            foreach ($envios['items'] as $envio) {
                // Creamos un elemento para cada una de las tareas de envío
                $idenvio = "". $envio['id']. "";
                print '<li title="'. $envio['notes']. '" id="'. $idenvio. '">'. $envio['title']. ' ('. $envio['notes']. ')';
                $coordenadas = "". $envio['notes']. "";
                print '<span class="accion"> (<a href="#"
onclick="abrirMaps('.$coordenadas. ');">Ver mapa</a>)</span>';
                print '<span class="accion"> (<a
href="'.$_SERVER['PHP_SELF'].'?accion=borrartarea&reparto='.$reparto['id'].'&envio='.$envio['id'].'">Borrar</a>)</span>';
                print '</li>';
            }
        print '</ul>';
        print '</div>';
    }
?>
</div>
<div class="pie">
    <?php print $error; ?>
</div>
</div>
</body>
</html>

```

[ajaxmaps.php](#)

```

<?php
/**
 * Desarrollo Web en Entorno Servidor
 * Tema 8 : Aplicaciones web híbridas
 * Ejemplo Rutas de reparto: ajaxmaps.php
 */

// Incluimos la librería Xajax
require_once("../libs/xajax_core/xajax.inc.php");

// Creamos el objeto xajax
$xajax = new xajax();

// Y registramos la función que vamos a llamar desde JavaScript
$xajax->register(XAJAX_FUNCTION,"obtenerCoordenadas");
$xajax->register(XAJAX_FUNCTION,"ordenarReparto");

// El método processRequest procesa las peticiones que llegan a la página
// Debe ser llamado antes del código HTML
$xajax->processRequest();

function ordenarReparto($coordenadasPuntosIntermedios)
{
    // Indicamos las coordenadas del almacén de donde sale la mercancía
    // Se podría añadir código para permitir al usuario indicarlo
    // o incluso coger por defecto la ubicación actual del usuario
    $coordenadasOrigen = "42.402497,-8.812001";

    // Se comienza y finaliza la ruta de reparto en el almacén
    $url = 'http://maps.google.es/maps/api/directions/json?origin='.$coordenadasOrigen;
    // Para obtener el resultado en XML en lugar de JSON, podríamos hacer:
    // $url = 'http://maps.google.es/maps/api/directions/xml?origin='.$coordenadasOrigen;
    $url .= '&destination='.$coordenadasOrigen;

    // Y se añaden los puntos de envío, indicando que optimice el recorrido
    $url .= '&waypoints=optimize:true|';
    $url .= $coordenadasPuntosIntermedios;
    $url .= '&sensor=false';

    // Como el resultado es JSON, lo procesamos de la siguiente forma
    $json = file_get_contents($url);
    $respuesta = json_decode($json);
}

```

```

    $orden = $respuesta->routes[0]->waypoint_order;

    // Si obtuviéramos un resultado en XML, habría que procesarlo de la siguiente forma
    /*
    $xml = simplexml_load_file($url);
    // Guardamos el recorrido óptimo calculado en un array
    foreach($xml->route[0]->waypoint_index as $parada) {
        $orden[] = (integer) $parada;
    }
    */

    // Y devolvemos el array obtenido
    $respuesta = new xajaxResponse();
    $respuesta->setReturnValue($orden);
    return $respuesta;
}

function obtenerCoordenadas($parametros)
{
    $respuesta = new xajaxResponse();
    $search =
'http://maps.google.com/maps/api/geocode/xml?address='.$parametros['direccion'].'&sensor=false
&appid=z9hiLa3e';
    $xml = simplexml_load_file($search);

    $respuesta->assign("latitud", "value", (string) $xml->result[0]->geometry->location->lat);
    $respuesta->assign("longitud", "value", (string) $xml->result[0]->geometry->location-
>lng);

    $respuesta->assign("obtenerCoordenadas", "value", "Obtener coordenadas");
    $respuesta->assign("obtenerCoordenadas", "disabled", false);

    return $respuesta;
}

?>

```

código.js

```

/**
 * Desarrollo Web en Entorno Servidor
 * Tema 8 : Aplicaciones web híbridas
 * Ejemplo Rutas de reparto: codigo.js
 */

// Indica si se está mostrando o no el diálogo de dirección / coordenadas
// para la introducción de un nuevo envío
var estadoDialogo = false;

function nuevoEnvio(idReparto) {
    $('#idrepartoactual').val(idReparto);
    mostrarDialogo();
}

function ordenarReparto(idReparto) {
    // Utilizamos jQuery para obtener una lista con las coordenadas de los puntos intermedios
    // que debemos ordenar
    // También se podrían haber almacenado por ejemplo en la sesión del usuario
    var paradas = $('#'+idReparto+' li').map(function() {
        return this.title;
    }).get().join('|');

    // Esta vez utilizamos el modo síncrono (esperamos a obtener una respuesta)
    var respuesta = xajax.request({xjxfun:"ordenarReparto"}, {mode:'synchronous', parameters:
[paradas]});
    if (respuesta) {
        // Si obtuvimos una respuesta, reordenamos los envíos del reparto
        // Cogemos la URL base del documento, quitando los parámetros GET si los hay
        var url = document.location.href.replace(/\?.*/,'');
        url = url.replace(/#$/,'');
        // Añadimos el código de la lista de reparto
        url += '?accion=ordenarEnvios&reparto='+idReparto;
        // Y un array con las nuevas posiciones que deben ocupar los envíos
        for(var r in respuesta) url += '&pos[]='+respuesta[r];

        window.location = url;
    }
}

```

```

    }
}

function getCoordenadas() {
    // Comprobamos que se haya introducido una dirección
    if($("#direccion").val().length < 10) {
        alert("Introduzca una dirección válida.");
        return false;
    }

    // Se cambia el botón de Enviar y se deshabilita
    // hasta que llegue la respuesta
    xajax.$('obtenerCoordenadas').disabled=true;
    xajax.$('obtenerCoordenadas').value="Un momento...";

    // Aquí se hace la llamada a la función registrada de PHP
    xajax_obtenerCoordenadas (xajax.getFormValues("formenvio"));

    return false;
}

function abrirMaps(coordenadas) {
    var url = "http://maps.google.com/maps?hl=es&t=h&z=17&output=embed&ll=";

    if(!coordenadas) {
        // Cogemos las coordenadas del diálogo
        url+=$("#latitud").val()+","+$("#longitud").val();
    }
    else {
        // Si hay coordenadas, las usamos
        url+=coordenadas;
    }

    window.open(url, 'nuevaventana', 'width=425,height=350');
}

function mostrarDialogo() {
    //Centramos en pantalla
    var anchoVentana = document.documentElement.clientWidth;
    var altoVentana = document.documentElement.clientHeight;
    var altoDialogo = $("#dialogo").height();
    var anchoDialogo = $("#dialogo").width();

    $("#dialogo").css({
        "position": "absolute",
        "top": altoVentana/2-altoDialogo/2,
        "left": anchoVentana/2-anchoDialogo/2
    });

    //Para IE6
    $("#fondonegro").css({"height": altoVentana});

    //Si no está visible el diálogo
    if(!estadoDialogo){
        // Se muestra el fondo negro
        $("#fondonegro").css({"opacity": "0.7"});
        $("#fondonegro").fadeIn("slow");
        // y el diálogo
        $("#dialogo").fadeIn("slow");

        $("#datosenvio").hide();
        estadoDialogo = true;
    }
}

function ocultarDialogo() {
    // Si está visible
    if(estadoDialogo){
        // Se oculta el fondo y el diálogo
        $("#fondonegro").fadeOut("slow");
        $("#dialogo").fadeOut("slow");
        estadoDialogo = false;
    }
}

```

estilos.css

```
@charset "utf-8";
body {
    font: 100%/1.4 Verdana, Arial, Helvetica, sans-serif;
    background: #699;
    margin: 0;
    padding: 0;
    color: #000;
}

h1, h2, h3, h4, h5, h6 {
    margin-top: 0;
    padding-left: 15px;
}

a:link {
    color:#414958;
    text-decoration: underline;
}
a:visited {
    color: #4E5869;
    text-decoration: underline;
}
a:hover, a:active, a:focus {
    text-decoration: none;
}

.accion {
    font-size: x-small;
}

.contenedor {
    min-width: 780px;
    background: #FFF;
    margin: 0 auto;
}

.encabezado {
    padding-top: 10px;
    background-color: #699;
}

.contenido {
    padding: 10px 0;
}

.contenido ul {
    padding: 0 15px 15px 40px;
    margin: 0;
}

.pie {
    color: #F00;
    padding: 10px 0;
    background: #699;
    position: relative;
    clear: both;
}

.titulo {
    padding-top: 5px;
    padding-right: 10px;
    padding-bottom: 0px;
    padding-left: 20px;
    font-size: large;
    font-weight: bold;
}

.clearfloat {
    clear:both;
    height:0;
    font-size: 1px;
    line-height: 0px;
}

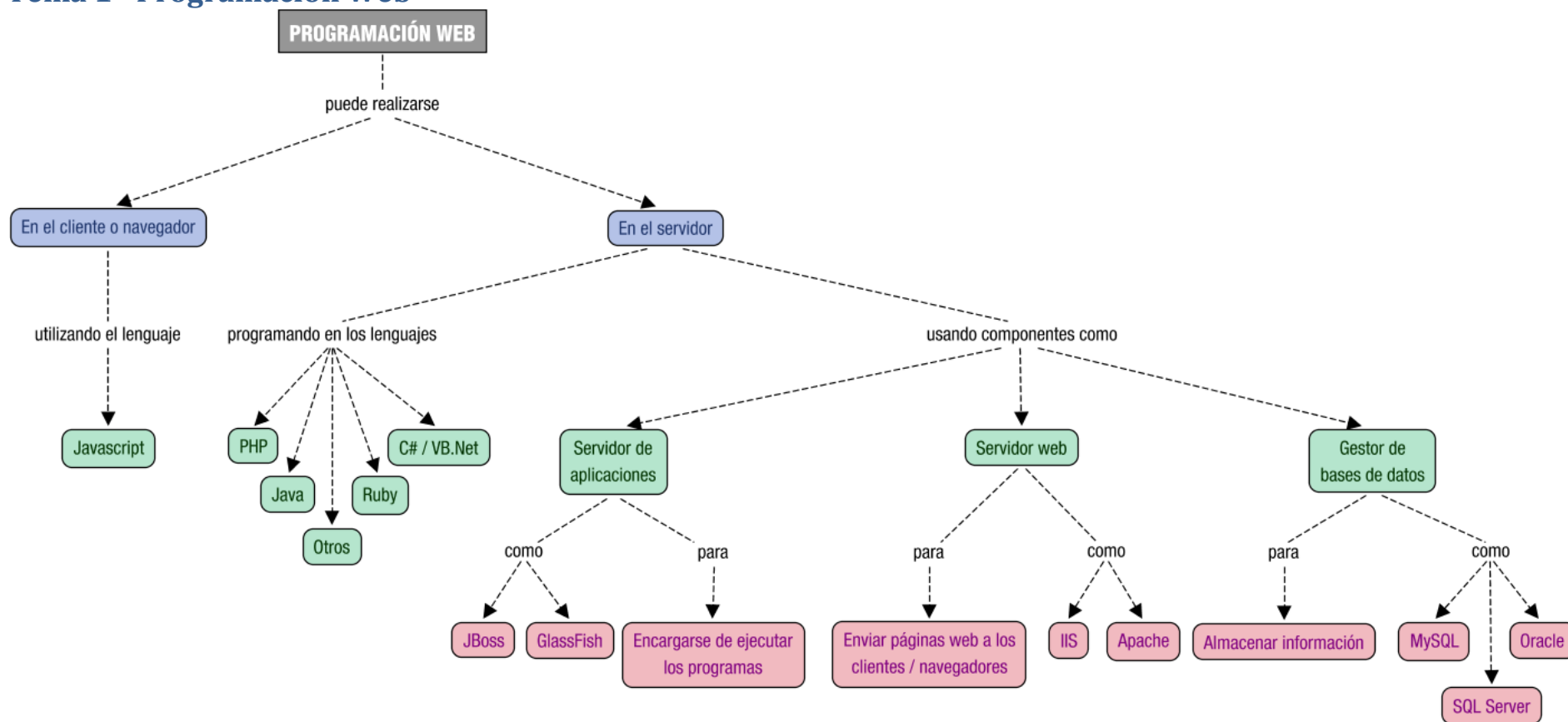
#dialogo {
    display:none;
    position:fixed;
    _position:absolute; /* para IE6*/
```

```
    height:320px;
    width:408px;
    background:#FFFFFF;
    border:2px solid #cecece;
    z-index:2;
    padding:12px;
    font-size:13px;
}
h1 {
    color:#039;
    font-size:24px;
    font-weight:700;
    padding-bottom:2px;
}

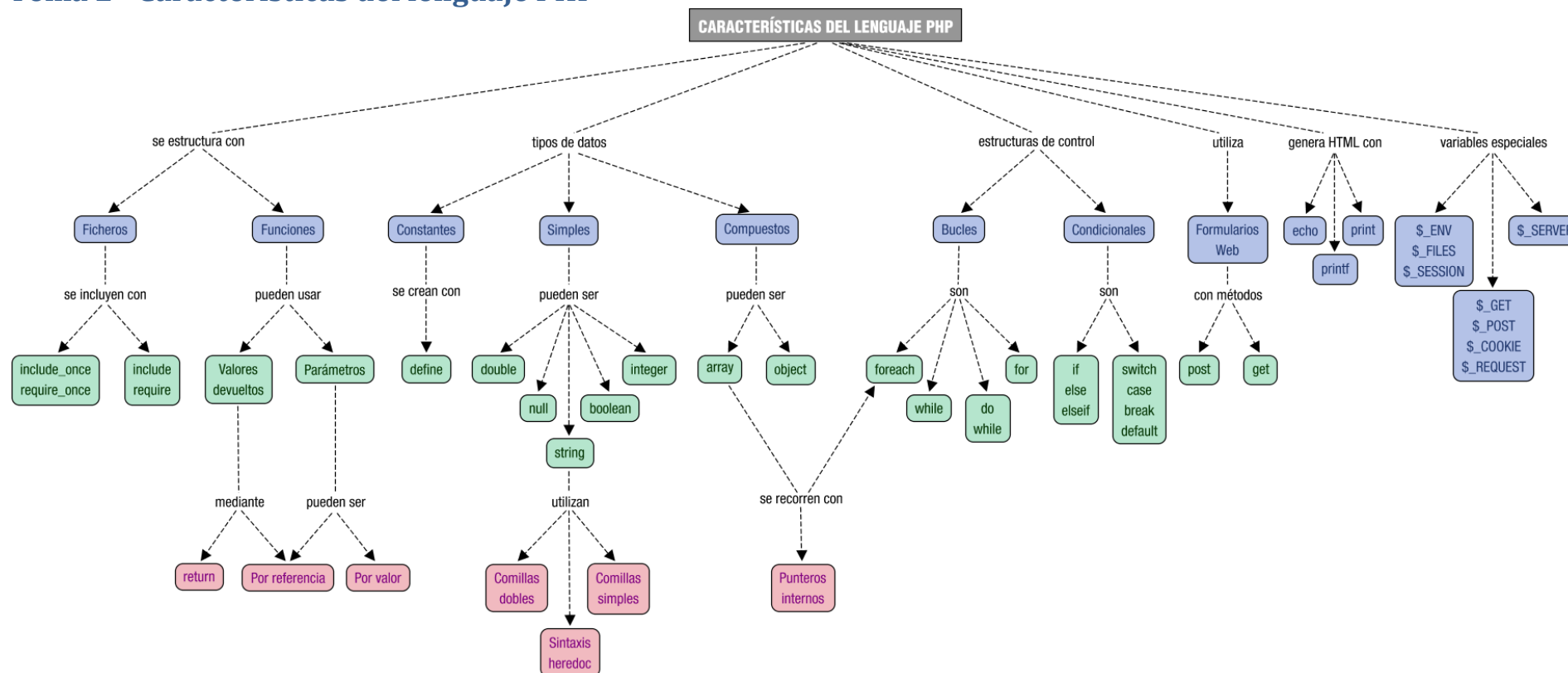
#cerrarDialogo {
    font-size:14px;
    line-height:14px;
    right:6px;
    top:4px;
    position:absolute;
    color:#6fa5fd;
    font-weight:700;
    display:block;
    cursor: pointer;
    text-decoration:none;
}

#fondonegro {
    display:none;
    position:fixed;
    _position:absolute; /* para IE6*/
    height:100%;
    width:100%;
    top:0;
    left:0;
    background:#000000;
    border:1px solid #cecece;
    z-index:1;
}
```

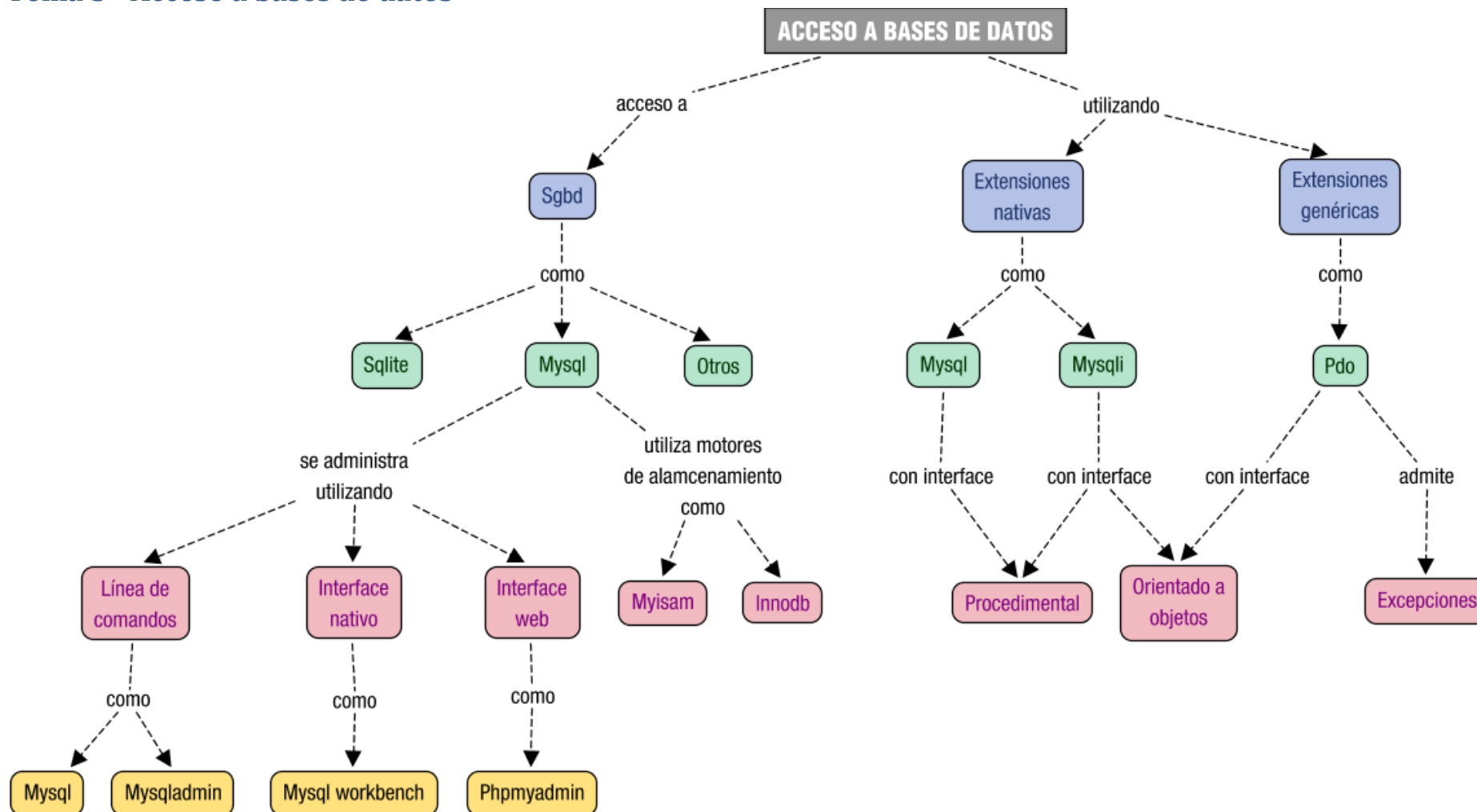
Tema 1 - Programación Web



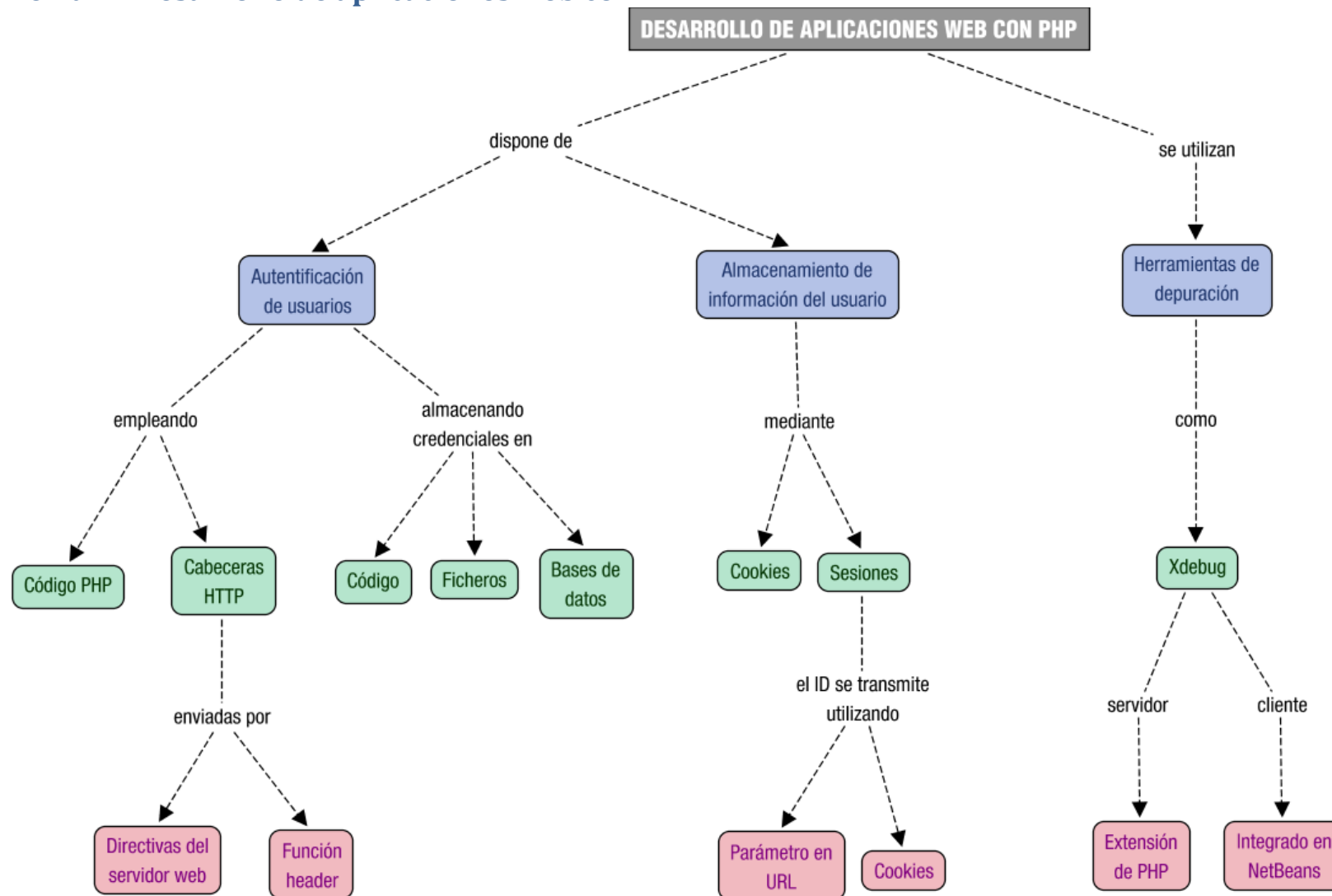
Tema 2 - Características del lenguaje PHP



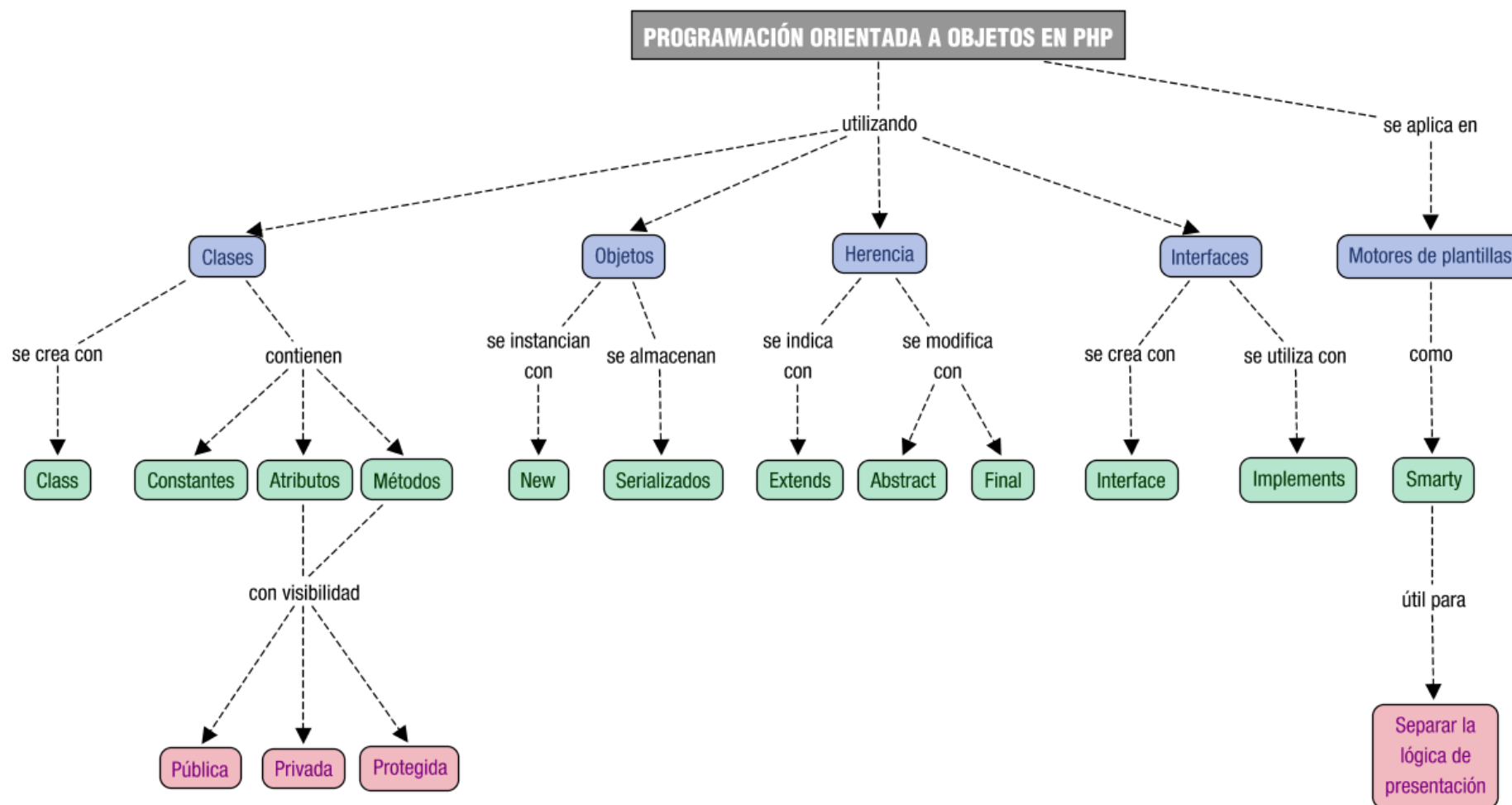
Tema 3 - Acceso a bases de datos



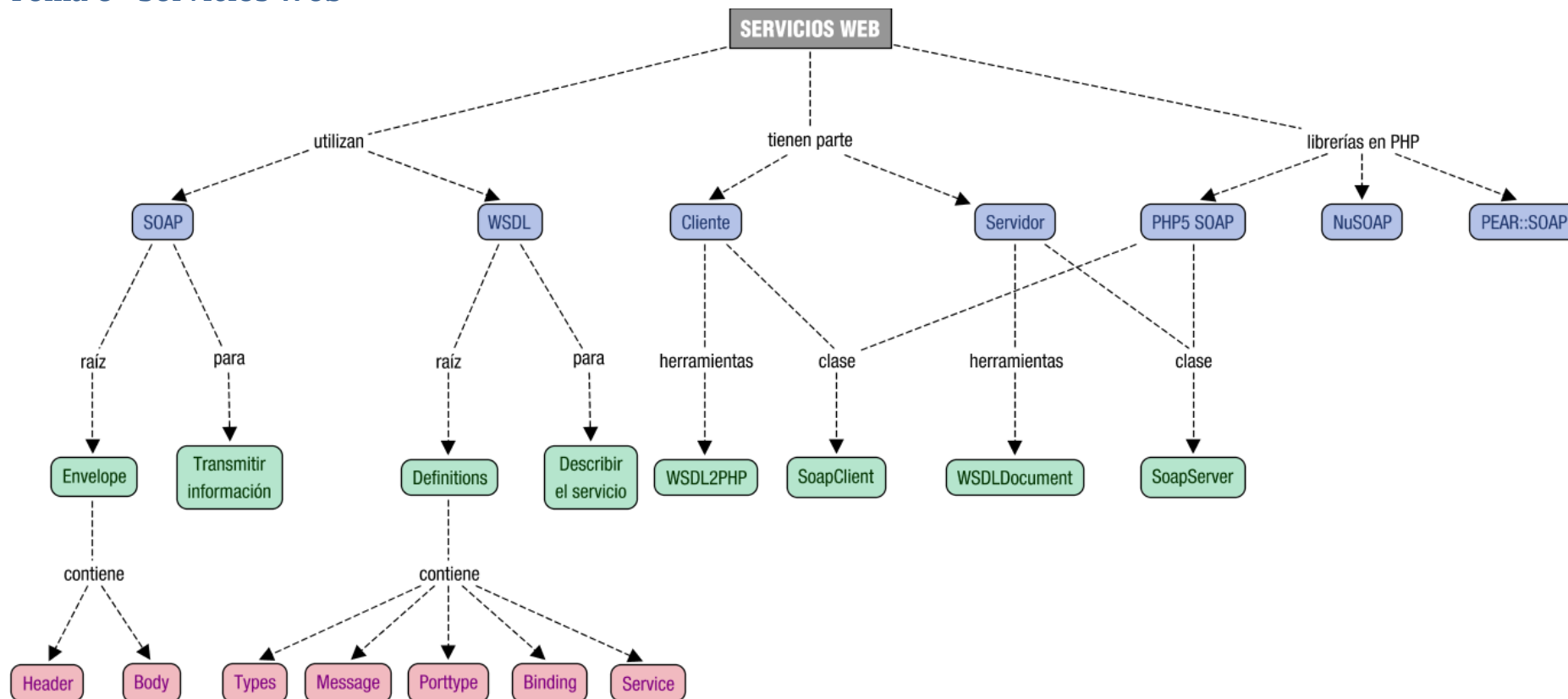
Tema 4 - Desarrollo de aplicaciones web con PHP



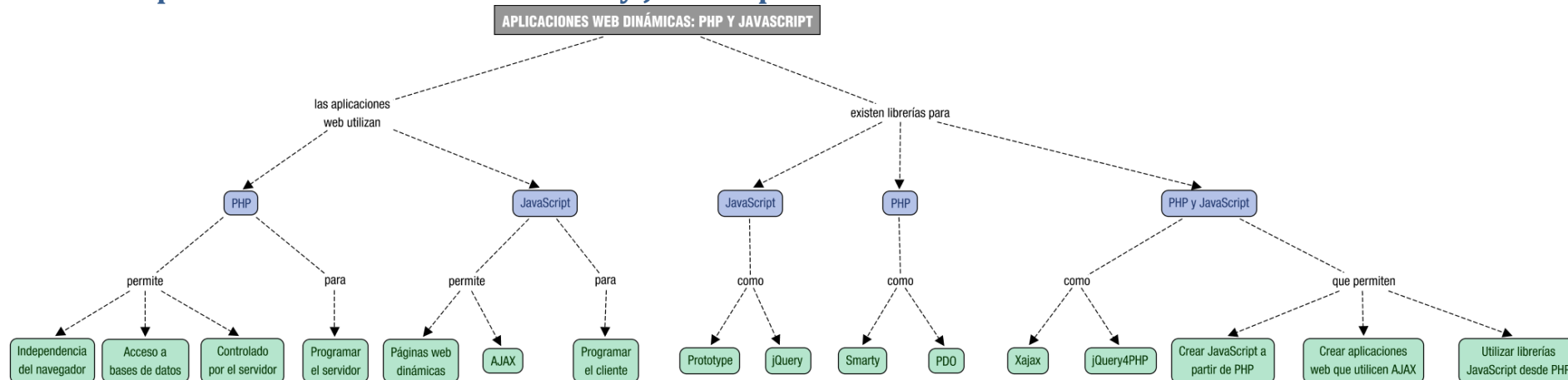
Tema 5 - Programación orientada a objetos en PHP



Tema 6 - Servicios Web



Tema 7 - Aplicaciones web dinámicas: PHP y JavaScript



Tema 8 - Aplicaciones web híbridas

