Bibliografía: PHP con acceso a bases de datos, Félix Mateos.

TEMA 1. INTRODUCCIÓN

Introducción a PHP y MySQL

Cuando escribimos una dirección en nuestro navegador web estamos solicitando que otro ordenador de Internet nos envíe un archivo con el contenido de la página que queremos ver. Nuestro ordenador está actuando como cliente de la aplicación de servidor web disponible en el otro ordenador.

Ese archivo debe estar codificado conforme a las normas del estándar HTML (XHTML), que es el que son capaces de interpretar los navegadores web.

Javascript, es un lenguaje de programación interpretado del lado del cliente (*client-side scripting*). Por ejemplo, en el siguiente archivo todo el código insertado entre los elementos <script> sería interpretado por la aplicación cliente cada vez que se accediese a la página, de modo que mostraría en cada acceso la hora configurada en el ordenador cliente.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es-ES" lang="es-ES">
   <head>
      <title>T&iacute;tulo de la p&aacute;gina</title>
      <meta http-equiv="content-type" content="text/html; charset=utf-8">
   </head>
   <body>
      <p>Ha accedido a las
            <script>
                var hora=new Date();
                document.write(hora.getHours()+':'+hora.getMinutes());
            </script>
      </p>
   </body>
</html>
```

Gracias al lenguaje PHP podremos insertar en los archivos HTML instrucciones que serán interpretadas por el servidor antes de enviar el archivo al cliente. Por ejemplo, el siguiente archivo mostraría la hora *del servidor* actualizada cada vez que se accediera a él.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es-ES" lang="es-ES">
   <head>
      <title>T&iacute;tulo de la p&aacute;gina</title>
      <meta http-equiv="content-type" content="text/html; charset=utf-8">
   </head>
   <body>
      <p>Ha accedido a las <?php echo date('G:i');?></p>
   </body>
</html>
```

Obsérva que se muestra la hora del servidor pues todo lo escrito entre los elementos `<?php` y `?>` es interpretado por el servidor antes de enviar el archivo al cliente Por este

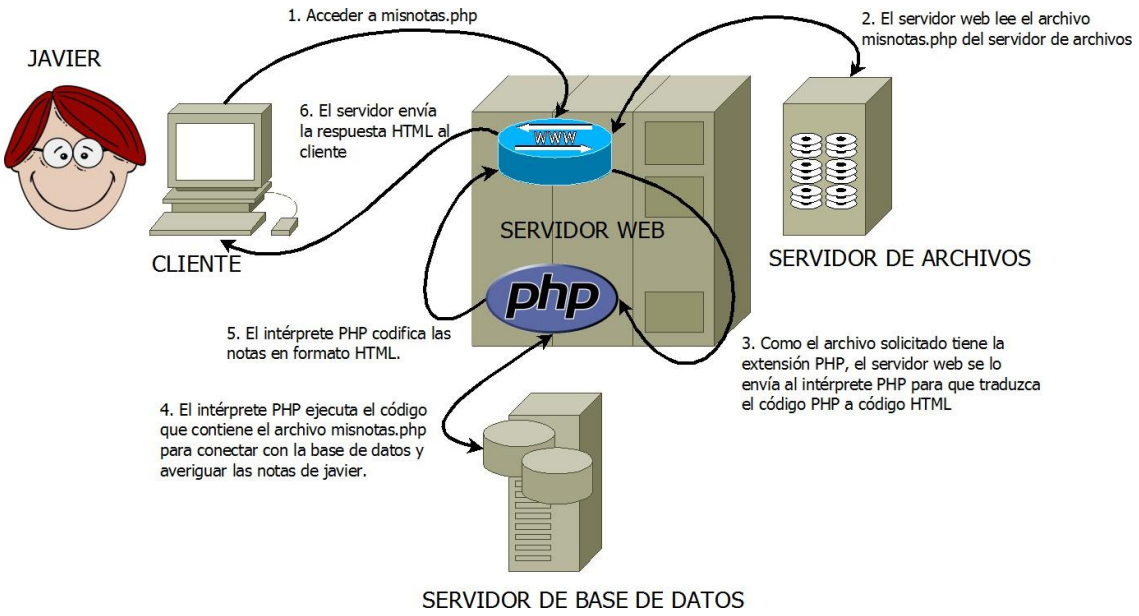
motivo PHP es un lenguaje interpretado del lado del servidor, es decir, *server-side scripting*.

¿En qué escenarios es conveniente utilizar PHP ? Justamente siempre que necesitemos manejar información sensible que no deseemos que esté al alcance del cliente.

MySQL es un sistema de gestión de bases de datos relacionales, multiusuario, multihilo y gratuito, pero cuya principal virtud es la compatibilidad con el lenguaje SQL (*Structured Query Language*).

PHP ha sido diseñado para facilitar la integración con MySQL, ofreciendo funciones muy cómodas para conectar con bases de datos MySQL y realizar consultas sobre ellas. La combinación entre PHP y MySQL conforma un tándem tan exitoso que ha sido elegido por sistemas tan relevantes como: Wikipedia, Joomla, Facebook, Moodle.

En la siguiente imagen se esquematiza el funcionamiento de PHP y MySQL.



PHP se creó en 1995 por Rasmus Lerdof(*Personal Home Page Tools*). La versión actual de PHP es la 8 (La anterior versión estable es la 7.4).

En 1995 Michael Widenius, David Axmark y Allan Larsson lanzaron la primera versión de su gestor de bases de datos relacionales MySQL. En 2008 fue adquirida por Sun Microsystems, que a su vez fue adquirida en 2010 por Oracle. La versión más reciente de MySQL en este momento es la 8.0.

Servidores web de desarrollo y explotación

Para crear un sitio web suelen utilizarse dos servidores: el de desarrollo y el de explotación.

El servidor de desarrollo es el que utilizan los programadores para desarrollar y depurar el código y, una vez que están convencidos de que funciona correctamente, lo trasladan al servidor de explotación.

Es importante que el servidor de desarrollo tenga unas característicasidénticas o lo más similares posible al de explotación para evitar problemas de compatibilidad.

Actualmente el servidor web más utilizado es Apache, sobre el que PHP suele instalarse como un módulo de éste.

XAMPP Server

Para configurar un ordenador normal como servidor web dinámico con acceso a bases de datos tendríamos que instalar e integrar en él los siguientes elementos:

- Una aplicación de tipo servidor web (la más popular es Apache).
- El intérprete PHP.
- El gestor de bases de datos relacionales (el más popular es MySQL, o MariaDB).

Existen distribuciones que "empaquetan" estos 3 elementos (Apache, PHP y MySQL), además de otros, en un único archivo de instalación, de modo que su instalación resulta tan sencilla como la de cualquier aplicación de escritorio. Estas distribuciones se conocen con el nombre genérico de distribuciones LAMP, por las siglas de Linux, Apache, MySQL y PHP/Perl/Python/Postgres y algunas de las variantes más populares son:

- XAMPP. Versión para Linux, Windows y MacOS.
- WAMP. Versión para Windows.
- MAMP. Versión para MacOs.

Hemos elegido crear nuestro servidor de desarrollo sobre Linux, e instalaremos XAMPP:

1. Si tenemos instalado apache lo desinstalamos:

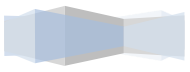
```
sudo service apache2 stop
```

```
sudo apt-get purge apache2 apache2-utils apache2.2-bin apache2-common

sudo apt-get autoremove

sudo rm -rf /etc/apache2
```

2. Descargamos el instalador de la página https://www.apachefriends.org



3. Desde la consola accedemos al directorio dónde hemos descargado el archivo y modificamos sus permisos con:

```
chmod +x xampp-linux-*-installer.run
```

4. Lanzamos el instalador

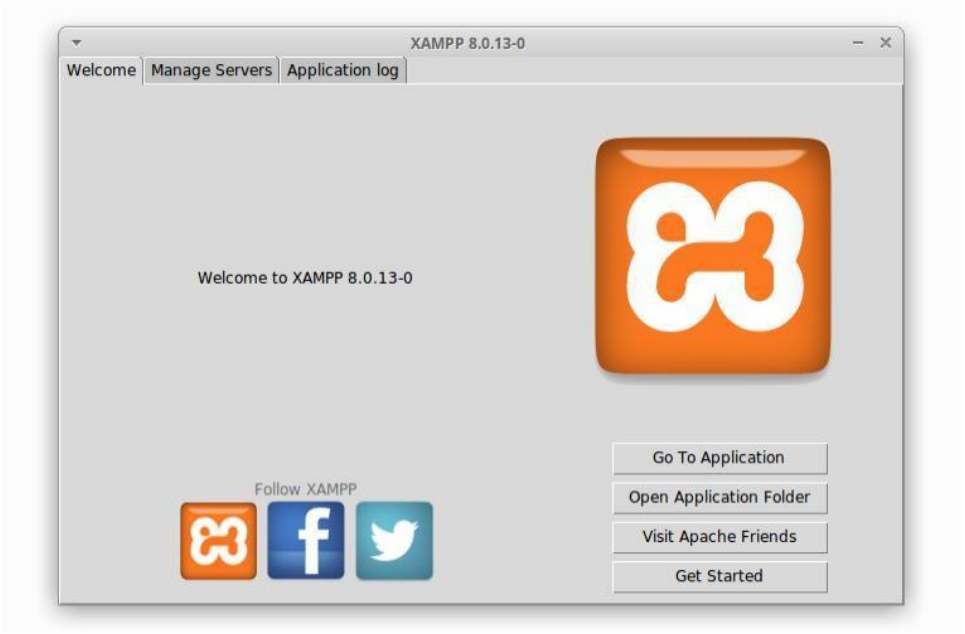
```
sudo ./xampp-linux-*-installer.run
```

5. Xampp estará instalado en el directorio /opt/lampp , desde ahí accederemos a todos los servicios.
6. Para arrancar tanto apache como mysql haremos:

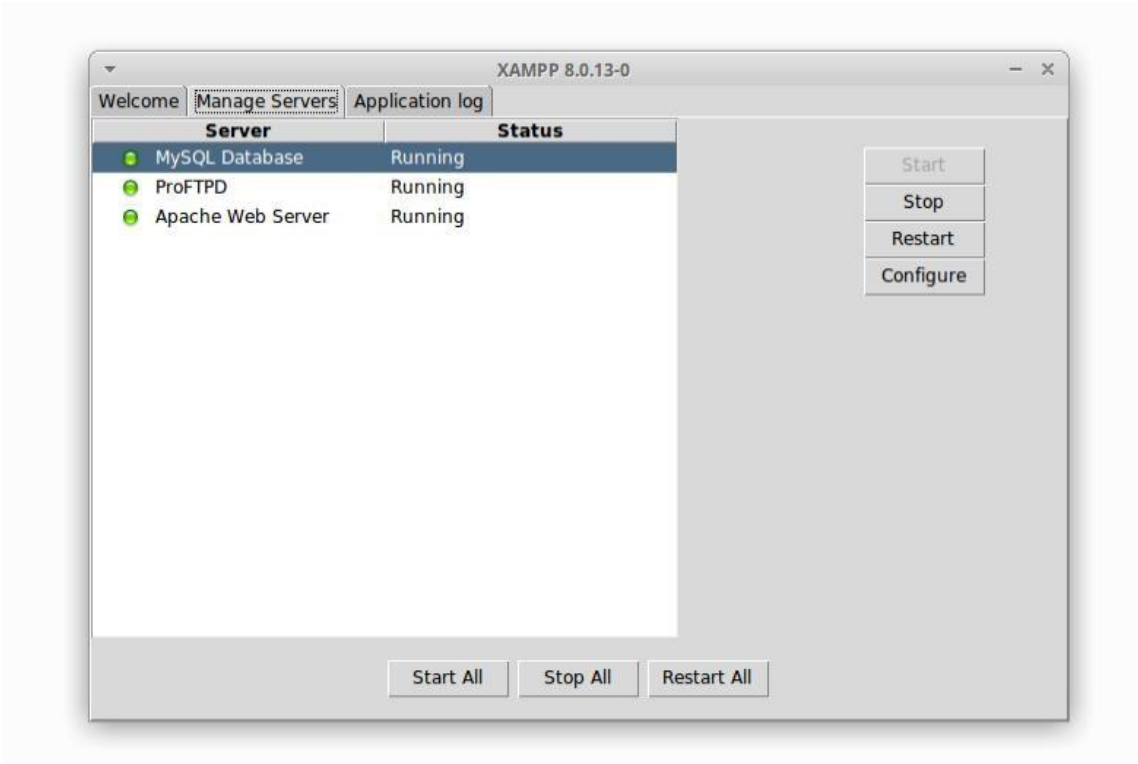
```
sudo /opt/lampp/lampp start
```

7. Si quiero que me lo ejecute en entorno gráfico, tendré que ejecutar en la consola en la ruta /opt/lampp el archivo

```
sudo ./manager-linux-x64.run
```

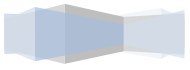


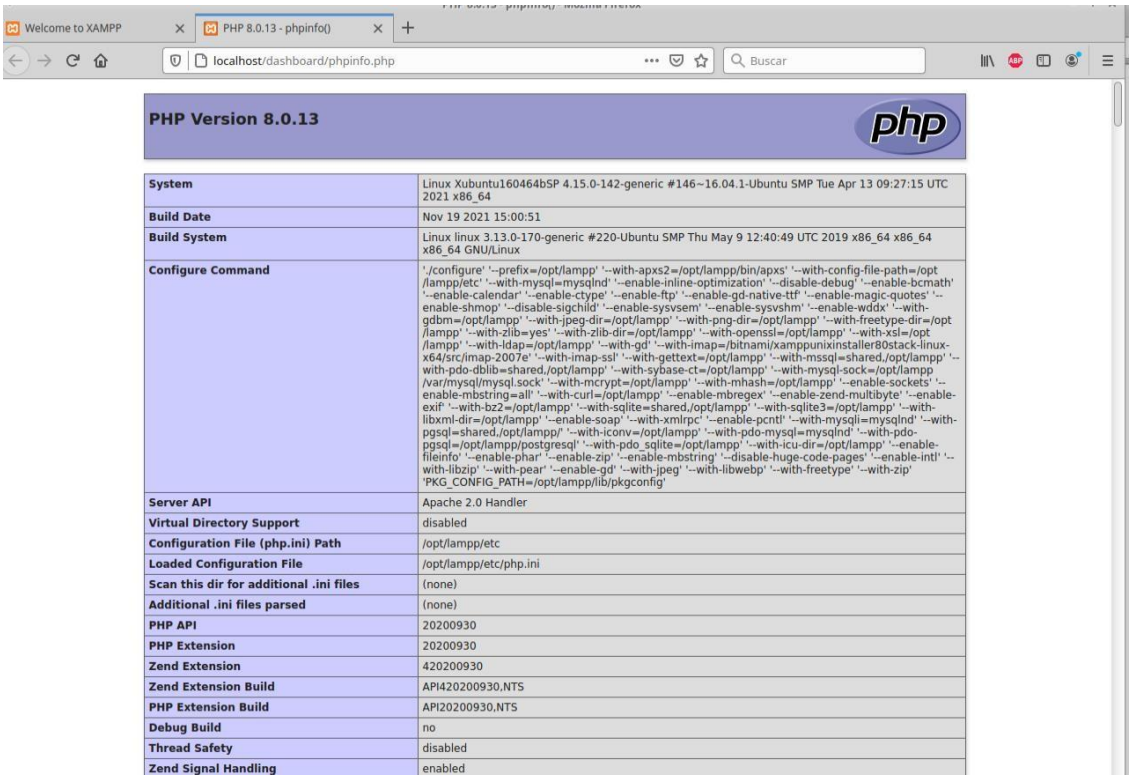
Desde aquí podemos acceder a la carpeta dónde está instalada la aplicación, ver en el log las distintas acciones producidas, apagar y encender todos los servicios, y acceder a los distintos archivos de configuración:



Si está corriendo MySQL y accedemos al navegador con localhost, se nos muestra la siguiente página:

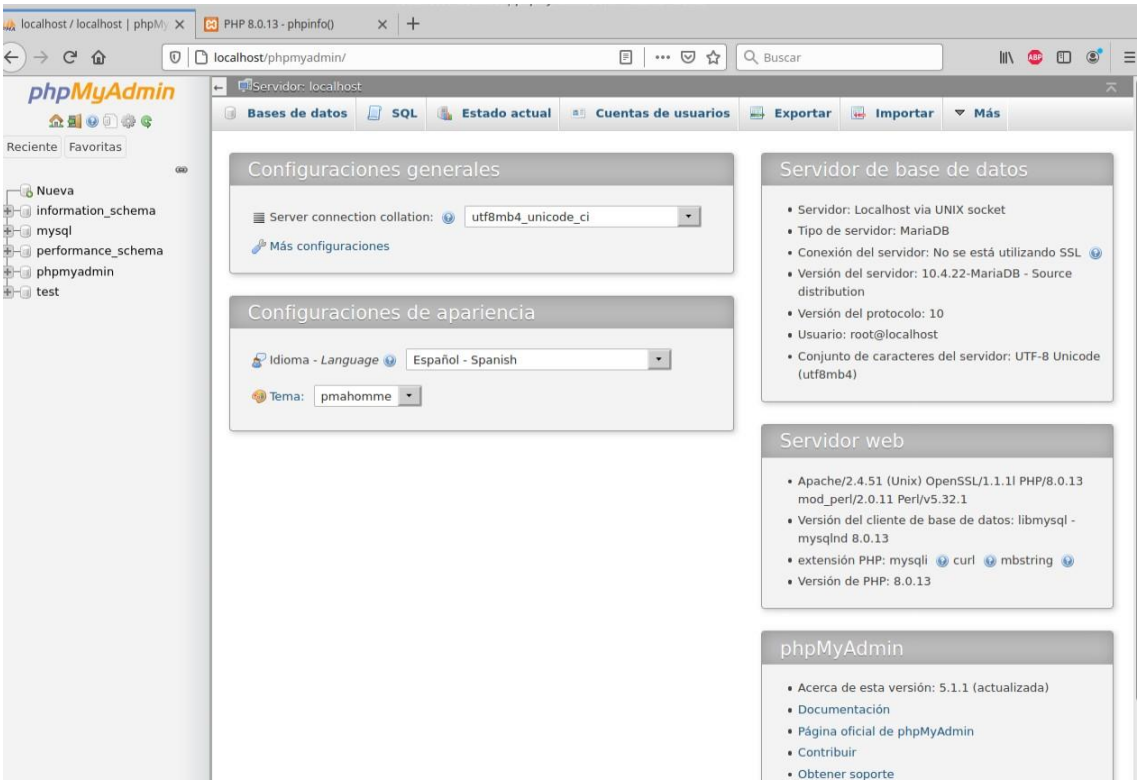
Accediendo a PHPInfo tenemos información sobre PHP:





System	Linux Xubuntu160464bSP 4.15.0-142-generic #146~16.04.1-Ubuntu SMP Tue Apr 13 09:27:15 UTC 2021 x86_64
Build Date	Nov 19 2021 15:00:51
Build System	Linux linux 3.13.0-170-generic #220-Ubuntu SMP Thu May 9 12:40:49 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
Configure Command	'./configure' '--prefix=/opt/lampp' '--with-apxs2=/opt/lampp/bin/apxs' '--with-config-file-path=/opt/lampp/etc' '--with-mysql=mysqlnd' '--enable-inline-optimization' '--disable-debug' '--enable-bcmath' '--enable-calendar' '--enable-ctype' '--enable-ftp' '--enable-gd-native-ttf' '--enable-magic-quotes' '--enable-shmop' '--disable-sigchild' '--enable-sysvsem' '--enable-sysvshm' '--enable-wddx' '--with-gdbm=/opt/lampp' '--with-jpeg-dir=/opt/lampp' '--with-png-dir=/opt/lampp' '--with-freetype-dir=/opt/lampp' '--with-zlib=yes' '--with-zlib-dir=/opt/lampp' '--with-openssl=/opt/lampp' '--with-xsl=/opt/lampp' '--with-ldap=/opt/lampp' '--with-gd' '--with-imap=/bitnami/xamppunixinstaller80stack-linux-x64/src/imap-2007e' '--with-imap-ssl' '--with-gettext=/opt/lampp' '--with-mssql=shared,/opt/lampp' '--with-pdo-dblib=shared,/opt/lampp' '--with-sybase-ct=/opt/lampp' '--with-mysql-sock=/opt/lampp/var/mysql/mysql.sock' '--with-mcrypt=/opt/lampp' '--with-mhash=/opt/lampp' '--enable-sockets' '--enable-mbstring=all' '--with-curl=/opt/lampp' '--enable-mbregex' '--enable-zend-multibyte' '--enable-exif' '--with-bz2=/opt/lampp' '--with-sqlite=shared,/opt/lampp' '--with-sqlite3=/opt/lampp' '--with-libxml-dir=/opt/lampp' '--enable-soap' '--with-xmlrpc' '--enable-pcntl' '--with-mysqli=mysqlnd' '--with-pgsql=shared,/opt/lampp/' '--with-iconv=/opt/lampp' '--with-pdo-mysql=mysqlnd' '--with-pdo-pgsql=/opt/lampp/postgresql' '--with-pdo-sqlite=/opt/lampp' '--with-icu-dir=/opt/lampp' '--enable-fileinfo' '--enable-phar' '--enable-zip' '--enable-mbstring' '--disable-huge-code-pages' '--enable-intl' '--with-libzip' '--with-pear' '--enable-gd' '--with-jpeg' '--with-libwebp' '--with-freetype' '--with-zip' 'PKG_CONFIG_PATH=/opt/lampp/lib/pkgconfig'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/opt/lampp/etc
Loaded Configuration File	/opt/lampp/etc/php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20200930
PHP Extension	20200930
Zend Extension	420200930
Zend Extension Build	API420200930,NTS
PHP Extension Build	API20200930,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled

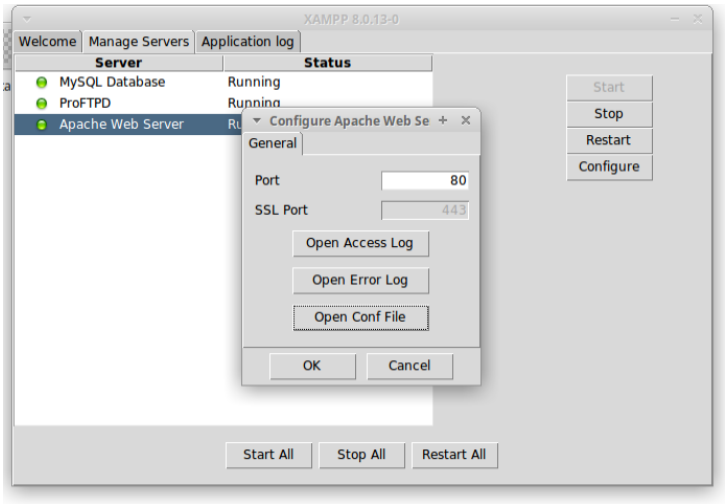
Y si vamos a phpMyAdmin, abrimos el gestor de base de datos de MySql:



En el que podremos gestionar nuestras bases de datos.

Configurar el funcionamiento de Apache: httpd.conf y .htaccess

El funcionamiento del servidor web Apache se configura en primera instancia a través de las directivas recogidas en el archivo httpd.conf, al que se puede acceder directamente desde el entorno gráfico, seleccionando Open ConfFile



El archivo se encuentra en la siguiente ruta: /opt/lampp/etc/httpd.conf.

Hay que tener en cuenta que este archivo se lee únicamente al arrancar Apache, de modo que si se introduce alguna modificación en él hay que reiniciar el servidor para que entre en efecto. El archivo httpd.conf está dividido en tres secciones principales:

1. Global environment. Las directivas de esta sección controlan el funcionamiento global de Apache, y algunas de las más interesantes son Listen (por defecto Apache escucha el puerto 80 de todas las interfaces de red disponibles en el ordenador, pero con la directiva `Listen` podemos indicarle que escuche sólo en algunas direcciones IP y/o puertos) y `LoadModule` (sirve para cargar módulos de Apache; por ejemplo, con esta directiva podemos cargar el módulo del intérprete PHP).
2. Main Server Configuration. En esta sección podemos encontrar directivas generales, y otras que se aplican a nivel de un directorio y todos sus subdirectorios descendientes (directiva `Directory`), o a nivel de un archivo (directiva `Files`). Algunas directivas interesantes que podemos encontrar en esta sección son `ServerAdmin` (indica el email del administrador que se ofrece en algunos mensajes de error para contactar con el webmaster), `DocumentRoot` (especifica qué carpeta de nuestro sistema de archivos queremos usar como raíz del servidor web, es decir, dónde están almacenadas nuestras páginas web), `DirectoryIndex` (especifica qué archivo del directorio debe servirse si se recibe una petición que no solicita ningún archivo concreto; generalmente se sirve por defecto el archivo index.php o, en su defecto, el archivo index.html), `Options +Indexes` (muestra un listado de los archivos que contiene una carpeta cuando se recibe una petición de una carpeta que no contiene ninguno de los archivos especificados por `DirectoryIndex`; cambiando el signo + por - impide que se muestre el

listado de contenidos), `AllowOverride` (nos permite especificar ciertas directivas para que puedan ser configuradas a nivel específico de cada carpeta mediante archivos .htaccess, que se explican a continuación).

3. Virtual Hosts. En esta sección podemos encontrar directivas similares a las de la sección anterior, pero anidadas dentro de directivas de tipo `VirtualHost`, de modo que sólo se apliquen a servidores virtuales concretos. Las directivas específicas de esta sección tienen precedencia sobre las directivas genéricas de la sección anterior.

Como se comentó anteriormente, mediante la directiva `AllowOverride` podemos indicar si queremos disfrutar de la posibilidad de configurar el funcionamiento de Apache a nivel de carpetas mediante archivos .htaccess. Este sistema nos permitirá incluir archivos llamados .htaccess en las carpetas de nuestro servidor web para establecer configuraciones específicas que anulen la configuración general establecida en httpd.conf. En otras palabras: si en httpd.conf indicamos mediante `AllowOverride` que queremos configurar alguna directiva concreta o todas a nivel de carpeta, el valor establecido en el archivo httpd.conf para esa directiva tendrá menor prioridad (será sobrescrito por) que el establecido a nivel de carpetas mediante archivos .htaccess. Además, los archivos .htaccess se heredan de carpetas a subcarpetas, de modo que las directivas establecidas en el .htaccess de una carpeta tendrán efecto en todas sus subcarpetas, a menos que alguna de ellas posea su propio .htaccess, en cuyo caso tendrán validez las directivas establecidas en él dentro de esa subcarpeta y todas sus descendientes. Los archivos .htaccess no se anulan al completo, sino a nivel de directivas; por ejemplo, si una carpeta posee un .htaccess en el que se configuran las directivas `Options` y `DirectoryIndex`, y una subcarpeta suya contiene otro .htaccess en el que se configuran las directivas `Options` (ya configurada a un nivel superior) y `ErrorDocument`, en esta subcarpeta y sus descendientes tendrán efecto las directivas `DirectoryIndex` del primer .htaccess, y las directivas `Options` y `ErrorDocument` del segundo .htaccess. Una diferencia sustancial entre los archivos .htaccess y el archivo httpd.conf es que los primeros se leen cada vez que se accede a una carpeta, no sólo cuando se arranca Apache como ocurre con httpd.conf, de modo que cualquier modificación realizada en un .htaccess entrará en vigor inmediatamente sin tener que reiniciar el servidor. Obviamente, el uso de archivos .htaccess supone una carga importante para el servidor, pues tiene que abrir e interpretar los archivos .htaccess de cada directorio y subdirectorio en cada petición. Por este motivo algunos servidores tienen impedido el uso de archivos .htaccess mediante la utilización de la directiva `AllowOverride None` en el archivo httpd.conf. Sin embargo, el escenario más frecuente es justamente el contrario, especialmente en servidores compartidos: en este caso como el propietario no puede permitir que cualquiera de los usuarios modifique el archivo httpd.conf, les ofrece cierta autonomía incluyendo en él directivas `AllowOverride` que les permitan configurar algunas directivas a través de archivos .htaccess.

Nota: En http://httpd.apache.org/docs/current/mod/quickreference.html puede encontrar un resumen de todas las directivas de configuración de Apache.

PRÁCTICA

En este ejercicio vamos a poner en práctica algunas directivas que nos conviene conocer como programadores de PHP.

1. Abre el archivo httpd.conf de XAMP y compruebe que la directiva `AllowOverride` impide en general el uso de archivos .htaccess (`AllowOverride None`), excepto a nivel de la carpeta opt/lampp/htdocs, en la que se permite reescribir cualquier directiva de httpd.conf mediante un .htaccess (`AllowOverride all`).

   Crea dentro de la carpeta htdocs una carpeta llamada padre y, dentro de ella, otra llamada hijo.

   Crea dentro de la carpeta padre dos archivos llamados index.html e index.php, que simplemente contengan el texto html y php respectivamente.

Si XAMP no está en ejecución, inícialo y accede con tu navegador a localhost/padre. ¿Qué archivo se ha abierto? ¿Por qué?

2. Crea dentro de la carpeta padre un archivo .htaccess con la directiva `DirectoryIndex index.php index.html`.

   Vuelve a acceder con tu navegador a la carpeta localhost/padre. ¿Qué archivo se ha abierto ahora? ¿Por qué?

3. Copia los archivos index.php e index.html a la carpeta hijo, y accede con su navegador a la dirección localhost/padre/hijo. ¿Qué archivo se abre? ¿Por qué?

4. Accede con tu navegador a la dirección localhost/padre/hijo/claves.txt. ¿Qué ocurre?

5. Añade al archivo .htaccess la directiva `ErrorDocument 404 /padre/error404.html` y crea en la carpeta un archivo llamado error404.html que simplemente contenga la frase "El archivo que ha solicitado no se encuentra en el servidor". Accede con su navegador nuevamente a la dirección localhost/padre/hijo/claves.txt. ¿Qué ocurre?

Nota: Otras páginas de error interesantes que podemos redirigir con `ErrorDocument` son 401 (requiere autorización), 403 (acceso prohibido), y 500 (error interno).

6. Renombra los archivos de la carpeta hijo como imagen.gif y documento.doc. Accede con su navegador a la dirección localhost/padre/hijo. ¿Qué ocurre?

7. Añade al archivo htaccess la directiva `IndexIgnore *.gif` y vuelva a acceder con el navegador a la dirección localhost/padre/hijo.¿Qué ocurre?

8. Añade al archivo htaccess la directiva `Redirect permanent /padre/hijo http://www.ibm.es` y vuelva a acceder con el navegador a la dirección localhost/padre/hijo. ¿Qué ocurre? ¿Para qué puede ser útil?

9. Otras directivas muy interesantes son `Allow from` y `Deny from`, que sirven para permitir o denegar el acceso a una carpeta (y sus subcarpetas) de todas las peticiones procedentes de cierta dirección IP o rango de direcciones. Por ejemplo `Allow from 207.123.45.23` permitiría el acceso desde esta dirección, mientras que `Allow from 207.123.45.` permitiría el acceso de cualquier ordenador de esa subred. Todas las directivas `Allow from` y `Deny from` del httpd.conf y los archivos htaccess se combinan y se procesan en el orden establecido por la directiva `Order`. Por ejemplo, con `Order Allow,Deny` se procesan primero todas las `Allow from` y si la IP de la petición no se encuentra en ninguna ellas se rechaza, luego se procesan todas las `Deny from` y si la IP se encuentra en alguna de ellas se rechaza; en otras palabras, una IP debe encontrarse en alguna de las `Allow from` y en ninguna de las `Deny from` para ser aceptada. De otra forma, se permiten las que están en Allow menos las que están en Deny. Por el contrario `Order Deny,Allow` procesa primero los `Deny from` y luego los `Allow from`, de modo que cualquier IP que no se encuentre en alguna de las `Deny from` será aceptada.Es decir , se deniegan todas las que están en Deny, excepto las que están en Allow. Como programadores PHP, las directivas `Order`, `Allow from` y `Deny from` pueden sernos muy útiles; por ejemplo, supongamos que acabamos de subir un sistema al servidor de explotación definitivo pero que antes de abrirlo al público queremos realizar pruebas desde cierto ordenador; podríamos usar las directivas `Order Allow,Deny` y `Allow from miIP` para permitir el acceso sólo desde miIP. Todas las peticiones rechazadas reciben el error 403 (forbidden); recuerda que puedes crear una página específica para este mensaje de error mediante la directiva `ErrorDocument`. Comprueba que cualquiera de sus compañeros puede acceder con su navegador a su carpeta padre. Esto es así porque en el archivo httpd.conf al no apareer las directivas anteriores, se asume: `Order Allow,Deny` y `Allow from All`. Elije la IP de uno de sus compañeros y añada al archivo htaccess las directivas `Order Deny,Allow`, `Deny from all` y `Allow from IP del compañero elegido`. Compruebe que ahora sólo ese compañero puede acceder a su carpeta padre. Introduce otra modificación en el archivo de configuración para que puedan acceder

a ese archivo todas las direcciones menos las de ese compañero. ¿Qué directivas has tenido que poner?

10. Otra posibilidad interesante de la configuración de Apache es proteger carpetas o archivos mediante contraseñas. Esto se consigue indicando en el httpd.conf o htaccess que deseamos utilizar un archivo de contraseñas .htpasswd. Este archivo contiene una línea para cada usuario autorizado, con la sintaxis `nombre_de_usuario:contraseña_encriptada`. La contraseña puede encriptarse con diversos algoritmos, pero se aplica por defecto la versión MD5, que consiste en una implementación de MD5 con *strectching* y *salting* propia de Apache. Introduce el siguiente código en su archivo htaccess para indicar dónde va a ubicar el archivo de contraseñas:

```
AuthType Basic
AuthUserFile "/opt/.htpasswd"
AuthName "USUARIO"
```

Nota: Por motivos de seguridad se recomienda ubicar el archivo .htpasswd por encima del `DirectoryRoot`, para que sea inaccesible a través del servidor.

A continuación, si quisiéramos proteger toda la carpeta padre simplemente añadiríamos la directiva `require valid-user`, pero en este ejemplo vamos a proteger sólo el archivo index.php, de modo que deberemos introducir el siguiente código:

```
<Files index.php>
require valid-user
</Files>
```

Ya sólo nos queda crear el archivo .htpasswd de contraseñas.
Crea un archivo con este nombre dentro de /opt y escribe en él una línea del tipo

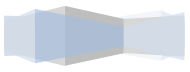
pepito: $apr1$s1r3m5c2$O4qk9./VvlD1iGP1kur3w.

(que corresponde al usuario pepito con la contraseña pepito), con tu nombre y primer apellido como usuario y contraseña la que quieras. Accede con tu navegador a la dirección padre/index.php para comprobar que Apache no le sirve este archivo a menos que se acredite convenientemente. Da un pantallazo y súbelo a la tarea.

Nota: Para crear tu propio usuario puedes ejecutar desde la línea de comando htpasswd -n nombre_usuario desde la carpeta bin de Apache, o bien recurrir a cualquier generador online, como Htpasswd Generator – Create htpasswd - hostingcanada.org. Si utilizas archivos htaccess y htpasswd es importante que te asegures de que está prohibido el acceso a ellos a través de un agente de usuario (navegador); esto se consigue incluyendo las siguientes directivas en el archivo httpd.conf (ya están incluidas en el de XAMP):

```
<FilesMatch "^\.ht">
    Order allow,deny
    Deny from all
</FilesMatch>
```

Nota: Una vez que nos hemos acreditado en un servidor Apache, el navegador suele enviar las credenciales en todas las cabeceras de las peticiones subsiguientes, de modo que nos evita tener que reescribirlas constantemente (en Firefox puede capturar las cabeceras mediante el complemento Live HTTP headers y verá una del estilo Authorization: Basic anVhbmZlOmp1YW5mZQ== que es la responsable de este comportamiento; si quiere borrar la información de autentificación seleccione **Herramientas>Limpiar el historial reciente>Sesiones activas**. Es aconsejable que instale este complemento y Web Developer en su navegador Firefox, pues le serán de utilidad a lo largo del curso.



11. Al modificar los archivos de configuración de Apache es posible que cometamos algún error y para ayudarnos a detectarlos podemos recurrir al archivo de registro de errores, accesible a través del menú de WAMP mediante **Apache>Apache error log**. Además, en este archivo también queda registrado cualquier intento de acreditación con unas credenciales erróneas, lo que puede ayudarnos a detectar ataques por fuerza bruta.
12. De forma similar, el archivo de registro de accesos, disponible a través del menú de WAMP mediante **Apache>Apache access log**, contiene un listado de todas las peticiones que ha recibido nuestro servidor y del código de respuesta:

    a. 200- OK
    b. 401 - No autorizado
    c. 403 - Acceso prohibido
    d. 404 - Documento inexistente
    e. 500 - Error interno del servidor

En la siguiente tabla se resumen las directivas de configuración de Apache que nos conviene conocer.

`Allow from`	Permite el acceso desde una IP o rango
`AllowOverride`	Permite sobrescribir directivas de httpd.conf a nivel de htaccess
`AuthName`	Indica al cliente qué credenciales se le están solicitando
`AuthType`	Permite elegir entre varios tipos de autentificación (Basic, digest, …)
`AuthUserFile`	Indica la ubicación del archivo de credenciales htpasswd
`Deny from`	Deniega el acceso desde una IP o rango
`Directory`	Agrupa directivas para un directorio concreto
`DirectoryIndex`	Indica qué archivo debe servirse por defecto
`DocumentRoot`	Indica la carpeta raíz del servidor web
`ErrorDocument`	Permite redirigir un mensaje de error a un archivo html
`Files`	Agrupa directivas de acceso para un fichero o ficheros concretos
`FilesMatch`	Tiene la misma utilidad que `Files` pero admite expresiones regulares para designar los archivos afectados
`IndexIgnore`	Sirve para evitar que ciertos archivos se muestren en los listados
`Listen`	Indica al servidor Apache en qué adaptadores de red y puertos debe escuchar
`LoadModule`	Sirve para cargar módulos de Apache
`Options [Indexes]`	Junto con `Indexes` permite configurar que se muestren o no los listados de archivos
`Order`	Establece en qué orden se procesan las directivas `Allow from` y `Deny from`

`Redirect [permanent]`	Permite redirigir un URL a otro de forma permanente
`require valid-user`	Obliga al usuario a acreditarse para acceder al URL
`ServerAdmin`	Indica el email de contacto del webmaster (aparece en algunos mensajes de error como el de tipo 500)
`VirtualHost`	Agrupa las directivas que atañen a un servidor virtual concreto

Configurar el funcionamiento de PHP: php.ini

En PHP existe un archivo llamado php.ini con directivas de configuración para el intérprete PHP y se ejecuta cada vez que se reinicia el servidor, en caso de que PHP esté funcionando como un módulo, o cada vez que se ejecuta un archivo PHP, en caso de que el intérprete esté funcionando como un CGI. (En phpinfo , podemos saber esto según e valor de **Server API**, si el intérprete PHP se está ejecutando como un CGI su valor será CGI/Fast CGI y si se está ejecutando como un módulo su valor será Apache Handler).

En XAMPP php.ini se encuentra en opt/lampp/etc.

En un servidor compartido generalmente no estaremos autorizados a modificar el archivo php.ini global.

Aparte de este php.ini global podemos encontrarnos con escenarios muy diversos, siendo éstos los más frecuentes:

- Algunos servidores (generalmente los que ejecutan PHP como CGI) permiten al usuario crear archivos php.ini personalizados (*custom php.ini*) a nivel de carpetas, pero las configuraciones establecidas en ellos casi siempre sólo son válidas para los archivos PHP contenidos en esa carpeta, no para sus subcarpetas.
- Algunos servidores si están ejecutando PHP como un módulo y tienen en httpd.conf la directiva `AllowOverride All` o al menos `AllowOverride Options`, el usuario podrá configurar el funcionamiento del intérprete PHP a través de directivas incluidas en los propios archivos .htaccess (aunque utilizando una sintaxis diferente a la de los archivos php.ini). No todas las directivas del php.ini son modificables en los archivos .htaccess.
- Existe una instrucción de PHP llamada `ini_set` que permite establecer el valor de directivas de configuración a nivel del script en el que están incluidas; en otras palabras: con esta instrucción podemos alterar la configuración del intérprete PHP temporalmente para la ejecución de un script concreto (al terminar su ejecución, se recupera automáticamente la configuración global del intérprete). No obstante, algunos servidores tienen deshabilitada la instrucción `ini_set`, bien explícitamente, o bien implícitamente porque tienen activado un modo del intérprete PHP llamado modo seguro (*safe mode*),. No todas las directivas del php.ini son modificables a través de `ini_set`.

Algunas directivas de configuración son:

- `display_errors`: Si esta directiva está activada y se produce un error en nuestro código PHP, el intérprete mostrará información sobre ese error en la pantalla del agente de usuario (el cliente). Es muy recomendable tener esta directiva activada en servidores de desarrollo, pero absolutamente desaconsejable en servidores de producción.

- `memory_limit`: Establece la cantidad de memoria máxima que puede consumir un script PHP en su ejecución. Lo complejidad de los scripts PHP ha provocado que el valor predeterminado de esta directiva haya aumentado con cada nueva versión de PHP, actualmente a 512M.
- `max_execution_time`: Establece el tiempo máximo en segundos que puede dedicar el intérprete PHP a ejecutar un script. Hay scripts especialmente complejos que requieren mucho tiempo, pero en lugar de cambiar este valor a nivel del php.ini global, quizás sea más aconsejable hacerlo mediante un `ini_set` en el propio script.
- `post_max_size`: Establece el tamaño máximo del conjunto de los datos recibidos de un formulario. Como existen formularios que permiten adjuntar archivos, esta directiva podría necesitar configurarse con un valor elevado. Como en el caso anterior, es mejor mantener un valor global bajo y aumentarlo en los scripts que lo necesiten mediante `ini_set`.
- `upload_max_filesize`: Establece el tamaño máximo para los archivos enviados a través de un formulario. Obviamente, por muy grande que sea el valor asignado a esta variable, siempre estará limitado por el de `post_max_size`. Por ejemplo, si `upload_max_filesize` tiene el valor 10M, pero `post_max_size` tiene el valor 2M., será imposible subir archivos de más de 2 MegaBytes.

Nota: El listado completo de las directivas de configuración de PHP se encuentra en http://www.php.net/manual/es/ini.list.php y en él puede consultar qué directivas son modificables a través de los archivos .htaccess y de la instrucción `ini_set`.

EJERCICIO

Vamos a aprender a configurar el funcionamiento del intérprete PHP mediante el archivo php.ini global, mediante archivos .htaccess y mediante instrucciones `ini_set`:

1. Añade al archivo index.php de tu carpeta padre el siguiente código que ejecuta la instrucción `phpinfo()`. Accede con su navegador a la dirección localhost/padre/index.php para ver el resultado. Fíjate en el valor de la directiva `max_execution_time`.

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

2. Modifica php.ini, en la carpeta etc. Localiza la línea correspondiente a la directiva `max_execution_time`, cambia su valor por 50 y guarda el archivo. Reinicia los servicios, accede nuevamente a la página localhost/padre/index.php y comprueba que ahora sí ha entrado en vigor el nuevo valor.
3. Introduce la directiva `php_value max_execution_time 75` en el archivo .htaccess de la carpeta padre. Observa que al configurar una directiva de PHP en un archivo .htaccess se utiliza una sintaxis diferente a la del archivo php.ini que consiste en anteponer la directiva `php_value`. Vuelve a acceder al archivo index.php y comprueba que ahora el valor local es 75 mientras que el global sigue siendo 50.
4. Modifica el archivo index.php para introducir la siguiente línea:

```
<?php
        ini_set("max_execution_time",150);
        phpinfo();
?>
```

Vuelve a acceder al archivo index.php y comprueba que ahora el valor local es 150 mientras que el global sigue siendo 50. En este caso la instrucción `ini_set` tiene precedencia sobre el archivo .htaccess, pero esto no siempre es así, recuerda que puede haber muchos escenarios diferentes, unos en los que no se permita

el uso de archivos php.ini personalizados, otros en los que no se permita el uso de archivo htaccess, otros en los que no se permita el uso de `ini_set`, etc…

Adaptar el servidor de desarrollo local a las características del servidor de explotación

Debemos adaptar la configuración del servidor de desarrollo local en el que vamos a llevar a cabo la programación a la del servidor de explotación.

Los tres niveles fundamentales de esta adaptación son:

- **Versiones**. Es conveniente que tengamos en nuestro servidor de desarrollo las mismas versiones de Apache, PHP y MySQL que el servidor de explotación.
- **Módulos**. Tanto Apache como PHP poseen módulos de ampliación. Por ejemplo, un módulo muy popular de Apache es rewrite_module, y uno muy popular de PHP es php_mysql. La información de `phpinfo()` nos indica cuáles son los módulos activos de PHP, pero para conocer los de Apache tendremos que contactar con la empresa de hosting o probar uno a uno si funcionan. Una vez hayamos averiguado cuáles son las extensiones habilitadas en el servidor de explotación, nos resultará muy fácil adaptar la configuración del servidor de desarrollo.
- **php.ini**. Una vez mimetizados los dos aspectos anteriores, ya sólo nos quedará editar el archivo php.ini del servidor de desarrollo para adaptar la configuración del intérprete PHP a la del servidor de explotación.

Nota: Cada versión de MySQL que instale tendrá sus propias bases de datos y tablas; en otras palabras: si ha creado varias bases de datos y tablas en una versión de MySQL, y cambia a otra versión de MySQL no espere encontrarlas en ella. Por este motivo es aconsejable establecer la versión de MySQL definitiva antes de empezar a trabajar con la base de datos.

Editando el fichero php.ini podemos ver cual es la mejor configuración para desarrollo o para producción.

```
;;;;;;;;;;;;;;;;;;;
; Quick Reference ;
;;;;;;;;;;;;;;;;;;;

; The following are all the settings which are different in either the production
; or development versions of the INIs with respect to PHP's default behavior.
; Please see the actual settings later in the document for more details as to why
; we recommend these changes in PHP's behavior.

; display_errors
;   Default Value: On
;   Development Value: On
;   Production Value: Off

; display_startup_errors
;   Default Value: On
;   Development Value: On
;   Production Value: Off

; error_reporting
;   Default Value: E_ALL
;   Development Value: E_ALL
;   Production Value: E_ALL & ~E_DEPRECATED & ~E_STRICT

; log_errors
;   Default Value: Off
;   Development Value: On
;   Production Value: On

; max_input_time
;   Default Value: -1 (Unlimited)
;   Development Value: 60 (60 seconds)
;   Production Value: 60 (60 seconds)
```