Capítulo 3
Utilizar las funciones PHP

1. Preámbulo

El objetivo de este capítulo es presentar las funciones más útiles para el desarrollo de un sitio web.

PHP ofrece numerosas funciones; la descripción de cada función está disponible en línea en el sitio www.php.net.

Versión 8

Desde la **versión 8**, es posible pasar parámetros a una función utilizando el nombre del parámetro en lugar de su posición. Esta funcionalidad se presenta en el capítulo Escribir funciones y clases PHP, pero puede utilizarse para las funciones propias del lenguaje PHP y, por tanto, para las funciones que se presentan en este capítulo. Sin embargo, en este capítulo, los nombres reales de los parámetros de las funciones no se presentan (están traducidos); para conocerlos, consulte la documentación en línea de las funciones…

Desde la **version 8.1**, pasar el valor `NULL` en un parámetro que no es explícitamente opcional es obsoleto, y por lo tanto, genera una alerta `E_DEPRECATED`.

Ejemplo

```
<?php
$x = null;
$n = strlen($x);
?>
```

Resultado

```
Deprecated: strlen(): Passing null to parameter #1 ($string) of type string
is deprecated in /app/scripts/index.php on line 3
```

2. Manipular las constantes, las variables y los tipos de datos

2.1 Constantes

PHP ofrece una serie de funciones útiles sobre las constantes:

Nombre	Función
`defined`	Indica si una constante está definida o no.
`constant`	Devuelve el valor de una constante.

defined

La función `defined` permite saber si una constante está definida o no.

Sintaxis

booleano `defined(`*cadena* `nombre)`

`nombre` Nombre de la constante.

La función `defined` devuelve `TRUE` si la constante está definida y `FALSE` en caso contrario.

Ejemplo

```
<?php
// Probar si la constante CONSTANTE está definida.
$ok = defined('CONSTANTE');
if ($ok) {
  echo 'CONSTANTE está definida.<br />';
} else {
  echo 'CONSTANTE no está definida.<br />';
};
// Definir la constante CONSTANTE
define('CONSTANTE','valor de la CONSTANTE');
// Probar si la constante CONSTANTE está definida.
$ok = defined('CONSTANTE');
if ($ok) {
  echo 'CONSTANTE está definida.<br />';
} else {
```

```
  echo 'CONSTANTE no está definida.<br />';
};
?>
```

Resultado

```
CONSTANTE no está definida.
CONSTANTE está definida.
```

constant

La función `constant` devuelve el valor de una constante cuyo nombre se pasa como parámetro.

Sintaxis

mixto constant(*cadena* nombre)

Donde

`nombre` Nombre de la constante.

Esta función es útil para recuperar el valor de una constante cuyo nombre no se conoce a priori.

Ejemplo

```
<?php
// definir el nombre de la constante en una variable
$nombreConstante = 'OTRA CONSTANTE';
// definir el valor de la constante
define($nombreConstante,'valor de la OTRA CONSTANTE');
// mostrar el valor de la constante
echo $nombreConstante,' = ',constant($nombreConstante);
?>
```

Resultado

```
OTRA CONSTANTE = valor de la OTRA CONSTANTE
```

Otras funciones permiten conocer el tipo de una constante (véase la sección Manipular las constantes, las variables y los tipos de datos - Tipos de datos).

2.2 Variables

PHP ofrece una serie de funciones útiles en las variables:

Nombre	Función
`empty`	Indica si una variable está vacía o no.
`isset`	Indica si una o varias variables están definidas o no.

Nombre	Función
unset	Elimina una o varias variables.
var_dump	Muestra la información sobre una o varias variables (tipo y valor).

empty

La función `empty` permite probar si una variable está vacía o no.

Sintaxis

```
booleano empty(mixto variable)
```

`variable` Variable que se va a probar.

`empty` devuelve TRUE si la variable está definida y FALSE en caso contrario.

Una variable se considera vacía si no ha sido asignada o si contiene una cadena vacía (""), una cadena igual a 0 ("0"), un 0, NULL, FALSE o una tabla vacía.

La función `empty` también se puede utilizar para probar si una expresión está vacía o no.

Ejemplo

```
<?php
// Prueba de una variable no inicializada.
$está_vacía = empty($variable);
echo '$variable no inicializada<br />';
if ($está_vacía) {
  echo '=> $variable está vacía.<br />';
} else {
  echo '=> $variable no está vacía.<br />';
}
// Prueba de una variable que contiene una cadena vacía.
$variable = '';
$está_vacía = empty($variable);
echo '$variable = \'\'<br />';
if ($está_vacía) {
  echo '=> $variable está vacía.<br />';
} else {
  echo '=> $variable no está vacía.<br />';
}
// Prueba de una variable que contiene una cadena igual a 0.
$variable = '0';
$está_vacía = empty($variable);
echo '$variable = \'',$variable,'\'<br />';
```

```
if ($está_vacía) {
  echo '=> $variable está vacía.<br />';
} else {
  echo '=> $variable no está vacía.<br />';
}
// Prueba de una variable que contiene 0.
$variable = 0;
$está_vacía = empty($variable);
echo '$variable = ',$variable,'<br />';
if ($está_vacía) {
  echo '=> $variable está vacía.<br />';
} else {
  echo '=> $variable no está vacía.<br />';
}
// Prueba de una variable que contiene una cadena no vacía.
$variable = 'x';
$está_vacía = empty($variable);
echo '$variable = \'',$variable,'\'<br />';
if ($está_vacía) {
  echo '=> $variable está vacía.<br />';
} else {
  echo '=> $variable no está vacía.<br />';
}
?>
```

Resultado

```
$variable no inicializada
=> $variable está vacía.
$variable = ''
=> $variable está vacía.
$variable = '0'
=> $variable está vacía.
$variable = 0
=> $variable está vacía.
$variable = 'x'
=> $variable no está vacía.
```

isset

La función `isset` permite probar si una o varias variables están definidas o no.

Sintaxis

`booleano` `isset(`*`mixto`* `variable[,...])`

`variable` Variable que se va a probar; pueden ser varias, separadas por una coma.

`isset` devuelve **TRUE** si la variable está definida y **FALSE** en caso contrario.

Si se facilitan varios parámetros, la función devuelve TRUE únicamente si se definen todas las variables.

Una variable se considera como no definida si no se ha visto asignada o si contiene NULL. A diferencia de la función empty, una variable que contiene una cadena vacía (""), una cadena igual a 0 ("0"), un 0, un FALSE o una tabla vacía, no se considera como no definida.

Ejemplo

```
<?php
// Prueba de una variable no inicializada.
$está_definida = isset($variable);
echo '$variable no inicializada<br />';
if ($está_definida) {
  echo '=> $variable está definida.<br />';
} else {
  echo '=> $variable no está definida.<br />';
}
// Prueba de una variable que contiene una cadena vacía.
$variable = '';
$está_definida= isset($variable);
echo '$variable = \'\'<br />';
if ($está_definida) {
  echo '=> $variable está definida.<br />';
} else {
  echo '=> $variable no está definida.<br />';
}
// Prueba de una variable que contiene una cadena igual a 0.
$variable = '0';
$está_definida = isset($variable);
echo '$variable = \'',$variable,'\'<br />';
if ($está_definida) {
  echo '=> $variable está definida.<br />';
} else {
  echo '=> $variable no está definida.<br />';
}
// Prueba de una variable que contiene 0.
$variable = 0;
$está_definida = isset($variable);
echo '$variable = ',$variable,'<br />';
if ($está_definida) {
  echo '=> $variable está definida.<br />';
} else {
  echo '=> $variable no está definida.<br />';
}
// Prueba de una variable que contiene una cadena no vacía.
$variable = 'x';
```

```
$está_definida = isset($variable);
echo '$variable = \'',$variable,'\'<br />';
if ($está_definida) {
  echo '=> $variable está definida.<br />';
} else {
  echo '=> $variable no está definida.<br />';
}
?>
```

Resultado

```
$variable no inicializada
=> $variable no está definida.
$variable = ''
=> $variable está definida.
$variable = '0'
=> $variable está definida.
$variable = 0
=> $variable está definida.
$variable = 'x'
=> $variable está definida.
```

unset

La función `unset` permite eliminar una o varias variables.

Sintaxis

```
unset(mixto variable)
```

`variable` Variable que se va a eliminar (para eliminar varias, deben estar separadas por una coma).

Después de la eliminación, la variable se encuentra en el mismo estado que si no hubiera sido asignada. El uso de la función `isset` en una variable eliminada devuelve FALSE.

Ejemplo

```
<?php
// Definir una variable.
$variable = 1;
// Mostrar la variable y probar si está definida.
$está_definida = isset($variable);
echo '$variable = ',$variable,'<br />';
if ($está_definida) {
  echo '=> $variable está definida.<br />';
} else {
  echo '=> $variable no está definida.<br />';
}
```

```
// Eliminar la variable.
unset($variable);
// Mostrar la variable y probar si está definida.
$está_definida = isset($variable);
echo '$variable = ',$variable,'<br />';
if ($está_definida) {
  echo '=> $variable está definida.<br />';
} else {
  echo '=> $variable no está definida.<br />';
}
?>
```

Resultado

```
$variable = 1
=> $variable está definida.
$variable =
=> $variable no está definida.
```

Observación

Al asignar un 0 o una cadena vacía a una variable, no se borra.

var_dump

La función `var_dump` muestra información sobre una o varias variables (tipo y contenido).

Sintaxis

```
var_dump(mixto variable, [,...])
```

`variable` Variable que se va a mostrar (pueden ser varias, separadas por una coma).

La función `var_dump` es especialmente interesante en las fases de desarrollo.

Ejemplo

```
<?php
// mostrar la información sobre una variable no inicializada
$variable = NULL;
var_dump($variable);
// inicializar la variable con un número entero
$variable = 10;
// mostrar la información sobre una variable
echo '<br />';
var_dump($variable);
// modificar el valor (y el tipo) de la variable
$variable = 3.14; // número decimal
```

```
// mostrar la información sobre una variable
echo '<br />';
var_dump($variable);
// modificar el valor (y el tipo) de la variable
$variable = 'abc'; // cadena de caracteres
// mostrar la información sobre la variable
echo '<br />';
var_dump($variable);
?>
```

Resultado

```
NULL
int(10)
float(3.14)
string(3) "abc"
```

Para una variable no inicializada, `var_dump` devuelve `NULL`. Para un número, `var_dump` indica el tipo (`int` = entero, `float` = número decimal), seguido por el valor entre paréntesis. Para una cadena, `var_dump` indica el tipo (`string`), seguido de la longitud entre paréntesis, seguido por el valor entre comillas.

PHP también ofrece las funciones `print_r` y `var_export`, que son similares a la función `var_dump`. La función `print_r` muestra o devuelve el contenido de la variable en una forma más legible, sin mencionar el tipo de datos. La función `var_export` muestra o devuelve una cadena que ofrece un código PHP de definición de la variable.

Observación

En la sección Tipos de datos de este capítulo, estudiaremos otras funciones que permiten determinar el tipo de una variable y realizar conversiones de tipos (de número a cadena, de cadena a número...).

2.3 Tipos de datos

2.3.1 Conversiones

PHP es capaz de realizar las conversiones automáticas implícitas de tipo.

Cuando un valor/expresión se asigna a una variable, la variable pasa a ser el tipo de valor/expresión.

Capítulo 3
WordPress y PHP

1. Introducción

WordPress es un CMS diseñado íntegramente en PHP que es un lenguaje orientado a objetos. Por lo tanto, es normal que WordPress haya desarrollado sus propias APIs, clases, métodos, funciones, etc. para simplificar la vida de los desarrolladores. Eso implica que como desarrollador, deberá familiarizarse con el funcionamiento de WordPress para poder aprovechar todo su potencial.

Este capítulo describe la preponderancia de PHP en WordPress, la estructura y funciones recurrentes de WordPress. Sirve como referencia para todo el libro. Si lo desea, puede omitirlo y consultarlo más adelante cuando personalice el archivo functions.php, cree temas avanzados o cuando cree extensiones, etc.

Este capítulo es un resumen de las posibilidades que ofrece WordPress. Se centra en lo esencial, así que no dude en consultar la documentación oficial, el códex o los numerosos tutoriales gratuitos disponibles en la Web, para profundizar en sus conocimientos sobre un tema específico.

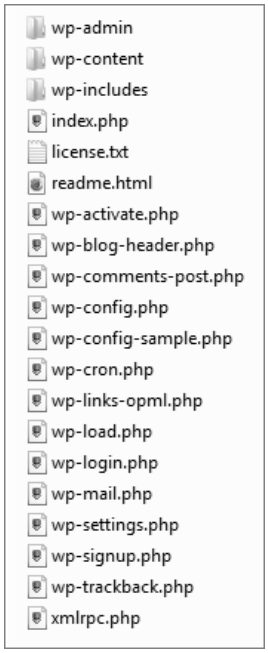
Todas las palabras clave y términos técnicos de cada capítulo le permiten realizar su propia investigación en la Red. Con una comunidad muy activa, sería una pena no aprovechar la experiencia de todas estas personas, dispuestas a ayudarle de forma gratuita. Encontrará muchos recursos, con muchos ejemplos de código.

2. La estructura de los archivos y carpetas WordPress

Es importante saber cómo se organizan los archivos en WordPress. La estructura es siempre la misma, tanto para un sitio web pequeño o para uno de mayor tamaño: el núcleo o corazón de WordPress no cambia.

2.1 Los archivos y carpetas en la raíz del sitio web

Todos los sitios de WordPress tienen la misma estructura. En la raíz, encontrará las carpetas wp-admin, wp-content, wp-includes y los archivos de WordPress.



Archivos y carpetas en la raíz del sitio web WordPress

No debe modificar o eliminar ningún archivo o carpeta nativo de WordPress. En el caso de aquellos que forman parte del core, es posible que el sitio web deje de funcionar. Además, la actualización de WordPress sobrescribe todos los cambios.

La carpeta **wp-admin** contiene todos los archivos relacionados con la administración del sitio web. No debe modificar archivos de esta carpeta.

La carpeta **wp-content** contiene todos los archivos por temas, extensiones y medios, idiomas, etc. En esta carpeta es donde se realizan las principales modificaciones y personalizaciones del sitio web.

La carpeta **wp-includes** contiene todos los archivos principales de WordPress (clases, funciones, scripts, etc.). No debe editar archivos de esta carpeta.

El archivo **wp-config.php** es un archivo generado durante la instalación de WordPress, a partir del archivo wp-config-sample.php. La información utilizada para configurar el sitio web se encuentra en este archivo: el nombre de la base de datos, la contraseña, la URL de la base de datos, el prefijo de las tablas, las claves de seguridad, la configuración del idioma principal del sitio, etc. Este archivo se modifica durante el cambio de host, la optimización del sitio web, la depuración, un cambio de idioma, etc. La eliminación del archivo wp -config.php vuelve a mostrar la página de configuración.

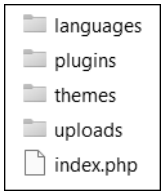
El resto de archivos .php se utilizan para el núcleo de WordPress y no se deben modificar ni eliminar en absoluto. Sin embargo, puede encontrar archivos en la raíz del sitio que hayan sido generados por un complemento, por WordPress o agregados por usted mismo. Por tanto, se pueden modificar algunos archivos: .htaccess, robots.txt o sitemap.xml.

2.2 La carpeta wp-content

La carpeta **wp-content** es la carpeta que se utiliza para crear temas, extensiones y funciones. Contiene las carpetas **languages**, **plugins**, **themes** y el archivo **index.php**.

Aparecen otras carpetas al agregar imágenes, como la carpeta **uploads** y durante las actualizaciones, como la carpeta **upgrade**.

Es posible que sigan apareciendo otras carpetas según la instalación de determinadas extensiones, como, por ejemplo, la carpeta **cache** y otras.



Carpetas y archivos en el directorio wp-content, al instalar WordPress

La carpeta **languages** contiene los archivos de idioma, temas y complementos de WordPress. La instalación manual se realiza descargando por FTP los archivos de idioma a esta carpeta y las subcarpetas relevantes. Los archivos de idioma tienen las extensiones .mo y .po/.pot.

La carpeta **plugins** contiene todas las extensiones instaladas. De base, la carpeta contiene dos extensiones: Akismet y Hello Dolly. La instalación manual se lleva a cabo descargando por FTP la carpeta que contiene los archivos de la extensión en la carpeta plugins. Todos los plugins se muestran en la administración, dentro de la pestaña **Plugins**.

La carpeta **themes** contiene todos los temas instalados. Hay tres temas básicos: Twenty Seventeen, Twenty Sixteen y Twenty Fifteen. La instalación manual se realiza descargando por FTP la carpeta que contiene los archivos y carpetas del tema en la carpeta themes. Todos los temas se muestran en la administración, dentro de la pestaña **Apariencia** - **temas**.

La carpeta **uploads** contiene todos los medios cargados. Para las imágenes, WordPress crea tres formatos de imagen y los clasifica según la configuración indicada en la administración, en la pestaña **Ajustes** - **Medios**. De forma predeterminada, WordPress clasifica los medios en carpetas con el nombre del año, conteniendo a su vez carpetas con el número de los meses (01, 02,…, 12). Algunas veces, ciertas extensiones crean carpetas para almacenar varios archivos. Todos los medios se muestran en la administración, pestaña **Medios**.

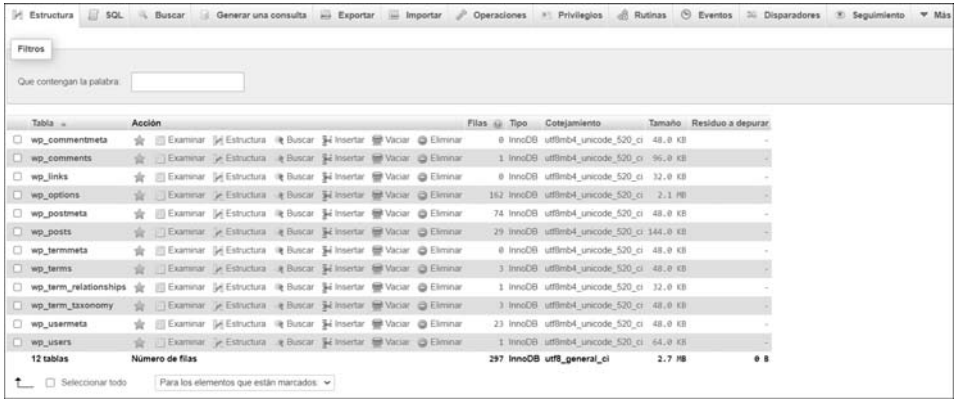
La carpeta **upgrade** aloja archivos comprimidos para actualizaciones de temas o extensiones. Antes de descomprimirlos, WordPress los descarga en esta carpeta. Si la actualización sale mal, encontrará el archivo comprimido en esta carpeta. Esta carpeta aparecerá cuando realice una actualización.

El archivo **index.php** solo se usa para la seguridad de la carpeta **wp-content**. Este archivo vacío se utiliza para mostrar una página en blanco en lugar del árbol del sitio web, en caso de un mal funcionamiento. Lo encontrará en diferentes carpetas.

3. La base de datos WordPress

WordPress instala doce tablas en la base de datos. Para ver esto, conéctese a su base de datos mediante phpMyAdmin. Aquí es donde se almacena toda la información sobre su sitio (contenido de artículos, páginas, categorías, etiquetas, configuración de WordPress, complementos, temas, información sobre los usuarios, etc.).

En este libro, los ejemplos se ilustran con el prefijo `msw_` (por defecto, WordPress usa el prefijo `wp_`). Normalmente, el prefijo se elige durante la instalación, por lo que podría ser diferente si lo cambia por razones de seguridad. Recuerde cambiar el prefijo cuando utilice los ejemplos.



Tablas en la base de datos durante la instalación de WordPress, en phpMyAdmin

Hay otras tablas que aparecen según los temas, extensiones y el uso del modo multisitio.

Observación

Referencia al códex: https://codex.wordpress.org/Database_Description

Aquí están las tablas nativas de WordPress, con el prefijo base wp:

– La tabla **wp_commentmeta** contiene información adicional sobre los comentarios. La utiliza la extensión **Akismet**. Se ha convertido en una extensión nativa de WordPress, imprescindible para evitar el spam. La encontrará en la administración en la pestaña **Plugins** y para la configuración, en la pestaña **Ajustes** - **Akismet Anti-Spam**.

– La tabla **wp_comments** contiene todos los comentarios de los artículos y las páginas.

– La tabla **wp_links**(opcional) agrupa todos los enlaces registrados a través de la pestaña **Enlaces** de la administración. La pestaña **Enlaces** ya no existe desde la versión 3.5 y requiere el uso de la extensión Link Manager, pero la tabla aún está presente para que las personas que la usaban antes no pierdan sus datos en las versiones recientes de WordPress. También puede habilitar el administrador de enlaces agregando el siguiente código al archivo functions.php:

```
add_filter('pre_option_link_manager_enabled',
'__return_true');
```

– La tabla **wp_options** contiene las configuraciones generales del sitio (introducidas durante la instalación del sitio) y extensiones, entre las más importantes. Durante la creación de temas avanzados, esta tabla también se usa para almacenar información, gracias a las funciones de WordPress.

– La tabla **wp_postmeta** contiene información adicional relacionada con los artículos o las páginas. Esta tabla está directamente relacionada con la tabla wp_posts.

– La tabla **wp_posts** es la más importante, ya que tiene todo el contenido del sitio. Encontrará toda la información sobre los artículos, páginas, fotos, archivos PDF y otros medios, o sobre los productos en el caso de las extensiones de comercio electrónico, así como el contenido de sus publicaciones personalizadas (custom post type), que aprenderemos a crear.

– La tabla **wp_termmeta** contiene los metadatos agregados a las categorías. Esta nueva tabla está disponible desde la versión 4.4.

– La tabla **wp_terms** contiene categorías y etiquetas. Esta tabla está directamente relacionada con las tablas wp_term_relationships y wp_term_taxonomy.
– La tabla **wp_term_relationships** vincula categorías y etiquetas a los diferentes artículos y páginas. Esta tabla está directamente relacionada con las tablas wp_terms y wp_term_taxonomy.
– La tabla **wp_term_taxonomy** se utiliza para diferenciar las categorías y las etiquetas. Allí encontrará la información adicional sobre las categorías y las etiquetas. Esta tabla está directamente relacionada con las tablas: wp_terms y wp_term_relationships.
– La tabla **wp_usermeta** contiene información adicional de todos los usuarios, así como su rol.
– La tabla **wp_users** contiene todos los usuarios, su contraseña, dirección de correo electrónico, etc.

Si olvida la contraseña, puede cambiarla directamente en su base de datos modificando la entrada después debe asignar al campo `user_pass` el valor de su nueva contraseña. No olvide poner el campo en `md5` antes de ejecutar la petición, lo que permite encriptar la contraseña por razones de seguridad. De esta forma, nadie podrá conocer la contraseña, ni siquiera una persona con acceso a la base de datos.

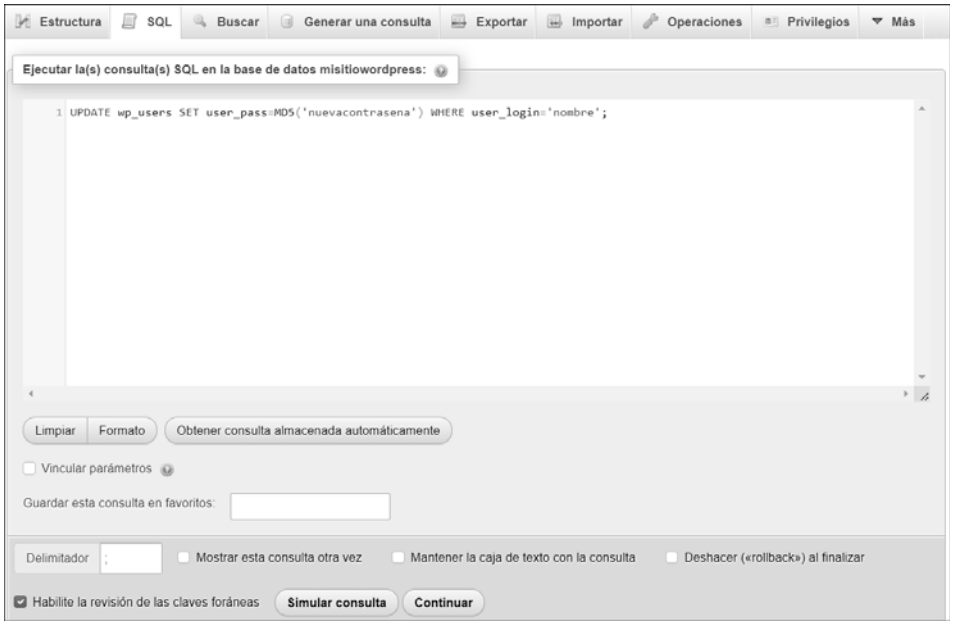
Observación

MD5(Message Digest 5) es una función de hash criptográfico que calcula un código digital único (huella digital), a partir de un archivo digital.

También puede ejecutar directamente la siguiente consulta SQL, cambiando primero los términos entre comillas (no olvide reemplazar el prefijo de la tabla si es diferente):

```
UPDATE wp_users SET user_pass=MD5('nuevacontraseña') WHERE
user_login='nombre';
```

En esta solicitud, se cambia "nuevacontraseña" por la contraseña que ha elegido y cambia "nombre" por el nombre de usuario.

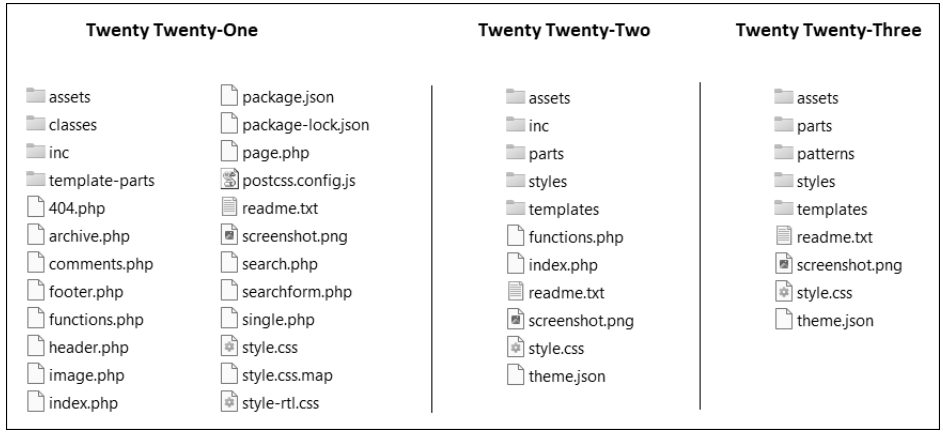


*Consulta en la pestaña **SQL** de phpMyAdmin*

4. La estructura y los archivos principales de un tema

Al instalar WordPress, hay tres temas presentes en el directorio **wp-content/themes**: Twenty Twenty-One, Twenty Twenty-Two, Twenty Twenty-Three, que son temas predeterminados desarrollados por WordPress.

Todos los temas tienen una estructura común que consta de archivos y carpetas principales, que están presentes en la mayoría de los temas. Se denominan archivos de plantilla o plantillas de página.



Archivos que componen los tres temas predeterminados de WordPress: Twenty Twenty-One, Twenty Twenty-Two, Twenty Twenty-Three

Observación

Referencia al códex: https://codex.wordpress.org/es:Template_Hierarchy

4.1 Los archivos principales

Entre los archivos en la raíz de los temas de WordPress, vamos a analizar el tema Twenty Twenty-One:

– Un archivo imagen **screenshot.png**: este archivo, específico a todos los temas, sirve como miniatura para el tema. WordPress detecta automáticamente la imagen y la muestra en la sección **Apariencia** - **Temas** de la administración.

– El archivo **style.css**: este archivo, específico a todos los temas, es el archivo CSS genérico de los temas, especialmente gracias al encabezado. En algunos temas, el código CSS se encuentra en una carpeta específica, a menudo una carpeta css. El archivo style.css está casi vacío, excepto por el encabezado.