 UNIVERSIDAD DON BOSCO	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERIA ESCUELA DE COMPUTACIÓN
CICLO 1	<p style="text-align: center;">GUIA DE LABORATORIO #6</p> <p>Nombre de la Practica: Gestión de archivos y directorios con PHP</p> <p>Lugar de Ejecución: Centro de Cómputo</p> <p>Tiempo Estimado: 2 horas con 30 minutos</p> <p>MATERIA: Desarrollo de Aplicaciones Web con Software Interpretado en el Servidor (DSS404)</p>

I. OBJETIVOS

Al desarrollar esta guía, el estudiante:

1. Comprenderá la importancia de la utilización de los archivos en aplicaciones web del lado del servidor.
2. Dominara las distintas funciones proporcionadas por el lenguaje PHP para la gestión de directorios y archivos.
3. Adquirirá habilidad en el manejo de archivos csv para gestionar información de forma organizada.
4. Logrará la habilidad para subir archivos al servidor desde el cliente realizando verificaciones de seguridad antes de procesar los datos.

II. INTRODUCCION TEORICA

Definición de archivo

Los **archivos** son una estructura de datos que se almacena en memoria secundaria y en la cual es posible guardar información de forma permanente.

Dentro del sistema de archivos del servidor los archivos son organizados en *directorios* o *carpetas*. Los *directorios* son un tipo especial de archivo creado para almacenar otros archivos. También pueden crearse jerárquicamente dentro de otros directorios, comenzando por el directorio raíz, conocido como *nivel superior*.

En los archivos se puede almacenar cualquier tipo de datos y también se puede almacenar mucha más información que la que puede soportar la memoria RAM. PHP facilita el trabajo con el sistema de archivos del servidor al incluir funciones que le permitan obtener información sobre los archivos, así como abrirlos, leerlos y escribir en ellos.



Trabajar con archivos en PHP

Al igual que muchos otros lenguajes de programación PHP proporciona una amplia gama de funciones para acceder al sistema de archivos del servidor con las cuales es posible desarrollar las operaciones típicas que se pueden realizar con archivos, como la **lectura** y la **escritura**, así como el **manejo de permisos**. Vamos a examinar las operaciones con archivos desde las más básicas hasta las más avanzadas, siendo estas:

- a) Abrir un archivo.
- b) Cerrar un archivo.
- c) Leer un archivo.
- d) Recorrer un archivo.
- e) Escribir en un archivo.

Abrir un archivo

Prácticamente la totalidad de las funciones del sistema de archivos carecen de operatividad sin la función `fopen()`, que es la función específica de PHP para abrir un archivo y que permite posteriormente, operaciones típicas como la lectura y la escritura sobre éste.

La sintaxis de la función `fopen()` es la siguiente:

```
resource fopen(  
string $filename, string $mode[, bool $use_include_path=false[, resource $context]]);
```

Donde, `$filename` es el nombre del archivo en disco que se desea abrir; `$mode`, representa el modo en el que la función lo abrirá (vea el detalle de los modos de apertura en la tabla que se muestra abajo); `$use_include_path`, puede ser usado si desea que PHP busque el archivo en la ruta definida en la directiva `include_path` del archivo de configuración `php.ini`. Hay que apuntar que el primer argumento, `$filename`, puede ser un nombre de archivo, una ruta local o una URL remota hacia el archivo. En el caso de tratarse del último caso, debe considerar que la directiva `allow_url_fopen` del **php.ini** debe estar habilitada para que PHP pueda abrir el archivo.

Con respecto al segundo argumento `$mode`, cabe mencionar que en PHP los archivos pueden ser abiertos en diversos modos. La siguiente tabla resume todos ellos:

Modo de apertura	Descripción
r	Abre el archivo en modo sólo lectura.
r+	Abre el archivo en modo lectura y escritura.
w	Abre el archivo en modo sólo escritura. Si el archivo no existe, se crea, y si ya existe, se sobrescribe.
w+	Abre el archivo en modo lectura y escritura. Si el archivo no existe, se crea, y si ya existe, se sobrescribe.
a	Abre el archivo en modo escritura. Si el archivo no existe, se crea, y si existe, le añade la nueva información.

Guía # 6: Gestión de archivos y directorios con PHP

a+	Abre el archivo en modo lectura y escritura. Si el archivo no existe, se crea, y si ya existe, la añade la nueva información.
x	
b	Abre el archivo en modo binario para lectura y escritura. Su aplicación es exclusiva para sistemas Windows, en donde se hace distinción entre archivos de texto y archivo binarios.

NOTA: A todos estos modos de apertura de archivos se les puede añadir las letras b o t, dependiendo de si el archivo es binario o de texto. Esto puede ser útil en sistemas en los que se distingue entre archivos binarios y de texto, como en Windows.

La función `fopen()` devuelve un recurso en caso de tener éxito, este recurso es un **descriptor de archivo** o un **puntero** o **referencia** a éste. Es recomendable asignar este valor devuelto a una variable, que a partir de ese momento se convertirá en la **referencia al archivo abierto** y con la cual deberá realizar las operaciones pertinentes de lectura y escritura.

En caso de producirse algún error al intentar abrir el archivo devolverá `False`. Si la apertura del archivo falla, se producirá un error de nivel `E_WARNING`, que puede ser ignorado haciendo uso del operador `@`.

Ejemplo:

```
//Abrir archivo en modo solo lectura
$fr = fopen("myfile.txt", 'r');
```

```
//Abrir archive en modo binario para lectura y escritura
$fb = fopen("myfile.txt", 'ba+');
```

```
//Abrir archivo en modo lectura y escritura, incluyendo la ruta definida en
include_path
$fw = fopen("myfile.txt", 'w+', true);
```

Cerrar un archivo

Siempre que se haya terminado de utilizar un archivo es recomendable cerrar las referencias que se hayan hecho sobre este a través de la función `fopen()`. Para tal efecto, debe utilizar la función `fclose()`, cuya sintaxis se muestra a continuación:

```
bool fclose(resource $handle);
```

La función `fclose()` recibe un descriptor de archivo como argumento y devuelve un valor booleano que será **True**, en caso de tener éxito o `False` en caso de producirse algún error.

Ejemplo:

```
$fr = fopen("listado.txt", 'w');
//Cerrar el archivo
fclose($fr);
```

Leer archivos

PHP proporciona muchas formas diferentes de leer un archivo y para cada una de estas formas se dispone de una o varias funciones que facilitan la tarea.

Leer un archivo carácter por carácter

Para realizar una lectura de archivo carácter a carácter se utiliza la función `fgetc()` que lee y devuelve un carácter a la vez de un archivo abierto. Devuelve falso en caso de alcanzar el final del archivo.

La sintaxis de la función es la siguiente:

```
string fgets(resource $handle);
```

La función `fgetc()` recibe un descriptor de archivo o un apuntador a éste y devuelve un solo carácter del archivo en forma de cadena de texto. Es preciso, si se va a leer todo el contenido de un archivo, utilizar esta función en conjunto con un ciclo repetitivo para acceder a todos los caracteres del archivo.

La sintaxis de la función `fgetc()` es la siguiente:

```
$filename = $_SERVER[DOCUMENT_ROOT] . "/examples/datos.dat";
$fh = fopen($filename, 'r');
while(!feof($fh){
    $car = fgetc($fh);
    if($car == '\n'){
        $car = "<br />";
    }
    print $car;
}
fclose($fh);
```

Leer archivos línea por línea

Existen dos funciones de PHP que permiten realizar la lectura de un archivo línea a línea, estas funciones son: `fgets()` y `fgetss()`. La diferencia que existe entre una y otra función es que la primera está diseñada para leer texto y la segunda para leer archivos html, de los que desea excluirse las etiquetas HTML.

La sintaxis de la función `fgets()` es la siguiente:

```
string fgets(resource $handle[, int $length]);
```

Donde, `$handle` es el manejador o descriptor de archivo del que se va a leer líneas y `$length` es un argumento opcional que se puede utilizar para indicar un número específico de bytes que se van a leer del archivo. La lectura puede terminar por una de tres razones:

- Se han leído ya `$length` bytes menos 1 (`$length-1`), en caso de que se haya proporcionado el segundo argumento,
- Se ha leído un carácter de fin de línea (`\n`), o
- Se ha llegado al final del archivo (EOF: End Of File).

Lo que suceda primero.

Ejemplo:

```
$filename = $_SERVER[DOCUMENT_ROOT] . "/examples/datos.dat";
$fh = fopen($filename);
while(!feof($fh)){
    $line = fgets($fh);
    print $line;
}
fclose($fh);
```

La sintaxis de la función `fgetss()` es la siguiente:

```
string fgetss(resource $handle[, int $length[, string $allowable_tags]]);
```

Esta función es en la práctica, igual que la función `fgets()`, la diferencia está en que de forma predeterminada elimina las etiquetas HTML en la operación de lectura del archivo. Si se desean conservar etiquetas HTML en la lectura, debe proporcionar en el tercer argumento la lista de etiquetas que no deberán eliminarse en la cadena devuelta por la función.

Ejemplo:

```
$filename = "pagina.html";
$fh = fopen($filename, 'r');
while(!feof($fh)){
    $content = fgetss($fh);
    echo $content;
}
fclose($fh);
```

Leer archivos una cantidad específica de bytes

Para leer de un archivo una cantidad específica de bytes puede hacer uso de la función `fread()`. Esta función trata a cualquier archivo que reciba como argumento como una secuencia de bytes, sin tener en cuenta finales de línea o cualquier otro tipo de carácter especial. En sistemas operativos de servidor en los que se diferencie entre archivos binarios y de texto, como Windows, el archivo debería ser abierto en modo binario ('b') al usar la función `fopen()`.

Si se va a leer el archivo completo, es aconsejable utilizar la función `filesize()` para obtener el número de bytes (o caracteres) del archivo y proporcionar el valor devuelto por esta función como argumento en la función `fread()`. La sintaxis de la función `fread()` es la siguiente:

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

Donde, el primer argumento `$handle` es el descriptor o manejador de archivo y el segundo argumento, `$length`, opcional, es el número de bytes que se van a leer del archivo. La función `fread()` devuelve los bytes leídos o el valor `False` en caso de que se haya producido algún error.

La lectura puede terminar por la primera de las tres condiciones siguientes que se llegue a producir:

- a) Se han leídos los `$length` bytes,
- b) Se ha alcanzado el final del archivo (EOF), o
- c) Un paquete se encuentra disponible o el tiempo límite del socket se agota (esto aplica para flujos de red).

Guía # 6: Gestión de archivos y directorios con PHP

Ejemplo:

```
$filename = "c:/wamp/www/examples/data.dat"; //Funcionará solo en Windows
$fr = fopen($filename, 'rt');
$contents = fread($fr, filesize($filename));
print "<pre>$contents</pre>";
fclose($fr);
```

En este otro ejemplo se ilustra cómo puede utilizarse la función `fread()` para leer un archivo de imagen:

```
$filename = "c:\\wamp\\www\\examples\\images\\tulips.jpg";
# $filename = "c:/wamp/www/examples/images/tulips.jpg";
$fh = fopen($filename, "rb");
$contents = fread($fh, filesize($filename));
header("Content-type: image/jpeg"); //Para indicar al navegador que lo que se va a
mostrar es una imagen
echo $content;
fclose($fh);
```

Acceso aleatorio a los datos del archivo

Hasta ahora, la forma en que se ha accedido a los archivos, ha sido de forma secuencial, esto es leer los datos del archivo en el orden en que están dispuestos. Sin embargo, en la práctica puede encontrarse con la necesidad de acceder a un dato específico que no necesariamente esté al inicio del archivo. Existe la posibilidad de mover el indicador de posición en el archivo de modo que se pueda comenzar a leer o a escribir en cualquier punto del archivo. A esto se le llama también recorrer un archivo.

Las funciones de PHP que se utilizan para este propósito son las siguientes:

- a) `fseek()`: Desplaza el indicador de posición a un punto específico en el archivo.
- b) `rewind()`: Mueve el indicador de posición al inicio del archivo.
- c) `ftell()`: Devuelve la posición actual del indicador de posición del archivo.

La sintaxis de la función `fseek()` es la siguiente:

```
int fseek(resource $handle, int $offset[, int $whence=SEEK_SET]);
```

En donde, `$handle` es el manejador de archivo, `$offset` establece el desplazamiento a partir de la posición indicada por `$whence`, cuando se proporcione, si no se proporciona, el desplazamiento dentro del archivo se hará a partir del inicio del archivo. El valor del tercer argumento, `$whence`, puede ser:

- a) `SEEK_SET`: Establece el indicador de posición al principio del archivo, más el desplazamiento indicado en el segundo argumento de la función `fseek()`, `$offset`.
- b) `SEEK_CUR`: Establece el indicador de posición en la posición actual dentro del archivo abierto más el desplazamiento especificado en `$offset`.
- c) `SEEK_END`: Establece el indicador de posición al final del archivo, más el desplazamiento establecido en el argumento `$offset`. Se utiliza con desplazamientos negativos para recorrer el archivo en forma inversa.

Guía # 6: Gestión de archivos y directorios con PHP

Los valores devueltos por `fseek()` serán 0 si el indicador se ha posicionado correctamente, o -1 se ocurrió algún problema.

La función `rewind()` se utiliza para regresar el indicador de posición dentro del archivo al inicio. Su sintaxis es la siguiente:

```
bool rewind(resource $handle);
```

Donde, `$handle` es el descriptor o manejador de archivo, y el valor devuelto por la función es un valor booleano que será `True`, si se hace la operación con éxito, o `False`, si se produce algún error.

La función `ftell()` se puede utilizar para una de tres cosas, obtener la posición en bytes actual del indicador de posición en un archivo sobre el que se ha realizado una operación de lectura, el número de bytes desde el principio del archivo en un determinado momento o la posición, en bytes, desde dónde comenzará la siguiente operación de lectura.

La sintaxis de la función `ftell()` es la siguiente:

```
int ftell(resource $handle);
```

La función recibe como argumento el descriptor o manejador de archivo y devuelve un valor entero con la posición del puntero al archivo referenciado. Si se produce algún error, devolverá `False`.

Trabajar con directorios

También es posible trabajar con directorios en PHP de la misma forma que con archivos, para lo cual se proporcionan una gran variedad de funciones equivalentes. Algunas de estas funciones utilizan un indicador de directorio, mientras que otras utilizan una cadena que contiene el nombre del directorio con el que se va a trabajar.

Un indicador de directorio es una variable especial que apunta a un directorio. Para obtener el indicador de directorio se puede utilizar la función `opendir()`.

```
$dh = opendir("/home/files"); //Funcionará en sistemas Unix/Linux
```

Al igual que con las operaciones con archivos, también existe una función para cerrar un directorio. Esa función es `closedir()`, que recibe como argumento el indicador de directorio.

```
closedir($dh);
```

Además de estas funciones está la función `readdir()` que se utiliza para obtener el nombre del siguiente archivo leído desde el directorio abierto con `opendir()`. Los nombres de archivo son devueltos en el orden en que son almacenados por el sistema de archivos. Cada directorio contiene una lista de entradas para cada uno de los archivos y subdirectorios que posee, así como dos entradas especiales que son: `.`, que representa el directorio y `..`, que representa el padre del directorio. PHP mantiene un puntero interno que hace referencia a la siguiente entrada de la lista, del mismo modo que un puntero de archivo apunta a la posición en un archivo donde debería ocurrir la siguiente operación de archivo.

Ejemplo:

```
$dirpath = "/home/files/images";  
if(!$handle = opendir($dirpath)) die("No se puede abrir el directorio");  
echo $dirpath . " contiene los siguientes directorios y archivos: <br />";
```

Guía # 6: Gestión de archivos y directorios con PHP

```
echo "<ul>";
while($file = readdir($handle)){
    if($file != "." && $file != "..") echo "<li>$file</li>";
}
echo "</ul>";
closedir($handle);
```

Subir archivos al servidor web

Esta es una tarea muy usual en los formularios web. Existen muchos sitios web que permiten a los usuarios subir sus archivos al servidor desde el navegador haciendo uso de una interfaz de usuario que forzosamente deberá incluir un campo de formulario del tipo file (`<input type="file" ... />`). Ejemplos prácticos de esto los puede encontrar en sitios web que le permiten subir fotos, su hoja de vida (curriculum vitae), música, etc. También lo puede encontrar en los sitios de correo electrónico web como Hotmail, Yahoo y Gmail, entre otros.

La tarea de crear una aplicación que le permita subir archivos al servidor debe dividirse en dos partes:

1. La página web que contendrá el formulario con el campo de formulario que le permitirá al usuario seleccionar el archivo que desea subir al servidor.
2. La programación PHP necesaria para procesar el archivo que ya ha sido transferido al servidor. En este punto habrá que acceder a dicho archivo para realizar las tareas que se consideren oportunas.

Creación del formulario para subir un archivo al servidor

En la página HTML necesita básicamente, un formulario que incluya una etiqueta input de tipo file. Esto se consigue con un código tan simple como el siguiente:

```
<form action="procesararchivo.php" method="POST" enctype="multipart/form-data">
  <!-- El valor establecido para el atributo value del input type="hidden" no debe ser superior al
  indicado en la   directiva upload_max_filesize del php.ini -->
  <input type="hidden" name="max_file_size" value="250000" />
  <label for="adjunto">Archivo a adjuntar:</label>
  <input type="file" name="adjunto" id="adjunto" /><br />
  <input type="submit" name="enviar" value="Subir" />
</form>
```

Note que en el elemento form que crea el formulario se ha utilizado el atributo enctype con el valor especial: multipart/form-data. Esto es requerido para que funcione correctamente la operación de subir el archivo al servidor web. Opcionalmente, se puede utilizar un campo oculto con el que se indicará el tamaño máximo del archivo a ser enviado a través del formulario. En este campo oculto se sugiere utilizar como nombre del campo el nombre max_file_size. Si no se hace esto se tomará de referencia el nombre de la directiva upload_max_filesize definida en el archivo php.ini.

Al pulsar el botón enviar del formulario el archivo seleccionado por el usuario será enviado al servidor y, como es de esperar, el archivo no es enviado a un directorio o carpeta al azar, ni tampoco al directorio donde está el script con el formulario. El archivo enviado es almacenado en un directorio indicado por la directiva upload_tmp_dir establecida en el archivo de configuración php.ini.

Guía # 6: Gestión de archivos y directorios con PHP

Para realizar las tareas de procesamiento en el servidor, una vez que el archivo está almacenado en el servidor, PHP proporciona la matriz asociativa `$_FILES`, que consta de dos dimensiones. En la primera columna de la matriz se utilizan como claves los identificadores (name) de los elementos input definidos en el formulario. En la segunda columna se utilizan claves especiales y predefinidas que se corresponden con las características de cada archivo. La siguiente tabla muestra los nombres de estas claves:

Clave	Descripción
name	Nombre original del archivo transmitido.
tmp_name	Nombre temporal del archivo que se establece en el momento que se termina de transferir el archivo al servidor. Si el archivo no se hubiera podido transmitir, por ejemplo, porque es demasiado grande en comparación con la directiva <code>upload_max_filesize</code> del <code>php.ini</code>, entonces obtendremos el valor <code>none</code>.
size	Tamaño en caracteres (bytes) del archivo.
type	El tipo MIME del archivo, indicado en el cliente.
error	Código de error devuelto que indica qué ha sucedido en la transferencia.

La última clave de la tabla anterior, puede ser consultada en caso de que se produzca algún error al intentar cargar el archivo al servidor. Los códigos de error que pueden devolverse son los siguientes:

Valor	Constante	Significado
0	<code>UPLOAD_ERR_OK</code>	No se han producido errores.
1	<code>UPLOAD_ERR_INI_SIZE</code>	El tamaño del archivo es superior al indicado en la directiva <code>upload_max_filesize</code> del archivo de configuración <code>php.ini</code>.
2	<code>UPLOAD_ERR_FORM_SIZE</code>	El archivo sobrepasa el tamaño indicado en el campo oculto <code>max_file_size</code>.
3	<code>UPLOAD_ERR_PARTIAL</code>	El archivo ha sido parcialmente transferido.
4	<code>UPLOAD_ERR_NO_FILE</code>	No se ha transferido el archivo.

También se proporcionan algunas funciones útiles para trabajar con archivos transferidos desde el cliente al servidor. La primera de ellas, denominada `is_uploaded_file($_FILES['adjunto']['tmp_name'])` nos indica si realmente el archivo ha sido transferido.

La segunda, denominada `move_uploaded_file(nombrearchivo, rutadestino)` sirve para renombrar y/o mover el archivo temporal a una ubicación permanente.

El siguiente código de ejemplo procesa el archivo una vez que llega al servidor:

```
<?php
foreach($_FILES['adjunto'] as $clave => $valor)
echo "\$_FILES[$clave] : $valor <br />";
if(!is_uploaded_file($_FILES['adjunto']['tmp_name'])){
    $error = $_FILES['adjunto']['error'];
    die("<h3>*** ERROR: Archivo no transferido: $error ***</h3>");
}
```

Guía # 6: Gestión de archivos y directorios con PHP

```
}  
if($_FILES['adjunto']['type'] != "application/x-zip-compressed"){  
    echo "<h3>** ERROR: El archivo enviado no es un archive comprimido **</h3>";  
}  
?>
```

III. MATERIALES Y EQUIPO

Para la realización de la guía de práctica se requerirá lo siguiente:

No.	Requerimiento	Cantidad
1	Guía de práctica #6: Archivos y directorios en PHP	1
2	Computadora con WampServer y PHP Designer 2007 instalado	1
3	Memoria USB	1

IV. PROCEDIMIENTO

Realice ordenadamente cada uno de los siguientes ejercicios. Algunos incluyen más de una script PHP junto con otros tipos de archivos.

Ejercicio #1: El siguiente ejemplo muestra cómo implementar con cuatro scripts PHP un administrador de archivos básico con interfaz web que permitirá: operaciones como navegación por el árbol de directorios (el directorio actual), borrado, renombrado y copiado de archivos y carpetas, entre otras operaciones típicas con directorios y archivos:

Archivo 1: administrador.php

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
    <meta charset="utf-8" />  
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />  
    <meta name="viewport" content="width=device-width, initial-scale=1"/>  
    <title>Administrador de archivos</title>  
    <link rel="stylesheet" href="css/filemanager.css" />  
    <!--[if IE]>  
        <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>  
    <![endif]>  
</head>  
<body>  
<header>  
    <h1>Administrador de archivos</h1>  
</header>  
<section>  
<article id="manager">  
<?php  
    if(!isset($_GET['directorio']) || $_GET['directorio'] == "."):  
        $directorio = ".";  
        $nombre_directorio = "/";
```

Guía # 6: Gestión de archivos y directorios con PHP

```

else:
    if(isset($_GET['directorio']) && $_GET['directorio'] != null):
        $directorio = $_GET['directorio'];
        $nombre_directorio = "\t" . basename($directorio);
    endif;
endif;

//Cambiar al directorio que se ha recibido como parámetro en la URL
if(!chdir($directorio)):
    die("<h3>ERROR: No se puede acceder a este directorio</h3>");
endif;

$tabla = "<table>\n\t";
$tabla .= "<caption>Elementos del directorio: $nombre_directorio</caption>\n\t";

//Abrir el manejador del directorio
$manejador = opendir(".");

//Procesar todos los elementos del directorio que también pueden ser directorios
while($selemento = readdir($manejador)):
    if(is_dir($selemento) && ($selemento != "." && !($directorio == "." && $selemento ==
".."))):
        if($selemento == ".."):
            $ruta = dirname($directorio);
            $item = "<span>Directorio anterior</span>";
        else:
            $ruta = $directorio . "/" . $selemento;
            $item = "<span>$selemento</span>";
        endif;
        $tabla .= "<tr>\n\t\t\t\t<td>\n\t\t\t\t\t";
        $tabla .= "<a href=\"administrador.php?directorio=";
        $tabla .= rawurlencode($ruta) . "\">\n\t\t\t\t\t";
        $tabla .= "<img src=\"img/openfolder.png\" alt=\"Cambiar a $selemento\"";
        $tabla .= "</a>\n\t\t\t\t";
        $tabla .= "</td>\n\t\t\t\t";
        $tabla .= "<td>\n\t\t\t\t\t";
        $tabla .= $selemento;
        $tabla .= "</td>\n\t\t\t\t</tr>\n\t\t\t\t";
        //echo $tabla;
    endif;
endwhile;

//Rebobinar el manejador de directorio
rewinddir($manejador);

//Procesar todos los elementos del directorio que son archivos
while($selemento = readdir($manejador)):
    if(!is_dir($selemento)):
        $tabla .= "<tr>\n\t\t\t\t";

```

Guía # 6: Gestión de archivos y directorios con PHP

```
$tabla .= "<td>\n\t\t\t";
$tabla .= "<a href=\"\" . $directorio . "/" . $elemento . "\">\n\t\t\t\t";
$tabla .= "<img src=\"img/file.jpg\" alt=\"Mostrar $elemento\" />\n\t\t\t\t";
$tabla .= "</a>\n\t\t\t";
$tabla .= "</td>\n\t\t\t";
$tabla .= "<td>$elemento</td>\n\t\t\t";
$tabla .= "<td>\n\t\t\t\t";
$tabla .= "<a href=\"operacionesarchivo.php?directorio=\"";
$tabla .= rawurlencode($directorio . "/" . $elemento) . "\">\n\t\t\t\t\t";
$tabla .= "<img src=\"img/toolsicon.png\" alt=\"Operaciones con $elemento\"";
/>\n\t\t\t\t\t";
$tabla .= "</a>\n\t\t\t\t";
$tabla .= "</td>\n\t\t\t";
$tabla .= "</tr>\n";

endif;
endwhile;
//Cerrar el manejador de directorio
closedir($manejador);
$tabla .= "</table>\n";
echo $tabla;
?>
</article>
</section>
</body>
<script src="js/modernizr.custom.lis.js"></script>
</html>
```

Archivo 2: operacionesdirectorio.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <title>Operaciones con directorios</title>
    <link rel="stylesheet" href="css/filemanager.css" />
</head>
<body>
<header>
    <h1>Operaciones con el directorio
    <?php
        error_reporting(E_ALL & ~E_NOTICE);
        if($_GET['directorio'] == "."):
            $nombre_directorio = " Raíz";
        else:
            $nombre_directorio = " " . basename($_GET['directorio']);
        endif;
        echo basename($nombre_directorio);
        $valor_directorio = rawurlencode($_GET['directorio']);
    ?>
```

Guía # 6: Gestión de archivos y directorios con PHP

```
</h1>
</header>
<section>
  <article>
    <?php
      $form = "<form method=\"POST\"
action=\"operacion.php?operacion=0&directorio=\";
      $form .= $valor_directorio . ">\n\t\t\t\t";
      echo $form;

    ?>

    <input type="image" name="crear" id="crear" src="img/newfolder.png" alt="Crear
directorio" />
    <label for="crear">Crear</label>
    <label for="nombre_directorio">Nombre directorio</label>
    <input type="text" name="nombre_directorio" id="nombre_directorio"
placeholder="Nombre del directorio" /><br />
    </form>
    <?php
      $form = "<form method=\"POST\" action=\"";
      $form .= "\"operacion.php?operacion=1&directorio=$valor_directorio\">\n\t\t\t\t";
      echo $form;

    ?>

    <input type="image" name="mostrar" id="mostrar" src="img/folderopenfiles.jpg"
alt="Mostrar directorio completo" />
    <label for="mostrar">Mostrar</label>
    </form>
    <?php
      $form = "<form method=\"POST\" action=\"operacion.php?operacion=2&directorio=\";
      $form .= "$valor_directorio\">\n\t\t\t\t";
      echo $form;

    ?>

    <input type="image" name="borrar" id="borrar" src="img/deletefolder.png"
alt="Borrar directorio" />
    <label for="borrar">Borrar</label>
    </form>
    <?php
      $form = "<form method=\"POST\" action=\"";
      $form .= "administrador.php?directorio=$valor_directorio\">\n\t\t\t\t";
      echo $form;

    ?>

    <input type="submit" name="volver" value="Volver al directorio" />
    </form>
  </article>
</section>
</body>
<script src="js/modernizr.custom.lis.js"></script>
</html>
```

Archivo 3: operacionesarchivo.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1"/>
    <title>Operaciones con archivos</title>
    <link rel="stylesheet" href="css/filemanager.css" />
    <link rel="stylesheet" href="css/demo.css" />
    <link rel="stylesheet" href="css/component.css" />
    <!--[if IE]>
        <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
</head>
<body>
<header>
    <h1>Operaciones con el archivo
    <?php
        error_reporting(E_ALL & ~E_NOTICE);
        if(!empty($_GET['directorio']) && $_GET['directorio'] !== null):
            echo basename($_GET['directorio']);
            $valor_directorio = rawurlencode($_GET['directorio']);
        endif;
    ?>
    </h1>
</header>
<section>
    <article>
    <?php
        $form = "<form method=\"POST\" action=\"\"";
        $form .= "operacion.php?operacion=3&directorio=" . $valor_directorio . "\"
class=\"ccform\">\n\t\t\t";
        echo $form;
    ?>
    <div class="ccfield-prepend">
        <input type="image" name="borrar" id="borrar" src="img/delete-file.gif"
alt="Borrar archivo" class="ccimagefield" />
        <label for="borrar">Borrar</label>
    </div>
    </form>
    <?php
        $form = "<form method=\"POST\" action=\"\"";
        $form .= "operacion.php?operacion=4&directorio=" . $valor_directorio . "\"
class=\"ccform\">\n\t\t\t";
        echo $form;
    ?>
    <div class="ccfield-prepend">
        <span class="ccform-addon">
```

Guía # 6: Gestión de archivos y directorios con PHP

```

        <i class="fa fa-user fa-2x fa-spin"></i>
    </span>
    <input type="text" name="destino" id="destino" placeholder="Nombre de la copia"
required class="ccformfield" />
    <input type="image" name="copiar" id="copiar" src="img/copyfile.jpg"
alt="Copiar archivo" class="ccimagefield" />
    <label for="copiar">Copiar</label>
</div>
</form>
<?php
    $form = "<form method=\"POST\" action=\"";
    $form .= "operacion.php?operacion=5&directorio=" . $valor_directorio . "\"
class=\"ccform\">\n\t\t\t";
    echo $form;
?>
    <div class="ccfield-prepend">
        <span class="ccform-addon">
            <i class="fa fa-envelope fa-2x"></i>
        </span>
        <input type="text" name="nuevo_nombre" id="nuevo_nombre" placeholder="Nuevo
nombre" required class="ccformfield" />
        <input type="image" name="renombrar" id="renombrar" src="img/renamefile.png"
alt="Renombrar archivo" class="ccimagefield" />
        <label for="renombrar">Renombrar</label>
    </div>
</form>
<?php
    $form = "<form method=\"POST\" action=\"";
    $form .= "administrador.php?directorio=" .
rawurlencode(dirname($_GET['directorio'])) . "\">\n\t\t\t";
    echo $form;
?>
    <div class="ccfield-prepend">
        <input type="submit" name="volver" value="Volver al directorio" class="ccbbtn"
/>
    </div>
</form>
</article>
</section>
</body>
<script src="js/modernizr.custom.lis.js"></script>
</html>

```

Archivo 4: operacion.php

```
<?php
//Función que muestra todos los elementos descendientes directos
//o indirectos del directorio que se pasa como parámetro
function mostrar_arbol($raiz){
    $nivel = "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&";
    echo $nivel . "<img src=\"directorio.gif\" alt=\"$raiz\" />";
```

Guía # 6: Gestión de archivos y directorios con PHP

```
        echo "<span>$raiz</span><br />";
        $manejador = opendir(".");
        while($selemento = readdir()):
            if(!is_dir($selemento)):
                echo $nivel . $nivel . "<img src=\"archivo.gif\" alt=\"$selemento\"
/>\n\t\t\t\t";
                echo $selemento . "<br />\n\t\t\t\t";
            endif;
        endwhile;
        rewinddir($manejador);
        while($selemento = readdir($manejador)):
            if(is_dir($selemento) && $selemento != "." && $selemento != ".."):
                chdir($selemento);
                mostrar_arbol("$raiz/$selemento");
            endif;
        endwhile;
        closedir($manejador);
        chdir("..");
    }

//Función que determina si ya existe en el directorio
//el nombre que se proporciona como parámetro
function existe_en_directorio($nombre){
    return file_exists($nombre);
}

//Función que determina si el directorio actual se encuentra vacío
function esta_vacio_directorio(){
    $manejador = opendir(".");
    $contador = 0;
    while($selemento = readdir($manejador)):
        $contador++;
    endwhile;
    closedir($manejador);
    return ($contador == 2);
}

//Función que escribe un botón que al ser pulsado vuelve
//a mostrar el directorio indicado como parámetro
function escribir_boton_volver($directorio){
    return "<form method=\"POST\" action=\"administrador.php?directorio=\" .
        rawurlencode("$directorio") . ">\n\t\t\t\t" .
        "<input type=\"submit\" name=\"volver\" value=\"Volver\" />\n\t\t\t\t" .
        "</form>\n\t\t";
}

//Función que muestra un mensaje de error y termina la ejecución del script
function error($numero, $directorio){
    $htmlstr = "<!DOCTYPE html>\n";
```


Guía # 6: Gestión de archivos y directorios con PHP

```
$htmlstr .= "<html lang=\"es\">\n";
$htmlstr .= "<head>\n\t";
$htmlstr .= "<title>Página de error</title>\n";
$htmlstr .= "</head>\n";
$htmlstr .= "<body>\n\t<header>\n\t\t";
$htmlstr .= "<h1>ERROR: No se puede ";
switch($numero):
    case 0:
        $htmlstr .= "acceder a este directorio</h1>\n\t";
        break;
    case 1:
        $htmlstr .= "crear este directorio</h1>\n\t";
        break;
    case 2:
        $htmlstr .= "borrar este directorio</h1>\n\t";
        break;
    case 3:
        $htmlstr .= "borrar este archivo</h1>\n\t";
        break;
    case 4:
        $htmlstr .= "copiar este archivo</h1>\n\t";
        break;
    case 5:
        $htmlstr .= "renombrar este archivo</h1>\n\t";
        break;
endswitch;
$htmlstr .= escribir_boton_volver($directorio) . "</header>\n</body>\n</html>";
die($htmlstr);
}

if(isset($_GET['directorio']) && ($_GET['directorio'] != null || $_GET['directorio'] !=
"")):
    $elemento = basename($_GET['directorio']);
    $ruta = dirname($_GET['directorio']);
endif;

//Cambio de directorio
if(isset($_GET['operacion'])):
    if(($_GET['operacion'] >= 0 && $_GET['operacion'] <= 2 && !chdir($_GET['directorio']))
|| ($_GET['operacion'] >= 3 && $_GET['operacion'] <= 5 && !chdir($ruta))):
        error(0, ".");
    endif;

//Ejecutar la operación requerida
switch($_GET['operacion']):
    case 0:
        if(empty($_POST['nombre_directorio']) ||
        existe_en_directorio($_POST['nombre_directorio']) ||
        !mkdir($_POST['nombre_directorio'], 0777)):
```

Guía # 6: Gestión de archivos y directorios con PHP

```
        error(1, $_GET['directorio']);
    endif;
    $ruta = $_GET['directorio'];
    break;

case 1:
    $htmlstr = "<!DOCTYPE html>\n";
    $htmlstr .= "<html lang=\"es\">\n";
    $htmlstr .= "<head>\n\t";
    $htmlstr .= "<title>Página de error</title>\n";
    $htmlstr .= "</head>\n";
    $htmlstr .= "<body>\n\t<header>\n\t\t";
    $htmlstr .= "<h1>Árbol completo de $elemento</h1>\n\t";
    $htmlstr .= "</header>\n";
    echo $htmlstr;
    break;

case 2:
    if(!esta_vacio_directorio() || !rmdir(".")):
        error(2, $_GET['directorio']);
    endif;
    break;

case 3:
    if(empty($elemento) || !existe_en_directorio($elemento) ||
    !unlink($elemento)):
        error(3, $ruta);
    endif;
    break;

case 4:
    if(empty($_POST['destino']) || existe_en_directorio($_POST['destino']) ||
    !copy($elemento, $_POST['destino'])):
        error(4, $ruta);
    endif;
    break;

case 5:
    if(empty($_POST['nuevo_nombre'])
    existe_en_directorio($_POST['nuevo_nombre']) ||
    !rename($elemento, $_POST['nuevo_nombre'])):
        error(5, $ruta);
    endif;
    break;

endswitch;
endif;

if(isset($_GET['operacion']) && $_GET['operacion'] != 1):
    header("Location: administrador.php?directorio=" . rawurlencode($ruta));
endif;

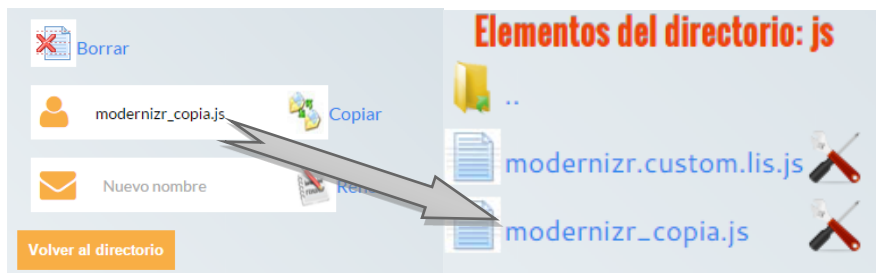
?>
```

Guía # 6: Gestión de archivos y directorios con PHP

En el navegador:



Navegando en carpetas:



Ejercicio #2: El siguiente ejemplo muestra la implementación de un libro de visitas. Para que el ejercicio funcione sin inconvenientes deberá crear una carpeta en la carpeta wamp, fuera de la carpeta www y en el mismo nivel de dicha carpeta. El nombre que debe asignar a esta carpeta es guest. Si cambia este nombre deberá cambiar también la línea de código donde se hace referencia a la ubicación donde se almacenará el archivo con datos.

Archivo 1: guestBook.class.php

```
<?php
class guestBook {
    //Propiedades
    protected $name;
```

Guía # 6: Gestión de archivos y directorios con PHP

```
protected $address;
protected $phone;
protected $birthday;
private $file;
//Métodos para escritura de las propiedades
function setName($name){
    $this->name = $name;
}
function setAddress($address){
    $this->address = $address;
}
function setPhone($phone){
    $this->phone = $phone;
}
function setBirthday($birthday){
    $this->birthday = $birthday;
}
function setFile($file){
    $this->file = $file;
}
//Métodos para lectura de las propiedades
function getName($name){
    return $this->name;
}
function getAddress($address){
    return $this->address;
}
function getPhone($phone){
    return $this->phone;
}
function getBirthday($birthday){
    return $this->birthday;
}
function getFile($file){
    return $this->file;
}
function showGuest(){
    echo "<div id=\"showGuest\">";
    echo "$this->name.<br>";
    echo "$this->address.<br>";
    echo "$this->phone.<br>";
    echo "$this->birthday.<br>";
    echo "</div>";
}
function saveGuest(){
    $outputstring = $this->name . " : " . $this->address . " : ";
    $outputstring .= $this->phone . " : " . $this->birthday . "\n";
    $path = "$_SERVER[DOCUMENT_ROOT]/../guest/$this->file";
    @$fh = fopen($path, "ab");
```

```
        if (!$fh) {
            $fh = fopen($path, "wb");
        }
        fwrite($fh, utf8_decode($outputstring), strlen($outputstring));
        fclose($fh);
        echo "<h3>Datos salvados en la ruta indicada $path";
    }
}
?>
```

Archivo 2: recordvisits.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Registrar usuarios en libro de visitas</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-
scale=1.0" />
    <link rel="stylesheet" href="css/style.css" />
    <link rel="stylesheet" href="css/responsive.css" />
    <style type="text/css">
        @import url(jscalendar/calendar-blue2.css);
    </style>
    <script src="jscalendar/calendar.js"></script>
    <script src="jscalendar/lang/calendar-es.js"></script>
    <script src="jscalendar/calendar-setup.js"></script>
</head>
<body>
<?php
    //Autocarga de la clase
    spl_autoload_register(function($classname){
        include_once("class/" . $classname . ".class.php");
    });

    if(isset($_POST['submit'])){
        extract($_POST); //Obtener los datos del formulario
        $entry = new guestBook();
        $entry->setName($yourname);
        $entry->setAddress($youraddress);
        $entry->setPhone($yourphone);
        $entry->setBirthDay($yourbirthday);
        $entry->setFile($yourfile);
        $entry->showGuest();
        $entry->saveGuest();
    }
    else{
?>
<section id="container">
```

Guía # 6: Gestión de archivos y directorios con PHP

```
<!-- <span class="chyron"><em><a href="http://www.hongkiat.com/blog/">&laquo; back to
the site</a></em></span> -->
<h2>Ingreso de datos</h2>
<form name="hongkiat" id="hongkiat-form" action="<?php echo $_SERVER['PHP_SELF'] ?>"
method="POST">
  <div id="wrapping" class="clearfix">
    <section id="aligned">
      <input type="text" name="yourname" id="yourname" placeholder="(Tu nombre)"
autocomplete="off" maxlength="40" tabindex="1" class="txtinput" />
      <input type="text" name="youraddress" id="youraddress" placeholder="(Tu
dirección)" autocomplete="off" maxlength="60" tabindex="2" class="txtinput" />
      <input type="tel" name="yourphone" id="yourphone" placeholder="(Tu número de
teléfono)" tabindex="4" maxlength="14" class="txtinput" />
      <input type="hidden" name="yourfile" id="yourfile" value="guestbook.txt" />
    </section>

    <section id="aside" class="clearfix">
      <section id="recipientcase">
        <h3>Fecha de nacimiento:</h3>
        <input type="date" name="yourbirthday" id="yourbirthday" placeholder="(Tu
fecha de nacimiento)" class="txtdate" />
        <br />
        <script type="text/javascript">
          Calendar.setup(
            {
              inputField : "yourbirthday", // ID of the input field
              ifFormat   : "%Y-%m-%d", // the date format
              button     : "imgcalendar", // ID of the button
              singleClick : true
            }
          );
        </script>
      </section>
    </section>
  </div>
  <section id="buttons">
    <input type="reset" name="reset" id="resetbtn" class="resetbtn" value="Restaurar"
/>
    <input type="submit" name="submit" id="submitbtn" class="submitbtn" tabindex="7"
value="Guardar" />
    <br style="clear:both;">
  </section>
</form>
</section>
<?php
}
?>
</body>
</html>
```

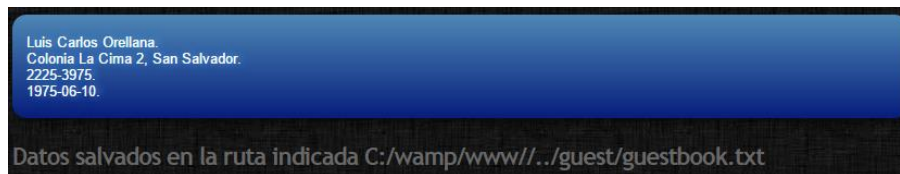
Guía # 6: Gestión de archivos y directorios con PHP

En el navegador:



The screenshot shows a web form titled "Ingreso de datos". It contains three input fields: a text field for the name "Luis Carlos Orellana", a text field for the address "Colonia La Cima 2, San Salvador", and a text field for the phone number "2225-3975". To the right of the name field is a label "Fecha de nacimiento:" followed by a date picker showing "10/06/1975". At the bottom of the form are two buttons: "Restaurar" (grey) and "Guardar" (green).

Al guardar los datos:



Ejercicio #3: La siguiente aplicación ilustra cómo crear un editor de texto en PHP, con la particularidad que permite mediante el uso de formularios crear archivos, listar los archivos creados y editarlos en una misma aplicación. Para la edición de los archivos existentes se hace uso de un campo de formulario del tipo textarea. Debe crear una carpeta llamada archivos para que funcione correctamente este ejercicio. En este ejemplo también utilizará la hoja de estilos enlaces.css.

Archivo 1: editortexto.php

```
<?php
//error_reporting(E_ALL ^ E_NOTICE);
require_once("funcioneseditor.php");
if(isset($_POST['savefile'])) {
    savefile();
}
elseif(isset($_GET['filename'])) {
    displayeditform();
}
elseif(isset($_POST['createfile'])) {
    createfile();
}
else {
    displayfilelist();
}
?>
```

Archivo 2: funcioneseditor.php

```
<?php
define("PATH", "archivos");

function displayfilelist($message = "") {
```

Guía # 6: Gestión de archivos y directorios con PHP

```
displaypageheader();
if(!file_exists(PATH)) die("No se encontró el directorio");
if(!($dir = dir(PATH))) die("No se puede abrir el directorio");
if($message){
    echo "<p class=\"error\">$message</p>";
}
//Imprimiendo encabezados de la tabla
$table = "<table id=\"file-data\">\n";
$table .= "<caption>Selecione el archivo a editar</caption>\n";
$table .= "<thead>\n<tr>\n<th>\nNombre archivo\n</th>\n";
$table .= "<th>\nTamaño\n</th>\n";
$table .= "<th>\nModificación</th>\n</tr>\n</thead>\n";
//Imprimiendo los archivos contenidos en el directorio y
//creando los enlaces para su edición
$table .= "<tbody>\n";
$numfilas = 0;
while($filename = $dir->read()){
    $filepath = PATH . "/" . $filename;
    if($filename != "." && $filename != ".." && !is_dir($filepath) && strrchr($filename,
".")) == ".txt"){
        $clase = ($numfilas%2 != 0) ? "odd" : "even";
        $table .= "<tr class=\"{$clase}\">\n<td>\n";
        $table .= "<a href=\"editortexto.php?filename=" . urlencode((binary)$filename) .
"\n">";
        $table .= $filename . "</a>\n</td>\n";
        $table .= "<td>\n" . filesize($filepath) . "\n</td>\n";
        $table .= "<td>\n" . date("M j, Y H:i:s", filemtime($filepath)) .
"\n</td>\n</tr>\n";
    }
    $numfilas++;
}
$dir->close();
$table .= "</tbody>\n</table>\n";
echo $table;
$form = "<form name=\"nuevo\" action=\"editortexto.php\" method=\"POST\">\n";
$form .= "<fieldset>\n<legend><span>Creando un nuevo archivo:</span></legend>\n";
$form .= "<ul>\n<li>\n";
$form .= "<div id=\"campo\">\n";
$form .= "<label for=\"filename\">Nombre del archivo: </label>\n";
$form .= "<input type=\"text\" name=\"filename\" id=\"filename\" placeholder=\"(Ingrese
el nombre del archivo)\" maxlength=\"100\" />\n";
$form .= "</div>\n</li>\n";
$form .= "<li>\n<div id=\"boton\">\n";
$form .= "<input type=\"submit\" name=\"createfile\" value=\"Crear archivo\" />\n";
$form .= "</div>\n</li>\n</ul>\n";
$form .= "</fieldset>\n</form>\n";
echo $form;
displaypagefooter();
}
```


Guía # 6: Gestión de archivos y directorios con PHP

```
function displayeditform($filename = ""){
    $archivo = isset($_GET['filename']) ? $_GET['filename'] : "";
    echo "<div id=\"info-file\">\n";
    //echo $archivo . "<br />\n";
    if(!$filename) $filename = basename($archivo);
    if(!$filename) die("Nombre de archivo inválido");
    $filepath = PATH . "/" . $filename;
    echo $filepath . "\n";
    echo "</div>\n";
    if(!file_exists($filepath)) die("Archivo no encontrado");
    displaypageheader();
    $editform = "<section id=\"formulario\">\n";
    $editform .= "<h2>Editando archivo: $filename</h2>\n";
    $editform .= "\t<form      name=\"creararchivo\"      action=\"editortexto.php\"
method=\"POST\">\n";
    $editform .= "\t\t<div style=\"width:40em\">\n";
    $editform .= "\t\t\t<input type=\"hidden\" name=\"filename\" value=\"$filename\" />\n";
    $editform .= "\t\t\t<textarea name=\"filecontents\" id=\"filecontents\" cols=\"80\"
rows=\"20\">\n";
    $editform .= file_get_contents($filepath) . "\n";
    $editform .= "</textarea>\n";
    $editform .= "\t\t\t<div style=\"clear:both\">\n";
    $editform .= "\t\t\t\t<input type=\"submit\" name=\"savefile\" value=\"Guardar archivo\"
/>\n";
    $editform .= "\t\t\t\t<input type=\"submit\" name=\"cancel\" value=\"Cancelar\" />\n";
    $editform .= "\t\t\t</div>\n";
    $editform .= "\t\t</div>\n\t</form>\n";
    $editform .= "</section>\n";
    echo $editform;
    displaypagefooter();
}

function savefile(){
    $archivo = isset($_POST['filename']) ? $_POST['filename'] : "";
    $filename = basename($archivo);
    $filepath = PATH . "/" . $filename;
    if(file_exists($filepath)){
        $filecontents = isset($_POST['filecontents']) ? $_POST['filecontents'] : "";
        if(file_put_contents($filepath, $_POST['filecontents']) === false)
            die("<p class=\"error\">No se ha podido guardar el archivo</p>\n");
        displayfilelist();
    }
    else{
        die("Archivo no encontrado...");
    }
}

function createfile(){
    $filename = basename($_POST['filename']);
    echo $filename . "<br />\n";
}
```

Guía # 6: Gestión de archivos y directorios con PHP

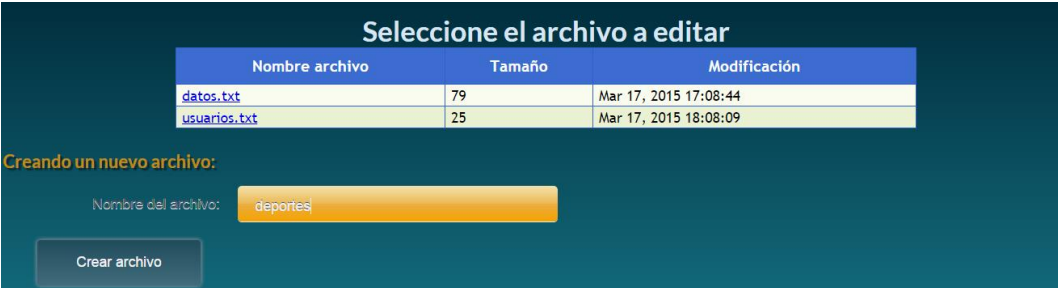
```
$filename = preg_replace("/[^A-Za-z0-9_- ]/", "", $filename);
if(!$filename){
    displayfilelist("<p class=\"error\">Nombre de archivo no válido. Pruebe con otro nombre.</p>");
    return;
}
$filename .= ".txt";
$filepath = PATH . "/" . $filename;
if(file_exists($filepath)){
    displayfilelist("El archivo $filename ya existe.");
}
else{
    if(file_put_contents($filepath, "") === false)
        die("No se ha podido crear el archivo");
    chmod($filepath, 0777);
    displayeditform($filename);
}
}

function displaypageheader(){
    echo "<!DOCTYPE html>\n";
    echo "<html lang=\"es\">\n";
    echo "<head>\n";
    echo "<title>Editor de texto basado en web</title>\n";
    echo "<meta charset=\"utf-8\" />\n";
    echo "<link rel=\"stylesheet\" href=\"css/page.css\" />\n";
    echo "</head>\n";
    echo "<body>\n";
}

function displaypagefooter(){
    echo "</body>\n";
    echo "</html>\n";
}

?>
```

En el navegador:



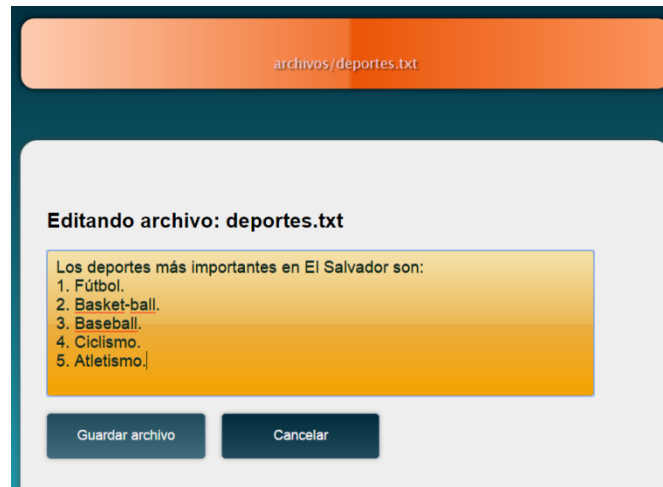
Nombre archivo	Tamaño	Modificación
datos.txt	79	Mar 17, 2015 17:08:44
usuarios.txt	25	Mar 17, 2015 18:08:09

Creando un nuevo archivo:

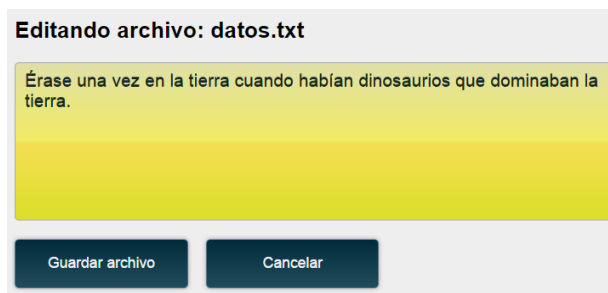
Nombre del archivo:

Creando un nuevo archivo con el editor:

Guía # 6: Gestión de archivos y directorios con PHP



Cargando un archivo previamente creado:



Ejercicio #4: Ejemplo de cómo implementar la carga de archivos al servidor desde un formulario web en el cliente. La aplicación incluye un script JavaScript que se utiliza para poder subir más de un archivo a la vez, creando dinámicamente un nuevo juego de controles de formulario del tipo file. Debe crear una carpeta llamada files dentro de la carpeta donde está guardando los scripts PHP de esta guía de práctica.

Archivo 1: uploadfile.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <title>Subir múltiples archivos</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" />
  <link rel="stylesheet" href="css/upload.css" />
  <script src="js/upload.js"></script>
</head>
<body>
<section>
  <article>
    <form name="formu" id="formu" action="upload.php" method="POST" enctype="multipart/form-data">
```

Guía # 6: Gestión de archivos y directorios con PHP

```
<dl>
  <dt>
    <h2>Archivos a Subir:</h2>
  </dt>
  <!-- Este div contendrá todos los campos file que creemos -->
  <dd>
    <div id="conteGeneral">
      <div class="contenedor">
        <input type="text" id="uploadFile0" placeholder="Seleccionar archivo"
disabled="disabled" />
        <div id="adjuntos" class="file-upload btn ">
          <!-- Hay que prestar atención a esto, el nombre
de este campo siempre debe terminar en [] como
un vector, y además debe coincidir con el nombre
que se da a los campos nuevos en el script js -->
          <span class="btn btn-primary span" id="spanadj">Adjunto</span>
          <input type="file" name="archivos[]" id="uploadBtn0" class="upload"
/><br/>
        </div>
      </div>
    </dd>
    <dt>
      <a href="javascript:void(0);" id="addfieldlink">Subir otro archivo</a>
    </dt>
    <dd>
      <input type="submit" value="Enviar" id="envia" name="envia" class="btn btn-
default" />
    </dd>
  </dl>
</form>
</article>
</section>
</body>
</html>
```

Archivo 2: upload.js

```
//Esta es una variable de control para mantener nombres
//diferentes de cada campo de texto creado dinámicamente.
var numero = 0;
var idText = 0;
var idField = 0;
var idSpan = 0;

window.onload = init;

function init() {
  //Asociando el enlace Subir otro archivo al enlace apropiado
  var link = document.getElementById("addfieldlink");
```

Guía # 6: Gestión de archivos y directorios con PHP

```
if(link.addEventListener){
    link.addEventListener("click", addCampo, false);
}
else if(link.attachEvent){
    link.attachEvent("onclick", addCampo);
}
// para añadirle funcionalidad al span del primer input file type
var span = document.getElementById("spanadj");
if (span.addEventListener) {
    span.addEventListener("click",function(){
        var adj = document.getElementById("uploadBtn0");
        adj.click();
        adj.onchange = function(){
            document.getElementById('uploadFile0').value = this.value;
        }
    },false);
}
}

//Esta funcion nos devuelve el tipo de evento disparado
evento = function (evt) {
    return (!evt) ? event : evt;
}

//Con esta funcion crea dinamicamente los nuevos campos file
addCampo = function () {
    //Contador para los elementos dinámicos que se generarán
    ++numero;

    //Creamos un nuevo div para que contenga el nuevo campo
    nDiv = document.createElement('div');
    //con esto se establece la clase del elemento div
    nDiv.className = 'contenedor';
    //Este es el id del div, aquí la utilidad de la variable numero
    //nos permite darle un id unico
    nDiv.id = 'file' + (numero);

    //Creando un elemento input que estará deshabilitado
    nInput = document.createElement('input');
    //Colocando los atributos para el elemento input type=text
    nInput.type = 'text';
    //Establecer el id del campo de texto deshabilitado
    nInput.id = 'uploadFile' + (numero);
    //Colocando el placeholder
    nInput.placeholder = 'Seleccionar archivo';
    //Agregando el atributo para que el control esté deshabilitado
    nInput.disabled = 'disabled';

    //Creamos otro div que contendrá los elementos
```

Guía # 6: Gestión de archivos y directorios con PHP

```
nDivAdjunto = document.createElement('div');
nDivAdjunto.className = "file-upload btn";

//Creando el elemento span que hará las veces del botón
//incluido en los campos input type=file cuando este elemento
//se oculte y se cambie por un span y un campo de texto
nSpan = document.createElement('span');
nSpan.className = "btn btn-primary";
nSpan.innerHTML = 'Adjunto';
nSpan.id = (numero);
nSpan.onclick = addOneClick;

//Creamos el input type=file para el formulario:
nCampo = document.createElement('input');
//Le damos un nombre, es importante que se nombre como vector/matriz,
//pues todos los campos compartiran el nombre en un arreglo,
//asi es mas facil procesar posteriormente con php
nCampo.name = 'archivos[]';
//Establecemos el tipo de campo
nCampo.type = 'file';
//Agregar id al campo input type=file
nCampo.id = 'uploadBtn' + (numero);
//Agregar la clase al elemento input type=file
nCampo.className = 'upload';

//Creando un elemento a href para poder eliminar un campo
//que ya no deseemos
a = document.createElement('a');
//El link debe tener el mismo nombre de la div padre, para efectos
//de localizarla y eliminarla
a.name = nDiv.id;
//Este link no debe ir a ningun lado
a.href = 'javascript:void(0)';
//Establecemos que dispare esta funcion en click
a.onclick = elimCamp;
//Con esto ponemos el texto del link
a.innerHTML = 'Eliminar';

//Integrar los elementos que se han creado al documento,
//primero usamos la función appendChild para adicionar
//el campo input type=file nuevo y luego los otros elementos
//que realizarán la funcionalidad de este mismo campo una vez
//que el input type=file sea ocultado para un mejor aspecto.
nDiv.appendChild(nInput);
nDiv.appendChild(nDivAdjunto);
nDivAdjunto.appendChild(nSpan);
nDivAdjunto.appendChild(nCampo);
nDivAdjunto.appendChild(a);
```

Guía # 6: Gestión de archivos y directorios con PHP

```
//Ahora si recuerdan, en el html hay una div cuyo id es 'adjuntos',
//bien con esta función obtenemos una referencia a ella para usar
//de nuevo appendChild y adicionar la div que hemos creado, la cual
//contiene el campo file con su link de eliminación:
container = document.getElementById('conteGeneral');
container.appendChild(nDiv);
}

//Con esta función eliminamos el campo cuyo link de eliminación sea presionado
elimCamp = function (evt){
    evt = evento(evt);
    nCampo = rObj(evt);
    console.log(nCampo.name);
    div = document.getElementById(nCampo.name);
    div.parentNode.removeChild(div);
}

addOneClick = function (evt){
    evt = evento(evt);
    nCampo = rObj(evt);
    div = document.getElementsByTagName("span")[this.id];
    file = "uploadBtn"+div.id;
    var fileUp = document.getElementById(file);
    console.log(fileUp);
    var adj = document.getElementById(fileUp.id);
    adj.click();
    adj.onchange = function(){
        document.getElementById('uploadFile'+div.id).value = this.value;
    }
}

//Con esta función recuperamos una instancia del objeto que disparó el evento
rObj = function (evt) {
    return evt.srcElement ? evt.srcElement : evt.target;
}
```

Archivo 3: upload.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <title>Subir múltiples archivos</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" />
    <link rel="stylesheet" href="css/upload.css" />
</head>
<body>
<?php
```

Guía # 6: Gestión de archivos y directorios con PHP

```
define("PATH", "archivos");
//Verificar que la matriz asociativa $_FILES["archivos"] haya sido definida
if (isset($_FILES["archivos"])) {
    //De ser así se procesa cada uno de los archivos.
    //Para poder hacerlo es conveniente obtener la cantidad
    //de elementos que tiene matriz $_FILES["archivos"]
    $total = count($_FILES["archivos"]["name"]);
    //este for recorre la matriz $_FILES
    for($i = 0; $i < $total; $i++){
        //Las propiedades definidas para cada archivo son:
        //1. 'tmp_dir': directorio temporal en el servidor donde se aloja el archivo
        //2. 'name': nombre original del archivo seleccionado por el usuario
        //3. 'size': tamaño en bytes del archivo
        //Para recorrer uno a uno los archivos que se hayan decidido subir al servidor
        //se utilizará el contador $i. En caso de que solo se suba un archivo el ciclo
        //se ejecutará una sola vez.
        $tmp_name = $_FILES["archivos"]["tmp_name"][$i];
        $name = $_FILES["archivos"]["name"][$i];
        $size = $_FILES["archivos"]["size"][$i];
        //echo "<h3>$size bytes</h3>";
        if($size > 2621440){
            echo "<h3>El tamaño del archivo es superior al admitido por el
servidor</h3><br>";
            echo "<a href='\"uploadfile.html\"'>Intentar de nuevo</a>";
        }
        echo "<h3 class='\"title\"'>Archivo " . ($i+1) . " :</h3>";
        echo "<b>el nombre original:</b> ";
        echo $name;
        echo "<br />";
        echo "<b>el nombre temporal:</b> \n";
        echo $tmp_name;
        echo "<br />";
        echo "<b>el tamaño del archivo:</b> \n";
        echo number_format($size, 2);
        echo " bytes<br />";
        //Verificar la carpeta en el servidor donde se alojarán los archivos
        //que se desean subir. Si no existe esta carpeta se creará y si no
        //es posible crearla se lanzará un error y se terminará el script
        if(!file_exists(PATH)){
            //Crear el directorio y asignar los permisos al mismo
            if(!mkdir(PATH, 0777, true)) {
                die('No se ha podido crear el directorio');
            }
        }
        //Una vez que es procesado cada archivo correctamente, se moverá
        //a una carpeta específica en el servidor, en este caso se usará
        //la carpeta files/.
        if(move_uploaded_file($tmp_name, PATH . "/" . utf8_decode($name))){
            echo "Se ha cargado correctamente el archivo " .
```


Guía # 6: Gestión de archivos y directorios con PHP

```

    "<a href=\"archivos/" . urldecode($name) . "\" target=\"_blank\">" .
$name . "</a>\n" . " en el servidor.\n<br />\n";
}
else{
    switch($_FILES['archivos']['error'][$i]){
        //No hay error, pero puede ser un ataque
        case UPLOAD_ERR_OK:
            echo "<p>Se ha producido un problema con la carga del archivo.</p>\n";
            break;
        //El tamaño del archivo es mayor que upload_max_filesize
        case UPLOAD_ERR_INI_SIZE:
            echo "<p>El archivo es demasiado grande, no se puede cargar.</p>\n";
            break;
        //El tamaño del archivo es mayor que MAX_FILE_SIZE
        case UPLOAD_ERR_FORM_SIZE:
            echo "<p>El archivo es demasiado grande, no se pudo cargar.</p>\n";
            break;
        //Solo se ha cargado parte del archivo
        case UPLOAD_ERR_PARTIAL:
            echo "<p>Sólo se ha cargado una parte del archivo.</p>\n";
            break;
        //No se ha seleccionado ningún archivo para subir
        case UPLOAD_ERR_NO_FILE:
            echo "<p>Debe elegir un archivo para cargar.</p>\n";
            break;
        //No hay directorio temporal
        case UPLOAD_ERR_NO_TMP_DIR:
            echo "<p>Problema con el directorio temporal. Parece que no
existe</p>\n";
            break;
        default:
            echo "<p>Se ha producido un problema al intentar mover el archivo "
. $name . "</p>\n";
            break;
    }
}
}
}
else{
    echo "<h3>No se han seleccionado archivos.</h3>";
}
?>
</body>
</html>
```

El resultado al visualizarlo en el navegador de su preferencia sería:

Guía # 6: Gestión de archivos y directorios con PHP

Archivos a Subir:

fondo1.jpg	Adjunto	
index.html	Adjunto	Eliminar
css.css	Adjunto	Eliminar
calle.jpg	Adjunto	Eliminar
Subir otro archivo		
<input type="button" value="Enviar"/>		

Archivo 1:

el nombre original: fondo1.jpg
el nombre temporal: C:\wamp\tmp\phpA3CE.tmp
el tamaño del archivo: 185,466.00 bytes
Se ha cargado correctamente el archivo fondo1.jpg en el servidor.

Archivo 2:

el nombre original: index.html
el nombre temporal: C:\wamp\tmp\phpA3DF.tmp
el tamaño del archivo: 317.00 bytes
Se ha cargado correctamente el archivo index.html en el servidor.

Archivo 3:

el nombre original: css.css
el nombre temporal: C:\wamp\tmp\phpA3E0.tmp
el tamaño del archivo: 136.00 bytes
Se ha cargado correctamente el archivo css.css en el servidor.

Archivo 4:

el nombre original: calle.jpg
el nombre temporal: C:\wamp\tmp\phpA400.tmp
el tamaño del archivo: 9,099.00 bytes
Se ha cargado correctamente el archivo calle.jpg en el servidor.

Carpeta en donde han sido alojados los archivos:

Este equipo > OS (C:) > wamp > www > lis > ciclo012016 > guias > guia7 > ejemplo6 > files				
<input type="checkbox"/>	Nombre	Fecha de modificación	Tipo	Tamaño
<input checked="" type="checkbox"/>	calle.jpg	15/03/2016 02:00 p.m.	Imagen JPEG	9 KB
<input type="checkbox"/>	css.css	15/03/2016 02:00 p.m.	Documento de hoja de estilos en cascada	1 KB
<input type="checkbox"/>	fondo1.jpg	15/03/2016 02:00 p.m.	Imagen JPEG	182 KB
<input type="checkbox"/>	index.html	15/03/2016 02:00 p.m.	Firefox HTML Document	1 KB

Ejercicio #5: En este ejemplo se busca dentro de un archivo el registro indicado por el primer nombre ingresado en la caja de texto para búsqueda. Abra el archivo datebook.txt y seleccione el primer nombre de un registro y digítelo en la caja de texto (Nombre a buscar). El script PHP debería mostrar todos los registros del archivo que coincidan con ese primer nombre.

Archivo 1: functions.lib.php

```
<?php
function showform(){
?>
<form method="POST" action="<?php echo $_SERVER['PHP_SELF']; ?>" id="searchthis">
    <input type="text" name="firstname" id="namanyay-search-box" placeholder="(Nombre a
buscar)"/>
    <input name="submit" id="namanyay-search-btn" value="Search" type="submit" />
</form>
<?php
}

function processfile(){
    $filename = "files/datebook.txt";
    //Obteniendo las líneas del archivo para asignarlas en una matriz
    $rows = file($filename);
    $firstname = trim($_POST['firstname']);
```

Guía # 6: Gestión de archivos y directorios con PHP

```
//Recorriendo una por una las líneas extraídas del archivo a través de la matriz
$count = 0;
foreach($rows as $register):
    $fields = explode(":", $register);
    $fullname = explode(" ", $fields[0]);
    $phone = $fields[1];
    $address = $fields[2];
    $birthday = $fields[3];
    $salary = $fields[4];
    if(strcasecmp($fullname[0], $firstname) == 0):
        $birthday = explode("/", $birthday);
        $newstring = implode("<br />",
            array($fields[0],
                $phone,
                $address,
                implode("/", $birthday),
                '$ ' . number_format(floatval($salary), 2, '.', ','));
        echo "<p>" . $newstring . "</p>";
        $count++;
    endif;
endforeach;
if($count == 0):
    echo "<p>El nombre " . $firstname . " no está en el archivo</p>";
endif;
}
?>
```

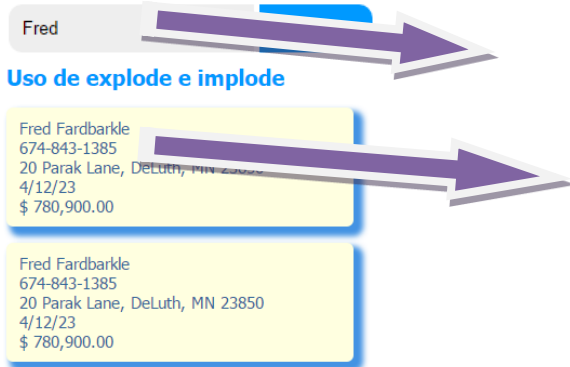
Archivo 2: implodefile.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <title>Accediendo a las líneas de un archivo</title>
    <link rel="stylesheet" href="css/form.css" />
</head>
<body>
<header>
    <h2>Uso de explode e implode</h2>
</header>
<section>
<article>
<?php
    include_once("functions.lib.php");
    if(!isset($_POST['submit'])):
        showform();
    else:
        processfile();
    endif;
```

```
?>
</article>
</section>
</body>
</html>
```

En el navegador de su preferencia podrá ver:

Uso de explode e implode



```
Steve Blenheim:238-923-7366:95 Latham Lane, Easton, PA 83755:11/1
Betty Boop:245-836-8357:635 Cutesy Lane, Hollywood, CA 91464:6/23
Igor Chevsky:385-375-8395:3567 Populus Place, Caldwell, NJ 23875:
Norma Corder:397-857-2735:74 Pine Street, Dearborn, MI 23874:3/28
Jennifer Cowan:548-834-2348:583 Laurel Ave., Kingsville, TX 83745
Jon DeLoach:408-253-3122:123 Park St., San Jose, CA 84886:7/25/53
Karen Evich:284-758-2857:23 Edgecliff Place, Lincoln, NB 92886:7/
Karen Evich:284-758-2867:23 Edgecliff Place, Lincoln, NB 92743:11
Karen Evich:284-758-2867:23 Edgecliff Place, Lincoln, NB 92743:11
Fred Fardbarkle:674-843-1385:20 Parak Lane, Deluth, MN 23850:4/12
Fred Fardbarkle:674-843-1385:20 Parak Lane, Deluth, MN 23850:4/12
Lori Gortz:327-832-5728:3465 Mirlo Street, Peabody, MA 34756:10/2
Paco Gutierrez:835-365-1284:454 Easy Street, Decatur, IL 75732:2/
Ephram Hardy:293-259-5395:235 Carlton Lane, Joliet, IL 73858:8/12/
```

V. DISCUSION DE RESULTADOS

1. Realice una aplicación PHP en la que se pueda llevar un log de visitas indicando en primer lugar la IP del visitante (utilice alguna de las matrices superglobales de servidor de PHP, `$_SERVER`, para obtener esta IP), el nombre del script que ejecutó y la fecha y hora al que se produjo esta visita. Los datos deben ser almacenados en un archivo de texto y luego un administrador debe poder abrir este archivo para verificar los datos de los visitantes que han ingresado al sitio mostrando los datos antes mencionados en alguna tabla HTML.

VI. INVESTIGACIÓN COMPLEMENTARIA

- Investigue sobre la utilización de la función de las funciones `file_get_contents()` y `file_put_contents()` indicando con detalle su sintaxis, argumentos que reciben y valores devueltos. Si es necesario explicar en detalle uno o varios de los argumentos de la función puede utilizar tablas u otro medio para realizar la explicación. Además de la sintaxis muestre un ejemplo detallado sobre cómo utilizar estas dos funciones, desde que abre el archivo, hasta cerrarlo.
- Investigue sobre la pseudo-clase `di` para el trabajo con directorios y los métodos que posee para realizar tareas como leer rebobinar y cerrar, así como las propiedades de consulta que posee. Además de la descripción de la clase, muestre ejemplos cortos sobre su utilización.

VII. BIBLIOGRAFIA

- Cabezas Granado, Luis Miguel. PHP 6 Manual Imprescindible. 1ra. Edición. Editorial Anaya Multimedia. Madrid, España. 2010.
- Doyle, Matt. Fundamentos de PHP Práctico. Editorial Anaya Multimedia. 1ª. Edición. Madrid, España. 2010.
- Gutierrez, Abraham / Bravo, Ginés. PHP 5 a través de ejemplos. 1ra Edición. Editorial Alfaomega. Junio 2005. México.
- Welling, Luke / Thomson, Laura. Desarrollo web con PHP y MySQL. Traducción de la 3ra Edición en inglés. Editorial Anaya Multimedia. 2005. Madrid, España.
- Gil Rubio / Francisco Javier, Villaverde / Santiago Alonso. Creación de sitios web con PHP5. 1a. edición en español. Editorial McGraw Hill. Madrid, España. 2006.
- John Coggeshall. LA BIBLIA DE PHP 5. 1ra. Edición. Editorial Anaya Multimedia. Madrid, España 2005.