ESCUELA DE CIENCIAS BÁSICAS TECNOLOGÍA E INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS

MATERIAL DIDÁCTICO DEL CURSO PROGRAMACIÓN DE SITIOS WEB - PHP

NOVIEMBRE DEL 2016

TABLA DE CONTENIDO

ASPECTOS DE PROPIEDAD INTELECTUAL Y VERSIONAMIENTO

El presente material del Curso Programación de Sitios Web – PHP, ha sido diseñado por los siguientes autores en el mes de abril del año 2007:

Ingeniero: Jorge Eduardo Salazar Zúñiga
Ingeniero: Orlando Puentes Andrade

La segunda versión del material, fue estructurada por capítulos independientes y sufrió leves cambios en su contenido en el año 2011, estando a cargo de:

Ingeniero: Yon Yerson Robles

La tercera versión del material ha sido actualizada a las nuevas plantillas de la universidad y referenciada con normas APA, también tiene ciertos cambios en la presentación y en el contenido, dicha modificación se realiza en noviembre del 2016, a cargo de:

Ingeniero: Francisco Javier Hilarión Novoa

¿QUÉ ES PHP?

PHP: HyperText Preprocessor. Es un lenguaje de tipo scripting, que actualmente está siendo utilizado como una de las mejores alternativas para desarrollar aplicaciones en la Web. Un lenguaje sencillo de aprender, porque basa su código en las estructuras y notaciones de otros muy populares lenguajes como C o Basic, además de ser potente y de alto rendimiento. PHP se considera como un lenguaje de programación del lado del servidor (se ejecuta en el servidor Web), rápido, que posee una gran cantidad de librería de funciones y una amplia documentación. Las páginas ejecutadas en el servidor pueden realizar accesos a bases de datos, conexiones en red, etc. El usuario o cliente recibe una página HTML resultante de la ejecución del código y página PHP.

El lenguaje PHP fue creado por Rasmus Lerdorf en 1994. Sin embargo al ser desarrollado en política de código abierto, ha recibido muchas contribuciones de otros desarrolladores. PHP se encuentra en la versión 4**,** que utiliza el motor Zend y cuenta con una extensa librería de funciones de soporte a los programadores (php.net, 2001).

Características de PHP

El código del lenguaje PHP se encuentra embebido en los documentos HTML. PHP puede interactuar con los principales, y más comunes, gestores de Bases de Datos en servidores Web. Se considera un lenguaje robusto y potente que está escrito en lenguaje C, con la gran ventaja que es gratuito y su código fuente, como el LINUX, está a disposición de los usuarios. PHP como todos los lenguajes creados pensando en Internet, soporta diversidad de protocolos de comunicaciones entre ellos FTP, HTTP, IMAP, etc.

Una de las grandes virtudes del lenguaje es que su código puede ser ejecutado en diversos sistemas operativos sin realizarle cambios; soportado por las versiones de Windows 95, 98, Me, NT, 2000, Unix y Linux. Cuando PHP, se monta en servidores Linux u Unix, es más rápido que muchos lenguajes como el caso de ASP y también aumenta la seguridad comparado con ambientes Windows; PHP permite configurar el servidor de modo que puede hacer al lenguaje más o menos seguro según necesidades específicas (Gonzalez, 2016).

¿Qué puedo hacer con PHP?

Con las contribuciones recibidas de los cientos de colaboradores interesados en el desarrollo del lenguaje, el PHP se transforma rápidamente en un lenguaje capacitado para realizar cualquier tarea. Entre ellas podemos destacar:

La gestión con las bases de datos donde el lenguaje permite interfaces con la mayoría de las bases de datos, incluyendo las de los sistemas Microsoft a través de ODBC.
Es muy sencillo con PHP enviar un e-mail a una persona por medio de su dirección electrónica o a una lista detallada de ellas

PHP facilita diversas tareas de tratamiento de imágenes a través de funciones, que serían demasiado tediosas utilizando alguna herramienta especialidad para ello.

Dentro del conjunto de librerías de PHP, se encuentran varias destinadas al proceso de gestión de archivo: crear, borrar, cambiar de nombre, mover, modificar, etc. Otras funciones también permitirán transferir archivos por FTP.

Con PHP se pueden tratar cookies con relativa facilidad. Así también otras tareas de los ambientes Internet.

El lenguaje proporciona en sus librerías diferentes funciones par tratamiento de textos, funciones matemáticas, y muchas otras más de uso general (hipertexto & Lamarca Lapuente, 2013).

PHP, el intérprete y el servidor

Para ejecutar aplicaciones desarrolladas en PHP, es necesario instalar el intérprete del lenguaje y un servidor Web. Generalmente las aplicaciones desarrolladas en PHP, se ejecutan sobre un navegador Web como Internet Explorer. Cuando el usuario realiza una llamada a un documento PHP, el navegador realiza un envío de solicitud al servidor a través del protocolo HTTP. El servidor identifica que el archivo es código fuente de PHP, por medio de su extensión, y ejecuta el intérprete. El intérprete ejecuta el programa obteniendo un resultado que es devuelto a través del servidor al Usuario visualizándose en el navegador.

Existen tres formas diferentes de instalar el intérprete de PHP: Como un intérprete externo (modo CGI), Como una extensión del servidor (vía NSAPI e ISAPI) o como módulo interno del servidor (sólo posible en Apache) (cdsphp, s.f.).
Reflexión
El código del lenguaje PHP se encuentra embebido en los documentos HTML, de forma que se considera muy sencillo procesar información dentro de una aplicación Web desarrollada en el lenguaje, pero obliga a que un programador que pretenda utilizar PHP, conozca a fondo las bondades de la programación con el lenguaje HTML.

Las aplicaciones para la Web, en ocasiones requieren del manejo y control de un conjunto amplio de datos que se intercambian en la red; PHP como todos los lenguajes de tipo scripting utilizan para ello Servidores Web como Apache y gestores de bases de datos como MySql. Los interesados en PHP, deben incrementar a su base de conocimientos, el lenguaje SQL para manejo de bases de datos que es aplicable a los diferentes gestores, y el manejo de un servidor WEB para poder interactuar en la red, ya sea en el ambiente Windows o en el ambiente LINUX.

Esta primera sesión propone, entonces, revisar los conocimientos necesarios para poder aprender el poderoso Lenguaje PHP.

Instalación y configuración del intérprete

Definiendo la instalación

El lenguaje PHP, se puede trabajar de tres formas: Ejecutando los scripts en el servidor, Ejecutando los scripts en la línea de comandos y En aplicaciones gráficas en clientes. La primera forma de trabajo es la más común y sobre la cual se enfoca el desarrollo de la guía. Se requiere entonces

el Intérprete del lenguaje PHP, el servidor Web y un navegador. El programa en PHP, es ejecutado y visto el resultado en el navegador. Al usuario le es transparente la interacción con el servidor y el intérprete.
La plataforma de Sistema Operativo donde se instala PHP define también el servidor Web a instalar. Son ejemplos Apache, IIS (Internet Information Server), PWS (Personal Web Server), Caudium, fhttpd, Nestcape, iPlanet, OmniHTTPd, Oreilly Website Pro, Sambar, Xitami, etc. Otra posibilidad es contratar el servicio de servidor en la red de forma que no se requiera instalarlo y se ejecuten los programas sin instalar el servidor en el equipo de trabajo.
En plataformas operativas Windows 95 o Windows 98 y para programadores nuevos de PHP, podría ser aconsejable utilizar el servidor PWS, que también sirve para aprender ASP. Otra posibilidad es utilizar el servidor Apache que a la larga puede ser una opción más completa. No obstante que el origen del servidor Apache es el ambiente UNIX o LINUX, ya existe una versión domo la
creada de forma específica para ambiente Windows.

Existen en la actualidad muchas versiones auto extraíbles de PHP que, son rápidas y fáciles de instalar. Este tipo de versiones no trae todas las herramientas y funciones del lenguaje pero son útiles cuando se está aprendiendo.
En las plataformas Windows NT, Windows 2000 y Windows XP, se puede utilizar de forma confiable IIS o Apache. Es bueno que los usuarios de PHP tengan en cuenta que PHP fue pensado para trabajar sobre Apache a la hora de tomar una decisión. IIS es una buena opción si se pretende ejecutar PHP y ASP en el mismo entorno.
El lenguaje PHP, presenta su mayor rendimiento en entornos operativos Unix o Linux y con la utilización de un servidor Apache, que es la combinación más común en la mayoría de los servidores de Internet (php.net, 2001).
Instalación de PHP

Como PHP, se puede trabajar en diferentes plataformas y con diferentes servidores existen múltiples combinaciones de realizar la instalación. De acuerdo al servidor escogido podemos diferenciar varios modos de instalación:

Como un intérprete externo al servidor Web (modo CGI).
Como un módulo interno estático o dinámico del servidor Apache. Como módulo ISAPI sobre Internet Information Server.
Como módulo NSAPI sobre Netscape Enterprise Server (php.net, 2001).

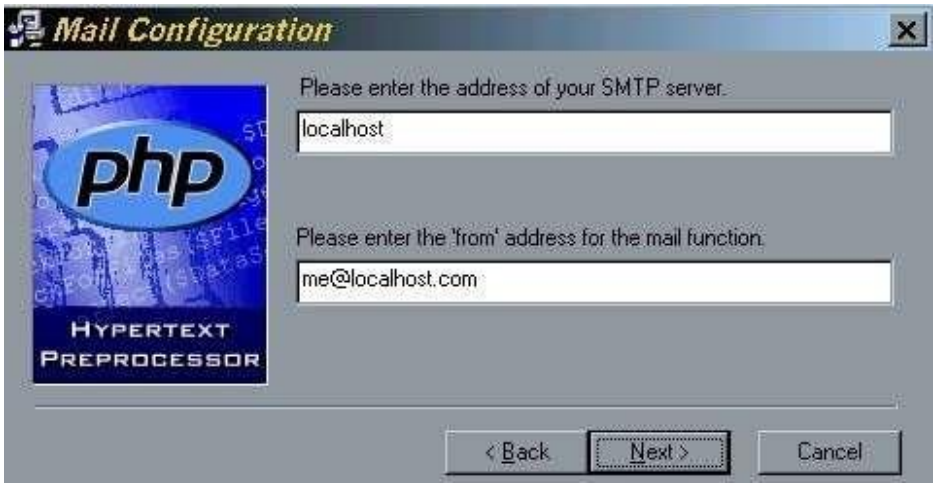
Instalación en Sistemas Win32 (Modo CGI)

Existen tres formatos del software de instalación en sistemas Win32.

Una versión instalable en modo CGI, que es ejecutada como cualquier aplicación Windows, y que paso a paso indica cómo se debe instalar el intérprete de PHP. La última versión 4.3.3 (1046Kb), contiene las principales funciones del lenguaje, y resulta ideal para usuarios nuevos. Después de la respectiva bienvenida, el despliegue y aceptación de la licencia, el usuario puede seleccionar una instalación estándar o una avanzada. En las siguientes ventanas se configura el directorio donde se almacena los archivos uploads y el directorio para almacenar las sesiones.

Finalizando la instalación podemos entonces configurar el correo electrónico: La dirección del servidor SMTP que generalmente es localhost y la dirección de correo que aparece en los mensajes de correo enviados desde los scripts PHP.

La siguiente ventana le permite configurar el nivel de notificación de errores bajo las tres opciones siguientes: Mostrar todos los errores, avisos y advertencias; Mostrar todos los errores y avisos; Mostrar todos los errores. Preferiblemente después de seleccionar la primera opción, se escoge el servidor Web seleccionado para el entorno PHP.



Luego de seleccionar el servidor se escoge la extensión de los archivos asociados al intérprete que por recomendación debe ser .php. Finalmente se procede a la instalación del intérprete. Al instalar el servidor Apache aún faltará la configuración del servidor para que el intérprete y sus archivos sean reconocidos. (Esto no sucede con otros servidores) (php.net, 2001).

Estos son los pasos a seguir para la configuración de PHP en Apache:

- Localice el archivo de configuración de Apache Server httd.conf.
- Realice una copia de seguridad antes de hacer cualquier modificación. (Por sí acaso)
- Por ser un archivo de tipo texto lo puede editar sin ningún problema, así que abra el archivo para modificarlo en el editor de su preferencia.
- En las directivas ScriptAlias agregue la siguiente: ScriptAlias /php4/
- "C:/PHP/"
- En las directivas AddType agregue la siguiente: AddType application/x-httpd-php4 .php
- En las directivas Action agregue la siguiente: Action application/x-httpd-php4 "/php4/php.exe"

Instalación en Sistemas Win32 (Modo ISAPI)

Este modo de instalación es más completo que el anterior. La versión se encuentra en formato ZIP, donde se incluye la versión CGI, librerías y utilidades que no están en la versión instalable descrita anteriormente. Las instrucciones de instalación y configuración se encuentran en el archivo install.txt que viene en el conjunto de archivos comprimidos. Como Apache es el servidor más utilizado describimos aquí los pasos para realizar su respectiva configuración (php.net, 2001).

- Descomprima la carpeta en el directorio deseado, para ejemplo utilizamos "C:\PHP"
- Localice el archivo de configuración de Apache Server httd.conf.
- Realice una copia de seguridad antes de hacer cualquier modificación. (Por sí acaso)
- Por ser un archivo de tipo texto lo puede editar sin ningún problema, así que abra el archivo para modificarlo en el editor de su preferencia.
- En las directivas LoadModule agregue: LoadModule
- php4_module c:/php/sapi/php4apache.dll
- En las directivas AddType agregue: AddType application/x-httpd-php
- .php
- Mueva el archivo php4ts.dll al directorio c:\windows\system o c:\windows\system32 para NT.
- Localice el archivo php.ini en su directorio c:\windows
- Muévalo como copia de seguridad a otro directorio o cámbiele de nombre, ejemplo:
- phpseg.ini.
- Copie el archivo php.ini-dist en el directorio c:\windows como php.ini.
- Reinicie su sistema, active nuevamente su servidor Apache para comenzar a trabajar.

Instalación en Sistemas UNIX

Instalar PHP en ambientes UNIX, es una tarea que varía de acuerdo a la plataforma completa en que se vaya a instalar, por tal motivo es indispensable de valerse de la documentación de instalación de PHP en el momento de realizar la instalación y configuración. Un conjunto de pasos genéricos de la instalación sería el siguiente (php.net, 2001):

Descomprimir las últimas versiones disponibles con el código fuente del servidor y del intérprete PHP.

- Configurar el código fuente del servidor Apache:

```
./configure --prefix=/usr/local/apache
```

- Configurar el código fuente de PHP y compilar el intérprete:

```
./configure  --with-mysql \
--with-apache=../apache_ \ (apache=directorio del servidor)
--enable-track-vars
make
make install
```

- En el directorio de módulos para Apache copiar la librería con las funciones para PHP.

```
cp libs/libphp4.a ../apache_/src/modules/php4/
```

- Volver a configurar Apache para incluir el módulo de PHP y construir el Servidor

```
./configure --active- module=src/modules/php4/libphp4.a make
```

make install

- Configurar de acuerdo al sistema el archivo **httpd.conf** y ponerlo en marcha.

Un programa php

Los programas PHP, se escriben compartiendo su código con código HTML. Para identificar cual es la parte correspondiente a PHP y cuál es la de HTML, utilizamos etiquetas, de forma que el servidor pueda enviar al intérprete sin problema la parte de PHP.

<?php Indica el comienzo del código PHP.
?> Indica el final del código

PHP.
<SCRIPT LANGUAGE="PHP"> Indica el comienzo del código PHP.
</SCRIPT> Indica el final del código PHP.

También es posible utilizar el modo abreviado, siempre y cuando este activa la directiva **short_open_tag** en el archivo de configuración **php.ini**.

<? Indica el comienzo del código PHP.
?> Indica el final del código PHP.

La última opción es activar la directiva **asp_tags** en el archivo de configuración y utilizar:

<% Indica el comienzo del código PHP.
%> Indica el final del código PHP.

Las páginas PHP, trabajan de manera similar a cualquier página dinámica de lado servidor: El servidor reconoce la extensión correspondiente a la página PHP (.php,.php4,etc.) y antes de enviarla al navegador se encarga de llamar el intérprete y ejecutar las instrucciones del lenguaje PHP involucradas en las etiquetas. Lo demás lo trabaja como cualquier código HTML.

Cada instrucción o script PHP debe concluirse con el caracter punto y coma ";". La única expresión que no lo requiere pero lo admite es la última colocada antes del cierre de etiqueta.

Los programas PHP, al igual que el código HTML, pueden ser creados en cualquier editor que maneje el tipo texto. Simplemente al nombre del programa se le coloca la extensión.php. Desde el modesto bloque de notas hasta los muy sofisticados como Dreamweaver; incluso existen editores específicos de PHP (Sintes Marco, 2016).

Comentarios

Una de las características principales de los buenos programas es la documentación, y el elemento primordial para hacerla son los comentarios. la forma de incluir estos comentarios en los programas PHP es variable dependiendo si queremos escribir una línea o más :

// Permite incluir comentarios de una línea.

```
# Permite incluir comentarios de una línea.
/* Permite incluir comentarios de varias líneas */
```

Salida de datos

Revisemos una de las funciones de salida de datos propios del lenguaje PHP, con el fin de escribir nuestro lo primeros y pequeños programas:

```
echo ["Listado de Mensajes"] ,[Listado de variables],...,[...];
```

Esta función nos permite imprimir mensajes o contenidos de variables en la página HTML resultante de la ejecución de los programas PHP. De todas maneras el programador puede optar por escribir los mensajes con código HTML. Observe el siguiente programa y luego ejecútelo (Manualdephp, s.f.).

```
<HTML>
<HEAD><TITLE>Prueba de programa en PHP</TITLE></HEAD>
<BODY>
Mensaje escrito en código HTML <BR>
<?PHP
#         Comentario 1 en código PHP
echo "Mensaje corto escrito en código PHP <BR>";
//         Comentario 2 en código PHP
echo "Mensaje largo escrito para mostrar ejemplo de código PHP
<BR>";
/* Comentario 3 de varias líneas. El programa explica la forma como escribir diferentes
tipo de mensajes utilizando código HTML y PHP */
?>
Adiós y Suerte aprendiendo PHP.
</BODY>
</HTML>
```

VARIABLES Y TIPOS DE DATOS

Una variable es un espacio, que se crea en la memoria del computador, que está destinado a guardar información durante el tiempo que dure la ejecución de un programa. Como su nombre lo indica la información almacenada puede cambiar durante este tiempo. A diferencia de los lenguajes estructurados, donde es necesario declarar y preparar las variables para que puedan almacenar datos de determinado tipo, en PHP las variables son creadas en el momento de empezar a utilizarlas y el tipo de información que almacena se define en el mismo momento en que se le almacena información.

Para definir variables en PHP utilizamos el caracter especial "$" anteponiéndoselo al nombre de la variable. El tipo de dato entonces depende del valor que le asignemos a la variable. Los nombres de las variables deben seguir algunas reglas:

El nombre empieza por el caracter "$" y continúa con una cadena de caracteres: $nombre,
$sueldo, $f.

La cadena debe empezar por una letra, pero puede tener números al interior y además acepta el caracter subrayado. Nunca puede empezar por un número:

$ala_norte, $clave11, $sitio1as.

Los nombres aceptan minúsculas y mayúsculas pero las diferencian: $Edad es diferente de $eDAd (elladodelgeek.com & El Youzghi, 2015).

Datos numéricos

Los tipos de datos numéricos pueden ser enteros o reales. Los números enteros no manejan cifras decimales y los reales sí. Los números enteros se pueden manejar en formato hexadecimal con el prefijo Ox u octal con el prefijo O. Los reales se pueden representar también en notación científica utilizando el carácter "e". Definimos datos numéricos cuando le asignamos información así (Gonzalez, 2016):

```
$numero = 256;
$Valor = 10254;
$Num1 = OxFF;
$Num2 = O17;
$Sueldo = 1636250.50;
$Xn = 0.4256e3;
```

Datos alfanuméricos.

Conjunto de letras y números llamados generalmente cadenas de caracteres se identifican al colocarlos entre comillas dobles (") o entre comillas sencillas ('). Si deseo mostrar como contenido del mensaje alguna de estas comillas, utilizo las otras para indicar que es cadena de caracteres, así (Gonzalez, 2016):

```
$Direccion = "Calle 5 sur #20-22";
$Leyenda = ' Dijo el coronel "Salve usted la patria" ';
$Nombre = 'Jesús Ramírez';
$Frase = "Utilice el signo '+' para la suma";
```

Como en C y en Perl, el carácter de barra invertida ("\") se puede usar para especificar caracteres especiales:

SECUENCIA	SIGNIFICADO
\n	Nueva
\r	Retorno de carro
\t	Tabulación horizontal
\\	Barra invertida
\$	Signo de pesos
\"	Comillas dobles

Datos Booleanos

Estos datos solo pueden tomar como contenido los valores False (Falso) o True (Verdadero), observemos (Gonzalez, 2016):

$Clave = True;
$Motivo
= False;

Datos de tipo Arreglo

Un arreglo es un conjunto de valores asociados a un único nombre de variable. A diferencia de la mayoría de los lenguajes en PHP, sí pueden ser de diferente tipo cada uno de los valores contenidos dentro del arreglo. El arreglo se define al asignarle el primer valor, y a medida que se le asignan valores al arreglo se define su tipo de dato. ejemplo:

$Semana[0] = "Lunes";
$Semana[1] = "Martes";

Si no indicamos el índice del elemento, el lenguaje lo asume automáticamente así (Gonzalez, 2016):

$Cuadro[]= "Primero";
$Cuadro[]=2;
$Cuadro[]=True;

Ámbito de las variables

Dos tipos de variables se pueden definir de acuerdo al ámbito donde se trabajan: Locales y Globales. Las variables locales se definen y trabajan dentro de una función, pero al abandonarla estás dejan de existir y sus contenidos también a menos que se hayan definido como estáticas. Las variables globales en cambio son reconocidas en cualquier parte de la aplicación y sus contenidos se mantienen disponibles en todo momento (Gonzalez, 2016).

Valores desde formularios

Una tarea importante, es la recuperación de valores a través de los formularios creados en código HTML. Esto es realmente sencillo, solo basta con referenciar los nombres asignados a las propiedades NAME de los campos con los nombres de las variables. Observe el siguiente ejemplo:

Este primer programa se almacena con el nombre de **prueba.php**.

<HTML>
<BODY>
<H1> Programa de Captura de Datos </H1>
<?PHP
ECHO "El nombre Capturado es : "; ECHO

```
$NOM;
?>
</BODY>
</HTML>
```

Este segundo programa se almacena con cualquier nombre. Es el programa a ejecutar primero y desde el cual se llama el anterior (prueba.php).

```
<HTML>
<BODY>
<FORM NAME="FORMULARIO" ACTION="PRUEBA.PHP"> Mi Nombre :
<INPUT TYPE="TEXT" NAME="NOM" SIZE="30">
<INPUT TYPE="SUBMIT" VALUE="ENVIAR">
<INPUT TYPE="RESET" VALUE="BORRAR">
</FORM>
</BODY>
</HTML>
```

Si el programa inicialmente genera error, asegúrese que en el archivo **php.ini** fue activada la directiva register_globals así:register_globals = on.

Otras variables importantes

Variables de variables.

En PHP, es posible crear variables especiales, que contienen como contenido nombres de otras variables. Esta funcionalidad es aprovechada cuando se desea generar código dinámico (Gonzalez, 2016).

```
$Nombre = "Jorge";    Asignación normal de una variable.
$$Nombre = 27;        Asigna la cantidad 27 a una variable nueva llamada Jorge.
echo $Jorge;   Imprime el número 27.
```

Variables de PHP

El lenguaje PHP, en sus librerías coloca a disposición del usuario un conjunto de variables de tipos especializados de datos, que pueden ser utilizadas en cualquier momento (Gonzalez, 2016):

VARIABLE	DESCRIPCIÓN
$HTTP_USER_AGENT	Informa sobre el sistema operativo y tipo de navegador y versión del mismo utilizado por el
$HTTP_ACCEPT_LANGUAGE	Devuelve la o las abreviaciones del idioma seleccionado como principal del navegador.
$HTTP_REFERER	Indica la URL de acceso por parte del usuario a la
$PHP_SELF	Indica una cadena con la URL del script que está siendo ejecutado por el servidor.
$HTTP_GET_VARS	Es un arreglo que almacena los nombres y contenidos de las variables enviadas al script por
$HTTP_POST_VARS	Es un arreglo que almacena los nombres y contenidos de las variables enviadas al script por medio de un
$HTTP_COOKIES_VARS	Es un arreglo que almacena los nombres y contenidos de las cookies relacionados con la
$PHP_AUTH_USER	Almacena la variable; usuario cuando se efectúa la entrada a páginas de acceso restringido.
$PHP_AUTH_PW	Almacena la variable password cuando se efectúa la entrada a páginas de acceso restringido.
$REMOTE_ADDR	Muestra la dirección IP del visitante.
$DOCUMENT_ROOT	Indica la ruta física en que se encuentra alojada la página en el servidor.
$PHPSESSID	Almacena el identificador de sesión del usuario.

Funciones de apoyo

Existen algunas funciona que ayudan a controlar el buen uso de las variables, cuando se envía sus valores desde formularios. Tenemos:

isset(Variable) Devuelve True si la variable ya está definida. empty(Variable) Devuelve True si la variable ya definida no tiene asignado valor.
Unset(Lista Variables) Libera recursos de las variables pasadas como parámetros

OPERADORES

Como todo lenguaje, PHP cuenta con un conjunto de operadores que permiten la construcción de expresiones, condiciones, estructuras, etc. que describimos a continuación. Los operadores se comportan respetando un jerarquía de operación que es importante tenerla en cuenta al momento de la programación (Php.net, 2001).

Asignación

Como en la mayoría de lenguajes el principal operador de asignación del lenguaje PHP es el caracter igual ("="). Su estructura de utilización es (Php.net, 2001):
Variable = [Constante][Variable][Expresión]; El conjunto completo de operadores es:

Operador Descripción

= Asignación de Valor

+= $Var = $Var +

-= $Var = $Var -

*= $Var = $Var *

/= $Var = $Var /

.= $Cad = $Cad .

Aritméticos

Estos operadores permiten realizar operaciones matemáticas entre las diferentes entidades componentes de una aplicación. Son iguales en función a los utilizados en lenguaje C (Php.net, 2001).

Operador	Descripción
+	Suma de dos entidades
-	Resta de dos entidades
*	Multiplicación de dos entidades
/	División entre dos entidades
%	Módulo de la división
++	Preincremento y Postincremento
--	Predecremento y Postdecremento

.	Concatena Cadenas (Suma)

Relacionales

Estos operadores relacionan dos entidades y devuelven un valor booleano (False o True) (Php.net, 2001).

Operador	Descripción
==	Igualdad diferente tipo de dato
===	Igualdad mismo tipo de dato
!=	Diferente
!===	Diferente en mismo tipo
<	Menor que
>	Mayor que
<=	Menor igual
>=	Mayor igual

Lógicos

Los siguientes operadores evalúan expresiones, y relaciones para determinar como respuesta un valor booleano, dependiendo de la relación lógica (Php.net, 2001).

Operador	Descripción
and	Y
or	O
xor	O exclusivo
!	Negación
&&	Y
||	O

De bits

Estos operadores realizan operaciones sobre el componente en binario de la información almacenada en las variables (Php.net, 2001).

Operador	Descripción

&	Y
|	O
^	O exclusivo
~	Negación
<<	Corrimiento a la izquierda
>>	Corrimiento a la derecha

Constantes

Las constantes como su nombre lo indica son espacios asignados en la memoria del computador, cuyo contenido se tiene la certeza de que no va a cambiar durante la ejecución del programa. En PHP las constantes se defines con la función define (Velásquez, 2013):

```
define("Constante",15);
define("EdadMayor",18);
define("A",3.5);
```

Expresiones

Una expresión es un conjunto y combinación de variables, constantes, funciones y operadores que ordenados sintáctica y semánticamente de forma correcta, permiten generar valores de resultado de diferentes tipo, como booleanos, numéricos o alfanuméricos (Velásquez, 2013). Ejemplos:

($a >= 5) && ($Valor < $Incremento)	Genera un valor Booleano.
$b * $b + ($c - ($a / 4))	Genera un valor numérico.
"Francisco González" . $h . $a	Genera un valor alfanumérico.

ESTRUCTURAS DE SELECCIÓN SIMPLES

If.

La estructura permite la ejecución condicional de fragmentos de código PHP. Si la condición se evalúa como TRUE, PHP ejecutará el conjunto de instrucciones, y si se evalúa como FALSE las ignorará (González, Bucles en PHP, 2006).

```
if (condición)
{
instrucciones... instrucciones...
}
```

Las condiciones se construyen a través de operadores relacionales y lógicos. El siguiente ejemplo compara dos números y determina cual es el mayor.

```
<HTML>
<BODY>
```

```
<H1> Programa de Mayor número </H1><BR>

<?PHP
$a=7;
$b=3;
if ($a > $b)
{
echo "$a es mayor que $b";
$b = $a;
}
?>
</BODY>
</HTML>
```

El siguiente ejemplo determina si un ciudadano tiene derecho a votar o no. Se incluye la función isset para asegurarse que las variables tienen valor asociado desde el formulario y pueden ser procesados. Caso contrario aparecerá un mensaje de error por estar las variables vacías.

```
<HTML>
<BODY>
<FORM NAME="FORMULARIO"> Mi Nombre :
<INPUT TYPE="TEXT" NAME="NOM" SIZE="30"> Edad :
<INPUT TYPE="TEXT" NAME="EDAD" SIZE="4">
<BR>
<INPUT TYPE="SUBMIT" VALUE="ENVIAR">
<INPUT TYPE="RESET" VALUE="BORRAR">
<BR>
<?PHP
if (isset($NOM) && isset($EDAD))
{
if ($EDAD >= 18)
{
ECHO "Señor : "; ECHO $NOM;
ECHO " Usted es mayor de edad y puede votar";
}
}
?>
</FORM>
</BODY>
</HTML>
```

If - Else

La estructura permite la ejecución condicional de fragmentos de código PHP. Si la condición se evalúa como TRUE, PHP ejecutará un conjunto específico de instrucciones, y si se evalúa como FALSE ejecutará un conjunto diferente de instrucciones (González, Bucles en PHP, 2006).

```
if (condición)
{
instrucciones... instrucciones...
}
else
```

{
instrucciones... instrucciones...
}

El ejemplo anterior puede ser modificado, para que aplicando una estructura if- else también determine si es menor de edad.

```
if (isset($NOM) && isset($EDAD))
{
if ($EDAD >= 18)
{
ECHO "Señor : "; ECHO $NOM;
ECHO " Usted es mayor de edad y puede votar";
}
else
{
ECHO "Señor : "; ECHO $NOM;
ECHO " Usted es menor de edad y no puede votar";
}
}
```

If - Else - If

La estructura elseif, es una combinación de las estructuras if y else - if. En este caso, por la parte else de la estructura, se extiende una nueva sentencia if para ejecutar un conjunto de instrucciones diferente en caso de que la expresión if original se evalúe como FALSE. Sin embargo, a diferencia de else, se ejecutará la expresión alternativa solamente si la expresión condicional elseif se evalúa como TRUE (González, Bucles en PHP, 2006).

```
if (condición)
{
instrucciones...
}
else if (Condición)
{
instrucciones...
}
:
:
else if (Condición)
{
instrucciones...
}
else
{
instrucciones...
}
```

El siguiente ejemplo muestra cómo se determina si un número es mayor que otro, o si son iguales a través de esta nueva estructura.

```
<HTML>
<BODY>
<FORM NAME="FORMULARIO"> Primer Número :
```

```
<INPUT TYPE="TEXT" NAME="a" SIZE="5"> Segundo Número :
<INPUT TYPE="TEXT" NAME="b" SIZE="5">
<BR>
<INPUT TYPE="SUBMIT" VALUE="ENVIAR">
<INPUT TYPE="RESET" VALUE="BORRAR">
<BR>

<?PHP
if (isset($a) && isset($b))
{
if ($a > $b)
{
echo "a : $a es mayor que b : $b";
}
elseif ($a == $b)
{
echo "a : $a es igual que b : $b";
}
else
{
echo "b : $b es mayor que a : $a";
}
}
?>
</FORM>
</BODY>
</HTML>
```

Switch

La estructura switch es similar a una serie de estructuras if en la misma expresión. En muchas ocasiones, es necesario comparar la misma variable con muchos valores diferentes, y ejecutar una parte de código distinta dependiendo de a qué valor es igual. Para ello sirve la sentencia switch (González, Bucles en PHP, 2006).

```
switch (variable)

{

case res1: instrucciones... break; case res2: instrucciones... break; case res3: instrucciones... break;
...

default: instrucciones...

}
```

Las instrucciones de la opción default, se ejecutan cuando la variable no ha tomado ninguno de los otros valores posibles. la instrucción break lleva el control del programa al final de la estructura. A diferencia de otros lenguajes en PHP es posible no-solo evaluar el contenido de una variable, también el de una expresión. Observemos las siguientes fracciones de código, donde se evalúa una variable de tipo entero y otra de tipo cadena:

```
switch ($i)

{
case 0: echo "i es igual a 0"; break; case 1: echo "i es igual a 1"; break; case 2: echo "i es igual a 2"; break; default: echo "i no es igual a 0, 1 o 2";
}

*******
switch ($equipo)

{

case "Flamengo": echo "Equipo de Brasil"; break;

case "Boca Junior": echo "Equipo de Argentina"; break; case "Manchester": echo "Equipo de Inglaterra"; break; default: echo "Equipo del resto del mundo";
}
```

Break

La instrucción break, es utilizada para abandonar estructuras de control sin que se halla ejecutado en su totalidad. puede ser utilizada en estructuras de selección o en estructuras iterativas.

Continue

Como antesala de las estructuras iterativas, adelantamos que la instrucción continue puede ser utilizada para interrumpir la ejecución normal de la estructura y continuar con la siguiente iteración.

ESTRUCTURAS ITERATIVAS

While

La forma como trabaja la estructura while es simple. Le dice a PHP que ejecute la(s) instrucciones contenidas en la estructura repetidamente, mientras que la condición while se evalúe como TRUE. El valor de la condición es evaluado cada vez al principio del ciclo, antes de iniciar una iteración (González, 2006).

```
while (Condición)
{
instrucciones ... instrucciones ...
}
```

El siguiente fragmento de código PHP, utiliza la estructura while para imprimir la tabla de multiplicar del número 2 para sus primeras 10 posiciones.

```
<?php
$i = 1;
$j = 2;
while ($i <= 10)
{

$k=$i*$j;
echo "$i * $j = $k <BR>";
$i++;

}
?>
```

Do - while

Las estructuras do..while son similares a las estructuras while, ejecutan un conjunto de instrucciones mientras la evaluación de la condición sea TRUE, con la diferencia que las condiciones se comprueban al final de cada iteración en vez de al principio. En la estructura do - while se garantiza la ejecución de la primera iteración si la primera evaluación de la condición es FALSE (González, 2006).

```
do{ instrucciones ...
```

instrucciones ...
}while (Condición);

El siguiente ejemplo imprime los primeros 20 número enteros.

```
<?php
$i = 1;
while ($i <= 20)
{
echo "Entero: $i <BR>";
$i++;
}
?>
```

ESTRUCTURAS ITERATIVAS

For.

Las estructuras for son los ciclos más complejos en PHP. Se comportan de forma similar a los de C. Sin embargo no son estructuras para, como las de algunos lenguajes, son más bien estructuras while compuestas.

```
for ( Inicializaciones ; Condiciones ; instrucciones )
{
instrucciones ... instrucciones ... instrucciones ...
}
```

Las inicializaciones se ejecutan incondicionalmente una vez al principio del ciclo; Si son varias se separan por comas (,). Las condiciones se evalúan al comienzo de cada iteración; Si se evalúan como TRUE, el ciclo continúa y se ejecutan las instrucciones de la estructura; Si se evalúan como FALSE, la ejecución del ciclo termina. Al final de cada iteración, se ejecutan las instrucciones; Si son varias se separan por comas (,).

Cada una de las partes de la estructura puede estar vacía. En el caso de las condiciones el vacío significa que la evaluación se considera como TRUE. El siguiente programa es una variación de la tabla de multiplicar del 2 pero utilizando estructura for (González, 2006).

```
<?php
for ($i=1,$j=2;$i <= 10;$i++)
{
$k=$i*$j;
echo "$i * $j = $k <BR>";
}
?>
```

Foreach

La estructura foreach está especializada, en el recorrido de estructuras complejas como arreglos. foreach recorre directamente cada uno de los elementos y asignarles un nombre que sea más fácil de procesar (González, 2006).

```
foreach (Arreglo As Variable)
{
instrucciones ... instrucciones ...
}
```

El siguiente ejemplo muestra un cómo se utiliza la estructura:

```
<?php
$Nombres[]="Zico";
$Nombres[]="Pelé";
$Nombres[]="Kaká";
$Nombres[]="Cafú";
$Nombres[]="Zizú";
$Nombres[]="Coco";
$i=1;
echo "<B>Ciclo foreach</B><BR>\n"; foreach ($Nombres as $autor)
{
echo "Jugador $i:<B> $autor</B><BR>\n";
$i++;
}
?>
```

SINTAXIS ALTERNATIVA

PHP ofrece una sintaxis alternativa para alguna de sus estructuras de control; a saber, if, while, for, y switch. En cada caso, la forma básica de la sintaxis alternativa es cambiar abrir-llave por dos puntos (:) y cerrar-llave por endif;, endwhile;, endfor;, or endswitch;, respectivamente (González, 2006). Observemos algunos ejemplos:

```
if ($a==5):
echo “A es igual a 5 “; endif;

if ($a == 5):

echo "a es igual a 5"; echo "..."; elseif ($a == 6):
echo "a es igual a 6"; echo "!!!"; else:
echo "a no es ni 5 ni 6"; endif;

$i = 1;
while ($i <= 10): echo $i;
$i++; endwhile;

for (i=0; i<10; i++): echo $i;
endfor;

switch ($i): case 0:
echo "i es igual 0"; break; case 1:
echo "i es igual a 1"; break; case 2:
echo "i es igual a 2"; break; default:
echo "i no es igual a 0, 1 o 2"; endswitch;
```

FUNCIONES

Una función, es un conjunto de instrucciones que realizan una tarea específica, se separan del programa principal, para en algún momento ser utilizadas. Las funciones en PHP al igual que en C, se pueden comportar de dos formas como un procedimiento o como una función como tal. La diferencia radica en que el procedimiento solo se limita a ejecutar las tareas asignadas, en cambio la función retorna un valor de respuesta para que sea utilizada en el programa o subprograma desde donde fue invocado. Tanto funciones como procedimientos pueden recibir valores para ser utilizados en su tarea llamados argumentos (Php.net, 2001):

```
function nombre ($arg_1, $arg_2, ..., $arg_n)
{
instrucciones ... instrucciones ... return $retval;
}
```

Dentro del cuerpo de una función se puede utilizar cualquier instrucción válida de PHP. A diferencia de lenguajes como C, en PHP4 no es necesario que las funciones se definan antes de ser referenciadas, pero no soporta no soporta la sobrecarga de funciones, ni es posible redefinir u ocultar funciones previamente declaradas. En cambio se es posible en PHP declarar funciones con longitud variable de parámetros, y soportar parámetros por defecto. La información puede suministrarse a las funciones mediante la lista de parámetros, que es una lista de variables y/o constantes separadas por comas. Un ejemplo de una función que eleva un número al cuadrado sería:

```
<?php
function Cuadrado($a)
{
$b=$a * $a; return $b;
}
$c= Cuadrado(5);
echo "El cuadrado de 5 es $c";
?>
```

Parámetros por valor y referencia

Por defecto, los parámetros de una función se pasan por valor, esto significa que el cambiar el valor del argumento dentro de la función, no se verá modificado fuera de ella. Si por el contrario se desea que una función modifique sus parámetros, se pasan por referencia. Para realizarlo se utiliza el operador ampersand (&), anteponiéndolo al nombre del parámetro en la definición de la función (Php.net, 2001):

```
function concatena (&$string)
{
$string .= ' adicionar información.';
}
$str = 'En espera de '; concatena($str);
echo $str; // 'Se escribe: "En espera de adicionar información.'
```

Otra forma de realizar el paso de una variable por referencia a una función que no toma el parámetro por referencia por defecto, es anteponiendo el operador ampersand (&) al nombre del parámetro en la llamada a la función:

```
function concatena ($string)
{
```

```
$string .= ' adicionando información.';
}

$str = 'En esperad de '; concatena(&$str); echo $str; // 'Esto es una cadena, y algo más.'
```

Parámetros por defecto

Una función puede definir valores por defecto para los parámetros escalares como se realiza en el lenguaje C. El valor por defecto debe ser una expresión constante. El lenguaje PHP permite también tener un grupo de parámetros por defecto. En caso de que no sea enviado el valor el argumento tampoco tomará valor alguno. Una regla básica es que cuando se usan parámetros por defecto, estos tienen que estar a la derecha de cualquier parámetro sin valor por defecto. Observemos los siguientes tres ejemplos y luego pruébelos (Php.net, 2001).

```
function comidas ($tipo = "Bandeja Paisa")
{
return "Orden de $tipo.";
}
echo comidas();
echo comidas ("Ajiaco");

function Donas($tipo = "Chocolate", $Sabor)
{
return "Haciendo dona de $type y $Sabor.";
}
echo Donas ("Chantilly");

function Donas ($Sabor, $tipo = "Canela")
{
return "Haciendo dona de $type $Sabor";
}
echo Donas ("Arequipe");
```

INCLUSIÓN DE CÓDIGO DESDE ARCHIVOS

Una de las principales necesidades buscadas en un lenguaje, por los programadores expertos, es la posibilidad de crear librerías. Entendemos como librería, una serie de código en cargado de realizar tareas específicas y que es común para todas las aplicaciones dentro del estilo de programación del usuario. El lenguaje PHP, dispone de dos constructores que permiten incluir código en programa de aplicación, código que se encuentra almacenado en archivos diferentes. Estas son las funciones:
require("Archivo.php") include("Archivo.php)

Lo que realizan las funciones es tomar el código almacenado en el archivo referenciado e incluirlo dentro del conjunto de código desde donde se invoca el archivo. Dentro de las diferencias existentes entre las dos funciones, tenemos que la función require() no acepta inclusión del código dentro de una condicional, mientras que con la estructura adecuada si se podría hacer con la función include() (Rolandocaldas, 2013).

No es posible:

```
require("libro1.php"); if($Archivo=="S")
{
require("libro2.php");
}
```

Si es posible:

```
include("libro1.php"); if($Archivo=="S")
{
include("libro2.php");
}
```

OBJETOS

El lenguaje PHP, permite utilizar conceptos de la POO (Programación orientada a objetos), así que se pueden construir objetos con sus respectivos atributos y métodos, y por supuesto aplicar herencia y redefinición de funciones (Álvarez, 2004). Observemos la estructura para la definición de clases con un ejemplo:
Definición de la clase. Class Empleado

```
{
var $NombreEmp = ""; var $SueldoEmp = 0;
function CapturaN($nombre)
{
$this->NombreEmp = $nombre;
}
function CapturaS($Basico)
{
$this->$SueldoEmp = $Basico;
echo $this->NombreEmp. " Gana : ". $this->$SueldoEmp;
}
}
```

Instrucciones de ingreso de datos:

```
$Emp = new Empleado;
$Emp->CapturaN("Jacinto Suárez");
$Emp->CapturaS(1500000);
```

MATRICES EN PHP

Cuando se definieron los datos de tipo Arreglo, se adelantó en gran parte del concepto de Matrices. Una Matriz es un conjunto de variables asociadas a un único nombre. A diferencia de otros lenguajes cada variable o elemento de la matriz puede manejar un tipo de dato diferente. En PHP se pueden tener dos tipos de Matrices: Matrices Indexadas o Matrices Asociativas (php.net, 2001).

En las Matrices Indexadas cada elemento de la matriz se encuentra numerado desde la posición 0, y la referencia del contenido del elemento se realiza a través de éste índice. Estos son ejemplo de una Matriz indexada con datos del mismo tipo y con datos de diferente tipo.

```
$Autos[0] = "Mazda";
$Autos[1] = "Renault";
$Autos[2] = "Chevrolet";
$Autos[3] = "Mitsubishi";
$Autos[4] = "Dodge";
$Hoja[0] = "Patricia Hernández";
$Hoja[1] = 33;
$Hoja[2] = 2506000;
$Hoja[3] = "52.054.019";
$Hoja[4] = "Enfermera";
```

Las Matrices asociativas, se conforman por pares clave-valor para realizar el acceso de acuerdo a la clave determinada. Adaptando el segundo ejemplo tenemos:

```
$Hoja['Nombre'] = "Patricia Hernández";
$Hoja['Edad'] = 33;
$Hoja['Sueldo'] = 2506000;
$Hoja['Cedula'] = "52.054.019";
$Hoja['Cargo'] = "Enfermera";
```

OPERACIONES EN MATRICES

Creación de Matrices

Las Matrices en el lenguaje PHP, pueden ser creadas de dos formas; de Forma implícita o utilizando alguna de las funciones especializadas array() o list().
Los ejemplos vistos anteriormente son creaciones de tipo implícito, donde simplemente se accede a los elementos y se les asigna un valor. Recuerde que también se puede sin utilizar el índice y estos se almacenan en estricto orden (php.net, 2001);

```
$Autos[] = "Mazda";
```

```
$Autos[] = "Renault";
$Autos[] = "Chevrolet";
$Autos[0] = "Mazda";
$Autos[2] = "Renault";
$Autos[4] = "Chevrolet";  // Aquí las posiciones 1 y 3 toman valor null.
$Hoja[0] = "Patricia Hernández";
$Hoja[1] = 33;
$Hoja[2] = 2506000;
```

Mediante la utilización de la función Array, también se pueden crear Matrices indexadas o Matrices asociativas. Observemos un ejemplo para entenderlo más claramente.

```
$Autos = array("Mazda","Renault","Chevrolet","Mitsubishi","Dodge");
$Hoja = array('Nombre' => "Patricia Hernández", 'Edad' => 33,
'Sueldo' => 2506000, 'Cedula' => "52.054.019", 'Cargo' => "Enfermera");
```

Las matrices tratadas se conocen como matrices unidimensionales, arreglos o vectores. PHP también permite crear las matrices bidimensionales y multidimensionales. Es como si en los elementos de una matriz tuviéramos otra matriz y así sucesivamente. Este tipo de matrices también pueden ser Indexadas o Asociativas y se pueden crear implícitamente o por medio de las funciones. En su interior Las matrices de varias dimensiones reciben elementos de diferente tipo (php.net, 2001). Observemos algunos ejemplos:

```
$Matriz[0][0] = "Mazda 323"
$Matriz[0][1] = "Mazda 626"
$Matriz[0][0] = "Renault 9"
$Matriz[0][1] = "Renault 19"
$Matriz[0][2] = "Renault 12"
$Matriz['Juan']['Telefono'] = "2154613";
$Matriz['Julio']['Edad'] = 23;
$Matriz['Julio']['Telefono'] = "2428659"
$Matriz['Jaime']['Edad'] = 13;
$Matriz = array(array('Juan','Fernando','Lucía'),array(11,51,23));
$Matriz = array(array('Nombre' => 'Julio', 'Edad' => 25; 'Sueldo' => 380000),
array("Rosa",1,"Flor"));
```

Recorrido de una Matriz

Recorrer una matriz es acceder a cada uno de sus elementos en un orden determinado, que puede ser del primero al último elemento o del último elemento al primero. Cuando una Matriz es indexada se realiza a través de los índices, pero es importante conocer cuántos elementos tiene la matriz, lo que hacemos con la función count(). Cuando la Matriz es asociativa, sólo es posible hacerlo a través de la función each(), que recupera los pares de clave y valor y avanza a través de cada una de las posiciones en la matriz (php.net, 2001).

```
for ($i=0; $i < count($Matriz); $i++) echo "$Matriz[$i]";
for(;$elem = each($Matriz);)
{
echo " $Elemento[0] <BR>"; echo " $Elemento[1] <BR>";
}
```

Navegación en una Matriz

Navegar sobre una matriz es sencillo en el caso de Matrices indexadas porque simplemente se hace referencia a la posición en la Matriz, en el caso de las Matrices Asociativas existen en PHP una serie de funciones que le permiten realizar dicha navegación (php.net, 2001).
reset(). El puntero va a la primera posición y devuelve su contenido. end(). El puntero va a la última posición y devuelve su contenido. next(). El puntero va a la siguiente posición y devuelve su contenido. prev(). El puntero va a la posición anterior y devuelve su contenido. current(). Determina la posición actual y devuelve su contenido. pos(). Determina la posición actual y devuelve su contenido.
key(). Determina la calve de la posición actual en Matrices Asociativas.

Inserción de Elementos en una matriz

Existen dos formas de adicionar elementos a una matriz; antes de la primera posición o después de la última posición. Para tal efecto usamos las funciones array_push(), cuando adicionamos al final de la matriz y array_unshift(), cuando adicionamos al principio de la matriz. Si utilizamos creación implícita siempre se adiciona al final. Las funciones devuelven un valor entero que representa la cantidad de elementos de la Matriz y se pueden adicionar varios elementos a la vez (php.net, 2001).

```
$Matriz[]=3;
$Matriz = array(1,2,3);
```

Agregamos un elemento al final de la matriz array_push($Matriz,4);

```
array_push(4,5,6); // Almacenados ( 1,2,3,4)
$Matriz = array(1,2,3);
array_unshift($Matriz,4); // Almacenados ( 4, 1,2,3 )
```

Eliminación de Elementos de una matriz

Para la eliminación en cambio se presenta 3 casos posibles: Eliminar el primer elemento de la Matriz, Eliminar el segundo elemento de la Matriz y Eliminar un elemento ubicado en alguna posición dentro de la Matriz, solo que aquí es necesario encontrar primero el elemento que se desea eliminar. Para tal caso PHP también dispone de funciones que permiten realizar esta labor (php.net, 2001).
array_shift(). Elimina el primer elemento de la Matriz. array_pop(). Elimina el último elemento de la Matriz. array_splice(). Elimina el elemento de la Matriz indicado.

Algunos ejemplos:

```
$Matriz = array(1,2,3); array_shift($Matriz); // Almacenados ( 2,3 )
$Matriz = array(1,2,3); array_pop($Matriz); // Almacenados ( 1,2 )
$Matriz = array(1,2,3);
array_splice($Matriz,1,1); // Almacenados ( 1,3 )
```

Ordenamiento de una Matriz

En el lenguaje PHP, existen diversas funciones que permiten el ordenamiento de las Matrices de acuerdo a criterios específicos que se deben aplicar de acuerdo a las

necesidades del usuario (php.net, 2001):

- sort(). Orden ascendente Matrices Indexadas.
- rsort(). Orden descendente Matrices Indexadas.
- asort(). Orden ascendente por valor Matrices Asociativas
- arsort(). Orden descendente por valor Matrices Asociativas.
- ksort(). Orden ascendente por clave Matrices Asociativas
- krsort(). Orden descendente por clave Matrices Asociativas.

CADENAS DE CARACTERES

Las cadenas de caracteres son una secuencia de números y letras que para su identificación en el lenguaje PHP utilizan tres tipos de delimitador a saber: comillas sencillas ('), comillas dobles (") y el conjunto de mayor que (<<<) en el caso de la incrustación de documentos. El uso de las comillas fue visto en la sesión de tipos de datos; La incrustación de documentos se utiliza para permitir representar cadenas de caracteres en varios renglones y se realiza de la siguiente manera (php.net, 2001):
<<<Identificador Cadena de caracteres... continuación cadena... continuación cadena... Identificador;
$Frase = <<<INI

Estas frases serán representadas como única cadena así cambien de línea. INI;

MANIPULACIÓN DEL CONTENIDO

PHP provee gran cantidad de funciones que permiten trabajar con el contenido de las cadenas de caracteres. Estas funciones le permiten a los desarrolladores presentar las cadenas en diferentes formatos, imprimirlas, modificarlas, etc. Estudiemos algunas de ellas:

Los caracteres disponibles en los lenguajes, generalmente están representados por el código ASCII, donde existe un número entre 0 y 255 que identifica cada carácter. Las funciones que permiten observar esta conversión son chr() y ord()

```
$Cadena1 = chr(65); // Almacena "A"

$Cadena2 = chr(64); // Almacena "@"

$i = Ord($Cadena1); // Almacena 65

$j = Ord("@"); // Almacena 64
```

Otra de las actividades importantes es la visualización de cadenas. El grupo de funciones diferentes a echo, para mostrar cadenas de caracteres es:

print() Muestra el contenido de una cadena de caracteres. print($Cadena);
print("La mejor época de la vida es la de los $Edad"); print("$a * $b = $c");

printf() Muestra los datos con un determinado formato, de acuerdo al tipo de dato.

Obsérvese la tabla:

SÍMBOLO	SIGNIFICADO
%	Símbolo de porcentaje. Indica el inicio de un formato.
b	Número entero en binario.
c	Número entero como caracter.
d	Número entero.
f	Número real.
o	Número entero en octal.
s	Cadena de caracteres.
x	Número entero en hexadecimal en minúsculas.
X	Número entero en hexadecimal en mayúsculas.

- sprintf() Realiza la misma función que printf y almacena el resultado en una variable de tipo cadena.
- sscanf() Toma los datos de una variable de tipo cadena y lo interprete con el formato indicado.

Algunas funciones fueron diseñadas para facilitar las tareas de recuperación de información desde archivos, donde son frecuentes algunos errores cuando se tratan los espacios en blanco o los caracteres de retorno, entre otros. Describamos algunas de ellas.

- chop() Elimina los saltos de línea.
- ltrim() Elimina los blancos a la izquierda de una cadena de caracteres.
- rtrim() Elimina los blancos a la derecha de una cadena de caracteres.
- trim() Elimina simultáneamente los blancos a la izquierda y derecha de la cadena de caracteres.
- str_pad() Ajusta el tamaño de una cadena de caracteres a una longitud determinada, y adiciona un caracter de relleno especificado o blancos por defecto. Se puede indicar también si el relleno se realiza por izquierda, por derecha o de manera proporcional.

Observemos los ejemplos:

```
$cadena = str_pad($cadena,10)

$cadena = str_pad($cadena,25)

$cadena = str_pad($cadena,10,"*",STR_PAD_LEFT)

$cadena = str_pad($cadena,25,"*",STR_PAD_RIGHT)

$cadena = str_pad($cadena,10,"*",STR_PAD_BOTH)
```

str_repeat() Permite repetir una cadena de caracteres un número específico de veces: str_repeat("*",10);
str_repeat("*",$i);

El lenguaje PHP, provee también una serie de funciones destinadas a cambiar la presentación de algunas cadenas de forma rápida, inclusive sin cambiar sus

contenidos.

- strtolower() Convierte una cadena de mayúsculas a minúsculas.
- strtoupper() Convierte una cadena de minúsculas a mayúsculas.
- ucfirst() Cambia todos los caracteres a minúsculas, menos el primero de la cadena.
- ucwords() Cambia todos los caracteres a minúsculas, excepto los primeros de cada palabra.

En algunas ocasiones se requiere realizar cambios dentro de las cadenas, y para ello PHP proporciona las siguientes funciones:

- str_replace() Esta función permite buscar una palabra o conjunto de caracteres en una cadena y remplazarla por otra.
- str_replace("Tirar","Lanzar",$cadena) Cambia Tirar por Lanzar en la variable
- $cadena.
- strtr() Permite cambiar caracteres encontrados por otros en la cadena especificada.
- substr_replace() Reemplaza una porción del contenido de la cadena por otra.

Otro grupo de funciones proporcionadas por el lenguaje PHP, permiten acceder al contenido de la cadena y obtener de él, algún resultado de evaluación que le permita al programador tomar una alternativa de trabajo. Dentro de este grupo tenemos las siguientes funciones:

- strlen() Devuelve la longitud de la cadena (Número de caracteres).
- count_chars() Devuelve el número de veces que hay un caracter dentro de una cadena.
- substr_count() Devuelve el número de veces que hay una subcadena dentro de otra.
- strchr() Determina la primera aparición de un caracter y devuelve la subcadena siguiente desde ese lugar de aparición.
- strrchr() Determina la última aparición de un caracter y devuelve la subcadena desde ese lugar de aparición.
- strstr() Determina la primera aparición de una subcadena teniendo en cuenta mayúsculas y minúsculas y devuelve la subcadena siguiente desde ese lugar de aparición.
- strpos() Determina la primera aparición de un subcadena y devuelve la posición donde se encuentra.
- strrpos() Determina la última aparición de un subcadena y devuelve la posición donde se encuentra.
- substr() Devuelve la porción de cadena original de una posición y una longitud dada.
- Otras funciones de aplicación interesante son las que permiten comparar contenidos de variables de tipo cadena y realizar algunas acciones después de su comparación. PHP proporciona las siguientes funciones:
- strcmp() Compara y determina la igualdad entre dos cadenas, diferenciando caracteres en mayúsculas y minúsculas.
- strcasesmp() Compara y determina la igualdad entre dos cadenas, sin diferenciar caracteres en mayúsculas y minúsculas.
- strncmp() Compara y determina igualdad para los n caracteres iniciales de las cadenas. strnatcmp() Compara y determina igualdad de acuerdo al método natural de comparación, diferenciando caracteres en mayúsculas y

minúsculas.

- strnatcasecmo() Compara y determina igualdad de acuerdo al método natural de comparación, sin diferenciar caracteres en mayúsculas y minúsculas (González, Funciones cadenas PHP: str_replace, strtolower, count_chars, strpos, trim, str_repeat, strstr, chr, 2006).

MANIPULACIÓN DE FECHAS

La manipulación de los datos de tipo fecha es muy importante, cuando se desarrollan aplicaciones que manejan entornos Web. Realizar una manipulación (obtención y validación) adecuada de estos datos, facilita tareas como el establecimiento de tiempos de caducidad, tiempos de espera y de trabajo en cookies y sesiones.

Obtener Fecha y Hora

El lenguaje PHP, utiliza la función time() para determinar la fecha y la hora actual. En realidad time devuelve un valor de entero correspondiente a la marca de tiempo. La marca de tiempo es el número de segundos transcurridos desde el día 1 de enero de 1970 a las 00:00:00 GMT hasta el momento de ejecutada la función.

```
$Hora=time();

echo "Son las $hora";
```

PHP dispone también de una función más manejable por el usuario que devuelve una matriz con los datos ya convertidos de la marca de tiempo a formato comprensible. Esta función es getdate(). La matriz asociativa obtenida contiene los siguientes elementos:

seconds	Segundos
minutes	Minutos
hours	Horas
mday	Día del mes
wday	Día de la semana (0 a 6)
mon	Mes del año
year	Año
yday	Día del año (0 a 364)
weekday	Cadena del día de la semana
month	Cadena del mes
0	Marca de tiempo obtenida

Esta secuencia de programa captura la hora del sistema, directamente desde la matriz:

```
<?php
function listar($matriz)
{
echo "<TABLE BORDER=1 WIDTH=500>\n"; foreach($matriz as $clave =>
$valor)
{
echo "<TR>\n";
echo "<TD ALIGN=CENTER>$clave</TD>\n";
echo "<TD ALIGN=CENTER>$valor</TD>\n"; echo "</TR>\n";
}
```

```
echo "</TABLE>\n";
}
$hora = getdate();
echo "HORA DEL SISTEMA \n"; listar($hora);
?>
```

Existen en el lenguaje PHP, otras funciones especializadas en el manejo de este tipo de datos; se pueden destacar como importantes las siguientes:

- localtime(). Recibe como parámetros la marca de tiempo y un valor booleano, para Determinar si la información se almacena sobre una matriz indexada o una asociativa.
- date(). Devuelve una cadena de caracteres que corresponde a la fecha a la que se le aplica un formato. La función evita el paso sobre la matriz asociativa de las funciones anteriores.
- gmdate(). Cumple la misma tarea de la función date(), pero tiene en cuenta la hora de Greenwich.
- strftime(). Determina el formato de una fecha, de acuerdo el idioma configurado en el sistema.
- Dependiendo de la función setlocale().
- gmstrftime(). Funciona igual a serftime(), pero teniendo en cuenta la hora de Greenwich.

Establecer fecha y hora

Las funciones descritas anteriormente, le permiten al usuario obtener una fecha del sistema en un formato establecido, pero también se requiere en algunas ocasiones determinar una fecha específica para realizar alguna tarea, para este caso el lenguaje PHP provee las siguientes funciones:

- mktime(). Determina la marca de tiempo de una fecha dada como parámetros de la función.
- gmktime(). Determina la marca de tiempo de una fecha dada como parámetros de la función que se supone está en horario greenwich.

Para validar las fechas que se capturan desde una página o portal Web, PHP dispone de las siguientes funciones (Manualdephp, s.f.):

- checkdate(). Determina con TRUE o FALSE si la fecha dada e válida.
- strtotime(). Convierte una cadena don presentación mm/dd/aaaa o dd mm aa y obtiene la Marca de hora.

CONCEPTOS BÁSICOS DE HTTP

El Protocolo de Transferencia de HiperTexto (HyperText Transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. La especificación completa del protocolo HTTP puede obtenerla desde su RFC en www.ietf.org. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web.

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en sencillas operaciones de solicitud / respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.

Los recursos u objetos que actúan como entrada o salida de un comando HTTP están clasificados por su descripción MIME. De esta forma, el protocolo puede intercambiar cualquier tipo de dato, sin preocuparse de su contenido. La transferencia se realiza en modo binario, byte a byte, y la identificación MIME permitirá que el receptor trate adecuadamente los datos.

Las principales características del protocolo HTTP son:

- Toda la comunicación entre los clientes y servidores se realiza a partir de caracteres de 8 bits. De esta forma, se puede transmitir cualquier tipo de documento: texto, binario, etc., respetando su formato original.
- Permite la transferencia de objetos multimedia. El contenido de cada objeto intercambiado está identificado por su clasificación MIME.
- Existen tres verbos básicos (hay más, pero por lo general no se utilizan) que un cliente puede utilizar para dialogar con el servidor: GET, para recoger un objeto, POST, para enviar información al servidor y HEAD, para solicitar las características de un objeto (por ejemplo, la fecha de modificación de un documento HTML).
- Cada operación HTTP implica una conexión con el servidor, que es liberada al término de la misma. Es decir, en una operación se puede recoger un único objeto.
- No mantiene estado. Cada petición de un cliente a un servidor no es influida por las transacciones anteriores. El servidor trata cada petición como una operación totalmente independiente del resto.
- Cada objeto al que se aplican los verbos del protocolo está identificado a través de la información de situación del final de la URL.
- HTTP se diseñó específicamente para el World Wide Web: es un protocolo rápido y sencillo que permite la transferencia de múltiples tipos de información de forma eficiente y rápida. Se puede comparar, por ejemplo, con FTP, que es también un protocolo de transferencia de ficheros, pero tiene un conjunto muy amplio de comandos, y no se integra demasiado bien en las transferencias multimedia (Uma, s.f.).

ETAPAS DE UNA TRANSACCIÓN HTTP

Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

- Un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el campo Location del cliente Web.
- El cliente Web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.
- Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente.
- Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD,…), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada (casi siempre HTTP/1.0) y un conjunto variable de información, que incluye datos sobre las capacidades del browser, datos opcionales para el servidor,…
- El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.
- Se cierra la conexión TCP.
- Este proceso se repite en cada acceso al servidor HTTP. Por ejemplo, si se recoge un documento HTML en cuyo interior están insertadas cuatro imágenes, el proceso anterior se repite cinco veces, una para el documento HTML y cuatro para las imágenes (ccm.net, 2016).
- Estructura de los mensajes HTTP
- El diálogo con los servidores HTTP se establece a través de mensajes formados por líneas de texto, cada una de las cuales contiene los diferentes comandos y opciones del protocolo. Sólo existen dos tipos de mensajes, uno para realizar peticiones y otro para devolver la correspondiente respuesta. La estructura general de los dos tipos de mensajes se puede ver en el siguiente esquema:

Mensaje de solicitud	Mensaje de respuesta
Comando HTTP + parámetros Cabeceras del requerimiento (línea en blanco) Información opcional	Resultado de la solicitud Cabeceras de la respuesta (línea en blanco) Información opcional

La primera línea del mensaje de solicitud contiene el comando que se solicita al servidor HTTP, mientras que en la respuesta contiene el resultado de la operación, un código numérico que permite conocer el éxito o fracaso de la operación. Después aparece, para ambos tipos de mensajes, un conjunto de cabeceras (unas obligatorias y otras opcionales), que condicionan y matizan el funcionamiento del protocolo.

La separación entre cada línea del mensaje se realiza con un par CR-LF (retorno de carro más nueva línea). El final de las cabeceras se indica con una línea en blanco, tras la cual se pueden incluir los datos transportados por el protocolo, por ejemplo, el documento HTML que devuelve un servidor o el contenido de un formulario que envía un cliente .

Comandos del Protocolo

Los comandos o verbos de HTTP representan las diferentes operaciones que se pueden solicitar a un servidor HTTP. El formato general de un comando es:

Nombre del comando	Objeto sobre el que se aplica	Versión de HTTP utilizada

Cada comando actúa sobre un objeto del servidor, normalmente un archivo o aplicación, que se toma de la URL de activación. La última parte de esta URL, que representa la dirección de un objeto dentro de un servidor HTTP, es el parámetro sobre el que se aplica el comando. Se compone de una serie de nombres de directorios y archivos, además de parámetros opcionales para las aplicaciones CGI.
El estándar HTTP recoge únicamente tres comandos, que representan las operaciones de recepción y envío de información y chequeo de estado:

- GET Se utiliza para recoger cualquier tipo de información del servidor. Se utiliza siempre que se pulsa sobre un enlace o se teclea directamente a una URL. Como resultado, el servidor HTTP envía el documento correspondiente a la URL seleccionada, o bien activa un módulo CGI, que generará a su vez la información de retorno.

- HEAD Solicita información sobre un objeto (Archivo): tamaño, tipo, fecha de modificación… Es utilizado por los gestores de cachés de páginas o los servidores proxy, para conocer cuándo es necesario actualizar la copia que se mantiene de un fichero.
- POST Sirve para enviar información al servidor, por ejemplo los datos contenidos en un formulario. El servidor pasará esta información a un proceso encargado de su tratamiento (generalmente una aplicación CGI). La operación que se realiza con la información proporcionada depende de la URL utilizada. Se utiliza, sobre todo, en los formularios.
- Un cliente Web selecciona automáticamente los comandos HTTP necesarios para recoger la información requerida por el usuario. Así, ante la activación de un enlace, siempre se ejecuta una operación GET para recoger el documento correspondiente. El envío del contenido de un formulario utiliza GET o POST, en función del atributo de <FORM METHOD="...">. Además, si el cliente Web tiene un caché de páginas recientemente visitadas, puede utilizar HEAD para comprobar la última fecha de modificación de un fichero, antes de traer una nueva copia del mismo.
- Adicional se han definido algunos comandos, que sólo están disponibles en determinadas versiones de servidores HTTP. La última versión de HTTP, denominada 1.1, recoge estas y otras novedades, que se pueden utilizar, por ejemplo, para editar las páginas de un servidor Web trabajando en remoto.
- PUT Actualiza información sobre un objeto del servidor. Es similar a POST, pero en este caso, la información enviada al servidor debe ser almacenada en la URL que acompaña al comando. Así se puede actualizar el contenido de un documento (ccm.net, 2016).
- DELETE Elimina el documento especificado del servidor. LINK Crea una relación entre documentos.
- UNLINK Elimina una relación existente entre documentos del servidor.

COOKIES

Cuando se navega por la red el usuario visita muchas páginas, algunas bastante complicadas que implementan distintos servicios de Internet. Estas páginas tienen que guardar información característica acerca del usuario. Para ello se tienen mecanismos en el servidor como bases de datos u otro tipo de contenedores, pero hay un mecanismo mucho más interesante de guardar esa información que los propios recursos del servidor, que es el la propia máquina del usuario.

En los computadores se almacenan muchos datos que necesitan conocer las páginas Web cada vez que se accede a la página, estas pequeñas informaciones son las cookies, definidos como: estados de variables que se conservan de una visita a otra en el ordenador del cliente. A manera de protección, de la salud de la máquina del usuario los cookies están muy restringidos. Por ejemplo: sólo se puede almacenar textos, nunca programas o imágenes, los textos no pueden ocupar mucho espacio (1k), y tienen fecha de caducidad.

Un ejemplo de cookies podría ser un contador de las veces que accede un usuario a una página. Podríamos poner una cookie en el ordenador del cliente donde tendríamos una variable que lleva la cuenta de las veces que ha accedido a la página y cada vez que se accede se incrementa en uno. La utilidad principal de las cookies es la de poder identificar al navegador una vez éste visita el sitio por segunda vez y así, en función del perfil del cliente dado en su primera visita, el sitio puede adaptarse dinámicamente a sus preferencias (lengua utilizada, colores de pantalla, formularios rellenados total o parcialmente, redirección a determinadas páginas...).

Una cookie es un conjunto de información integrada por varios elementos que se describen a continuación:

ELEMENTO	CONTENIDO
nombre	Nombre de la cookie
valor	Valor asociado de la cookie
fecha expiración	Fecha de expiración de la cookie
path	Subconjunto URL donde la cookie es válida
dominio	Rango de dominios donde la cookie es
segura	Determina transmisión segura HTTPS o no

Para crear un archivo cookies, modificar o generar una nueva cookie se utiliza la función SetCookie(). Esta función incluye los seis elementos descritos como parámetros pero solo es obligatorio el primero.

Ejemplo:

```
setcookie("InfoPriv",$datos,time()+86400*365);
```

Esta instrucción crea una cookie llamada InfoPriv que tiene como valor el contenido de la variable $datos y que tiene una duración de 1 año a partir de su creación. Las llamadas a la función setcookie() deben ser colocadas antes de la etiqueta HTML. Si una cookie se crea con el mismo nombre de una ya existente la borra, es decir que sobre escribe el archivo texto que identifica la cookie. Una precaución importante es no definir variables en nuestro script con el mismo nombre que las cookies porque

PHP dará prioridad al contenido de la variable local sin mostrar mensaje de error.

```
<?php
$nombre = "InfoPriv";
$datos = 105;
setcookie($nombre, $datos, time()+86400*365, "", "", 0); echo "Cookie:
$nombre, $datos";
?>
```

Para eliminar una cookie, solo basta con realizar un nuevo envío utilizando como único parámetro el nombre de la cookie que se desea borrar.

```
<?php
$nombre = "InfoPriv"; setcookie($nombre); echo "Cookie: Borrada";
?>
```

En PHP es posible recuperar en una variable tipo array el conjunto de cookies almacenadas en el disco duro del ínter nauta mediante la variable de servidor $HTTP_COOKIES_VARS (allaboutcookies.org, 2015)

ABRIR FICHEROS

PHP dispone de la función fopen() para abrir ficheros, y su estructura es la siguiente:
int fopen (string fichero, string modo [, int ruta]);

La función fopen() nos devuelve un valor numérico (indicador de archivo) de tipo entero que nos servirá para hacer referencia al archivo abierto.
El parámetro fichero le indica a la función el nombre del archivo que se desea abrir, el parámetro modo determina la forma de acceso al archivo (Obsérvese tabla), y el parámetro ruta es opcional y sólo toma el valor 1 para que tome la directiva include_path del fichero php.ini (php.net, 2001)

ATRIBUTO	DESCRIPCIÓN
r	Sólo lectura
r+	Lectura y escritura
w	Sólo escritura. Si no existe el archivo lo crea, si ya existe lo sobre escribe.
w+	Lectura y escritura. Si no existe el archivo lo crea, si ya existe lo sobre escribe.
a	Solo lectura. Sino existe el archivo lo crea, si ya existe empieza a escribir al final del archivo.
a+	Lectura y escritura. Sino existe el archivo lo crea, si ya existe empieza a escribir al final del

Ejemplo: $ind = fopen("Archivo1.txt","r");

PHP utiliza la función die(), para que en caso de error al intentar abrir un archivo se despliegue un mensaje que indique el tipo de error que ocurrió:

$ind = fopen("Archivo1.txt","r") or die("Error de Apertura");

La función utilizada para cerrar un archivo es fclose(), que devuelve TRUE si no existe problema al cerrar o FALSE en caso contrario. La estructura es muy sencilla, y recibe el indicador del fichero como único parámetro:

int fclose(int indicador)

Recuperar datos de ficheros

Existen en PHP diferentes funciones de lectura o recuperación de datos desde ficheros, pero en todas ellas es importante conocer en que posición del fichero se encuentra. Lo primero es identificar el inicio y el fin del archivo. Cuando se efectúa una instrucción de apertura de archivo el puntero interno del archivo se coloca en al primera posición, y para determinar el fin utilizamos la función feof(), al recibir como parámetro el indicador de archivo determina con TRUE si se está apuntando al final del archivo.
feof($ind)

Las funciones utilizadas para la recuperación de los datos en los ficheros son:

fread(). string fread (int indicador, int num_bytes) Lee bytes especificados.
fgetc(). string fgetc(int indicador) Lee caracter.
fgtes(). string fgets(int indicador, int num_bytes Lee Línea restringido por bytes.

```
$archivo = "prueba.dat"
$ind = fopen($archivo,"r");
$linea = 0;
while (!feof($ind))
{
$linea++;
$contenido = fgets($ind,4096); echo "$contenido <BR>";
}
fclose ($ind);
```

fgestss(). fgetss(int indicador, int num_bytes [, string ver_tags]); Lee línea de archivo HTML fscanf(). Valor fscanf (int indicador, string formato [, string var1...]); Obtiene datos bajo un formato.

```
$ind = fopen("Datos.txt","r"); while (!feof($ind))
{
$datos = fscanf($ind, "%s %d \n", $nombre, $sueldo); echo "$nombre <BR>"; echo "$sueldo <BR>";
}
fclose($fichero);
```

file(). array file(string nomarchivo [, int ruta]); Lee todo el archivo en una sola acción.

ESCRITURA EN FICHEROS

La escritura de archivos mantiene la misma estructura de procesamiento de la lectura: primero se abre el archivo, luego se escriben los datos y por último se cierra el archivo. La función utilizada de apertura utiliza la misma función fopen(), pero utilizando los parámetros de apertura w o a.

$ind = fopen("Datos.txt","w"); $ind = fopen("Datos.txt","a"); Las funciones utilizadas en PHP para escritura en ficheros es la siguiente:

fputs(). int fputs(int fichero, string cadena [, int longitud]); Escribe cadenas en el fichero.
fwrite(). int fwrite(int fichero, string cadena [, int longitud]); Escribe cadenas en el fichero.

MANIPULACIÓN DE FICHEROS

Acceso directo a ficheros

Cuando se requiere que el acceso sobre los archivos no se haga de forma secuencial, PHP provee de algunas funciones para realizar un acceso directo. Estas son:

fseek() int fseek(int indicador, int posicion, [int base]); Ubica el puntero de lectura en cualquier posición. La base del fichero puede ser SEEK_SET para contar desde el principio del fichero, SEEK_CUR para contar desde la posición actual y SEEK_END para que la base sea el final del fichero (Uc3m, 2011).

rewind(). int rewind(int indicador); Se ubica en la primera posición del archivo.
ftell(). int ftell (int indicador); Recupera la posición del puntero.

Otras funciones

fpassthru(). int fpassthru (int indicador) Muestra el contenido referenciado por el manejador de archivo.
set_file_buffer. int set_file_buffer(int indicador, int Cap_buffer) Define tamaño del buffer.
readfile(). int readfile(string Nomarchivo [,int usar_include]);

MANIPULACIÓN DE DIRECTORIOS

Creación, eliminación y cambio de directorios.

Existe una función para cada operación de directorio a saber creación, eliminación y cambio.
Estas son:

- chdir(). boolean chdir (string ruta_directorio) Cambio de directorios
- rmdir(). boolean rmdir (string ruta_directorio) Creación de directorios
- mkdir(). boolean mkdir (string ruta_directorio, int permisos)

Eliminación de directorios

Se debe tener en cuenta que para ambientes Windows se pueden utilizar indistintamente los caracteres slash "/" y backslash "\", pero en entornos UNIX/LINUX sólo slash "/". El carácter punto "." indica directorio actual y la cadena "..", indica el directorio padre del directorio actual (Álvarez R. , 2002).

Procesamiento de archivos en un directorio

Para tener acceso a los archivos o ficheros almacenados en un directorio es necesaria la creación de un manejador de directorio, que no es más que un puntero a una lista de los archivos almacenados en el directorio. Para tal efecto PHP utiliza la función opendir() (blog.openalfa.com, 2016)

int opendir(string ruta_directorio)

Cuando ya se ha creado el directorio se pueden utilizar las demás funciones:

- readdir(). string readdir (int manejador) Devuelve el nombre del siguiente elemento del directorio.
- rewinddir(). void rewindir (int manejador) Se sitúa en el primer elemento del directorio.
- closedir(). void closedir (int manejador) Libera el manejador de directorio.

Copiar, borrar y renombrar ficheros

Estas tareas, sólo pueden ser ejecutadas si se poseen los permisos adecuados para realizarlas, y corresponden a las siguientes funciones (InformaticaPC, s.f.):

- copy(). boolean copy (string fichero_origen, string fichero_destino) Copia ficheros.
- unlink(). boolean unlink (string nombre_fichero) Borra ficheros.
- rename(). boolean rename (string nombre_viejo, string nombre_nuevo) Renombra ficheros.

MANIPULACIÓN DE FICHEROS Y DIRECTORIOS

El lenguaje PHP proporciona a los usuarios, diferentes funciones que le permiten trabajar con los atributos de los ficheros y directorios, que son propiedades que los identifican ante los demás, y que definen su funcionalidad.

- file_exists(). boolean file_exists (string elemento) Determina existencia del elemento.
- filesize(). int filesize (string nom_fichero) Determina el tamaño de un fichero.
- fileatime(). int fileatime (string nom_fichero) Devuelve el tiempo del último acceso.
- filemtime(). int filemtime (string nom_fichero) Devuelve el tiempo de la última modificación.
- filectime(). int filectime (string nom_fichero) Devuelve el tiempo del último cambio.
- filetype(). string filetype (string elemento) Devuelve el tipo de elemento.

Ver tabla.

TIPO	DESCRIPCIÓN
block	Dispositivo de bloques
char	Dispositivo de caracteres
dir	Directorio
fifo	Directorio FIFO
file	Fichero
link	Enlace simbólico
unknown	Indeterminado

PHP tiene implementada la función chmod() para permitir la protección ficheros y directorios: boolean chmod (string elemento, int permisos)

Esta función recibe en sus parámetros el nombre del elemento y un número entero en representación octal que determina los permisos asignados al elemento. Esta tabla cambia en las plataformas Windows y UNIX/LINUX.

fileperms().	int fileperms (string elemento)	Determina los atributos del elemento.
is_readable(). es	is_readble (string elemento)	Determina si de lectura.
is_writeable().	is_writeable (string elemento)	Determina si es de escritura.
is_executable().	is_executable (string elemento)	Determina si es ejecutable.
is_dir().	boolean is_dir (string elemento)	Determina si es elemento.
is_file().	boolean is_file (string elemento)	Determina si es Fichero.

La información que se obtiene de las funciones anteriores puede ser obtenidas con la utilización de la función stat(), que es una matriz indexada con 13 valores (González, Leer y escribir archivos de texto con PHP, 2006).

FUNCIONES PARA UNIX/LINUX

fileowner(). int fileowner (string elemento)
filegroup(). int filegroup (string elemento)
fileinode(). int fileinode (string elemento)
chown(). boolean chown (string elemento, string usuario)
chgrp(). boolean chgrp (string elemento, string grupo)
umask(). int umask (int mascara)
is_link(). boolean is_link (string elemento) linkinfo().
link(). boolean link (string elemento, string enlace)
symlink(). boolean symilink (string elemento, string enlace_simb)
readlink(). readlink (string enlace_simb)

GESTORES DE BASES DE DATOS

Existen varios tipos de Sistemas gestores de bases de datos SGBD, según el modelo de datos que utilizan. Son estos: bases jerárquicas, en red, relacional, y bases de datos orientadas a objetos. El modelo más extendido es el relacional, creado primero como formulación teórica y después implementado en programas. Entre sus ventajas figuran la sencillez de concepción y manejo y su flexibilidad, que permite su aplicación en muchas situaciones reales de gestión de información.

La entidad básica de organización de datos es la tabla; una tabla es una estructura bidimensional para contener datos, dispuestos en filas y columnas: cada fila (tupla) contiene la descripción de las entidades definidas en un problema y cada columna almacena los valores de los atributos (propiedades, variables) de todas las entidades; la intersección de filas y columnas, las celdas, reciben el nombre de campos. Las tablas de una base de datos se enlazan mediante claves (primaria y externa) que son atributos compartidos (repetidos) por una dos o más tablas. Además, las bases de datos relacionales se pueden gestionar mediante un lenguaje independiente de los programas informáticos concretos: “lenguaje estructurado para consulta de bases de datos” (Structured Query Language, SQL). No obstante, los SGBR no se adaptan bien a ciertas necesidades de gestión de datos, lo que ha impulsado el nacimiento y difusión de las bases de datos orientadas a objetos.

Nombre	Cargo	Edad	Antigüedad	Sueldo
Gonzalo	Gerente	45	15	5.360.00
Jaime	Director	36	12	3.500.00
Sandra	Secretaria	34	17	850.500
Felipe	Mensajero	21	3	520.000

Los sistemas gestores de bases de datos, fueron diseñados principalmente para corregir los errores presentados en el mantenimiento formal de un Base de datos. Los SGBD protegen los datos, en lo posible, frente a manejos indebidos, y proporcionan al usuario herramientas de gestión, que faciliten las labores de mantenimiento de una base de datos (PowerData, 2015).

INSTRUCCIONES BÁSICAS SQL

El lenguaje SQL (Structured Query Language, SQL) o, “lenguaje estructurado para consulta de bases de datos” está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos (w3schools, 2016).

Para la creación de la base de datos y su activación contamos con los comandos CREATE DATABASE y USE.

Existen dos tipos de comandos SQL:

Los DLL que permiten crear y definir nuevas bases de datos, campos e índices.
Los DML que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.
Dentro de los comandos DDL tenemos:

CREATE. Utilizado para crear nuevas tablas, campos e índices.

```
CREATE TABLE
NombreTabla (Atributo1 Tipo(Longitud) [Not Null], (Atributo2 Tipo(Longitud) [Not Null],
AtributoN Tipo(Longitud) [Not Null], [PRIMARY
KEY (ListaAtributos) , ]
[FOREIGN KEY (Atributo) REFERENCES
NombreTabla ON DELETE CASCADE | RESTRICT | SET NULL ON UPDATE
CASCADE | RESTRICT | SET NULL] )
CREATE [UNIQUE] INDEX
NombreIndice ON NombreTabla
(ListaAtributos);
```

DROP. Empleado para eliminar tablas e índices.

```
DROP TABLE NombreTabla;
```

ALTER. Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

```
ALTER TABLE
NombreTabla [ADD NombreAtributo Definición]
[CHANGE AtributoAntiguo AtributoNuevo Definición] [DROP NombreAtributo];
```

En los comandos DML se encuentran:

SELECT. Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado.

```
SELECT * |
ListaAtributos FROM
ListaTablas [WHERE
ListaCondiciones] [GROUP BY
ListaAtributos [HAVING
ListaCondiciones]] [ORDER BY
ListaAtributos];
```

INSERT. Utilizado para cargar lotes de datos en la base de datos en una única operación.

```
INSERT INTO NombreTabla[(Atri1, Atri2, ..., Atrin)] VALUES (ValAtri1, ValAtri2, ...,
ValAtrin);
```

UPDATE. Utilizado para modificar los valores de los campos y registros especificados.

```
UPDATE
NombreTabla SET Atributo1
= Valor1, SET Atributo2 =
Valor2, SET
Atributo3 = Valor3,
...
SET Atributon
= Valorn WHERE
Condición];
```

DELETE. Utilizado para eliminar registros de una tabla de una base de datos.

```
DELETE FROM NombreTabla [WHERE Condición];
```

BASES DE DATOS EN PHP

Antes de poder trabajar Bases de datos, con lenguaje PHP es necesario instalar el Gestor de Bases de datos y configurarlo, gestión que depende del SGBD que se haya escogido. El más utilizado en la actualidad es MySQL, pero en general las tareas estándar son similares de operación en la mayoría de gestores.
Dos formas generales de conectarse con las bases de datos a través de ODBC (Open DataBase Connectivity) y por medio de funciones nativas de cada gestor.

Conexión con ODBC

Cuando se trabaja en entorno Windows, la tarea de conexión es muy sencilla. Basta acceder al Panel de Control y seleccionar la herramienta **Fuentes de Datos ODBC**, allí se configura la fuente de datos o DSN determinando el origen con el que se desea establecer la conexión, como dbase, access, foxpro, etc. Luego se procede a dar de alta un DSN de usuario o de sistema.

Ya al interactuar en PHP se deben tener en cuenta las siguientes funciones:

odbc_connect(). Función para realizar la conexión a la base.

```
$conexion = odbc_connect("basededatos", "usuario", "contraseña"); if (!$conexion)
die ("Error en la conexión");
```

odbc_do(). Función para ejecutar el comando SQL.

```
$resultado = odbc_do($conexion, $consultaSQL); if (!$resultado) die ("Error en el comando");
```

odbc_close(). Cierra la conexión.

```
odbc_close($conexion);
```

odbc_fetch_row(). Hace accesible las filas devueltas por una consulta.
odbc_num_fields(). Determina el número de atributos devueltos
odbc_result(). Devuelve el contenido del atributo de la fila activa del cursor.
odbc_free_result().

Libera recursos asociados al cursor. **odbc_num_rows().**Devuelve el número de filas de la consulta.

Conexión con MySQL.

Una vez configurado el gestor de acuerdo a las necesidades de la versión, se puede entonces acceder a Bases de datos a través de PHP y MySQL. En el lenguaje PHP se tiene un gran número de funciones MySQL que empiezan por el prefijo "mysql" (W3schools, 2016). Algunas funciones son:

mysql_connect (host, usuario, contraseña). Establece conexión con el servidor.

```
$conex = mysql_connect("localhost", "usuario", "clave") or die("No hay conexión");
```

mysql_select_db (base, conexión).

Selecciona la base de datos.

```
mysql_select_db("BaseDatos", $conex);
```

mysql_query (consulta, conexión). Ejecuta la consulta SQL indicada.

```
$consulta = "SELECT * FROM Alumnos";
$res = mysql_query($consulta,$conex);
```

mysql_num_fields (cursor). Devuelve el número de atributos en el cursor.

```
$Campos = mysql_num_fields($res);
```

mysql_fetch_rows (cursor). Avanza a la siguiente fila o tupla.

```
$Filas = mysql_num_rows($res);
```

mysql_free_result (cursor). Libera los recursos asociados al cursor.

```
mysql_free_result($res);
```

mysql_close (conexión). Cierra la conexión establecida.

```
mysql_close($conex);
```

Dentro de las funciones de PHP para interacción con MySQL, son importantes las de manejo de error y las conexiones persistentes:

mysql_errno(). Devuelve el número del error producido.

mysql_error(). Devuelve la descripción del error.

mysql_pconnect(). Realiza una conexión persistente. (La función mysql_close() no cierra este tipo de conexiones, así que es necesario liberar los recursos antes de finalizar procesos)

PHP permite al usuario acceder a estructuras de las bases de datos de forma fácil, incluso sin conocer la composición de las tablas. Estas s o n :

mysql_list_dbs (conexión). Devuelve los nombres de las bases de datos.
mysql_list_tables (conexión). Devuelve los nombres de las tablas.
mysql_field_name(cursor, columna). Devuelve el nombre del campo del cursor.
mysql_field_type (cursor, columna). Devuelve el tipo del campo del cursor.

mysql_tablename (cursor, fila). Devuelve el nombre de la tabla o base del cursor y fila.

mysql_field_flags (cursor, columna). Devuelve características del atributo del cursor.

mysql_field_len (cursor, columna). Devuelve la longitud del campo del cursor.

FORMATOS GRÁFICOS

En Internet se utilizan principalmente dos tipos de archivos gráficos GIF y JPG, pensados especialmente para optimizar el tamaño que ocupan en disco, ya que los archivos pequeños se transmiten más rápidamente por la Red. El formato de archivo GIF se usa para las imágenes que tengan dibujos, mientras que el formato JPG se usa para las fotografías. Los dos comprimen las imágenes para guardarlas. La forma de comprimir la imagen que utiliza cada formato es lo que los hace ideales para unos u otros propósitos.

Adicionalmente, se puede usar un tercer formato gráfico en las páginas Web, el PNG. Este formato no tiene tanta aceptación como el GIF o JPG por varias razones, entre las que destacan el desconocimiento del formato por parte de los desarrolladores, que las herramientas habituales para tratar gráficos generalmente no lo soportan y que los navegadores antiguos también tienen problemas para visualizarlas. Sin embargo, el formato se comporta muy bien en cuanto a compresión y calidad del gráfico conseguido, por lo que resultaría útil si se llega a extender su uso (desarrolloweb.com, 2016).

DISEÑO GRÁFICO

El diseño gráfico de una página Web es tan solo una parte del diseño de la misma, ya que, además, hay que considerar un conjunto más o menos extenso de condicionantes que van a limitar la libre creatividad del diseñador. En primer lugar, las páginas Web se deben descargar de un servidor Web remoto por medio de Internet, por lo que el ancho de banda de las conexiones de los usuarios va a ser un factor clave en la velocidad de visualización. La mayoría de los usuarios se conectan todavía a Internet con un módem, con velocidades teóricas de 56 Kbps, pero rara vez alcanzan estos niveles de velocidad.

Los elementos gráficos, ya tengan formato de mapa de bits o vectorial, suelen traducirse en ficheros de bastante peso, dependiendo del tamaño de la imagen y del formato en que se guarde. Esto origina que las páginas que contienen en su diseño muchas imágenes, o pocas pero de gran tamaño, tarden mucho tiempo en ser descargadas desde el servidor Web y presentadas en la ventana del ordenador del usuario, que no suele ser muy paciente. Otro aspecto a tener en cuenta es que las páginas Web son visualizadas en unas aplicaciones específicas, los navegadores Web, que imponen grandes limitaciones al diseño de las mismas. La ventana de un navegador es eminentemente rectangular, con unas medidas concretas (dadas por la resolución empleada por el usuario en su monitor) y con unas capacidades de interpretación de colores que varían mucho según el ordenador usado, el sistema operativo, el monitor y la tarjeta gráfica. Estos factores imponen fuertes limitaciones al diseñador Web, que debe buscar siempre que sus páginas puedan ser visualizadas correctamente por el mayor número de usuarios.

PHP proporciona al usuario la posibilidad de realizar montajes dinámicos, a partir de funciones específicas para la construcción de gráficos. La utilización de estas funciones presenta algunas ventajas al usuario, por ejemplo en algunos casos no es necesario almacenar las imágenes, sino que estas se crean en el mismo momento de la carga de la página. Las funciones del lenguaje PHP para tratamiento de gráficos se pueden clasificar así:

Funciones de tratamiento de color. Funciones de tratamiento de píxeles. Funciones gráficas de tratamiento de texto.
Funciones de acceso a propiedades de los gráficos. Funciones para trazado de gráficas vectoriales.
Las funciones gráficas, disponibles en PHP están relacionadas con la librería GD, para activarlas es necesario incluirlas en el directorio de trabajo (Configuración de PHP) e incluidas en el fichero de configuración php.ini

extension=php_gd.dll

Los colores de las imágenes en PHP, son tratados bajo las condiciones del formato RGB. Las imágenes en PHP pueden ser enviadas al navegador o pueden ser almacenadas en ficheros.
En ambos casos se utiliza las funciones imagegif(), imagepng(), imagejpeg(),...

imagejpeg($imagen); //Salida en le navegador
imagejpeg($imagen, "grafo.jpg"); //Salida en fichero

Los tipos de imágenes que pueden ser soportados por PHP, se pueden obtener a través de la función ImageTypes() (desarrolloweb.com, 2016).

CREACIÓN DE IMÁGENES

Se ha definido una secuencia ordenada de la forma como se deben trabajar los gráficos en PHP que se describe a continuación:

Creación de la imagen. Definición de colores. Tratar la imagen. Elaboración de la página para la imagen. Liberación de recursos consumidos.

Para crear imágenes se tienen dos posibilidades. La primera cuando se pretende crear la imagen desde cero, donde a través de la función imagecreate(), se le especifica el ancho y la altura en píxeles de la imagen (w3schools, 2016).
int imagecreate (int ancho, int altura);

La segunda posibilidad es cuando se desean modificar imágenes ya existentes. Aquí se debe tener en cuenta el formato de imagen origen y se utilizan las siguientes funciones:
int imagecreatefromjpeg (string nombre_archivo);
int imagecreatefromwbmp (string nombre_archivo);
int imagecreatefrompng (string nombre_archivo);
int imagecreatefromgif (string nombre_archivo);

Después de creada la imagen se envía al navegador para que sea mostrada. Esta operación se realiza con las siguientes funciones, que dependen del formato de la imagen.
int imagejpg (int identificador_imagen, [string nombre_archivo [int calidad]]);
int imagewbmp (int identificador_imagen, [string nombre_archivo [int calidad]
]);
int imagepng (int identificador_imagen, [string nombre_archivo [int calidad
]]);
int imagegif (int identificador_imagen, [string nombre_archivo [int calidad]]);

Ejemplo:

header("Content-type: image/jpeg"); //Preparación de la salida de la imagen

```
$imagen = @imagecreatefromjpeg("foto.jpg"); if (!$imagen)

{

echo "Error en creación";

}

else

{

imagejpeg($imagen);

}
```

Para liberar los recursos utilizados por la imagen se utiliza la función imagedestroy(), que relaciona la imagen con el identificador obtenido en la función de creación (Php.net, 2001).

int imagedestroy (int identificador_imagen);

PROPIEDADES DE LOS GRÁFICOS

PHP, dispone de tres funciones que le permiten acceder a las propiedades de los gráficos. Lo que es muy importante porque no siempre las imágenes van a ser utilizadas como están (Php.net, 2001).
Estas funciones son:
getimagesize().
array getimagesize (string, nombre_archivo, [array datos_imagen] Devuelve arreglo de atributos.

Posición	Descripción
0	Ancho de la imagen en
1	Alto de la imagen en píxeles
2	Formato de imagen (número)
3	Cadena con Altura y Anchura

imagesx(). imagen.	int imagesx (int identificador_imagen);	Devuelve el ancho de la
imagesy(). imagen.	int imagesy (int identificador_imagen);	Devuelve la altura de la

TRATAMIENTO DEL COLOR

El lenguaje PHP proporciona al usuario varias funciones para que pueda manipular el color en una imagen, tarea que para los desarrolladores Web resulta interesante con miras a satisfacer las necesidades del cliente. Las funciones que se relacionan con la

definición de colores y paletas son:

- imagecolorallocate(). int imagecolorallocate (int identificador_imagen, int rojo, int verde, int azul) Devuelve un valor de tipo entero que representa la posición de la paleta de colores que define el color. El color se define en formato RGB.
- imagecolordeallocate(). int imagecolordeallocate (int identificador_imagen, int identificador_paleta) Elimina la definición del color (paleta).
- imagecolorset(). bool imagecolorset (int identificador_imagen, int identificador_paleta, int rojo, int verde, int azul) Modifica color (paleta).
- imagecolorstotal(). int imagecolorstotal (int identificador_imagen) Devuelve el número de colores de la paleta de la imagen.
- imagecolorsforindex(). int imagecolorsforindex (int identificador_imagen, int identificador_paleta) Devuelve una matriz asociativa con los valores de los componentes básicos del color.
- imagecolortransparent(). int imagecolortransparent (int identificador_imagen, int [color]) Define el color de transparencia.
- imagecolorat(). int imagecolorat (int identificador_imagen, int x, int y) Devuelve un entero de la ubicación del color del píxel indicado.
- imagecolorclosest(). int imagecolorclosest (int identificador_imagen, int rojo, int verde, int azul) Devuelve la posición del color más cercano al color indicado.
- imagecolorexact(). int imagecolorexact (int identificador_imagen, int rojo, int verde, int azul
-) Devuelve la posición del color especificado dentro de la paleta de colores.
- imagecolorresolve(). imagecolorresolve (int identificador_imagen, int rojo, int verde, int azul) Devuelve la posición del color especificado o del más cercano en la paleta de colores.

Otras funciones se refieren directamente al manejo de los componentes gráficos de las imágenes los píxeles:

- imagesetpixel(). int imagesetpixel (int, identificador_imagen, int x, int y, int color) Dibuja un píxel en la imagen y en el color especificado.
- imagecopy(). int imagecopy (int identificador_imagen_destino, int indentificador_imagen_origen, int Xdestino, int Ydestino, int Xorigen, int Yorigen, int ancho_origen, int alura_origen) copia un pedazo de imagen en otra.
- imagecopyresized(). int imagecopyresized (int identificador_imagen_destino, int indentificador_imagen_origen, int Xdestino, int Ydestino, int Xorigen, int Yorigen, int ancho_destino, int altura_destino, int ancho_origen, int alura_origen) copiar un pedazo de imagen y lo redimensiona en la imagen destino.

```
<?php

$avion = imagecreatefromjpeg("avion");

$ancho = imagesx($avion);

$altura = imagesy($avion);
```

```
$avionsote = imagecreate(2* $ancho, 2 * $altura);
imagecopyresized($avionsote,$avion,0,0,0,0,$ancho-1,$altura-1,$ancho- 1,$altura-1);
$blanco = imagecolorclosest($avionsote, 255, 255, 255);

imagesetpixel($avionsote, 100, 310, $blanco); header("Content-type: image/png");
imagepng($avionsote); imagedestroy($avionsote); imagedestroy($avion);
?>
```

GRÁFICAS VECTORIALES

Las gráficas vectoriales fueron diseñadas para crear figuras geométricas o formas que se construyen a través de fórmulas matemáticas que permiten construirlas en un tamaño especificado. Revisemos la estructura de algunas de ellas:

- imageline(). int imageline (int identificador_imagen, int x1, int y1, int x2, int y2, int color) Dibuja una línea sobre la imagen especificada..
- imagedashedline(). int imagedashedline (int identificador_imagen, int x1, int y1, int x2, int y2, int color) Dibuja una línea punteada sobre la imagen especificada.
- imagearc(). int imagearc (int identificador_imagen, int centrox, int centroy, int radiox, int radioy, int anguloi, int angulof, int color) Dibuja un arco de circunferencia, si los radios son iguales, o de elipse. Si el ángulo inicial es 0 y ángulo final 360 será una circunferencia o una elipse completa.
- imagerectangle(). int imagerectangle (int identificador_imagen, int x1, int y1, int x2, int y2, color) Dibuja un cuadrado o un rectángulo en la imagen especificada.
- imagepolygon(). int imagepolygon (int identificador_imagen, array puntos, int num_puntos, int color) Dibuja un polígono, de acuerdo a los puntos especificados en la imagen.

Para rellenar de color las figuras dibujadas se disponen en PHP de las siguientes funciones:

- imagefill(). int imagefill (int identificador_imagen, int x, int y, int color) Rellena sin incluir el borde una área cerrada.
- imagefilltoborder(). int imagefilltoborder (int identificador_imagen, int x, int y, int color_borde, int color_fondo) Rellena teniendo en cuenta el borde un área cerrada.
- imagefilledrectangle(). int imagefilledrectangle (int identificador_imagen, int x1, int y1, int x2, int y2, int color) Dibuja un rectángulo y lo rellena con el color especificado.
- imagefilledpolygon(). int imagefilledpolygon (int identificador_imagen, array puntos, int num_puntos, int color) Dibuja in polígono y lo rellena con el color especificado (Moreno, 2005).

EXPRESIONES REGULARES

Las expresiones regulares son una serie de caracteres que forman un patrón, normalmente representativo de un grupo de caracteres, de tal forma que podemos compararlo con otro conjunto de caracteres para ver las coincidencias. Un ejemplo de un patrón y una comparación sería:

am Patrón. am coincide panorama coincide ambición coincide campamento coincide mano no coincide

Tenemos un patrón (am) que podemos comparar con otro conjunto de caracteres para ver cuando coinciden o no. Las expresiones regulares son un magnifico instrumento de programación, presente en PHP y en muchos otros lenguajes de programación que nos permiten, por ejemplo, comprobar la corrección de entrada de datos por un usuario en un formulario, o hacer búsquedas y sustituciones a lo largo de un texto dado.
Un patrón puede estar formado por un conjunto de caracteres o por meta caracteres que representan otros caracteres, o permiten una búsqueda contextual. Los meta-caracteres reciben este nombre porque no se representan a ellos mismos, sino que son interpretados de una manera especial. Los mas usados son los signos ^ y $. Cuando usamos el signo ^ queremos decir que el patrón debe aparecer al principio del conjunto de caracteres comparado. Cuando usamos el signo $ estamos indicando que el patrón debe aparecer al final del conjunto de caracteres:

- ^am Patrón am coincide cama no coincide ambidiestro coincide Pam no coincide caramba no coincide
- am$ Patrón am coincide salam coincide ambar no coincide Pam coincide
- ^am$ Patrón am coincide salam no coincide ambar no coincide
- Si el patrón está compuesto por uno de los signos que representan meta caracteres, tenemos que indicarlo con un carácter de escape, la barra invertida
- \. Un patrón definido como \$12 no coincide con un número terminado en 12, y sí con $12.

La lista de meta caracteres es la siguiente:

- . * ? + [ ] ( ) { } ^ $ | \
- . Representa cualquier carácter, menos *nueva línea*.
- * Coincide si el carácter o grupo de caracteres que le precede está presente 0 o mas veces: ab* coincide con "a", "ab", "abbb", etc.

- ? Coincide si el carácter o grupo de caracteres que precede está presente 0 o 1 vez: ab?
- coincide con "a", "ab", no coincide con "abb".
- + Coincide si el carácter o grupo de caracteres que le precede está presente al menos 1 o mas veces. ab+ coincide con "ab", "abbb",etc. No coincide con “a”.
- [abc] Coincide si existe una "a" o una "b" o una "c".
- [a-c] Coincide si existe una letra en el rango ("a", "b" o "c")
- (abc) Coincide si está presente el conjunto abc
- {x,y} Coincide si la letra o grupo que le precede esta presente *entre* x, y

veces: "ab{2}" coincide con "abb": exactamente dos ocurrencias de "b". "ab{2,}" coincide con "abb", "abbbb"

- ... Como mínimo dos ocurrencias de b, máximo indefinido. "ab{3,5}" coincide con "abbb", "abbbb", o "abbbbb": Mínimo 2 ocurrencias, máximo 5.
- ^ Coincide si el patrón esta al comienzo de la frase.
- $ Coincide si el patrón esta al final de la frase.

- x|y Coincide si esta presente x ó y: (Sr|Sra) coincide si precede "Sr" o "Sra" Ampliando:
- En el caso de los paréntesis cuadrados [], que incluidos en un patrón, dan coincidencia si en la cadena a comparar existe cualquiera de los caracteres que encierra. Se aclara que, lo que hay entre corchetes es el rango de los caracteres coincidentes. Basta que exista cualquiera de ellos para que se de la condición:
- [abc] El patrón coincide con la cadena si en esta hay cualquiera de estos tres caracteres: a, b, c.
- c[ao]sa Coincide con casa y con cosa.
- [^abc] El patrón coincide con la cadena si en esta NO hay ninguno de estos tres caracteres: a, b, c. El signo ^ aquí tiene un valor excluyente.
- c[^ao]sa Coincide con cesa, cusa, cisa (etc); no coincide con casa ni cosa.
- [0-9] Coincide con una cadena que contenga cualquier número entre el 0 y el 9.
- [^0-9] Coincide con una cadena que NO contenga ningún número.
- [A-Z] Coincide con cualquier carácter alfabético, en mayúsculas. No incluye números.
- [a-z] Como el anterior, en minúsculas.

Como estos patrones se usan una y otra vez, hay caminos abreviados con la misma significancia.

Abreviatura	Equivalente	Significado
\d	[0-9]	Números de 0 a 9
\w	[0-9A-Za-z]	Cualquier número o cualquier letra
\s	[\t\n\r]	Espacio en blanco: incluye espacio, tabulador, nueva línea o retorno
\D	[^0-9]	El contrario de \d
\W	[^0-9A-Za-z]	Contrario de \w, un carácter que no sea letra ni número
\S	[^ \t\n\r]	Contrario de \s, cualquier carácter que no sea espacio

Los meta caracteres vistos informan si el patrón coincide con la cadena a comparar. Pero ¿y si queremos comparar con una cadena un patrón que puede estar una o más veces, o puede no estar ? Para esto usamos un tipo especial de meta caracteres: los

multiplicadores:

- ? Coincide si el carácter o grupo de caracteres que precede a ? esta presente 0 o 1 vez
- cant?a Coincide con canta y cana d?el Coincide con del y el (ala)?cena Coincide con cena y alacena
- * Coincide si el carácter o grupo de caracteres que le precede esta presente 0 o mas veces.
- cant*a Coincide con canta, cana, canttta
- + Coincide si el carácter o grupo de caracteres que le precede está presente 1 o mas veces.
- cant+a Coincide con canta, canttttta, NO coincide con cana.
- Si se desea es confrontar que un caracter patrón exista un numero de veces determinado en la cadena, se usa los cuantificadores {} (Php.net, 2001).
- a{2,3} Coincide con casaa, casaaa.
- a{2, } Coincide con cualquier palabra que tenga al menos dos "a" o mas: casaa o casaaaaaa, no con casa.
- a{0,3} Coincide con cualquier palabra que tenga 3 o menos letras "a".
- a{5} Exactamente 5 letras "a".

SESIONES

Hasta ahora los programas que se ejecuten y utilicen variables que sólo existen en el archivo ejecutado, se pierden al cargar otra página distinta, a menos que se pasaran por la URL o se inscribieran en las cookies o en un formulario para su posterior explotación. Estos métodos, aunque útiles, no son todo lo prácticos que podrían ser en determinados casos en los que la variable que se quiere conservar ha de ser utilizada en varios scripts diferentes y distantes los unos de los otros.

Se puede pensar que ese problema queda resuelto con las cookies ya que se trata de variables que pueden ser invocadas en cualquier momento. El problema, ya lo hemos dicho, es que las cookies no son aceptadas ni por la totalidad de los usuarios ni por la totalidad de los navegadores lo cual implica que una aplicación que se sirviera de las cookies para pasar variables de un archivo a otro no sería 100% infalible. Es importante a veces pensar en "la inmensa minoría", sobre todo en aplicaciones de comercio electrónico donde se debe captar la mayor cantidad de clientes posibles y los scripts PHP deben estar preparados ante cualquier eventual deficiencia del navegador del cliente.

Resulta necesario entonces poder declarar ciertas variables que puedan ser reutilizadas tantas veces como queramos dentro de una misma sesión. Este problema se corrige a partir de las variables de sesión. Una sesión es considerada como el intervalo de tiempo empleado por un usuario en recorrer las páginas hasta que abandone el sitio Web o deje de actuar sobre él durante un tiempo prolongado o bien, sencillamente, cierre el navegador.

PHP permite almacenar variables llamadas de sesión que, una vez definidas, podrán ser utilizadas durante este lapso de tiempo por cualquiera de los scripts del sitio de trabajo. Estas variables serán específicas del usuario de modo que varias variables sesión del mismo tipo con distintos valores pueden estar coexistiendo para cada una de las sesiones que están teniendo lugar simultáneamente. Estas sesiones tienen además su propio identificador de sesión que será único y específico.
Iniciar una sesión se puede hacer de dos formas distintas:

-Declarar abiertamente la apertura de sesión por medio de la función session_start(). Esta función crea una nueva sesión para un nuevo visitante o bien recupera la que está siendo llevada a cabo.

-Declarar una variable de sesión por medio de la función session_register('variable'). Esta función, además de crear o recuperar la sesión para la página en la que se incluye también sirve para introducir una nueva variable de tipo sesión.

Las sesiones han de ser iniciadas al principio de los script. Antes de abrir cualquier etiqueta o de imprimir cualquier cosa. En caso contrario se recibirá un error (Álvarez, Tutorial de sesiones en PHP, 2001). Un ejemplo clásico: un contador. Este contador deberá aumentar de una unidad cada vez que recargue la página o realicemos el enlace:

```
<?php session_register('contador'); If (isset($contador)==0)

{

$contador=0;

}

++$contador;

echo "<a href=\"contador.php\">Recargado página $contador veces</a>";

?>
```

Otras funciones útiles para la gestión de sesiones son:

Función	Descripción
Session_id()	Nos devuelve el identificador de la sesión
Session_destroy()	Da por abandonada la sesión eliminando variables e identificador.
Session_unregister('variable')	Abandona una variable sesión

BIBLIOGRAFÍA

Álvarez, M. Á. (1 de Enero de 2001). *Formatos gráficos para páginas web*. Obtenido de Desarrolloweb: http://www.desarrolloweb.com/articulos/19.php

Álvarez, M. Á. (1 de Enero de 2001). *Tutorial de sesiones en PHP*. Obtenido de DesarrolloWeb: http://www.desarrolloweb.com/articulos/235.php

Álvarez, M. Á. (21 de Junio de 2004). *Programación orientada a objetos en PHP*. Obtenido de DesarrolloWeb: http://www.desarrolloweb.com/articulos/1540.php

Álvarez, R. (21 de Octubre de 2002). *Gestión de directorios por PHP*. Obtenido de DesarrolloWeb: http://www.desarrolloweb.com/articulos/935.php

allaboutcookies.org. (2015). *las cookies*. Obtenido de allaboutcookies.org: http://www.allaboutcookies.org/es/

Blog Openalfa. (2013). *Cómo procesar los ficheros de un directorio en PHP*. Obtenido de Blog Openalfa: http://blog.openalfa.com/como-procesar-los-ficheros-de-un-directorio-en-php

ccm.net. (Noviembre de 2016). *El protocolo HTTP*. Obtenido de Ccm: http://es.ccm.net/contents/264-el-protocolo-http

cdsphp. (s.f.). *Cliente Servidor*. Obtenido de cdsphp: https://cdsphp.wordpress.com/cliente-servidor/

elladodelgeek.com, & El Youzghi, M. (1 de Mayo de 2015). *Variables, constantes y tipos de datos*. Obtenido de El lado del geek: http://www.elladodelgeek.com/curso-php-variables-tipos-datos/

González, E. (2006). *Bucles en PHP*. Obtenido de Aprenderaprogramar: http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=565:tipos-de-bucles-ciclos-de-repeticion-en-php-while-dowhile-y-for-ejercicios-y-ejemplos-resueltos-cu00822b&catid=70:tutorial-basico-programador-web-php-desde-cero&Itemid=193

González, E. (2006). *Estructuras repetitivas o Bucles en PHP*. Obtenido de Aprenderaprogramar: http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=565:tipos-de-bucles-ciclos-de-repeticion-en-php-while-dowhile-y-for-ejercicios-y-ejemplos-resueltos-cu00822b&catid=70:tutorial-basico-programador-web-php-desde-cero&Itemid=193

González, E. (2006). *Funciones cadenas PHP: str_replace, strtolower, count_chars, strpos, trim, str_repeat, strstr, chr*. Obtenido de Aprender a programar: http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=574:funciones-cadenas-php-strreplace-strtolower-countchars-strpos-trim-strrepeat-strstr-chr-cu00828b&catid=70:tutorial-basico-programador-web-php-desde-cero&Itemid=193

González, E. (2006). *Leer y escribir archivos de texto con PHP*. Obtenido de Aprenderaprogramar: http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=585:leer-y-escribir-archivos-de-texto-con-php-funcion-fopen-modo-fgets-fputs-fclose-y-feof-ejemplo-cu00836b-&catid=70:tutorial-basico-programador-web-php-desde-cero&Itemid=193

Gonzalez, E. (1 de noviembre de 2016). *Funciones cadenas PHP*. Obtenido de Aprenderaprogramar:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=607:generar-html-usando-lenguaje-php-ejemplos-sencillos-que-estudiar-para-ser-programador-web-cu00733b&catid=69:tutorial-basico-programador-web-html-desde-cero&Itemid=192

hipertexto, & Lamarca Lapuente, M. J. (8 de Diciembre de 2013). *Hipertexto, el nuevo concepto de documento en la cultura de la imagen*. Obtenido de Hipertexto: http://www.hipertexto.info/documentos/b_datos.htm

InformaticaPC. (s.f.). *Borrar líneas en un archivo de texto*. Obtenido de InformaticaPC: http://informaticapc.com/tutorial-php/manejo-de-archivos.php

Manualdephp. (s.f.). *Entrada y salida en PHP*. Obtenido de Manualdephp: http://www.manualdephp.com/manualdephp/entrada-y-salida-en-php/

Manualdephp. (s.f.). *Fechas PHP*. Obtenido de Manualdephp: http://www.manualdephp.com/manualdephp/fechas-php/

Moreno, L. (2 de Febrero de 2005). *Gráficos vectoriales*. Obtenido de DesarrolloWeb: http://www.desarrolloweb.com/articulos/1806.php

php.net. (2001). *Arrays*. Obtenido de php.net: http://php.net/manual/es/language.types.array.php

php.net. (2001). *Cadenas de caracteres (Strings)*. Obtenido de php.net: http://php.net/manual/es/language.types.string.php

Php.net. (2001). *Expresiones Regulares*. Obtenido de Php.net: http://php.net/manual/es/book.pcre.php

php.net. (2001). *Funcion fopen*. Obtenido de Php.net: http://php.net/manual/es/function.fopen.php

Php.net. (2001). *Funciones*. Obtenido de Php.net: http://php.net/manual/es/language.functions.php

php.net. (2001). *Instalacion PHP*. Obtenido de php.net: http://php.net/manual/es/faq.installation.php

Php.net. (2001). *Listado de Funciones y Métodos*. Obtenido de Php.net: http://php.net/manual/es/indexes.functions.php

Php.net. (2001). *Operadores*. Obtenido de Php.net: http://php.net/manual/es/language.operators.php

php.net. (2001). *Que es PHP*. Obtenido de http://php.net/manual/es/intro-whatis.php

PowerData. (12 de Agosto de 2015). *Tipos y función de los gestores de bases de datos*. Obtenido de PowerData: http://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/406547/Tipos-y-funci-n-de-los-gestores-de-bases-de-datos

Rolandocaldas. (22 de Junio de 2013). *Uso de Include y require - PHP paso a paso*. Obtenido de Rolandocaldas: https://rolandocaldas.com/php/usando-require-include-php

Sintes Marco, B. (28 de Septiembre de 2016). *Primeras páginas en PHP*. Obtenido de mclibre: http://www.mclibre.org/consultar/php/lecciones/php_primeraspaginas.html

Uc3m. (2011). *Lectura y escritura de ficheros*. Obtenido de Uc3m: http://www.it.uc3m.es/abel/as/MMC/L2/FilesDef_es.html

Uma. (s.f.). *El protocolo HTTP*. Obtenido de Uma: http://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/http.html

Velásquez, R. (16 de Julio de 2013). *PHP desde cero: Variables y Constantes*.

Obtenido de Codehero: http://codehero.co/php-desde-cero-variables-constantes/

w3schools. (2016). *Colors picker*. Obtenido de w3schools: http://www.w3schools.com/sql/

w3schools. (2016). *HTML Color Picker*. Obtenido de w3schools: http://www.w3schools.com/colors/colors_picker.asp

W3schools. (2016). *PHP 5 Tutorial*. Obtenido de W3schools: http://www.w3schools.com/php/default.asp