PHP:

UN PASO AL ÉXITO

Fundamentos de programación web
y desarrollo con bases de datos

Wilson Mamani Rodrigo
Kevin Arturo Quispe Apaza
Julio César Llocllli Champi
Wilson Mamani Rodrigo, editor

PHP
UN PASO AL ÉXITO

Wilson Mamani Rodrigo
Kevin Arturo Quispe Apaza
Julio César Lloclli Champi

HOJA DE CRÉDITOS

Título de la obra:
PHP: UN PASO AL ÉXITO
Fundamentos de programación web y desarrollo con bases de datos

Autores:
Wilson Mamani Rodrigo
Kevin Arturo Quispe Apaza
Julio César Lloclli Champi

Editado por:
Wilson Mamani Rodrigo
Domicilio: Av. Garcilazo del Vega Calle 01 LT. 3, Abancay - Perú
Correo electrónico: rodrigo.peruvian@gmail.com

Correo electrónico:
rodrigo.peruvian@gmail.com

Edición:
Primera edición digital, abril 2026

Publicación disponible en: https://nova.innovationunamba.tech/

Año de publicación:
2026|

Lugar de publicación:
Perú

ISBN: _____
Depósito Legal: _____

Primera edición publicada en el Perú, 2026.

ÍNDICE

INTRODUCCIÓN AL PHP 1
1.1 Ejecución en el lado del cliente 1
1.2 Ejecución en el lado del servidor 1
1.3 Instalación de un servidor local 2
1.4 Proceso de instalación del AppServ 3
1.5 El servidor 10
1.6 Editores para php 11
1.7 Descarga del sublime text 11
1.8 Instalación del sublime text 12
1.9 Estructura del PHP 13
1.10 Mi primer programa en php 14
VARIABLES Y COMENTARIOS 16
2.1 Comentarios en PHP 16
2.2 Las variables 17
2.3 Diferencias en echo y print 22
2.4 Flujo de ejecución 22
2.5 Las funciones. 23
2.6 Ámbito de las variables 26
2.7 Declarando una variable global 27
2.8 Declarando una variable super global 28
2.9 Variables estáticas 29
STRINGS Y OPERADORES DE COMPARACIÓN 32
3.1 Strings 32
3.2 Comparación de cadena de caracteres entre sí. 35
3.3 Operadores de comparación 37
3.4 Archivo VALIDACION.PHP 44
3.5 Constantes 46
3.5.1 Declaración de constantes en PHP 46
3.5.2 Aspectos a tener en cuenta al momento de usar las constantes. 46
3.5.3 Constantes predefinidas. 47
FUNCIONES MATEMÁTICAS Y CASTINGS 49
4.1 Operadores matemáticos 49

4.2 Paso de parámetros a una función 52
4.3 Operadores de incremento y decremento 57
4.4 Funciones matemáticas y castings 57
4.5 Uso de funciones matemáticas 57
4.6 Casting 59
CONDICIONALES: USO DEL OPERADOR IF 61
5.1 Operadores lógicos 61
5.2 Prioridad de operadores 61
5.3 Precedencia de operadores 62
5.4 Ejemplo del uso de condicionales 64
5.5 Operador ternario 68
5.5.1 Sintaxis del operador ternario. 68
ESTRUCTURAS CONDICIONALES 70
6.1 SWITCH - CASE 70
BUCLE WHILE Y DO WHILE 73
7.1 Definición de bucles 73
7.2 Tipos de bucles 73
7.3 Flujo de ejecución del while 73
7.4 Sintaxis en código while. 74
7.5 Bucle infinito 74
7.6 Bucle do – while 75
FOR 78
8.1 Bucle for 78
8.2 Sintaxis del bucle for 78
8.3 Sentencias de interrupción de bucle 78
8.4 Instrucción break 80
8.5 Instrucción continue 80
FUNCIONES EN PHP 83
9.1 Funciones 83
9.2 Tipos de funciones 83
9.2.1 Funciones predefinidas. 83
9.2.2 Funciones propias 86
9.3 Parámetro de funciones 86
9.4 Parámetros por valor y por referencia 89
BASE DE DATOS CON PHP 94

10.1 Definición de MySQL 94
10.2 Base de datos relacional. 94
10.2.1 Multihilo 94
10.2.2 Multiusuario 94
10.3 Servidores MySQL 95
10.3.1 Servidor local. 95
10.3.2 Acceso al servidor MySql 95
10.4 Creación de una base de datos desde PhpMyAdmin 99
10.5 Eliminar una base de datos desde PhpMyAdmin 100
10.6 Creación de una base de datos desde consola 100
10.7 Ver las bases de datos que tenemos en el servidor 101
10.8 Eliminar una base de datos por consola 102
CREACIÓN Y MANIPULACIÓN DE TABLAS EN MYSQL 103
11.1 Tabla 103
11.2 Creación de tablas. 103
11.3 Insertar registros 105
11.4 Tipos de datos en MySql 106
11.4.1 Tipos de datos numéricos 107
11.4.2 Tipos de datos cadenas 108
11.4.3 Tipo de datos fechas y horas 109
CONEXIÓN DE LA BASE DE DATOS CON PHP 111
12.1 Consultas SQL 111
12.2 Conexión de la base de datos con MySql 111
12.2.1 Dirección de la base de datos 112
12.3 Manejo de errores en la conexión con la base de datos 116
12.4 Recorrido del array de resultados SQL 118
IMPORTACIÓN DE TABLAS 122
13.1 Optimizando la conexión a la base de datos 122
13.2 Importación de tablas a la base de datos en mysql 124
13.3 Modificando el acceso a registros de la base de datos 127
13.4 Cerrar la base de datos 129

PRESENTACIÓN

El curso está dividido en dos partes, en la primera parte se encuentra las generalidades de PHP, la utilidad que tiene, en que consiste el lenguaje en el lado del servidor, componentes de una aplicación PHP, los programas o softwares que vamos a necesitar instalado en nuestro computador, tipos de datos, variables, constantes, bucles, condicionales, arrays, etc. Es decir, todo lo que se necesita.

El presente curso es para iniciar de cero, esta direccionado para una persona que no tiene conocimientos de PHP.

La segunda parte veremos el PHP con las Bases de Datos, como manejar una base de datos con MySql, veremos el lenguaje estructurado de consultas SQL, vamos a manejar la consola de PhpMyAdmin, crear copias de seguridad, borrar registros, insertar registros, copias de bases de datos, conectar a una base de datos, trabajaremos con formularios, trabajos frecuentes que haremos con PHP, manejaremos Cookies y Sesiones, seguridad, manejo de errores y crearemos una aplicación práctica.

Al final de cada tema haremos un ejercicio.

TEMA 01

INTRODUCCIÓN AL PHP

PHP es un lenguaje que se ejecuta en un servidor

1.1 Ejecución en el lado del cliente

Cuando abrimos el navegador, introducimos una dirección WEB en la barra de direcciones (URL), y presionamos ENTER, en ese instante se produce una PETICIÓN al servidor para que nos devuelva la página HTML que queremos visitar. El servidor nos da una RESPUESTA, esta respuesta es la página WEB, y siempre que esa página tenga código HTML, JavaScript o cualquier código que se ejecuta del lado del cliente, es mi navegador quien procesa el código de programación que hay dentro de esa página WEB.

A continuación, mostramos programas que se ejecutan del lado del cliente:

- JavaScript
- VBScript
- Applets Java
- HTML
- CSS

1.2 Ejecución en el lado del servidor

El proceso que se ejecuta del lado del servidor es diferente: abrimos el navegador, introducimos la URL, presionamos ENTER y en ese momento se realiza la **PETICIÓN** (hasta aquí todo igual), pero, cuando esa petición llega al servidor, el servidor procesa el código PHP que contiene la página WEB que queremos visitar, luego el servidor **RESPONDE** enviándonos un archivo HTML en el que podemos ver la información que pedimos.

A continuación, mostramos lenguajes que se ejecutan del lado del servidor tenemos:

- PHP (Preprocesador de Hipertexto)
- JSP (Java Server Pages)
- Perl

- ASP (Active server Pages)

Estos lenguajes nos son interpretados en el ordenador, sino que se ejecutan y se interpretan en el servidor, y una vez que el servidor los ha interpretado, nos proporciona una respuesta en formato HTML en función de lo que haya interpretado en esos lenguajes.

Al querer programar en PHP estamos obligados a tener disponible un servidor WEB, para poder ver el resultado de nuestras páginas WEB, y para hacer esto tenemos dos alternativas:

- **CONTRATAR UN HOSTING** que trabaje con PHP y para realizar las pruebas de cómo va nuestro código, subimos las páginas al hosting, después abrimos el navegador y visitamos esa página para ver si va bien nuestro programa, es muy engorroso porque cada que hiciéramos un cambio debemos de realizar todos los pasos mencionados anteriormente. Por lo tanto, cada vez que modifiquemos, nuestra página será visible por todos los cibernautas, con todo y fallas.

- La otra alternativa es **INSTALAR UN SERVIDOR LOCAL**, que consiste en convertir nuestro computador en un servidor. Nos beneficia mucho pues nuestra aplicación seria subida cuando esté libre de errores.

1.3 Instalación de un servidor local

Necesitamos instalar 3 programas:

- Servidor WEB : Apache
- Lenguaje de Programación : PHP
- Gestor de Base de datos : MySql

Utilizaremos estas aplicaciones primeramente porque son gratuitos, hay una gran comunidad que brindan soporte cuando necesitamos hacer una consulta, tanto para el servidor WEB apache como para el PHP y para el gestor de base de datos MySql. En segundo lugar, porque son tecnologías que se entienden bastante bien entre ellas, PHP puede trabajar con otros gestores de base de datos, pero es, esencialmente sencillo cuando trabaja con MySql.

Se puede instalar uno por uno, pero, hay una alternativa de instalarlos en conjunto, en un paquete que traen en su interior estos tres softwares y además traen herramientas

adicionales bastante útiles, por lo cual solo tendríamos que hacer una única instalación, el paquete ya instalara todo lo necesario.

Si optamos por la instalación separada, lo primero que debemos de instalar es el servidor WEB, luego el PHP y el MySql. Porque, si primero instalas el PHP o el Gestor de Base de Datos, tendrás problemas.

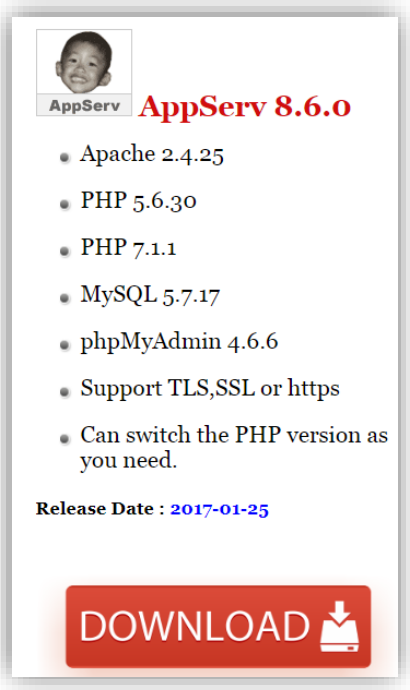
Optaremos la instalación del paquete, y podemos encontrar los siguientes:

Concretamente, nosotros trabajaremos con AppServ. Todos los programas funcionan de manera similar, lo único que varía es la interfaz, pero el objetivo de estos paquetes es el mismo. Estos paquetes incluyen el PhpMyAdmin que es una consola de administración para el manejo de las Base de Datos

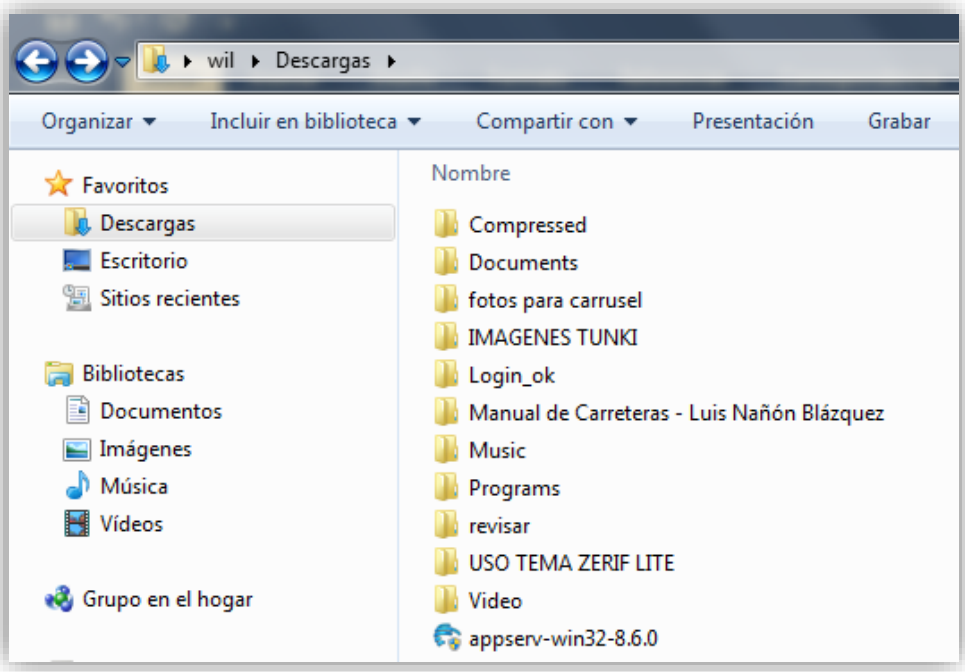
1.4 Proceso de instalación del AppServ

Descargaremos el AppServ de la página:

En esta página daremos clic en **DOWNLOAD**: (Probablemente aparecerá una nueva página, pero es lo mismo, así que daremos nuevamente clic en **DOWNLOAD**)



El instalador iniciará su proceso de descarga, y lo descargará en la ruta de descargas que su sistema operativo tenga configurado. En la mayoría de los casos las descargas se hacen en la carpeta descargas de la Unidad C.



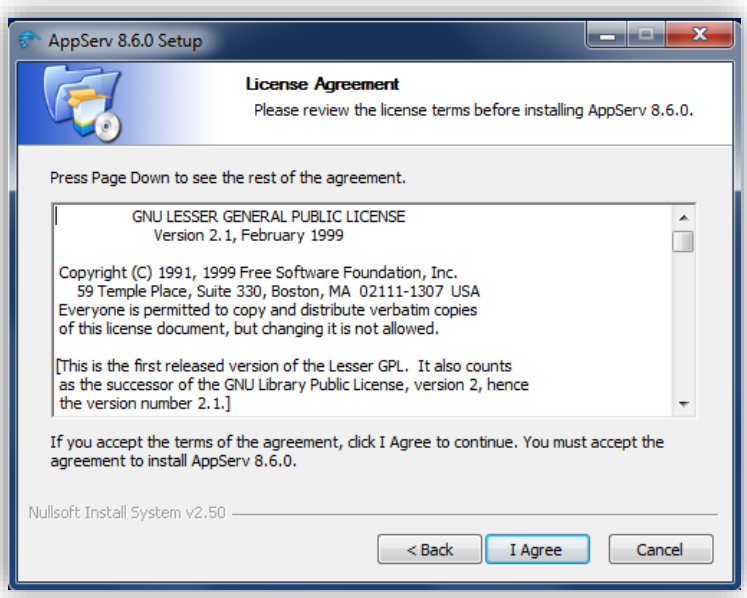
Haremos doble clic en el archivo **appserv-win32-8.6.0** y aparecerá una pregunta rigurosa de Windows:

Le daremos clic en **SÍ**.

Luego se procederá con la carga del instalador, en donde nos aparece una bienvenida al programa de instalación del AppServ 8.6.0. en la cual recomienda cerrar otras aplicaciones, debido a que el sistema operativo debe de reiniciarse después de la Instalación.

Después de haber leído el mensaje y haber cerrado las demás aplicaciones abiertas, procedemos haciendo clic en **NEXT.**

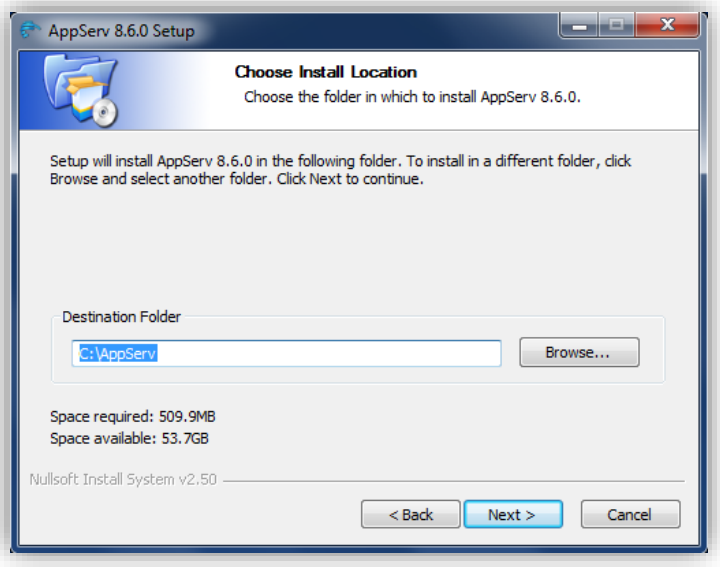
Luego de ejecutar en NEXT, aparece la licencia del producto, dar una lectura y dar clic en **I AGREE,** el continuar con la instalación es como si aceptáramos los términos de contrato.



A continuación, podemos indicar la ubicación en donde se crearán los archivos necesarios para el uso de nuestro servidor local, puede aceptar por defecto la unidad **C:\AppServ** o puede cambiar la ruta de almacenamiento si así lo desea.

También podemos observar el espacio requerido para la instalación, que en este cado necesita 509.9 Megabytes de espacio libre en su disco o unidad donde se instalara el **AppServ.**

Para nuestro caso elegiremos la instalación en la unidad **C:\AppServ** que viene configurado por defecto:

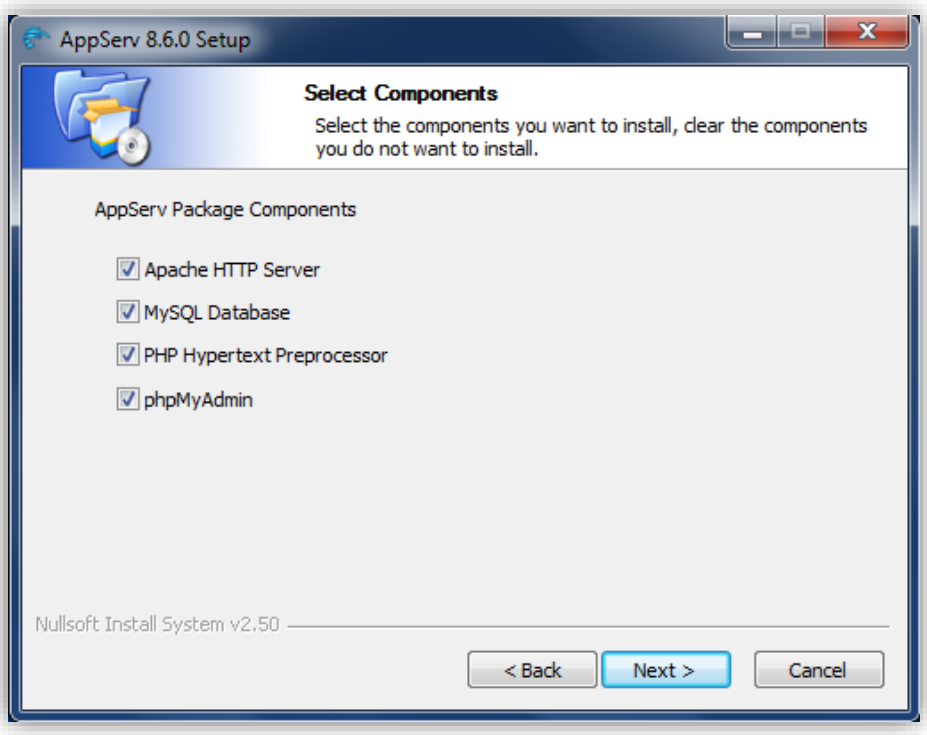


Daremos clic en **NEXT,** y en este formulario nos aparecerá los componentes que necesitamos instalar. Por defecto aparecerán seleccionadas las siguientes opciones:

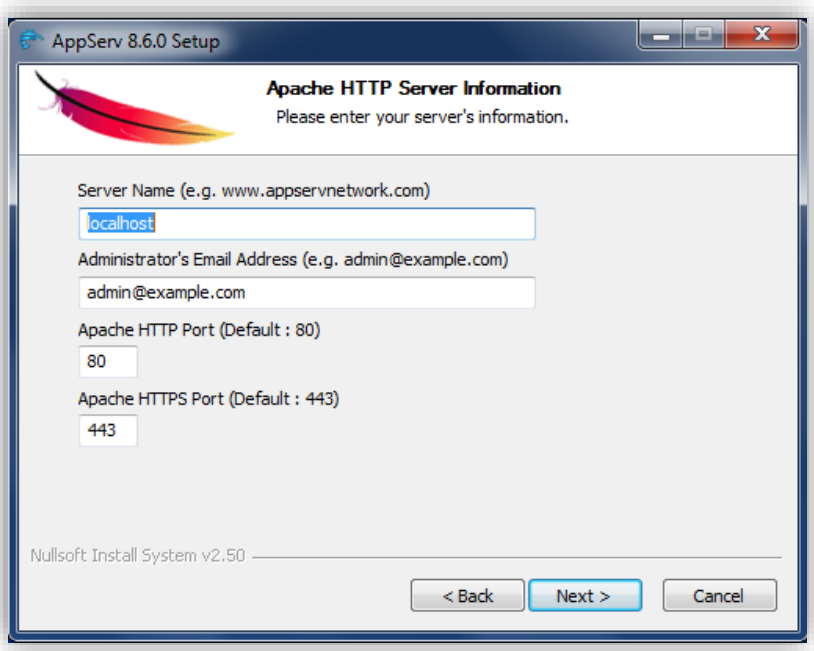
- Servidor WEB Apache
- Servidor de Base de Datos MySql
- Lenguaje de programación PHP
- PHPMyAdmin

Todos estos programas o aplicaciones las necesitamos, por lo que deberemos de verificar que todas estas casillas estén activadas, en caso que no estén activadas, debemos de activarlas.

Luego procedemos a hacer clic en **NEXT.**

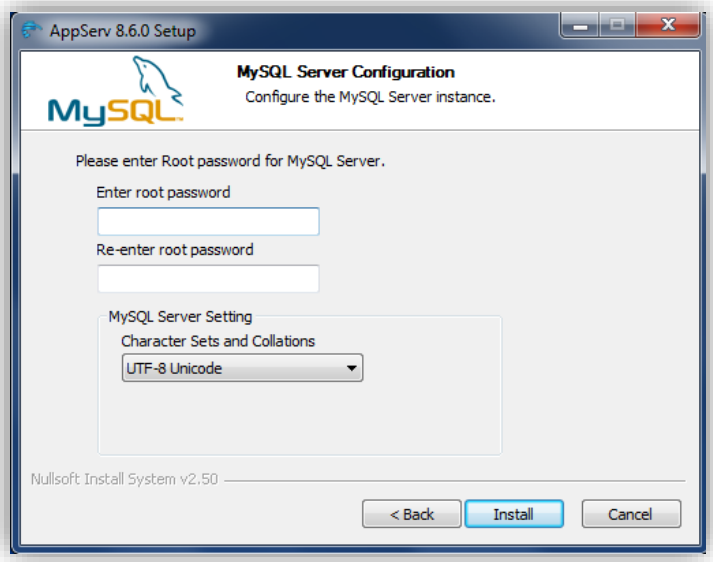


Ahora presenciamos el nombre del servidor: **localhost,** debemos de dejar todo como esta por defecto y hacemos clic en **NEXT.**



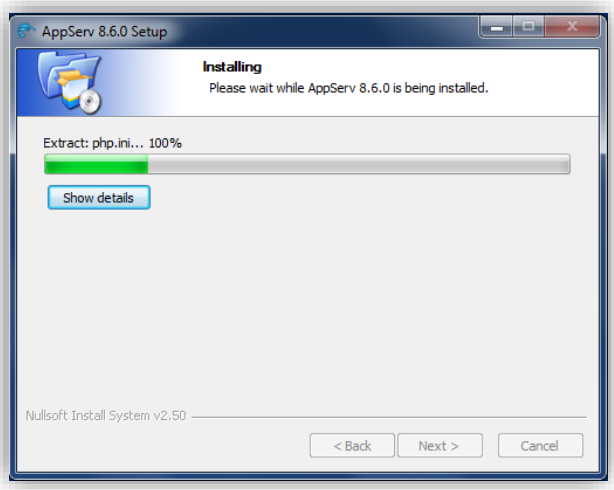
En la siguiente ventana ingresaremos el password[1] para acceder a nuestro servidor de Base de datos. En donde nuestro usuario será **ROOT** y nuestro password será el que ingresamos en este formulario.

Debemos de ingresar el password 02 veces, dejar tal y como esté las demás opciones, luego dar clic en **INSTALL**

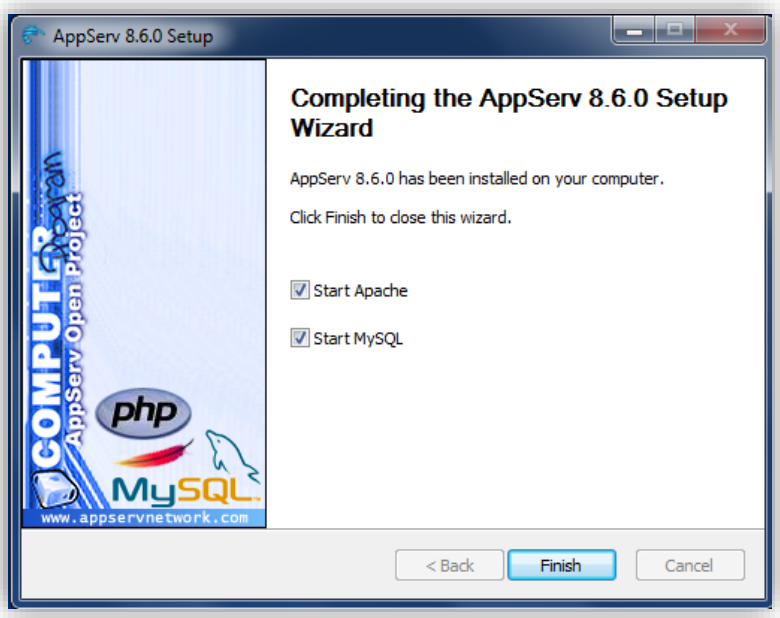


[1] Guardar el password o almacenarlo en un lugar que pueda recordar, pues es muy importante para acceder a todas sus bases de datos creadas en el servidor local.

El programa iniciará la instalación:

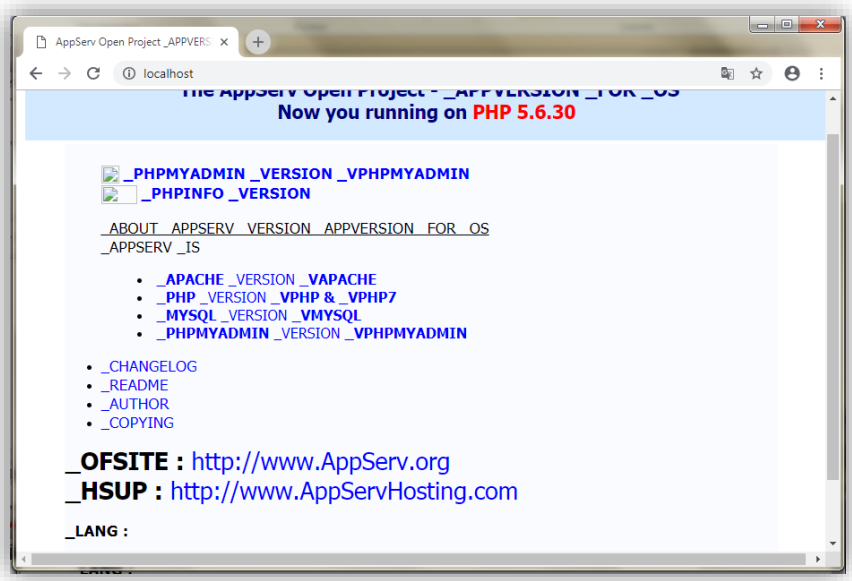


Para que funcione nuestros dos servidores instalados, deben de estar **INICIADOS,** es por ello que, debe de estar seleccionado Start Apache y Start MySql. Luego daremos clic en **FINISH**



Ahora ya podemos utilizar nuestro servidor.

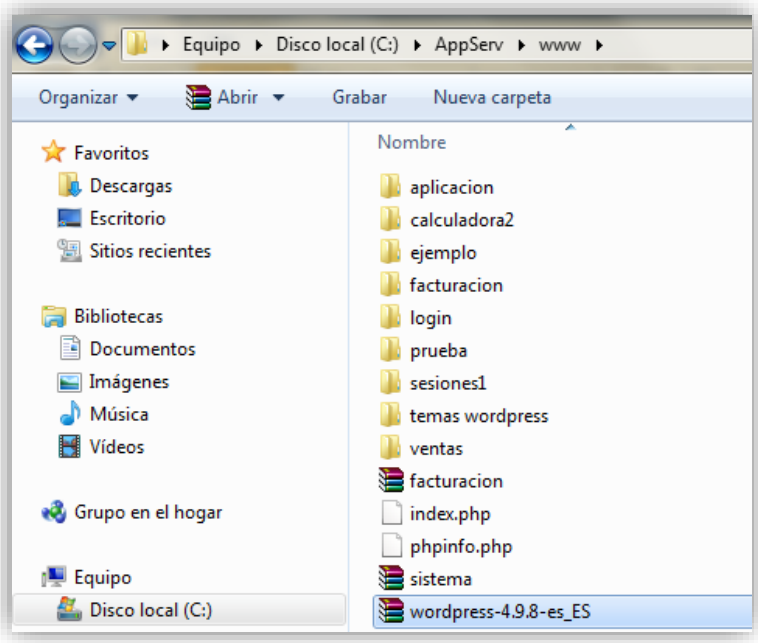
Para ver si nuestro servidor está en funcionamiento debemos de abrir nuestro navegador preferido y escribir **localhost**, si vemos que se abre la siguiente pantalla: (todo está OK)



1.5 El servidor

Al contrario de lo que ocurre en un curso HTML donde podemos almacenar nuestros documentos WEB en cualquier lugar de nuestro computador, con PHP no ocurre lo mismo. Tenemos que guardar nuestra aplicación PHP en una carpeta en concreto del servidor WEB apache para que pueda funcionar. Esta carpeta en mención es la carpeta **WWW** que se encuentra donde instalamos **AppServ**

Nuestro servidor WEB se localiza dentro de la unidad instalada, en mi caso sería así:



Cuando creemos un proyecto WEB, deberemos de crear una carpeta que identifique a nuestro proyecto, esta carpeta debe de estar guardado dentro de la carpeta www. En caso que lo grabe fuera de esta carpeta, no funcionará nuestro PHP pues no estaría dentro del servidor.

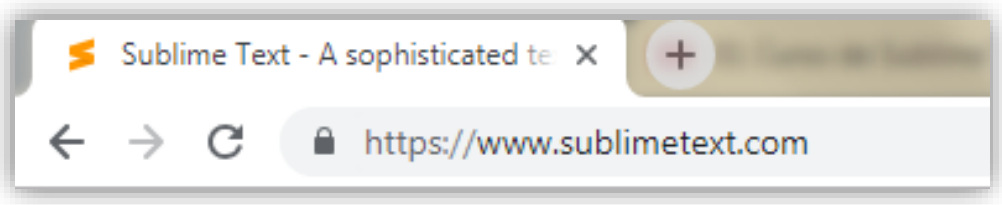
Como podemos observar en la imagen de arriba, cada carpeta es un proyecto diferente, y dentro de esta carpeta debemos de crear otras carpetas para colocar imágenes, css o cualquier otro detalle de organización referente a nuestra aplicación.

1.6 Editores para php

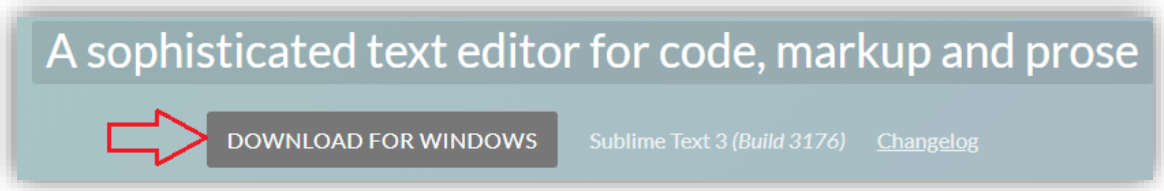
Tenemos múltiples editores para trabajar con PHP, así podemos mencionar al Dreamweaver, Brackets, Sublime text, etc. En nuestro caso vamos a utilizar el **SUBLIME TEXT.**

1.7 Descarga del sublime text

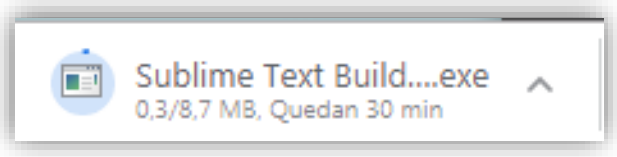
Lo primero que vamos a hacer es descargar sublime text de la página:



Daremos clic en **DOWNLOAD FOR WINDOWS**



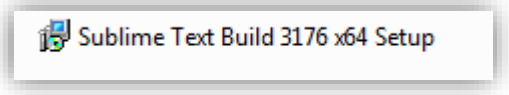
Automáticamente iniciará el proceso de descarga:



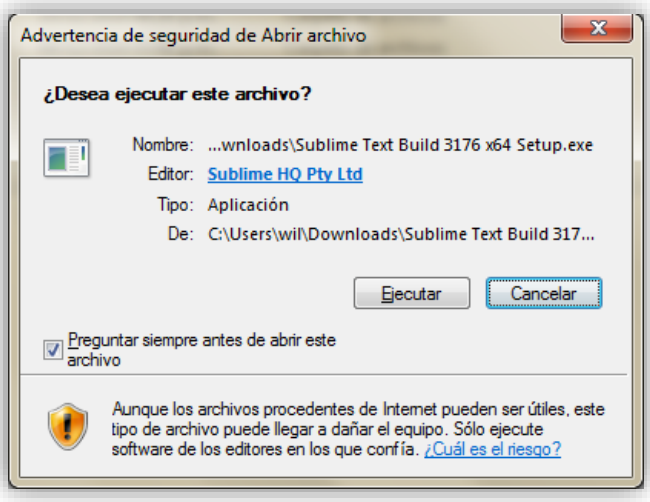
El archivo se encuentra en la carpeta de descargas y se grabará de la siguiente manera:

1.8 Instalación del sublime text

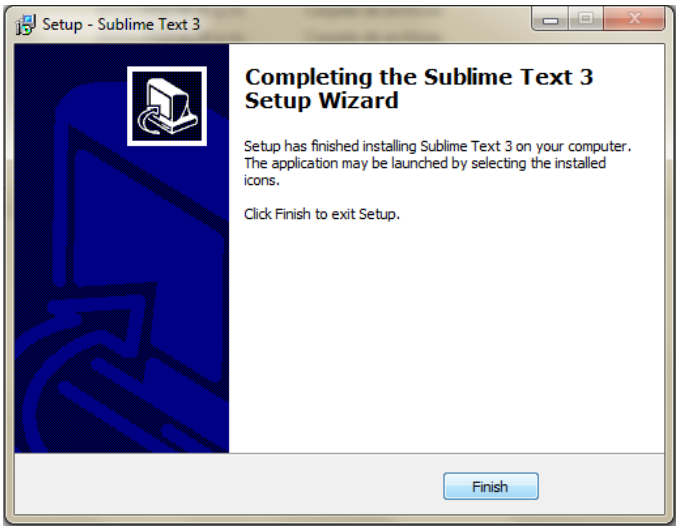
Hacer doble clic en el instalador de sublime text:



Luego de ejecutar el instalador, aparecerá el formulario de **Advertencia de seguridad de Abrir archivo.** Nos pregunta: ¿desea ejecutar el archivo?, a lo que responderemos que deseamos ejecutar el archivo y para ello daremos clic en **EJECUTAR**.



Y luego damos clic en siguiente a todas las opciones que salgan, para luego finalmente dar clic en **FINISH**



Para utilizar este programa solo hace falta abrirlo y empezar a programar.

1.9 Estructura del PHP

Para programar en PHP necesitamos saber HTML, por lo menos, la estructura básica del HTML y su funcionamiento.

Primeramente, abriremos el Sublime Text y grabaremos el archivo en el servidor, previamente en el servidor crearemos la carpeta **EJERCICIOS**. Y es dentro de esta carpeta que crearemos nuestro primer archivo llamado **ejemplo1.php**. se graba con esa extensión debido a que se trata de un programa en PHP y además después de que fue grabado, el **sublime text** automáticamente asimila que se trata de una aplicación web y pondrá a disposición las herramientas que se requiere para el PHP, por lo tanto, ni bien escribamos **<h** nos aparecerá las posibles etiquetas que inician con **<h** y generara automáticamente toda la estructura del HTML y sucede lo mismo cuando querramos trabajar con PHP.

```
<html>
<head>
        <title></title>
</head>
<body>
</body>
</html>
```

Lo anterior pertenece a las etiquetas básicas del HTML, y si queremos programar en PHP nos introduciremos dentro de la etiqueta <body> y todo el código PHP debe de comenzar con una etiqueta de apertura **<?php** y una etiqueta de cierre **?>** y se verá de la siguiente forma:

```
<html>
<head>
        <title></title>
</head>
<body>
<?php
// Todo dentro de esta etiqueta es considerado
// lenguaje de programación PHP
?>
</body>
</html>
```

Todo el código comprendido entre estas dos etiquetas, será nuestro código PHP. Se incluye estas instrucciones para que el servidor WEB apache sepa que esta porción de código lo tenga que interpretar él, mientras que el resto de código será interpretado por nuestro navegador.

1.10 Mi primer programa en php

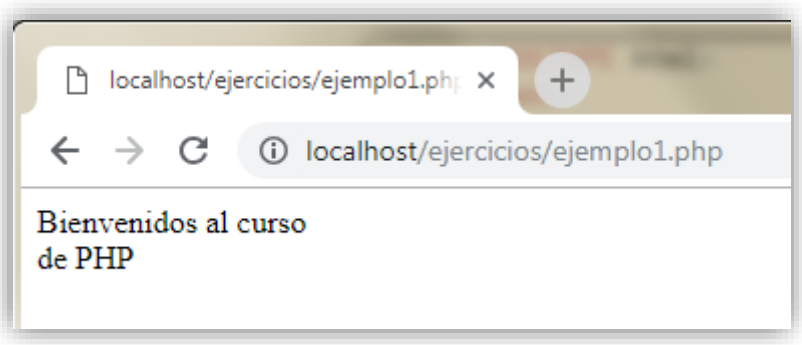
Print Permite imprimir en pantalla un mensaje, los que se encuentra dentro de las comillas es una cadena de caracteres, es lo que aparecerá en el navegador y el
 es una etiqueta que permite el salto de línea, es decir, el siguiente texto aparecerá abajo del comentario.

```
<html>
<head>
        <title></title>
</head>
<body>
<?php
```

```
print "Bienvenidos al curso <br>";
print "de PHP";
?>
</body>
</html>
```

Recordemos que el archivo fue grabado como **ejemplo1.php** dentro de la carpeta ejercicios. No se olvide de grabar siempre antes de ejecutar el programa, en caso contrario no se podrán ver los cambios.

Para ver si funciona, debes de abrir una ventana de tu navegador preferido, en mi caso usare el Google Chrome, y escribir: **localhost/ejercicios/ejemplo1.php** y los resultados se verán así:



NOTA. Cuando grabe una página, es conveniente no utilizar tildes y no dejar espacios.

TEMA 02

VARIABLES Y COMENTARIOS

2.1 Comentarios en PHP.

Los comentarios son pequeñas anotaciones que sirven a modo de ayuda al programador.

```
<?php
//esto es un comentario de una sola línea a partir de la doble barra
/* todo lo que está dentro de la barra y el asterisco es un comentario, este
comentario se usa cuando el comentario es de varias líneas */
?>
```

El comentario es invisible para nuestro programa cuando se ejecuta, cuando el comentario es en una sola línea se pone doble barra seguido del comentario. Cuando el comentario será de varias líneas escribimos /* */ dentro de ello escribimos todo el texto y será considerado como comentario.

Ejemplo:

```
<html>
<head>
        <title></title>
</head>
<body>
<?php
print "Bienvenidos al curso <br>"; // Imprime y hace salto de
línea
/* el salto de línea es <br> */
print "de PHP";
?>
</body>
</html>
```

Los comentarios también se pueden utilizar para hacer pruebas, es decir, para invalidar código, para ver qué sucedería si se quita algunas líneas de código PHP. Para ello el código que no queremos que se ejecute, lo pondríamos dentro de comentarios.

2.2 Las variables

Las variables son espacios en la memoria del computador donde se almacenará un valor que puede cambiar durante la ejecución del programa.

A la hora de programar en cualquier lenguaje, surge la necesidad de almacenar valores en algún sitio para utilizarlos en el futuro, por ejemplo, si necesitamos almacenar la **edad** de una persona, eso quiere decir que en un futuro vamos a utilizar el valor de esta variable. Para ello debemos de almacenar en algún sitio esa **edad**, se necesita si o si una variable.

Las variables en PHP siempre inician con el símbolo dólar ($), las variables tienen reglas, como, por ejemplo:

- No deben incluir símbolos extraños
- No debe de comenzar con valores numéricos
- Pueden llevar números, pero no al inicio
- No debe haber espacios en blanco
- Cada línea de código debe de terminar en punto y coma (;)

Lo más habitual en PHP es crear una variable y darle un valor, por ejemplo, si quiero almacenar mi nombre lo haría de la siguiente manera:

```
<?php
$nombre = "Wilson Mamani Rodrigo";
?>
```

Todas las sentencias en PHP terminan en punto y coma **(;)** y los valores de texto o cadena de caracteres va entre comillas dobles o comillas simples.

Lo que hicimos fue reservar un espacio en la memoria del computador donde se almacenara un valor y este valor está almacenado en la variable nombre. Y por ser variable puede cambiar. Como, por ejemplo:

```
<?php
$nombre = ‘Wilson Mamani Rodrigo’;
$nombre = ‘Luis Beltran’;
?>
```

Ahora la variable $nombre ya no contiene a **Wilson Mamani Rodrigo,** sino contendría a **Luis Beltran.** (A esto se le llama reasignación de variable)

En php no es necesario indicar el tipo de dato de la variable (entero, flotante o caracter), PHP es más flexible en cuestión a tipo de dato.

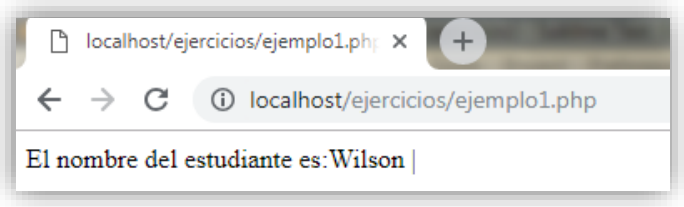
Las cadenas van entre comillas, por lo tanto, todo lo que va entre comillas dobles o comillas simples es considerado como si fuese un tipo de dato cadena de caracteres. En caso que querramos asignar un tipo de dato entero, únicamente debemos asignar ese tipo de dato numérico como entero. Para visualizar lo datos usaremos el **print.**

```
<html>
<head>
        <title></title>
</head>
<body>
<?php
$nombre = "Wilson";
$edad = 42;
print $nombre;
?>
</body>
</html>
```

El ejemplo anterior imprimirá en pantalla el contenido de la variable $nombre. Y ese contenido es **Wilson**. También puede funcionar una concatenación, y para concatenar utilizaremos en operador punto (.) de la siguiente manera:

```
<html>
<head>
        <title></title>
</head>
<body>
<?php
$nombre = "Wilson";
$edad = 42;
print "El nombre del estudiante es:". $nombre;
?>
</body>
</html>
```

Lo que saldrá en pantalla será:

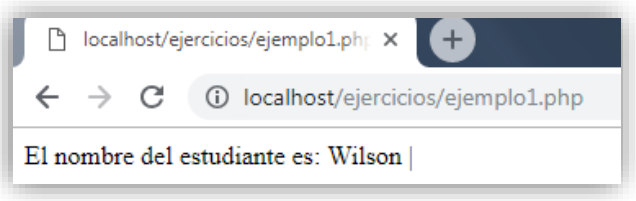


Otra forma de trabajar el ejemplo anterior sería:

```
<html>
<head>
        <title></title>
</head>
<body>
<?php
$nombre = "Wilson";
```

```
$edad = 42;
print "El nombre del estudiante es: $nombre";
?>
</body>
</html>
```

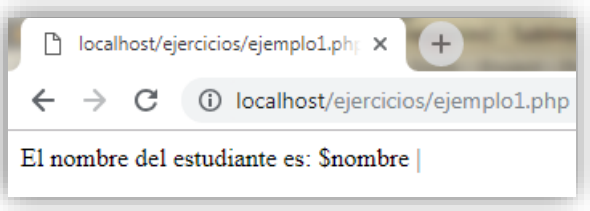
Podemos notar que hacemos llamada al valor de la variable dentro de la doble comilla, el resultado será:



Ahora, si sustituimos las comillas dobles con las comillas simples, el resultado será totalmente diferente, (debemos de recordarle que, lo que se encuentra dentro de comillas simples o comillas dobles son cadena de caracteres, con una diferencia de que las comillas dobles reconocen las variables, mientras que las comillas simples no lo reconocen, es decir lo toman como si fuera una simple cadena). Ejemplo:

```
<html>
<head>
        <title></title>
</head>
<body>
<?php
$nombre = "Wilson";
$edad = 42;
print 'El nombre del estudiante es: $nombre';
//la comilla simple no reconoce la variable.
?>
</body>
</html>
```

Imprimirá en pantalla el siguiente resultado:



Observe que la variable $nombre no lo reconoció como variable, sino, como una simple cadena.

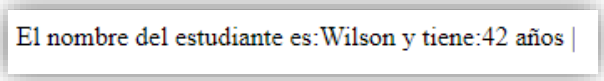
Para imprimir en pantalla también podemos utilizar la instrucción **echo** en lugar de **print,** funciona de similar manera: ejemplo:

echo “El nombre del estudiante es: $nombre”;

Otra forma de concatenación múltiple tenemos:

```
<html>
<head>
        <title></title>
</head>
<body>
<?php
$nombre = Wilson;
$edad = 42;
print "El nombre del estudiante es:" . $nombre . " y tiene:" . $edad . "
años";
?>
</body>
</html>
```

El resultado sería:



2.3 Diferencias en echo y print

Código fuente	Resultado

<html>
<head>
<title></title>
</head>
<body>
<?php
$nombre = Wilson;
$edad = 42;
echo $nombre, $edad; //estamos separando con comas diferentes variables
?>
</body>
</html>
	Wilson42

Echo $nombre, $edad; /*esto es algo que print no admite, más específicamente, print no admite comas */

El más utilizado es el **echo**

2.4 Flujo de ejecución

El flujo de ejecución de un programa va de arriba abajo, el navegador al cargar una página, lo primero que hace es ejecutar cualquier cosa que esté antes de la zona de PHP. Por ejemplo:

Código fuente	Resultado
<html> <head> <title></title> </head> <body> <?php	Este es el primer mensaje Este es el segundo mensaje

echo "Este es el primer mensaje "; echo "Este es el segundo mensaje "; ?> </body> </html>|	

Podemos ver que se está ejecutando de arriba abajo, línea a línea. Este tema es muy importante tenerlo presente, sobre todo cuando interviene las estructuras que interrumpen el flujo del programa, así podemos mencionar: *las condicionales, los bucles y las funciones.* Estas estructuras, rompen con el flujo de ejecución del programa porque en principio de lectura de arriba abajo se ve alterado volviendo otra vez hacia arriba o pegando saltos en su curso hacia abajo.

2.5 Las funciones.

Las funciones en PHP se declaran de la siguiente forma:

Se escribe la palabra reservada **function,** a continuación, el nombre que le queremos dar a esa función seguida de dos paréntesis, luego se abre y se cierra llaves. Ejemplo:

```
<?php
function obtener_datos()
{

}
?>
```

Estas estructuras rompen con el flujo de ejecución de un programa que es de arriba a abajo. Pues una función no ejecuta el código que hay dentro de la función a menos que esta función haya sido llamada.

El ámbito de la función o zona de actuación es desde la llave de apertura (**{**) hasta la llave de cierre (**}**), todo lo que esté dentro de estas dos llaves forma parte de la función, y es un código que no será ejecutado hasta que no se llame a la función.

```
1. function obtener_datos()
2. {
```

```
3.          $nombre = "Wilson";
4.          echo "Mi nombre es".$nombre;
5.  }
6.  obtener_datos();
7.  ?>
```

La forma de llamar a la ejecución de la función se encuentra en la línea 6 (**obtener_datos()**), si no incluimos esta instrucción, la función no será ejecutada, por lo tanto el orden de ejecución de flujo del programa seria así:

- Del 1 al 5 no se ejecuta
- La línea 6 se ejecuta y llama a la función
- Se ejecuta la línea 1, 2, 3, 4, 5, 7

Se puede llamar a la función antes de escribir la función o después de escribir la función.

```
1.  <?php
2.  obtener_datos();
3.  function obtener_datos()
4.  {
5.          $nombre = "Wilson";
6.          echo "Mi nombre es".$nombre;
7.  }
8.  ?>
```

En el anterior caso el orden de ejecución de flujo cambia y sería el siguiente: 1, 2, 3, 4, 5, 6, 7, 8

Otra forma de presentar las funciones es que en un bloque podemos tener las funciones y en otro bloque las llamadas a la función, así:

```
<?php
function obtener_datos()
{
        $nombre = "Wilson";
```

```
        echo "Mi nombre es".$nombre;
}
?>
<?php
obtener_datos();
?>
```

Otra forma de trabajar en PHP es que podemos utilizar códigos externos creados por nosotros o por otro programador y que podemos utilizar, para ello es necesario conocer de manera precisa el flujo de ejecución de un programa.

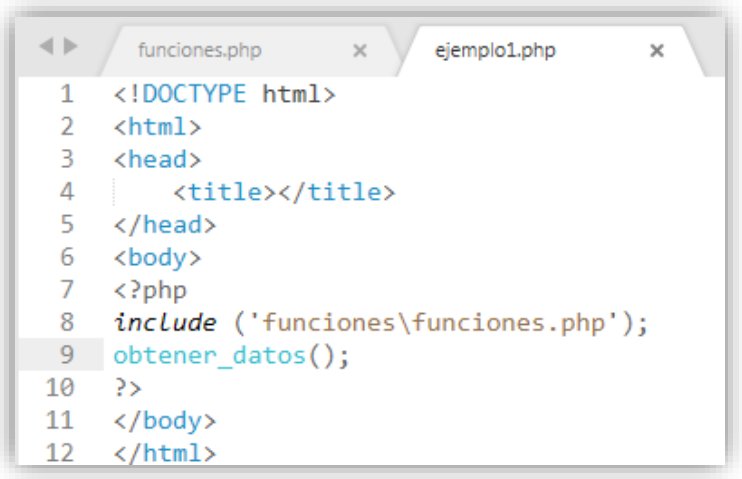
Cuando consideramos funciones muy útiles podemos crear un archivo PHP en donde almacenaremos todas nuestras funciones. Estas funciones estarán listas para cuando se les va a llamar o utilizar.

La forma de llamar a esta función que está ubicada en otro archivo es utilizando la instrucción **include (**'nombre del archivo donde se encuentra grabado las funciones a utilizar'**)**

/* utilizar comillas simples*/

Ejemplo: Dentro del servidor (www) se encuentra nuestra carpeta **ejercicios,** dentro de ésta crear la carpeta de nombre ***funciones*** así mismo dentro de esa carpeta guardar el archivo llamado **funciones.php** ahí podemos guardar todas nuestras funciones, para nuestro ejemplo ahí codificaremos la función obtener_datos

Y por otro lado dentro de la carpeta ejercicios tenemos grabado el archivo **ejemplo1.php** en donde hacemos la llamada a la función **obtener_datos**



De idéntica manera podemos utilizar la instrucción **require,** que funciona de similar manera que la instrucción **include**

2.6 Ámbito de las variables

Cuando hablamos de ámbito de variables nos referimos al lugar o zona de actuación de la variable.

Cuando declaramos una variable, esta variable tiene una zona de actuación, que depende de donde la declares y de cómo la declares. En PHP tenemos 3 formas de declaración de variables:

Ámbito local. Es cuando la variable es declarada dentro de una función, esto quiere decir que la variable funciona solo dentro de esa función, y fuera de la función esa variable no existe, es decir, no podemos mencionarla ni utilizarla

Ámbito global. La variable global la podemos declarar en cualquier lugar del PHP tanto dentro de una función como fuera de una función y es accesible desde cualquier lugar del código.

Ámbito super global. Sirve para poder acceder desde fuera del programa, es decir puedes acceder a la variable que está en otro archivo. Este ámbito es utilizado para enviar

información de un formulario a otro archivo de PHP. La variable súper global se declara como array.

Ejemplo: ¿Qué imprime el siguiente código?

Respuesta: Imprime 40.

La variable edad que está dentro de la función nada tiene que ver con la variable edad que está fuera de la función. Para PHP son dos variables totalmente diferentes. Por esta razón, si le decimos fuera de la función que imprima el contenido de la variable edad, este imprimirá el valor de la variable edad que este fuera de la función. Imprimirá 40 porque la línea 13 del código no accede en ningún caso a lo que hay almacenado dentro de la función y viceversa dentro de la función en ningún caso podemos acceder a lo que hay fuera.

Por ese motivo cuando le pedimos que se ejecute la función DameEdad(), hizo su trabajo, pero no accede a la variable edad que está fuera porque se trata de variables diferentes

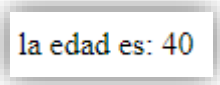
2.7 Declarando una variable global

Debemos de declarar la variable edad como variable global (esta declaración debe ser dentro de la función, no fuera de la función). Ahí si considera a edad como variable que podría hacer cambios en la primera variable.

ambito.php

```
</head>
<body>
    <?php
        $edad = "40";
        function DameEdad()
        {   global $edad;
            $edad="la edad es: ".$edad;
        }
        DameEdad();
        echo $edad;
    ?>
</body>
</html>
```

El resultado en pantalla sería



2.8 Declarando una variable super global

Para mostrar su funcionamiento crearemos un archivo **obtener.php** en la cual declararemos una variable dato. El valor de esta variable lo podremos utilizar en cualquier otro archivo haciendo uso de la instrucción **include o require.**

ambito.php | obtener.php

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
<?php
  $dato = "hola";
?>
</body>
</html>
```

ambito.php obtener.php

```
5   </head>
6   <body>
7       <?php
8           include ('funciones\obtener.php');
9           echo $dato;
10      ?>
11  </body>
12  </html>
```

Si ejecutamos **ambito.php** el resultado será:

hola

2.9 Variables estáticas

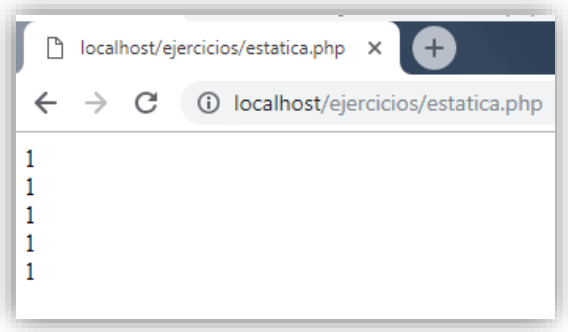
Cuando una función termina (hablamos del fin de función con la llave de cierre }), el valor de sus variables internas o locales se destruyen. En otras palabras, cuando ejecutamos por primera vez una función, las variables adoptan los valores, pero cuando sale de la función los valores que tenían dichas variables se destruyen.

estatica.php

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <title></title>
5   </head>
6   <body>
7   <?php
8       function incrementa()
9       {
10          $numero = 0;
11          $numero++;
12          echo $numero . "<br>";
13      }
14      incrementa();
15      incrementa();
16      incrementa();
17      incrementa();
18      incrementa();
19  ?>
20  </body>
21  </html>
```

En el ejemplo anterior el valor de las variables se destruye al finalizar la función, es por ello que las 5 llamadas que se tiene a la función siempre imprimirán el valor de 1.

Ejecutando el código nos muestra la siguiente respuesta:



Ahora que sucedería si requerimos que al finalizar la función los valores de las variables se mantengan, de tal forma que, si llamamos a la función varias veces, los valores de las variables se actualicen.

Para ello usamos **LA VARIABLE ESTÁTICA**, esto se consigue anteponiendo la instrucción **static** antes de la variable.

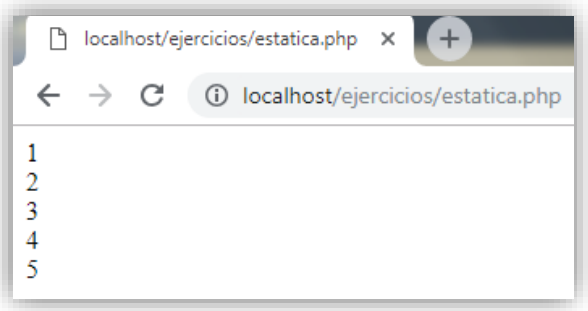
Por ejemplo: si declaramos ***static $numero = 0;***

Con esta declaración conseguimos que la línea donde hemos declarado la variable estática, se ejecute nada más que la primera vez que llamas a la función y por otro lado lo que consigues es que cuando la función finaliza, el valor de la variable se conserva.

estatica.php

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title></title>
5  </head>
6  <body>
7  <?php
8      function incrementa()
9      {
10         static $numero = 0;
11         $numero++;
12         echo $numero . "<br>";
13     }
14     incrementa();
15     incrementa();
16     incrementa();
17     incrementa();
18     incrementa();
19 ?>
20 </body>
21 </html>
```

El resultado es:



En resumen, Las variables estáticas conservan su valor al salir de su ámbito.

TEMA 03

STRINGS Y OPERADORES DE COMPARACIÓN

3.1 Strings

En este tema vamos a tratar de las formas en que se debe de declarar un String en PHP y de cómo comparar Strings. El String se refiere a cadenas de texto que se muestran en una página web.

Vamos a trabajar con Dreamweaver, en esta sesión vamos a ver las diferentes formas de declarar un String, para ello primeramente vamos a crear una hoja de estilos CSS para resaltar un texto, para ello abrimos dentro de head la etiqueta Style, para ello creamos un selector de clase al cual denominamos **resaltar,** y que va encargarse de resaltar el texto en color **rojo,** y en negrita, luego cerramos el **style**.

```
<head>
<meta charset="utf-8">
<title>Documento sin título</title>
        <style>
                .resaltar{
                        color:#f00;
                        font-weight: bold
                }
        </style>
</head>
```

Luego, dentro del body vamos a crear nuestra zona PHP, e imprimimos por pantalla la instrucción **echo** e incluimos un párrafo (dentro de PHP podemos incluir etiquetas HTML), “Esto es un ejemplo, luego cerramos la etiqueta de párrafo”, para ver cambios presionamos F12.

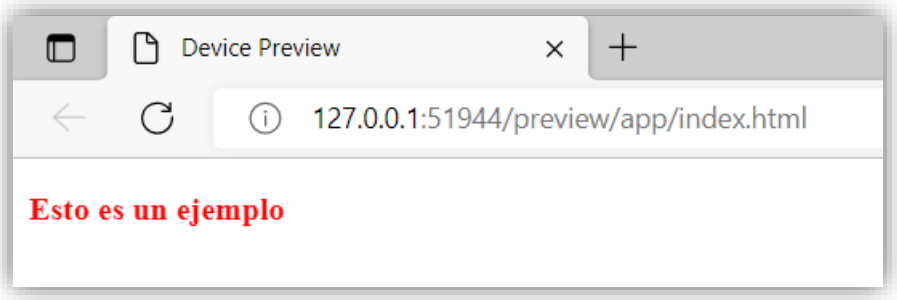
```
<?php
        echo "<p>Esto es un ejemplo</p>"
?>
```

Ahora debemos de resaltarla utilizando la hoja de estilos interna que hemos creado anteriormente, para ello debemos de agregar dentro la etiqueta de apertura **p** el atributo **class** y dentro del atributo class utilizamos la clase **resaltar** de nuestro CSS creado. Esta clase debe de estar dentro de comilla simple, porque no puede haber comilla doble dentro de comillas dobles. Luego guardamos y ejecutamos.

Para sacar comilla simple, puede usar ALT + 39

```
<html>
<head>
<meta charset="utf-8">
<title>Documento sin título</title>
        <style>
                .resaltar{
                        color:#f00;
                        font-weight: bold
                }
        </style>
</head>
<body>
<?php
  echo "<p class = 'resaltar'> Esto es un ejemplo</p>"
?>
</body>
</html>
```

Presionamos F12 y podremos ver el resultado. Podemos notar que ahora el texto esta resaltado en rojo y negrita.



Cuando vemos el caso **echo " Esto es un ejemplo ",** donde la comilla doble esta fuera de la cadena de texto. También se puede invertir la posición de las cadenas: **' Esto es un ejemplo ',** es decir, las comillas simples afuera del texto y las comillas dobles dentro del texto, podrá notar que se obtiene el mismo resultado.

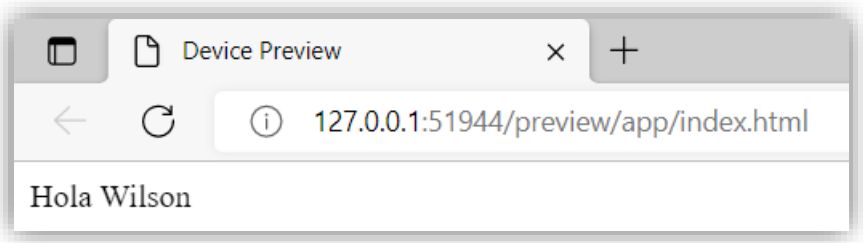
Cuando usemos en toda la cadena pura comilla doble, podemos usar la barra invertida para realizar un escape de caracteres en un String, con esto le estamos diciendo a PHP que el carácter que va a continuación de la barra invertida no forma parte del string, con lo cual estaríamos escapando de ese caracter, por ejemplo:

```
<?php
 echo "<p class = \"resaltar\"> Esto es un ejemplo</p>"
?>
```

Funciona de igual manera con la comilla simple.

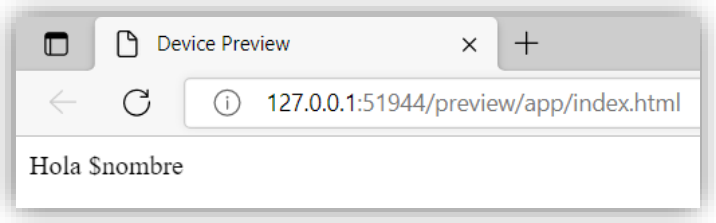
También podemos trabajar con variables, sobre todo cuando necesitamos concatenar **Strings** con valores de variable. por ejemplo:

```
<?php
$nombre = "Wilson";
 echo "Hola $nombre";
?>
```



Concatena el String con lo que contiene la variable, se debe de tener en cuenta que cuando hagas la llamada al contenido de una variable no debes usar comillas simples. Si en caso usaste comillas simples la llamada al valor de una variable **$nombre** se imprimirá $sombre, mas no imprimirá el valor de la variable. Ejemplo:

```php
<?php
$nombre = "Wilson";
  echo 'Hola $nombre';
?>
```



3.2 Comparación de cadena de caracteres entre sí.

Se usa para ver si dos cadenas son iguales o si no son iguales. en PHP tenemos dos funciones que se encargan de comparar cadenas.

Strcmp que es una abreviatura de String compare, que permite comparar valores de tipo strings, teniendo en cuenta si estas comparando mayúsculas y minúsculas.

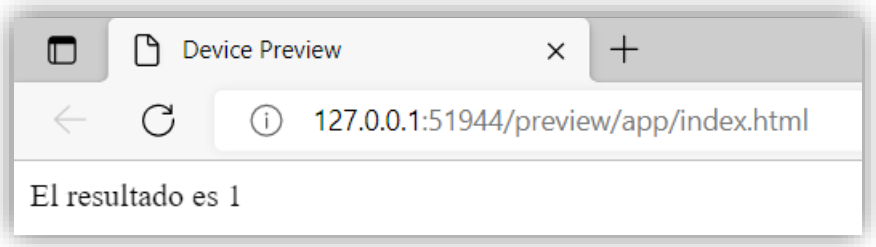
Strcasecmp compara valores de tipo string ignorando si están en mayúscula o en minúscula.

Ambas funciones devuelven un valor de **cero,** si los valores coinciden y un **uno** si los valores comparados no coinciden.

Recordemos que en programación 1 = True y 0 = False.

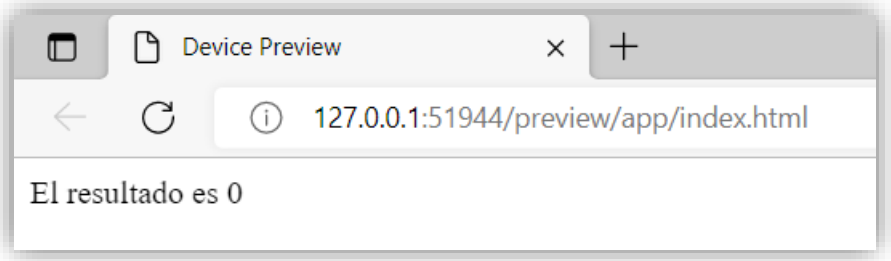
Ejemplo:

```php
<?php
        $variable1 = "wilson";
        $variable2 = "WILSON";
        $resultado = strcmp($variable1, $variable2);
        echo "El resultado es $resultado";
?>
```



El resultado es 1 porque utilice la función **strcmp,** por lo que las cadenas no son iguales, sin embargo, si utilizamos la otra función **strcasecmp** las cadenas si son iguales pues es indistinto si son mayúsculas o minúsculas.

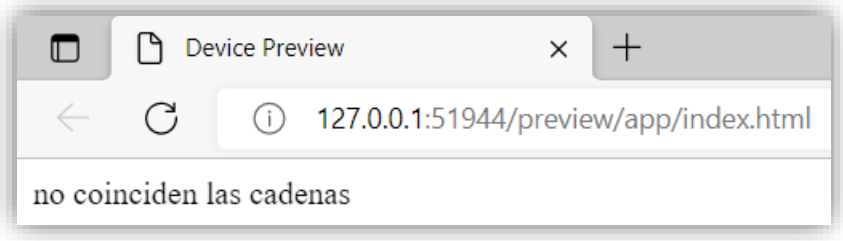
```
<?php
        $variable1 = "wilson";
        $variable2 = "WILSON";
        $resultado = strcasecmp($variable1,
$variable2);
        echo "El resultado es $resultado";
?>
```



Luego de analizar las funciones, podemos usar la instrucción **IF** para poder evaluar las comparaciones. Por ejemplo:

```
<?php
        $variable1 = "wilson";
        $variable2 = "WILSON";
        $resultado = strcmp($variable1, $variable2);
        if($resultado){
                Echo "no coinciden las cadenas";
        }
```

```
        else
        {
                echo "las cadenas coinciden";
        }
?>
```



if($resultado) significa que todo lo que está dentro del paréntesis es asumido como verdadero, o sea True o 1

if(!$resultado) significa que todo lo que está dentro del paréntesis es asumido como falso, esto por haber agregado el símbolo !

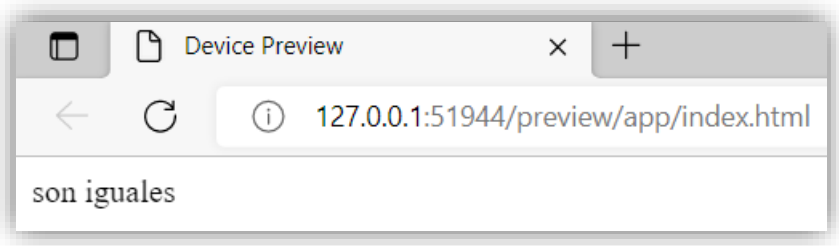
3.3 Operadores de comparación

Operador	Nombre	Ejemplo	Devuelve verdadero cuando
==	Igual	$a == $b	$a es igual a $b
===	Idéntico	$a === $b	$a es igual a $b y ambos son del mismo tipo
!=	Distinto	$a != $b	$a es distinto de $b
!==	No idéntico	$a !== $b	$a no es idéntico a $b
<	Menor que	$a < $b	$a es menor que $b
>	Mayor que	$a > $b	$a es mayor que $b
<=	Menor o igual	$a <= $b	$a es menor o igual que $b
>=	Mayor o igual	$a >= $b	$a es mayor o igual que $b

Vamos a realizar un ejemplo para poder explicar este tema

Ejemplo 1. Comparación utilizando el operador igual

```php
<?php
	$var1 = 20;
	$var2 = "20";
	$var3 = "hola";
	if($var1==$var2)
	{
		echo "son iguales";
	}
	else
	{
		echo "no son iguales";
	}
?>
```



Ejemplo 2. Comparación utilizando el operador idéntico a.

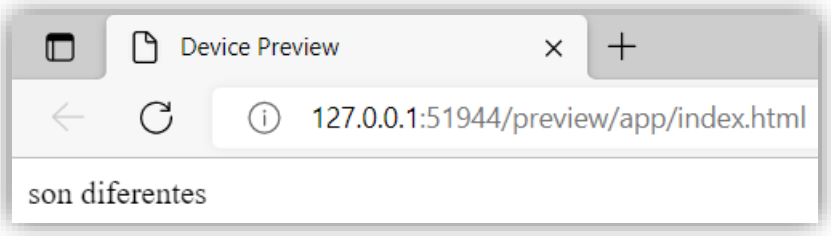
```php
<?php
	$var1 = 20;
	$var2 = "20";
	$var3 = "hola";
	if($var1===$var2)
	{
		echo "son idénticos";
	}
	else
	{
		echo "no son idénticos";
	}
?>
```

El resultado será:

no son idénticos

Ejemplo 3. Comparación utilizando el operador es distinto de.

```php
<?php
        $var1 = 20;
        $var2 = "20";
        $var3 = "hola";
        if($var1!=$var3)
        {
                echo "son diferentes";
        }
        else
        {
                echo "son iguales";
        }
?>
```



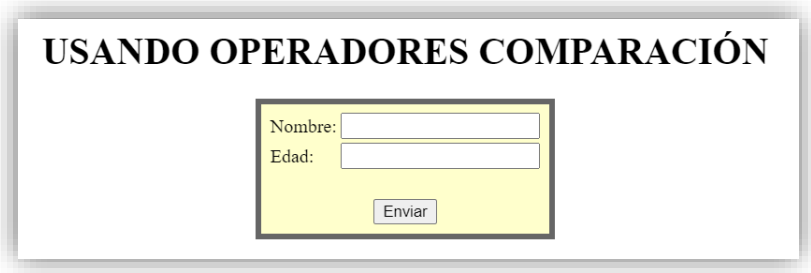
Vamos a revisar el siguiente código fuente.

```html
<html>
<head>
<meta charset="utf-8">
<title>Documento sin título</title>
<style>
        h1{
                text-align:center;
        }
        table{
                background-color:#FFC;
```

```
            padding:5px;
            border:#666 5px solid;
        }
        .no_validado{
            font-size:18px;
            color:#F00;
            font-weight:bold;
        }
        .validado{
            font-size:18px;
            color:#0C3;
            font-weight:bold;
        }
</style>
</head>
<body>
<h1>USANDO OPERADORES COMPARACIÓN</h1>
<form action="" method="post" name="datos_usuario" id="datos_usuario">
 <table width="15%" align="center">
  <tr>
   <td>Nombre:</td>
   <td><label for="nombre_usuario"></label>
   <input type="text" name="nombre_usuario" id="nombre_usuario"></td>
  </tr>
  <tr>
   <td>Edad:</td>
   <td><label for="edad_usuario"></label>
   <input type="text" name="edad_usuario" id="edad_usuario"></td>
  </tr>
  <tr>
   <td> </td>
   <td> </td>
  </tr>
```

```
   <tr>
    <td colspan="2" align="center"><input type="submit" name="enviando" id="enviando" value="Enviar"></td>
   </tr>
  </table>
</form>
</body>
</html>
```

Lo que tenemos en este archivo es una hoja interna dentro del **head,** unos selectores de etiqueta y unos selectores de clase, luego dentro del **body** tenemos un formulario, y si ejecutamos con F12 veremos el siguiente formulario.



Podemos observar un cuadro de texto para insertar un nombre y un cuadro de texto para insertar la edad, el resto es la decoración que se hizo con CSS y HTML.

Debemos de resaltar el nombre con el cual se ha identificado a los 2 cuadros de texto, y también de cómo hemos identificado al botón de enviar.

El formulario está identificado como **Id = "datos_usuario",** el cuadro de texto lo hemos identificado como **id = "nombre_usuario"**, la edad está identificado como **id = "edad_ usuario"**, y el botón lo hemos identificado con **id = "enviando"**

```
<form action="validacion.php" method="post" name="datos_usuario" id="datos_usuario">
  <table width="15%" align="center">
    <tr>
      <td>Nombre:</td>
      <td><label for="nombre_usuario"></label>
      <input type="text" name="nombre_usuario" id="nombre_usuario"></td>
    </tr>
    <tr>
      <td>Edad:</td>
      <td><label for="edad_usuario"></label>
      <input type="text" name="edad_usuario" id="edad_usuario"></td>
    </tr>
    <tr>
      <td> </td>
      <td> </td>
    </tr>
    <tr>
      <td colspan="2" align="center"><input type="submit" name="enviando" id="enviando"
    </tr>
  </table>
</form>
```

Problema. Implementar acceso restringido para una página según datos ingresados.

Estos nombres de id tenemos que tenerlo presente para evaluar si el nombre introducido en el cuadro de texto es un nombre en concreto o no lo es, para darle un acceso a una página en función de que, si el nombre es válido o no es válido, también trabajaremos con la edad y los operadores de comparación, funcionaría como si la persona que está introduciendo datos en el formulario es menor de 18 años, salga un mensaje de que no puede ingresar, y si es mayor de 18 si se le permita acceder.

Para resolver este problema vamos a situarnos debajo de la finalización del formulario, es ahí donde programaremos nuestro código PHP. Debe ser después del formulario, porque si programamos antes del formulario, los mensajes saldrán antes. Luego creamos nuestra zona PHP.

Luego tenemos que almacenar en una variable lo que el usuario haya introducido en: **id = "nombre_usuario"**, **id = "edad_ usuario"**, y también tenemos que decirle al programa que haga su trabajo siempre y cuando el usuario haya pulsado en el botón enviar del formulario.

if(isset($_POST["enviando"]))

isset es una función predefinida de PHP que tiene múltiples usos, uno de los usos de esta función es comprobar si se ha pulsado el botón de enviar o no.

enviando es el nombre del botón que habíamos definido en el identificador del botón del formulario.

El código anterior comprueba si hemos pulsado el botón de enviar en el formulario, y en caso de que hemos pulsado el botón, el flujo del programa se introducirá dentro del **if** y leerá el código que está dentro del él.

$_POST en PHP es una variable superglobal, estas variables son arrays, es por ello que tiene corchetes, y con ello almacenamos el valor de una variable a una variable denominada $usuario, es decir, se almacena en la variable, lo que el usuario haya introducido en el cuadro de texto Nombre.

```php
<?php

    if(isset($_POST["enviando"]))
    {
        $usuario = $_POST["nombre_usuario"];
        $edad = $_POST["edad_usuario"];

        if($usuario=="wilson")
        {
            echo "puedes ingresar";
        }
        else
        {
            echo "no puedes ingresar";
        }

    }

?>
```

También podemos usar conceptos de Strings, como, por ejemplo, podemos usar los estilos **no_validado** y **validado** que habíamos definido anteriormente para que el mensaje **echo** sea más atractivo. Para ello añadimos las etiquetas <p> en el texto y configuramos la clase.

```css
.no_validado{
    font-size:18px;
    color:#F00;
    font-weight:bold;
}

.validado{
    font-size:18px;
    color:#0C3;
    font-weight:bold;
}
```

Si insertamos CSS en nuestro programa quedaría.

```php
<?php

    if(isset($_POST["enviando"]))
    {
        $usuario = $_POST["nombre_usuario"];
        $edad = $_POST["edad_usuario"];

        if($usuario=="wilson" && $edad >= 18)
        {
            echo "<p class='validado'>puedes ingresar</p>";
        }
        else
        {
            echo "<p class='no_validado'>no puedes ingresar</p>";
        }
    }
?>
```

Si en la casilla nombre ingreso **wilson** y en edad ingreso 46, el resultado sería:

puedes ingresar

Si en la casilla ingresamos **juan y en edad** 23, el resultado sería:

no puedes ingresar

NOTA. Para que ingrese debe de cumplir ambas condiciones especificadas en el IF.

3.4 Archivo VALIDACION.PHP

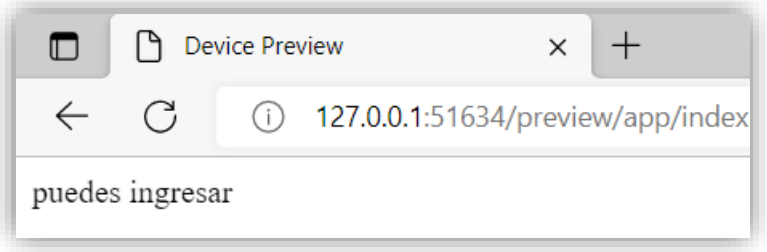
Podemos crear un nuevo archivo llamado validación.php y en ahí podemos insertar el código PHP que hemos realizado anteriormente. Y la forma de comunicar que el formulario al hacer clic en **enviar,** lea la información que hay en el archivo php.

Esto lo hacemos en el formulario desde el atributo **acction,** esta etiqueta action por defecto está vacío, y si dentro de las comillas ponemos validación.php, lo que estamos haciendo, es que el formulario debe comunicarse con este archivo gracias a las variables superglobales, podemos trasladar la información insertada en los cuadros de texto y es ahí donde realmente trabajan las variables superglobales.

En el archivo operadores.php debemos de modificar el action del formulario con validación.php como se muestra abajo.

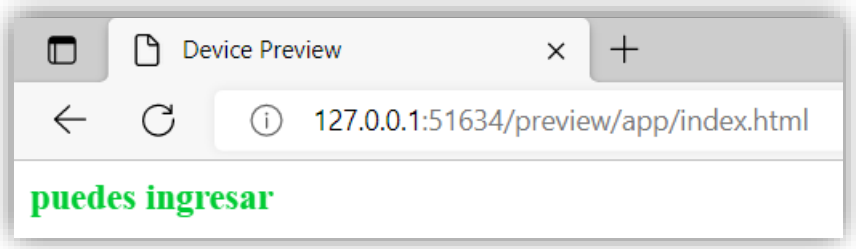
```
<form action="validacion.php" method="post" name="datos_usuario" id="datos_usuario">
  <table width="15%" align="center">
    <tr>
      <td>Nombre:</td>
      <td><label for="nombre_usuario"></label>
      <input type="text" name="nombre_usuario" id="nombre_usuario"></td>
```

Luego ejecutamos el archivo operadores.php y el resultado se muestra así:



No se puede visualizar los CSS que hemos aplicado porque está haciendo la llamada a otro archivo que es validación.php

Para solucionar este problema, lo que podemos hacer es llevar o copiar los estilos CSS a nuestro archivo de validación.php y luego quedaría todo solucionado.



3.5 Constantes

En este tema vamos a ver lo que son las constantes propias (aquellas que podemos crear nosotros), y las constantes predefinidas (son aquellas constantes que vienen definidas en el lenguaje de programación).

Una constante es lo contrario de una variable, es decir, una constante es un espacio que se reserva en la memoria del computador, donde se almacena un valor, pero este valor no podrá cambiar durante la ejecución del programa.

En una variable podemos cambiar su valor, pero en una constante no.

3.5.1 Declaración de constantes en PHP

Se utiliza una función define, y dentro de esta función que admite 3 parámetros, donde 2 son obligatorios y 1 es opcional, se introduce entre comillas el nombre de la constante y a continuación el valor.

define(“Nombre de la constante”, valor)

Donde el valor puede ser numérico y de tipo texto (en este caso va entre comillas).

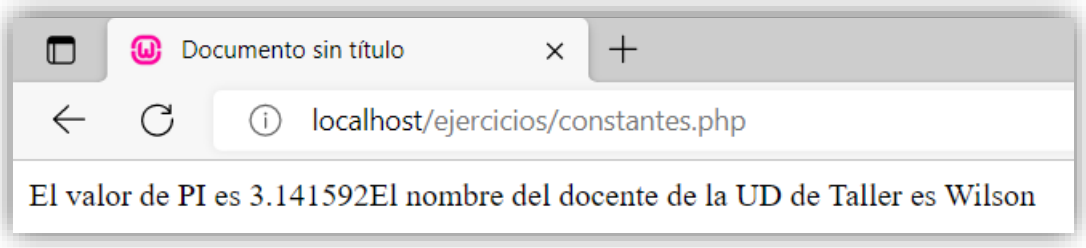
3.5.2 Aspectos a tener en cuenta al momento de usar las constantes.

- Por convenio el nombre de las constantes debe ir en mayúscula, pero si utilizamos un tercer argumento podemos ignorar esta recomendación, pero siempre veremos las constantes en mayúsculas.
- El nombre de las constantes no debe de llevar el símbolo dólar $
- Es obligatorio el uso de la función define() para definir constantes.
- El ámbito de una constante por defecto es global.
- Las constantes no se pueden redefinir.

- Las constantes solo pueden almacenar valores escalares (un valor escalar es un valor que no se puede dividir en partes más pequeñas, un array no puede ser una constante)

Ejemplo de declaración de constantes

```php
<?php
    define("PI", 3.141592);
    define("DOCENTE_TALLER", "Wilson");
    echo "El valor de PI es ". PI;
    echo "El nombre del docente de la UD de Taller es ". DOCENTE_TALLER;
?>
```



Debo mencionar que una constante no puede visualizar el valor de la constante como si fuera una variable, es decir, no podemos poner el nombre de la constante dentro de las comillas porque lo tomaría como si fuese una cadena y no una constante. Es por eso que, para llamar al valor de una constante, debemos de llamarla con la concatenación.

Y la llamada debe ser también en mayúscula, si la llamamos en minúscula tendremos error. Si queremos que el nombre de la constante sea insensible a las mayúsculas, debemos de poner **true** como tercer parámetro.

3.5.3 Constantes predefinidas.

Las constantes predefinidas son propias del lenguaje de programación php y vienen en mayúsculas y vienen precedidas de dos guiones bajos y terminan con dos guiones bajos. En php también lo conocen como constantes mágicas. Estas constantes estarán presentes dependiendo de la versión de PHP, en __LINE__ quiere decir que cuando pongo esa constante nos va decir el número de línea dentro del código php donde se encuentra esa instrucción. Las funciones de las demás constantes las muestro a continuación:

Nombre	Descripción
__LINE__	El número de línea actual en el fichero.
__FILE__	Ruta completa y nombre del fichero con enlaces simbólicos resueltos. Si se usa dentro de un include, devolverá el nombre del fichero incluido.
__DIR__	Directorio del fichero. Si se utiliza dentro de un include, devolverá el directorio del fichero incluído. Esta constante es igual que dirname(__FILE__). El nombre del directorio no lleva la barra final a no ser que esté en el directorio root.
__FUNCTION__	Nombre de la función.
__CLASS__	Nombre de la clase. El nombre de la clase incluye el namespace declarado en (p.e.j. Foo\Bar). Tenga en cuenta que a partir de PHP 5.4 __CLASS__ también funciona con traits. Cuando es usado en un método trait, __CLASS__ es el nombre de la clase del trait que está siendo utilizado.
__TRAIT__	El nombre del trait. El nombre del trait incluye el espacio de nombres en el que fue declarado (p.e.j. Foo\Bar).
__METHOD__	Nombre del método de la clase.
__NAMESPACE__	Nombre del espacio de nombres actual.
ClassName::class	El nombre de clase completamente cualificado. Véase también ::class.

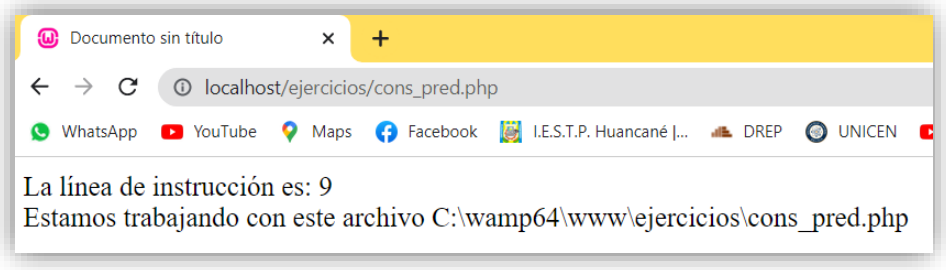
Ejemplo.

Realizar un programa que muestre la línea de código y el nombre del archivo incluido la ruta de ubicación.

Solución:

```php
<?php
echo "La línea de instrucción es: ".__LINE__ . "<br>";
echo "Estamos trabajando con este archivo ".__FILE__;
?>
```

Resultado.



Fíjese que con la declaración de constantes predefinidas por php, con __LINE__ obtenemos el número de línea donde se llama a la constante. Y con __FILE__ obtenemos la ruta y el nombre del archivo del código fuente.

En la página de php.net además de venir las constantes mágicas, ofrece otras constantes predefinidas, también podrá ver algunos ejemplos.

TEMA 04

FUNCIONES MATEMÁTICAS Y CASTINGS

4.1 Operadores matemáticos

Vamos a ver cómo realizar cálculos matemáticos en este lenguaje de programación, para ello vamos a conocer los operadores matemáticos.

Operador	Nombre
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo
++ +=	Incremento
-- -=	Decremento

Tenemos al operador suma que nos permite sumar dos valores, el operador resta para restar, el operador asterisco que sirve para multiplicar, la barra invertida que permite dividir, el operador módulo que nos devuelve el resto de una división, el ++ incrementa en 1 el valor que tenga almacenado en una variable, el – decrementa o resta en uno el valor que hay almacenado en una variable.

Vamos a crear un formulario con dos cuadros de texto, que es ahí donde vamos a introducir los valores numéricos, un menú desplegable que es donde vamos a escoger la operación a realizar y el botón de enviar.

Para el funcionamiento del formulario, el usuario deberá de introducir dos números, uno en cada cuadro de texto, en el desplegable debe elegir la operación a realizar, y al pulsar el botón enviar, toda la información ingresada se enviará al servidor, luego el servidor procesa los datos enviados y envía una respuesta para que aparezca después del formulario.

El código HTML para esa visualización sería:

```
<body>
<p> </p>
<form name = "form1" method="post" action = "">
    <p>
    <label for="num1"></label>
    <input type="text" name="num1" id="num1">
    <label for="num2"></label>
    <input type="text" name="num2" id="num2">
    <label for="operacion"></label>
    <select name="operacion" id="operacion">
        <option>Suma</option>
        <option>Resta</option>
        <option>Multiplicación</option>
        <option>División</option>
        <option>Módulo</option>
    </select>
    </p>
    <p>
    <input type="submit" name="button" id="button"
           value="enviar" onclic="prueba">
    </p>
</form>
</body>
```

Para que la operación sea resuelta primero debemos de determinar el lugar donde queremos que salga la respuesta, encima o debajo del formulario. Si queremos que la respuesta aparezca debajo del formulario debemos de programar PHP debajo del código del formulario, si queremos que salga por encima, debemos programarlo por encima.

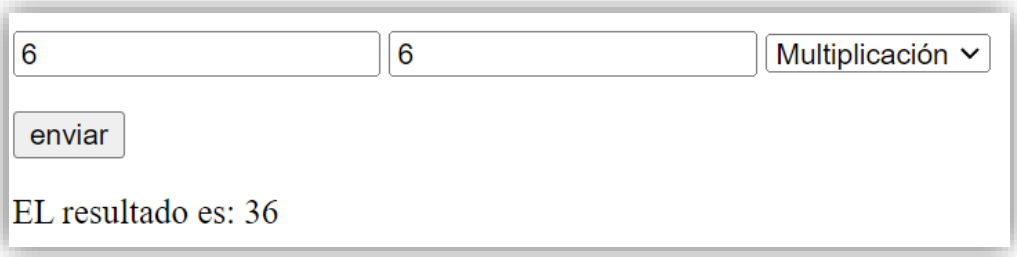
Nota. Recuerde que la función strcmp() devuelve como resultado 0 si las comparaciones son idénticas, es por eso que negamos al resultado de la comparación para que, lo que está dentro del if sea 1 o true e ingrese a la iteración.

Las comparaciones en el if deben ser iguales a como aparecen en el desplegable, sino habrá error.

El código PHP para el programa es:

```php
<?php
if(isset($_POST["button"]))
{
    $numero1=$_POST["num1"];
    $numero2=$_POST["num2"];
    $operacion=$_POST["operacion"];

    if(!strcmp("Suma",$operacion))
    {
        echo "EL resultado es: ". ($numero1+$numero2);
    }
    if(!strcmp("Resta",$operacion))
    {
        echo "EL resultado es: ". ($numero1-$numero2);
    }
    if(!strcmp("Multiplicación",$operacion))
    {
        echo "EL resultado es: ". ($numero1*$numero2);
    }
    if(!strcmp("División",$operacion))
    {
        echo "EL resultado es: ". ($numero1/$numero2);
    }
    if(!strcmp("Módulo",$operacion))
    {
        echo "EL resultado es: ". ($numero1%$numero2);
    }
}

?>
```

Si ejecuto la aplicación y pruebo el programa, obtenemos:



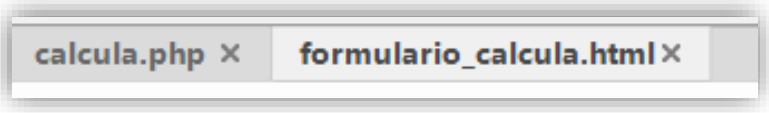
4.2 Paso de parámetros a una función

Ejercicio. Modificar el programa anterior y convertirlo en dos archivos, un archivo HTML que contendrá el formulario, y un archivo PHP que contendrá todo el código PHP. De tal forma que cuando pulsemos el botón enviar del formulario, el formulario enviará la información que hay en los cuadros de texto al archivo PHP que será el encargado de procesar la información y de devolvernos los resultados. También utilizaremos las funciones a las cuales vamos a realizar llamadas. En esta ocasión vamos a utilizar el término de paso de parámetros a una función.

Solución

Vamos a crear dos archivos, en un archivo pondremos todo el código HTML y en el otro archivo pondremos todo el código PHP, y tenemos que tener presente algunas modificaciones.

El archivo **calcula.php** no tiene estructura HTML porque lo único que hará es hacer o procesar un cálculo. Debemos notar que el archivo que contiene el HTML sea de extensión PHP, tranquilamente puede ser HTML (si tuviese por alguna razón código PHP siempre debe ser guardado como PHP)



NOTA. Para poder enviar información que tenemos en un formulario HTML a un archivo diferente en PHP, tenemos que ir a la etiqueta de apertura **form1** y en el atributo **action** debemos poner el nombre del archivo PHP al que queremos enviar la información, en este caso pondremos **calcula.php**

```
<body>
<p> </p>
<form name = "form1" method="post" action = "calcula.php">
    <p>
    <label for="num1"></label>
    <input type="text" name="num1" id="num1">
    <label for="num2"></label>
    <input type="text" name="num2" id="num2">
```

Ahora vamos a modificar el archivo PHP utilizando funciones. En donde, vemos aparentemente está bien, pero si ejecutamos el programa tendremos errores con el ámbito de las variables, donde las variables num1, num2 y operación no son accesibles desde la función.

```php
<?php
if(isset($_POST["button"]))
{
    $numero1=$_POST["num1"];
    $numero2=$_POST["num2"];
    $operacion=$_POST["operacion"];
    calcular();
}

function calcular()
{
    if(!strcmp("Suma",$operacion))
    {
        echo "EL resultado es: ". ($numero1+$numero2);
    }
    if(!strcmp("Resta",$operacion))
    {
        echo "EL resultado es: ". ($numero1-$numero2);
    }
    if(!strcmp("Multiplicación",$operacion))
    {
        echo "EL resultado es: ". ($numero1*$numero2);
    }
    if(!strcmp("División",$operacion))
    {
        echo "EL resultado es: ". ($numero1/$numero2);
    }
    if(!strcmp("Módulo",$operacion))
    {
        echo "EL resultado es: ". ($numero1%$numero2);
    }
}
```

Para ejecutar el programa debemos de llamar al archivo donde se encuentra el HTML, y al realizar la operación veremos errores o en su defecto no realizará ninguna operación.

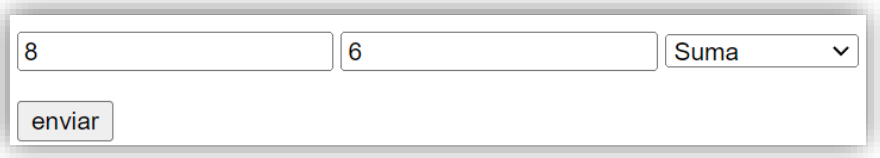
Para solucionar este problema explicare el concepto de paso de parámetros: en la función podemos decirle que puede recibir un valor cuando se le hace la llamada y se hace declarando una variable dentro del paréntesis. Además, vemos que no reconoce las variables numero1 ni numero2 porque están declaradas dentro de un if, y nosotros lo estamos queriendo utilizar dentro de una función que esta fuera de ese if. (el ámbito de la variable declarada numero1 y numero2 solo pertenece dentro del if , nada mas)

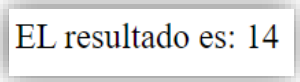
```php
if(isset($_POST["button"]))
{
    $numero1=$_POST["num1"];
    $numero2=$_POST["num2"];
    $operacion=$_POST["operacion"];
    calcular($operacion);
}
```

Entonces esas variables son invisibles fuera de esta instrucción, la solución es declarar numero1 y numero2 como variables globales para que sean reconocidas en cualquier lugar de este código PHP y eso lo hacemos declarando las variables como globales en cada if en la que se le llama.

```php
<?php
if(isset($_POST["button"]))
{
    $numero1=$_POST["num1"];
    $numero2=$_POST["num2"];
    $operacion=$_POST["operacion"];
    calcular($operacion);
}

function calcular($dato_operador)
{
    if(!strcmp("Suma",$dato_operador))
    {
        global $numero1;
        global $numero2;
        echo "EL resultado es: ". ($numero1+$numero2);
    }
    if(!strcmp("Resta",$dato_operador))
    {
        global $numero1;
        global $numero2;
        echo "EL resultado es: ". ($numero1-$numero2);
    }
    if(!strcmp("Multiplicación",$dato_operador))
    {
        global $numero1;
        global $numero2;
        echo "EL resultado es: ". ($numero1*$numero2);
    }
    if(!strcmp("División",$dato_operador))
    {
        global $numero1;
        global $numero2;
        echo "EL resultado es: ". ($numero1/$numero2);
    }
    if(!strcmp("Módulo",$dato_operador))
    {
        global $numero1;
        global $numero2;
        echo "EL resultado es: ". ($numero1%$numero2);
    }
}

?>
```

El resultado sería:





Otra forma de solucionar seria enviando todas las variables como paso de parámetros, y seria de la siguiente manera:

```php
<?php
if(isset($_POST["button"]))
{
    $numero1=$_POST["num1"];
    $numero2=$_POST["num2"];
    $operacion=$_POST["operacion"];
    calcular($operacion,$numero1,$numero2);
}

function calcular($dato_operador,$numero1,$numero2)
{
    if(!strcmp("Suma",$dato_operador))
    {

        echo "EL resultado es: ". ($numero1+$numero2);
    }
    if(!strcmp("Resta",$dato_operador))
    {

        echo "EL resultado es: ". ($numero1-$numero2);
    }
    if(!strcmp("Multiplicación",$dato_operador))
    {

        echo "EL resultado es: ". ($numero1*$numero2);
    }
    if(!strcmp("División",$dato_operador))
    {

        echo "EL resultado es: ". ($numero1/$numero2);
    }
    if(!strcmp("Módulo",$dato_operador))
    {

        echo "EL resultado es: ". ($numero1%$numero2);
    }
}

?>
```

Incluir una hoja de estilos en PHP

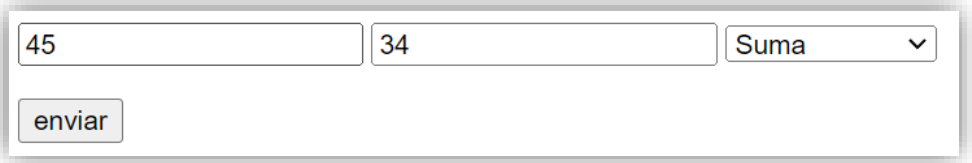
Vamos a incluir una hoja de estilos al programa.

```
<style>
    .resultado{
        color: #EB072B;
        font-weigt:bold;
        font-size: 32px;
    }

</style>
    <?php
```

Como en la clase anterior podemos hacer la llamada a la hoja de estilo cuando mostramos resultados, e incluso podemos modificar nuestro programa adicionando una variable $result, y aquí podemos efectuar la suma y podemos afectar el estilo porque el resultado lo colocamos dentro de las comillas.

```
function calcular($dato_operador,$numero1,$numero2)
{
    if(!strcmp("Suma",$dato_operador))
    {
        $result = $numero1+$numero2;
        echo "<p class='resultado'>EL resultado es: $result </p>";
    }
```

Realizando cambios, procedemos a probar la modificación.



EL resultado es: 79

El resultado mostrado con la hoja de estilo, solo funcionará para la suma, ahora usted deberá de completar para que funcione con los resultados de las demás operaciones matemáticas.

Trabajo de implementación

Implementar el ejercicio anterior utilizando librería

4.3 Operadores de incremento y decremento

Para explicar este punto, podemos usar directamente variables y ver cómo afecta su uso.

```php
<?php
$incrementa = 1;
$incrementa++;
$decrementa = 1;
$decrementa--;
echo $incrementa."<br>";
echo $decrementa."<br>";
$num = 1;
$num += $num;
echo $num;
?>
```

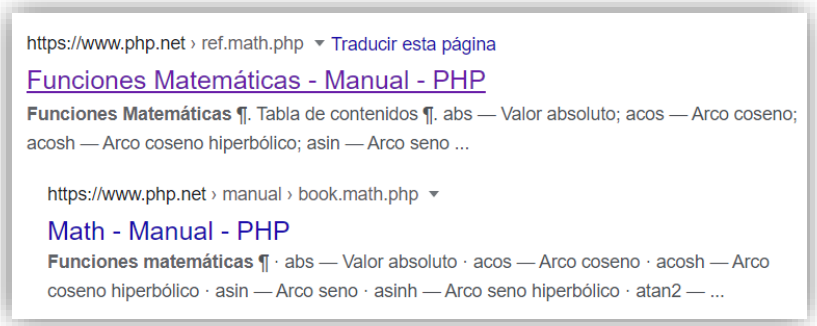
4.4 Funciones matemáticas y castings

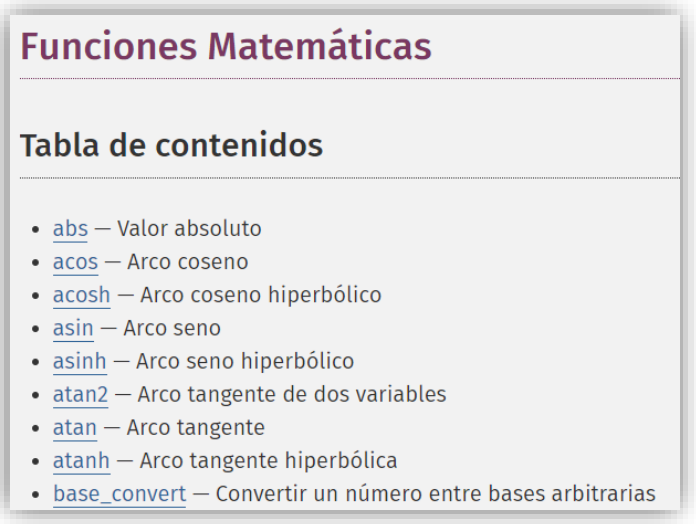
En el tema anterior hemos visto el uso de operadores aritméticos para resolver cálculos matemáticos, como sumas, restas, multiplicaciones, etc.

Con el uso de funciones matemáticas predefinidas podemos solucionar más problemas matemáticos, como hallar la potencia de un número, la raíz cuadrada de un número, etc. También hablaremos sobre casting, que es convertir un tipo dato en otro tipo de dato.

4.5 Uso de funciones matemáticas

En este tema no podremos tratar todas las funciones, lo que podemos hacer es buscar en Google como funciones matemáticas en PHP y podrán ver todas las funciones matemáticas predefinidas en PHP.





Funciones Matemáticas

Tabla de contenidos

- abs — Valor absoluto
- acos — Arco coseno
- acosh — Arco coseno hiperbólico
- asin — Arco seno
- asinh — Arco seno hiperbólico
- atan2 — Arco tangente de dos variables
- atan — Arco tangente
- atanh — Arco tangente hiperbólica
- base_convert — Convertir un número entre bases arbitrarias

Podemos ver una lista de funciones matemáticas ordenado alfabéticamente y que nos ofrece PHP. Si quieren aprender alguna función, solo deben de ingresar a esa función y estudiar su comportamiento, por ejemplo, **round** que se encarga de redondear un float.

Ejemplo: Realizar un programa que genere un número aleatorio.

```php
<?php
$numero = rand();
echo "El numero generado aleatoriamente es ". $numero;
?>
```

Resultado

```
El numero generado aleatoriamente es 1861400212
```

Según el manual que vimos en Google, vimos que había una función rand(void), donde void es vacío, y que también la función rand podía soportar dos argumentos, estos dos argumentos son valores para generar un numero entre esos números, ojo que también debe de notar que en la explicación de la página dice que esos argumentos son de tipo entero.

Ejemplo: Realizar un programa que genere un número aleatorio entre 1 y 200.

```php
<?php
$numero = rand(1,200);
echo "El numero generado aleatoriamente es ". $numero;
?>
```

Resultado

```
El numero generado aleatoriamente es 7
```

De la misma manera que usamos la función **rand()**, pueden usar cualquier otra función.

TRABAJO DE IMPLEMENTACIÓN 2

Implementar las siguientes funciones, utilizando formularios HTML con 4 opciones.

- Valor absoluto
- Obtener el valor de PI
- Obtener el exponente de un número
- Hallar la raíz cuadrada de un número

4.6 Casting

Convierte implícitamente un string a entero, y luego lo podemos convertir a flotante, y PHP lo hace sin la necesidad de estar invocando a funciones como lo hace C++.

```php
<?php
$numero = "23";
$numero += 4;
$numero += 43.456;
echo "El numero es ". $numero;
?>
```

Resultado:

```
El numero es 70.456
```

Por ejemplo: si tenemos una variable

$numero1 = “10”;

$num = $numero1;

Estaríamos diciendo que $num1 es 10 pero de tipo string.

Para solucionar ese problema, debemos de poner el tipo de dato al que queremos convertir, y a esta asignación le llamamos CASTING.

Por ejemplo:

```php
<?php
$numero = "23";
$num=(int)$numero;
?>
```

Donde PHP asume que $num es de tipo entero.

TEMA 05

CONDICIONALES: USO DEL OPERADOR IF

5.1 Operadores lógicos

Tenemos:

OPERADOR	NOMBRE
&&	Y LÓGICO
AND	Y LÓGICO
||	O LÓGICO (ALT+124)
OR	O LÓGICO
XOR	O EXCLUSIVO
!	NEGACIÓN (NOT)

5.2 Prioridad de operadores

Ejemplo: 2 + 3 * 6 = 20

Es 20 porque matemáticamente primero se debe de operar la multiplicación, es decir, el operador * tiene precedencia al operador suma. Sin embargo, con la colocación de paréntesis podemos modificar la prioridad de los operadores.

Por ejemplo: (2 + 3) * 6 = 30

Esta prioridad también ocurre con los operadores lógicos y de comparación, hay algunos que tienen prioridad respecto a otros.

Para ver, pueden escribir en Google operadores lógicos php

Ejemplo	Nombre	Resultado
$a and $b	And (y)	**true** si tanto *$a* como *$b* son **true**.
$a or $b	Or (o inclusivo)	**true** si cualquiera de *$a* o *$b* es **true**.
$a xor $b	Xor (o exclusivo)	**true** si *$a* o *$b* es **true**, pero no ambos.
! $a	Not (no)	**true** si *$a* no es **true**.
$a && $b	And (y)	**true** si tanto *$a* como *$b* son **true**.
$a || $b	Or (o inclusivo)	**true** si cualquiera de *$a* o *$b* es **true**.

5.3 Precedencia de operadores

Se trata de la prioridad de unos operadores en relación a otros, y se muestra según la siguiente tabla:

Asociatividad	Operadores	Información adicional
no asociativo	clone new	clone and new
izquierda	[	array()
derecha	**	aritmética
derecha	++ - - ~ (int) (float) (string) (array) (object) (bool) @	tipos e incremento/decremento
no asociativo	instanceof	tipos
derecha	!	lógico
izquierda	* / %	aritmética
izquierda	+ - .	aritmética y string
izquierda	<< >>	bit a bit
no asociativo	< <= > >=	comparación
no asociativo	== != === !== <> <=>	comparación
izquierda	&	bit a bit y referencias
izquierda	^	bit a bit
izquierda	|	bit a bit
izquierda	&&	lógico
izquierda	||	lógico
derecha	??	comparación
izquierda	? :	ternario
derecha	= += - = *= **= /= .= %= &= |= ^= <<= >>=	asignación
izquierda	and	lógico
izquierda	xor	lógico
izquierda	or	lógico

En esta tabla se muestra los operadores que tienen más prioridad en la parte superior y los que tienen menos prioridad se encuentran en la parte inferior. Por ejemplo, el operador && tiene una mayor prioridad que el operador AND, a pesar que ambos son Y LÓGICO. También el operador = tiene más prioridad que el AND.

Ejemplo:

Realizar un programa que analice la prioridad entre && e =, y muestre resultado, y compare con el resultado que ofrece cuando en lugar de && utiliza AND.

Establezca las diferencias y sustente su respuesta.

```php
<?php
/*Vamos a utilizar el operador y && que usualmente se utiliza en las
condicionales, es decir, el operador && (y) obliga a que se cumplan
todas las considiciones en un condicional*/
    $var1 = true;
    $var2 = false;
    $result = $var1 && $var2;
    //estamos obligando a que result evalúe a var1 y var2
    //var1 = V
    //var2 = F
    //según las tablas de verdad eso es FALSO
    // se evalúa primero el operador && y luego el =, según
    //la tabla de prioridades
    //por lo tanto result = FALSO
    if($result==true)
    {
        echo "VERDAD";
    }
    else
    {
        echo "FALSO";
    }
?>
```

Utilizando en operador AND

```php
<?php
    $var1 = true;
    $var2 = false;
    $result = $var1 AND $var2;
    //en este caso el operador = tiene mayor precedencia que AND
    //por lo tanto, primero ejecutará el operador =
    //entonces result sera igual a var1, es decir result = true
    //imprimira VERDAD
    if($result==true)
    {
        echo "VERDAD";
    }
    else
    {
        echo "FALSO";
    }
?>
```

Para solucionar el problema anterior podemos añadir los paréntesis, encerrando var1 y var2, entonces se ejecuta primero lo de los paréntesis.

```php
<?php
    $var1 = true;
    $var2 = false;
    $result = ($var1 AND $var2);
    //en este caso el operador hemos añadido paréntesis
    //por lo tanto, primero ejecutará todo dentro del paréntesis
    //y despues ese resultado se asigna a la variable result
    //imprimira FALSO
    if($result==true)
    {
        echo "VERDAD";
    }
    else
    {
        echo "FALSO";
    }
?>
```

5.4 Ejemplo del uso de condicionales

A continuación, tenemos un formulario con el atributo action, y lo que hace es llamar a un archivo externo php, el cual se llama validacion_cond.php. Es decir, nuestro programa consta de dos archivos: uso_condicionales.php y validacion_cond.php

El archivo de validacion_cond.php no tiene nada de especial, salvo una hoja de estilos, y una zona de php vacía.

El formulario se verá de la siguiente manera:

USO DE LAS CONDICIONALES

Lo que haremos será introducir una edad y dependiendo de la edad ingresada el servidor nos debe de dar diferentes respuestas, por ejemplo, si introducimos 15, este 15 irá al servidor y el servidor nos dará como respuesta que es **menor** de edad, pero si introducimos 18 nos devolverá **joven**, si introducimos 45 nos dirá **maduro.**

Ahora la solución sería:

Archivo uso_condicionales.php

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Documento sin título</title>
	<style>
	h1{
		text-align:center;
	}
	table{
		background-color:#E18B8C;
		padding:5px;
		border:#666 5px solid;
	}
	.no_validado{
		font-size:18px;
		color:#F00;
		font-weight:bold;
	}
	.validado{
		font-size:18px;
```

```
                color:#0C3;
                font-weight:bold;
        }
</style>
</head>
<body>
<h1>USO DE LAS CONDICIONALES</h1>
        <form action="validacion_cond.php" method="post" name="datos_usuario" id="datos_usuario">
        <table width="15%" align="center">
   <tr>
     <td>Nombre:</td>
     <td><label for="nombre_usuario"></label>
     <input type="text" name="nombre_usuario" id="nombre_usuario"></td>
   </tr>
   <tr>
     <td>Edad:</td>
     <td><label for="edad_usuario"></label>
     <input type="text" name="edad_usuario" id="edad_usuario"></td>
   </tr>
   <tr>
     <td> </td>
     <td> </td>
   </tr>
   <tr>
     <td colspan="2" align="center"><input type="submit" name="enviando" id="enviando" value="Enviar"></td>
   </tr>
        </table>
</form>
</body>
</html>
</body>
```

```
</html>
```

Archivo validacion_cond.php

Nos dirigimos al archivo validacion_cond.php que es lo que el formulario está llamando, en ese archivo debemos programar en php, y decirle al programa de que, si el usuario pulsa el botón enviar, el programa debe evaluar lo que hay en el cuadro de texto de la edad.

```
<?php
        if(isset($_POST["enviando"]))
        //se verifica si el usuario pulso el botón de nombre enviando.
        {
         $edad = $_POST["edad_usuario"];
        //en edad almacenamos lo que el usuario a introducido en el cuadro de texto edad
        //evaluamos lo introducido.
                if($edad<=18)
                {  echo "eres menor de edad";
                }else if($edad<=40)
                        {  echo "eres joven";
                        } else if ($edad<=65)
                                { echo "eres maduro";
                                }else { echo "cuidate";
                                        }
        }
?>
```

El uso del else if, dice que todo forma un bloque, es decir, están relacionados unos con otros. Y el ultimo else, esta relacionado con el if anterior, y también con el anterior y con el de más adelante (con los 3 if)

```php
if($edad<=18)
{
    echo "eres menor de edad";
}else if($edad<=40)
    {
        echo "eres joven";
    }else if ($edad<=65)
        {
            echo "eres maduro";
        }else
            {
            echo "cuídate";
            }
```

Si no utilizamos el else, y si usamos puro if, lo tomará cada bloque de manera independiente.

5.5 Operador ternario

Sirve para construir estructuras condicionales sencillas (ojo solo utilizar cuando es una estructura sencilla), tienen que devolver un único valor si esta estructura se cumple y otro valor diferente se esa condición no se cumple. No es muy habitual ver este operador.

5.5.1 Sintaxis del operador ternario.

Condición ? Valor si verdadero: Valor si Falso

Ejemplo con el caso anterior.

```php
<?php
    $var1 = true;
    $var2 = false;
    $result = ($var1 AND $var2);
    echo $result == true ? "VERDAD" : "FALSO";
?>
```

El resultado mostrado será FALSO.

Este tipo de implementaciones reduce la cantidad de código fuente.

Otra forma de resolver esta operación es almacenando la expresión en una variable y luego mostramos el resultado de la variable.

```php
<?php
    $var1 = true;
    $var2 = false;
    $result = ($var1 AND $var2);
    $dato = $result == true ? "VERDAD" : "FALSO";
    echo $dato;
?>
```

Si necesitamos evaluar más de una variable o condición, podemos usar dos operadores lógicos.

```php
<?php
    $num1 = 5;
    $num2 = 10;
    $dato = $num1>0 && $num2<20 ? "LOS NUMEROS ESTAN ENTRE 0 Y 20" : "LOS
    NUMEROS NO CUMPLEN CON LA CONDICION";
    echo $dato;
?>
```

TEMA 06

ESTRUCTURAS CONDICIONALES
USO DEL SWITCH

6.1 SWITCH - CASE

Esta estructura es de uso en casi todos los lenguajes de programación, tales como C, C++, Java, Java script, visual Basic, etc. Se utiliza cuando debemos evaluar muchas condiciones, en este caso utilizamos el switch case, aunque también puedes utilizar puro if.

6.1.1 Sintaxis.

En PHP se maneja dos sintaxis y la diferencia entre estos es el uso de las llaves. Ambas sintaxis son válidas en PHP.

Si bien el CASE funciona como varios IF, y si quisiéramos poner un ELSE, utilizaríamos el DEFAULT que es como si estuviéramos escribiendo ELSE, entonces se ejecuta todo el código que este dentro de DEFAULT siempre y cuando nada de lo anterior se haya cumplido.

Switch(Condición a evaluar)
{ Case **valor 1:**
//Código;
Break;
Case **valor 2:**
//Código;
Break;
Case **valor 3:**
//Código;
Break;
Default:
}
	Switch(Condición a evaluar):
	Case **valor 1:**
	//Código;
	Break;
	Case **valor 2:**
	//Código;
	Break;
	Case **valor 3:**
	// Código;
	Break;
	Default:
	Endswitch

Ejercicio 1: Realizar un programa que permita comparar if anidado con la herramienta CASE.

```php
<html>
<head>
<meta charset="utf-8">
<title>Documento sin título</title>
</head>

<body>
    <?php
    $var = 2;
    if($var==1)
    {
        echo "var es igual a 1";
    }else if($var==2){
        echo "var es igual a 2";
    }else if($var==3){
        echo "la variable es igual a 3";
    }else{
        echo "var no es 1, ni 2 ni 3";
    }
    ?>
</body>
</html>
```

```php
<html>
<head>
<meta charset="utf-8">
<title>Documento sin título</title>
</head>

<body>
    <?php
    $var = 2;
    switch($var){
        case 1: echo "var es 1";
                break;
        case 2: echo "var es 2";
                break;
        case 3: echo "var es 3";
                break;
        default: echo "var no es 1, 2 ni 3";
    }
    ?>
</body>
</html>
```

El resultado en ambos casos es: $var es igual a 2.

Si cambiamos $var = 2 por $var = 5 al inicio del código, el resultado sería: $var no es igual a 1, 2 o 3.

Una diferencia importante respecto a las instrucciones if ... else if ... else es que se requiere de la instrucción break para salir del switch al terminar las instrucciones correspondientes a un case. Sólo podemos especificar un valor en cada case, no se admite indicar más de un valor. Sin embargo, si dejamos un case en blanco y omitimos el break, damos lugar a que se ejecuten ciertas instrucciones si el valor coincide con alguno de los case en juego. Esto lo veremos más claro con un ejemplo.

Ejercicio 2.

```php
<html>
<head>
<meta charset="utf-8">
<title>Documento sin título</title>
</head>

<body>
    <?php
    $var = 2;
    if($var==1)
    {
        echo "var es igual a 1";
    }else if($var==2 OR $var==3){
        echo "var es 2 o 3";
    }else{
        echo "var no es 1, ni 2 ni 3";
    }
    ?>
</body>
</html>
```

```php
<html>
<head>
<meta charset="utf-8">
<title>Documento sin título</title>
</head>

<body>
    <?php
    $var = 2;
    switch($var){
        case 1: echo "var es 1";
                break;
        case 2:
        case 3: echo "var es 2 o 3";
                break;
        default: echo "var no es 1, 2 ni 3";
    }
    ?>
</body>
</html>
```

En cualquiera de las dos formas mostradas, la salida obtenida sería: $var es 2 o 3.

En este caso, al evaluarse la segunda instrucción CASE, no se encuentra ningún break por lo que se siguen ejecutando las instrucciones siguientes al tercer case, aun cuando $var sea diferente de 3, ya que al no existir un break anterior los casos 2 y 3 quedan “agrupados”. Más aún, si borramos el break del tercer case también se ejecutarán las instrucciones siguientes a default, y la salida sería entonces:

$var es 2 o 3.

$var no es 1, 2 ni 3.

Por ello es muy importante que cuando escribas una instrucción switch pongas break en todos los case, o bien que si dejas algún break sin escribir sea porque conscientemente quieras hacerlo y no por olvido.

TEMA 07

BUCLE WHILE Y DO WHILE

7.1 Definición de bucles

Los bucles son estructuras repetitivas, es decir, repiten una y otra vez el código que está en su interior. En ocasiones necesitamos que una o varias líneas de código se repitan. La solución sería meter en un bucle las líneas de código que queremos que se repitan.

7.2 Tipos de bucles

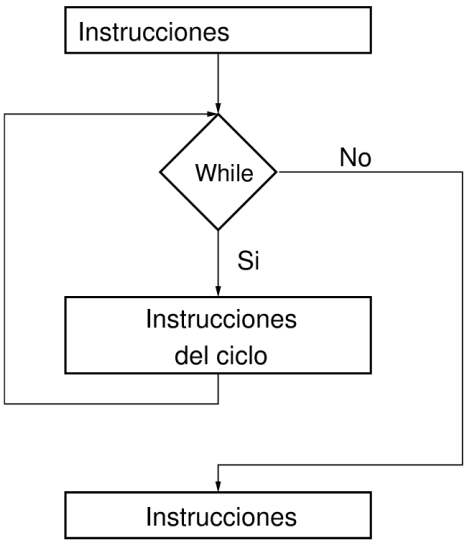
Bucles indeterminados. Son aquellos en los que no sabemos antes de ejecutar el programa, cuantas veces deberá repetirse el código que hay en el interior, dependerá de las circunstancias.

Como bucles indeterminados tenemos: While y do while

Bucles determinados. Son aquellos que sabemos que incluso antes de ejecutar el programa, cuantas veces repetirán el código que hay en su interior.

Tenemos al bucle for, y es un bucle que tienen todos los lenguajes de programación.

7.3 Flujo de ejecución del while



El esquema anterior explica el flujo de ejecución del while, cuando el flujo de ejecución llegue al **while,** se evaluará una condición, esa condición puede ser verdadera o puede ser falta, es como la condición **if,** y si la condición es verdad, entonces entrará en el flujo de ejecución dentro del bucle, y realizará el código que hay en el interior, y lo realizará mientras la condición sea verdad, (while = mientras), llegara un momento en que la condición pase a ser **falsa,** entonces la ejecución se sale del **while.**

Y si la condición del while es falsa, nunca entrará en el bucle.

7.4 Sintaxis en código while.

El código se va ejecutar las veces necesarias hasta que $var1 sea menor que 6,

```php
<?php
$var1 = 1;
while($var1<6)
{
    echo "se ejecuta el bucle "."$var1". "<br>";
    $var1++;
}
echo "hemos salido del bucle";
?>
```

Resultado

```
se ejecuta el bucle 1
se ejecuta el bucle 2
se ejecuta el bucle 3
se ejecuta el bucle 4
se ejecuta el bucle 5
hemos salido del bucle
```

Si en $var1 ponemos que es igual a 7, entonces nunca va entrar en el while, por lo que imprimiría **hemos salido del bucle.**

7.5 Bucle infinito

Debemos de tener cuidado de no crear bucles infinitos, un bucle infinito es cuando la condición del while nunca pasa a ser falsa. Por ejemplo, si dentro del bucle while no incrementamos la variable $var1, esto da que cuando el programa entra dentro del while, la condición es cierta, y como no vamos a modificar la variable $var1, la condición

siempre será verdadera y siempre entrará dentro del while, por lo que ahí estaríamos creando un bucle infinito. (Probar el cambio y comente)

7.6 Bucle do – while

Funciona prácticamente igual, do = hacer, while = mientras, y la única diferencia es que el bucle do while ejecuta la instrucción, aunque la condición sea falsa.

Analice el siguiente código:

```php
<?php
$var1 = 7;
do
{
    echo "se ejecuta el bucle "."$var1". "<br>";
    $var1++;
}while($var1<6);

echo "hemos salido del bucle";
?>
```

Resultado

```
se ejecuta el bucle 7
hemos salido del bucle
```

si cambiamos el valor de la variable $var1 = 1, entonces funcionará como en el while.

PRÁCTICA DIRIGIDA

1. Realizar un programa que realice la tabla de multiplicar del 5 mediante el while y do while.

```php
<?php
$numero = 1;
while($numero<=10)
{
    $result = $numero * 5;
    echo "5 * $numero = $result <br>";
    $numero++;
}

?>
```

5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50

```php
<?php
$numero = 1;
do
{
    $result = $numero * 5;
    echo "5 * $numero = $result <br>";
    $numero++;
}while($numero<=10);

?>
```

5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50

2. Realizar un programa que permita sumar números del 1 al 2000 de 5 en 5.

```php
<?php
$num = 1;
$suma = 0;
while($num<=2000)
{
    $suma = $suma + $num;
    $num = $num + 5;
}
echo "$suma";
?>
```

399400

3. Realizar un programa mediante el uso de while o do while, el programa debe hallar el promedio de 5 estudiantes y cada estudiante debe tener 3 notas de 0 a 20, generados aleatoriamente, promedio de exámenes tienen un porcentaje de 60%, promedio de trabajos encargados tienen un porcentaje de 30% y actitudinal que tiene un peso del 10%. El programa solo debe visualizar las 5 notas de los estudiantes.

```php
<?php
$num_est = 1;
while($num_est<=5)
{
    $prom_exam = rand(0,20);
    $prom_trab = rand(0,20);
    $prom_act = rand(0,20);

    echo "las notas obtenidas por el estudiante $num_est fueron: <br>";

    echo "Promedio de examenes = $prom_exam<br>";
    echo "Promedio de trabajos = $prom_trab<br>";
    echo "Promedio actitudinal = $prom_act<br>";

    $prom_final = $prom_exam*0.6 + $prom_trab*0.3 + $prom_act*0.1;
    echo "El promedio final es $prom_final<br><br>";
    $num_est++;
}

?>
```

las notas obtenidas por el estudiante 1 fueron:
Promedio de examenes = 15
Promedio de trabajos = 8
Promedio actitudinal = 5
El promedio final es 11.9

las notas obtenidas por el estudiante 2 fueron:
Promedio de examenes = 15
Promedio de trabajos = 15
Promedio actitudinal = 12
El promedio final es 14.7

las notas obtenidas por el estudiante 3 fueron:
Promedio de examenes = 9
Promedio de trabajos = 19
Promedio actitudinal = 5
El promedio final es 11.6

las notas obtenidas por el estudiante 4 fueron:
Promedio de examenes = 12
Promedio de trabajos = 4
Promedio actitudinal = 2
El promedio final es 8.6

las notas obtenidas por el estudiante 5 fueron:
Promedio de examenes = 15
Promedio de trabajos = 0
Promedio actitudinal = 20
El promedio final es 11

TEMA 08

FOR

8.1 Bucle for

El bucle FOR es un bucle determinado, porque sabemos desde un inicio que antes de ejecutar el programa se repetirá el código que hay en su interior, mientras que con los otros bucles indeterminados a veces lo sabemos, pero otras veces no.

8.2 Sintaxis del bucle for

```
For (Inicio de bucle; condición del bucle; incremento o decremento de bucle)
{
}
```

Este bucle es difícil de entender, sobre todo cuando alguien se inicia en la programación.

Inicio de bucle indica el punto en el que partimos para ejecutar el bucle.

Condición de bucle indica la condición que tiene que cumplir el bucle for (funciona como un condicional IF, esta condición será verdad, pero llegará un momento en el que dejará de ser verdad para pasar a ser falsa, y en el momento que la condición es falsa el ciclo del bucle termina).

Incremento o decremento de bucle nos permite decirle al bucle el como se va incrementar o cómo se va decrementar para que repita el código del interior del bucle.

Al finalizar el bucle FOR, no finaliza con punto y coma, mas bien debe tener una llave de inicio y cierre { }

8.3 Sentencias de interrupción de bucle

Son sentencias que nos permiten salir de un bucle antes de que éste finalice.

Ejemplo: Realizar un programa que imprima “Estamos dentro del bucle for”, una cantidad de 10 veces.

```php
<?php
for($i=1; $i<=10; $i++)
{
    echo "estamos dentro del bucle for<br>";
}

?>
```

estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for

Recordemos que el flujo de un programa es de arriba hacia abajo, lo primero que hacemos es declarar una variable i, donde i es igual a 0, luego el programa toma nota y almacena en memoria en i = 0, el segundo bucle del programa es lo que llamamos condición, y funciona de similar forma que el condicional if. En el código de arriba le estamos diciendo que ingrese al FOR hasta que i sea menor o igual que 10, si esa condición es verdad, entonces baja a ejecutar la instrucción de abajo, mas no se ejecuta aun el incremento del bucle, únicamente después de haberse ejecutado el código del interior del FOR, recién se incrementa el i, ahora el i se actualiza y se vuelve a preguntar si i es menor o igual que 10, se vuelve a ejecutar las instrucciones de abajo, luego se incremente el i, y así sucesivamente hasta que ya no se cumpla la condición o que i sea igual a 11 (para este caso), entonces saldría del bucle for.

El ejercicio anterior podemos resolverlo de otra manera, con decremento, la cual muestro a continuación. Además, realice usted el análisis de ese algoritmo.

```php
<?php
for($i=10; $i>=1; $i--)
{
    echo "estamos dentro del bucle for<br>";
}

?>
```

estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for

Para continuar conociendo el FOR, otra forma de resolver el mismo ejercicio, podría ser la siguiente estructura, y realice su análisis mediante una prueba de escritorio.

```php
<?php
for($i=0; $i<20; $i+=2)
{
    echo "estamos dentro del bucle for<br>";
}

?>
```

estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for

8.4 Instrucción break

Qué tendríamos que hacer si necesitamos que se salga del FOR cuando el código muestre 4 veces el texto “estamos dentro del bucle for”, luego el programa debe mostrar el mensaje “Bucle interrumpido”, y se debe salir del bucle sin llegar a culminar con que i sea menor o igual a 10.

Este problema se soluciona utilizando la sentencia **break,** y lo hacemos a continuación:

```php
<?php
for($i=1; $i<=10; $i++)
{
    echo "estamos dentro del bucle for<br>";
    if($i==4)
    {
        echo "Bucle interrumpido";
        break;
    }
}
```

estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
estamos dentro del bucle for
Bucle interrumpido

Realizar la prueba de escritorio del programa anterior.

8.5 Instrucción continue

Esta instrucción no hace que se salga del bucle, sino que impide que una vuelta de bucle no se ejecute.

Ejemplo: Realizar un programa que imprima la tabla del 6.

A continuación, mostramos el programa de la tabla de multiplicar, pero sin la instrucción continue, es decir, se va ejecutar todo el for.

```php
<?php
for($i=0; $i<=10; $i++)
{
    echo "6 x $i = " . 6*$i . "<br>";
}
echo "fuera del bucle";

?>
```

```
6 x 0 = 0
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
fuera del bucle
```

Ejemplo: Realice un cambio de tal manera que en el programa anterior no se ejecute únicamente cuando tenga que multiplicar con 5, es decir 6 x 5 = 30 no se debe mostrar, pero si las demás instrucciones.

```php
<?php
for($i=0; $i<=10; $i++)
{
    if($i==5)
    {
        continue;
    }
    echo "6 x $i = " . 6*$i . "<br>";
}
echo "fuera del bucle";

?>
```

```
6 x 0 = 0
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
fuera del bucle
```

Podemos apreciar que cuando i es igual a 5, el programa no ejecuta, pero salta a la siguiente instrucción.

Ejemplo: Realizar la tabla de dividir del 8 y hacer que se salte la división entre 0. Y que envíe un mensaje de que no existe la división entre cero.

```php
<?php
for($i=0; $i<=10; $i++)
{
    if($i==0)
    {
        echo "No existe la division entre cero <br>";
        continue;
    }
    echo "8 / $i = " . 8/$i . "<br>";
}
echo "fin del bucle";

?>
```

No existe la division entre cero
8 / 1 = 8
8 / 2 = 4
8 / 3 = 2.6666666666667
8 / 4 = 2
8 / 5 = 1.6
8 / 6 = 1.3333333333333
8 / 7 = 1.1428571428571
8 / 8 = 1
8 / 9 = 0.88888888888889
8 / 10 = 0.8
fin del bucle

TEMA 09

FUNCIONES EN PHP

9.1 Funciones

Las funciones permiten automatizan tareas, esto es parecido a las funciones que utilizamos en Excel, estas tareas son tareas repetitivas, esto nos va a permitir eliminar código repetitivo en el programa.

Por ejemplo, si tenemos una tarea de realizar limpieza o aseo personal todos los días, se puede crear una función limpieza( ), todo el proceso está dentro de la función, por lo que, después solo llamaríamos a la función para que se ejecute la tarea.

9.2 Tipos de funciones

9.2.1 Funciones predefinidas.

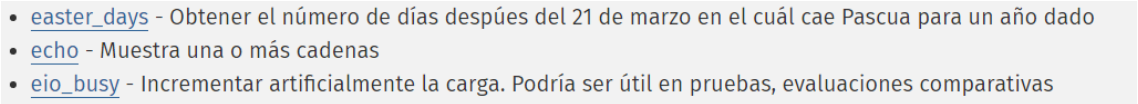
Estas funciones vienen con el lenguaje de programación PHP, estas funciones están disponibles para que podemos utilizarla. Por ejemplo, la función echo, o las funciones matemáticas, etc.

Las funciones predefinidas las podemos ver en (o podemos escribir en una ventana de Google *listado de funciones en php* ***nos va mostrar el vínculo de las funciones***):

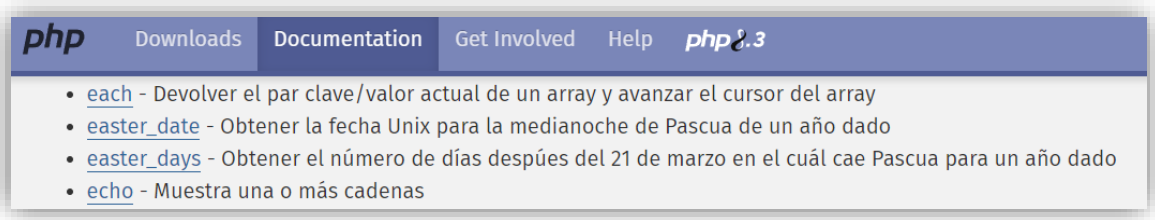
https://www.php.net/manual/es/indexes.functions.php

Si hacemos clic en la letra e, la página nos va mostrar todas las funciones que inician con la letra e.

Podemos ver la función echo.



Si pulsamos en la función echo, la página web nos va mostrar las definiciones y modo de uso de la función, incluido los parámetros y la sintaxis de la función.



Si hacemos clic en **echo** la pagina web nos mostrara una información mas ampliada de la función **echo,** donde muestra la sintaxis con los argumentos que puede recibir la función. Los argumentos también son denominados parámetros de una función y va dentro de los paréntesis. Hay funciones predefinidas que no reciben ningún argumento y hay otras funciones que si tienen parámetros.

Estos argumentos son como una especie de variables que utiliza la función para almacenar valores y las utiliza en la tarea que tiene que realizar.

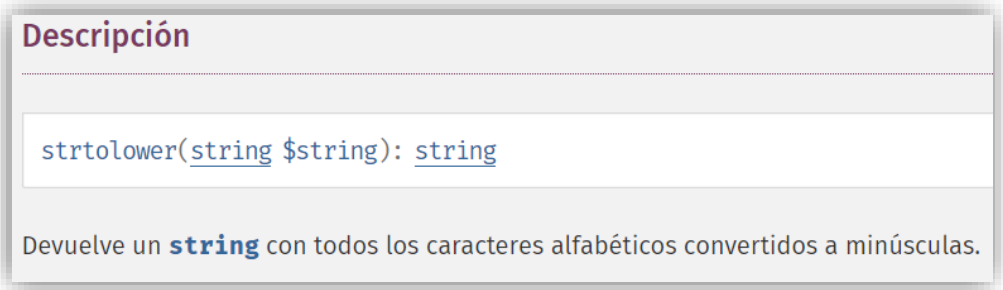
Además de darnos la sintaxis, la página web nos da una descripción de que es lo que hace una función. También más abajo podremos ver varios ejemplos de cómo utilizar dicha función.

Vamos a ver unos ejemplos de cómo es el uso de algunas funciones predefinidas.

Vamos a utilizar dos funciones que sirven para cambiar de mayúsculas a minúsculas y viceversa, y también veremos otras funciones relacionadas.

Ejemplo de uso de funciones pre definidas.

Strtolower Esta función convierte una cadena de texto a minúsculas.
Strtoupper Esta función convierte una cadena de texto a mayúsculas.
Ucwords Convierte de una frase la primera letra de todas las palabras en mayúscula.



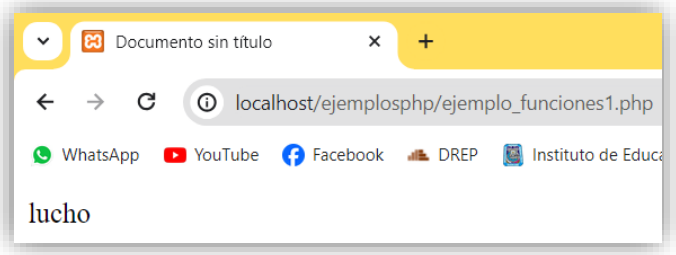
Problema. Realizar un programa que permita cambiar un texto de mayúsculas a minúsculas.

Solución

Vamos a crear nuestra zona php

```php
<?php
    $dato = "LUCHO";
    echo(strtolower($dato));

?>
```

Se imprime en pantalla el siguiente resultado:



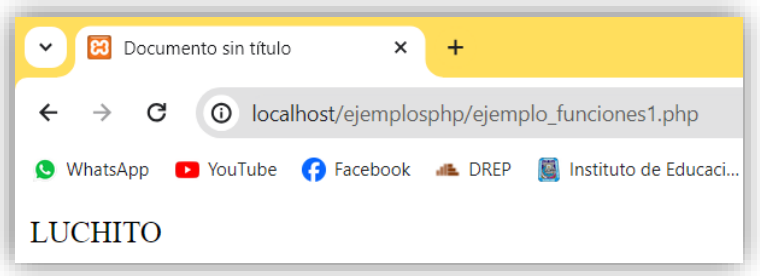
Como puede ver, ya ha sido convertido la cadena en minúscula

Problema. Realizar un programa que permita cambiar un texto de minúsculas a mayúsculas.

Solución.

```php
<?php
    $dato = "luchito";
    echo(strtoupper($dato));

?>
```

Resultado:



9.2.2 Funciones propias

Son las que creamos nosotros como programadores con el objetivo de poder reutilizarlas en un futuro para que no tengamos que escribir una y otra vez el mismo código.

Para trabajar con funciones tenemos que utilizar la palabra reservada **function** luego debemos escribir el nombre de la función, en este caso le vamos a llamar **suma** y luego asignamos paréntesis, y dentro de este paréntesis irían lo que son los argumentos, (puede ser que la función que estamos construyendo, no tenga la necesidad de tener argumentos, en cuyo caso los paréntesis quedarían vacíos, una función puede tener uno o más de un argumento)

9.3 Parámetro de funciones

En este caso la función que vamos a crear va recibir dos argumentos y deben ir separados por comas. Y dentro de la función se debe poner el cálculo que debe hacer.

En esta ocasión vamos a utilizar la función **return,** que nos va permitir que la función devuelva un valor, en este caso nos devolverá lo que contiene la variable **$resultado.**

Ejemplo: Realizar un programa que, dados dos números, y mediante una función se realice la suma de ambos números dados.

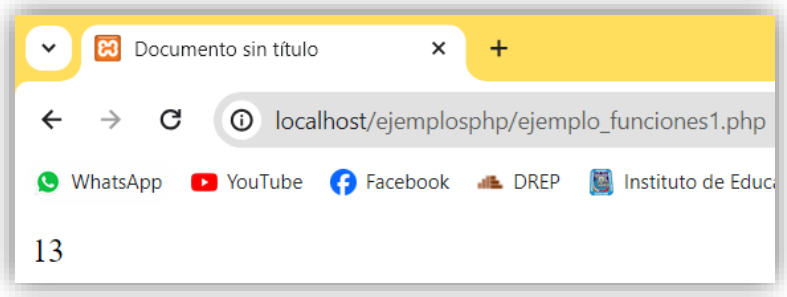
Solución:

Cuando hacemos la llamada a la función, la maquina se asimila a lo que dice la función, por ejemplo, hemos llamado a la función con **suma(5, 8)** en donde $num1 se asume como 5 y $num2 se asume como 8. Luego en $resultado se almacena 13.

Líneas más abajo con la función echo estamos imprimiendo el valor de la suma, justamente el valor que retorna la función suma.

Es decir, el resultado dependerá de los datos que ingresemos como argumentos a la función, ojo que, si no llamamos a la función, nunca se ejecutará la función.

```php
<?php
    function suma($num1, $num2)
    {
        $resultado = $num1+$num2;
        return $resultado;
    }
    $valor = suma(5,8);
    echo $valor;
?>
```



En cuanto al paso de parámetros php tiene el paso de un parámetro por defecto (no todos lo tienen)

Ejemplo: Realizar un programa que se encargue de pasar de mayúsculas a minúsculas y que además también sea capaz de convertir la primera letra de las palabras de una frase en mayúscula.

Solución

Para esto vamos a crear una función propia (para nombrar a una función seguimos las mis reglas que para definir una variable).

Voy a llamar a esta función como **palabras_mayus_pri,** y esta función va a recibir dos argumentos, el primero lo llamare **$palabras** y el segundo argumento será el argumento por defecto al cual le llamare **$convertir** el cual a este parámetro le diré que por defecto sea **true**

Le vamos a decir a la función que almacene en $palabras, precisamente la variable $palabras, obviamente con el uso de la función **Strtolower**

Luego mediante if, preguntamos si el argumento $convertir es igual a **true,** si es cierto, que en una variable $resultado poner la primera letra de la frase en Mayúscula, y para ello usaremos la función **ucwords,** esta función nos permitirá poner la primera letra de toda la frase, en mayúscula.

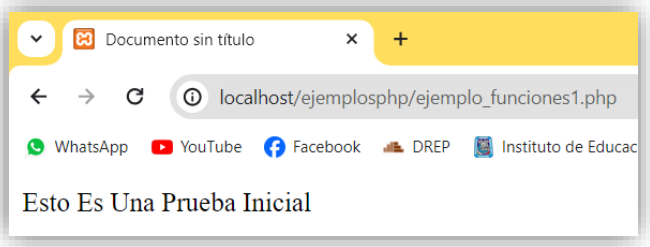
Luego, decimos que, si no es cierto la condición anterior, debe almacenar en la variable $resultado las palabras en mayúsculas.

Y por último nos debe de devolver con **return** el resultado.

Luego debemos de hacer el llamado a la función.

```php
<?php
    function palabras_mayus_pri($palabras, $convertir = true)
    {
        $palabras = strtolower($palabras);
        if($convertir==true)
        {
            $resultado=ucwords($palabras);
        }
        else{
            $resultado=strtoupper($palabras);
        }
        return $resultado;
    }
    echo palabras_mayus_pri("esto es una prueba inicial");
?>
```

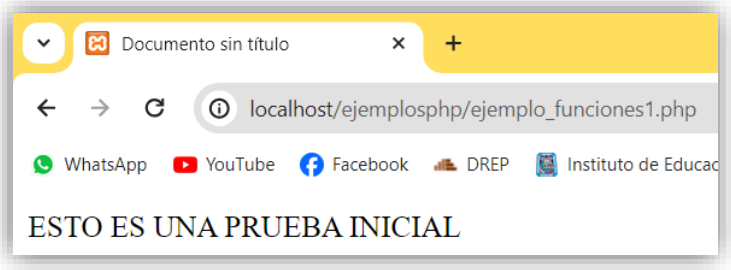
Fíjense que en la llamada a la función solo hemos hecho la llamada con un argumento, la cadena viaja y se almacena en la variable $palabras, pero no le hemos pasado el segundo argumento. Entonces lo que hace la función es tomar el valor por defecto. Lo que hará el programa es entrar en el if, ya que cumple con la condición de la función. Entonces nos pondrá la primera letra de la frase en mayúsculas.



Ahora, si la pasamos un segundo argumento a la función, como **false,** en ese caso se a cambiado el valor por defecto que era **true,** entonces ahora la llamada la hará al **else.**

```php
<?php
    function palabras_mayus_pri($palabras, $convertir = true)
    {
        $palabras = strtolower($palabras);
        if($convertir==true)
        {
            $resultado=ucwords($palabras);
        }
        else{
            $resultado=strtoupper($palabras);
        }
        return $resultado;
    }
    echo palabras_mayus_pri("esto es una prueba inicial",false);
?>
```

Resultado



9.4 Parámetros por valor y por referencia

Cuando hacemos un paso por valor, no tiene nada de especial, solamente pasamos como argumento un valor, y en este caso ese valor lo estamos incrementando en 1.

Sin embargo, en el paso de parámetro por referencia, a la hora de declarar el parámetro, podemos aumentar el símbolo & (ampersand) delante del parámetro declarado. Esto implica que cuando llamemos a esta función, ese parámetro declarado será pasado por referencia.

Paso de parámetro por valor

```
Function ejemplo($parametro)
{
        $parametro++;
}
```

Paso de parámetro por referencia

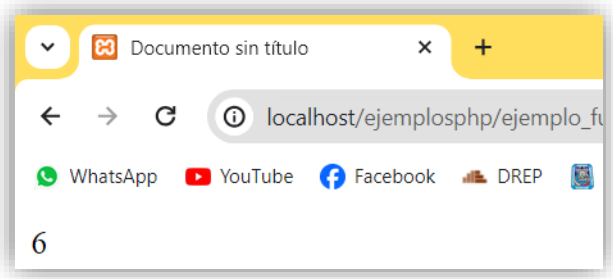
```
Function ejemplo(&$parametro)
{
        $parametro++;
}
```

¿Qué implica pasar un parámetro por valor?

Ejemplo: vamos a declarar una función incrementa, con un parámetro $valor, y lo que vamos a hacer es incrementar el valor de ese argumento en 1, y para terminar le vamos a decir a la función que retorne el parametro $valor.

Luego vamos a imprimir en pantalla lo que devuelve la función incrementa.

```php
<?php
    function incrementa($valor)
    {
        $valor++;
        return $valor;
    }
    echo incrementa(5);
?>
```

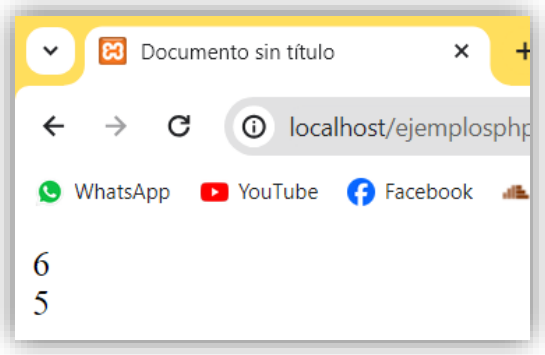


Otra forma de resolver. (el valor de una variable lo podemos pasar a la función, nos dará el mismo resultado)

```php
<?php
    function incrementa($valor)
    {
        $valor++;
        return $valor;
    }
    $num = 5;
    echo incrementa($num);
?>
```

Ahora vamos a concatenar un salto de línea, y después de llamar a la función, lo que haremos será imprimir el valor de $num.

```php
<?php
    function incrementa($valor)
    {
        $valor++;
        return $valor;
    }
    $num = 5;
    echo incrementa($num)."<br>";
    echo $num;
?>
```



Interpretación

Lo que estamos diciendo es que el argumento que recibe es por valor, eso quiere decir que lo que hay almacenado en $num, ósea 5, viaja y se almacena en $valor, pero se pierde toda conexión entre lo que sería la función y lo que hay después, es decir, ese valor 5 esta encapsulado dentro de la función, y la función no sabe nada de lo que hay fuera de ella, es decir, la función no sabe lo que es la variable $num, es decir, recibe el 5, lo incrementa en 1, se convierte en 6 y luego nos devuelve el nuevo $valor.)

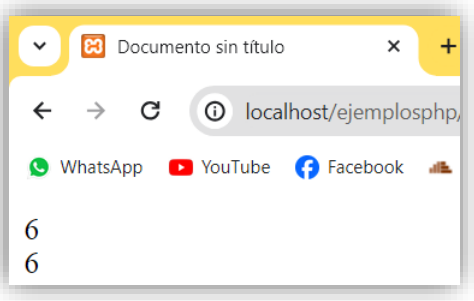
¿Qué implica pasar un parámetro por referencia?

Ahora teniendo una nueva versión del código, vamos a aumentar el & delante del parámetro $valor, (lo que estamos haciendo es que estamos cambiando por completo el sentido del programa)

Cuando incluimos el &, lo que estamos haciendo es crear una conexión o referencia con el origen, que en este caso es $num, existe una especie de vinculo, que es la referencia, y eso quiere decir que la función si sabe que es lo que hay fuera de ella, con lo cual, al incluir el &, el valor de $num viaja y se almacena en $valor, pero ahora la clave

está en $valor++, cuando incrementamos en 1, lo que hay almacenado en $valor, $valor pasa a ser 6, pero como había un vinculo con el origen ósea con $num, también se incrementa $num, porque está vinculado, porque hay una referencia, por lo que en este caso, lo que va imprimir en pantalla será dos veces el seis.

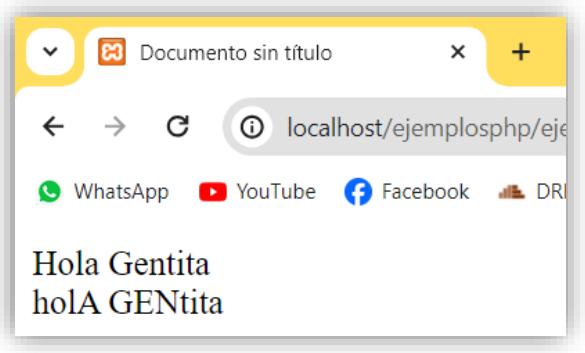
```php
<?php
    function incrementa(&$valor)
    {
        $valor++;
        return $valor;
    }
    $num = 5;
    echo incrementa($num)."<br>";
    echo $num;
?>
```



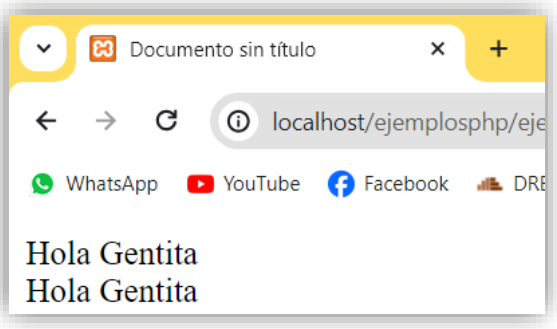
Bien, lo que ahora debemos ver es cual es la utilidad de este suceso, es decir, casos en los que una función modifique valores que están fuera de su función, concretamente valores pasados por parametro.

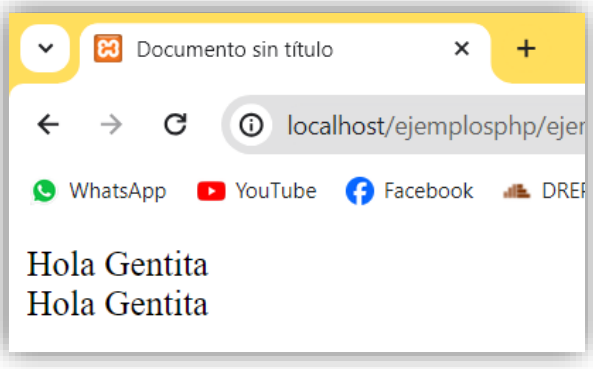
Ejemplo: Crear una función que modifica las mayúsculas a minúsculas de el parametro que le pasemos.

```php
<?php
    function cambia_mayuscula($parametro)
    {
        $parametro=strtolower($parametro);
        $parametro=ucwords($parametro);

        return $parametro;
    }
    $cadena = "holA GENtita"; //texto con anormalidades
    echo cambia_mayuscula($cadena)."<br>";
    echo $cadena;
?>
```



Ahora que sucede si trabajamos con el paso de valor por referencia.





Pregunta: Explique este suceso.

TEMA 10

BASE DE DATOS CON PHP

10.1 Definición de MySQL

MySql es un gestor de base de datos relacional, multihilo y multiusuario.

10.2 Base de datos relacional.

Una base de datos relacional es cuando la base de datos en donde la información está relacionada, es decir, una base de datos relacional está formada por varias tablas en donde la información está directamente relacionada entre sí.

Si vemos una tabla clientes está relacionada con la tabla pedidos, como por ejemplo para conocer la lista de clientes que habían hecho ciertos pedidos.

10.2.1 Multihilo

Multihilo quiere decir que la base de datos puede soportar varios procesos al mismo tiempo, es decir, nosotros podremos realizar varias peticiones de información a la base de datos de manera simultánea.

10.2.2 Multiusuario

Quere decir que en una base de datos pueden conectarse varios usuarios simultáneamente o trabajando la base de datos, o pidiendo información o modificando información en la base de datos. Con esto queremos decir que cada usuario tendrá sus usuario y contraseña y sus permisos de acceso que puede ser diferente en cada caso.

Una característica especial en MySql es LIBRE, esto implica que no tendremos que pagar por el uso del gestor, es decir, es gratuito, pero si vamos a utilizarlo en una empresa deberemos de pedir una licencia. (ver en la página web oficial del MySql las demás bondades).

Otra característica es que se puede descargar el código fuente del MySql, y podemos modificarlo a nuestro gusto, claro que para este detalle necesitamos conocimientos avanzados de programación.

10.3 Servidores MySQL

10.3.1 Servidor local.

Cuando hemos instalado el AppServ u otro paquete de servidores locales, lo que se hizo fue que hemos instalado en la computadora nuestros propios gestores de bases de datos MySql, sin embargo, estos gestores de BD que instalamos en nuestra PC, normalmente lo usamos para hacer pruebas.

Para nuestra página web que vamos a colgar en el internet vamos a necesitar utilizar un gestor de BD MySql, es decir un **servidor remoto**, esto nos va a proporcionar nuestro servidor WEB o nuestro servidor de servicios internet (ISP).

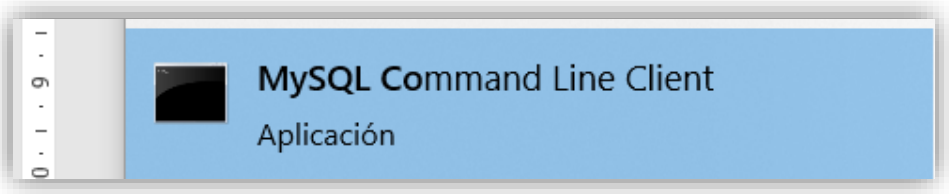
Es decir, una vez que comprobemos que muestra base de datos funciona en servidor local, lo podemos subir a un servidor remoto. Pero para este paso, debemos de cambiar ciertos parámetros porque en local no es lo mismo que en remoto.

10.3.2 Acceso al servidor MySql

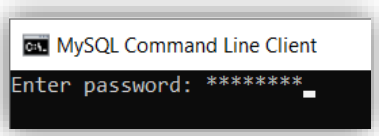
Por consola

Los paquetes como Wamp Server, AppServ y otros nos ofrecen acceso por consola, y todo lo que hagas por consola también se puede hacer por PhpMyAdmin, la única diferencia es que por consola solo digitas código, sin embargo, PhpMyAdmin tiene una consola grafica con el propósito de facilitar las tareas que podemos hacer con MySql.

Para abrir por consola debemos de escribir en el buscador de aplicaciones de Windows:

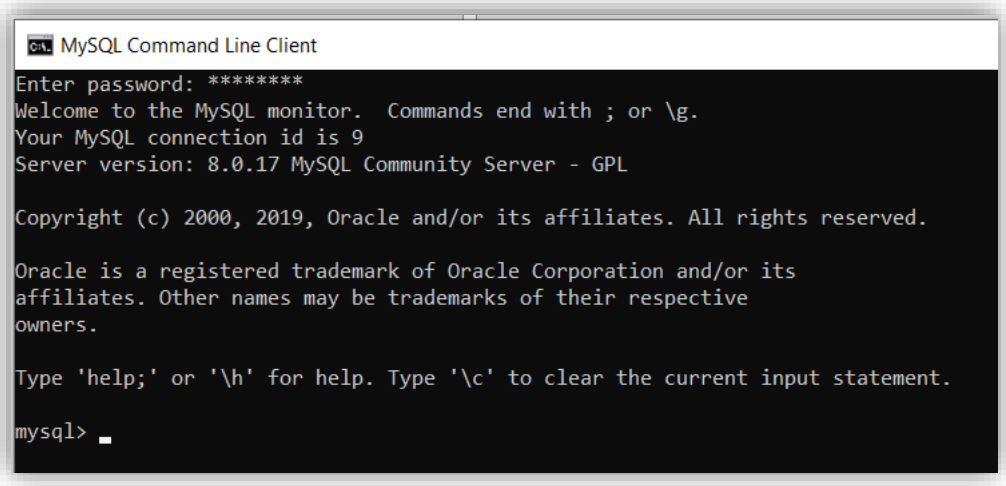


Luego debemos de ingresar la clave del root, que en nuestro caso iniciamos con 12345678



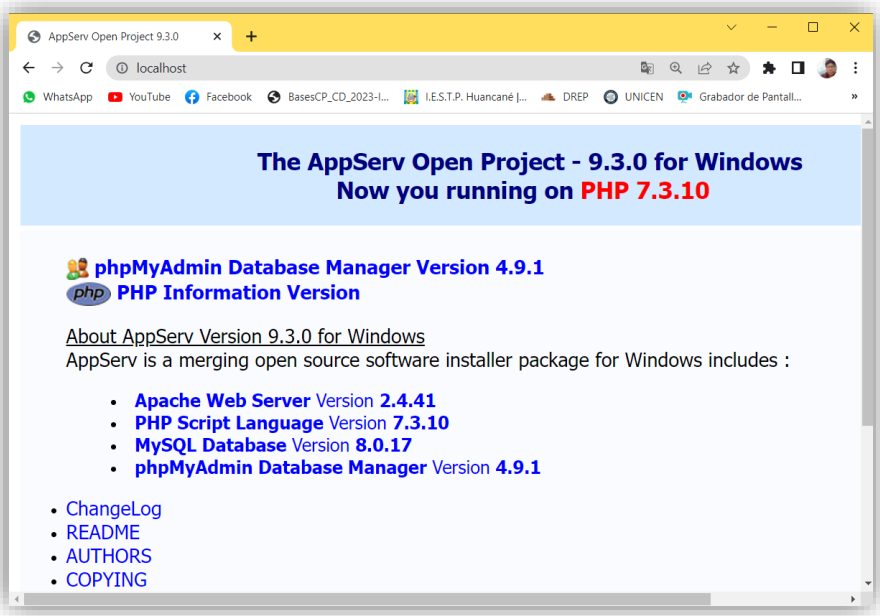
Luego presionamos Enter, y entonces se mostrará la siguiente pantalla:

En esta pantalla podremos escribir nuestra codificación en SQL como si lo estuviéramos haciendo en PhpMyAdmin.



Por PhpMyAdmin

Cuando utilicemos en PhpMyAdmin debemos de abrir un navegador de internet de su preferencia, y en la barra de direcciones escribir localhost, luego presionamos Enter. En caso no funcione con localhost, puede intentar con 127.0.0.1

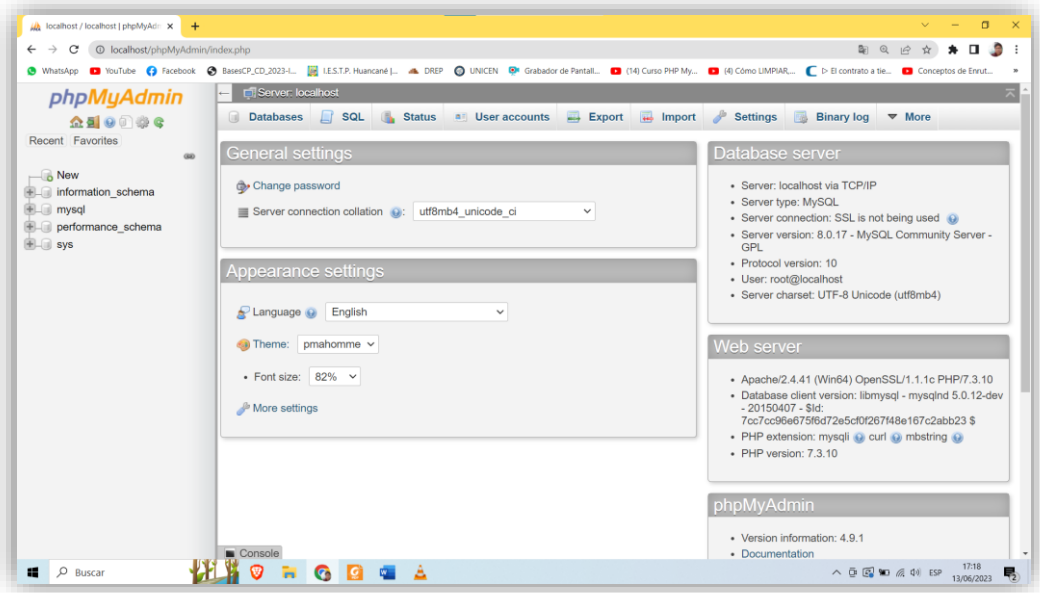


Algunos otros programas como el wamp necesitan que los servidores se inicialicen.

Para abrir PhpMyAdmin debemos de hacer clic en **PhpMyAdmin Database Manager Versión 4.9.1**

Se abrirá una ventana en donde el usuario es **root** y la contraseña es el número que se le ha solicitado al momento de la instalación de su servidor local, en nuestro caso he colocado como 12345678. Luego hacemos clic en **go.** En otros casos en lugar de **go** aparecerá **continue**

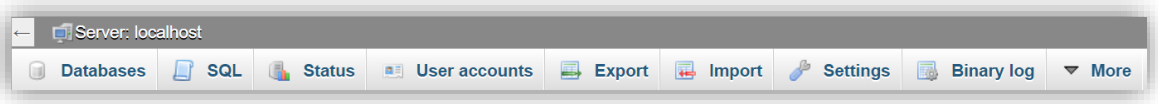
Luego de ejecutar **go,** si el password era el correcto aparecerá la siguiente ventana:



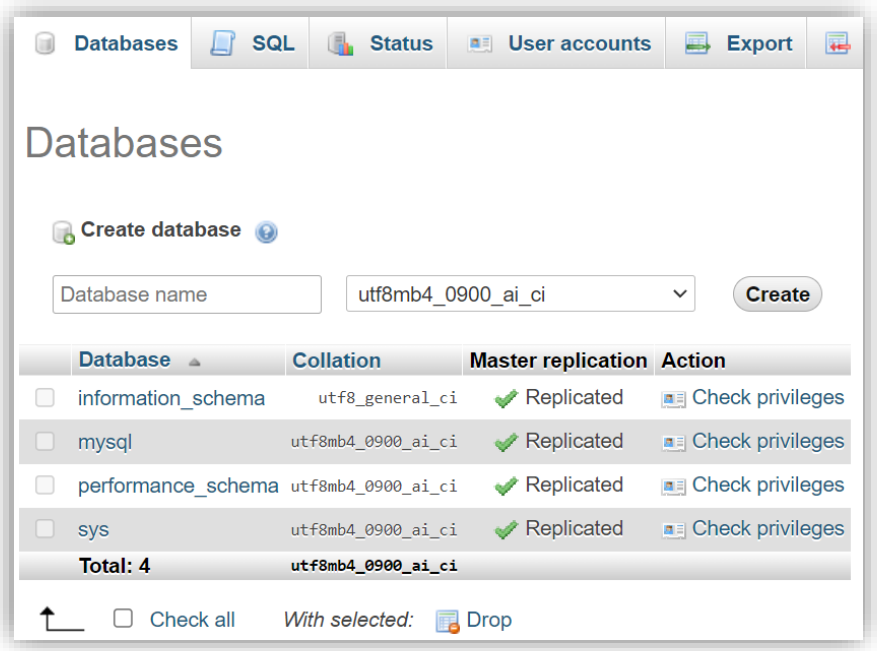
La imagen anterior muestra la interfaz gráfica que nos va permitir realizar todas las tareas que requieran base de datos con MySql.

Al lado derecho podemos observas algunas bases de datos que ya están instaladas, estas bases de datos han sido instaladas por defecto con el paquete del servidor local y que no se debe de eliminar, porque contienen información para que todo el gestor de base de datos MySql funcione de manera correcta.

En la parte de la derecha tenemos una barra de herramientas, por ejemplo, si hacemos clic en base de datos, vemos como carga la base de datos que se cuenta hasta el momento, incluso tiene una opción para crear una nueva base de datos.



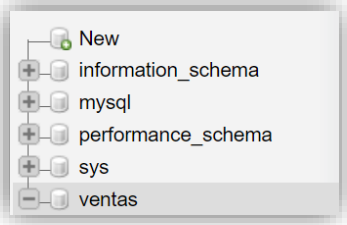
La opción drop es para eliminar una base de datos. Donde dice total: 4, indica la cantidad de base de datos que existe en nuestro servidor.



10.4 Creación de una base de datos desde PhpMyAdmin

Para crear una base de datos, lo primero que debemos hacer es hacer clic en el botón base de datos y luego aparece un menú para crear la base de datos, debemos de poner un nombre a la base de datos y en cotejamiento seleccionar **utf8_spanish2_ci,** luego pulsamos en el botón **create**:

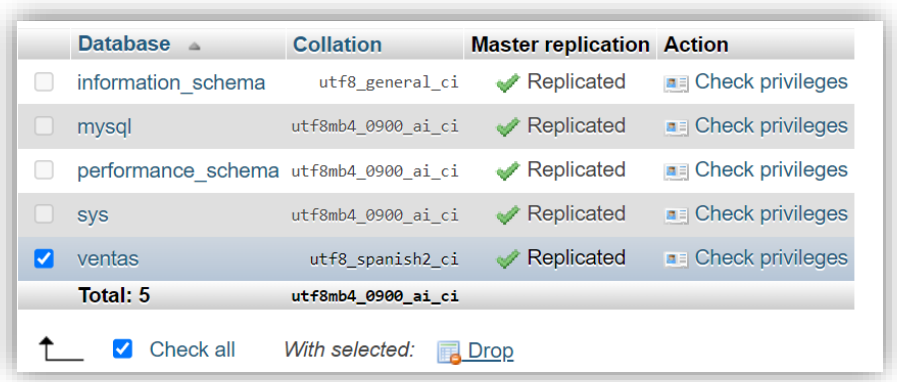
Luego de pulsar en create, debe de salir un mensaje de **base de datos creada satisfactoriamente.** Debemos de observar que al costado izquierdo debió actualizarse las bases de datos, ahora debe incluir la nueva base de datos creada.



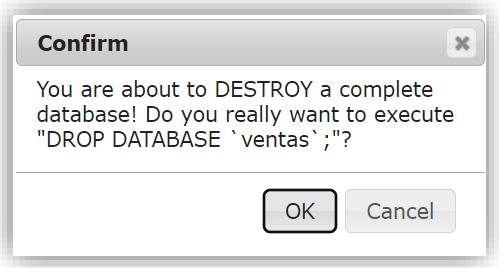
Claro que hasta el momento nuestra base de datos está vacía, es decir, no hay tablas en su interior.

10.5 Eliminar una base de datos desde PhpMyAdmin

En el botón de base de datos, donde se encuentran nuestras bases de datos, debemos de seleccionar con un check la base de datos que queremos eliminar, luego haremos clic en **drop o en eliminar** según el idioma de su PhpMyAdmin.



Luego de presionar en **Drop,** aparecerá un mensaje de confirmación para eliminar la base de datos **ventas,** e incluso en dicha ventana nos muestra la instrucción SQL que permite eliminar una base de datos, en este caso la instrucción es: **DROP DATABASE ventas;**

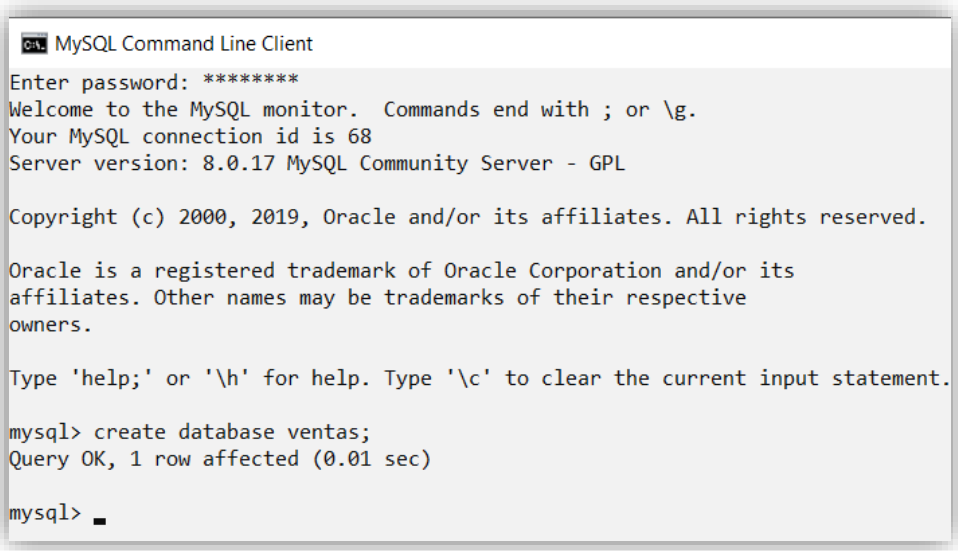


Luego presionamos OK. Si no hubo problemas, se habrá eliminado la base de datos.

10.6 Creación de una base de datos desde consola

Primeramente, debemos de abrir la consola de MySql, colocaremos la contraseña, y crearemos ahí nuestra base de datos.

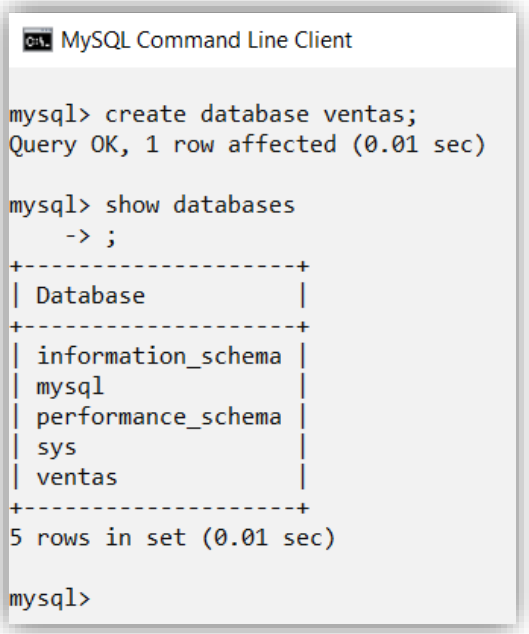
En el promt escribir **create database ventas;** y presione Enter, debe de salir un mensaje de **Query OK.**



Nos dice que se ha ejecutado una consulta y que una fila a sido afectada.

10.7 Ver las bases de datos que tenemos en el servidor

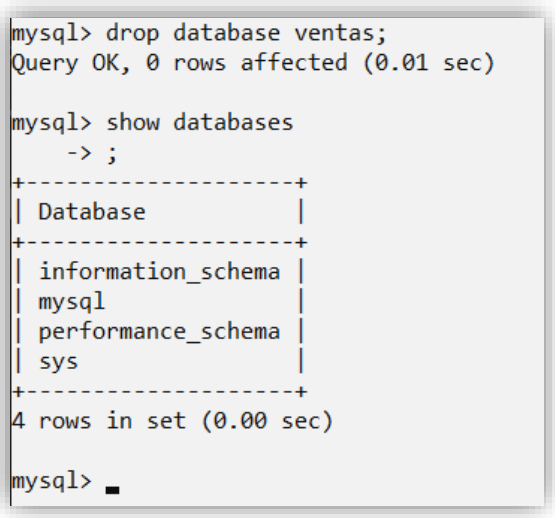
Para ver las bases de datos debemos de escribir la instrucción: **show databases;** luego Enter. Puede observar que fue creado la base de datos **ventas** y las otras 4 base de datos que vino preinstalado.



10.8 Eliminar una base de datos por consola

Para eliminar una base de datos debemos escribir la instrucción: **drop database ventas;**

Presionamos Enter y luego veremos que ya no está la base de datos con la instrucción **show databases;**

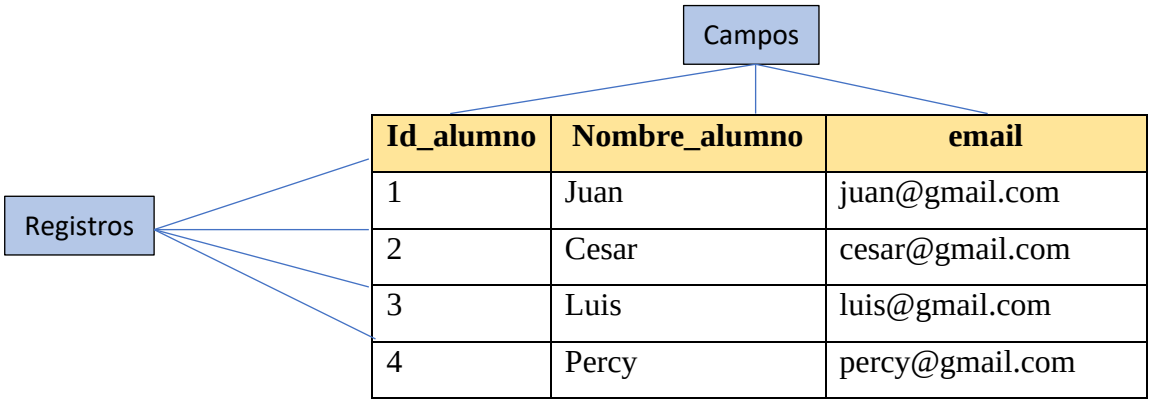


TEMA 11

CREACIÓN Y MANIPULACIÓN DE TABLAS EN MYSQL

11.1 Tabla

Una tabla es una estructura que se divide en columnas y en filas, en una tabla a los encabezados de las columnas se les denomina **campos,** y a las filas se les denomina **registros.** la finalidad de una tabla es el de almacenar información.



Esta información presenta una tabla que contiene nombres de amistades y su correo electrónico, en caso que se quiera obtener información de sus correos electrónicos, entonces haremos uso del SQL para hacer una consulta y obtener información.

Cada campo debe de tener un nombre, en este caso tenemos 3 campos de nombres: Id_alumno, Nombre_alumno, email. Otra cosa importante es que se debe de indicar el **tipo de dato** que almacenará cada campo, como por ejemplo el campo **Nombre_alumno** almacenará datos de tipo texto. Si en nuestros registros hay números y nosotros le decimos a la BD que es de tipo texto, entonces la BD tomar este campo como tipo texto, solo si quisiéramos hacer operaciones matemáticas con estos datos, entonces lo estaríamos declarando como de **tipo numérico.**

11.2 Creación de tablas.

Sabemos que antes de crear nuestras tablas debemos tener creada nuestra base de

datos, para ello abrimos la consola de MySql o mediante PhpMyAdmin, dependiendo de cómo guste trabajar. En nuestro caso vamos a trabajar desde consola para que se vaya familiarizando con la ejecución de código SQL.

Use usuarios: con esta instrucción le estamos diciendo a nuestro servidor que vamos a trabajar con la base de datos usuarios, porque se supone que podemos tener N bases de datos, por lo que para crear tablas o hacer manipulaciones primero debemos entrar a la base de datos que queremos trabajar.

Los nombres de las tablas no deben contener caracteres extraños ni espacios.

VARCHAR nos permite ingresar hasta 255 caracteres, es por ello que le debemos de ingresar la longitud, pues un nombre es case imposible que tenga 250 caracteres, por lo que se debe de poner una cantidad de caracteres razonable.

```
mysql> create database usuarios;
Query OK, 1 row affected (0.02 sec)

mysql> use usuarios;
Database changed
mysql>  create table datos_usuarios (nombre varchar(30), clave varchar(10));
Query OK, 0 rows affected (0.08 sec)

mysql>
```

Si queremos ver las tablas creadas debemos de usar el comando **show,** si es para ver bases de datos será **show databases** y si es para ver tablas será **show tables**

```
mysql> show tables
    -> ;
+--------------------+
| Tables_in_usuarios |
+--------------------+
| datos_usuarios     |
+--------------------+
1 row in set (0.00 sec)

mysql>
```

Si queremos eliminar una tabla debemos de usar el comando **DROP**. La instrucción para eliminar una tabla sería: **drop table *nombre_tabla;***

Note usted que, si escribe mal el nombre de la tabla o de la instrucción, dará como resultado un error, por lo que debemos de tener un diccionario de datos, en donde tendremos todas nuestras tablas, de tal manera que debemos de escribir correctamente el nombre de la tabla

```
mysql> drop table datos_usuario;
ERROR 1051 (42S02): Unknown table 'usuarios.datos_usuario'
mysql> drop table datos_usuarios;
Query OK, 0 rows affected (0.03 sec)

mysql>
```

Ahora utilizaremos la instrucción **describe,** pero como hemos eliminado la tabla, debemos de crearla nuevamente, en caso contrario, si no hay tablas nos enviara un mensaje de error.

Describe *nombre tabla* nos muestra la estructura de la tabla como los campos de la tabla, el tipo de datos, la cantidad de caracteres y todos los demás atributos de cada campo.

Nota. Con las flechas del teclado podemos ubicar la instrucción con la que hemos creado la tabla datos_usuarios, poner punto y coma (;) y presionar Enter.

```
mysql>  create table datos_usuarios (nombre varchar(30), clave varchar(10));
Query OK, 0 rows affected (0.03 sec)

mysql> describe datos_usuarios;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| nombre | varchar(30) | YES  |     | NULL    |       |
| clave  | varchar(10) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql>
```

11.3 Insertar registros

Para insertar registros usaremos el comando **insert.** Podemos indicar los campos

en los cuales vamos a insertar registros o bien no lo podemos indicar como se a continuación.

```
mysql> insert into datos_usuarios values ("Wilson", "12345678");
Query OK, 1 row affected (0.01 sec)

mysql>
```

Indicando los campos:

```
mysql> insert into datos_usuarios (nombre, clave) values ("Wilson", "12345678");
Query OK, 1 row affected (0.01 sec)

mysql>
```

Con la instrucción **select** podemos ver lo que hemos ingresado por teclado.

```
mysql> select * from datos_usuarios;
+--------+----------+
| nombre | clave    |
+--------+----------+
| Wilson | 12345678 |
| Wilson | 12345678 |
+--------+----------+
2 rows in set (0.00 sec)
```

11.4 Tipos de datos en MySql

Datos numéricos	Datos cadena	Datos de fechas y horas
TINYINT	CHAR	DATE
SMALLINT	VARCHAR	TIME
MEDIUMINT	BINARY	DATETIME
INTEGER	VARBINARY	TIMESTAMP
BIGINT	TINYBLOB	
DECIMAL	TINYTEXT	
NUMERIC	BLOB	
FLOAT	TEXT	
DOUBLE	MEDIUMBLOB	
	MEDIUMTEXT	
	LONGBLOB	

	LONGTEX ENUM SET	

11.4.1 Tipos de datos numéricos

Los datos numéricos corresponden a datos expresados en números. Por ejemplo, la edad, cantidad de personas de una población, el precio de un producto, tipo de cambio de una moneda, etc. Escoger cuál tipo de dato es el más adecuado al momento de crear los campos dependerá del análisis previo realizado.

Los tipos de datos numéricos se dividen en dos grupos: datos numéricos enteros y datos numéricos decimales.

Numéricos enteros. Los datos numéricos enteros son aquellos que carecen de un punto decimal. Las opciones que tenemos para almacenar este tipo de datos, son:

TINYINT. Bytes: 1, valor con signo mínimo – máximo: -128 a 127, valor sin signo: 0 a 255.

SMALLINT. Bytes: 2, valor con signo mínimo – máximo: -32768 a 32767, valor sin signo: 0 a 6553

MEDIUMINT. Bytes: 3, valor con signo mínimo – máximo: -8388608 a 8388607, valor sin signo: 0 a 16777215

INTEGER. Bytes: 4, valor con signo mínimo – máximo: -2147483648 a 2147483647, valor sin signo: 0 – 4294967295

BIGINT. Bytes 8, permite números desde -9223372036854775808 hasta 9223372036854775807. Si se define como UNSIGNED (sin signo) permite números desde 0 hasta 18446744073709551615.

Numéricos reales. Los datos numéricos decimales están compuestos por una parte entera y otra decimal. Las opciones que tenemos para almacenar este tipo de datos, son las siguientes:

DECIMAL. Definir con precisión muy exacta. Números significativos. No se debe perder de vista la exactitud.

NUMERIC. Definir con precisión muy exacta. Números significativos. No se debe perder de vista la exactitud.

FLOAT. Precisión simple, la exactitud no tiene mucha relevancia. Rango de precisión entre 0 y 24.

DOUBLE. Precisión doble, la exactitud no tiene mucha relevancia. Rango de precisión entre 25 y 53.

11.4.2 Tipos de datos cadenas

Los datos de tipo cadena representan datos alfanuméricos que pueden incluir letras, números, espacios y caracteres especiales. Las opciones con las que se cuenta son:

CHAR. El tipo de dato CHAR almacena una cadena de datos de longitud fija de hasta 255 caracteres, que se reserva en la memoria, aunque no se la utilice toda.

VARCHAR. VARCHAR almacena la misma cantidad de caracteres que CHAR de 255 caracteres, pero la longitud es variable. La longitud dependerá del contenido que se almacena en la memoria. Por ejemplo, el texto “Framework” consumirá diez caracteres, nueve para las letras y uno para la longitud del texto.

BINARY. BINARY es similar a los tipos de datos CHAR y VARCHAR, la diferencia radica en que no almacena caracteres sino bytes. Asimismo, la diferencia con VARBINARY radica en la cantidad de bytes que almacena**.**

BLOB. Los datos de tipo BLOB guardan información en formato binario de gran tamaño, generalmente se utiliza este tipo de datos para guardar imágenes, sonido y archivos. Las diferencias de TYNYBLOB, MEDIUMBLOB y LONGBLOB se diferencian por la cantidad máxima de almacenamiento.

TINYBLOB. Longitud máxima en Bytes: 255

BLOB. Longitud máxima en Bytes: 65535

MEDIUMBLOB. Longitud máxima en Bytes: 16777215

LONGBLOB. Longitud máxima en Bytes: 4 GB

TEXT. Es utilizado para guardar cualquier tipo de texto de gran tamaño, bajo este formato se pueden almacenar blogs enteros, noticias, comentarios, publicaciones, etc. Las diferencias entre TINYTEXT, MEDIUMTEXT y LONG TEXT es la cantidad de caracteres.

TINYTEXT. Longitud máxima en Bytes: 255

TEXT. Longitud máxima en Bytes: 65535

MEDIUMTEXT. Longitud máxima en Bytes: 16777215

LONGTEXT. Longitud máxima en Bytes: 4294967295

ENUM. (enumeración) es un tipo de datos especial que se utiliza para definir valores predeterminados de una lista, los cuales deben estar separados por comas (solo estos valores son permitidos en el campo al momento de ingresar el dato). Se puede almacenar hasta 65535 valores diferentes.

SET. Es un tipo de dato que representa un conjunto de cadenas que puede contener 1 o más valores, similar a ENUM con la diferencia que se puede almacenar más de un valor en el campo.

11.4.3 Tipo de datos fechas y horas

Los tipos de datos de fechas y horas representan un periodo determinado en el tiempo. Las opciones con las que se cuenta son:

DATE. DATE es un tipo de dato que permite almacenar fechas en el formato "YYYY-MM-DD", donde YYYY representa el año, MM representa el mes y DD representa el día. Permite almacenar fechas en un rango de 1000-01-01 a 9999-12-31.

TIME. TIME es similar a DATE, pero sirve para almacenar horas, minutos y segundos; el formato es "HH:MM:SS" donde HH representa la hora, MM los minutos y SS los segundos.

DATETIME. DATETIME es un tipo de dato que nos permite registrar con exactitud un determinado periodo de tiempo, almacena las fechas y horas. El formato es "YYYY-MM-DD HH:MM:SS".

TIMESTAMP. TIMESTAMP es un tipo de dato similar a DATETIME con la diferencia de que el rango de fechas utilizado es el presente (desde 1970-01-01 hasta 2037-12-31). Cuenta con tres tipos de formato "YYYY-MM-DD HH:MM:SS", "YYYY-MM-DD" y "YY-MM-DD".

TEMA 12

CONEXIÓN DE LA BASE DE DATOS CON PHP

En este tema vamos a tratar sobre como conectar una base de datos MySql desde una página web. Esto se hace con el objetivo de obtener información de la base de datos, es cuando buscamos un registro o grupos de registros que se encuentra almacenada en esa base de datos.

Después de saber ¿el cómo conectarnos a la base de datos?, debemos de saber el ¿cómo hacer consultas a la base de datos con instrucciones SQL?

En base de datos tenemos dos grandes consultas, las cuales son las consultas de selección y las consultas de acción. Donde las consultas de acción son aquellas que modifican la información que se encuentra almacenada en la base de datos, y las consultas de selección que permite capturar información de una base de datos sin modificar la información almacenada en la base de datos.

12.1 Consultas SQL

Las consultas en SQL se realizan utilizando la instrucción **select,** luego se pone los campos que se quiera ver como resultado de la consulta, estos campos deben estar separados entre comas, o si desea ver todos los campos de la tabla deberá de poner *. Luego ponemos la instrucción **from** y luego en nombre de la tabla.

También podemos agregar filtros, como el **where.**

Sintaxis de la instrucción SELECT

SELECT campos a visualizar **FROM** nombre de la tabla

12.2 Conexión de la base de datos con MySql

Para conectar una base de datos MySql con una página web PHP, vamos a necesitar 4 datos importantes:

12.2.1 Dirección de la base de datos

Viene a ser la dirección del servidor de base de datos, esta dirección se coloca en la barra de direcciones de un navegador web. Usualmente esta dirección es: **localhost** o **127.0.0.1**

Ahora, si estamos trabajando con un hosting y dominio, esta dirección nos la dará nuestro proveedor de servicio.

- **Nombre de la base de datos.** También es necesario tener el nombre de la base de datos con la que vamos a interactuar.
- **Usuario de la base de datos.** El usuario de la base de datos, suele ser **root,** esto en caso estemos trabajando como servidor local.
- **Contraseña de la base de datos.** La contraseña de la base de datos, es la contraseña con el que se ingresa al servidor de la base de datos, podemos decir que es la que ponemos en el **root.** También puedo aclarar que esta contraseña es la que pusimos al momento de instalar el AppServ.

Para poder obtener la información de las tablas, podemos trabajar de dos formas:

- **Con instrucciones orientadas a objetos.** Si utilizamos orientado a objetos utilizaremos la clase **Mysqli** con todas sus propiedades y métodos.
- **Con instrucciones por procedimientos.** Su trabajamos con procedimientos utilizaremos la función **mysqli_connect** (antiguamente se usaba la instrucción mysql_connect, pero han quedado desfazadas con la nueva versión de PHP, si funciona, pero se recomienda actualizarnos con las nuevas instrucciones)

Empecemos, debemos de ver que todos nuestros servicios del servidor, estén funcionando. Realizaremos una página web que llamaremos **conexión.php**

Almacenaremos los 4 datos que necesitaremos para conectarnos con la base de datos, primeramente, vamos a hacer la conexión con la base de datos mediante **procedimientos.**

```
$host_bd = "localhost";
$nombre_bd = "academico";
$usuario_bd = "root";
$clave_bd = "12345678";
```

Vamos a tener que crear una variable llamada $**conexión** y diremos que es igual a la función **mysqli_connect,** en donde la función nos pide parámetros, el primer parámetro es la dirección de la base de datos, luego el usuario de la base de datos, luego la contraseña, y por último el nombre de la base de datos.

```
$conexion = mysqli_connect($host_bd, $usuario_bd, $clave_bd, $nombre_bd);
```

En algunas computadoras puede que haya algún error, por lo que, se recomienda que ese error lo pongan en el Google y vean las soluciones que pueda haber en páginas web o algunos foros. Estos errores pueden deberse a la ausencia de una librería o alguna configuración.

Bien, ahora para ver si se está accediendo a la base de datos vamos a realizar una consulta, en este caso, vamos a crear una variable a la que llamaremos **$consulta.**

```
$consulta = "SELECT * FROM DATOS";
```

Como podemos ver, tenemos almacenado en la variable consulta una instrucción SQL, donde nos va almacenar en la variable **$consulta** todos los registros de todos los campos de la tabla **datos.**

El siguiente paso es ejecutar la consulta, y para ejecutar esa consulta tenemos que usar la función **mysqli_query** que nos va pedir dos parámetros: la conexión y la sentencia SQL que tenemos almacenado en la variable **$consulta.**

Creamos una variable a la cual llamaremos **$resultado**

```
$resultado = mysqli_query($conexion, $consulta);
```

Nota. Puede ver que estamos utilizando variables para almacenar datos como la conexión, pero, si queremos podemos poner de frente las conexiones o consultas en las funciones. Yo lo hago para que no haya confusión, o se entienda un poco mejor.

En el momento en que se ejecuta la sentencia **mysqli_query,** lo que se hizo es crear una tabla en memoria donde se carga toda la información que nos devuelve la sentencia SQL, en este caso nos devuelve 4 campos y los registros que tenga almacenado. Y esa tabla virtual lo hemos almacenado en la variable **$resultado**

El siguiente paso es mirar lo que hay en la variable **$resultado.** Para la que utilizaremos otra función: **myqli_fetch_row (**Esta función lo que hace es ir viendo línea a línea los datos que hay en la tabla**).** Para esto debemos crear una variable a la que llamaremos **$fila** que será igual a la función Mysqli_fetch_row, y ahí debemos de poner la tabla virtual que hemos almacenado previamente en **$resultado**

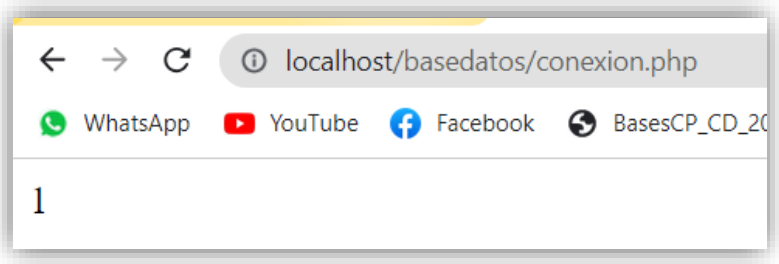
```php
$fila = mysqli_fetch_row($resultado);
```

$fila realmente es un array que almacena lo que hay en **$resultado.** Lo que queda ahora es preguntar al array por lo que hay almacenado en la primera fila.

Por ejemplo, vamos a decirle al array que nos diga que tenemos almacenado en la posición 0.

```php
<?php
//el primer paso es guardar en una variable los datos que usaremos.

$host_bd = "localhost";
$nombre_bd = "academico";
$usuario_bd = "root";
$clave_bd = "12345678";

// el siguiente paso es conectarnos a la base de datos

$conexion = mysqli_connect($host_bd, $usuario_bd, $clave_bd, $nombre_bd);
$consulta = "SELECT * FROM datos";
$resultado = mysqli_query($conexion, $consulta);
$fila = mysqli_fetch_row($resultado);
echo $fila[0];

?>
```

Resultado.

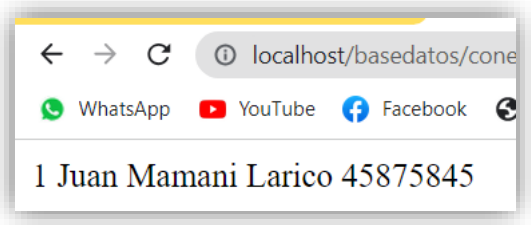


Puede ver que nos da como resultado el primer valor de nuestro primer registro y primer campo de la tabla datos.

Y si queremos ver más datos de nuestra fila o registro, solo debemos de indicar al programa que nos muestre los demás datos.

```php
<?php
//el primer paso es guardar en una variable los datos que usaremos.

$host_bd = "localhost";
$nombre_bd = "academico";
$usuario_bd = "root";
$clave_bd = "12345678";

// el siguiente paso es conectarnos a la base de datos

$conexion = mysqli_connect($host_bd, $usuario_bd, $clave_bd, $nombre_bd);
$consulta = "SELECT * FROM datos";
$resultado = mysqli_query($conexion, $consulta);
$fila = mysqli_fetch_row($resultado);
echo $fila[0]." ";
echo $fila[1]." ";
echo $fila[2]." ";
echo $fila[3];

?>
```

Resultado.



12.3 Manejo de errores en la conexión con la base de datos

Partiremos del ejercicio anterior, en el ya hicimos la conexión la base de datos. Podemos imaginarnos que hemos cometido el error de escribir incorrectamente el nombre del host. Por ejemplo, para este caso vamos a equivocarnos en poner el nombre del host.

```
$host_bd = "localhos";
```

En algunos casos les aparecerá mensajes de error, y en otros simplemente no hará nada, no va mostrar la consulta debido a que no hay conexión con la base de datos.

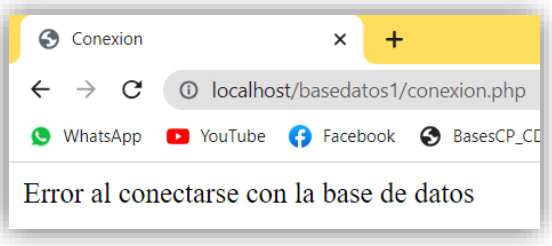
Lo que debemos hacer es que después de la conexión con la base de datos, es agregar un condicional, en este caso vamos a usar la instrucción **mysqli_connect_errno()**

Esta función se va ejecutar siempre y cuando no haya una conexión con la base de datos, y acá decimos que, si se ejecuta esa función, hay un error, entonces enviaremos un mensaje de que hay una falla al conectar con la base de datos, y luego tenemos que decirle que salga del código PHP, porque el programa va intentar ejecutado las demás líneas de código como: la consulta, los resultados y la fila. Entonces para esto usaremos la función **exit( ).**

Para probar el programa, debemos de ejecutar con el error involuntario que provocamos anteriormente.

```php
<?php
$host_bd = "localhos";
$nombre_bd = "academico";
$usuario_bd = "root";
$clave_bd = "12345678";

$conexion = mysqli_connect($host_bd, $usuario_bd, $clave_bd, $nombre_bd);
if(mysqli_connect_errno())
{
    echo "Error al conectarse con la base de datos";
    exit();
}
$consulta = "SELECT * FROM datos";
$resultado = mysqli_query($conexion, $consulta);
$fila = mysqli_fetch_row($resultado);
echo $fila[0]." ";
echo $fila[1]." ";
echo $fila[2]." ";
echo $fila[3];

?>
```

Resultado



Después de ver el error, debemos de revisar cual es el motivo por el que no se esta conectando la base de datos, corrigiendo el error volvemos a probar el código hasta que funciones nuestra consulta.

Cuando tengamos el problema de que no está reconociendo alguna letra como la eñe, debemos de ejecutar el siguiente código:

```php
mysqli_set_charset($conexion, "utf8");
```

Con este código le estamos diciendo al php, que usaremos el utf8 para caracteres especiales. Por lo que, al ejecutar este código veremos que corrige los caracteres.

Trabajo

Otro error que puede ocurrir es que el nombre de la base de datos no corresponda, también saldrá un error. También podemos decirle al php que nos avise si el error corresponde al nombre de la base de datos.
Explicar mediante un ejemplo que parte del código podemos modificar o agregar código para que el sistema nos envíe un mensaje de que el nombre de la base de datos no es correcto.

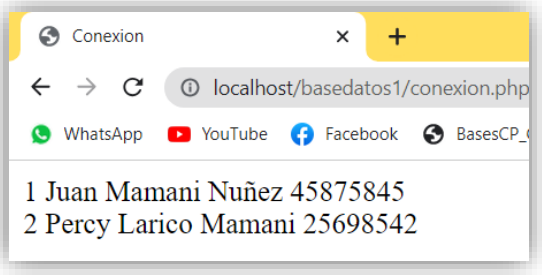
12.4 Recorrido del array de resultados SQL

Recordemos que cuando ejecutamos el programa anterior, únicamente nos muestra el primer registro, es decir, cuando llamamos a la función **mysqli_fetch_row( ),** lo que se hizo fue crear un **resulset**, la función **mysqli_fetch_row( )** en la primera llamada accede al primer registro que hay almacenado en la tabla virtual. Pero si llamamos nuevamente a **mysqli_fetch_row( ),** lo que hará es que se hará una llamada al segundo registro, y si llamamos por tercera vez, se accede al tercer registro, y así sucesivamente.

Veamos como funcionaria si hacemos una segunda llamada a la función **mysqli_fetch_row( )**

```php
<?php
$host_bd = "localhost";
$nombre_bd = "academico";
$usuario_bd = "root";
$clave_bd = "12345678";

$conexion = mysqli_connect($host_bd, $usuario_bd, $clave_bd, $nombre_bd);
if(mysqli_connect_errno())
{
    echo "Error al conectarse con la base de datos";
    exit();
}

mysqli_set_charset($conexion, "utf8");

$consulta = "SELECT * FROM datos";
$resultado = mysqli_query($conexion, $consulta);
$fila = mysqli_fetch_row($resultado);
echo $fila[0]." ";
echo $fila[1]." ";
echo $fila[2]." ";
echo $fila[3]."<br>";

$fila = mysqli_fetch_row($resultado);
echo $fila[0]." ";
echo $fila[1]." ";
echo $fila[2]." ";
echo $fila[3];

?>
```

Resultado.



Podemos ir ejecutando nuevamente la función **mysqli_fetch_row( ),** pero eso no es dable porque si tuviéramos que mostrar 1000 registros, tendríamos que copiar la función 1000 veces.

Lo que podemos hacer es poner la función **mysqli_fetch_row( )** en un bucle, y se ejecute hasta que no haya más registros en la tabla.

```php
<?php
$host_bd = "localhost";
$nombre_bd = "academico";
$usuario_bd = "root";
$clave_bd = "12345678";

$conexion = mysqli_connect($host_bd, $usuario_bd, $clave_bd, $nombre_bd);
if(mysqli_connect_errno())
{
    echo "Error al conectarse con la base de datos";
    exit();
}

mysqli_set_charset($conexion, "utf8");

$consulta = "SELECT * FROM datos";
$resultado = mysqli_query($conexion, $consulta);

    $registros = 1;

    while($registros<=2)
    {
        $fila = mysqli_fetch_row($resultado);
        echo $fila[0]." ";
        echo $fila[1]." ";
        echo $fila[2]." ";
        echo $fila[3]."<br>";
        $registros++;
    }
?>
```

Ahora, analicemos, la condición anterior no es de utilidad cuando no sabemos cuántos registros hay, lo que debemos de decirle al while es que ejecute mientras siga habiendo algo de información: mientras la variable $fila sea igual a mysqli_fetch_row($resultado) sea verdad hará lo que esta abajo.

```php
<?php
$host_bd = "localhost";
$nombre_bd = "academico";
$usuario_bd = "root";
$clave_bd = "12345678";

$conexion = mysqli_connect($host_bd, $usuario_bd, $clave_bd, $nombre_bd);
if(mysqli_connect_errno())
{
    echo "Error al conectarse con la base de datos";
    exit();
}

mysqli_set_charset($conexion, "utf8");

$consulta = "SELECT * FROM datos";
$resultado = mysqli_query($conexion, $consulta);

    while(($fila=mysqli_fetch_row($resultado))==true)
    {
        echo $fila[0]." ";
        echo $fila[1]." ";
        echo $fila[2]." ";
        echo $fila[3]."<br>";
    }
 ?>
```

Ahora lo que debemos de aprender es a cerrar sesión en la base de datos, y eso lo hacemos con **mysqli_close($conexion),** y obviamente debemos de especificar la conexión que deseamos cerrar. Esto se hace porque desde una aplicación php podemos conectarnos a más de una base de datos.

TEMA 13

IMPORTACIÓN DE TABLAS

13.1 Optimizando la conexión a la base de datos

En el anterior tema en nuestra primera conexión del PHP con la base de datos, donde teníamos almacenados los datos necesarios en variables, ahora que pasaría si en nuestra aplicación tenemos varios archivos que queremos conectar con la misma base de datos.

Lo que tendríamos que hacer es volver a repetir el código de datos de configuración en cada archivo, sin embargo, nos podemos ahorrar esa escritura, si guardamos los datos de configuración en un archivo PHP, independiente, y después con la función **require** o con la función **include.**

En un archivo nuevo de PHP, vamos a crear un archivo llamado **datos_de_conexion.php** en la cual van a estar únicamente los datos de conexión.

```php
<?php
$host_bd = "localhost";
$nombre_bd = "academico";
$usuario_bd = "root";
$clave_bd = "12345678";
?>
```

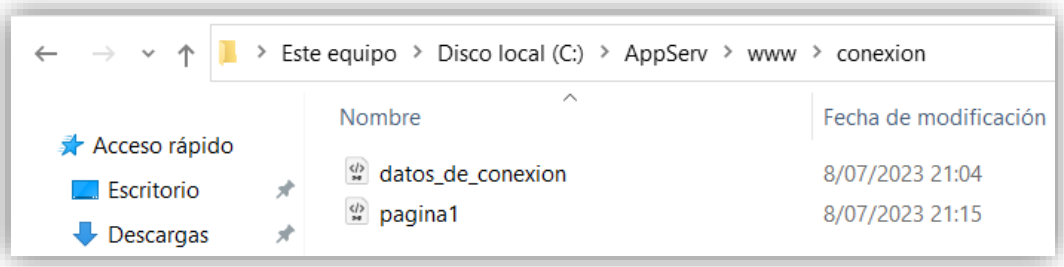
Ahora, en cada archivo que vayamos a querernos conectar con la base de datos, únicamente haremos uso de la función **require("datos_de_conexion.php")**

En **require** se pone la dirección donde está ubicado la función, en caso que esté en el mismo directorio, no es necesario poner la ruta, y con esa acción ya tenemos la conexión.

El archivo nuevo que va utilizar los datos anteriores lo llamaremos **pagina1.php**

Y en todos los demás archivos que queramos conectar con la base de datos, lo haremos haciendo la llamada con la instrucción require.

La carpeta que en la que grabo los dos archivos lo he llamado conexión y esta dentro del servidor (pueden llamarlo como gusten), y dentro de esta carpeta esta grabado los dos archivos y queda asi:

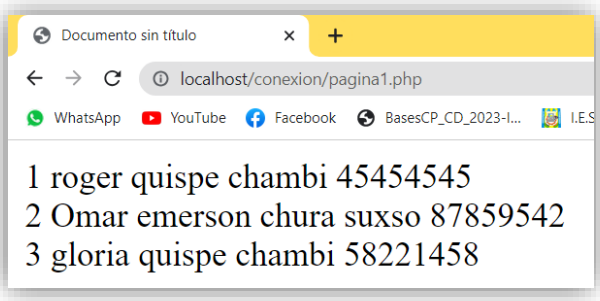


El código **pagina1.php** quedaría de la siguiente manera:

datos_de_conexion.php* × pagina1.php ×

```php
<html>
<head>
<meta charset="utf-8">
<title>Documento sin título</title>
</head>

<body>
<?php
require("datos_de_conexion.php");

$conexion = mysqli_connect($host_bd, $usuario_bd, $clave_bd, $nombre_bd);

if(mysqli_connect_errno())
{
    echo "Error al conectarse con la base de datos";
    exit();
}

$consulta = "SELECT * FROM datos";
$resultado = mysqli_query($conexion, $consulta);

    $registros = 1;

    while($registros<=3)
    {
        $fila = mysqli_fetch_row($resultado);
        echo $fila[0]." ";
        echo $fila[1]." ";
        echo $fila[2]." ";
        echo $fila[3]."<br>";
        $registros++;
    }
?>
</body>
</html>
```

El resultado de la ejecución de página1.php es:

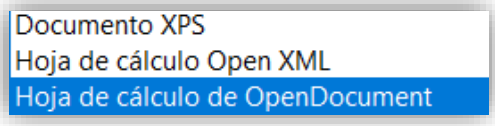


Como sabemos en nuestra base de datos solo tenemos 3 registros, las cuales hemos insertado directamente y de forma manual en el PhpMyAdmin.

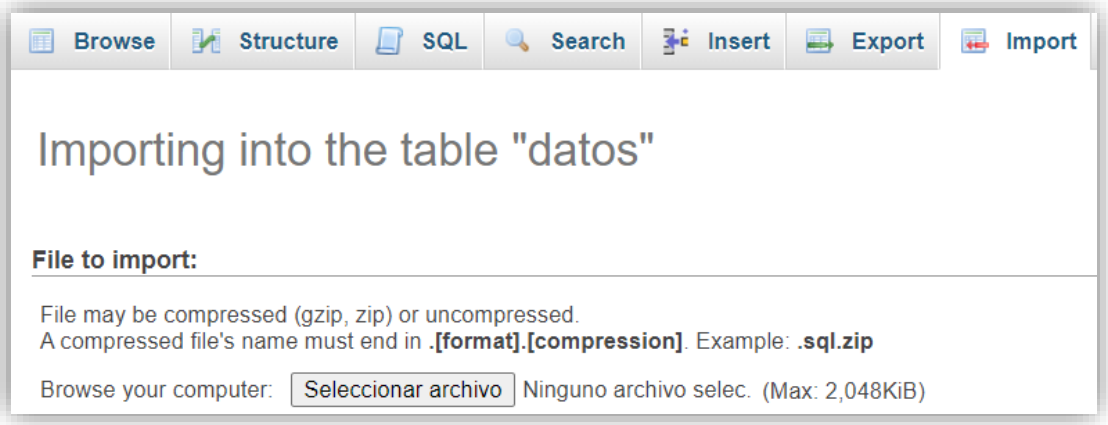
13.2 Importación de tablas a la base de datos en mysql

Para importar tablas, vamos a usar un archivo que llamado **datos.ods** (open document), este archivo puede ser abierto por Excel. Esta tabla tiene almacenado mas datos para poder insertar en nuestra tabla anterior.

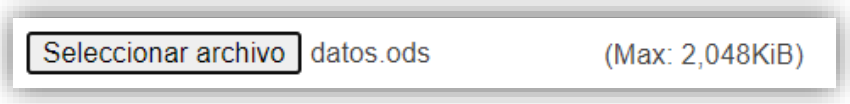
En caso en Excel tengamos información y esta grabado con la extensión xlsx, debemos de cambiarlo a la extensión ods (open document, mediante guardar como y buscar esa opción)



Estando en nuestra base de datos **académico,** debemos ir a la opción **importar** y luego hacer clic en **seleccionar archivo.**



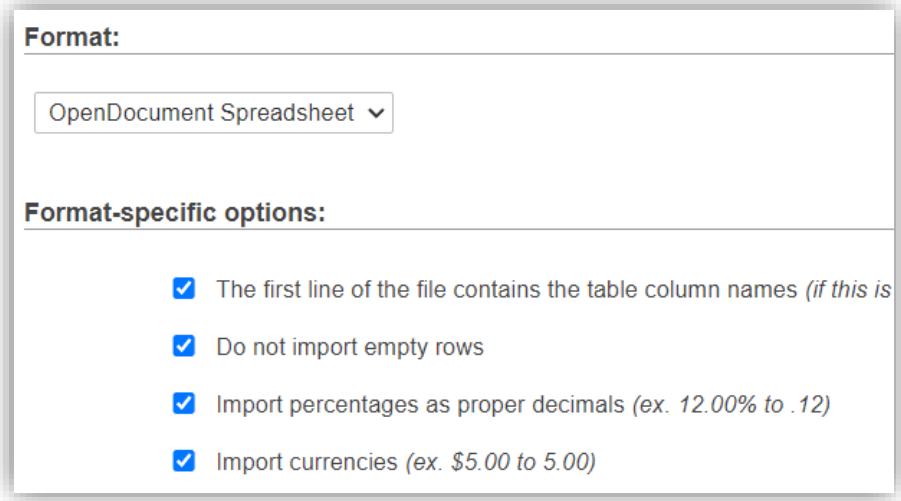
Luego, debemos de buscar y seleccionar el archivo ODS que vamos a importar.



Aparecerá el archivo elegido al costado de seleccionar archivo, luego debemos de asegurarnos que en **format** aparezca **OpenDocument** y más abajo en **Opciones especificas al formato** debemos de seleccionar el check en **La primera línea del archivo contiene los nombres de columna de la tabla (**fijarse en la tabla ods si tiene encabezado, en caso lo tenga debemos de activar esta casilla, en el caso mío está en inglés**)**

Si nos fijamos en el archivo ODS, la primera fila corresponde a los campos de mi tabla:

	A	B	C	D
1	id_datos	nombre	apellidos	dni
2	4	YHON CLEVER	AJAHUANA MAMANI	73523887
3	5	JUAN CARLOS	APAZA APAZA	46891437
4	6	FABIAN FABRICIO	BENITEZ RODRIGO	73418727
5	7	OMAR DAVID	CANAZA MOLLEAPAZA	70140004



Luego le hacemos clic en **go o continuar**

NOTA. Si cuando importemos el archivo ODS y queremos que los datos se inserten en nuestra tabla que ya teníamos creado, en este caso en la tabla **datos,** debemos

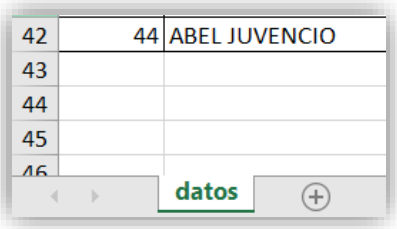
de considerar que la hoja de Excel debe tener el mismo nombre que nuestra tabla **datos,** y los nombres de los campos también deben ser los mismos, así como la cantidad de campos deben de coincidir para evitar problemas. No olvidar que teníamos 3 registros con **id_datos** desde el 1 al 3 por lo que nuestro archivo ODS debe iniciar desde el 4, así:

id_datos	nombre	apellidos	dni
4	YHON CLEVER	AJAHUANA MAMANI	73523887
5	JUAN CARLOS	APAZA APAZA	46891437
6	FABIAN FABRICIO	BENITEZ RODRIGO	73418727
7	OMAR DAVID	CANAZA MOLLEAPAZA	70140004
8	XIOMARA JULIZA	CAPIA QUISPE	46248004
9	DAVID ERNESTO	CHIPANA CHEJE	73525790
10	OLGA MARINA	CHIPANA HANCCO	77334322
11	ZULEMA YAQUELIN	CHIPANA URBINA	74349950
12	ANGEL MOISES	CHUQUIMAMANI CONDORI	70508260
13	PAVEL SAMUEL	CONDORI QUISPE	70364233

Por lo tanto, después de la importación del archivo ODS la tabla quedaría de la siguiente manera:

	id_datos	nombre	apellidos	dni
Edit Copy Delete	1	roger	quispe chambi	45454545
Edit Copy Delete	2	Omar emerson	chura suxso	87859542
Edit Copy Delete	3	gloria	quispe chambi	58221458
Edit Copy Delete	4	YHON CLEVER	AJAHUANA MAMANI	73523887
Edit Copy Delete	5	JUAN CARLOS	APAZA APAZA	46891437
Edit Copy Delete	6	FABIAN FABRICIO	BENITEZ RODRIGO	73418727
Edit Copy Delete	7	OMAR DAVID	CANAZA MOLLEAPAZA	70140004
Edit Copy Delete	8	XIOMARA JULIZA	CAPIA QUISPE	46248004
Edit Copy Delete	9	DAVID ERNESTO	CHIPANA CHEJE	73525790
Edit Copy Delete	10	OLGA MARINA	CHIPANA HANCCO	77334322
Edit Copy Delete	11	ZULEMA YAQUELIN	CHIPANA URBINA	74349950
Edit Copy Delete	12	ANGEL MOISES	CHUQUIMAMANI CONDORI	70508260
Edit Copy Delete	13	PAVEL SAMUEL	CONDORI QUISPE	70364233
Edit Copy Delete	14	WILBER	CONDORI RAMOS	76599626
Edit Copy Delete	15	EDWIN IVAN	CUEVA MAMANI	77132794
Edit Copy Delete	16	RIVALDO	GONZA CONDORI	63266494
Edit Copy Delete	17	JHOJAN YAMPIER ANDY	GUTIERREZ TORRES	70168775
Edit Copy Delete	18	GLENIA IRIS DEL PILA	HUANCA BAUTISTA	73428544
Edit Copy Delete	19	FLOR MELIN	JUAREZ MENDOZA	72426096
Edit Copy Delete	20	FERNANDO JOSE	LARICO LIRA	72121944

Si desea importar tablas ODS que no correspondan a nuestras tablas que tenemos creadas, lo puede hacer sin ningún problema, pues esta tabla se va a crear en la base de datos con el nombre de la hoja que tiene definido en el archivo ODS.



13.3 Modificando el acceso a registros de la base de datos

Cuando mostramos los registros en nuestro código PHP, únicamente se muestra 3 registros, pero no siempre vamos a tener esa cantidad de registros, puede haber más, es por tal motivo que debemos de modificar el código mostrado: (en página1.php)

```php
$registros = 1;

while($registros<=3)
{
    $fila = mysqli_fetch_row($resultado);
    echo $fila[0]." ";
    echo $fila[1]." ";
    echo $fila[2]." ";
    echo $fila[3]."<br>";
    $registros++;
}
```

Lo que tenemos que hacer es que la instrucción que muestra los registros debemos de meterlo en un bucle, de tal manera que muestre los registros hasta que no haya más en la tabla. Lo que tenemos que decirle al while es que ejecute lo que hay en su interior mientras siga habiendo información, y eso lo vamos a hacer de la siguiente forma:

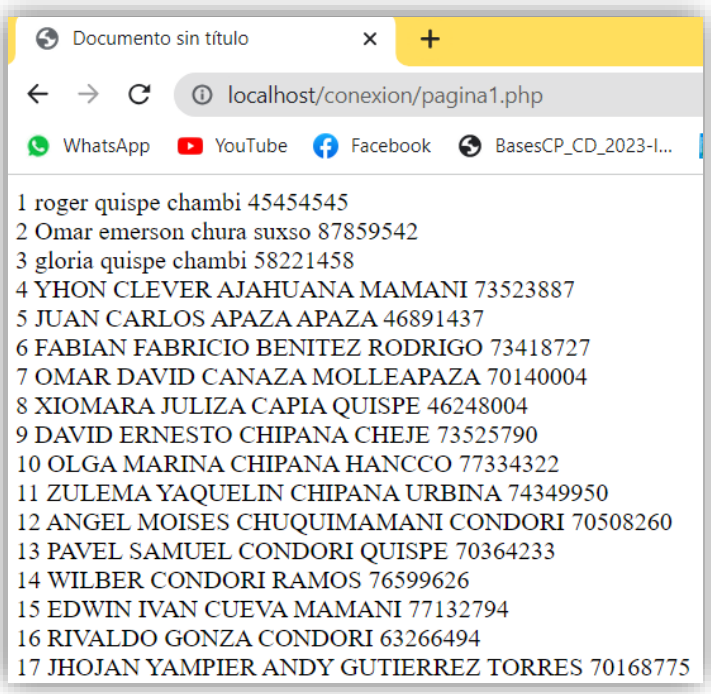
Trato de decir que mientras $fila sea igual a verdad, se ejecute las líneas de abajo

```php
$consulta = "SELECT * FROM datos";
$resultado = mysqli_query($conexion, $consulta);

    while(($fila = mysqli_fetch_row($resultado)) == true)
    {
        echo $fila[0]." ";
        echo $fila[1]." ";
        echo $fila[2]." ";
        echo $fila[3]."<br>";

    }
```

El código de pagina1.php quedaría de la siguiente manera:

```php
<?php
require("datos_de_conexion.php");

$conexion = mysqli_connect($host_bd, $usuario_bd, $clave_bd, $nombre_bd);

if(mysqli_connect_errno())
{
    echo "Error al conectarse con la base de datos";
    exit();
}

$consulta = "SELECT * FROM datos";
$resultado = mysqli_query($conexion, $consulta);

    while(($fila = mysqli_fetch_row($resultado)) == true)
    {
        echo $fila[0]." ";
        echo $fila[1]." ";
        echo $fila[2]." ";
        echo $fila[3]."<br>";

    }
?>
```

Resultado de la ejecución del código anterior seria: (En la imagen no se muestra todos los datos que hemos importado, pero en realidad hay más datos)



El anterior código también se puede simplificar de la siguiente manera: (Porque se asume que $fila = mysqli_fetch_row($resultado) dentro del while y sin ninguna otra condición se asume como verdad), el código funcionar de idéntica manera al anterior.

```
while($fila = mysqli_fetch_row($resultado))
{
    echo $fila[0]." ";
    echo $fila[1]." ";
    echo $fila[2]." ";
    echo $fila[3]."<br>";

}
```

13.4 Cerrar la base de datos

Como sabemos hemos abierto la base de datos con:

$conexion = mysqli_connect($host_bd, $usuario_bd, $clave_bd, $nombre_bd);

Y puede consumir recursos, por lo que, para optimizar recursos, lo que debemos hacer es cerrar la base de datos, claro eso debemos de hacer cuando ya no necesitamos acceder a la base de datos, cerraremos la conexión con la función Mysqli_close(acá especificar la conexión que queremos cerrar).

```
$consulta = "SELECT * FROM datos";
$resultado = mysqli_query($conexion, $consulta);

    while($fila = mysqli_fetch_row($resultado))
    {
        echo $fila[0]." ";
        echo $fila[1]." ";
        echo $fila[2]." ";
        echo $fila[3]."<br>";
    }
    mysqli_close($conexion);
?>
```

A veces en una misma página web podemos acceder a más de una base de datos, es por este motivo que debemos de indicar a muchas de las funciones que cierre la conexión.

Si queremos conectar con 6 bases de datos diferentes, tendremos que realizar 6 conexiones diferentes. Entonces para cerrar la conexión debemos de especificar que conexión deseamos cerrarla.

Bibliografía

Cobo, Á., Gómez, P., Pérez, D., & Rocha, R. (2005). *PHP y MySQL*. Díaz de Santos. https://www.editdiazdesantos.com/wwwdat/pdf/9788479787066.pdf

López Quijado, J. (2018). *Domine PHP y MySQL* (2.ª ed.). RA-MA Editorial. https://books.google.com/books?id=jo-fDwAAQBAJ

Torres Remón, M. Á. (2021). *Desarrollo de Aplicaciones web con PHP y MySQL*. Macro Editorial. https://editorialmacro.com/wp-content/uploads/2021/02/9786123042554.pdf

Minera, F. (2014). *PHP y MySQL desde cero*. USERS. https://elhacker.info/manuales/Desarrollo%20web/PHP-Mysql/PHP%2BMySQL%20Desde%20Cero%2C%20USERS%20-%20Francisco%20Minera.pdf

Nixon, R. (2015). *Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5* (4th ed.). O'Reilly Media. https://www.bluebooksoft.com/DISENO_PROGRAMACION_WEB/1349.pdf

¿Quieres aprender a programar en PHP con bases de datos desde cero y llegar al éxito en el desarrollo web?

Este libro te guiará paso a paso en el fascinante mundo del desarrollo web y la programaeción con PHP, desde lo más basico hasta la integración con bases de datos MySQL.

LO QUE APRENDERÁS

- ✓ Instalar y configurar un servidor local para PHP
- ✓ Dominar los fundamentos de PHP desde cero
- ✓ Trabajar con variables, operadores, strings, y funciones
- ✓ Crear y conectar bases de datos, MySQL con PHP
- ✓ Realizar consultas SQL: INSERT, SELECT, UPDATE, DELETE
- ✓ Manipular y gestionar la información almacenada
- ✓ Desarrollar aplicaciones web dinamicas y seguras
- ✓ Consejos y buenas prácticas en programacion

Al final de cada sección, encontrarás ejercicios prácticos para reforzar tus conocimientos.

¡Prepárate para dar el paso hacia una carrera exitosa en el desarrollo web con PHP!