

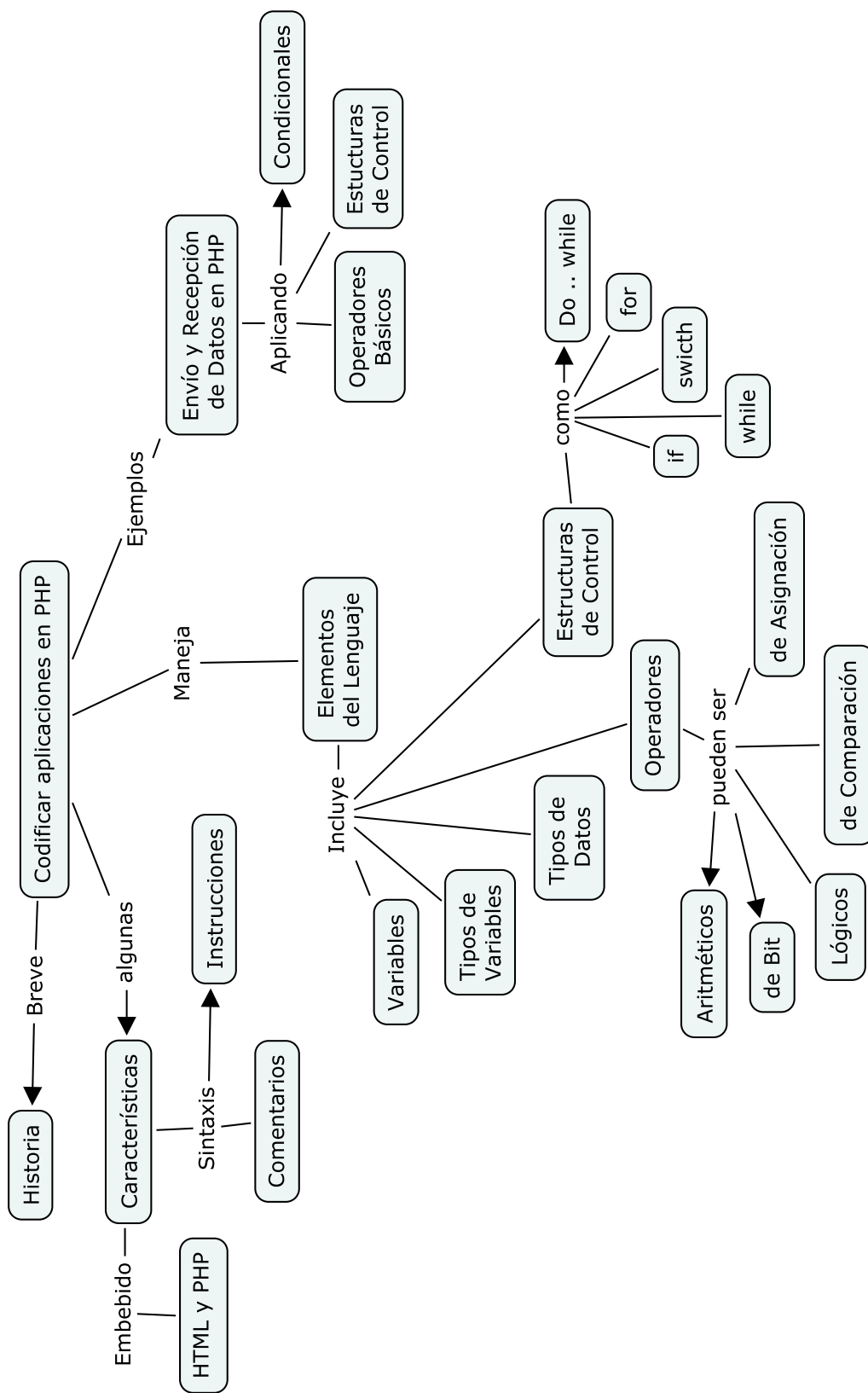
Fundamentos de programación con PHP

INTRODUCCIÓN	3
1. GENERALIDADES DEL LENGUAJE	4
1.1. Historia de PHP.....	5
1.2. Programar en PHP.....	6
1.3. Características del Lenguaje	9
1.3.1. Comentarios	10
1.3.2. Instrucciones	11
2. ELEMENTOS DEL LENGUAJE	11
2.1. Variables	11
2.2. Obtener información sobre el tipo de una variable	12
2.3. Tipos de Datos	13
2.4. Operadores	14
Aritméticos.....	14
De Asignación	14
Operadores de Bit	15
Operadores de comparación.....	16
Operadores Lógicos	16
2.5. Estructuras de Control	17
Condicional Simple (IF)	17
Ciclo Mientras (WHILE)	18
Ciclo Haga mientras (DO ... WHILE).....	20
Ciclo repita n veces (FOR)	21

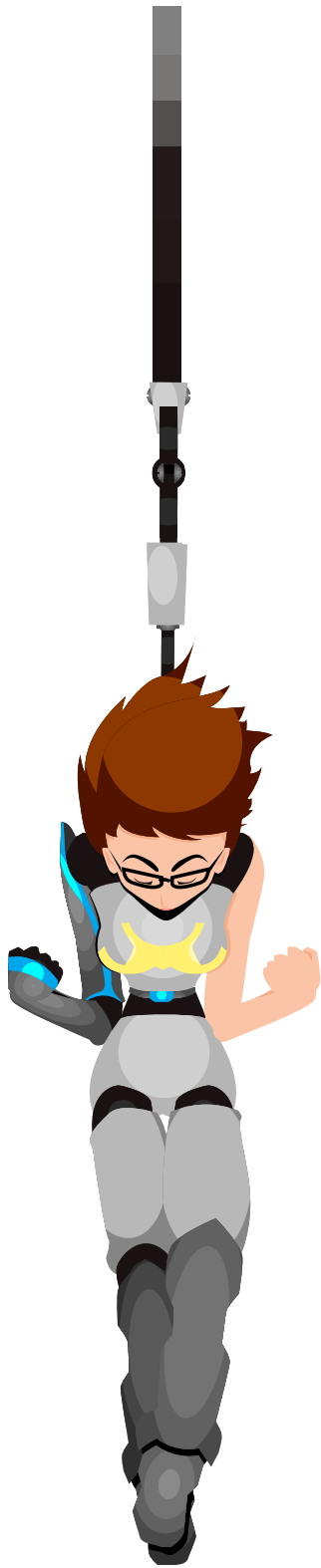


Decisión basada en múltiples opciones (SWITCH)	22
3. ENVÍO Y RECEPCIÓN DE DATOS EN PHP.....	25
3.1. Ejemplo con operadores básicos en PHP.....	27
3.2. Ejemplo de condicionales sencillos en PHP	31
3.3. Ejemplo de la estructura cíclica hacer para en PHP	33
3.4. Ejemplo del manejo de un control combobox en PHP	35
GLOSARIO.....	37
BIBLIOGRAFÍA	38





INTRODUCCIÓN



Dentro del proceso de codificación, especialmente en la parte web, se cuenta con un lenguaje de código abierto muy popular conocido como PHP (Hypertext Pre-Processor), dicho lenguaje está especialmente adecuado para el desarrollo web y puede ser incrustado en HTML.

Mediante la construcción de los ejercicios guiados y con toda la información incluida en este Objeto de Aprendizaje, logrará desarrollar habilidades de codificación en PHP.

Para iniciar el trabajo con PHP, se tomarán algunos algoritmos básicos y

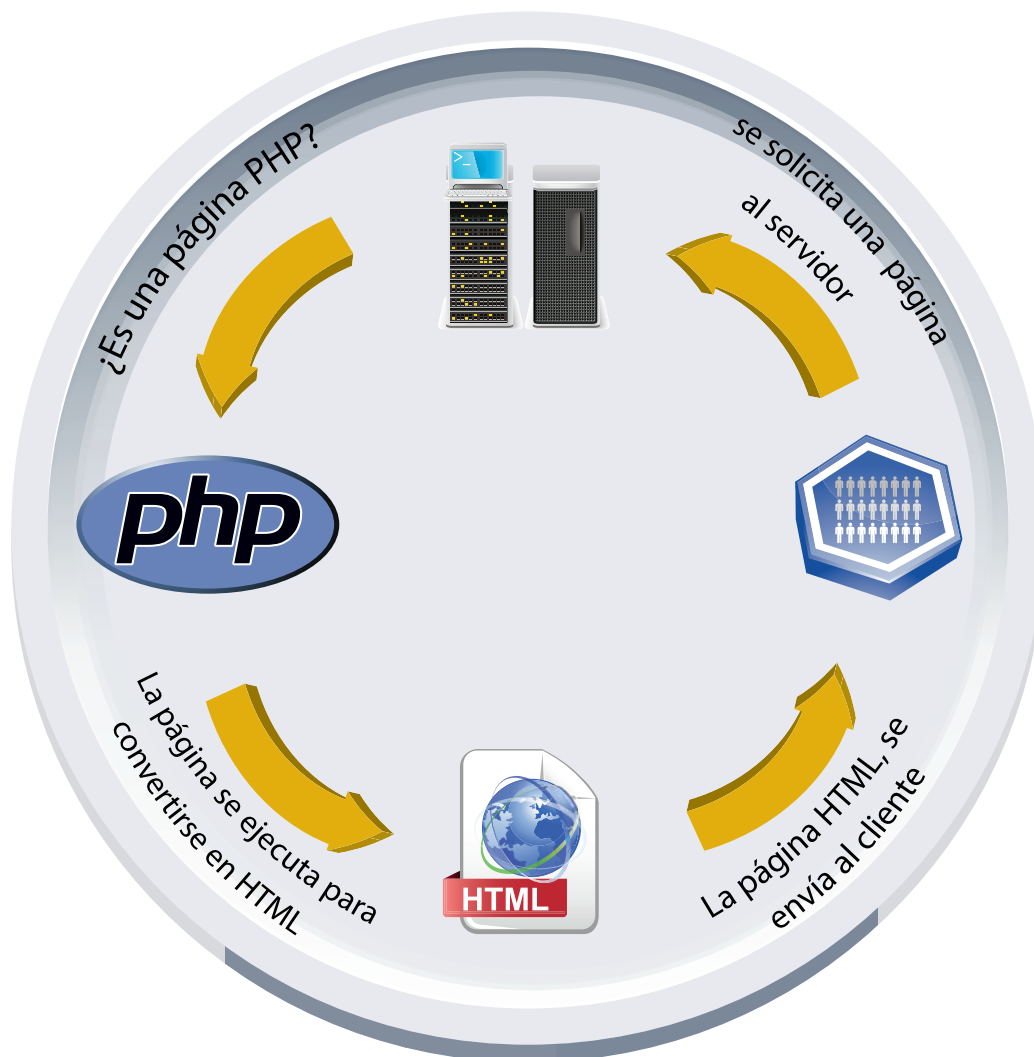
se desarrollarán con esta herramienta, como estrategia pedagógica y con el fin de evidenciar que quien maneja la lógica y la estructura de datos en lenguajes de programación, puede utilizar cualquier herramienta; en algunos casos se realizará un paralelo con el lenguaje de programación JAVA y PHP.

Una vez se tengan claros los fundamentos de programación en PHP, se pasará al manejo de Bases de datos desde este lenguaje, dando como resultado el desarrollo de habilidades en cuanto a codificación en PHP se refiere.

1. GENERALIDADES DEL LENGUAJE

"P.H.P., es un lenguaje de programación del lado del servidor (se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente), gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación."

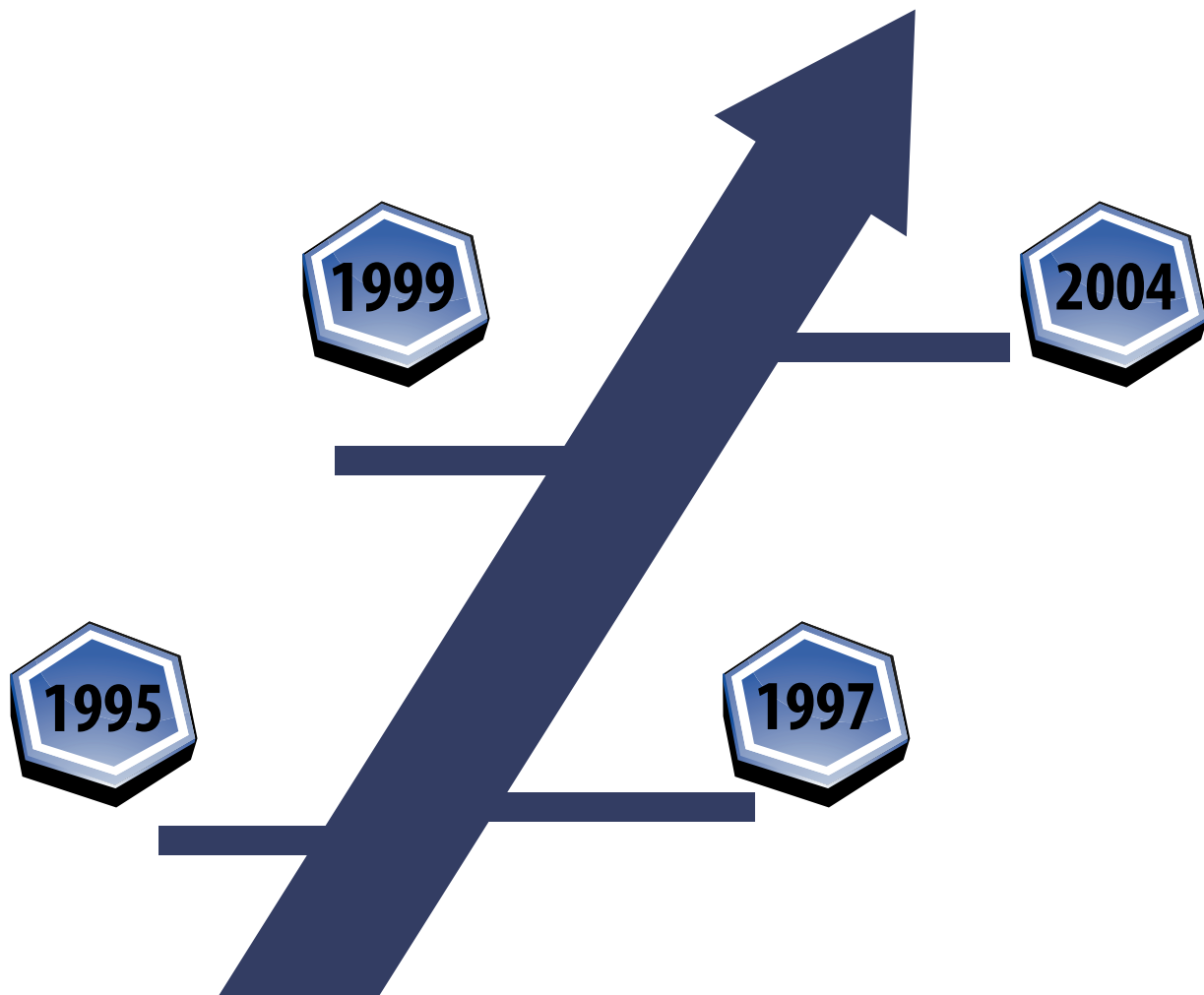
(<http://www.desarrolloweb.com/articulos/392.php>)



- Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos y otras tareas, el cliente solamente recibe una página con el código HTML resultante de la ejecución de la página de PHP.

Como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores.

1.1. Historia de PHP



PHP/FI 1995

Creado por Rasmus Lerdorf permitía desarrollar pequeñas aplicaciones web dinámicas.



PHP 3.0 1997

Creado por Andy Gutmans y Zeev Zuraski. Una de sus grandes características era su gran extensibilidad. El soporte de sintaxis orientado a Objetos fue otra característica clave. Esta versión fue liberada oficialmente en Junio de 1998.



PHP 4.0 1999

Utilizó el motor “Motor Zend” de Zeev, Andi. Fue Oficialmente liberado en mayo de 2000. Además de la mejora de ejecución, incluía otras características claves como soporte para la mayoría de servidores web, sesiones HTTP, buffers de salida y muchas nuevas construcciones del Lenguaje.



PHP 5 2004

Utiliza el motor Zend Engine II. Entre sus características se encuentran: Mejor soporte a la programación orientada a Objetos. Mejoras de Rendimiento. Mejoras para MySQL y XML. Soporte integrado para SOAP. Manejo de Excepciones.

1.2. Programar en PHP

Para desarrollar en PHP, es necesario contar con un servidor que tenga un entorno que soporte PHP, por regla general este servidor depende del sistema operativo que se tenga instalado en la máquina.

Algunos ejemplos que incorporan Apache (Servidor de páginas web), MySQL (Bases de Datos) y soporte al lenguaje PHP, son:

- LAMP (Linux).

- WAMP (Windows).
- MAMP (MacOSX).

En las versiones Express, para crear una estructura de almacenamiento se crean esquemas *“Un esquema es un espacio de almacenamiento donde residen los objetos (tablas, vistas, índices, procedimientos almacenados...) que son propiedad de un usuario”*.

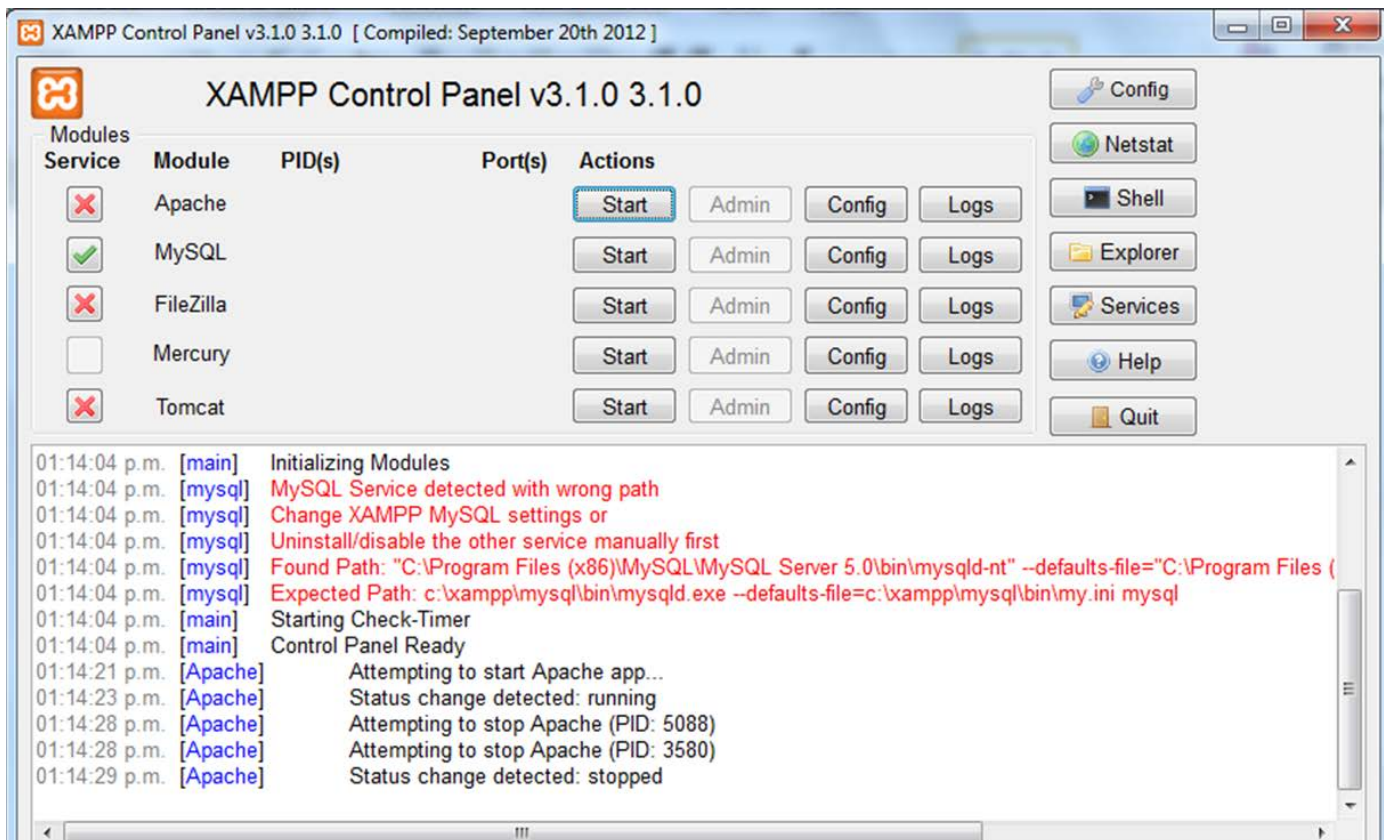
Para la construcción de las tablas, campos, relaciones y restricciones de la estructura de almacenamiento, se debe tomar como referencia el diseño de la base de datos previamente realizado. La base de Datos con la cual se va a trabajar se debe llamar citas y va a contener cinco tablas así: Pacientes, Medicos, Consultorios, Citas y Tratamientos.

Una alternativa rápida de instalación del servidor, es utilizar herramientas como XAMPP que a través de un solo paquete de instalación permite activar el servidor. Estos paquetes están disponible para los diferentes Sistemas Operativos (SO), en este objeto de aprendizaje se utilizará el SO Windows.



XAMPP

Una vez instalado el paquete, se debe iniciar el servidor mediante el panel de control de la herramienta. Es importante aclarar que el panel de control puede variar, dependiendo de la versión de XAMPP que se tenga instalada.



Un aspecto fundamental al momento de trabajar con PHP, es la ruta donde se encuentran las páginas que debe procesar el servidor. Para el caso de Xampp, se debe ubicar la carpeta de instalación "xampp" y dentro de ésta carpeta ubicar la subcarpeta "htdocs", donde se debe ubicar todas las páginas que se desarrollen dando una organización adecuada por carpetas.

Para visualizar los resultados de una página PHP, se debe acceder desde el navegador web ingresando al sitio del servidor local, el cual se identifica con la palabra localhost. Solo se mostrarán resultados de las páginas publicadas en el directorio web "htdocs".

1.3. Características del Lenguaje

El código PHP, está embebido dentro del código HTML, esto significa que el intérprete solo ejecuta las instrucciones que están incluidas dentro de las etiquetas manejadas por el lenguaje. Todo lo que esté fuera de las etiquetas PHP se deja tal como está, mientras que el resto se interpreta como código.

Se tienen ya predefinidos un conjunto de etiquetas que pueden ser usadas para representar bloques de código PHP.

Etiqueta 1



Representa la salida por pantalla,
Un echo en PHP es como un, ←
System.out.println en JAVA

```
<? php  
echo (" Es compatible con archivos  
XHTML o XML, es la etiqueta  
utilizada normalmente");  
?>
```

Etiqueta 2



```
<?  
echo ("salida por pantalla,  
funciona en versiones PHP y  
anteriores");  
?>
```

Etiqueta 3



```
<? script language ="php"  
echo ("Si se está trabajando desde  
FrontPage, no reconoce las  
instrucciones PHP");  
?>
```

Etiqueta 4



```
< %  
echo ("Puede ser compatible con  
ASP");  
% >
```

1.3.1. Comentarios

PHP, al igual que todos los lenguajes de programación maneja comentarios de una sola línea y de múltiples líneas, soporta el estilo de comentarios de los lenguajes de programación "C", "C++" y de interfaz de comandos de UNIX.

```
<? php  
echo ("Ejemplo de comentario");  
// Estilo de una sola línea  
/* Estilo  
de comentario  
con varias líneas */  
?>
```

```
<? php  
echo ("Comentario estilo sell de UNIX");  
# Estilo de una sola línea  
?>
```

1.3.2. Instrucciones

Todas las instrucciones en PHP, finalizan con punto y coma. Es muy importante, que antes de incluir el conjunto de instrucciones, se inicie el bloque PHP (<? php) y una vez se terminen las instrucciones, se finalice el bloque (?>)

Para construir las sentencias, se necesitan diferentes elementos, que ya se han trabajado en otros lenguajes de programación, estos elementos son, variables, constantes, operadores, estructuras de control, entre otros.

Antes de iniciar con nuestros ejercicios, tomemos los siguientes aspectos como referente.

1. Crear el archivo con extensión php, el cual contiene el respectivo código HTML, estos conceptos los trabajaron previamente en el Objeto de aprendizaje "HMTL".
2. Dentro del código HTML, se debe vincular las respectivas instrucciones PHP.
3. Los archivos PHP, deben estar incluidos dentro de la carpeta htdocs del xampp.

2. ELEMENTOS DEL LENGUAJE

2.1. Variables

En PHP, las variables se representan con el signo monetario (\$), seguido del respectivo nombre de variable. Utiliza los estándares internacionales para renombrar variables y PHP es sensible a mayúsculas y minúsculas. PHP, por defecto maneja la asignación de variables "por valor", aunque también se pueden manejar variables "por referencia", para esto tiene el



```
<? php
//Declaración de variables
$nombre=" Pedro";
$apellido=" Perez";
$edad=35;
//Salida por pantalla
echo ("Su nombre es $nombre" );
echo ("Su apellido es $apellido");
echo ("Su edad es $edad");
"?>
```

símbolo ampersand (&), para poder manejar las variables por referencia, la variable fuente ya se debió haber declarado, un ejemplo de esto se representa en el siguiente ejemplo:

```
<? php
//Declaración de variables
$salario= 500000;
$incremento= &$salario*0,05; // la variable incremento toma
por referencia el valor de la variable salario y lo
incrementa en 5%
//Salida por pantalla
echo ("El incremento de su salario es $incremento" );
"?>
```

2.2. Obtener información sobre el tipo de una variable

En muchas ocasiones se hace necesario capturar el tipo de una variable, para este proceso PHP cuenta con unas funciones predefinidas, estas funciones empiezan por la palabra `is_`. Si al utilizar dichas funciones, la

respuesta es verdadera, significa que la variable es del tipo que maneja la función, si la respuesta es falsa, es porque la variable es de otro tipo.

Nombre	Tipo de Variable que maneja
<code>is_array()</code>	Si la respuesta es verdadera, la variable es una matriz
<code>is_bool()</code>	Si la respuesta es verdadera, la variable es de tipo booleano
<code>is_float()</code> , <code>is_double()</code> o <code>is_real()</code>	Si la respuesta es verdadera, la variable es de tipo numero con coma flotante
<code>is_int</code> , <code>is_integer</code> o <code>is_long()</code>	Si la respuesta es verdadera, la variable es un numero entero
<code>is_object()</code>	Si la respuesta es verdadera, la variable es un objeto
<code>is_resource()</code>	Si la respuesta es verdadera, la variable es un recurso
<code>is_scalar()</code>	Si la respuesta es verdadera, la variable es de tipo escalar(integer, float, string o booleano)
<code>is_string()</code>	Si la respuesta es verdadera, la variable es una cadena
<code>is_null()</code>	Si la respuesta es verdadera, la variable contiene el valor null
<code>is_numeric()</code>	Si la respuesta es verdadera, la variable es de tipo numero o una cadena numérica
<code>gettype()</code>	Devuelve directamente el tipo de variable que se pasa como argumento

2.3. Tipos de Datos

PHP, al igual que todos los lenguajes de programación soporta varios tipos de datos, estos datos son:

Tipo	Ejemplo
Booleanos	<code>\$Sexo=true;</code>
Enteros	<code>\$num=1234;</code>
Numeros en punto flotante (double)	<code>\$def=4.5;</code>
Cadenas	<code>\$nombre="Carolina"</code>
Objetos	Para inicializar un objeto, se utiliza la sentencia <code>new</code> .

2.4. Operadores

Los operadores se utilizan para realizar operaciones entre objetos, datos, identificadores y/o constantes, y devuelven un valor.

Los operadores en PHP se pueden dividir en las siguientes categorías:

Aritméticos

Operador	Uso	Descripción	Ejemplo
+	$\$op1 + \$op2$	Suma op1 y op2	$5 + 7 = 12$
-	$\$op1 - \$op2$	Resta op1 y op2	$7 - 5 = 2$
*	$\$op1 * \$op2$	Multiplica op1 y op2	$2 * 2 = 4$
/	$\$op1 / \$op2$	Divide op1 y op2	$6 / 3 = 2$
%	$\$op1 \% \$op2$	Obtiene el resto de dividir op1 por op2	$6 \% 3 = 0$
Negación	$\$a$	El opuesto de a	$\$3 = 3$

De Asignación

El operador de asignación básico es el igual (=) y se utiliza para asignar un valor a otro. Mediante la utilización de estos operadores se pueden realizar operaciones aritméticas, lógicas, de bit y de asignación con un único operador.

Supongamos que necesitamos sumar un número a una variable y almacenar el resultado en la misma variable, como a continuación, lo que en algoritmos conocimos como acumulador:

```
$acu = $acu + 2;
```

En PHP se puede abreviar esta sentencia con el operador de atajo +=, de la siguiente manera:

```
$acu += 2;
```

La siguiente tabla muestra los operadores de asignación y su expresión equivalente:

Operador	Uso	Expresión Equivalente	Ejemplo
+=	\$op1 += \$op2	\$op1 = \$op1 + \$op2	para \$num1=3 \$num1 += 2
-=	\$op1 -= \$op2	\$op1 = \$op1 - \$op2	para \$num2=4 \$num2 -= 3
*=	\$op1 *= \$op2	\$op1 = \$op1 * \$op2	para \$mul=5 \$mul *= 2
/=	\$op1 /= \$op2	\$op1 = \$op1 / \$op2	para \$div=10 \$div /= 2
%=	\$op1 %= \$op2	\$op1 = \$op1 % \$op2	para \$mod=5 \$mod %= 2

Operadores de Bit

Operador	Uso	Operación	Ejemplo
>>	\$op1>>\$op2	Desplaza los bits de op1 a la derecha op2 veces	256 >> 4 Retorna: $256/2^4 = 64$. $2^4 = (2*2*2*2) = 16$ $256/16 = 64$
<<	\$op1<<\$op2	Desplaza los bits de op1 a la izquierda op2 veces	16 << 2 Retorna: $16 * 2^2 = 64$ $2^2 = (2*2) = 4$ $16*4 = 64$

Operadores de comparación

Un operador de comparación, como su nombre lo indica compara dos valores y determina la relación existente entre ellos. La respuesta que se obtiene es true o false.

Operador	Uso	Devuelve true
>	\$op1 > \$op2	Si op1 es mayor que op2
>=	\$op1 >= \$op2	Si op1 es mayor o igual que op2
<	\$op1 < \$op2	Si op1 es menor que op2
<=	\$op1 <= \$op2	Si op1 es menor o igual que op2
==	\$op1 == \$op2	Si op1 y op2 son iguales
!=	\$op1 != \$op2	Si op1 y op2 son diferentes

Operadores Lógicos

Su uso radica en la manera como queremos que se evalúe la condición, si se hace necesario que se evalúe las dos partes se utiliza el AND simple (&) o el OR simple (|). El comportamiento del AND y el OR lo muestra la siguiente tabla:

Operador	Nombre	Uso	El resultado
&&	AND	\$op1 && \$op2	True si op1 y op2 son true. False los demás casos. Si op1 es false, YA no se evalúa op2
	OR	\$op1 \$op2	True si op1 u op2 son true. Si op1 es true, YA no se evalúa op2
!	Negación	! \$op	True si op false y false si op es true
&	AND	\$op1 & \$op2	True si op1 y op2 son true. False los demás casos. <u>Siempre</u> se evalúa op2
	OR	\$op1 \$op2	True si op1 u op2 son true. <u>Siempre</u> se evalúa op2

2.5. Estructuras de Control

En PHP, las estructuras de control, funcionan similar a la de otros lenguajes de programación, entre las más comunes tenemos:

Condicional Simple (IF)

Realiza las instrucciones a partir de una decisión evaluada.

Sintaxis:

```
<?php
if (expr)
    Sentencia; // en caso de condición verdadera
else
    Sentencia; // en caso de condición falsa
?>
```

Ejemplo:



```
<!-- Manual de PHP -->
<html>
<head>
  <title>Ejemplo de condicional sencillo</title>
</head>

<body>

Ejemplo de un condicional Sencillo
<BR><BR>

<?php
    echo "Codigo PHP con uso condicional <br><br>";
$a = 8;
$b = 3;
if ($a < $b)
{
    echo "$a es menor que $b";
}
else
{
    echo "$a no es menor que $b";
}
?>

</body>
</html>
```

Ciclo Mientras (WHILE)

Recordemos que en esta estructura, el número de iteraciones del ciclo depende de la condición que se plantee al comienzo del proceso.



Sintaxis:

```

<?php
while (expr) //evalúa la condición
Sentencia; //instrucción que se repite mientras
           // condición sea verdadera
?>

```

El ejemplo a continuación imprime los números del 1 al 10.

```

<! - - Manual de PHP - ->
<html>
<head>
  <title>Ejemplo de while</title>
</head>

<body>

Ejemplo de un while
<BR><BR>

<?php
  echo "Codigo PHP con uso de while <br><br>";
  $i=0;
  while ($i<10)
  {
    echo "El valor de i es"; $i, <br>
    $i++;
  }
?>

</body>
</html>

```

Ciclo Haga mientras (DO ... WHILE)

Este bucle es similar al While, lo que los diferencia es que la condición se evalúa hasta el final de cada iteración y no al comienzo.

Recordemos que en esta estructura, el número de iteraciones del ciclo depende de la condición que se plantee al final del proceso

Sintaxis:

```
<?php
do
    Sentencia;
While (expr) // evalúa condición y repite ciclo en
//caso de ser verdadera
?>
```

```
<!-- Manual de PHP -->
<html>
<head>
    <title>Ejemplo de dowhile</title>
</head>
```

```
<body>
```

Ejemplo de uso de dowhile

```
<BR><BR>
```

```
<?php
    echo "Codigo PHP con uso de do while <br><br>";
    $i=1;
    do{
        echo "El valor de i es"; $i, <br>";
        $i=$i+1;
    }
    while ($i<=5)
?>

</body>
</html>
```



Ciclo repita n veces (FOR)

Recordemos que en esta estructura el número de iteraciones se conoce antes de ejecutarse el ciclo.

Sintaxis:

```
<?php  
for (expr1; expr2; expr3)  
    Sentencia;  
  
?>
```

Dónde:

expr1 inicializa el contador

expr2 determina el número de repeticiones

expr3 establece el incremento en cada repetición



```
<! - - Manual de PHP - ->
<html>
<head>
    <title>Ejemplo de Uso For</title>
</head>

<body>
```

Ejemplo de un ciclo

```
<BR><BR>
```

```
<?php
    echo "Codigo PHP con uso FOR <br>";

    for($i=0;$i<10;$i++)
    {
        echo "Linea "; . $i. "<br>";
    }
?>

</body>
</html>
```

Decisión basada en múltiples opciones (SWITCH)

También es conocida como case, y se utiliza para evaluar el contenido de una variable y determinar diferentes acciones a ejecutar dependiendo de ese valor, que puede ser cualquier expresión que evalúe a un tipo simple.



Sintaxis:

```
<?php
switch (variable_a_evaluar)
{
    case valor1:
        sentencias;
        break;
        case valor2:
            sentencias;
            break;
    case valorn:
        sentencias;
        break;
    default // si no corresponde con ninguno de los
            // valores anteriores
        sentencias ;
}
?>
```

Ejemplo:




```
<!-- Manual de PHP -->
<html>
<head>
  <title>Ejemplo de Uso switch</title>
</head>
```

```
<body>
```

```
Ejemplo de un switch
```

```
<BR><BR>
```

```
<?php
  $dia = 12;
```

como la variable tiene el valor 12, la salida del sistema se irá por el bloque default.

```
    switch($dia) {
    case 1: // Bloque 1
      echo "El dia es Lunes";
      break;
    case 2: // Bloque 2
      echo "El dia es Martes";
      break;
    case 3: // Bloque 3
      echo "El dia es Miercoles";
      break;
    case 4: // Bloque 4
      echo "El dia es Jueves";
      break;
    case 5: // Bloque 5
      echo "El dia es Viernes";
      break;
    case 6: // Bloque 6
      echo "El dia es Sabado";
      break;
    case 7: // Bloque 7
      echo "El dia es Domingo";
      break;
    case 8: // Bloque 8
      echo "La variable contiene otro valor";
      echo "distinto a los dias de la semana";
    }
  ?>
```

```
</body>
</html>
```



3. ENVÍO Y RECEPCIÓN DE DATOS EN PHP

Al diseñar un formulario, se debe indicar el nombre de la página PHP, que procesará la información que se digita en el formulario, junto con el método por el que se le pasará la información a la página encargada de procesarla.

Un ejemplo de esto sería:

Página 1, para inserción de Datos

Iniciando el manejo de formularios

Introduzca su nombre:

Cuando se digite el nombre en la caja de texto, y se haga clic en el botón enviar, este dato se pasará a otra página, que se convertiría en la página de respuesta.

Página 2, para visualización de los datos

Iniciando el manejo de formularios

El nombre que ha introducido es: Catrine Rojas

El procedimiento para realizar este ejercicio, sería el siguiente:

A. Construir una página PHP, para este ejemplo se va a utilizar el nombre `Frmingreso.php`, con el siguiente código:

```

<! -- Manual de PHP -->
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Iniciando el manejo de formularios</H1>
Introduzca su nombre:

```

```

      1          2          3
<FORM ACTION= "FrmpIngreso.php" METHOD="GET">
4  <INPUT TYPE="text" NAME="nombre"><BR>
      5
  <INPUT TYPE="submit" VALUE="enviar">
</FORM>
</body>
</html>
</html>

```

Resaltemos los siguientes aspectos a tener en cuenta:

1. Se debe manejar una acción al formulario, es decir informar la página a donde se van a enviar los datos para ser visualizados.
2. El nombre de la página debe escribirse dentro de comillas, y con la extensión correspondiente, en este ejemplo es "FrmpIngreso.php".
3. Método, para el ejemplo se utilizará el get
4. Input: las diferentes opciones del input, se trabajó en HTML, para nuestro caso, recordemos que tenemos una caja de texto.
5. El input donde se digitará el dato a enviar tiene como name la palabra "nombre", el nombre de esta variable es muy importante recordarlo, porque será el enlace entre el formulario de inserción de datos (FrmpIngreso.php) y el de salida (FrmpIngreso.php)



B. Después de construir la página de inserción de Datos, se procede a codificar la página que visualizará los datos que previamente se incluyeron en el punto A, esta página también sería de tipo PHP, para este ejemplo se va a utilizar el nombre `FrmIngreso.php`, con el siguiente código:

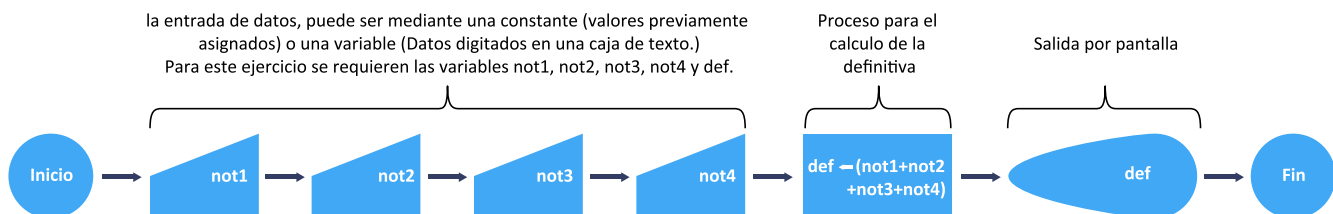
```
<!-- Manual de PHP -->
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
  <H1>Iniciando el manejo de formularios</H1>
  El nombre que ha introducido es: <?php echo $_GET['nombre'] ?>
  <br>
</body>
</html>
```

este es el nombre de la variable que se utilizó en la página de inserción de datos (`FrmIngreso.php`).

C. Recuerde que estas dos páginas deben estar ubicadas en la carpeta `xampp\htdocs`.

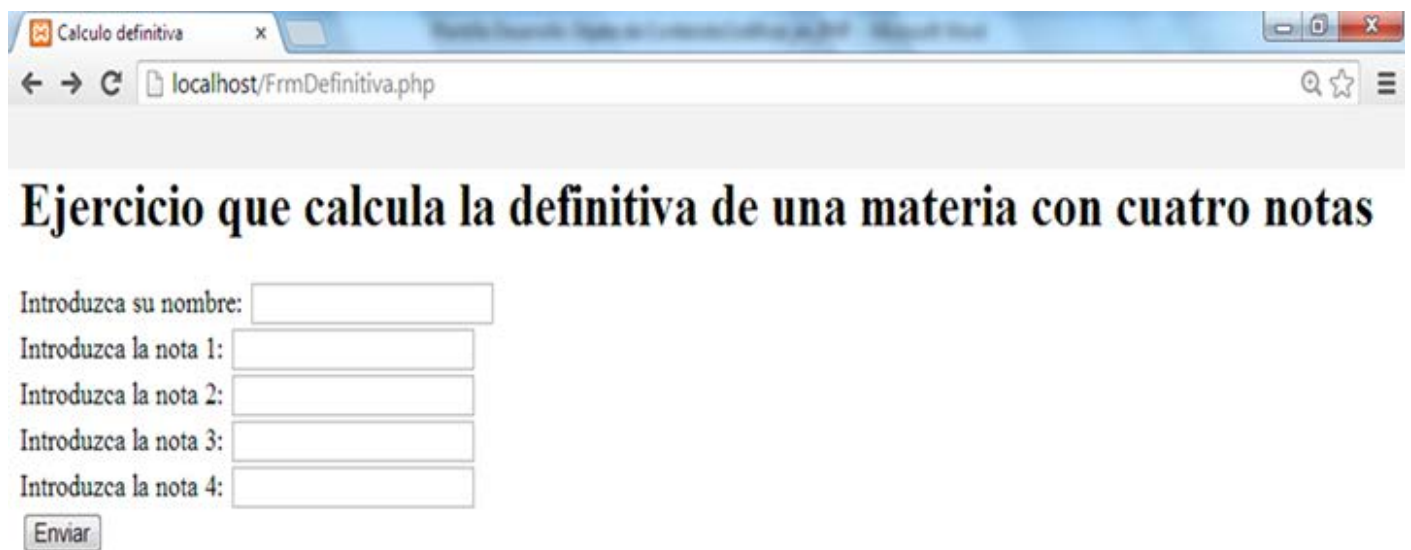
3.1. Ejemplo con operadores básicos en PHP

Para este ejemplo vamos a tomar un algoritmo básico, calcula la definitiva de una materia que tiene cuatro números, estas cuatro notas son digitadas en cajas de texto, y el resultado de la definitiva se visualizará por pantalla, a continuación el algoritmo de este ejercicio.



El proceso para realizar la respectiva codificación en PHP, sería:

a. Construir una página PHP, se propone como nombre FrmDefinitiva.php, en esta página se debe incluir el código necesario para construir las cajas de texto donde se digitaran las cuatro notas, el diseño de esta página podría ser:



The screenshot shows a web browser window with the title 'Calculo definitiva'. The address bar shows 'localhost/FrmDefinitiva.php'. The page content includes the title 'Ejercicio que calcula la definitiva de una materia con cuatro notas' and a form with the following fields:

- Introduzca su nombre:
- Introduzca la nota 1:
- Introduzca la nota 2:
- Introduzca la nota 3:
- Introduzca la nota 4:
-

Una propuesta del código para llegar a la página podría ser:


```

<!-- Manual de PHP -->
<html>
<head>
  <title>Calculo definitiva</title>
</head>
<body>
<H1>Ejercicio que calcula la definitiva de una materia
con cuatro notas</H1>
  1
  2
  <FORM ACTION= "FrmpDefinitiva.php" METHOD= "post">
    Introduzca su nombre:
    <INPUT TYPE="text" NAME="nombre"><BR>
    Introduzca la nota 1:
    <INPUT TYPE="text" NAME="nota1"><BR>
    Introduzca la nota 2:
    <INPUT TYPE="text" NAME="nota2"><BR>
    Introduzca la nota 3:
    <INPUT TYPE="text" NAME="nota3"><BR>
    Introduzca la nota 4:
    <INPUT TYPE="text" NAME="nota4"><BR>
    <INPUT TYPE="submit" VALUE="Enviar">
  </FORM>
</body>
</html>
  3

```

Línea	Explicación
1	El action del formulario nos llevará a la página de resultados, en este ejemplo se llamará FrmpDefinitiva.php (Frm, de formulario, p, de presentación, Definitiva, nombre del proceso que se realiza).
2	Method, para este caso se utilizó el método POST. La diferencia entre el método get y el Post, es que este último, no permite ver el nombre de las variables que se están utilizando, al llamar la página FrmpDefinitiva.php, esto facilita la seguridad de los datos.
3	Corresponde a la construcción de las cajas de texto, y a la asignación de nombres para cada una de ellas. RECUERDE que este nombre es el que se utilizará al llamar las variables en la página de presentación de datos (FrmpDefinitiva.php).



b. Construir la página PHP, que visualizará la definitiva, una vez realizado el respectivo cálculo. Una propuesta para la codificación de esta página sería:

```
<!-- Manual de PHP -->
<html>
<head>
  <title>Calculo definitiva</title>
</head>
<body>
<H1>Calculo definitiva</H1>
```

1

```
<?php echo $_POST ['nombre'] ;
```

```
$nota1= $_POST ['nota1'];
$nota1= $_POST ['nota2'];
$nota1= $_POST ['nota3'];
$nota1= $_POST ['nota4'];
```

2

3

```
$def= ($nota1+$nota2+$nota3+$nota4)/4;
```

<?

4

Su definitiva es: <?php echo \$def ?>

```
<br>
</body>
</html>
```

Línea	Explicación
1	Iniciamos el código php y traemos la variable nombre, que se declaró en la página FrmDefinitiva.php
2	Declaramos las variables nota1, nota2, nota3, nota4 y le asignamos los datos que vienen de las cajas de texto creadas en la página FrmDefinitiva.php
3	Realizamos el proceso para calcular la definitiva, aplicando prioridades de operadores en la formula.
4	Se vincula el código PHP, para que se visualice por pantalla la definitiva.



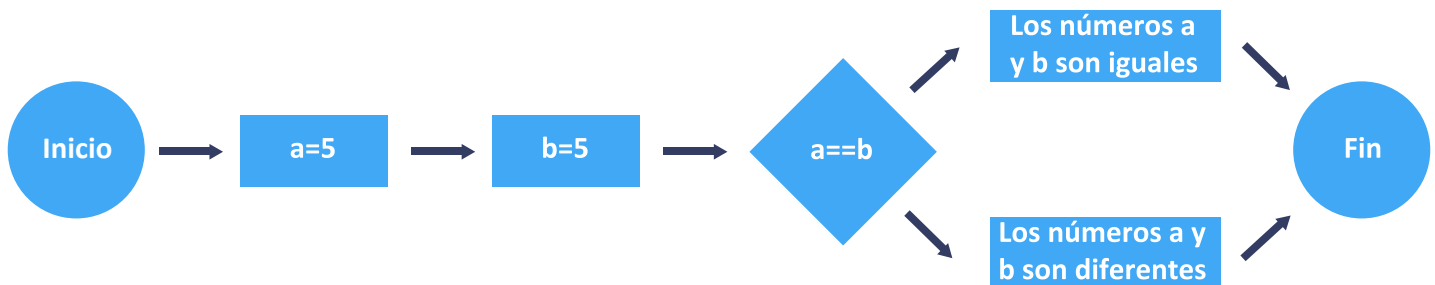
La salida en el sistema probablemente sería la siguiente, para el caso que digitáramos los siguientes datos: nombre: Patricia Rojas; nota1: 4,3; nota 2: 4,1; nota3: 3,9; nota4: 3,5.

Calculo definitiva

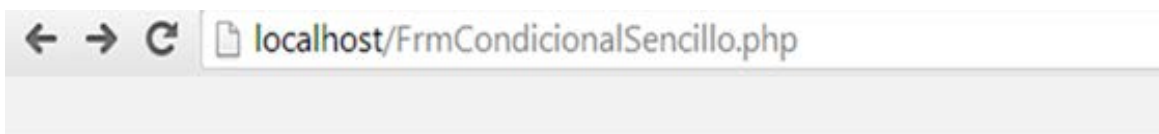
Patricia Rojas Su definitiva es: 3.5

3.2. Ejemplo de condicionales sencillos en PHP

Para este ejemplo se tomará como referente el diagrama de flujo, que compara dos números, e informa si éstos son iguales o diferentes, el procedimiento es similar al anterior ejercicio, una página para la inserción de datos y otra para la respuesta del sistema.



Los componentes básicos para el diseño de este formulario, podrían ser:



Ejercicio que compara dos numeros

Introduzca el numero 1:

Introduzca el numero 2:

La codificación básica para construir esta página podría ser:

```
<! - - Manual de PHP - ->
<html>
<head>
  <title>Condicionales Sencillos</title>
</head>
<body>
<H1>Ejercicio que compara dos numeros</H1>
<FORM ACTION= "FrmpCondicionalSencillo.php" METHOD= "post">
  Introduzca el numero 1:
  <INPUT TYPE="text" NAME="num1"><BR>
  Introduzca el numero 2:
  <INPUT TYPE="text" NAME="num2"><BR>
  <INPUT TYPE="submit" VALUE="Enviar">
</FORM>
</body>
</html>
</html>
```

Diagrama de anotaciones:

- 1: `"FrmpCondicionalSencillo.php"`
- 2: `METHOD= "post">`
- 3: `<INPUT TYPE="text" NAME="num1">
` y `<INPUT TYPE="text" NAME="num2">
`

Se debe recordar que en la línea 1, se coloca el nombre de la página php que va a realizar los procesos y a visualizar los resultados, en nuestro caso `FrmpCondicionalSencillo.php`; en el ítem 2, el método a utilizar, para este caso `post`, por el manejo de seguridad de la información; y en el ítem 3, la construcción y asignación de nombres a las cajas de texto.

La codificación de la página de salida podría ser:

```
<! -- Manual de PHP -->
<html>
<head>
  <title>Comparación de dos numeros</title>
</head>
<body>
<H1>Ejercicio que lee dos numeros y los compara</H1>
```

```
<?php
```

```
$num1= $_POST ['num1'];
$num2= $_POST ['num2'];
```

Declaración de variables y asignación de valores, digitados en la pagina de inserción de datos.

```
if ($num1==$num2)
{
  mensaje= "iguales";
}
else
{
  mensaje= "Diferentes";
}
```

```
?>
```

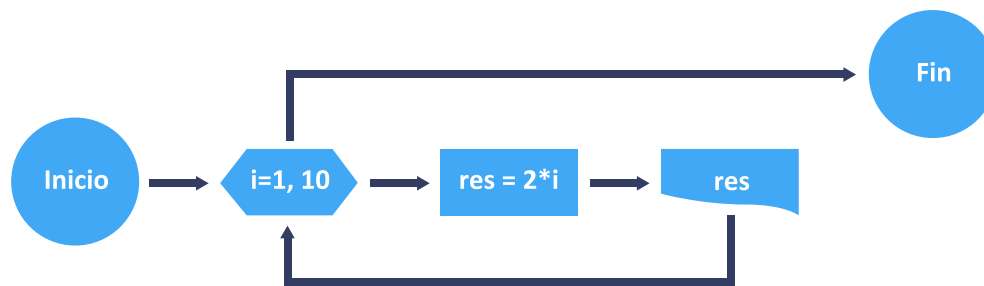
```
Los numero son: <?php echo $mensaje ?>
```

Respuesta del Sistema

```
<br>
</body>
</html>
```

3.3. Ejemplo de la estructura cíclica hacer para en PHP

Recordemos que en esta estructura el número de iteraciones se conoce antes de ejecutarse el ciclo, tomemos como referencia el siguiente diagrama de flujo:



Como se puede observar, este diagrama no tiene ningún dato de entrada por teclado, por lo tanto, este ejemplo solo tiene una página php, para visualizar los datos.

```
<! - - Manual de PHP - ->
<html>
<head>
    <title>Manejo del FOR en PHP</title>
</head>
<body>
<H1>Ejercicio que visualiza la tabla del 2</H1>

<?php
for ($i=1;$i<10;$i++)

{
$res = 2*$i;

echo "2 * " . $i . " = " . $res. "<br>";

}
?>
</body>
</html>
```

El operador punto(.), permite unir las variables, con el rotulo de la salida, es decir el texto que esta entre comillas.

La salida por pantalla sería de la siguiente manera:



Ejercicio que visualiza la tabla del 2

```

2 * 1=2
2 * 2=4
2 * 3=6
2 * 4=8
2 * 5=10
2 * 6=12
2 * 7=14
2 * 8=16
2 * 9=18
2 * 10=20

```

3.4. Ejemplo del manejo de un control combobox en PHP

```

<!-- Manual de PHP -->
<html>
<head>
    <title>Manejo de combobox</title>
</head>
<body>
<H1>Manejo de combobox en PHP</H1>

<FORM ACTION= "FrmpCombo.php" METHOD="post">

Introduzca su nombre:
<INPUT TYPE="text" NAME="nombre"><BR>
Gustos: <br>
<select multiple name="gustos[]">
<option value="Bailar">Bailar</option>
<option value="Cine">Ir a cine</option>
<option value="Nadar">Nadar</option>
<option value="Cantar">Cantar</option>
</select><br>
Dirección
<INPUT TYPE="text" NAME="dirección"><BR>
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
</body>
</html>
</html>

```

Manejo de combobox en PHP

Con la palabra reservada select, se informa al sistema que se va a construir un combo, y después de asignar el nombre al combo se incluyen [], para indicar que se va a incluir un conjunto de valores.

El siguiente fragmento de código permite enviar el dato que se seleccionó en el combo y que la página de visualización lo reciba correctamente:

```
<!-- Manual de PHP -->
<html>
<head>
    <title>Calculo Nuevo Salario</title>
</head>
<body>
<H1>Incremento Salario</H1>
<?php echo $_POST 1['nombre'] ;
echo "Usted viden en". $direccion= $_POST
['direccion'];
$gusto=$_POST['gustos'];
for ($i0=;$i<count($gusto);$i++)
{
echo "<br>El gusto que usted selecciono
fue:". $gusto [$i];
} ?>

<br>
</body>
</html>
```

Note que debe recorrer el arreglo gusto, para poder visualizar la información seleccionada previamente al combo



Glosario

Htdocs: Carpeta donde deben estar ubicados todos los archivos PHP, para que éstos puedan ser ejecutados

HTML: Hyper Text Markup Language. Lenguaje que sirve para escribir hipertexto a través de un estándar que define una estructura básica y código para definir el contenido de una página web.

LAMP: Servidor que contiene un entorno que soporta PHP, que incluye Apache y MySQL diseñado para que corra en sistemas operativos Linux.

MAMP: La abreviatura corresponde a Macintosh, Apache, PHP y MySQL, Este servidor que contiene un entorno que soporta PHP, diseñado para que corra en sistemas operativos Mac OS x.

WAMP: Servidor que contiene un entorno que soporta PHP, que incluye Apache y MySQL diseñado para que corra en sistemas operativos Windows.

XAMPP: Herramienta informática que tiene todo lo necesario para que con una sola instalación tengamos un servidor que soporte PHP, incluye Apache, MySQL y está disponible para sistemas operativos Microsoft Windows, GNU/Linux, Solaris y MacOS X.



Bibliografía

Álvarez, Miguel Ángel. Qué es PHP?. Consultada el 26/08/2013 en <http://www.desarrolloweb.com/articulos/392.php>,

Vásquez Mariño, Carlos. Programación en PHP5. Nivel Básico. Ferrol, disponible en: http://administraciondesistemas.pbworks.com/f/Manual_PHP5_Basico.pdf



CRÉDITOS

OBJETO DE APRENDIZAJE FUNDAMENTOS DE PROGRAMACIÓN CON PHP

Desarrollador de contenido Experto temático	Magda Milena García Gamboa
Asesor Pedagógico	Rafael Neftali Lizcano Reyes
Productor Multimedia	Rafael Ricardo Valdés Prada Yessenia A. Cárdenas Rodríguez
Programador	Daniel Eduardo Martínez Díaz
Líder expertos temáticos	Ana Yaqueline Chavarro Parra
Líder línea de producción	Santiago Lozada Garcés



Atribución, no comercial, compartir igual

Este material puede ser distribuido, copiado y exhibido por terceros si se muestra en los créditos. No se puede obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de licencia que el trabajo original.

Creative Commons



El logo de PHP® es una marca registrada.

Registered trademark