Universidad Don Bosco
FACULTAD DE INGENIERÍA
ESCUELA DE COMPUTACIÓN
Lenguaje Interpretado en el Servidor
Guía 1: Fundamentos de Programación en PHP

I. OBJETIVOS

En esta guía de práctica se pretende:

1. Conseguir que los estudiantes tengan un primer contacto con la programación y sintaxis de PHP.
2. Desarrollar las habilidades necesarias para crear secuencias de comando con PHP.
3. Lograr el dominio de los tipos de datos del lenguaje PHP.
4. Dominar la forma en que se crearán scripts PHP y del proceso para visualizarlos a través de un navegador.

II. INTRODUCCIÓN TEÓRICA

Definición de PHP

PHP es, hoy en día, un lenguaje de programación diseñado para desarrollar páginas web dinámicas que son ejecutadas en un servidor web y, luego, devueltas en formato HTML al navegador del usuario que las solicita. No obstante, en sus principios PHP fue más bien un lenguaje de secuencias de comando, guiones o scripts del lado del servidor desarrollado por Rasmus Lerdorf en 1995 con el propósito de crear un conjunto de scripts que le permitieran contabilizar el número de visitantes que accedían a su hoja de vida que había puesto en línea para conseguir empleo o contratos de trabajo.

Debido a que PHP estaba diseñado para trabajar en un ambiente web y que este se podía insertar directamente dentro del código (X)HTML propició que se volviera muy popular para procesar datos ingresados a través de formularios.

El significado actual de PHP es Hypertext PreProcessor y a partir de la versión 4.0 es considerado todo un lenguaje de programación y una plataforma sólida para el desarrollo de aplicaciones web del lado del servidor.

Evolución de PHP

Rasmus Lerdorf desarrolló inicialmente PHP en Perl y luego en C, liberando la versión PHP/FI 2.0 en 1997. En ese momento, PHP era utilizado en unos 50,000 dominios. En 1998, se lanzó PHP 3.0, desarrollado por Andi Gutmans y Zeev Zurasky, con mejor extensibilidad y soporte para bases de datos, alcanzando un 10% de uso en servidores. PHP 4.0, lanzado en 2000, introdujo un nuevo motor Zend, mejor rendimiento, soporte para orientación a objetos y seguridad mejorada. En 2007, se suspendió el soporte para PHP 4, pero se continuaron lanzando actualizaciones de seguridad. PHP 5.0, lanzado en 2004, mejoró el soporte para objetos, rendimiento, MySQL, XML, SQLite, SOAP, iteradores y excepciones.

Mejoras en la sintaxis de PHP 7.2

PHP 7.2 es una actualización significativa de este popular lenguaje de programación, que mejora el rendimiento y la seguridad, permitiendo a los desarrolladores escribir código más eficiente y crear aplicaciones web de mayor calidad. Esta nueva versión introduce una serie de cambios que optimizan tanto la programación como la seguridad.

Una de las principales mejoras en PHP 7.2 está relacionada con los *type hints* (indicadores de tipo) en la declaración de argumentos. Ahora, los desarrolladores podrán declarar objetos genéricos como argumentos o métodos, lo que facilita el paso de información entre ellos y mejora la claridad del código.
Además, PHP 7.2 incluye una funcionalidad en la que las funciones correctamente declaradas especificarán el tipo de variable esperado, incluso si no se ha declarado explícitamente en el código. También se permite omitir el tipo de una subclase sin que esto rompa el código, lo que otorga mayor flexibilidad al desarrollo.
En cuanto a seguridad, PHP 7.2 se enfoca en este aspecto al ser el primer lenguaje de programación que incluye bibliotecas criptográficas modernas por defecto, sin necesidad de APIs externas, lo que fortalece la protección de datos.

PHP es ampliamente utilizado para:

- Desarrollar aplicaciones web del lado del servidor: Esta es la aplicación más común de PHP y la que le ha valido su popularidad.
- Ejecutar scripts desde la línea de comandos: Estos scripts se pueden ejecutar sin necesidad de un servidor web o navegador.
- Crear aplicaciones con interfaces gráficas: PHP también se puede usar para crear aplicaciones con interfaz gráfica, utilizando la extensión PHP-GTK, que debe ser instalada adicionalmente.
- Tipos de aplicaciones que se pueden desarrollar con PHP

PHP es ideal para crear aplicaciones como:

- Comercio electrónico
- Educación a distancia
- Foros de discusión
- Sistemas de gestión de contenidos (CMS)
- Blogs
- Exámenes en línea
- Aplicaciones de correo electrónico

Requerimientos para desarrollar aplicaciones con PHP
Para poder realizar scripts de PHP en el lado del servidor es necesario tener instalado:

- ✪ El intérprete de PHP (CGI o módulo).
- ✪ Un servidor web (Apache Web Server es ideal para trabajar con PHP; sin embargo, hoy en día se puede instalar fácilmente en Internet Information Server también).
- ✪ Un navegador web (Internet Explorer, Chrome, Firefox, Safari y Opera son los más difundidos).
- ✪ Un gestor de bases de datos (MySQL es la mejor opción de base de datos para trabajar con PHP).
- ✪ Un editor de texto, de preferencia especializado en sintaxis de PHP.

Sintaxis básica.
El lenguaje PHP es bastante sencillo en cuanto a su sintaxis. Alguien con experiencia en programación con lenguaje C o Perl, no debería tener ningún problema de adaptación. Sin embargo, hay que decir que es más complicado que simplemente escribir código HTML.

Delimitadores de bloque de código PHP.
Existen cuatro diferentes tipos de delimitadores de código PHP. Estos son:

a) Delimitadores estilo XML:

```
<?php
  //instrucciones php
```

```
?>
```

b) Delimitador estilo script:

```
<script language="php">
  //instrucciones php
</script>
```

c) Delimitador corto:

```
<?
  //instrucciones php
?>
```

d) Delimitador estilo ASP:

```
<%
  //instrucciones php
%>
```

Hay que mencionar que solamente los primeros dos tipos de delimitadores están disponibles de forma automática, sin necesidad de realizar configuración alguna en el archivo php.ini. Los últimos dos delimitadores deben ser habilitados en dicho archivo de configuración. Este archivo está en la carpeta de instalación de PHP y deberá modificarlo si necesita utilizar etiquetas cortas o las de estilo ASP y reiniciar los servicios del Wamp para que funcionen.

Delimitador de sentencias.

El terminador o delimitador de sentencias en PHP es el punto y coma (;), el mismo que se utiliza en C/C++. Por lo tanto, cuando desee terminar una sentencia para iniciar otra debe utilizar el punto y coma. Existen dos casos en los que se puede omitir el punto y coma. Uno es cuando sólo existe una instrucción PHP en el script y el otro es cuando la instrucción sea la última línea del bloque de código PHP. Se sugiere que siempre utilice el punto y coma al final de cualquier instrucción, incluso en los casos antes mencionados para evitar cometer errores por tratar de recordar estos casos especiales. Es más fácil recordar que siempre debe utilizarse que recordar cuando puede omitirse.

Comentarios.

PHP utiliza los estilos de comentarios del lenguaje C/C++ y del *shell* de la interfaz de comandos de Unix y Linux. Estos son el comentario de una sola línea "//" y el comentario de bloque "/* … */", también utilizados en el lenguaje C. Además, se puede utilizar el comentario de una sola línea utilizado en el *shell* de Unix y Linux, "#"

Veamos algunos ejemplos:

1) Comentario de una sola línea "//" estilo C++:

```
<?php
  $valor = 5.2;
  //Este es un comentario de una sola línea estilo C
  echo "El valor es: " . $valor;
?>
```

2) Comentario de una sola línea "#" estilo *shell*:

```
<?php
  $nombre = "Julio";
  #Este es un comentario de una sola línea estilo shell
  echo "Su nombre es:. " . $nombre;
```

```
?>
```

3) Comentario de bloque (varias líneas) "/* … */":

```
<?php
  $precio = 25;
  $total = $precio * $POST['descuento'];
  /*  En este caso estamos utilizando un comentario
     de bloque, esto quiere decir que todo este texto
     que está leyendo es comentario y que por tanto será
     ignorado por el intérprete de PHP
  */
  echo "El total es " . $total;
?>
```

Salida a pantalla.

En PHP existen dos formas principales de mandar a imprimir texto en la ventana de un navegador. La primera es utilizando la instrucción *echo* y la segunda es utilizar la función *printf*().

echo

La sentencia *echo* es fácil de utilizar, su sintaxis es la siguiente:

```
echo cadena_de_texto;
```

Donde, *cadena_de_texto* puede ser un literal de cadena delimitado por comillas que pueden ser simples (') o dobles ("). La diferencia más importante es que entre comillas dobles se interpretan las variables y ciertos caracteres especiales, incluyendo etiquetas HTML. En cambio, con comillas simples sólo se interpretan la comilla simple y la barra invertida, por tanto, para evitar que sean interpretados estos caracteres deberá hacer uso de secuencias de escape.

printf()

La función *printf()* es mucho más versátil que la instrucción *echo*. Con esta función se pueden mandar a imprimir varios tipos de variables a la vez, utilizando códigos de formato, que indican cómo debe ser formateada la variable que se desea mostrar a la salida. La sintaxis es la siguiente:

```
printf("cadena_de_texto [%s %d %f %c]", $cadena, $entero, $flotante, $caracter);
```

Donde, *cadena_de_texto* es una cadena delimitada por comillas dobles que puede incluir ciertos códigos de formato opcionales. Si se desea imprimir el contenido de variables deben especificarse códigos de formato para formatear la salida adecuadamente. Los códigos de formato más importantes son:

Elemento	Tipo de variable
%s	Cadena de caracteres
%d	Número sin decimales
%f	Número con decimales
%c	Carácter ASCII
Aunque existen otros tipos, estos son los más importantes.	

Variables

Las variables en PHP se definen anteponiendo el símbolo dólar ($) al nombre de la variable. A diferencia de otros lenguajes, PHP posee una gran flexibilidad a la hora de operar con variables. En efecto, cuando definimos una variable asignándole un valor, PHP le atribuye un tipo. Si por ejemplo definimos una variable entre comillas, la variable será considerada de tipo cadena:

```
$variable = "5";    //esto es una cadena
```

Ahora bien, si en el script se realiza una operación matemática con esta variable, no se lanzará ningún mensaje de error sino que la variable cadena será convertida automáticamente en numérica al incluirla en una expresión que involucre un operador matemático:

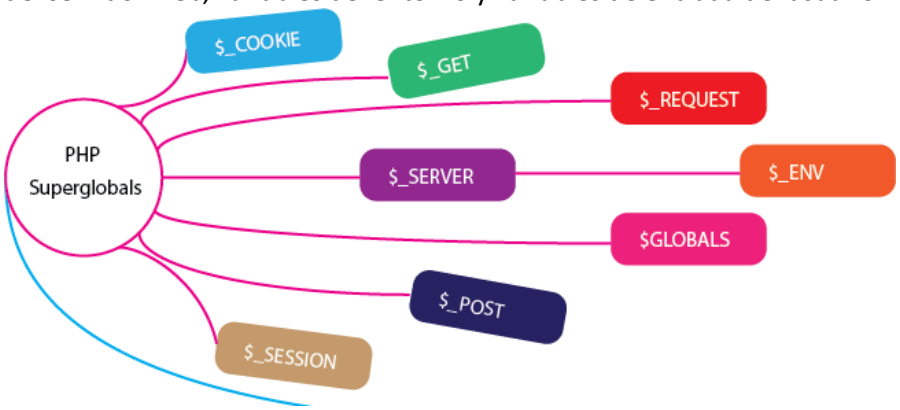
```
<?php
  $cadena = "5";     //esto es una cadena
  $entero = 3;       //esto es un entero
  echo $cadena + $entero
?>
```

Este *script* dará como resultado "8". La variable cadena con valor de "5", ha sido asimilada como entero (aunque su tipo sigue siendo cadena) para poder realizar la operación matemática. Del mismo modo, podemos operar entre variables tipo entero y real. No debemos preocuparnos de nada, PHP se encarga durante la ejecución de interpretar el tipo de variable necesario para el buen funcionamiento del programa.

Sin embargo, si hay que tener cuidado en no cambiar mayúsculas por minúsculas en el identificador de la variable, ya que, en este sentido, PHP es sensible. Conviene por lo tanto, trabajar ya sea siempre en mayúsculas, o siempre en minúsculas para evitar este tipo de malentendidos a veces muy difíciles de localizar. Durante las prácticas de laboratorio se convendrá que los nombres de **variables** se digitarán siempre **en minúsculas** y las **constantes en mayúsculas**.

Variables predefinidas de PHP

Estas son variables que están disponibles para cualquier script PHP que se ejecute en un servidor web con el módulo PHP instalado. Algunas de ellas pueden ser muy útiles para obtener información del cliente o del mismo servidor. El comportamiento y disponibilidad de estas variables depende del servidor sobre el que se estén ejecutando, específicamente de su configuración, de la versión de PHP y de otros factores.
A partir de la versión 4.1.0 PHP dispone de un conjunto de matrices predefinidas que contienen variables del servidor web, variables del entorno y variables de entrada del usuario.



Estas matrices son automáticamente globales o, también llamadas, superglobales. Entre estas matrices se pueden mencionar:

$GLOBALS, es una matriz asociativa que contiene una referencia a cada variable disponible en el ámbito de las variables globales del script. La forma de acceder a las variables es utilizando el nombre de las variables globales entre comillas (dobles o simples) como índice de la matriz.

$_SERVER, es una matriz asociativa que contiene información como cabeceras, rutas y ubicaciones de scripts. Las entradas de esta matriz se crean en el servidor web. No hay garantía alguna de que el servidor vaya a proveer estos valores realmente. Dentro de las entradas que pueden encontrarse en esta matriz se pueden mencionar:
'PHP_SELF': que proporciona el nombre del archivo de script ejecutándose actualmente, relativo a la raíz del documento.
'SERVER_ADDR': que proporciona la dirección IP del servidor web en el que se está ejecutando el script actual.
'SERVER_NAME': que proporciona el nombre del servidor web bajo el que está siendo ejecutado el *script* actual. Si se ejecuta en un host virtual devolverá el nombre definido para tal host.
'SCRIPT_FILENAME': que proporciona la ruta absoluta del nombre del *script* que está siendo ejecutado actualmente.

Existen muchas más entradas para la matriz $_SERVER que pueden consultar en el manual oficial de PHP.

$_GET, que es una matriz asociativa que contiene variables proporcionadas al script por medio del método HTTP GET. Esto significa que cuando define que las variables de un formulario serán pasadas por el método GET, es en esta matriz donde se almacenarán sus valores de acuerdo al nombre que asignó al control de formulario HTML.

$_POST, es una matriz asociativa que contiene las variables pasadas al script a través de método HTTP POST. Al igual que $_GET, cuando se define que el método de paso de valores provenientes de un formulario será POST, la matriz contendrá dichos valores y para tener acceso a ellos deberá usar como llave de la matriz el nombre que le dio al control de formulario.

$_COOKIE, es una matriz asociativa que contiene las variables pasadas al script mediante cookies HTTP.

$_SESSION, es una matriz asociativa que contiene las variables de sesión disponibles en el script actual.

$_REQUEST, es una matriz asociativa que contiene cualquiera de los contenidos de las matrices superglobales $_GET, $_POST y $_COOKIE.

Existen otras matrices superglobales que se dejan como investigación al estudiante.

Ejemplo:

```
<?php
  $cad  = "El script que est&aacute;s ejecutando: " . $_SERVER['PHP_SELF'] . ". ";
  $cad .= "En el servidor: " . $_SERVER["SERVER_NAME"] . ".<br>";
  echo "<h3>" . $cad . "</h3>";
?>
```

Constantes.
Sintaxis
Se puede definir una constante utilizando la función *define()*. Una vez definida, no se puede modificar ni eliminar.
Sólo se puede definir como constantes valores escalares (boolean, integer, float y string).
Para obtener el valor de una constante únicamente es necesario especificar su nombre. A diferencia de las variables, no se tiene que especificar el prefijo $. También se puede utilizar la función *constant()*, para obtener el valor de una constante, en el caso de que queramos expresarla de forma dinámica Usa la función *get_defined_constants()* para obtener una lista de todas las constantes definidas.

Nota: Las constantes y las variables (globales) se encuentran en un espacio de nombres distinto. Esto implica que por ejemplo TRUE y $TRUE son diferentes.

Cuando se utiliza una constante que todavía no ha sido definida, PHP asume que se está refiriendo al nombre de la constante en sí. Se lanzará un aviso si esto sucede. Usa la función *define()* para comprobar la existencia de dicha constante.
Estas son las diferencias entre constantes y variables:

- Las constantes no son precedidas por un símbolo de dólar ($)
- Las constantes sólo pueden ser definidas usando la función *define()*, nunca por simple asignación.
- Las constantes pueden ser definidas y accedidas sin tener en cuenta las reglas de alcance del ámbito.
- Las constantes no pueden ser redefinidas o eliminadas después de establecerse; y
- Las constantes solo pueden albergar valores escalares.

Ejemplo. Definiendo constantes

```
<?php
  define("CONSTANT", "LIS.");
  echo CONSTANT;        // muestra el mensaje "LIS."
  echo "<br>", Constant; // muestra "Constant".
?>
```

Tratamiento de cadenas
Existen tres formas de asignar cadenas a una variable en PHP, que son: delimitándolas entre comillas dobles, entre comillas simples y usando delimitadores tipo Perl (HereDoc).

Para asignar a una variable una cadena **usando comillas dobles** debe hacer una declaración de este tipo:
$cadena = "Esta es la información de mi variable";

Para mostrar el valor de una variable pueden usarse la instrucción *echo* o la función *print()*:
echo $cadena //obtendríamos: Esta es la información de mi variable;
echo "Esta es la información de mi variable"; //daría el mismo resultado

Algo importante con respecto a los delimitadores de comillas dobles es que interpretan variables si son colocadas dentro de comillas dobles. En terminología de programación, se dice que son interpoladas. Por ejemplo:

```
<?php
   $cadena1 = "Lenguajes";
   $cadena2 = " Interpretados en el Servidor";
   $cadena3 = "Materia: $cadena1 $cadena2";
   echo $cadena3          //El resultado es: Lenguajes Interpretados en el Servidor
?>
```

También podemos introducir variables dentro de nuestra cadena lo cual nos puede ayudar mucho en el desarrollo de nuestros scripts. Lo que veremos no es el nombre, sino el valor que almacena la variable:

```
<?
   $a=22;
   $mensaje = "Tengo $a años";
   echo $mensaje //El resultado es: "Tengo 22 años"
?>
```

Puede ser que, en lugar de imprimir el valor de la variable, lo que se desee es imprimir el nombre mismo de la variable. Al colocarlo entre comillas dobles, como se hizo en el ejemplo anterior, no sería posible. La única solución sería encerrarlo entre comillas simples o utilizar código o secuencias de escape en la cadena delimitada por comillas dobles. Como se muestra a continuación:

```
<?
   $a=22;
   $mensaje = "Tengo \$a años";
   echo $mensaje //El resultado es: "Tengo 22 años"
?>
```

Si se usan **comillas simples** debe realizar una instrucción como la siguiente:

```
$cadena = 'Coloque acá su cadena';
```

Para poder mostrar una comilla simple dentro de una cadena delimitada por comillas simples debe utilizarse una secuencia de escape colocando el símbolo de barra invertida antes de ella. Así:

```
$cadena='Todos lo llamaban \'el mesías\'';
```

Los únicos caracteres que deben escaparse cuando se encierran entre comillas simples son la comilla simple y la barra invertida.

Otra forma de delimitar cadenas es mediante el uso de la **sintaxis heredoc ("<<<")**. Debe indicarse un identificador después de la secuencia <<<, luego la cadena, y luego el mismo identificador para cerrar la cita.

```
 <?
   $frase = <<<ANILLOS
         "Hay muchos vivos que merecen la muerte y hay
                  muchos muertos que merecen la vida,
                  ¿quién eres tú para impartir ese derecho?" – Gandalf.
ANILLOS;
   echo $frase;
?>
```

El identificador de cierre debe comenzar en la primera columna de la línea. Asimismo, el identificador usado debe seguir las mismas reglas que cualquier otra etiqueta en PHP: debe contener solo caracteres alfanuméricos y de subrayado, y debe iniciar con un caracter no-dígito o de subrayado.
Unos aspectos importantes a considerar acerca de esta sintaxis son:

- La línea con el identificador de cierre no puede contener otros caracteres, excepto quizás por un punto-y-coma (;). Esto quiere decir en especial que el identificador no debe usar sangría, y no debe haber espacios o tabuladores antes o después del punto-y-coma.
- Es importante también notar que el primer caracter antes del identificador de cierre debe ser un salto de línea, tal y como lo defina su sistema operativo. Esto quiere decir \r en Macintosh, por ejemplo.

Códigos o secuencias de escape

En PHP, al igual que en otros lenguajes existen ciertos caracteres que generan problemas cuando se encuentran dentro de cadenas, debido a que dentro del lenguaje se interpretan de alguna forma especial. Para poder visualizar correctamente dichos caracteres en una operación de salida se utilizan secuencias de escape que involucran dichos caracteres especiales. La siguiente tabla muestra varios de estos caracteres especiales con su correspondiente secuencia de escape:

Secuencia de escape	Significado
\b	Espacio hacia atrás (*backspace*)
\f	Cambio de página (*form feed*)
\n	Cambio de línea (*line feed*)
\r	Retorno de carro (*carriage return*)
\t	Tabulación horizontal
\\	Barra inversa (*backslash*)
\'	Comilla simple
\"	Comilla doble
\$	Carácter dólar ($)
\###	Carácter ASCII octal (#: 0-7)
\x##	Carácter ASCII hexadecimal (#: 0-F)

Operadores

Los operadores son símbolos especiales que se utilizan en los lenguajes de programación para poder realizar operaciones con las expresiones. Como lo que se obtiene en dichas operaciones es un valor, el resultado también será una expresión.
Los operadores se pueden clasificar como: unarios (que operan sobre un único valor o expresión), binarios (que operan sobre dos valores o expresiones) y ternarios (que consta de tres valores o expresiones)
En PHP existen diversos tipos de operadores y se pueden clasificar de la siguiente manera: aritméticos, lógicos, de cadena, de ejecución, de comparación, de asignación, de incremento/decremento, etc.

Operadores aritméticos

Son los operadores que permiten realizar operaciones numéricas sobre las variables y expresiones. Se muestran en la siguiente tabla:

Símbolo	Nombre	Ejemplo	Resultado
+	Adición	$a + $b	Suma de $a y $b.
-	Substracción	$a - $b	Diferencia entre $a y $b.
*	Multiplicación	$a * $b	Producto de $a y $b.

/	División	$a / $b	Cociente de $a y $b.
%	Módulo	$a % $b	Resto de $a dividido por $b.

Debe tener en cuenta que el operador de división "/" devuelve un número con punto flotante en todos los casos, incluso cuando los dos operandos son enteros.

Operadores lógicos

Los operadores lógicos se utilizan para realizar comparaciones entre expresiones. Pueden combinarse para formar expresiones de comparación más complejas. Los operadores lógicos de PHP se muestran en la siguiente tabla:

Símbolo	Nombre	Ejemplo	Resultado
and	Y	$a and $b	**TRUE** si tanto $a como $b son **TRUE**.
or	O	$a or $b	**TRUE** si cualquiera de $a o $b es **TRUE**.
xor	O exclusivo (Xor)	$a xor $b	**TRUE** si $a o $b es **TRUE**, pero no ambos.
!	No	! $a	**TRUE** si $a no es **TRUE**.
&&	Y	$a && $b	**TRUE** si tanto $a como $b son **TRUE**.
||	O	$a || $b	**TRUE** si cualquiera de $a o $b es **TRUE**.

Operadores de cadena

Estos operadores se utilizan en combinación con variables o expresiones de cadena. Los dos operadores válidos en PHP para operar con cadenas son el operador de concatenación que se representa con un símbolo de punto (.) y el operador de asignación sobre concatenación, representado por un punto seguido de un símbolo de igual que (.=). Vea la siguiente tabla:

Símbolo	Nombre	Ejemplo	Resultado
.	Concatenación	$var1 = "Hola "; $var2 = "mundo"; $saludo = $var1 . $var2;	Se imprimirá en pantalla: Hola mundo
.=	Concatenación y asignación	$var1 = "Hola mundo "; $var1.="cruel y perverso";	Se imprimirá en pantalla: Hola mundo cruel y perverso.

NOTA: Observe que no debe olvidar colocar espacios en blanco al terminar o al comenzar la subcadena para que no queden palabras unidas.

Operadores de comparación

Se utilizan para verificación de condiciones en ciertas expresiones, sobre todo en expresiones condicionales. En la siguiente tabla se muestra la lista completa de ellos:

Símbolo	Nombre	Ejemplo	Resultado
==	Igual	$a == $b	**TRUE** si $a es igual a $b.
===	Idéntico	$a === $b	**TRUE** si $a es igual a $b, y son del mismo tipo. (PHP 4 o superior)
!=	Diferente	$a != $b	**TRUE** si $a no es igual a $b.
<>	Diferente	$a <> $b	**TRUE** si $a no es igual a $b.
!==	No idénticos	$a !== $b	**TRUE** si $a no es igual a $b, o si no son del mismo tipo. (PHP 4 o superior)

<	Menor que	$a < $b	**TRUE** si $a es estrictamente menor que $b.
>	Mayor que	$a > $b	**TRUE** si $a es estrictamente mayor que $b.
<=	Menor o igual que	$a <= $b	**TRUE** si $a es menor o igual que $b.
>=	Mayor o igual que	$a >= $b	**TRUE** si $a es mayor o igual que $b.

Operadores de asignación

Existe un solo operador básico de asignación, que se lee "se asigna a" y no "es igual a", como podría parecer. Además de este operador de asignación existen operadores combinados con operador de asignación que también se mostrarán en esta parte. Vea la siguiente tabla:

Símbolo	Nombre	Ejemplo
=	Se asigna a	$a = "Hola"
.=	Concatenación y asignación	$a .= " mundo"
+=	Adición y asignación	$a += $b
-=	Sustracción y asignación	$a -= $b
*=	Multiplicación y asignación	$a *= $b
/=	División y asignación	$a /= $b
%=	Módulo y asignación	$a %= $b

Operadores de ejecución

PHP soporta un operador de ejecución: las comillas invertidas (``). ¡Note que no se trata de comillas sencillas! PHP intentará ejecutar el contenido entre las comillas como si se tratara de un comando del intérprete de comandos; su salida será devuelta (es decir, no será simplemente volcada como salida; puede ser asignada a una variable).

Ejemplo:

```
<?php
  $salida = `ls - al`;
  echo "<pre>$salida</pre>";
?>
```

Operadores de incremento/decremento

Estos operadores son, en realidad, una mejora a los operadores aritméticos de adición y sustracción, para el caso muy particular en que uno de los operandos sea la unidad. Existen variantes para este operador dependiendo si primero se hace la asignación y luego el incremento/decremento o viceversa. Veamos la siguiente tabla:

Símbolo	Nombre	Finalidad
++$var	Pre-incremento	Incrementa el valor de $var en 1 y luego retorna el nuevo valor de $var
$var++	Post-incremento	Retorna primero el valor actual de $var y después incrementa este valor en 1
--$var	Pre-decremento	Decrementa el valor de $var en 1 y luego retorna el nuevo valor de $var
$var--	Post-decremento	Retorna primero el valor actual de $var y después decrementa este valor en 1

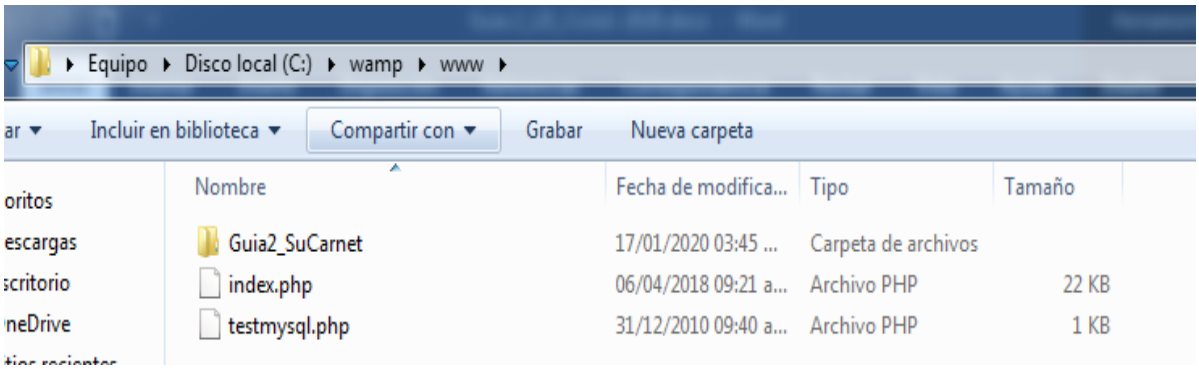
III. MATERIALES Y ESQUIPOS

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

No.	Requerimiento	Cantidad
1	Guía de práctica #1: Introducción a la programación web con PHP	1
2	Computadora con WampServer y Visual Studio Code	1
3	Memoria USB o disco flexible	1

IV. PROCEDIMIENTO

1. Es importante que se asegure de que el servidor web esté en funcionamiento antes de poder visualizar las páginas con código PHP. Para ello, debe iniciar el programa que utilizará como servidor web, WampServer, que incluye el servicio necesario para activar el servidor Apache.
2. Los archivos creados con el editor visual studio code deben guardarse en una ubicación específica, denominada carpeta web, donde se almacenan todas las páginas gestionadas por el servidor web. En la instalación predeterminada de WampServer, esta carpeta se llama "www". Puede encontrarla dentro de la carpeta "wamp", donde se encuentra la carpeta "www".



1. Por cuestión de orden deberá crear una carpeta con el nombre Guia1_SuCarnet dentro de esta ira creando una carpeta para cada ejercicio y almacenar dentro de ella **cada recurso, el cual debe descargar de la página de la Universidad**.

Ejercicio #1:

1. Utilice Visual Studio Code para digitar el siguiente código que mostrará cómo se inserta código PHP dentro de un documento web realizado con lenguaje HTML:

Primer script: partehtmlphp.php

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Demostración de HTML y PHP</title>
  <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0" />
  <!-- Incluir Bootstrap CSS -->
          <link      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" />
```

```
</head>
<body>

<div class="container my-5">
  <header class="text-center mb-4">
    <h1 class="display-4">Demostración de HTML y PHP</h1>
    <p class="lead">Compara las diferencias en el uso de HTML y PHP con ejemplos visuales</p>
  </header>

  <!-- Sección HTML y PHP lado a lado -->
  <section class="row d-flex justify-content-between">

    <!-- Sección HTML alineada a la izquierda -->

    <!-- Sección PHP alineada a la derecha -->

  </section>
</div>

<!-- Incluir Bootstrap JS y dependencias -->
<script                                                    src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

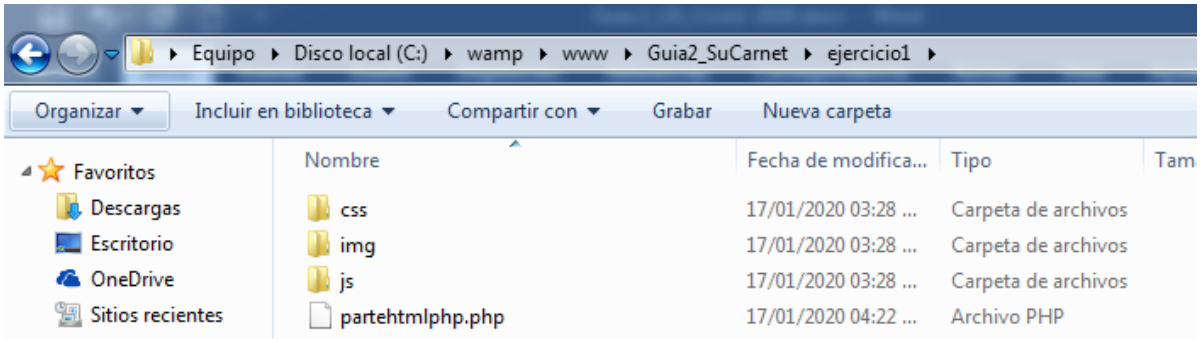
2. Agregar el código html

```
<!-- Sección HTML y PHP lado a lado -->
<section class="row d-flex justify-content-between">
    <!-- Sección HTML alineada a la izquierda -->
    <div class="col-md-5">
        <div class="section-title">HTML</div>
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Código HTML</h5>
                <div class="code-block">
                    <p>&lt;!DOCTYPE html&gt;</p>
                    <p>&lt;html&gt;</p>
                    <p class="highlight">&lt;!-- Parte de HTML normal --&gt;</p>
                    <p>&lt;/html&gt;</p>
                </div>
                <p>Este es un fragmento de código HTML básico que estructura la página.</p>
            </div>
            <div class="card-footer text-center">
                <img src="img/html5.png" alt="Logo de HTML5" class="img-fluid" />
            </div>
        </div>
    </div>
```

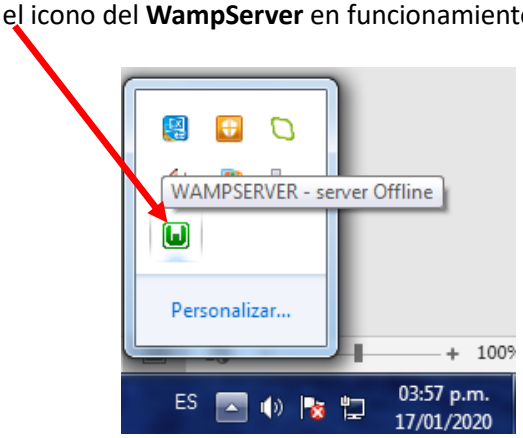
3. Agregar el código php

```
    <!-- Sección PHP alineada a la derecha -->
    <?php
    echo '
    <div class="col-md-5">
        <div class="section-title">PHP</div>
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Código PHP</h5>
                <div class="code-block">
                    <p>&lt;?php</p>
                    <p class="highlight">echo "&lt;h1&gt;Hola desde PHP&lt;/h1&gt;";</p>
                    <p>&lt;?&gt;</p>
                </div>
                <p>Este es un ejemplo de cómo PHP se integra en una página web para generar contenido dinámico.</p>
            </div>
            <div class="card-footer text-center">
                <img src="img/php.png" alt="Logo de PHP" class="img-fluid" />
            </div>
        </div>
    </div>
    ';
    ?>
</section>
</div>
```

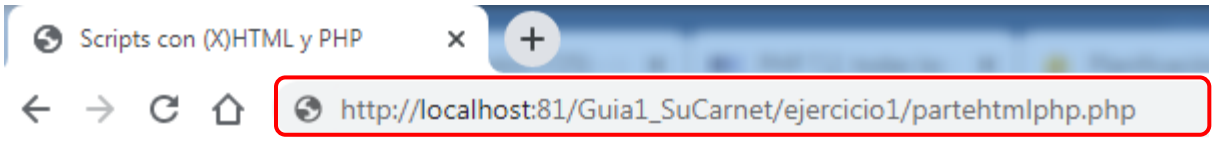
4. Guarde el primer archivo con el nombre **partehtmlphp.php** en la carpeta predeterminada para los documentos web de su servidor web Apache. En una instalación predeterminada del **WampServer** debería ser: C:\wamp\www. Si usted modificó esta carpeta debe asegurarse de que sea en la carpeta correcta de los documentos web.



5. Para visualizar el resultado en el navegador verifique que esté iniciado el servidor web Apache. Si lo está deberá observar en el área de estado de la barra de tareas de Windows (donde aparece la hora) el icono del **WampServer** en funcionamiento:

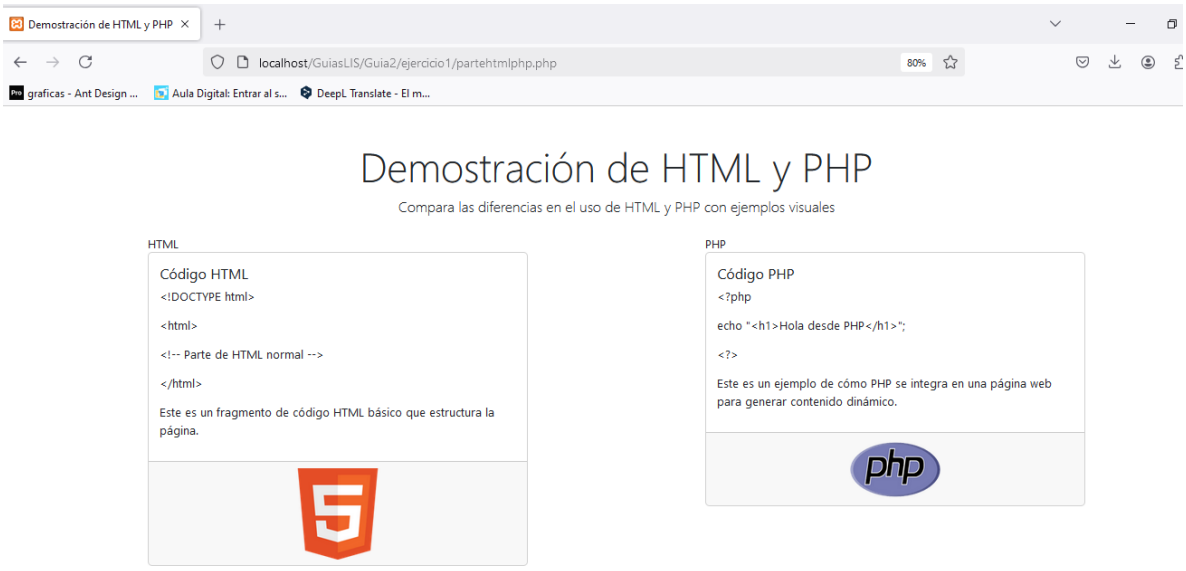


6. Por último, para ver el resultado de los scripts php, diríjanse con el navegador a http://localhost/tuarchivo.php (donde «tuarchivo.php» es el archivo php que queremos correr.), en el caso de ejemplo de la guía seria de la siguiente forma:



Nota: De forma predeterminada, al ejecutar la aplicación WAMP en un servidor Apache, el puerto host local se establece en el **puerto 80**. Sin embargo, puede ajustar esta configuración en su servidor mediante la edición del archivo **"http.conf"** y colocar un nuevo número de puerto. Al cambiar el puerto local de **WAMP**, estará instruyéndole a su servidor Apache para que envíe comunicaciones desde una ubicación diferente, así como se muestra en la imagen del ejemplo.

7. Observe la página PHP que acaba de cargarse en el navegador y verifique que el resultado es el esperado. Es recomendable que verifique en todos los navegadores disponibles: Chrome, Firefox, Internet Explorer, Safari y Opera



NOTA: De ahora en adelante solamente se le proporcionará el código HTML combinado con PHP que debe digitar, para volver a cargar una página en el navegador ya sabe cuáles deben ser los pasos a seguir. Lo mismo será con cualquier otro archivo adicional como hoja de estilo o secuencia de comando de JavaScript, etc.

Ejercicio #2: El siguiente ejemplo muestra cómo opera PHP la declaración de las variables, En este ejemplo hemos definido tres variables, $a, $b y $c y con la instrucción echo hemos impreso el valor que contenían, insertando un salto de línea entre ellas.
Primer script: uso_variable.php

```
<!DOCTYPE html>
<html lang="es">
<head>
```

```
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ejemplo de Variables en PHP</title>
  <!-- Incluir Bootstrap desde su CDN -->
      <link    href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
  <div class="container mt-5">
    <div class="row">
      <div class="col-12">
        <h1 class="text-center">Ejemplo de Variables en PHP</h1>
          <p class="lead text-center">A continuación se muestran los valores de las variables
definidas en PHP:</p>
        <div class="card">
          <div class="card-body">
            <h5 class="card-title">Valor de la variable $a (entero):</h5>
            <p class="card-text">
              <?php
              $a = 10; // Variable con valor entero
              echo $a; // Imprimir el valor de $a
              ?>
            </p>
            <h5 class="card-title">Valor de la variable $b (texto):</h5>
            <p class="card-text">
              <?php
              $b = "Hola Mundo"; // Variable con valor de texto
              echo $b; // Imprimir el valor de $b
              ?>
            </p>
            <h5 class="card-title">Valor de la variable $c (decimal):</h5>
            <p class="card-text">
              <?php
              $c = 3.14; // Variable con valor decimal
              echo $c; // Imprimir el valor de $c
              ?>
            </p>
          </div>
        </div>

      </div>
    </div>
  </div>
  <!-- Incluir jQuery y Bootstrap JS desde sus CDN -->
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
                                                                                <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js"></script>
```

```
                                                                    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

El resultado en el navegador sería el siguiente:

Ejercicio #3: El siguiente ejemplo muestra cómo opera PHP los distintos tipos de operadores aritméticos, Los operadores de PHP son muy parecidos a los de C y JavaScript, si usted conoce estos lenguajes le resultaran familiares y fáciles de reconocer.

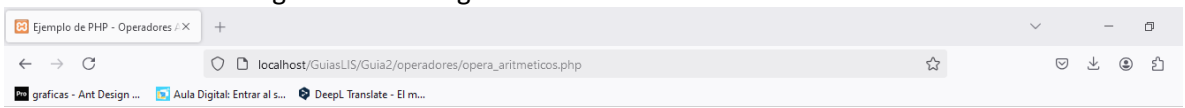
Primer script: opera_aritmeticos.php

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ejemplo de PHP - Operadores Aritméticos</title>
  <!-- Incluir Bootstrap desde su CDN -->
       <link    href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
  <style>
    body {
      font-family: "Arial Black";
    }
    .container {
      margin-top: 50px;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1 class="text-center mb-4">Operadores Aritméticos en PHP</h1>
    <p class="lead text-center">Ejemplo de cómo realizar operaciones aritméticas con PHP:</p>
```

```
    <div class="card">
      <div class="card-body">
        <h5 class="card-title text-center">Operaciones Aritméticas</h5>
        <p class="card-text">
          <?php
          $a = 8;
          $b = 3;
          ?>
          <strong>Resultado de la suma:</strong> <?php echo $a + $b; ?> <br>
          <strong>Resultado de la resta:</strong> <?php echo $a - $b; ?> <br>
          <strong>Resultado de la multiplicación:</strong> <?php echo $a * $b; ?> <br>
          <strong>Resultado de la división:</strong> <?php echo $a / $b; ?> <br>
          <strong>Valor de $a después del incremento:</strong> <?php $a++; echo $a; ?> <br>
          <strong>Valor de $b después del decremento:</strong> <?php $b--; echo $b; ?> <br>
        </p>
      </div>
    </div>
  </div>

  <!-- Incluir jQuery y Bootstrap JS desde sus CDN -->
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
                                                                    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js"></script>
                                                                    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

El resultado en el navegador sería el siguiente:



Operadores Aritméticos en PHP

Ejemplo de cómo realizar operaciones aritméticas con PHP:

Operaciones Aritméticas

Resultado de la suma: 11
Resultado de la resta: 5
Resultado de la multiplicación: 24
Resultado de la división: 2.6666666666667
Valor de $a después del incremento: 9
Valor de $b después del decremento: 2

Ejercicio #4: El siguiente ejemplo muestra cómo opera PHP los distintos tipos de datos que maneja (enteros, flotantes, cadenas y booleanos) y el uso de constantes ocupando variables locales definidas directamente en el código.

Primer archivo: tiposdatos.php

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Conversión de Cadenas</title>
                                                    <link                              rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body>
<div class="container mt-5">
  <header class="text-center mb-4">
    <h1 class="display-5 fw-bold">Trabajando con los Tipos de Datos en PHP</h1>

  </header>

  <section>

  <?php
  error_reporting(E_ALL & ~E_WARNING);
  ini_set('display_errors', '1'); // Habilita la visualización de errores configurados

    echo "<div class='container-fluid py-5 text-bg-dark'>";
    echo "<p class='col-md-12 fs-4' id='datos'>" ;
    echo "<strong>Variables:</strong><br>" ;
        define("SALTO", "\n");
        $cad = "10 metros";
        $str = "3D";
        $val = 20.5;
        $num = 7;

        echo "\$cad = \"$cad\"<br>\n";
        echo "\$str = \"$str\"<br>\n";
        echo "\$val = $val<br>\n";
        echo "\$num = $num<br>\n";
    echo "</p>";
    echo "</div>";
    echo "<div class='row'>";
      // Primera operación
      echo '<div class="col-md-6 mb-4">';
      echo '<div class="card">';
      echo '<div class="card-body">';
      $concat = $cad + $val;
```

```
echo "<p>El valor de <strong>\$concat</strong> = \$cad + \$val es:</p>";
echo "<p><strong>\$cad= \"$cad\" + $val</strong></p>";
echo "<p><strong>Resultado:</strong> $concat</p>";
echo "<p><strong>Tipo de dato:</strong> " . gettype($concat) . "</p>";
echo '</div></div></div>';

// Segunda operación
echo '<div class="col-md-6 mb-4">';
echo '<div class="card">';
echo '<div class="card-body">';
$concat = $str + $num;
echo "<p>El valor de <strong>\$concat</strong> = \$str + \$num es:</p>";
echo "<p><strong>\$cad= \"$str\" + $num</strong></p>";
echo "<p><strong>Resultado:</strong> $concat</p>";
echo "<p><strong>Tipo de dato:</strong> " . gettype($concat) . "</p>";
echo '</div></div></div>';

// Tercera operación
echo '<div class="col-md-6 mb-4">';
echo '<div class="card">';
echo '<div class="card-body">';
$concat = $cad + $num;
echo "<p>El valor de <strong>\$concat</strong> = \$cad + \$num es:</p>";
echo "<p><strong>\$cad= \"$cad\" + $num</strong></p>";
echo "<p><strong>Resultado:</strong> $concat</p>";
echo "<p><strong>Tipo de dato:</strong> " . gettype($concat) . "</p>";
echo '</div></div></div>';

// Cuarta operación
echo '<div class="col-md-6 mb-4">';
echo '<div class="card">';
echo '<div class="card-body">';
$concat = $str + $val;
echo "<p>El valor de <strong>\$concat</strong> = \$str + \$val es:</p>";
echo "<p><strong>\$cad= \"$str\" + $val</strong></p>";
echo "<p><strong>Resultado:</strong> $concat</p>";
echo "<p><strong>Tipo de dato:</strong> " . gettype($concat) . "</p>";
echo '</div></div></div>';

// Quinta operación
echo '<div class="col-md-6 mb-4">';
echo '<div class="card">';
echo '<div class="card-body">';
$concat = $cad + $str;
echo "<p>El valor de <strong>\$concat</strong> = \$cad + \$str es:</p>";
echo "<p><strong>\$cad= \"$cad\" + \"$str\"</strong></p>";
echo "<p><strong>Resultado:</strong> $concat</p>";
echo "<p><strong>Tipo de dato:</strong> " . gettype($concat) . "</p>";
```

```
echo '</div></div></div>';

// Sexta operación
echo '<div class="col-md-6 mb-4">';
echo '<div class="card">';
echo '<div class="card-body">';
$concat = $val + $cad;
echo "<p>El valor de <strong>\$concat</strong> = \$val + \$cad es:</p>";
echo "<p><strong>\$cad= $val + \"$cad\"</strong></p>";
echo "<p><strong>Resultado:</strong> $concat</p>";
echo "<p><strong>Tipo de dato:</strong> " . gettype($concat) . "</p>";
echo '</div></div></div>';

// Séptima operación
echo '<div class="col-md-6 mb-4">';
echo '<div class="card">';
echo '<div class="card-body">';
$concat = $num + $str;
echo "<p>El valor de <strong>\$concat</strong> = \$num + \$str es:</p>";
echo "<p><strong>\$cad= $num + \"$str\"</strong></p>";
echo "<p><strong>Resultado:</strong> $concat</p>";
echo "<p><strong>Tipo de dato:</strong> " . gettype($concat) . "</p>";
echo '</div></div></div>';

// Octava operación
echo '<div class="col-md-6 mb-4">';
echo '<div class="card">';
echo '<div class="card-body">';
$concat = $num + $cad;
echo "<p>El valor de <strong>\$concat</strong> = \$num + \$cad es:</p>";
echo "<p><strong>\$cad= $num + \"$cad\"</strong></p>";
echo "<p><strong>Resultado:</strong> $concat</p>";
echo "<p><strong>Tipo de dato:</strong> " . gettype($concat) . "</p>";
echo '</div></div></div>';

// Novena operación
echo '<div class="col-md-6 mb-4">';
echo '<div class="card">';
echo '<div class="card-body">';
$concat = $val + $str;
echo "<p>El valor de <strong>\$concat</strong> = \$val + \$str es:</p>";
echo "<p><strong>\$cad= $val + \"$str\"</strong></p>";
echo "<p><strong>Resultado:</strong> $concat</p>";
echo "<p><strong>Tipo de dato:</strong> " . gettype($concat) . "</p>";
echo '</div></div></div>';

// Décima operación
echo '<div class="col-md-6 mb-4">';
```

```
        echo '<div class="card">';
        echo '<div class="card-body">';
        $concat = $str + $cad;
        echo "<p>El valor de <strong>\$concat</strong> = \$str + \$cad es:</p>";
        echo "<p><strong>\$cad= \"$str\" + \"$cad\"</strong></p>";
        echo "<p><strong>Resultado:</strong> $concat</p>";
        echo "<p><strong>Tipo de dato:</strong> " . gettype($concat) . "</p>";
        echo "</div>";
        ?>
    </section>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

El resultado en el navegador sería el siguiente:

Ejercicio #5: Este último ejemplo ilustrará cómo obtener los datos ingresados en un formulario y procesarlos con un script PHP sencillo que, de momento, no utilizará ninguna validación del lado del servidor, solamente validación del lado del cliente con JavaScript que ya conoce. En posteriores prácticas abordaremos el tema de la validación en el lado del servidor.

Primer script: ingresodatos.html

```
<!DOCTYPE html>
<html lang="es">
<head>
```

```
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Formulario de ingreso de datos</title>
        <link     href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
                                                                                <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
  <script src="js/validar.js"></script>
</head>
<body>
<header class="bg-primary text-white text-center py-3">
  <h1>Ingreso de datos</h1>
</header>
<section class="container my-5">
  <article class="card shadow">
    <div class="card-body">
      <h2 class="card-title text-center mb-4">Formulario</h2>
      <form action="procesar.php" method="POST" >
        <div class="mb-3">
          <label for="client" class="form-label">Cliente:</label>
         <input type="text" id="client" name="client" class="form-control" placeholder="Ingrese
el nombre del cliente" required>
        </div>
        <div class="mb-3">
          <label for="product" class="form-label">Producto:</label>
                    <input   type="text"   id="product"   name="product"   class="form-control"
placeholder="Ingrese el nombre del producto" required>
        </div>
        <div class="mb-3">
          <label for="price" class="form-label">Precio:</label>
                     <input   type="number"   id="price"   name="price"   class="form-control"
placeholder="Ingrese el precio" required>
        </div>
        <div class="mb-3">
          <label for="quantity" class="form-label">Cantidad:</label>
                <input  type="number"  id="quantity"  name="quantity"  class="form-control"
placeholder="Ingrese la cantidad" required>
        </div>
        <div class="text-center">
                     <button   type="submit"   id="enviar"   name="submit"   class="btn   btn-
primary">Enviar</button>
        </div>
      </form>
    </div>
  </article>
</section>
</body>
</html>
```

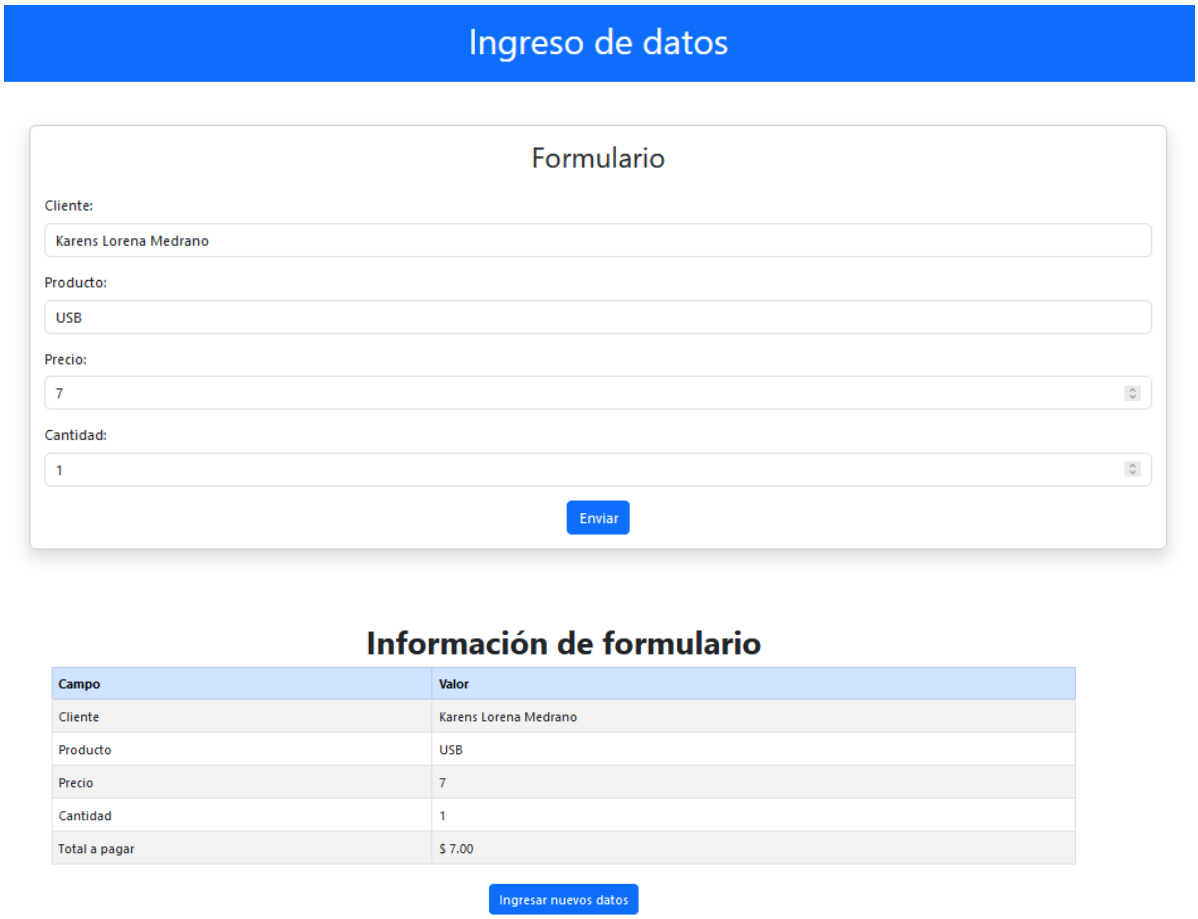
Segundo script: procesar.php

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
   <title>Información recibida</title>
          <link     href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
                                                                                    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
<section class="container my-5">
<article>
<h1 class="text-center fw-bold">Información de formulario</h1>
<div id="info" class="table-responsive">
<table class="table table-striped table-bordered">

<thead class="table-primary">
   <tr>
      <th scope="col">Campo</th>
      <th scope="col">Valor</th>
   </tr>
</thead>
<tbody>
<?php
if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['submit'])):
   echo "\t<tr>\n";
   echo "\t\t<td>Cliente</td>\n";
   // Accediendo a los datos del formulario usando la función extract()
   extract($_POST);
   $cliente = !empty($client) ? $client : "<a href='ingresodatos.html' class='text-danger'>No ingresó
el cliente.</a>";
   echo "\t\t<td>" . $cliente . "</td>\n";
   echo "\t</tr>\n";
   echo "\t<tr>\n";
   echo "\t\t<td>Producto</td>\n";
   $producto = !empty($product) ? $product : "<a href='ingresodatos.html' class='text-danger'>No
ingresó el producto.</a>";
   echo "\t\t<td>" . $producto . "</td>\n";
   echo "\t</tr>\n";
   echo "\t<tr>\n";
   echo "\t\t<td>Precio</td>\n";
   $precio = !empty($price) ? $price : "<a href='ingresodatos.html' class='text-danger'>No ingresó
el precio</a>";
   echo "\t\t<td>" . $precio . "</td>\n";
```

```
    echo "\t</tr>\n";
    echo "\t<tr>\n";
    echo "\t\t<td>Cantidad</td>\n";
    $cantidad = !empty($quantity) ? $quantity : "<a href='ingresodatos.html' class='text-danger'>No
ingresó la cantidad</a>";
    echo "\t\t<td>" . $cantidad . "</td>\n";
    echo "\t</tr>\n";
    echo "\t<tr>\n";
    echo "\t\t<td>Total a pagar</td>\n";
    if(isset($cliente) && isset($producto) && floatval($precio)>0 && floatval($cantidad)>0):
        echo "\t<td>$ " . number_format($precio * $cantidad, 2, '.', ',') . "</td>\n";
    else:
        echo "\t<td>Nada que cobrar</td>\n";
    endif;
    echo "\t</tr>\n";
else:
    echo "\t<tr>\n";
        echo "\t\t<td colspan='2' class='text-danger'>No se han ingresado datos desde el
formulario.</td>\n";
    echo "\t</tr>\n";
endif;
?>
</tbody>
</table>
<div id="link" class="text-center mt-4">
    <a href="ingresodatos.html" class="btn btn-primary">Ingresar nuevos datos</a>
</div>
</div>
</article>
</section>
</body>
</html>
```

Resultado en el navegador:



V. EJERCICIOS COMPLEMENTARIOS

El instructor evaluará el desempeño de su trabajo dentro la clase. Estos ejercicios se evaluarán en la misma hora de clase.

1) Basándote en el ejercicio 3, este ejercicio tiene como objetivo mostrar cómo operan los **operadores de comparación** y **operadores lógicos** en PHP.
Los **operadores de comparación** se utilizan para comparar valores y determinar relaciones entre ellos. Algunos de los operadores de comparación más comunes en PHP son:

- ==: Igual a
- !=: Diferente de
- >: Mayor que
- <: Menor que
- >=: Mayor o igual que
- <=: Menor o igual que
- ===: Identico (compara valor y tipo)

- !==: No idéntico (compara valor y tipo)

Los **operadores lógicos** se emplean para combinar múltiples condiciones o comparaciones y devolver un resultado basado en esas evaluaciones. Los operadores lógicos más utilizados son:

- &&: AND (devuelve verdadero si ambas condiciones son verdaderas)
- ||: OR (devuelve verdadero si al menos una condición es verdadera)
- !: NOT (invierte el valor de una condición)

V. DISCUSIÓN DE RESULTADOS

2) Realice un script PHP que muestre mediante la utilización de variables sus datos personales: nombre completo, lugar de nacimiento (departamento y país, si es extranjero), edad y carnet de la universidad. Muestre estos datos en una tabla.

3) Cree un script PHP que utilice un formulario para solicitar una cantidad que debe ser ingresada en dólares y muestre como respuesta a cuánto equivale esa cantidad en euros. Su respuesta debe indicar la cantidad en dólares que se ingresó en una columna de una tabla de resultados y en la columna siguiente su equivalente en euros. Utilice buena presentación de la tabla y validación de la entrada del usuario.

VI. INVESTIGACION COMPLEMENTARIA

1) Investigue las constantes de PHP conocidas como constantes predefinidas o "mágicas" y las constantes para el tratamiento de errores que brinda el lenguaje. Muestre en una tabla de dos columnas la constante y su utilidad. Por ejemplo:

E_ALL	**Todos los errores y advertencias soportados.**

2) Investigue todas las funciones que dispone PHP para determinar el tipo de una variable. Haga una tabla con dos columnas, una con la sintaxis de la función y otra con la descripción. Por ejemplo:

boolean is_numeric($arg)	**Devuelve true si la variable es un número o una cadena numérica o falso en otro caso.**

VII. BIBLIOGRAFIA

1. Cabezas Granado, Luis Miguel. PHP 6 Manual Imprescindible. 1ra. Edición. Editorial Anaya Multimedia. Madrid, España. 2010.
2. Doyle, Matt. FUNDAMENTOS PHP PRÁCTICO. 1ra Edición. Editorial Anaya Multimedia. Madrid, España. 2010.
3. Tim Converse / Steve Suehring. LA BIBLIA DE PHP 6 y MySQL. Editorial Anaya Multimedia. 1a. Edición. Madrid, España. 2009.
4. Welling, Luke / Thomson, Laura. DESARROLLO WEB CON PHP Y MySQL. Traducción de la 3ra Edición en inglés. Editorial Anaya Multimedia. Madrid, España 2005.
5. Gutiérrez, Abraham / Bravo, Ginés. PHP 5 A TRAVÉS DE EJEMPLOS. 1ra. Edición. Editorial Alfaomega. México, junio 2005.
6. Ellie Quigley / Marko Gargenta. PHP y MySQL Práctico para Diseñadores y Programadores Web. Anaya Multimedia / Prentice Hall. 1a. edición. 2007. Madrid, España.

7. Gil Rubio, Francisco Javier / Villaverde, Santiago Alonso. CREACIÓN DE SITIOS WEB CON PHP 5. 1ra. Edición. Editorial McGraw-Hill/Interamericana. Madrid, España 2006.
8. John Coggeshall. LA BIBLIA DE PHP 5. 1ra. Edición. Editorial Anaya Multimedia. Madrid, España 2005.