RESUMENPHP

Contenido

TEMA 1. Variables y operadores.

- Variables: Los nombres de las variables comienzan con el símbolo dólar ($) y no es necesario definirlas. La misma variable puede contener distintos tipos de dato.
  La función var_dump(vble1, vble2, ...) muestra tipo y valor de cada variable.
  La función isset(vble) devuelve verdadero si la vble tiene definido algún valor.
- Operadores para String
  . concatena dos Strings
  .= añade al final del String de la izquierda, el String de la derecha
- Operadores aritméticos

Operador	Nombre	Ejemplo	Descripción
+	suma	20 + $x	suma dos números
-	resta	$a - $b	resta dos números
*	multiplicación	10 * 7	multiplica dos números
/	división	$altura / 2	divide dos números
%	módulo	5 % 2	devuelve el resto de la división entera
++	incremento	$a++	incrementa en 1 el valor de $a
--	decremento	$a--	decrementa en 1 el valor de $a

Si ++ o -- se pone delante de la variable en una expresión se ejecuta antes de resolverla.
rand(mínimo, máximo) genera un aleatorio entero entre dos valores.
floor(numero con decimales) devuelve la parte entera sin decimales redondeando
ceil(numero con decimales) devuelve la parte entera redondeando hacia arriba siempre
intval(numero con decimales) devuelve la parte entera ignorando los decimales (truncar)

- Operadores de comparación

Operador	Nombre	Ejemplo	Devuelve verdadero cuando…
==	Igual	$a == $b	$a es igual $b (aunque sean de diferente tipo)
===	Igual	$a === $b	$a es igual $b (y además son del mismo tipo)
!=	Distinto	$a != $b	$a es distinto $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
<=>	Nave espacial	$a <=> $b	-1 si $a es menor 0 si son iguales 1 si $b es menor

- Operadores lógicos
  Y ->&&, and Ejemplos: (7>2) && (2<4), (7>2) and (2<4)
  O -> ||, or Ejemplos: (7>2) || (2<4), (7>2) or (2<4)
  Negación -> ! Ejemplos: ! (7>2)
- Recogida de datos de formularios (si no se define method, por defecto se usa GET):
  $_GET[‘nombre’] : (array asociativo para método get y parámetros por url ->Mi enlace)
  $_POST[‘nombre’] : (array asociativo para método post.OBLIGATORIO con sesiones)
  $_REQUEST[‘nombre’] : (array asociativo para métodos get, post y cookies)
  enctype = "multipart/form-data" : en un formulario indica que se envia texto y ficheros (es obligatorio usar metodo post)
  <input type=””>: usar los tipos adecuados (number, color, date, email, url …), para restringir los tipos de datos enviados a la pagina destino.

- Envio de datos por URL, en vez de usar formularios
  Mi enlace

TEMA 2. Estructuras de control.

IF Opción 1

```
if (condición) {
        sentencias a ejecutar cuando la condición es cierta }
else {
        Sentecias a ejecutar cuando la condición es falsa }
```

IF Opción 2

```
(condición) ? expresión1 : expresión2
```

SWITCH

```
switch(variable) {
case valor1:
sentencias
break;
case valor2:
sentencias
break;
. . .
default:
sentencias
}
```

```
switch(true) {
case ExpresionBooleana1:
sentencias
break;
case ExpresionBooleana2:
break;
. . .
default:
sentencias
}
```

BUCLES

```
for (expresion1 ; expresion2 ; expresion3) {
sentencias
}
```

```
foreach ($array as $elemento) {
echo $elemento;
} //para arrays normales
```

```
foreach ($array as $indice=>$valor) {
echo $elemento;
} //para arrays asociativos
```

```
while (expresion) {
sentencias
}
```

```
do {
sentencias
} while (expresion)
```

TEMA 3. Arrays

Arrays clásicos de tamaño variable

```
$v[0] = 16; $v[1] = 15; $v[2] = 17; $v[3] = 15; $v[4] = 16;
$v = array(16, 15, 17, 15, 16);
$color = ["verde", "amarillo", "rojo", "azul", "blanco", "gris"];
$v = explode("caracter", $texto); separa el texto en cada carácter y lo almacena en el array
$texto = implode ("caracter", $v); une los elementos del array en una cadena, con el
    carácter entre los elementos del array
$v[]=14; Añade el valor en la última posición
echo $v[4]; acceso al valor del índice 4 del array
```

Arrays clásicos de tamaño fijo:

```
$v = new SplFixedArray(10); (los valores no inicializados son null)
```

Arrays asociativos:

```
$edades['Rosa']=16; $edades['Ignacio']=25; $edades['Daniel']=17; $edades['Rubén']=18;
$edades = array("Rosa" => 16, "Ignacio" => 25, "Daniel" => 17, "Rubén" => 18);
$edades = ["Rosa" => 16, "Ignacio" => 25, "Daniel" => 17, "Rubén" => 18];
echo "Daniel:  ", $edades['Daniel'];
```

Paso de un array en un formulario

```
<input type=”text” name=”nombreArray[índice1]”>
<input type=”text” name=”nombreArray[índice2]”>
…
```

Recogida de un array en página de destino.

```
$nombreArray=$_GET[‘nombreArray’];
```

Arrays bidimensionales:

```
$v = array(array(5, 6, 2), array(4, 7, 1, 6, 3), array(5, 9));
$v = [[5, 6, 2], [4, 7, 1, 6, 3], [5, 9]];
$persona = array (
array( "nombre" =>"Rosa", "estatura" => 168, "sexo" =>"F"),
array( "nombre" =>"Ignacio", "estatura" => 175, "sexo" =>"M"),
array( "nombre" =>"Daniel", "estatura" => 172, "sexo" =>"M"),
array( "nombre" =>"Rubén", "estatura" => 182, "sexo" =>"M")
);
$persona = [['ana'=>'pepe', 'julia'=>'juan'],
['luisa'=>'adrian'],
['eva'=>'wally', 'sandra'=>'antonio','maria'=>'jose']];
```

Funciones Ordenación de Arrays:

count($array): devuelve la dimensión de un array

in_array($buscado, $array): devuelve booleano si el elemento buscado está en array

array_key_exists($buscado,$array): devuelve booleano si elemento buscado en índices del array asociativo

Recorrido array bidimensional clásico

```
for($i=0; $i<count($m); $i++){
for ($j=0; $j<count($m[$i]) ; $j++){
echo $m[$i][$j],' - ';
}
echo '<br>';
}
```

Recorrido array bidimensional asociativo

```
for($i=0; $i<count($persona); $i++){
foreach ($persona[$i] as $m=>$h){
echo $m, '  casada con  ',$h,' ; ';
}
echo '<br>';
}
```

TEMA 4. Funciones

function nombreFuncion ($par1, $par2, …) {
instrucciones
return valor;
}

Librerias: fichero php que solo contiene funciones entre <?php y ?>, para usar las funciones es necesaria la instrucción include fichero.php; o include_one fichero.php;

Parametros: los **datos primitivos** se pasan por valor, si anteponemos & delante del parametro en la definición de los parámetros de la función, se pasa por referencia, **los objetos siempre se pasan por referencia**.

Parametros opcionales: para establecer un parámetro como opcional, es necesario darle un valor por defecto. Si se quiere omitir algún valor en la llamada se pone null en lugar del valor, y por tanto el parametro en la función tomará el valor por defecto. Ejemplos:

Declaración: function nombreFuncion ($par1=valor1, $par2=valor2, …) {

Llamada: nombreFuncion (valor1, null, valor3, …)

Sobrecarga: En PHP no se pueden sobrecargar funciones, podemos hacer un apaño:

```
function opera($x, $y, $z) {
if (!isset($y)) {
return $x * $x;
} else if (!isset($z)) {
return $x * $y;
} else {
return $x + $y + $z; }
}
```

Diferencia entre include, include_once, require y require_once

– **include()**:

```
//incluye el fichero functions.php
include("functions.php");
```

Esta función nos permite incluir el archivo tantas veces como lo pidamos, sin importar que esté incluido con anterioridad o no, simplemente evalúa el archivo y lo incluye en nuestro documento. En caso de que el archivo no exista, nos devolverá un warning pero el script continuará ejecutándose.

– **include_once()**:

```
//evalúa si functions.php está incluido, si no lo está lo incluye
include_once("functions.php");
```

La diferencia respecto a la función include() y tal y como muestra su nombre, evaluará si el archivo ya ha sido incluido y si es así, no volverá a incluirlo, es decir, se incluye una única vez. En el caso de error por no encontrar el archivo se comporta igual que include().

– **require()**:

```
//incluye y obliga a que functions.php esté incluido
require("functions.php");
```

Posee el mismo comportamiento que include(), la única diferencia reside en el caso de error, pues cuando usamos require(), si el archivo no existe lanza un error fatal que para la ejecucción del script.

– **require_once()**:

```
//evalúa y obliga a estar incluido, si lo está, no hace nada
require_once("functions.php");
```

Al igual que ocurre entre include() y include_once(), el comportamiento de require_once() es el mismo que el de require(), tan sólo que el primero evalúa si el archivo ya ha sido incluido y si es así, no vuelve a incluirlo de nuevo. El tratamiento del error es igual al que realiza require().

TEMA 5. Funciones de texto principales

strcmp ($cadena1 , $cadena2) devuelve 0 si son iguales >0 si cadena1 mayor y <0 si es menor en orden alfabético según código ASCII(las mayúsculas están después de todas las minúsculas)

$saludo = "Hola, estamos trabajando con cadenas"; //la posición del primer carácter es 0
echo "
Todo en minúsculas: " . **strtolower($saludo)**;
echo "
Todo en mayúsculas: " . **strtoupper($saludo)**;
echo "
Primera letra mayuscula: " . **ucfirst($saludo)**;
echo "
Primeras palabras mayúsculas" . **ucwords($saludo)**;
echo "
Eliminamos espacios: " . **trim($saludo)**;
echo "
Repetimos la cadena: " . **str_repeat($saludo, 5)**;
echo "
Contamos los caracteres: " . **strlen($saludo)**;
echo "
Busqueda de cadenas: " . **strstr($saludo, 'la')**; //devuelve la cadena donde la encuentre hasta el final
echo "
Remplazando cadenas: " . **str_replace(["Hola","Buenas","Hello"], "Adios", $saludo)**; //sustituye una o varias cadenas de un array por otra cadena
echo "
Extraer cadenas: " . **substr($saludo, 2, 8)**; //extrae desde la posición 2 hasta la 10 (2+8) desde la posición 2 mas 8 caracteres.

echo “
Encontrar posición primera ocurrencia: ”.**mb_stripos($saludo, “estamos”);** //si no encuentra devuelve false
echo "
Encontrar posición primera ocurrencia: " . **strpos($saludo, "estamos")**; //si no encuentra devuelve cadena vacia
echo "
Encontrar posición última ocurrencia: " . **strrpos($saludo, "estamos")**; //si no encuentra devuelve cadena vacia
echo "
Devuelve todo el string desde la primera ocurrencia" . **strstr($saludo, "estamos")**;
preg_match("/patron/i", $saludo); //devuelve un entero con las veces que se cumple el patrón en el string en $saludo sin distinguir mayúsculas, si se quita la i, si distingue mayúsculas y minúsculas. El patrón se define como una expresión regular, caracteres comodín:

- “.” coincide una vez con cualquier carácter excepto “\n”
- “?” 0 o 1 ocurrencias del carácter que le precede
- “*” cero o más ocurrencias del carácter que le precede
- “+” una o más ocurrencias del carácter que le precede
- “^” comienza por esa cadena, con “/^patron/m” da nº de líneas coincidentes
- “$” termina por esa cadena con “/$patron/m” da nº de líneas coincidentes

Explicado en url: https://diego.com.es/expresiones-regulares-en-php
echo "
Voltea un string dado".**strrev($saludo)**; //voltea o invierte una cadena
$array = **str_split ($saludo)**; devuelve un array con cada carácter en una posición
ord(string): devuelve el código ASSCII del primer carácter de la cadena pasada por parámetro
chr(código ascii): devuelve el carácter correspondiente al código ascii pasado por parámetro

FECHAS

checkdate ($mes, $dia, $año); //devuelve true o false según la fecha sea correcta o no (para el mes y el dia adminte 1 o 2 dígitos, y para el año 2 o 4 dígitos, sabiendo que con 2 dígitos los valores entre 00-69 hacen referencia a 2000-2069 y 70-99 a 1970-1999).

Funcion date (formato [, fecha]): Devuelve un String con la fecha/hora formateada actual u opcionalmente, una indicada por parámetro en formato fecha (número entero correspondientes a los segundos transcurridos desde 1900).

$fecha=date("d/m/Y"); //La fecha de hoy es:02/09/2018 (el carácter '/' se puede cambiar)
$fecha=date("j/n/y"); //La fecha de hoy es:2/9/18 (el carácter '/' se puede cambiar)
$hora=date("H:i:s"); //La hora actual es:15:06:31 (el carácter ':' se puede cambiar)
Caracteres dentro de la función date () con su aplicación práctica:
a -> Imprime "am" o "pm"
A -> "AM" o "PM"
h -> La hora en formato (01-12)
H -> Hora en formato 24 (00-23)
g -> Hora de 1 a 12 sin un cero delante
G -> Hora de 1 a 23 sin cero delante
i -> Minutos de 00 a 59
s -> Segundos de 00 a 59
d -> Día del mes (01 a 31)
j -> Día del mes sin cero (1 a 31)
w -> Día de la semana (0 a 6). El 0 es el domingo
m -> Mes actual (01 al 12)
n -> Mes actual sin ceros (1 a 12)
Y -> Año con 4 dígitos (2004)
y -> Año con 2 dígitos (04)
z -> Día del año (0 a 365)
t -> Número de días que tiene el mes actual
L -> 1 or 0, según si el año es bisiesto o no

Funcion strtotime(cadena): convierte un string a fecha/hora (entero que representa un dia/hora concreto)
echo strtotime("now"); //Fecha y hora actual
echo strtotime("10 September 2000"); // 10 de septiembre de 2000
echo strtotime("9/10/00"); // 10 de septiembre de 2000 (Fíjate en el orden "m/d/Y")
echo **strtotime("$año-$mes-$dia")**; fecha creada a partir de 3 variables (Fíjate en el orden)
echo strtotime("+1 day"); // actual + 1 dia
echo strtotime("+1 week"); //actual + 1 semana
echo strtotime("+1 week 2 days 4 hours 2 seconds"); //actual + 1 semana 2 dias 4 hrs y 2 seg
echo strtotime("next Thursday"); //próximo jueves
echo strtotime("last Monday"); //ultimo lunes que hayamos pasado

Sumar días, semanas, meses, años a una fecha
$fecha_actual = date("d-m-Y"); //$fecha_actual se podría suprimir, por defecto es la actual

```
echo date("d-m-Y",strtotime($fecha_actual."+ 1 days")); //sumo 1 día
echo date("d-m-Y",strtotime($fecha_actual."+ 1 week")); //sumo 1 semana
echo date("d-m-Y",strtotime($fecha_actual."+ 1 month")); //sumo 1 mes
echo date("d-m-Y",strtotime($fecha_actual."+ 1 year")); //sumo 1 año
```

Comparar fechas: Las fechas tienen que ser comparadas en formato fecha UNIX(entero)

```
$fecha_actual = strtotime(date("d-m-Y H:i:00",time()));
$fecha_entrada = strtotime("19-11-2008 21:00:00");
if($fecha_actual > $fecha_entrada){
echo "La fecha actual es mayor a la comparada.";
}else{
echo "La fecha comparada es igual o menor";
}
```

TEMA 6. Sesiones y Cookies

Sesiones

Al principio, antes de html incluir inicio de sesión:

```
<?php session_start();?>
//Para evitar warnings podemos poner
<?php if ( session_status() == PHP_SESSION_NONE) {  session_start(); }?>
```

Acceso a variables de sesión a través del array asociativo $_SESSION[‘variable’]

```
if(!isset($_SESSION['visitas'])) { $_SESSION['visitas']=valor;}
```

Destruir sesión (borrar todas las variables de sesión)

```
session_destroy(); //destruye la sesion
header("refresh: 0;"); // refresca la página
```

Importante: Trabajando con sesione se debe utilizar el método POST en formularios.

Cookies

Las cookies se crean con la función setcookie() al principio, antes de html:

```
setcookie(nombre, valor, segundos hasta expiración);
Ejemplo: setcookie("usuario", "Luis", time() + 7*24*60*60);
```

Acceso a valores almacenados en cookie a través del array asociativo $_COOKIE[‘nombre’]

```
if (isset($_COOKIE['usuario'])) {$_SESSION[‘usuario’] = $_COOKIE['usuario'];}
```

Borrar una cookie

```
setcookie("usuario", NULL, -1);
```

TEMA 7. Ficheros

La función **fopen(fichero, modoDeApertura)** sirve para abrir ficheros (archivos)

Modo	Observaciones
r	Abre el archivo sólo para lectura. La lectura comienza al inicio del archivo.
r+	Abre el archivo para lectura y escritura. La lectura o escritura comienza al inicio del archivo, machacando el contenido previo según se va escribiendo en él.
w	Abre el archivo sólo para escritura. La escritura comienza al inicio del archivo, y elimina el contenido previo. Si el archivo no existe, intenta crearlo.
w+	Abre el archivo para escritura y lectura. La lectura o escritura comienza al inicio del archivo, y elimina el contenido previo. Si el archivo no existe, intenta crearlo.
a	Abre el archivo para sólo escritura. La escritura comenzará al final del archivo, sin afectar al contenido previo. Si el fichero no existe se intenta crear.
a+	Abre el archivo para lectura y escritura. La lectura o escritura comenzará al final del fichero, sin afectar al contenido previo. Si el fichero no existe se intenta crear.

La función **fgets(identificadorDelFichero[, bytes])** recupera el contenido de una línea, o los bytes (caracteres) indicados opcionalmente o hasta alcanzar el salto de linea.

La función **fgetc(identificadorDelFichero)** recupera un carácter de un archivo.

La función **fgetcsv($f, 1000, ",")**es similar a fgets en archivosCSV, devolviendo un array con los campos en la línea leída. 1000 es la longitud máxima de la línea o 0 para cualquier longitud, y "," es el delimitador de campos en el CSV.

La función **fputs(identificadorDelFichero)** escribe una línea en un archivo. (usar constante **PHP_EOL** para salto de línea)

La función **fwrite(identificadorDelFichero)** escribe una línea en un archivo. (similar a fputs)

La función **fclose(identificadorDelFichero)** cierra un archivo abierto.

La función **ftell(identificadorDelFichero)** devuelve la posición del cursor del archivo (contando todos los caracteres del mismo, incluido los saltos de línea)

La función **rewind(identificadorDelFichero)** posiciona el cursor al principio

La función **file_exists('ruta de fichero')** devuelve booleano si existe fichero o no

La función **$array = file("fichero.txt")** almacena cada línea del fichero en un array

```php
<?php
  $fp = fopen("fichero.txt", "r");
  while(!feof($fp)) {
        $linea = fgets($fp);
        echo $linea . "<br />";
  }fclose($fp);
?>
```

```php
<?php
$file = fopen("archivo.txt", "a");
foreach ($array as $cadena) {
  fwrite($file, $cadena . PHP_EOL);
}
fclose($file);
?>
```

TEMA 8. Acceso a Bases de Datos PDO (PHP Data Objet)

Crear conexión a la BD

```
try {
  $conexion = new PDO("mysql:host=localhost", "root", "root");
  echo "Se ha establecido una conexión con el servidor de bases de datos.";
} catch (PDOException $e) {
  echo "No se ha podido establecer conexión con el servidor de bases de datos.<br>";
  die ("Error: " . $e->getMessage());
}
```

Recorrido de una tabla

```
$consulta = $conexion->query("SELECT dni FROM cliente");
while ($cliente = $consulta->fetchObject()) {
  echo 'br'.$cliente->dni;
}
```

Consulta del numero de filas de una consulta

```
echo $consulta->rowCount();
```

Cerrar conexión a la BD

```
$conexion->close();
```

Insertar, modificar o borrar un registro

```
$insercion = "INSERT INTO cliente (dni, nombre, direccion, telefono) VALUES
('$_POST[dni]','$_POST[nombre]','$_POST[direccion]','$_POST[telefono]')";
$delete = "DELETE FROM cliente WHERE dni=" . $_POST['dni'];
$update= "UPDATE cliente SET  nombre=\"$_POST[nombre]\",
direccion=\"$_POST[direccion]\", telefono=\"$_POST[telefono]\" WHERE
dni=\"$_POST[dni]\"";
$conexion->exec($insercion);
```

Funciones SQL

LIMIT valor1, valor2 => al final de un SELECT, selecciona a partir de la fila valor1, un numero de filas valor2

TEMA 9. POO Programación Orientada a Objetos en PHP

- **La clase se debe crear en un fichero con extensión php, con nombre igual al de la clase.**

```
class NombreClase {
        private $atributo1;
        private $atributo2;
        public function __construct($atr1, $atr2) { // método mágico Constructor
        $this->atributo1 = $atr1;
        $this->atributo2 = $atr2;
        }
        public function setAtributo1($atr1) {
        $this->atributo1 = $atr1;
        }
        public function getAtributo1() {
        return $this->atributo1;
```

```
        }
        public function nombreFuncion($parametro1, $parametro2, …) {
        …
        return valor;
        }
        public function __toString() { // método mágico toString
        return "texto $this->atributo1 texto $this->atributo2”;
    }
}
```

- **Uso de parámetros opcionales en el constructor.**

```
public function __construct($atr1=valor_por_defecto, $atr2=valor_por_defecto, …)
```

Si se omiten algunos valores al crear el objeto, se escribe null en el parámetro omitido.

```
$objeto = new ClaseObjeto (valor1, null, valor3, ...)
```

- **Uso de una clase almacenada en otro fichero.**

```
include_once 'NombreClase.php';
$objeto = new NombreClase(valor1, valor2);
echo $objeto->getAtributo1(); //imprime el atributo1 con llamada al método get
$objeto->setAtributo1(valor); //establece un nuevo valor para el atributo1
$variable=$objeto->nombreFuncion();
echo $objeto; //llama a método toString
var_dump($objeto); //imprime el tipo y valor de todos los atributos
```

- **Crear una clase que hereda de otra (normal o abstracta: abstract class)**

```
include_once 'ClasePadre.php'; //se deben incluir la clase de la que herede
class ClaseHija extends ClasePadre {
  prívate $atributo2;
  public function __construct($atr1, $atr2) {
        parent::__construct($atr1); //llama al método constructor de la clase padre
      $this->atributo2=$atr2;
  }
  public function nombreFuncion() { //si el nombre existe en la clase padre, lo redefine
        return parent::metodoClasePadre()."texto"; //llamada a un método padre
  }
}
```

- **Atributos y métodos de clase**

```
class NombreClase {
  private static $atributoEstatico = valor; // atributo de clase
  public static function nombreFuncionEstatica() { // método de clase
        return nombreClase::$atributoEstatico;
  }
        prívate $atributo; // atributo de instancia
  public function nombreFuncion($atr){ // método de instancia
        $this->atributo = $atr;
        NombreClase::$atributoEstatico += $atr; // acceso a un atributo estático
  }
}
```

- **Aclaraciones**
  - El acceso a un método de clase (estático) es con :: en vez de -> (eje: Coche::kmsTotales)
  - La sentencia require_once es idéntica a require excepto que PHP verificará si el archivo ya ha sido incluido y si es así, no se incluye (require) de nuevo.
  - require es idéntico a include excepto que en caso de fallo producirá un error fatal de nivel E_COMPILE_ERROR. En otras palabras, éste detiene el script mientras que include sólo emitirá una advertencia (E_WARNING) lo cual permite continuar el script.
  - Para almacenar un objeto en una base de datos o en una variable de sesion, primero hay que serializarlo para convertirlo en texto y para recuperar el objeto a partir del texto hay que unserilizarlo:

    ```
    $cadena= serialize($objeto); //$cadena se guarda como texto en la BD o en la sesion
    $objeto= unserialize($cadena); //se recupera el objeto a partir de la cadena
    ```

Visiualizar u ocultar errores y warnings

Instrucción a colocar al principio del php, para configurar los avisos de errores:

error_reporting (número);

Ejemplo: error_reporting (5);

El número se calcula según los tipos de errores que se deseen mostrar en la página con la siguiente combinación:

- 1 - Errores Normales de Funciones (Normal Function Errors)
- 2 - Avisos Normales (Normal Warnings)
- 4 - Errores del Analizador de código (Parser Errors)
- 8 - Avisos (Notices, advertencia que puedes ignorar, pero que puede implicar un error en tu código.

Ejemplos: 1+4=5 solo errores; 1+4+2=7 errores y warnings (Valor por defecto)

TEMA 10. Modelo Vista Controlador

TEMA 11. TWIG – Motor de plantillas

TEMA 12. Servicios Web

Example#2 **http_build_query()** con elementos indexados numéricamente.

```
<?php
$datos = array('foo', 'bar', 'baz', 'boom', 'vaca' => 'leche',
               'php' => 'procesador de hipertexto');

echo http_build_query($datos) . "\n";
echo http_build_query($datos, 'mivar_');
?>
```

El resultado del ejemplo seria:

```
0=foo&1=bar&2=baz&3=boom&vaca=leche&php=procesador+de+hipertexto
mivar_0=foo&mivar_1=bar&mivar_2=baz&mivar_3=boom&vaca=leche&php=proces
ador+de+hipertexto
```

SublimeText plugins

Package Control (principal para los demás)
Autofilename
codeformatter (indicar ruta php.exe en default settings del plugin o incluir en path del sistema)
Emmet
Php Companion
SideBarEnhancements
SublimeCodeintel
BracketHighLighter
(Mirar video explicativo https://youtu.be/IyjBAZ9uZHc)
*plugins que no viene en el video:
php getters and setters
php-twig
HTMLbeautify (formatea el código con tabulaciones)

LARAVEL

- Las rutas se definen en: ‘routes\web.php’
- Los controladores se crean en la carpeta ‘app\Http\Controllers’
- Las vistas se crean en la carpeta ‘resources\views'
- Los datos de conexión a la BD se configuran en el archivo ‘.env’
- Las clases de las tablas de la BD se crean en la carpeta ‘app’