

---

# Introducción a PHP

Implantación de Aplicaciones Web

Curso 2025/2026

# Índice general

|          |   |          |
|----------|---|----------|
| <b>1</b> | <b>Introducción a PHP</b>                                 | <b>1</b> |
| 1.1      | Entorno de desarrollo con Docker                          | 1        |
| 1.1.1    | Preparación del entorno de desarrollo                     | 1        |
| 1.1.2    | Cómo crear un contenedor Docker con persistencia de datos | 1        |
| 1.1.3    | Crear un entorno de desarrollo LAMP con Docker Compose    | 2        |
| 1.2      | Conceptos básicos   | 2        |
| 1.3      | ¿Qué es PHP?  | 2        |
| 1.4      | ¿Cómo funciona PHP?                                       | 3        |
| 1.5      | ¿Qué sitios web utilizan PHP?                             | 3        |
| 1.6      | Frameworks PHP  | 3        |
| 1.7      | Sintaxis básica   | 4        |
| 1.7.1    | Etiquetas de apertura y cierre de PHP                     | 4        |
| 1.7.2    | Carácter separador de instrucciones                       | 4        |
| 1.7.3    | Comentarios   | 5        |
| 1.8      | PHP embebido en HTML                                      | 5        |
| 1.8.1    | Sintaxis alternativa para estructuras de control          | 6        |
| 1.8.2    | Etiqueta de impresión corta (<?=>)                        | 6        |
| 1.8.3    | Escapado de datos para evitar ataques XSS                 | 6        |
| 1.9      | Tipos de datos  | 7        |
| 1.10     | Variables   | 7        |
| 1.11     | Ámbito de las variables                                   | 7        |
| 1.12     | Constantes  | 7        |
| 1.13     | Cómo visualizar variables                                 | 8        |
| 1.13.1   | echo  | 8        |
| 1.13.2   | var_dump  | 8        |
| 1.13.3   | print_r   | 8        |
| 1.14     | Ejercicios de introducción                                | 9        |
| 1.15     | Estructuras de control (if, else, switch)                 | 10       |
| 1.15.1   | if  | 10       |
| 1.15.2   | if - else   | 10       |
| 1.15.3   | elseif/else if  | 11       |
| 1.15.4   | switch  | 12       |
| 1.15.5   | Ejercicios  | 13       |
| 1.16     | Bucles (for, while, do-while)                             | 14       |
| 1.16.1   | for   | 14       |
| 1.16.2   | while   | 15       |

---

|          |  |           |
|----------|--|-----------|
| 1.16.3   | do - while                                       | 15        |
| 1.16.4   | Ejercicios bucle for                             | 16        |
| 1.16.5   | Ejercicios bucle while                           | 16        |
| 1.16.6   | Ejercicios bucle do - while                      | 17        |
| 1.17     | Arrays y bucles (for, while, do-while y foreach) | 17        |
| 1.17.1   | Arrays con índices                               | 17        |
| 1.17.2   | Consultar su contenido con print_r               | 17        |
| 1.17.3   | Cómo conocer el tamaño de un array con count     | 17        |
| 1.17.4   | Cómo recorrer un array indexado con for          | 18        |
| 1.17.5   | Arrays asociativos                               | 18        |
| 1.17.6   | Cómo recorrer un array asociativo con foreach    | 18        |
| 1.17.7   | Ejercicios                                       | 18        |
| 1.18     | Ejercicios - Funciones                           | 20        |
| 1.19     | Ejercicios - Formularios                         | 21        |
| 1.20     | Ejercicios - Subida de ficheros                  | 22        |
| 1.21     | Ejercicios - Objetos                             | 22        |
| 1.22     | Ejercicios - JSON                                | 23        |
| 1.23     | Acceso a bases de datos                          | 23        |
| 1.23.1   | Uso de MySQLi orientado a objetos                | 23        |
| 1.23.2   | Sentencias preparadas                            | 24        |
| 1.23.3   | Uso de PDO                                       | 24        |
| 1.24     | Ejercicios - Acceso a bases de datos (Consulta)  | 25        |
| 1.25     | Ejercicios - Acceso a bases de datos (Inserción) | 26        |
| 1.26     | Ejercicios - Acceso a bases de datos (Edición)   | 26        |
| 1.27     | Ejercicios - Acceso a bases de datos (Borrado)   | 26        |
| 1.28     | Sesiones   | 26        |
| <b>2</b> | <b>Referencias</b>                               | <b>27</b> |
| <b>3</b> | <b>Licencia</b>                                  | <b>28</b> |

# 1 Introducción a PHP

## 1.1. Entorno de desarrollo con Docker

### 1.1.1. Preparación del entorno de desarrollo

Para el desarrollo de las prácticas vamos a hacer uso de contenedores [Docker](#) y [Docker Compose](#).

Para poder ejecutar contenedores [Docker](#) es necesario tener instalado [Docker Community Edition \(CE\)](#) y [Docker Compose](#) en nuestro equipo.

En la web oficial encontrará la información necesaria para realizar la instalación de [Docker CE](#) sobre [Windows](#), [macOS](#), [Ubuntu](#), [Debian](#), [Fedora](#) y [CentOS](#).

### 1.1.2. Cómo crear un contenedor Docker con persistencia de datos

Si queremos que los datos del contenedor sean persistentes tenemos que crear un **volumen** donde vamos a indicar el directorio de nuestra máquina local que queremos vincular con el directorio `/var/www/html` del contenedor [Docker](#), que es el directorio que utiliza [Apache HTTP Server](#) por defecto para servir una página web.

El comando que podríamos usar para lanzar nuestro contenedor [Docker](#) es el siguiente:

```
1 docker run -d --rm --name my-apache-php-app -p 80:80 -v "$PWD":/var/www/html
  php:7.4-apache
```

- `docker run` es el comando que nos permite crear un contenedor a partir de una imagen Docker.
- El parámetro `-d` nos permite ejecutar el contenedor en modo *detached*, es decir, ejecutándose en segundo plano.
- El parámetro `--rm` hace que cuando salgamos del contenedor, éste se elimine y no ocupe espacio en nuestro disco.
- El parámetro `--name` nos permite asignarle un nombre a nuestro contenedor. Si no le asignamos un nombre [Docker](#) nos asignará un nombre automáticamente.
- El parámetro `-p` nos permite mapear los puertos entre nuestra máquina local y el contenedor. En este caso, estamos mapeando el puerto 80 de nuestra máquina local con el puerto 80 del contenedor.
- `php:7.4-apache` es el nombre de la imagen y la versión que vamos a utilizar para crear el contenedor. Si no se indica lo contrario buscará las imágenes en el repositorio oficial [Docker Hub](#). La imagen `php:7.4-apache` contiene un **servidor Apache HTTP** y los módulos de **PHP 7.4** necesarios.

### 1.1.3. Crear un entorno de desarrollo LAMP con Docker Compose

Si queremos crear un entorno de desarrollo con la pila LAMP completa podemos hacer uso del repositorio que se indica a continuación, que contiene la configuración necesaria para crear el entorno con Docker Compose.

- <https://github.com/josejuansanchez/lamp-docker>

El archivo `docker-compose.yml` del repositorio contiene la definición de tres servicios:

- Servidor web apache configurado con los módulos necesarios para ejecutar código PHP y poder conectar desde PHP a MySQL.
- Sistema gestor de bases de datos relacionales MySQL.
- Servidor web con phpMyAdmin para poder administrar MySQL desde una interfaz web.

Para iniciar los servicios en su entorno de desarrollo, en primer lugar tendrá que clonar el repositorio en su equipo.

```
1 git clone https://github.com/josejuansanchez/lamp-docker
```

Una vez que haya clonado el repositorio en su equipo sólo tiene que acceder al directorio del repositorio.

```
1 cd lamp-docker
```

Para iniciar los servicios que se han definido en el archivo `docker-compose.yml` puede ejecutar el siguiente comando.

```
1 docker-compose up -d
```

- El parámetro `-d` nos permite ejecutar los contenedores en modo *detached*, es decir, ejecutándose en segundo plano.

Para detener los servicios podemos hacerlo con el comando:

```
1 docker-compose down
```

Si quisiéramos eliminar el volumen que hemos definido para MySQL podemos utilizar el parámetro `-v`. De modo que el comando que tendríamos que ejecutar sería el siguiente:

```
1 docker-compose down -v
```

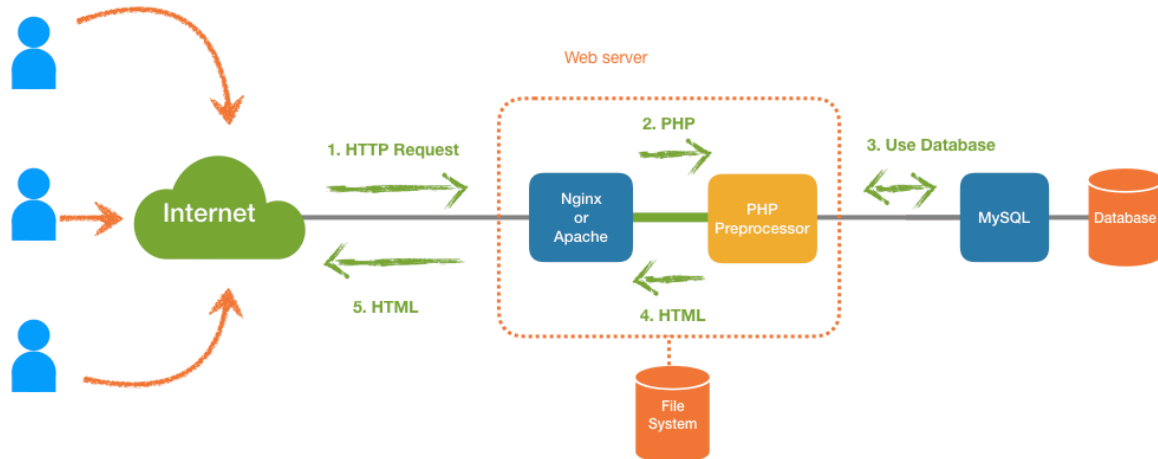
## 1.2. Conceptos básicos

### 1.3. ¿Qué es PHP?

PHP es un lenguaje de programación de uso general, especialmente adecuado para el desarrollo web.

El código PHP puede ser interpretado y ejecutado desde la interfaz de línea de comandos (CLI) o desde un servidor web que tenga implementado un intérprete PHP.

## 1.4. ¿Cómo funciona PHP?



<https://josejuansanchez.org>

## 1.5. ¿Qué sitios web utilizan PHP?

En la actualidad, PHP está siendo utilizado en gran cantidad de sitios web. Entre los sitios web más destacados podemos encontrar:

- [Wordpress](#)
- [PrestaShop](#)
- [Drupal](#)
- [Joomla](#)

Sitios como **Facebook** o **Wikipedia**, hacen uso del [lenguaje de programación Hack](#) que es una extensión de PHP y se ejecuta en la [máquina virtual HHVM](#) (HipHop Virtual Machine).

## 1.6. Frameworks PHP

Un *framework* es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. (Fuente: [Wikipedia](#))

A la hora de desarrollar software es muy común hacer uso de *frameworks* ya que nos ayudarán a reducir a cantidad de código que tenemos que escribir y nos proporcionarán una gran cantidad de bibliotecas para realizar funciones de uso común.

Algunos de los *frameworks* PHP más utilizados actualmente son:

- [Laravel](#).
- [Symfony](#).
- [CodeIgniter](#).
- [Yii](#).
- [Slim](#).
- [CakePHP](#).
- [Zend Framework](#).

## 1.7. Sintaxis básica

### 1.7.1. Etiquetas de apertura y cierre de PHP

El código PHP se encierra entre las etiquetas de apertura y cierre: `<?php` y `?>`. Todo el código que se encuentre entre estas dos etiquetas será interpretado como código PHP.

**Ejemplo:**

```
1 <?php
2
3 echo "¡Hola mundo!";
4
5 ?>
```

Es posible omitir la etiqueta de cierre `?>` cuando el contenido del archivo sólo sea código PHP.

**Ejemplo:**

```
1 <?php
2
3 echo "¡Hola mundo!";
```

### 1.7.2. Carácter separador de instrucciones

Todas las instrucciones en PHP terminan con el carácter punto y coma (;).

**Ejemplo:**

```
1 <?php
2     echo "¡Hola ";
3     echo "mundo!";
4 ?>
```

El único caso donde se puede omitir el carácter punto y coma (;) en la última instrucción de un bloque PHP, ya que la etiqueta de cierre de un bloque PHP (?>) implica un punto y coma.

**Ejemplo:**

```
1 <?php
2     echo "¡Hola ";
3     echo "mundo!";
4 ?>
```

### 1.7.3. Comentarios

Podemos escribir comentarios de una sola línea con: // y #, y comentarios multilínea con /\* ... \*/.

**Ejemplo:**

```
1 <?php
2     // Esto es un comentario de una línea
3     echo "Frase 1";
4
5     # Esto es otro comentario de una línea
6     echo "Frase 2";
7
8     /* Este comentario
9     es un comentario de múltiples líneas */
10    echo "Frase 3";
11 ?>
```

## 1.8. PHP embebido en HTML

El uso de estas etiquetas de apertura y cierre, nos permite embeber código PHP en documentos HTML. De modo que sólo el código que aparezca entre las etiquetas <?php y ?> será interpretado por el intérprete de PHP y el resto de etiquetas serán ignoradas.

**Ejemplo de código PHP embebido en un documento HTML:**

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Ejemplo</title>
5 </head>
6 <body>
7     <p>Contenido escrito en HTML</p>
8     <?php echo "Contenido escrito desde PHP"; ?>
9 </body>
10 </html>
```

### 1.8.1. Sintaxis alternativa para estructuras de control

PHP ofrece una **sintaxis alternativa** para las estructuras de control (**if**, **while**, **for**, **foreach** y **switch**) que es mucho más legible cuando se mezcla con HTML, ya que evita el uso excesivo de llaves { } que pueden perderse entre etiquetas HTML.

- En lugar de abrir una llave { se usan dos puntos :.
- En lugar de cerrar con } se usa `endif;`, `endforeach;`, `endwhile;`, `endfor;` o `endswitch;`.

#### Ejemplo de **if**:

```
1 <?php if ($usuario_logueado): ?>
2   <p>Bienvenido!</p>
3 <?php else: ?>
4   <a href="login.php">Iniciar sesión</a>
5 <?php endif; ?>
```

#### Ejemplo de **for**:

```
1 <ul>
2   <?php for ($i = 1; $i <= 10; $i++): ?>
3     <li>Elemento <?= $i ?></li>
4   <?php endfor; ?>
5 </ul>
```

### 1.8.2. Etiqueta de impresión corta (<?=>

Para mostrar valores de variables en el HTML, se recomienda usar la etiqueta corta `<?= $variable ?>` en lugar de la etiqueta completa `<?php echo $variable; ?>`. Es más limpia y reduce el ruido visual en las vistas.

#### Ejemplo:

```
1 <p>Bienvenido, <?= $nombre ?></p>
```

### 1.8.3. Escapado de datos para evitar ataques XSS

Cualquier dato que provenga de una fuente externa como una base de datos, entrada del usuario, una API, etc. y se vaya a mostrar en una página HTML debe ser escapado para prevenir ataques de **Cross-Site Scripting (XSS)**.

En PHP, la forma estándar de hacerlo es mediante la función `htmlspecialchars()`.

Esta función convierte caracteres especiales como `<` o `>` en sus entidades HTML equivalentes (`&lt;` y `&gt;`), evitando que el navegador los interprete como etiquetas de código malicioso.

#### Ejemplo:

```
1 <p>Nombre: <?= htmlspecialchars($nombre) ?></p>
```

## 1.9. Tipos de datos

[Documentación oficial de PHP.](#)

## 1.10. Variables

Las variables en PHP se representan con el símbolo del dólar (\$) seguido por el nombre de la variable. El nombre de la variable es sensible a mayúsculas y minúsculas.

### Ejemplo:

```
1 <?php
2     $nombre = "Pepe";
3     $edad = 30;
4 ?>
```

[Documentación oficial de PHP.](#)

## 1.11. Ámbito de las variables

[Documentación oficial de PHP.](#)

## 1.12. Constantes

Las constantes no pueden modificar su valor durante la ejecución del script. El nombre de la constante **no tiene** que ir precedido por el símbolo del dólar y es sensible a mayúsculas y minúsculas. Por convención, las constantes siempre se declaran en mayúsculas.

### Ejemplo:

```
1 <?php
2     // Ejemplo de una constante numérica de tipo real
3     define("PI", 3.141592);
4
5     // Ejemplo de una constante de tipo string
6     define("CONSTANTE", "Hola mundo");
7 ?>
```

[Documentación oficial de PHP.](#)

## 1.13. Cómo visualizar variables

### 1.13.1. echo

`echo` es una construcción del lenguaje (no es una función) que nos permite mostrar cadenas de texto y el contenido de las variables.

#### Ejemplo:

```
1 <?php
2
3 echo "Esto es una cadena de texto.";
4
5 // Con echo también podemos mostrar el contenido de una variable
6 $numero = 10;
7
8 echo "El contenido de la variable es: $numero";
9
10 ?>
```

[Documentación oficial de echo.](#)

### 1.13.2. var\_dump

`var_dump` es una función que nos permite mostrar el contenido de una variable. Esta función muestra el tipo de dato y el valor de la variable.

#### Ejemplo:

```
1 <?php
2
3 $nombre = "Pepe";
4 $edad = 30;
5 $nota = 7.5;
6
7 var_dump($nombre);
8 // string(4) "Pepe"
9
10 var_dump($edad);
11 // int(30)
12
13 var_dump($nota);
14 // float(7.5)
15
16 ?>
```

[Documentación oficial de la función var\\_dump.](#)

### 1.13.3. print\_r

`print_r` es una función que nos permite mostrar el contenido de una variable de una forma legible.

**Ejemplo:**

```
1 <?php
2
3 $lista = array("Pepe", "María", "Juan");
4
5 print_r($lista);
6
7 //Array
8 //(
9 //     [0] => Pepe
10 //     [1] => María
11 //     [2] => Juan
12 //)
13
14 ?>
```

[Documentación oficial de la función print\\_r.](#)

## 1.14. Ejercicios de introducción

### 1. Escribe un script PHP que muestre información sobre la configuración de PHP que hay en el servidor.

*Notas:*

- [Documentación oficial de la función phpinfo.](#)

### 2. Revise la documentación oficial de PHP para ver qué información podemos obtener de la variable *superglobal* `$_SERVER`. Escribe un script haciendo uso de la variable *superglobal* `$_SERVER` que muestre lo siguiente:

- La dirección IP del servidor donde se está ejecutando el script.
- El nombre del host del servidor donde se está ejecutando el script.
- El software que está utilizando el servidor para servir el script.
- Información sobre el agente de usuario (*User Agent*) desde el que se está solicitando el script.
- La dirección IP del cliente que está solicitando el script.

*Notas:*

- [Documentación oficial de la variable \*superglobal\* `\$\_SERVER`.](#)

### 3. Revise la documentación oficial para conocer todas las variables *superglobals* que existen. Con ayuda de la función `print_r` muestra el contenido de cada una de las variables *superglobals*.

*Notas:*

- [Documentación oficial de las variables \*superglobals\*.](#)
- [Documentación oficial de la función print\\_r.](#)

## 1.15. Estructuras de control (if, else, switch)

### 1.15.1. if

La estructura de control **if** permite la ejecución condicional de fragmentos de código PHP.

Sintaxis:

```
1 <?php
2
3 if (condicion_es_cierta) {
4     bloque_de_sentencias
5 }
6
7 ?>
```

#### Ejemplo:

El siguiente ejemplo mostraría **a es mayor que b** si la variable **\$a** es mayor que **\$b**:

```
1 <?php
2
3 $a = 10;
4 $b = 5;
5
6 if ($a > $b) {
7     echo "a es mayor que b";
8 }
9
10 ?>
```

### 1.15.2. if - else

Sintaxis:

```
1 <?php
2
3 if (condicion_es_cierta) {
4     bloque_de_sentencias_1
5 } else {
6     bloque_de_sentencias_2
7 }
8
9 ?>
```

#### Ejemplo:

El siguiente ejemplo puede mostrar **a es mayor que b** si la variable **\$a** es mayor que **\$b** o **a NO es mayor que b** en caso contrario.

```
1 <?php
2
3 $a = 1;
```

```
4 $b = 5;
5
6 if ($a > $b) {
7     echo "a es mayor que b";
8 } else {
9     echo "a NO es mayor que b";
10 }
11
12 ?>
```

### 1.15.3. elseif/else if

Sintaxis `elseif`:

```
1 <?php
2
3 if (condición_1) {
4     bloque_de_sentencias_1
5 } elseif (condición_2) {
6     bloque_de_sentencias_2
7 } else {
8     bloque_de_sentencias_3
9 }
10
11 ?>
```

Sintaxis `else if`:

```
1 <?php
2
3 if (condición_1) {
4     bloque_de_sentencias_1
5 } else if (condición_2) {
6     bloque_de_sentencias_2
7 } else {
8     bloque_de_sentencias_3
9 }
10
11 ?>
```

**Ejemplo:**

```
1 <?php
2
3 $a = 1;
4 $b = 1;
5
6 if ($a > $b) {
7     echo "a es mayor que b";
8 } elseif ($a == $b) {
9     echo "a es igual que b";
10 } else {
11     echo "a es menor que b";
12 }
```

```
13
14 ?>
```

### 1.15.4. switch

Sintaxis:

```
1 <?php
2
3 switch ($variable) {
4     case valor1:
5         bloque_de_sentencias_1
6         break;
7
8     case valor2:
9         bloque_de_sentencias_2
10        break;
11
12    default:
13        bloque_de_sentencias_3
14 }
15
16 ?>
```

Esta estructura de control es equivalente a:

```
1 <?php
2
3 if ($variable == valor1) {
4     bloque_de_sentencias_1
5 } elseif ($variable == valor2) {
6     bloque_de_sentencias_2
7 } else {
8     bloque_de_sentencias_3
9 }
10
11 ?>
```

**Ejemplo:**

```
1 <?php
2
3 $numero = 2;
4
5 switch ($numero) {
6     case 1:
7         echo "La variable es igual a 1";
8         break;
9
10    case 2:
11        echo "La variable es igual a 2";
12        break;
13
14    default:
```

```
15     echo "La variable es un número distinto a 1 y 2";
16 }
17
18 ?>
```

### 1.15.5. Ejercicios

**1. Escribe un script que simule el comportamiento de lanzar una moneda al aire y muestre una imagen con la cara o la cruz de la moneda.**

*Notas:*

- [Documentación de la función rand.](#)
- [Documentación de la estructura de control if.](#)
- [Documentación de la estructura de control else.](#)
- [Documentación del elemento de imagen <img> en HTML.](#)

**2. Escribe un script PHP que genere un número aleatorio entre 1 y 10, simulando una nota numérica y muestre un mensaje indicando la calificación obtenida teniendo en cuenta los siguientes rangos:**

- Insuficiente: [0, 5)
- Suficiente: [5, 6)
- Bien: [6, 7)
- Notable: [7, 9)
- Sobresaliente: [9, 10]

*Notas:*

- [Documentación de la estructura de control if.](#)
- [Documentación de la estructura de control else.](#)

**3. Escribe un script PHP que genere un número aleatorio entre 1 y 7, y muestre un mensaje indicando a qué día de la semana corresponde. Por ejemplo, 1 sería lunes, 2 martes, etc.**

*Notas:*

- [Documentación de la estructura de control switch.](#)

**4. Escribe un script PHP que realice la simulación de lanzar un dado y muestre una imagen con un valor aleatorio entre 1 y 6. Resuelva el ejercicio utilizando la estructura de control if - else.**

*Notas:*

- [Documentación de la función rand.](#)
- [Documentación de la estructura de control if.](#)
- [Documentación de la estructura de control else.](#)
- [Documentación del elemento de imagen <img> en HTML.](#)

*Imágenes:* 1, 2, 3, 4, 5 y 6.

**5. Escribe un script PHP que realice la simulación de lanzar un dado y muestre una imagen con un valor aleatorio entre 1 y 6. Resuelva el ejercicio utilizando la estructura de control `switch`.**

Notas:

- Documentación de la función `rand`.
- Documentación de la estructura de control `switch`.
- Documentación del elemento de imagen `<img>` en HTML.

Imágenes: 1, 2, 3, 4, 5 y 6.

**6. Escribe un script PHP que realice la simulación de lanzar un dado y muestre una imagen con un valor aleatorio entre 1 y 6. Resuelva el ejercicio sin utilizar las estructuras de control `if - else` y `switch`.**

**7. Escribe un script PHP que realice la simulación de lanzar dos dados y muestre una imagen con los valores obtenidos en cada uno de los dados.**

## 1.16. Bucles (`for`, `while`, `do-while`)

### 1.16.1. `for`

Sintaxis:

```
1 for (expr1; expr2; expr3) {
2     sentencias;
3 }
```

**Ejemplo:**

El siguiente ejemplo muestra los números del 1 al 10.

```
1 <?php
2
3 for ($i = 1; $i <= 10; $i++) {
4     echo $i;
5     echo "<br>";
6 }
7
8 ?>
```

**Ejemplo:**

El siguiente ejemplo muestra los números del 10 al 1.

```
1 <?php
2
3 for ($i = 10; $i >= 1; $i--) {
4     echo $i;
5     echo "<br>";
6 }
7
8 ?>
```

### 1.16.2. while

Sintaxis:

```
1 while (condicion_es_verdadera) {
2     sentencias;
3 }
```

#### Ejemplo:

El siguiente ejemplo muestra los números del 1 al 10.

```
1 <?php
2
3 $i = 1;
4 while ($i <= 10) {
5     echo $i;
6     echo "<br>";
7     $i++;
8 }
9
10 ?>
```

#### Ejemplo:

El siguiente ejemplo muestra los números del 10 al 1.

```
1 <?php
2
3 $i = 10;
4 while ($i >= 1) {
5     echo $i;
6     echo "<br>";
7     $i--;
8 }
9
10 ?>
```

### 1.16.3. do - while

```
1 do {
2     sentencias;
3 } while (condicion_es_verdadera)
```

#### Ejemplo:

El siguiente ejemplo muestra los números del 1 al 10.

```
1 <?php
2
3 $i = 1;
4 do {
5     echo $i;
6     echo "<br>";
```

```
7     $i++;
8 } while ($i <= 10);
9
10 ?>
```

**Ejemplo:**

El siguiente ejemplo muestra los números del 10 al 1.

```
1 <?php
2
3 $i = 10;
4 do {
5     echo $i;
6     echo "<br>";
7     $i--;
8 } while ($i >= 1);
9
10 ?>
```

#### 1.16.4. Ejercicios bucle for

**1. Escribe un script PHP que muestre los números del 1 al 10 en una tabla de una fila y 10 columnas. Utiliza un bucle for**

*Notas:*

- [Documentación del bucle for](#).
- [Tablas HTML](#).

**2. Escribe un script PHP que muestre los números del 1 al 10 en una tabla de una columna y 10 filas. Utiliza un bucle for.**

**3. Escribe un script PHP que muestre en una tabla los números pares que existen entre 1 y 100. Utiliza un bucle for.**

**4. Escribe un script PHP que muestre la tabla de multiplicar de un número aleatorio. Utiliza un bucle for**

**5. Escribe un script PHP que muestre las tablas de multiplicar del 1 al 10. Utiliza un bucle for**

#### 1.16.5. Ejercicios bucle while

**1. Escribe un script PHP que muestre los números del 1 al 10 en una tabla de una fila y 10 columnas. Utiliza un bucle while**

*Notas:*

- [Documentación de la función while](#).
- [Tablas HTML](#).

### 1.16.6. Ejercicios bucle do - while

**1. Escribe un script PHP que muestre los números del 1 al 10 en una tabla de una fila y 10 columnas. Utiliza un bucle while**

Notas:

- [Documentación de la función do-while.](#)
- [Tablas HTML.](#)

## 1.17. Arrays y bucles (for, while, do-while y foreach)

Un array es una estructura de datos que nos permite almacenar varios valores en una única variable.

### 1.17.1. Arrays con índices

Para crear un array es suficiente con hacer:

```
1 $productos = array();
```

Existen dos formas de inicializar los valores de un array indexado:

```
1 $productos = array("Disco SSD", "Memoria RAM", "Monitor");
```

o también asignando los valores posición a posición:

```
1 $productos[0] = "Disco SSD";  
2 $productos[1] = "Memoria RAM";  
3 $productos[2] = "Monitor";
```

Tenga en cuenta que los arrays siempre empiezan por la posición 0.

### 1.17.2. Consultar su contenido con print\_r

```
1 print_r($productos);
```

### 1.17.3. Cómo conocer el tamaño de un array con count

```
1 $productos = array("Disco SSD", "Memoria RAM", "Monitor");  
2 echo count($productos);
```

#### 1.17.4. Cómo recorrer un array indexado con for

```
1 $productos = array("Disco SSD", "Memoria RAM", "Monitor");
2 $numero_de_elementos = count($productos);
3
4 for ($i = 0; $i < $numero_de_elementos; $i++ ) {
5     echo $productos[$i];
6     echo "<br>";
7 }
```

#### 1.17.5. Arrays asociativos

Los arrays asociativos nos permiten usar claves en lugar de índices, para acceder a los valores del array.

Existen dos formas de inicializar los valores de un array asociativo:

```
1 $edades = array("Juan" => "25", "María" => "28", "Paco" => "27");
```

o también asignando los valores a cada clave:

```
1 $edades["Juan"] = "35";
2 $edades["María"] = "35";
3 $edades["Paco"] = "35";
```

#### 1.17.6. Cómo recorrer un array asociativo con foreach

```
1 $edades = array("Juan" => "25", "María" => "28", "Paco" => "27");
2
3 foreach ($edades as $clave => $valor) {
4     echo "Clave: " . $clave . " - Valor: " . $valor;
5     echo "<br>";
6 }
```

#### 1.17.7. Ejercicios

##### 1. Escribe un script PHP que realice las siguientes acciones:

- Inicializar un array de 10 elementos, con valores aleatorios entre 1 y 30.
- Una vez que ha inicializado el array, imprimir todos los valores que almacena.

##### 2. Escribe un script PHP que realice las siguientes acciones:

- Inicializar un array de 10 elementos, con valores aleatorios entre 1 y 30.
- Una vez que ha inicializado el array, imprima todos los valores que almacena.
- Calcular el **valor medio** de los valores del array.
- Mostrar el valor medio que ha calculado.

##### 3. Escribe un script PHP que realice las siguientes acciones:

- Inicializar un array de 10 elementos, con valores aleatorios entre 1 y 30.
- Una vez que ha inicializado el array, imprima todos los valores que almacena.
- Buscar el **valor máximo** de los valores del array.
- Muestre el valor máximo que ha encontrado.

#### 4. Escribe un script PHP que realice las siguientes acciones:

- Inicializar un array de 10 elementos, con valores aleatorios entre 1 y 30.
- Una vez que ha inicializado el array, imprima todos los valores que almacena.
- Buscar el **valor mínimo** de los valores del array.
- Muestre el valor mínimo que ha encontrado.

#### Escribe un script PHP que sobre un array de temperaturas realice las siguientes operaciones:

- Calcular la media.
- Calcular el valor máximo.
- Calcular el valor mínimo.
- Mostrar todos los valores calculados.

El array de temperaturas lo vamos a generar con números aleatorios. El array será de 10 elementos y los valores aleatorios generados estarán entre 1 y 30.

#### 5. Resuelva el ejercicio utilizando bucles **for**.

Notas:

- [Documentación de la función \*\*for\*\*](#).

#### 6. Resuelva el ejercicio utilizando bucles **while**.

Notas:

- [Documentación de la función \*\*while\*\*](#).

#### 7. Resuelva el ejercicio utilizando bucles **do-while**.

Notas:

- [Documentación de la función \*\*do-while\*\*](#).

#### 8. Escribe un script PHP que sobre un array de temperaturas realice las siguientes operaciones:

- Mostrar el listado ordenado de mayor a menor.
- Mostrar el listado ordenado de menor a mayor.

El array de temperaturas lo vamos a generar con números aleatorios. El número de elementos del array será especificado mediante un formulario y los valores aleatorios generados estarán entre 1 y 30.

Notas:

- [Documentación de la función \*\*sort\*\*](#).
- [Documentación de la función \*\*rsort\*\*](#).

#### 9. Escribe un script PHP que permita ordenar el siguiente array asociativo:

```
1 array("Antonio"=>"31", "María"=>"28", "Juan"=>"29", "Pepe"=>"27")
```

- De forma ascendente ordenado por valor.
- De forma ascendente ordenado por clave.
- De forma descendente ordenado por valor.
- De forma descendente ordenado por clave.

Notas:

- [Documentación de la función asort.](#)
- [Documentación de la función arsort.](#)
- [Documentación de la función ksort.](#)
- [Documentación de la función krsort.](#)

**10. Escribe un script PHP que muestre el siguiente array asociativo ordenado por la clave. El resultado deberá seguir el siguiente patrón:**

```
1 La capital de ITALIA es ROMA
```

**Tenga en cuenta que tendrá que utilizar una función para convertir las claves y los valores del array en mayúscula.**

```
1 array("Italy"=>"Rome", "Luxembourg"=>"Luxembourg", "Belgium"=>"Brussels", "Denmark"=>"Copenhagen", "Finland"=>"Helsinki", "France" =>"Paris", "Slovakia"=>"Bratislava", "Slovenia"=>"Ljubljana", "Germany" =>"Berlin", "Greece" =>"Athens", "Ireland"=>"Dublin", "Netherlands"=>"Amsterdam", "Portugal"=>"Lisbon", "Spain"=>"Madrid", "Sweden"=>"Stockholm", "United Kingdom"=>"London", "Cyprus"=>"Nicosia", "Lithuania"=>"Vilnius", "Czech Republic"=>"Prague", "Estonia"=>"Tallin", "Hungary"=>"Budapest", "Latvia"=>"Riga", "Malta"=>"Valetta", "Austria" =>"Vienna", "Poland"=>"Warsaw");
```

Notas:

- [Documentación de la función foreach.](#)
- [Documentación de la función strtoupper.](#)

**11. Escribe un script PHP que convierta el array del ejercicio anterior en un objeto JSON.**

Notas:

- [JSON \(JavaScript Object Notation\).](#)
- [Documentación de la función json\\_encode.](#)

## 1.18. Ejercicios - Funciones

**1. Escribe una función que reciba un número como parámetro de entrada y que imprima su tabla de multiplicar.**

Notas:

- [Funciones definidas por el usuario en PHP.](#)

2. Escribe una función que reciba dos parámetros de entrada (inicio y fin) y que imprima las tablas de multiplicar entre esos dos números. Utilice la función del ejercicio anterior.

3. Escribe una función llamada `inicializar_array` que reciba tres parámetros llamados `numero_de_elementos`, `min` y `max`, y que devuelva un array de números enteros comprendidos entre los valores `min` y `max`. El número de elementos que contiene el array será el especificado en el parámetro de entrada `numero_de_elementos`

Notas:

- [Cómo devolver valores en una función PHP.](#)

4. Escribe una función llamada `calcular_media` que reciba un array como parámetro de entrada y que devuelva la media de todos los valores que contiene.

5. Escribe una función llamada `calcular_maximo` que reciba un array como parámetro de entrada y que devuelva cuál es el máximo valor del array.

6. Escribe una función llamada `calcular_minimo` que reciba un array como parámetro de entrada y que devuelva cuál es el mínimo valor del array.

7. Escribe una función llamada `imprimir_array` que reciba un array como parámetro de entrada y muestre su contenido en una tabla con dos columnas. La primera columna mostrará la posición del array y la segunda el valor que hay en esa posición.

8. Crea un archivo llamado `funciones.php` que contenga todas las funciones creadas en los ejercicios anteriores. Escriba un script PHP que incluya el archivo `funciones.php` y haga uso de cada una de ellas.

Notas:

- [Documentación de la sentencia `include` en PHP.](#)
- [Documentación de la sentencia `require` en PHP.](#)

## 1.19. Ejercicios - Formularios

1. Escribe un script que muestre un formulario con un campo de texto y que permita enviarlo usando el método `GET`. El mismo script será capaz de recibir el dato enviado por el formulario y lo mostrará.

Notas:

- [Documentación de la variable `superglobal` `\$\_GET`.](#)
- [Documentación de la función `empty`.](#)
- [Formularios HTML.](#)

2. Escribe un script que muestre un formulario con un campo de texto y que permita enviarlo usando el método `POST`. El mismo script será capaz de recibir el dato enviado por el formulario y lo mostrará.

Notas:

- Documentación de la variable *superglobal* `$_POST`.
- Documentación de la función `empty`.
- Formularios HTML.

**3. Escribe un script que muestre un formulario que permita introducir un número y mostrar su tabla de multiplicar.**

**4. Escribe un script que mediante un formulario permita seleccionar el número de monedas que se desean lanzar (de 1 a 20) y el tipo de moneda (Dólar Estadounidense, Euro, Yen japonés, Libra esterlina, Franco suizo). El comportamiento tiene que ser similar al de la web [random.org](https://random.org).**

*Notas:*

- Documentación de la función `rand`.
- Documentación de la estructura de control `if`.
- Documentación de la estructura de control `else`.
- Documentación del bucle `for`.
- Documentación sobre el tipo `array` en PHP.
- Documentación del elemento de imagen `<img>` de HTML.
- Documentación del elemento `<select>` de HTML.

## 1.20. Ejercicios - Subida de ficheros

**1. Escribe un script PHP que mediante un formulario solamente permita subir archivos de imágenes.**

*Notas:*

- Documentación sobre subida de ficheros en PHP.

## 1.21. Ejercicios - Objetos

Recursos:

- Documentación oficial sobre clases y objetos en PHP.
- Introducción a la Programación Orientada a Objetos (POO) en PHP por Diego Lázaro.

1. Define una clase llamada `Persona` que cumpla los siguientes requisitos:

- Debe tener las siguientes **propiedades privadas**:
  - `nombre`
  - `apellido1`
  - `apellido2`
  - `edad`
- Debe tener un **constructor** que permita inicializar los valores de las propiedades.
- Debe tener **métodos públicos** `get` y `set` para cada una de las propiedades.

- Debe incluir un **método público llamado imprimir** que muestre todas las propiedades del objeto.

Una vez definida la clase, realice dos instancias y utilice todos los métodos que ha creado.

## 1.22. Ejercicios - JSON

Recursos:

- [Documentación de la función json\\_decode.](#)
- [Documentación de la función json\\_encode.](#)

1. Escribe un script que haga uso de la [API de OpenWeatherMap](#) y muestre la previsión metereológica de la ciudad que se indique en un formulario web.

## 1.23. Acceso a bases de datos

Para interactuar con una base de datos MySQL o MariaDB desde PHP, existen principalmente dos extensiones que se utilizan hoy en día: **MySQLi** (MySQL Improved) y **PDO** (PHP Data Objects).

- **MySQLi:** Es una extensión específica para MySQL. Permite trabajar tanto de forma procedimental como orientada a objetos.
- **PDO:** Es una capa de abstracción de bases de datos que permite trabajar con múltiples sistemas (MySQL, PostgreSQL, SQLite, etc.) utilizando la misma interfaz.

### 1.23.1. Uso de MySQLi orientado a objetos

El proceso estándar para trabajar con una base de datos es:

1. **Conectar:** Establecer el enlace con el servidor.
2. **Consultar:** Enviar la sentencia SQL.
3. **Procesar:** Trabajar con los datos devueltos (si es un **SELECT**).
4. **Cerrar:** Liberar la conexión.

**Ejemplo de conexión y consulta básica:**

```
1 <?php
2 // Configuración de las variables relacionadas con la conexión
3 // ...
4
5 // 1. Crear la conexión
6 $conexion = new mysqli($servidor, $usuario, $password, $base_datos);
7
8 // Verificar si hay errores
9 if ($conexion->connect_error) {
10     die("Error de conexión: " . $conexion->connect_error);
11 }
```

```
12
13 // 2. Ejecutar una consulta
14 $sql = "SELECT id, nombre FROM productos";
15 $resultado = $conexion->query($sql);
16
17 if ($resultado->num_rows > 0) {
18     // 3. Recorrer los resultados
19     while($fila = $resultado->fetch_assoc()) {
20         echo "Producto: " . $fila["nombre"] . " (ID: " . $fila["id"] . ")<br>";
21     }
22 }
23
24 // 4. Cerrar la conexión
25 $conexion->close();
26 ?>
```

### 1.23.2. Sentencias preparadas

Podemos mejorar la seguridad de nuestra aplicación haciendo uso de **sentencias preparadas** para evitar ataques de **inyección SQL**.

La forma más profesional de manejar parámetros en nuestras consultas SQL es mediante **sentencias preparadas**, por este motivo debemos **evitar concatenar variables del usuario directamente en la sentencia SQL**.

#### Ejemplo:

```
1 <?php
2 // ...
3
4 // Preparamos la consulta con un marcador de posición (?)
5 $stmt = $conexion->prepare("SELECT nombre FROM productos WHERE id = ?");
6
7 // Vinculamos el parámetro, "i" indica que es un integer
8 $stmt->bind_param("i", $id);
9
10 // Ejecutamos la consulta
11 $stmt->execute();
12
13 // Obtenemos los resultados
14 $resultado = $stmt->get_result();
15
16 // ...
17 ?>
```

### 1.23.3. Uso de PDO

**PDO (PHP Data Objects)** es la opción recomendada por la mayoría de los desarrolladores profesionales, ya que permite trabajar con múltiples tipos de bases de datos mediante la misma interfaz y ofrece un manejo de errores basado en **excepciones**.

#### Ejemplo:

```
1 <?php
2 // Configuración de las variables relacionadas con la conexión
3 // ...
4
5 try {
6     // 1. Definir el DSN (Data Source Name)
7     $dsn = "mysql:host=$host;dbname=$db_name;charset=utf8mb4";
8
9     // 2. Crear la conexión (objeto PDO)
10    $conexion = new PDO($dsn, $user, $pass);
11
12    // Configurar para que lance excepciones en caso de error
13    $conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
14
15    // 3. Preparar la consulta (usando nombres de parámetros -> :id)
16    $stmt = $conexion->prepare("SELECT nombre FROM productos WHERE id = :
17                               id_buscar");
18
19    // 4. Vincular y ejecutar en un solo paso
20    $id_buscado = 5;
21    $stmt->execute(['id_buscar' => $id_buscado]);
22
23    // 5. Obtener resultados
24    $producto = $stmt->fetch(PDO::FETCH_ASSOC);
25
26    if ($producto) {
27        echo "Nombre: " . $producto['nombre'];
28    }
29 } catch (PDOException $e) {
30     // Manejo de errores
31     echo "Error en la base de datos: " . $e->getMessage();
32 }
33
34 // 6. Cerrar (opcional, PHP lo hace solo al terminar)
35 $conexion = null;
36 ?>
```

Una gran ventaja de **PDO** es que permite utilizar **marcadores con nombre** (: nombre\_parametro) en lugar de simples signos de interrogación, lo que hace que las consultas sean mucho más fáciles de leer y mantener.

## 1.24. Ejercicios - Acceso a bases de datos (Consulta)

*Notas:*

- [Documentación sobre la extensión mysqli](#) para acceder a bases de datos.
- [Documentación sobre la extensión PDO](#) para acceder a bases de datos.
- [Ejemplo sencillo de un CRUD](#) (Create, Read, Update, Delete) en PHP y MySQL.
- [Ejemplo sencillo de un CRUD](#) (Create, Read, Update, Delete) con un sistema de login, en PHP y MySQL.
- [Ejemplo de una aplicación LAMP](#) para seleccionar el delegado de clase.

## 1.25. Ejercicios - Acceso a bases de datos (Inserción)

Notas:

- Documentación sobre la extensión `mysqli` para acceder a bases de datos.
- Documentación sobre la extensión `PDO` para acceder a bases de datos.
- Ejemplo sencillo de un **CRUD** (Create, Read, Update, Delete) en PHP y MySQL.
- Ejemplo sencillo de un **CRUD** (Create, Read, Update, Delete) con un sistema de login, en PHP y MySQL.
- Ejemplo de una aplicación LAMP para seleccionar el delegado de clase.

## 1.26. Ejercicios - Acceso a bases de datos (Edición)

Notas:

- Documentación sobre la extensión `mysqli` para acceder a bases de datos.
- Documentación sobre la extensión `PDO` para acceder a bases de datos.
- Ejemplo sencillo de un **CRUD** (Create, Read, Update, Delete) en PHP y MySQL.
- Ejemplo sencillo de un **CRUD** (Create, Read, Update, Delete) con un sistema de login, en PHP y MySQL.

## 1.27. Ejercicios - Acceso a bases de datos (Borrado)

Notas:

- Documentación sobre la extensión `mysqli` para acceder a bases de datos.
- Documentación sobre la extensión `PDO` para acceder a bases de datos.
- Ejemplo sencillo de un **CRUD** (Create, Read, Update, Delete) en PHP y MySQL.
- Ejemplo sencillo de un **CRUD** (Create, Read, Update, Delete) con un sistema de login, en PHP y MySQL.

## 1.28. Sesiones

Notas:

- Documentación sobre el manejo de sesiones en PHP.

## 2 Referencias

Los recursos que vamos a utilizar en esta práctica son los siguientes:

- [PHP. Wikipedia.](#)
- [Manual de PHP oficial.](#)
- [Desarrollo de sitios web con PHP y MySQL.](#)
- [Programación web en PHP.](#)
- [PHP. The right way.](#)
- [Best practices for modern PHP development.](#)
- [PHP Pandas.](#)
- [PHP Examples. w3schools.](#)
- [PHP Tutorials for beginners. w3resource.](#)
- [PHP Examples. w3schools.](#)
- [PHP Tutorials for beginners. w3resource.](#)
- [Bobby Tables: A guide to preventing SQL injection.](#)
- [Introducción a PHP. Apuntes del módulo DWES de Alfredo Moreno.](#)
- [Arquitectura MVC. Apuntes del módulo DWES de Alfredo Moreno.](#)

## **3 Licencia**

Esta página forma parte del curso Implantación de Aplicaciones Web de José Juan Sánchez Hernández y su contenido se distribuye bajo una licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.