Inserción de código en aplicaciones web

DESARROLLO WEB EN ENTORNO SERVIDOR. UNIDAD 2

2º DESARROLLO DE APLICACIONES WEB. ARCIPRESTE DE HITA. 2025/2026

AARÓN MONTALVO

2. Inserción de código en aplicaciones web. Criterios de evaluación

a) Se han reconocido los mecanismos de generación de páginas web a partir de lenguajes de marcas con código embebido.

b) Se han identificado las principales tecnologías asociadas.

c) Se han utilizado etiquetas para la inclusión de código en el lenguaje de marcas.

d) Se ha reconocido la sintaxis del lenguaje de programación que se ha de utilizar.

e) Se han escrito sentencias simples y se han comprobado sus efectos en el documento resultante.

f) Se han utilizado directivas para modificar el comportamiento predeterminado.

g) Se han utilizado los distintos tipos de variables y operadores disponibles en el lenguaje.

h) Se han identificado los ámbitos de utilización de las variables.

2. Inserción de código en aplicaciones web. Contenidos

2.1. Herramientas y aplicaciones PHP

2.2. Sintaxis básica de PHP

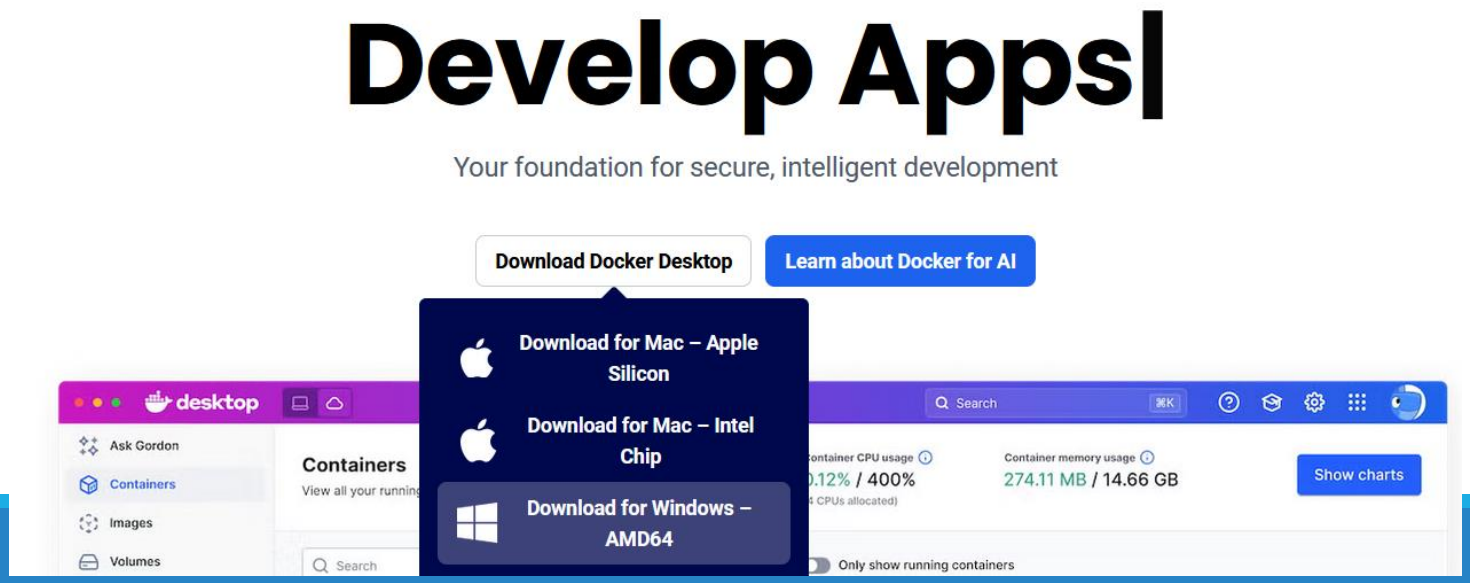
2.3. Tipos de datos en PHP

2.1. Herramientas y aplicaciones PHP. Plataforma

Contenedor Docker: Apache + PHP 8.2 + María DB

Contiene Servidor Web, PHP y Sistema gestor de BD.

1. Descarga e instala Docker Desktop para tu sistema operativo.
   - https://www.docker.com/ -> Download Docker Desktop



2.1. Herramientas y aplicaciones PHP. Plataforma

2. Descarga y descomprime el fichero `docker-dwes.zip`.
3. Con Docker Desktop abierto, abre un terminal, entra en la carpeta descomprimida y ejecuta el comando correspondiente para crear y desplegar los contenedores:

   ```
   docker-compose up -d --build
   ```

4. Prueba que la plataforma funciona accediendo en el navegador a http://localhost:8080

Para visualizar una web desarrollada en entorno servidor hay que ver la página en el navegador web a través del servidor, con HTTP.

2.1. Herramientas y aplicaciones PHP. Plataforma. Directivas de Apache

Las directivas de configuración del servidor web Apache se encuentra en el archivo **`apache-php/apache2.conf`**.

Algunas directivas importantes son:

- **ServerRoot "/etc/apache2"**. Ubicación del directorio raíz de Apache.
- **DocumentRoot " var/www/html"**. Carpeta de los archivos web a servir.
- **Listen 80**. Puerto a utilizar para atender las peticiones.
- **LoadModule php_module /usr/lib/apache2/modules/libphp.so**. Carga del módulo PHP para Apache.
- **DirectoryIndex index.php index.html**. Orden de búsqueda de archivos dentro del directorio raíz cuando no se especifica uno concreto.

2.1. Herramientas y aplicaciones PHP. Plataforma. Directivas de PHP

Las directivas de configuración del módulo PHP XAMPP se encuentra en el archivo **`php.ini`** o en otros dentro de **`php-config`**:

Algunas directivas importantes son:

- **max_execution_time=120**. Tiempo de ejecución máximo en segundos antes de que el analizador termine.
  - Sirve para prevenir que *scripts* mal escritos bloqueen el servidor.
  - El valor por defecto es 30 (desde la línea de comandos es 0).
- **memory_limit=512M**. Máximo de memoria (bytes) que se pueden consumir.
  - Ayuda a prevenir que *scripts* mal escritos consuman la memoria del servidor.
  - El valor por defecto es 128M. Se puede eliminar el límite con el valor -1.

2.1. Herramientas y aplicaciones PHP. Plataforma. Directivas de PHP

- **upload_max_filesize=50M**. Tamaño máximo en bytes de los archivos que pueden ser subidos al servidor.
  - El valor por defecto es 2M.
- **display_errors=On**. Establece los errores del código en el navegador.
  - El valor por defecto es "on", para que se muestren errores.
- **output_buffering=Off**. Guarda datos en un *buffer* interno antes de enviarlos.
  - Se suele utilizar un valor de 4096 en desarrollo.
  - Su valor a implica evitar enviar cabeceras después de haber enviado datos visibles.

2.1. Herramientas y aplicaciones PHP. Plataforma

Para empezar a utilizar los contenedores recuerda siempre

1. **Arrancar Docker Desktop**
2. **Levantar los contenedores desde la terminal**
   - `docker-compose up -d`

2.1. Herramientas y aplicaciones PHP. Editor de código

VSCodium (licencia MIT): https://github.com/VSCodium/vscodium/releases

VisualStudio Code (licencia propietaria): https://code.visualstudio.com/download

Utiliza como worskpace la carpeta `docker-dwes\www`

Además, instala las ***extensiones***

- PHP Intelephense
- PHP Debug

2.1. Herramientas y aplicaciones PHP. Depurador XDebug

Un depurador es un programa que se utiliza para probar una aplicación en desarrollo. Ayuda a detectar y corregir errores

XDebug es una extensión de PHP que proporciona varias características útiles para el desarrollo:

- Depuración paso a paso
- Informes de errores
- *Tracing*

https://xdebug.org/docs/

2.1. Herramientas y aplicaciones PHP. Depurador XDebug. Configuración

El fichero 20-xdebug.ini contiene la configuración necesaria para ejecutar el depurador:

```
zend_extension=xdebug
xdebug.mode=debug
xdebug.start_with_request=yes
xdebug.client_host=host.docker.internal
xdebug.client_port=9003
```

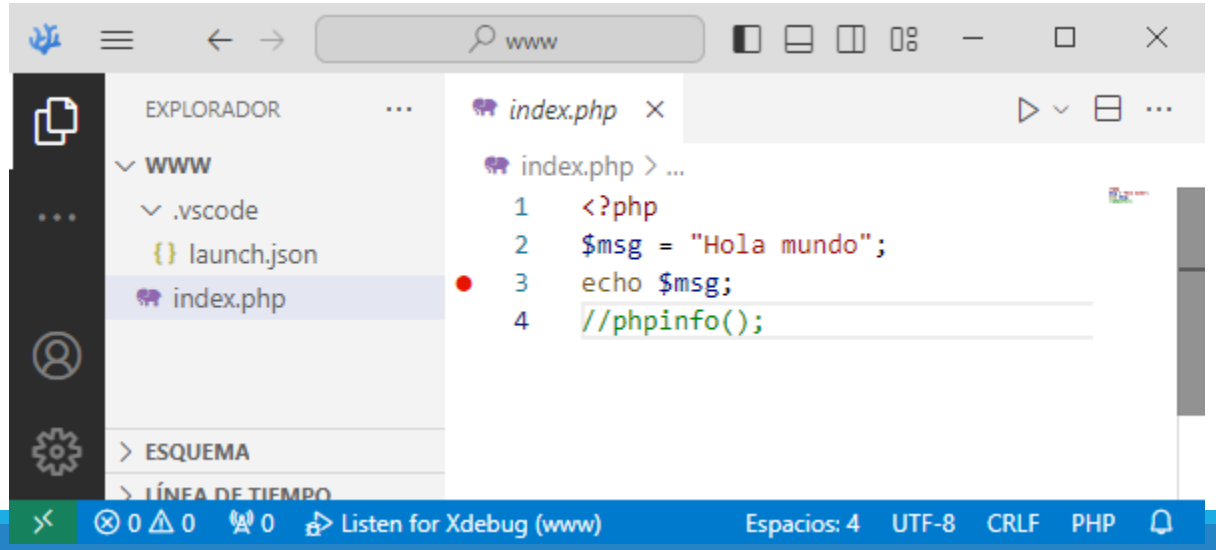
Puedes instalar la extensión **Xdebug helper** en el navegador, aunque la configuración proporcionada no lo hace necesario

https://addons.mozilla.org/es/firefox/addon/xdebug-helper-for-firefox/

https://chromewebstore.google.com/detail/xdebug-helper/eadndfjplgieldjbigjakmdgkmoaaaoc

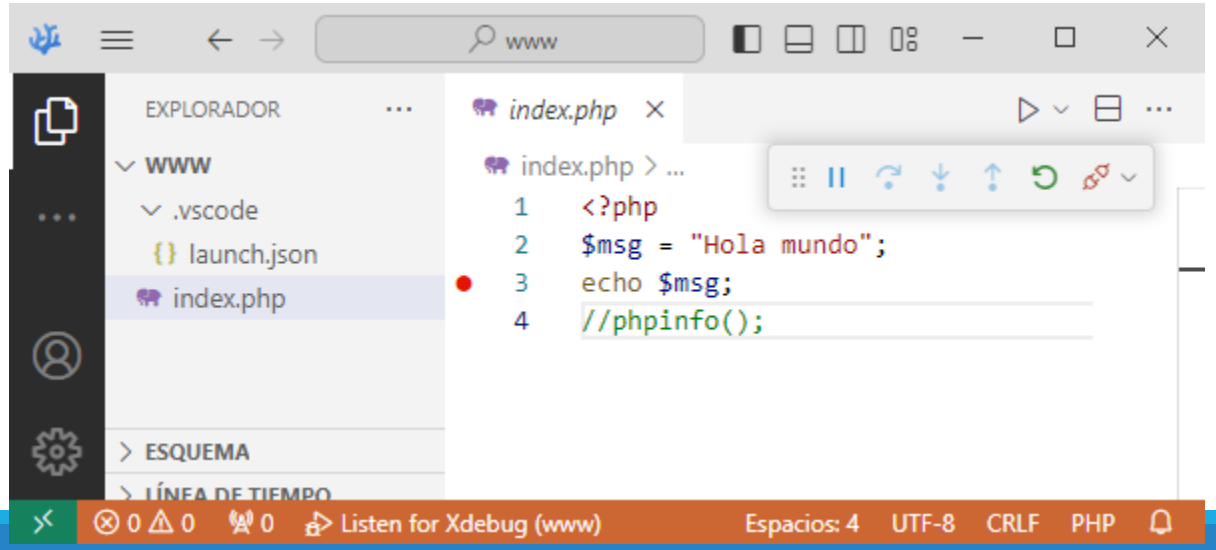
2.1. Herramientas y aplicaciones PHP. Depurador XDebug. Uso (1/3)

1. Abre un archivo PHP en el explorador.
   - Puedes usar `index.php`
2. Pon un punto de depuración (*breakpoint*).
   - Haz *click* sobre la izquierda de una línea ejecutable.



2.1. Herramientas y aplicaciones PHP. Depurador XDebug. Uso (2/3)

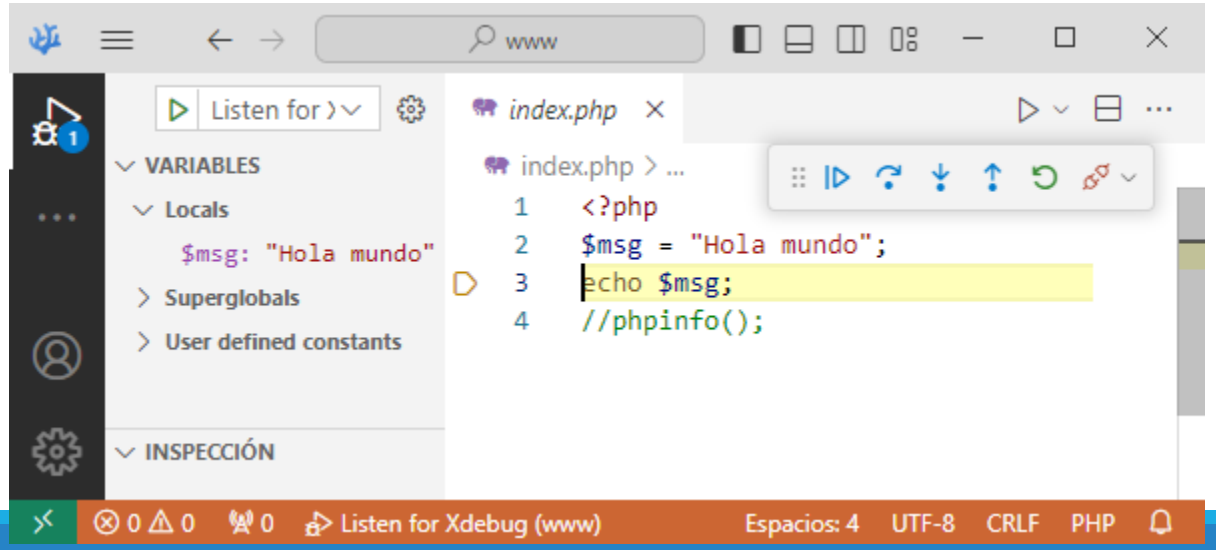
3. Depura el archivo PHP actual .
   - *Ejecutar -> Iniciar depuración*
     - O pulsa la tecla F5
   - El fondo de a barra de estado se volverá naranja.



2.1. Herramientas y aplicaciones PHP. Depurador XDebug. Uso (3/3)

4. En el navegador, pide al servidor la página PHP con el *breakpoint*.
   - **`http://localhost:8080`**`/index.php`.

5. Comprueba que el control de la ejecución lo tiene el IDE.
   - La página no terminará de cargar.
   - La ejecución se parará donde se haya colocado el *breakpoint*.



2.2. Sintaxis básica de PHP

Una aplicación web en PHP está compuesta por archivos HTML, CSS y PHP (y JavaScript para el entorno cliente).

El código PHP puede diseñarse de dos maneras:

- Embebido dentro del código HTML
- Separando el código de apoyo a plantillas

Un archivo PHP tiene extensión .php

2.2. Sintaxis básica de PHP

Todo código PHP debe

- comenzar con la etiqueta <?php
- terminar con ?>

Desde PHP 7.0

- la etiqueta final es optativa si no hay código HTML detrás
- otros mecanismos de inserción están obsoletos

2.2. Sintaxis básica de PHP
Mezclando PHP y HTML

Cualquier texto fuera de las etiquetas <php […]?> se devuelve tal cual.

Un archivo PHP puede tener múltiples etiquetas de apertura y cierre.

```
<?php if ($exp == true): ?>
Texto a mostrar si exp es verdad.
<?php else: ?>
Texto a mostrar si exp es falso.
<?php endif; ?>
```

2.2. Sintaxis básica de PHP. Hola mundo

```php
<?php
    echo "Hola mundo";
?>
```

2.2. Sintaxis básica de PHP. Características

PHP es un lenguaje interpretado.

Todas las sentencias deben terminar en punto y coma **;**

- Igual que en Java o C.

Los espacios en blanco, tabuladores y retornos del carro son ignorados.

Las instrucciones y nombres son sensibles a las mayúsculas y minúsculas.

2.2. Sintaxis básica de PHP. Características

Para mandar texto al cliente sin formato desde el código PHP se pueden utilizar las la construcciones del lenguaje `print()` o `echo`.

```
<?php
if ($exp == true):
echo 'Texto a mostrar si exp es verdad.';
else:
print('Texto a mostrar si exp es falso.');
endif;?>
```

2.2. Sintaxis básica de PHP. Comentarios

Los comentarios ayudan a clarificar el código. En PHP pueden ser

- Hasta final de línea: // o #
- Multilínea: /* …….. */

```
<?php
/* Programa de pedidos web
Pedido generado el 01/09/2021 */
$npedido=10982; // Indica el numero de pedido
$ncliente=4942; # Indica el numero de cliente
?>
```

2.2. Sintaxis básica de PHP. Reutilización de código

`include(archivo)`. Permite incluir un archivo en la posición de la llamada.

- Genera una advertencia (*warning*) si no encuentra el archivo.

`require(archivo)`. Permite incluir un archivo en la posición de la llamada

- Genera un error si no encuentra el archivo.

Los archivos se incluyen según la ruta de acceso dada o, si no se hubiera especificado ninguna, según el `include_path`.

2.2. Sintaxis básica de PHP. Reutilización de código

Existen las sentencias `require_once()` e `include_once()`, que se aseguran que el archivo solo se incluye una vez.

- Si hay funciones definidas en el archivo incluido y el archivo se incluye dos o más veces, PHP arrojará un error fatal, ya que las funciones ya han sido declaradas.
- Estas sentencias son más lentas que las anteriores, pero se recomiendan.

Siempre que el archivo sea llamado por cualquiera de estas instrucciones, PHP no examinará la extensión del archivo.

- El código será procesado como PHP.

2.2. Sintaxis básica de PHP. Reutilización de código

Una alternativa para es utilizar opciones de configuración del archivo php.ini

- auto_prepend_file
- auto_append_file

Se comportan de forma análoga a la instrucción `include()`, evitando tener que poner dicha instrucción en cada archivo, pero obligando a que esos archivos sean incluidos en todas las páginas.

2.2. Sintaxis básica de PHP. Novedades de PHP 8.0

Argumentos nombrados

- Solamente hay que especificar los parámetros obligatorios, pudiendo omitir los opcionales y especificándolos en cualquier orden.
- `htmlspecialchars($string, double_encode: false);`

Promoción de propiedades constructivas

- Se pueden definir e inicializar propiedades de forma compacta.
- ```
  class Point {
    public function __construct(
      public float $x = 0.0) {}
  }
  ```
  ```

2.2. Sintaxis básica de PHP. Novedades de PHP 8.0

Tipos de unión

- Se puede usar una declaración de tipo unión nativa que será validada en el momento de ejecución.
- `private int|float $number;`

Expresiones `match`

- Juntan expresiones `switch` de forma sencilla.
- No hay que romperlas y devuelven un valor.
-
```
echo match (8.0) {
  '8.0' => "Este valor no se imprimirá",
  8.0 => "Este valor sí se imprimirá ",
};
```

2.2. Sintaxis básica de PHP. Novedades de PHP 8.0

Operador *nullsafe*

- Cuando la evaluación de un elemento falla, la ejecución de la entire cadena es abortada y la cadena entera es evaluada como nula.
- `$country = $session?->user?->getAddress()?->country;`

Comparaciones inteligentes entre cadenas y números

- Si la cadena es numérica se convierte a número y se compara así. Si no, convierte el número a una cadena y compara las cadenas.
- `0 == 'foobar' // false`

Errores consistentes para funciones internas

- Las funciones internas dan error de excepción si el parámetro no es validado.

2.2. Sintaxis básica de PHP. Novedades de PHP 8.1

Enumeraciones

- Se pueden definir tipos enumerados directamente.
- ```
  enum Status {
    case Bien; case Mal;
  }
  ```

Propiedades de solo lectura

- Nuevo modificador `readonly` para atributos cuyo valor no puede cambiar.
- `public readonly Status $status;`

Sintaxis First-class Callable

- Se puede obtener una referencia a cualquier función.
  ```

2.2. Sintaxis básica de PHP. Novedades de PHP 8.1

Expresión *new* en constructor

- Los objetos instanciados pueden ser utilizados como parámetros por defecto, variables estáticas, constantes globales y argumentos de atributos.
- ```
  public function __construct(
        Logger $logger = new NullLogger(),) {
        $this->logger = $logger;
  }
  ```

Tipos de intersección pura

- Para un valor que necesite resolver varias restricciones de tipado a la vez.
- ```
  function count_and_iterate(Iterator&Countable $value) {
    /*...*/}
  ```

2.2. Sintaxis básica de PHP. Novedades de PHP 8.1

Tipo de retorno *never*

- Indica que no devolverá valor: producirá una excepción o finalizará con una llamada a `die()`, `exit()` o similar.
-
```
function redirect(string $uri): never {
  header('Location: ' . $uri);
  exit();
}
```

Constantes de clase final

- Para que la clase no pueda ser sobrescrita en las clases heredadas.
- `class Foo { final public const XX = "foo"; }`

2.2. Sintaxis básica de PHP. Novedades de PHP 8.1

Notación numérica octal explícita

- Es posible escribir números octales
- `0o16 === 16; // false`
- `0o16 === 14; // true`

Fibers

- Implementación ligera de concurrencia.
- Bloques de código que puedan pausarse y reanudarse.

Soporte de expansión de matrices

- También para matrices con claves de tipo cadena de caracteres.
- `$result = ['a' => 0, ...$arrayA, ...$arrayB];`

2.2. Sintaxis básica de PHP. Novedades de PHP 8.2

Clases de solo lectura

- Nuevo modificador `readonly` para clases.
- `readonly class BlogData { /*...*/ }`

Tipos de forma normal disyuntiva (DNF)

- Permiten combinar unión y intersección de tipos
- Deben agruparse con paréntesis
- `public function bar((A&B)|null $entity) { /*...*/ }`

Tipos independientes *`null`*, *`false`* y *`true`*

- `public function alwaysFalse(): false { /* ... */ *}`

2.2. Sintaxis básica de PHP. Novedades de PHP 8.2

Propiedades dinámicas en desuso

- `class User { public $name; }`
- `$user = new User();`
- `$user->last_name = 'Doe'; // Deprecated`
- `$user = new stdClass();`
- `$user->last_name = 'Doe'; // Still allowed`

Nueva extensión *random*

- Nueva API orientada a objetos para la generación de números aleatorios.

Constantes en rasgos

- Se puede acceder a la constante solo a través de la clase que utiliza el rasgo.

2.3. Tipos de datos en PHP. Tipos de datos: Escalares

`bool / boolean`

- Booleano. Solo admite los valores `true` y `false`.

`int / integer`

- Número entero

`float / double`

- Número de coma flotante. Cuidado con las comparaciones (épsilon).
- Anteriormente también conocido como `real`.

`string`

- Cadena de caracteres.

2.3. Tipos de datos en PHP. Tipos de datos: Compuestos

`array`

- Matriz. Mapa ordenado. Asocia *valores* con *claves.*

`object`

- Instancia de una clase

`callable`

- Llamadas de retorno.

`Iterable`

- Pseudotipo para usar con `foreach`.
- Introducido en PHP 7.1.

2.3. Tipos de datos en PHP. Tipos de datos: Especiales

`resource`

- Contiene una referencia a un recurso externo.

`null`

- Variables sin valor asignado, asignados a `null` o destruidas con `unset()`.

`mixed`

- Acepta cualquier valor. Desde PHP 8.0.
- `object|resource|array|string|float|int|bool|null`

`void`

- Tipo de retorno únicamente. Indica que la función termina sin retornar valor.

2.3. Tipos de datos en PHP.
Tipos de datos

El tipo de una variable no lo declara el programador

- Lo decide PHP en tiempo de ejecución dependiendo del contexto en el que se emplea la variable.

PHP es un lenguaje de control de tipos muy débil.

- **Las variables cambian de tipo a medida que se les asignan valores diferentes.**

Se puede obtener el tipo y el valor de una expresión con la función `var_dump(mixed $value, mixed ...$values): void`.

Se puede obtener el tipo de una variable con la función `gettype(mixed $value): string`.

2.3. Tipos de datos en PHP. Control de tipos

```php
<?php
$res = 0.00;
$tipo = gettype($res);
echo "<p>Valor: $res, tipo $tipo</p>";

$res = "Cero";
$tipo = gettype($res);
echo "<p>Valor: $res, tipo $tipo</p>";
?>
```

2.3. Tipos de datos en PHP. Control de tipos

Para comprobar los tipos se puede determinar el tipo de una variable con otras funciones:

- `is_array(mixed $value): bool`
- `is_bool(mixed $value): bool`
- `is_float(mixed $value): bool`
- `is_int(mixed $value): bool`
- `is_object(mixed $value): bool`
- `is_string(mixed $value): bool`

2.3. Tipos de datos en PHP. Control de tipos

Comprobar si existe una variable:

- `isset(mixed $var, mixed ...$vars): bool`

Destruir una variable:

- `unset(mixed $var, mixed ...$vars): void`

Si existe una variable y está vacía:

- `empty(mixed $var): bool`

2.3. Tipos de datos en PHP. Forzado de tipos

Se puede tratar una variable como si fuera de un tipo diferente.

Se escribe el tipo deseado entre paréntesis delante de la variable a forzar o convertir.

```php
$cantidad = 10;
$resultado = (double)$cantidad;
var_dump($cantidad);
var_dump($resultado);
```

La variable en coma flotante toma el valor entero en coma flotante.

La variable entera no se ve modificada.

2.3. Tipos de datos en PHP. Forzado de tipos

Solo están permitidas las siguientes conversiones explícitas:

```
(int) // (integer)
(bool) // (boolean)
(float) // (double)
(real) // alias de float, en desuso desde PHP 8.0
(string) // (binary)
(array)
(object)
```

2.3. Tipos de datos en PHP. Forzado de tipos

La función `settype()` también sirve para establecer el tipo de una variable:

```
settype(mixed &$var, string $type): bool
```

Establece el tipo de la variable `var` a `type`, que puede ser cualquiera de los tipos aceptados en una conversión explícita.

2.3. Tipos de datos en PHP. Reinterpretación de variables

Toman una variable y la devuelven convertida al tipo apropiado.

```
intval(mixed $value, int $base = 10): int
```

```
floatval(mixed $value): float
```

- `doubleval(mixed $value): float //alias`

```
strval(mixed $value): string
```

2.3. Tipos de datos en PHP. Constantes

Permiten almacenar valores escalares que no se podrán modificar en ningún punto del programa.

Se pueden definir constantes de matrices (desde PHP 7), siempre que contengan solo expresiones escalares.

La función define permite definir una constante:

```
define(string $constant_name, mixed $value, bool $case_insensitive =
false): bool
```

2.3. Tipos de datos en PHP. Constantes

También se puede usar la palabra clave `const`.

- `const PI=3.14;`

La diferencia principal es que `const` define las constantes en tiempo de compilación, y `define()` en tiempo de ejecución.

- Además, `const` no permite expresiones para definir la constante.

2.3. Tipos de datos en PHP. Constantes

Por convenio, los nombres de las constantes se escriben en MAYÚSCULAS para diferenciarlas de las variables.

Para referenciar a una constante se utiliza su nombre sin el signo $.

```
define ('PRECIO', 100);
echo PRECIO;
```

2.3. Tipos de datos en PHP. Variables

Una variable es un espacio en la memoria que almacena un valor en la memoria del ordenador y se puede referenciar con un nombre.

Para crear una variable en PHP hay que escribir el nombre de la variable precedido de $ y asignarle un valor.

- PHP no requiere declarar las variables antes de utilizarlas.

```php
$nombre = "Mundo";
echo "Hola, $nombre";
```

2.3. Tipos de datos en PHP. Variables. Características de los nombres

Descriptivos. Deben identificar la información que contienen.

Longitud apropiada. Suficientemente largos para ser descriptivos, pero no tanto como para que resulte tedioso escribirlos.

Sin símbolos. Excepto guion bajo (_).

Sin espacios: Se usan guiones bajos (_), o *camelCase* coherentes.

No pueden empezar por un número.

Un nombre de variable puede ser igual al de una función, pero no se recomienda.

Son sensibles a mayúsculas y minúsculas: `user ≠ User ≠ USER`

2.3. Tipos de datos en PHP. Variables. Asignación

Las variables se asignan por defecto por valor.

Se puede asignar una variable por referencia con el operador &.

Si no se inicializa una variable tendrá un valor por defecto de acuerdo al contexto en el que son usadas.

- las booleanas se serán `false`.
- los números enteros y flotantes serán cero,
- las cadenas se establecen como una cadena vacía
- las matrices se convierten en una matriz vacía

2.3. Tipos de datos en PHP. Variables. Asignación

```php
$foo = 'Bob';          // Asigna el valor 'Bob' a $foo
$bar = &$foo;          // Referencia $foo usando $bar
$bar = "Yo soy $bar";// Modifica $bar y también $foo
echo $bar;
echo $foo;
```

2.3. Tipos de datos en PHP. Variables variables

Son nombres de variables que se pueden definir y usar de forma dinámica.

Toman el valor de una variable y lo tratan como el nombre de una variable.

```
$a = 'hola';
$$a = 'mundo';
echo "<p>$a {$$a}</p>";
echo "<p>$a $hola</p>";
```

2.3. Tipos de datos en PHP. Variables. Ámbito

Ámbito: Lugares dentro del programa donde una variable es visible.

Variables superglobales. Visibles desde cualquier lugar del programa.

- `$GLOBALS`. Matriz con referencias a las variables globales.
- `$_SERVER`. Matriz con información del servidor y del entorno de ejecución.
- `$_GET`. Matriz con las variables pasadas por parámetros URL (*query string*).
- `$_POST`. Matriz con las variables pasadas con el método POST de HTTP.
- `$_FILES`. Matriz con los elementos subidos con el método POST.
- `$_COOKIE`. Matriz con las *cookies* actuales.
- `$_SESSION`. Matriz con las variables de sesión actuales.
- `$_REQUEST`. Matriz con el contenido de `$_GET`, `$_POST` y `$_COOKIE`.
- `$_ENV`. Matriz con las variables de entorno.

2.3. Tipos de datos en PHP. Variables. Ámbito

Variables globales. Visibles en el bloque de código en el que son declaradas, pero no dentro de las funciones.

Variables de función. Visibles solo dentro de la función donde han sido declaradas.

Variables de función globales. Hacen referencia dentro de una función a la variable global del mismo nombre.

- Para acceder dentro de una función a un variable externa con dicho nombre se puede usar la palabra clave `global` o el array `$GLOBALS`.

2.3. Tipos de datos en PHP. Variables. Ámbito

```php
$a = 1; // variable global
function test(){
    echo "<p>$a</p>"; // variable global, no definida
    $b = 2;
    global $a;
    echo "<p>$a</p>"; // acceso a la variable global
    echo "<p>$b</p>"; // acceso la variable local
}
test();
```

2.3. Tipos de datos en PHP. Operadores

Los operadores son símbolos que representan una acción concreta que se realizará sobre variables o valores.

Aritméticos. Realizan las operaciones aritméticas más comunes.

- suma (+). Si es unario, es identidad, o conversión a `int`/`float`.
- resta (-). Si es unario, es la negación, o número opuesto.
- multiplicación (*)
- división (/)
- módulo (%)
- exponenciación (**)

2.3. Tipos de datos en PHP. Operadores

De asignación (=). Asigna el valor, resultado de la expresión o contenido de la variable de la derecha del operador a la variable situada a la izquierda del mismo.

```
$cantidad = 5;

$precio = 12.5 * 1.20;

$total = $precio * $cantidad;
```

Combinación de operaciones aritméticas con asignación.

```
$precio += 5;   // $precio = $precio + 5;

$cantidad *= 2; // $cantidad = $cantidad * 2;
```

2.3. Tipos de datos en PHP. Operadores

Incremento y decremento (++ y --). Permiten incrementar o decrementar en una unidad (1) el valor de una variable.

- Si el operador precede a la variable primero modifica el valor de la variable y después lo utiliza (preincremento y predecremento)
- Si el operador va detrás primero utiliza el valor de la variable y luego la modifica (postincremento y postdecremento).

```
$a = 4;

$b = $a++; // $b valdrá 4 y $a valdrá 5

$c = ++$a; // $c y $a valdrán 6
```

2.3. Tipos de datos en PHP. Operadores

Comparación. Permiten evaluar si se cumple una condición entre dos valores. Devuelve un valor booleano (`true` o `false`).

- igual (==)
- idéntico (===)
- diferente (!= o <>)
- no idéntico (!==)
- menor que (<)
- mayor que (>)
- menor o igual que (<=)
- mayor o igual que (>=)
- nave espacial (<=>). Devuelve `-1`, `0` o `1`. Solo para enteros

2.3. Tipos de datos en PHP. Operadores

De matrices. Realizan las operaciones sobre matrices más comunes.

- unión (+)
- igualdad (==)
- identidad (===)
- desigualdad (!= o <>)
- no identidad (!==)

2.3. Tipos de datos en PHP. Operadores

Lógicos. Permiten combinar los resultados de la condiciones y comparaciones.

- NOT (!)
- AND (&& , and)
- OR (|| , or)
- XOR, OR exclusivo (xor)

Ambas versiones de los operadores AND y OR tienen precedencias diferentes.

2.3. Tipos de datos en PHP. Operadores

Bit a bit. Permiten tratar a un dato como una secuencia de bits.

- NOT (~). Operador unario.
- AND (&)
- OR (|)
- XOR, OR exclusivo (^)
- Desplazamiento a la izquierda (<<)
- Desplazamiento a la derecha (>>)

2.3. Tipos de datos en PHP. Operadores

Concatenación (.). Genera una cadena como unión de otras.

- Se puede mezclar asignación sobre concatenación (.=)

```
$texto = "DAW2 - ";
echo $texto."Desarrollo Web";
```

Referencia (&). Permite asociar el valor de una variable a otra

- Si se modifica el valor de la segunda variable, la primera también se modifica.
- Hay que pensar en ellos como alias, no como punteros.

```
$a = 5;
$b = &$a;
$a = 7;
```

2.3. Tipos de datos en PHP. Operadores

Ternario (?:). Similar al de la estructura `if-else`.

- `condicion ? valor_si_verdad : valor_si_falso`

```php
$nota >= 5 ? 'Aprobado' : 'Suspenso';
```

Fusión de nulo (??). Devuelve de izquierda a derecha el primer operador que exista.

- Es la combinación del operador ternario y la función `isset()`.

```php
$user = $_GET['user'] ?? 'yo';
```

```php
$user = isset($_GET['user']) ? $_GET['user'] : 'yo';
```

2.3. Tipos de datos en PHP. Operadores

Ejecución (``). Permite ejecutar comandos del sistema operativo

- Son 2 acentos invertidos o graves, uno antes y otro después del comando.

```
$out = `dir c:`; // Windows
echo "<pre>$out</pre>";
```

2.3. Tipos de datos en PHP. Operadores.

Control de errores (@). Permite eliminar los avisos de advertencias (*warnings*) que se pueden generar al ejecutar una instrucción.

- No puede anteponerse a definiciones de funciones ni clases, ni a estructuras condicionales
- Desde PHP no suprime errores fatales, como división por cero

De tipo (`instanceof`). Determina si una variable es un objeto instanciado de una clase dada.

- `$varObj instanceof className`

2.3. Tipos de datos en PHP. Operadores

Supresión de error: permite eliminar los avisos de error que se pueden generar al ejecutar una instrucción.

```
$a = @(5/0); //No se mostrará el error de división por cero
```

instanceof: determinar si un objeto pertenece a una clase

```
$varObj instanceof className
```

2.3. Tipos de datos en PHP. Operadores. Precedencia

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

Actividad 2.1 (1/2)

Analiza el siguiente código y describe cual sería la salida por pantalla. Compruébalo ejecutando el código en el servidor.

a)

```php
$resultado = 0.00;
$tipo = gettype($resultado);
echo "Resultado vale: $resultado y es de tipo $tipo\n";
$resultado = "Cero";
$tipo = gettype($resultado);
echo "y ahora vale: $resultado y es de tipo $tipo\n";
```

Actividad 2.1 (2/2)

b)

```php
$resultado = 0;
$tipo = gettype($resultado);
echo "Resultado vale: $resultado y es de tipo $tipo\n";
$resultado2 = (double)$resultado;
$tipo = gettype($resultado2);
echo "y Resultado2: $resultado2 y es de tipo $tipo\n";
$tipo = gettype($resultado);
echo "y Resultado vale: $resultado y es de tipo $tipo";
```

Actividad 2.2

a. Crea una variable de tipo entero con valor 5 y muéstrala en pantalla indicando su tipo. A continuación, asigna a la variable el valor 5.5 y vuelve a mostrarla en pantalla indicando su tipo, ¿se ha modificado?

b. Crea una variable de tipo doble con valor 5.5 y muéstrala en pantalla indicando su tipo. A continuación, conviértela a tipo entero y vuelve a mostrarla en pantalla indicando su tipo, ¿se ha modificado?

c. Crea las siguientes variables: nombre, apellidos, dirección, código postal, localidad y provincia. Asigna valores a las variables y muéstralos en una tabla indicando su tipo:

Variable	Valor	Tipo
nombre	Juan	string
apellidos	Perro	string
direccion	C. Jardín Botánico	string
codigoPostal	28014	integer
localidad	Madrid	string
Provincia	Madrid	string

Actividad 2.3

a. Define PI como constante con 4 decimales y muestra su valor.
   A continuación, utiliza la constante para calcular la longitud de una circunferencia y el área de un círculo cuyos radios vengan dados por la variable "radio".

b. Escribe un programa que permita obtener el siguiente resultado utilizando el operador necesario en cada caso:
   a. valor1: 8
   b. valor2: 3
   c. suma: 11
   d. resta: 5
   e. producto: 24
   f. cociente: 2,66666667
   g. resto de la division: 2
   h. incremento de valor 1: 9
   i. decremento de valor2: 2

Actividad 2.4

a. Escribe el valor que tendrán las variables tras ejecutar el siguiente código:

`$num1 = 7;`
`$num2 = 5;`
`$result = $num1;`
`$result += $num2++;`
	`$num1 = 7;`
	`$num2 = 5;`
	`$result = &$num1;`
	`$result += ++$num2;`
		`$num1 = 7;`
		`$num2 = 5;`
		`$result = &$num1;`
		`$result += ++$num1;`

`$num1 =`
`$num2 =`
`$result =`
	`$num1 =`
	`$num2 =`
	`$result =`
		`$num1 =`
		`$num2 =`
		`$result =`

b. Escribe el resultado que devolverán las siguientes comparaciones:

`$num1 = 0;`
`$num2 = 0.0;`
`var_dump ($num1==$num2);`
	`$num1 = 0;`
	`$num2 = 0.0;`
	`var_dump ($num1===$num2);`

`Salida:`	`Salida:`

Actividad 2.5

Escribe los resultados que se mostrarán en pantalla al ejecutar el siguiente código:

```
$num1 = 5.5; $num2 = &$num1;
echo "num1: $num1 - num2: $num2\n";
echo "Existe la variable num1: ";
var_dump(isset($num1));
echo "Esta vacía la variable num1: ";
var_dump(empty($num1));
echo "Es de tipo entero la variable num1: ";
var_dump(is_int($num1));
echo "Es de tipo float la variable num1: ";
var_dump(is_float($num1));
unset ($num1);
echo "Existe la variable num1: "; var_dump(isset($num1));
echo "Existe la variable num2: "; var_dump(isset($num2));
echo "num1: $num1 - num2: $num2";
```

Actividad 2.6

Indica el tipo y el valor de las siguientes variables tras ejecutar el código:

`$num1 = “7”;`
`$num2 = 5;`
`$result = intval($num1) / $num2;`
	`$num1 = "Lote: ";`
	`$num2 = 724;`
	`$result = $num1.$num2;`

`$num1 =`
`$num2 =`
`$result =`
	`$num1 =`
	`$num2 =`
	`$result =`

Actividad 2.7

Prueba a realizar conversiones de tipos en diferentes variables y elabora una tabla en la que describas que conversiones son posibles y en qué casos:

Tipo origen	Tipo destino	¿Posible?	¿En qué casos?
Entero	Coma flotante	Sí	Siempre
Entero	Cadena		
Coma flotante	Entero		
Coma flotante	Cadena		
Cadena	Entero		
Cadena	Coma flotante		