

# **Módulo PHP**

## **1. Introducción a PHP**

### **¿Qué es PHP?**

¡PHP es un lenguaje increíble y popular!

¡Es lo suficientemente potente como para estar en el núcleo del sistema de blogs más grande de la web (WordPress)!

¡Es lo suficientemente profundo para ejecutar la red social más grande (Facebook)!

¡También es bastante fácil de utilizar, al ser el primer idioma del “lado del servidor” y por eso puede ser utilizado por principiantes! Lenguajes que se instalaban en el servidor, el procesamiento se hacía allí, no hacía falta tener una máquina muy potente.

### **¿Qué es un archivo PHP?**

Los archivos PHP pueden contener textos, HTML, CSS, JavaScript, y código PHP, a su vez.

El código PHP se ejecuta en el servidor y el resultado se devuelve al navegador como HTML sin formato.

Los archivos PHP tienen la extensión “.php”.

### **¿Qué puede hacer PHP?**

Es bastante similar a C, condicionales, bucles, procede de ese lenguaje C, orientado a objetos, cierta similitud. Puede ejecutar muchos elementos.

- PHP puede generar contenido de página dinámico
- PHP puede crear, abrir, leer, escribir, eliminar y cerrar archivos en el servidor
- PHP puede recopilar datos de formularios
- PHP puede enviar y recibir cookies (es bastante importante, cada vez hace más hincapié en que cookies estamos almacenando).
- PHP puede agregar, eliminar, modificar datos en su base de datos
- PHP se puede utilizar para controlar el acceso de los usuarios
- PHP puede cifrar datos (se pueden cifrar contenidos para que sean cifradas, etc.).

\*Con PHP no estas limitado a generar HTML. Puede generar imágenes, archivos PDF e incluso películas Flash. También puede generar cualquier texto, como XHTML y XML.

*Cookie no es lo mismo que sesión. A veces la sesión se guarda en una cookie. Borraremos todas las cookies menos las necesarias. Otra persona ha puesto su nombre. Dos filas, dos registros.*

### **¿Por qué utilizar PHP?**

Porque tengo que utilizar PHP, básicamente porque se puede utilizar en bastantes sistemas operativos.

- PHP se ejecuta en varias plataformas (Windows, Linux (apache), Unix, Mac OS X, etc.)

- PHP es compatible con casi todos los servidores que se utilizan en la actualidad (Apache, IIS, etc.)
- PHP admite una amplia gama de bases de datos (php my admin) (My SQL)
  - En el caso de Windows – Microsoft se trata de Acces
- PHP es gratis. Descárguelo del recurso oficial de PHP: php.net
- PHP es fácil de aprender y se ejecuta de manera eficiente en el lado del servidor.

### Novedades de PHP 8

PHP 8.0 es una actualización importante del lenguaje PHP. Contiene muchas características nuevas y optimizaciones que incluyen argumentos con nombre, tipos de unión, atributos, promoción de propiedades del constructor, expresión de coincidencia, operador nullsafe, JIT y mejoras en el sistema de tipos, manejo de errores y consistencia.

- La vulnerabilidad y la seguridad, depuración de errores, es uno de los motivos más importantes de seguridad. Evolucionar en ese idioma.
- Tiempos de ejecución también es importante. Cada vez se intenta hacer más eficiente el lenguaje. → Si el lenguaje es más eficiente, se cargará mejor.

**GitHub:** la web por excelencia de compartición de códigos.

### Sintaxis PHP

Se ejecuta un script PHP en el servidor y el resultado HTML sin formato se envía de vuelta al navegador.

➔ Sintaxis básica:

- Se puede colocar un script PHP en cualquier lugar del documento
- Un script PHP comienza con `<?php` y termina con `?>`
  - El comentario va con `//` barras

\*La extensión de archivo predeterminada para los archivos PHP es `“.php”`.

\*Un archivo PHP normalmente contiene etiquetas HTML y algo de código de programación PHP.

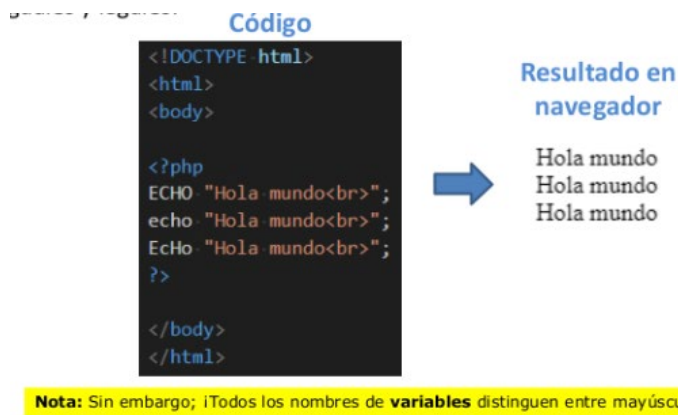
Diapositiva nº 12: normalmente se pone una línea por sentencia.

### Sensibilidad de mayúsculas y minúsculas

En PHP, las palabras clave (por ejemplo, `if`, `else`, `while`, `echo`, etc.), clases, funciones, y funciones definidas por el usuario no distinguen entre mayúsculas y minúsculas.

“PHP Case Sensitivity”

En el siguiente ejemplo, las tres declaraciones de `“echo”` a continuación iguales y legales.



## 2. XAMPP

### ¿Qué es XAMPP?

Tendremos una carpeta diferenciada que será nuestro XAMPP. XAMPP es el entorno más popular de desarrollo con PHP.

XAMPP es una distribución de Apache completamente gratuita y fácil de instalar que contiene MariaDB, PHP y Perl.

El paquete de instalación de XAMPP ha sido diseñado para ser increíblemente fácil de instalar y usar.

*Apachefriends.org:*

Cuando hayamos abierto estos, los archivos que quedamos abrir interpretados por el servidor hay que abrirlo con localhost/

- ➔ Habrá que ir al navegador como el Chrome y poner,
  - Localhost
  - 127.0.0.1

La carpeta htdocs: es la conexión con local host

## 3. Editores de Código PHP

Ayudan a escribir y editar el código PHP, dar sentido a la estructura de tu archivo y proyecto, y cometer menos errores:

- Saltos de línea,
- Indentación
- Autocompletar
- Diferentes vistas y modos de edición
- Finalización de código
- Búsqueda y comparación de funcionalidades

- Sugerencias inteligentes de código

Y ediciones avanzadas (mejorar el código, errores y mantenimiento) :

- Depuración de código
- Control de versiones (por ejemplo, Github)
- Refactorización

#### 1. [Visual Studio Code](#)

2. [Atom](#)
3. [Brackets](#)
4. [GNU Emacs](#)
5. [Vim](#)
6. [Bluefish](#)
7. [jEdit](#)
8. [Notepad++](#)
9. [RJ TextEd](#)
10. [TextMate](#)

Editores de código PHP



## Editores de código PHP

Editores de código PHP (Premium)

#### 1. [Sublime Text 3](#)

2. [UltraEdit](#)
3. [Rapid PHP Editor](#)
4. [Smultron](#)

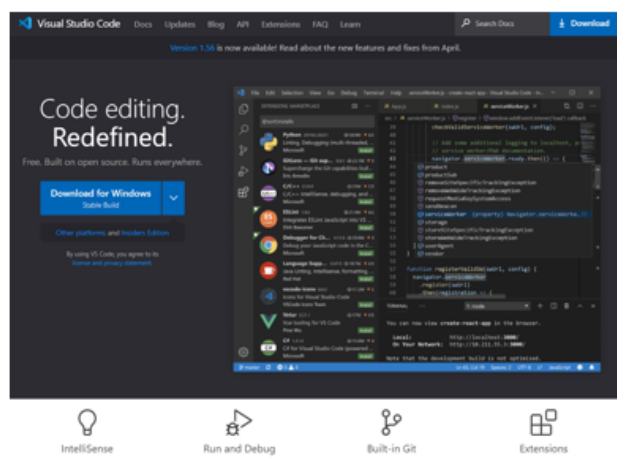
```
<html>
<head>
<title> PHP Test </title>
</head>
<body>
<?php echo '<p>Hello World</p>'; ?>
</body>
</html>
```

## Visual Studio Code

- Sistemas operativos compatibles: Windows, Linux, MacOS
- Licencia: Licencia del MIT
- Código

fuelle: <https://github.com/microsoft/vscode>

- Lenguajes soportados: PHP, HTML, CSS, SCSS, Less, JavaScript, JSON, TypeScript, Markdown, PowerShell, C++, Java, Python, Go, T-SQL, C#, .NET Core, y muchos más



## Las mejores características:

- resaltado de sintaxis, autocompletado y navegación de código
- la finalización del código inteligente con [IntelliSense](#)
- selector de temas de color

- integración de **Git** y **GitHub** incorporada
- soporte de **Emmet** incorporado
- extensibilidad (se puede crear tu propia extensión)
- depuración, refactorización
- **terminal** integrado
- **compartir** en vivo para la programación en pareja (**con extensión**)

## Atom

- Sistemas operativos compatibles: Windows, Linux, macOS, FreeBSD
- Licencia: Licencia del MIT
- Código  
fuente: <https://github.com/atom/atom>
- Lenguajes soportados: PHP, HTML, CSS, JavaScript, Java, C, C#, Objective C, Perl, Python, Ruby, Go, XML, y muchos más



*“Editor de texto hackeable del siglo XXI”*

### **Las mejores características:**

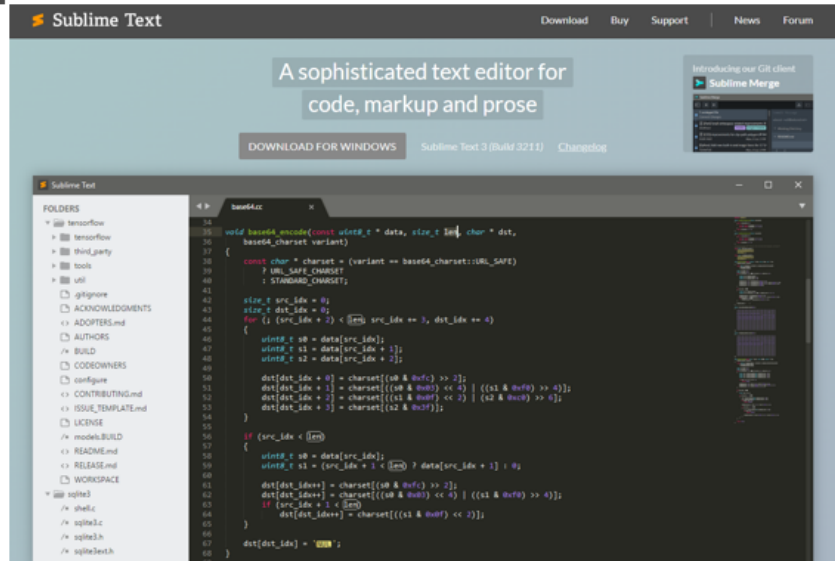
- UI personalizable con más de [3.000 temas Atom de código abierto](#)
- resaltado de sintaxis y autocompletado inteligente
- la navegación por el sistema de archivos y los múltiples paneles
- herramienta avanzada de búsqueda y reemplazo
- gestor de paquetes integrados (llamado [apm](#))
- Paleta de comandos de fácil acceso
- extensibilidad
- colaboración de código en tiempo real (con el [paquete interno Teletype](#))
- integración directa de Git y GitHub (con el [paquete GitHub](#) de construcción propia)

## Sublime Text 3



## Sublime Text 3

- Sistemas operativos compatibles: Windows, Linux, MacOS
- Precio: 80 dólares con 3 años de actualizaciones; la descarga es gratuita
- Licencia: [ver EULA](#)
- Lenguajes soportados: HTML, CSS, Sass, LESS, Markdown, JavaScript, PHP, Python, Perl, Ruby, Java, y muchos más



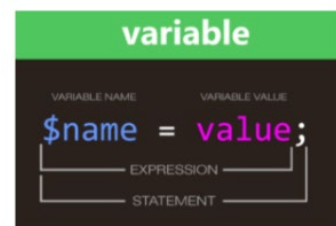
### Las mejores características:

- recortes personalizables, menús, macros, enlaces de teclas (con archivos JSON)
- paleta de comandos
- definiciones *goto* con un índice de funciones, clases y métodos para todo el proyecto
- funcionalidad de edición avanzada como vista dividida, mapa de documentos, selecciones múltiples, autocompletado y otras
- el administrador de paquetes integrado
- extensible a través de plugins

## 4. Variables PHP

Las **variables** son contenedores que nos permiten guardar información temporalmente para después poder trabajar con esa información y hacer lo que necesitemos, como por ejemplo hacer cálculos, traer esa información y mostrarla en pantalla, etc.

Las variables son un tema importante en cualquier lenguaje de programación y también PHP.



### // NOTAS:

## Creación (declaración) de variables PHP

En PHP, una variable comienza con el signo \$, seguido del nombre de la variable:

Después de la ejecución de las declaraciones anteriores, la variable \$x tendrá el valor Hola Mundo, la variable \$y, tendrá el valor 5 y la variable \$z tendrá el valor 10.5

- Nota: Cuando asigne un valor de texto a una variable, coloque comillas alrededor del valor.

A diferencia de otros lenguajes de programación, PHP no tiene ningún comando para declarar una variable. Se crea en el momento en que le asigna un valor por primera vez.

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = "Hola mundo";
$y = 5;
$z = 10.5;

echo $x;
echo "<br>";
echo $y;
echo "<br>";
echo $z;
?>

</body>
</html>
```

Si quiero abrir ese archivo con las variables PHP, va a escribir básicamente algo que ha interpretado. Lo que nosotros estamos creando desde aquí es el servidor. Es el servidor el que se encarga de enviárselo al navegador. El será el que procese y ejecute.

- El código está ejecutándose por un lado y la carga por otro, en el caso de AJAX. Si cambia la carga se volverá a revisar.

Las direcciones del dominio, son los puertos que salen. 80 → local host 443 → 127.0.0.1.

Si lo abro por el navegador el archivo no lo interpreta. Hay que abrirlo desde el servidor para que sea interpretado correctamente.

➔ Función gettype (\$nombre)

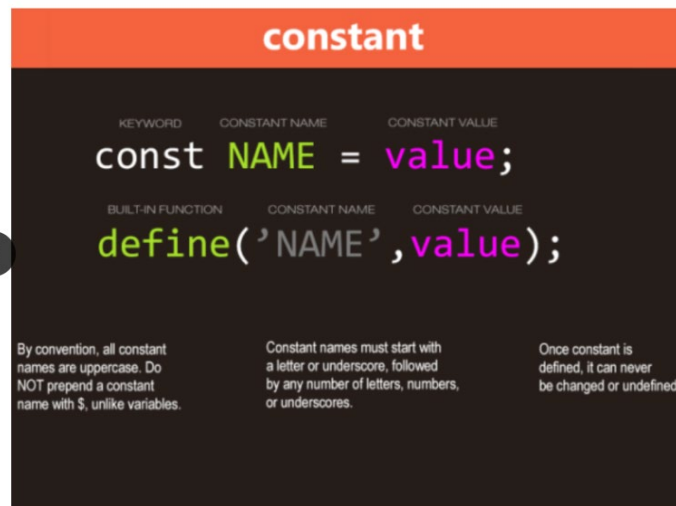
En el caso de la variable boolean si es un resultado afirmativo será un 1 y si es falso

## 5. Constante

Las constantes son contenedores similares a las variables que nos permiten guardar información para después poder trabajar con esa información y hacer lo que necesitemos, como por ejemplo hacer cálculos, traer esa información y mostrarla en pantalla, etc.

- ➔ En este caso, las constantes, cuando son definidas, no pueden cambiar de valor.

```
<?php
const ... ;
define(...);
?>
```



## Definición de constantes PHP

Cuando las llamamos, nos acordamos de que esa variable es constante. Si son en mayúsculas, serían constantes, si son en minúsculas podrían ser variable. Sabes determinar que son una o la otra, constante **en mayúscula**.

### ➔ 1. Función define()

- Primero le dices el nombre, y luego le dices el valor.

```
<?php
```

```
define("CONSTANTE", "Hola mundo.");
```

```
echo CONSTANTE; // muestra "Hola mundo."
```

```
?>
```

\*Nota: En la función define() podía haber un tercer parámetro, true); para definir que la CONSTANTE no fuera case-sensitive y podía llamarse en minúsculas o mayúsculas indistintamente, pero desde la versión PHP 7.3 y PHP8 ya no funciona. **Keysensitive: diferenciar mayúscula y minúscula.**

### ➔ 2. Palabra Clave **Const** (primero pones la palabra, luego el nombre = el valor)

```
<?php
```

```
// Funciona a partir de PHP 5.3.0
```

```
const CONSTANTE = 'Hola Mundo';
```

```
echo CONSTANTE;
```

```
?>
```

Nota: Las constantes son automáticamente GLOBALES y se pueden utilizar en todo el script. En cualquier función o diferentes bloques del código PHP, se van a poder utilizar. Con la palabra LET para crear variable no se puede utilizar en todos sitios, no es global. En el caso de las constantes, siempre son globales y se puede utilizar en todo el código/script.

\*Los nombres de las constantes deben empezar con una palabra o un guión bajo seguido de cualquier número, letra o guiones bajos.

Parte de código:



// La constante PI es un tipo de valor Double (significa que es decimal), en el caso que fuera un número entero (sería integer).

Si has definido una constante, será eso todo el rato.

➔ Instalación PHP Pluggin: Extensions: PHP Intelephense

Aquí sí que puedo tener otra variable.

Sí que podemos cambiar el valor de las variables. El de las constantes no. Lo voy mostrando mientras voy cambiando el valor de las variables. **Si ya has definido esa constante de una manera, no puedes darle otro valor.**

**Para concatenar en JS** se utiliza el **+** en cambio para concatenar en PHP se utiliza el **.** (punto).

## **6. Declaraciones PRINT ECHO COMMENTS**

### Declaraciones print/echo

Echo: muestra en pantalla más rápido.

➔ El proceso en ambos es igual.

### Declaraciones print / echo

Las diferencias son pequeñas: **echo** no tiene valor de retorno, mientras que print tiene un valor de retorno de 1, por lo que puede usarse en expresiones. Echo puede tomar múltiples parámetros (aunque tla uso es raro), mientras que print puede tomar un argumento. (print en php.net).

echo	print
It is faster than print.	It is slower than echo.
It accepts single data as well as multiple data also. Multiple data separated with comma (,).	It accepts only single data.
It doesn't return any type of value.	It always returns 1.

➔ El eco, por ejemplo, en las funciones, no nos devolverá ningún valor.

### Declaración/sentencia print

Print no es realmente una función (es una construcción del lenguaje) por lo que no se requiere el uso de de paréntesis.

\*Solo se puede mostrar un parámetro: muestra etiquetas HTML:

<?php

```

$txt1 = "Aprende PHP";
$txt2 = "pr0j3ct.com";
$x = 5; $y = 4;
print "<h2>" . $txt1 . "</h2>";
print "Estudia PHP con " . $txt2 . "<br>";
print $x + $y; //9
?>

```

Implode

### Declaración/sentencia echo

La declaración echo se puede utilizar con o sin paréntesis: echo o echo ().

Muestra todos los parámetros:

```

<? php
echo "<h2> ¡PHP es divertido! </h2>";
echo "¡Hola mundo! <br>";
echo "¡Estoy a punto de aprender PHP! <br>";
echo "Esta", " cadena de texto ", "se ha", "creado", con múltiples parámetros.";?>

```

*Puede tener múltiples parámetros separados por comas y etiquetas HTML*

Recomendación: prueba este echo en un archivo .PHP para comprobar que funciona

**Echo** no es realmente una función (es una construcción del lenguaje), por lo que no se requiere el uso de paréntesis.

**Echo** (a diferencia de otras construcciones del lenguaje) no se comporta como una función, es decir no siempre se puede usar en el contexto de una función. Además, se quiere pasar más de un parámetro a echo, éstos no deben estar entre paréntesis.

Si queremos, utilizar comillas dobles siempre, deberíamos hacer lo siguiente:

Ejemplo:

➔ Echo "5 Para escapar caracteres se hace \"así\""

En PHP, se pueden utilizar variables también variables dentro. Igual que en el print, también se puede mostrar en talla el elemento. Si puedes modificar un string concreto puedes ir a modificar una variable.

### ¿¿ **el tema array** ¿¿

En una concatenación con un string tienes que poner ese espacio, para que figure ese espacio.

El tema de la variable End también es importante. No devuelve ningún valor. Print si que devuelve un 1. Print se puede utilizar como función función, echo no.

- ➔ (\$variable) ¿ print 'true' : print false; // print también es una construcción, pero se comporta como función. Puede usarse en este contexto.
- ➔ Echo \$variable ? 'true' : 'false' ;
  - Aquí se va a mostrar el resultado final, no una variable en concreto.

### Comentarios en PHP

Como hemos visto en otros lenguajes, es una forma muy sencilla de ocultar part del código o colocar explicaciones para programadores en el código. Tenemos 3 formas posibles.

En una concatenación con un string tienes que poner ese espacio, para que figure ese espacio.

Primera forma:

```
➔ <?php
    //Comentarios de una sola línea
?>
```

Segna forma:

```
➔ <?php
    #Comentarios de una sola línea
?>
```

Tercera forma:

```
➔ <?php
    /*Comentarios
    De más de una
    Línea*/ ?>
```

## **7. Tipos de datos**

Las variables pueden almacenar datos de diferentes tipos y los diferentes tipos de datos pueden hacer cosas diferentes.

PHP admite los siguientes tipos de datos:

- ➔ String (cadena de texto):
- ➔ Integer (número enteros)
- ➔ Float/double (números de coma flotante decimal)
- ➔ Boolean (true/false)
- ➔ Array (Matricues "arreglos")
- ➔ Object (clases y objetos)
- ➔ Nulll (variables sin definir el tipo)

*Si no se dice lo contrario cada variable tendrá el tipo de dato que le asignemos.*

### PHP String

String es uan secuencia de caracteres, como "¡Hola munda!"

Un string puede ser cualquier texto entre comillas. Se puede utilizar comillas simples o dobles.

<?php

### PHP Integer

Més info: Nos devuelve un warning si no se puede sumar un número integer con un string que no es número”.

### PHP Float

los números de coma flotante (también conocidos como “flotantes”, “dobles” o “números reales”) se pueden especificar mediante cualquiera de las siguientes sintaxis.

La comunidad dice que hay pocas diferencias entre el Double y el Float. Así que se tratan de igual forma, son decimales.

- ➔ No hay diferencia práctica y realista. Nosotros vamos a colocar la asignación de la variable y ya está. Diferencias pocas. EL hilo del overflow y prácticamente no tiene ninguna diferencia.

<?php

```
$foo = True; // asigna el valor TRUE a $foo
```

?>

Si es true será un 1 y es false, no aparecerá ni el 0. Para especificar un tipo boolean se emplean las constantes true o false. Ambas no son susceptibles a mayúsculas y minúsculas. En JavaScript daba true o false literalmente, y aquí nos lo pone de esta manera.

Normalmente el resultado de un operador que devuelve un valor de tipo boolean es pasado o a una estructura de control (if/else, while, for...)

### PHP Boolean

Un booleano representa dos estados posibles: Verdadero o Falso, true/falso, abierto/cerrado, 0 o 1.

Este es el tipo más simple. Un boolean expresa un valor que indica verdad. Puede ser true (verdadero) o false (falso).

### PHP Array (arrays indexadas)

Es una variable que puede guardar uno o más valores en si misma. Hasta teníamos un solo valor y con array, podemos tener más de un valor en una misma variable.

- ➔ Guarda un conjunto de valores que se puede asemejar en una base de datos.

Este tipo de dato está optimizado para varios usos diferentes:

- ➔ Matriz
- ➔ Lista (vector)
- ➔ Tabla hash (una implementación de un mapa)
- ➔ Diccionario
- ➔ Colección
- ➔ Pila

## → Cola

Como los valores de la matriz pueden ser otras matrices, también son posibles árboles y matrices multidimensionales.

### PHP array (asociativas)

Array: también llamada arreglo o matriz. Una matriz almacena varios valores en una variable. U

n array (arreglo es\_US) PHP es en realidad un mapa ordenado. Un mapa es un tipo que asocia valores a claves (=>)

Para ir cogiendo las variables de la array se pone el nombre de la variable y entre paréntesis las diferentes variables del array:

```
<?php
$array = array(
    1 => "a",
    "1" => "b",
    1.5 => "c",
    true => "d",
);
var_dump($array);

// Sintaxis corta
$array = [
    "clave" => "valor1",
    "clave2" => "valor2",
];
?>
```

```
$profesor = array('nombre' => 'Marc', 'telefono' => 665533, 'edad' => 35, 'apellido' => 'Esteve', 'ciudad' => 'Sant Boi');

# Igual que en los arrays indexados, en los asociativos también podemos modificar sus valores simplemente accediendo a ellos.
$profesor['apellido'] = 'Esteve García';

echo 'El teléfono ' . $profesor['telefono'] . ' es de ' . $profesor['nombre'] . ' que vive en ' . $profesor['ciudad'] . ' . ' . '<br>';

echo 'El nombre del profesor es ' . $profesor['nombre'] . ' . ' . $profesor['apellido'] . ' . ', su edad es ' . $profesor['edad'] . ' . ', su teléfono es el ' . $profesor['telefono'] . ' . ' y imparte clases en ' . $profesor['ciudad'] . ' . ' . '<br>';
```

→ Cuando queremos llamar a un valor de la variable, lo haremos con diferentes <br>.

→ Cuando queramos llamar a todos los elementos lo haremos a través de var implode

- El var implode (aguanta el \$glu que aguanta dos argumentos)
  - El echo implode nos muestra toda la información
- El var dump (solo soporta un argumento que colocamos el array).

### Array multidimensionales

```
# Los arrays multidimensionales nos permiten tener arrays dentro de otros arrays, como una MATRIZ.
$hola = "Hola";

$amigos = array(
    array('Marc', 35, true),
    array('Héctor', 33, 3.14),
    array('Daniel', 38, $hola) //incluso puede tener otra variable dentro
```

Los array multidimensionales aparecen cuando tenemos más variables de tipo array dentro de otras array y de esta manera se convierten en matrices de datos con múltiples “columnas y filas” y para acceder a ellas lo haríamos por dos coordenadas. \$variable[x][y];

```
<?php
```

```
$amigos = array(
array('Marc', 35, true),
array('Héctor', 33, 3.14),
array('Daniel',
```

\*En la base de datos tendrías como unas tablas relacionadas, con toda la información. Irías a ordenarlo en base al campo. Ordénamelo en base al precio, ordénalo en base a no se que.

*\*Exponente que no sea en base 10. Si hacemos 1.3e3, es 1.3 por 10 al cubo. En cambio si no se puede hacer de la siguiente manera:*

### Función count

Gracias a la función **count** podemos contabilizar una variable. Con el **count vamos** a saber cual es la posición. Cuantos valores tiene ese objeto o ese array. Nos devuelve un integer.

- ➔ **En índice, el 0 también es un valor, y por lo tanto existe.**
- ➔ Si el array va modificándose y expandiéndose necesitas poder contar hasta donde será el último.

```
echo count($meses); //función que cuenta valores de array
echo "<br>";
$ultimo_mes = count($meses) - 1; //guardas el último valor
echo $meses[$ultimo_mes]; //muestras la última posición del array
echo "<br>";
```

- ➔ **Con el count -1, no tendré que cambiar el valor, valga lo que valga. Será diciembre normal.** Me daría febrero x1000 o el valor que sea, correspondiente allí.

Por lo tanto, hasta ahora, las funciones que hemos visto que se pueden utilizar:

```
echo implode($meses);
echo "<br>";
echo gettype($meses);
echo "<br>";
var_dump($meses);
```

En el caso **del var dump** (muestra toda la información, así como la tipología de la variable. De la misma manera solo agrupa un argumento. La diferencia con el **implode** es que solo nos muestra todos los valores de la variable, así como **aceptar** más de un argumento. De esa manera le podemos incorporar primero ("un string", \$elnombre de la variable).

### Recorrer array. Foreach

Se puede recorrer un array y ejecutar determinadas instrucciones por cada elemento del array. Es la forma más fácil de recorrello.

- ➔ La variable días, el primer valor (le asigna alias = as a una nueva variable). Es un recorrido por lo tanto solo va a recorrer todos los variables.

```
foreach($meses as $mes){  
    echo '<li>' . $mes . '</li>'; // echo $meses[0]; solo  
    repetiria el primero  
}
```

### Ordenar array resort / sort

- ➔ sort (\$array) // ordena Menor ➔ Mayor (0 al 100 / A la Z)
- ➔ rsort (&array) // Ordena inverso que sort (100 al 0 / Z a la A).

En teoría en un mismo array tendría que a ver

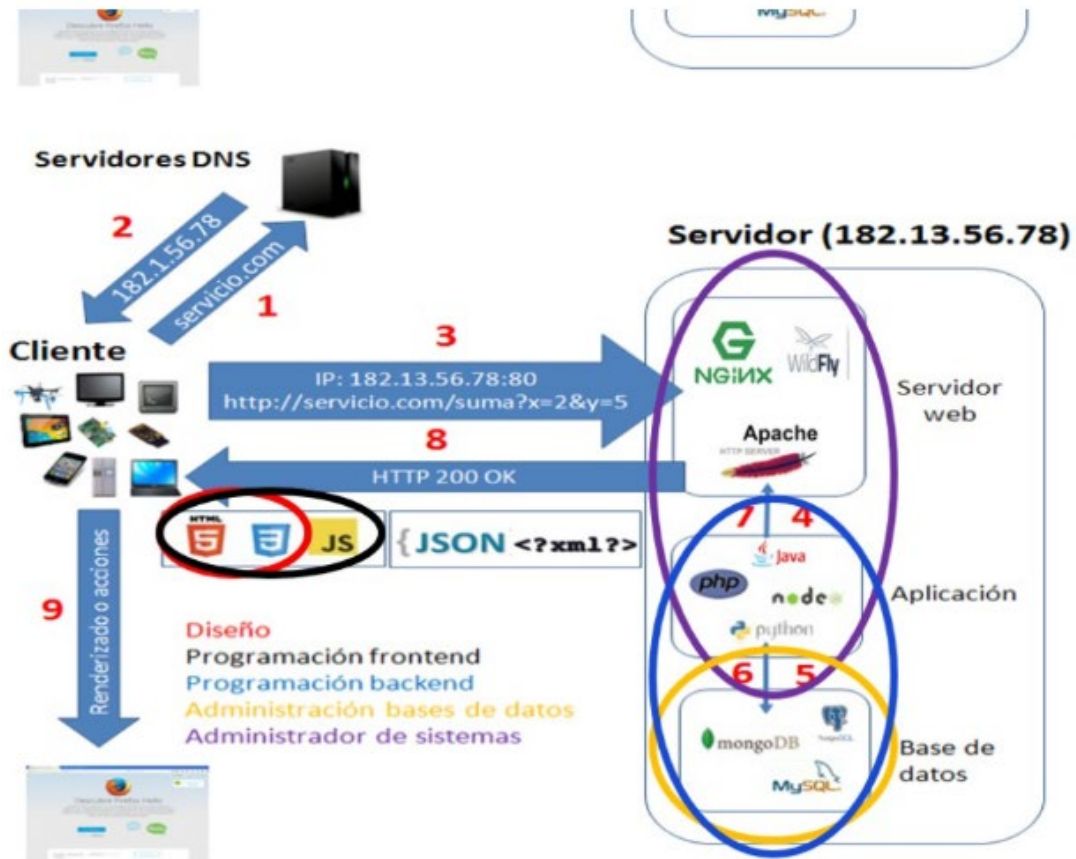
```
<?php  
    foreach($numeros as $numero){  
        echo '<li>' . $numero . '</li>';  
    }  
?>  
<?php  
    foreach($meses as $mes){  
        echo '<li>' . $mes . '</li>';  
    }  
?>
```

### ESTRUCTURA RELACIÓN SERVIDOR - CLIENTE

\*L'arxiu és tipus PHP:

- ➔ BACK END
- Dins hi ha un arxiu PHP dins hi ha una part HTML (on hi ha incrustat PHP)
- ➔ FRONT END
- Curs HTML.

## PETICIÓN CLIENTE SERVIDOR (FRONTEND – BACKEND)



Llamamos al servidor, sabemos la web pero no al IP. Domain Name Server. Te manda la IP que realmente tiene es dominio.

### PHP Object y class

\*Instanciar. Cuando hablamos de instanciar, estamos hablando de instanciar una variable, eso lo que hace es convertirla en un objeto. Es decir en vez de que sea string, numérica o boolean, lo que hacemos es convertirla en un objeto.

1. Lo que tenemos que hacer en primer lugar es abrir una clase con la que definiremos una serie de características a través de variables y/o funciones.
2. Métodos: hablamos de las diferentes acciones que llevará a cabo el objeto.
3. La instanciación de la variable. En este momento la estamos convirtiendo en un objeto. Ya no será una tipología de variable sino que se habrá convertido en un objeto. Si hacemos `echo gettype($nombre del objeto)` nos mostrará tipo "object".

```
$tesla = new Coche(); // Realizamos una instancia de la clase Coche
// Crea el objeto $tesla, heredando los atributos y métodos
echo "<br>" . 'El color del coche es ' . $tesla->color . "<br>";
echo 'La velocidad del coche es ';
$tesla->acelera("200 km/h"); //Asignamos valores a la variable
argumento $velocidad
```



- `Var_dump(pow)(2,8);`

Los array multidimensionales aparecen cuando tenemos más variables de tipo array dentro de otras array y de esta manera se convierten en matrices de datos con múltiples “columnas y filas” y para acceder a ellas lo haríamos por dos coordenadas. `$variable[x][y];`

- ➔ Las clases pueden ser:
- Public (la puedes utilizar)
  - Privada (puedes utilizar una clase heredada)
  - Protected

*Ejemplo 24-05-2022*

```
➔ <?php
➔ class Fruit {
➔     public $name;
➔     function set_name($name) {
➔         $this->name = $name;
➔     }
➔ }
➔ $apple = new Fruit();
➔ echo $apple->name ;//Undefined
➔ $apple->set_name("Apple");
➔ echo $apple-> name; //Aquí si que aparecerá Apple
➔ $apple->name="Orange"
➔ echo $apple->name; //Orange
➔
➔ $orange = new Fruit();
➔ $orange-> set_name("Orange.")
➔
➔ ?>
```

Si yo quiero llamar a los dos objetos, tendré que utilizar this. Es la garantía de que me sirve para los dos. Las diferentes formas de llamamiento son:

- ➔ `$objeto->metodo()`  
➔ `$objeto->atributo`

Llamando a la función no puedes poner un echo `$orange-> set name` (“Orange”, porque en la función no hay un return y por lo tanto no va a mostrar nada. Si queremos que muestre algo, tendría que poner un echo y un return. O bien se hace modificando directamente el atributo y se pone un echo. (`$orange->name="naranja";`)

```
<!DOCTYPE html>
<html>
<body>
<?php
class Fruit {
    public $name="pepito";
    function set_name($name) {
        $this->name = $name;
```

```

    }
}
echo 'Este es el Objeto $apple: ';
$apple = new Fruit();
echo $apple->name;//pepito
$apple->set_name("Apple");
echo $apple->name;//Apple
$apple->name="Orange";
echo $apple->name;//Orange
echo '<br> Este es otro Objeto $orange: ';
$orange = new Fruit();
echo $orange->name;//pepito
$orange->set_name("Orange");
$orange->name="naranja";
echo "<br> Mi fruta preferida es la " . $orange->name;
?>
</body>
</html>

```

## PHP object y class

Las clases y los objetos son los dos aspectos principales de la programación orientada a objetos (OOP\*).

- ➔ Una **class** es una plantilla para los **objects**
- ➔ Un **object** es una instancia de una **class**

## OOP Programación Orientada a Objetos

La OOP es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación, hay que destacar que en POO todo se define como un objeto.

Un objeto en la programación se define como na clase que contiene atributos y métodos, esta clase tiene la posibilidad de heredar datos de otras y también poder definir algunos métodos especiales.

## EJEMPLO BÁSICO

Una persona puede tener **propiedades**: nombre, edad, calzado...Esto son los **atributos** del objeto en OOP (ID, título, CONtenido...), es decir, definir **variables**

Una persona puede realizar **acciones**: correr, hablar, escuchar...Esto son los **métodos** del objeto en OOP (Crear, editar, eliminar, ver...), es decir, definir **funciones**.

## OOP Diferencia con el resto de programación

- ➔ **Programación lineal**: Es cuando desarrollamos todo el código disponiendo instrucciones PHP alternando con HTML de la página.
- ➔ **Programación Estructurada**: Es cuando planteamos funciones que agrupan actividades a desarrollar y luego dentro de la página llamamos a dichas funciones que pueden estar dentro del mismo archivo o en una librería separada.

➔ **Programación OOP:** Cuando planteamos **clases** y definimos objetos de las mismas.

```
<?php
//Creamos la clase Profesor
class Profesor {
//Atributos
public $nombre = 'Marc';
//echo $nombre; no funciona
//Métodos
public function hablar($mensaje){
echo $mensaje;
}
}
$persona = new Profesor();
// echo $persona->nombre;
// $persona->hablar("Un cordial saludo"); ?>
```

PHP Object / class

Programación OOP:

Es cuando planteamos clases y definimos objetos de las mismas.

En este ejemplo se crea la class Profesor{} con un **atributo variable \$** nombre y un método

## **8. Operadores en PHP**

### Operadores aritméticos

Los operadores aritméticos de PHP se utilizan con valores numéricos para realizar operaciones aritméticas comunes, como sumas, restas, multiplicaciones, etc.

En la siguiente tabla se muestran ejemplos con \$x=10 y \$y=6 para los siguientes resultados sencillos aritméticos.

Operador	Nombre	Ejemplo	Resultado
+	Suma	$\$x + \$y$	10
-	Resta	$\$x - \$y$	4
*	Multiplicación	$\$x * \$y$	60
/	División	$\$x / \$y$	1
%	Resto	$\$x \% \$y$	4
**	Exponente	$\$x ** \$y$	1.000.000

#### Operadores de asignación PHP

Operador	Similar a	Descripción
$x = y$	$x = y$	El operando de la izquierda se establece en el valor de la expresión de la derecha.
$x += y$	$x = x + y$	Suma
$x -= y$	$x = x - y$	Resta
$x *= y$	$x = x * y$	Multiplicación
$x /= y$	$x = x / y$	División
$x \% = y$	$x = x \% y$	Resto

#### Operadores de comparación de PHP

Los operadores de comparación de PHP se utilizan para comparar dos valores.

#### Operadores de comparación de PHP

Los operadores de comparación de PHP se utilizan para comparar dos valores.

Los operadores de comparación de PHP se utilizan para comparar dos valores.

Operador	Nombre	Ejemplo	Resultado
==	Igual que	\$x == \$y	Devuelve verdadera si \$x es igual a \$y
===	Idéntica	\$x === \$y	true si es igual valor y tipo
!=	No igual	\$x != \$y	true si es diferente valor
<>	No igual	\$x <> \$y	true si es diferente valor
!==	No idéntica	\$x !== \$y	true si es diferente en valor y <b>tipo</b>
>	Mayor que	\$x > \$y	true si es mayor x que y
<	Menor que	\$x < \$y	true si es menor x que y
>=	Mayor igual	\$x >= \$y	true si x es mayor o igual que y
<=	Menor igual	\$x <= \$y	true si x es menor o igual que y
<=>	Spaceship*	\$x <=> \$y	-1 si x<y, 0 si x=y, 1 si x>y

\*Spaceship: es como una especie de boolean, de hecho por eso se llama comparativo. Si es más pequeño o más grande da un un menos uno. Todos son booleanos.

#### Operadores de incremento / decremento

Los operadores de incremento/decremento de PHP se utilizan para incrementar/decrementar el valor de una variable.

Operador	Nombre	Descripción
++\$x	Preincremento	Incrementa \$x en uno, luego devuelve \$x
\$x++	Postincremento	Devuelve \$ x, luego incrementa \$ x en uno
--\$x	Predecremento	Decrementa \$x en uno, luego devuelve \$x
\$x--	Postdecremento	Devuelve \$x, luego decrementa \$x en uno

Primero puedo incrementar o mostrar y posteriormente se guarda. Si primero muestro muestro y luego sumo, no se verá el incremento hasta la siguiente sentencia.

#### Operadores de cadenas PHP

PHP tiene dos operadores que están especialmente diseñados para cadenas.

Operador	Nombre	Ejemplo	Resultado
.	Concatenar	\$x . \$y	Concatenación de ambos textos o variables
.=	Concatenar asignando	\$x .= \$y	A \$x le añade el valor de \$y

La diferencia con los operadores de asignación es que uno es numérico y cambio los otros son string.

```
$x"Hola";
```

```
$y = "¿Qué tal?";
```

```
//$sx = $y →
```

```
---
```

```
X += y → x = x + y
```

```
X .y → x = x . y
```

*Nos sirven para inputs del usuario.* Le podrías dar valores a un string a datos que el usuario está generando.

### Operadores lógicos de PHP

Los operadores lógicos de PHP se utilizan para combinar declaraciones condicionales

Operador	Nombre	Ejemplo	Resultado
and	And	\$x and \$y	Verdadero si tanto \$x como \$y son verdaderos
or	Or	\$x or \$y	Verdadero si \$x o \$y son verdaderas
xor	Xor	\$x xor \$y	Verdadero si \$x o \$y son verdaderas, pero no ambas
&&	And	\$x && \$y	Ídem que <i>and</i>
	Or	\$x != \$y	Ídem que <i>or</i>
!	Not	\$x > \$y	Verdadero si \$x no es cierto

Con el **and** solo cuando las condiciones (tanto x como y) son verdaderos. Si se cumple, se ejecuta.

**Or:** si la condición 1 o la condición 2 es verdadero. Yo lo consigo. La condición uno no se cumple pero la otra. Solo si, la condición 1 o la 2 será false. Sería como la suma, o se cumple o se cumple la otra.

**Xor:** Siempre y cuando alguna de ellas sea verdadera. Esto lo que hace es unir condiciones. Establecemos que son verdaderas, con el and, o con shore.

**&&**

**||**

**¿**

Es importante el tema de lo del código. Por ejemplo si asignamos:

```
&nombre =
```

Un if con un and, siempre que sea dos iguales asignación será incompatible con un más grande del mismo número.

\*Si hay algún xor y posteriormente hay un or, el xor es más intransigente y por lo tanto se va a hacer antes.

\*En el caso del If con una variable que es un bool

### Operadores de matriz PHP

Los operadores de matriz de PHP se utilizan para comparar matrices.

Operador	Nombre	Ejemplo	Resultado
+	Unión	$\$x + \$y$	Unión de $\$x$ y $\$y$
==	Igualdad	$\$x == \$y$	Devuelve verdadero si $\$x$ y $\$y$ tienen los mismos pares clave/valor
===	Identidad	$\$x === \$y$	Devuelve verdadero si $\$x$ y $\$y$ tienen los mismos pares clave/valor en el mismo orden y de los mismos tipos
!=	Desigualdad	$\$x != \$y$	Devuelve verdadera si $\$x$ no es igual a $\$y$
<>	Desigualdad	$\$x <> \$y$	Devuelve verdadera si $\$x$ no es igual a $\$y$
!==	No idéntica	$\$x !== \$y$	Devuelve verdadera si $\$x$ no es idéntica a $\$y$

\*En el caso del + con array, hace que se puedan sumar los diferentes valores.

\*Print\_R() muestra en pantalla la suma y en este caso, la suma es una fusión de las claves y valores de ambas matrices. Dentro del print r vamos a incorporar ( $\$x + \$y$ ).

```
$x = array("a" => "10", "b" => "green");
$y = array("b" => "green", "a" => 10 );

// var_dump($x == $y); //BOOL(TRUE)
// var_dump($x === $y); //BOOL(FALSE)
```

\*Si se pone shift y comillas entre lo que se quiere poner

\*Shift +  $\text{c}$  se comenta

Cuando sea == (igualdad) tiene que ser igual el valor, pero no el tipo.

En cambio cuando sea === tiene que ser tanto la clave como el valor, del mismo tipo y del mismo valor.

### Operadores de asignación condicional de PHP

Los operadores de asignación condicional de PHP se utilizan para establecer un valor en función de las condiciones

Operador	Nombre	Ejemplo	Resultado
?:	Ternario	$\$x = expr1 ? expr2 : expr3$	Devuelve el valor de \$x. El valor de \$x es expr2 si expr1 = VERDADERO. El valor de \$x es expr3 si expr1 = FALSE
??	Fusión nula	$\$x = expr1 ?? expr2$	Devuelve el valor de \$x. El valor de \$x es expr1 si expr1 existe y no es NULL. Si expr1 no existe, o es NULL, el valor de \$x es expr2.



**\* Nota: Introducida en PHP7**

**La función empty:** le estamos preguntando si está vacía la variable (si está vacía nos devuelve un true, si no nos devuelve un false).

Combinar la función empty con el operador ternario y la fusión nula.

A la variable &status = (empty(&user)).

➔ Está vacío el valor de la variable. Le asigna el segundo valor si es verdadero, sino le asigna el tercero.

&color="blue".

Echo &color = \$color ¿? "rojo";

## 9. Math en PHP

PHP tiene un conjunto de funciones matemáticas que le permite realizar tareas matemáticas con números

Función PHP pi ()

```
<?php
echo(pi()); // 3.1415926535898
?>
```

La **función pi()** devuelve el valor de PI:

Tal como se enseña en la escuela, PI es el número que se obtiene al dividir la longitud de una circunferencia por su diámetro. Por eso muchos lo asocian exclusivamente a la geometría, más específicamente, al cálculo del perímetro y área de un círculo.

Otros cálculos: que se pueden hacer:

➔ Los relojes de péndulo

➔ Cálculos espaciales GPS

➔ En la voz del móvil (T.Fourier).



### Función PHP Min () y Max ()

Las funciones min() y max() se pueden usar para encontrar el valor más bajo o más alto en una lista de argumentos:

```
<?php
echo(min(0, 150, 30, 20, -8, -200)); // -200
echo(max(0, 150, 30, 20, -8, -200)); // 150
?>
```

### Función PHP abs()

La función abs() devuelve el valor absoluto (positivo) de un número:

```
<?php
echo(abs(-6.7)); // 6.7
?>
```

### Función PHP sqrt()

La función sqrt() devuelve la raíz cuadrada de un número:

```
<?php
echo(sqrt(64)); // 8
?>
```

### Función PHP round()

La función round() redondea un número de punto flotante a su entero más cercano:

```
<?php
echo(round(0.60)); // 1
echo(round(0.49)); // 0
?>
```

### Números al azar rand()

La función rand() genera un número aleatorio:

```
<?php
echo(rand());
?>
```

Para tener más control sobre el número aleatorio, puede agregar los parámetros mínimos y máximos opcionales para especificar el entero más bajo y el entero más alto que se devolverá.

Por ejemplo, si desea un número entero aleatorio entre 10 y 100 (inclusive), use rand (10, 100):

```
<?php
echo(rand(10, 100));
?>
```

## 10. Estructuras de control PHP


En muchos lenguajes de programación existen un tipo de estructuras para repetir sucesos, realizar acciones en un determinado momento o cuando se cumplen una serie de condiciones, hasta un determinado valor de variables y constantes se harán u.

PHP admite las siguientes estructuras de control:

### IF ELSE ELSEIF

Cuando necesitamos realizar acciones en nuestro código de PHP, a través de condiciones de estados o valores de variables o constantes, podremos utilizar condicionales como IF, y siguiendo una estructura de sintaxis determinada, conseguir realizar acciones concretas.

```
<?php
IF (CONDICION){
    ACCION
}
ELSE{
    //ACCION
}
?>
```

**Nota: Sintaxis** 

**Función isset()** revisa si una variable tiene valor establecido o no.

```
if (isset($edad)) {
    $edad = $edad;
} else {
    $edad = 'El usuario no establecio su edad';
}
echo 'Edad: ' . $edad;
```

\*Podemos poner más de una variable que se asignará en función de cada caso o de cada situación de los condicionales. Cuando quiero tener muchos casos haría a lo largo.

➔ Operador ternario o atajo IF (condición) ? (para true este valor) : (para false este valor).

Cuando necesitamos realizar acciones en nuestro código de PHP, a través de condiciones de estados o valores de variables o constantes, podremos utilizar condicionales como IF, y siguiendo una estructura de sintaxis determinada, conseguir realizar acciones concretas.

*Con el else podemos poner diferentes condiciones.*

```
$mes = 'Junio';
$verdadero = true;

// if ($verdadero) {
//     echo "Verdadero";
// } else {
```

```
// echo "Falso";
// }

if ($mes == 'Diciembre') {
    echo "Feliz Navidad";
} else if($mes == 'Enero'){
    echo "Feliz Año Nuevo";
} else if ($mes == 'Julio'){
    echo "Feliz Julio";
} else if($mes == 'Junio'){
    echo "Bona revetlla";
} else if ($mes == 'Agosto'){
    echo "Felices vacaciones";
} else {
    echo "El mes no tiene celebracion";
}
```

Si dos else / else if se cumplen, saldrá el primero. La recomendación es que existan paratensis

### Switch

Cuando necesitamos realizar acciones nuestro código de PHP, a través de condiciones de estados o valores de variables o constantes, podremos utilizar condicionales como SWITCH, y siguiendo una estructura de sintaxis determinada, conseguir realizar acciones concretas.

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all
        labels;
}
```

en

### Bucles

#### Estructura Foreach

Lo primero declaramos los dos array: el normal indexado y el asociativo.

\*Con el foreach va a recorrer los meses del año.

\*Podemos recorrer arrays asociativos también:

```
Foreach($nombrevariable as $nombre final variable => $elvalor){
    Echo '<li>' . $nombrefinal . ' : ' . $elvalor . '</li>';
}
}
```

#### Estructura While

Mientras un elemento vaya de aquí aquí. Será esto.

```
while($x >= 1){  
    echo $x . '<br>';  
  
    $x--;  
}
```

### Estructura DO + WHILE

El bucle lo hará si se cumple la condición del final. Es más eficiente que el while porque ya ha llegado cuando ha realizado la acción.

➔ Tiene que ir junto el do con el while. Si no hay condición no se ejecuta.

### Estructura For

➔ Tiene:

- Inicialización
- Condición
- Incremento/decremento
- Y luego además tenemos acciones

Para una serie de elementos se mostrar algo. Puedes colocar hasta 3 argumentos en la función, entre ellos el valor de la variable (la inicialización de la variable, la condición y el incremento).

## Concepto scope

- ➔ En castellano es alcance. Cuando tienes un scope dentro de una i, podría darte la función. Si tu utilizaras esa fuera del bucle for.
- Hasta donde llega o no llega esa variable.
  - Una variable podría estar definida dentro y solo dentro.

Ejemplo de clase:

- Let solo va funcionar dentro del for.
- Si hacemos un echo sum dentro de la función dará un resultado. Si lo hacemos fuera no estará declarada.

Ex:

- ➔ En la línea 5 y 9: se mostrará lo que da la variable. Seguirá siendo un 7. Antes y después se mostrará un determinado elemento.
- ➔ La función sin argumentos, es como si fuera ciega, si no me llaman yo ejecuto mis argumentos. No me fijo en nada más.
- Pero si la variable a la que llamas dentro no está definida, hay que inicializarla.
  - Si no llamo a la función no se ejecuta. El número está definido. Pero está definido dentro.
  - A no ser que esté dentro del bloque la variable, fuera de la función seguirá dando 7.

- Hay que entender el scope de cada una de las variables. Aunque se llame variable global, pero no llega a todos los sitios.
- Pasa también en JavaScript. TypeScript. Java C++ y otros lenguajes de programación.

➔ Para que podamos **utilizarla dentro de la función**, habrá que colocarla como argumento/parámetro. Habrá que hacer la llamada de la función:

- La llamada está en la línea 16. Si no llamamos a la función no se hace nada en principio.

```
function mostrarNumero($numero){
    echo $numero
}
mostrarNumero($numero);
```

➔ Si queremos **sacar el valor de la función**, cuando llamemos a la función hay que utilizar el return dentro de la función. Me devuelve el número.

- Con el return va a guardar el valor. Lo va a devolver. Pero no lo va a mostrar.
- En cambio si le ponemos un echo también lo va a devolver, pero además lo mostrará cuando se llame a la función.

```
function mostrarNumero(){
    $numero = 10;
    return $numero;
}

echo mostrarNumero();
$numero = mostrarNumero();
echo $numero;
```

## 11. Funciones

El verdadero poder de PHP proviene de sus funciones.

- Funciones integradas de PHP: PHP tiene más de 1000 funciones integradas que se pueden llamar directamente, desde un script, para realizar una tarea específica.
  - Consulte nuestra referencia de PHP para obtener una descripción general completa de las [funciones integradas de PHP](#).
- Funciones definidas por el usuario de PHP:
  - Además de funciones PHP integradas, es posible crear tus propias funciones.
    - Una función es un bloque de declaraciones que se pueden usar repetidamente en un programa.
    - Una función no se ejecutará automáticamente cuando se cargue una página.

- Una función se ejecutará **mediante una llamada a la función**.

Se coloca la función con los argumentos entre comas ( , )

## Funciones PHP

Crear una función definida por el usuario en PHP

Una declaración de función definida por el usuario comienza con la palabra function:

### Sintaxis

```
<?php
function nomFunc(){
    //código de la acción
}
nomFunc(); //llamada
?>
```

\* El nombre de una función debe comenzar con una letra o un guión bajo. Los nombres de las funciones NO distinguen entre mayúsculas y minúsculas.

Consejo: ¡Asigna a la función un nombre que refleje lo que hace la función!

En el siguiente ejemplo, creamos una función llamada "escribirMensaje()". La llave de apertura ( { ) indica el comienzo del código de la función y la llave de cierre ( } ) indica el final de la función. La función emite "Mensajeescrito". Para llamar a la función, simplemente hay que escribir su nombre seguido de paréntesis ( ):

```
<?php
function escribirMensaje() {
    echo "Mensaje escrito";
}

escribirMensaje(); // llamada a la
función
?>
```

**Nota:** Prueba el código y cambia las mayúsculas y minúsculas de la función al llamarla.

## Consejos de Visual Studio Code

---

*Si subrayas y pones luego shift y luego un paréntesis.*

*Si subrayas y pones shift 2 y luego se pone comillas.n*

---

Cuando podemos **result**, se guarda el resultado pero no se muestra.

- ➔ Documento "funciones".
  - El valor después de echo saludo, es el argumento que se le está otorgando a la función saludo.

Cuando ponemos phpinfo nos da información sobre php que es interesante tenerla en cuenta.

## Funciones cadenas de texto

**Función htmlspecialchars:** lo utilizamos cuando queremos mostrar tal cual la información. Por problema de seguridad, no deberías interpretarlo como script. Que lo coja como texto y punto.

- ➔ Estaría llevando a cabo cualquier tipo de acción. Es un filtro mínimo, para recibir un string.

FOTO

**Función echo trim** Si le hemos puesto espacios a un nombre, en la base de datos se guardará mal. Eso podría rellenar de una forma no demasiado buena esa base datos. Quitar espacios

FOTO

**Función strlen (&string);** combinando las dos podemos coger desde la cifra que nos diga hasta el último.

FOTO

**Función echo substr:** corta el string desde el primer valor de 0 hasta el 4. Podríamos almacenarlo además en otra variable.

FOTO

+Copiar siempre las variables, aunque estén mal. No poner ni números la principio.

**Función strtoupper:** lo convierte en mayúscula todo, aunque ya haya alguna en minúscula.

**Función strtolower:** Va a poner todo lo que hay en ese string en minúscula.

**Función strpos:** solo tiene dos argumentos. Sobre todo sirve para identificar la posición de un carácter y número.

**Función strrev:** lo que hace es el reverse. Le da la vuelta. Le da la vuelta a la cadena de texto. En vez de Roma → Amor.

**Función str\_replace:**

- ➔ Se reemplaza un texto por otro. (DONDE PONÍA, LO QUE PONE AHORA, LA VARIABLE A LA QUE HAGO REFERENCIA). Se cambia de esa manera, con el paréntesis y con lo que pone ahora.
- ➔ Hay 3 argumentos, lo que ponía originalmente, lo que pone ahora y la variable a la que esto haciendo referencia.

Función die

Es una función por la cual el resto del código se deja de ejecutar.

```
<body>
  <h1>Hola Mundo</h1>
  <?php die(); ?> <!--Las siguientes declaraciones no se muestran
```

Funciones array

**La función extract:** lo que hace es que las diferentes claves del array pasan a ser variables cada una por separadas. Solo sirve con la array asociativa. Las convierte TODAS, no pueden seleccionar cuales si o cuales no.

Cuando hacemos el **array pop** tipo `&ultimo_dia = array_pop($semana)` ; extrae el último valor de un array (indexado) y además se lo quita al array inicial. Lo hemos probado con el foreach.

En el caso de la función **array push**, lo que hacemos es añadir una serie de variables a la array push.

El **echo join**: une todos los valores de un array con un elemento. "Podría ser un elemento un salto de línea". Todas las funciones que puedan eliminar elementos de concatenación siempre simplificarían el tema.

**\$ARRAY\_reverse**: lo que va a hacer es leer el array al revés. Es decir le va a dar la vuelta al array.

### Funciones matemáticas

#### Include require index

El **include** y el **require** hacen lo mismo. En el segundo caso es obligatorio para compilar y ejecutar el código. Coge todo el código de un PHP y lo incluye en otro. La diferencia es el nivel de exigencia entre los dos.

Ejercicio: hacen la misma llamada. Lo diferente aquí es el que le nombre de las funciones ha cambiado. PHP es uno de los lenguajes más dinámicos. Podemos tener la variación de un código en un mismo documento y que nos de cosas diferentes sin solo cambiando la raíz.

- Se puede hacer con los dos métodos. Tanto con HTML como con PHP. El resultado va a ser casi lo mismo. (no tienes que ponerlo en los dos documentos).
- Sirven en las páginas webs, para hacer un require, Ver más, entonces llamas al resto de código.

## **12. App Formulario de contacto PHP**

La forma de conectar el **input** con el label, es a través del FOR. El for conecta la palabra hombre con el radio.

El método es como lo envío y el action es donde lo envío.

➔ El for conecta con la id:

Estructura:

- ➔ Type: checkbox
- ➔ Name: terminos
- ➔ Id "términos"
- ➔ Value: ok
- ➔ <br>
- ➔ <input type="submit" value="Enviar">

### Ejemplo form- index:

El resultado del ejemplo index lo envío a otro documento. En el otro ejemplo se queda en el mismo formulario.



## Variable Get

**El recíbe get, solo se podrá abrir desde el VSC.** Al enviarla por el método get, se guardará a través de un array asociativa. \$\_GET, variable global en PHP. Obtenemos un array de valores, asociativo.

De un archivo Form-index.php, enviamos una serie de datos con el método get. Lo recibe -get.`h`-

Cuando nosotros accedemos de form index con ese formulario y le damos a enviar.

Form-index.php → (Método GET admin)→ Recibe -Get. Php \$GET (array datos form).

**\*Es una variable global, ya existe de por sí.** Cuando enviamos el método get ya obtiene esa variable. Lo veremos en el navegador y lo guardaríamos en la variable Get. No guarda nada, simplemente lo archivo. Tiene que proceder de otro sitio. Solo lo recibe de un sitio concreto, tendrá efectividad.

Desde post se recibe en oculto. Post sería el más seguro. Echo en torno de pruebas.

Print\_r(\$\_POST):

➔ **Le asignamos a un variable nombre una parte del array post que se llama nombre.**

\*Aprovechando que la variable post está guardando arrays, aprovechamos para sacar cada uno de la expresión.

Hacemos una concatenación, después de vincular el formulario.

\*El Get no sería la opción recomendada para una web. Podríamos decir que es Front-end. No haría falta ni si quiera PHP. **En cambio, en el caso de Post, es Back-End, no lo muestra ni si quiera en la dirección.**

## Método post

**Con el método post, nos da más seguridad. Además tenemos la opción de que nos redirija directamente al formulario.**

```
if(!$ _POST) { //Si no hay datos daría error
    header('Location:http://localhost/cursoweb2022/PHP/UF1845formulari/12
.ejemplo-form-index2.php'); //Carpeta y archivo para rellenar el form
correcto
}

$nombre = $_POST['nombre'];
$sexo = $_POST['sexo'];
$fecha = $_POST['fecha'];
$terminos = $_POST['terminos'];

echo 'Hola ' . $nombre . ' eres ' . $sexo . ' y has escogido la fecha: ' .
$fecha ;
```

En cada posición. Le estamos asignando a la variable un elemento. Ese nombre es que el usuario ha escrito al formulario. Recibiremos lo que el usuario ha establecido.

**Cuando se abre directamente el archivo recibe post nos va a re direccionar.** No tiene ningún valor. No podemos abrir el post directamente. Hay que añadirsele mediante un formulario previamente.

Se le podría poner la variable que preferíamos. En vez de asignárselo a otras variables se le podría llamar ahí. Nos podemos ahorrar bastante texto.

\*Establecer que nos salga: **“el usuario no ha aceptado los términos”** si no nos sale nada”.

### Valida datos formulario

En la parte principal tenemos un filtro bastante largo.

Tenemos 3 inputs.

- ➔ Tipo texto
- ➔ Tipo texto
- ➔ Tipo submit.
  - Cuando nosotros clicamos es cuando podemos poner un action.

### Filtro IF que se está aplicando.

Submit ya tiene un valor enviar. Si está seteado. Si tiene valor significa que hemos hecho clic en el button.

- ➔ Le asignamos a un valor a las dos variables.
- ➔ Si no está vacío
  - Que el nombre tenga un filter\_var(\$nombre, FILTER\_SANITIZE\_STRING).
    - Sanear significa quitarle los espacios por delante y por detrás. Posibles códigos en lugares donde no tendría que haber código.
    - Validar significa, es correo o no es correcto. Es lo que nos está preguntando aquí.
  - Si está vacío
    - Cuando guardamos una variable queremos que se guarde correctamente. De ahí el trim.
    - También cuando escribimos código y no se puede ejecutar. Tendríamos que poner htmlspecialchars.
    - Miramos un email, filtramos saneado el email. No podremos poner ni caracteres especiales, ni espacios, sin barras.
    - FILTER\_SANITIZE\_STRING → No funciona. Hay que poner los otros tres:
      - Trim
      - htmlspecialchars
      - stripslashes(&xx)
    - Filter Validate Email: Le
  - If empty email
    - Tendremos un filtro, de que se cumplan las condiciones del correo. Sanea el correo
    - Luego validará el formato del correo, si es el que tendría que ser perfecto si no, pondrá : “Por favor escribe un correo válido”.
- ➔ Se le puede dar un estilo con el CSS.

### Ejercicio clase

Las cambiamos las variables.

- ➔ Si en vez de poner una concatenación ponemos un = estamos haciendo la asignación, así no tendremos que declarar las variables arriba por separado. Si queremos declararlas, entonces ya podremos asignarlas y llevar a cabo la concatenación.

### Documento de valida datos.

Ejercicio 1:

Estamos poniendo que nos muestro cual es el tipo de envío.

Ejercicio 2:

Poner dos formularios con tipo GET.

Ejercicio 3:

Nos pide reconocer que tipo de método es. Antes de nada.

- ➔ Si es tipo GET, entra dentro del IF. Y mostrará tanto para formulario 1 como para formulario 2. Los dos elementos.
- ➔ Si es tipo POST se irá al ELSE.

### Aplicativo concreto

VIEW → "HTML" plantilla (require PHP)

STYLE → "CSS" formato

LÓGICA → "PHP" COMPORTA -variables, - method,

Comporta

- Variables
- Method
- Submit
- Action

### Estilo

Le hemos quitado los margin y los paddings. Al body, le pedimos un color gris.

**Se utiliza la clase.wrap que es de Bootstrap.**

Con la hoja de estilo le estamos dando un formato importante al formulario. EL index view es el html que pedirá el PHP index.php

Puedes poner también cuales son los mensajes alert. Cual es la clase de esots mensajes.

- ➔ Alert error: hay algún problema de envío. Texto rojo.
- ➔ Alert succes: color correcto, en verde. Todo lo demás hay varios inputs. +
- ➔ Alert son todos: hace un recuadro para todos.

.btn es el botón. Btn y btn primary establecen el estilo.

Además tenemos el hover para cuando pasamos el ratón por encima.

### **13. Sesiones //Acabar apunte de sesiones que no está acabado.**

#### **Sesiones en PHP**

**Una sesión es una forma de almacenar información (en variables) que se utilizará en varias páginas.** A diferencia de una cookie, **la información no se almacena en el dispositivo del usuario.**

#### **¿Qué es una sesión PHP?**

Cuando se trabaja con una aplicación, **se abre, se hace algunos cambios y luego se cierra.** Esto es muy parecido a una sesión. El dispositivo sabe quién eres. Sabe cuándo inicia la aplicación y cuándo finaliza. Pero en Internet hay un problema: el servidor web no sabe quién eres ni qué haces, porque la dirección HTTP no mantiene el estado.

Las **variables de sesión** resuelven este problema al almacenar la información del usuario que se utilizará en varias páginas (por ejemplo, nombre de usuario, color favorito, etc.). **De forma predeterminada, las variables de sesión duran hasta que el usuario cierra el navegador.**

Entonces; **Las variables de sesión contienen información sobre un solo usuario y están disponibles para todas las páginas en una aplicación.**

#### **Sesiones en PHP**

Las variables con las que funcionan las sesiones son:

- ➔ `Session_start()` (con esta es con la sesión que empezamos). Así se inicia una sesión.
- ➔ `$_SESSION`: Las variables de sesión se establecen con la variable global de PHP: `$_SESSION`.
- ➔ `Session_destroy()`: Se cierra una sesión con la función `session_destroy()`.

\*Si se necesita almacenamiento permanente, es posible que desees almacenar los datos en una **base de datos**.

➔

### **14. Cookies**

Una cookie se usa a menudo para **identificar a un usuario**. Una cookie es un pequeño archivo que el servidor incrusta en el navegador del dispositivo del usuario. Cada vez que el mismo dispositivo solicita una página con un navegador, también enviará la cookie. **Con PHP, se puede crear y recuperar valores de cookies.**

\*A diferencia de la sesión que se guarda en esa variable global, la cookie también se guarda en el navegador. También enviará la cookie cuando se solicita una página del navegador.

- ➔ Remarketing: cuando te sale algo de lo que te estabas informando en otra página web.
  - Sabes que esa persona ha visitado otra web anterior. Si tú tienes esa alerta de que se produce una cookie de un Publisher, de alguien que

quiere hacer publicidad, aparece una cookie de otra web, sino mostrará una publicidad aleatoria.

\*Es un texto que se guarda de manera codificada o no.

**Las cookies son como archivos de texto que se guardan en el ordenador.** A petición de un servidor web, el navegador crea un archivo de este tipo. Una vez que ocurre esto el servidor puede leer y escribir contenido en este archivo. Puede parecer peligroso, pero existen una serie de restricciones para hacerlo lo más seguro posible:

- ➔ **Los servidores web sólo pueden acceder a cookies establecidas a su propio dominio.** Este dominio es establecido por el navegador cuando el servidor crea una nueva cookie, y sólo puede ser un dominio o subdominio del servidor.
- ➔ Según el **protocolo HTTP**, las cookies no pueden ser más grandes de 4096 Bytes (4KB). Hay un límite de cookies por dominio. Depende del navegador, pero suelen ser 20 cookies.
- ➔ También hay **un límite en el número total de cookies** en el disco duro del cliente. Suele ser de unas 300 cookies. Cuando se llega a este número, una cookie antigua se elimina antes de crear la nueva.

Las cookies **tienen una fecha de expiración**. Esta fecha permite al navegador eliminar cookies antiguas cuando ya no son requeridas por el servidor web. Si la fecha de expiración está vacía, **la cookie se eliminará cuando finalice la conexión con el servidor**. Esto ocurrirá cuando el usuario cierre la ventana o pestaña del sitio web, o directamente cierre el navegador. Estas cookies, normalmente denominadas sesión cookies, son usadas principalmente para guardar ajustes temporales.

\*Sin cookies habría cosas que no funcionarían bien. Por ejemplo, si no tuviéramos cookies el carrito de las ecommerce no funcionarían bien. Si borramos las cookies, el inicio de sesión se habrá borrado. Las cookies, **normalmente son cookies técnicas**. Si por casualidad, has visitado diferentes productos, artículos. En base a los artículos de productos o de blog, puede almacenar información y darte información personalizada en función de lo que has visitado.

**1 cookie no es una persona. La misma persona puede tener varias cookies.**

\*A través de navegador se guarda por cookies. **En una app, se guarda a través de usuario**. Ese usuario es equivalente a otro usuario de aplicación. Tiene que ver con los usuarios que se han generado.

- ➔ **Cookies sería más bien para navegadores.**
- ➔ **Sesiones es para usuarios.**

### Cookies PHP

Setcookie()

\*Se crea una cookie con la función setcookie()

### SINTAXIS

Setcookie (name\*, value\*, expire\*, path\*, domain\*, secure, httponly, options)

- ➔ **Name**=nombre de la cookie

- ➔ **Value=valor** asignado a la cookie (\$\_COOKIE['name']=value)
- ➔ **Expire**= fecha de expiración
- ➔ **Path**= La ruta dentro del servidor en la que la cookie estará disponible. Ej. '/' todo el dominio. El dominio de la cookie quiere decir donde está disponible.
- ➔ **\$\_COOKIE**
  - Una variable tipo array asociativa de variables pasadas al script actual a través de Cookies HTTP.

**Eliminar cookie:** Utilizar la función **setcookie()** con una fecha de expiración en el pasado (negativa).

#### Ejemplo de color red PHP

Veremos que además de los diferentes **valores se le puede insertar una duración determinada a la cookie**. Esto es importante porque es lo que va a durar la cookie.

Si queremos que nos alga la cookie, habrá que quitar el seteo para que nos pueda aparecer. Si no,

#### Ejercicio de clase

1. Crear una cookie color de fondo en un archivo llamado fondo.php
2. Fondo php → enlace a texto.php
3. Nombre = fondo  
Valor = color concreto  
Expira = 35
4. Modif. Texto.php
  - a. CSS → texto

### **15. Bases de datos //Acabar apuntes de base de datos, no están listos.**

#### Niveles de abstracción

Simplifica usabilidad de usuarios con Bases de datos (BBDD) y Sistema.

- ➔ Nivel físico: Dispositivos de almacenamiento Local o remoto. Nivel más bajo de abstracción. Dónde tenemos nuestra BD.
- ➔ Nivel conceptual: Definición + Relación de DATOS: Interno, como está construido.
- ➔ Nivel visión: Usuario visualiza BD en el sistema. Nivel más alto de abstracción. Diferentes vistas

#### Diagrama/modelo Entidad – Relación

La clave primaria es un identificador único. Solo lo tiene el esa clave primaria.

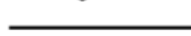
Elementos:



Objeto concreto o abstracto que figura en la BD



Establece relaciones entre entidades (mapeo y cardinalidad)



UNIÓN entre ENTIDADES



Características de las entidades (Nombre, Edad...)



Atributo principal o clave primaria (único por entidad y no puede ser nulo)



Atributo foráneo o clave secundaria

Clave foránea: Una que relaciona con otras tablas.

Cuando solo es uno se pone un 1:

Cuando es muchos se pone una n:

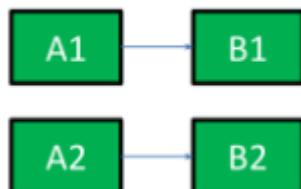
Quien obtiene la clave foránea es quien tiene muchos.

### Diagrama Entidad – Relación: Grado de cardinalidad

Unión de dos cardinalidades de dos entidades.

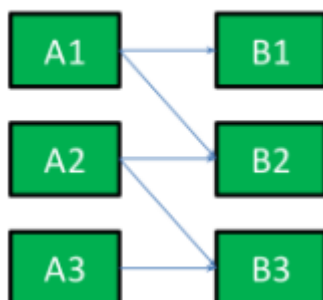
1:1. Uno a uno. Una entidad, solo puede tener otra entidad.

#### 1:1 Uno a uno



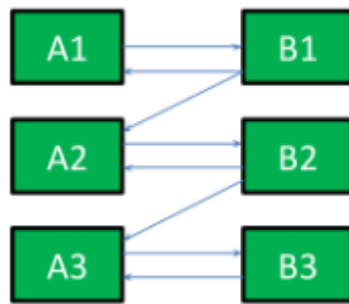
1:N Uno a muchos: Una entidad puede tener muchas entidades.

#### 1:N Uno a muchos



N:M Mucho a muchos

### N:M Muchos a muchos



### Diagrama Entidad – Relación



- ➔ **Ventajas:** Diseño de alto nivel, ya que refleja con bastante precisión el esquema conceptual. Los diagramas ER permiten mantener una visión global del diseño y favorecer la comunicación entre diseñadores.
- ➔ **Desventajas:** Falta de soporte formal, ya que los SGBD no lo implantan directamente. Casi siempre hay que transformarlo en un modelo de más bajo nivel, esto significa, más cercano a la comprensión del sistema – máquina:
  - **Un nivel intermedio:** El lenguaje SQL
  - **Un nivel alto:** el usuario lo puede entender más fácilmente pero la máquina que computa el código, no tanto.
  - **Un nivel bajo:** más cercano a la comprensión del sistema – máquina. Entiende bytes pero no los textos.

### Unified Modeling Language

Es un modelo internacional, que ayudaba al diseño e implementación de sistemas de software complejos. Cuando quieres hacer un flujo de bases de datos (como se hace un préstamo por parte de un usuario con entidades, Estudiantes-preéstamos-libros). Se crea para los diagramas que definen la estructura y el comportamiento.

El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. UML tiene aplicaciones más allá del desarrollo de software, p. ej., en el flujo de procesos en la fabricación.

Los diagramas UML describen los límites, la estructura y el comportamiento del sistema y los objetos que contiene.



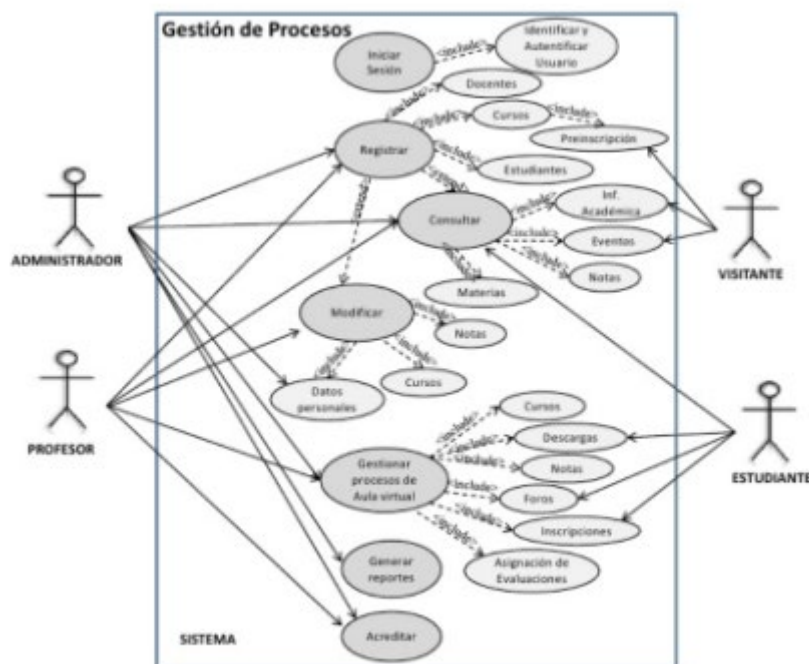
UML no es un lenguaje de programación, pero existen herramientas que se pueden usar para generar código en diversos lenguajes usando los diagramas UML.

Vamos a establecer información relacionada, se acerca a la base de datos, pero no va a hacer que yo pueda insertar el préstamo de un libro en ningún sitio concreto. El UML no es un lenguaje de base de datos pero es más avanzado que el Modelo Entidad-Relación.

\*Cuando hacemos el modelo Entidad Relación ponemos atributos, pero hemos establecido la relación que tenga la forma en la que un libro es leído por un alumno. Que sepas la parte del libro que ha sido leído.

### UML: Diagrama de casos de usos

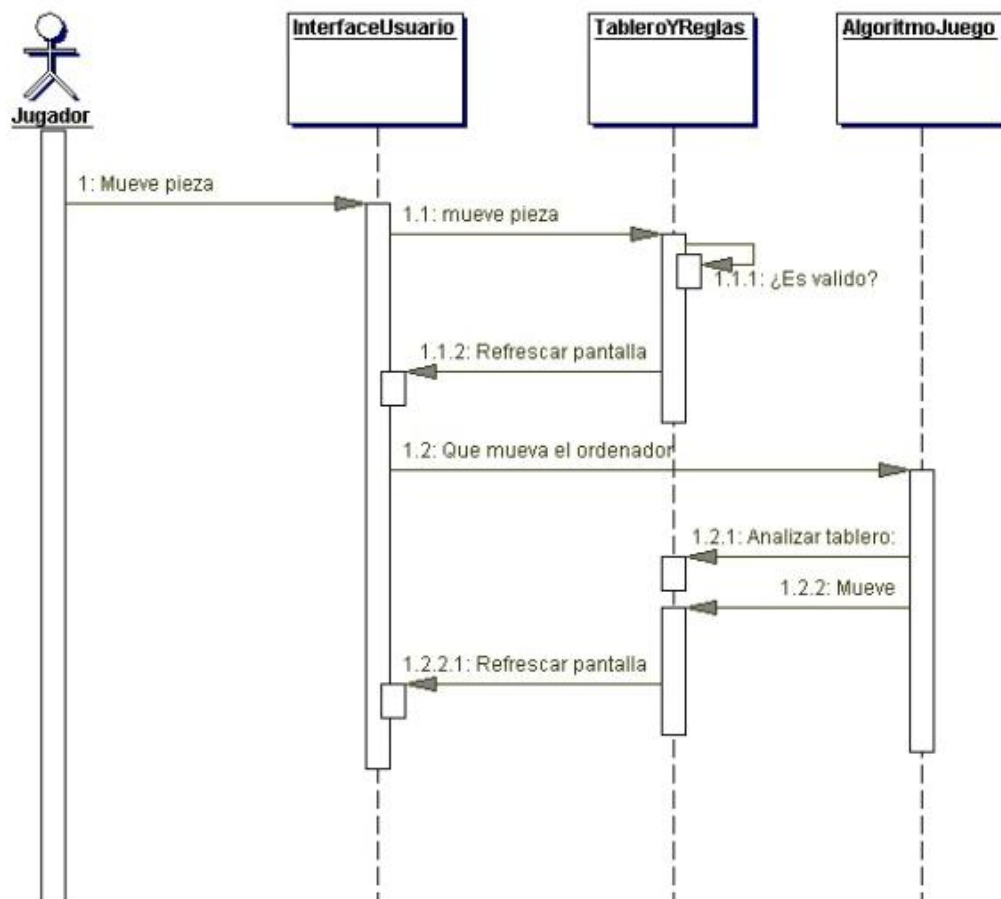
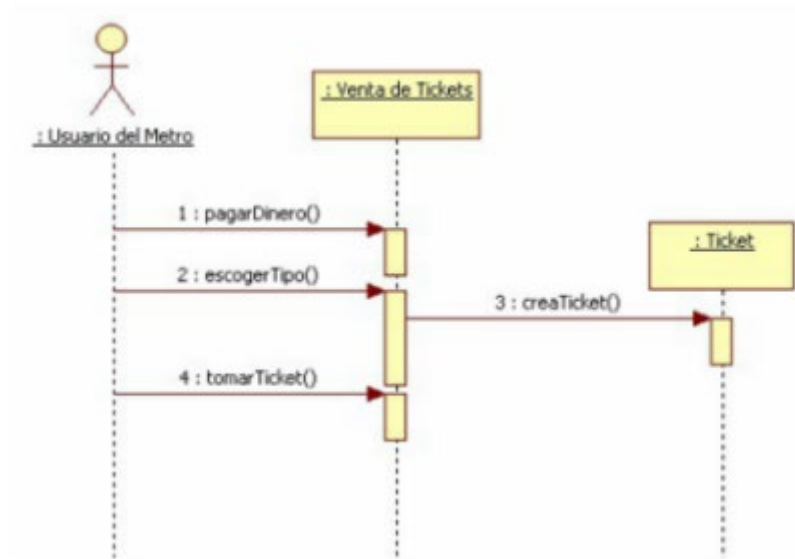
Representa gráficamente los casos de usos, interacción con el sistema.



### UML: Diagrama de clases

Muestra un conjunto de clases interfaces y sus relaciones





EJERCICIO: Diagrama de secuencia de un juego “Buscaminas”.

## 15.2 Concepto y origen de las Bases de Datos (DDBB) y de los Sistemas Gestores de Bases de Datos (DBMS). Evolución de la Base De Datos (BD).

Si trataremos los datos solo con ficheros, no tendríamos información relacionada.

\* En el mundo actual es fundamental el procesamiento de datos para obtener información que permita tomar decisiones, cada vez se genera y se utilizan más datos, lo cual dificulta su procesamiento.

Etapas de evolución del tratamiento de información:

- 1ª etapa, principios de lo 1960, primeras aplicaciones integran datos en programas

en forma de CONSTANTES, se analizan partes del mundo científico

- 2ª etapa, mediados 60s, aparecen los FICHEROS que representan partes del mundo

real, solo se accedían de forma secuencial y poco después de forma aleatoria.

- 3ª etapa, finales 60s, para evitar problemas en ficheros aparecen las BASES DE
- DATOS, que pretenden modelar grandes partes del mundo real mediante la relación de varios archivos, evitando la redundancia e incoherencias.

Una base de datos es “una colección de datos que están relacionados entre sí, que tienen una definición y una descripción comunes y que están estructurados de forma particular”

Los elementos que forman las bases de datos.

1. Los datos: representación de hechos, conceptos o instrucciones, estructurados, para comunicar, interpretar y representar datos de informacions almacenados y relacionados entre si:
  - Microdatos: datos simples, como por ejemplo nombres, direcciones,...
  - Macrodatos: aglomeración de datos, como las tablas.
2. El diccionario de datos (DD): conjunto de metadatos que contiene datos sobre los datos, que da lugar a una metabase, una BD que describe otra BD
  - Esquemas externos, conceptuales e internos
  - Descripciones de registros, campos y relaciones a cada nivel
  - Procedimientos de autorización de accesos y validación de datos
  - Correspondencias externas/conceptual y conceptual/interna
  - Referencias cruzadas
    - Un programa qué datos utiliza
    - Un datos y qué programas lo utilizan o lo modifican

//Es lo que describe la base de datos. Sus esquemas internos. Todos los campo y relaciones describen la base de datos.
3. **El sistema de gestión de la base de datos (SGBD):** Software o conjunto de programas que gestionan las BD.El software encargado de administrar y elaborar bases de datos.
  - Funciones: Crear, actualizar, consultar, proteger las BBDD
4. Los usuarios: todas las personas que utilizan las BD
  - Informáticos (especializados y comprenden la parte interna)
    - Administradores (ABD): gestionan la base de datos y proporcionan herramientas para el resto de usuarios

- Desarrolladores: complementan a los administradores, analizan y programan aplicaciones para usuarios finales.
- No informáticos: serán los tipos de usuarios a los que va dirigida la Base de Datos, utilizarán datos para el trabajo cotidiano y accederán los datos según sus permisos para facilitar el acceso y manipulación

Una cosa es la biblioteca de datos y otra cosa es la definición de estas bases de datos.