

DESARROLLO WEB EN ENTORNO SERVIDOR

DAW

César López - clopezs@iescomercio.com



¿Qué vamos a hacer esta semana?

Conocernos

Establecer las reglas del juego para este curso

Presentar la asignatura

Realizar evaluación inicial



Conocernos



Establecer las reglas del juego para este curso



Presentar la asignatura

DESARROLLO WEB EN ENTORNO SERVIDOR



¿DESARROLLO WEB?
¿ENTORNO SERVIDOR?



DESARROLLO WEB EN ENTORNO SERVIDOR

El módulo de Desarrollo Web en Entorno Servidor se encuadra dentro de las especificaciones del título de Técnico Superior en Desarrollo de Aplicaciones Web, integrado en la Familia Profesional de Informática y Comunicaciones, recogidas en el Real Decreto 686/2010, de 20 de mayo y en la Orden 21/2011, de 10 de octubre, y su actualización descrita en el Real Decreto 405/2023 e instrucciones para aplicarlo en la C.A. de La Rioja dadas en la resolución 26/2025 .

Este módulo equivale a 12 ECTS, unas 250 horas y lo abordamos en el aula razón de

8 horas semanales, unas 140 horas lectivas.



DESARROLLO WEB EN ENTORNO SERVIDOR:

Objetivos Generales, Resultados de Aprendizaje

- RA1. Selecciona las arquitecturas y tecnologías de programación Web en entorno servidor, analizando sus capacidades y características propias.
- RA2. Escribe sentencias ejecutables por un servidor Web reconociendo y aplicando procedimientos de integración del código en lenguajes de marcas.
- RA3. Escribe bloques de sentencias embebidos en lenguajes de marcas, seleccionando y utilizando las estructuras de programación.
- RA4. Desarrolla aplicaciones Web embebidas en lenguajes de marcas analizando e incorporando funcionalidades según especificaciones.



DESARROLLO WEB EN ENTORNO SERVIDOR: Objetivos Generales, Resultados de Aprendizaje

RA5. Desarrolla aplicaciones Web identificando y aplicando mecanismos para separar el código de presentación de la lógica de negocio.

RA6. Desarrolla aplicaciones de acceso a almacenes de datos, aplicando medidas para mantener la seguridad y la integridad de la información.

RA7. Desarrolla servicios Web analizando su funcionamiento e implantando la estructura de sus componentes.



DESARROLLO WEB EN ENTORNO SERVIDOR:

Objetivos Generales, Resultados de Aprendizaje

RA8. Genera páginas Web dinámicas analizando y utilizando tecnologías del servidor Web que añadan código al lenguaje de marcas.

RA9. Desarrolla aplicaciones Web híbridas seleccionando y utilizando librerías de código y repositorios heterogéneos de información.



Contenidos básicos y distribución temporal.

Unidad Didáctica	Trimestre	Horas	Resultados de Aprendizaje
UD1: Plataformas de programación web en entorno servidor. Aplicaciones LAMP.	1	5	RA1, RA2
UD2: Características del lenguaje PHP	1	20	RA2, RA3
UD3: Desarrollo de aplicaciones web con PHP	1	30	RA4
UD4: Trabajar con bases de datos desde PHP	1	10	RA5
UD5: Programación en capas MVC	1	10	RA6
UD6: Servicios Web	1	10	RA7
UD7: Aplicaciones web dinámicas: PHP y JavaScript	1	10	RA8
UD8: Aplicaciones web híbridas.	1	15	RA9
UD9: Proyectos integradores	2	70	RA1, RA2, RA3, RA4, RA5, RA6, RA7, RA8, RA9
TOTAL		180	



Contenidos actitudinales

Se destacan valores actitudinales importantes para el entorno profesional y en empresa como:

- **Trabajo en equipo**, fomentando el intercambio de opiniones entre los alumnos.
- **Delegación** en el alumno del **proceso cognitivo** de resolución de problemas.
- **Reflexión** del trabajo realizado mediante crítica/feedback reportado por el docente.
- **Reconocimiento del valor** del trabajo/software/producto desarrollado por terceros.
- **Adecuación** entre el esfuerzo desempeñado y el esfuerzo efectivo.
- **Interés** por la colaboración entre iguales y entre no iguales.
- **Exposición pública del trabajo individual** y sometimiento a la crítica colectiva



METODOLOGÍA

"Había que meterse todo aquello en la cabeza del modo que fuera, disfrutándolo o aborreciéndolo. Tamaña coerción produjo en mí un desaliento tan grande que, tras mi examen final, pasé un año entero sin encontrar el más mínimo placer en la consideración de ningún problema científico."

Albert Einstein



METODOLOGÍA CONECTIVISTA: APRENDIZAJE SIGNIFICATIVO Y CONSTRUCTIVISMO

El aprendizaje significativo es un tipo de aprendizaje en que un estudiante asocia la información nueva con la que ya posee; reajustando y reconstruyendo ambas informaciones en este proceso.

El Constructivismo promueve la exploración libre de un estudiante dentro de un marco o de una estructura dada. Se considera al alumno como centro de la enseñanza y como sujeto mentalmente activo en la adquisición del conocimiento, al tiempo que se toma como objetivo prioritario el potenciar sus capacidades de pensamiento y aprendizaje.



METODOLOGÍA CONECTIVISTA: APRENDIZAJE POR ACCIÓN Y DESCUBRIMIENTO

El aprendizaje por descubrimiento se produce cuando el docente le presenta todas las herramientas necesarias al alumno para que éste descubra por sí mismo lo que se desea aprender. Constituye un aprendizaje muy efectivo, pues cuando se lleva a cabo de modo idóneo, asegura un conocimiento más duradero en el tiempo y fomenta hábitos de investigación y rigor en los individuos.



METODOLOGÍA CONECTIVISTA ... en la era digital

El conectivismo es una teoría del aprendizaje para la era digital, que toma como base el análisis de las limitaciones del conductismo, el cognitivismo y el constructivismo, para aplicar el efecto que la tecnología tiene sobre el aprendizaje.

El aprendizaje debe ser humano, relevante, práctico, significativo; es algo que **el alumno debe hacer,** no algo que sea dado hecho por el docente.



RECURSOS

- Google Classroom para colgar apuntes y actividades propuestas en el aula.
- Google Suite para documentar, para proyectar soluciones u otros materiales..
- PC con Windows 10 + AMPPS + Entorno de Desarrollo (Notepad++, Sublimetext, Visual Studio Code, PHP Storm, ...) + VirtualBox o VMWare



EVALUACIÓN

Se emplearán distintos instrumentos para evaluar al alumno durante el curso académico.

- 1) Prácticas evaluables entendidas como exámenes prácticos durante los dos trimestres.
- 2) Exámenes teórico-prácticos al final del trimestre.



EVALUACIÓN

(*) En caso de haber más de un examen práctico se realizará la media aritmética.

(**) En la segunda evaluación, en caso de tener proyecto integrador sustituyendo exámenes teórico-prácticos, se considerará la nota obtenida en él como el 100% de la nota de dicha evaluación.

En caso de que en alguna evaluación no hubiera prácticas evaluables, el peso de los exámenes sería el 100% en dicha evaluación, manteniéndose invariable el peso respecto la nota final del módulo

Evaluación	Peso exámenes prácticos (*)	Peso examen teórico	Peso respecto nota final
1 ^a	75%	25%	70%
2 ^a (**)	75%	25%	30%



BIBLIOGRAFÍA Y WEBOGRAFÍA

- Desarrollo web en entorno servidor. Editorial Síntesis. Xabier Ganzábal García
- Desarrollo web en entorno servidor. Editorial Garceta. Juan Luis Vicente Carro.
- PHP 8.x
- Ammps - Xampp - Docker
- www.php.net
- www.laravel.com



Realizar autoevaluación inicial



UNIDADES DIDÁCTICAS DEL MÓDULO

UD1: Plataformas de programación web en entorno servidor. Aplicaciones LAMP

UD2: Características del lenguaje PHP

UD3: Desarrollo de aplicaciones web con PHP

UD4: Trabajar con bases de datos desde PHP

UD5: Programación en capas MVC

UD6: Servicios Web

UD7: Aplicaciones web dinámicas: PHP y JavaScript

UD8: Aplicaciones web híbridas (APIs).



Sesión I

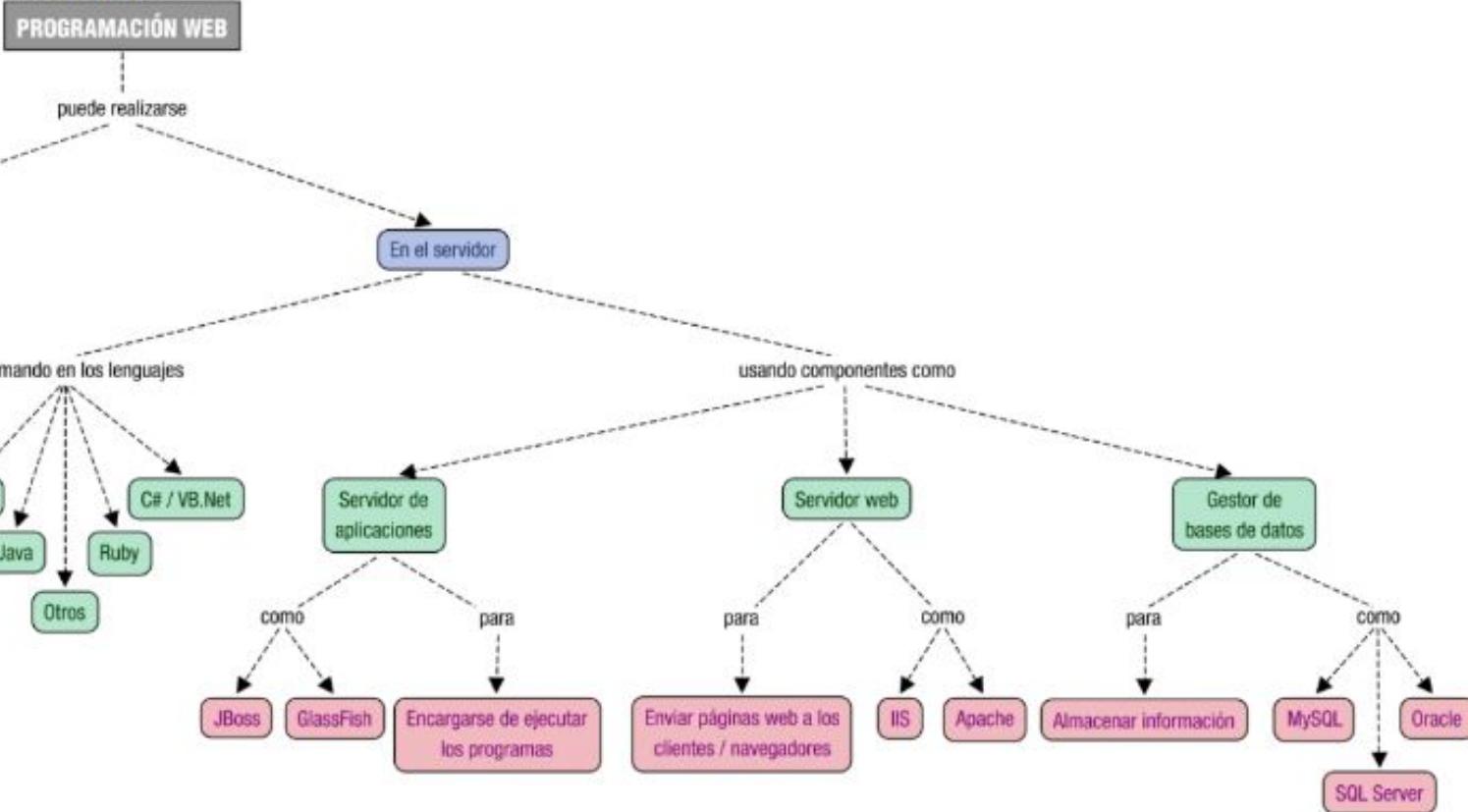


BLOQUE 1

UD1 (4h) - Plataformas de programación web en entorno servidor. Aplicaciones LAMP.



Tema 1 - Programación Web



UD 1 - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP.

Glosario

Aplicación web. Aplicación informática a la que se accede mediante una interfaz web utilizando un navegador.

Cliente. En el modelo cliente-servidor, los clientes solicitan funcionalidad a los servidores.

Framework. Un framework es una plataforma para el desarrollo de aplicaciones. Puede incluir librerías y metodologías.

HTML. *Hyper Text Markup Language*, el lenguaje básico para la creación de páginas web. Es un estándar del W3C.

HTTP. *Hyper Text Transfer Protocol*, protocolo de transferencia de hipertexto. Es el protocolo que utilizan clientes y servidores web para comunicarse. Es un estándar del W3C.

HTTPS. Versión segura del HTTP.

IDE. *Integrated Development Environment*, entorno de desarrollo integrado. Programa que integra herramientas útiles para programar, como editores, compiladores o control de versiones.

Protocolo. Según la RAE, conjunto de reglas que se establecen en el proceso de comunicación entre dos sistemas

Servidor. En el modelo cliente-servidor, los servidores proveen servicios a los clientes.

W3C. *World Wide Web Consortium*, organismo que elabora y mantiene varios de los estándares más importantes en Internet, como el HTTP o el HTML.



UD 1 - Plataformas de programación web en entorno servidor.

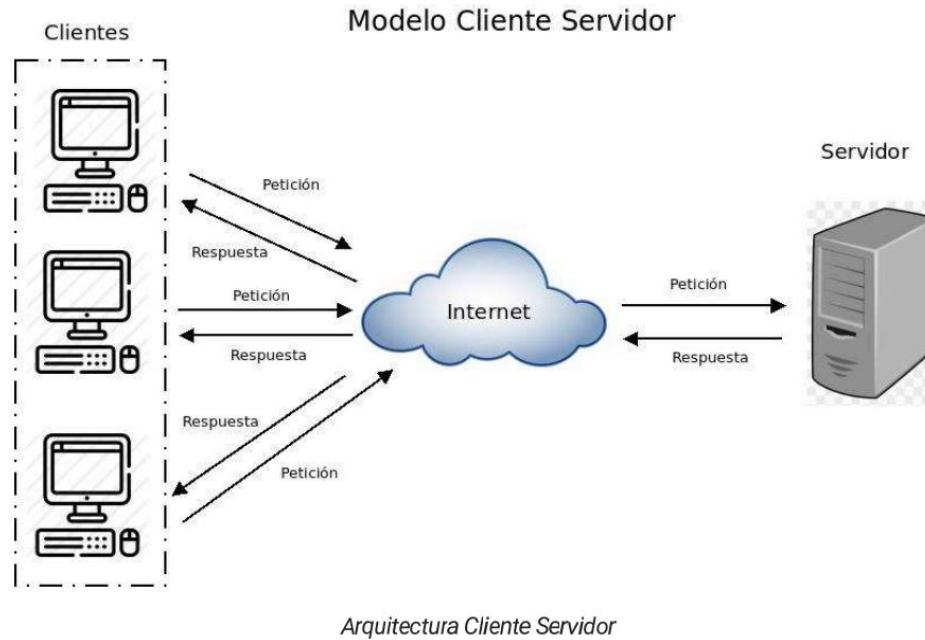
Aplicaciones LAMP.

- Modelos de programación: Cliente vs. Servidor. Diferencias (Apuntes)
- La generación dinámica de páginas web. Ventajas:
 - Mecanismos de ejecución de código en un servidor web
 - Lenguajes de programación web en entorno servidor
 - Integración con los lenguajes de marca
 - Herramientas de programación en entorno servidor: editores y compiladores
- Servidores de aplicaciones. Funcionalidades y uso.
- Integración con los servidores web.



UD 1 - Plataformas de programación web en entorno servidor.

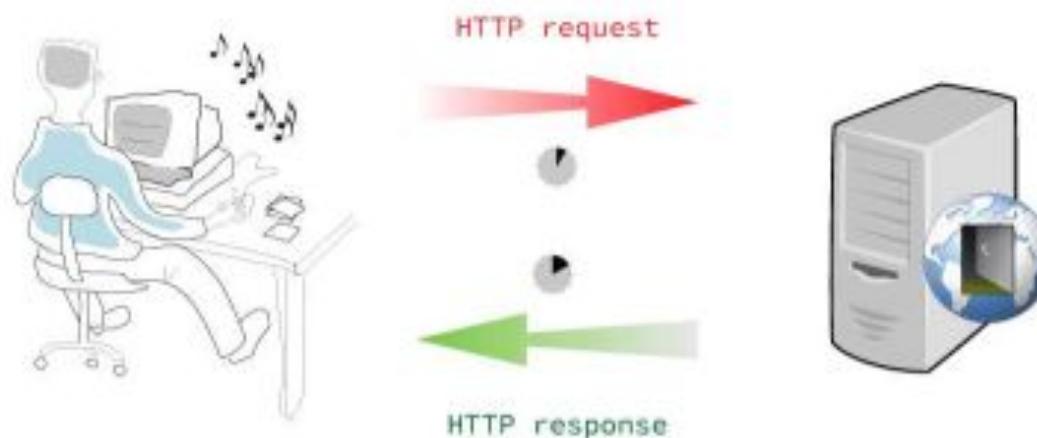
Aplicaciones LAMP.



UD 1 - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP.

Static Website

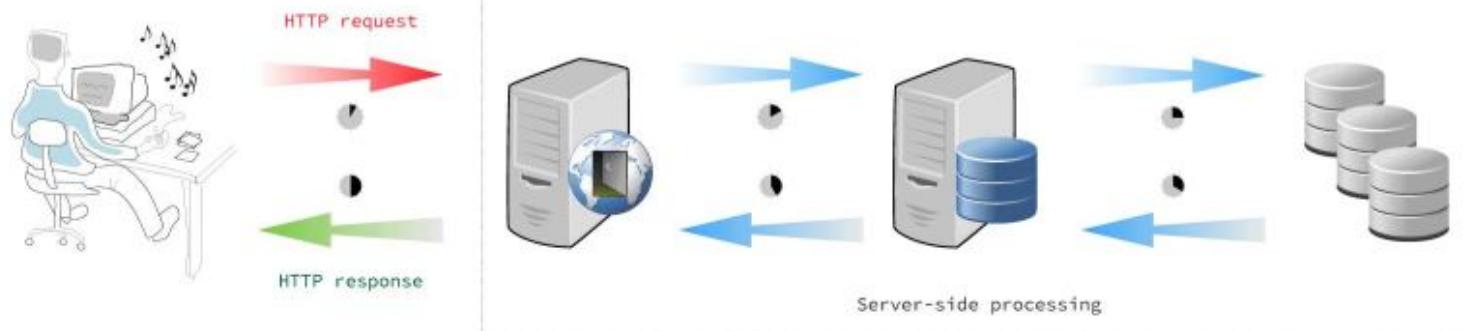


UD 1 - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP.

SCHEME 1

Dynamic Website



Página web dinámica



UD 1 - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP.

Perfil	Herramienta	Tecnología
<i>Front-end / cliente</i>	Navegador Web	HTML + CSS + JavaScript
<i>Back-end / servidor</i>	Servidor Web + BBDD	PHP, Python, Ruby, Java / JSP, .Net / .asp



UD 1 - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP: herramientas



UD 1 - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP.

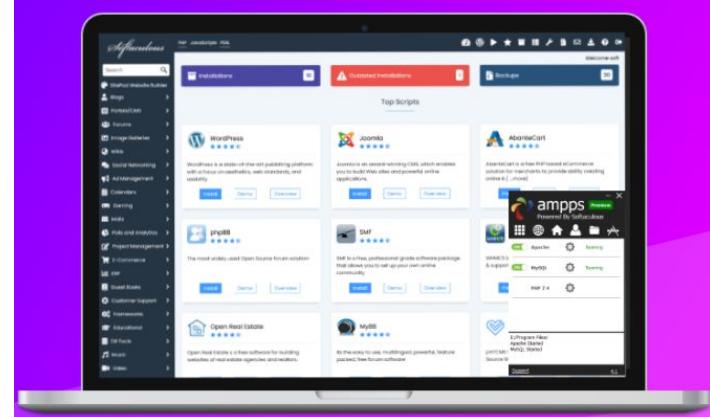
- **(AC) Descargar BoilerPlate**

<https://html5boilerplate.com/>

- Investiga el contenido de la plantilla
- Abre el índice, ¿Qué contiene?
- Modifica el índice para que la página muestre “Hola + TuNombre”

- **(AC) Descargar AMPPS**

- localiza la carpeta para los proyectos web
- copia la plantilla anterior
- visualizala con tu navegador



UD 1 - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP.

DESARROLLO WEB EN ENTORNO SERVIDOR

```
3      <head>
4          <title>Número aleatorio</title>
5      </head>
6      <body>
7          <?php
8              echo rand(0, 100);
9          ?>
10     </body>
11 </html>
```

HTML5 ★ BOILERPLATE

(AC) Crear página dinámica

- modifica el index.html anterior
- incluye el siguiente código php que genera un número aleatorio.
- accede desde tu cliente web varias veces a la página.
- ¿Qué ocurre?

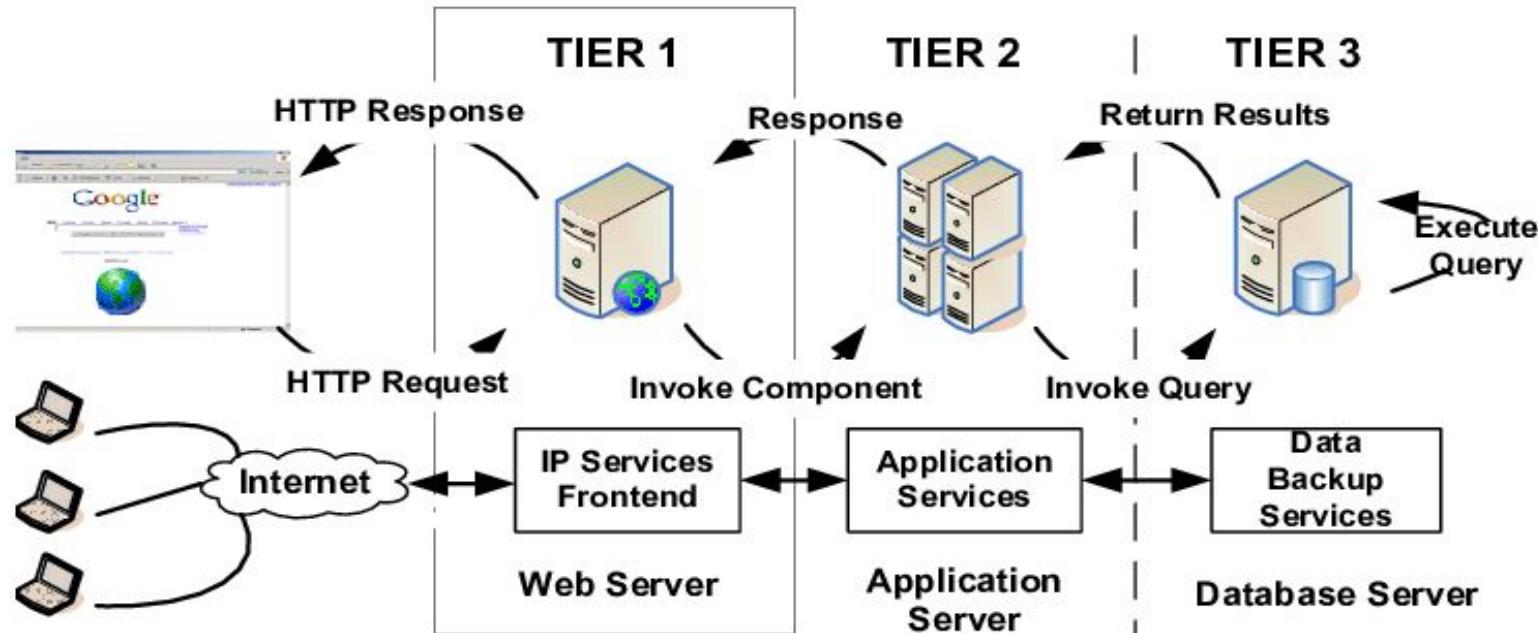


Sesión II



UD 1 - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP: Capas Físicas

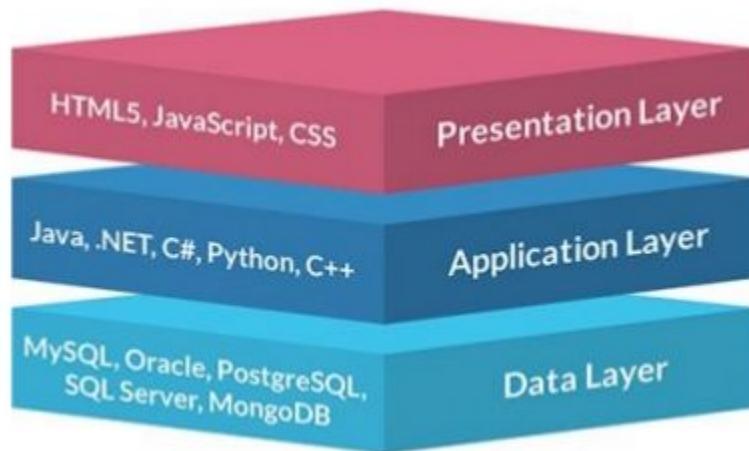


Arquitectura de tres capas físicas



UD 1 - Plataformas de programación web en entorno servidor.

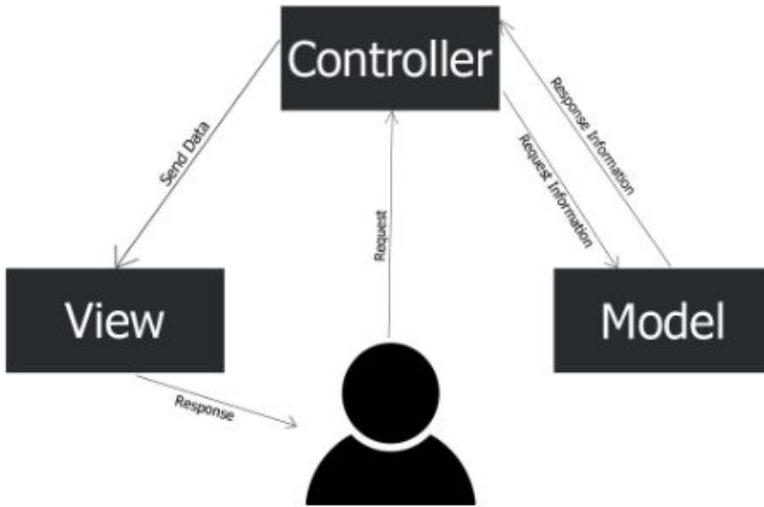
Aplicaciones LAMP: Capas Lógicas



UD 1 - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP: MVC

Model-View-Controller



Sistema monolítico

- Ningún patrón usado. Todas las capas se encuentran unidas en un único fichero. Útil en proyectos muy básicos

Vista / Controlador (dos capas)

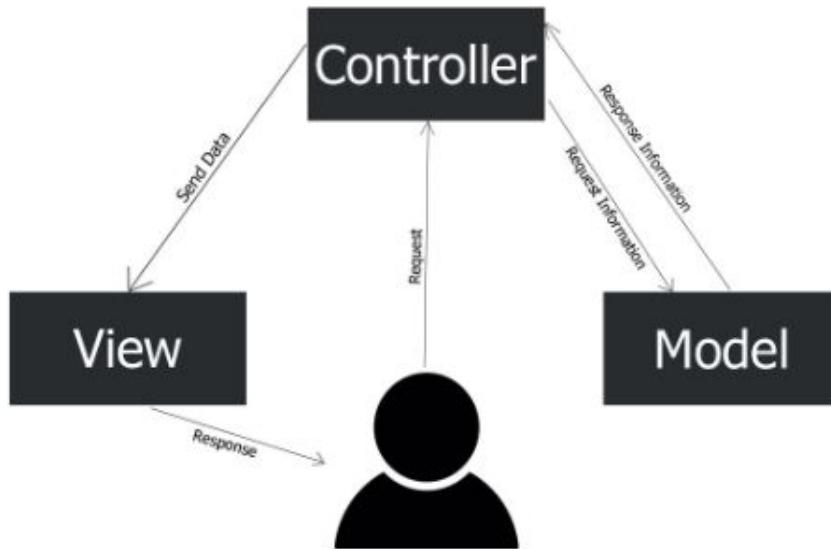
- Un **controlador** encargado de comunicarse con la BD y pasar el control a la vista (archivo *index.php*).
- Una **vista** encargada de presentar los datos recibidos del controlador (archivo *showAllArticles.php*).



UD 1 - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP: MVC

Model-View-Controller



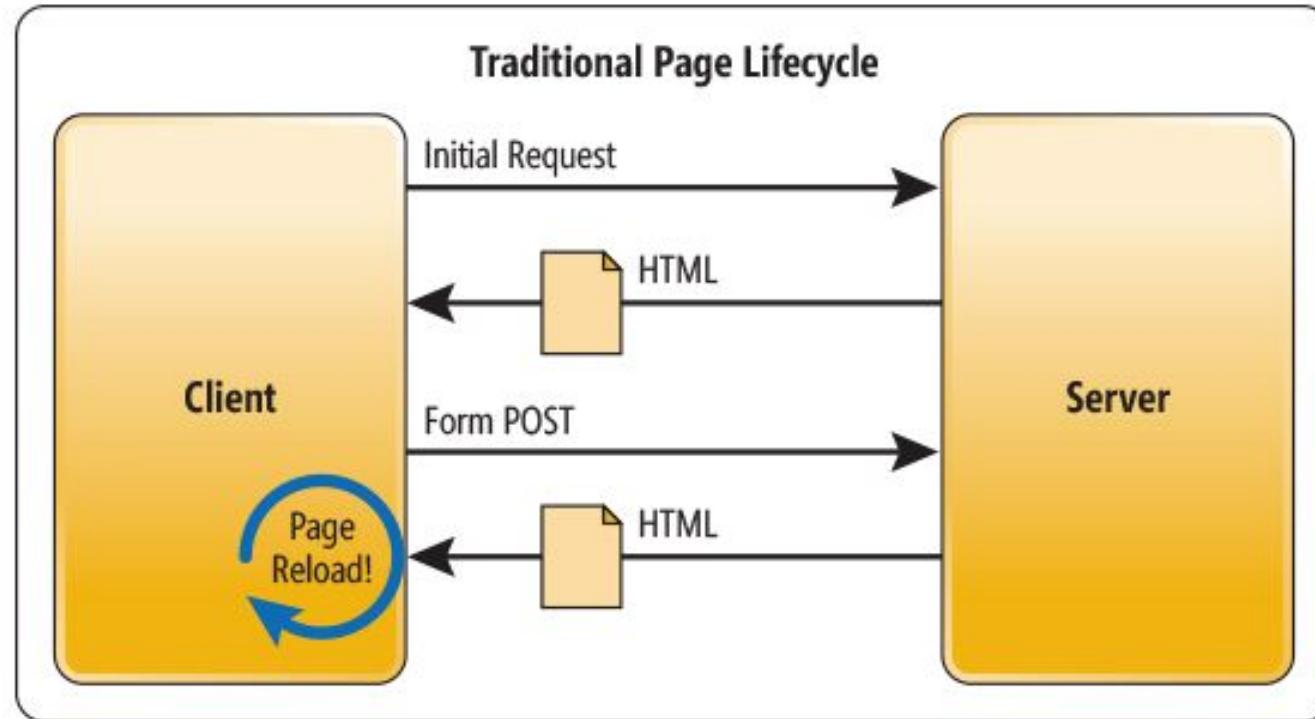
Modelo / Vista / Controlador

- Un **modelo**: se encargará de acceder a la base de datos y empaquetar el resultado de la consulta en un array (*articles.php*). Contendrá una clase con un método que.
- Una **vista**: idem al anterior ejemplo (*archivo showAllArticles.php*).
- Un **controlador**: comunica modelo y vista, creando una capa de seguridad intermedia (*index.php*)



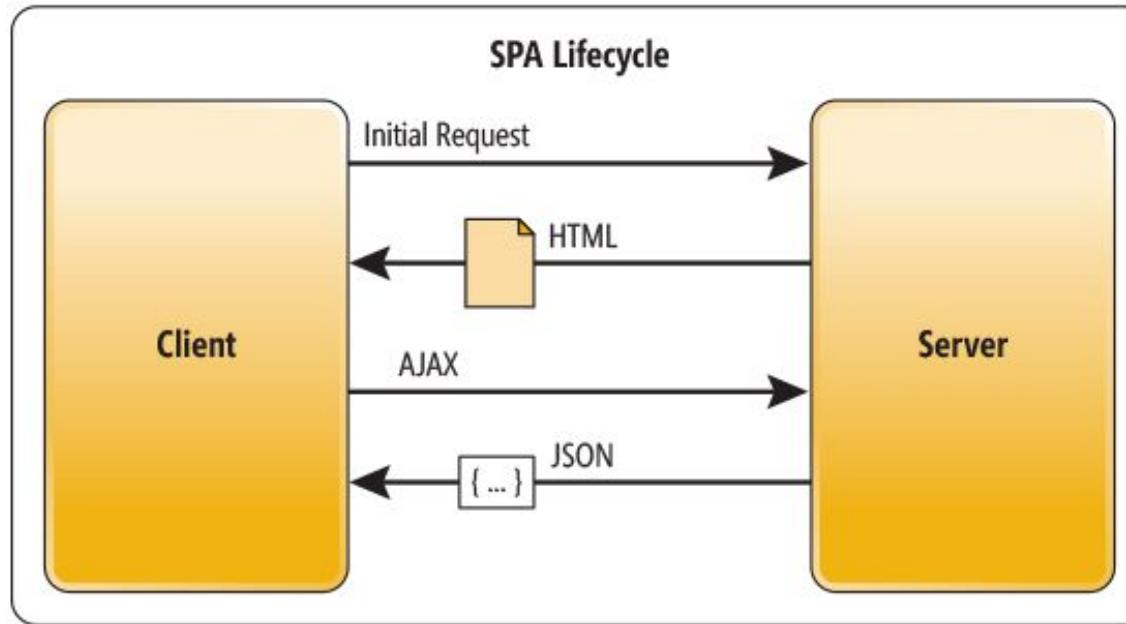
UD 1 - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP.



UD 1 - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP.



Arquitectura tradicional vs SPA



UD 1 (4h) - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP.

(AC) Descubriendo otras tecnologías para programación WEB

- Averigua qué caracteres se utilizan para embeber código JSP y ASP dentro HTML

(AC) PhpMyAdmin

- Investiga cómo se utiliza PhpMyadmin y crea bbdd, tablas y restricciones.



UD 1 (5h) - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP.

(AC) Instalar primer IDE: SublimeText o Notepad++

- Descarga e instala Sublimetext en tu equipo.
- Configúralo para trabajar con PHP.

● (AE) Web estáticas y dinámicas.



Sesión III



UD 1 (5h) - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP.

- (AE) Web estáticas y dinámicas.
 - Corrigiendo en clase
 - Diferenciar entre página estática y dinámica.



UD 1 (5h) - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP.

ACTIVIDADES DE AUTOEVALUACIÓN

1. El W3C:
 - a) Elabora servidores de aplicaciones.
 - b) Elabora y mantiene estándares de tecnologías web.
 - c) Elabora regulación de obligado cumplimiento para los desarrolladores.
2. El lenguaje básico para la elaboración de páginas web es:
 - a) HTTP.
 - b) HTML.
 - c) HTTPS.
3. En el modelo a tres capas, la lógica de la interfaz gráfica se sitúa en la capa de:
 - a) Presentación.
 - b) Datos.
 - c) Negocio.
4. En el modelo a tres capas no hay comunicación directa entre:
 - a) La capa de datos y la de negocio.
 - b) La capa de presentación y la de negocio.
 - c) La capa de datos y la de presentación.
5. En el modelo cliente-servidor, la comunicación la inicia:
 - a) El cliente.
 - b) El servidor.
 - c) Cualquiera de los dos.



UD 1 (5h) - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP.

6. ¿Cuál es el puerto asignado al protocolo HTTP?

- a) 80.
- b) 21.
- c) 8000.

7. ¿Cuál es el puerto asignado al protocolo HTTPS?

- a) 80.
- b) 553.
- c) 8000.

8. XAMPP incluye:

- a) Servidor web y PHP.
- b) Servidor web, PHP y base datos.
- c) Servidor de bases de datos y PHP.

9. ¿Cuál de los siguientes no es un estándar del W3C?

- a) HTML.
- b) CGI.
- c) HTTP.

10. Si se utiliza CGI, las peticiones se responden:

- a) Usando un módulo del servidor web.
- b) Usando un ejecutable en el servidor web.
- c) Usando un *servlet* en el servidor web.



UD 1 (5h) - Plataformas de programación web en entorno servidor.

Aplicaciones LAMP.

Resumen

- Las aplicaciones web siguen la arquitectura cliente-servidor.
- El modelo a 3 capas es una evolución del modelo cliente-servidor.
- El modelo a 3 capas separa la lógica de las aplicaciones en tres partes para conseguir código más reusable.
- Las capas son: presentación, negocio y datos.
- El W3C mantiene varios de los estándares más importantes para el desarrollo web.
- La comunicación entre clientes y servidores se hace con el protocolo HTTP.
- Se pueden generar páginas dinámicas usando lenguajes de programación.
- El lenguaje de programación más extendido en el lado del servidor es PHP.
- El código del lenguaje se inserta dentro del HTML. El servidor se encarga de ejecutarlo antes de enviar la respuesta.
- XAMPP instala Apache con PHP y MySQL.



BLOQUE 1

Comenzar con la UD2 (20h)

OBJETIVO PRINCIPAL: Conocer los fundamentos de PHP.



UD 2 - Características del lenguaje PHP

Glosario

Array. Es un tipo de dato compuesto habitual en los lenguajes de programación. Aunque los detalles varían entre lenguajes, en general, se parecen a vectores o listas ordenadas.

Bucle. Estructura de programación que permite repetir instrucciones.

Estructura condicional. Posibilita ejecutar o no una instrucción según se cumpla una condición.

Función. Conjunto de instrucciones que realiza una tarea concreta.

Librería. Las funciones relacionadas entre sí se agrupan en librerías o bibliotecas.

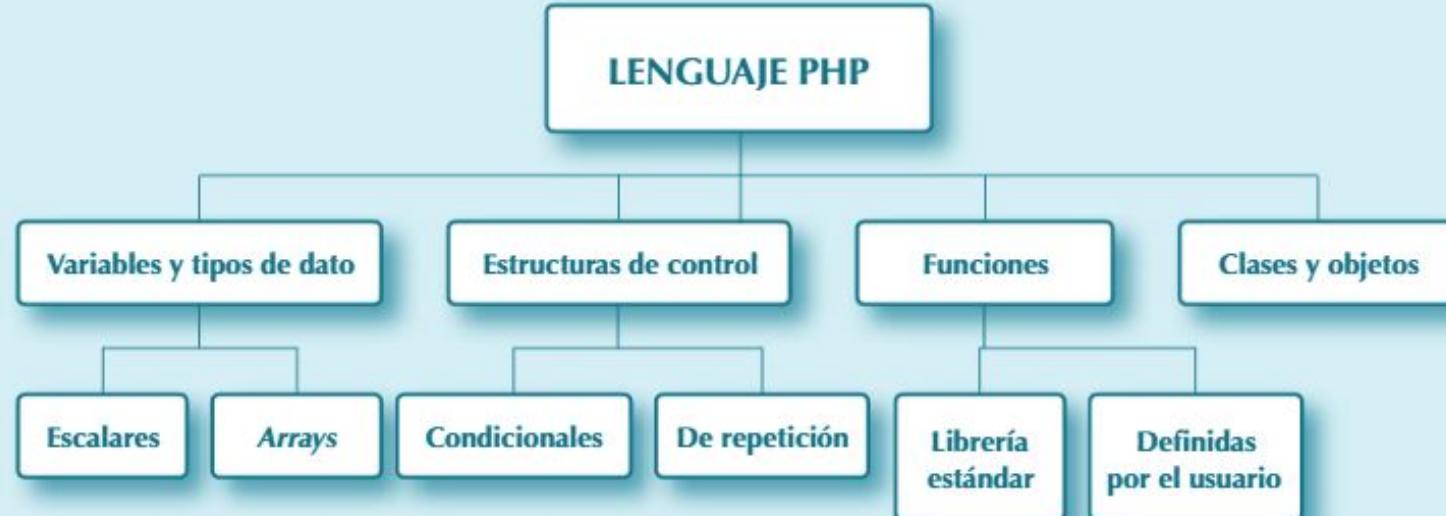
Programación orientada a objetos. Paradigma de programación basado en la idea de objetos, elementos que agrupan variables y funciones.

Script. Programa sencillo, habitualmente ejecutado por un intérprete en lugar de ser compilado.

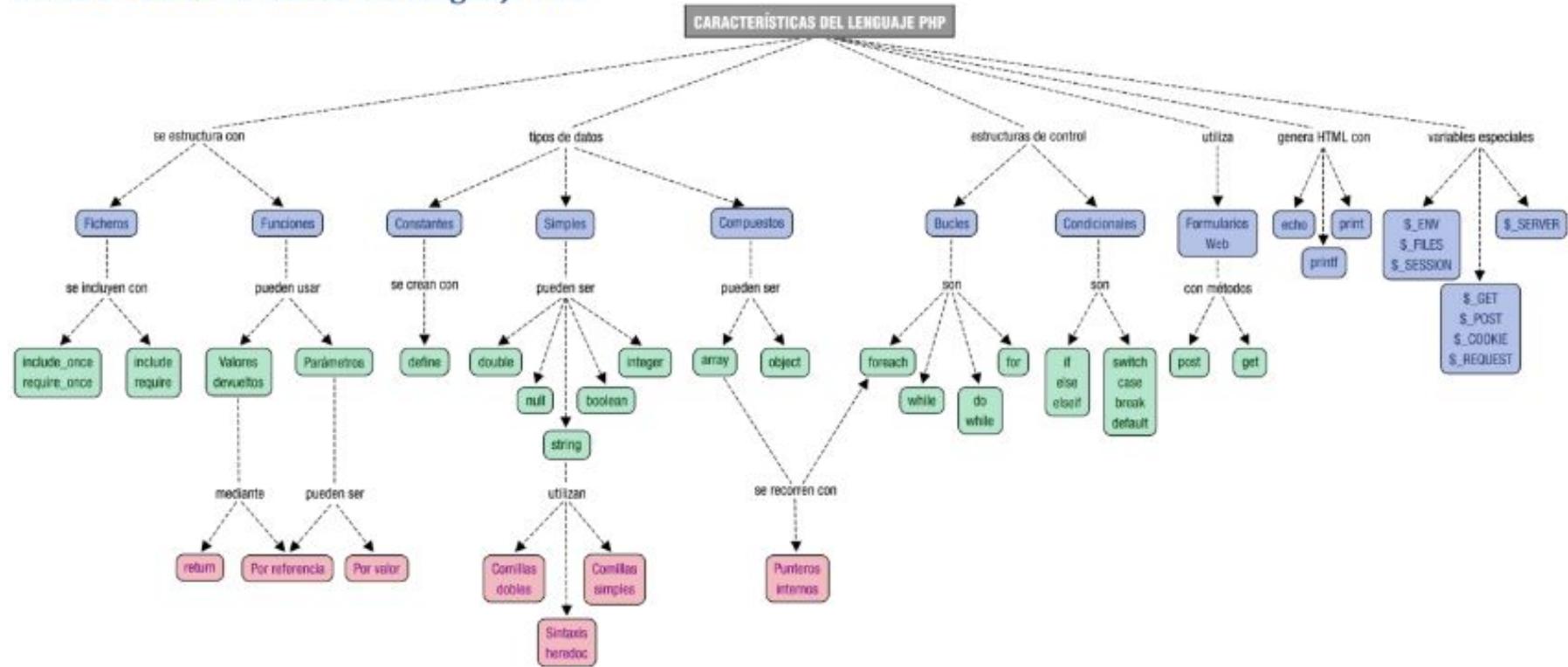
Variable. Posición en la memoria del ordenador identificada por un nombre. La variable almacena datos. Estos datos son el valor de la variable.



UD 2 - Características del lenguaje PHP



Tema 2 - Características del lenguaje PHP



UD 2 - Características del lenguaje PHP: la programación embebida

- Condiciones (tomas de decisión)
- Bucles
- Tipos de datos compuestos:
 - o arrays
 - o objetos
 - Clases y objetos
 - Interfaces
 - Herencia y Polimorfismo



UD 2 - Características del lenguaje PHP

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Hola mundo</title>
5      </head>
6      <body>
7          <?php
8              echo "Hola mundo";
9          ?>
10     </body>
11 </html>
```



UD 2 - Características del lenguaje PHP

```
1 <?php
2 /* declaración de variables */
3 $entero = 4; // tipo integer
4 $numero = 4.5; // tipo coma flotante
5 $cadena = "cadena"; // tipo cadena de caracteres
6 $bool = TRUE; //tipo booleano
7 /* cambio de tipo de una variable */
8 $a = 5; // entero
9 echo gettype($a); // imprime el tipo de dato de a
10 echo "<br>";
11 $a = "Hola"; // cambia a cadena
12 echo gettype($a); // se comprueba que ha cambiado
```



UD 2 - Características del lenguaje PHP

```
<?php
$var1 = 100;
$var2 = &$var1; // asignación por referencia

$var3 = $var1; // asignación por copia
echo "$var2<br>";// muestra 100
$var2 = 300; // cambia el valor de $var2
echo "$var1<br>";// $var1 también cambia
$var3 = 400; // este cambio no afecta a $var1
echo $var1;
```



UD 2 - Características del lenguaje PHP

```
1 <?php  
2 $var1 = 100;  
3 $var3 = 100 + $var2; // $var2 no existe, se toma como  
4 echo "$var3 <br>"; // muestra 100  
5 $var3 = 100 * $var2; // $var2 no existe, se toma como  
6 echo "$var3 <br>"; // muestra 0
```



Sesión IV

(W3Schools antes de Arrays)



UD 2 - Características del lenguaje PHP

```
<?php
    echo PHP_INT_SIZE.'<br>';
    echo PHP_INT_MIN.'<br>';
    echo PHP_INT_MAX.'<br>';
    $a = 0b100; // en binario
    $a = 0100; // octal
    $a = 0x100; // hexadecimal
    $a = 3/2; // la división entre enteros no da problemas
    echo $a.'<br>'; // 1.5
    $b = 7.5;
    $a = (int)$b; //casting a int
    echo $a.'<br>'; // 7, se trunca
    $b = 7e2; // notación científica
    $b = 7E2;
```



UD 2 - Características del lenguaje PHP



TOMA NOTA

A la hora de formatear la salida hay que tener en cuenta que esta va a ser procesada como HTML por un navegador web, es decir, los saltos de línea se ignoran y los espacios en blanco consecutivos colapsan. Los caracteres de escape como '\n', '\r' y '\t' son ignorados por el navegador. Para introducir un salto de línea la opción más sencilla es incluir la etiqueta de salto de línea de HTML ("
") en la salida, como se puede ver en los ejemplos anteriores.



UD 2 - Características del lenguaje PHP

CUADRO 2.1

Conversión implícita a *boolean*

Tipo	Valor como <i>boolean</i>
integer	Si es 0 se toma FALSE, en otro caso como TRUE
float	Si es 0.0 se toma FALSE, en otro caso como TRUE
string	Si es una cadena vacía o "0", se toma como FALSE, en otro caso como TRUE
variables no inicializadas	FALSE
null	FALSE
array	Si no tiene elementos se toma FALSE, en otro caso como TRUE



UD 2 - Características del lenguaje PHP

CUADRO 2.2
Tipos de dato en PHP

Tipo	Descripción
integer	Números enteros
float	Números reales en coma flotante
string	Cadenas de caracteres
boolean	Booleanos, TRUE o FALSE
array	Colección de elementos identificados
object	Un objeto es una instancia de una clase
callable	Para las funciones de <i>callback</i>
null	Para representar variables no asignadas
resource	Representa recursos externos



UD 2 - Características del lenguaje PHP

El *script* **global_server.php** muestra algunos de los datos disponibles.

```
<?php  
echo "Ruta dentro de htdocs: ". $_SERVER['PHP_SELF'];  
echo "Nombre del servidor: ". $_SERVER['SERVER_NAME'];  
echo "Software del servidor: ". $_SERVER['SERVER_SOFTWARE'];  
echo "Protocolo: ". $_SERVER['SERVER_PROTOCOL'];  
echo "Método de la petición: ". $_SERVER['REQUEST_METHOD'];
```



UD 2 - Características del lenguaje PHP

CUADRO 2.3

Variables *superglobales*

Nombre	Descripción
\$GLOBALS	Variables globales definidas en la aplicación
\$_SERVER	Información sobre el servidor
\$_GET	Parámetros enviados con el método GET (en la URL)
\$_POST	Parámetros enviados con el método POST (formularios)
\$_FILES	Ficheros subidos al servidor
\$_COOKIE	Cookies enviadas por el cliente
\$_SESSION	Información de sesión
\$_REQUEST	Contiene la información de \$_GET, \$_POST y \$_COOKIE
\$_ENV	Variables de entorno



UD 2 - Características del lenguaje PHP: condicionales.php

```
<?php
    $var = 3;
    if ($var == 1) {
        echo "Es un uno";
    }elseif ($var == 2) {
        echo "Es un dos";
    }elseif ($var == 3) {
        echo "Es un tres";
    }else{
        echo "No es un uno, ni un dos, ni un tres"
    }
```



UD 2 - Características del lenguaje PHP: switch.php

```
<?php
$var = 3;
switch($var){
    case 1:
        echo "Es un 1";
        break;
    case 2:
        echo "Es un 2";
        break;
    case 3:
        echo "Es un 3";
        break;
    default:
        echo "No es un 1, ni un 2, ni un 3";
}
```

RECUERDA

- ✓ Si no hay un *break* al final de un *case*, la ejecución continúa con el siguiente.



UD 2 - Características del lenguaje PHP: switch.php

```
<?php
    echo "Primer for anidado: <br>";
    for ($i = 0; $i < 3; $i++) {
        for ($j = 0; $j < 3; $j++) {
            echo "i: $i j: $j <br>";
            if ($j == 1) {
                break; //es lo mismo que poner break 1
            }
        }
    }
    echo "Segundo for anidado: <br>";
    for ($i = 0; $i < 3; $i++) {
        for ($j = 0; $j < 3; $j++) {
            echo "i: $i j: $j <br>";
            if ($j == 1){
                break 2;
            }
        }
    }
}
```



UD 2 - Características del lenguaje PHP: switch.php

Actividad propuesta 2.1



Escribe un programa que calcule el factorial de un número.

Recuerda que el factorial solo está definido para números enteros mayores o iguales que cero.



UD 2 - Características del lenguaje PHP: switch.php

El ejemplo **requerir.php** incluye otro fichero.

```
<?php  
    $a = "variable del principal";  
    require "ejerequerido.php";  
    $b = "otra variable del principal";  
    echo "En el script principal";
```

El fichero requerido es el siguiente:

```
<?php  
    echo "En el fichero requerido <br>";  
    echo $a;  
    echo $b;
```



UD 2 - Características del lenguaje PHP: switch.php

TOMA NOTA



La inclusión de ficheros tiene una diferencia con otros lenguajes. Supongamos que el fichero A incluye al fichero B. Las rutas relativas que aparezcan en B se interpretarán a partir del directorio de A.

Para solucionarlo, en el fichero B se utiliza `dirname(__FILE__)` que devuelve la ruta del fichero:

```
include( dirname(__FILE__)."\\".'fichero.php');
```



UD 2 - Características del lenguaje PHP: Operadores

PHP cuenta con los operadores habituales para operaciones aritméticas, lógicas, de manipulación de cadenas y demás.

Entre los operadores de comparación cabe señalar los operadores “**`==`**” y “**`!=`**”, llamados Idéntico y No Idéntico, que no están presentes en todos los lenguajes. El operador Idéntico se usa para comparar dos expresiones y se evalúa como verdadero cuando las dos expresiones tienen el mismo valor y además el mismo tipo de dato. Se diferencia del operador Igual, “**`=`**”, en que este, cuando las expresiones no tienen el mismo tipo de dato, intenta convertirlas antes de compararlas. El operador Idéntico es útil para evitar sorpresas inesperadas en la conversión de datos. En el ejemplo **`identico.php`** se puede comprobar la diferencia.



UD 2 - Características del lenguaje PHP: operadores.php

```
<?php
    $a = 3;
    $b = "3";
    if ($a == $b){
        echo "Son iguales <br>";
    }else{
        echo "No son iguales <br>";
    }
    if ($a === $b){
        echo "Son idénticos <br>";
    }else{
        echo "No son idénticos <br>";
    }
}
```



UD 2 - Características del lenguaje PHP: Operadores

Operadores de comparación

e1 === e2

Idéntico. Verdadero si las dos expresiones son del mismo tipo y tienen el mismo valor.

e1 == e2

Igual. Verdadero si las dos expresiones son iguales tras la conversión de tipos, si es necesaria.

e1 !== e2

No idéntico.

e1 != e2, e1 <> e2

No igual.

e1 >= e2, e1 > e2, e1 <= e2, e1 < e2

Mayor o igual, mayor, menor o igual, menor.

e1 ?? e2 ?? e3

Comenzando por la izquierda, devuelve la primera expresión no nula.



Sesión V

(Arrays - Funciones)



UD 2 - Características del lenguaje PHP: Arrays

Los *arrays* en PHP son una estructura muy flexible y potente. Unifica en un solo tipo lo que en otros lenguajes se consigue con *arrays* básicos, vectores, listas o diccionarios. Los elementos de un *array* se identifican por una clave, que puede ser un entero o una cadena. Los elementos guardan un orden dentro del *array*. Este orden está determinado por el orden de los elementos al declarar el *array* o al añadir nuevos.

Para recorrer un *array* lo habitual es utilizar el bucle **foreach**.

Si se pretende modificar el contenido del *array* al recorrerlo con un bucle **for**, es necesario utilizar una referencia al declarar las variables del bucle. En el ejemplo **foreach vs for** en [php](#), el primer bucle no utiliza referencias y, por tanto, no modifica el *array*. Esto es la forma correcta de hacerlo.



UD 2 - Características del lenguaje PHP: Arrays

```
<?php
    $arr1 = array(
        "Viernes" => 22,
        "Sábado" => 34
    );
    /* no modifica el array */
    foreach ($arr1 as $cantidad) {
        $cantidad = $cantidad * 2;
    }
    print_r($arr1);
    echo "<br>";
    /* modifica el array */
    foreach ($arr1 as &$cantidad) {
        $cantidad = $cantidad * 2;
    }
    print_r($arr1);
```



UD 2 - Características del lenguaje PHP: Arrays

Algunos operadores también están disponibles para los *arrays*, pero su significado no es exactamente el mismo. Por ejemplo, el operador “Idéntico” solo será verdadero si los dos *arrays* tienen todos los elementos iguales tanto en clave como en valor y además están en el mismo orden. Para el operador “Igual”, el orden de las claves no importa.

El operador “+” utilizado con dos *arrays* devuelve su unión. El *array* resultante contendrá primero los elementos del primer *array* (el que aparece a la izquierda del operador) y a continuación los del segundo, pero sin repetir claves. Si hay elementos con la misma clave en los dos *arrays*, el del segundo no se añade al resultado.



UD 2 - Características del lenguaje PHP: Arrays

CUADRO 2.5 Operadores para arrays

Operador	Descripción
<code>\$a1 === \$a2</code>	Idéntico. Verdadero si los dos arrays tienen las claves y valores iguales, en el mismo orden y del mismo tipo.
<code>\$a1 !== \$a2</code>	No idéntico.
<code>\$a1 == \$a2</code>	Igual. Verdadero si los dos arrays tienen las claves y valores iguales.
<code>\$a1 != \$a2, \$a1 <> \$a2</code>	No igual.
<code>\$a1 + \$a2</code>	Unión. Devuelve un array con los elementos de ambos.



Funciones



UD 2 - Características del lenguaje PHP: la programación embebida

- Programación modular
 - o Funciones. Paso de parámetros y devolución de valores
 - o Objetos. Instanciación y uso
- Comentarios
- La interacción con el usuario. Formularios:
 - o Recuperación de la información en los formularios
 - o Procesamiento de la información del formulario.



UD 2 - Características del lenguaje PHP: funciones de variables

```
<?php
$var1 = 4;
$var2 = NULL;
$var3 = FALSE;
$var4 = 0;
echo "var 1";
var_dump(isset($var1)); // TRUE
var_dump(is_null($var1)); // FALSE
var_dump(empty($var1)); // FALSE
echo "var 2";
var_dump(isset($var2)); // FALSE
var_dump(is_null($var2)); // TRUE
var_dump(empty($var2)); // TRUE
echo "var 3";
var_dump(isset($var3)); // TRUE
var_dump(is_null($var3)); // FALSE
var_dump(empty($var3)); // TRUE
echo "var 4";
var_dump(empty($var4)); // TRUE, EL 0 COMO BOOLEAN ES FALSE
echo "unset";
unset($var1);
var_dump(isset($var1)); // FALSE
```



UD 2 - Características del lenguaje PHP: funciones

Funciones de variables

`isset($var)` TRUE si la variable está inicializada y no es NULL

[Ir a la pág
siguiente](#)

`is_null($var)` TRUE si la variable es NULL

`empty($var)` TRUE si la variable no está inicializada o su valor es FALSE

`is_int($var), is_float($var),
is_bool($var), is_array($var)` Para comprobar el tipo de dato de \$var

`intval($var), floatval($var),
boolvar($var), strval($var)` Para obtener el valor de \$var como otro tipo de dato



UD 2 - Características del lenguaje PHP: funciones

Funciones de cadenas

strlen(\$cad)	Devuelve la longitud de \$cad
explode(\$cad, \$token)	Parte una cadena utilizando \$token como separador. Devuelve un array de cadenas
implode(\$token, \$array)	Crea una cadena larga a partir de un array de cadenas, entre cadena y cadena se introduce \$token
strcmp(\$cad1, \$cad2)	Compara las dos cadenas. Devuelve 0 si son iguales, -1 si \$cad1 es menor y 1 si \$cad1 es mayor
strtolower(\$cad), strtoupper(\$cad)	Devuelven \$cad en mayúsculas o minúsculas, respectivamente
str(\$cad1, \$cad2)	Busca la primera ocurrencia de \$cad2 en \$cad1. Si no aparece devuelve FALSE, si aparece devuelve \$cad1 desde donde comienza la ocurrencia

UD 2 - Características del lenguaje PHP: funciones

Funciones de arrays

ksort(\$arr), krsort(\$arr)	Ordena el <i>array</i> por clave en orden ascendente o descendente
sort(\$arr), rsort(\$arr)	Ordena el <i>array</i> por valor en orden ascendente o descendente
array_values(\$arr)	Devuelve los valores de \$arr
array_keys(\$arr)	Devuelve las claves de \$arr
array_key_exists(\$arr, \$cla)	Devuelve verdadero si algún elemento de \$arr tiene clave \$cla
count(\$arr)	Devuelve el número de elementos del <i>array</i>



UD 2 - Características del lenguaje PHP: funciones definidas por el usuario

```
<?php
    function suma($a, $b)  {
        return $a + $b;
    }
    echo suma(4,8). '<br>';
    $var1 = 35;
    $var2 = 5;
    $var3 = suma($var1, $var2);
    echo $var3.'<br>';
```

Es posible especificar valores por defecto para los argumentos. Si al llamar a la función no se usa el argumento, se toma el valor por defecto.

```
<?php
    function saludar($nombre = 'usuario'){
        echo "Hola $nombre <br>";
    }
    saludar();
    saludar("Ana");
```



UD 2 - Características del lenguaje PHP: funciones

Actividad propuesta 2.2



Escribe una función para calcular potencias. Recibirá como argumentos la base y el exponente, que es opcional y tiene valor por defecto 2 (elevar al cuadrado).



Sesión VI

STARTInnova

Repaso Express
Terminar funciones ...

W3School

AC2_EV_ArraysFunciones



UD 2 - Características del lenguaje PHP: la programación embebida

- Programación modular
 - o **Funciones. Paso de parámetros y devolución de valores**
 - o Manejo de errores.
 - o Objetos. Instanciación y uso
- La interacción con el usuario. Formularios:
 - o Recuperación de la información en los formularios
 - o Procesamiento de la información del formulario.



UD 2 - Características del lenguaje PHP: paso de argumentos por copia y valor

En PHP los argumentos se pasan por copia. Esto quiere decir que, cuando se llama a una función con una variable como argumento, se crea una variable local a la función en la que se copia el valor del argumento. Por tanto, si la función modifica el argumento, estos cambios no tienen efecto en la variable original.

Esto es lo que ocurre en la función `duplicarMal()` del siguiente ejemplo. El argumento se multiplica por dos, pero el valor de `$var1` no cambia. Para solucionarlo hay dos opciones. Devolver el nuevo valor y reasignar, como se hace con la función `duplicar()`, o utilizar una referencia, como se hace en `duplicar2()`. Para utilizar una referencia solo hace falta añadir el símbolo “&” antes del argumento.



UD 2 - Características del lenguaje PHP: paso de argumentos por copia y valor

```
<?php
    function duplicarMal($a){
        $a = $a *2;
    }
    function duplicar($a){
        return $a *2;
    }
    function duplicar2(&$a){
        $a = $a *2;
    }
    $var1 = 5;
    duplicarMal($var1);
    echo "$var1 <br>";
    $var1 = duplicar($var1);
    echo "$var1 <br>";
    duplicar2($var1);
    echo "$var1 <br>";
```



UD 2 - Características del lenguaje PHP: funciones como argumentos

```
<?php
    function calculador($operacion, $numa, $numb){
        $resul = $operacion($numa, $numb);

        return $resul;
    }
    function sumar($a, $b){
        return $a + $b;
    }
    function multiplicar($a, $b){
        return $a * $b;
    }
    $a = 4;
    $b = 5;
    $r1 = calculador("multiplicar", $a, $b);
    echo "$r1 <br>";
    $r2 = calculador("sumar", $a, $b);
    echo "$r2 <br>";
```

Actividad propuesta 2.3



Escribe una función para calcular el factorial de un número, que recibirá como argumento. Devolverá el factorial o -1 si el argumento no es válido.



Sesión VII

Repasar Arrays

Ejercicio básicos de Arrays

AC2_EV_IntroducciónPHP



UD 2 - Características del lenguaje PHP: funciones como argumentos

(AE) UD2_AC1_EV ArraysFunciones

- Lee el ejercicio planteado y analízalo.
- Crea tu solución en PHP.
- Envíalo vía aula virtual.



UD 2 - Características del lenguaje PHP: funciones como argumentos

<https://account.jetbrains.com/a/okghdorr>



Sesión VIII y IX

Realizar ejercicio en aula

AC2_EV_IntroducciónPHP



Sesión IX

Realizar ejercicio en aula

AC2_EV_IntroducciónPHP

Corregir ejercicio en Aula



Sesión X

Manejo de Excepciones

Clases y Objetos



UD 2 - Características del lenguaje PHP: Excepciones y errores

Código	Constante	Descripción
1	E_ERROR	Error fatal en tiempo de ejecución. La ejecución del <i>script</i> se detiene
2	E_WARNING	Advertencia en tiempo de ejecución. El <i>script</i> no se detiene
4	E_PARSE	Error de sintaxis al compilar
8	E_NOTICE	Notificación. Puede indicar error o no
16	E_CORE_ERROR	Error fatal al iniciar PHP
32	E_CORE_WARNING	Advertencia al iniciar PHP
64	E_COMPILE_ERROR	Error fatal al compilar
128	E_COMPILE_WARNING	Advertencia fatal al compilar
256	E_USER_ERROR	Error generado por el usuario
512	E_USER_WARNING	Advertencia generada por el usuario
1024	E_USER_NOTICE	Notificación generada por el usuario
2048	E_STRICT	Sugerencias para mejorar la portabilidad
4096	E_RECOVERABLE_ERROR	Error fatal capturable
8192	E_DEPRECATED	Advertencia de código obsoleto
16384	E_USER_DEPRECATED	Como la anterior, generada por el usuario
32767	E_ALL	Todos los errores



UD 2 - Características del lenguaje PHP: Excepciones y errores

(ver libro)

(AC) Modificar el fichero php.ini

- Localiza el fichero php.ini de tu servidor web.
- Configúralo para que los errores E_Notify no se muestren.
- Configúralo para que los errores se muestren en un archivo de LOGs
“./mislogs/”



UD 2 - Características del lenguaje PHP: Excepciones y errores

Este código te dará un Warning

```
<?php  
  
echo "<h1> UD2 - Gestión de Errores </h1>";  
$var1 = 5;  
$var3 = $var1 + $var2;  
  
echo "El valor de var3 es: $var3 <br>";  
|
```



UD 2 - Características del lenguaje PHP: Excepciones y errores

; Muestra y registra todos los errores en desarrollo:

error_reporting = E_ALL ; Reporta todos los errores

display_errors = On ; Muestra los errores por pantalla

log_errors = On ; Registra los errores en un fichero

error_log = "/var/log/php_errors.log" ; Ruta del log de errores

; Para PRODUCCIÓN deberías usar:

error_reporting = E_ALL ; Registra todos los errores

display_errors = Off ; No mostrar errores en la salida al usuario

log_errors = On ; Activar registro de errores

error_log = "/var/log/php_errors.log"



UD 2 - Características del lenguaje PHP: Excepciones y errores

```
; --- mis configuraciones ---  
error_reporting = E_ALL  
display_errors = On  
log_errors = Off  
error_log = "mislogs.txt"
```

Con esta configuración se mostrarán todos los errores en pantalla pero no se registran en archivo externo de LOGs

```
; --- mis configuraciones ---  
error_reporting = E_ALL &  
~E_WARNING  
display_errors = Off  
log_errors = On  
error_log = "mislogs.txt"
```

Con esta configuración se reportarán todos los errores menos los WARNINGs, NO se mostrarán en pantalla pero se registran en archivo externo de LOGs

Nota: si también quieres ocultar *notices* y *deprecated*, añade ~E_NOTICE, ~E_DEPRECATED, etc.



UD 2 - Características del lenguaje PHP: Excepciones y errores

<https://www.php.net/manual/es/ini.list.php>

Directivas de configuración de PHP en php.ini



UD 2 - Características del lenguaje PHP: error_reporting()

// Deshabilita todos los reportes de error
error_reporting(0);

// Reporta solo errores graves
error_reporting(E_ERROR | E_WARNING | E_PARSE);

// Reporta todos los errores excepto avisos (E_NOTICE)
error_reporting(E_ALL ^ E_NOTICE);

// Reporta absolutamente todos los errores posibles
error_reporting(E_ALL);
error_reporting(-1); // Equivalente, incluso ante futuros cambios



UD 2 - Características del lenguaje PHP: set_error_handler()

bool handler (int \$errno, string \$errstr [, string \$errfile [, int \$errline [, array \$errcontext]]]);

```
<?php
    function manejadorErrores($errno, $str, $file, $line){
        echo "Ocurrió el error: $errno";
    }
    set_error_handler("manejadorErrores");
    $a = $b; // causa error, $b no está inicializada
```

¿Te parece suficiente información este ejemplo para el uso de este
manejador de errores? ... busca información en ... php.net



UD 2 - Características del lenguaje PHP: set_error_handler()

¿Te parece suficiente información este ejemplo para el uso de este manejador de errores? ... busca información en ... php.net

¿Tienes claro cuando y porque utilizar este tipo de funciones? ...
¿Buscamos información sobre ello? ... vamos con la IA



UD 2 - Características del lenguaje PHP: set_error_handler()

```
<?php
    function dividir($a, $b){
        if ($b==0){
            throw new Exception('El segundo argumento es 0');
        }
        return $a/$b;
    }
    try{
        $resul1 = dividir(5, 0);
        echo "Resul 1 $resul1". "<br>";
    }catch(Exception e){
        echo "Excepción: ". $e->getMessage(). "<br>";
    }finally{
        echo "Primer finally";
    }
    try{
        $resul2 = dividir(5, 2);
        echo "Resul 2 $resul2". "<br>";
    }catch(Exception e){
        echo "Excepción: ". $e->getMessage(). "<br>";
    }finally{
        echo "Segundo finally";
    }
}
```



UD 2 - Características del lenguaje PHP: Excepciones

Actividad propuesta 2.3



Escribe una función para calcular el factorial de un número, que recibirá como argumento. Devolverá el factorial o -1 si el argumento no es válido.



Actividad propuesta 2.5

Adapta la actividad 2.3 para que controle si el argumento es negativo utilizando una excepción.



UD 2 - Características del lenguaje PHP: Excepciones Error predefinidas

Nombre	Descripción	Hereda de
Error	Clase base para las excepciones Error	
ArithmeticError	Error en operaciones matemáticas. Hereda del anterior	Error
DivisionByZeroError	Intento de división por cero. Hereda del anterior	ArithmeticError
AssertionError	Ocurre cuando falla una llamada a assert()	Error
ParseError	Error al compilar	Error
TypeError	Ocurre cuando una expresión no tiene el tipo de dato que se espera	Error
ArgumentCountError	Ocurre al llamar a una función con menos argumentos de los necesarios. Hereda del anterior	TypeError



Sesión X

POO - Programación Orientada a Objetos



UD 2 - Características del lenguaje PHP: POO - Clases y objetos

```
class Clase{  
    private $att1 = 10; // con valor por defecto  
    private $atr2; // sin valor por defecto  
    private static $atr3 = 0; // estático  
    ...  
}
```



Lenguaje PHP: Clases

```
<?php
class Persona {
    private $DNI;
    private $nombre;
    private $apellido;
    function __construct($DNI, $nombre, $apellido) {
        $this->DNI = $DNI;
        $this->nombre = $nombre;
        $this->apellido = $apellido;
    }
    public function getNombre() {
        return $this->nombre;
    }
    public function getApellido() {
        return $this->apellido;
    }
    public function setNombre($nombre) {
        $this->nombre = $nombre;
    }

    public function setApellido($apellido) {
        $this->apellido = $apellido;
    }
    public function __toString() {
        return "Persona: ".$this->nombre." ".$this->apellido;
    }
}
```



UD 2 - Características del lenguaje PHP: POO - Clases y objetos

```
class Cliente extends Persona{
    private $saldo = 0;
    function __construct($DNI, $nombre, $apellido, $saldo){
        parent::__construct($DNI, $nombre, $apellido);
        $this->$saldo = $saldo;
    }
    public function getSaldo(){
        return $this->saldo;
    }
    public function setSaldo($saldo){
        $this->saldo = $saldo;
    }
    public function __toString(){
        return "Cliente: ". $this->getNombre();
    }
}
```



UD 2 - Características del lenguaje PHP: POO - Clases y objetos

Resumen

- PHP es el lenguaje más extendido para el desarrollo web en el lado del servidor.
- Los ficheros de PHP mezclan HTML y PHP. El servidor sustituye los bloques de PHP por su salida.
- En PHP no hace falta declarar las variables, se declaran la primera vez que se usan.
- Tampoco es necesario definir su tipo de dato, se infiere del valor de inicialización.

- El paso de parámetros y la asignación se realizan por defecto por copia, pero se pueden usar referencias.
- Las funciones de PHP pueden ser parámetros de otras funciones. Es una característica muy usada en las librerías PHP.
- PHP cuenta con muchas funciones integradas para las tareas más habituales: manipulación de cadenas, *arrays*, bases de datos, ficheros...
- Los *arrays* de PHP son un tipo de dato muy potente similar a los que en otros lenguajes se llaman *mapas* o *diccionarios*.
- Hay tres sistemas de control de errores en PHP: errores, excepciones y excepciones de clase Error.
- PHP tiene soporte completo para la programación orientada a objetos.



UD 2 - Características del lenguaje PHP: POO - Clases y objetos

Hacer Test del libro de la Editorial Síntesis



UD 2 - Características del lenguaje PHP: POO - Herencia.

Las **clases**, **propiedades** y **métodos** se definen mediante los **modificadores** *public*, *protected*, *private*, *final* o *abstract* seguido de una declaración de variable normal:

1. Public: la **propiedad** o **método** podrá usarse en cualquier parte del script
2. Private: la **propiedad** o **método** sólo podrá usarse en la clase a la que pertenece
3. Protected: la **propiedad** o **método** se podrá usar por la clase a la que pertenece y por sus descendientes.
4. Final: la **clase** o **método** no puede ser sobreescrito en clases descendientes.
5. Abstract: la **clase** o **método** no puede ser usado directamente, ha de ser heredado primero para usarse.



UD 2 - Características del lenguaje PHP: POO - Abstracción de clases.

Las clases definidas como abstractas no se pueden instanciar.

Cualquier clase que contiene al menos un método abstracto debe ser definida como clase abstracta.

Los métodos definidos como abstractos simplemente declaran la firma del método, pero no pueden definir la implementación.

<https://www.php.net/manual/es/language.oop5.abstract.php>



UD 2 - Características del lenguaje PHP: POO - Abstracción de clases.

Cuando se hereda de una clase abstracta, todos los métodos definidos como abstractos en la declaración de la clase madre deben ser definidos en la clase hija

Además, estos métodos deben ser definidos con la misma visibilidad (o con una menos restrictiva)

<https://www.php.net/manual/es/language.oop5.abstract.php>



UD 2 - Características del lenguaje PHP: POO - Abstracción de clases.

Una clase abstracta A puede ser extendida por una clase abstracta B.

Y esta última, B, puede implementar o no los métodos abstractos de su antecesora A.

Si no fuera abstracta, A, si que estaría obligada, B, a implementar los métodos.

<https://www.php.net/manual/es/language.oop5.abstract.php>



UD 2 - Características del lenguaje PHP: POO - Interfaces de objetos

- Definen los métodos que deben ser implementados por una clase.
- Se declara con la palabra “interface” y los métodos sin contenido.
- Los métodos deben ser **PÚBLICOS**
- Utilizar “implements” para implementar la interfaz.



UD 2 - Características del lenguaje PHP: interfaces de objetos

- Las interfaces se pueden extender con “extends”
- Las constantes de las interfaces NO pueden ser re-escritas.
- Se pueden heredar múltiples interfaces. (NO clases)

<https://www.php.net/manual/es/language.oop5.interfaces.php>



UD 2 - Características del lenguaje PHP: interfaces vs clases abstractas

- *Las interfaces **NO** pueden tener propiedades, mientras que las clases abstractas pueden.
- Todos los métodos de interfaz deben ser públicos, mientras que los métodos de clase abstracta son públicos o protegidos.

[ver documento sobre interfaces](#)



UD 2 - Características del lenguaje PHP: interfaces vs clases abstractas

- Todos los métodos en una interfaz son abstractos, por lo que no se pueden implementar en el código y la palabra clave abstracta no es necesaria.
- Las clases pueden implementar una interfaz mientras heredan de otra clase al mismo tiempo.

[ver documento sobre interfaces](#)



UD 2 - Características del lenguaje PHP: interfaces vs clases abstractas

Característica	Interfaz	Clase Abstracta
Herencia	Múltiple	Simple
Métodos	Solo declaración	Puede contener implementación parcial
Propiedades	*No permitidas	Permitidas
Uso típico	Definir contrato	Definir comportamiento base



Sesión XI

Continuación POO



Sesión XII, XIII y XIV
*Realizar ejercicio en aula
AC3_TrabajandoConClases



UD 2 - Características del lenguaje PHP: traits

Los traits son un mecanismo de reutilización de código en un lenguaje de herencia simple como PHP.

Un trait es similar a una clase, pero solo sirve para agrupar funcionalidades de una manera interesante.

No es posible instanciar un Trait en sí mismo. Es un añadido a la herencia tradicional, que permite la composición horizontal de comportamientos.

<https://www.php.net/manual/es/language.oop5.traits.php>



UD 2 - Características del lenguaje PHP: traits

Un trait es similar a una clase, pero solo tiene el propósito de agrupar funcionalidad de una manera fácilmente reutilizable en varias clases independientes. Ayuda a suplir la carencia de herencia múltiple.

```
trait Loggable {
    public function log($message) {
        echo "[LOG] {$message}\n";
    }
}
```

```
class Product {
    use Loggable;

    public function create() {
        // ... lógica para crear un producto ...
        $this->log("Producto creado");
    }
}
```

<https://www.php.net/manual/en/language.oop5.traits.php>



UD 2 - Características del lenguaje PHP: ventajas

- Reutilización de código: puedes reutilizar un conjunto de funcionalidades en múltiples clases sin necesidad de heredar de una clase común.
- Componer comportamientos: puedes incorporar múltiples traits en una clase para componer sus comportamientos.
- Si dos traits tienen un método con el mismo nombre, PHP generará un error fatal para evitar confusiones. Sin embargo, puedes resolver estos conflictos en la clase que usa los traits

<https://www.php.net/manual/en/language.oop5.traits.php>



UD 2 - Características del lenguaje PHP: ventajas

```
trait A {
    public function test() { echo "Trait A"; }
}

trait B {
    public function test() { echo "Trait B"; }
}

class MyClass {
    use A, B {
        A::test insteadof B; // Esto usará `test` de A
    }
}
```

<https://www.php.net/manual/en/language.oop5.traits.php>



UD 2 - Características del lenguaje PHP: Sobrecarga mágica

- La sobrecarga mágica en PHP permite "crear" dinámicamente propiedades y métodos. Estas entidades dinámicas son tratadas a través de métodos mágicos establecidos que se pueden posicionar en una clase para diversos tipos de acciones.
- Los métodos mágicos de sobrecarga son llamados durante la interacción con propiedades o métodos que no han sido declarados o no son **visibles** en el contexto actual. El resto de esta sección utiliza los términos de propiedades inaccesibles y de métodos inaccesibles para referirse a esta combinación de declaración y visibilidad.

<https://www.php.net/manual/es/language.oop5.overloading.php>



UD 2 - Características del lenguaje PHP: Métodos Mágicos

- Los métodos mágicos son métodos especiales que sobreescriben acciones por defecto cuando se realizan ciertas acciones sobre un objeto.
- `__construct()`, `__destruct()`, `__call()`, `__callStatic()`, `__get()`, `__set()`, `__isset()`, `__unset()`, `__sleep()`, `__wakeup()`, `__serialize()`, `__unserialize()`, `__toString()`, `__invoke()`, `__set_state()`, `__clone()`, y `__debugInfo()`.

<https://www.php.net/manual/es/language.oop5.magic.php>



Sesión XII, XIII y XIV
Realizar ejercicio en aula
AC3_TrabajandoConClases



UD 2 - Características del lenguaje PHP: Clases anónimas

Definición: "Las clases anónimas, introducidas en PHP 7, permiten la creación de objetos sin necesidad de definir una clase con nombre.

```
<?php  
  
$greeting = new class {  
    public $message = "¡Hola, mundo!";  
  
    public function sayHello() {  
        return $this->message;  
    }  
};  
  
echo $greeting->sayHello(); // Esto imprimirá: ¡Hola, mundo!  
?>
```



UD 2 - Características del lenguaje PHP: Clases anónimas - VENTAJAS

- Reducción de la sobrecarga de clases.
- Código más legible y conciso.
- Rapidez en la prototipación.
- Uso en patrones de diseño y pruebas.
- Encapsulación temporal.
- Menor contaminación del espacio de nombres.



UD 2 - Características del lenguaje PHP: Clases anónimas - VENTAJAS

- No reutilizables.
- Dificultad en la depuración.
- Potenciales desafíos de mantenimiento.
- Limitaciones en extensión e implementación.
- Rendimiento e incompatibilidad con versiones antiguas.
- Comprensión del código por terceros.



UD 2 - Características del lenguaje PHP: namespaces

En el mundo de PHP, los espacios de nombres están diseñados para resolver dos problemas que enfrentan los autores de bibliotecas y aplicaciones al reutilizar elementos como clases o bibliotecas de funciones:

1. Colisiones de nombres entre el código que se crea, las clases, funciones o constantes internas de PHP, o las de bibliotecas de terceros.
2. La capacidad de crear alias o acortar nombres como Nombres_Extreadamente_Largos para ayudar a resolver el primer problema y mejorar la legibilidad del código.

Los espacios de nombres de PHP proporcionan un medio para agrupar clases, interfaces, funciones o constantes.

<https://www.php.net/manual/en/language.namespaces.definition.php>

UD 2 - Características del lenguaje PHP: strptime() DEPRECATED

- Cuando detectamos un método obsoleto heredado de algún código fuente, acudir a php.net para localizar la nueva solución propuesta y actualizar el código antiguo.
- <https://www.php.net/manual/en/function.strptime.php>

Changelog

Version	Description
8.1.0	This function has been deprecated. Use date_parse_from_format() instead (for locale-independent parsing), or IntlDateFormatter::parse() (for locale-dependent parsing)

Sesión



PASE DE PARÁMETROS GET Y POST



UD 2 - Características del lenguaje PHP: comunicación cliente-servidor (GET)

```
<?php  
echo "Hola ". $_GET[ "nombre" ];
```



UD 2 - Características del lenguaje PHP: paso de parámetros

Actividad propuesta 3.1



Escribe un fichero que reciba dos parámetros, num1 y num2, y muestre su suma. Hay que comprobar que los dos argumentos existan y sean números.



FORMULARIOS



UD 2 - Características del lenguaje PHP: formularios (POST)

```
<!DOCTYPE html>
<html>
    <head>
        <title>Formulario de login</title>
        <meta charset = "UTF-8">
    </head>
    <body>
        <form action = "login_basico.php" method = "POST">
            <input name = "usuario" type = "text">
            <input name = "clave" type = "password">
            <input type = "submit">
        </form>
    </body>
</html>
```



Sesión



UD 2 - Características del lenguaje PHP: formularios - header

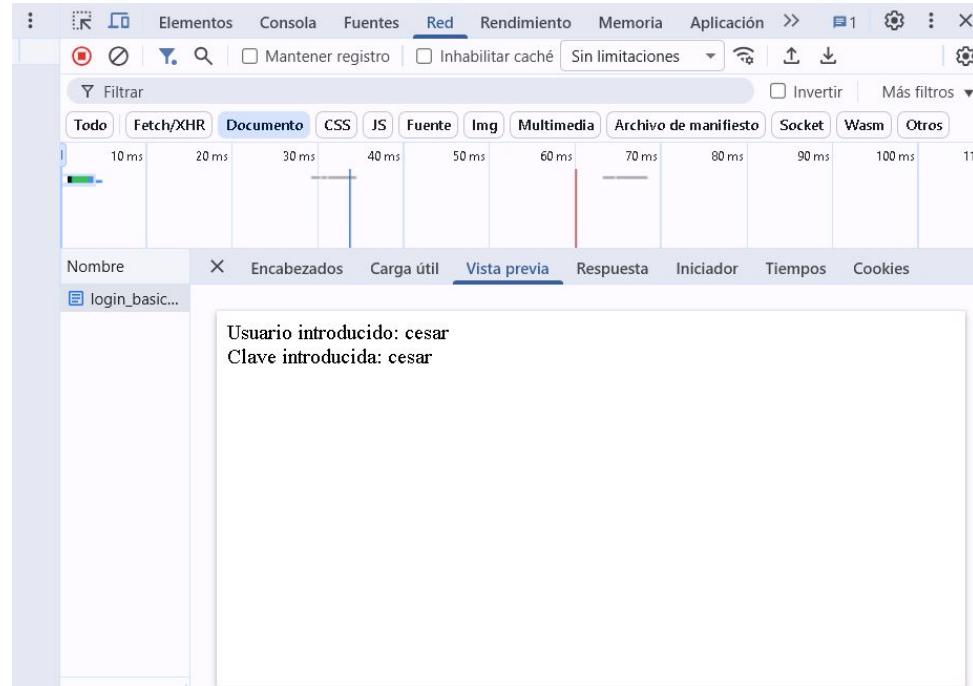
```
<?php
/*si va bien redirige a bienvenido.html
si va mal, mensaje de error */
if ($_POST['usuario']=="usuario" and $_POST["clave"]=="1234") {
    header("Location:bienvenido.html");
} else {
    header("Location:error.html");
}
```

RECUERDA

- ✓ Hay que enviar las cabeceras antes de empezar con el cuerpo de la respuesta. Esto implica que hay que utilizar la función header() antes de que se empiece a escribir la salida. Si se intenta llamar a header() después de haber realizado un echo, se producirá un error.



UD 2 - Características del lenguaje PHP: formularios - navegador - developer tools



UD 2 - Características del lenguaje PHP: formularios - form_en_uno.php

```
<?php
/* si va bien redirige a principal.php si va mal, mensaje de error */
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if($_POST['usuario']=="usuario" and $_POST["clave"]=="1234"){
        header("Location: principal.php");
    }else{
        $err = true;
    }
}
?>
<!DOCTYPE html>
<html>
```



UD 2 - Características del lenguaje PHP: formularios - form_en_uno.php

```
<head>
    <title>Formulario de login</title>
    <meta charset = "UTF-8">
</head>
<body>
    <?php if(isset($err)){
        echo "<p> Revise usuario y contraseña</p>";
    }?>
    <form method = "POST"
        action=<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>>
        <label for = "usuario">Usuario</label>
        <input value = "<?php if(isset($usuario))echo $usuario;?>">
        id = "usuario" name = "usuario" type = "text">
        <label for = "clave">Clave</label>
        <input id = "clave" name = "clave" type = "password">
        <input type = "submit">
    </form>
</body>
</html>
```



UD 2 - Características del lenguaje PHP: formularios - formularios todo en uno ...

Cuando el formulario llama al mismo fichero se recomienda usar:

`action = "<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>"`

en lugar del nombre del fichero como aparece en el ejemplo. La variable *superglobal* `$_SERVER["PHP_SELF"]` contiene el nombre del fichero y la función `htmlspecialchars()` sirve para filtrar los caracteres por seguridad.



UD 2 - Características del lenguaje PHP: formularios

Métodos GET y POST:

- **GET:** Los datos del formulario se anexan a la URL en el campo de dirección del navegador. No es seguro para datos sensibles.
- **POST:** Los datos del formulario se envían en el cuerpo de la solicitud HTTP. Es más seguro que GET y no tiene restricciones en la cantidad de datos.



FORMULARIOS SUBIDA DE FICHEROS



UD 2 - Características del lenguaje PHP: formularios

enctype =multipart/formdata

```
<!DOCTYPE html>
<html>
    <body>
        <form action="procesar_subida.php" method="post"
              enctype="multipart/form-data">
            Escoja un fichero
            <input type="file" name="fichero">
            <input type="submit" value="Subir fichero">
        </form>
    </body>
</html>
```



UD 2 - Características del lenguaje PHP: formularios

En el *script* que recibe el formulario, la variable global `$_FILES` contiene información sobre el fichero que se está subiendo. Se trata de un *array* bidimensional. La primera dimensión identifica el fichero según el atributo `name` en el formulario, en la segunda las claves son:

- `name`: el nombre del fichero en el cliente.
- `size`: el tamaño del fichero en *bytes*.
- `type`: el tipo MIME del fichero.
- `tmp_name`: el nombre temporal con el que se ha subido al servidor.
- `error`: código de error asociado a la subida.



UD 2 - Características del lenguaje PHP: procesar_subida.php

```
<?php
$tam = $_FILES["fichero"]["size"];
if($tam > 256 *1024){
    echo "<br>Demasiado grande";
    return;
}
echo "Nombre del fichero: ". $_FILES["fichero"]["name"];
echo "<br>Nombre temporal del fichero en el servidor: ".
    $_FILES["fichero"]["tmp_name"];
$res = move_uploaded_file($_FILES["fichero"]["tmp_name"],
    "subidos/".$_FILES["fichero"]["name"]);
if($res){
    echo "<br>Fichero guardado";
} else {
    echo "<br>Error";
}
```



UD 2 - Características del lenguaje PHP: funciones de validación

<https://www.php.net/trim>

<https://www.php.net/manual/es/book.var.php>

<https://www.php.net/manual/es/filter.examples.validation.php>



SESIÓN
corrección AC_Formularios
Los alumnos presentan su
solución.



COOKIES



UD 2 - Características del lenguaje PHP: \$_COOKIES[]

Las *cookies* son pequeños ficheros que dejan los servidores web en los ordenadores de los clientes. Pueden almacenar información sobre la fecha de la última visita o preferencias de idioma, por ejemplo. Cuando un cliente realiza una petición web, envía al servidor las *cookies* que pudiera tener de este.

Para manejar las *cookies* se usa la función **setcookie()**, que tiene la siguiente cabecera:

```
bool setcookie (string $name [, string $value = "" [, int $expire = 0 [, string $path = "" [, string  
$domain = "" [, bool $secure = FALSE [, bool $httponly = FALSE ]]]]]])
```

Los tres primeros argumentos son los más importantes:

1. El primer argumento es el nombre de la *cookie*.
2. El segundo, el valor que se le quiere dar.
3. El tercero es la fecha en la que expira la *cookie*. Se especifica como una fecha Unix, es decir, el número de segundos pasados desde el comienzo de 1970. Normalmente se utiliza la función **time()**, que devuelve la fecha actual y se le suma un periodo de tiempo expresado en segundos.



UD 2 - Características del lenguaje PHP: \$_COOKIES[]

RECUERDA

- ✓ Las *cookies* se envían como cabeceras de las peticiones HTTP. Hay que enviar las cabeceras antes de empezar con el cuerpo de la respuesta. Esto implica que hay que utilizar la función `setcookie()` antes de que se empiece a escribir la salida. Si se intenta llamar a `setcookie()` después de haber realizado un `echo`, se producirá un error.



UD 2 - Características del lenguaje PHP: \$_COOKIES[]

Cookies

PHP soporta las cookies [HTTP](#) de manera transparente. Las cookies son un mecanismo de almacenamiento de información en el cliente, y de lectura de dicha información. Este sistema permite identificar y seguir a los visitantes. Se puede enviar una cookie con la función [setcookie\(\)](#) o [setrawcookie\(\)](#). Las cookies forman parte de los encabezados [HTTP](#), lo que impone que [setcookie\(\)](#) sea llamada antes de cualquier visualización de texto. Estas son las mismas limitaciones que para [header\(\)](#). Se pueden utilizar las funciones [de bufferización de salida](#) para retrasar la visualización de su script hasta que se haya decidido enviar una cookie o encabezados.

Todas las cookies enviadas al servidor por el cliente serán automáticamente incluidas en un array superglobal [\\$_COOKIE](#) si [variables_order](#) contiene "C". Si se desea asignar múltiples valores a una sola cookie, se debe añadir [] al nombre de la cookie.

Para más detalles, incluyendo notas sobre errores de los navegadores, ver las funciones [setcookie\(\)](#) y [setrawcookie\(\)](#).

<https://www.php.net/manual/es/features.cookies.php>



UD 2 - Características del lenguaje PHP: contador_visitas.php

```
<?php
    if (!isset($_COOKIE['visitas'])) { // si no existe
        setcookie('visitas', '1', time() + 3600 * 24);
        echo "Bienvenido por primera vez";
    } else { // si existe
        $visitas = (int) $_COOKIE['visitas'];
        $visitas++;
        setcookie('visitas', $visitas, time() + 3600 * 24);
        echo "Bienvenido por $visitas vez";
    }
}
```



SESIÓN



UD 2 - Características del lenguaje PHP:

AC: Ver Cookies desde el navegador del cliente

- ¿Cuál es el número máximo de cookies por dominio?
150 - 180
- ¿Pone límite el navegador al número de cookies almacenadas?
- ¿Cuánta información se puede almacenar en una Cookie?
- ¿Cómo se elimina una Cookie?

Actividad propuesta 3.2



Añade un vínculo para borrar la cookie al ejemplo anterior.



UD 2 - Características del lenguaje PHP:

Just an example to clarify the use of the array options, especially since Mozilla is going to deprecate / penalise the use of SameSite = none, which is used by default if not using array options.

```
<?php
$arr_cookie_options = array (
    'expires' => time() + 60*60*24*30,
    'path' => '/',
    'domain' => '.example.com', // leading dot for compatibility or use subdomain
    'secure' => true, // or false
    'httponly' => true, // or false
    'samesite' => 'None' // None || Lax || Strict
);
setcookie('TestCookie', 'The Cookie Value', $arr_cookie_options);
?>
```

<https://www.php.net/manual/en/function.setcookie.php>



SESIÓN AC_Cookies

Solucionar práctica de Cookies



SESIONES

SEGURIDAD: usuarios y roles



UD 2 - Características del lenguaje PHP: sesiones_uso_basico.php

```
<?php
    session_start();
    if (!isset($_SESSION['count'])) {
        $_SESSION['count'] = 0;
    }else {
        $_SESSION['count']++;
    }
    echo "hola ".$_SESSION['count'];
    echo "<br><a href='sesiones_uso_basico2.php'>Siguiente</a>";

```



UD 2 - Características del lenguaje PHP: sesiones_uso_basico2.php

```
<?php  
    session_start();  
    echo "La variable count vale: ". $_SESSION[ 'count' ];
```



UD 2 - Características del lenguaje PHP: ejemplo de sesiones

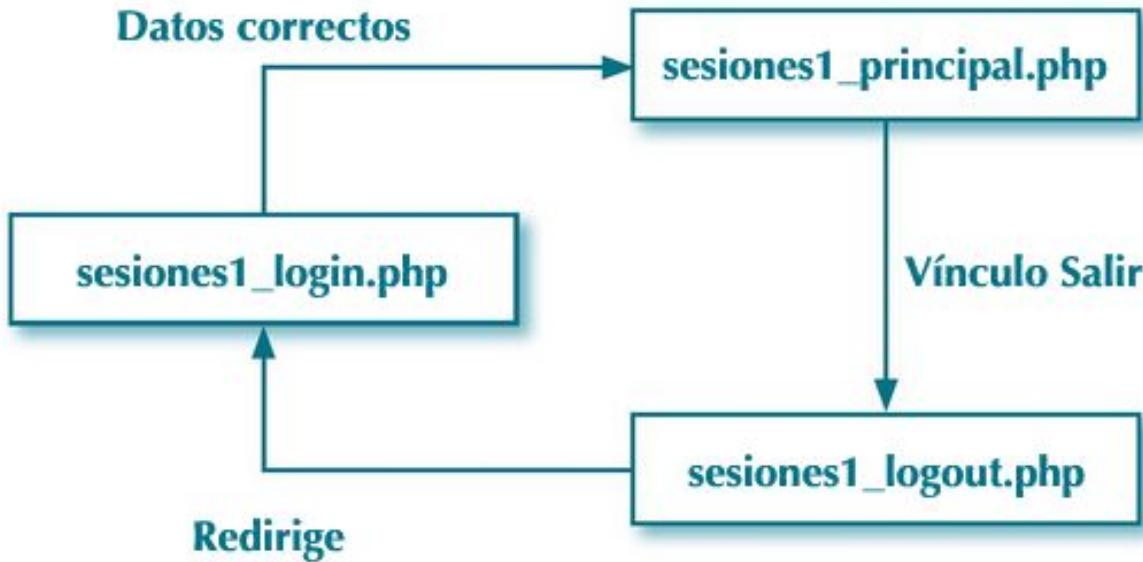


Figura 3.4
Diagrama para el ejemplo de sesiones.



UD 2 - Características del lenguaje PHP: destruir sesión

```
<?php
    session_start(); // unirse a la sesión
    $_SESSION = array();

    session_destroy(); // eliminar la sesión
    // eliminar la cookie

    setcookie(session_name(), 123, time() - 1000);
    header("Location: sesiones1_login.php");
```



SESIÓN

Sesiones y AC_Sesiones



UD 2 - Características del lenguaje PHP: ejemplo de sesiones

Repasar manejo de sesión: login -> principal -> logout -> login

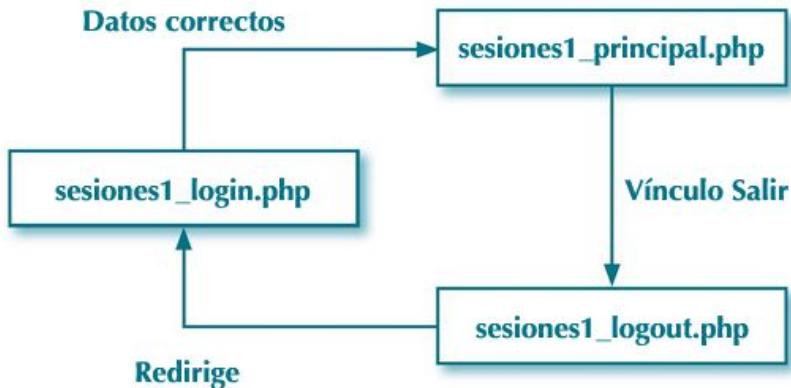


Figura 3.4
Diagrama para el ejemplo de sesiones.



UD 2 - Características del lenguaje PHP: funciones para manejo sesiones

session_start()
session_status()
session_unset()
session_destroy()
...

<https://www.php.net/manual/en/ref.session.php>



UD 2 - Características del lenguaje PHP: login

```
<?php
/*formulario de login
habitual
si va bien abre sesión, guarda el nombre de usuario y redirige a principal.
php
si va mal, mensaje de error */
function comprobar_usuario($nombre, $clave){
if($nombre === "usuario" and $clave === "1234"){


```



UD 2 - Características del lenguaje PHP: login

```
$usu[ 'nombre' ] = "usuario";
$usu[ 'rol' ] = 0;
return $usu;
}elseif($nombre === "admin" and $clave === "1234"){
    $usu[ 'nombre' ] = "admin";
    $usu[ 'rol' ] = 1;
    return $usu;
} else return FALSE;
}
if ($_SERVER[ "REQUEST_METHOD" ] == "POST" ) {
    $usu = comprobar_usuario($_POST[ 'usuario' ], $_POST[ 'clave' ]);
    if($usu==FALSE){
        $err = TRUE;
        $usuario = $_POST[ 'usuario' ];
    }else{
        session_start();
        $_SESSION[ 'usuario' ] = $_POST[ 'usuario' ];
        header("Location: sesiones1_principal.php");
    }
}
```



UD 2 - Características del lenguaje PHP: login

```
<!DOCTYPE html>
<html>
    <head>
        <title>Formulario de login</title>
        <meta charset = "UTF-8">
    </head>
    <body>
        <?php if(isset($_GET["redirigido"])){>
            echo "<p>Haga login para continuar</p>";
        }?>
        <?php if(isset($err) and $err == true){>
            echo "<p> revise usuario y contraseña</p>";
        }?>
        <form method = "POST" action = "<?php echo htmlspecialchars($_
SERVER["PHP_SELF"]);?>" >
            Usuario
            <input value = "<?php if(isset($usuario))echo $usuario;?>">
            id = "usuario" name = "usuario" type = "text">
            Clave
            <input id="clave" name = "clave" type = "password">
            <input type = "submit">
        </form>
    </body>
</html>
```



UD 2 - Características del lenguaje PHP: principal.php

```
<?php
    session_start();
    if(!isset($_SESSION['usuario'])){
        header("Location:sesiones1_login.php?redirigido=true");
    }
?>
<!DOCTYPE html>
<html>
    <head>
        <title>Página principal</title>
        <meta charset = "UTF-8">
    </head>
    <body>
        <?php echo "Bienvenido ".$_SESSION['usuario'];?>
        <br><a href = "sesiones1_logout.php"> Salir <a>
    </body>
</html>
```



UD 2 - Características del lenguaje PHP: logout.php

```
<?php
    session_start(); // unirse a la sesión
    $_SESSION = array();

    session_destroy(); // eliminar la sesión
    // eliminar la cookie

    setcookie(session_name(), 123, time() - 1000);
    header("Location: sesiones1_login.php");
```



SESIÓN



EXPRESIONES REGULARES

(ver apuntes expreg)



UD 2 - Características del lenguaje PHP: Expresiones Regulares

Una expresión regular es un patrón que se compara con una cadena de sujeto de izquierda a derecha. La mayoría de los caracteres representan a sí mismos en un patrón, y coinciden con los caracteres correspondientes en el sujeto. Como ejemplo trivial, el patrón **The quick brown fox** coincide con una parte de una cadena de sujeto que es idéntica a sí misma.



UD3

Utilidades y Bases de Datos



UTILIDADES



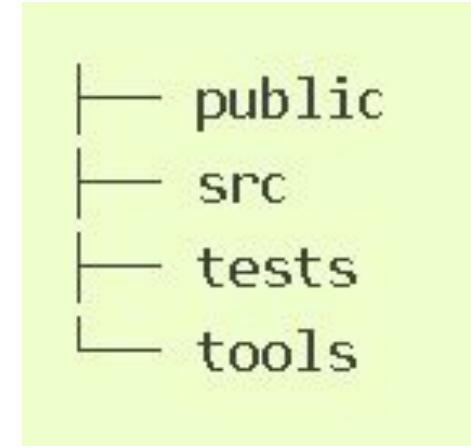
UD 3 - BBDD y Utilidades

Composer - Instalar composer

(desactivar XDebug para poder instalarlo)

- composer –version
- composer self-update
- composer init

Ver estructura de archivos del proyecto.



Instalar phpMailer - (usar SMTP de GMAIL - [Ver miniguía](#))

- código de ejemplo en GitHub de phpmailer.



UD 3 - BBDD y Utilidades

PHPUnit - [ver enlace sobre instalación y uso](#)

jpGraph - [ver enlace sobre instalación y uso](#)

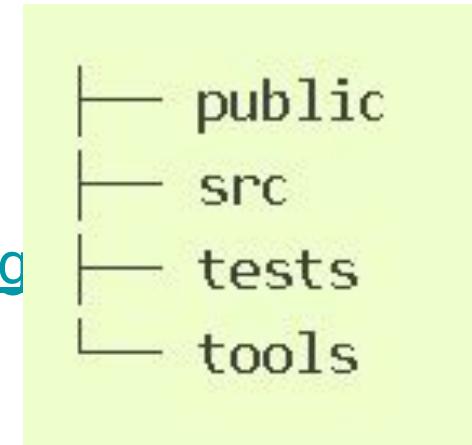
phpDocumentor -

mpdf - creador de pdf

Google Api Client <https://github.com/googleapis/google-api-php-client>

MonoLog

WebCrawler



ACTIVIDAD

UD3_AC_EñviarCorreo (1 + 1/2 SESIONES)



CONEXIÓN A BASES DE DATOS



Conexión a través de PDO ([ver php.net](#))

Visor Sintesis Libro

Ejemplos de ([w3schools](#))

- conexión
- consulta
- transacciones



UD 3 - BBDD y Utilidades

AMPPs y PHPMyAdmin

Crear tabla para el control de usuarios: id, nombre, apellido, edad, clave, rol, email, *ListalPs (lista de IPs de sus conexiones)



UD 3 - BBDD y Utilidades

Example (PDO)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
} catch(PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
}
?>
```



Recuperación y presentación de datos



UD 3 - BBDD y Utilidades

El método `query($cad)` de la clase PDO ejecuta la cadena que recibe como argumento en la base de datos, como se haría desde la línea de comando SQL. El argumento `$cad` tiene que ser una instrucción SQL válida. Devuelve FALSE si hubo algún error o un objeto PDOStatement si la cadena se ejecutó con éxito.

Si se trata de una consulta, es posible recorrer las filas devueltas con un `foreach`. En cada iteración del bucle se tendrá una fila, representada como un `array` en que las claves son los nombres que aparecen en la cláusula `select`.



UD 3 - BBDD y Utilidades

```
<?php
$cadenaConexion = 'mysql:dbname=empresa;host=127.0.0.1';
$usuario = 'root';
$clave = '';
try {
    $bd = new PDO($cadenaConexion, $usuario, $clave);
    echo "Conexión realizada con éxito";
    $sql = 'SELECT nombre, clave, rol FROM usuarios';
    $usuarios = $bd->query($sql);
    echo $usuarios->rowCount(). "<br>";
    foreach ($usuarios as $row) {
        print $row['nombre']. "\t";
        print $row['clave']. "\t";
    }
}
```



UD 3 - BBDD y Utilidades: instrucciones preparadas

También es posible obtener instrucciones preparadas que permiten utilizar parámetros. Se inicializan una sola vez con el método `prepare()` y luego se ejecutan las veces que sea necesario con `execute()`, con diferentes valores para los parámetros. Las instrucciones preparadas permiten reutilizar las consultas, previenen la inyección de código y mejoran el rendimiento.

Hay dos opciones para indicar los parámetros de la consulta, por posición y por nombre. En el primer caso se utiliza el símbolo de interrogación para indicar un parámetro. Al ejecutarla, se asocian por orden los símbolos de interrogación con los valores del *array* que se pasa como argumento a `execute()`.



UD 3 - BBDD y Utilidades: instrucciones preparadas por posición

```
$preparada = $bd->prepare("select nombre from usuarios where
    rol = ?");
$preparada->execute( array(0));
echo "Usuarios con rol 0: ". $preparada->rowCount(). "<br>";
foreach ($preparada as $usu) {
    print "Nombre: ". $usu['nombre']. "<br>";
}
```



UD 3 - BBDD y Utilidades: instrucciones preparadas por nombre

```
$preparada_nombre=$bd->prepare("select nombre from usuarios where rol =:rol");
$preparada_nombre->execute(array(':rol' => 0));
echo "Usuarios con rol 0: ". $preparada->rowCount(). "<br>";
foreach ($preparada_nombre as $usu) {
    print "Nombre: ". $usu['nombre']. "<br>";
}
```



ACTIVIDAD

UD3_AC_LoginBBDD (1/2 SESIÓN)



Inserción, borrado y actualización



UD 3 - BBDD y Utilidades: inserción, borrado y actualización

Sólo debemos ejecutar la sentencia SQL correspondiente.

```
<?php
// datos conexión
$cadenaConexion = 'mysql:dbname=empresa;host=127.0.0.1';
$usuario = 'root';
$clave = '';
try {
    // conectar
    $bd = new PDO($cadenaConexion, $usuario, $clave);
```



UD 3 - BBDD y Utilidades: inserción, borrado y actualización

```
echo "Conexión realizada con éxito<br>";
// insertar nuevo usuario
$ins = "insert into usuarios(nombre, clave, rol)
        values('Alberto', '33333', '1');";
$resul = $bd->query($ins);
//comprobar errores
if($resul) {
    echo "insert correcto <br>";
    echo "Filas insertadas: ". $resul->rowCount(). "<br>";
} else print_r ($bd -> errorinfo());
// para los autoincrementos
echo "Código de la fila insertada".$bd->lastInsertId()."<br>";
// actualizar
$upd = "update usuarios set rol = 0 where rol = 1";
$resul = $bd->query($upd);
//comprobar errores
if($resul){
    echo "update correcto <br>";
    echo "Filas actualizadas: ". $resul->rowCount(). "<br>";
} else print_r($bd -> errorinfo());
// borrar
$del = "delete from usuarios where nombre = 'Luisa'";
$resul = $bd->query($del);
//comprobar errores
if($resul){
    echo "delete correcto <br>";
    echo "Filas borradas: ". $resul->rowCount(). "<br>";
} else print_r($bd -> errorinfo());
} catch (PDOException $e) {
    echo 'Error con la base de datos: '. $e->getMessage();
}
```



Transacciones



UD 3 - BBDD y Utilidades: transacciones

Una transacción consiste en un conjunto de operaciones que deben realizarse de forma atómica. O se realizan todas o ninguna.

Ejemplo: realizar un pago por transferencia a un proveedor entre clientes del mismo banco:

- Operación 1: Descontar el saldo de la cuenta bancaria origen.
- Operación 2: Incrementar el saldo de la cuenta bancaria destino.

Si “*operación 2*” falla, hay que deshacer “*operación 1*”.

- *beginTransaction()*
- *commit()*
- *rollBack()*



UD 3 - BBDD y Utilidades: transaccion.php

- En este ejemplo la segunda operación falla por duplicar campo “*unique*”

```
<?php
$cadena_conexion = 'mysql:dbname=empresa;host=127.0.0.1';
$usuario = 'root';
$clave = '';
try {
    $bd = new PDO($cadena_conexion, $usuario, $clave);
    echo "Conexión realizada con éxito<br>";
    // comenzar la transacción
    $bd->beginTransaction();
    $ins = "insert into usuarios (nombre, clave, rol)
            values('Fernando', '33333', '1')";
    $resul = $bd->query($ins);
    // se repite la consulta
    // falla porque el nombre es unique
    $resul = $bd->query($ins);
    if(!$resul){
        echo "Error: ". print_r($bd->errorinfo());
        // deshace el primer cambio
        $bd->rollback();
        echo "<br>Transacción anulada<br>";
    }else{
        // si hubiera ido bien
        $bd->commit();
    }
} catch (PDOException $e) {
    echo 'Error al conectar: '. $e->getMessage();
}
```



UD 3 - BBDD y Utilidades

AC - Crear usuarios.

Crea una función que te ayude a añadir usuarios a tu tabla de usuarios.
Crea usuarios y roles tomados desde un archivo (json o txt). Usa una query “preparada”.

Crea una función que te ayude a mostrar todos los usuarios de un mismo rol. (consulta w3schools o el libro de DWES)



ACTIVIDAD

UD3_AC_ManejobBDD

(3 SESION)



UD 3 - BBDD y Utilidades

*AC - Crear BBDD con SQL

Crea una función para iniciar tu aplicación:

- Crear BD para los Hobbies.

Crea una función para crear las tablas de tus hobbies (2)

- Crear tablas para cada Hobby de tu práctica anterior.



UD 3 - BBDD y Utilidades

Validación entrada de datos en App PHP

Cuando un usuario envía un formulario (por ejemplo, el login), la información llega a PHP a través del array `$_POST`. Sin embargo, esa información no es fiable:

- Puede venir vacía.
- Puede contener espacios al inicio o al final.
- Puede contener caracteres que no queremos.
- Puede venir manipulada por un atacante.

Por eso utilizamos dos niveles de validación, cada uno con una función distinta y complementaria.

1a capa -> limpieza con `trim()`

2a capa -> validación dentro de las clases (setters)



UD 3 - BBDD y Utilidades

Validación 1a Capa trim()

Al leer los datos del formulario hacemos:

```
$user = trim($_POST['usuario']);  
$pass = trim($_POST['password']);
```

`trim()` elimina espacios al principio y al final:

- “cesar” → “cesar”
- “cesar ” → “cesar”
- “ cesar ” → “cesar”
- “” (vacío) sigue siendo vacío

Esto evita errores tontos por espacios y prepara los datos para validar correctamente. Pero OJO: **! `trim()` NO valida** Solo “limpia” los datos. La validación real debe hacerse dentro de las clases.



UD 3 - BBDD y Utilidades

Validación 2a Capa setters()

La validación importante se hace en los métodos **setUser()** y **setPass()** de la clase **Usuario**.

Estos métodos usan el **trait Validador**, que contiene funciones reutilizables.

```
public function setUser(string $user)

{
    if (!$this->campoObligatorio($user)) { //campoObligatorio($user) -> método de Validador.php (trait)

        throw new Exception("El usuario no puede estar vacío");

    }

    $this->user = $user;
}
```

Si el dato está mal:

- Los setters lanzan una excepción.
- **index.php** la captura en el bloque **try/catch**.
- Se muestra un mensaje amable al usuario.



UD 3 - BBDD y Utilidades

<https://www.php.net/manual/es/class pdo.php>

<https://www.php.net/manual/es/pdo.construct.php>

<https://www.php.net/manual/es/pdo.query.php>

<https://www.php.net/manual/es/pdo.prepare.php>

<https://www.php.net/manual/es/pdo.beginTransaction.php>

https://www.w3schools.com/php/php_mysql_intro.asp



Ficheros



UD 3 - BBDD y Utilidades

En php los ficheros se manejan de forma similar a C, usando métodos como: fopen, fclose, fgets, fgetc, fwrite, feof, fscanf

...

Cuando abrimos un fichero le tenemos que decir con que tipo de permiso lo queremos manejar:

\$file = fopen("mifichero.txt", "r");

```
<?php  
$fich = fopen("fichero_ejemplo.txt", "r");  
if ($fich === FALSE){  
    echo "No se encuentra el fichero o no se pudo leer<br>";  
}else{  
    while( !feof($fich) ){  
        $car = fgetc($fich);  
        echo $car;  
    }  
}  
fclose($fich);
```

Modo	Descripción
r	Solo lectura. Si el fichero no existe, devuelve FALSE
r+	Lectura y escritura. Si el fichero no existe, devuelve FALSE
w	Solo escritura. Si el fichero no existe, se crea; si existe, se trunca (es decir, se borra el contenido anterior). Si no puede crearlo, devuelve FALSE
w+	Lectura y escritura. Si el fichero no existe, se crea; si existe, se trunca. Si no puede crearlo, devuelve FALSE
a	Solo escritura. Las escrituras se realizan siempre al final de fichero (append). Si el fichero no existe, se crea; si existe, se trunca. Si no puede crearlo, devuelve FALSE
a+	Lectura y escritura. Las escrituras se realizan siempre al final de fichero. Si el fichero no existe, se crea; si existe, se trunca. Si no puede crearlo, devuelve FALSE



UD 3 - BBDD y Utilidades

```
$valores = fscanf($fichero, $formato, $var1, ...);
```

Ejemplo de formato

Supongamos que tenemos un fichero de texto que representa una matriz de 4 filas y 4 columnas.

```
23 234 611 5  
1233 565 123 5  
123 54 757 12  
77 88 9 99
```

Para indicar que cada línea está formada por 4 números separados por espacios se usa como formato:

```
"%d %d %d %d"
```

Cada %d representa un número en formato decimal, y se deja un espacio entre ellas.

```
$num = fscanf($fich, "%d %d %d %d");  
echo "$num[0] $num[1] $num[2] $num[3] <br>";
```



UD 3 - BBDD y Utilidades

Función	Descripción
fgets(\$fich)	Devuelve una cadena con los caracteres desde el indicador de posición
fputs(\$fich, \$cad)	Escribe una cadena en el fichero
fseek(\$fich, \$pos)	Sitúa el cursor del fichero en la posición indicada
ftell(\$fich)	Devuelve el indicador de posición del fichero
rewind(\$fich)	Sitúa el indicador de posición al principio del fichero
fopen(\$ruta, modo)	Abre un fichero
fclose(\$fich)	Cierra un fichero
fread(\$fich, \$longitud)	Lee \$longitud bytes
fwrite(\$fich, \$cadena)	Escritura una cadena en un fichero
copy(\$origen, \$destino)	Copia un fichero
unlink(\$borra)	Borra un fichero
move(\$actual, \$nuevo)	Mueve un fichero
filesize(\$ruta)	Devuelve el tamaño del fichero en bytes
filetype(\$ruta)	Devuelve el tipo de fichero
is_file(\$ruta)	Para comprobar si la ruta corresponde a un fichero
is_dir(\$ruta)	Para comprobar si la ruta corresponde a un directorio
rename(\$actual,\$nuevo)	Cambia el nombre a un fichero



UD 3 - BBDD y Utilidades

manejar archivos csv

```
<?php  
$row = 1;  
if (($handle = fopen("test.csv", "r")) !== FALSE) {  
    while (($data = fgetcsv($handle, 1000, ",")) !== FALSE) {  
        $num = count($data);  
        echo "<p> $num fields in line $row: <br /></p>\n";  
        $row++;  
        for ($c=0; $c < $num; $c++) {  
            echo $data[$c] . "<br />\n";  
        }  
    }  
    fclose($handle);  
}  
?>
```

<https://www.php.net/manual/en/function.fgetcsv.php>

fgetcsv(
resource \$stream,
?int \$length = null,
string \$separator = " , ",
string \$enclosure = "\\",
string \$escape = "\\\"
): array | false



UD 3 - BBDD y Utilidades

manejar archivos csv

```
<?php
```

```
$list = [  
    ['aaa', 'bbb', 'ccc', 'dddd'],  
    ['123', '456', '789'],  
    ["aaa", "bbb"]  
];  
  
$fp = fopen('file.csv', 'w');  
  
foreach ($list as $fields) {  
    fputcsv($fp, $fields, ',', "", "");  
}  
  
fclose($fp);  
?>
```

<https://www.php.net/manual/en/function.fputcsv.php>

```
fputcsv(  
    resource $stream,  
    array $fields,  
    string $separator = ',',  
    string $enclosure = "\\"",  
    string $escape = "\\",  
    string $eol = "\n"  
): int|false
```



UD 3 - BBDD y Utilidades

manejar archivos csv

<https://www.php.net/manual/en/function.file-get-contents.php>

<https://www.php.net/manual/en/function.file-put-contents.php>

<https://www.php.net/manual/en/function.parse-ini-file.php>



BLOQUE 3

Comenzar con la UD4 - UD7



CONTACTO

clopezs@iescomercio.com

DEPARTAMENTO de INFORMÁTICA

