Ud2. Tecnologías entorno servidor. PHP.

1. Historia.
2. Características generales y funcionamiento.
3. Instalación y configuración de PHP.
4. Línea de comandos.
5. Buenas prácticas.
6. Estructura y sintaxis básica.
7. Elementos del lenguaje.

1. Historia.

Personal Home Page históricamente, oficialmente acrónimo recursivo de php: “PHP Hypertext Preprocesor”

https://es.wikipedia.org/wiki/PHP

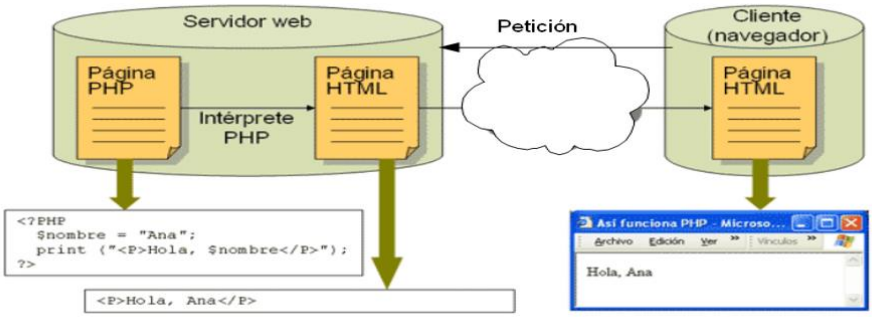
2. Características generales y funcionamiento.

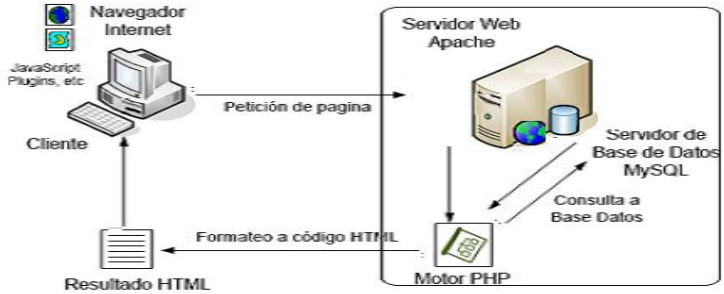
- Lenguaje de programación:
  - Tecnología de servidor interpretada.
  - Multiplataforma.
  - Paradigma: Imperativo (cómo realizar una tarea) y OO
  - Tipado: Débil (control débil en los tipos de datos) y dinámico.
  - Basado en C, C++, Java, Awk, Perl y Bash (shell script de Unix).
  - Software libre bajo licencia GPL:
    - Es gratuito
    - El código fuente está disponible y existe el permiso para modificarlo.
  - Existe una gran cantidad de módulos y añadidos para complementar y aumentar sus prestaciones.
  - Rápido.
  - Curva de aprendizaje baja.

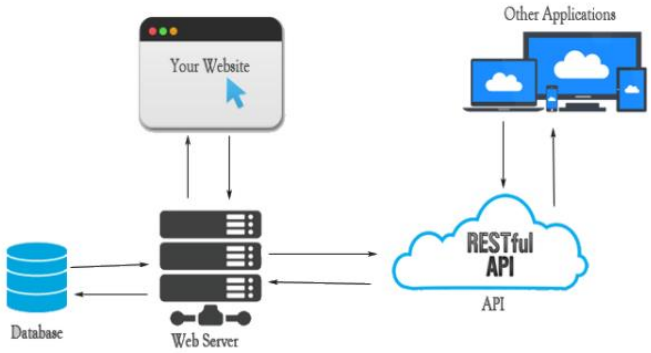
- ¿Qué podemos hacer?
  - Aplicaciones web.
  - Scripting en línea de comandos.
  - Aplicaciones de escritorio utilizando GTK.

- Multitud de aplicaciones realizadas con php.
  - Redes sociales: Algunas redes sociales como Facebook, Twenty lo incluyen entre sus tecnologías de desarrollo.
  - Gestores de contenido: Wordpress, Drupal, Joomla
  - Ecommerce: Magento, OsCommerce
  - Elearning: Moodle

Funcionamiento.







3. Instalación y configuración de PHP.

Instalación

Para poder desarrollar aplicaciones web con acceso a base de datos necesitaremos: servidor web, lenguaje de programación del lado del servidor y servidor de base de datos.

En nuestro caso vamos a utilizar: Apache, PHP y MySQL

Debemos tener claro que son tecnologías independientes y la elección siempre dependerá del tipo de aplicación a desarrollar.

La instalación se hará de la manera habitual en la que el software se intala en nuestro sistema.:

- Apache
- PHP
- MySQL

Existe software alternativo que incorpora estas aplicaciones y que facilita el proceso de instalación sin sistemas Windows y Linux.

- XAMP

Actualmente, es más común utilizar crear entornos de desarrollo con Docker, creando contenedores que incluyan exactamente los servicios que necesitaremos en nuestra aplicación. Esta metodología de trabajo facilita en gran medida el despliegue posterior de las aplicaciones.

Docker

Configuración

El archivo de configuración. **php.ini**

El fichero de configuración (php.ini) es leído al arrancar PHP. En las versiones en que PHP funciona como módulo de servidor, esto sucede únicamente cuando se inicia el servidor. En las versiones CGI y CLI, esto ocurre en cada ejecución.
El fichero php.ini se busca en las siguientes ubicaciones (en orden):

- Ubicación específica en el servidor web (directiva PHPIniDir en Apache 2).Un lugar indicado por la variable de entorno PHPRC. Antes de PHP 5.2.0, esta ubicación se comprobaba después de la clave de registro mencionada más abajo.
- A partir de PHP 5.2.0, se puede establecer la ubicación del fichero php.ini para diferentes versiones de PHP. Se examinan en orden las siguientes claves de registro: [HKEY_LOCAL_MACHINE\SOFTWARE\PHP\x.y.z], [HKEY_LOCAL_MACHINE\SOFTWARE\PHP\x.y] y [HKEY_LOCAL_MACHINE\SOFTWARE\PHP\x], donde 'x', 'y' y 'z' significan la versión mayor, menor, y de edición de PHP. Si existiera un valor para IniFilePath en cualquiera de estas claves, la primera en ser encontrada se utilizaría como ubicación del fichero php.ini (solo en Windows).
- [HKEY_LOCAL_MACHINE\SOFTWARE\PHP], valor de IniFilePath (solo en Windows).
- El directorio actual de trabajo (excepto CLI)
- El directorio del servidor web (para módulos SAPI), o el directorio de PHP (excepto en Windows)
- El directorio de Windows (C:\windows o C:\winnt) (para Windows), o la opción en tiempo de compilación—with-config-file-path.

Para facilitar el el desarrollo php viene con algunas configuraciones por defecto del fichero php.ini para poder utilizarlas.

- Entorno de desarrollo: php.ini-development
- Entorno de producción: php.ini-production.

Algunas directivas:

short_open_tag
- ✓ short_open_tag = Off

error_reporting
- ✓ error_reporting = E_ALL & ~E_NOTICE & ~E_DEPRECATED
- ✓ error_reporting = E_ALL & ~E_STRICT
- ✓ error_reporting = E_ALL ; Valor recomendado

output_buffering
- ✓ output_buffering = 4096
- ✓ output_buffering = Off ; Valor recomendado

extensión pdo_mysql
- ✓ extension=php_pdo_mysql.dll

date.timezone
- ✓ date.timezone = Europe/Madrid

post_max_size (tamaño máximo de lo datos enviados al servidor)
- ✓ post_max_size = 8M

upload_max_filesize
- ✓ upload_max_filesize = 2M

Funciones sobre configuración.

string phpversion ([string $extension])
bool phpinfo ([int $what = INFO_ALL]

4. Línea de comandos.

Desde PHP 5.1.0, SAPI CLI ofrece una consola interactiva si se usa con el modificador -a y PHP está compilado con la opción --with-readline.

Al usar la consola interactiva, se puede escribir directamente código PHP que se ejecuta al momento.

Ejercicios propuestos.

1.- Localizar el fichero de configuración php.ini y explicar el uso y estado de algunas de las directivas de configuración.
2.- Encuentra las diferencias más significativas entre php.ini-development y php.ini-production.
3.- Directivas de configuración para acceso a bases de datos.
4.- Desde la línea de comandos utiliza la opción -v de php para comprobar la versión instalada.
5.- Desde la consola php muestra el resultado de 5+8

5. Buenas prácticas.

Debug.

http://xdebug.org/

Estandar de codificación (zend).

http://zfdes.com/manual/es/coding-standard.html

- Etiqueta inicial de PHP en forma completa (<?php") para empezar con el código. Si el archivo contiene únicamente código PHP, la etiqueta de cierre no es requerida ("?>"), más bien se recomienda omitirla para evitar inyección de espacios en blanco en la respuesta.
- Se recomienda usar indentación de 4 espacios. Se debe configurar el editor de texto para convertir las tabulaciones a espacios en blanco.
- Se recomienda que el ancho de una línea no supere los 80 cacteres, siendo el tamaño máximo de 120 caracteres.
- Internamente, los archivos PHP no deben contener caracteres de retorno de carro ([0x0D] o [0x0D, 0x0A]). Sino que las líneas deben acabar con el carácter de fin de línea ([0x0A]).
- Los nombres de los archivos sólo pueden contener caracteres alfanuméricos, guiones bajos (_) y guiones (-). No se permiten espacios en blanco.
- Archivos que contengan únicamente código php, deben terminar con la extensión.php.

Documentación integrada.

https://www.phpdoc.org/

- Los "docblocks" (bloques de documentación) deben ser compatibles con el formato de phpDocumentor.
- Cualquier archivo con código PHP debe tener un docblock al inicio del archivo.
- Todas las funciones y clases deben contener un dockblock para describirlas.

Ejemplos de documentación:

Script

```
/**
*       Este Archivo contiene funciones utilizadas dentro del programa
*
*       @package    MiProyecto
*       @subpackage Comun
*       @license    http://opensource.org/licenses/gpl-license.php  GNU Public License
*       @author     Pedro Picapiedra < pedropicapiedra@yabadabado.com>
*/
```

Función.

```
/**
*       Calcula la suma del cuadrado de un arreglo
*
*       Ciclo que recorre el arreglo, obtiene el valor lo eleva al cuadrado y se lo
*       suma y retorna el total
*
*       @param array $arr
*       @return int
*       @throws Exception Si el elemento no es un entero
*/
 function sumaDeCuadrados($arr)
 {
   $total = 0;
   foreach ($arr as $val) {
      if (!is_int($val)) {
         throw new Exception("El elemento no es un entero !");
      }
     $total = $val * $val;
   }
   return $total;
}
```

Principios como desarrolladores.

Principio 1: RTFM - "Lee el Maldito Manual"
RTFM es una sigla que significa "lee el maldito manual", algo muy usado en los foros como respuesta hacia los novatos que lo último que hacen es leerlos (lamentablemente *todo* se encuentra ahí)
http://www.es.wikipedia.org/wiki/RTFM

Principio 2: DRY - "No Te Repitas" "No te repitas" significa algo muy simple: si cuando desarrollas ves que al programar "copias" un código para "pegarlo en otro lado", es muy probable que estés haciendo algo mal, ya que ese código debería estar aislado y ser usado a través de parámetros. Generalmente no existe razón para tener que duplicar el código, si estamos muy apurados, seguro, lo pagaremos muy caro dentro de poco.
http://es.wikipedia.org/wiki/DRY

Principio 3: KISS - "Mantenlo Simple, Estúpido!" es un **principio** del diseño que establece que todo sistema funciona mejor cuanto más sencillo es.
http://es.wikipedia.org/wiki/Principio_KISS.

6. Estructura y sintaxis básica.

En una aplicación web el código PHP se encuentra integrado dentro del documento HTML, teniendo en cuenta que los patrones de diseño tienden a separar lo más posible vistas, lógica de negocio y modelo de datos. Esto es, HTML, PHP y bases de datos.

Saliendo de HTML.

Hay cuatro formas de incluir código PHP en HTML

1.
```
<?
...
 ?>
```

Ejemplo:
```
<?
echo (“Hola mundo”);
?>
```

2.
```
<?php
...
?>
```

Ejemplo:
```
<?php
echo(“Hola mundo”);
?>
```

3.
```
<script language=”php”>
...
</script>
```

Ejemplo:
```
<script language=”php”>
echo (“Hola mundo”);
</script>
```

4.
```
 <%
…
%>
```

Ejemplo:
```
<%
echo (“Hola mundo”);
%>

<%= $variable;   # Esto es una forma abreviada de “<%echo ..”
%>
```

Separación de instrucciones.

Las instrucciones terminan con un **punto y coma.**
La etiqueta de cierre *?>* también implica fin de sentencia.

```
<?php
   echo “Hola mundo”;
?>
```

Equivale a

```
<?php
echo “Hola mundo”
?>
```

Comentarios

PHP soporta comentarios tipo C, C++ y shell de Unix.
Caracteres para insertar comentarios en una línea o después de una instrucción:

```
//
#
```

Caracteres para insertar comentarios de varias líneas.

```
/*
…
*/
```

Ejemplos:

```
<?php
echo “Hola mundo”; // Comentario tipo c++ para una línea
/* Primera línea del comentario.
   Segunda línea del comentario.*/
echo “Hola mundo”; # Comentario para una línea tipo shell
?>
```

El tipo de comentario de “una línea” sólo comenta hasta el final de la línea o el final del bloque actual de código PHP.

Tipos de datos.

Tipos simples.

Enteros.

PHP permite trabajar de manera directa con diferentes sistemas de numeración.

```
$a = 1234; # número decimal
$a = -123; # un número negativo
$a = 0123; # número octal (equivalente al 83 decimal)
$a = 0x12; # número hexadecimal (equivalente al 18 decimal)
```

En caso de desbordamiento se convierten en punto flotante

Punto flotante.

Los números en punto flotante ("double") se pueden especificar utilizando cualquiera de las siguientes sintaxis:

```
$a = 1.234;
$a = 1.2e3;
```

Cadena de caracteres.

Conjunto de caracteres agrupados utilizando uno de los siguientes delimitadores.

- *Comillas dobles (" ")*. Si la cadena está encerrada entre dobles comillas ("), las variables que estén dentro de la cadena serán expandidas (sujetas a ciertas limitaciones de interpretación).
- *Comillas simples (' ')*. Cuando una cadena va encerrada entre comillas simples, los únicos caracteres de escape que serán comprendidos son "\\" y "\'". Esto es por convenio, así que se pueden tener comillas simples y barras invertidas en una cadena entre comillas simples. Las variables no se expandirán dentro de una cadena entre comillas simples.

El carácter de barra invertida ("\") se puede usar para especificar caracteres especiales.
El carácter '.' (punto) es el operador de concatenación.
Algunas relgas del estándar de codificación de Zend a nivel de cadenas:

- Cuando un string no contiene sustitución de caracteres, deben usasrse las comillas simples como delimitadores.

```
$a = 'Hola mundo';
```

- Cuando un string contiene apóstrofes, se permite delimitar dicho string con "comillas dobles".

```
$sql = "SELECT 'img' FROM 'tweets' where 'uid' = '1' ";
```

- Se debe dejar un espacio antes y después del operador de concatenación.

```
$msg = 'Hola ' . ' Mundo';
```

- Cuando se concatenan varias línea de texto para mejojar la legibilidad, cada línea debe iniciarse con espacios en blanco para que el operador "." quede alineado debajo del operador "=" de la primera línea:

```
$sql  = "SELECT 'user', 'user_img', 'tweet' "
      . "FROM 'tweets' "
      . "WHERE 'if_active' = 1 "
      . "ORDER BY 'date' desc";
```

Booleanos.

Pueden tomar dos valores posibles: true , false.
PHP es capaz de convertir **cualquier tipo** de dato en **booleano** según las siguientes reglas:

false: Número entero 0
Número decimal 0.000...
Cadena vacía (“”)
Cadena igual a 0 (“0”)
Matriz vacía.
Objeto vacío.
Constante NULL.

true: Resto

Por ejemplo, un valor igual a **-1** se convierte en **TRUE** con PHP.

Tipos compuestos.

Se estudiarán más adelante.

Arrays.

Objetos.

Tipos especiales

Recurso.

NULL

PHP permite conversión explícita de tipos a través de notaciones y de una función específica.
Notación:

(int)
(bool)
(real)
(string)
(array)
(object)

Función:

booleano settype(variable, tipo)

Algunas funciones para trabajar con tipos:

is_*: is_array, is_real, is_null, is_string, …

Constantes

PHP trabaja con un conjunto de constantes predefinidas y dispone de mecanismos para definir constantes de usuario.

Constantes predefinidas.

PHP dispone de un conjunto de constantes predefinidas y constantes mágicas (su valor depende del entorno de ejecución)

__FILE__
El nombre del **archivo de comandos que está siendo interpretado actualmente**. Si se usa dentro de un archivo que ha sido incluido o requerido, entonces se da el nombre del archivo incluido, y no el nombre del archivo padre.

__LINE__
El número de línea dentro del archivo que está siendo interpretado en la actualidad. Si se usa dentro de un archivo incluido o requerido, entonces se da la posición dentro del archivo incluido.

PHP_VERSION
La cadena que representa la versión del analizador de PHP en uso en la actualidad.

PHP_OS
El nombre del sistema operativo en el cuál se ejecuta el analizador PHP; e.g. 'Linux'.

TRUE
Valor verdadero.

FALSE
Valor falso.

E_ERROR, E_WARNING, E_PARSE, E_NOTICE
Constantes para el manejo de errores. Se usan habitualmente con la función error_reporting() para configurar el nivel del informes de error.

Constantes de usuario.

PHP permite la definición de constantes con la función define() y la palabra reservada const.

```
<?php
   define("CONSTANTE", "Hola mundo.");
   echo CONSTANTE; // muestra "Hola mundo."
?>

<?php
   const CONSTANTE = "Hola mundo.";
   echo CONSTANTE; // muestra "Hola mundo."
?>
```

Variables.

En PHP las variables se nombran con un **signo de dólar** seguido por el nombre de la variable. El

nombre de la variable es **sensible a minúsculas y mayúsculas.**

```
$var = "IES";
$Var = "GC";
echo "$var, $Var"; // produce la salida "IES,GC"
```

- La sustitución de variables está permitida en cualquiera de estas dos maneras:

```
$retweet = "$tweet (via @$user_from)";
$retweet = "{$tweet} (via @{$user_from})";
```

Algunas relgas del estándar de codificación de Zend a nivel de variables:

- Los nombres de las variables deben contener caracteres alfanuméricos. No se permiten los guiones bajos (_) y no se recomiendan los números.
- Las variables de instancia que se declaran con los modificadores "private" o "protected", deben tener en el primer carácter un guión bajo (_), mientras que el nombre de las variables declaradas como "public" no puede empezar con un guión bajo.
- Los nombres de las variables deben empezar con una letra minúscula siguiente la norma camel case.
- Los nombres de variables deben ser descriptivos de los datos que el programador almacena en ellas. No se aconsejan nombres de variables como " $i " o " $n " excepto para el contexto de loops o ciclos pequeños (menores a unas 20 líneas de código).

Ámbito de las variables.

El ámbito de una variable es el contexto dentro del que la variable está definida. La mayor parte de las variables PHP tienen un ámbito simple. Este ámbito simple también abarca los ficheros incluidos y los requeridos. Por ejemplo:

```
$a = 1;
include "b.inc";
```

Variables Variables.

A veces es conveniente tener variables que almacenan el nombre de otras variables, lo que permite su utilización de forma dinámica. Una variable variable toma el valor de una variable y lo trata como el nombre de una variable.

```
$a = 'hola';
$$a = 'mundo';
echo "$a ${$a}";
echo "$a $hola";
```

Algunas funciones para trabajar con variables:

empty. Indica si una variable está vacía o no.
isset. Indica si una variable está definida o no.
unset. Elimina una variable.
var_dump. Muestra información sobre una variable (tipo y valor)

Visualización de cadenas.

void echo (string $arg1 [, string $...])

echo Es una construcción del lenguaje (no es una función) y no requiere el uso de paréntesis. Se utiliza para escribir texto, por lo general HTML que será enviado al cliente en respuesta a una petición.
No hay salto de línea automático en el resultado de la ejecución de echo. Cuando sea necesario incluir una nueva línea, se debe insertar la etiqueta html
.
echo posee una sintaxis abreviada, donde se puede poner el símbolo igual justo después de la etiqueta de apertura de PHP.

<?=$var?>

Operadores.

Asignación por valor: =
Asignación por referencia: = &
Aritmeticos:
+
-
*
%
-
/
++operando
operando++
--operando
operando--
Concatenación de cadena: .
Comparación
==
= = =
!=
!==
<
<=
>
>=
Operadores lógicos
and, &&
or, ||
xor
!
Operador condicional
exp1?exp2:exp3
Operadores combinados.
+=
-=
*=
etc

Expresiones.

Una expresión es un conjunto de operadores, variables y/o constantes sintácticamente bien construida según las normas del lenguaje de programación y que será evaluada por éste. Las formas más básicas de expresiones son las constantes y las variables.

Ejemplos

```
$a
3+6
$a*$b
$a >= 0
```

Ejercicios propuestos.

1.- Ficha personal con los datos cargados en variables.
2.- Script para calcular el área de un círculo cargando el valor del radio en una variable.
3.- Script que escriba el resultado de la suma de dos números almacenados en dos variables.
4.- Script que cargue las siguientes variables:

```
$x=10;
$y=7;
```

y muestre

```
10 + 7 = 17
10 -    7 = 3
10 * 7 = 70
10 / 7 = 1.4285714285714
10 % 7 = 3
```

5.- Escribir un script que declare una variable y muestre la siguiente información en pantalla:

```
Valor actual 8.
Suma 2. Valor ahora 10.
Resta 4. Valor ahora 6.
Multipica por 5. Valor ahora 30.
Divide por 3. Valor ahora 10.
Incrementa el valor en 1. Valor ahora 11.
Decrementa el valor en 1. Valor ahora 11.
```

6.- A veces es necesario conocer exactamente el contenido de una variable. Piensa como puedes hacer esto y escribe un script con la siguiente salida:

```
string(5) “Harry”
Harry
int(28)
NULL
```

7.- Escribir un script que utilizando variables permita obtener el siguiente resultado:

```
Valor es string.
Valor es double.
Valor es boolean.
Valor ess integer.
Valos is NULL.
```