

# DESARROLLO WEB FULLSTACK -INTERMEDIO

Autor de contenido

**Andrés Fernando Pineda Guerra**



# Tabla de Contenido



## Presentación

En el curso de desarrollador Full Stack como componente intermedio, podrán adquirir las habilidades y lenguajes necesarios para el desarrollo web, enfocándose en sus grandes pilares, como lo son Front End, Back End, Diseño y modelamiento de aplicaciones y documentación de código.

El curso trata temas emergentes tales como, la seguridad informática, desarrollo de aplicaciones móviles, gestión de base de datos, todo esto basado en la metodología Scrum. De la misma manera, se hace énfasis en el manejo de proyectos tanto en los módulos de desarrollo como los módulos de gestión de proyectos de TI.

## Objetivos del curso (competencias)



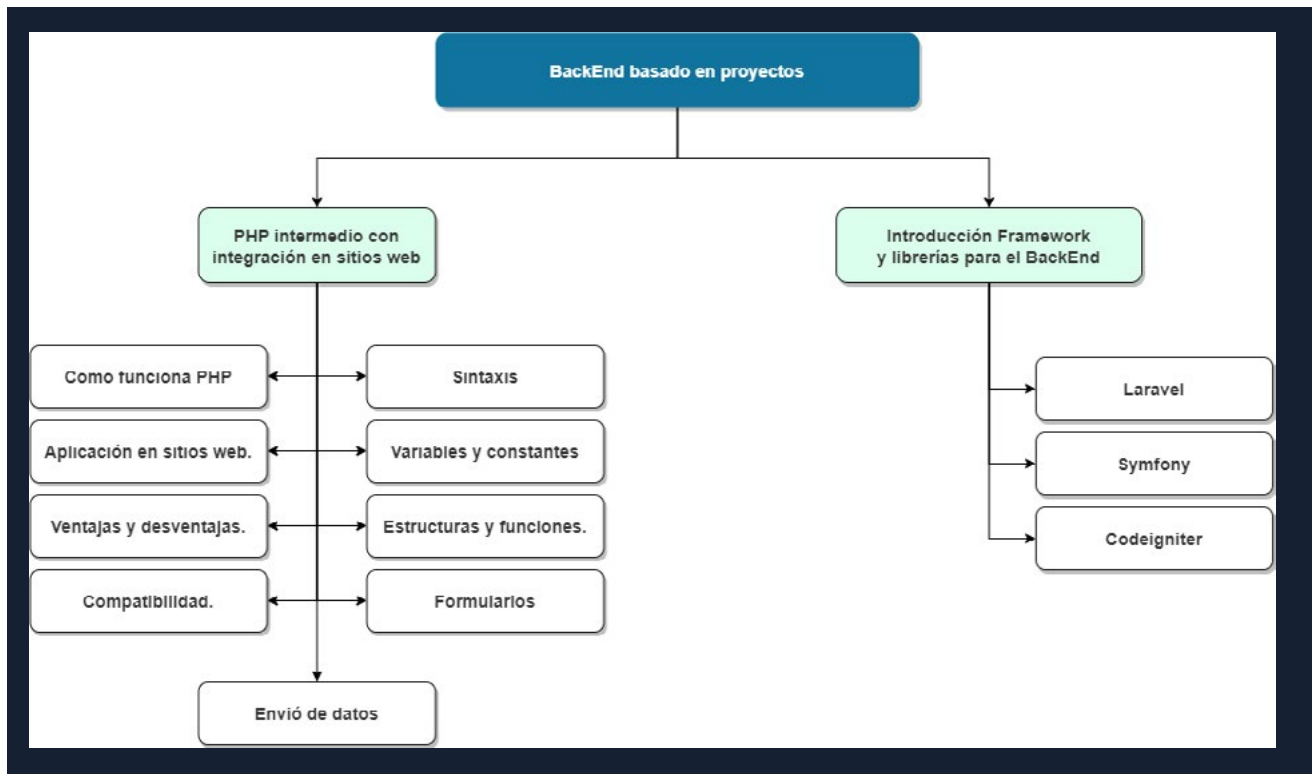
### Objetivo general

Formar a los participantes en el desarrollo web en todo el ciclo de vida del software, en donde adquieran los conocimientos básicos para implementar soluciones web.

### Objetivo específico

- Conocer los conceptos y teoría básica del desarrollo web.
- Identificar y conocer los diferentes lenguajes de programación y herramientas para el desarrollo web.
- Aplicar las diferentes tecnologías web, tendencias y herramientas en el desarrollo de soluciones web enfocadas a proyectos.
- Diseñar, desarrollar e implementar soluciones web básicas en donde se integren los componentes de Front End, Back End, seguridad, redes y buenas prácticas utilizando metodologías ágiles.
- Identificar y conocer los conceptos básicos para el desarrollo móvil, así como aplicar su desarrollo en aplicaciones básicas.

## Mapa de contenido de la unidad



## Módulo 6 Backend basado en proyectos

### Ideas clave

Repaso general de los lenguajes de Back End, enfoque en Python, Java, PHP en donde se hace énfasis en el envío de datos, formularios, estructuras y funciones, variables y constantes, ventajas de PHP, framework de trabajo.

## 6.1. PHP intermedio con integración en sitios web



### Introducción a PHP.

Es uno de los lenguajes de programación más utilizados en el ámbito web, este permite ejecutar comandos o abrir conexiones de red desde el servidor. Estas propiedades hacen que, por omisión, sea inseguro todo lo que se ejecute en un servidor web.

Dado que hay muchas vías para ejecutar PHP, existen muchas opciones de configuración para controlar su comportamiento. Al haber una extensa selección de opciones se garantiza poder usar PHP para un gran número de propósitos, pero a la vez significa que existen combinaciones que conllevan una configuración menos segura.

La flexibilidad de configuración de PHP rivaliza igualmente con la flexibilidad de su código. PHP puede ser usado para construir completas aplicaciones de servidor, con toda la potencia de un usuario de consola, o se puede usar sólo desde el lado del servidor implicando un menor riesgo dentro de un entorno controlado. El cómo construir ese entorno, y cómo de seguro es, depende del desarrollador PHP.

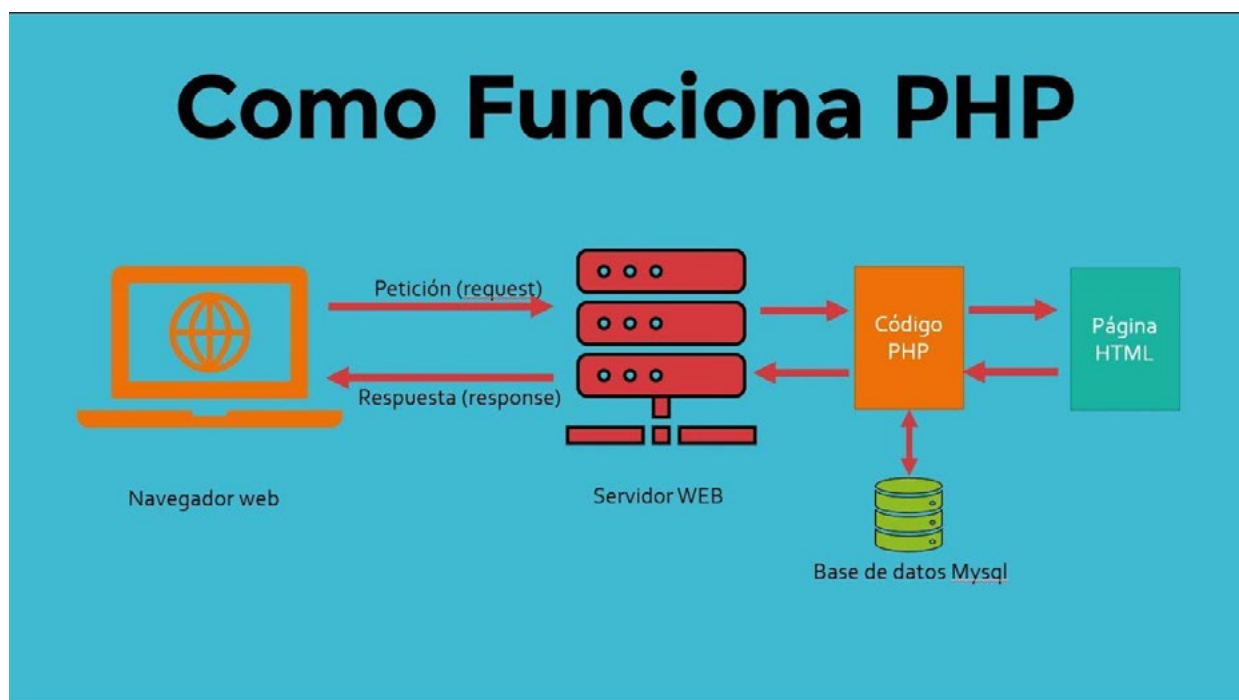
Este capítulo comienza con algunos consejos generales de seguridad, explica las diferentes combinaciones de opciones de configuración y las situaciones en que pueden ser útiles, y describe diferentes consideraciones relacionadas con la programación de acuerdo a diferentes niveles de seguridad (Luna et al., 2018).

Es uno de los lenguajes de programación más documentados y sostenibles en el tiempo.

Sus principales características son.

- Cuenta con una gran comunidad
- Existen diversos frameworks
- Se puede ejecutar en cualquier sistema operativo gracias a su gran compatibilidad
- Código abierto
- Fácil de entender y aprender
- Es seguro
- Utiliza una estructura separada

## Como funciona PHP



PHP es un lenguaje de scripts que se ejecuta en el servidor. Esto significa que cuando un usuario solicita una página que contiene código PHP, realiza una petición al servidor.

El intérprete de PHP ejecuta el código o script y produce la salida, que se envía al navegador web del usuario.

El código PHP se incrusta en el código HTML, utilizando etiquetas especiales de PHP.

El código entre estas etiquetas es procesado por el intérprete de PHP, mientras que el resto del código HTML se trata como texto plano y se muestra en el navegador (Luna et al., 2018).

## Aplicaciones en sitio web

PHP es uno de los lenguajes de programación más famosos y más utilizados para el desarrollo, debido a esto es importante saber que tipo de aplicación tiene este lenguaje en los diversos ambientes web, como por ejemplo en los Gestores de contenido, WordPress, Joomla, Drupal, osCommerce, Moodle, entre otros.

A continuación se presenta un modelo de comunicaciones cliente servidor en las aplicaciones web.



## Qué aplicaciones se pueden crear con PHP

El uso de PHP sirve para crear blogs, intranets, tiendas virtuales (e-commerce), plataformas de educación, sitios de noticias, entre otros.

Los servicios web se basan en protocolos de comunicación diseñados específicamente para sus propósitos, unos de los más conocidos con REST y SOAP.

PHP permite el consumo de los servicios web, así como la exposición de los mismos.

PHP se puede usar en la creación de aplicaciones de escritorio, usando librerías como wx PHP, el cual permite modelar todo el software.

PHP es utilizado para la creación de aplicaciones móviles, usando KikAppTools.

## Ventajas

- Fácil de usar y de aprender ya que puede ser integrado con HTML, esto facilita el aprendizaje y la creación de páginas dinámicas.
- Es un lenguaje que permite ser versátil, ya que integra la conexión a base de datos, la creación de páginas web dinámicas, uso de imágenes, envíos de correo electrónico, desarrollo de formularios, entre otros.
- Es gratis
- Tiene diversa documentación gracias al fuerte apoyo de la comunidad.
- Domina muchos gestores de contenidos
- Amplia oferta laboral
- Separación de estructuras

## Compatibilidad

PHP es el preprocesador de hipertexto (PHP) se puede utilizar con Sun Java System Web Server utilizando una de las tres APIs admitidas: CGI, NSAPI y FastCGI. PHP es un lenguaje de creación de secuencias de comando de página disponible a través del grupo PHP (Luna et al., 2018).

- CGI API es la interfaz más estable que se puede usar pero tiene los inconvenientes inherentes a CGI.
- NSAPI utiliza la API original de Sun Java System Web Server para ejecutar el software PHP en la memoria del servidor web. Esta configuración proporcionará el mejor rendimiento, pero corre el riesgo de bloquear el servidor si se utilizan módulos PHP que no sean seguros para los subprocesos.
- La interfaz FastCGI brinda un compromiso entre el rendimiento y la estabilidad.



FastCGI permite que el software PHP permanezca en ejecución después de responder a las solicitudes, a la vez que continúa ejecutándose fuera de la memoria del servidor web. Si se usa un módulo PHP que no sea estable, esto no hará que falle el servidor web. Por este motivo es mejor utilizar la interfaz de FastCGI con el software PHP.

La interfaz FastCGI es compatible con Web Server mediante la instalación del complemento FastCGI, disponible en <http://www.sun.com/download/products.xml?id=-42d693c3>.

Cuando se ejecuta como proceso FastCGI, el software PHP utiliza las variables de entorno siguientes para controlar el ciclo de vida de los procesos PHP:

- `PHP_FCGI_CHILDREN` determina el número de procesos PHP que se crearán para responder a las solicitudes.
- `PHP_FCGI_MAX_REQUESTS` determina el número de solicitudes que un proceso PHP puede responder antes de cerrarse a sí mismo y ser sustituido por un nuevo proceso PHP.

Es un lenguaje compatible con todos los navegadores, diversos frameworks, entornos de trabajo y gestores de contenidos, por lo cual es una gran ventaja al utilizar este lenguaje de programación.



## Sintaxis

### Etiquetas de PHP

Cuando PHP analiza un fichero, busca las etiquetas de apertura y cierre, que son `<?php` y `?>`, y que indican a PHP dónde empezar y finalizar la interpretación del código. Este mecanismo permite embeber a PHP en todo tipo de documentos, ya que todo lo que esté fuera de las etiquetas de apertura y cierre de PHP será ignorado por el analizador.

PHP también permite la etiqueta de apertura abreviada `<?` (la cual está desaconsejada debido a que sólo está disponible si se habilita con la directiva `short_open_tag` del fichero de configuración `php.ini`, o si PHP se configuró con la opción `--enable-short-tags`).

Si un fichero contiene solamente código de PHP, es preferible omitir la etiqueta de cierre de PHP al final del mismo. Así se previene la adición de espacios en blanco o nuevas líneas accidentales después de la etiqueta de cierre, lo cual causaría efectos no deseados debido a que PHP comenzará la salida del búfer cuando no había intención por parte del programador de enviar ninguna salida en ese punto del script (Luna et al., 2018).

## Embeber con HTML

Cualquier cosa fuera de un par de etiquetas de apertura y cierre es ignorado por el intérprete de PHP, lo que permite que los ficheros de PHP tengan contenido mixto. Esto hace que PHP pueda ser embebido en documentos HTML para, por ejemplo, crear plantillas.

```
<p>Esto va a ser ignorado por PHP y mostrado por el navegador.</p>
<?php echo 'Mientras que esto va a ser interpretado.'; ?>
<p>Esto también será ignorado por PHP y mostrado por el navegador.</p>
```

Este ejemplo funciona como estaba previsto, porque cuando PHP intercepta las etiquetas de cierre `?>`, simplemente comienza a imprimir cualquier cosa que encuentre (a excepción de una nueva línea inmediatamente después; véase la separación de instrucciones) hasta que dé con otra etiqueta de apertura a menos que se encuentre en mitad de una sentencia condicional, en cuyo caso el intérprete determinará el resultado de la condición antes de tomar una decisión de qué es lo que tiene que saltar.

## Separación de instrucciones.

Como en C o en Perl, PHP requiere que las instrucciones terminan en punto y coma al final de cada sentencia. La etiqueta de cierre de un bloque de código de PHP automáticamente implica un punto y coma; no es necesario usar un punto y coma para cerrar la última línea de un bloque de PHP. La etiqueta de cierre del bloque incluirá la nueva línea final inmediata si está presente.

```
<?php
    echo 'Esto es una prueba';
?>
```

```
<?php echo 'Esto es una prueba' ?>
```

```
<?php echo 'Hemos omitido la última etiqueta de cierre';
```

## Comentarios.

PHP admite comentarios al estilo de 'C', 'C++' y de consola de Unix (estilo de Perl). Por ejemplo:

```
<?php
    echo 'Esto es una prueba'; // Esto es un comentario al estilo de c++ de una sola línea
    /* Esto es un comentario multilínea
       y otra línea de comentarios */
    echo 'Esto es otra prueba';
    echo 'Una prueba final'; # Esto es un comentario al estilo de consola de una sola línea
?>
```

Los comentarios al estilo de “una sola línea” solo comentan hasta el final de la línea o del bloque actual de código de PHP, lo primero que suceda. Esto implica que el código HTML después de // ... ?> o # ... ?> SERÁ impreso: ?> sale del modo PHP y vuelve al modo HTML, por lo que // o # no pueden influir en eso. Si la directiva de configuración asp\_tags está activada, actúa igual que // %> y # %>. Sin embargo, la etiqueta </script> no sale del modo PHP en un comentario de una sola línea.

```
<?php
    /*
        echo 'Esto es una prueba'; /* Este comentario causará un problema*/
    */
?>
```

## Variables y constantes

- **Constantes**

Una constante es un identificador (nombre) para un valor simple. Como el nombre sugiere, este valor no puede variar durante la ejecución del script (a excepción de las constantes mágicas, que en realidad no son constantes). Por defecto, una constante distingue mayúsculas y minúsculas. Por convención, los identificadores de constantes siempre se declaran en mayúsculas.

El nombre de una constante sigue las mismas reglas que cualquier otra etiqueta de PHP. Un nombre de constante válido empieza por una letra o guión bajo, seguido por cualquier número de letras, números o guiones bajos. Usando una expresión regular, se representaría de la siguiente manera: `[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*`

Ejemplo:

```
/*Declaración de constantes*/

const y = 89;
const x = 1.5;
const z = 56;
const i = 9;
const CONSTANTE = 'Hola Mundo';
```

- **Variables**

En PHP las variables se representan con un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

Los nombres de variables siguen las mismas reglas que otras etiquetas en PHP. Un nombre de variable válido tiene que empezar con una letra o un carácter de subrayado (underscore), seguido de cualquier número de letras, números y caracteres de subrayado. Como expresión regular se podría expresar como: `'[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'`

```
/*Declaración de variables*/

$y = 89;
$x = 1.5;
$z = 56;
$i = 9;
$CONSTANTE = 'Hola Mundo';
```

## • Expresiones

Las expresiones son los bloques de construcción más importantes de PHP. En PHP casi todo lo que se escribe es una expresión. La manera más simple y acertada de definir lo que es una expresión es «cualquier cosa que tiene un valor».

Las formas más básicas de expresiones son las constantes y las variables. Cuando se escribe “\$a = 5”, se está asignando ‘5’ a \$a. ‘5’, obviamente, tiene el valor 5, o en otras palabras, ‘5’ es una expresión con el valor de 5 (en este caso, ‘5’ es una constante entera).

Después de esta asignación, se espera que el valor de \$a sea 5 también, por lo que si se escribe \$b = \$a, se espera que esto se comporte tal como si se escribiera \$b = 5. En otras palabras, \$a es también una expresión con el valor 5. Si todo funciona bien, esto es exactamente lo que sucederá.

Un ejemplo de expresiones algo más complejo son las funciones. Por ejemplo, considere la siguiente función:

```
/*EXPRESIONES*/

function foo ()
{
    return 5;
}
```

Asumiendo que está familiarizado con el concepto de función (si no lo está, échele una ojeada al capítulo sobre funciones), asumirá que escribir \$c = foo() es esencialmente igual que escribir \$c = 5. Y está en lo cierto. Las funciones son expresiones con el valor de sus valores devueltos. Ya que foo() devuelve 5, el valor de la expresión ‘foo()’ es 5. Normalmente las funciones no sólo devuelven un valor estático, sino que computan algo.

Por supuesto, los valores en PHP no tienen que ser enteros, y con frecuencia no lo son. PHP soporta cuatro tipos de valores escalares: valores enteros (integer), valores de coma (punto) flotante (float), valores de cadena (string) y valores booleanos (boolean) - (valores escalares son aquellos que no se pueden descomponer en piezas más pequeñas, a diferencia de las matrices, por ejemplo). PHP también soporta dos tipos compuestos (no escalares): matrices (arrays) y objetos. Cada uno de estos tipos de valores pueden ser asignados a variables o devueltos desde funciones.

PHP lleva las expresiones mucho más allá, de la misma manera que lo hacen otros lenguajes. PHP es un lenguaje orientado a expresiones, en el sentido de que casi todo es una expresión. Considere el ejemplo que ya hemos tratado, '\$a = 5'. Es fácil de ver que aquí hay dos valores involucrados, el valor de la constante entera '5', y el valor de \$a que ha sido actualizado a 5 también. Aunque la verdad es que existe aquí un valor adicional involucrado, que es el valor de la asignación misma. La asignación evalúa al valor asignado, en este caso 5. En la práctica, esto significa que '\$a = 5', sin importar lo que haga, es una expresión con el valor 5. De este modo, escribir algo como '\$b = (\$a = 5)' es igual que escribir '\$a = 5; \$b = 5;' (el punto y coma marca el final de una sentencia). Ya que las asignaciones se analizan de derecha a izquierda, también se puede escribir '\$b = \$a = 5'.

Otro buen ejemplo de orientación a expresiones es el pre- y post-incremento y decremento. Los usuarios de PHP y de otros muchos lenguajes pueden estar familiarizados con la notación variable++ y variable--. Éstos son los operadores de incremento y decremento. En PHP, al igual que en C, hay dos tipos de incrementos - pre-incremento y post-incremento. Ambos esencialmente incrementan la variable, y el efecto sobre la variable es idéntico. La diferencia está con el valor de la expresión de incremento. Pre-incremento, que se escribe '++\$variable', evalúa al valor incrementado (PHP incrementa la variable antes de leer su valor, de ahí el nombre de 'pre-incremento'). Post-incremento, que se escribe '\$variable++' evalúa el valor original de \$variable, antes de que sea incrementado (PHP incrementa la variable después de leer su valor, de ahí el nombre de 'post-incremento').

Un tipo de expresiones muy comunes son las expresiones de comparación. Estas expresiones evalúan si algo es false (falso) o true (verdadero). PHP soporta > (mayor que), >= (mayor o igual que), == (igual), != (distinto), < (menor que) y <= (menor o igual que). El lenguaje también soporta un conjunto de operadores de equivalencia estricta: === (igual y del mismo tipo) y !== (diferente o de distinto tipo). Estas expresiones se usan mayormente dentro de ejecuciones condicionales, tales como la sentencia if.

El último ejemplo de expresiones que trataremos aquí es una combinación de expresiones operador-asignación. Ya sabe que si quiere incrementar \$a en 1, puede simplemente escribir '\$a++' o '++\$a'. Pero si lo que quiere es añadirle más de uno, por ejemplo 3, podría escribir '\$a++' varias veces, pero esto, obviamente, no es una manera muy eficiente o cómoda. Una práctica mucho más común es escribir '\$a = \$a + 3'. '\$a + 3' evalúa al valor de \$a más 3, y se vuelve a asignar a \$a, lo que resulta en incrementar \$a en 3. En PHP, como en otros lenguajes como C, se puede escribir esto de una manera más abreviada, lo que con el tiempo se podría convertir en una forma más clara y rápida de entenderlo. Añadir 3 al valor actual de \$a puede ser escrito '\$a += 3'. Esto significa exactamente "toma el valor de \$a, añádele 3 y asígnele de nuevo a \$a". Además de ser más corto y claro, también resulta en una ejecución más rápida. El valor de '\$a += 3', al igual que el valor de una asignación normal, es el valor asignado. Observe que NO es 3, sino



el valor combinado de \$a más 3 (éste es el valor que es asignado a \$a). Se puede usar cualquier operador compuesto de dos partes en este modo de operador-asignación, por ejemplo '\$a -= 5' (restar 5 del valor de \$a), '\$b \*= 7' (multiplicar el valor de \$b por 7), etc.

- **Operadores**

Un operador es algo que toma uno más valores (o expresiones, en la jerga de programación) y produce otro valor (de modo que la construcción en si misma se convierte en una expresión).

Los operadores se pueden agrupar de acuerdo con el número de valores que toman. Los operadores unarios toman sólo un valor, por ejemplo ! (el operador lógico de negación) o ++ (el operador de incremento). Los operadores binarios toman dos valores, como los familiares operadores aritméticos + (suma) y - (resta), y la mayoría de los operadores de PHP entran en esta categoría. Finalmente, hay sólo un operador ternario, ? ;, el cual toma tres valores; usualmente a este se le refiere simplemente como “el operador ternario” (aunque podría tal vez llamarse más correctamente como el operador condicional).

Una lista completa de operadores de PHP sigue en la sección Precedencia de Operadores. La sección también explica la precedencia y asociatividad de los operadores, las cuales gobiernan exactamente cómo son evaluadas expresiones que contienen varios diferentes operadores.

El operador se utiliza para realizar la operación.

Los operadores se dividen principalmente en tres grupos.

- **Operadores Unarios que toman un valor**
- **Operadores binarios que toman dos valores**
- **operadores ternarios que toman tres valores**

Los operadores se dividen principalmente en tres grupos que son en total diecisiete tipos.

a. Operador aritmético

Operador	Símbolo
Suma	+
Resta	-
Multiplicación	*
División	/
Módulo	%
Exponenciación	**

b. Operador de asignación

Operador	Símbolo
Igual a	=

c. Operador de matriz

Operador	Símbolo
Unión	+
Igualdad	==



Identidad	===
Desigualdad	!=
Desigualdad	<>
Sin identidad	!==

d. Operador bit a bit

Operador	Símbolo
y	&
xor	^
no	
desplazar a la izquierda	<<
desplazar a la derecha	>>

e. Operador de comparación

Operador	Símbolo
igual	==
idéntico	===

no igual	!=
no idéntico	!==
no igual	<>
menor que	<
menor que o igual	<=
mayor que	>
mayor o igual	>=

f. Operador de incremento/decremento

Operador	Símbolo
PreIncremento	++\$a
PostIncremento	\$a++
PreDecremento	--\$a
Posdecremento	\$a--

g. Operador lógico

Operador	Símbolo
Y	&&
O	
No	!
y	y
xor	xor
o	o

h. Operador de cadena

Operador	Símbolo
Operador de concatenación	.
Operador de asignación de concatenación	.=

- Tipos de datos

Los valores posibles para la cadena devuelta son:

Dato	Descripción
Boolean	Un valor que puede ser verdadero (true) o falso (false)
int	Un valor numérico entero con signo
Float	Un valor numérico de punto flotante con signo
String	Cadena de caracteres - texto
null	Representa la ausencia de valor para una variable.
resource	Recursos no representados de una forma nativa por PHP, por ejemplo las conexiones de una base de datos y manejadores de archivos.

## Estructuras y funciones.

Un script PHP está contruido a partir de sentencias. Una sentencia puede ser una asignación, una llamada a una función, un loop, una sentencia condicional o una sentencia vacía. Las sentencias normalmente finalizan con un punto y coma.

Las estructuras de control son sentencias que permiten controlar cómo el código fluye en nuestro script basándose en ciertos factores. Por ejemplo, cuando queremos realizar una acción sólo si cierta variable está definida, o cuando queremos mostrar un array de datos a través de un loop.

Las estructuras de control son mayoritariamente condicionales (if, switch, etc) o loops (for, foreach, etc).

- **if**

La estructura de control if permite la ejecución de fragmentos de código.

```
/*Ejemplo if*/

if ($x > $y) {
    echo "$x es mayor que $y";
}
```

- **else**

Con frecuencia se desea ejecutar una sentencia si una determinada condición se cumple y una sentencia diferente si la condición no se cumple. Esto es para lo que sirve else. El else extiende una sentencia if para ejecutar una sentencia en caso que la expresión en la sentencia if se evalúe como false.

```
/*Ejemplo else*/

if ($x > $y) {
    echo "$x es mayor que $y";
} else {
    echo "$y es mayor que $x";
}
```

- **elseif / else if**

elseif, como su nombre lo sugiere, es una combinación de if y else. Del mismo modo que else, extiende una sentencia if para ejecutar una sentencia diferente en caso que la expresión if original se evalúe como false. Sin embargo, a diferencia de else, esa expresión alternativa sólo se ejecutará si la expresión condicional del elseif se evalúa como true.

```
/*Ejemplo elseif/ else if */

if ($x > $y) {
    echo "$x es mayor que $y";
} elseif ($x == $y) {
    echo "$x es igual que $y";
} else {
    echo "$y es mayor que $x";
}
```

- **While**

Los bucles while son el tipo más sencillo de bucle en PHP. Se comportan igual que su contrapartida en C. La forma básica de una sentencia while es:

**while (expr)**  
**sentencia**

El significado de una sentencia while es simple. Le dice a PHP que ejecute las sentencias anidadas, tanto como la expresión while se evalúe como true. El valor de la expresión es verificado cada vez al inicio del bucle, por lo que incluso si este valor cambia durante la ejecución de las sentencias anidadas, la ejecución no se detendrá hasta el final de la iteración (cada vez que PHP ejecuta las sentencias contenidas en el bucle es una iteración). A veces, si la expresión while se evalúa como false desde el principio, las sentencias anidadas no se ejecutarán ni siquiera una vez.

```
/*Ejemplo while*/

$i = 1;
while($i <= 10){
    echo $i;
    $i++;
}
```

- **do-while**

Los bucles do-while son muy similares a los bucles while, excepto que la expresión verdadera es verificada al final de cada iteración en lugar que al principio. La diferencia principal con los bucles while es que está garantizado que corra la primera iteración de un bucle do-while (la expresión verdadera sólo es verificada al final de la iteración), mientras que no necesariamente va a correr con un bucle while regular (la expresión verdadera es verificada al principio de cada iteración, si se evalúa como false justo desde el comienzo, la ejecución del bucle terminaría inmediatamente).

```
/*Ejemplo do while*/

$i = 0;
do {
    echo $i;
} while ($i > 0);

/*Ejemplo for*/

// EJEMPLO 1
for($i = 1; $i <= 10; $i++) {
    echo $i;
}

// EJEMPLO 2
for ($i = 1; ; $i++){
    if($i > 10) {
        break;
    }
    echo $i;
}

// EJEMPLO 3
$i = 1;
for( ; ; ){
    if($i > 10){
        break;
    }
    echo $i;
    $i++;
}

// EJEMPLO 4
for ($i = 1, $j = 0; $i <= 10; $j += $i, print $i, $i++);
```

- **for**

Los bucles for son los más complejos en PHP. Se comportan como sus homólogos en C. La sintaxis de un bucle for es:

**for (expr1; expr2; expr3)  
sentencia**

La primera expresión (expr1) es evaluada (ejecutada) una vez incondicionalmente al comienzo del bucle.

En el comienzo de cada iteración, se evalúa expr2. Si se evalúa como true, el bucle continúa y se ejecutan la/s sentencia/s anidada/s. Si se evalúa como false, finaliza la ejecución del bucle.

Al final de cada iteración, se evalúa (ejecuta) expr3.

Cada una de las expresiones puede estar vacía o contener múltiples expresiones separadas por comas. En expr2, todas las expresiones separadas por una coma son evaluadas, pero el resultado se toma de la última parte. Que expr2 esté vacía significa que el bucle debería ser corrido indefinidamente (PHP implícitamente lo considera como true, como en C). Esto puede no ser tan inútil como se pudiera pensar, ya que muchas veces se debe terminar el bucle usando una sentencia condicional break en lugar de utilizar la expresión verdadera del for.

- **foreach**

El constructor foreach proporciona un modo sencillo de iterar sobre arrays. foreach funciona sólo sobre arrays y objetos, y emitirá un error al intentar usarlo con una variable de un tipo diferente de datos o una variable no inicializada. Existen dos sintaxis:

**foreach (expresión\_array as \$valor)  
sentencias**  
**foreach (expresión\_array as \$clave => \$valor)  
sentencias**

La primera forma recorre el array dado por expresión\_array. En cada iteración, el valor del elemento actual se asigna a \$valor y el puntero interno del array avanza una posición (así en la próxima iteración se estará observando el siguiente elemento).

La segunda forma además asigna la clave del elemento actual a la variable \$clave en cada iteración.



```
/*Ejemplo foreach*/

$array = array(1, 2, 3, 4);
foreach ($array as &$value){
    $value = $value * 2;
}
// cada valor del array vale ahora : 2, 4, 6, 8
unset($value);

/*Ejemplo break*/

$array = array('uno', 'dos', 'parar', 'tres');
while(list(, $valor) = each($array)){
    if($valor == 'parar'){
        break;
    }
    echo "$valor<br>";
}
// Ejemplo con valor numérico
$i = 0;
while (++$i){
    switch($i){
        case 5:
            echo "He llegado a 5 <br>";
            break 1; // Aquí sólo saldría del
switch
            case 10:
                echo "He llegado a 10 <br>";
                break 2; // Sale del switch y del while
            default:
                break;
    }
}
```

- **break**

break finaliza la ejecución de la estructura for, foreach, while, do-while o switch en curso.

break acepta un argumento numérico opcional que indica de cuántas estructuras anidadas circundantes se debe salir. El valor predeterminado es 1, es decir, solamente se sale de la estructura circundante inmediata.

- **continue**

continue se utiliza dentro de las estructuras iterativas para saltar el resto de la iteración actual del bucle y continuar la ejecución en la evaluación de la condición, para luego comenzar la siguiente iteración.

```
/*Ejemplo continue*/

for ($i=0; $i < 10; $i++) {
    if($i % 2 == 0)
        continue;
    print "$i ";
} // Devuelve 1 3 5 7 9
```

- **switch**

La sentencia switch es similar a una serie de sentencias IF en la misma expresión. En muchas ocasiones, es posible que se quiera comparar la misma variable (o expresión) con muchos valores diferentes, y ejecutar una parte de código distinta dependiendo de a qué valor es igual. Para esto es exactamente la expresión switch.

```
/*Ejemplo switch*/

switch ($i) {
    case "perro":
        echo "\$i es un perro";
        break;
    case "gato":
        echo "\$i es un gato";
        break;
    case "avestruz":
        echo "\$i es un avestruz";
        break;
}
```

- **declare**

El constructor declare es usado para fijar directivas de ejecución para un bloque de código. La sintaxis de declare es similar a la sintaxis de otros constructores de control de flujo:

**declare (directive)**  
**statement**

La sección directive permite que el comportamiento de declare sea configurado. Actualmente, solamente están reconocidas tres directivas: ticks (véase abajo para más información sobre la directiva ticks), encoding (véase abajo para más información sobre la directiva encoding) y strict\_types (para más información, véase la sección strict de la página de Argumentos de funciones).

```
/*Ejemplo declare*/

function handler() {
    echo "Hola";
}

register_tick_function("handler");
$i = 0;
declare(ticks = 3) {
    while($i < 9)
        echo ++$i;
} // Devuelve 123Hola456Hola789Hola
```

- **return**

return devuelve el control del programa al módulo que lo invoca. La ejecución vuelve a la siguiente expresión después del módulo que lo invoca.

- **include / include\_once**

La sentencia include incluye y evalúa el archivo especificado.

La sentencia include\_once incluye y evalúa el fichero especificado durante la ejecución del script. Tiene un comportamiento similar al de la sentencia include, siendo la única

diferencia de que si el código del fichero ya ha sido incluido, no se volverá a incluir, e `include_once` devolverá true. Como su nombre indica, el fichero será incluido solamente una vez.

```
/*Ejemplo include / include_once*/

include 'archivo1.php';

$string = get_include_contents('archivo.php');
```

- **requiere / requiere\_once**

`require` es idéntico a `include` excepto que en caso de fallo producirá un error fatal de nivel `E_COMPILE_ERROR`. En otras palabras, éste detiene el script mientras que `include` sólo emitirá una advertencia (`E_WARNING`) lo cual permite continuar el script.

```
/*require hace lo mismo que include pero en caso de fallar devuelve un
error fatal de nivel E_COMPILE_ERROR, por lo que no puede continuar el
script.
include sólo emite un E_WARNING que permite continuar el script.
require_once es igual que require, pero PHP comprobará si el archivo
ya ha sido incluido, y si es así no se incluirá otra vez.*/
```

## Formularios y envío de datos

Para el envío de datos por medio de PHP, se tiene que tener en cuenta la etiqueta `form` y etiquetas `input`. También se deben tener en cuenta los atributos, `action`, `method`, `type` y `name` (HEURTEL, 2016).

En el siguiente formulario simple se muestra la estructura básica de un formulario:

```
<form action="accion.php" method="post">
  <p>Su nombre: <input type="text" name="nombre" /></p>
  <p>Su edad: <input type="text" name="edad" /></p>
  <p><input type="submit" /></p>
</form>
```

Tal como se muestra en la imagen

**Etiqueta form:** El elemento HTML form (<form>) representa una sección de un documento que contiene controles interactivos que permiten a un usuario enviar información a un servidor web.

**Es posible usar las pseudo-clases de CSS:** valid e :invalid para darle estilos a un elemento form.

**Atributo action:** La URI de un programa que procesa la información enviada por medio del formulario. Este valor puede ser sobrescrito por un atributo formaction en un <button> o en el elemento <input>.

**Atributo method:** El método HTTP que el navegador usa para enviar el formulario. Valores posibles son:

**post:** Corresponde al método POST HTTP ; los datos del formulario son incluidos en el cuerpo del formulario y son enviados al servidor.

**get:** Corresponde al método GET HTTP; los datos del formulario son adjuntados a la URI del atributo action , con un '?' como separador, y la URI resultante es enviada al servidor. Use este método cuando el formulario no tiene efectos secundarios y contiene solo caracteres ASCII.

Este valor puede ser sobrescrito por un atributo formmethod en un <button> o elemento <input>.

**Etiqueta input:** El elemento HTML <input> se usa para crear controles interactivos para formularios basados en la web con el fin de recibir datos del usuario. Hay disponible una amplia variedad de tipos de datos de entrada y widgets de control, que dependen del dispositivo y el agente de usuario (user agent (en-US)). El elemento <input> es uno de los más potentes y complejos en todo HTML debido a la gran cantidad de combinaciones de tipos y atributos de entrada.

**Atributo type:**

El tipo de control a mostrar. Su valor predeterminado es text, si no se especifica este atributo. Los posibles valores son:

**button:** Botón sin un comportamiento específico.

**checkbox:** Casilla de selección. Se debe usar el atributo value para definir el valor que se enviará por este elemento. Se usa el atributo checked para indicar si el elemento está seleccionado. También se puede usar el atributo indeterminate (el cual solo se puede establecer programáticamente) para indicar que la casilla está en un estado indeterminado (en la mayoría de las plataformas, se dibuja una línea horizontal a través de la casilla).

**color:** Control para especificar un color. Una interfaz de selección de color no requiere más funcionalidad que la de aceptar colores simples como texto (más información).

**date:** Control para introducir una fecha (año, mes y día, sin tiempo).

**datetime:** El elemento HTML `<time>` representa un periodo específico en el tiempo. Puede incluir el atributo `datetime` para convertir las fechas en un formato interno legible por un ordenador, permitiendo mejores resultados en los motores de búsqueda o características personalizadas como recordatorios.

**email:** Campo para introducir una dirección de correo electrónico. El valor introducido se valida para que contenga una cadena vacía o una dirección de correo válida antes de enviarse. Las pseudo-clases `:valid` y `:invalid` son aplicadas según corresponda.

**file:** Control que permite al usuario seleccionar un archivo. Se puede usar el atributo `accept` para definir los tipos de archivo que el control podrá seleccionar.

**hidden:** Control que no es mostrado en pantalla, pero cuyo valor es enviado al servidor.

**image:** Botón de envío de formulario con gráfico. Se debe usar el atributo `src` para definir el origen de la imagen y el atributo `alt` para definir un texto alternativo. Se puede usar los atributos `height` y `width` para definir el tamaño de la imagen en píxeles.

**month:** Control para introducir un mes y año, sin zona horaria específica.

**number:** Control para introducir un número de punto flotante.

**password:** Control de línea simple cuyo valor permanece oculto. Se puede usar el atributo `maxlength` para especificar la longitud máxima del valor que se puede introducir.

**radio:** Botón radio. Se debe usar el atributo `value` para definir el valor que se enviará por este elemento. Se usa el atributo `checked` para indicar si el elemento está seleccionado de forma predeterminada. Los botones radio que tengan el mismo valor para su atributo `name` están dentro del mismo "grupo de botones radio". Solo un botón radio dentro de un grupo puede ser seleccionado a la vez.

**reset:** Botón que restaura los contenidos de un formulario a sus valores predeterminados.

**search:** Cuadro de texto de línea simple para introducir textos de búsqueda. Los saltos de línea son eliminados automáticamente del valor introducido.

**submit:** Botón que envía el formulario.

**tel:** Control para introducir un número telefónico. Los saltos de línea son eliminados automáticamente del valor introducido, pero no hay otra sintaxis forzada. Se pueden usar atributos como `pattern` y `maxlength` para restringir los valores introducidos en este control. Las pseudo-clases CSS `:valid` y `:invalid` son aplicadas según corresponda.

**text:** Campo de texto de línea simple. Los saltos de línea son eliminados automáticamente del valor introducido.

**time:** Control para introducir un valor de tiempo sin zona horaria específica.

**url:** Campo para editar una URL. El valor introducido se valida para que contenga una cadena vacía o una ruta URL absoluta antes de enviarse. Los saltos de línea y espacios en blanco al principio o al final del valor son eliminados automáticamente. Se pueden usar atributos como `pattern` y `maxlength` para restringir los valores introducidos en el control. Las pseudo-clases `:valid` y `:invalid` son aplicadas según corresponda.

**week:** Control para introducir una fecha que consiste en número de semana del año y número de semana sin zona horaria específica.

Atributo name: El atributo name permite a un script acceder a su contenido. Es preferible utilizar los dos atributos con el mismo identificador, por motivos de compatibilidad.

## 6.2. Introducción Framework y librerías para el BackEnd

Estos marcos de trabajo permiten tener una buena base de código y facilita el desarrollo con un pool de elementos integrados, los cuales permiten la creación de soluciones web profesionales e íntegras.

Manejar marcos de trabajo, permiten tener una arquitectura más robusta y escalable en donde se puede realizar diversas aplicaciones enfocadas en mantener las buenas prácticas de la programación (Laaziri et al., 2019).

### ¿Qué aspectos tener en cuenta para la elección de un Framework?

Debe tenerse en cuenta, principalmente la cantidad de herramientas y ayudas que le ofrece al desarrollador, de allí de parte para los siguientes aspectos:

#### Curvas de aprendizaje.

Los frameworks deben permitir un aprendizaje sencillo e intuitivo, en donde el progreso en sus conocimientos sea exponencial y progresivo, sin barreras que ralenticen o frustren proyectos.

#### Documentación y Documentación

Es importante que el framework cuente con una comunidad robusta, en donde se puedan solucionar las diversas dudas que se vayan teniendo en el transcurso de los desarrollos, así mismo poder encontrar tutoriales y documentación específica.

A continuación, se muestran algunos de los Frameworks más utilizados:



## Laravel



Es un **framework** PHP gratis y de código abierto el cual brinda un conjunto de recursos y herramientas con el fin de crear aplicaciones modernas. Laravel cuenta con un amplio repertorio de paquetes y extensiones compatibles, los cuales incrementan la funcionalidad de los desarrollos.

Hay que tener claro que Laravel es un **framework de back end**, que si bien aporta y tiene funciones que mejoran el Frontend, como por ejemplo los sistemas de validación, paginación, consultas dinámicas y en tiempo real, haciendo más “sencilla” la labor del desarrollador, permitiendo optimizar tiempos y concentrando al desarrollador en trabajo de lógica (Laaziri et al., 2019).

Laravel tiene diversos campos de acción en la construcción de soluciones web, permitiendo facilitar diversos procesos, las aplicaciones más habituales son:

- One page, o aplicaciones de una sola página que requiere hacer consultar sin recargar las páginas.
- Sitios de redes sociales.
- E-commerce.
- Sistemas dinámicos en la administración de contenidos.

### Aspectos principales de Laravel

**Base de datos:** el ORM “Eloquent” de Laravel proporciona la mejor abstracción de bases de datos de su clase sin dolores de cabeza. Consulta y actualiza tus datos sin esfuerzo. Eloquent se combina perfectamente con MySQL, Postgres, SQLite y SQL Server.



**Queues:** envía trabajos en segundo plano para realizar tareas lentas como enviar correos electrónicos y generar informes mientras mantienes tiempos de respuesta ultrarápidos. El robusto sistema de colas de Laravel puede procesar trabajos usando Redis, Amazon SQS o incluso MySQL y Postgres.

**WebSockets:** Laravel Echo y la transmisión de eventos hacen que sea muy fácil crear experiencias de usuario modernas y en tiempo real. Crea increíbles aplicaciones en tiempo real mientras potencias sus WebSockets con PHP puro, Node.js o soluciones sin servidor como Pusher y Ably.

**Autenticación:** deja de preocuparte por esto. Laravel proporciona la bases para una autenticación segura basada en sesiones, mientras que Laravel Sanctum proporciona una autenticación sencilla para las API y las aplicaciones móviles.



## Symfony



Symfony se compone por una colección de más de 30 bibliotecas PHP que pueden descargarse y utilizarse como pack o de forma individual.

Más allá de los componentes estándar, el framework también cuenta con módulos adicionales que aumentan su alcance o que pueden utilizarse por separado. De acuerdo con información de SensioLabs, en la última década se han realizado más de 500 millones de descargas de packs individuales así como del framework completo, lo que se refleja en su gran expansión global. Multitud de proyectos, como el sistema de gestión de contenidos Drupal, la herramienta de análisis web Piwik o el software para crear comunidades online phpBB, están basados en componentes Symfony.

Todos los packs del framework son reutilizables y están disponibles para su descarga sin ningún coste en la página oficial del proyecto.

### **Características de Symfony:**

Es un framework multiplataforma ya que se puede instalar en cualquier sistema operativo.

Su instalación es muy sencilla, se puede configurar muy rápidamente.

Es muy intuitivo su manejo. Es una de las razones por las que es habitualmente utilizado por desarrolladores principiantes. Pero los más expertos también la utilizan por su profesionalidad.

### **Componentes para el uso de Symfony:**

**Asset:** módulo para generar URL y controlar las versiones de archivos de imágenes, hojas de estilo CSS y aplicaciones JavaScript.

**ClassLoader:** con él se cargan las clases PHP propias automáticamente.

**Debug:** proporciona herramientas de depuración del código PHP que encuentran y clasifican errores.

**DependencyInjection:** define las normas para la producción de objetos de cada proyecto web.

**EventDispatcher:** son los componentes básicos que regulan la comunicación de los módulos individuales en forma de eventos.

**Form:** contiene herramientas que ayudan a crear, de manera sencilla, formularios HTML reutilizables.

**Templating:** instrumentos para la creación de un sistema de plantillas.

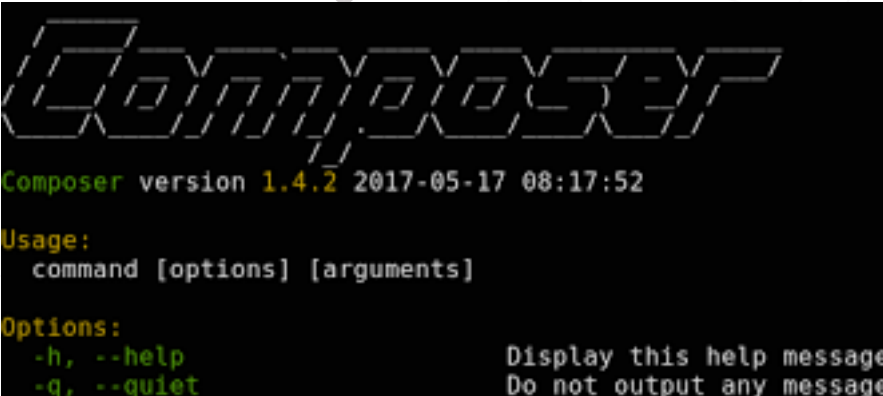
**Translation:** módulo para la internacionalización del proyecto.

**Validator:** permite validar las clases creadas.

**Yaml:** carga y guarda archivos .yaml.

## Librerías para PHP.

### Symfony Console Component



```

Composer
Composer version 1.4.2 2017-05-17 08:17:52
Usage:
  command [options] [arguments]

Options:
  -h, --help                Display this help message
  -q, --quiet               Do not output any message
  
```

La interfaz de línea de comandos (CLI) es uno de los elementos centrales de los principales frameworks de PHP, incluyendo Laravel, Symfony, CodeIgniter y otros. Esta biblioteca proporciona una interfaz de línea de comandos fácil de entender en Symfony. Su integración en la aplicación también es bastante sencilla y está hecha precisamente para construir interfaces de línea de comandos comprobables.

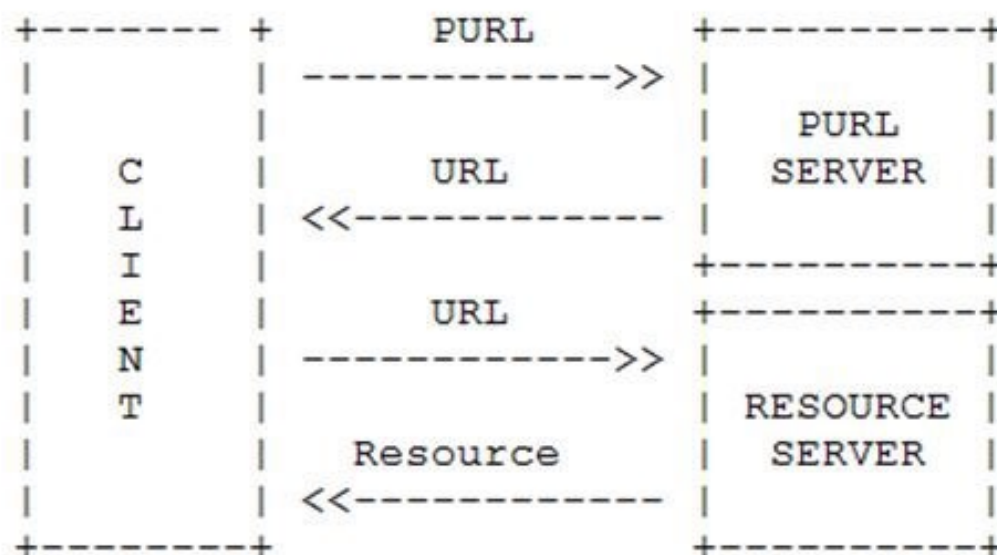
El componente de consola Symfony permite crear comandos CLI personalizados en las aplicaciones PHP. Si nunca has trabajado con Laravel o Symfony, podría ser consciente de las herramientas CLI que proporcionan con el fin de facilitar las operaciones diarias como:

- generar código de andamios
- cachés de claro
- instalación, habilitación y deshabilitando servicios adicionales
- corriente de base de datos de migraciones
- y más

### PHP-code-coverage

Si deseas medir cuánto código fuente de un programa se ejecuta durante una prueba en particular, esta biblioteca te ayuda a medir ese grado de código. La biblioteca te proporciona una funcionalidad de recopilación y representación del código PHP ejecutado para que puedas tener una mejor idea sobre el trozo de código probado y cómo resolver los errores en él.

### Purl



Librería de PHP ligera para trabajar con URLs. Con Purl se puede componer rutas complejas atributo por atributo, extrayendo datos de las URLs.

## PHPExcel



Un conjunto de clases PHP que permiten a los desarrolladores implementar de forma sencilla hojas de cálculo editando sus apps. Esta librería puede leer y crear hojas de cálculo en un sinnúmero de formatos, incluyendo Excel (.xls y .xlsx), OpenDocument (.ods) y CSV por nombrar algunos.

## Otros materiales para profundizar

### Recursos de video



TICnoticos (Director). (2021). Los 5 MEJORES frameworks php explicados en 5 minutos | desarrolladores web. <https://www.youtube.com/watch?v=PU5eRjIQGpc>

TICnoticos (Director). (2021). Los 5 MEJORES frameworks php explicados en 5 minutos | desarrolladores web. <https://www.youtube.com/watch?v=PU5eRjIQGpc>

### Referencias bibliográficas de la unidad



Acosta, J. C., Greiner, C. L., Dapozo, G. N., & Estayno, M. G. (2012, octubre). Medición de atributos POO en frameworks de desarrollo PHP. XVIII Congreso Argentino de Ciencias de la Computación. <http://sedici.unlp.edu.ar/handle/10915/23734>

PHP Avanzado. (s. f.). USERSHOP.

Deitel, H. M., & Deitel, P. J. (2003). Cómo programar en Java. Pearson Educación.

FERNANDEZ, A. (2013). Python 3 al descubierto—2a ed. Alfaomega Grupo Editor.

HEURTEL, O. (2016). PHP 7: Desarrollar un sitio web dinámico e interactivo. Ediciones ENI.

Laaziri, M., Benmoussa, K., Khouilji, S., & Kerkeb, M. L. (2019). A Comparative study of PHP frameworks performance. Procedia Manufacturing, 32, 864-871. <https://doi.org/10.1016/j.promfg.2019.02.295>

Luna, F., Millahual, C. P., & Iacono, M. (2018). PROGRAMACION WEB Full Stack 13 - PHP: Desarrollo frontend y backend - Curso visual y práctico. RedUsers.



**ALCALDÍA MAYOR  
DE BOGOTÁ D.C.**  
SECRETARÍA DE EDUCACIÓN



**ATENEA**  
AGENCIA DISTRITAL PARA LA EDUCACIÓN  
SUPERIOR LA CIENCIA Y LA TECNOLOGÍA



**UNIVERSIDAD DISTRITAL  
FRANCISCO JOSÉ DE CALDAS**  
Acreditación Institucional de Alta Calidad