

```
<!--INGENIERIA DE SISTEMAS-->
```

PHP {

```
<Por="Juan David Martínez",  
"Tomas Mogollon",  
"Daniel Mauricio Bermúdez",  
"Kevin Chaparro",  
"Cristian López";  
>
```

}



# Contenidos

01

Historia

02

Versión Actual.

03

Ranking

04

Utilidad

05

Clases y objetos

06

Contenedores

07

Herencia/Polimorfismo

08

GUI EJEM

# HISTORIA {

PHP fue creado en 1994 por Rasmus Lerdorf como "Personal Home Page Tools". Posteriormente, en 1995, se lanzó la primera versión pública conocida como PHP/FI. En 1997, Andi Gutmans y Zeev Suraski reescribieron el lenguaje, lo que resultó en el lanzamiento de PHP 3. Más tarde, en 2000, se introdujo PHP 4, el cual incorporó el Zend Engine, mejorando significativamente el rendimiento del lenguaje. En 2004, con la llegada de PHP 5, se implementó el Zend Engine II y se mejoró notablemente el soporte para la programación orientada a objetos. A continuación, en 2015, se lanzó PHP 7, que trajo consigo mejoras sustanciales en rendimiento y sintaxis. Finalmente, en 2020, PHP 8 introdujo el compilador JIT (Just-In-Time) y nuevas características, elevando aún más el rendimiento y las capacidades del lenguaje.

- 1994: Creación por Rasmus Lerdorf como "Personal Home Page Tools".
- 1995: Primera versión pública (PHP/FI).
- 1997: PHP 3 lanzado; reescrito por Andi Gutmans y Zeev Suraski.
- 2000: Lanzamiento de PHP 4 con Zend Engine.
- 2004: PHP 5 introdujo Zend Engine II y mejor soporte para POO.
- 2015: PHP 7 mejoras en rendimiento y sintaxis.
- 2020: PHP 8 con JIT (Just-In-Time) compiler y nuevas características.

}

















VERSIÓN ACTUAL {

PHP 8.2 (Última Versión)

La versión más reciente de PHP, 8.2.6, fue lanzada el 9 de mayo de 2023. Esta versión trae varias mejoras y nuevas características en comparación con la versión anterior, PHP 8.1.

}

# RANKING JUNIO 2024 {

Jun 2024	Jun 2023	Change	Programming Language		Ratings	Change
1	1			Python	15.39%	+2.93%
2	3	^		C++	10.03%	-1.33%
3	2	v		C	9.23%	-3.14%
4	4			Java	8.40%	-2.88%
5	5			C#	6.65%	-0.06%
6	7	^		JavaScript	3.32%	+0.51%
7	14	^^		Go	1.93%	+0.93%
8	9	^		SQL	1.75%	+0.28%
9	6	v		Visual Basic	1.66%	-1.67%
10	15	^^		Fortran	1.53%	+0.53%
11	11			Delphi/Object Pascal	1.52%	+0.27%
12	19	^^		Swift	1.27%	+0.33%
13	10	v		Assembly language	1.26%	-0.03%
14	12	v		MATLAB	1.26%	+0.14%
15	8	vv		PHP	1.22%	-0.52%
16	13	v		Scratch	1.17%	+0.15%

}

## Utilidad {

- **Desarrollo Web:** Sitios web, aplicaciones web
- **Manipulación de Bases de Datos:** Conectarte bases, realizar consultas
- **Procesamiento de Formularios:** Datos formularios, información usuarios
- **Generación de Contenido Dinámico:** Contenido web, preferencias usuario
- **APIs Web:** Consumir APIs, servicios externos
- **Manipulación de Archivos y Directorios:** Crear archivos, gestionar recursos
- **Automatización de Tareas:** Automatizar tareas, envío correos
- **Sistemas de Autenticación:** Implementar sistemas, usuarios autorizados

}

# CLASES Y OBJETOS {

## CLASES;

Una clase en PHP es una plantilla o estructura que define las propiedades y métodos comunes a todos los objetos de un tipo específico. Define la estructura básica y el comportamiento de los objetos.

```
1  <?php
2  class Persona {
3      public $nombre;
4      public $edad;
5  }
```

## OBJETOS;

Definición: Un objeto es una instancia específica de una clase. Representa una entidad concreta que tiene sus propias características (propiedades) y puede realizar acciones (métodos) definidas en la clase.

}

## CLASES Y OBJETOS {

```
10     public function saludar(){
11         echo "Hola, mi nombre es " . $this->nombre . " y tengo " .
        $this->edad . " años.";
12     }
13 }
14
15 $persona1 = new Persona("Juan", 30);
16 $persona1->saludar(); // Salida: Hola, mi nombre es Juan
```

### PROPIEDADES;

Las propiedades son variables que pertenecen a un objeto y describen sus características o estado. Cada objeto tiene su propio conjunto de valores para estas propiedades.

```
1  <?php
2  class Coche {
3      public $marca;
4      private $modelo;
5      protected $precio;
6  }
```

}



# CLASES Y OBJETOS {

## MÉTODOS;

Los métodos son funciones definidas dentro de una clase que representan el comportamiento de los objetos de esa clase. Pueden acceder y manipular las propiedades del objeto para realizar acciones específicas.

```
<?php
public function __construct($nombre) {
    $this->nombre = $nombre;
}

// Método que imprime un saludo
public function saludar() {
    echo "Hola, soy " . $this->nombre;
}
```

## ENCAPSULAMIENTO;

Principio de la POO que agrupa propiedades y métodos que operan en esos datos dentro de una sola unidad, llamada clase. Utiliza modificadores de acceso (public, protected, private) para controlar la visibilidad y proteger los datos de la clase.

```
<?php
class Carro {
    public $marca;
    protected $modelo;
    private $precio;
}
```

}

# Contenedores {

## Definición fundamental

Son una herramienta que permite gestionar y organizar las dependencias de una aplicación de forma eficiente.

Actúan como un "inversor de control", donde las clases no crean sus dependencias directamente, sino que las solicitan a través del contenedor. Esto facilita la inyección de dependencias y promueve un diseño de software más modular y mantenible.

```
// Contenedor.php
<?php
1 reference | 0 implementations
class Contenedor {
    3 references
    private $servicios = [];

    // Registrar un servicio en el contenedor
    1 reference | 0 overrides
    public function registrar($nombre, $objeto) {
        $this->servicios[$nombre] = $objeto;
    }

    // Obtener un servicio del contenedor
    1 reference | 0 overrides
    public function obtener($nombre) {
        if (isset($this->servicios[$nombre])) {
            return $this->servicios[$nombre];
        } else {
            throw new Exception("Servicio no registrado: " . $nombre);
        }
    }
}
?>
```

```
// index.php
<?php
include 'Contenedor.php';

// Crear el contenedor
$contenedor = new Contenedor();

// Registrar servicios en el contenedor
$miObjeto = new stdClass(); // Crear un objeto simple de PHP para demostrar
$miObjeto->nombre = 'Mi Servicio Simple';
$contenedor->registrar('servicioSimple', $miObjeto);

// Obtener y usar un servicio del contenedor
$servicioRecuperado = $contenedor->obtener('servicioSimple');

// Mostrar información del servicio recuperado
echo "Nombre del servicio: " . $servicioRecuperado->nombre;
?>
```

}

# Herencia {

La herencia es un principio de programación bien establecido y PHP hace uso de él en su modelado de objetos. Este principio afectará la manera en que muchas clases y objetos se relacionan unas con otras.

¿Para que sirve?

Esto es útil para la definición y abstracción de la funcionalidad y permite la implementación de funcionalidad adicional en objetos similares sin la necesidad de reimplementar toda la funcionalidad compartida.

}

```
<!--INGENIERÍA DE SISTEMAS-->
```

Gracias {

}