



Ingeniería de Software

Capacitación especializada y experiencia profesional

Composer

Curso PHP orientado a objetos



Qué es Composer

- Es el **gestor de dependencias** del lenguaje PHP. Es el equivalente a npm en JavaScript.
- Permite **crear proyectos** e incorporar fácilmente librerías de terceros.
- Requiere previa instalación de **PHP**
- Su primera versión estable data de abril del 2016 (PHP 7.0 7.0)

<https://getcomposer.org/>



A Dependency Manager for PHP

Latest: **2.4.4** ([changelog](#))

[Getting Started](#)

[Download](#)

[Documentation](#)

[Browse Packages](#)

[Issues](#)

[GitHub](#)

Authors: Nils Adermann, Jordi Boggiano and many community contributions

Sponsored by:



Logo by: [WizardCat](#)



Instalación de Composer

- Para **Windows** existe un instalador que facilita el proceso de instalación.
- Para **Linux** la primera opción es usar la versión que trae el gestor de paquetes de cada distribución. La segunda opción es seguir las instrucciones del sitio para descargarlo e instalarlo.

Actualizar la versión de Composer

Es posible actualizarlo con la siguiente instrucción

```
composer self-update
```

```
C:\WINDOWS\system32>composer self-update
Upgrading to version 2.4.4 (stable channel).

Use composer self-update --rollback to return to version 2.3.6

C:\WINDOWS\system32>
```

Tanto en Linux como en Windows es necesario tener los privilegios suficientes para realizar la actualización porque implica la escritura de archivos.

En Windows basta con ejecutar la consola como Administrador y en Linux, usaría sudo.



Creación de un proyecto usando Composer

Para crear un proyecto, primero crea el directorio para alojarlo

```
mkdir HolaMundo
```

A continuación, ejecuta el comando especificando el directorio de trabajo:

```
composer init --name danielbg/hola-mundo -d ./HolaMundo
```



Creación de un proyecto usando Composer

```
Welcome to the Composer config generator
```

```
This command will guide you through creating your composer.json config.
```

```
Package name (<vendor>/<name>) [danielbg/hola-mundo]: —
```

```
Description []: Primer proyecto con composer
```

```
Author [Daniel Barajas Gonzalez <ldanielbg@comunidad.unam.mx>, n to skip]:
```

```
Minimum Stability []: dev
```

```
Package Type (e.g. library, project, metapackage, composer-plugin) []: project
```

```
License []: Apache
```



Creación de un proyecto usando Composer

```
Define your dependencies.
```

```
Would you like to define your dependencies (require) interactively [yes]? no
```

```
Would you like to define your dev dependencies (require-dev) interactively [yes]? no
```

```
Add PSR-4 autoload mapping? Maps namespace "Danielbg\HolaMundo" to the entered relative path. [src/, n to skip]:-
```



Creación de un proyecto usando Composer

```
{
  "name": "danielbg/hola-mundo",
  "description": "Primer proyecto con composer",
  "type": "project",
  "license": "Apache",
  "autoload": {
    "psr-4": {
      "Danielbg\\HolaMundo\\": "src/"
    }
  },
  "authors": [
    {
      "name": "Daniel Barajas Gonzalez",
      "email": "ldanielbg@comunidad.unam.mx"
    }
  ],
  "minimum-stability": "dev",
  "require": {}
}

Do you confirm generation [yes]?
Generating autoload files
```




Estructura de un proyecto

- El archivo `composer.json` contiene la configuración del proyecto desde el punto de vista de Composer.
- El directorio `src` es donde podremos organizar los subdirectorios del proyecto.
- El directorio `vendor` es donde Composer instalará las dependencias que instalemos.

```
Directorio de C:\code\php\HolaMundo
17/11/2022 11:30 a. m. <DIR> .
17/11/2022 11:30 a. m. <DIR> ..
17/11/2022 11:30 a. m. 428 composer.json
17/11/2022 11:30 a. m. <DIR> src
17/11/2022 11:30 a. m. <DIR> vendor
```



Incorporar dependencias

- Una dependencia es una librería de terceros que deseas usar en tu proyecto.
- El primer paso es identificar el nombre del proyecto, compuesto por el nombre del vendor y el nombre de la librería. Tiene el formato **vendor/librería**

<https://packagist.org/>

The screenshot shows the Packagist website interface. At the top, there is a dark header with the text "Packagist The PHP Package Repository" on the left and a "Browse" button on the right. Below the header is a search bar with the placeholder text "Search packages...". The search results for "nesbot/carbon" are displayed below the search bar. The package name "nesbot/carbon" is shown in a large font. Below the package name, there is a command to install the package: "composer require nesbot/carbon". At the bottom of the screenshot, there is a description of the package: "An API extension for DateTime that supports 281 different languages."



Integrar la librería con Composer

Para instalar la librería usa el comando desde el directorio del proyecto

```
composer require <vendor/librería>
```

Por ejemplo:

```
composer require nesbot/carbon
```

- Composer descarga los archivos de dicha librería y los guarda en “./vendor”
- Actualizará el archivo composer.json con la nueva dependencia y creará “composer.lock” con el número de versión de las dependencias instaladas para asegurar que el proyecto funcione de manera consistente siempre que se instale.



Eliminar dependencias

- Para eliminar una dependencia del proyecto, abre el archivo `composer.json` y borra la referencia a la librería.
- Posteriormente ejecuta el comando: `composer update`

```
{.} composer.json
1  {
2    "name": "danielbg/hola-mundo",
3    "description": "Primer proyecto con composer",
4    "type": "project",
5    "license": "Apache",
6    "autoload": {
7      "psr-4": {
8        "Danielbg\\HolaMundo\\": "src/"
9      }
10   },
11   "authors": [
12     {
13       "name": "Daniel Barajas Gonzalez",
14       "email": "ldanielbg@comunidad.unam.mx"
15     }
16   ],
17   "minimum-stability": "dev",
18   "require": {
19     "nesbot/carbon": "2.x-dev"
20   }
21 }
22
```



```
{.} composer.json > ...
1  {
2    "name": "danielbg/hola-mundo",
3    "description": "Primer proyecto con composer",
4    "type": "project",
5    "license": "Apache",
6    "autoload": {
7      "psr-4": {
8        "Danielbg\\HolaMundo\\": "src/"
9      }
10   },
11   "authors": [
12     {
13       "name": "Daniel Barajas Gonzalez",
14       "email": "ldanielbg@comunidad.unam.mx"
15     }
16   ],
17   "minimum-stability": "dev",
18   "require": {
19   }
20 }
21 }
22
```



Dependencias y la estabilidad

- Observa que la estabilidad que establecemos para el proyecto determina la estabilidad de sus dependencias.

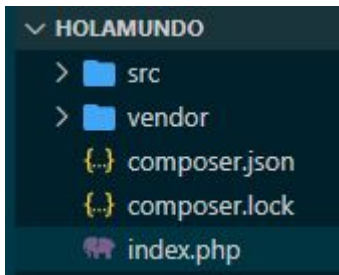
```
{..} composer.json
1 {
2   "name": "danielbg/hola-mundo",
3   "description": "Primer proyecto con composer",
4   "type": "project",
5   "license": "Apache",
6   "autoload": {
7     "psr-4": {
8       "Danielbg\\HolaMundo\\": "src/"
9     }
10  },
11  "authors": [
12    {
13      "name": "Daniel Barajas Gonzalez",
14      "email": "ldanielbg@comunidad.unam.mx"
15    }
16  ],
17  "minimum-stability": "dev",
18  "require": {
19    "nesbot/carbon": "2.x-dev"
20  }
21 }
22
```



```
{..} composer.json > ...
1  √ {
2    "name": "danielbg/hola-mundo",
3    "description": "Primer proyecto con composer",
4    "type": "project",
5    "license": "Apache",
6    √ "autoload": {
7      √ "psr-4": {
8        "Danielbg\\HolaMundo\\": "src/"
9      }
10   },
11   √ "authors": [
12     √ {
13       "name": "Daniel Barajas Gonzalez",
14       "email": "ldanielbg@comunidad.unam.mx"
15     }
16   ],
17   "minimum-stability": "stable",
18   "require": {
19     "nesbot/carbon": "^2.63"
20   }
21 }
```



Importar y usar las librerías



```
index.php ×
index.php > ...
1  <?php
2
3  require_once("../vendor/autoload.php");
4
5  use Carbon\Carbon;
6
7  $ahora = Carbon::now();
8
9  ?>
10
11 <h1>Hola, Mundo!</h1>
12
13 <br>
14
15 La fecha y hora actual es: <?php echo $ahora; ?>
16
```

Autoload

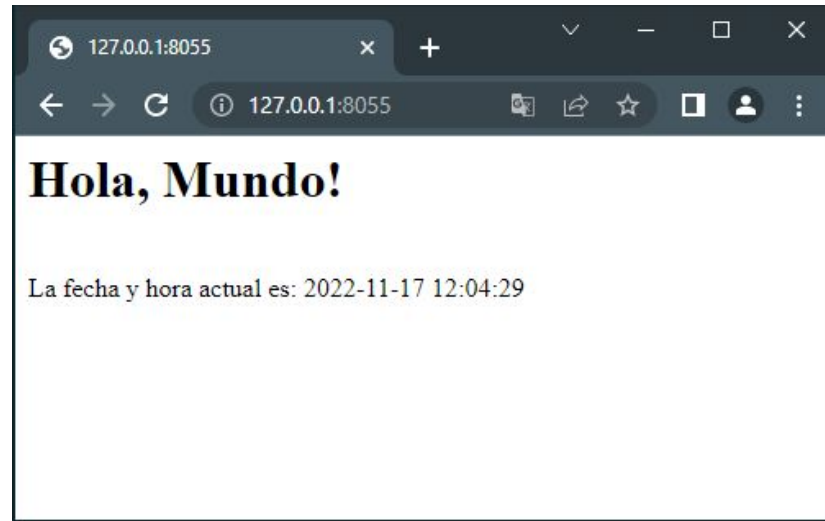
Importación

Uso

Levantamos servidor y probamos

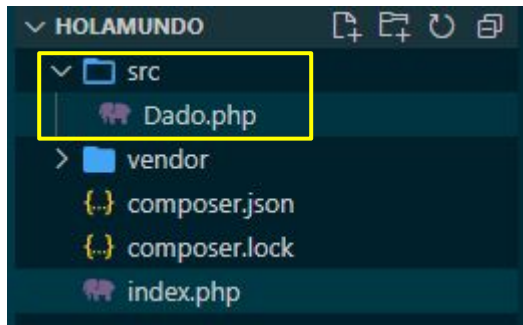
Nos ubicamos dentro del directorio del proyecto y ejecutamos:

```
php -S 127.0.0.1:8055 -t .
```





Creando clases propias



```
Dado.php x
src > Dado.php > ...
1  <?php
2
3  namespace Danielbg\HolaMundo;
4
5  class Dado {
6
7      protected $numero = null;
8
9      public function __construct()
10     {
11         $this->lanzar();
12     }
13
14     public function lanzar() {
15         $this->numero = random_int(1, 6);
16         return $this->numero;
17     }
18
19     public function getNumero(){
20         return $this->numero;
21     }
22 }
```

```
{.} composer.json x
{.} composer.json > {} require
1  {
2      "name": "danielbg/hola-mundo",
3      "description": "Primer proyecto con co
4      "type": "project",
5      "license": "Apache",
6      "autoload": {
7          "psr-4": {
8              "Danielbg\\HolaMundo\\": "src/
9          }
10 }
```




Usando las clases creadas

```
index.php x
index.php > ...
1  <?php
2
3  require_once("../vendor/autoload.php");
4
5  use Carbon\Carbon;
6  use Danielbg\HolaMundo\Dado;
7
8  $ahora = Carbon::now();
9
10 $dado = new Dado();
11
12 ?>
13
14 <h1>Hola, Mundo!</h1>
15
16 <br>
17 La fecha y hora actual es: <?php echo $ahora; ?>
18
19 <br>
20 El dado cayó en: <?php echo $dado->getNumero(); ?>
21
```

Importación

Uso