Becas de FORMACIÓN en TIC

DGTIC

Ingeniería de Software

Capacitación especializada y experiencia profesional

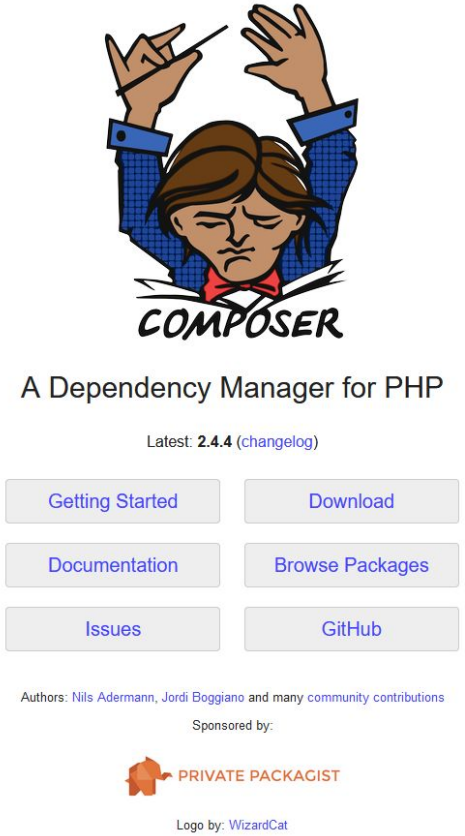
Composer

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/



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

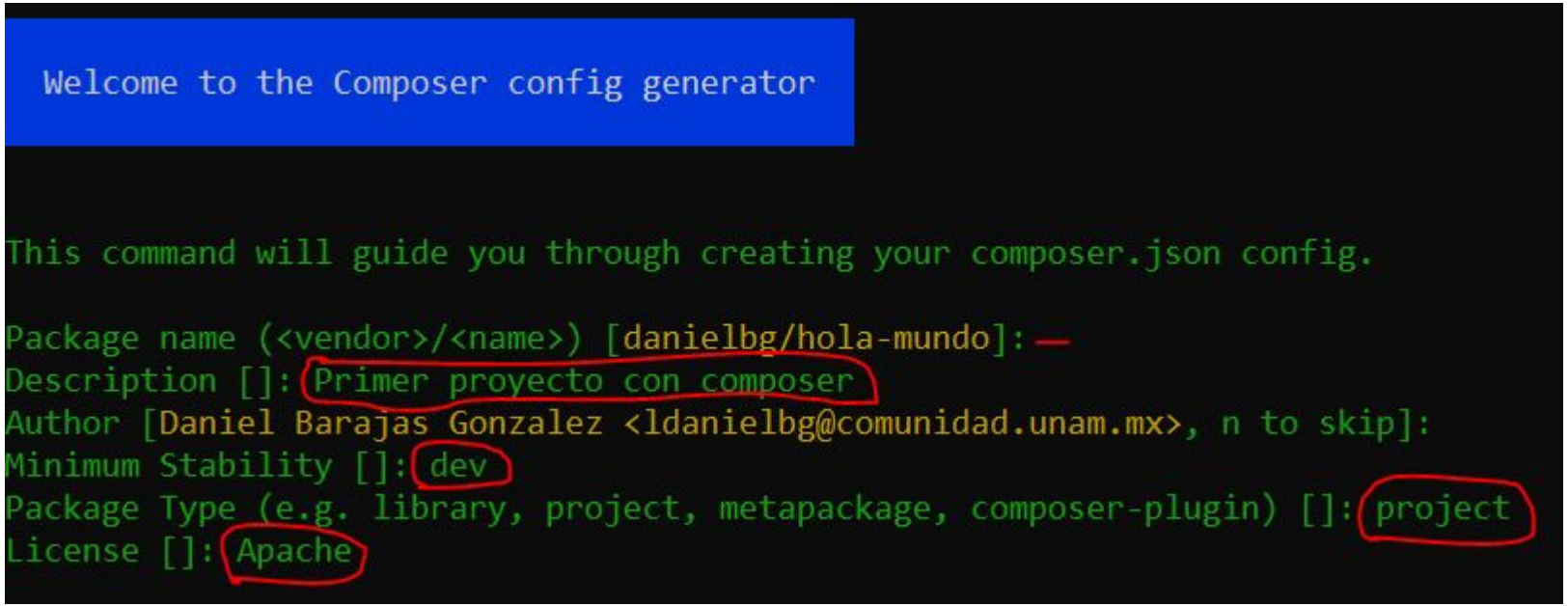
Para crear un proyecto, primero crea el directorio para alojarlo

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



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
```

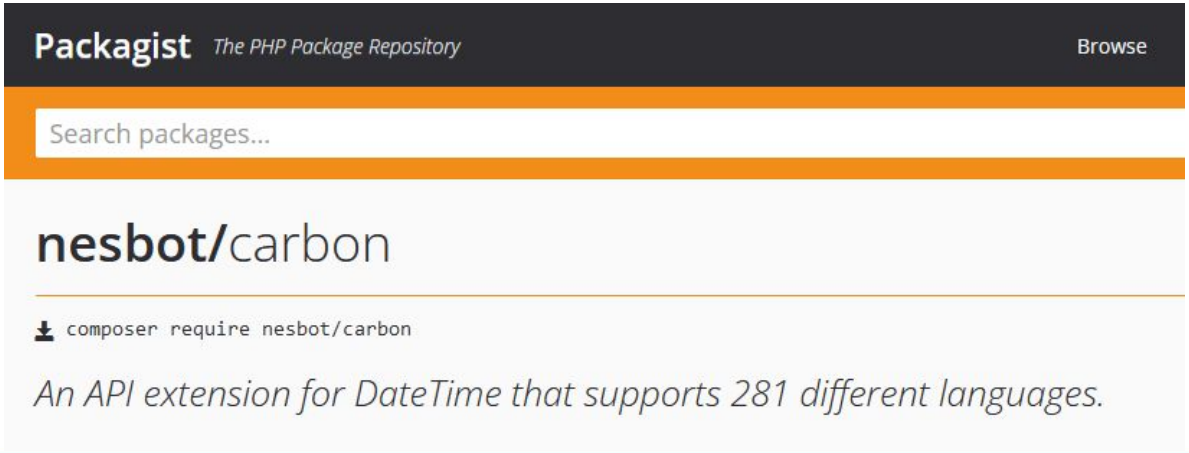
Estructura de un proyecto

- 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.

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/



Integrar la librería con Composer

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

composer require <vendor/librería>

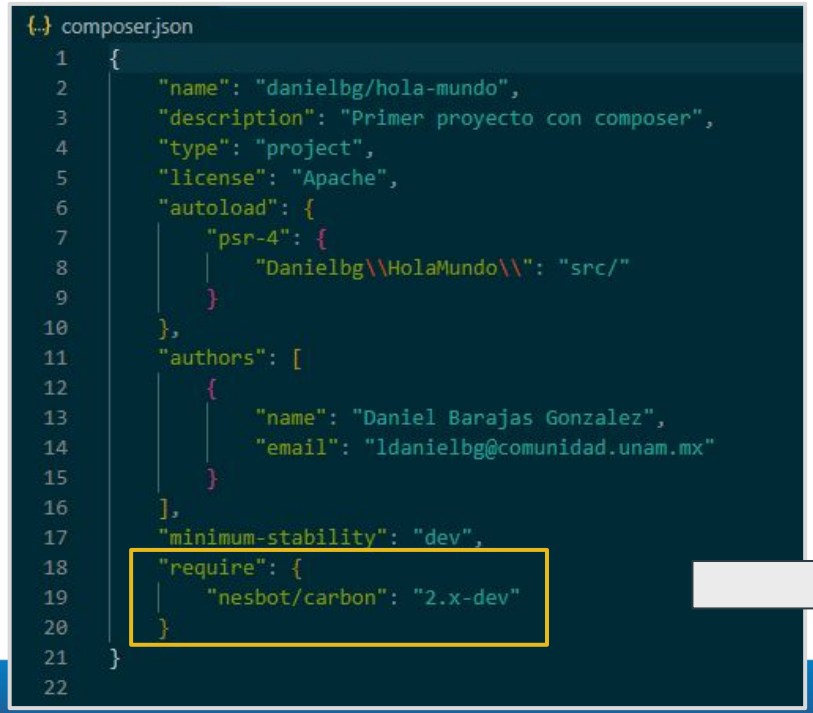
Por ejemplo:

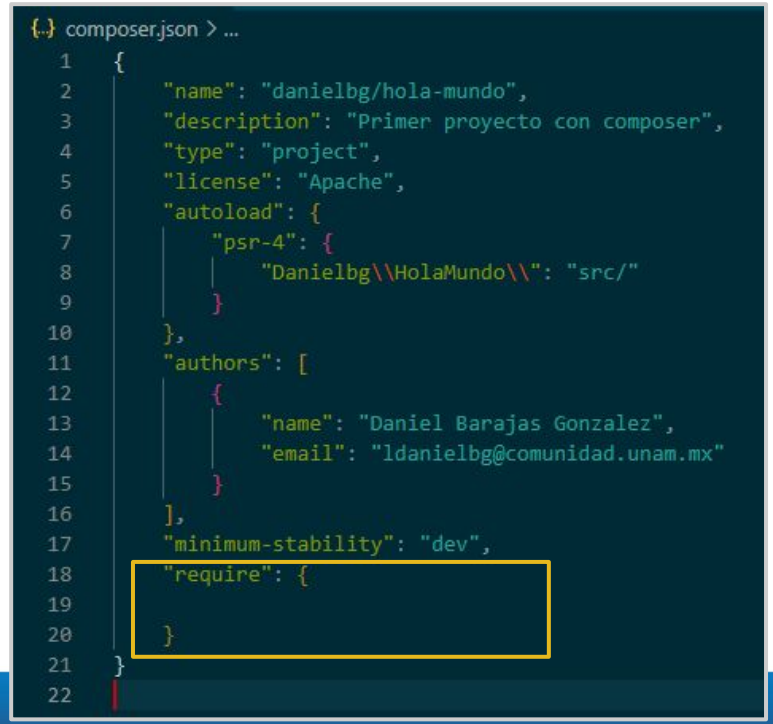
composer require nesbot/carbon

- 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





Dependencias y la estabilidad

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

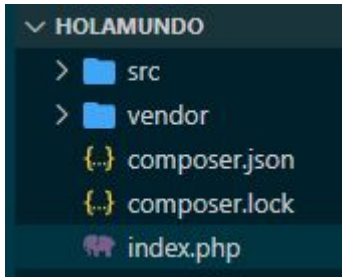
composer.json

```
{
    "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": {
        "nesbot/carbon": "2.x-dev"
    }
}
```

composer.json > ...

```
{
    "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": "stable",
    "require": {
        "nesbot/carbon": "^2.63"
    }
}
```

Importar y usar las librerías



index.php

```php
<?php

require_once("./vendor/autoload.php");

use Carbon\Carbon;

$ahora = Carbon::now();

?>

<h1>Hola, Mundo!</h1>

<br>

La fecha y hora actual es: <?php echo $ahora; ?>
```

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

src > Dado.php > ...

```php
<?php

namespace Danielbg\HolaMundo;

class Dado {

    protected $numero = null;

    public function __construct()
    {
        $this->lanzar();
    }

    public function lanzar() {
        $this->numero = random_int(1, 6);
        return $this->numero;
    }

    public function getNumero(){
        return $this->numero;
    }
}
```

composer.json

composer.json > { } require

```json
{
    "name": "danielbg/hola-mundo",
    "description": "Primer proyecto con co
    "type": "project",
    "license": "Apache",
    "autoload": {
        "psr-4": {
            "Danielbg\\HolaMundo\\": "src/
        }
    },
```

Usando las clases creadas

index.php

```php
<?php

require_once("./vendor/autoload.php");

use Carbon\Carbon;
use Danielbg\HolaMundo\Dado;

$ahora = Carbon::now();

$dado = new Dado();

?>

<h1>Hola, Mundo!</h1>

<br>
La fecha y hora actual es: <?php echo $ahora; ?>

<br>
El dado cayó en: <?php echo $dado->getNumero(); ?>
```

Importación

Uso