

Índice

1. Introducción.....	1
2. Entornos de ejecución.....	1
3. Configuración del entorno.....	2
3.1 Dependencias.....	2
3.1.1 Extensiones PHP.....	2
3.1.2 Librerías y frameworks.....	3
3.2 Instalación de composer.....	4
3.2.1 Instalación de composer.....	5
4. Instalación de Laravel.....	5
4.1 Directorios y permisos.....	6
4.1.1 Directorios con permisos de escritura.....	7
4.2 Variables de entorno.....	9
4.3 Configuración del server block.....	10
4.4 Trabajo a realizar.....	12
5. Referencias.....	13

1. Introducción

Una vez instalados los **diferentes componentes** que forman el STACK de la aplicación, deberemos configurar su entorno de ejecución que vendrá determinado, en gran medida, por la **arquitectura de la aplicación** a desplegar (*frameworks utilizados, gestores de dependencias,...*) y las **tareas** que realiza (*envío de mails, tratamiento de imágenes, integraciones con servicios de terceros,...*).

A lo largo del tema analizaremos los diferentes aspectos a tener en cuenta en la configuración del entorno de ejecución de una aplicación web, ejemplificándolo con despliegue de una aplicación construida mediante el **framework Laravel**.

2. Entornos de ejecución

Antes de configurar el **entorno de ejecución** de la aplicación deberemos tener en cuenta que durante la construcción de una aplicación, salvo que desarrolles directamente sobre el servidor de producción (*escenario no recomendable*), dispondrás de diferentes entornos:

- **Desarrollo:** este es el entorno que utilizan los programadores cuando modifican la aplicación para añadir nuevas características y/o corregir errores. Generalmente se

trata de un entorno local. (*Computador del desarrollador*)

- **Pruebas:** este entorno se utiliza para ejecutar automáticamente las pruebas unitarias de la aplicación. (*En la denominada integración continua, se trata de una instancia temporal que se despliega individualmente en cada ejecución*)
- **Intermedio** (o "**staging**"): lo utiliza el cliente final para probar la aplicación e informar sobre los errores que ha encontrado o las características que faltan en la aplicación. (*Pruebas alfa*)
- **Producción:** este es el entorno en el que se ejecuta la aplicación que utilizan los **usuarios finales**.

3. Configuración del entorno

3.1 Dependencias

Las **librerías, módulos o extensiones** son el principal mecanismo de reutilización de código y uno de las **pilares** en los que se basa la **construcción de software** actualmente. Existen una gran cantidad de proyectos distribuidos en forma de librerías **privativas, libres y open source** que implementan tareas habituales como:

- Manejo de fechas y horas
- Acceso a sistemas gestores de base de datos (Mongo, MySQL, Redis, Postgres,...)
- Compresión de archivos (BZip2, Zip)
- Cifrado/Descifrado (openssl)
- Internalización (iconv)
- Procesamiento de imágenes

Toda aplicación necesitará de una serie de dependencias para su funcionamiento que tendremos que tener en cuenta durante su despliegue. En el **lenguaje PHP** podemos diferenciar 2 tipos: **extensiones** y **librerías**.

3.1.1 Extensiones PHP

Son instaladas a través del **sistema operativo** y extienden las capacidades del lenguaje con nuevas primitivas o tratamientos. Generalmente son descargadas como paquetes binarios (**.so** files) y tienen sus propios archivos de configuración. Muchas de ellas dependen, a su vez, de una aplicación o librería específica instalada en el sistema operativo como es el caso de [openssl](#) o [curl](#).

En sistemas operativos **unix** este tipo de extensiones son muy fáciles de configurar, solo debemos hacer uso del **gestor de paquetes**, que se encargará tanto de su instalación como de su activación a través del fichero con extension **.ini** correspondiente.

```
$ sudo apt install php-curl
```

```
root@root /usr/lib/php/20170718$ sudo apt install php-curl
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
¿Desea continuar? [S/n] S
Des:1 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 php7.2-curl amd64
7.2.24-0ubuntu0.18.04.1 [28,8 kB]
Des:2 http://archive.ubuntu.com/ubuntu bionic/main amd64 php-curl all 1:7.2+60ubuntu1
[1.996 B]
Descargados 30,8 kB en 5s (5.722 B/s)
Seleccionando el paquete php7.2-curl previamente no seleccionado.
(Leyendo la base de datos ... 105767 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../php7.2-curl_7.2.24-0ubuntu0.18.04.1_amd64.deb ...
Desempaquetando php7.2-curl (7.2.24-0ubuntu0.18.04.1) ...
Seleccionando el paquete php-curl previamente no seleccionado.
Preparando para desempaquetar .../php-curl_1%3a7.2+60ubuntu1_all.deb ...
Desempaquetando php-curl (1:7.2+60ubuntu1) ...
Procesando disparadores para php7.2-fpm (7.2.24-0ubuntu0.18.04.1) ...
....
Configurando php7.2-curl (7.2.24-0ubuntu0.18.04.1) ...
...
Creating config file /etc/php/7.2/mods-available/curl.ini with new version
```



Los archivos **.ini** contienen la configuración del módulo y la directiva de carga del mismo. Podemos encontrar sus correspondientes binarios (archivos con extension **.so**) en el path **/usr/lib/php/20190902/**

3.1.2 Librerías y frameworks

Puedes encontrarlas en diferentes repositorios online y son gestionadas a través de un gestor de dependencias como es [composer](#). Ejemplo de este tipo de librerías son: [Guzzle](#) o [phpUnit](#). En muchos casos, dependerán de las extensiones para llevar a cabo la tarea.

Actividad 1 (0.5 puntos)

- Consulta la **documentación** del [framework Laravel](#) y de la [librería phpUnit](#) ¿Qué **módulos o extensiones** necesita cada una de ellas para su funcionamiento? Indica

el tipo de cada una de ellas.

- En un proyecto desarrollado con la tecnología PHP se necesita obtener la nacionalidad de cada una de las peticiones http a través de la dirección IP de la que proviene la solicitud. Busca información sobre las posibles librerías o extensiones existentes.
- Qué **extensiones** tenemos **activadas** en el servicio php-fpm. Busca el directorio de configuración del servicio `/etc/php/7.4/fpm/conf.d/`. ¿Qué módulos tienes actualmente disponibles en el sistema operativo?

3.2 Instalación de composer

Composer puede ser instalado a **nivel global** para cualquier proyecto que se encuentre albergado en el servidor, o a **nivel local** para un proyecto concreto. Nosotros lo instalaremos a nivel global.

Para su instalación, podemos hacer uso de los repositorios de ubuntu o instalarlo de forma manual, tal y como indican sus desarrolladores en la [página oficial](#).

```
$ sudo apt install composer
```



Si optamos por la forma manual, podemos tener problemas con el comando `copy` debido a la configuración de **IPv6** del sistema, podemos solucionarlo asignando prioridad al tráfico **IPv4** en la descarga.

```
sudo sh -c "echo 'precedence ::ffff:0:0/96 100' >> /etc/gai.conf"
```

Una vez instalado, debemos poder ejecutar el comando `composer` desde cualquier terminal.

```
[alecogi@alex:/var/www/html/001-task-list/quickstart$ composer]
Composer 1.6.3 2018-01-31 16:28:17
```

Como ya sabrás, **composer** es un gestor de dependencias tanto internas como externas de la aplicación, es decir, se va a encargar de **importar y cargar** las diferentes **clases y ficheros** que necesitemos tanto si pertenecen al **código fuente de nuestro proyecto** como si son **librerías de terceros** que estemos utilizando desde nuestra aplicación (phpMailer, PHP dotEnv,...).

En entornos de producción podemos mejorar el rendimiento de las búsquedas de cada uno de los ficheros mediante la creación de un fichero estático de referencias. Para ello ejecutaremos el comando `composer install` con la opción `--optimize-autoloader`

```
$ composer install --optimize-autoloader
```

3.2.1 Instalación de composer

La herramienta **composer** nos permite mantener diferentes dependencias para el entorno en el que nos encontremos, lo que nos permitirá **optimizar** los **tiempo de carga** de los ficheros asociados a cada dependencia, a la vez que optimizaremos el **espacio** necesario para el **despliegue** de la aplicación.

En el ejemplo anterior, ejecutando la instrucción con el parámetro `--no-dev`, nos permitiría indicar que solo queremos que se instalen las dependencias que sean para el entorno de producción.

```
$ composer install --optimize-autoloader --no-dev
```

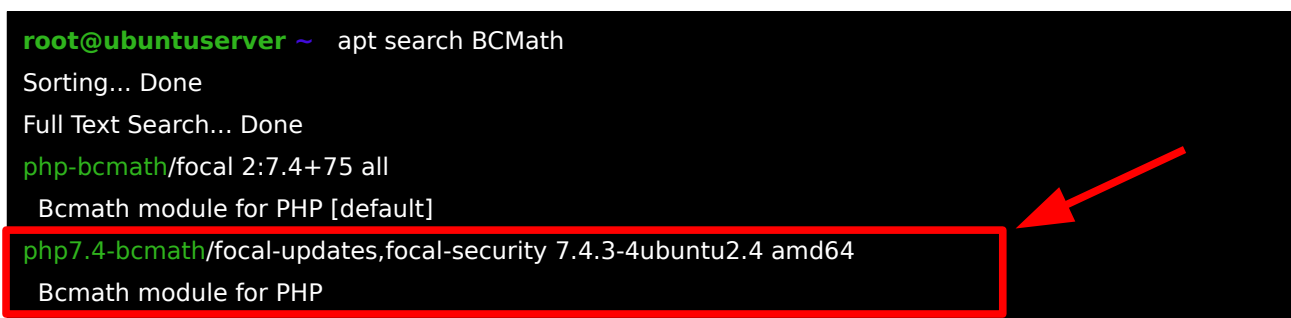
4. Instalación de Laravel

Antes de instalar Laravel, será necesario que lleves a cabo la instalación de las extensiones o **módulos de PHP** necesarios y que habrás conte

stado en la **actividad 1**. Si estamos utilizando un entorno linux, solo **debemos buscar** el nombre de los módulos con los que se ha registrado la extensión mediante el gestor de paquetes apt. Podemos hacer uso de la función `apt search modulo`. Generalmente estos módulos vendrán registrados con un **nombre descriptivo de la extensión**, con el prefijo `php-` de la siguiente forma `php-mbstring`.

```
$ sudo apt search BcMath
```

```
root@ubuntuuserver ~ apt search BCMath
Sorting... Done
Full Text Search... Done
php-bcmath/focal 2:7.4+75 all
  Bcmath module for PHP [default]
php7.4-bcmath/focal-updates,focal-security 7.4.3-4ubuntu2.4 amd64
  Bcmath module for PHP
```



Una vez encontrada, la instalaremos en el sistema operativo para tenerla disponible de forma global en el proyecto que queremos desplegar.

```
$ sudo apt install php7.4-bcmath
```

Antes de la búsqueda e instalación, **actualizaremos** la lista de **paquetes disponibles** y sus **versiones** para asegurarnos que estén en su versión más reciente.

```
$ sudo apt update
```

4.1 Directorios y permisos

Una vez instaladas las dependencias, crearemos el directorio donde vamos a desplegar el proyecto y que constituirá el **directorio base** de despliegue de la aplicación y le damos los permisos y el propietario correspondientes.

```
$ sudo mkdir /var/www/ddaw-example/html -p
$ sudo chown $USER:$USER /var/www/ddaw-example/html
$ sudo chmod 755 /var/www/ddaw-example/html
```

De esta forma, nuestro usuario, será capaz de escribir en el directorio, y por tanto llevar a cabo el **despliegue de la aplicación**.

Por último, nos situaremos en la carpeta donde acabamos de definir el document root de nuestro proyecto y, con la ayuda de **composer**, crearemos un proyecto **Laravel** vacío.

```
$ cd /var/www/ddaw-example/html  
$ composer create-project laravel/laravel myNewApp --prefer-dist
```



Si al ejecutar el comando que nos crea **el nuevo proyecto de laravel** nos encontramos con errores como el siguiente, significará que nos hemos dejado alguna extensión por instalar. En el caso del ejemplo, nos falta instalar la extensión `php-dom`.

```
acoloma@ubuntuserver:/var/www/ddaw-example/html$ composer create-project laravel/laravel  
myNewApp --prefer-dist --no-dev
```

```
Creating a "laravel/laravel" project at "./myNewApp"
```

```
Your requirements could not be resolved to an installable set of packages.
```

Problem 1

```
- phpunit/phpunit 9.5.x-dev requires ext-dom * -> the requested PHP extension dom is missing from  
your system.
```

4.1.1 Directorios con permisos de escritura

El directorio de despliegue de la aplicación pertenece al usuario con el que estamos gestionando la configuración, por lo que tendrá todos los privilegios sobre el mismo (rwx). No obstante, el **servidor de aplicaciones** (que utiliza el usuario `www-data`) **no podrá escribir en él (php-FPM)**.

De forma general, durante la ejecución de una aplicación, existen ciertos directorios que deben ser escribibles por el servidor de aplicaciones como son:

- **Directorio de logs de la aplicación:** donde la aplicación generará y almacenará los ficheros que nos permitirán llevar a cabo una traza de cada petición http.
- **Directorio de subida de imágenes y documentos:** si nuestra aplicación implementa funciones que requieren la subida de ficheros como **imágenes** o **documentos**, debe poder escribir en el **directorio** de subidas.
- **Directorios de caché:** Utilizados por la aplicación para mejorar el tiempo de respuesta, mediante la generación de ficheros estáticos.



Debemos tener en cuenta que este tipo de directorios pueden ser altamente vulnerables, por lo que deberemos asegurarnos que los archivos que se generan no tienen permisos de ejecución, que su contenido es el permitido (`image/jpeg`, `image/png`, `application/pdf`), y que sean los mínimos necesarios para el funcionamiento de nuestra aplicación.

En el caso de **Laravel**, el servidor de aplicaciones debe poder escribir en los siguientes directorios `bootstrap/cache` y `storage` situados en el **directorio raíz** de **nuestro proyecto**.

El procedimiento a seguir será asignar como grupo propietario de los directorios al grupo con el tenemos configurado el pool de conexiones del **servidor php-fpm** para que interprete los **ficheros .php**, es el siguiente:

```
$ sudo chgrp -R www-data storage bootstrap/cache
$ sudo chmod -R ug+rw storage bootstrap/cache
```

Estos directorios se encuentran dentro de `/var/www/ddaw-example/html/myNewApp`

Actividad 2 (0.5 puntos)

- La [documentación oficial](#) del framework nos proporciona información sobre el uso que hace el framework de cada uno de los directorios anteriores, consúltala y explica por qué necesitamos acceso de escritura a los directorios especificados.



Si vamos a **compartir** el servidor de aplicaciones, una mejor aproximación sería crear un grupo para cada una de las aplicaciones y configurar un **pool** de conexiones **específico** para cada aplicación, de modo que cada **pool de conexiones** solo tuviera acceso de escritura sobre los directorios correspondientes de su aplicación.

4.2 Variables de entorno

Como se ha comentado en el **punto 2**, el desarrollo y despliegue de una aplicación requiere de diferentes entornos de ejecución cada uno con diferentes características que deben ser configurables como:

- **Nivel de log:** En entornos de producción no es deseable que nuestros usuarios reciban notificación por pantalla de los diferentes errores que se van produciendo.
- **Capa de Persistencia:** Las bases de datos, los **SGBD** encargados de su gestión y las credenciales de acceso no deben ser las mismas para cada entorno, sino que deben ser independientes.
- **Servicios de terceros:** Si utilizamos servicios de 3ros como una plataforma de envío de **mailing** o un método de pago online, deberemos utilizar diferentes credenciales o incluso crear un mock del servicio durante el desarrollo.

Para poder mantener esta independencia se definen las **variables de entorno**. Estas pueden ser establecidas en la **propia configuración del servidor web/aplicaciones** o en el **propio proyecto**, pero *nunca deberán guardarse en el repositorio del Sistema de Control de Versiones* que estemos utilizando, sino que se crearán a partir de un plantilla base durante el despliegue de la aplicación.

En el caso de Laravel y la mayoría de **frameworks PHP** actuales se hace uso de la librería **DotEnv**. Esta librería se encargará de buscar un archivo **.env** en el directorio raíz de despliegue de la aplicación y convertirá cada una de las entradas `APP_ENV=production` en **variables de entorno**, accesibles de forma global desde cualquier parte de la aplicación web.

Definición de ficheros .env

El siguiente paso para desplegar la aplicación será crear el archivo **.env** para la configuración del entorno y su modificación de acuerdo a las credenciales y parámetros que necesitamos establecer. En primer lugar crearemos el fichero a partir de la copia de la plantilla facilitada junto al proyecto.

```
$ cp .env.example .env
```

A partir de aquí, editaremos el fichero **.env** creado y lo modificaremos de acuerdo a las

credenciales y opciones que necesitemos en el entorno que estamos configurando.

```
/var/www/ddaw-example/html/.env
APP_ENV=production
APP_DEBUG=false
APP_KEY=b809vCwvtawRbsG0BmPltWgnlXQypSKf
APP_URL=http://example.com
DB_HOST=127.0.0.1
DB_DATABASE=laravel
DB_USERNAME=laraveluser
DB_PASSWORD=password
```



El valor de la variable `APP_KEY` proporciona una clave de cifrado para las sesiones y otros artefactos de la aplicación. Podemos **generarla** mediante la **herramienta artisan** que nos facilita el propio framework.

```
$ php artisan key:generate
```

Nota*: El propio comando, modificará el fichero `.env` con la nueva clave generada

Actividad 3 (0.5 puntos)

- Consulta la siguiente [documentación oficial](#) del framework y explica para que se utilizan las variables `APP_ENV`, `APP_DEBUG`, `APP_KEY` y `APP_URL`. ¿Qué valores podemos especificar para la variable `APP_ENV`? ¿Qué funcionalidad nos proporcionan?

4.3 Configuración del server block

Configuraremos ahora un nuevo **server block** para que se encargue de publicar la nueva aplicación creada. Solo deberemos asegurarnos de que el modificador `default_server` de la directiva `listen` aparezca en un único server block de **nginx**. En caso contrario, el servidor **nginx** no se inicializará.

```
server {
    listen 80;
    listen [::]:80;
    ...
    root /var/www/ddaw-example/html/myNewApp/public
    index index.php index.html index.htm
    server_name ddaw.example.com www.ddaw.example.com;
    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }
    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php7.4-fpm.sock;
    }
}
```

A continuación pasamos a detallar algunas de las directivas utilizadas en la definición del server root anterior.

- **try_files:** la directiva `try_files` comprueba que el recurso al que se intenta acceder exista de forma explícita (**fichero + path**). Cuando hacemos uso de las **url amigables** el fichero al que se intenta acceder no existe como tal, si no que todo el tráfico pasa por un **controlador frontal**, en el caso de laravel **index.php** que se encargará de pasar las peticiones al controlador correspondiente. Por lo que se incluye el path a ese controlador.

```
#Intenta comprobar que el recurso existe con back slash #(/), en
#caso contrario lo redirige al controlador frontal que se
#encargará de procesar las urls amigables
try_files $uri $uri/ /index.php?$query_string;
```

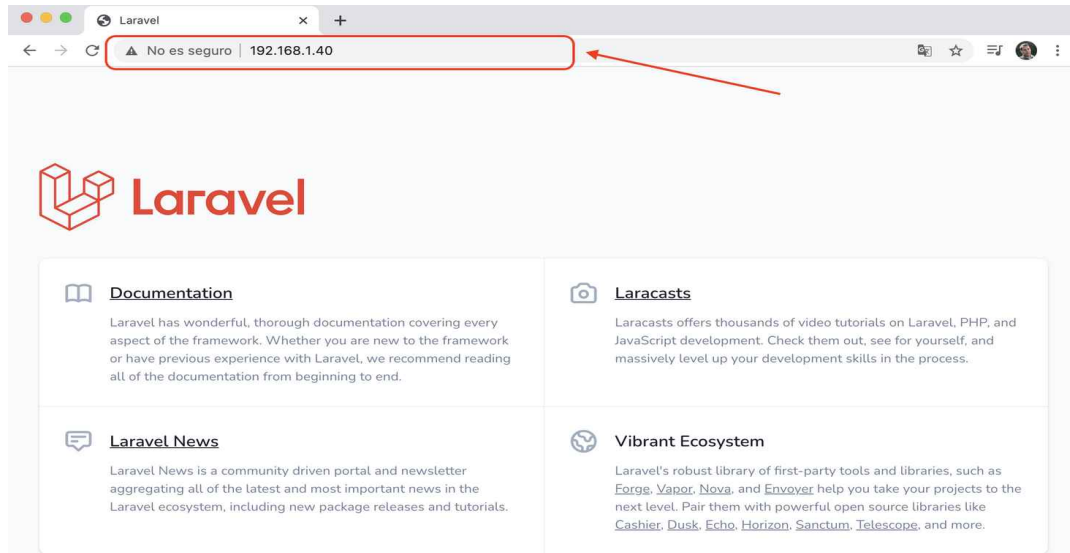
- **root:** El **document root** deja de ser el directorio raíz donde hemos desplegado la aplicación y se convierte en la carpeta `/public` del proyecto.

```
#Establece como documentRoot la carpeta public de la aplicación.
root /var/www/ddaw-example/html/myNewApp/public
```



En la [documentación oficial](#) de laravel, podemos encontrar un archivo optimizado para la configuración del **server block** de nginx que podemos utilizar en nuestro proyecto.

Si todo ha ido bien, cuando accedamos desde un navegador a la IP de nuestro servidor, podremos ver la página de bienvenida de Laravel.



En caso de que Laravel nos indique mensajes de error, deberemos revisar los pasos anteriores, asegurándonos de que los **ficheros** de la aplicación tienen los **permisos correctos** y que se han **establecido las variables** de entorno en los ficheros **.env** del proyecto.

4.4 Trabajo a realizar

Actividad 4 (3 puntos)

- Utilizando el **entorno LEMP**, creado en la **práctica anterior**, despliega el [siguiente proyecto](#) teniendo en cuenta las siguientes especificaciones.
 - **Entorno:** development
 - **Nombre de dominio:** dev.quickstart.ddaw.es
 - **Directorio de despliegue:** /var/www/002-ddaw-es-quickstart-dev/html
 - **Usuario del SO para el despliegue de la aplicación:** dev-ddaw. Bázate en este [link](#)

- Para llevar a cabo el despliegue del proyecto deberás: instalar composer, instalar las dependencias del framework Laravel y disponer de un usuario específico de la base de datos con permisos DML y DDL. Puedes utilizar el siguiente [link](#) como una guía.
- Lleva a cabo la **optimización** indicada en el **punto 3.2** para la **carga de dependencias** de la aplicación con composer. ¿Qué contiene el fichero `vendor/composer/autoload_classmap.php`?

5. Referencias

- Digital Ocean . How to deploy a laravel application with Nginx on Ubuntu 18.04. "<https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-laravel-with-lemp-on-ubuntu-18-04>".
- Php.net. Documentación oficial PHP. "<https://www.php.net/manual/es/index.php>"
- Laravel. Documentación oficial. "<https://laravel.com/docs/7.x>"
- Composer. Documentacion oficial. "<https://getcomposer.org/>"