# «Como instalar y configurar Laravel en un entorno LEMP en Ubuntu 18.04» por Erika Heidi

- **En castellano:** Este artículo es una traducción del inglés al castellano, publicado bajo licencia (en idioma inglés) «Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) » escrito por Erika Heidi, publicado en línea por la empresa de alojamiento de máquinas virtuales DigitalOcean. El título original en idioma inglés es «***How to Install and Configure Laravel with LEMP on Ubuntu 18.04***» y hemos creado una copia en Wayback Machine para prevenir un posible enlace roto a futuro.

## Introducción

Laravel es un *entorno de trabajo* escrito en lenguaje PHP bajo licencia de *código abierto* que ofrece un conjunto de herramientas y recursos para construir modernas aplicaciones en PHP. Con un completo *ecosistema* aprovechando sus características integradas, la popularidad de Laravel ha crecido rápidamente en los últimos años, con muchos desarrolladores adoptándolo como su entorno de trabajo preferido para un proceso de desarrollo simplificado.

En este artículo usted instalará y configurará una nueva aplicación de Laravel en un servidor Ubuntu versión 18.04, usando **Composer** (software) para descargar y administrar las librerías del entorno de trabajo. Cuando usted finalice tendrá una aplicación demostrativa totalmente funcional, la cual obtiene información desde una base de datos **MySQL**.

## Prerrequisitos

Para completar esta guía, usted primero necesita realizar las siguientes tareas en vuestro servidor Ubuntu 18.04:

- Crear un usuario con derechos de administrador ( *sudo* ) y habilitar el muro de fuego de Ubuntu, **UFW**. Para configurar esto usted puede seguir el procedimiento «*Configuración inicial de un servidor con Ubuntu 18.04*».
- Instalar un conjunto **LEMP**. Si usted no ha hecho esto aún, puede seguir lo descrito en este enlace «*Como instalar Nginx, MySQL y PHP en Ubuntu 18.04*».
- Instalar **Composer (software)**. Usaremos **Composer** para instalar Laravel y sus librerías. Usted puede instalar **Composer** de acuerdo a esta guía acerca de «*Como instalar Composer en Ubuntu 18.04*».

## Paso 1 — Instalando los módulos PHP requeridos

Antes de usted poder instalar Laravel necesita unos cuantos módulos **PHP** que son requeridos por el entorno de trabajo. Usaremos **apt** para instalar los siguientes módulos: **php-mbstring**, **php-xml** y **php-bcmath**. Esas extensiones **PHP** proporcionan soporte adicional para manejar la codificación o juego de caracteres, XML y las matemáticas de precisión.

Si esta es la primera vez usando **apt** en esta sesión de trabajo, usted primero debería ejecutar el parámetro **update** para actualizar el caché del administrador de paquetes de software:

```
$ sudo apt update
```

Ahora usted puede instalar los paquetes requeridos con:

```
$ sudo apt install php-mbstring php-xml php_bcmath
```

Su sistema ahora está listo para ejecutar la instalación de **Laravel** por medio de **Composer**, pero antes usted necesitará una base de datos para su aplicación.

## Paso 2 — Creación de una base de datos para la aplicación

Para demostrar una instalación básica de **Laravel** y su utilización, crearemos una aplicación de muestra denominada *travel list* (lista de viaje) para mostrar una serie de lugares a los cuales un usuario le gustaría viajar y una lista de sitios que haya visitado. Esto puede ser almacenado en una simple tabla llamada **places** ("lugares" o "sitios") con un campo para las ciudades o regiones el cual llamaremos **name** ("nombre") y, además, otro campo para registrar como *visitado* o *no visitado,* el cual llamaremos **visited** ("visitado"). De manera adicional incluiremos un campo llamado **id** (abreviatura de *identifier* o "identificador") para identificar dicha entrada de manera singular y única.

Para conectar a la base de datos desde la aplicación Laravel crearemos un usuario a propósito y exclusivo para ello en **MySQL** y le otorgaremos a este usuario completos privilegios sobre la base de datos que llamaremos **travel_list**.

Para comenzar inicie usted sesión en la línea de comandos con **MySQL** con credenciales de *usuario raíz* o **root**:

```
mysql> CREATE DATABASE travel_list;
```

Ahora usted puede crear un nuevo usuario y garantizarle todos los privilegios en la base de datos personalizada que acaba de crear. En este ejemplo estamos creando un usuario llamado **travel_user** con la contraseña password, aunque debe usted cambiar a una contraseña segura de su elección:

```
mysql> GRANT ALL ON travel_list.* TO 'travel_user
'@'localhost' IDENTIFIED BY 'password' WITH GRANT OPTION;
```

Todo esto le proporcionará al usuario **travel_user** completos privilegios sobre la base de datos **travel_list**, a la vez que previene que modifique y/o adicione otras bases de datos en el servidor.

A continuación salga de la interfaz de **MySQL**:

```
mysql> exit
```

Puede vd. ahora probar si el nuevo usuario tiene los permisos apropiados para conectar a la consola de **MySQL**, conecte de nuevo pero con la identidad de dicho usuario:

```
$ mysql -u travel_user -p
```

Nótese el parámetro **-p** en este comando, el cual solicitará la contraseña creada cuando fue creado el usuario **travel_user**. Después de registrarse en la ventana terminal con **MySQL**, verifique que usted tiene acceso a la base de datos **travel_list**:

```
mysql> SHOW DATABASES;
```

Este comando mostrará por pantalla lo siguiente:

*Ejecución y visualización:*

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| travel_list        |
+--------------------+
2 rows in set (0.01 sec)
```

Ahora agregue una tabla llamada **places** en la base de datos **travel_list**. Desde la terminal con MySQL ordene usted la siguiente sentencia:

```
mysql> CREATE TABLE travel_list.places (
mysql>   id INT AUTO_INCREMENT,
mysql>   name VARCHAR(255),
mysql>   visited BOOLEAN,
mysql>   PRIMARY KEY(id)
mysql> );
```

Ahora rellene la tabla **places** con algunos datos de ejemplo:

```
mysql>    INSERT INTO travel_list.places (name, visited)
mysql>    VALUES ("Tokyo", false),
mysql>    ("Budapest", true),
mysql>    ("Nairobi", false),
mysql>    ("Berlin", true),
mysql>    ("Lisbon", true),
mysql>    ("Denver", false),
mysql>    ("Moscow", false),
mysql>    ("Olso", false),
mysql>    ("Rio", true),
mysql>    ("Cincinati", false),
mysql>    ("Helsinki", false);
```

Para verificar que los datos fueron agregados sin problema alguno ejecute la siguiente sentencia:

```
mysql> SELECT * FROM travel_list.places;
```

Verá algo parecido a esto:

*Ejecución y visualización:*

```
+----+-----------+---------+
| id | name      | visited |
+----+-----------+---------+
|  1 | Tokyo     |       0 |
|  2 | Budapest  |       1 |
|  3 | Nairobi   |       0 |
|  4 | Berlin    |       1 |
|  5 | Lisbon    |       1 |
|  6 | Denver    |       0 |
|  7 | Moscow    |       0 |
|  8 | Oslo      |       0 |
|  9 | Rio       |       1 |
| 10 | Cincinati |       0 |
| 11 | Helsinki  |       0 |
+----+-----------+---------+
11 rows in set (0.00 sec)
```

Después de confirmar de que usted tiene datos válidos en su tabla de pruebas, salga de la terminal de comandos MySQL:

```
mysql> exit
```

Ahora usted se encuentra preparado para crear la aplicación y configurar así para que conecte a la nueva base de datos.

## Paso 3 — Creando una nueva aplicación en Laravel

Ahora usted hará una nueva aplicación para **Laravel** por medio del comando **composer create-project**. Este comando de **Composer** generalmente es usado para crear nuevas aplicaciones basadas en entornos de trabajo con librerías previamente creadas para tal efecto y con el fin de usarlas en sistemas de manejo de contenido.

En toda esta guía usaremos **travel_list** como una aplicación de ejemplo, pero usted convertir esto a lo que necesite o desee. El programa **travel_list** mostrará una lista de ubicaciones traídas desde una base de datos local manejada por **MySQL**, con el propósito de demostrar la configuración básica de **Laravel** y demostrar que tiene usted conexión con dicha base de datos.

Primero vaya usted a su carpeta de usuario:

```
$ cd ~
```

El próximo comando creará un directorio de **travel_list** el cual tiene un armazón de **Laravel** basada en los valores por defecto:

```
$ composer create-project --prefer-dist laravel/laravel travel_list
```

Obtendrá algo como esto:

*Ejecución y visualización:*

```
Installing laravel/laravel (v5.8.17)
  - Installing laravel/laravel (v5.8.17): Downloading (100%)
Created project in travel_list
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 80 installs, 0 updates, 0 removals
  - Installing symfony/polyfill-
ctype (v1.11.0): Downloading (100%)
  - Installing phpoption/phpoption (1.5.0): Downloading (100%)
  - Installing vlucas/phpdotenv (v3.4.0): Downloading (100%)
```

```
  - Installing symfony/css-selector (v4.3.2): Downloading (100%)
...
```

Cuando la instalación finalice, acceda a la carpeta de la aplicación y corra el comando **artisan** para comprobar que todos los componentes fueron agregados con éxito:

```
$ cd travel_list
$ php artisan
```

Observará esta similitud:

```
Laravel Framework 5.8.29

Usage:
  command [options] [arguments]

Options:
  -h, --help            Display this help message
  -q, --quiet           Do not output any message
  -V, --version         Display this application version
      --ansi            Force ANSI output
      --no-ansi         Disable ANSI output
  -n, --no-interaction  Do not ask any interactive question
      --env[=ENV]       The environment the command should run under
  -v|vv|vvv, --verbose  Increase the verbosity of messages: 1 for normal
output, 2 for more verbose output and 3 for debug

(...)
```

Esta salida por pantalla reafirma que los archivos de la aplicación están en su sitio, y las herramientas por línea de comandos trabajan como se espera. Sin embargo aún necesitamos configurar el programa para con la base de datos y unos cuantos detalles más.

## Paso 4 — Configurando a Laravel

Los archivos de configuración de Laravel se encuentran en un directorio llamado **config**, dentro del directorio principal de la aplicación. Además, cuando usted instala **Laravel** con **Composer**, automáticamente se crea un *archivo de entorno*. Este archivo contiene configuraciones específicas para el entorno actual en el que se ejecuta el programa, y tendrá prioridad sobre los valores establecidos en los archivos de configuración normales ubicados en el directorio de configuración. Cada instalación en un nuevo entorno requiere un archivo de entorno personalizado para definir elementos como la configuración de conexión con la base de datos, las opciones de depuración, la URL de la aplicación, entre otros elementos que pueden variar según el entorno en el que se ejecuta la aplicación.

**Advertencia:** el fichero de configuración contiene información delicada acerca de vuestro servidor, incluyendo credenciales para conectar con la base de datos y *llaves de seguridad.* Por esa razón usted nunca debe compartir este archivo de manera pública.

Ahora editaremos el fichero **.env** usando el editor de texto de su preferencia. Acá usaremos el editor **nano**:

Aunque hay muchas variables de configuración en este archivo, no es necesario configurarlas todas ahora. La siguiente lista contiene una descripción general de las variables que requieren atención inmediata:

- **APP_NAME**: nombre del programa, usado para mensajes y notificaciones.
- **APP_ENV**: entorno actual de la aplicación.
- **APP_KEY**: utilizada para generar *salts* y *hashes*, esta clave única se genera automáticamente cuando instalamos **Laravel** con **Composer** así que no es necesario cambiarla.
- **APP_DEBUG**: para elegir si mostrar información de depuración del lado del cliente (usuario final).
- **APP_URL**: URL base para el programa, usada para generar los enlaces de toda la aplicación.
- **DB_DATABASE**: nombre de la base de datos.

- **DB_USERNAME**: nombre del usuario para conectar a la base de datos.
- **DB_PASSWORD**: contraseña para conectar a la base de datos.

De manera predeterminada esos valores vienen listos para un entorno de desarrollo local basado en **Homestead**, un prepaquete para **Vagrant** hecho por **Laravel**. Cambiaremos esos valores para reflejar el conjunto de ajustes locales de nuestra aplicación de ejemplo.

En el caso que usted esté instalando **Laravel** en un ambiente de **desarrollo** o de **prueba**, puede dejar la opción **APP_DEBUG** habilitada, así le dará importante información al depurar mientras prueba la aplicación desde un navegador web. Siendo esto así, el valor **APP_ENV** debe ser cambiado a alguno de estos dos valores: **development** (desarrollo) o **testing** (prueba).

En otra opción, si usted está instalando **Laravel** en un entorno *de producción* usted deber deshabilitar la opción **APP_DEBUG** porque esta el mostrará información delicada al usuario acerca de la aplicación. El siguiente archivo **.env** coloca nuestro ejemplo en modo de **aplicación en desarrollo** (no producción):

**Nota:** la variable **APP_KEY** contiene una clave que es única y que fue autogenerada cuando usted instaló **Laravel** con **Composer**. No necesita cambiar este valor. Si desea generar una nueva clave segura puede echar mano del comando **php artisan key:generate**.

Archivo **/var/www/travel_list/.env**

```
APP_NAME=TravelList
APP_ENV=development
APP_KEY=APPLICATION_UNIQUE_KEY_DONT_COPY
APP_DEBUG=true
APP_URL=http://domain_or_IP

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
```

```
DB_PORT=3306
DB_DATABASE=travel_list
DB_USERNAME=travel_user
DB_PASSWORD=password

...
```

Coloque el valor de sus propias variables en cada renglón. Cuando haya usted finalizado guarde y cierre el archivo para mantener fijos sus cambios. Si usted está usando el editor de texto **nano**, puede hacer esto con las teclas CTRL + X, luego pulse Y y a continuación Enter para confirmar.

Su aplicación en Laravel está ahora lista, pero necesitamos ajustar el servidor web a fin de que sea capaz de acceder a él por medio de un navegador web. En el próximo paso configuraremos a **Nginx** para que de servicio a su programa hecho sobre **Laravel**.

## Paso 5 — Configurando a Nginx

Hemos instalado **Laravel** en un directorio local de su carpeta de usuario en el servidor remoto, y mientras que esto trabaja muy bien en ambientes de desarrollo local, no es una práctica recomendada para servidores web que están expuestos al público en Internet. Moveremos la carpeta de la aplicación hacia **/var/www**, el cual es la ubicación normal para las aplicaciones web que corren bajo **Nginx**.

Primero use el comando **mv** para mover el directorio de la aplicación junto con todo su contenido a **/var/www/travel_list**:

```
$ sudo mv ~/travel_list /var/www/travel_list
```

Ahora necesitamos darle al servidor web acceso como usuario que pueda escribir en las carpetas **cache** y **storage**, lugares donde **Laravel** almacena los ficheros que son generados para la aplicación:

sa

```
$ sudo chown -R www-data.www-data /var/www/travel_list/storage
$ sudo chown -R www-data.www-data /var/www/travel_list/bootstrap/cache
```

El siguiente fichero de configuración contiene los ajustes recomendados para las aplicaciones Laravel sobre **Nginx**:

Fichero **/etc/nginx/sites-available/travel_list**

```
server {
    listen 80;
    server_name server_domain_or_IP;
    root /var/www/travel_list/public;

    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Content-Type-Options "nosniff";

    index index.html index.htm index.php;

    charset utf-8;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location = /favicon.ico { access_log off; log_not_found off; }
    location = /robots.txt  { access_log off; log_not_found off; }

    error_page 404 /index.php;

    location ~ \.php$ {
        fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.(?!well-known).* {
        deny all;
    }
}
```

Copie todo este contenido a su archivo **/etc/nginx/sites-avalaible/travel_list** y, si es necesario, ajuste los valores resaltados en color rojo para que coincidan con su propia configuración.

Para activar el nuevo fichero de configuración de anfitrión, deberá crear un enlace simbólico a **travel_list** en **sites-enabled**:

```
$ sudo ln -s /etc/nginx/sites-
available/travel_list /etc/nginx/sites-enabled/
```

Si usted tiene otro fichero de anfitrión virtual configurado para el mismo **server_name** utilizado en el anfitrión virtual de **travel_list** , puede que usted necesite desactivar la vieja configuración por medio de la remoción del correspondiente enlace simbólico en **/etc/nginx/sites-enabled/** .

Para confirmar que la configuración hecha no contiene error alguno, puede usar:

```
$ sudo nginx -t
```

Apreciará una vista como esta:

```
$ nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
$ nginx: configuration file /etc/nginx/nginx.conf test is successful
```

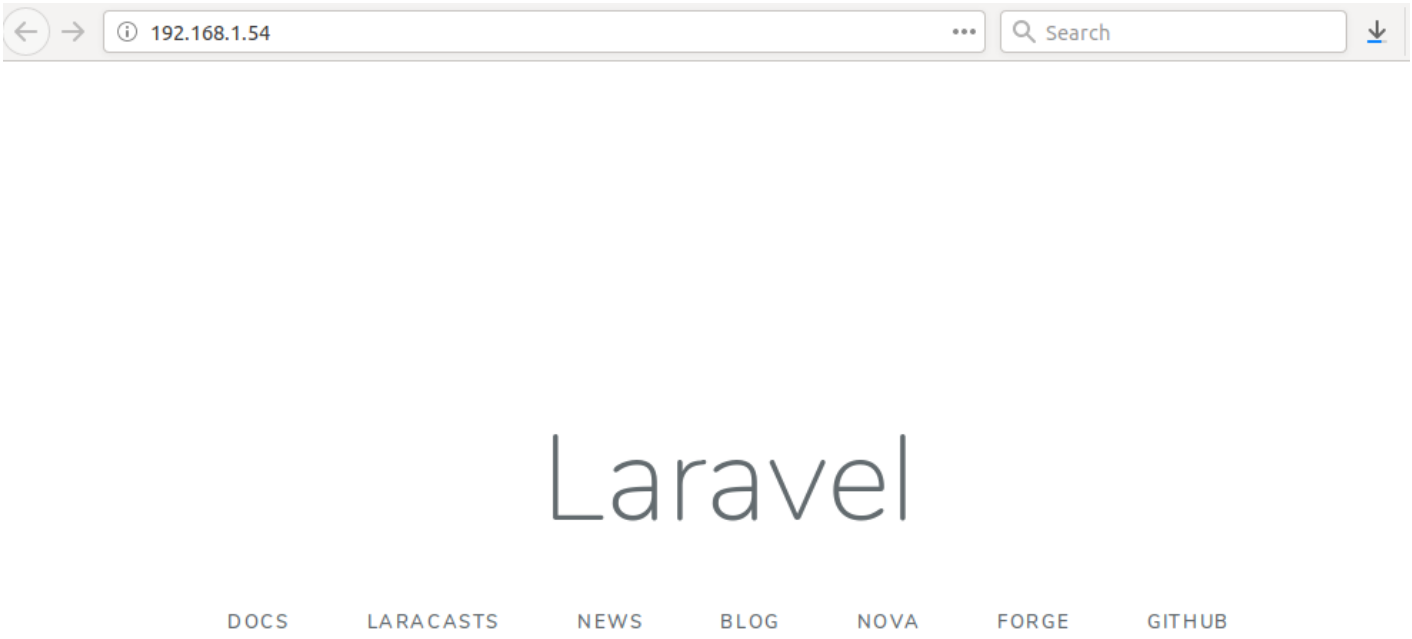
Para que los cambios tomen efecto, reinicie **nginx**:

```
$ sudo systemctl reload nginx
```

Vaya a su navegador web y acceda a la aplicación usaindo el dominio web del servidor o su dirección IP, la cual fue definida por la directiva **server_name** en su archivo de configuración:

```
$ http://server_domain-or_IP
```

Verá lo siguiente, más o menos:




Esto corrobora que su servidor **nginx** está puesto a punto de manera adecuada para servir **Laravel**. Desde este punto usted puede comenzar a construir su aplicación desde el principio de su armazón provisto en la instalación por defecto.

En el siguiente paso modificaremos la ruta principal del programa para consultar información en la base de datos por medio de la fachada **DB** de **Laravel**.

## Paso 6 — Personalizando la página principal

Asumiendo que haya seguido todos los pasos de esta guía hasta el momento, debería de tener una aplicación Laravel en funcionamiento y una tabla de base de datos llamada **places** que

contenga algunos datos de muestra.

Ahora editaremos la ruta principal de la aplicación para consultar la base de datos y devolver los contenidos a la *vista* de aplicación.

Abra el fichero de ruta principal, **/rooutes/web.php**:

Archivo **routes/web.php**