

::: Instalar Servidor Web Completo en Ubuntu para Trabajar con Laravel :::

By Freddy Alcarazo (surflaweb)

<https://github.com/alcarazolabs>

== creatorpart@gmail.com ==

(Actualizado 5 de Agosto del 2021)

Presentación

¡Hola, soy Freddy Alcarazo !

En este manual aprenderás a instalar desde 0 un servidor web en el sistema operativo linux ubuntu 20.0. Vamos a instalar todo lo necesario para correr proyectos del framework php laravel u nodejs.

Lo que instalaremos:

Instalaremos apache, mysql, php, phpmyadmin, composer, crear aplicación laravel, dar permisos, crear virtualhost, editar hosts, crear sistema de autenticación laravel ui, instalar nodejs y npm.

¡¡Adelante!!

Pasos:

1. Instalar Apache2

```
$ sudo apt install apache2
```

Luego de la instalación. Los proyectos estarán dentro de la carpeta html:

```
freddy@freddy-ubuntu:/$ ls /var/www/html/  
index.html
```

Iniciar, restart, recargar detener servicio apache2:

```
sudo systemctl start apache2  
sudo systemctl restart apache2  
sudo systemctl reload apache2
```

```
sudo systemctl stop apache2
```

Confirmar que apache2 esta funcionando:

```
sudo systemctl is-enabled apache2
```

Nota:

Los archivos de configuración de Apache2 se encuentran en el directorio `/etc/apache2` y el archivo de configuración principal es `/etc//etc/apache2/apache2.conf`. Y la raíz del documento predeterminada para almacenar sus archivos web es `/var/www/html/`.

Verificar en el navegador que se puede acceder al localhost o usar ip de la máquina.

2. Instalar base de datos MySQL

```
$ sudo apt-get install mysql-server
```

Los archivos de configuración del servidor mysql están dentro de `/etc/mysql`

Iniciar, restart, detener servicio MySQL:

```
sudo sytemctl start mysql
sudo systemctl restart mysql
sudo systemctl stop mysql
```

Confirmar que apache2 esta funcionando:

```
sudo systemctl is-enabled apache2
```

Establecer contraseña al usuario 'root':

```
$ sudo mysql
```

```
$ ALTER USER 'root'@'localhost' IDENTIFIED WITH  
mysql_native_password BY 'password';
```

```
$ exit;
```

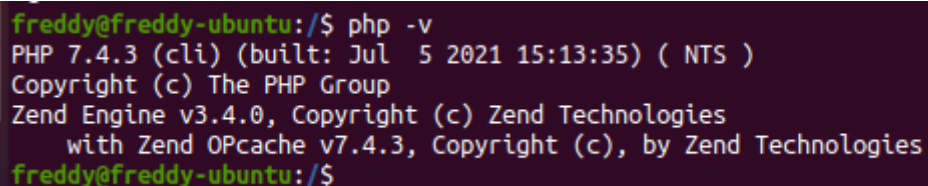
//Probar loguearse luego de cambiar la contraseña:

```
$ sudo mysql -u root -p
```

3. Instalar PHP 7.4

La versión actual de Laravel 8 trabaja con la versión de php ≥ 7.4 por lo tanto se instalar php 7.4:

```
$ sudo apt install -y php7.4-gd php7.4-mbstring php7.4-xml php-zip  
libapache2-mod-php7.4 php7.4-mysql  
$ php -v
```

A terminal window with a dark purple background. The text is as follows:

```
freddy@freddy-ubuntu:/$ php -v  
PHP 7.4.3 (cli) (built: Jul  5 2021 15:13:35) ( NTS )  
Copyright (c) The PHP Group  
Zend Engine v3.4.0, Copyright (c) Zend Technologies  
    with Zend OPcache v7.4.3, Copyright (c), by Zend Technologies  
freddy@freddy-ubuntu:/$
```

Los archivos de configuración de PHP se encuentran dentro de /etc/php/7.4

Además, dependiendo de su proyecto, es posible que desee instalar algunas extensiones PHP requeridas por su aplicación. Puede buscar una extensión PHP como se muestra:

```
$ sudo apt-cache search php | grep php-  
#show all php packages
```

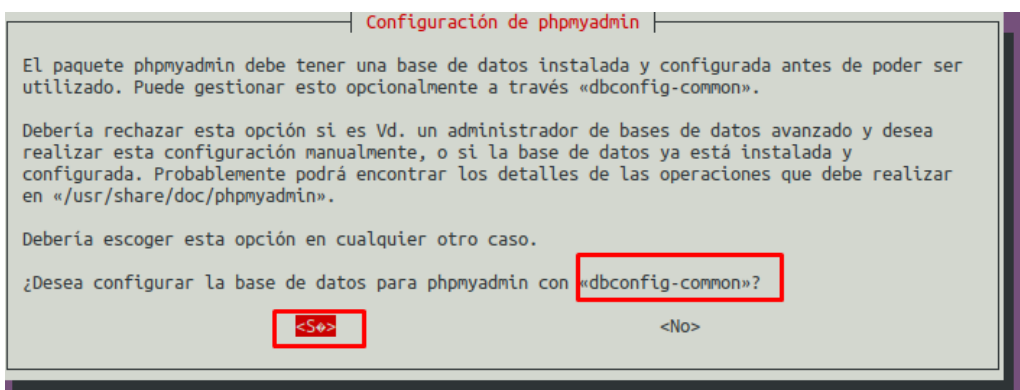
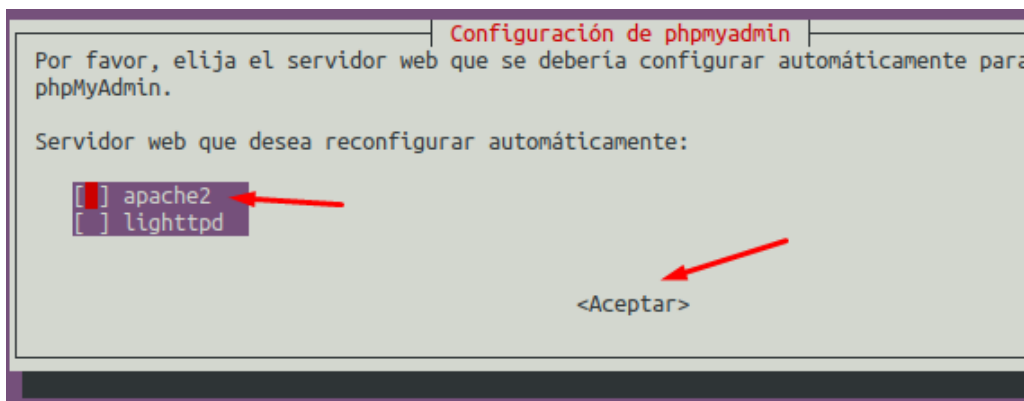
Después de encontrar la extensión, puede instalarla. Por ejemplo, estoy instalando módulos PHP para el caché en memoria de Redis y la herramienta de compresión Zip (Esto es opcional).

```
$ sudo apt instalar php-redis php-zip
```

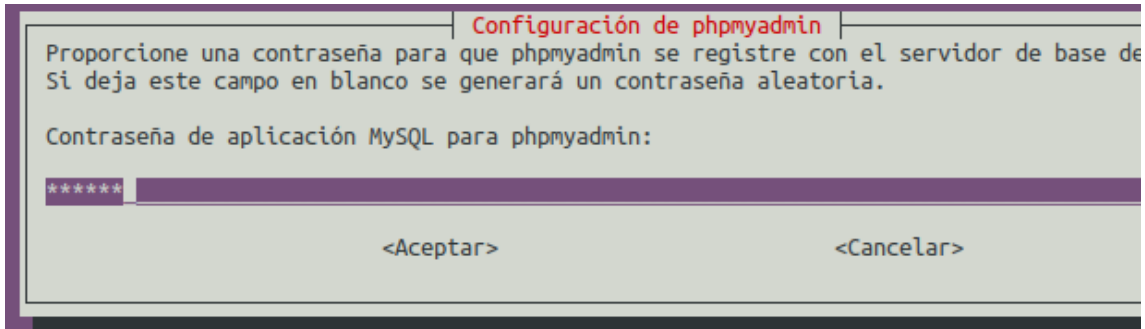
4. Instalar phpMyAdmin

Al momento de hacer esta instalación la versión de phpmyadmin es compatible con php 7.4, si se instala phpmyadmin y luego tenemos problemas solo actualizamos la versión (<https://youtu.be/A6uXY2lZQYo?t=504>)

```
$ sudo apt-get install phpmyadmin
```



Luego nos pide una clave, ponemos la misma que usamos para el usuario root:

A screenshot of a terminal window showing the 'Configuración de phpmyadmin' dialog box. The dialog has a title bar with the text 'Configuración de phpmyadmin'. Inside, it says: 'Proporcione una contraseña para que phpmyadmin se registre con el servidor de base de datos. Si deja este campo en blanco se generará un contraseña aleatoria.' Below this, it asks for the 'Contraseña de aplicación MySQL para phpmyadmin:' and shows a text input field with six asterisks. At the bottom, there are two buttons: '<Aceptar>' and '<Cancelar>'.

Configuración de phpmyadmin

Proporcione una contraseña para que phpmyadmin se registre con el servidor de base de datos. Si deja este campo en blanco se generará un contraseña aleatoria.

Contraseña de aplicación MySQL para phpmyadmin:

<Aceptar> <Cancelar>

Una vez que se completa el proceso de instalación, los archivos de configuración de phpMyAdmin se encuentran en /etc/phpmyadmin y su archivo de configuración principal es /etc/phpmyadmin/config.inc.php. Otro archivo de configuración importante es /etc/phpmyadmin/apache.conf, que se utiliza para configurar Apache2 para que funcione con PhpMyAdmin.

A continuación, debe configurar Apache2 para que sirva al sitio phpMyAdmin. Ejecute el siguiente comando para vincular simbólicamente el archivo /etc/phpmyadmin/apache.conf a /etc/apache2/conf-available/phpmyadmin.conf

```
$ sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf-available/phpmyadmin.conf
```

```
$ sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf-available/phpmyadmin.conf
```

Luego habilite los archivos de configuración phpmyadmin.conf para Apache2 y reinicie el servicio Apache2 para aplicar los cambios recientes.

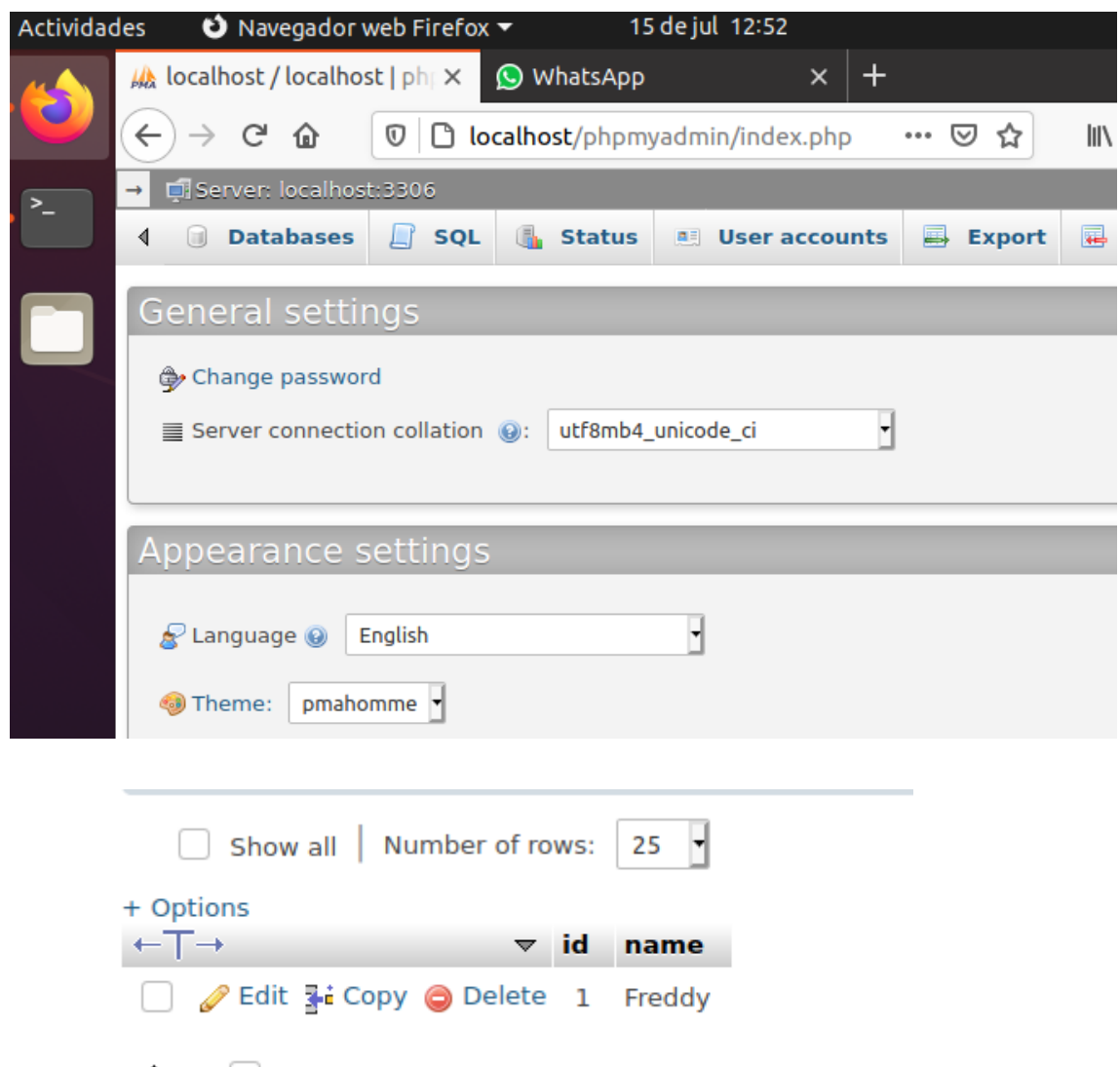
Luego habilite los archivos de configuración phpmyadmin.conf para Apache2 y reinicie el servicio Apache2 para aplicar los cambios recientes.

```
$ sudo a2enconf phpmyadmin.conf
```

```
$ sudo systemctl reload apache2
```

```
freddy@freddy-ubuntu:/$ sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf-available/phpmyadmin.conf
freddy@freddy-ubuntu:/$ sudo a2enconf phpmyadmin.conf
Enabling conf phpmyadmin.
To activate the new configuration, you need to run:
systemctl reload apache2
freddy@freddy-ubuntu:/$ sudo systemctl reload apache2
freddy@freddy-ubuntu:/$
```

y listo! Ya tenemos phpmyadmin funcionando:



5. Instalar Composer

Con composer se podrá instalar modulos a laravel, crear el proyecto, actualizar dependencias, ejecutar comandos etc.

Pasos:

1. Instalar curl: `sudo apt-get install curl`
2. Ejecutar los siguientes comandos:

Antes ejecutar el primer comando, debemos de ubicarnos dentro de la carpeta Descargas del usuario:

Porque en la raíz no debemos de descargar archivos.

```
freddy@freddy-ubuntu:~/Descargas$ pwd  
/home/freddy/Descargas  
freddy@freddy-ubuntu:~/Descargas$
```

Seguir los siguientes pasos:

Paso 1: Dependencias adicionales

Además de las dependencias que ya deben estar incluidas en su sistema de Ubuntu 20.04, como `git` y `curl`, Composer requiere `php-cli` para ejecutar las secuencias de comandos PHP en la línea de comandos y `unzip` para extraer los archivos comprimidos. Instalaremos estas dependencias ahora.

- `sudo apt install php-cli unzip`

Paso 2: Descargar e instalar Composer

- `sudo curl -sS https://getcomposer.org/installer -o composer-setup.php`

```
freddy@freddy-ubuntu:~/Descargas$ ls  
composer-setup.php  
freddy@freddy-ubuntu:~/Descargas$
```

A continuación, verificaremos que el instalador descargado coincida con el hash SHA-384 para el instalador más reciente disponible en la página Composer Public Keys/Signatures. A fin de facilitar el paso de verificación, puede utilizar el siguiente comando para obtener de forma programática el hash más reciente de la página de Composer y almacenarlo en una variable de shell:

- `HASH=`curl -sS https://composer.github.io/installer.sig``

Si desea verificar el valor obtenido, puede ejecutar lo siguiente:

- `echo $HASH`

```
freddy@freddy-ubuntu:~/Descargas$ echo $HASH
756890a4488ce9024fc62c56153228907f1545c228516cbf63f885e036d37e9a59d27d63f46af1d4d07ee0f76181c7d3
freddy@freddy-ubuntu:~/Descargas$
```

Ahora, ejecute el siguiente código PHP, como se indica en la página de descarga de Composer, para verificar que la secuencia de comandos de instalación se pueda ejecutar de forma segura:

```
php -r "if (hash_file('SHA384', 'composer-setup.php') === '$HASH') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
```

```
freddy@freddy-ubuntu:~/Descargas$ php -r "if (hash_file('SHA384', 'composer-setup.php') === '$HASH') {
echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PH
P_EOL;"
Installer verified
freddy@freddy-ubuntu:~/Descargas$
```

Si aparece el mensaje `Installer corrupt`, tendrá que volver a descargar la secuencia de comandos de instalación y verificar nuevamente si utilizó el hash correcto. Luego, repita el proceso de verificación. Cuando cuente con un instalador verificado, podrá continuar.

Para instalar `composer` de manera global, utilice el siguiente comando que lo descargará e instalará en todo el sistema como un comando llamado `composer`, en `/usr/local/bin`:

- `sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer`

Verá un resultado similar a este:

```
freddy@freddy-ubuntu:~/Descargas$ sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer
[sudo] contraseña para freddy:
All settings correct for using Composer
Downloading...

Composer (version 2.1.3) successfully installed to: /usr/local/bin/composer
Use it: php /usr/local/bin/composer

freddy@freddy-ubuntu:~/Descargas$
```

Y listo!

Prueba:

\$ composer

Créditos: <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-composer-on-ubuntu-20-04-es>

6. Crear una aplicación laravel con composer

A continuación, se crea un proyecto laravel con la versión 8:

Primero nos ubicamos dentro de la carpeta “html” del servidor apache:

\$ cd /var/www/html

Dentro de la carpeta “html” se crearan los proyectos. Pero antes debemos de darle permisos de escritura al usuario “freddy” en la carpeta “html”, no vamos a usar directamente chmod 777 a la carpeta html por que seria peligroso.

Ahora cambiamos de propietario a la carpeta html con el comando chown

\$ sudo chown tuusuario:www-data /var/www/html

Luego debemos de agregar nuestro usuario al grupo de Ubuntu www-data

\$ sudo usermod -a -G APACHEGROUP MYUSER

```
freddy@freddy-ubuntu:/var/www/html$ sudo usermod -a -G www-data freddy
freddy@freddy-ubuntu:/var/www/html$
```

Listo ahora ya deberíamos de poder crear un proyecto laravel:

\$ composer create-project laravel/laravel laravelapp 8.0

```
freddy@freddy-ubuntu:/var/www/html$ composer create-project laravel/laravel laravelapp 8.0
Creating a "laravel/laravel" project at "./laravelapp"
Installing laravel/laravel (v8.0.0)
- Downloading laravel/laravel (v8.0.0)
- Installing laravel/laravel (v8.0.0): Extracting archive
Created project in /var/www/html/laravelapp
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 104 installs, 0 updates, 0 removals
- Locking asm89/stack-cors (v2.0.3)
- Locking brick/math (0.9.2)
- Locking doctrine/inflector (2.0.3)
- Locking doctrine/instantiator (1.4.0)
- Locking doctrine/lexer (1.2.1)
- Locking dragonmantank/cron-expression (v3.1.0)
```

Si composer se queda trabado y no descarga nada y al final sale un problema de timeout por problemas de conexión ejecutar el siguiente comando:

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

Así no se priorizara ipv6

<https://www.phpcentral.com/pregunta/1802/composer-en-linux-no-descarga-paquetes-ni-conecta-connection-timed-out>

Sin embargo a pesar de tener el proyecto ya creado aún falta definir permisos de escritura a ciertos archivos para ello vamos a ejecutar el siguiente script “fix.sh” que hará todo el trabajo desde agregando el usuario al grupo www-data de Ubuntu a darle permisos a la carpeta storage de laravel y otros archivos:

***Copiar el siguiente código/descargarlo ver link mas abajo, y guardarlo como “fix.sh” dentro del proyecto laravel, editar el usuario:**

```
#!/usr/bin/env bash

#By juchipilo

#Configure these two variables
MYUSER="freddy"
APACHEGROUP="www-data"


SCRIPTPATH=`pwd -P`
BOOTSTRAP="$SCRIPTPATH/bootstrap/"
BOOTSTRAPCACHE="$SCRIPTPATH/bootstrap/cache/"
STORAGE="$SCRIPTPATH/storage"
LOGS="$STORAGE/logs"


#add my user to the web server group
sudo usermod -a -G ${APACHEGROUP} ${MYUSER}


#make www-data own everything in the directory
sudo chown -R ${MYUSER}:${APACHEGROUP} ${SCRIPTPATH}


#change permissions on files to 644
sudo find ${SCRIPTPATH} -type f -exec chmod 0644 {} \;


#change permissions on directories to 755
sudo find ${SCRIPTPATH} -type d -exec chmod 0755 {} \;


#if i have any bash scripts in there, make them executable
sudo find ${SCRIPTPATH} -type f -iname "*.sh" -exec chmod +x {} \;
```

```
if test ! -d "$BOOTSTRAPCACHE"
then
    MKDIRCOMMAND=`mkdir -p ${BOOTSTRAPCACHE}`
    $MKDIRCOMMAND
fi

chown ${MYUSER}:${APACHEGROUP} ${BOOTSTRAP}

chown ${MYUSER}:${APACHEGROUP} ${BOOTSTRAPCACHE}

chmod 0775 ${BOOTSTRAPCACHE}

if [ -f ${SCRIPTPATH}/bootstrap/cache/services.php ];
then
    chmod 0664 ${SCRIPTPATH}/bootstrap/cache/services.php
fi

if [ ! -d ${SCRIPTPATH}/storage ]; then
    mkdir -p ${SCRIPTPATH}/storage
fi

STORAGEFIXCOMMAND=`chown -R ${MYUSER}:${APACHEGROUP}
${SCRIPTPATH}/storage`
$STORAGEFIXCOMMAND
```

```
chmod -R 0775 ${STORAGE}
```

```
echo 'Permisos establecidos correctamente'
```

Descarga script: <https://pastebin.com/KbLz9a4Q>

Opción 2: <https://github.com/alcarazolabs/surflaweb-scripts/blob/main/fix.sh>

(*) Copiar el script y pegarlo dentro del proyecto laravel creado.

Darle permisos de ejecución al script:

```
$ sudo chmod +x fix.sh
```

```
freddy@freddy-ubuntu:/var/www/html/laravelapp$ ls -al |grep fix
-rwxr-xr-x 1 freddy www-data 1367 jul 15 16:50 fix.sh
freddy@freddy-ubuntu:/var/www/html/laravelapp$
```

7. Ejecutar script:

```
$ sudo ./fix.sh
```

```
freddy@freddy-ubuntu:/var/www/html/laravelapp$ sudo ./fix.sh
Permisos establecidos correctamente
```

Luego de haber otorgado los permisos al proyecto generamos la llave con php artisan:

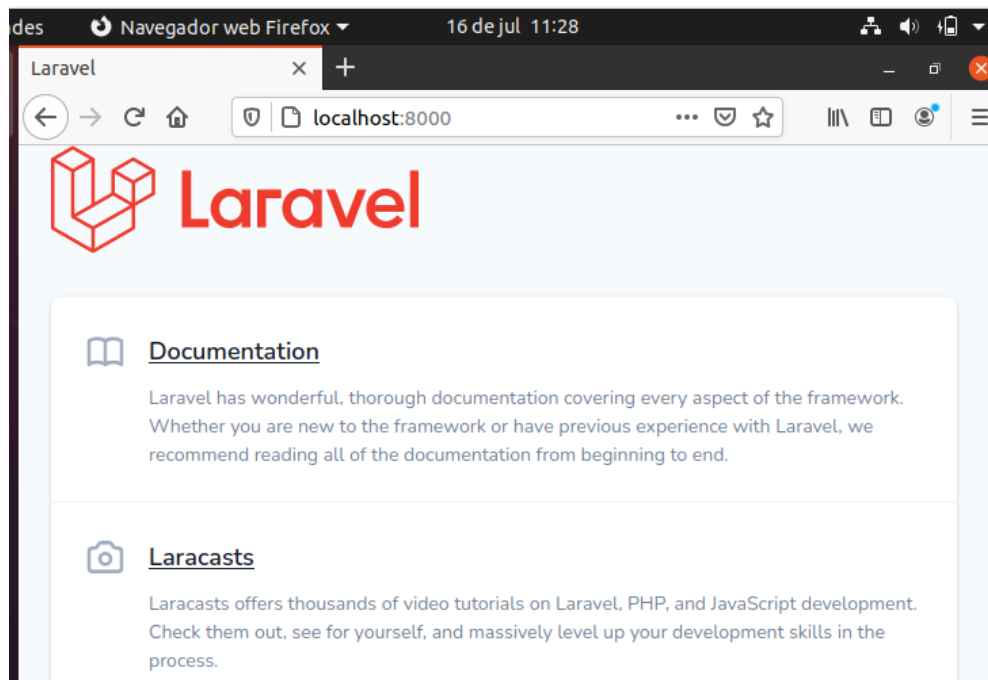
```
$ php artisan key:generate
```

```
freddy@freddy-ubuntu:/var/www/html/laravelapp$ php artisan key:generate
Application key set successfully.
freddy@freddy-ubuntu:/var/www/html/laravelapp$
```

Ejecutar el proyecto:

```
$ php artisan serve
```

Y Listo! La aplicación ya está corriendo:



¿Pero como accederíamos a la aplicación si necesidad de estar ejecutando cada vez que encendemos la máquina el comando `php artisan serve`?

La solución es crear un virtualhost como el siguiente:

```
<VirtualHost *:80>

    ServerName laravelapp

    ServerAlias laravelapp.test

    ServerAdmin creatorpart@gmail.com

    DocumentRoot /var/www/html/laravelapp/public

    <Directory /var/www/html/laravelapp/public>

        Options FollowSymLinks

        AllowOverride All

        Require all granted
```

```
</Directory>

ErrorLog "/var/log/apache2/laravelapp-error_log"

CustomLog "/var/log/apache2/laravelapp-access_log" combined

</VirtualHost>
```

Descargar archivo virtualhost:

<https://github.com/alcarazolabs/surflaweb-scripts/blob/main/laravelapp.test.conf>

Notar que:

- ServerName: Tiene el nombre de la aplicación
- ServerAlias: Un dominio local para probar, si compraron uno y están un servidor como de digitalOcean entonces agreguen ahí el dominio.
- ServerAdmin: El correo del webmaster del app.
- DocumentRoot: Tiene la ruta en donde se encuentra la aplicación que hemos creado, en este caso esta dentro de la carpeta "html".
- ErrorLog: Tiene la ruta en donde guardaremos los logs de error de la aplicación.
- CustomLog: Guarda los log de acceso.

Modifica el VirtualHost de acuerdo al nombre de tu aplicación y nombre de la carpeta.

Para agregar este virtualhost a apache lo haremos de la siguiente manera:

Copiamos el contenido del archivo "000-default.conf" de apache a un nuevo archivo para nuestro virtualhost, este archivo nuevo se creara con el contenido original de "000-default.conf" luego lo actualizaremos por el código anterior. Al hacer la copia de este a un nuevo archivo, a este nuevo archivo le vamos a llamar

“laravelapp.test.conf”. Como vemos tiene el nombre del dominio de la aplicación y termina en .conf (La configuración por defecto de Ubuntu requiere que cada archivo de configuración de Virtual Host termine en .conf)

Ejecutamos el siguiente comando:

```
$ sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/laravelapp.test.conf
```

```
freddy@freddy-ubuntu:/$ sudo cp /etc/apache2/sites-available/000-default.conf /  
etc/apache2/sites-available/laravelapp.test.conf
```

Luego hacemos un “ls” para verificar que el archivo se creo dentro de “sites-available”:

```
freddy@freddy-ubuntu:/$ ls /etc/apache2/sites-available/  
000-default.conf default-ssl.conf laravelapp.test.conf  
freddy@freddy-ubuntu:/$
```

Ahora editamos el archivo creado con “nano”:

```
$ sudo nano /etc/apache2/sites-available/laravelapp.test.conf
```

Y reemplazamos todo su contenido:

```
GNU nano 4.8 /etc/apache2/sites-available/laravelapp.test.conf mo  
<VirtualHost *:80>  
    ServerName laravelapp  
    ServerAlias laravelapp.test  
        ServerAdmin creatorpart@gmail.com  
        DocumentRoot /var/www/html/laravelapp/public  
    <Directory /var/www/html/laravelapp/public>  
        Options FollowSymLinks  
        AllowOverride All  
        Require all granted  
    </Directory>  
    ErrorLog "/var/log/apache2/laravelapp-error_log"  
    CustomLog "/var/log/apache2/laravelapp-access_log" combined  
</VirtualHost>
```

Ctrl+O guardamos y Ctrl+x salimos.

Habilita los nuevos Archivos Virtual Host

Ahora que hemos creado nuestros archivos virtual hosts, debemos habilitarlos. Apache incluye herramientas que nos permiten hacer esto.

Podemos usar la herramienta `a2ensite` para habilitar cada uno de nuestros sitios haciendo esto:

```
sudo a2ensite laravelapp.test.conf
```

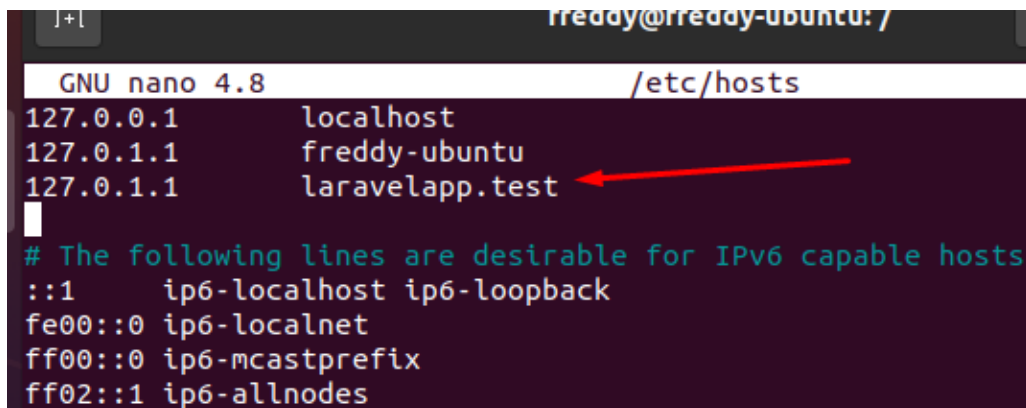
```
freddy@freddy-ubuntu:/$ sudo a2ensite laravelapp.test.conf
Enabling site laravelapp.test.
To activate the new configuration, you need to run:
    systemctl reload apache2
freddy@freddy-ubuntu:/$
```

Ahora debemos reiniciar el servidor ya que no lo dice:

```
$ sudo systemctl reload apache2
```

Aún falta agregar el dominio al archivo “hosts”:

```
$ sudo nano /etc/hosts
```

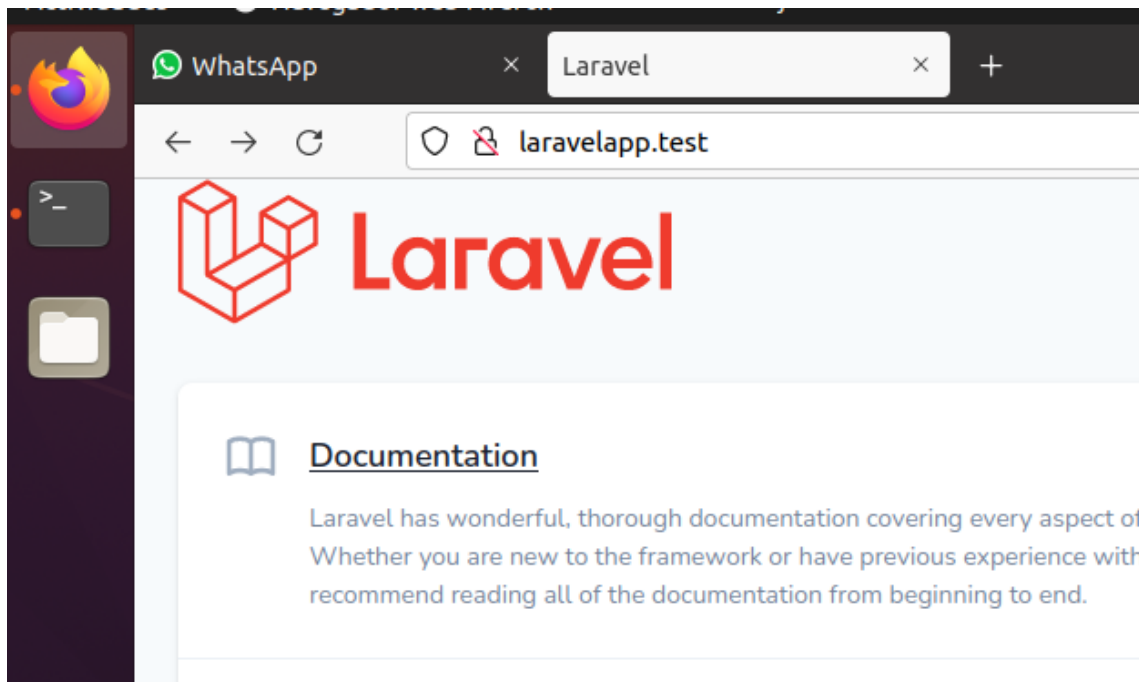


```
freddy@freddy-ubuntu: /
GNU nano 4.8 /etc/hosts
127.0.0.1    localhost
127.0.1.1    freddy-ubuntu
127.0.1.1    laravelapp.test
# The following lines are desirable for IPv6 capable hosts
::1          ip6-localhost ip6-loopback
fe00::0      ip6-localnet
ff00::0      ip6-mcastprefix
ff02::1      ip6-allnodes
```

Guardamos y probamos.

Nota si están en un VPS, en lugar de 127.0.1.1 deben de agregar la IP de su servidor.

Listo funciona:

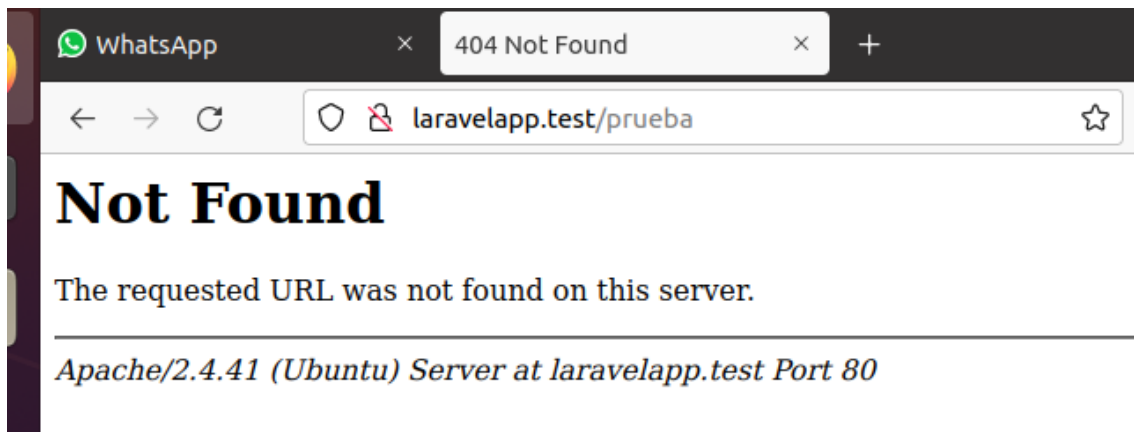


Funciona! Así hacemos lo mismo si creamos otro proyecto dentro de la carpeta “html” le creamos su archivo .conf de virtualhost y luego agregamos el dominio al archivo hosts.

Ahora agregamos una ruta para probar en el archivo web/routes.php

```
16 Route::get('/', function () {  
17     return view('welcome');  
18 });  
19  
20 Route::get('/prueba', function() {  
21  
22     return 'Ruta funcionando!!';  
23 });
```

Sin embargo luego de guardar el archivo y probar en el navegador:



Obtuvimos un 404 not found.

Solución:

Debemos de habilitar el módulo de apache **“rewrite”**. Este módulo `mod_rewrite` permite crear direcciones URL alternativas a las dinámicas generadas por la programación de nuestro sitio web (blog, foro, portal...), de tal modo que sean más legibles y fáciles de recordar.

Ejecutar comando:

```
$ sudo a2enmod rewrite
```

```
laravelapp$ sudo a2enmod rewrite
```

Luego reiniciamos el servidor:

```
$ sudo systemctl restart apache2
```

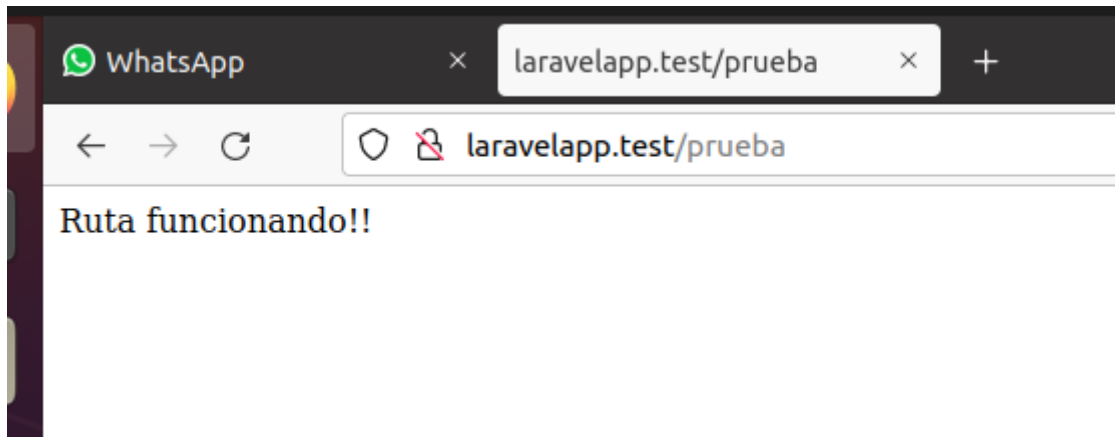
```
/laravelapp$ sudo systemctl restart apache2
```

Luego les recomiendo que ejecuten el comando dentro del proyecto:

```
$ php artisan optimize:clear
```

```
$ composer dump-autoload
```

Con eso sería suficiente para limpiar los caches y rutas nuevas. Probamos:



Y Listo!

Conclusión

Si me has seguido, deberás tener un servidor respondiendo a dos dominios separados. Ahora puedes expandir este procedimiento siguiendo los pasos que hemos llenado arriba para crear Virtual Hosts adicionales.

No hay límite de software en el número de dominios que Apache pueda manejar, así que eres libre de agregar tantos como tu servidor pueda soportar.

Excelente!

Algo más!

Instalar sistema de autenticación laravel/ui

\$ composer require laravel/ui

```

freddy@freddy-ubuntu:/var/www/html/laravelapp$ composer require laravel/ui
Using version ^3.3 for laravel/ui
./composer.json has been updated
Running composer update laravel/ui
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
- Locking laravel/ui (v3.3.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Downloading laravel/ui (v3.3.0)
- Installing laravel/ui (v3.3.0): Extracting archive
Package fzaninotto/faker is abandoned, you should avoid using it. No replacement was suggested.
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: fruitcake/laravel-cors
Discovered Package: laravel/tinker
Discovered Package: laravel/ui
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
74 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
freddy@freddy-ubuntu:/var/www/html/laravelapp$

```

Crear la UI y la autenticación directamente:

```
$ php artisan ui bootstrap --auth
```

```

freddy@freddy-ubuntu:/var/www/html/laravelapp$ php artisan ui bootstrap --auth
Bootstrap scaffolding installed successfully.
Please run "npm install && npm run dev" to compile your fresh scaffolding.
Authentication scaffolding generated successfully.
freddy@freddy-ubuntu:/var/www/html/laravelapp$

```

Listo! Pero ahora necesitaremos del instalador de dependencias “**NPM**” para poder ejecutar “npm install && npm run dev” para poder compilar nuestro sistema de autenticación y además **NodeJs**.

Instalar NodeJs y NPM juntos:

Nodejs para linux

<https://github.com/nvm-sh/nvm>

Nos ubicamos dentro de la carpeta descargas y ejecutamos el comando:

```
$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
```

```
freddy@freddy-ubuntu:~/Descargas$ ls
composer-setup.php  fix.sh
freddy@freddy-ubuntu:~/Descargas$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
% Total % Received % Xferd Average Speed Time Time Time Current
         Dload Upload   Total   Spent    Left   Speed
100 14926 100 14926    0     0 29673      0 --:--:-- --:--:-- --:--:-- 29673
=> Downloading nvm from git to '/home/freddy/.nvm'
=> Clonando en '/home/freddy/.nvm'...
remote: Enumerating objects: 347, done.
remote: Counting objects: 100% (347/347), done.
remote: Compressing objects: 100% (294/294), done.
remote: Total 347 (delta 38), reused 170 (delta 28), pack-reused 0
Recibiendo objetos: 100% (347/347), 203.80 KiB | 1.63 MiB/s, listo.
Resolviendo deltas: 100% (38/38), listo.
* (HEAD desacoplada en FETCH_HEAD)
  master
=> Compressing and cleaning up git repository

=> Appending nvm source string to /home/freddy/.bashrc
=> Appending bash_completion source string to /home/freddy/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
freddy@freddy-ubuntu:~/Descargas$
```

Luego de instalar ejecutar el comando:

```
$ source ~/.bashrc
```

```
freddy@freddy-ubuntu:~/Descargas$ source ~/.bashrc
freddy@freddy-ubuntu:~/Descargas$
```

Uso:

- nvm --version
- nvm install node #instala la última versión de nodejs
- nvm install --lts #Instala la version lts

* Tambien instala npx.

Instalaremos nodejs lts:

```
$ nvm install --lts
```

```
freddy@freddy-ubuntu:~/Descargas$ nvm install --lts
Installing latest LTS version.
Downloading and installing node v14.17.3...
Downloading https://nodejs.org/dist/v14.17.3/node-v14.17.3
#####
```

Ahora ya tenemos instalado NodeJs 14 y NPM 6

```
Now using node v14.17.3 (npm v6.14.13)
Creating default alias: default -> lts/* (-> v14.17.3)
freddy@freddy-ubuntu:~/Descargas$ node -v
v14.17.3
freddy@freddy-ubuntu:~/Descargas$ npm -v
6.14.13
freddy@freddy-ubuntu:~/Descargas$
```

Ahora si instalamos las dependencias dentro del proyecto y compilamos la autenticación de laravel con Bootstrap instalada:

\$ npm install && npm run dev

```
freddy@freddy-ubuntu:/var/www/html/laravelapp$ npm install && npm run dev
npm WARN deprecated axios@0.19.2: Critical security vulnerability fixed in v0.2
1.1. For more information, see https://github.com/axios/axios/pull/3410
npm WARN deprecated popper.js@1.16.1: You can find the new Popper v2 at @popper
js/core - this package is dedicated to the legacy v1
```

Sin embargo, la compilación no fue exitosa:

```
/js/app.js 2.92 MiB /js/app [ewitted] /js/app

ERROR in ./resources/sass/app.scss
Module build failed (from ./node_modules/css-loader/index.js):
ModuleBuildError: Module build failed (from ./node_modules/sass-loader/dist/cjs.js):
TypeError: this.getOptions is not a function
    at Object.loader (/var/www/html/laravelapp/node_modules/sass-loader/dist/index.js:25:24)
    at /var/www/html/laravelapp/node_modules/webpack/lib/HotModule.js:316:20
    at /var/www/html/laravelapp/node_modules/loader-runner/lib/LoaderRunner.js:367:11
    at /var/www/html/laravelapp/node_modules/loader-runner/lib/LoaderRunner.js:233:18
    at processTicksAndRejections (internal/process/task_queues.js:95:5)
    @ ./resources/sass/app.scss

ERROR in ./resources/sass/app.scss (.node_modules/css-loader??ref--5-21./node_modules/postcss-loader/src??postcss01./node_modules/resolve-url-loader??ref--5-41./node_modules/sass-loader/dist/cjs.js??ref--5-51./resources/sass/app.scss)
Module build failed (from ./node_modules/sass-loader/dist/cjs.js):
TypeError: this.getOptions is not a function
    at Object.loader (/var/www/html/laravelapp/node_modules/sass-loader/dist/index.js:25:24)
    @ ./resources/sass/app.scss 2:14-253

npm ERR! code ELIFECYCLE
npm ERR! errno 2
npm ERR! @ development: 'cross-env NODE_ENV=development node_modules/webpack/bin/webpack.js --progress --hide-modules --config=node_modules/laravel-mix/setup/webpack.config.js'
npm ERR! Exit status 2
npm ERR!
npm ERR! Failed at the @ development script.
npm ERR! This is probably not a problem with npm. There is likely additional logging output above.

npm ERR! A complete log of this run can be found in:
npm ERR! /home/freddy/.npm/_logs/2021-07-16T20_46_17_233Z-debug.log
npm ERR! code ELIFECYCLE
npm ERR! errno 2
npm ERR! @ dev: 'npm run development'
npm ERR! Exit status 2
npm ERR!
npm ERR! Failed at the @ dev script.
npm ERR! This is probably not a problem with npm. There is likely additional logging output above.

npm ERR! A complete log of this run can be found in:
npm ERR! /home/freddy/.npm/_logs/2021-07-16T20_46_17_312Z-debug.log
freddy@freddy-ubuntu: /var/www/html/laravelapp$
```

Esto es un problema con la dependencia “sass-loader” tenemos que cambiar la versión a la 10. Abrimos el package.json del proyecto y ponemos la versión 10:

"sass-loader": "^10",

Vemos que por defecto trae la 11:

```
{
  "devDependencies": {
    "axios": "^0.19",
    "bootstrap": "^4.6.0",
    "cross-env": "^7.0",
    "jquery": "^3.6",
    "laravel-mix": "^5.0.1",
    "lodash": "^4.17.19",
    "popper.js": "^1.16.1",
    "resolve-url-loader": "^3.1.0",
    "sass": "^1.32.11",
    "sass-loader": "^11.0.1",
    "vue-template-compiler": "^2.6.14"
  }
}
```



```

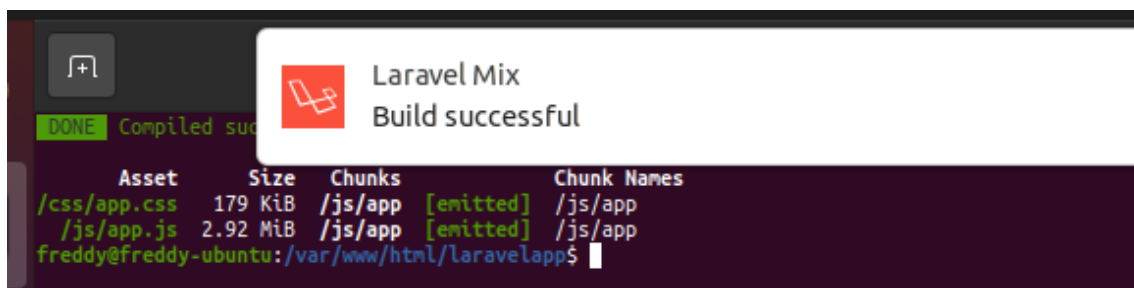
    "devDependencies": {
      "axios": "^0.19",
      "bootstrap": "^4.6.0",
      "cross-env": "^7.0",
      "jquery": "^3.6",
      "laravel-mix": "^5.0.1",
      "lodash": "^4.17.19",
      "popper.js": "^1.16.1",
      "resolve-url-loader": "^3.1.0",
      "sass": "^1.32.11",
      "sass-loader": "^10"
      "vue-template-compiler": "^2.6.14"
    }
  }
}

```

Ctrl+O guardamos y salimos con Ctrl+X si lo editamos con nano.

Y volvemos a instalar y compilar.

\$ npm install && npm run dev



Y listo!

¿Volver a limpiar caches y hacer composer dump-autoload?

Para evitar escribir manualmente estos comandos php artisan guardemos el siguiente script bash con extensión .sh y lo ponemos dentro del proyecto:

```

#!/usr/bin/env bash

php artisan log:clear
php artisan config:clear
php artisan config:cache
php artisan clear-compiled
php artisan route:clear
php artisan view:clear
php artisan cache:clear

```

```
composer dump-autoload -o  
php artisan route:cache  
php artisan optimize  
php artisan optimize:clear
```

Le llamamos “clear_caches.sh”

```
freddy@freddy-ubuntu:/var/www/html/laravelapp$ sudo nano clear_cache.sh  
freddy@freddy-ubuntu:/var/www/html/laravelapp$ cat clear_cache.sh  
#!/usr/bin/env bash  
php artisan log:clear  
php artisan config:clear  
php artisan config:cache  
php artisan clear-compiled  
php artisan route:clear  
php artisan view:clear  
php artisan cache:clear  
composer dump-autoload -o  
php artisan route:cache  
php artisan optimize  
php artisan optimize:clear  
freddy@freddy-ubuntu:/var/www/html/laravelapp$
```

Darle permisos de ejecución al script:

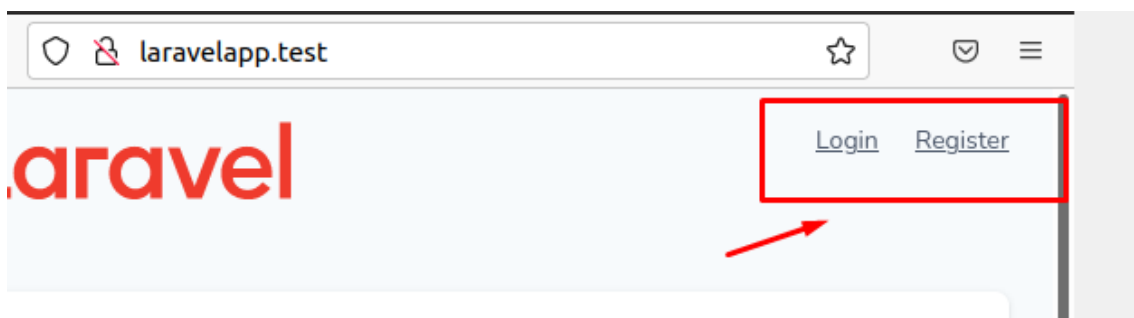
```
$ sudo chmod +x clear_cache.sh  
$ sudo ./clear_cache.sh
```

```
There are no commands defined in the "log" namespace.

Configuration cache cleared!
Configuration cache cleared!
Configuration cached successfully!
Compiled services and packages files removed!
Route cache cleared!
Compiled views cleared!
Application cache cleared!
Do not run Composer as root/super user! See https://getcomposer.org/root for details
Continue as root/super user [yes]? y
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: fruitcake/laravel-cors
Discovered Package: laravel/tinker
Discovered Package: laravel/ui
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
Generated optimized autoload files containing 4757 classes
Route cache cleared!
Routes cached successfully!
Configuration cache cleared!
Configuration cached successfully!
Route cache cleared!
Routes cached successfully!
Files cached successfully!
Compiled views cleared!
Application cache cleared!
Route cache cleared!
Configuration cache cleared!
Compiled services and packages files removed!
Caches cleared successfully!
freddy@freddy-ubuntu:/var/www/html/laravelapp$
```

Listo! Cada vez que tengamos que limpiar caches solo ejecutamos este script.

Ahora verificamos que nuestro sistema de autenticación funciona, aún falta configurar la base de datos para poder registrarnos.



Register

Name

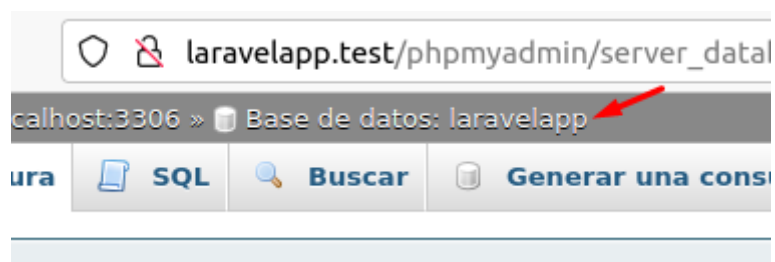
E-Mail Address

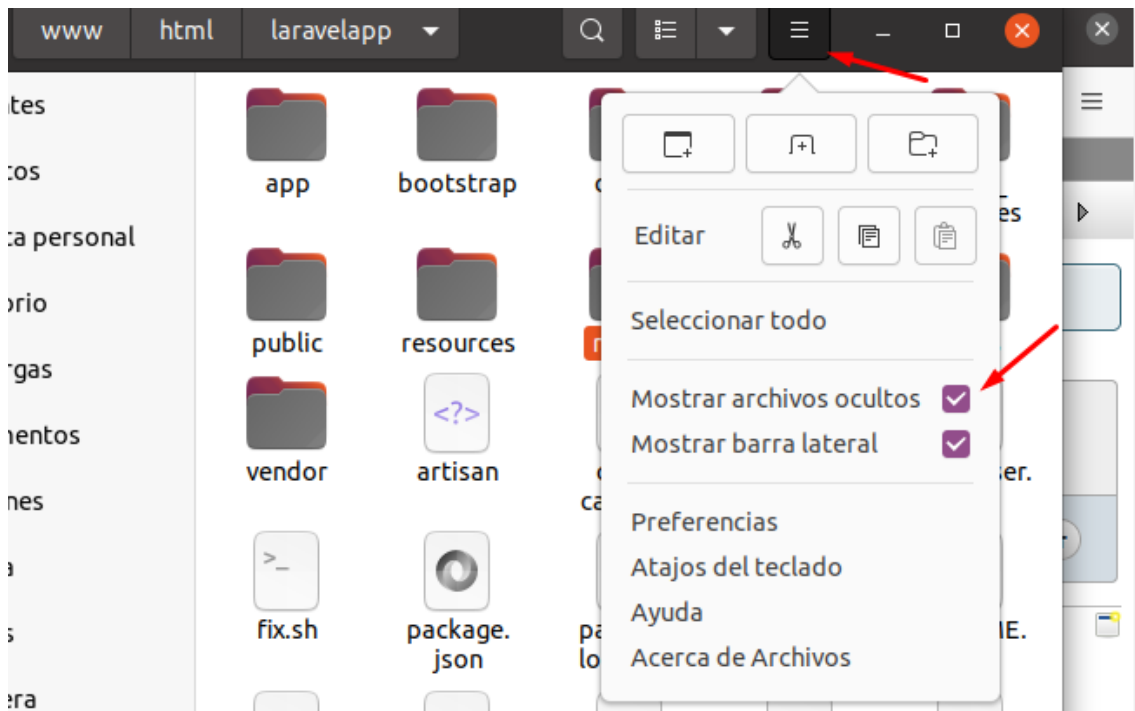
Password

Confirm Password

Please fill out this f

Ahora creamos una base de datos de ejemplo en phpMyAdmin y luego agregamos el nombre de la BD creada en el archivo .env (esta oculto) y la clave del usuario root y el usuario:





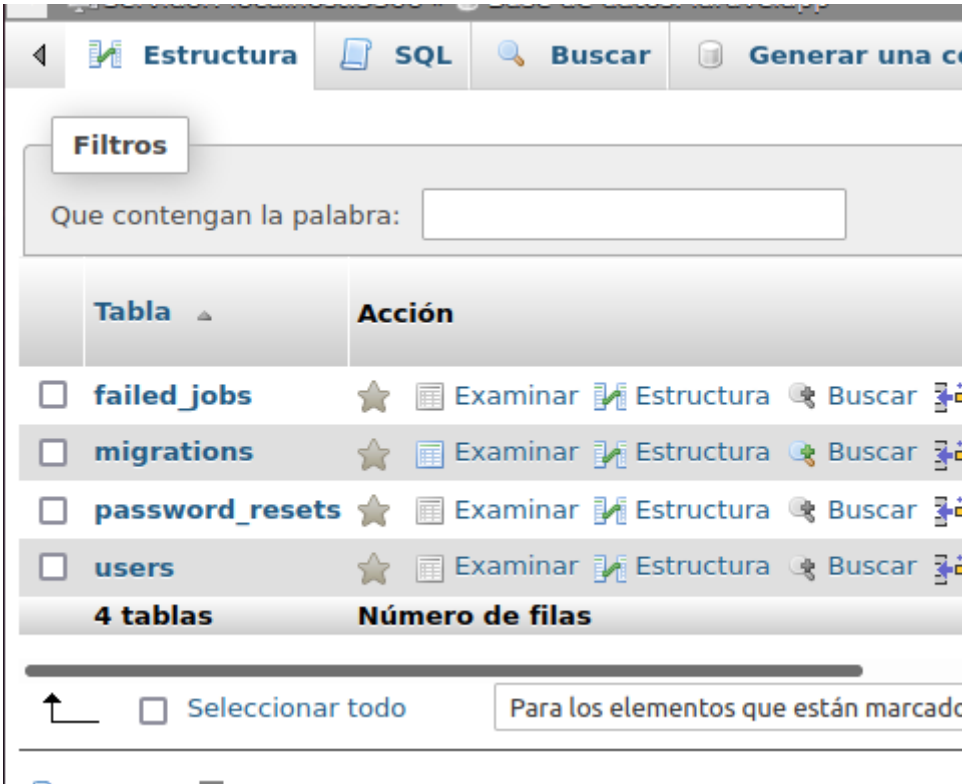
Abrimos el .env y agregamos el nombre de la base de datos creada y el usuario:

```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:zwLSXTMrpmo+tgtpUaNc5MxHDeAJ/XmA3WULQwZ5oyE
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8
9 DB_CONNECTION=mysql
10 DB_HOST=127.0.0.1
11 DB_PORT=3306
12 DB_DATABASE=laravelapp
13 DB_USERNAME=root
14 DB_PASSWORD=123456
15
16 BROADCAST_DRIVER=log
17 CACHE_DRIVER=file
```

Ejecutamos las migraciones:

\$ php artisan migrate

```
freddy@freddy-ubuntu:/var/www/html/laravelapp$ php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (110.55ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (76.19ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (96.17ms)
freddy@freddy-ubuntu:/var/www/html/laravelapp$
```



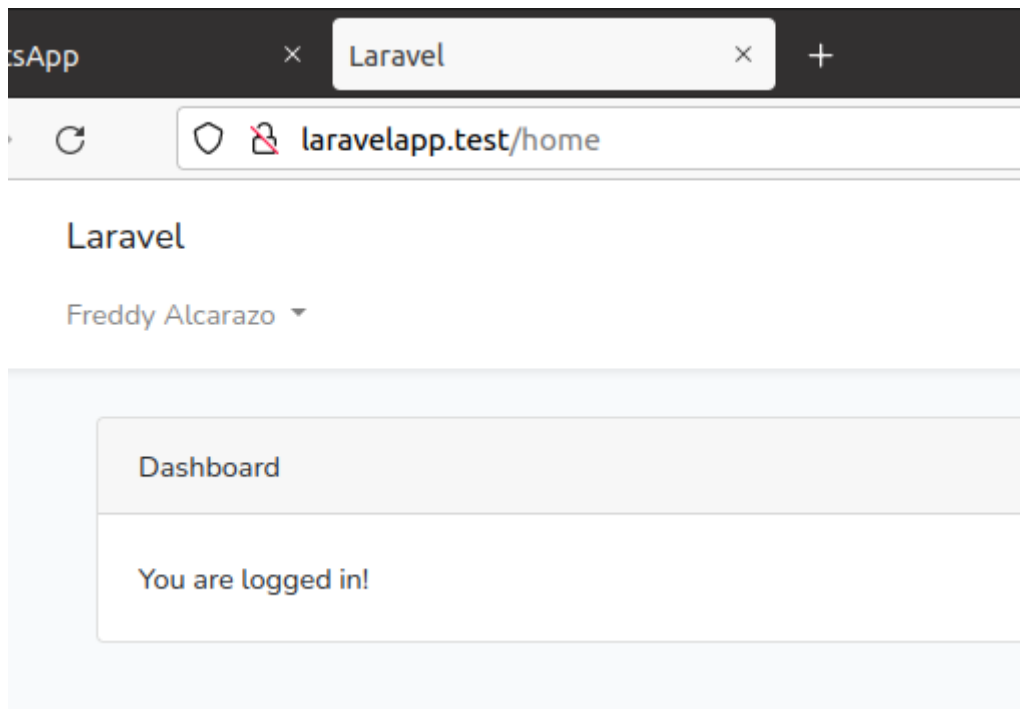
Ahora si nos registramos:

The screenshot shows a web browser window with the address bar displaying "laravelapp.test/register". The page has a light blue header with the title "Register". Below the header, there are four input fields stacked vertically:

- Name:** The input field contains the text "Freddy Alcarazo".
- E-Mail Address:** The input field contains the text "creatorpart@gmail.com".
- Password:** The input field contains six dots, indicating a masked password.
- Confirm Password:** The input field contains six dots and a vertical cursor line, indicating a masked password.

At the bottom of the form, there is a blue button labeled "Register".

Y listo! Ya estamos logueados:



¡¡Todo funciona a la perfección!!

Otras publicaciones relacionadas de mi autoría:

Hacer deploy de una aplicación Flask Python en el hosting Heroku

- <https://github.com/alcarazolabs/surflaweb-scripts/blob/main/5-Manual%20para%20Desplegar%20Flask%20App%20en%20Heroku.pdf>

Consultar más manuales de mi autoría en Github

- <https://github.com/alcarazolabs/surflaweb-scripts>

Enlaces recomendados:

Implementar medidas de seguridad para phpmyadmin:

- <https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-phpmyadmin-on-ubuntu-20-04-es>

Configurando virtual host:

- <https://www.digitalocean.com/community/tutorials/como-configurar-virtual-host-de-apache-en-ubuntu-14-04-lts-es>