

Educación Informativa UCC

Hecho en:

Laravel 4.2

Servidor de desarrollo: Droplet [DigitalOcean](#)

Características del server: 1 CPU, 1 Gb RAM, 30 Gb SSD

OS: Debian 8.2 (Jessie/Stable)

Servidor web: nginx

Ubicación: New York - US

Instrucciones de Instalación

Para ejecutar el proyecto en un servidor nginx en Ubuntu GNU/Linux 14.04 o Debian Jessie 8.2 x64. Seguir las siguientes instrucciones:

1. Instalar los componentes del Backend

Necesitaremos correr las actualizaciones en nuestro sistemas operativo; además instalar nginx y sus respectivas librerías en PHP 5

```
sudo apt-get update
```

```
sudo apt-get install nginx php5-fpm php5-cli php5-mcrypt git
```

2. Modificar la configuración de PHP

Lo primera que necesitamos es abrir el archivo de configuración principal de PHP (`php.ini`), para el procesador `PHP-fpm` que utiliza `nginx` . Abrir el archivo con permisos de administrador con un editor de texto

```
sudo nano /etc/php5/fpm/php.ini
```

Una vez abierto el archivo, modifique la línea con el parámetro: `cgi.fix_pathinfo` (posiblemente está comentado o con un valor de 1) y la cambias a cero

```
cgi.fix_pathinfo=0
```

Esto le indica a PHP que no intente ejecutar scripts con nombres similares si el archivo que busca no es encontrado. Una vez finalizado, guardar y cerrar el archivo.

La última pieza de la administración de PHP a modificar es activar la extensión `mcrypt` de la cual `Laravel` depende.

```
sudo php5enmod mcrypt
```

Ahora reiniciamos el servicio `php5-fpm` para que se implementen los cambios realizados

```
sudo service php5-fpm restart
```

3. Configurar Nginx y la ruta web

Crearemos un directorio para alojar nuestro proyecto, para eso necesitamos hacerlo con permisos de administrador, en este caso será:

```
sudo mkdir -p /var/www/mooc_ucc
```

Ahora que tenemos una ubicación para los componentes de `Laravel`, podemos mover y editar un servidor `nginx`, para eso abrimos el archivo de configuración por defecto con permisos de administrador:

```
sudo nano /etc/nginx/sites-available/default
```

Dentro del archivo agregamos nuestra configuración del servidor

```

server {
    listen 80;
    listen [::]:80 default_server ipv6only=on;

    root /var/www/mooc_ucc/public;
    index index.php index.html index.htm;

    server_name 127.0.0.1;
    ssl_certificate /etc/nginx/ssl/nginx.crt;
    ssl_certificate_key /etc/nginx/ssl/nginx.key;

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

    location ~ \.php$ {
        try_files $uri /index.php =404;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        fastcgi_pass unix:/var/run/php5-fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

}

```

Guardar y cerrar el archivo cuando hayas finalizado. Después para que el servidor nginx lea los cambios, lo reiniciamos ejecutando la siguiente línea:

```
sudo service nginx restart
```

Ahora, descomprimos el archivo `.zip` del proyecto, por ejemplo `mooc_ucc.zip` y copiamos la carpeta del proyecto a `/var/www`

```
sudo cp -R mooc_ucc /var/www
```

Ahora nuestros archivos quedan en el directorio `/var/www/mooc_ucc` pero quedan con cuenta de administrador (grupo root), por lo cual vamos a cambiarlos a otro (grupo web):

```
sudo chown -R :www-data /var/www/mooc_ucc
```

Nuestro siguiente paso es cambiar los permisos de `/var/www/mooc_ucc/app/storage` los cuáles necesitan tener permisos de escritura para que la aplicación funcione

```
sudo chmod -R 775 /var/www/mooc_ucc/app/storage
```

El último paso es reiniciar nuestro servidor nginx

```
sudo service nginx restart
```

Una vez hecho esto podremos ingresar a localhost y nos debería permitir ver nuestra página index

Si por algún motivo obtenemos un error referente a "couldn't find driver", debemos instalar y habilitar la extensión de `postgresql` en PHP

```
sudo apt-get install php5-pgsql
```

Si luego de instalar el módulo aún nos sale el error, debemos activarlo manualmente, así que accedemos a `/etc/php/php.ini` y descomentamos la línea:

```
;extension=pgsql.so y
```

```
;extension=pdo_pgsql.so (en el caso de archlinux)
```

NOTA: Para descomentar basta con eliminar el `;` inicial.

Entorno de desarrollo

La carpeta `mooc_ucc` es la raíz del proyecto, así que basta con abrir con el editor de texto favorito (por ejemplo [atom](#) o [sublime text](#)) la carpeta.

Árbol del proyecto

```
|— app
|— artisan
|— bootstrap
|— composer.json
|— composer.json~
|— composer.lock
|— CONTRIBUTING.md
|— phpunit.xml
|— Procfile
|— public
|— README.md
|— server.php
|— vendor
```

En la carpeta `app` podremos encontrar los archivos y código fuente del MOOC (modelos, vistas y controladores). El archivo `composer.json` contiene los paquetes,

dependencias y librerías utilizadas en éste proyecto. En la carpeta `public` se encontrarán los assets como archivos `css` , `javascript` , imágenes entre otros archivos estáticos.

Ahora bien, si miramos la estructura de nuestra carpeta `app` :

```
|— commands
|— config
|— controllers
|— database
|— filters.php
|— lang
|— library
|— models
|— routes.php
|— start
|— storage
|— tests
|— views
```

Podremos observar nuestras tres carpetas principales: `models` (modelos/clases), `controllers` (controladores) y `views` (vistas).

Dentro de la carpeta `models` se pueden ver todas las clases/modelos de nuestra aplicación. Se puede apreciar que todas las clases poseen la extensión `php` .

En la carpeta `controllers` se pueden apreciar los controladores de la aplicación, los cuáles también son clases y conectan a los modelos con las vistas.

Por último, la carpeta `views` contiene nuestras vistas, los cuáles son archivos también de extensión `.php` y que contienen código HTML5/PHP y son las responsables de la parte visual del **MOOC**. Están ubicadas dentro de carpetas con el mismo nombre de las clases y cada modelo tiene diferentes vistas, por ejemplo, un curso tiene vista lista, view y form, las cuáles son la vista general, específica y formulario/creación respectivamente. El archivo `index.blade.php` contiene el código del front-end de la vista index (o vista principal) del **MOOC**.

En el archivo `routes.php` encontraremos las rutas de las vistas con sus correspondientes verbos HTTP : `GET` , `POST` , etc.

4. Configurar servicio del chat

Para el correcto funcionamiento del chat se debe correr un servicio por medio de

`artisan` el cuál está basado en [ember.js](#) y que además, debe estar corriendo siempre en el servidor, de lo contrario la funcionalidad del chat sería interrumpida. Para lograr esto debemos utilizar un paquete o librería que nos permita correr el servicio perpetuamente y que se inicie aún cuando el servidor sea reiniciado también. Para ello utilizaremos el paquete [supervisor](#) para [Debian Jessie](#)

El comando es:

```
php artisan chat:serve
```

Procedemos con la instalación de [supervisor](#):

```
sudo apt-get install supervisor
```

Ahora creamos un archivo de configuración para el `supervisord`

```
sudo nano /etc/supervisor/conf.d/laravel_queue.conf
```

Y agregamos la siguiente configuración:

```
command=/usr/local/bin/run_queue.sh
autostart=true
autorestart=true
stderr_logfile=/var/log/laraqueue.err.log
stdout_logfile=/var/log/laraqueue.out.log
```

Damos permisos de ejecución:

```
sudo chmod +x /etc/supervisor/conf.d/laravel_queue.conf
```

Ahora creamos el archivo `run_queue.sh` y lo editamos

```
sudo nano /usr/local/bin/run_queue.sh
```

Agregamos lo siguiente:

```
#!/bin/bash
php /home/dev/mooc_ucc/artisan chat:serve
```

El anterior archivo abre una terminal en bash y ejecuta mediante php el comando mencionado anteriormente para la ejecución del servicio del chat en la aplicación del MOOC.

Instrucciones tomadas de [Github-Gist](#)

Iniciamos el servicio de supervisor con:

```
sudo service supervisord start
```

Ahora le decimos a supervisor que lea el archivo de configuración que agregamos:

```
sudo supervisorctl reread
```

Ahora le decimos que actualice:

```
sudo supervisorctl update
```

Lo siguiente es agregar un script para permitir el inicio automático de supervisord y sus servicios cuando se reinicie el SO.

```
sudo nano /etc/init.d/supervisor
```

Y agregamos la información incluida en el siguiente gist:

[Supervisord script](#)

Guardamos y damos permisos de ejecución al script:

```
sudo chmod +x /etc/init.d/supervisord
```

Para programar el daemon:

```
sudo update-rc.d supervisord defaults
```

Por último detenemos e iniciamos el servicio nuevamente:

```
service supervisord stop
```

```
service supervisord start
```

Si por algún motivo el chat no inicia tras reiniciar el SO, verificar las instrucciones adicionales en el enlace de referencia:

Instrucciones tomadas de [Server Fault](#)

Con esto ya tendremos corriendo siempre el servicio del chat.