Supuestos.

- Se trabaja con sistema operativo Linux.
- Se tiene instalado las siguientes aplicaciones:
  - PHP composer.
  - GIT
  - Se debe trabajar en la carpeta de trabajo HOME (~/)

Contenedor Laravel PHP

Seguir los siguientes pasos:

```
cd ~
git clone https://github.com/laravel/laravel.git laravel-app
```

```
cd laravel-app
```

A continuacion instalamos en la imagen docker de la aplicacion en Laravel composer para evitar instalarla localmente en el S.O.

```
docker pull composer
```

Copiamos el contenido del directorio de laravel-app al contenedor y la carpeta vendor de laravel se copie donde corresponda.

```
docker run --rm -e $(pwd):/app composer install
```

Establecemos los permisos del usuario actual que esta utilizando en toda la carpeta (debe tener acceso sudo):

```
sudo chown -R $USER:$USER ~/laravel-app
```

Creación de docker-compose.yml

Defeniremos la comunicación entre contenedores, los directorios "bind" con el host y la configuración en modo bridge para la comunicación entre sí.

Para simplificar, crearemos tres contenedores.

a) Para la aplicación
b) Para el nginx
c) Para la base de datos.

Ver archivo docker-compose.yml en la raiz del proyecto con la especificación app, webserver y bd respectivamente.

- El archivo que contiene los datos de la bd mysql se llama dbenviame
- /var/www enlazado al directorio ~/laravel-app que permite que la app se monte en /var/www
- Archivo de configuración ubicado en host ~/laravel-app/php/local.ini enlazado a /usr/local/etc/php/conf.d/local.ini
- Archivo my.cnf ubicado en ~/laravel-app/mysql/my.cnf enlazado con /etc/mysql/my.cnf en el contenedor nginx.

Creación de archivo Dockerfile para generar el contenedor de aplicación Laravel.

Es necesario crear un contenedor Dockerfile el que contendrá los pasos necesarios para que laravel entre en funcionamiento con:

- Una imagen debian que contiene PHP-FPM FastCGI.
- Las librerias o paquetes necesarios: mcrypt, pdo_mysql, mbstring e imagick
- Establecemos usuario www para seguridad extra (en vez de root)
- Exponemos el puerto 9000 la app.

Permisos en la bd al usuario configurado.

```
docker-compose exec db bash
```

```
GRANT ALL ON enviame.* TO 'enviame'@'%' IDENTIFIED BY 'enviame_pass';
```

```
docker-compose exec app php artisan migrate
docker-compose exec app php artisan config:cache
```

Creacion de faker

Este paso no se detallará pero es como sigue:

- Creamos un modelo Empresa, con id y nombre.
- Creamos un factory y le agregamos un faker.
- Creamos una ruta en web.php /empresa

Luego llamamos a la url en un navegados

> http://localhost/empresa

y tenemos nuestro faker creado en nuestra bd.

CRUD.

Con Postman pueden generar, segun el verbo HTTP, ejemplos.

- GET (Todas las empresas): http://localhost/api/empresas
- GET (Una empresa): http://localhost/api/empresa?id=1
- POST (Crear una empresa): http://localhost/api/empresas
  - key: nombre
  - value: Unimarc
- UPDATE (Actualizar una empresa): http://localhost/api/empresas
  - key: id, value:
  - key: nombre, value: Los Mercantes
- DELETE (Eliminar una empresa): http://localhost/api/empresas
  - key: id, value:

Notas

- Mysql fallo al iniciar en la version latest y en la 5.2.X, entonces, para simplificar, usamos postgres.
- Lo anterior implica tambien cambiar configuraciones de instalacion de la app en el dockerfile para que incluya el pdo para postgres.
- Luego en Debian no existian candidatos para pdo-pgsql en laravel.
- Forzamos la reinstalacion con la version de mysql 5.7.22
- Lidiar con el error empty continuation line in Dockerfile
- composer en laravel me obligo a subir a la version php 7.3 y tuve que sacar la extensión zip por fallos en la compilacion.