SA04AAQ01.–
Configuració de l'entorn d'execució de l'aplicació Laravel Framework.

Índex

1. Introducció

Una vegada instal·lats els **diferents components** que formen l'stack de l'aplicació, haurem de configurar el seu **entorn d'execució** que vindrà determinat, en gran manera, per:

- L'arquitectura **de l'aplicació** a desplegar (frameworks utilitzats, gestors de dependències…).
- Les **tasques** que realitza (enviament de mails, tractament d'imatges, integracions amb serveis de tercers…).

Al llarg del tema analitzarem els diferents aspectes a tindre en compte en la configuració de l'entorn d'execució d'una aplicació web, exemplificant-lo amb desplegament d'una aplicació construïda mitjançant el **framework Laravel.**

2. Entorns d'execució.

Abans de configurar l'entorn **d'execució** de l'aplicació haurem de tindre en compte que durant la construcció d'una aplicació, llevat que desenvolupes directament sobre el servidor de producció (escenari no recomanable), disposaràs de diferents entorns:

- **Desenvolupament:** aquest és l'entorn que utilitzen els programadors quan modifiquen l'aplicació per a afegir noves característiques o corregir errors. Generalment, es tracta d'un entorn local. (Computador del desenvolupador).
- **Proves o testing**: aquest entorn es fa servir per a executar automàticament les proves unitàries de l'aplicació. (En la denominada integració contínua, es tracta d'una instància temporal que es desplega individualment en cada execució).
- **Preproducció** (o "**staging**"): l'utilitza el client final per a provar l'aplicació i informar sobre els errors que ha trobat o les característiques que falten en l'aplicació. (Proves alfa)
- **Producció**: aquest és l'entorn en el qual s'executa l'aplicació que fan servir els **usuaris finals.**

3. Configuració de l'entorn.

A l'hora de configurar l'entorn necessari perquè desplegar una aplicació haurem de tenir en compte les llibreries i mòduls que utilitza l'aplicació en el dit entorn i mantenir-les al mínim. Per ficar un exemple, a **l'entorn de proves** serà necessària una llibreria com pot ser **phpUnit** per a poder passar els tests unitaris. En canvi, a **l'entorn de producció** aquesta llibreria serà innecessària.

També necessitarem configurar els directoris i els permisos d'acord amb el tipus de tasca que realitzen. No necessitarem els mateixos permisos a un directori on guardem els logs o la cau i en el que ha d'escriure el servidor d'aplicacions, que el directori on tenim els fitxers font que sols haurà de llegir.

Per finalitzar, haurem de configurar les **variables d'entorn** amb els **paràmetres de configuració**; contrasenyes, noms d'usuari, IP's de servidors de base de dades, noms de domini, etc. Aquestes variables d'entorns les encontrem generalment en els fitxers amb extensió .env.

3.1 Dependències.

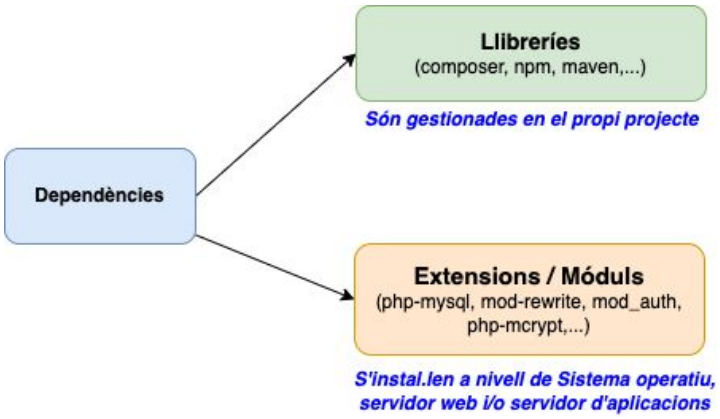
Les **llibreries, mòduls o extensions** són el principal mecanisme de reutilització de codi i

un dels **pilars** en els quals es basa la **construcció de programari** actualment.

Existeixen una gran quantitat de projectes distribuïts en forma de llibreries **privatives, lliures** i **open-source** que implementen tasques habituals com:

- Tractament de dates i hores
- Accés a sistemes gestors de base de dades (Mongo, MySQL, Redis, Postgress...)
- Compressió d'arxius (BZip2, Zip).
- xifrat/Desxifrat (openssl).
- Internacionalització (Iconv).
- Processament d'imatges.

Tota aplicació necessitarà una sèrie de dependències per al seu funcionament que haurem de tindre en compte durant el seu desplegament. En el **llenguatge PHP** podem diferenciar 2 tipus: **extensions** i **llibreries**.

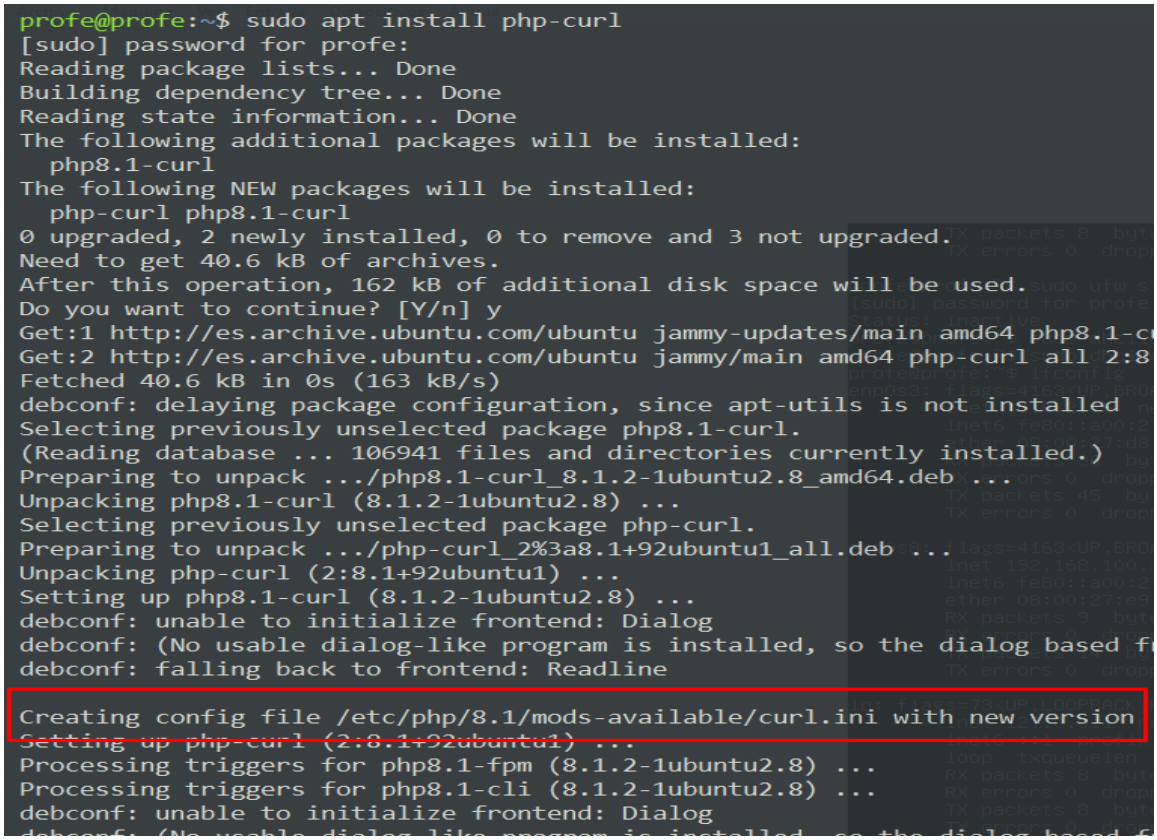


3.1.1 Extensions PHP.

Són instal·lades a través del **sistema operatiu** i estenen les capacitats del llenguatge amb noves primitives o tractaments. Generalment, són baixades com a paquets binaris amb **extensió .so** i tenen els seus propis arxius de configuració. Moltes d'elles, depenen al seu torn, d'una aplicació o llibreria específica instal·lada al sistema operatiu com és el cas d'Open-SSL, o Curl.

Als sistemes operatius **Unix** i **like-Unix** aquest tipus d'extensions són molt fàcils de configurar, només hem de fer ús del **gestor de paquets**, que s'encarregarà tant de la seua instal·lació com de la seua activació a través del fitxer de configuració amb extensió **.ini**.

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



Els arxius **.ini** contenen la configuració del mòdul i la directiva de càrrega d'aquest. Podem trobar els seus corresponents binaris (arxius amb extensió **.so**) al path **/usr/lib/php/20210902/**

3.1.2 Llibreries i frameworks.

Pots trobar-les en diferents repositoris en línia i són gestionades a través d'un gestor de dependències com és Composer. Exemple d'aquesta mena de llibreries són: Guzzle o phpUnit. En molts casos, dependran de les extensions per a dur a terme la seua tasca.

Activitat 1

- Consulta la **documentació** del **framework Laravel** i de la **llibreria phpUnit** Quins

mòduls o extensions necessita cadascuna d'elles per al seu funcionament? Indica el tipus de cadascuna d'elles.

- En un projecte desenvolupat amb la tecnologia PHP cal obtindre la nacionalitat de cadascuna de les peticions HTTP a través de l'adreça IP de la qual prové la sol·licitud. Busca informació sobre les possibles llibreries o extensions existents.
- Quines **extensions** tenim **activades** en el servei php-fpm. Cerca el directori de configuració del servei /etc/php/8.3/fpm/conf.d/. Quins mòduls tens actualment disponibles en el sistema operatiu?

3.2 Instal·lació de Composer.

Composer pot ser instal·lat en l'àmbit **global** per a qualsevol projecte que es trobe desplegat al servidor, o en l'àmbit **local** d'un projecte concret. Nosaltres ho instal·larem en l'àmbit global.

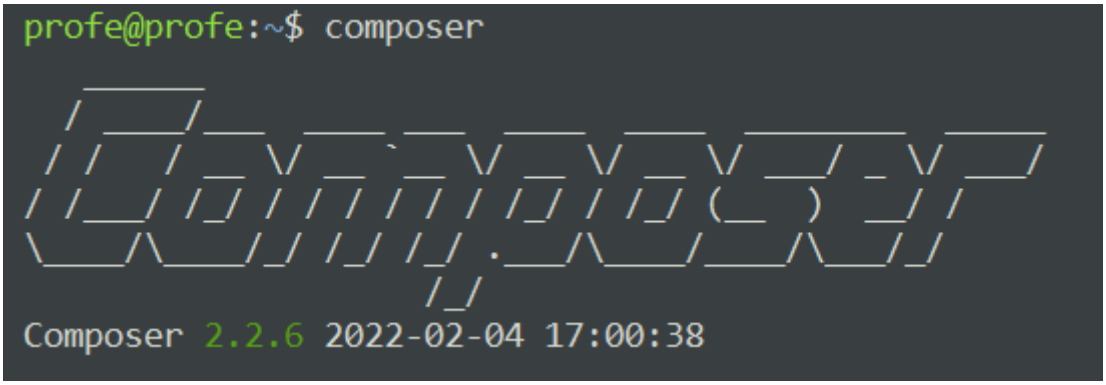
Per a la seua instal·lació, podem fer ús dels repositoris d'Ubuntu o instal·lar-lo de manera manual, tal com indiquen els seus desenvolupadors en la pàgina oficial**.**

```
$ sudo apt install composer
```

Si optem per la forma manual, podem tindre problemes amb el comandament «copy», a causa de la configuració d'Ipv6 del sistema, podem solucionar-lo assignant prioritat al trànsit **IPv4** en la descàrrega.

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

Una vegada instal·lat, hem de poder executar la instrucció «composer» des de qualsevol terminal.



Composer és un gestor de dependències (tant internes com externes) de l'aplicació. S'encarregarà d'importar **i carregar** les diferents **classes** i **fitxers** que necessitem, tant si pertanyen al **codi font** del **nostre projecte,** com si són **llibreries de tercers** que estiguem utilitzant des de la nostra aplicació (phpMailer, PHP dotEnv…).

En entorns de producció podem millorar el rendiment de les cerques de cadascun dels fitxers mitjançant la creació d'un fitxer estàtic de referències. Per a això executarem el comandament «composer install» amb l'opció «--ptimize-autoloader» (Haurem d'ejecutar el comandament posicionant el terminal al directori del projecte).

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

L'eina **Composer** ens permet mantindre diferents dependències per a l'entorn en el qual ens trobem, la qual cosa ens permetrà **optimitzar** els **temps de càrrega** dels fitxers associats a cada dependència, alhora que optimitzarem l'espai necessari per al **desplegament** de l'aplicació.

En l'exemple anterior, executant la instrucció amb el paràmetre --no-dev, ens permetrà indicar que només volem que s'instal·len les dependències que siguen per a l'entorn de producció.

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

3. Instal·lació de Laravel.

Abans d'instal·lar Laravel, serà necessari que dugues a terme la instal·lació de les extensions o **mòduls de php** necessaris i que hauràs contestat en **l'activitat 1.**

Abans de la cerca i instal·lació, **actualitzarem** la llista de paquets **disponibles** i les seues **versions** per a assegurar-nos que estiguen en la seua versió més recent.

```
$ sudo apt update
```

Si estem utilitzant un entorn Linux, només **haurem de buscar** el nom dels mòduls amb els quals s'ha registrat l'extensió mitjançant el gestor de paquets apt. Podem fer ús de la funció "apt search nom-modul". Generalment, aquests mòduls vindran registrats amb un **nom descriptiu de l'extensió**, amb el prefix php-. Per a l'**extensió amb nom mbstring** el nom del mòdul tindrà la forma php-mbstring. A continuació, és mostra com fer una cerca de l'extensió "BcMath".

```
$ sudo apt search BcMath
```

```
profe@profe:~$ sudo apt search BcMath
[sudo] password for profe:
Sorting... Done
Full Text Search... Done
php-bcmath/jammy 2:8.1+92ubuntu1 all
  Bcmath module for PHP [default]

php8.1-bcmath/jammy-updates,jammy-security 8.1.2-1ubuntu2.8 amd64
  Bcmath module for PHP
```

Una vegada trobada, la instal·larem al sistema operatiu.

```
$ sudo apt install php8.3-bcmath
```

La instal·lació d'aquest paquet ens crearà un fitxer `/etc/php/8.3/mods-available/bcmath.ini` que conté la configuració del mòdul e instal·larà l'arxiu `bcmath.so` en el path `/usr/lib/php/20210902/` per a la seua càrrega en l'intèrpret de PHP.

3.1 Directoris i permisos.

Una vegada instal·lades les dependències, crearem el directori on desplegarem el projecte i que constituirà el **directori base** de desplegament de l'aplicació i li donem els permisos i el propietari corresponents.

```
$ sudo mkdir /var/www/ddaw-example/html -p

$ sudo chown -R $USER:www-data /var/www/ddaw-example

$ sudo chmod -R 755 /var/www/ddaw-example
```

D'aquesta manera, el nostre usuari, serà capaç d'escriure en el directori i, per tant, dur a terme el **desplegament de l'aplicació**.

Finalment, ens situarem en la carpeta on acabem de definir el document root del nostre projecte i, amb l'ajuda de composer, crearem un projecte **Laravel** buit.

```
$ cd /var/www/ddaw-example/html

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

Si en executar la instrucció que ens crea **el nou projecte de laravel** ens trobem amb errors com el següent, significarà que ens hem deixat alguna extensió per instal·lar. En el cas de l'exemple, ens falta instal·lar l'extensió php-xml.

```
$> composer create-project laravel/laravel myNewApp --prefer-dist --no-dev
Creating a laravel/laravel "" project at "./myNewApp"
Your requirements could not be resoleu to an installable set of packages.
  Problem 1
    - phpunit/phpunit 9.5.x-dev requires php-xml
        -> the requested PHP extension xml is missing from your system.
```

3.1.1 Directoris amb permisos d'escriptura

El directori de desplegament de l'aplicació pertany a l'usuari amb el qual estem gestionant la configuració, per la qual cosa tindrà tots els privilegis sobre el mateix (rwx). No obstant això, el **servidor d'aplicacions** (que utilitza l'usuari www-data) **no podrà escriure en ell (php-fpm)**.

De manera general, durant l'execució d'una aplicació, existeixen uns certs directoris als quals el servidor d'aplicacions ha de tindre permisos d'escriptura, com són:

- **Directori de logs de l'aplicació;** on l'aplicació generarà i emmagatzemarà els fitxers que ens permetran dur a terme una traça de cada petició HTTP.
- **Directori de pujada d'imatges i documents**; si nostra implementa funcions que requereixen la pujada de fitxers com a imatges o **documents**, ha de poder escriure en el **directori** de pujades.
- **Directoris cau**: Fets servir per l'aplicació per a millorar el temps de resposta, mitjançant la generació de fitxers estàtics.

Hem de tindre en compte que aquest tipus de directoris poden ser altament vulnerables, per la qual cosa haurem d'assegurar-nos que els arxius que es generen no tenen permisos d'execució, que el seu contingut és el permés (image/jpeg, image/png, application/pdf), i que siguen els mínims necessaris per al funcionament de la nostra aplicació.

Al cas de **Laravel**, el servidor d'aplicacions ha de poder escriure als directoris «**bootstrap/cache»** i **«storage» situats** al **directori arrel** del **nostre projecte**.

El procediment a seguir serà assignar com a grup propietari dels directoris al grup configurat al pool de connexions del **servidor php-fpm** perquè interprete els **fitxers .php,** és el següent:

```
$> cd myNewApp;
$> sudo chgrp -R www-data storage bootstrap/cache database;
```

```
$> sudo chmod -R ug+rwx storage bootstrap/cache database;
```

Activitat 2

La documentació oficial del framework ens proporciona informació sobre l'ús que fa el mateix de cadascun dels directoris. Consulta-la i explica perquè necessitem accés d'escriptura a cadasqun dels directoris especificats.

Si compartirem el servidor d'aplicacions, una millor aproximació seria crear un grup per a cadascuna de les aplicacions i configurar un **pool** de connexions **específic** per a cada aplicació, de manera que cada **pool de connexions** només tinguera accés d'escriptura sobre els directoris corresponents de la seua aplicació.

3.2 Variables d'entorn.

Com hem comentat al **punt 2,** el desenvolupament i desplegament d'una aplicació requereix diferents entorns d'execució cadascun amb diferents característiques que han de ser configurables com:

- **Nivell de log:** En entorns de producció no és desitjable que els nostres usuaris reben notificació per pantalla dels diferents errors que es van produint.
- **Capa de Persistència:** Les bases de dades, els **SGBD** encarregats de la seua gestió i les credencials d'accés no han de ser les mateixes per a cada entorn, sinó que han de ser independents.
- **Serveis de tercers:** Si utilitzem serveis de 3rs com una plataforma d'enviament de mailing o un mètode de pagament en línia, haurem d'utilitzar diferents credencials o fins i tot crear un mock del servei durant el desenvolupament.

Per a poder mantenir aquesta independència es defineixen les **variables d'entorn**. Aquestes poden ser establides en la **mateixa configuració del servidor**

web/aplicacions o en el **mateix projecte**, però **mai hauran de guardar-se en el repositori del Sistema de Control de Versions** que estiguem utilitzant, sinó que es crearan a partir d'una plantilla base durant el desplegament de l'aplicació.

En el cas de Laravel i la majoria de frameworks **PHP** actuals, es fa ús de la llibreria **DotEnv**. Aquesta llibreria s'encarregarà de buscar un arxiu **.env** en el directori arrel de desplegament de l'aplicació i convertirà cadascuna de les entrades APP_ENV=production en **variables d'entorn**, accessibles de manera global des de qualsevol part de l'aplicació web.

Definició de fitxers .env

El següent pas per a desplegar l'aplicació serà crear l'arxiu **.env** per a la configuració de l'entorn i la seua modificació d'acord amb les credencials i paràmetres que necessitem establir.

En primer lloc crearem el fitxer a partir de la còpia de la plantilla facilitada al costat del projecte.

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

A partir d'ací, editarem el fitxer .env creat i ho modificarem d'acord amb les credencials i opcions que necessitem en l'entorn que estem configurant.

/var/www/ddaw-example/html/.env

```
APP_ENV=production
APP_DEBUG=false
APP_KEY=b809vCwvtawRbsG0BmP1tWgnlXQypSKf
APP_URL=http://example.com
DB_HOST=127.0.0.1
DB_DATABASE=laravel
DB_USERNAME=laraveluser
DB_PASSWORD=password
```

Una volta configurat el projecte, hem de migrar les dades de l'aplicació (inicialitzar la BBDD, i inserir les dades necessàries perquè l'aplicació funcione) fent servir el

comandament artisan migrate.

```
$ php artisan migrate
```

El valor de la variable APP_KEY proporciona una clau de xifratge per a les sessions i altres artefactes de l'aplicació. Podem **generar-la** mitjançant l'eina A**rtisan** que ens facilita el propi framework.

$ php artisan key:generate

Nota: la pròpia instrucció modificarà el fitxer .env amb la nova clau generada

Activitat 3

Consulta la següent documentació oficial del framework i explica perquè s'utilitzen les variables APP_ENV, APP_DEBUG, APP_KEY. Quins valors podem especificar per a la variable APP_ENV? Quina funcionalitat ens proporcionen?

3.3 Configuració del server bloc.

Seguint els passos de la **pràctica anterior**, configurarem on nou **server block** perquè s'encarregue de publicar la nova aplicació creada. Només haurem d'assegurar-nos que el modificador default_server de la directiva listen, només aparega en un únic server block de NGinx. En cas contrari, el servidor **NGinx** no s'inicialitzarà.

```
server {
        listen 80;
        listen [::]:80;
        server_name example.com www.example.com;
        root /var/www/ddaw-example/html/myNewApp/public/;
        index index.htm index.html index.php;
        location / {

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

```
                include snippets/fastcgi-php.conf;
                fastcgi_pass 127.0.0.1:9000;
        }
}
```

A continuació passem a detallar algunes de les directives utilitzades en la definició del server root anterior.

- **try_files:** la directiva try_files comprova que el recurs al qual s'intenta accedir existisca de forma explícita **(fitxer + path)**.

- Quan fem ús de les **URL amigables** el fitxer al qual s'intenta accedir no existeix com a tal, sinó que tot el trànsit passa per un **controlador frontal,** en el cas de Laravel **index.php** que s'encarregarà de passar les peticions al controlador corresponent. Pel que s'inclou el path a aquest controlador.

```
# Intenta comprovar que el recurs existeix amb i son back slash #(/),
# en cas contrari ho redirigeix al controlador frontal que
# s'encarregarà de processar les URL amigables

try_files $uri $uri/ /index.php?$query_string;
```

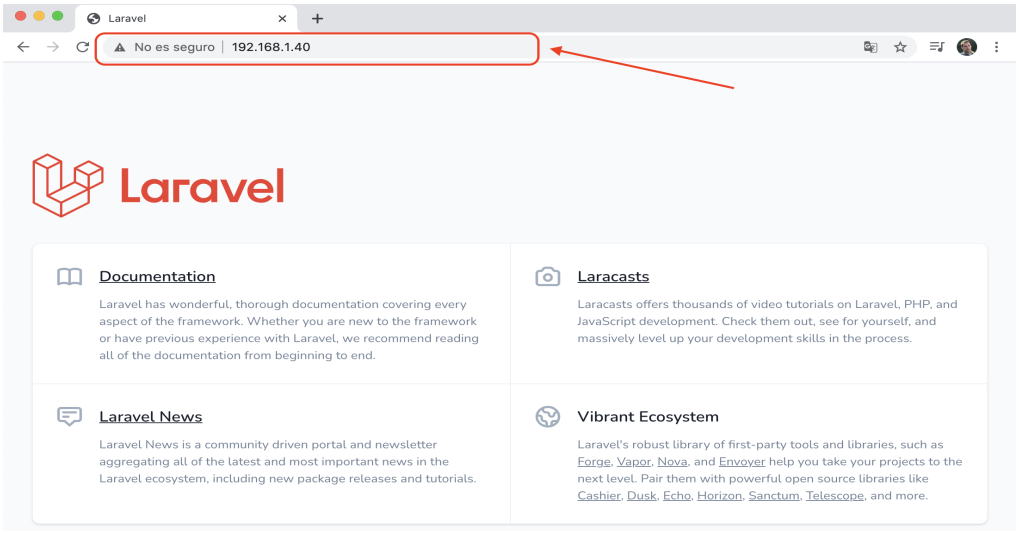
- **root:** El **document root deixa de ser el directori arrel** on hem desplegat l'aplicació i es converteix en la carpeta /public del projecte.

```
#Estableix com documentRoot la carpeta public de l'aplicació.

root /var/www/ddaw-example/html/myNewApp/public
```

A la documentació oficial de Laravel, podem trobar un arxiu optimitzat per a la configuració del **server block** de NGinx que podem utilitzar al nostre projecte.

Si tot ha anat bé, quan accedim des d'un navegador a la IP del nostre servidor, podrem veure la pàgina de benvinguda de Laravel.



En cas que Laravel ens indique missatges d'error, haurem de revisar els passos anteriors, assegurant-nos que els **fitxers** de l'aplicació tenen els **permisos correctes** i que s'han **establit les variables** d'entorn en els fitxers .env del projecte.

3.4 Treball a fer

Activitat 4

- Utilitzant l'entorn **LEMP**, creat en la **pràctica anterior**, desplega el següent projecte tenint en compte les següents especificacions (com a alternativa a aquest projecte, pot desplegar un projecte Laravel que hages realitzat en el mòdul de PWES).
  - **Entorn**: development
  - **Nom de dominio**: dev.todo.cipfpbatoi.es
  - **Directori de desplegament:** /var/www/002-es-cipfpbatoi-todo-dev/html
  - **Usuari del SO per al desplegament de l'aplicació**: dev-ddaw
- Per a dur a terme el desplegament del projecte hauràs de:
  - Instal·lar Composer.
  - Instal·lar les dependències del framework Laravel.
  - **Crear un usuari específic de la base de dades amb permisos DML i DDL** així

com una base de dades y modificar el fitxer `.env` de configuració de l'aplicació per a que l'aplicació faça ús d'aquests recursos.

Per demostrar que l'aplicació funciona correctament hauràs de mostrar, com a minim, les següents captures: log d'accés a l'aplicació al fer una petició a la pàgina principal i al donar d'alta una nova tasca, una captura del navegador accedint, una captura on aparega la nova entrada del arxiu /etc/hosts configurat, una captura de la configuració de la interficíe de xarxa, es parámetres del arxiu .env que has establit i l'usuari de la base de dades que has donat de alta junt als permissos de que disposa.

- Du a terme l'optimització indicada en el **punt 3.2** per a la **càrrega de dependències** de l'aplicació amb Composer.
  - Què conté el fitxer /vendor/composer/autoload_classmap.php

4. Referències

- 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. Documentació oficial PHP. "https://www.php.net/manual/es/index.php"
- Laravel. Documentació oficial. "https://laravel.com/docs/7.x"
- Composer. Documentació oficial. "https://getcomposer.org/"