

**CCI - SIO Atelier 2 - SACCHETTO Vladimir le 04/07/2022**

Installation et utilisation de Laravel

Rédaction de documentation utilisateur efficace



SOMMAIRE

1. Introduction. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
2. Installer le framework laravel. . . . . . . . . . . . . . . . . . . . . . . . 4
3. Routing de base. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
4. Les vues avec Blade. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
5. Les contrôleurs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
6. Vues et contrôleurs. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
7. Les migrations et les modèles. . . . . . . . . . . . . . . . . . . . . . . . . 11
8. L’ORM Eloquent. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
9. CRUD. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
10. Introduction

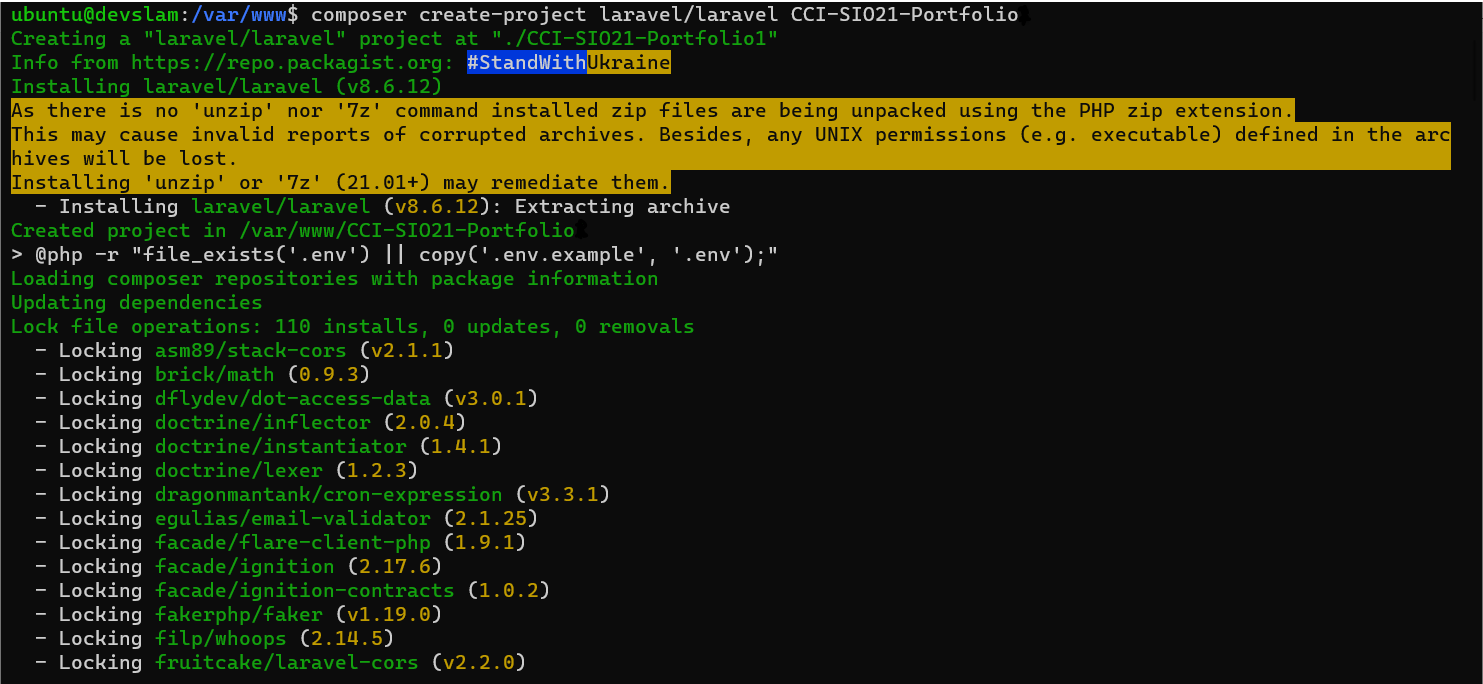
Laravel est un Framework web open-source écrit en PHP respectant le principe **modèle-vue-contrôleur** et entièrement développé en programmation orientée objet. Laravel est distribué sous licence MIT, avec ses sources hébergées sur [GitHub](https://github.com/laravel/laravel). Laravel a été créé par Taylor Otwell en juin 2011.

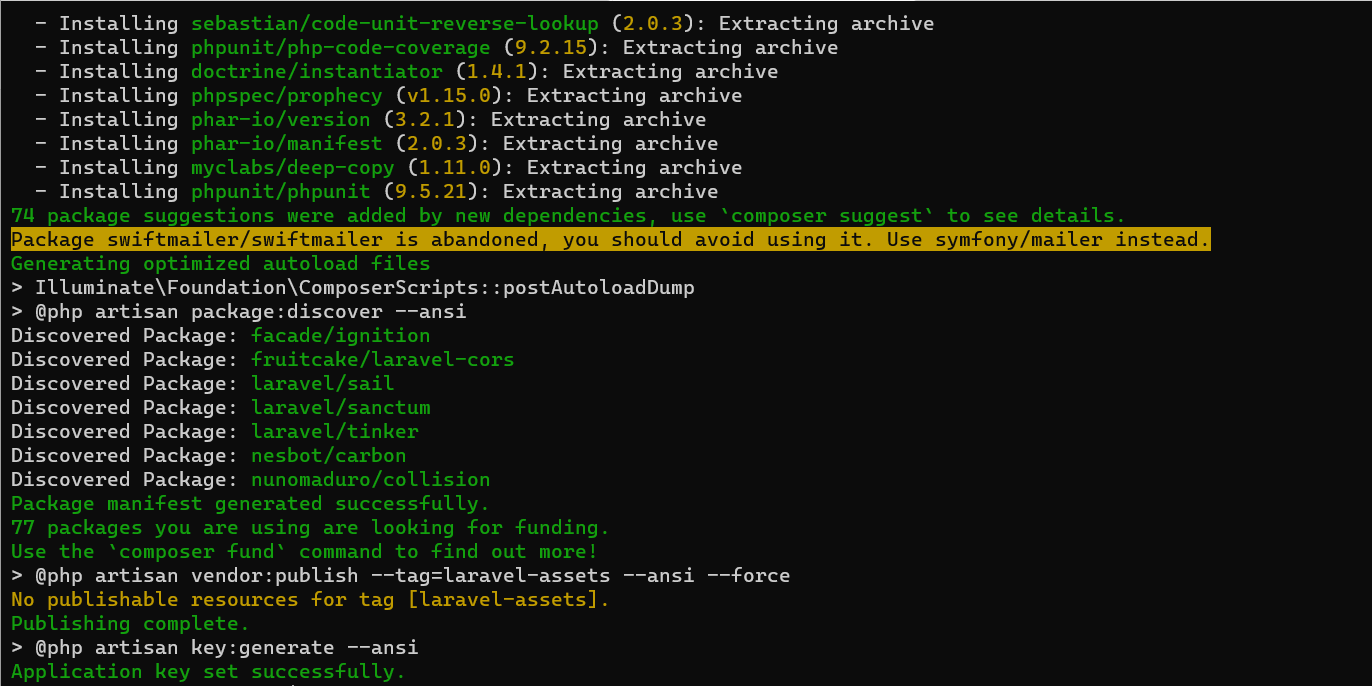
En peu de temps, une communauté d'utilisateurs du Framework s'est constituée, et il est devenu en 2016 le projet PHP le mieux noté de GitHub. Laravel reste pourtant basé sur son grand frère Symfony, pour au moins 30 % de ses lignes (utilisation de "Symfony component").

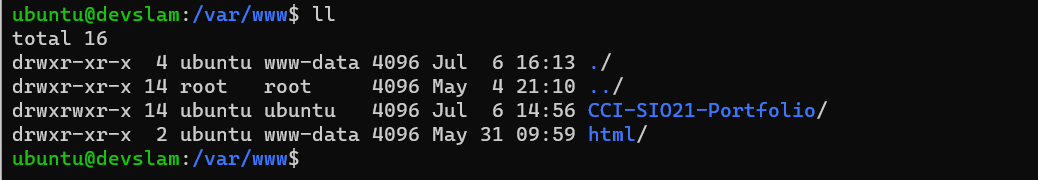
1. Installer le Framework Laravel

Nous allons installer le Framework Laravel à travers le gestionnaire de dépendances **composer** que nous avons installé lors de la mise en place de notre environnement de travail *(doc 3)*.

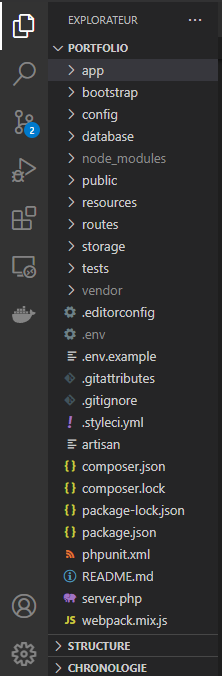
Pour ce faire nous allons nous rendre dans notre machine de développement dans **/var/www** et lançons la commande **composer** **create-project laravel/laravel *(nom du répertoire app par défaut)* *(CCI-SIO21-Portfolio dans mon cas)***







Notre projet vient d’être installé. Si nous ouvrons **Visual Studio Code** nous pouvons observer le contenu du dossier avec tous les répertoires et les fichiers de Laravel.



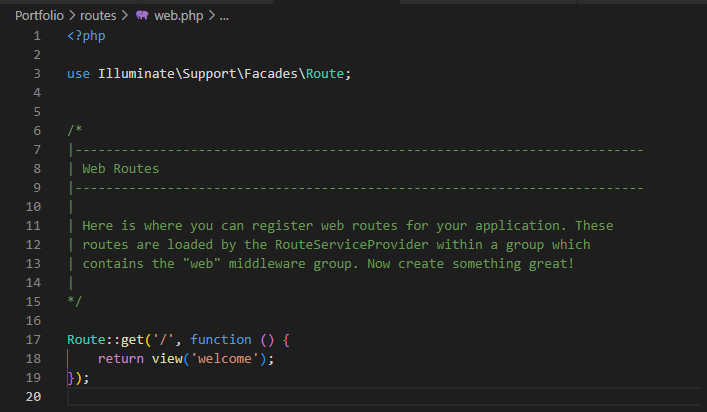
Nous pouvons retrouver toutes les informations sur le Framework sur leur page officielle [Laravel](https://laravel.com/).

1. Routing de base

Afin de coder en Laravel nous devons d’abord comprendre le principe des routes.

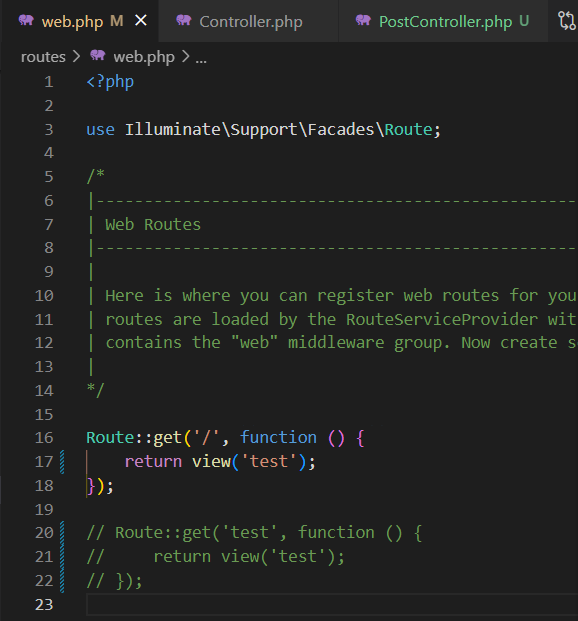
Dans le dossier **routes** nous avons différents fichiers PHP.

Nous allons ouvrir le fichier *web.php* qui sera celui qui nous servira à afficher nos pages web. Nous pouvons observer que le fichier contient des class PHP ‘’*use Illuminate\Support\Facades\Route;’’* et une fonction **Route::get** que Laravel propose par défaut.



La **Route::get** commence à la racine ‘’/ ’’ et contient une fonction qui affiche la vue *‘’welcome’’*. Cette vue fait référence au fichier *welcome.blade.php* que Laravel possède par défaut dans le dossier **/ressources/views.**

Nous allons créer une route qui retourne la vue *‘’test’’* pour l’exemple de ce document

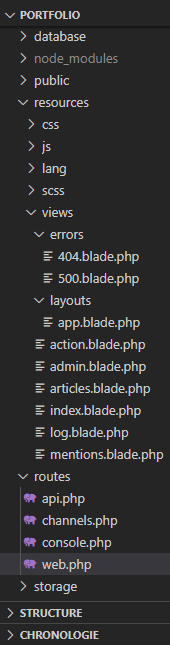


1. Les vues avec Blade

Comme nous l’avons vu dans le fichier *web.php* les routes en Laravel affichent des vues qui sont stockés dans le dossier **/ressources/views.**

Les vues sont des fichiers en format PHP qui ont l’extension ‘’*.blade.php’’*. Dans ces fichiers pouvons coder en HTML et PHP à l’aide des propriétés de **Blade**. Blade est le moteur de Template utilisé par Laravel. Son but est de permettre d’utiliser du PHP sur notre vue mais d’une manière assez particulière. Pour créer un fichier qui utilise le moteur de Template Blade nous sommes obligés d’ajouter l’extension *“.blade.php”*.

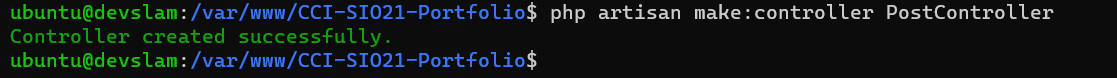
Voici les vues crées pour le projet **« Portfolio »**

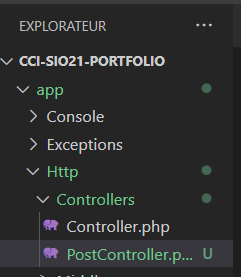


1. Les contrôleurs

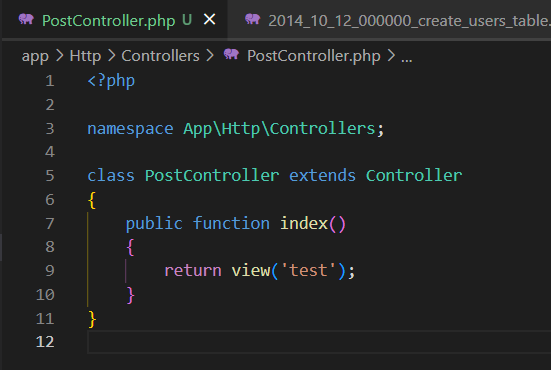
Afin que tout fonctionne correctement, Laravel utilise les **contrôleurs**. Le contrôleur est un moyen pour faire afficher les pages qui sont dans le répertoire **/ressources/views** de manière optimisée. Ce processus utilise une logique bien définie.

Nous pouvons retrouver les contrôleurs dans le répertoire **app\Http\Controllers**. Afin de créer un nouveau contrôleur Laravel nous propose une commande **php artisan make:controller *(nom du contrôleur)*** nous créons pour l’exemple de ce document un fichier *PostController.php.*



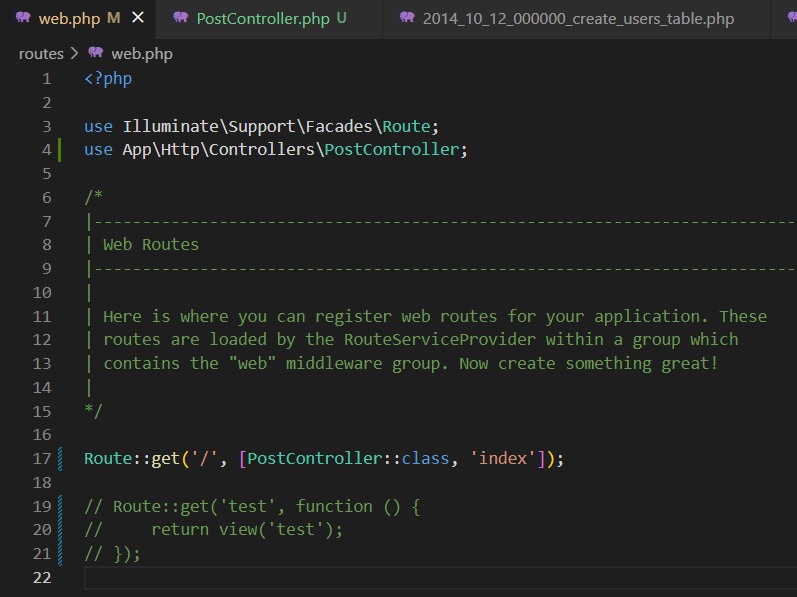


C’est donc dans ce fichier que nous allons insérer une fonction qu’on appelle *index* qui va rappeler la vue *‘’test’’*.



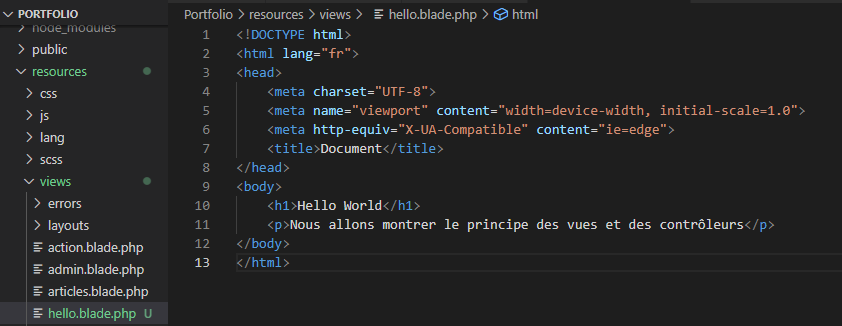
Ce fichier php contient le **namespace** avec le chemin où le fichier se trouve **app\Http\Controllers;** ainsi que la *class* **PostController** qui étend la classe de base de Laravel *Controller.php* dans **app\Http\Controllers\Controller.php**

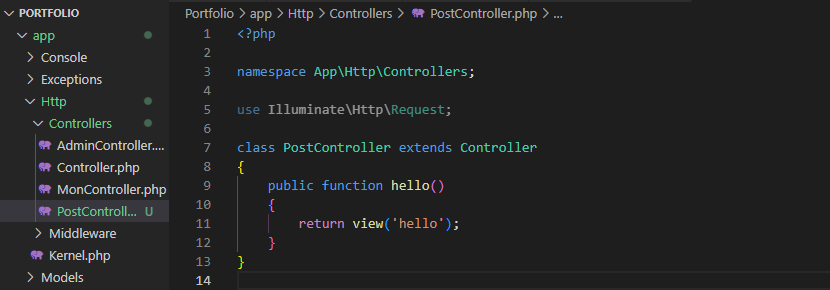
Une fois que nous avons indiqué la fonction dans le contrôleur nous allons modifier la route en indiquant la *class* du contrôleur et le nom de la fonction.

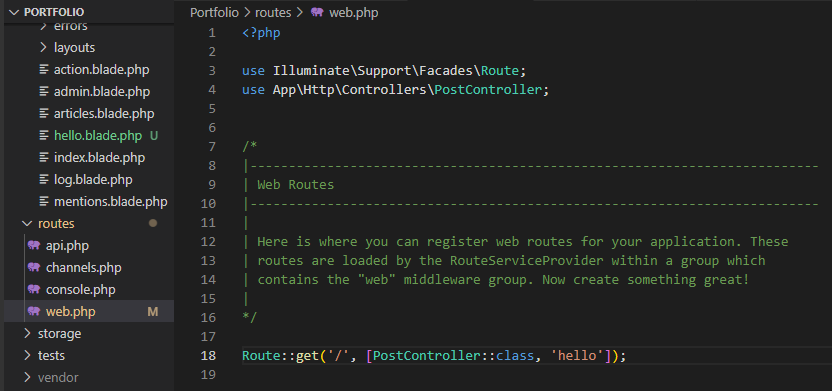


1. Vues et contrôleurs

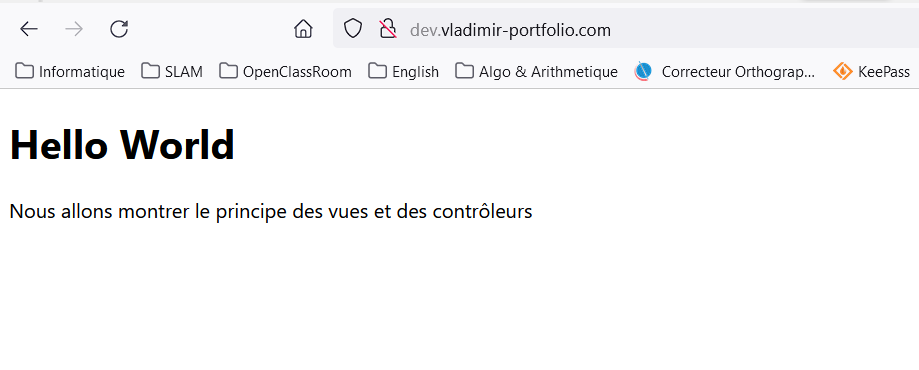
Les vues sont directement liées à la route et le contrôleur. Comme nous l’avons expliqué dans les précédents paragraphes, nous pouvons créer une vue où nous insérons ce que la page va nous afficher en HTML et/ou PHP (avec les propriétés Blade) et rappeler dans le contrôleur la fonction qui sera affiché par la vue.







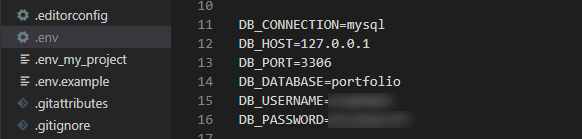
De cette manière si nous nous rendons sur notre page web nous allons afficher **Hello World**



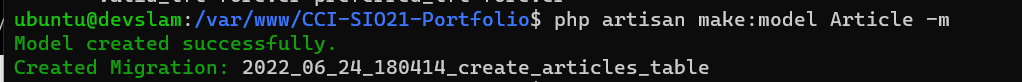
1. Les migrations et les modèles

Les migrations nous permettent de nous connecter à la Base de données.

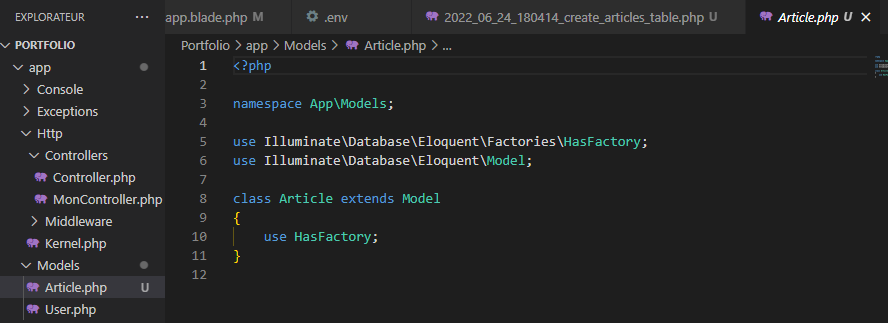
Nous allons modifier le fichier **.env** en ajoutant les informations sur la BDD qu’on utilise *(dans mon cas mysql)*. Ainsi nous allons indiquer le nom de la BDD, l’user et le mot de passe pour la connexion.

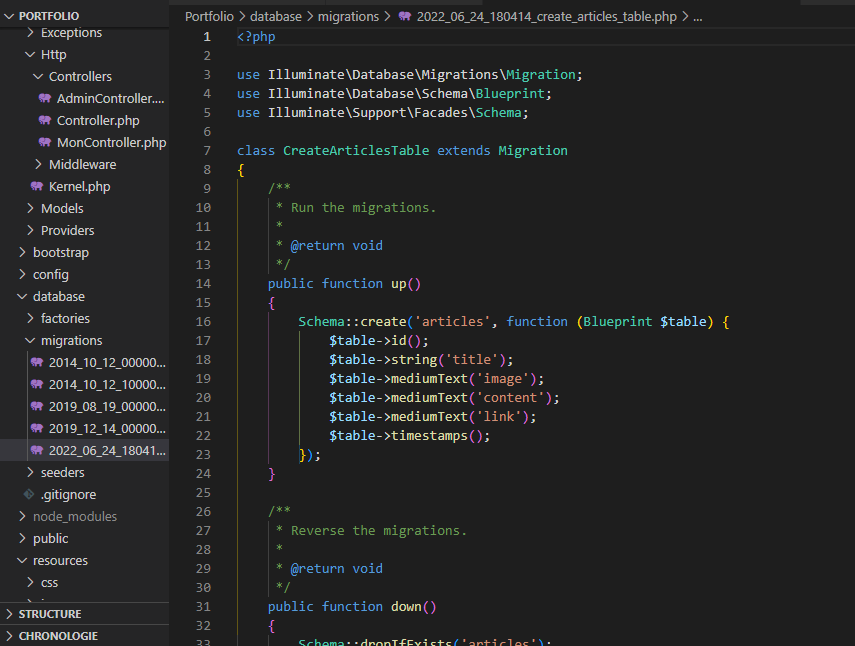


Nous allons ensuite utiliser la commande **php artisan make:model *(nom de la table)* -m** afin de créer notre migration ainsi que notre modèle.

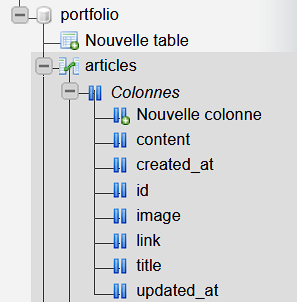


Nous aurons un modèle dans le dossier **app/Models** avec le nom de la table (dans mon cas **Article.php**) et une migration qui va se créer automatiquement dans le dossier **database/migration**

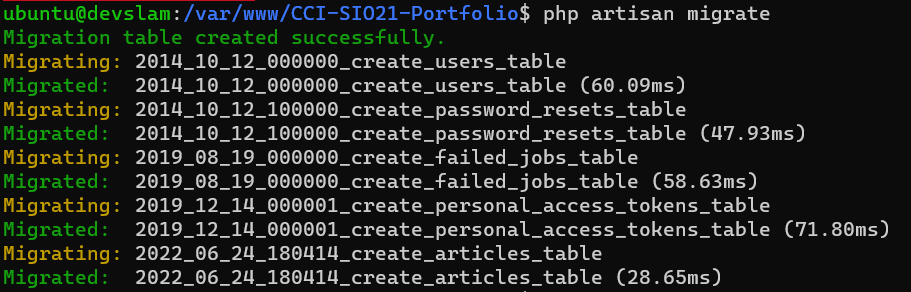


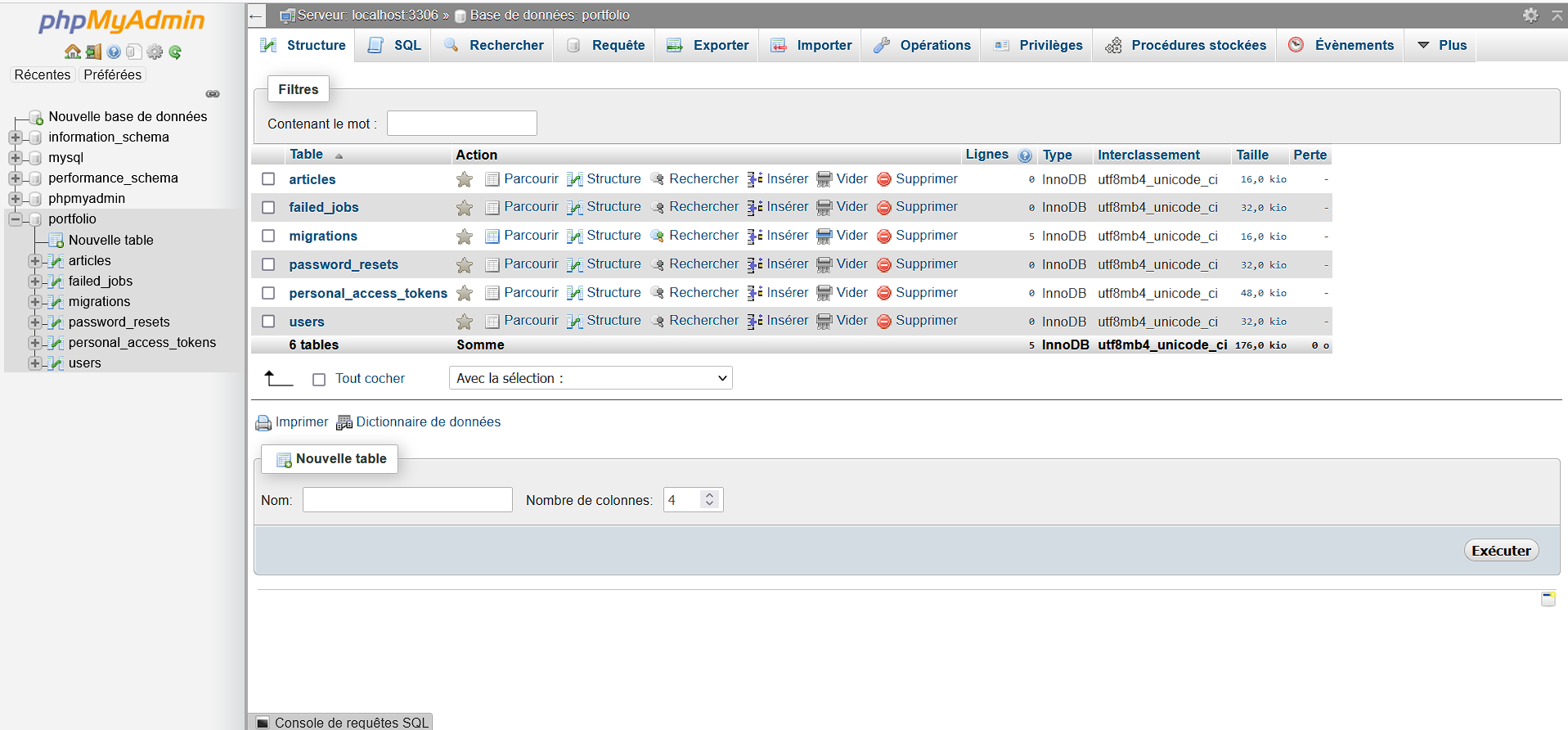


C’est dans la migration que nous allons rajouter des champs et des tables. Dans notre cas on rajoute un **id, title** « string »**, image** « mediumText »**, content** « mediumText » **et link** « mediumText »**.** Par défaut des paramètres sont déjà présentsdans la BDDdont **created\_at** et **updated\_at.**



Nous allons ensuite utiliser la commande **php artisan migrate** pour communiquer avec la base de données et ajouter tous nos fichiers de migration

****

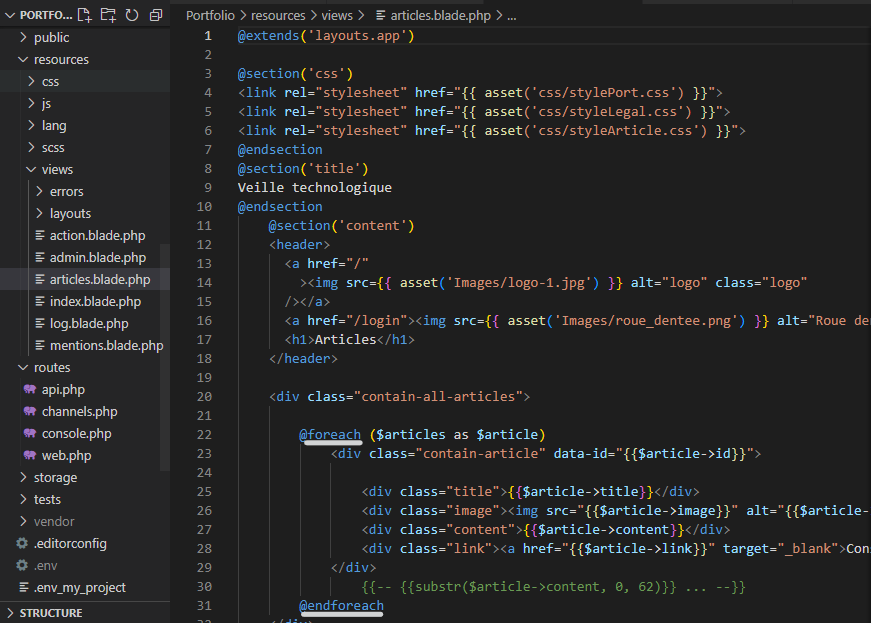
****

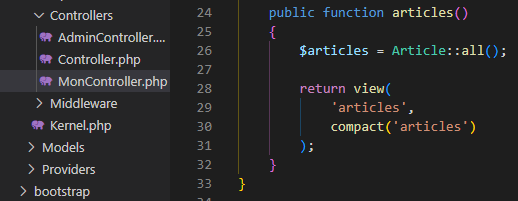
Nous pouvons effectuer la commande php artisan migrate:fresh ainsi tout le contenu de nos tables est effacé et toutes les migrations sont réécrites à vide.

1. L’ORM Eloquent

Laravel offre un environnement de développement très fonctionnel, ainsi que des interfaces de ligne de commande intuitives et expressives. En outre, Laravel utilise la cartographie objet-relationnel (ORM) pour simplifier l’accès et la manipulation des données.

Nous allons utiliser ce principe afin d’afficher les articles présents dans notre BDD sur une page web que je vais nommer *articles.blade.php*

Dans cette page grâce à une propriété *blade* je peux utiliser un *@foreach* qui me permettra d’afficher les articles présents dans ma BDD que j’ai appelé également dans mon contrôleur avec le paramètre *compact()*.**



Ainsi ma page affichera les articles présents dans la BDD qui sont défini dans le modèle *Article.php.* Le contrôleur en effet présente la *class use app\Models\Article* afin de récupérer les informations



1. CRUD

En utilisant le principe des migrations et des modèles ainsi que le MVC (modèle-vue-contrôleur), nous pouvons créer des liaisons avec la base de données et ainsi mettre en place un CRUD. L'acronyme informatique anglais CRUD *(pour Create, Read, Update, Delete)* désigne les quatre opérations de base pour la persistance des données, en particulier le stockage d'informations en base de données.

Soit :

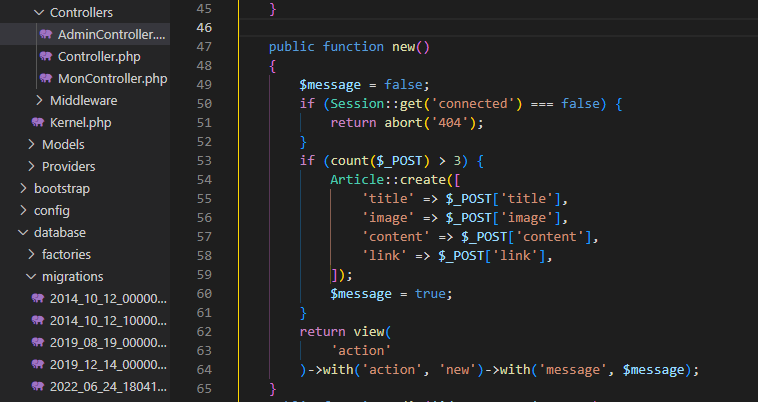
* create : créer
* read : lire
* update : mettre à jour
* delete : supprimer

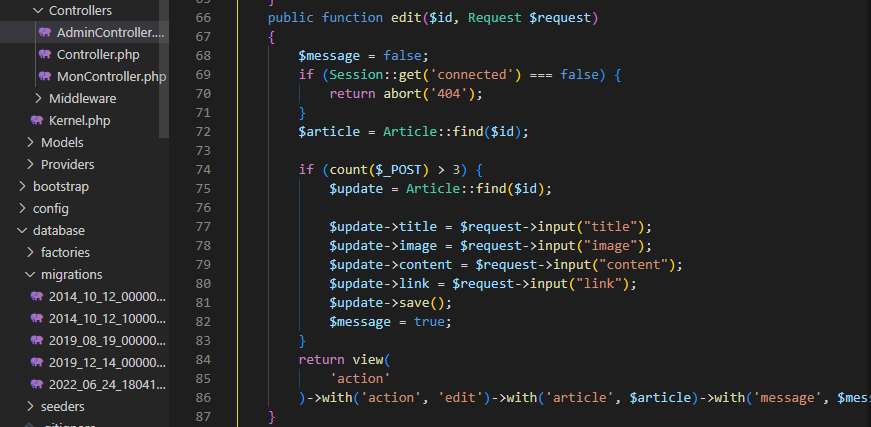
Plus généralement, il désigne les opérations permettant la gestion d'une collection d'éléments. Ce terme est aussi un jeu de mot en anglais sur l'adjectif crude (en français brut ou rudimentaire).

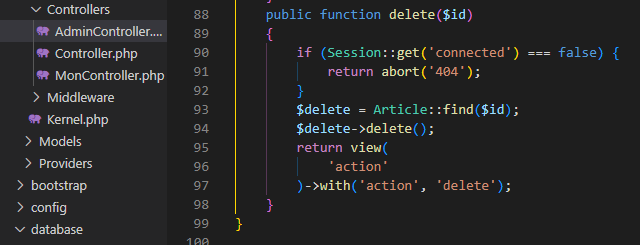
Nous allons pouvoir créer, afficher, modifier et supprimer nos articles directement depuis notre page web.

Le CRUD va donc être établi dans le contrôleur ainsi que la vue.

J’ai créé un contrôleur qui va contenir des fonctions new, edit et delete qui feront en sorte de pouvoir ajouter, modifier et supprimer les articles directement depuis la BDD







J’ai ensuite indiqué dans une vue les trois fonctions qui envoient un formulaire pour chacun des actions et crée des chemins dans la route.





