11/01/2023

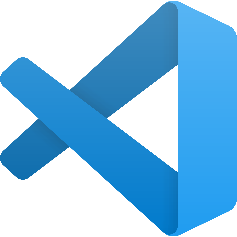
Youssef El khanchoufi

Premier Application Laravel

Laravel Version 8



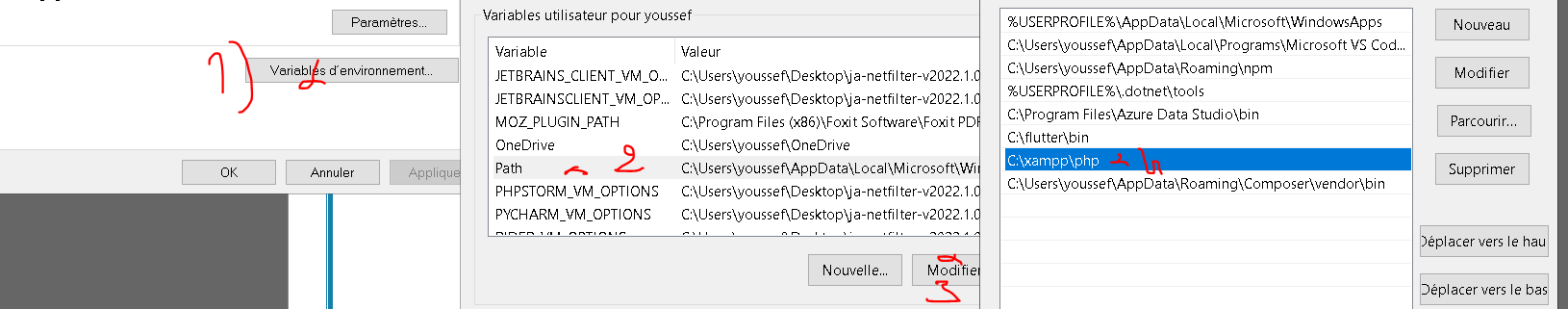
* Télécharger Visual Studio.



* Télécharger Composer.



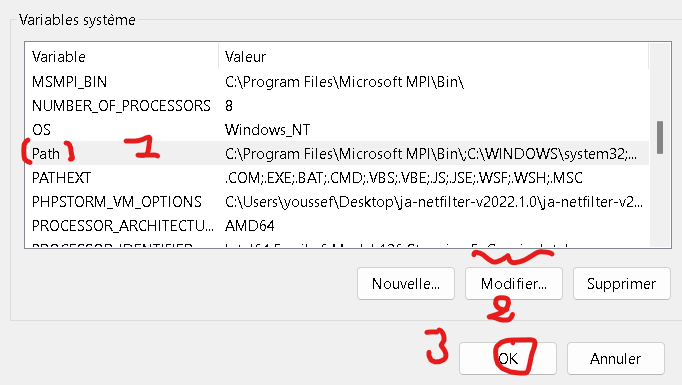
* Télécharger Xampp.
* Telecharger Php 8.

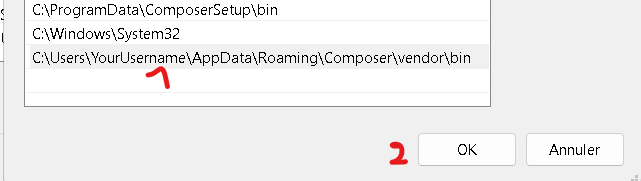
1. Taper : modifier les variable d’environnement dans recherche
2. Click sur : variables d’environnement.
3. Modifier variable path .
4. Ajouter nouvelle path vers dossier php 8.
5. Préparez votre environnement de travail
6. Installer laravel installer :

* **Laravel** **Installer** est un outil en ligne de commande qui facilite l'installation et la configuration initiale de Laravel sur votre système. Il vous permet de créer rapidement de nouveaux projets Laravel en utilisant une seule commande.

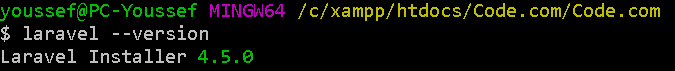


* Ajoute de path C:\Users\youssef\AppData\Roaming\Composer\vendor\bin
* Dans Environment variable

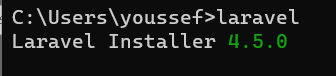




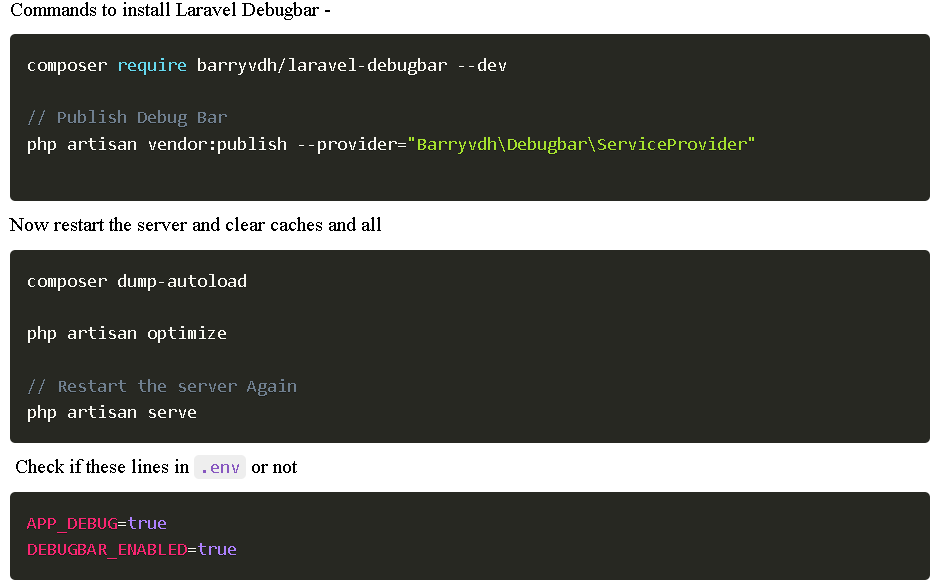
* **bash**



* **Cmd**



1. Installer Laravel Debugbar :

* Pou r Laravel Version 8 :

**Sommaire**

[I. Préparez votre environnement de travail 2](#_Toc138665646)

[1) Installer laravel installer : 2](#_Toc138665647)

[2) Installer Laravel Debugbar : 3](#_Toc138665648)

[II. Création d’un premier Projet Laravel 8 : 6](#_Toc138665649)

[1) Création du Projet Laravel : 6](#_Toc138665650)

[3) Création d’un nouveau contrôler : 6](#_Toc138665651)

[4) Passer des paramètres : 7](#_Toc138665652)

[5) Ajouter CSS et JS dans Blade template et Organisation des Pages : 8](#_Toc138665653)

[6) Blade Templates: 10](#_Toc138665654)

[7) System de Routage : 11](#_Toc138665655)

[III. Création et gestion des basés Données : 12](#_Toc138665656)

[1. Création d’un basé donnée dans MySQL 12](#_Toc138665657)

[2. Migration : 12](#_Toc138665658)

[1.1 Commande php artisan make :migration 12](#_Toc138665659)

[1.2 Method Up et Down : 13](#_Toc138665660)

[1.3 Exécution de migrations : 13](#_Toc138665661)

[1.4 Restauration des migrations 13](#_Toc138665662)

[3. Création Tables **Erreur ! Signet non défini.**](#_Toc138665663)

[1.5 Schema Builder 14](#_Toc138665664)

[1.6 Façade Schéma 14](#_Toc138665665)

[1.7 Classe Blueprint 14](#_Toc138665666)

[1.8 Type Des donnée avec Classe « Blueprint » 15](#_Toc138665667)

[4. Add fake data using factories (factories) 19](#_Toc138665668)

[5. Stocker des données dans la basé donnée à l’aide classe seeder (seeders) 19](#_Toc138665669)

[a. Calling Additional Seeders 20](#_Toc138665670)

[b. Calling Additional Seeders 20](#_Toc138665671)

[8) Obtention des données de basse donnée et afficher dans View 21](#_Toc138665672)

[9) Query Builder : 21](#_Toc138665673)

[10) Paggination 23](#_Toc138665674)

[IV. Forms : 24](#_Toc138665675)

[1) Ajouter un enregistrement 24](#_Toc138665676)

[2) Validation de la demande de formulaire 26](#_Toc138665677)

[3) Personnalisation des messages d'erreur : 26](#_Toc138665678)

[4) Validation de la demande de formulaire : 27](#_Toc138665679)

[5) Modifier les donnée : 29](#_Toc138665680)

[6) Supprimer les donnée : 31](#_Toc138665681)

[7) Uploade les fichiers : 32](#_Toc138665682)

[8) Update les Images : 33](#_Toc138665683)

[9) Validation les fichiers ajoutés : 33](#_Toc138665684)

[V. ) Éloquent : ORM : 34](#_Toc138665685)

[VI. Laravel Jetstream : 36](#_Toc138665686)

[1) Installation Package 36](#_Toc138665687)

[2) Désinstaller Package jetstream : 36](#_Toc138665688)

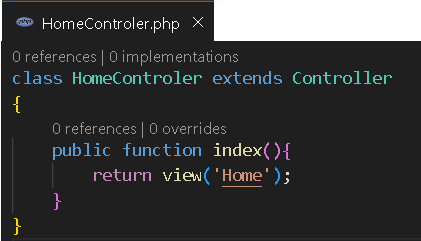
[3) Laravel Mix : 37](#_Toc138665689)

1. Création d’un premier **Projet Laravel 8** :
2. Création du Projet Laravel :

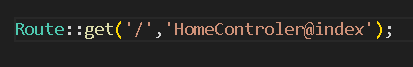
* Accéder au Dossier c:/xmapp/htdocs/Nom\_dossier
* Ouvrir Git/Bash/here
* Composer create-project laravel/laravel:^8.0 Laravel\_8
* Ouvrir dossier dans visual studio

1. Création d’un nouveau contrôler :

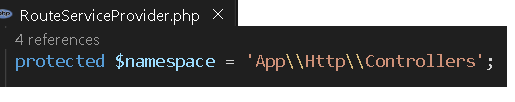
* Dans Git.bash : Cd Nom\_Projet
* Php artisan make:controller HomeControler
* Les contrôleurs créer dans chemin : app/http/Controllers/
* Créer method Index : public index function() { return “Youssef”; }



* Ajoute Route vers Controler dans chemin : app/routes/web.php.



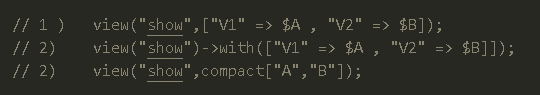
* Dans App/Provider/ RouteServiceProvider = protected $namespace = 'App\\Http\\Controllers';



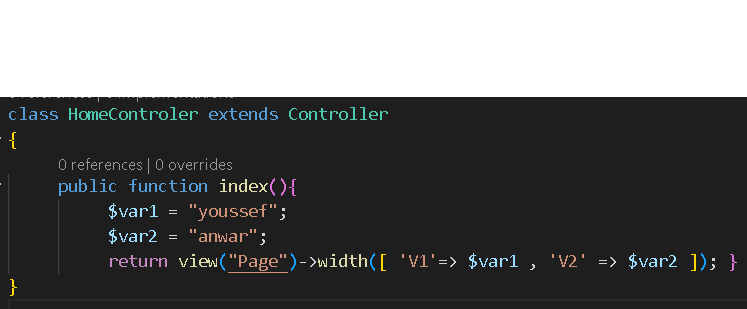
* Demarer Server: php artisan serv
* Accéder page par adresse : https ::/127.0.0.1.8000

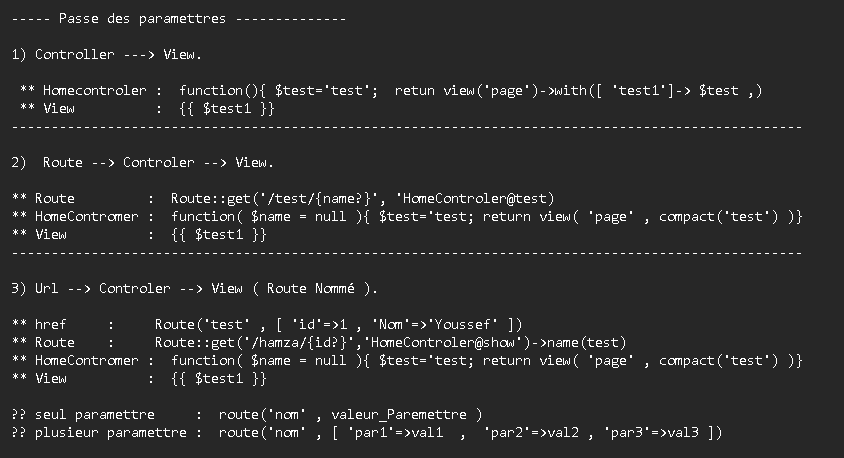
1. Passer des paramètres :

* ***Controler ----> View :***
* Dans View: {{$**V1**}} {{$**V2**}} pour concaténation : {{ **V1** . ' '. **V2**}}
* Il y a trois method pour passe des données à une vue.



* ***URL -- Controler --> View* :**
* Route::get(‘/{name?}', 'HomeControler@index'} ); (? = Si parameter n’existe pas)
* Dans Controler: Public function index($**name**=NULL) { return view("Page")->width([ **'Name'**=> $**name**]) }
* Dans View : {{**Name**}}



******

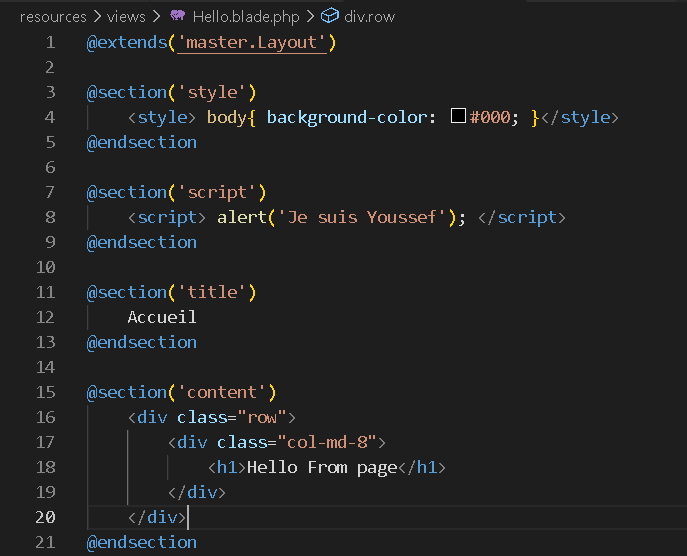
1. Ajouter CSS et JS dans Blade template et Organisation des Pages :

* Les views créer dans dossier : **App\ressources\views**
* Créer **Layout.blade.php** : Ajouter Structure du page Web
* Dans Home page: [Home.blade.php](http://Home.blade.php) .
* Ajouter **Bootstrap** Dans Page Html.



**NB:**

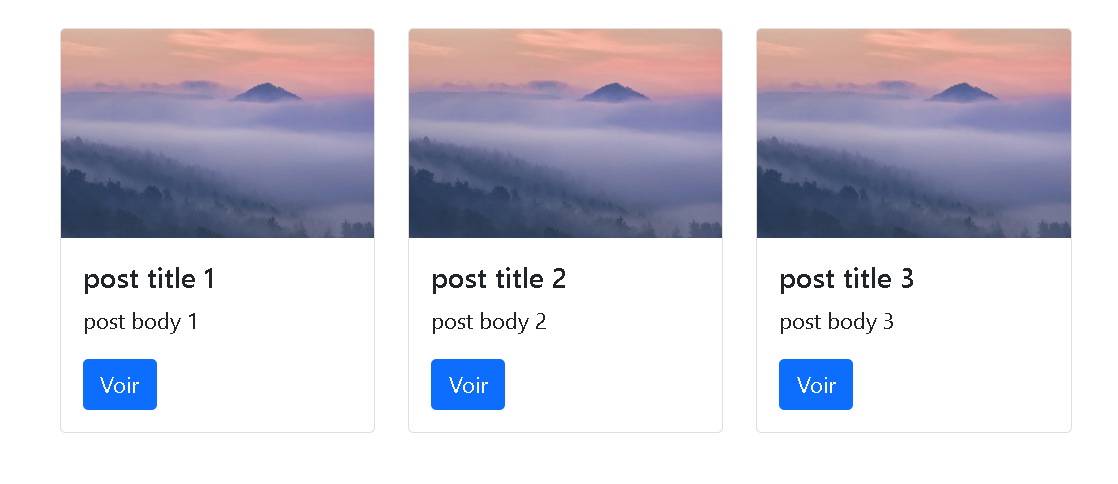
* Pour appeller la page Layout.blade dans Home.blade : @extends('Nom blade').
* Pour déclarer une variable dans Layout.blade : @yield(‘nom\_variable’).
* Pour appeler la variable dans Home.blade : @Section(‘Nom\_Varaiable’) @endSection

- Dans **HomeControler** Créer une Tableau de **Posts** :



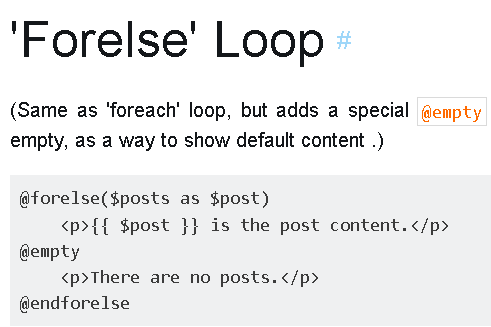
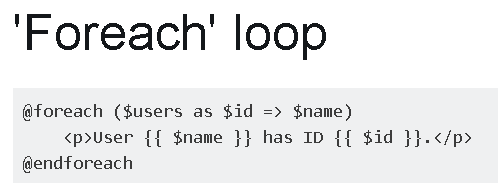
* Dans **View** Hello Afficher Tableau de **Posts** dans **Card** :

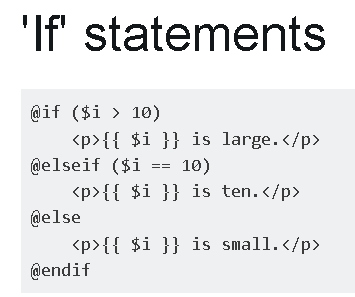
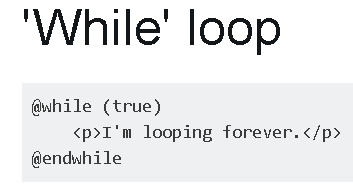




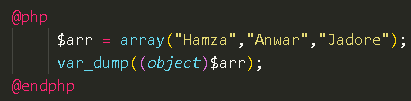
1. Blade Templates:

* ***View* :**
* Un View peut étre appelée à l’aide de la view fonction d’assistance = view( path , array(data))
* Les fichier dans les sous-dossier : view (‘part.header.navigation’)

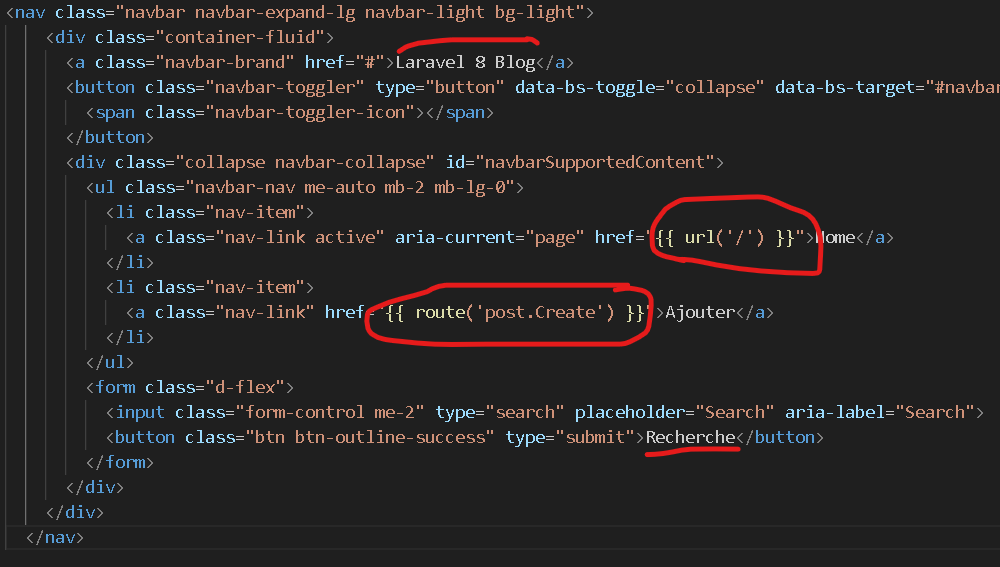




* ***Affichage des variables* :**
* Afficher variable : {{ **$variable** }}
* Element array : {{ **$array[$key]** }}
* Object : {{ **$obj-> preperty** }}
* Cheching for exist : **isset($variable)** {{ **$variable or ‘Default’** }}
* ***Ajouter Code PHP dans Blade* :**



* Ajouter Navbar dans layout :



* **Function URL (‘/’):** la fonction **url('/')** est utilisée pour générer l'URL de base de l'application. Cela renvoie l'URL racine de l'application Laravel.
* Example : ( $url = url('/'); ) = ( $url = [**http://localhost:8000/**](http://localhost:8000/) )

1. System de Routage :

**Requête http** : Le client demande une ressource au serveur en envoyant une **requête** et le serveur réagit en envoyant une **réponse**, en général une **page Html**.

Les méthodes de **requête** **HTTP** définissent l'action que vous souhaitez effectuer sur une **ressource**.

**GET** : Utilisée pour **récupérer** des données. Elle **ne** **modifie** **pas** l'état du serveur ou des données.

**POST** : Utilisée pour **envoyer** des données. **Créer** de **nouvelles** **ressources** ou soumettre des formulaires. Les **données** sont **envoyées** dans le **corps** de la **requête** et **ne** **sont** **pas** visibles dans l'URL.

**PATCH**, **PUT** : Utilisée pour **mettre** **à** **jour** une **ressource** **existante** avec les **données** **fournies**. **remplacer** entièrement une ressource existante.

**DELETE** : Utilisée pour **supprimer** une **ressource** **spécifiée**.

Syntaxe : Route::type requête( ‘Route’ , callback )

* **Par défaut**, le formulaire utilise la **méthode** **"post"**.
* Mais pour utilise les autres méthodes comme patch et put et delete ;
* Vous pouvez utiliser la directive **@method()** pour indiquer explicitement la méthode à utiliser.

<form **action**= "**#**" method="**Post**" >

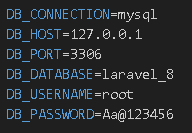
**@csrf**

**@method(PUT)**

</form>

1. Création et gestion des basés Données :
2. Création d’un basé donnée dans MySQL

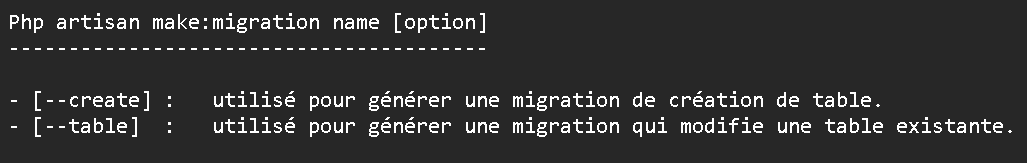
* Dans XAMPP Control Panel Activer Service Apache et MySQL.
* Tapez Nouvelle base de données : Laravel \_8
* Dans Projet Laravel : Modifier Fichier : **.env** (nom , nom de l’utilisateur , mot de passe )

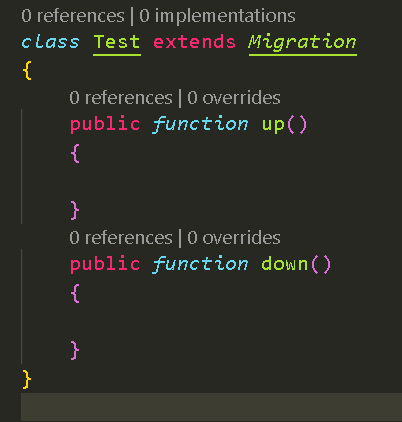


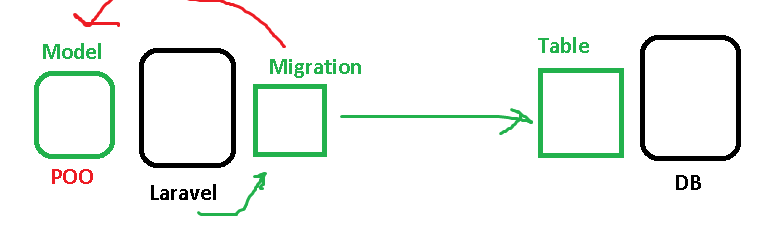
1. Migration :

* La **Schema facade** offre un moyen pratique d'interagir avec le schéma de la base de données. La façade Schéma vous permet d'effectuer diverses opérations sur les tables de base de données, telles que la création, la **modification** et la **suppression** de tables, ainsi que l'ajout ou la modification de colonnes.
* Une classe de migration contient deux méthodes : **up** et **down**. La **up** méthode est utilisée pour ajouter de nouvelles tables, colonnes ou index à votre base de données, tandis que la **down** méthode doit **inverser** les opérations effectuées par la **up** méthode.

1. Commande php artisan make :migration

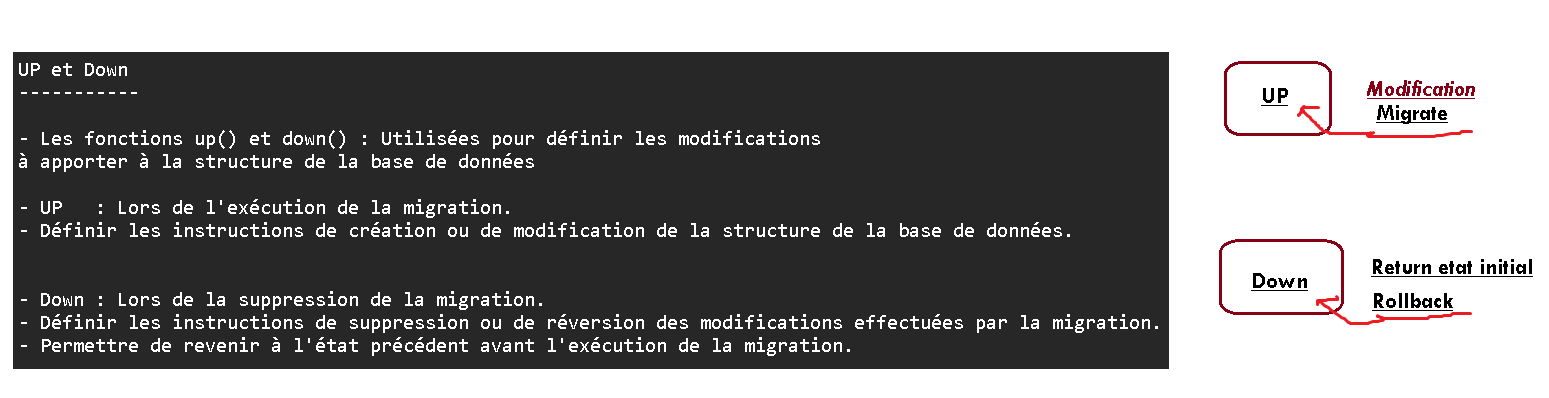
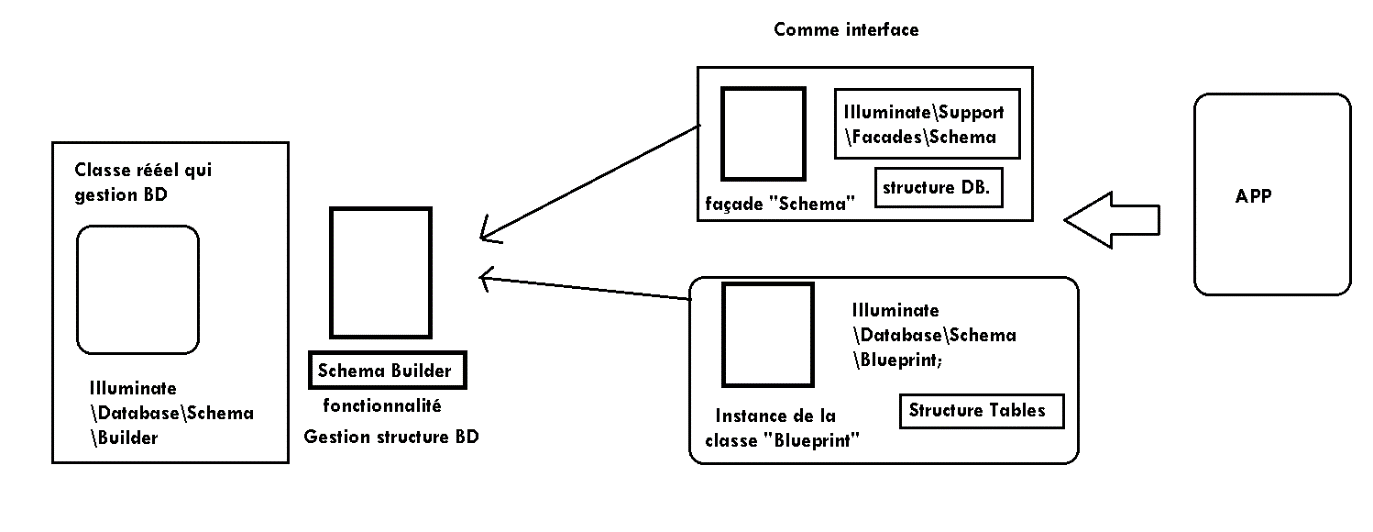
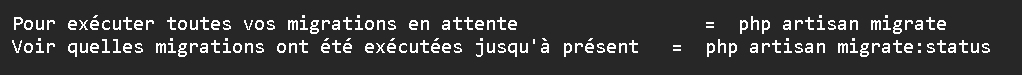




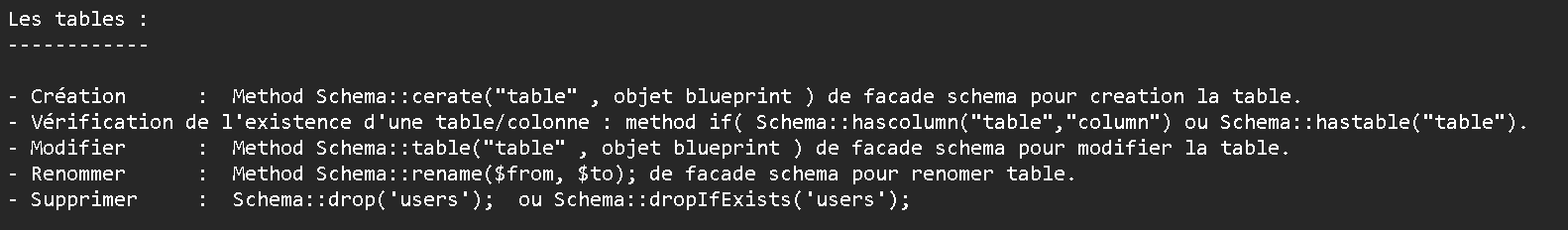


* Chaque **Migration** correspondant **table** en **DataBase**.
* Chaque **Model** Lie avec **Migration**.

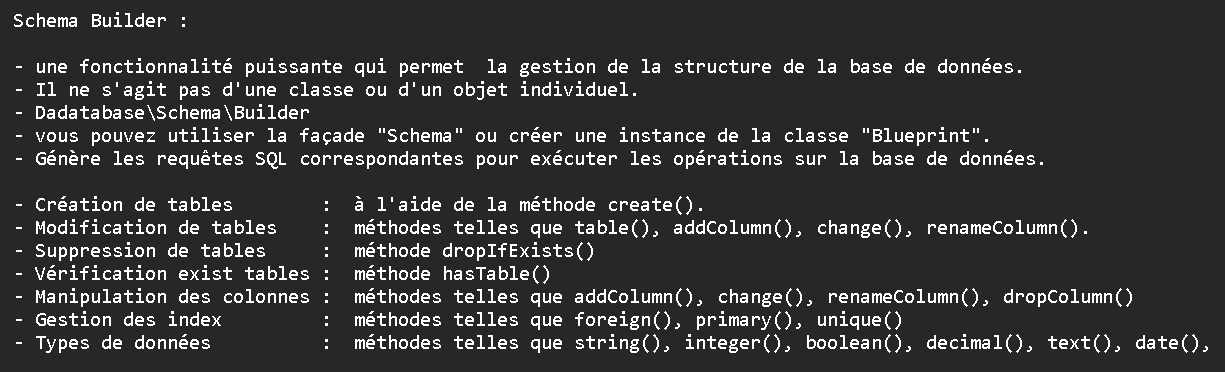
1. Method Up et Down :

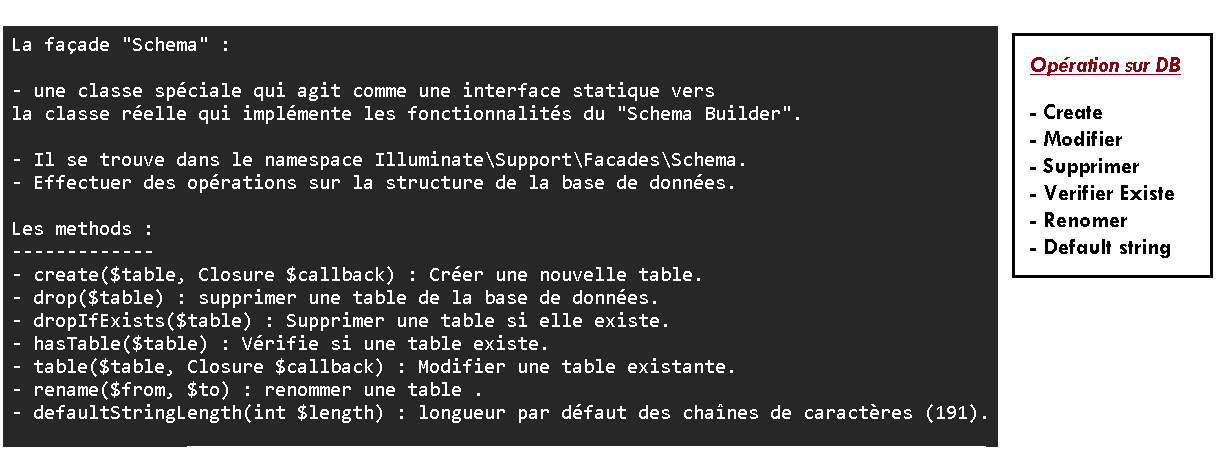
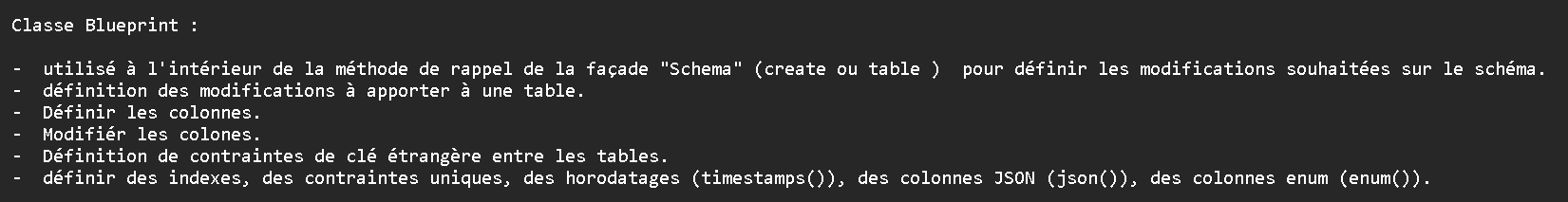


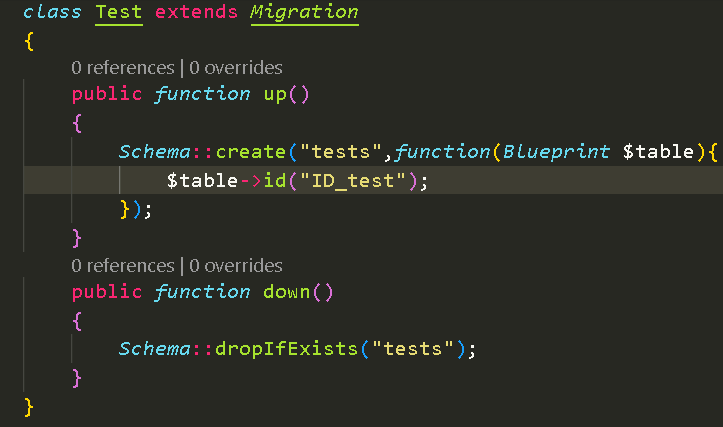
1. Exécution de migrations :
2. [Restauration des migrations](https://laravel.com/docs/8.x/migrations#rolling-back-migrations)
3. Gestion la structure des tables



1. Schema Builder



1. Façade Schéma
2. Classe Blueprint
3. Creation la Table Migration



* Nom Suivi mot Create C’est Nom de la table, Si créate Model en convention pluriel.

YYYY\_MM\_DDHHSS\_**Creation**\_(**Nom**)\_Table

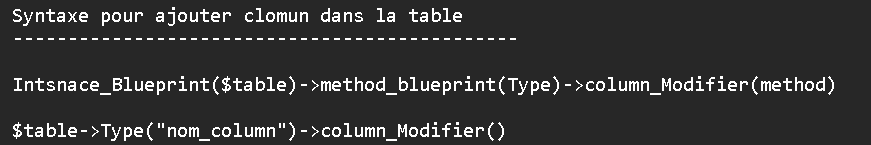
YYYY\_MM\_DDHHSS\_**ADD**\_(**Column**\_)\_**To**\_(**Nom**)

YYYY\_MM\_DDHHSS\_ **Modifier**\_(**Column**\_)\_**IN**\_(**Nom**)

YYYY\_MM\_DDHHSS\_ **Renomer**\_(**Column**\_)\_**IN**\_(**Nom**)

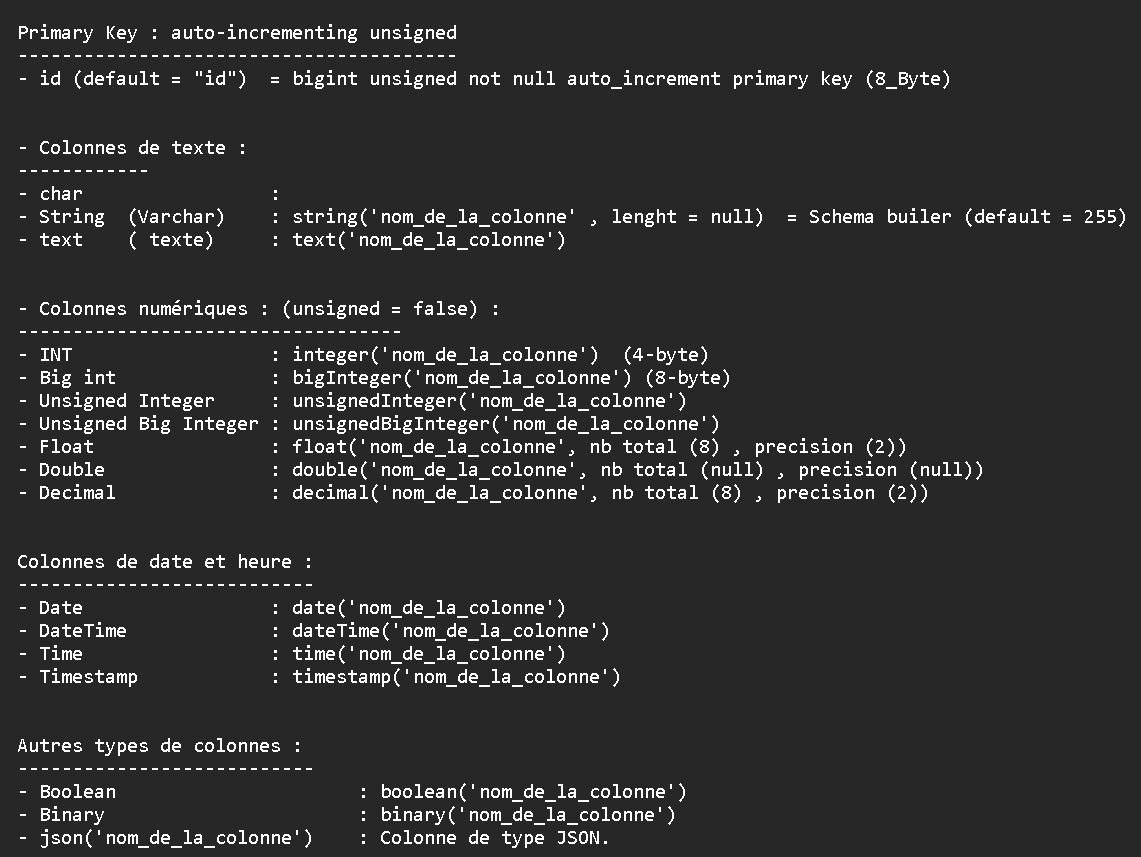
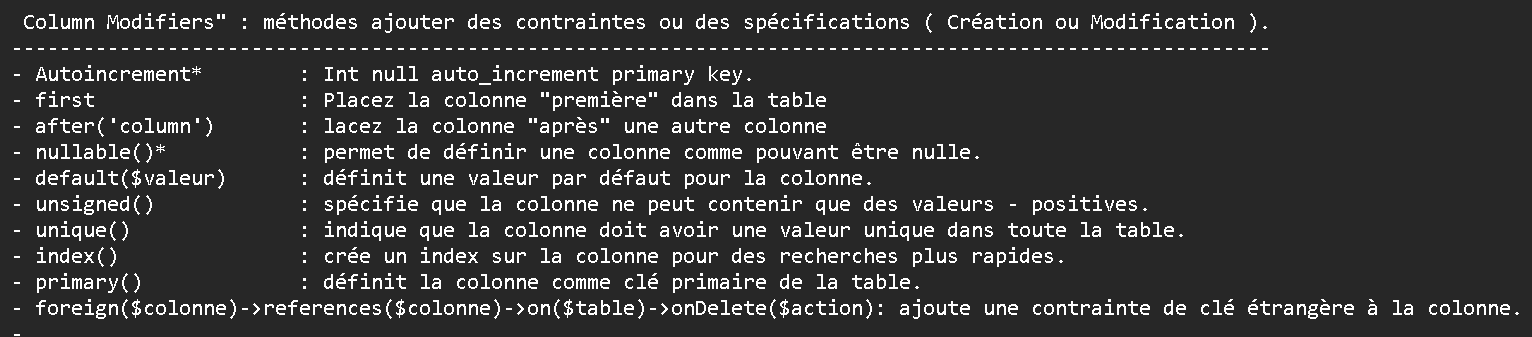
YYYY\_MM\_DDHHSS\_**Supprimer**\_(**Column**\_)\_**IN**\_(**Nom**)



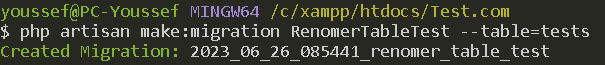


1. Type Des donnée avec Classe « Blueprint ».

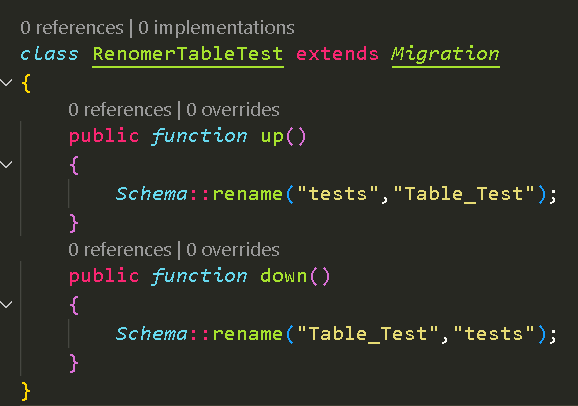
* Les méthodes du Blueprint de Laravel correspondant à différents types de colonnes.



1. Changer du nom de la Table :



* Création migration avec –table=nom\_table pour renomer nom de la table.



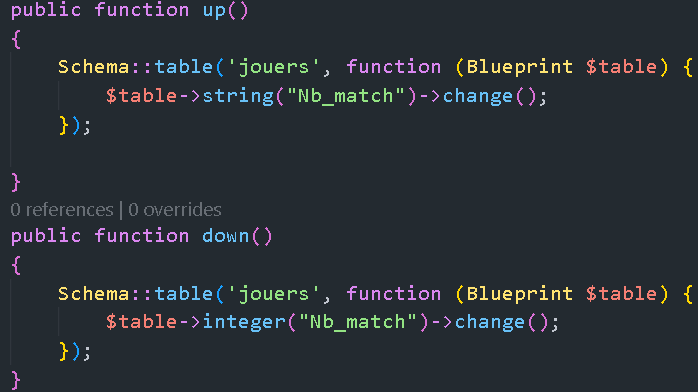
* Method **Static** de **façade** **schéma** pour renommer table.
* Dans **down** inverse opération dans **Up** pour **rollback** **migration** **changer** à **état** **initial**.

1. Ajouter Columns dans la table :

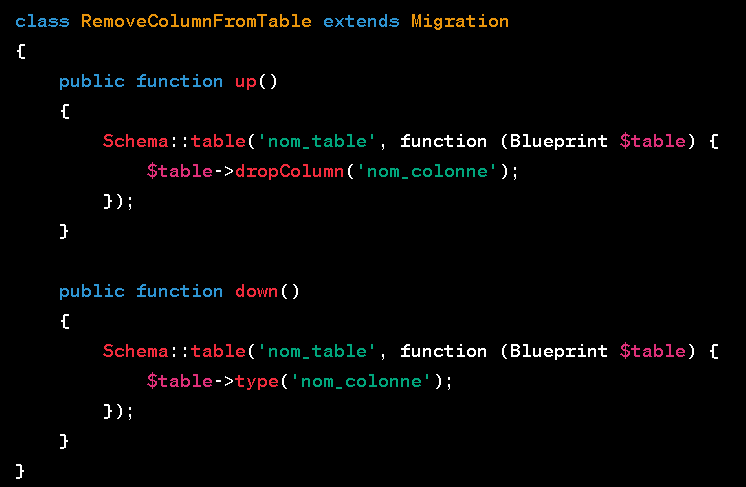


1. Modifier et renomer Columns dans la table :

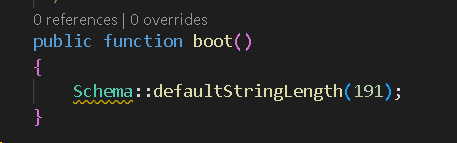




1. Supprimer Columns dans la table :

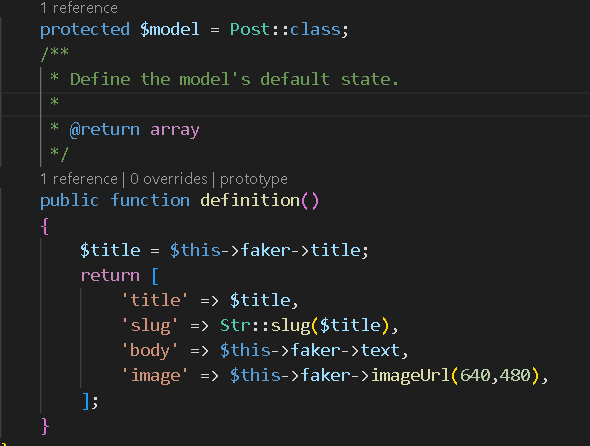






1. Add fake data using factories (factories)

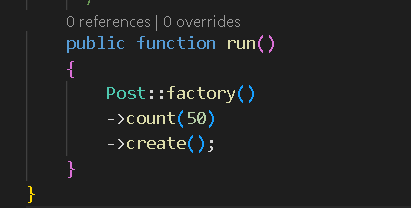
* Creer un factory pour model Post : **php artisan make:factory PostFactory** , App/database/factories
* Dans fichier PostFactories Ajouter :



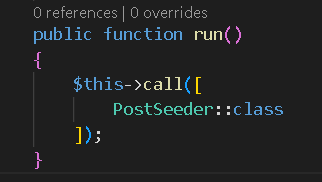
* Faker : est une bibliothèque PHP qui génère de fausses données pour vous.
* $faker->( **name** , **title** ) : name
* $faker->( **adresse** , **city** ) : adresse
* $faker-> **phoneNumber()**
* $faker-> **realText()** , **word()** = un seul mots , **words()** = plusieurs morts dans tableau , text = chaine
* $faker-> **randomDigit** = number entre 0 – 9
* $faker->**boolean()** = true ou false
* $faker-> **datetime()** = , **time()**

1. Stocker des données dans la basé donnée à l’aide classe seeder (seeders)

* Créer **seeder** à l’aide commande suivante : ***php artisan make:seeder PostSeeder***
* Tous les seeders générés par le framework seront placés dans le répertoire database/seeders
* Ajouter dans App/databse/Seeders/PostSeeder : use App\Models\Post avec modification
* Une classe seeder ne contient qu'une seule méthode par défaut : **run**. Cette méthode est appelée lorsque la commande **db:seed** Artisan est exécutée.

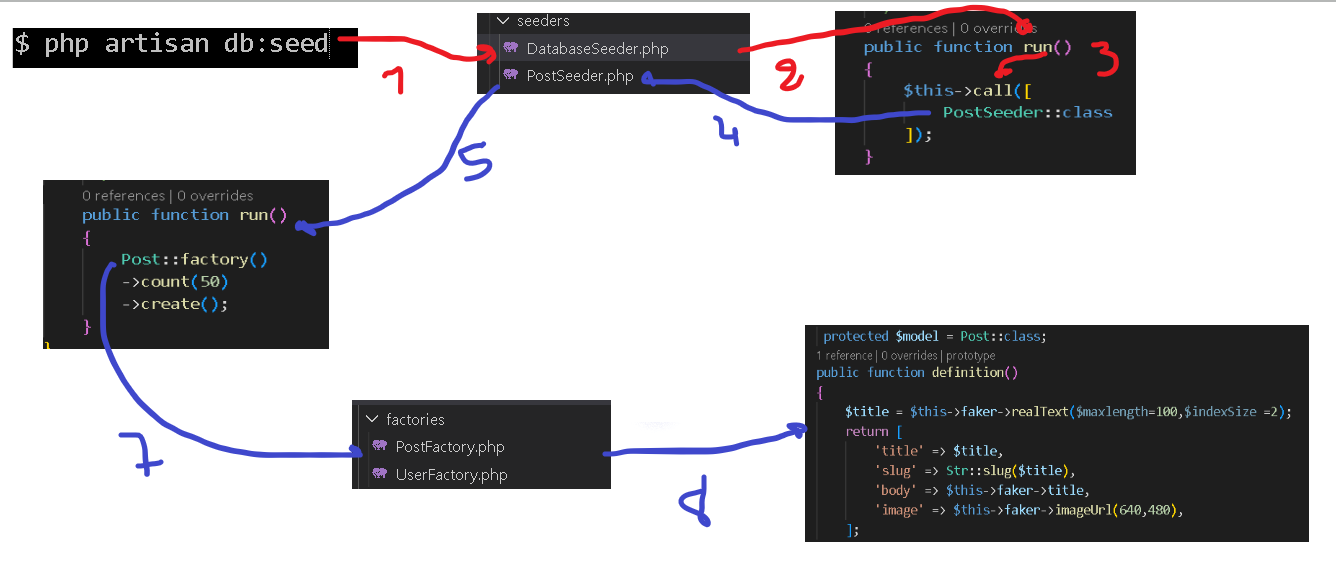


* 1. Calling Additional Seeders
* Dans Fichier App/databse/Seeders/databseSeeder.php
* La méthode call pour exécuter des classes de départ supplémentaires.



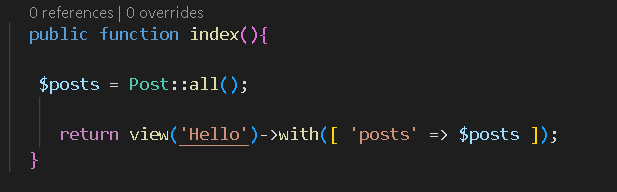
* 1. Calling Additional Seeders
* La commande **db:seed** Artisan pour amorcer votre base de données : php artisan db :seed
* La commande **db:seed** exécute la classe **Database\Seeders\DatabaseSeeder**, qui peut à son tour invoquer d'autres classes de départ.
* Exctuction un seul class : php artisan db :seed –class=PostSeeder

***Resume :***



1. Obtention des données de basse donnée et afficher dans View

* Dans HomeController ajouter : Use App\Models\Post;
* Post::all(); return arrayes de modules;



* Dans src de Image Ajouter : {{ asset($post->image) }} Recherche dans dossier public
* La asset fonction génère une URL pour un actif en utilisant le schéma actuel de la requête (HTTP ou HTTPS)

1. Query Builder :

// La table méthode renvoie une instance de ( **query builder** ) fluide pour la table donnée.

// Récupérer finalement les résultats de la requête à l'aide de la get méthode.

1. *Récupération de toutes les lignes d'une table*

$noms = DB::table("tests")->get();

1. *Récupération d'une seule ligne / colonne d'un tableau*

$noms = DB::table("tests")->where("id",2)->first();

$noms = DB::table("tests")->where("id",2)->value("nom");

$noms = DB::table("tests")->find("2"); par id

1. *Récupération d'une liste de valeurs de colonne*

$noms = DB::table('tests')->pluck('id');

$noms = DB::table('tests')->pluck('id',"nom"); colonne que la collection résultante doit utiliser comme clés

1. *Agrégats*

$count = DB::table("tests")->count(); // count

$max = DB::table("tests")->max("id"); // max

$avg = DB::table("tests")->avg("id"); // moyenne

1. *Déterminer si des enregistrements existent*

/\* if(DB::table("tests")->where("id",1)->exists()){

echo "Existe";

}

if(DB::table("tests")->where("id",1)->doesntExist()()){

echo "Existe";

}

exists() ou doesntExist() : déterminer s'il existe des enregistrements correspondant aux contraintes

\*/

1. *Personnaliser les noms des colonnes*

/\* $noms = DB::table("tests")

->select('id','nom as name')

->get();

\*/

Select : spécifier une clause "select" personnalisée pour la requête

Distinct : forcer la requête à renvoyer des résultats distincts :

1. *Ajouter une colonne à sa clause select existante, vous pouvez utiliser la addSelectméthode :*

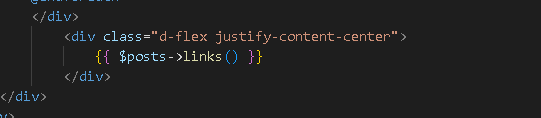
$qeury = DB::table("tests")->select('id');

$noms = $qeury->addSelect("nom")->get();

1. Paggination
2. Dans HomeControler :



1. Dans fichier Html Ajouter cette Div :



1. Ajouter dans fichier : **App/AppServiceProvider**

use Illuminate\Pagination\Paginator;

 public function boot()

    {

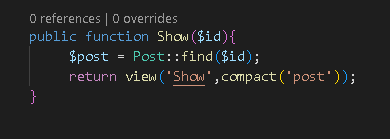
        Schema::defaultStringLength(191);

        Paginator::useBootstrap();

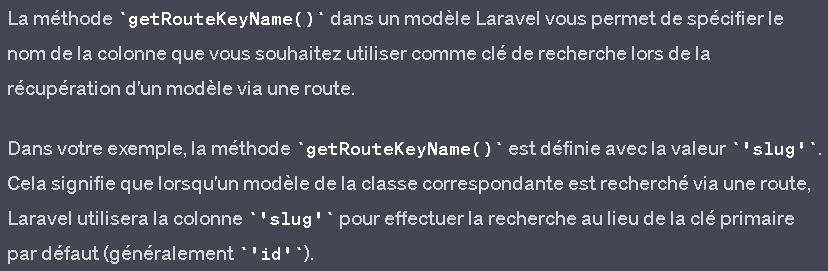
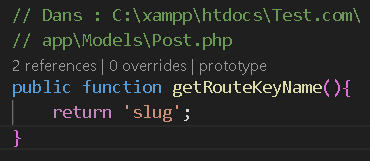
    }

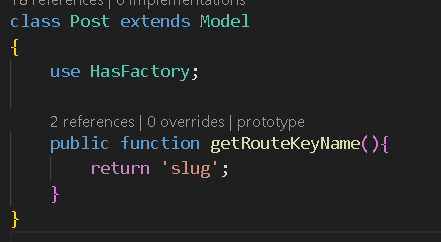
1. Passe un autre page pour afficher les information Post :





* Utilisation Slug comme clé de rechercehe :
* C:\xampp\htdocs\ Test.com\app\Models\Post.php :

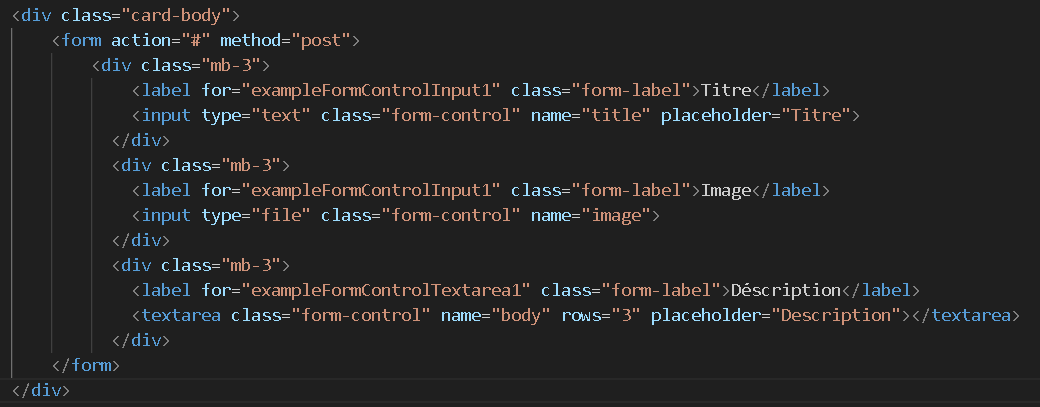




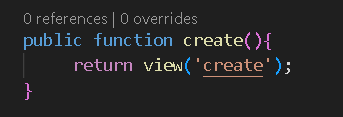
1. Forms ( CRUD ) :
2. Création d’ un enregistrement :

* Dans web config ajouter d’un route vers view create :





* Creation d’un view createpost :
* Dans controler ajouter function return view :



* Traitement des données d’un formulaire et ajout dans une base de données :

1. Ajoute d’un router vers fonction pour store des données



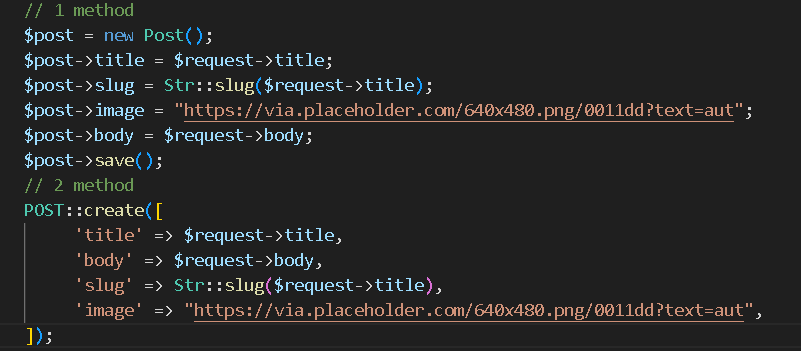
1. Dans view « Cérate post »

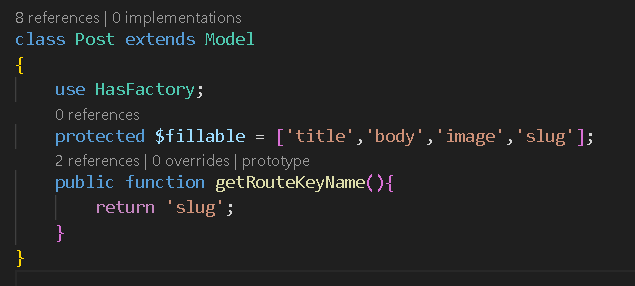


1. Ajouter Tokken @CSRF dans view Create après balise <form > : **Problème de Page\_Expiration**

* @csrf est une directive utilisée pour générer un jeton **CSRF** (Cross-Site Request Forgery) dans les formulaires.
* Le jeton CSRF est inclus dans la requête et Laravel le vérifiera automatiquement avant de traiter la requête. pour s'assurer que la requête est légitime et provient du même site.
* Il est important d'inclure la directive **@csrf** dans tous les formulaires POST générés par **Laravel** pour assurer une protection appropriée contre les attaques CSRF.



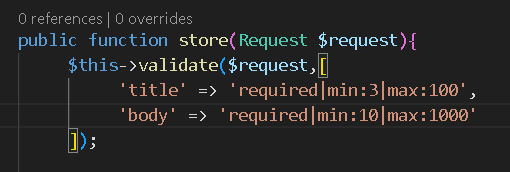
* Enregistré les données dans la base de données (DB) :
* Dans Model Ajouter la solution de problème mass assignment.
* La propriété fillable spécifie quels attributs de modèle peuvent être affectés en masse lors de l'utilisation de méthodes telles que create ou update :



* Dans Example seules les colonnes **title** et **body** et **image** et **slug** : est **autorisé** à **cérate** ou **update** .

1. Validation de la demande de formulaire

**\*** Dans HomeControler ajouter function **validate**() : est utilisée pour valider les données entrantes dans une requête HTTP.

****

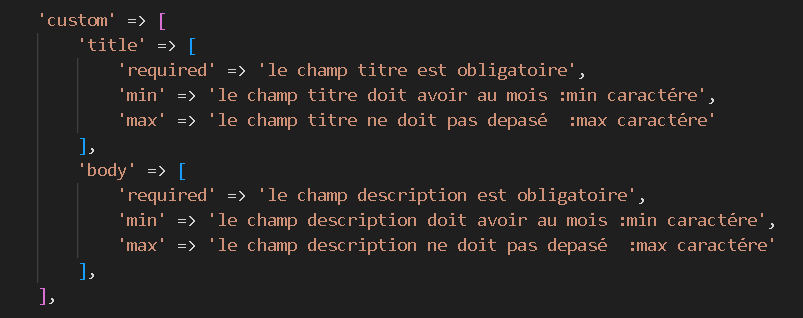
\* Dans **Blade** Cérate ajouter Code Suivante : pour afficher les **erreurs**

**

1. Personnalisation des messages d'erreur :

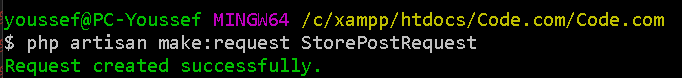
- Dans fichier : **ressources/lang/validate()** :

- Le variable **:min** et **:max** change avec le valeur déclare dans la fonction Validate du fichier **homeControler**

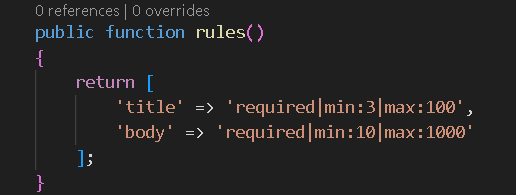


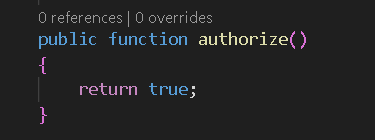
1. Validation de la demande de formulaire :

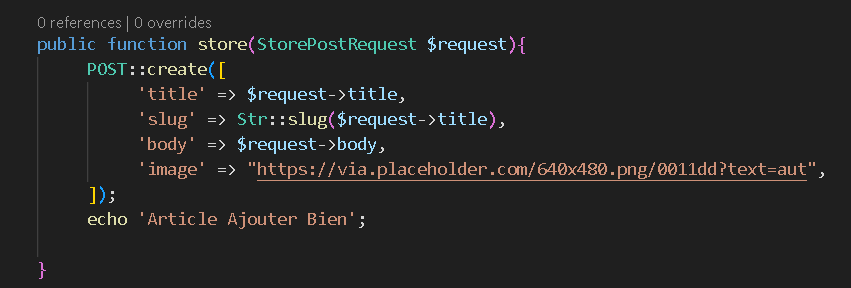
* créer une classe de requête de formulaire

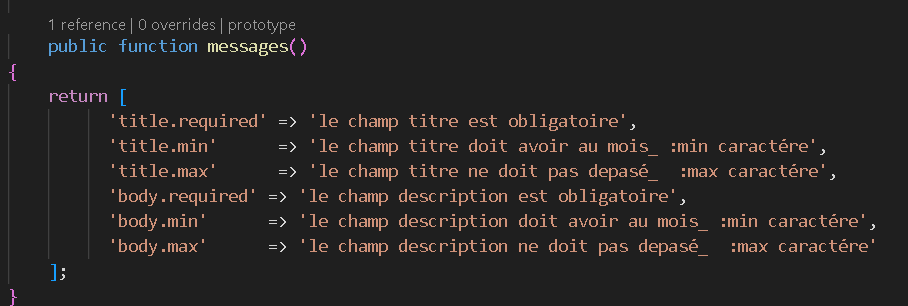


* La classe de demande de formulaire générée sera placée dans le **app/Http/Request**.
* Chaque requête de formulaire générée par Laravel a deux méthodes : authorizeet rules.



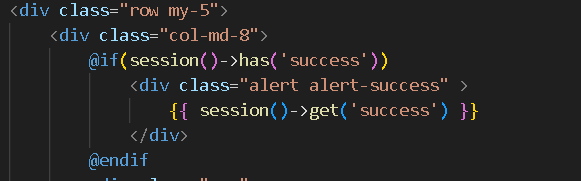






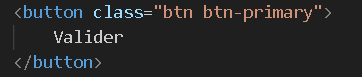




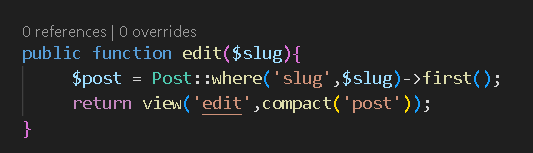
* Redirection vers page d’acceuil avec affichagge flash message pour ajouter

1. Modifier les donnée :
2. Creation d’un view modifier (Edit) :

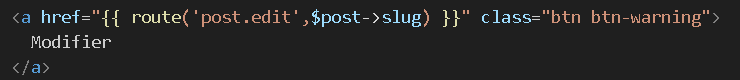




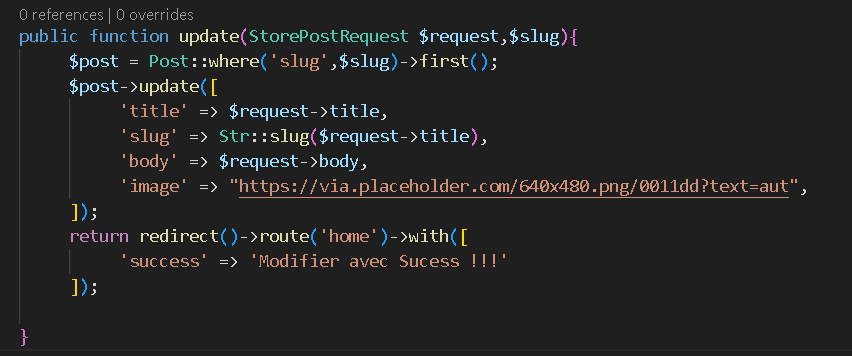
1. Ajouter Fonction dans Homme Contrôler pour return view Edit avec post :



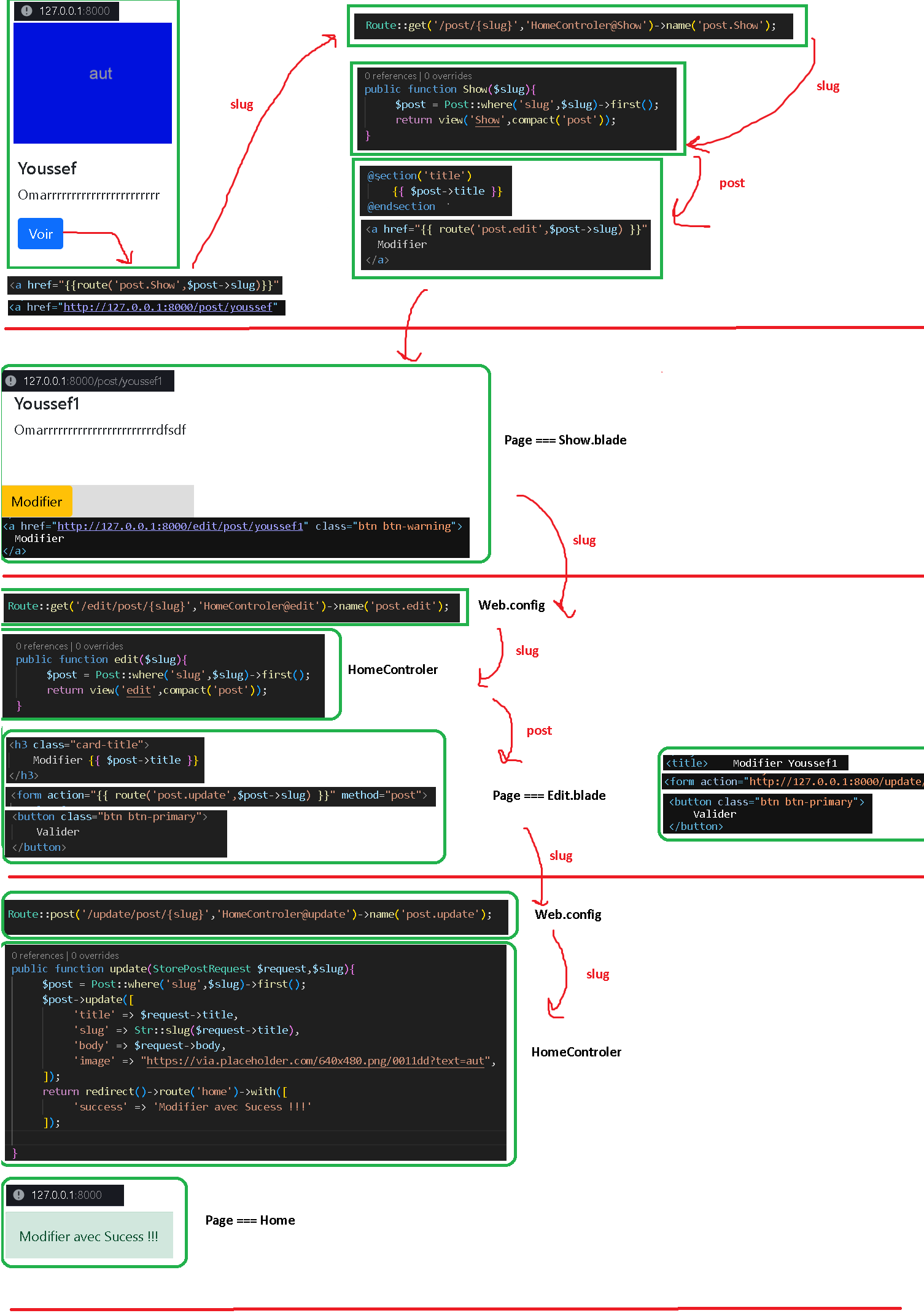
1. Route pour Édit :
2. Dans 4) Button Modifier en view Show :



1. Quand click sur Valide les donnée Modifier :





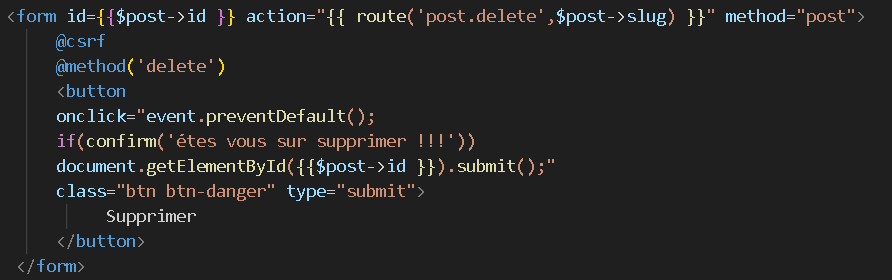


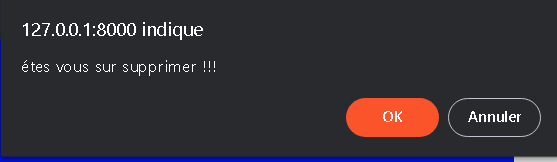
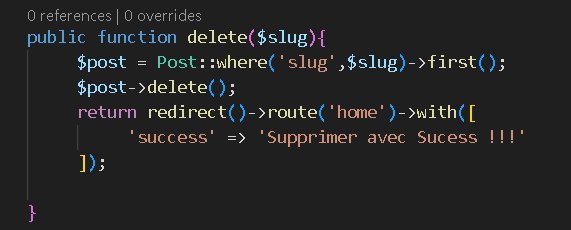
SERVEUR

SERVEUR

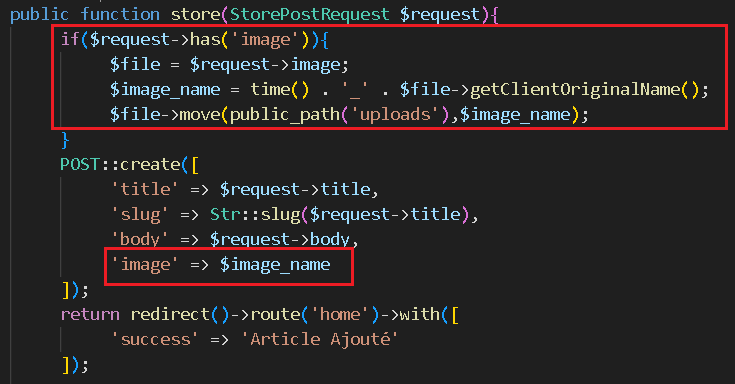
SERVEUR

1. Supprimer les donnée :





1. Uploade les fichiers :





* Ajouter attribut **enctype** dans form qui envoyer pour ajouter les données

L'attribut **enctype** est utilisé pour spécifier le type d'**encodage** à utiliser lors de l'**envoi** de **données** à partir d'un **formulaire** **HTML**. Cependant, lorsque vous avez besoin de soumettre des fichiers ou des données binaires, vous devez utiliser un encodage différent, tel que **multipart/form-data**.



La fonction **asset()** prend le chemin relatif du fichier par rapport au répertoire public et génère l'**URL** complète en ajoutant le **nom** de **domaine** et le **chemin** de base de votre application.

* **http://127.0.0.1:8000/post/aaaaaaaaaaaaaaa**

**public/uploads/aaaaaaaaaaaaaaa**

1. Update les Images :





* Ajouter attribut **enctype** = "**multipart/form-data**"

**Unlink()** : La **méthode** **unlink()** est **utilisée** pour **supprimer** un **fichier** du **système** de fichiers.

1. Validation les fichiers ajoutés :

* Dans fichier App/http/Request/StorePostRequest.php



1. ) Éloquent : ORM :

* Un mappeur relationnel objet (ORM) : Un **ORM** (**Object-Relational Mapping**) est un logiciel permettant la **conversion** des **données** **relationnelles** d’une base de données en objets afin de pouvoir les manipuler dans notre application en **POO** (**Programmation Orientée Objet**).
* Création :

**Model** + **Migration** + **factory** + **seeder** + **controller** + **ressources** :

* php artisan make:model test -f -m -s -c -r
* ***Nom de table :***
* **Dynamique : nom** **pluriel** de la classe sera utilisé comme **nom** de **table**. Model (**Eleve**) = **Eleves**
* **Manuellement :** protected $table = Eleves;
* ***Primary Key :***
* **Dynamique :** Clé primaire nommée id
* **Manuellemsent :** protected $primaryKey = 'flight\_id';
* **Eloquent** suppose que la clé primaire est une valeur entière incrémentielle
* N'est pas un **entier** et non **incrémentielle**

Public $incrementing = false;

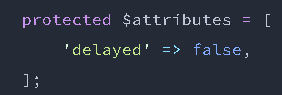
protected $keyType = 'string';

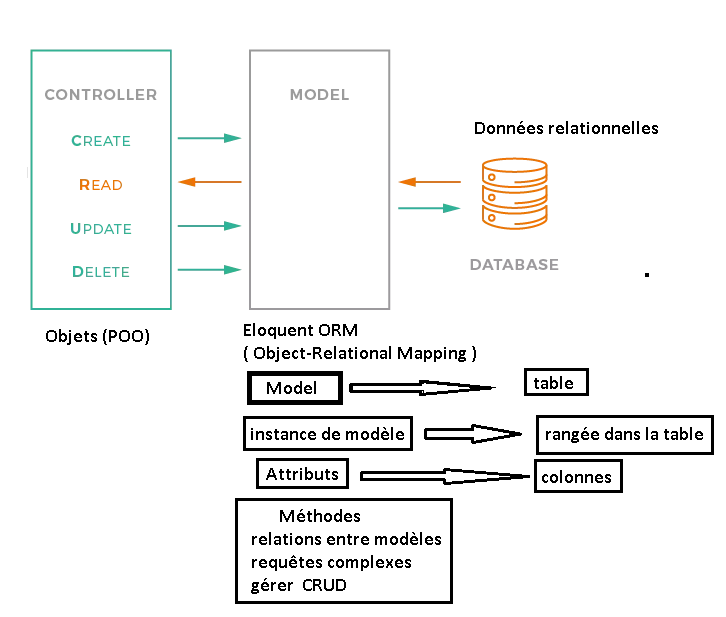
* ***Timestamps:***
* Eloquent définira automatiquement les valeurs de ces colonnes lors de la création ou de la mise à jour des modèles.
* Vous ne souhaitez pas que ces colonnes soient automatiquement : public $timestamps = false;
* Personnaliser le format : protected $dateFormat = 'U';
* Personnaliser les noms des colonnes :

const CREATED\_AT = 'creation\_date';

const UPDATED\_AT = 'updated\_date';

* ***Connexions à la base de données:***
* Tous les modèles Éloquent utiliseront la connexion à la base de données par défaut configurée pour votre application.
* Spécifier une connexion différente : protected $connection = 'sqlite';
* ***Valeurs d'attribut par défaut :***





1. Laravel Jetstream :
2. Installation Package

* **Composer** **require** laravel/jetstream:\* : est utilisé pour installer le **package** **Laravel** Jetstream dans votre **projet** **Laravel**.
* **php** **artisan** jetstream:install **livewire**
* **npm** install
* npm run dev
* php artisan migrate

1. Désinstaller Package jetstream :

* Composer remove laravel/jetstream

Résumer :

Tailwind CSS = Un framework CSS

* Il ne fournit aucune fonctionnalité dynamique ou interactive en soi, mais se concentre uniquement sur le style.
* Tailwind CSS fonctionne bien avec n'importe quel framework JavaScript frontal, y compris Livewire.

Livewire = Livewire est un framework complet

* Livewire est un framework complet pour Laravel qui simplifie la création d'interfaces dynamiques
* la création d'interfaces Web dynamiques à l'aide de Laravel et PHP.

Laravel Sanctum is a popular package for Laravel, a PHP web framework.

Jetstream est conçu à l'aide de Tailwind CSS et offre votre choix d'échafaudages Livewire.

Après modifier le fichier

resources\views\layouts\guest.blade.php et resources\views\layouts\app.blade.php

Supprimer :

@vite(['resources/css/app.css', 'resources/js/app.js'])

Ajouter :

<script src="{{ asset('js/app.js') }}" defer></script>

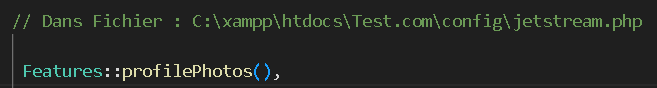
link href="{{ asset('css/app.css') }}" rel="stylesheet">

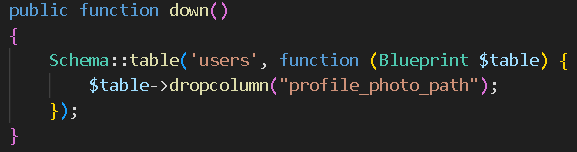
1. Laravel Mix :

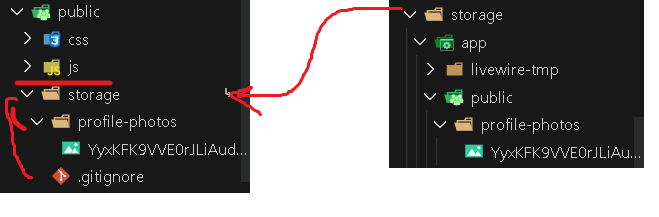
* Laravel Mix, vous pouvez facilement compiler et regrouper vos ressources CSS, JavaScript et autres.
* Installing Laravel Mix : **npm install**
* Running Mix : **npm run dev**
* Login et register:
* **Modifier App\_Url dans fichier .env :**



* **Modifier Users Login Ajouter Photo de profil :**

****

* **Ajouter column dans Table Users :**
* **php** **artisan** make:**migration** add\_photo\_path\_column\_to\_users --table="users"
* **php artisan storage:link :**
* Créer un lien symbolique entre le répertoire de stockage (storage) et le répertoire public (public) :

****