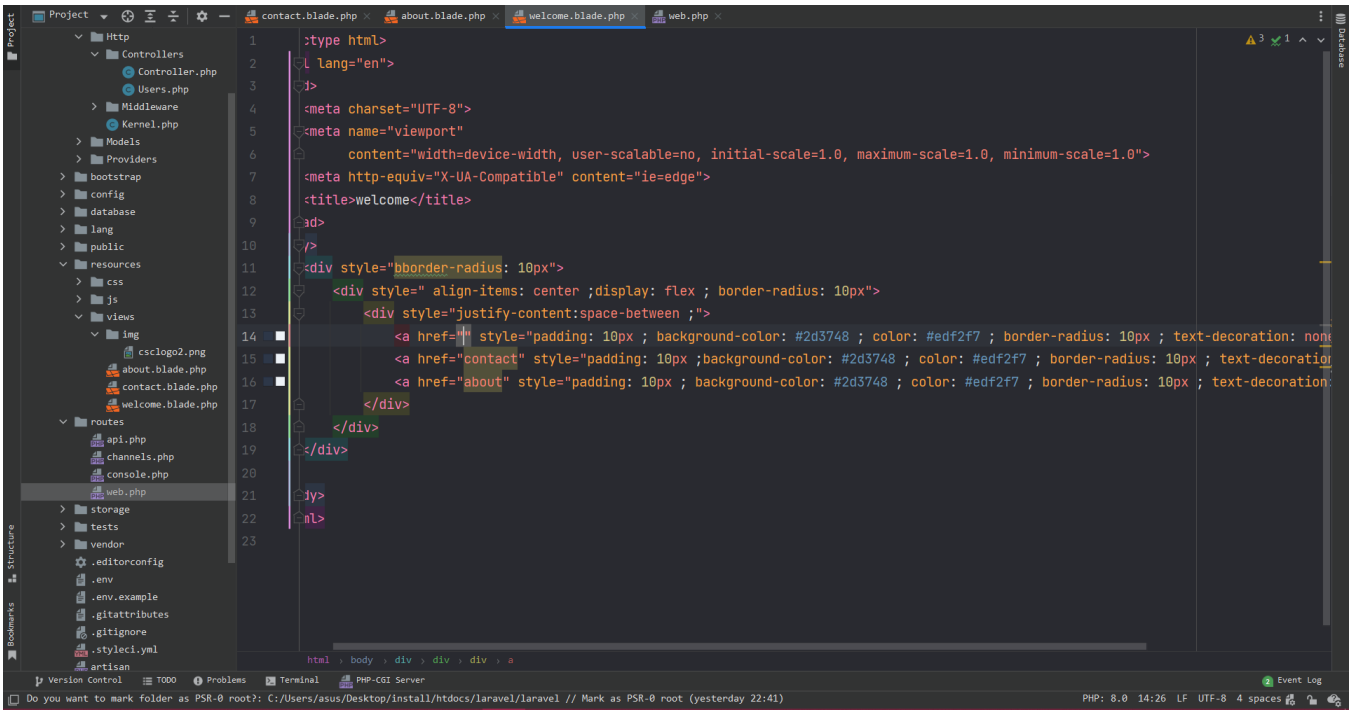
Laravel 9

Achraf KHABAR

vendredi 25 février 2022
00:20

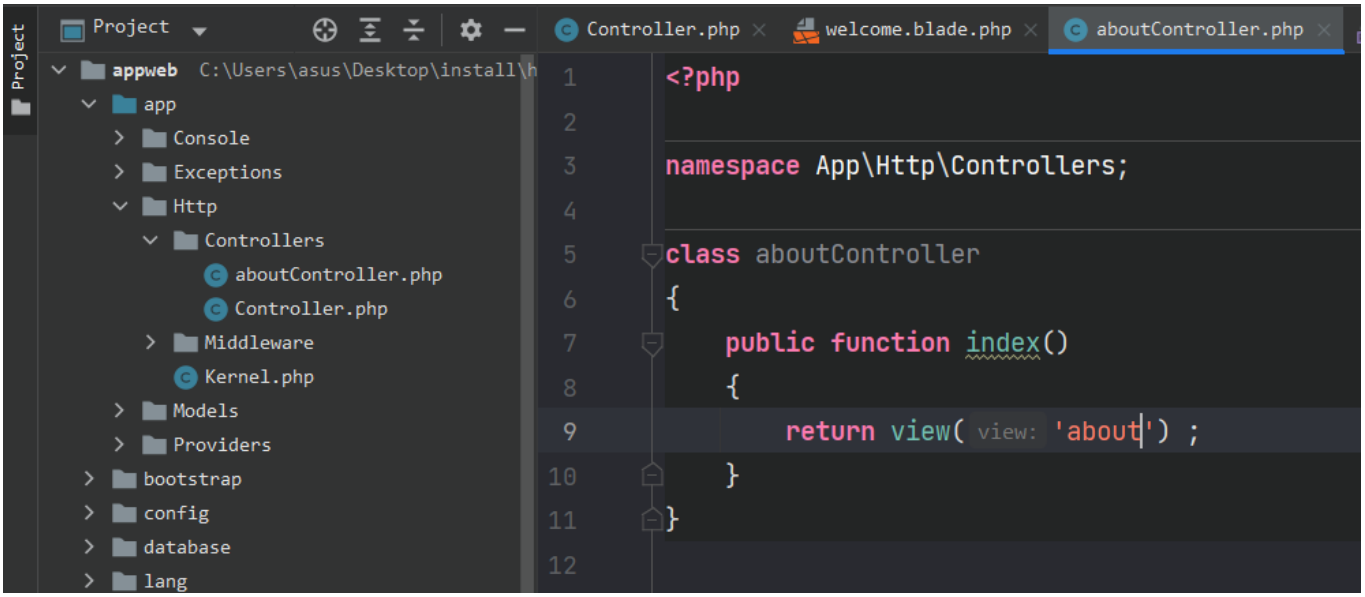
Routing :



I can return such things not only views :

```
{"title":"mon super title","description":"ma super description "}
```

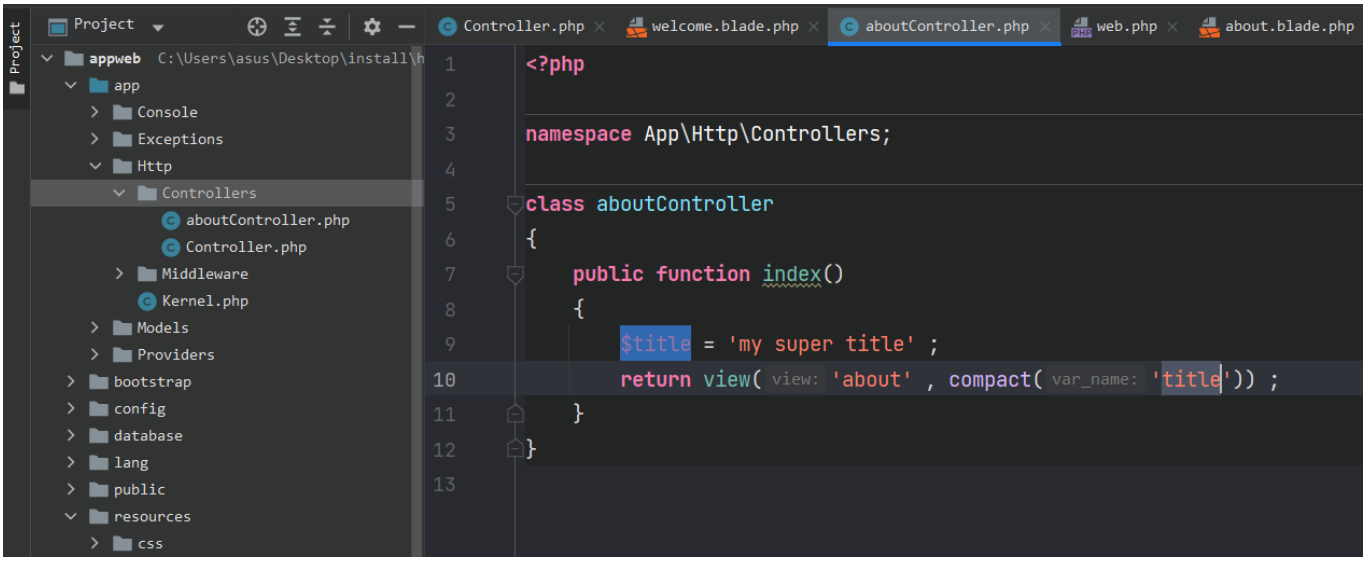
Controllers :



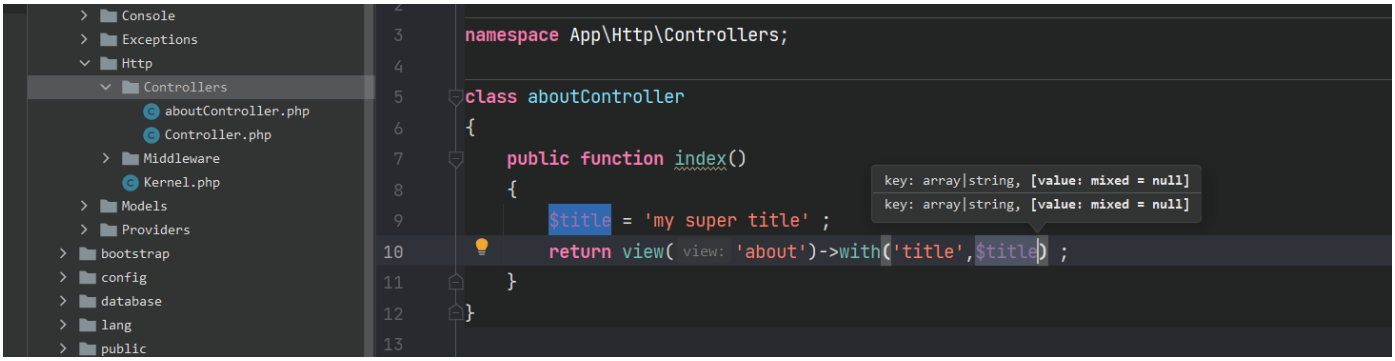
An other way :

```php
use App\Http\Controllers\aboutController;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. Thes
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get( uri: '/about', [aboutController::class ,'index']) ;
```

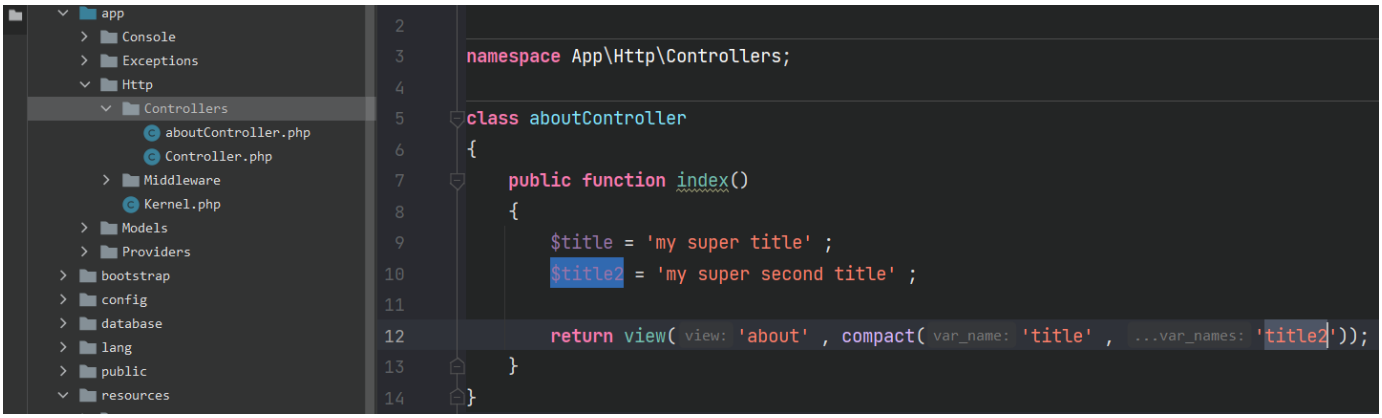
How to use a variable in laravel :

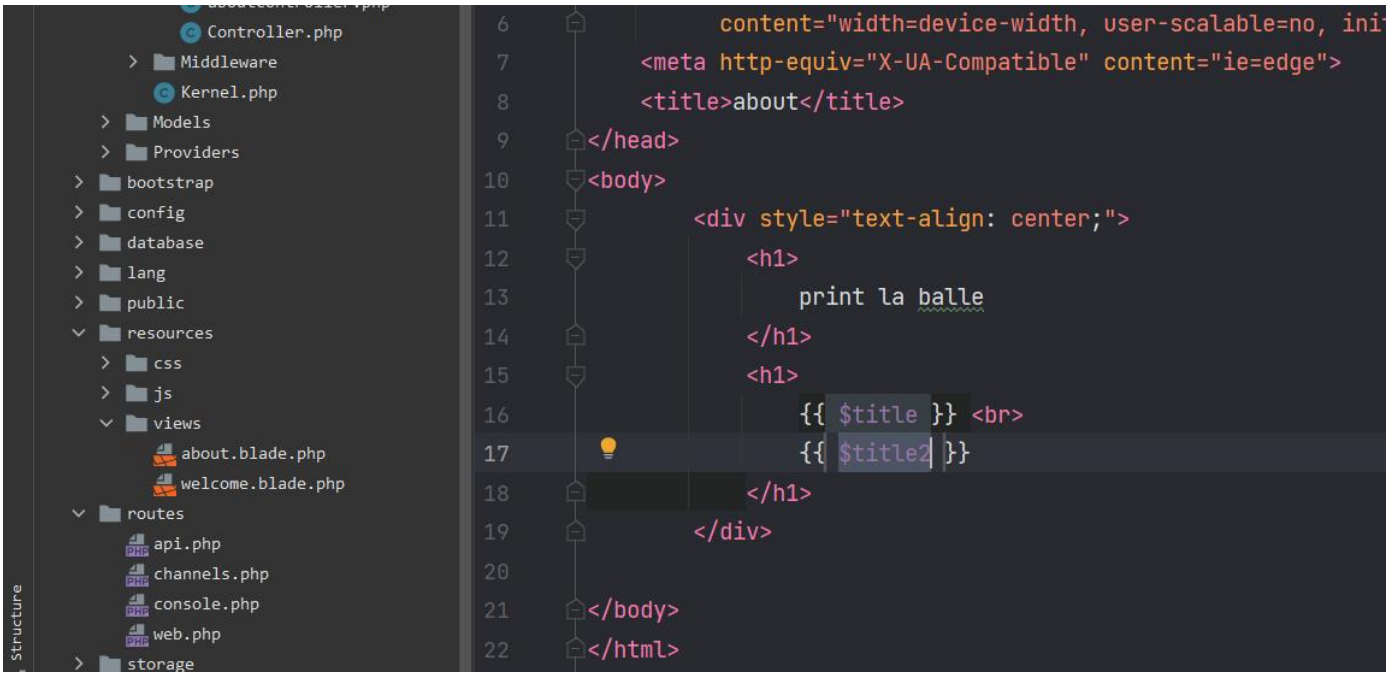


An other way :



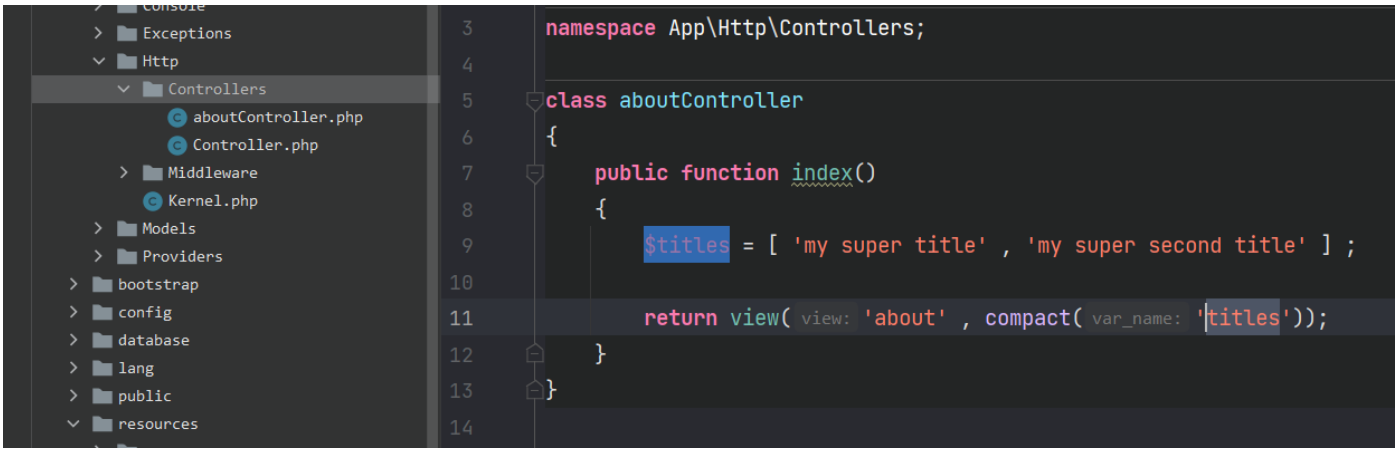
How to pass more than variable :



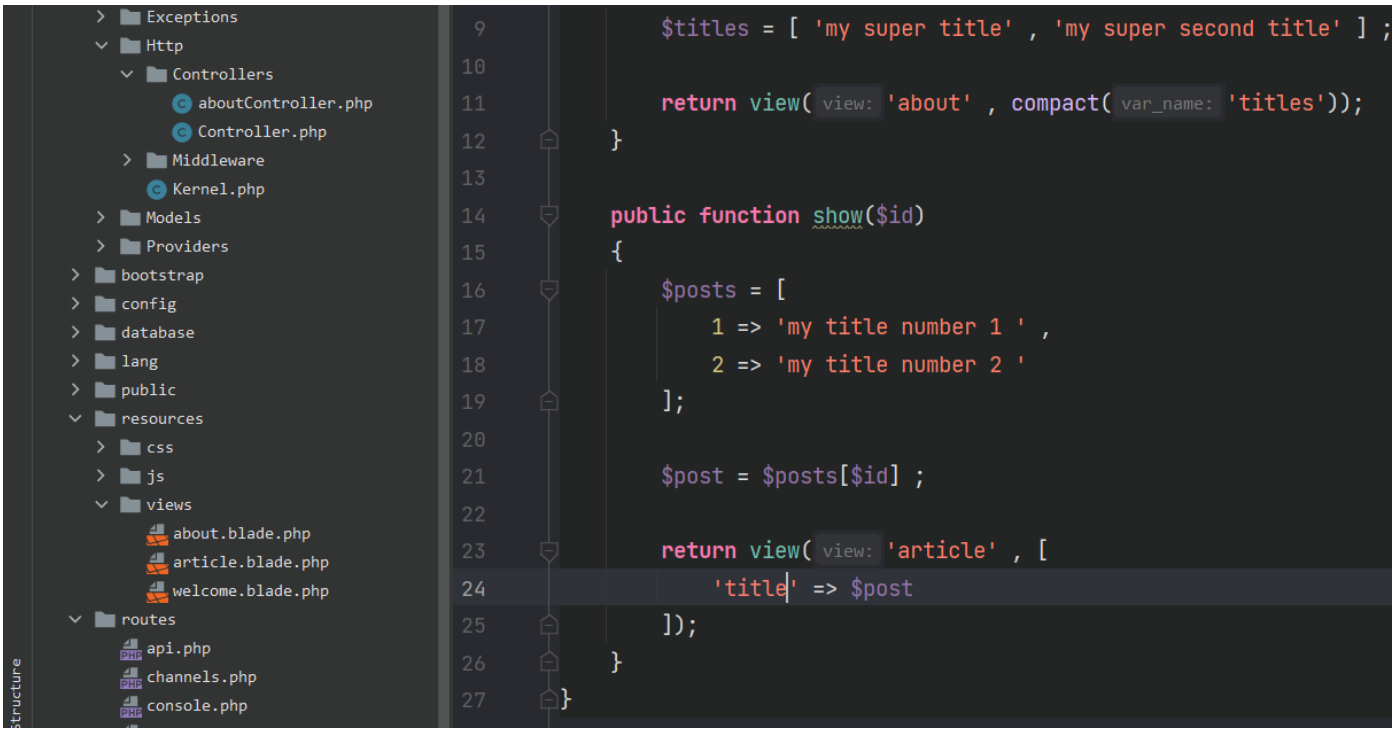


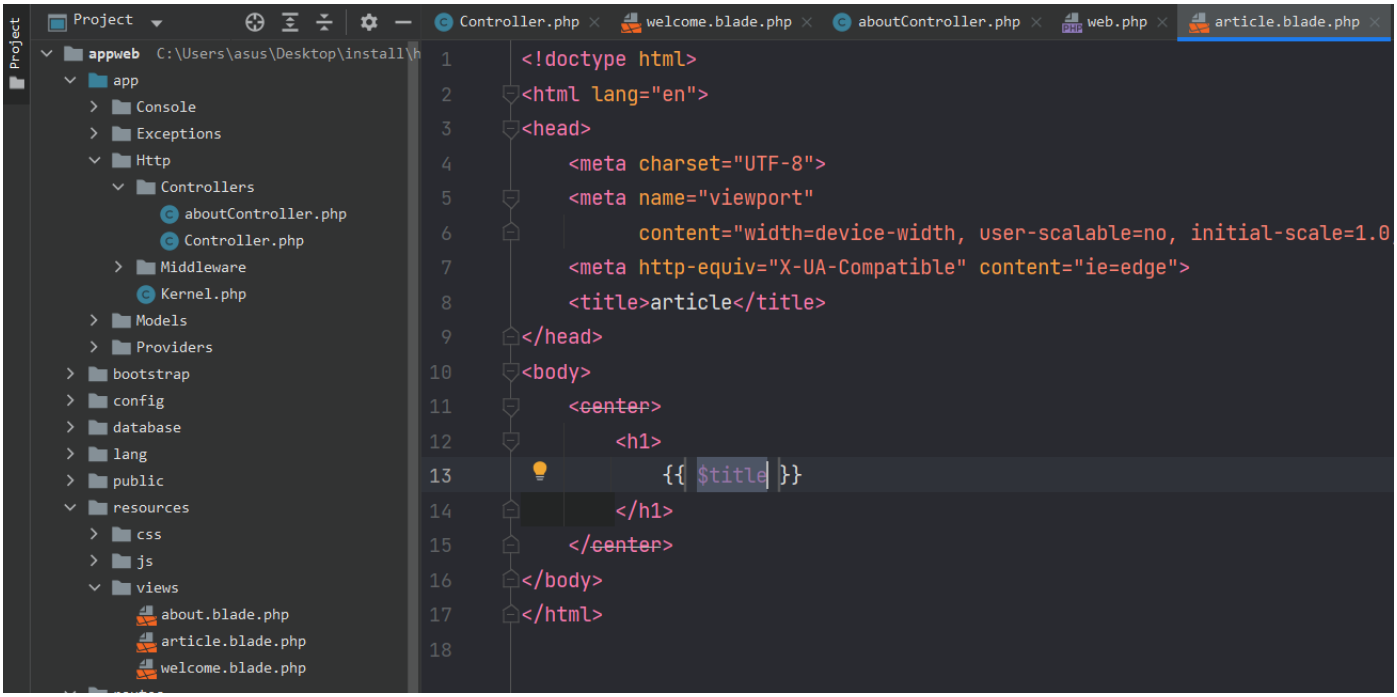
Or we can do an array :

How to pass them within array :



Blade and views :

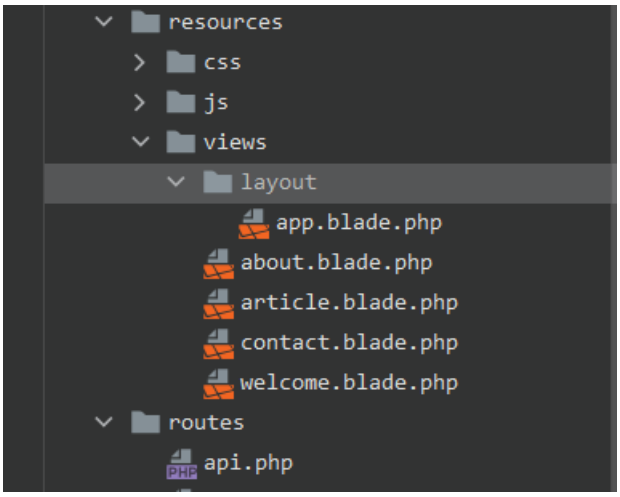




Add condition :

Aussi :

Adventage of .blade (how to not repete the coman code between views) :



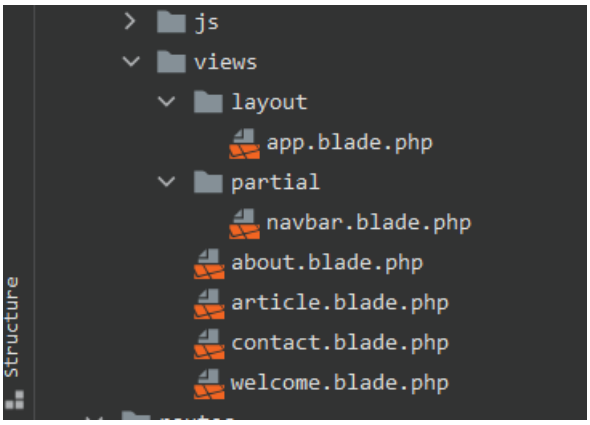
Controller.php | welcome.blade.php | app.blade.php | aboutController.php | article.blade.php | contact.blade.php | web.php | about.blade.php

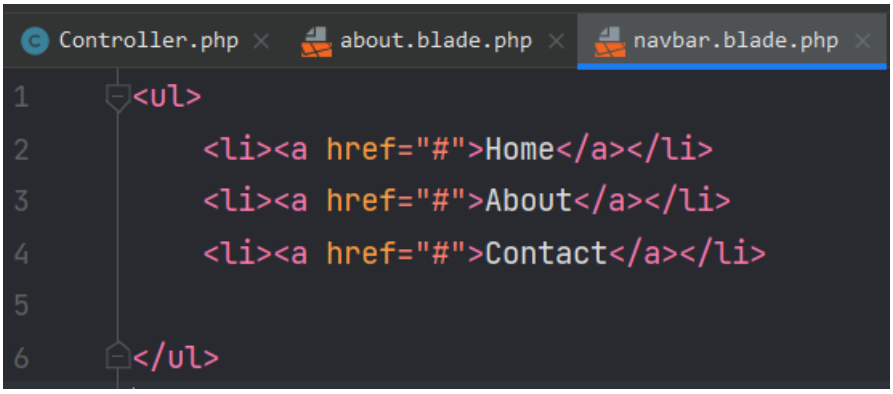
```blade
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>AKC</title>
</head>
<body>
    @yield('content')
</body>
</html>
```

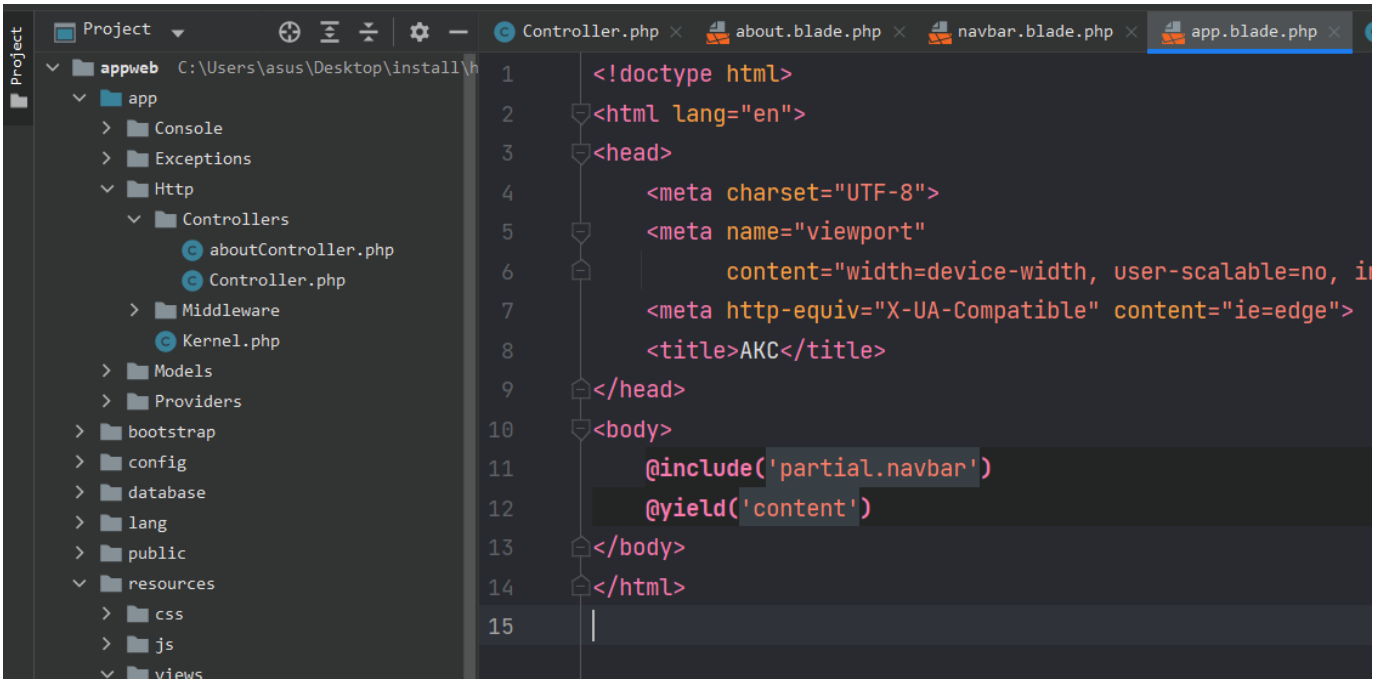
Controller.php | about.blade.php | app.blade.php | aboutControll

```blade
@extends('layout.app')

@section('content')
    <div class="premiere-div" style="...">
        <h1 class="grand-titre">
            print la balle
        </h1>
        <h1>
            @foreach( $titles as $title)
                <h6>
                    {{ $title }}
                </h6>
            @endforeach
        </h1>
    </div>
@endsection
```

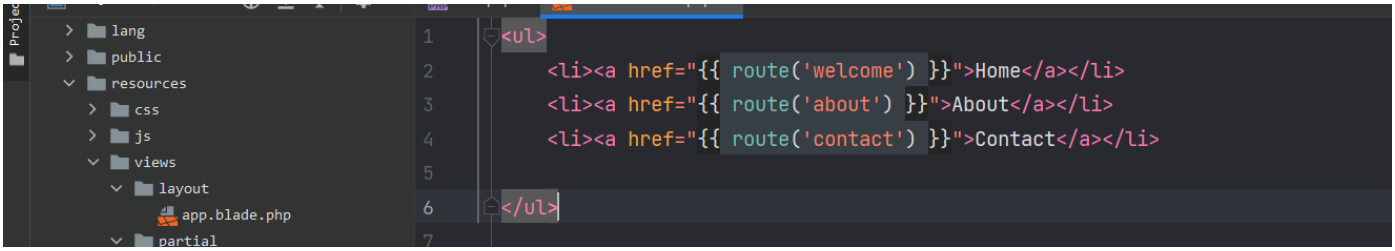
Include in blade :



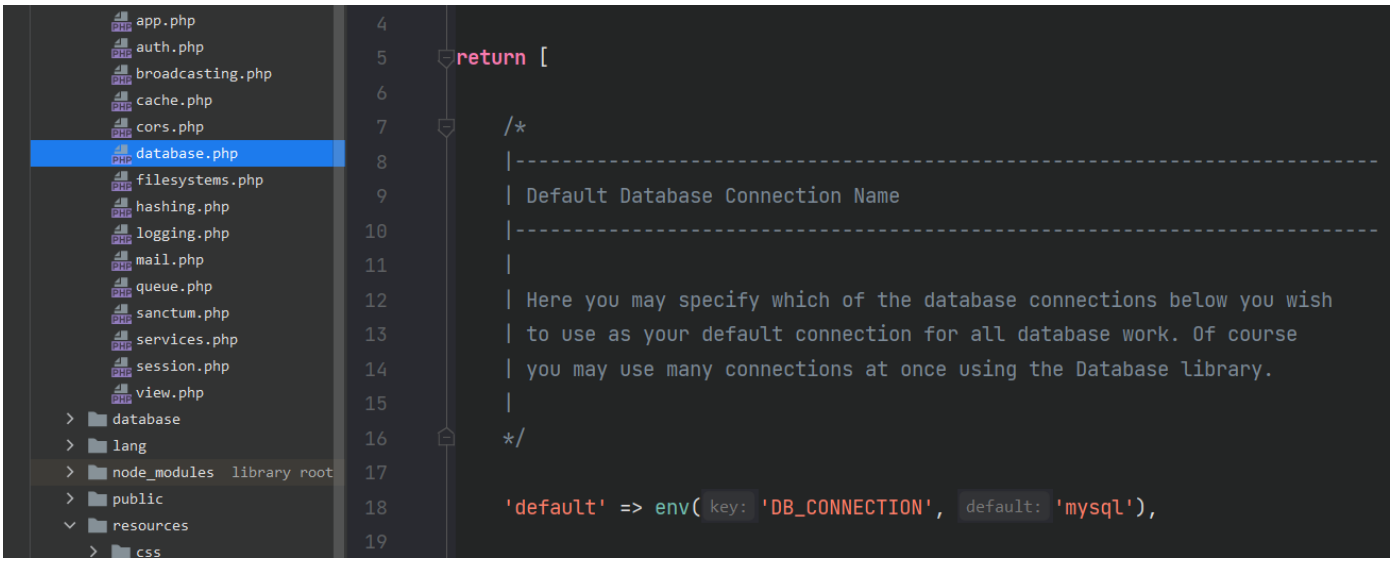


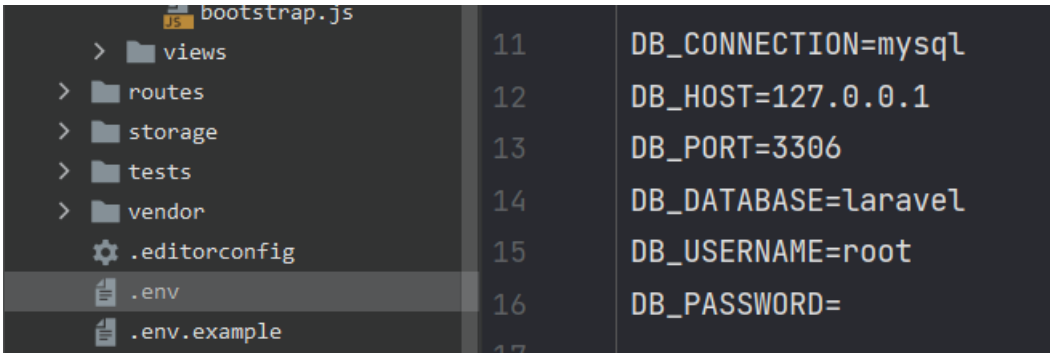


How to name routes :



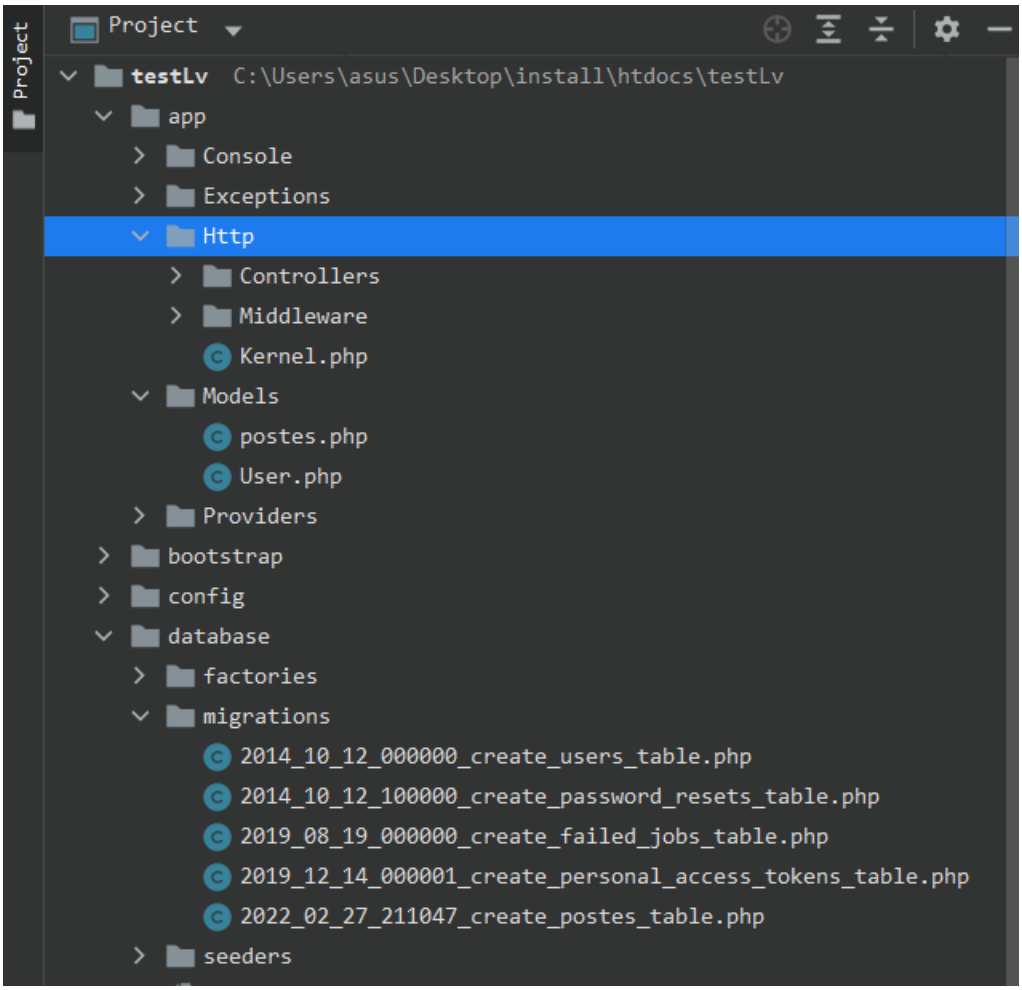
Base de donnee avec laravel :



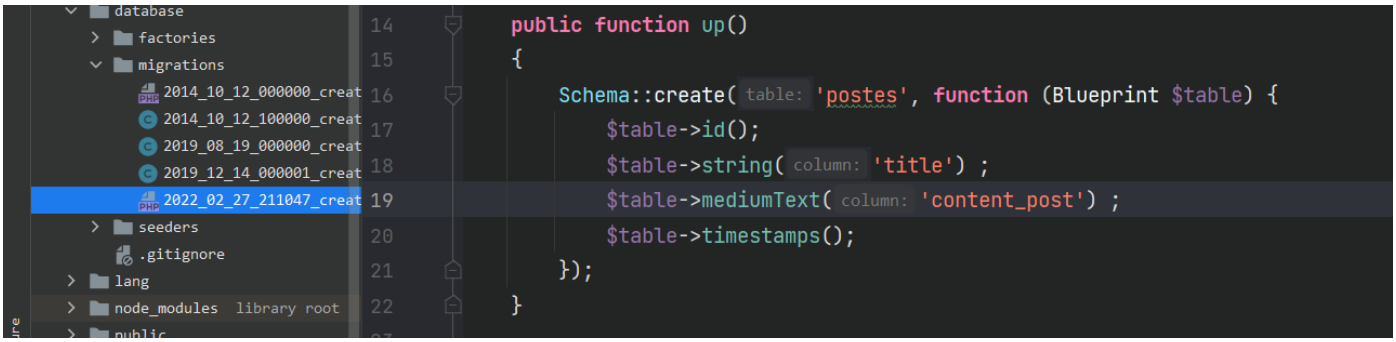


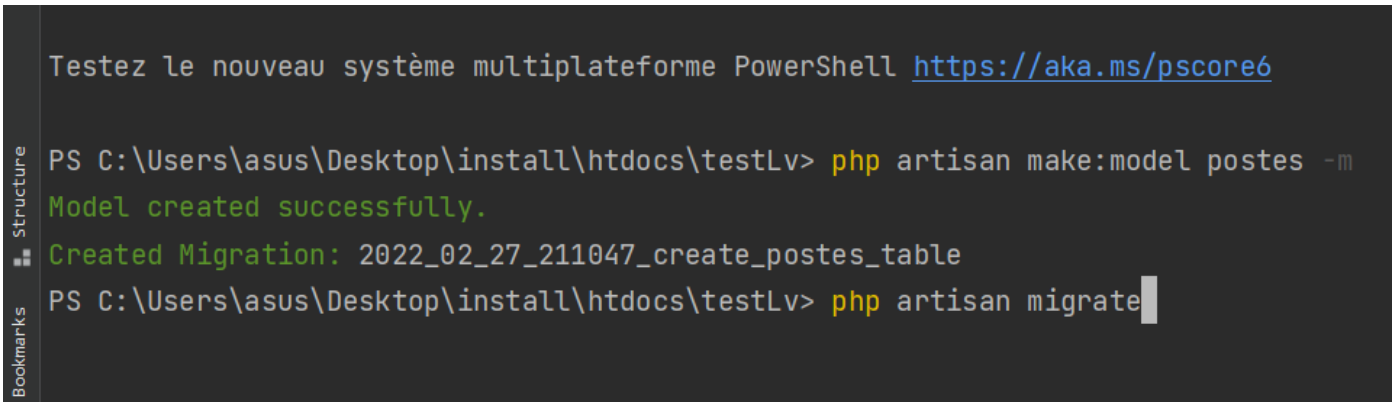
How to create a model and his migration (migration in order to create tables) :

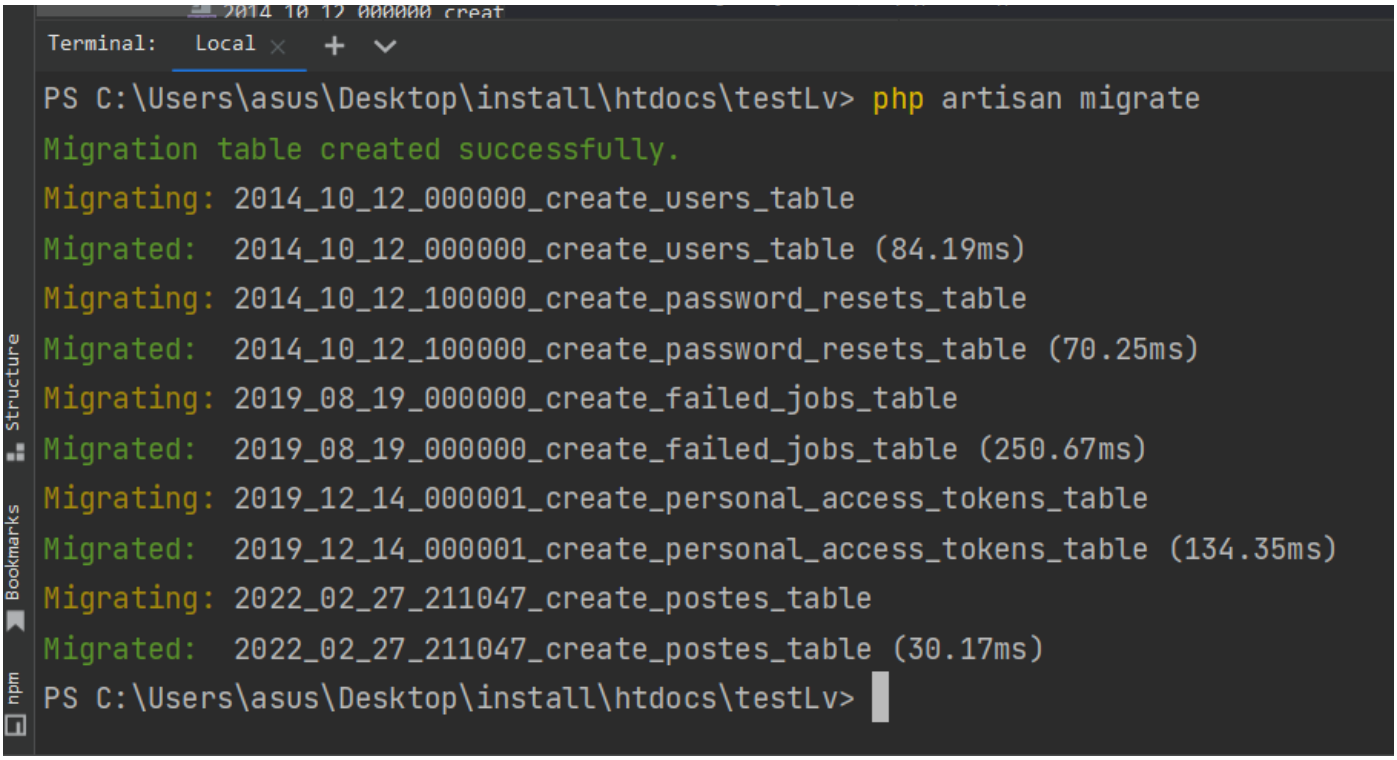
```
PS C:\Users\asus\Desktop\install\htdocs\testLv> php artisan make:model postes -m
```



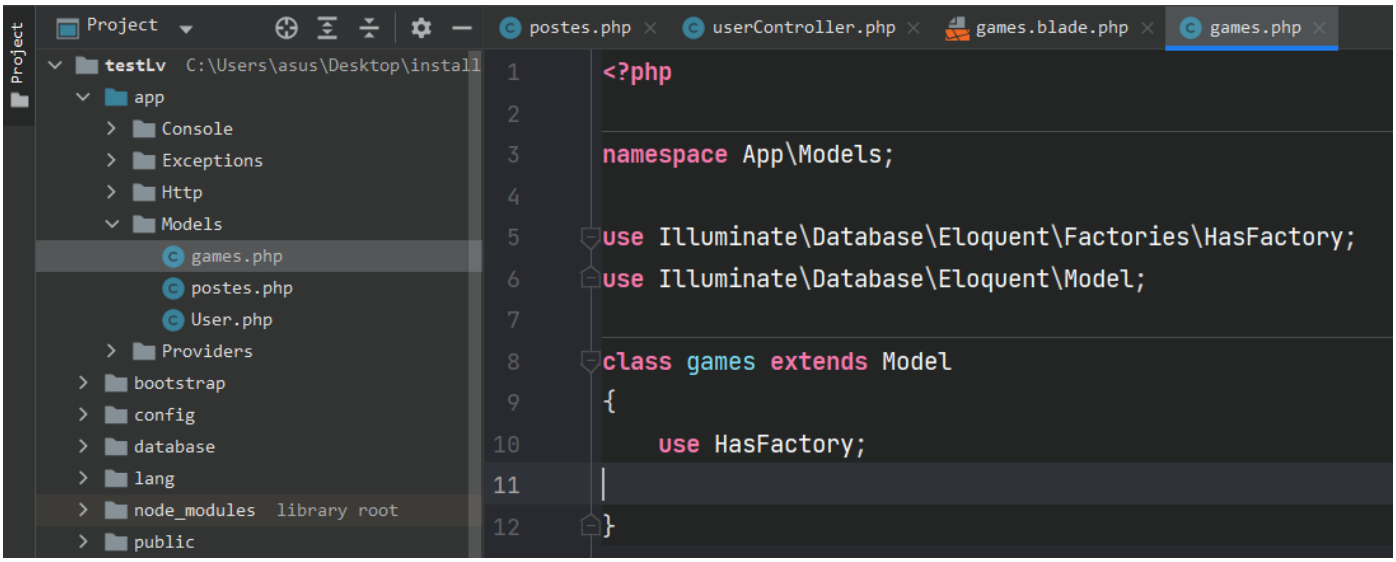
Create a database in laravel 8 :



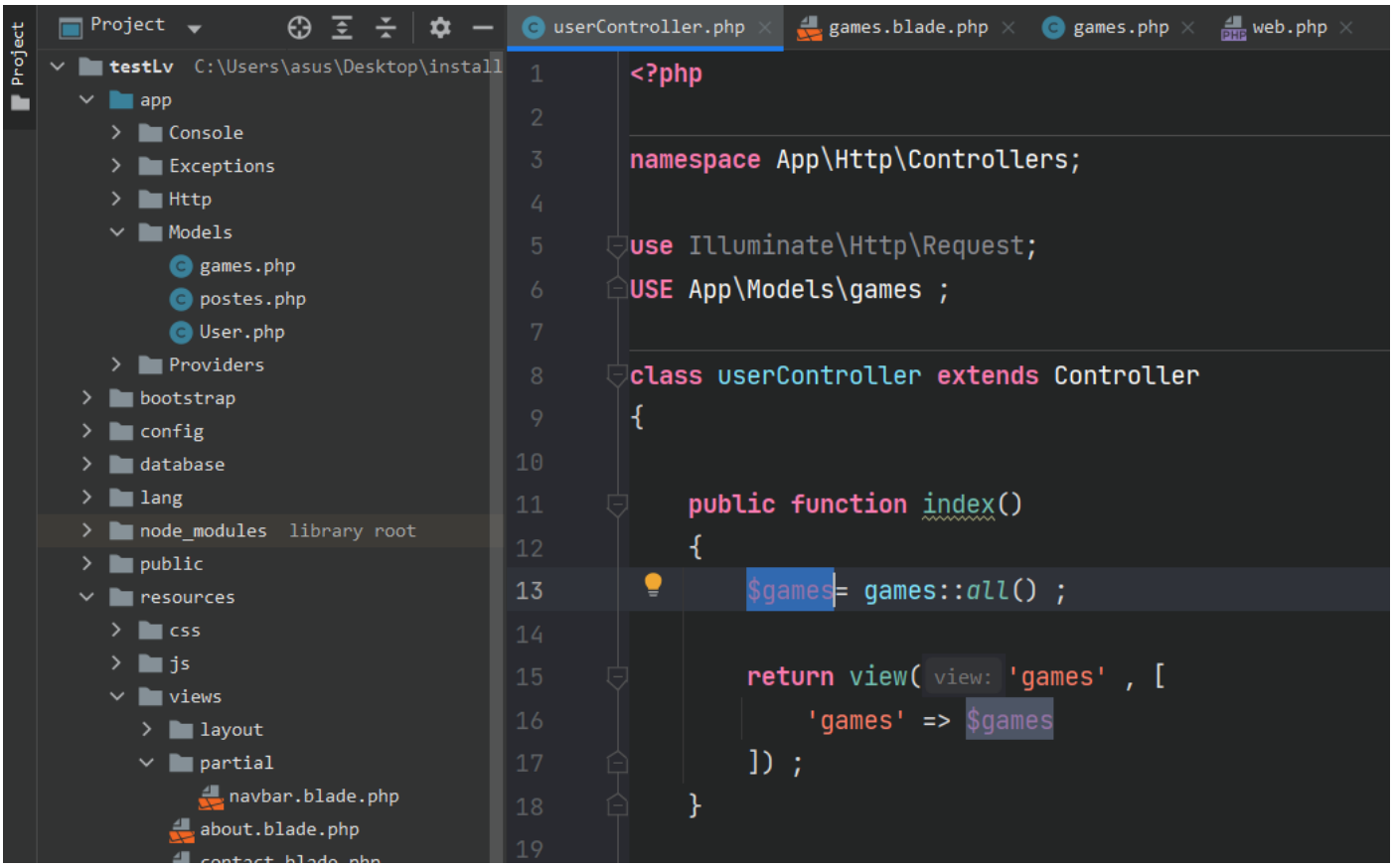


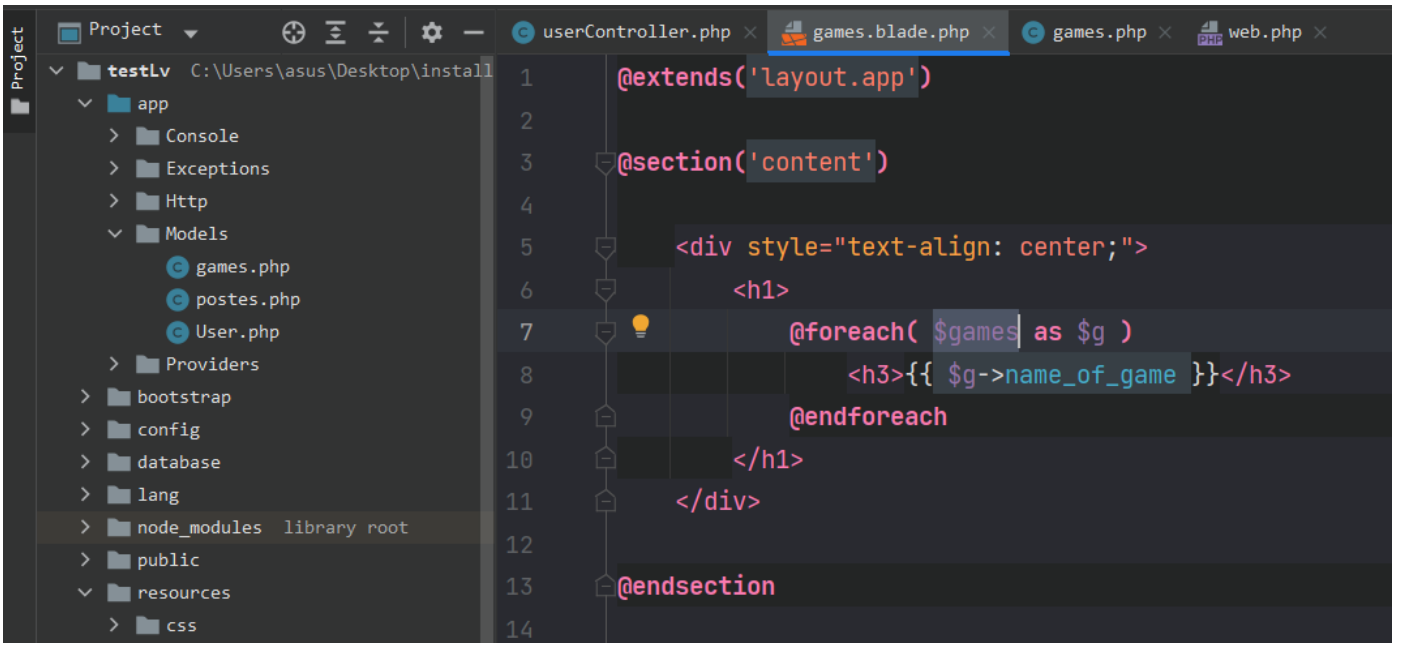


Eloquent : (fetch data from the data base)


Project
testLv C:\Users\asus\Desktop\install
app
Console
Exceptions
Http
Models
games.php
postes.php
User.php
Providers
bootstrap
config
database
lang
node_modules library root
public
postes.php
userController.php
games.blade.php
games.php
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class games extends Model
{
use HasFactory;
}


resources
css
js
views
layout
partial
navbar.blade.php
about.blade.php
contact.blade.php
games.blade.php
home.blade.php
routes
api.php
channels.php
console.php
web.php
| contains the "web" middleware group. Now create something great!
|
*/
Route::get(uri: '/home' , [userController::class , 'home'])->name(name: 'Home') ;
Route::get(uri: '/about' , [userController::class , 'about'])->name(name: 'about') ;
Route::get(uri: '/contact' , [userController::class , 'contact'])->name(name: 'contact') ;
Route::get(uri: '/games' , [userController::class , 'index'])->name(name: 'games') ;





The condition of fetching datas from database :

How to fetch a data from database by id :

```
Route::get( uri: '/games/{id}' , [userController::class , 'show'] )->name( name: 'postShow') ;
```


```
public function show($id)
{
    $games = games::find($id) ;

    return view( view: 'display',[
        'games' => $games
    ]);
```

```blade
@extends('layout.app')

@section('content')

    <center>
        <h1>{{ $games->name_of_game }}</h1>
    </center>

@endsection
```

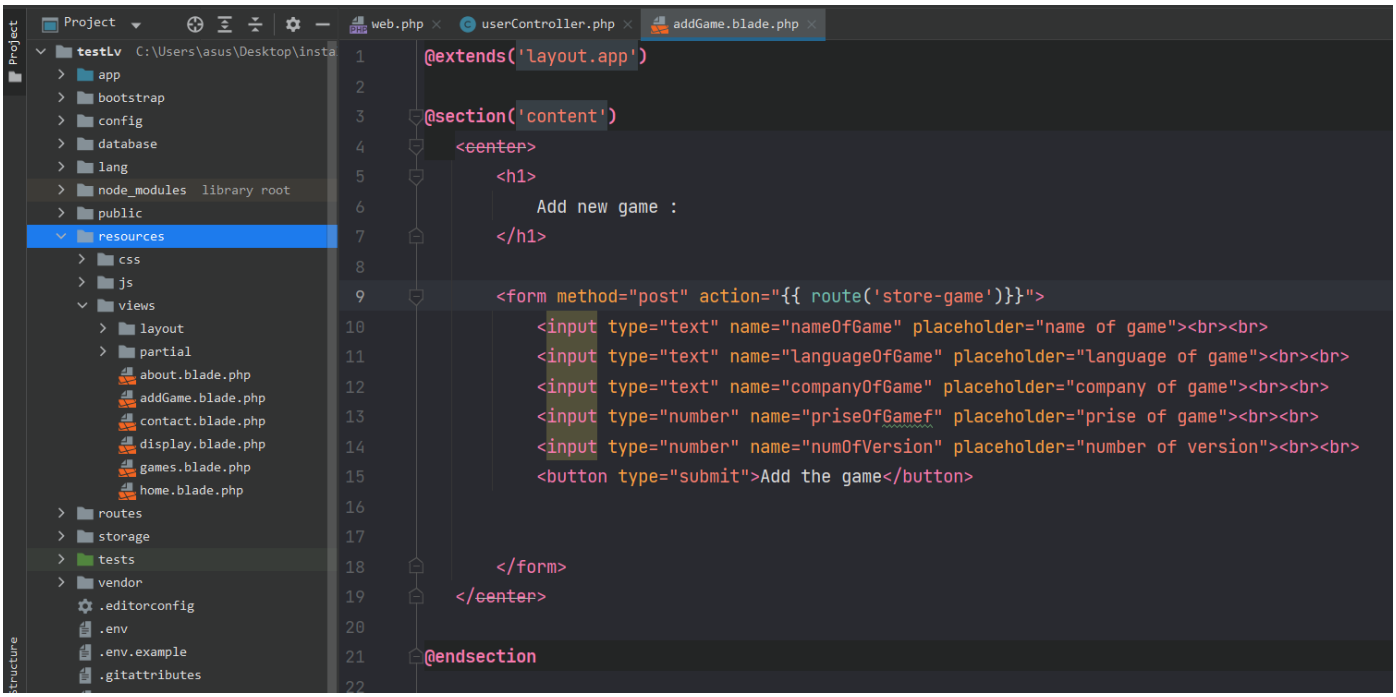
How to fetch data by other colomn no an id :

```php
        ]) ;
    }

    public function show($id)
    {
        $games = games::where('name_of_game' , '=' , 'GTA')->first() ;
        dd($games) ;

        return view( view: 'display',[
            'games' => $games
        ]);

    }
```

How to add something in database :

```php
Route::get( uri: '/games/addGame' , [userController::class , 'add'])->name( name: 'add-game') ;
```

```php
Route::get( uri: '/games/addGame' , [userController::class , 'add'])->name( name: 'add-game') ;
Route::post( uri: '/games/addGame' , [userController::class , 'store'])->name( name: 'store-game') ;
```


Project
testLv C:\Users\asus\Desktop\insta
app
bootstrap
config
database
lang
node_modules library root
public
resources
css
js
views
layout
partial
about.blade.php
addGame.blade.php
contact.blade.php
display.blade.php
games.blade.php
home.blade.php
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
web.php
userController.php
addGame.blade.php
@extends('layout.app')
@section('content')

<h1>
Add new game :
</h1>
<form method="post" action="{{ route('store-game')}}">
<input type="text" name="nameOfGame" placeholder="name of game">

<input type="text" name="languageOfGame" placeholder="language of game">

<input type="text" name="companyOfGame" placeholder="company of game">

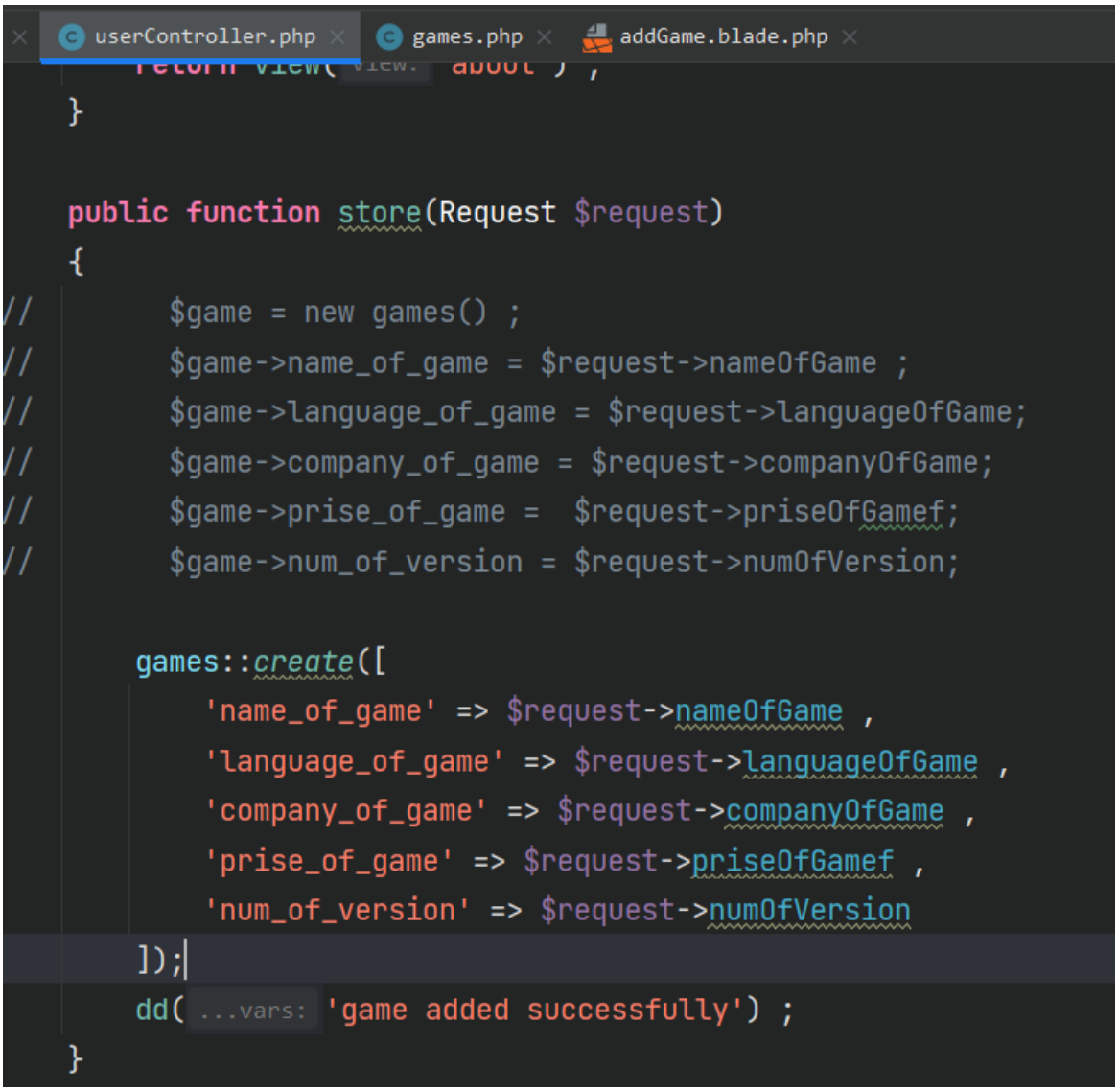
<input type="number" name="priseOfGamef" placeholder="prise of game">

<input type="number" name="numOfVersion" placeholder="number of version">

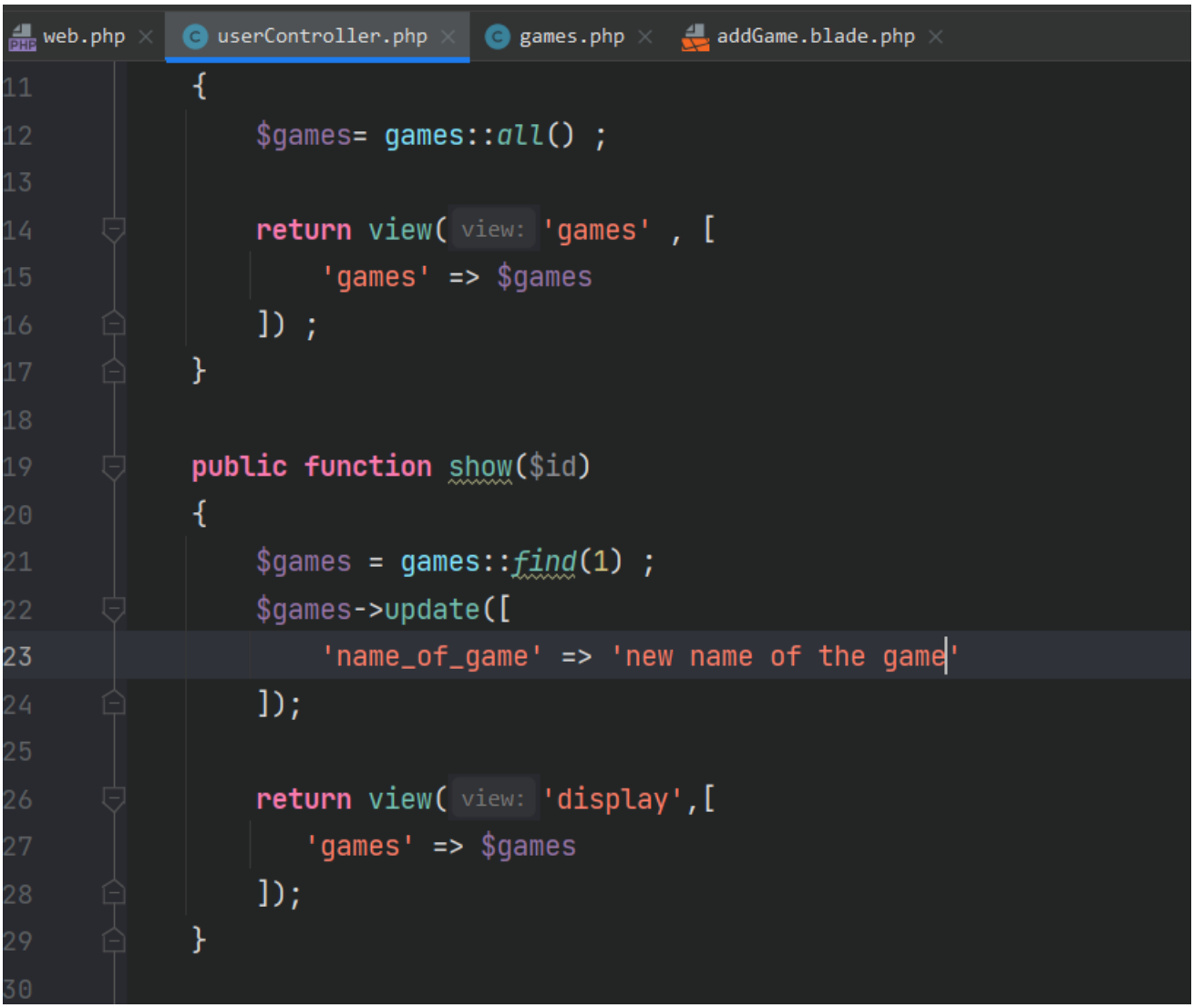
<button type="submit">Add the game</button>
</form>

@endsection


web.php
userController.php
games.php
addGame.blade.php
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class games extends Model
{
use HasFactory;
protected $fillable = ['name_of_game' , 'language_of_game' , 'company_of_game' , 'prise_of_game' , 'num_of_version '] ;
}



Update somthing in database :



Data base and Laravel

samedi 5 mars 2022
23:08

Relation one to many :

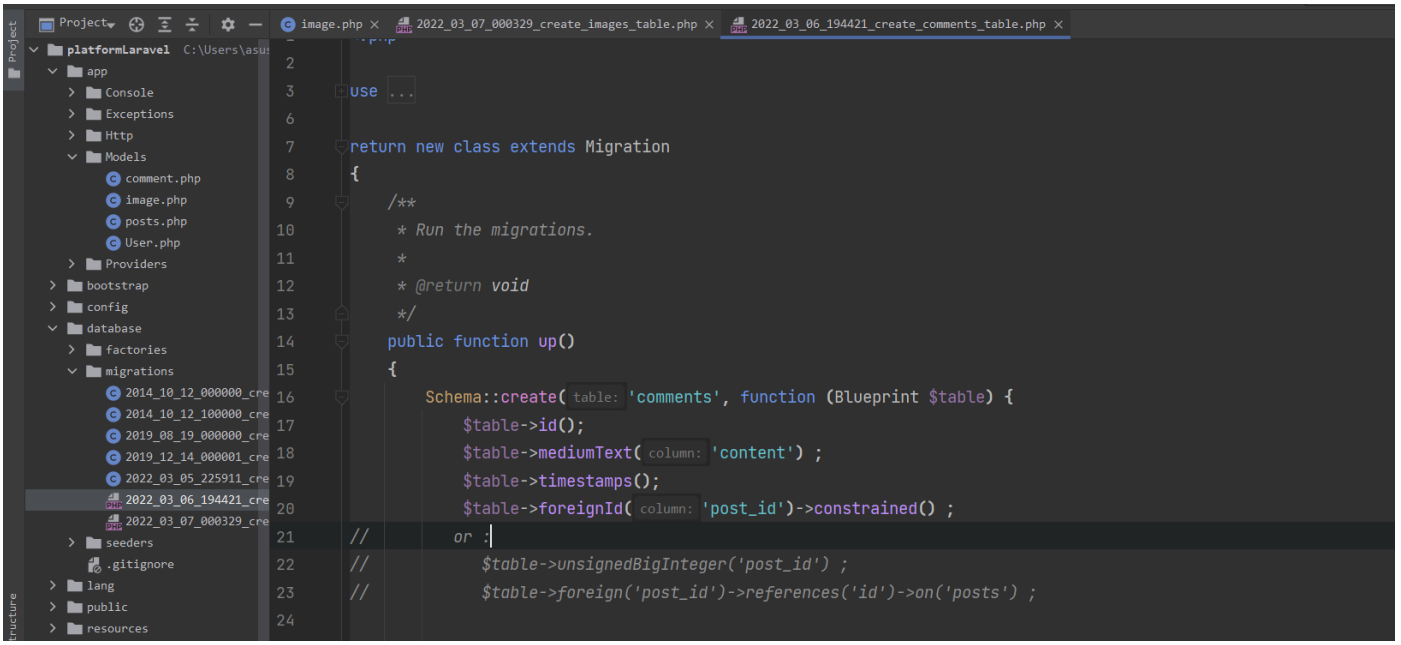
Posts :

Id	Title	Conten	Created_at	Updated_at

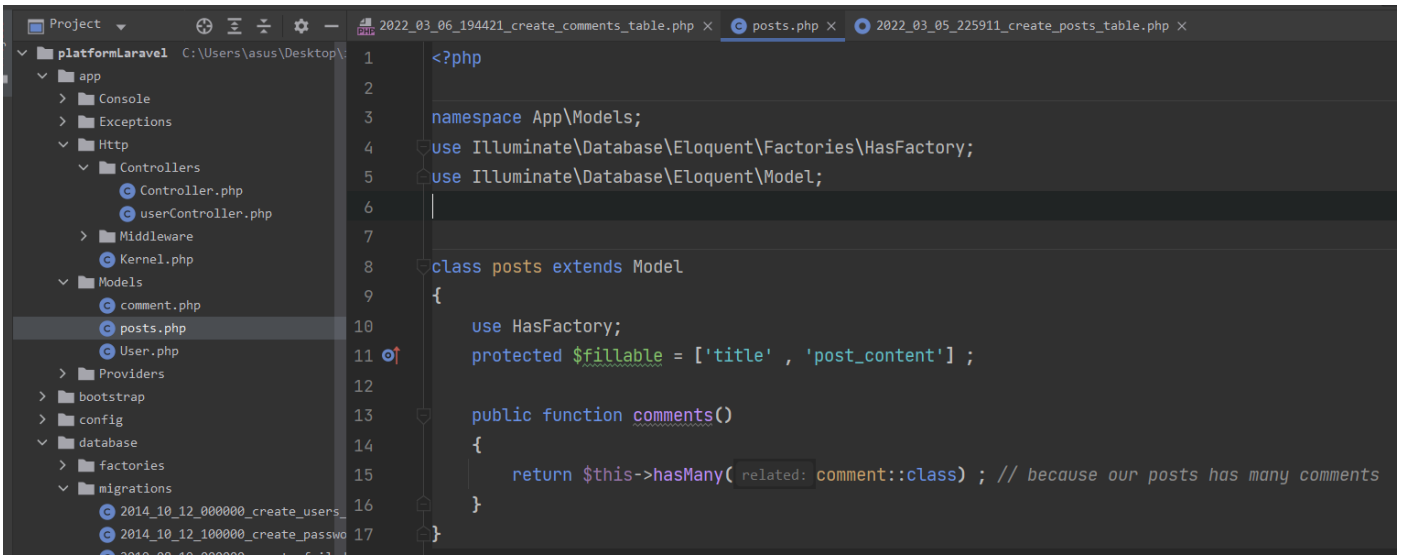
If we want add a comments , we should add ather table (comment) because il we mae it as colomn in (postes table) we will have a lot of ligne with the same values exept the comment value , that s why we gonna make in a new table

Comments :

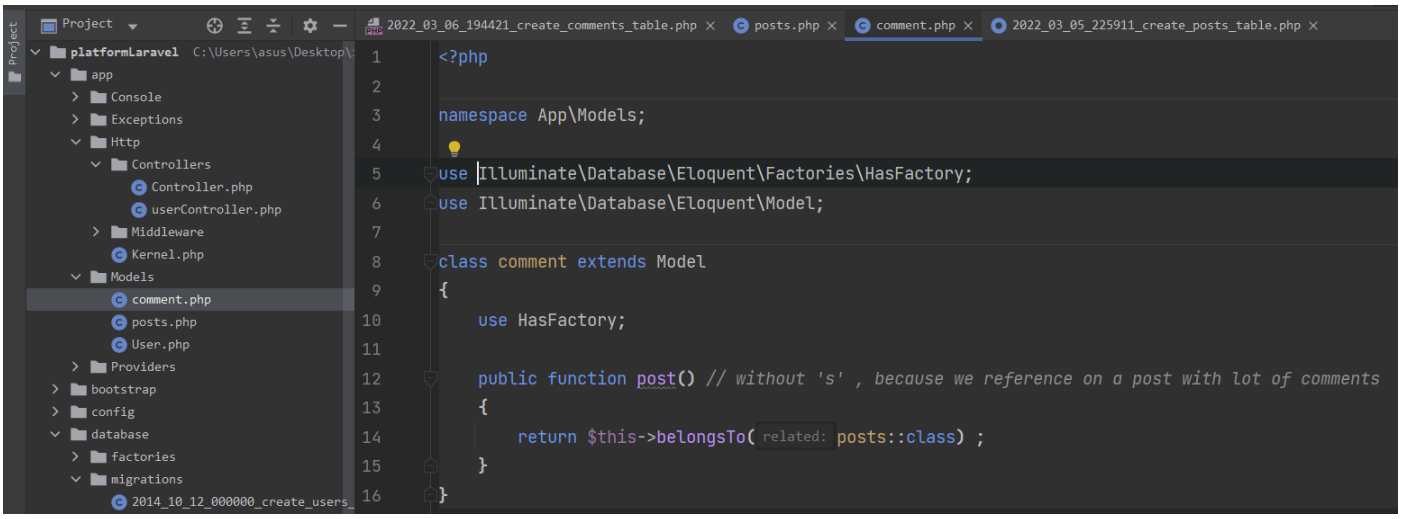
Id	Content	Post_id	Created_at	Updated_at



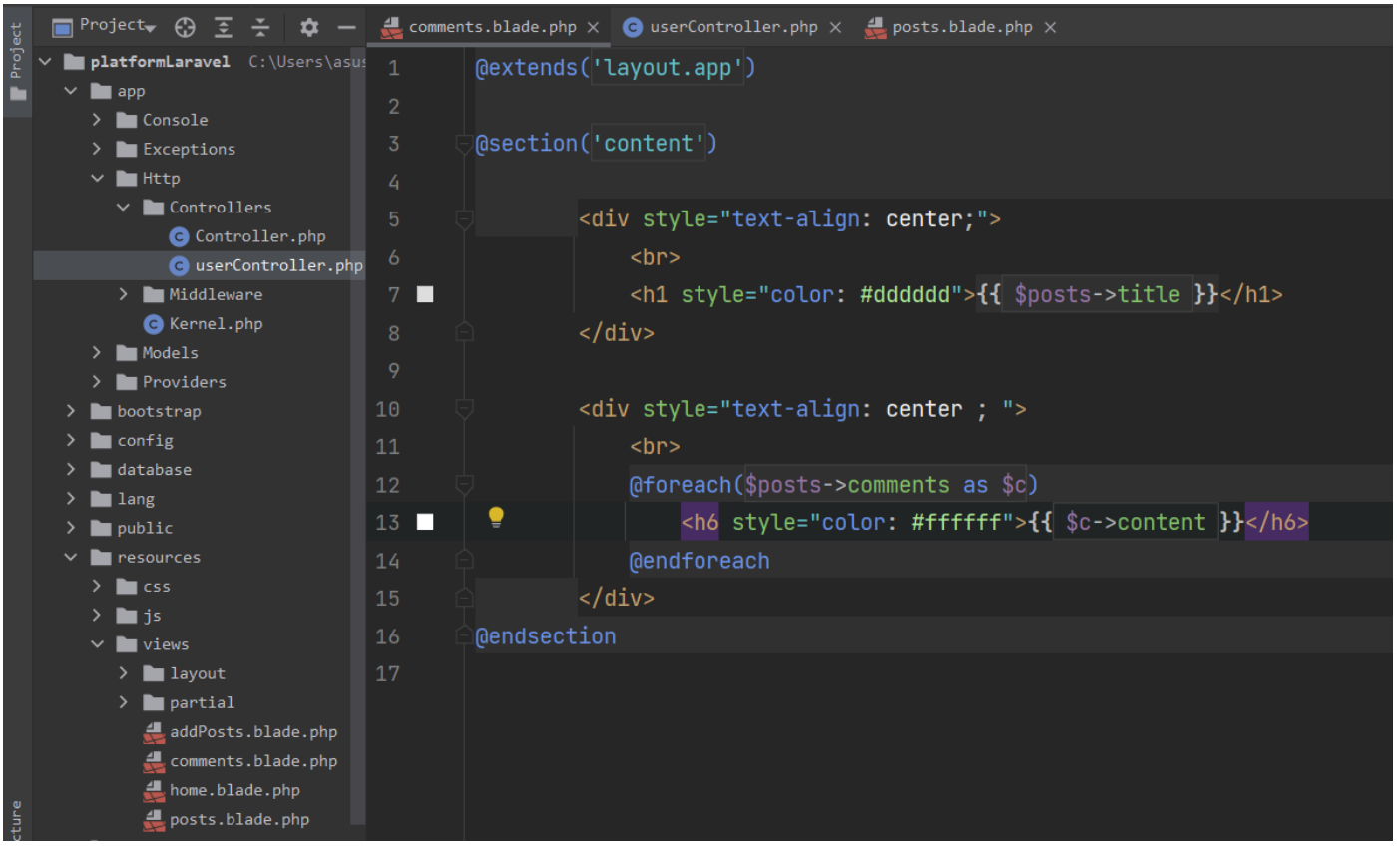
How to define a forieng key in database by the two lines above .



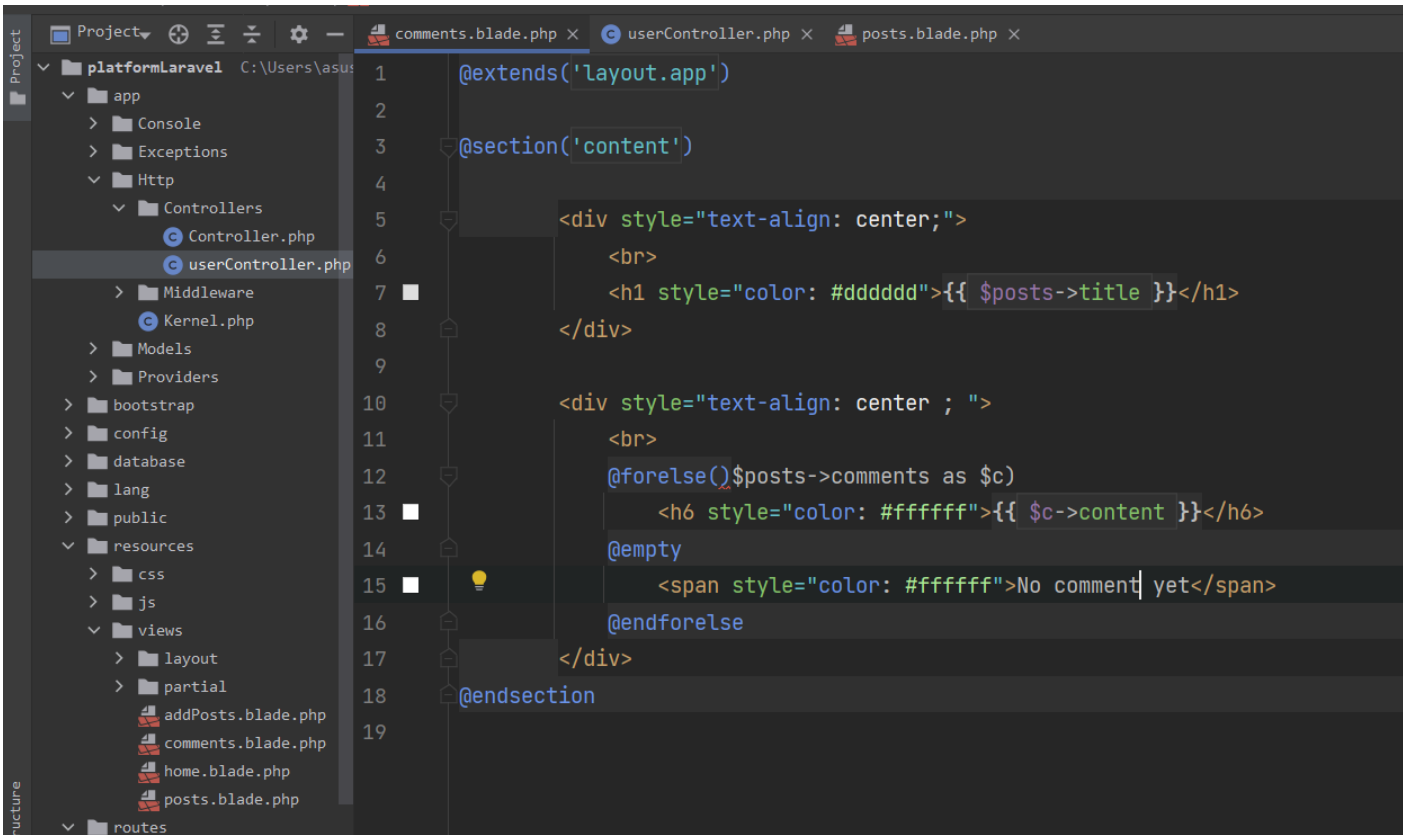
Because our posts has so many comment , at least one comment or 0 .



How to display data :



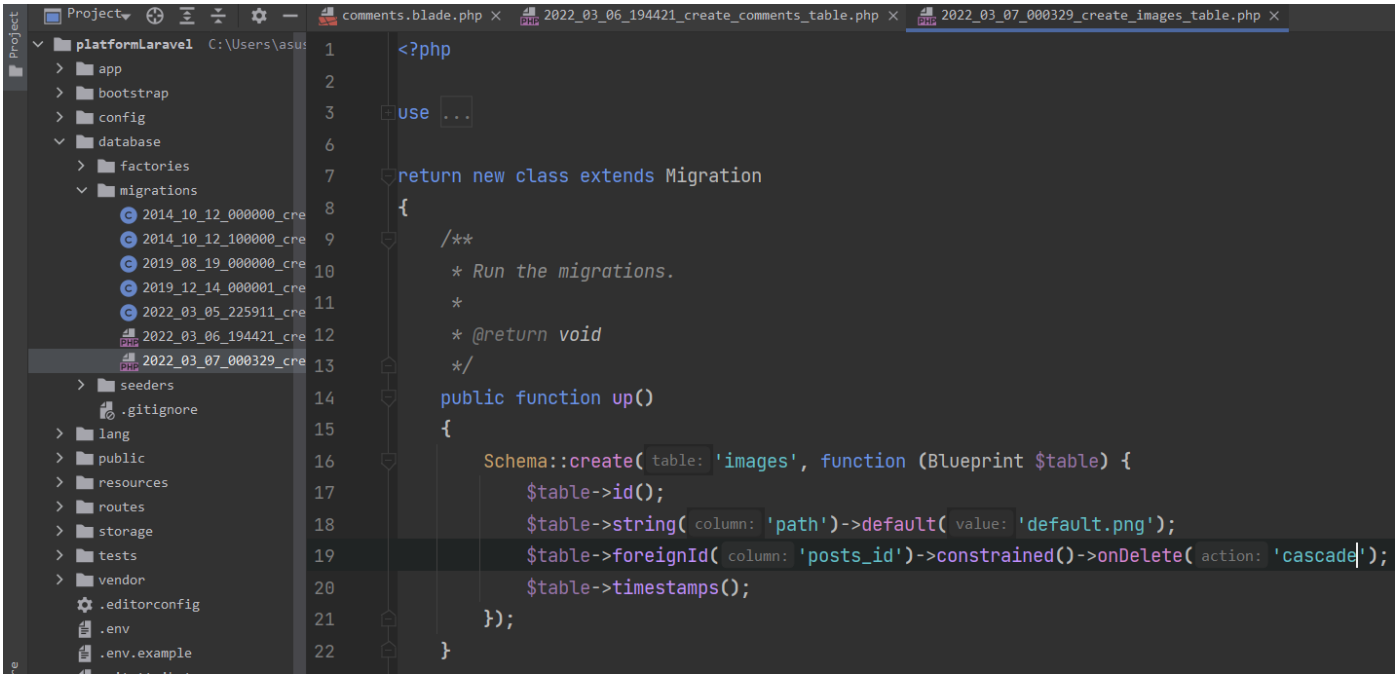
Or :



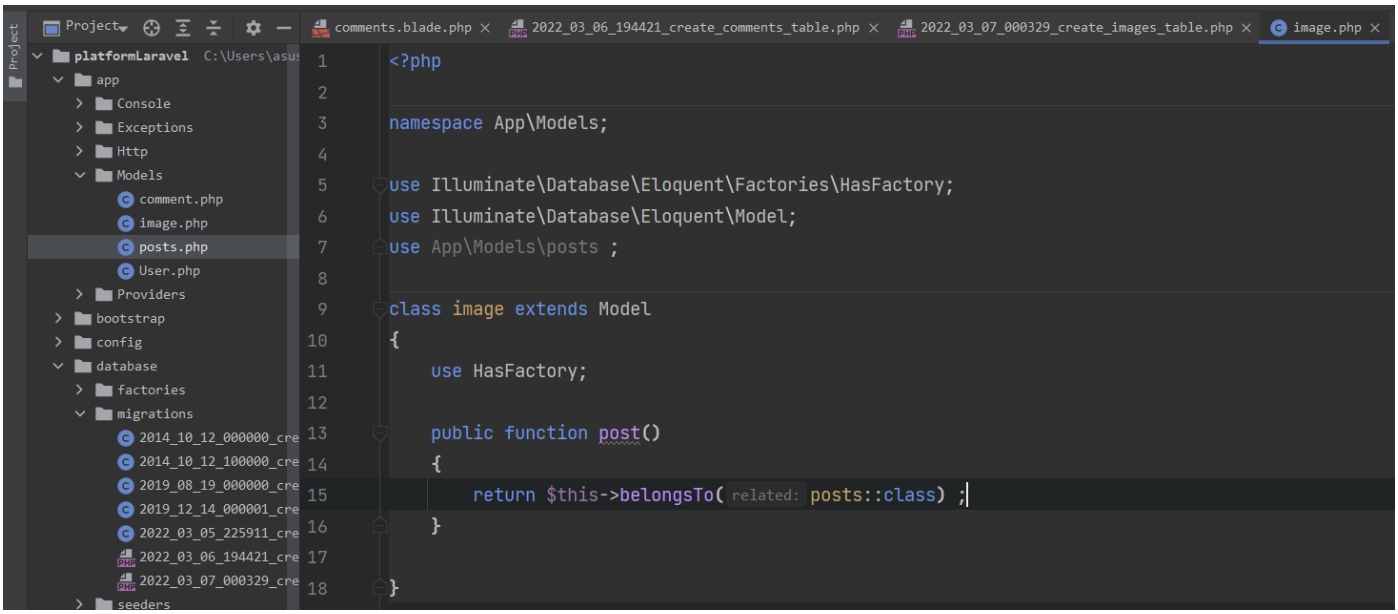
Relation one to one:

Every post has one image and only one , so we need to crate an other table image :

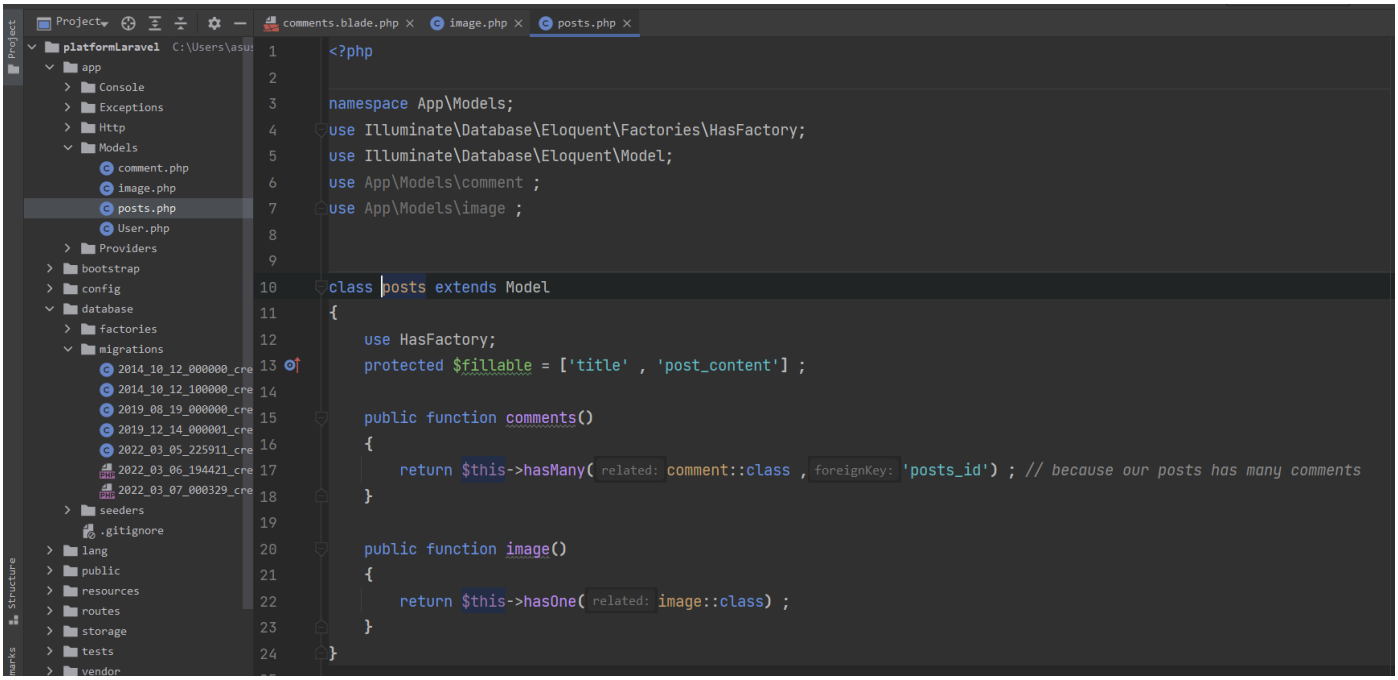
Id	Path	Post_id	Created_at	Updated_at



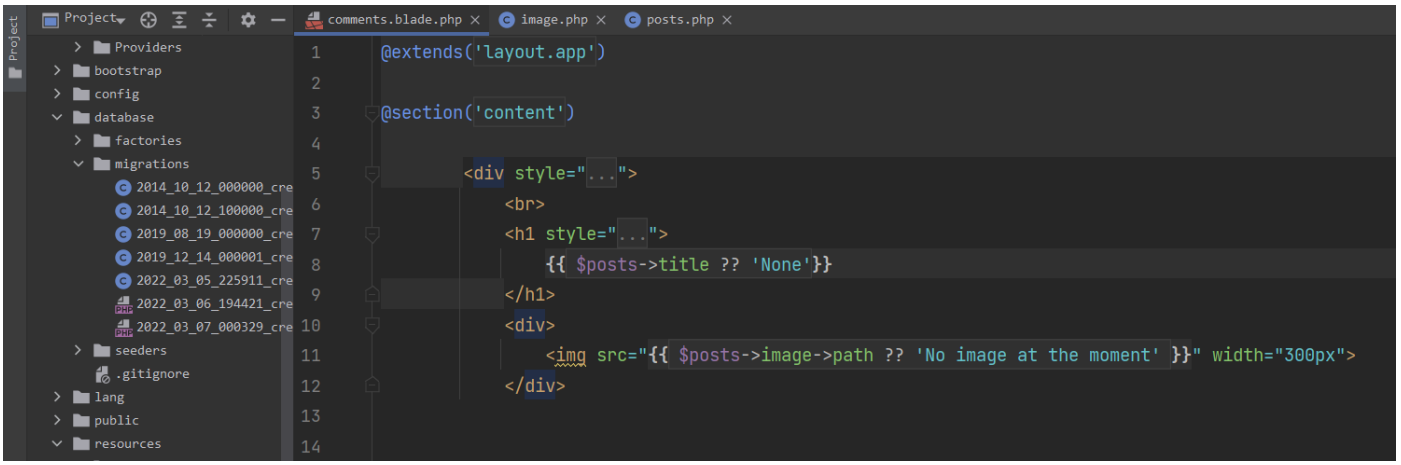
We first create out table and we add the 'foreing key for the' field (See above) .



```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use App\Models\posts ;

class image extends Model
{
    use HasFactory;

    public function post()
    {
        return $this->belongsTo( related: posts::class) ;
    }

}
```



```php
<?php

namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use App\Models\comment ;
use App\Models\image ;


class posts extends Model
{
    use HasFactory;
    protected $fillable = ['title' , 'post_content'] ;

    public function comments()
    {
        return $this->hasMany( related: comment::class , foreignKey: 'posts_id') ; // because our posts has many comments
    }

    public function image()
    {
        return $this->hasOne( related: image::class) ;
    }
}
```



Relation many to many :

A tags can be present a one post or lot of ones , but in the same time a post can have a lots of tags and it s the meaning of many to many relation .

We define a tag table as :

Id	Name	Created_at	Updated_at

We don't need here to define a foreing key , we need just to define a new table (intermediate table) this table wil containe bouth ids of 'posts' and 'tags' .

We will define it as : post_tag

Post_id	Tag_id	Created_at	Updated_at

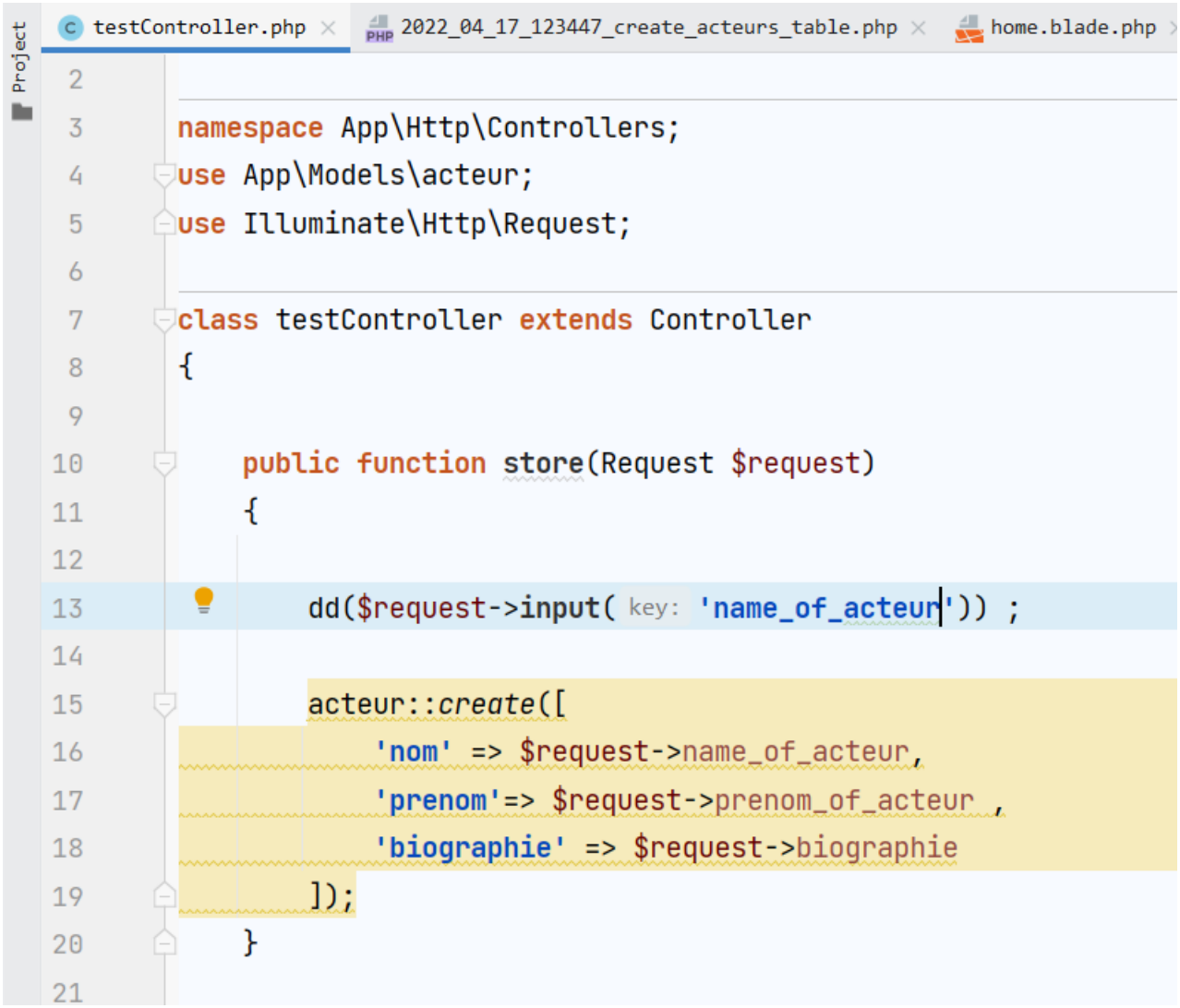
And we will have an exaple like this :

Post_id	Tag_id
1	3
1	2
2	5
2	2
3	5

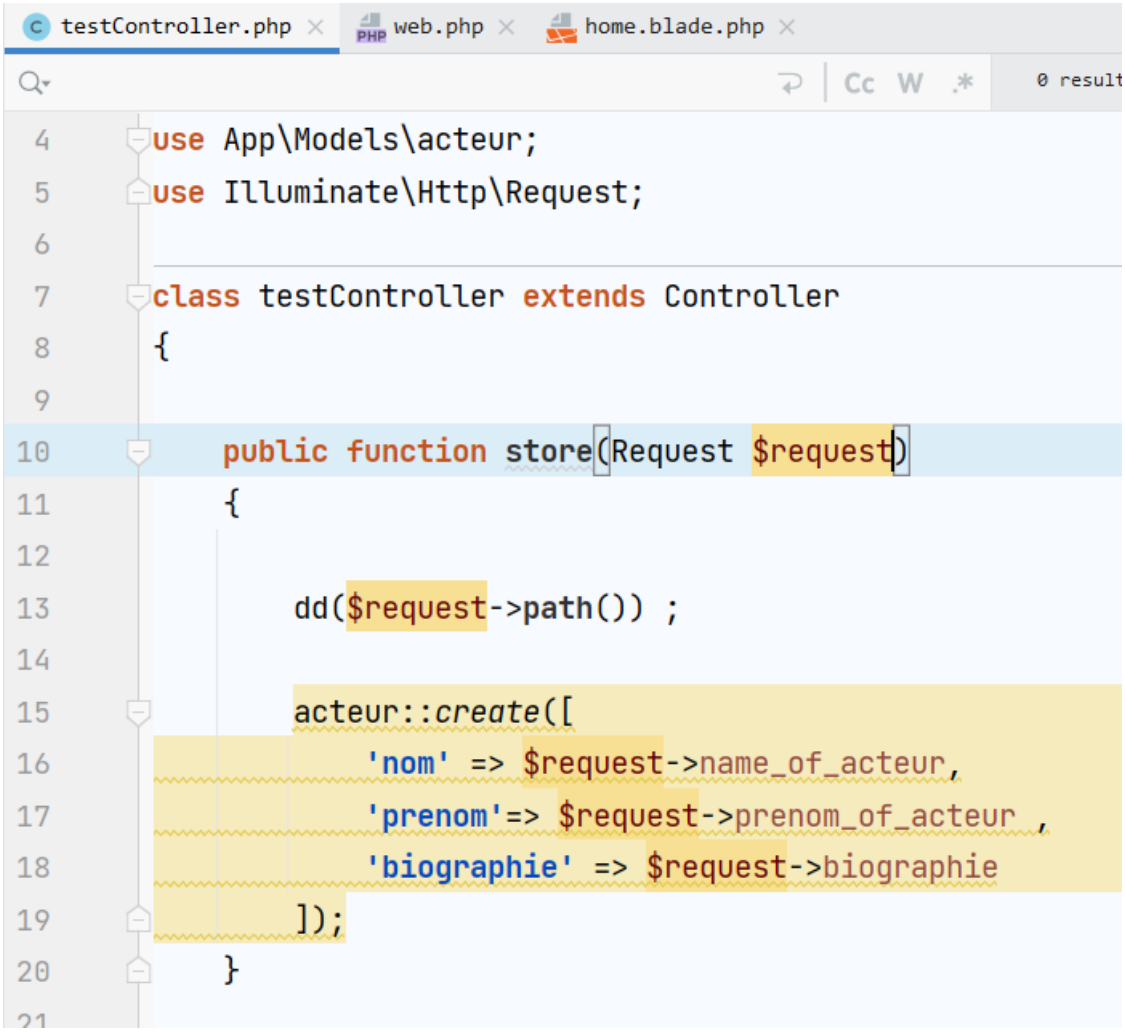
Request

dimanche 17 avril 2022
11:31

La fonction 'input' dans l'objet 'request' : (allow to fetch the value from the key)



Return the path with the request object :



Verfy if we are in specifique URL :

```php
class testController extends Controller
{

    public function store(Request $request)
    {
        dd($request->is( ...patterns: '/')) ;

        acteur::create([
            'nom' => $request->name_of_acteur,
            'prenom'=> $request->prenom_of_acteur ,
            'biographie' => $request->biographie
        ]);
    }
```

Verify if we are in a specific route :

```php
Route::get( uri: '/' , [testController::class,'home'])->name( name: 'home') ;
Route::post( uri: '/' , [testController::class , 'store'])->name( name: 'store_acteur') ;
```

```php
    public function store(Request $request)
    {
        dd($request->routeIs( ...patterns: 'store_acteur')) ;

        acteur::create([
            'nom' => $request->name_of_acteur,
            'prenom'=> $request->prenom_of_acteur ,
            'biographie' => $request->biographie
        ]);
    }
```

How to get the url :

```php
public function store(Request $request)
{
    dd($request->url()) ;

    acteur::create([
        'nom' => $request->name_of_acteur,
        'prenom'=> $request->prenom_of_acteur ,
        'biographie' => $request->biographie
    ]);
}
```

How to know if the method is post ou get :

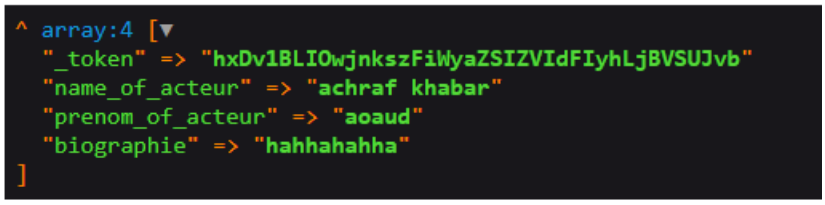
```php
    dd($request->isMethod( method: 'post')) ;

    acteur::create([
        'nom' => $request->name_of_acteur,
        'prenom'=> $request->prenom_of_acteur ,
        'biographie' => $request->biographie
    ]);
}
```

And how to return the method used :

```php
{
    dd($request->method()) ;

    acteur::create([
        'nom' => $request->name_of_acteur,
        'prenom'=> $request->prenom_of_acteur ,
        'biographie' => $request->biographie
    ]);
}
```

How to know the possible input :

```
        dd($request->all()) ;

        acteur::create([
            'nom' => $request->name_of_acteur,
            'prenom'=> $request->prenom_of_acteur ,
            'biographie' => $request->biographie
        ]);
    }
```



! : for the _token is a key generated randomly due to the @csrf .

Validation of formulaire :

jeudi 21 avril 2022
22:10

Inserting null object in a form :

To ovoid this issue we neet a request function named 'validate' :

```php
public function store(Request $request)
{
    $request->validate([
       'title' => 'required' ,
        'post_content' => 'required'
    ]);

    posts::create ([
        'title' => $request->title ,
        'post_content' => $request->post_content
    ]);

}
```

How to display the error message when the form submition failed :

addPosts.blade.php | userController.php

```blade
@extends('layout.app')

@section('content')

    <section data-aos="fade-up-right" class="contact-clean">
        @if( $errors->any())
            @foreach($errors->all() as $E)
                <center>
                    <span style="color: red">{{ $E }}</span> <br><br>
                </center>
            @endforeach
        @endif
        <form method="post" action="{{route('store-posts')}}"><h2 class="text-center">Add post</h2>
            @csrf
            <div class="mb-3">
                <input class="form-control" type="text" name="title" placeholder="title">
            </div>
            <div class="mb-3"></div>
            <div class="mb-3"><textarea class="form-control" name="post_content" placeholder="content" rows="14">
                </textarea>
            </div>
            <div class="mb-3">
                <button class="btn btn-primary" type="submit">send</button>
            </div>
        </form>
    </section>
@endsection
```

If we need that the input field needs at least 8 characters :

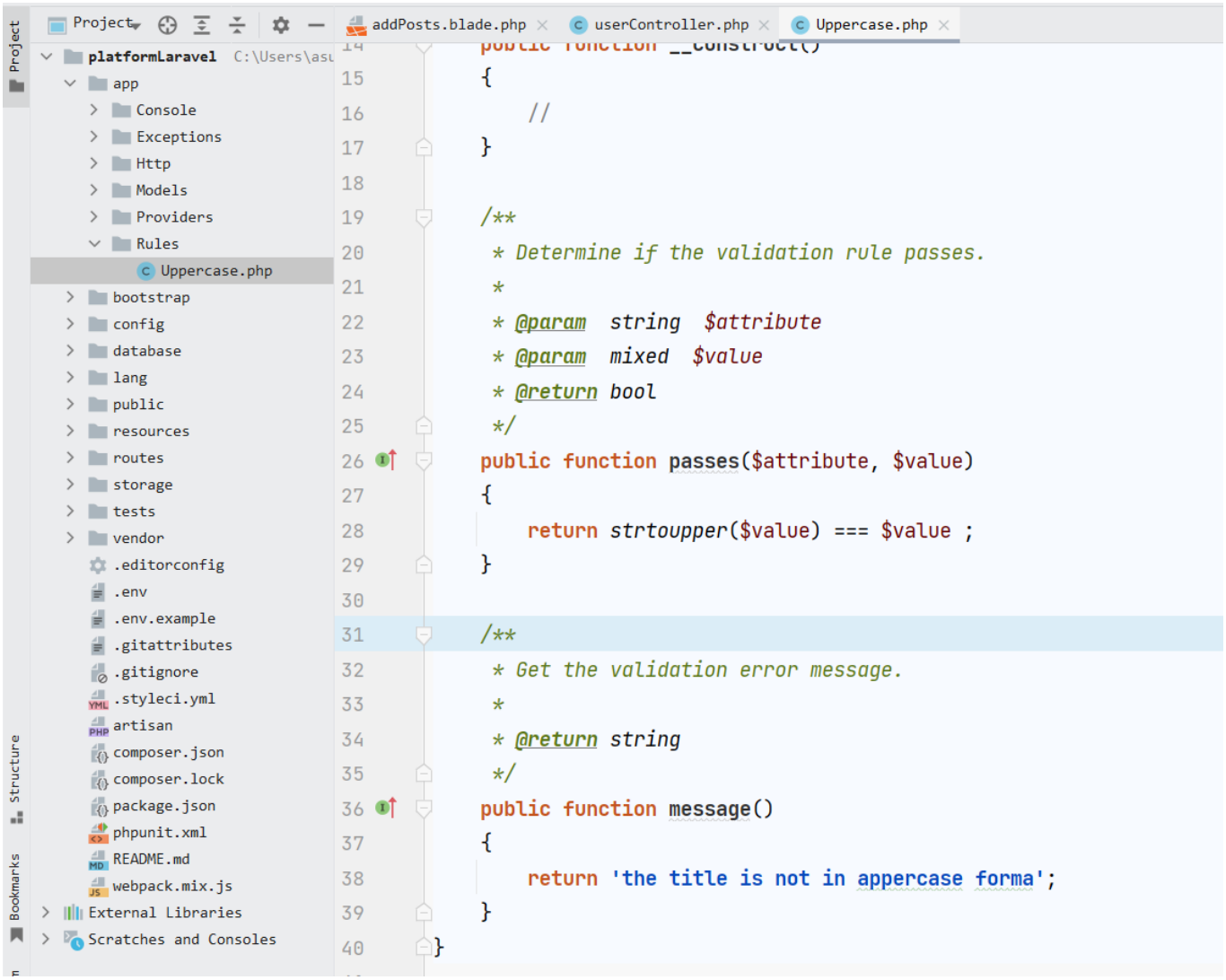
```
}

public function store(Request $request)
{
    $request->validate([
       'title' => 'required|min:8' ,
        'post_content' => 'required'
    ]);

    posts::create ([
        'title' => $request->title ,
        'post_content' => $request->post_content
    ]);

}
```

Other rules :

```
public function store(Request $request)
{
    $request->validate([
       'title' => 'required|min:8|max:255|unique:posts',
        'post_content' => 'required'
    ]);

    posts::create ([
        'title' => $request->title ,
        'post_content' => $request->post_content
    ]);
```

How to define a new rule :

```
PS C:\Users\asus\Desktop\install\htdocs\platformLaravel> php artisan make:rule Uppercase
Rule created successfully.
```




```php
public function store(Request $request)
{
    $request->validate([
       'title' => ['required' , 'max:255' , 'min:8' , 'unique:posts' , new Uppercase],
        'post_content' => 'required'
    ]);

    posts::create ([
        'title' => $request->title ,
        'post_content' => $request->post_content
    ]);
```

Files

vendredi 22 avril 2022
23:09

Forme update :

```
@endif
<form method="post" action="{{route('store-posts')}}" enctype="multipart/form-data"><h2 class="text-center
    @csrf
    <div class="mb-3">
        <input class="form-control" type="text" name="title" placeholder="title">
    </div>
    <div class="mb-3"></div>
    <div class="mb-3"><textarea class="form-control" name="post_content" placeholder="content" rows="14">
        </textarea>
    </div>
    <div class="mb-3">
        <input class="form-control" type="file" name="title" placeholder="title">
    </div>
    <div class="mb-3">
        <button class="btn btn-primary" type="submit">send</button>
    </div>
```

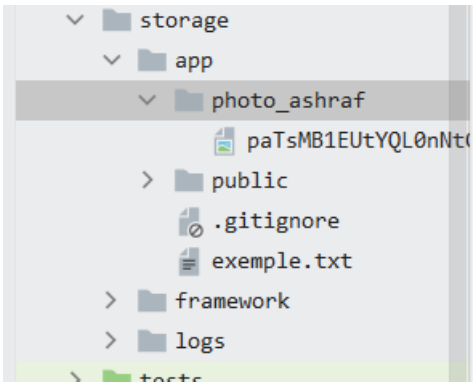
Storage in the local disk :

```
public function store(Request $request)
{
    storage::disk( name: 'local')->put( path: 'exemple.txt' , contents: 'My content') ;
    die();

    $request->validate([
       'title' => ['required' , 'max:255' , 'min:8' , 'unique:posts' , new Uppercase],
        'post_content' => 'required'
    ]);

    posts::create ([
        'title' => $request->title ,
        'post_content' => $request->post_content
    ]);
```

How to fetch a file from form uploeaded from the pc :

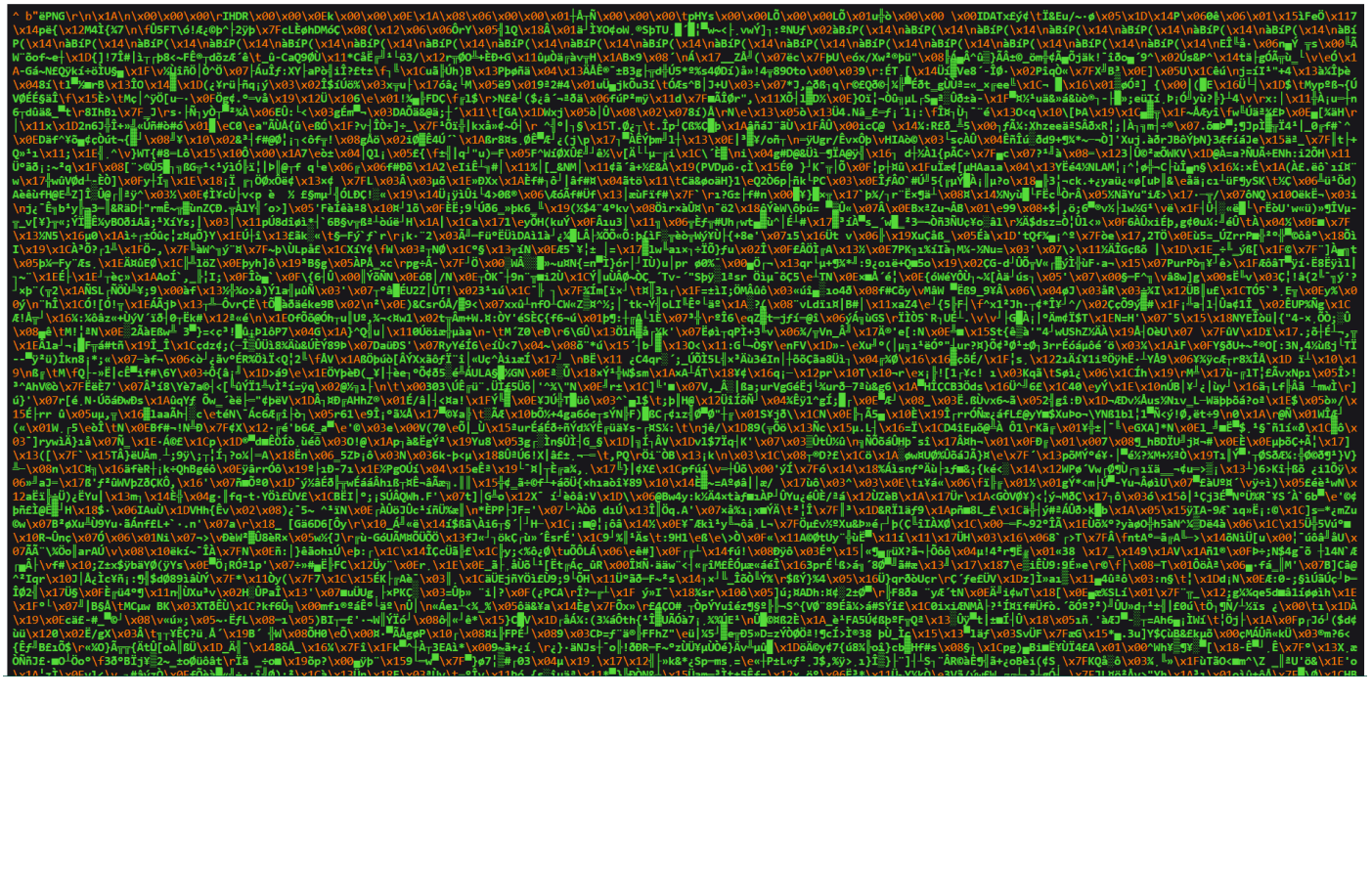
```
@endif
<form method="post" action="{{route('store-posts')}}" enctype="multipart/form-data"><h2 class="text-center">Add post</h2>
    @csrf
    <div class="mb-3">
        <input class="form-control" type="text" name="title" placeholder="title">
    </div>
    <div class="mb-3"></div>
    <div class="mb-3"><textarea class="form-control" name="post_content" placeholder="content" rows="14">
        </textarea>
    </div>
    <div class="mb-3">
        <input class="form-control" type="file" name="photo_ashraf" placeholder="title">
    </div>
    <div class="mb-3">
        <button class="btn btn-primary" type="submit">send</button>
    </div>
```

```
public function store(Request $request)
{
    storage::disk( name: 'local')->put( path: 'photo_ashraf' , $request->file( key: 'photo_ashraf')) ;
    die();
    $request->validate([
        'title' => ['required' , 'max:255' , 'min:8' , 'unique:posts' , new Uppercase],
        'post_content' => 'required'
    ]);

    posts::create ([
        'title' => $request->title ,
        'post_content' => $request->post_content
    ]);

}
```



Retrieving Files :

```php
public function store(Request $request)
{
    $name =  storage::disk( name: 'local')->put( path: 'photo_ashraf' , $request->file( key: 'photo_ashraf')) ;
    dd(storage::get($name)) ;
    $request->validate([
        'title' => ['required' , 'max:255' , 'min:8' , 'unique:posts' , new Uppercase],
        'post_content' => 'required'
    ]);

    posts::create ([
        'title' => $request->title ,
        'post_content' => $request->post_content
    ]);

}
```



Verify if a file exists or not :

```
public function store(Request $request)
{
    $name =  storage::disk( name: 'local')->put( path: 'photo_ashraf' , $request->file( key: 'photo_ashraf')) ;
    dd(storage::disk( name: 'local')->exists($name)) ;
    $request->validate([
       'title' => ['required' , 'max:255' , 'min:8' , 'unique:posts' , new Uppercase],
        'post_content' => 'required'
    ]);

    posts::create ([
        'title' => $request->title ,
        'post_content' => $request->post_content
    ]);

}
```

```
public function store(Request $request)
{
    $name =  storage::disk( name: 'local')->put( path: 'photo_ashraf' , $request->file( key: 'photo_ashraf')) ;
    dd(storage::disk( name: 'local')->missing($name)) ;
    $request->validate([
       'title' => ['required' , 'max:255' , 'min:8' , 'unique:posts' , new Uppercase],
        'post_content' => 'required'
    ]);

    posts::create ([
        'title' => $request->title ,
        'post_content' => $request->post_content
    ]);

}
```

How to give the user the right to dowload a file :

```php
public function store(Request $request)
{
    $name =  storage::disk( name: 'local')->put( path: 'photo_ashraf' , $request->file( key: 'photo_ashraf')) ;
    return storage::download($name) ;

    $request->validate([
       'title' => ['required' , 'max:255' , 'min:8' , 'unique:posts' , new Uppercase],
        'post_content' => 'required'
    ]);

    posts::create ([
        'title' => $request->title ,
        'post_content' => $request->post_content
    ]);
```

Renvoyer la taille du fichier :

```php
public function store(Request $request)
{
    $name =  storage::disk( name: 'local')->put( path: 'photo_ashraf' , $request->file( key: 'photo_ashraf')) ;
    dd(storage::size($name)) ;

    $request->validate([
       'title' => ['required' , 'max:255' , 'min:8' , 'unique:posts' , new Uppercase],
        'post_content' => 'required'
    ]);
```

How to store a file in the disk :

```php
public function store(Request $request)
{
//      $name =  storage::disk('local')->put('photo_ashraf' , $request->file('photo_ashraf')) ;
    dd(storage::put( path: 'fichier_ashraf' , $request->photo_ashraf)) ;

    $request->validate([
       'title' => ['required' , 'max:255' , 'min:8' , 'unique:posts' , new Uppercase],
        'post_content' => 'required'
    ]);

    posts::create ([
        'title' => $request->title ,
        'post_content' => $request->post_content
    ]);
```

If we want the file as unique :

```php
$path = $request->file('avatar')->storeAs(
    'avatars', $request->user()->id
);
```

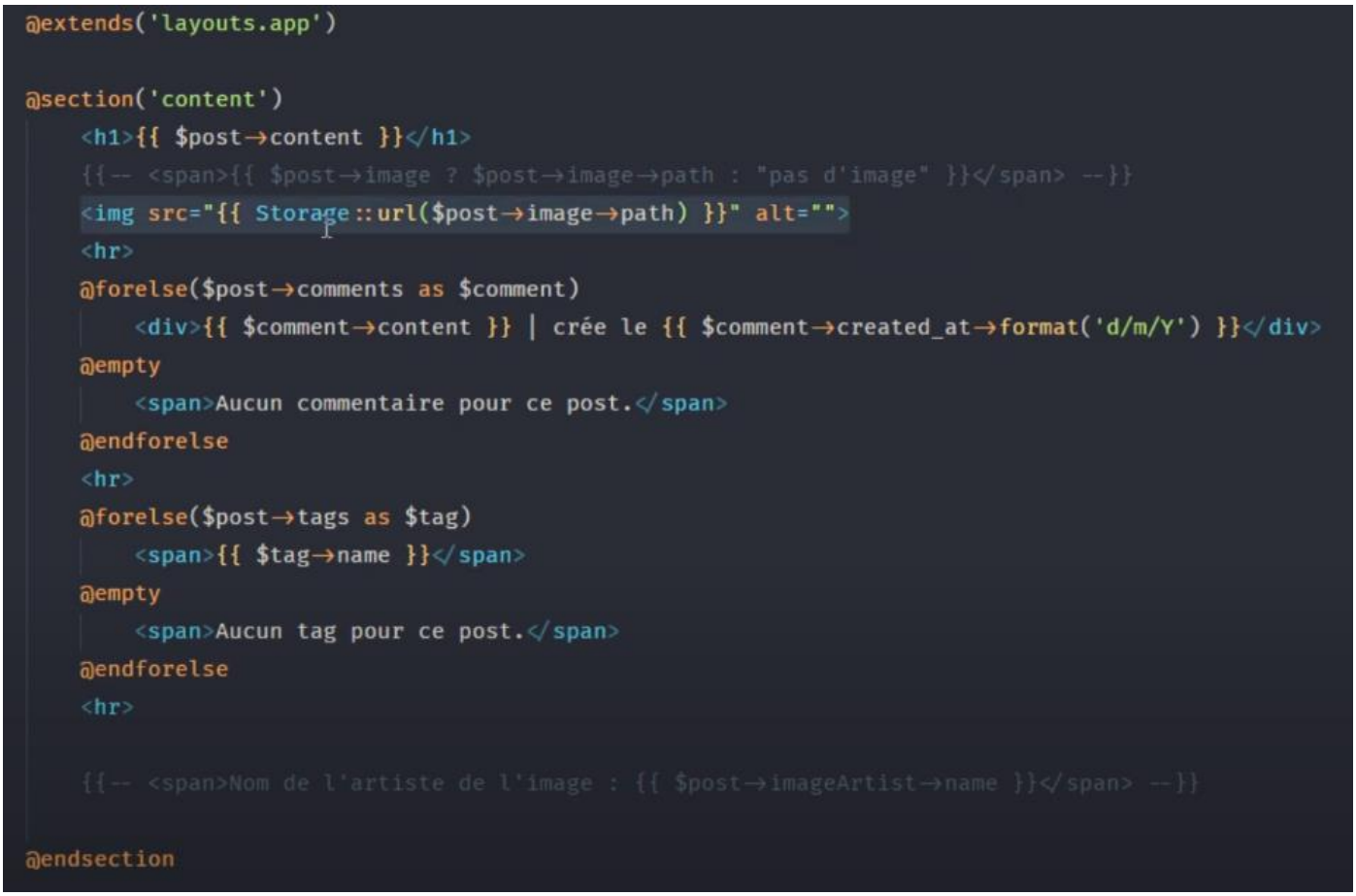
How to generate a randome files :

```php
public function store(Request $request)
{
    $filename = time().'.'.$request->file( key: 'photo_ashraf')->extension() ;

    dd($filename) ;
    $name =  storage::disk('local')->put('photo_ashraf' , $request->file('photo_ashraf')) ;
    dd(storage::put( path: 'fichier_ashraf' , $request->photo_ashraf)) ;

    $request->validate([
       'title' => ['required' , 'max:255' , 'min:8' , 'unique:posts' , new Uppercase],
        'post_content' => 'required'
    ]);

    posts::create ([
        'title' => $request->title ,
        'post_content' => $request->post_content
```

How to store an image for each post :

```php
public function store(Request $request)
{
      $request->validate([
          'title' => ['required' , 'max:255' , 'min:8' , 'unique:posts' , new Uppercase],
          'post_content' => 'required'
      ]);

    // crerer un directory named photos dans publique avec une photo named randomlly xxxxx.pnj .
    $filename = time().'.'.$request->file( key: 'photo_ashraf')->extension() ;
    $path =  $request->file( key: 'photo_ashraf')->storeAs(
            path: 'photos' ,
            $filename ,
            options: 'public'
    );

    $post = posts::create ([
        'title' => $request->title ,
        'post_content' => $request->post_content
    ]);

    $image = new Image() ;
    $image->path = $path ;

    $post->image()->save($image) ;
```

```
PS C:\Users\asus\Desktop\install\htdocs\platformLaravel> php artisan storage:link
```



```
@extends('layouts.app')

@section('content')
    <h1>{{ $post->content }}</h1>
    {{-- <span>{{ $post->image ? $post->image->path : "pas d'image" }}</span> --}}
    <img src="{{ Storage::url($post->image->path) }}" alt="">
    <hr>
    @forelse($post->comments as $comment)
        <div>{{ $comment->content }} | crée le {{ $comment->created_at->format('d/m/Y') }}</div>
    @empty
        <span>Aucun commentaire pour ce post.</span>
    @endforelse
    <hr>
    @forelse($post->tags as $tag)
        <span>{{ $tag->name }}</span>
    @empty
        <span>Aucun tag pour ce post.</span>
    @endforelse
    <hr>

    {{-- <span>Nom de l'artiste de l'image : {{ $post->imageArtist->name }}</span> --}}

@endsection
```

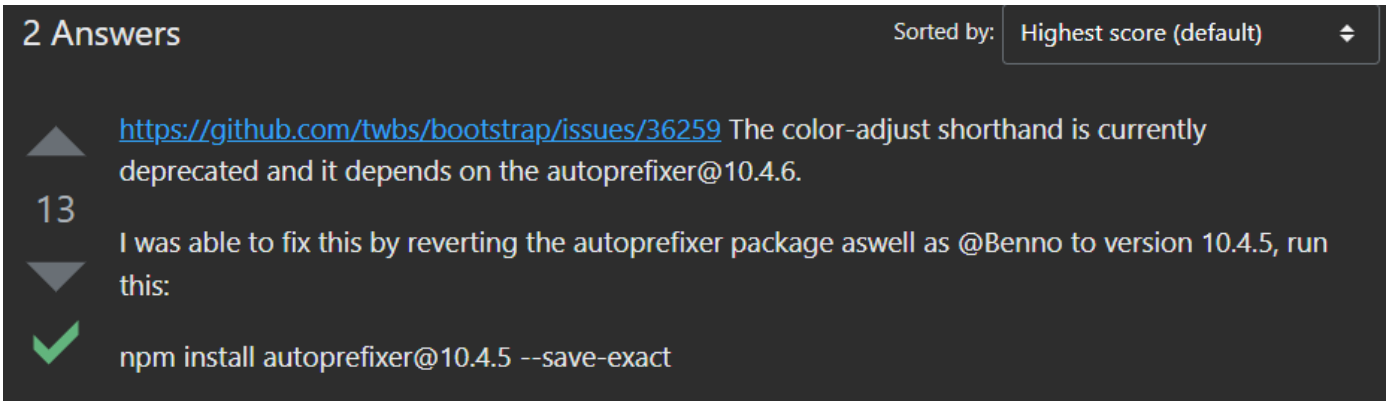
Authentification

samedi 23 avril 2022
12:18

How to fix the npm run dev problem :

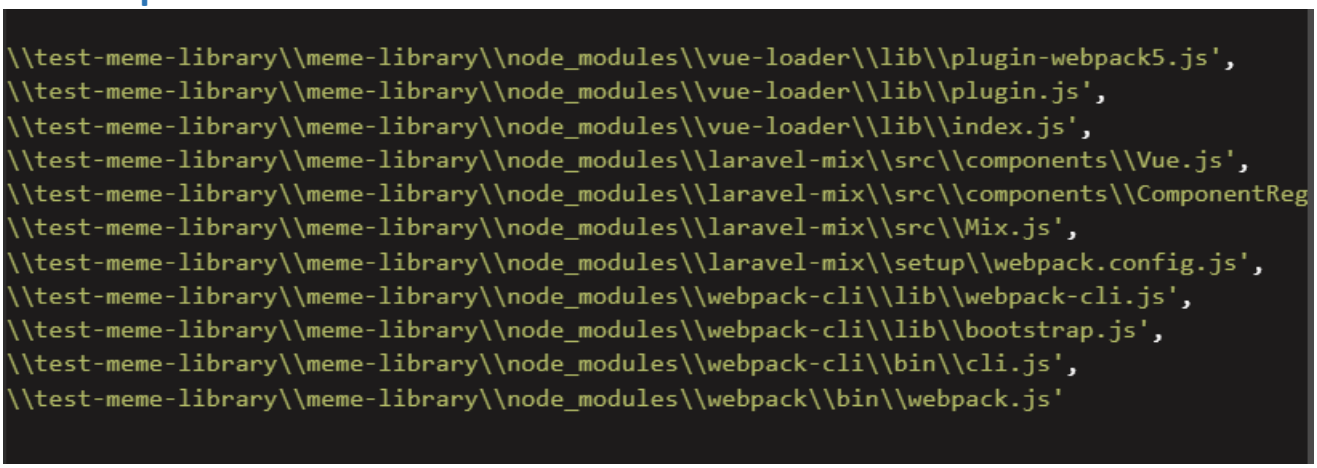
https://github.com/twbs/bootstrap/issues/36259

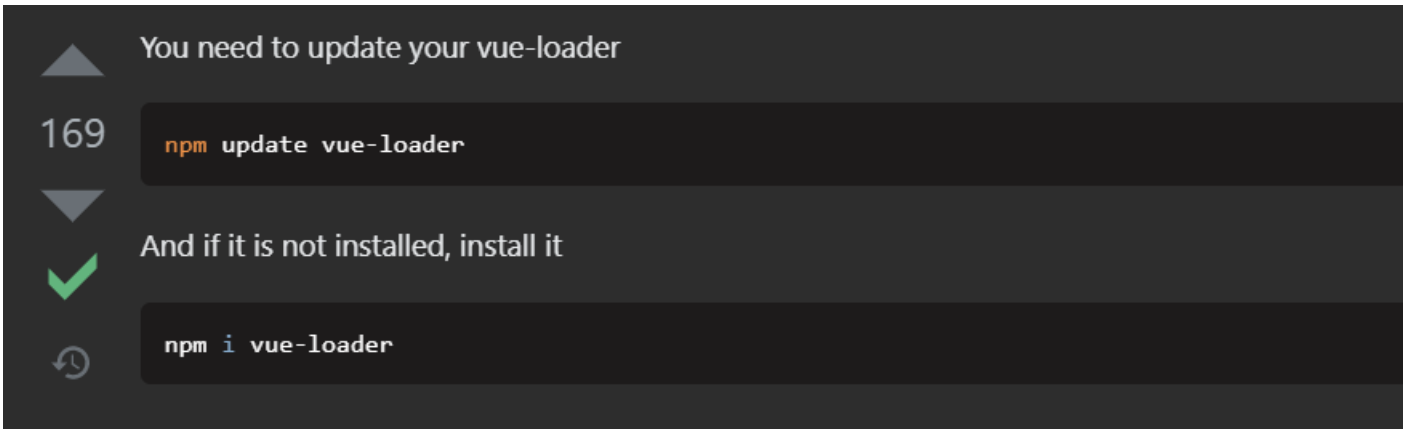
- autoprefixer: Replace color-adjust to print-color-adjust. · Issue #36259 · twbs/bootstrap (github.com)
- laravel - 1 WARNING in child compilations (Use 'stats.children: true' resp. '--stats-children' for more details) - Stack Overflow



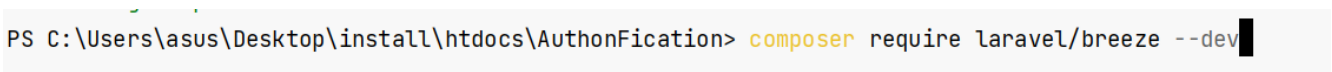
Npm install autoprefixer@10.4.5 --save-exact

Other npm run dev's issues :



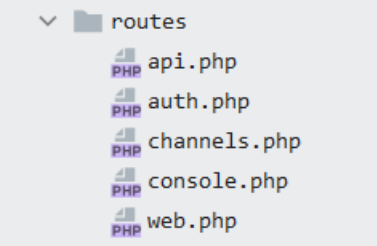


We need to install the laravel breez if we need to have login proprties :



```
Please execute the "npm install && npm run dev" command to build your assets.
PS C:\Users\asus\Desktop\install\htdocs\AuthonFication> php artisan breeze:install
```

We notice that we have a new folder inside route named auth :




```
PS C:\laragon\www\laravel-authentication> npm install
```

```
PS C:\Users\asus\Desktop\install\htdocs\AuthonFication> npm run dev
```

The login system work perfectlly .

We need now to make a controller test :

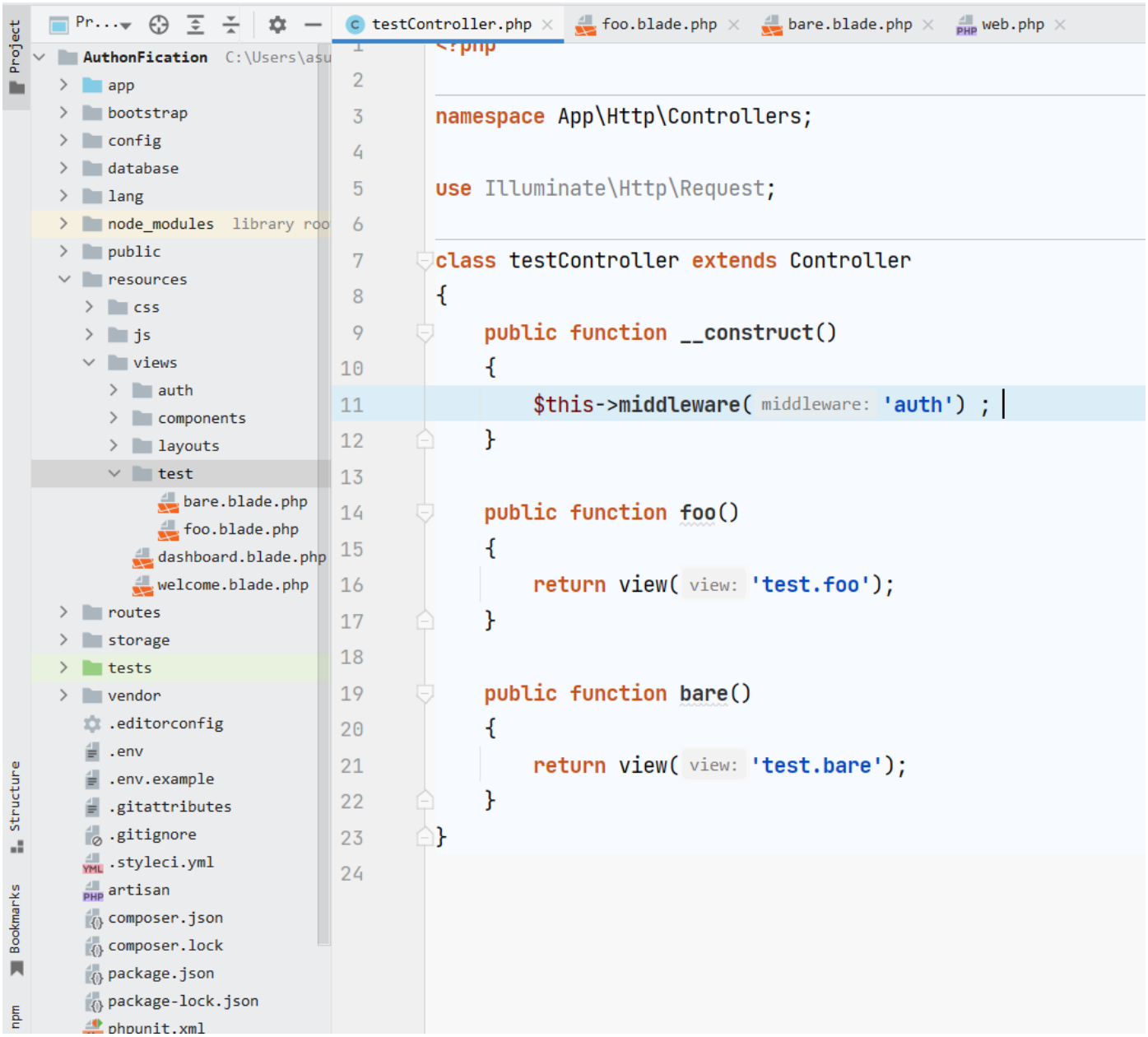
```
PS C:\Users\asus\Desktop\install\htdocs\AuthonFication> php artisan make:controller testController
```

```php
use App\Http\Controllers\testController;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get( uri: '/', function () {
    return view( view: 'welcome');
});

Route::get( uri: '/foo' , [testController::class,'test'])->name( name: 'foo') ;
Route::get( uri: '/bare' , [testController::class,'test'])->name( name: 'bare') ;

Route::get( uri: '/dashboard', function () {
    return view( view: 'dashboard');
})->middleware(['auth'])->name( name: 'dashboard');

require __DIR__.'/auth.php';
```

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class testController extends Controller
{
    public function foo()
    {
        return view( view: 'test.foo');
    }

    public function bare()
    {
        return view( view: 'test.bare');
    }
}
```

After creating these two views we need to make a middleware in order to verify some condition to allow acces to these two methods :

Now we gonna verify if the personn who want to get access to the two view is connected or not .



If we need filter the views which i can access without authentification :

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class testController extends Controller
{
    public function __construct()
    {
        $this->middleware( middleware: 'auth')->except(['bare','foo']) ;
    }

    public function foo()
    {
        return view( view: 'test.foo');
    }

    public function bare()
    {
        return view( view: 'test.bare');
    }
}
```

If we need display somthing just onlly if i m connected :

```blade
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
<center>
    <h1>
        bare
        @auth()
            <span>
                You are connected !!!
            </span>
        @endauth
    </h1>
</center>
</body>
</html>

```

With @auth i can display the name of the user and his informations :

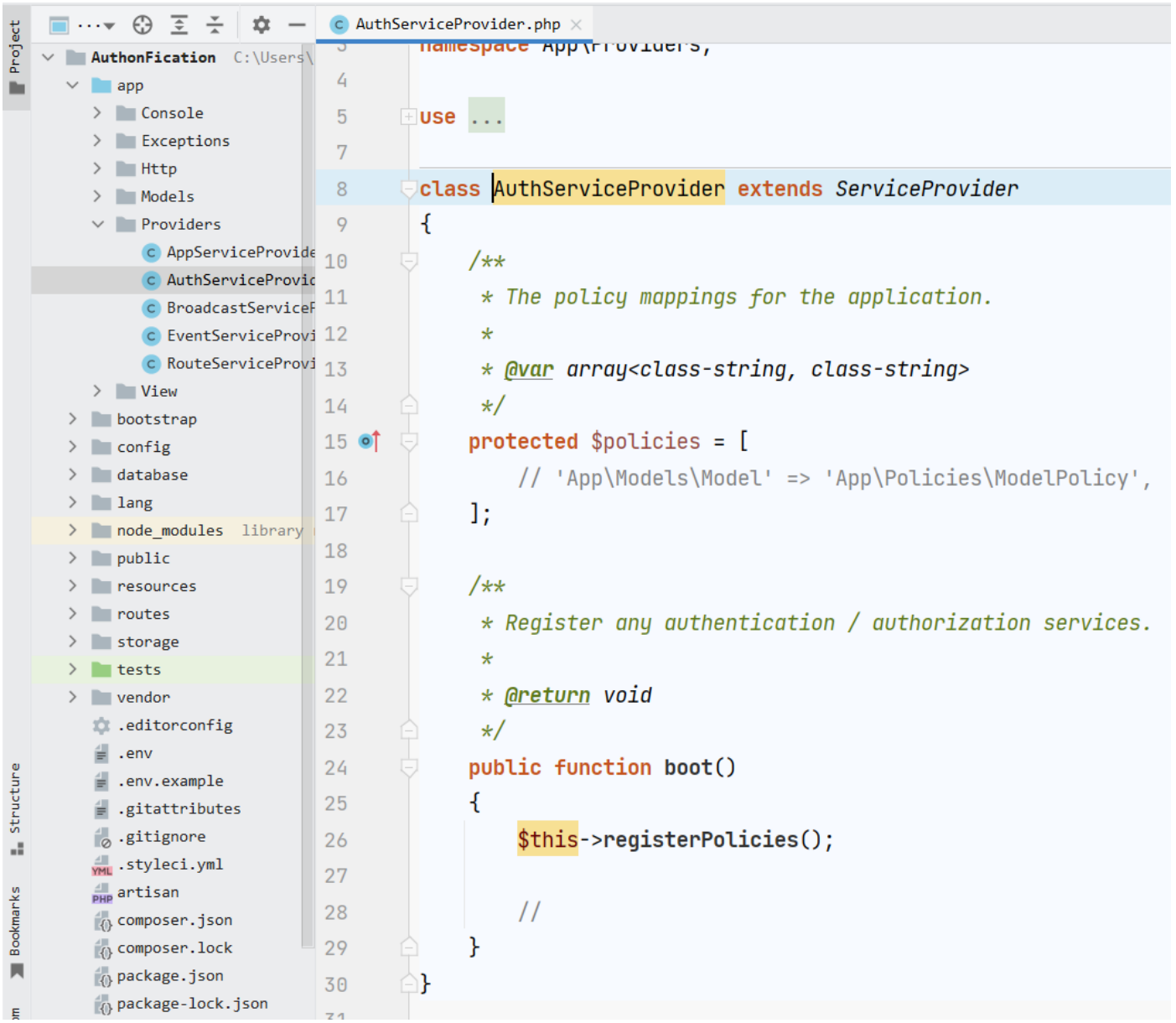
I can do the same thing if i want to display somthing if we are only guest :

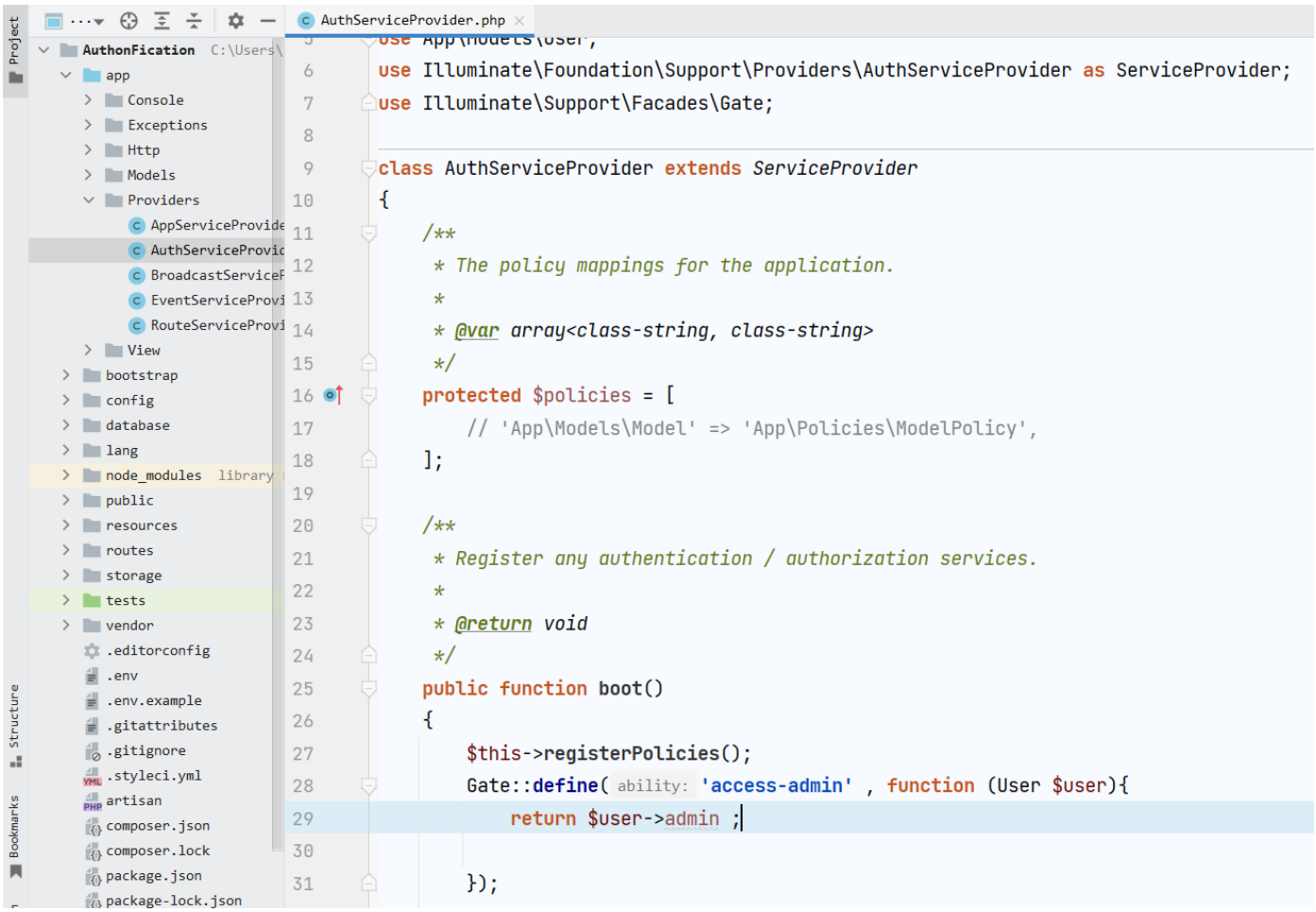
@guest :

Autorisation gates :

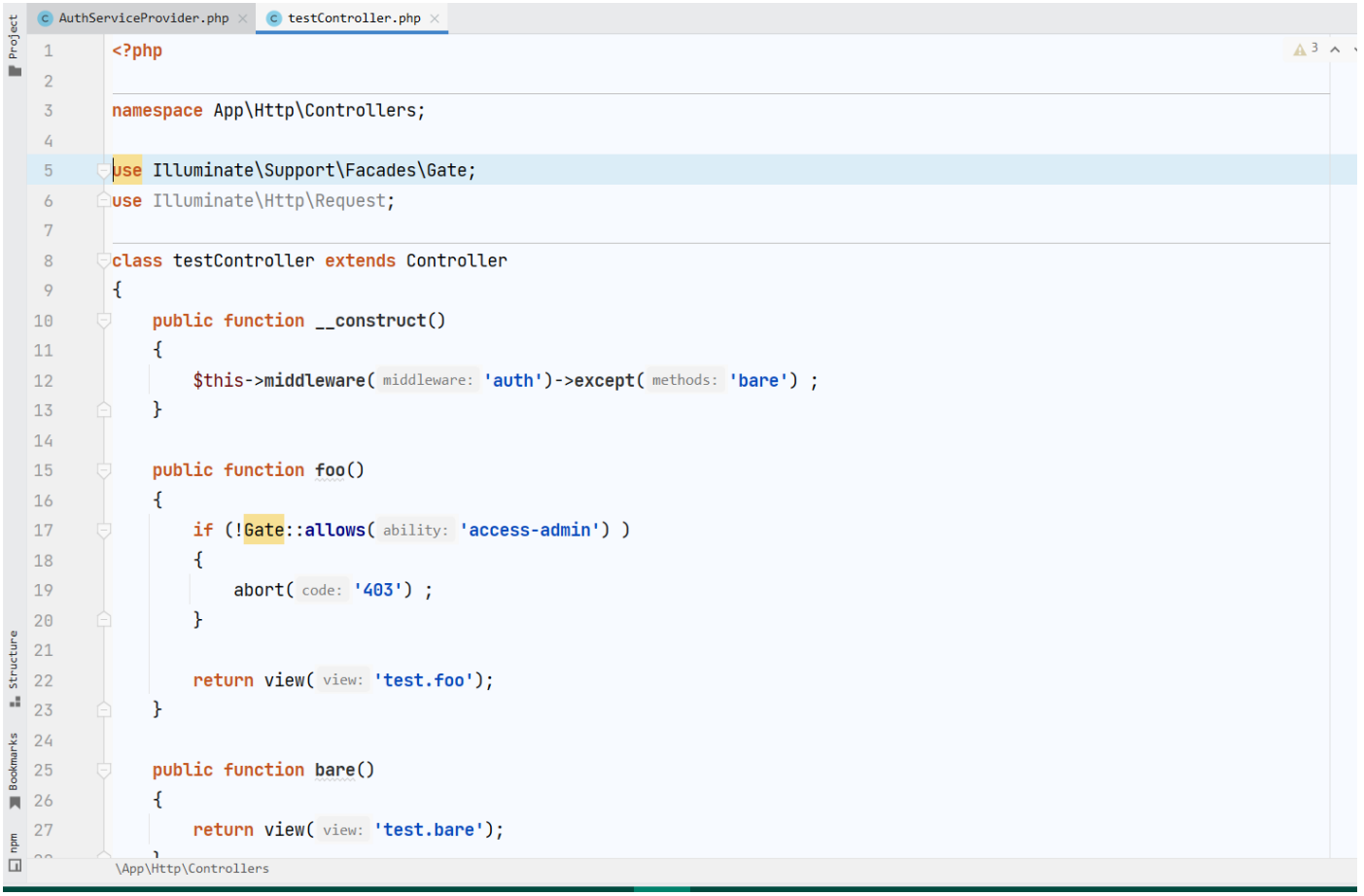
```
PS C:\Users\asus\Desktop\install\htdocs\AuthonFication> php artisan make:migration add_admin_column_to_users_table --table=users
```

```php
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table( table: 'users', function (Blueprint $table) {
            $table->boolean( column: 'admin')->default( value: 0) ;
        });
    }

    /**
     * Reverse the migrations.
     *
```


Project
AuthonFication C:\Users\
app
Console
Exceptions
Http
Models
Providers
AppServiceProvide
AuthServiceProvic
BroadcastServiceF
EventServiceProvi
RouteServiceProvi
View
bootstrap
config
database
lang
node_modules library
public
resources
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
.styleci.yml
artisan
composer.json
composer.lock
package.json
package-lock.json
Structure
Bookmarks
AuthServiceProvider.php
namespace App\Providers;
use ...
class AuthServiceProvider extends ServiceProvider
{
/**
* The policy mappings for the application.
*
* @var array<class-string, class-string>
*/
protected $policies = [
// 'App\Models\Model' => 'App\Policies\ModelPolicy',
];
/**
* Register any authentication / authorization services.
*
* @return void
*/
public function boot()
{
$this->registerPolicies();
//
}
}



If we need to access to 'foo' :

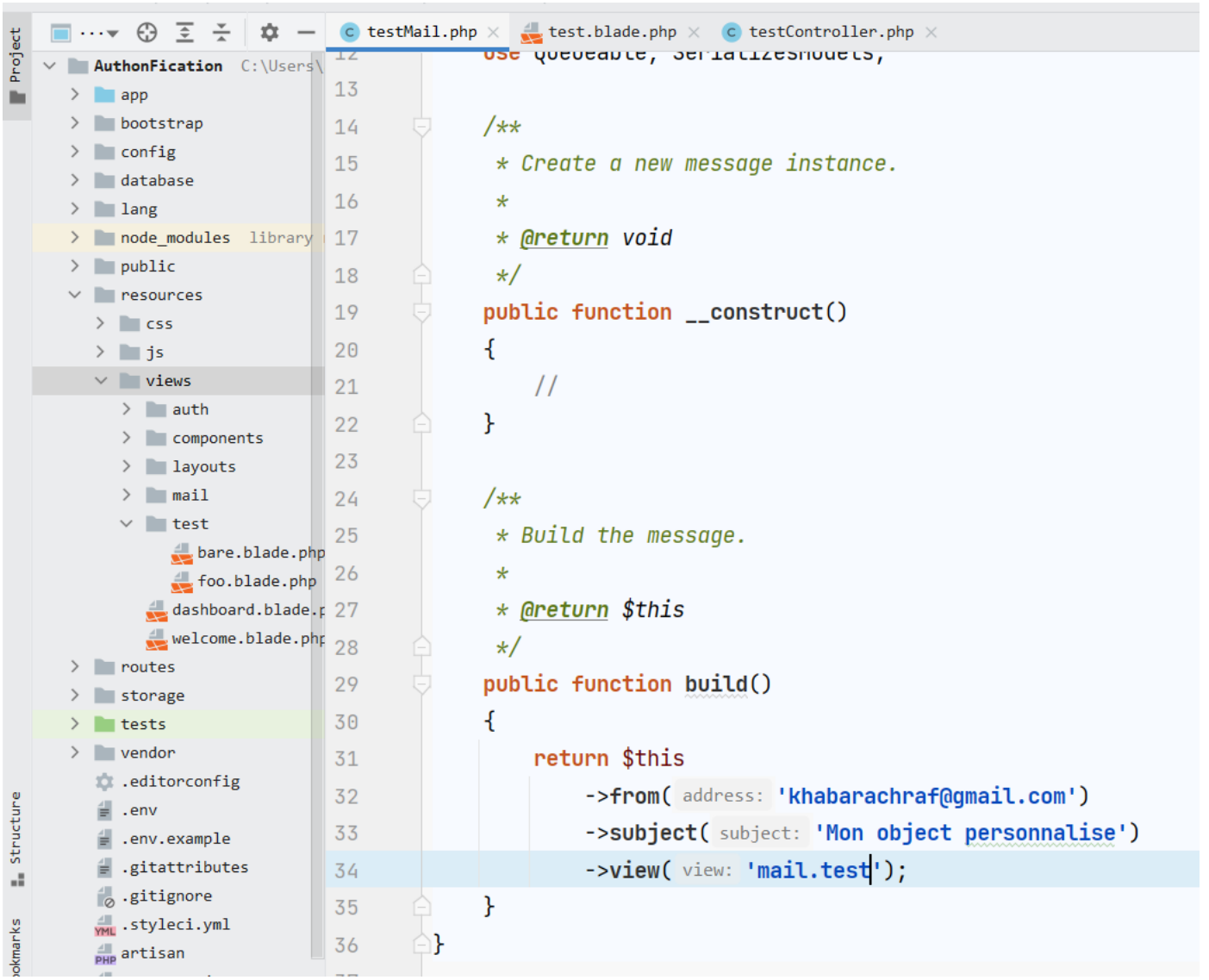


Emails

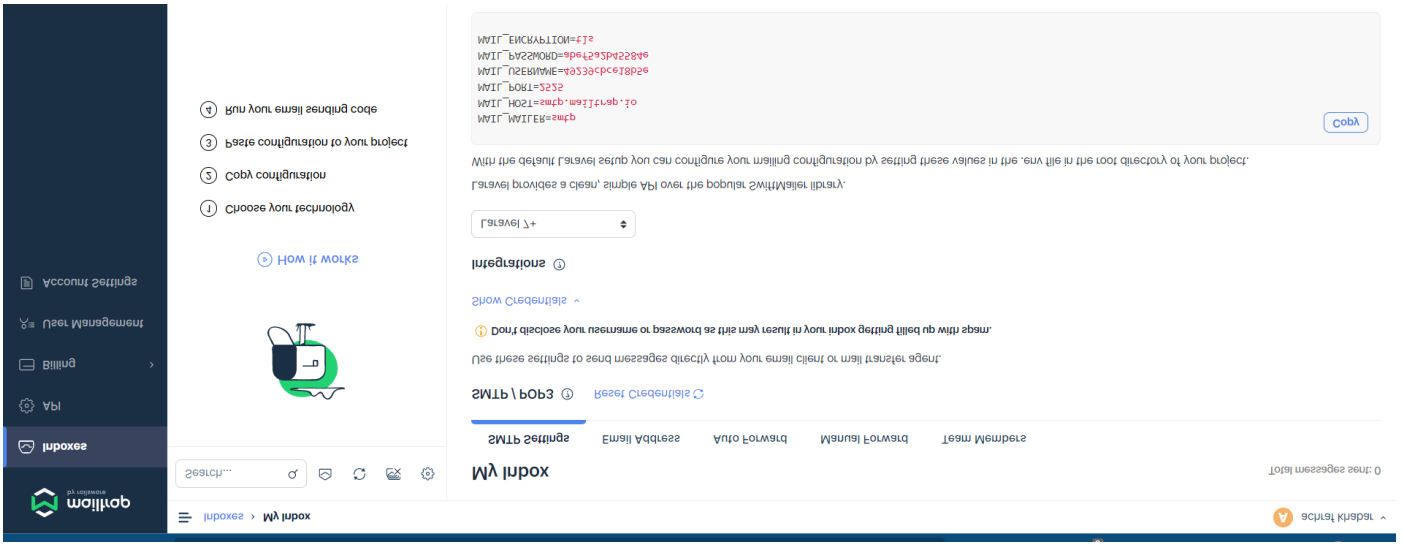
samedi 23 avril 2022
23:32

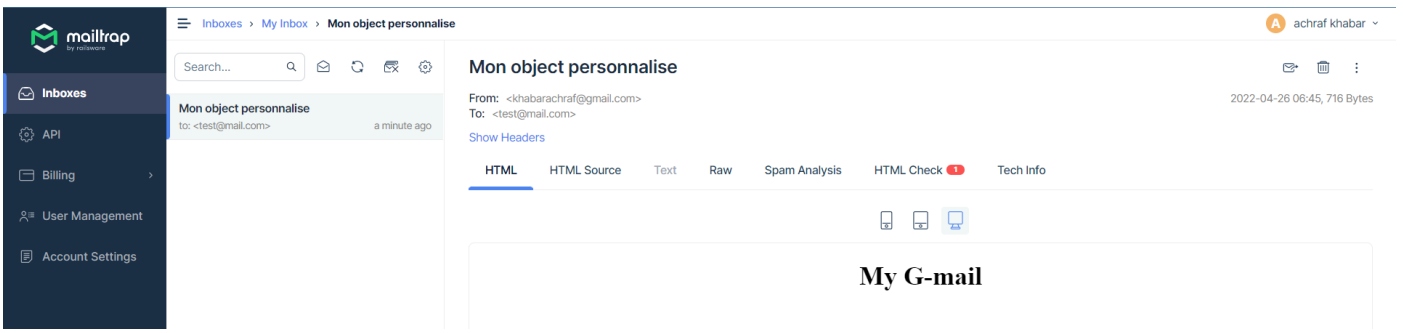
HOW TO CONNECT YOUR DATABASE WITH YOUR PHP CODE :

```
Mail created successfully.
PS C:\Users\asus\Desktop\install\htdocs\AuthonFication> php artisan make:mail testMail
```


Project
AuthonFication C:\Users\
app
bootstrap
config
database
lang
node_modules library
public
resources
css
js
views
auth
components
layouts
mail
test
bare.blade.php
foo.blade.php
dashboard.blade.p
welcome.blade.php
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
.styleci.yml
artisan
Structure
Bookmarks
testMail.php
test.blade.php
testController.php
use Queueable, SerializesModels;
/**
* Create a new message instance.
*
* @return void
*/
public function __construct()
{
//
}
/**
* Build the message.
*
* @return $this
*/
public function build()
{
return $this
->from(address: 'khabarachraf@gmail.com')
->subject(subject: 'Mon object personnalise')
->view(view: 'mail.test');
}
}


testMail.php
test.blade.php
testController.php
AuthonFication C:\Users\asu
app
Console
Exceptions
Http
Controllers
Auth
Controller.php
testController.ph
Middleware
Requests
Kernel.php
Mail
testMail.php
Models
Providers
View
bootstrap
config
database
lang
node_modules library roo
public
resources
css
js
views
auth
public function __construct()
{
$this->middleware(middleware: 'auth')->except(methods: 'bare') ;
}
public function foo()
{
if (!Gate::allows(ability: 'access-admin'))
{
abort(code: '403') ;
}
return view(view: 'test.foo');
}
public function bare()
{
Mail::to(users: 'test@mail.com')->send(new testMail()) ;
return view(view: 'test.bare');
}
}


How it works
Choose your technology
Copy configuration
Paste configuration to your project
Run your email sending code
My Inbox
SMTP Settings
Email Address
Auto Forward
Manual Forward
Team Members
SMTP / POP3
Integrations
Inboxes
API
Billing
User Management
Account Settings


mailtrap
Inboxes › My Inbox › Mon object personnalise
achraf khabar
Search...
Inboxes
API
Billing
User Management
Account Settings
Mon object personnalise
to: <test@mail.com>
a minute ago
Mon object personnalise
From: <khabarachraf@gmail.com>
To: <test@mail.com>
2022-04-26 06:45, 716 Bytes
Show Headers
HTML
HTML Source
Text
Raw
Spam Analysis
HTML Check
Tech Info
My G-mail

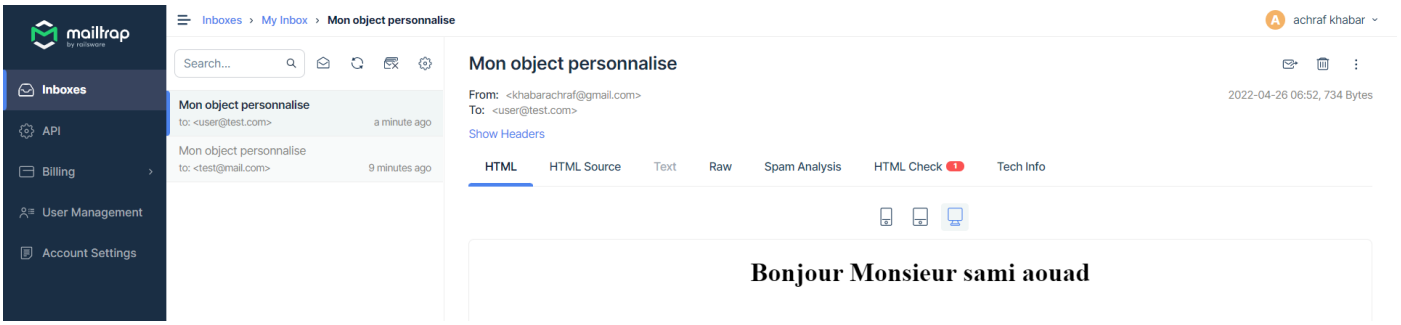
How to make a dynamic mail :


testMail.php
test.blade.php
testController.php
Project
Structure
Bookmarks
use App\Models\User;
use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Mail\Mailable;
use Illuminate\Queue\SerializesModels;
use PhpParser\Node\Expr\Array_;
class testMail extends Mailable
{
use Queueable, SerializesModels;
public $data = [] ;
public function __construct(Array $user)
{
$this->data = $user ;
}
public function build()
{
return $this
->from(address: 'khabarachraf@gmail.com')
->subject(subject: 'Mon object personnalise')
->view(view: 'mail.test');
}
}

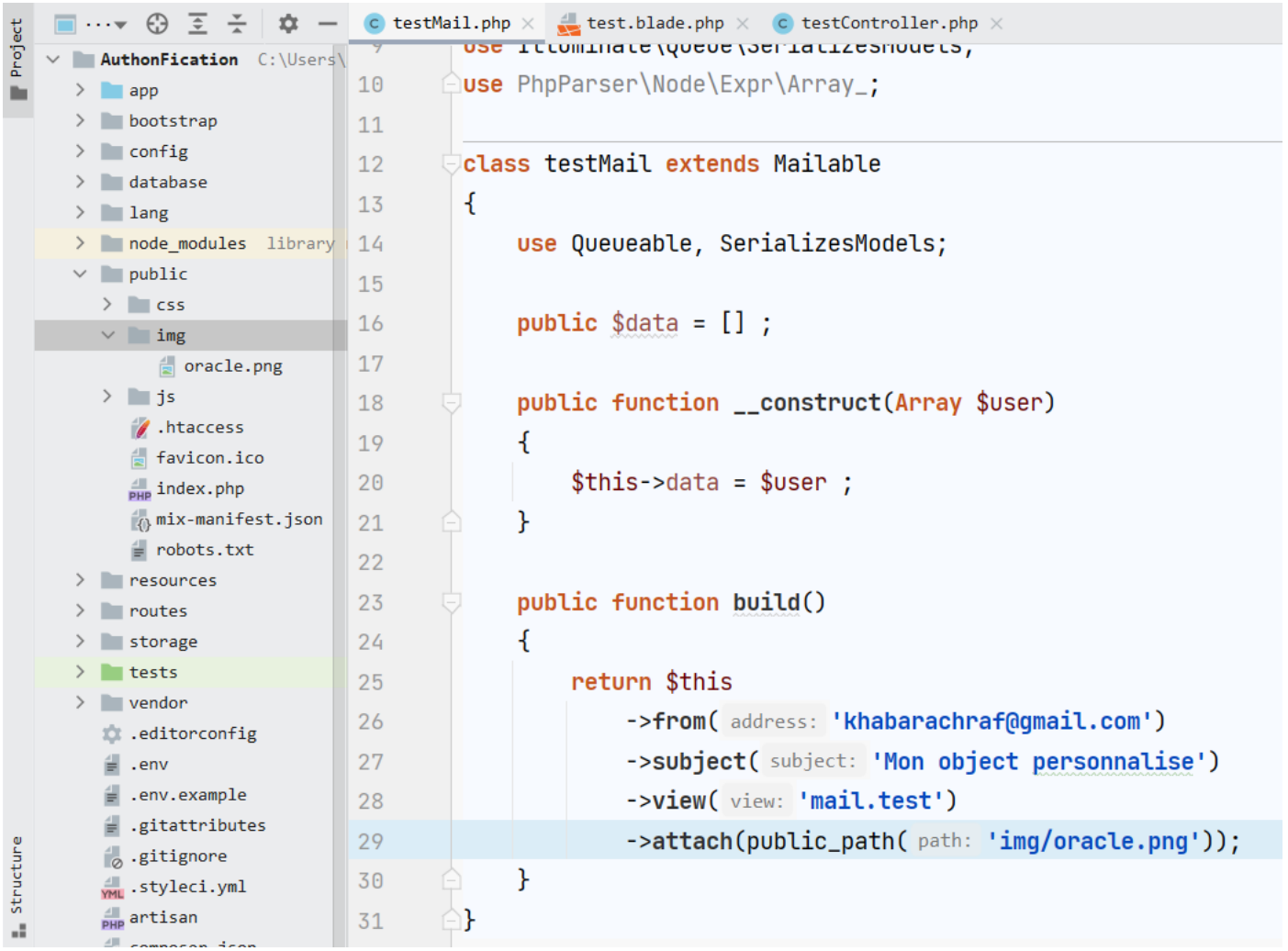
Project

testMail.php | test.blade.php | testController.php

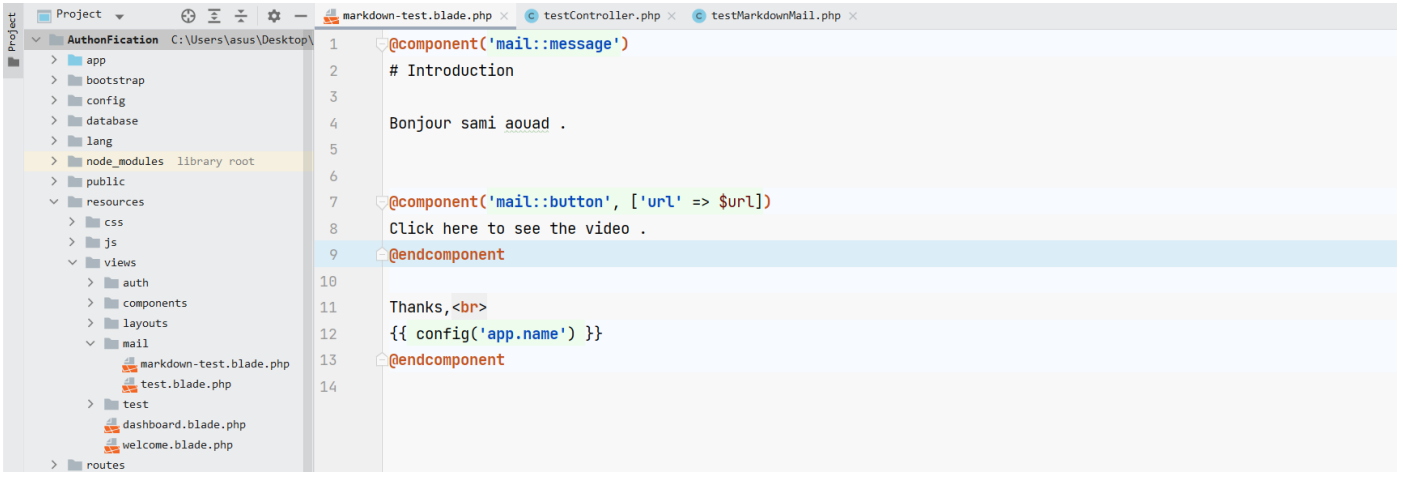
```
1  <!doctype html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport"
6            content="width=device-width, user-scalable=no
7      <meta http-equiv="X-UA-Compatible" content="ie=edge
8      <title>Document</title>
9  </head>
10 <body>
11     <center>
12         <h1>
13             Bonjour Monsieur {{ $data['name'] }}
14         </h1>
15     </center>
16
17 </body>
18 </html>
19
```

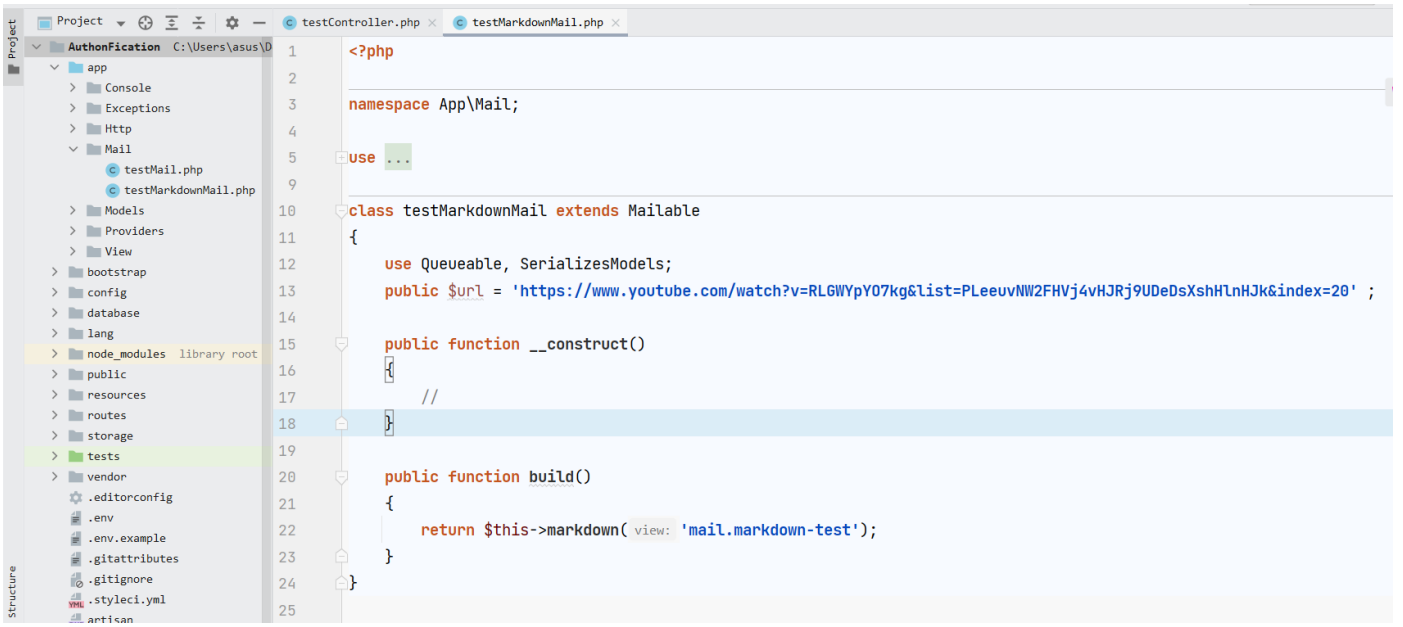


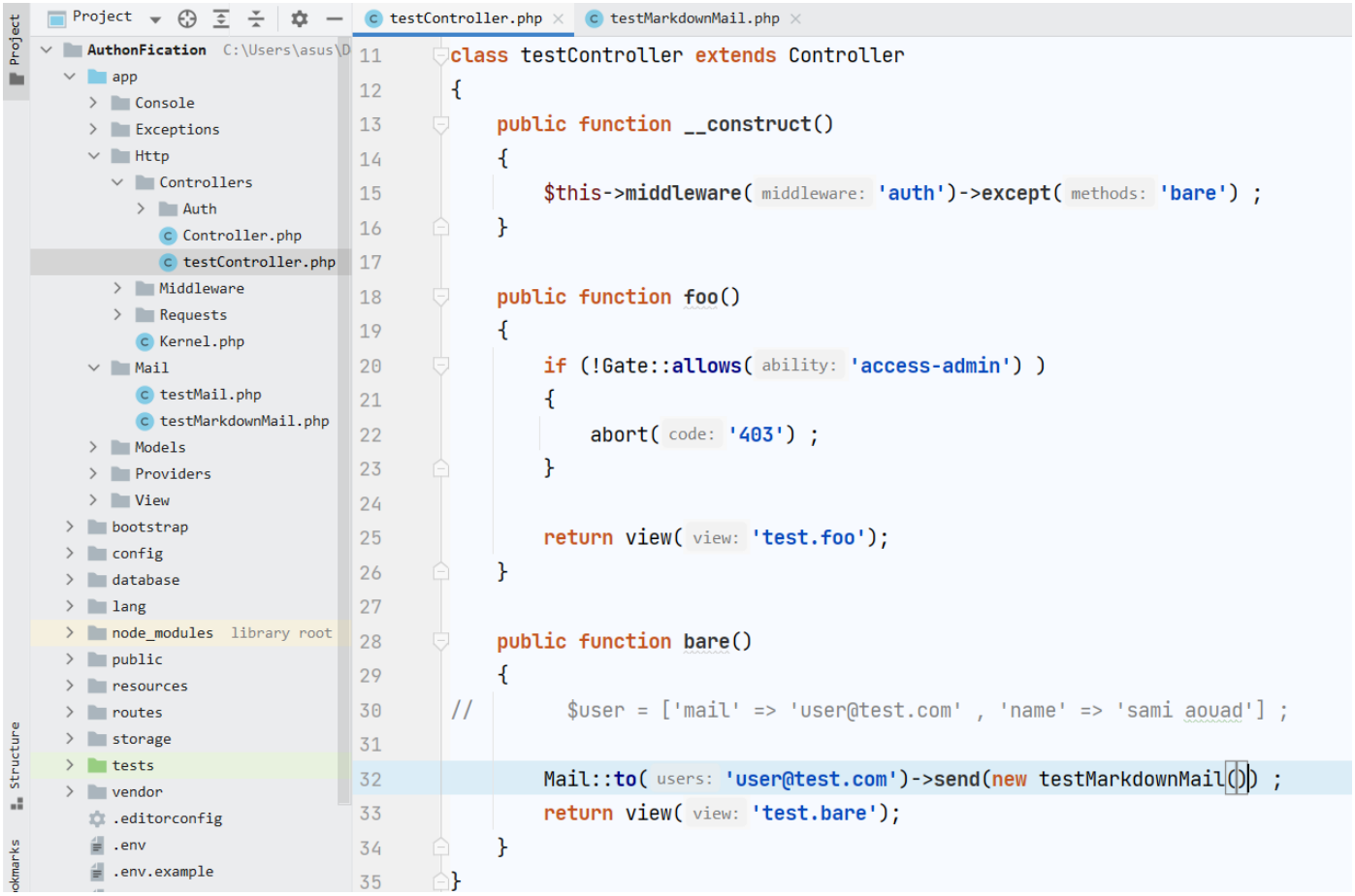
Attachements :



Generating mails with the mark down :


Project
AuthonFication C:\Users\asus\Desktop\
app
bootstrap
config
database
lang
node_modules library root
public
resources
css
js
views
auth
components
layouts
mail
markdown-test.blade.php
test.blade.php
test
dashboard.blade.php
welcome.blade.php
routes
markdown-test.blade.php
testController.php
testMarkdownMail.php
@component('mail::message')
Introduction
Bonjour sami aouad .
@component('mail::button', ['url' => $url])
Click here to see the video .
@endcomponent
Thanks,
{{ config('app.name') }}
@endcomponent

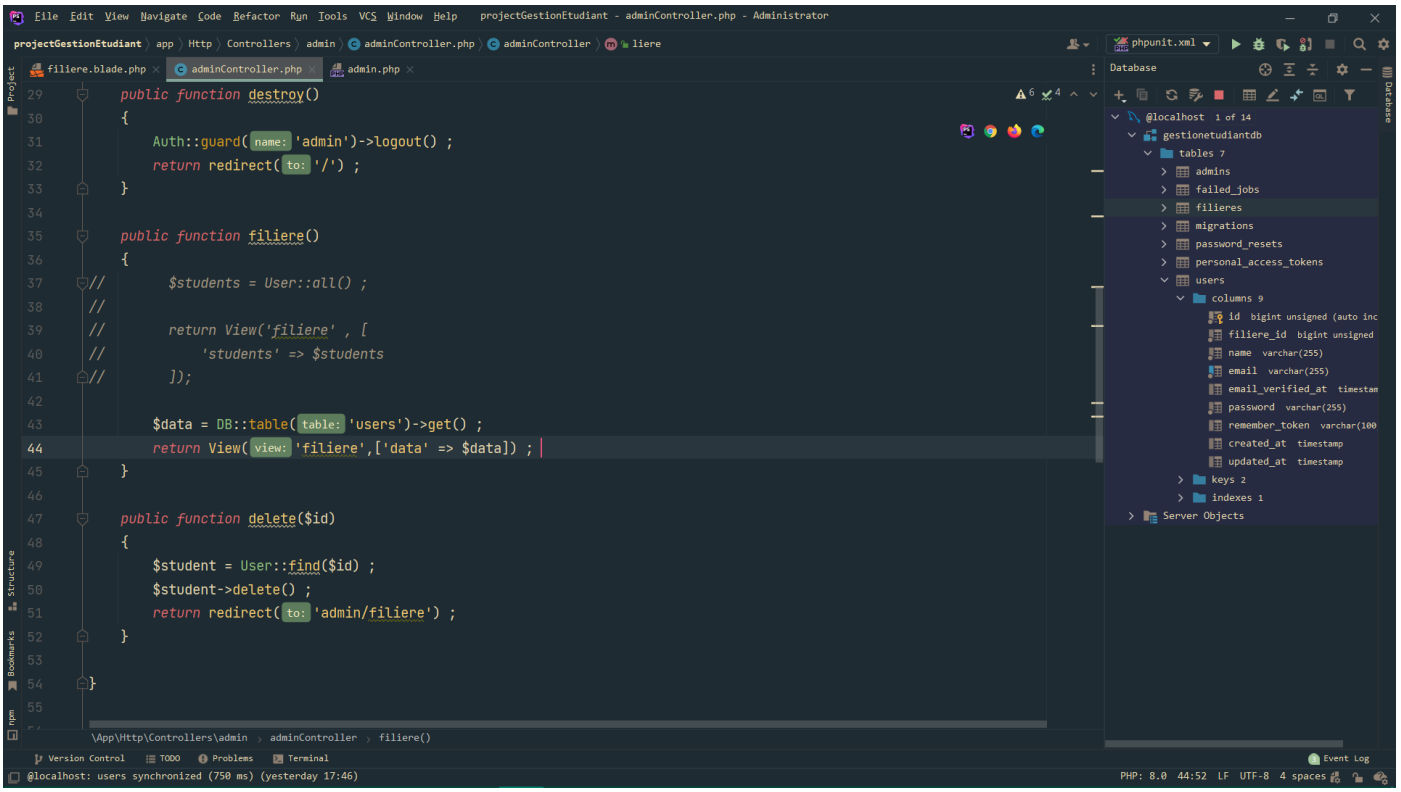

Project
AuthonFication C:\Users\asus\D
app
Console
Exceptions
Http
Mail
testMail.php
testMarkdownMail.php
Models
Providers
View
bootstrap
config
database
lang
node_modules library root
public
resources
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
.styleci.yml
artisan
Structure
testController.php
testMarkdownMail.php
<?php
namespace App\Mail;
use ...
class testMarkdownMail extends Mailable
{
use Queueable, SerializesModels;
public $url = 'https://www.youtube.com/watch?v=RLGWYpYO7kg&list=PLeeuvNW2FHVj4vHJRj9UDeDsXshHlnHJk&index=20' ;
public function __construct()
{
//
}
public function build()
{
return $this->markdown(view: 'mail.markdown-test');
}
}



Query builder

vendredi 29 avril 2022
00:45

Fetch all data from data base :



Use a query with where condition :


filiere.blade.php
adminController.php
admin.php
Project
public function destroy()
{
Auth::guard(name: 'admin')->logout() ;
return redirect(to: '/') ;
}
public function filiere()
{
// $students = User::all() ;
//
// return View('filiere' , [
// 'students' => $students
//]);
$data = DB::table(table: 'users')
->where(column: 'id', operator: 1)
->get() ;
return View(view: 'filiere',['data' => $data]) ;
}

Find :


filiere.blade.php
adminController.php
admin.php
Project
Structure
Bookmarks
npm
29 public function destroy()
30 {
31 Auth::guard(name: 'admin')->logout() ;
32 return redirect(to: '/') ;
33 }
34
35 public function filiere()
36 {
37 // $students = User::all() ;
38 //
39 // return View('filiere' , [
40 // 'students' => $students
41 //]);
42
43 $data = DB::table(table: 'users')->find(id: 1);
44
45 return View(view: 'filiere',['data' => $data]) ;
46 }
47
48 public function delete($id)
49 {
50 $student = User::find($id) ;
51 $student->delete() ;
52 return redirect(to: 'admin/filiere') ;
53 }
54
55 }

Count :


filiere.blade.php
adminController.php
admin.php
Project
Structure
Bookmarks
npm
29 public function destroy()
30 {
31 Auth::guard(name: 'admin')->logout() ;
32 return redirect(to: '/') ;
33 }
34
35 public function filiere()
36 {
37 // $students = User::all() ;
38 //
39 // return View('filiere' , [
40 // 'students' => $students
41 //]);
42
43 $data = DB::table(table: 'users')->count(columns: 'id');
44
45 return View(view: 'filiere',['data' => $data]) ;
46 }
47
48 public function delete($id)
49 {
50 $student = User::find($id) ;
51 $student->delete() ;
52 return redirect(to: 'admin/filiere') ;
53 }
54
55 }

Insert :


filiere.blade.php
adminController.php
admin.php
Project
Structure
Bookmarks
public function destroy()
{
Auth::guard(name: 'admin')->logout() ;
return redirect(to: '/') ;
}
public function filiere()
{
// $students = User::all() ;
//
// return View('filiere' , [
// 'students' => $students
//]);
$data = DB::table(table: 'users')
->insert([
'name' => 'sami aouad',
'email' => 'samiaouad@gmail.com' ,
]) ;
return View(view: 'filiere',['data' => $data]) ;
}
public function delete($id)
{
$student = User::find($id) ;
$student->delete() ;

Update :

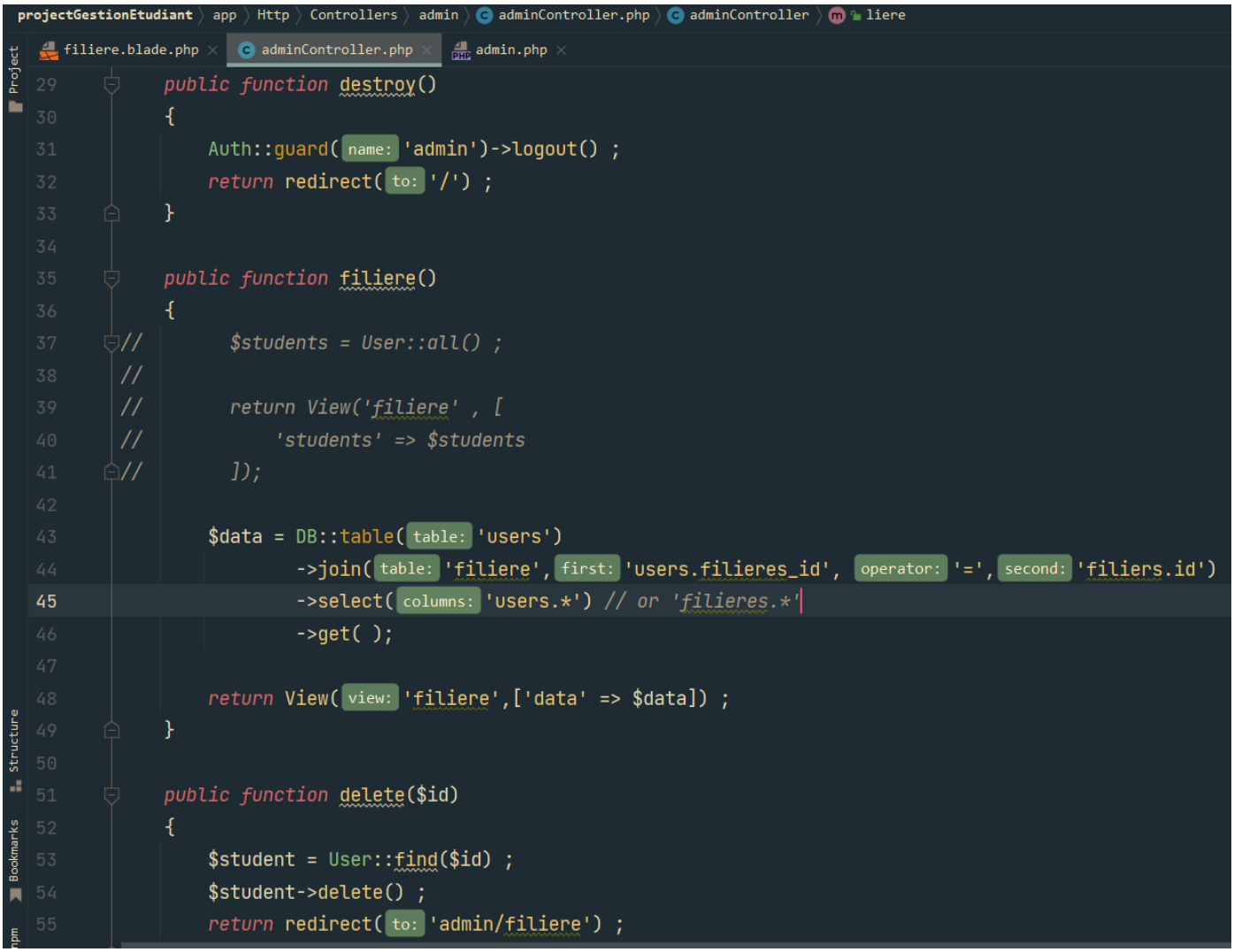

projectGestionEtudiant app Http Controllers admin adminController.php adminCont
filiere.blade.php
adminController.php
admin.php
Project
Structure
Bookmarks
29 public function destroy()
30 {
31 Auth::guard(name: 'admin')->logout() ;
32 return redirect(to: '/') ;
33 }
34
35 public function filiere()
36 {
37 // $students = User::all() ;
38 //
39 // return View('filiere' , [
40 // 'students' => $students
41 //]);
42
43 $data = DB::table(table: 'users')
44 ->where(column: 'id', operator: 6)
45 ->update([
46 'name' => 'sami aouad',
47 'email' => 'samiaouad@gmail.com' ,
48]) ;
49
50 return View(view: 'filiere',['data' => $data]) ;
51 }
52
53 public function delete($id)
54 {
55 $student = User::find($id) ;

Delete :

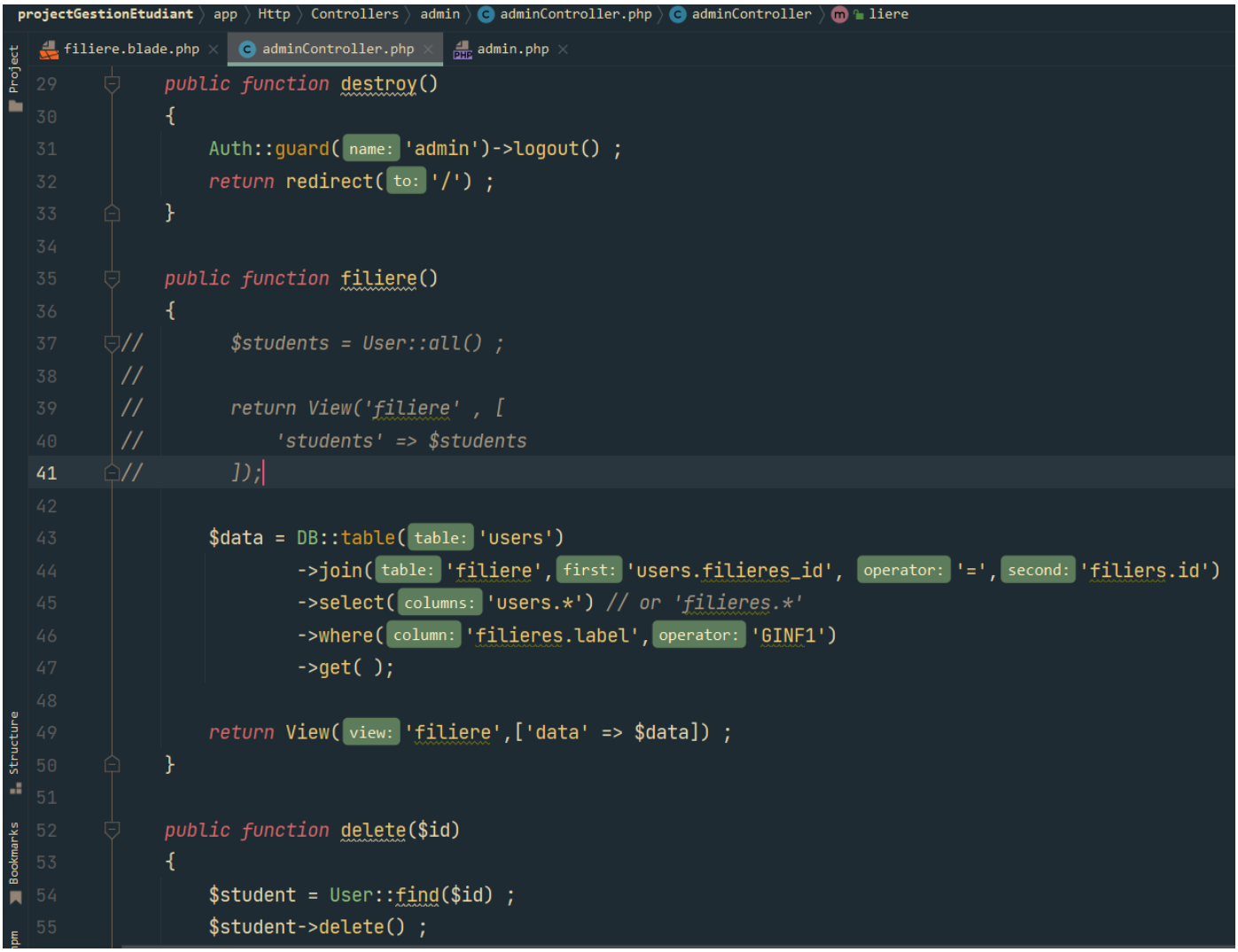

projectGestionEtudiant app Http Controllers admin adminController.php adminCo
filiere.blade.php
adminController.php
admin.php
Project
Structure
Bookmarks
pm
29 public function destroy()
30 {
31 Auth::guard(name: 'admin')->logout() ;
32 return redirect(to: '/') ;
33 }
34
35 public function filiere()
36 {
37 // $students = User::all() ;
38 //
39 // return View('filiere' , [
40 // 'students' => $students
41 //]);
42
43 $data = DB::table(table: 'users')
44 ->where(column: 'id', operator: 22)
45 ->delete() ;
46
47 return View(view: 'filiere',['data' => $data]) ;
48 }
49
50 public function delete($id)
51 {
52 $student = User::find($id) ;
53 $student->delete() ;
54 return redirect(to: 'admin/filiere') ;
55 }

Join :

Get data from a specific table :



Add condition in join :



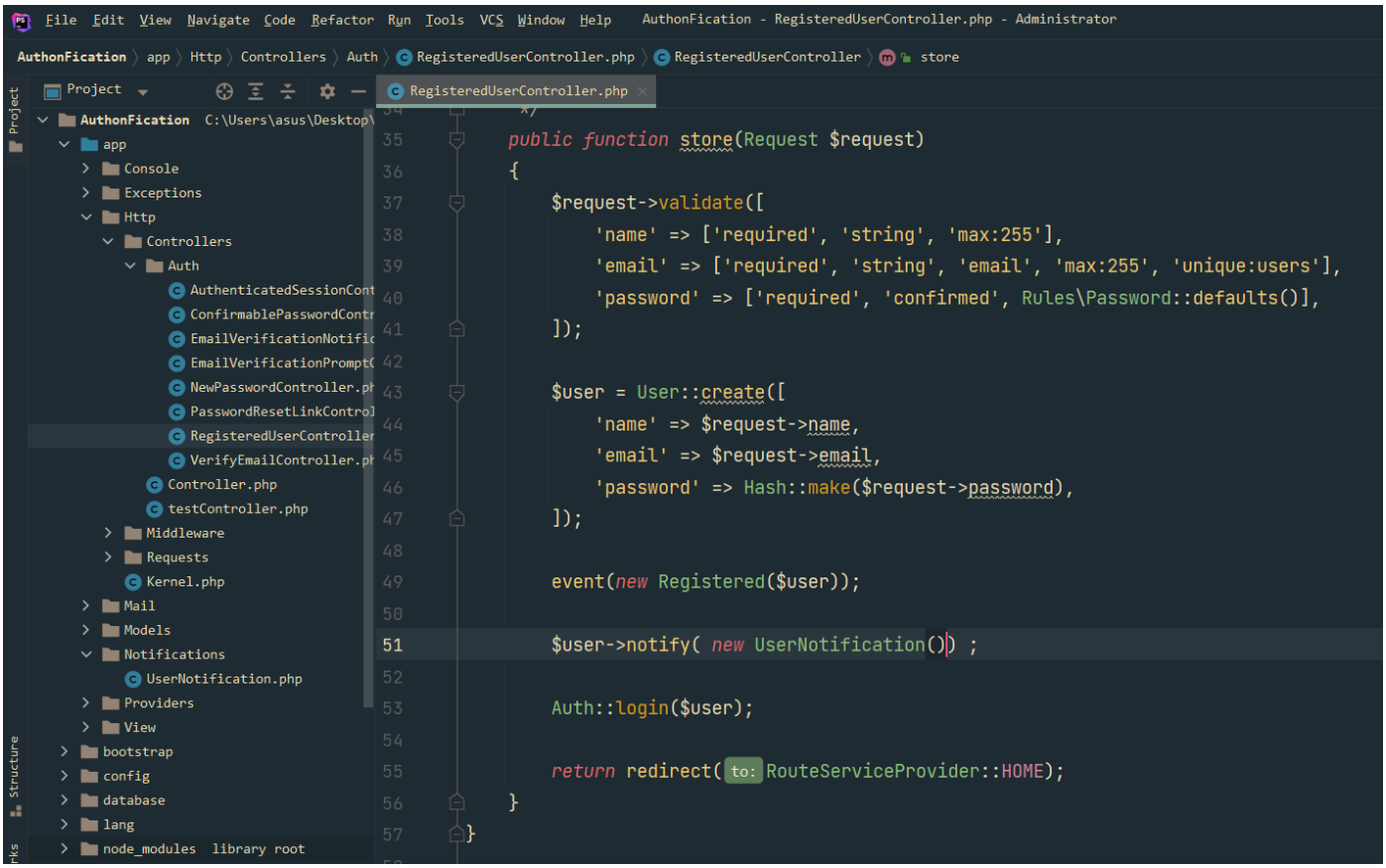
Notifications

dimanche 1 mai 2022
00:39

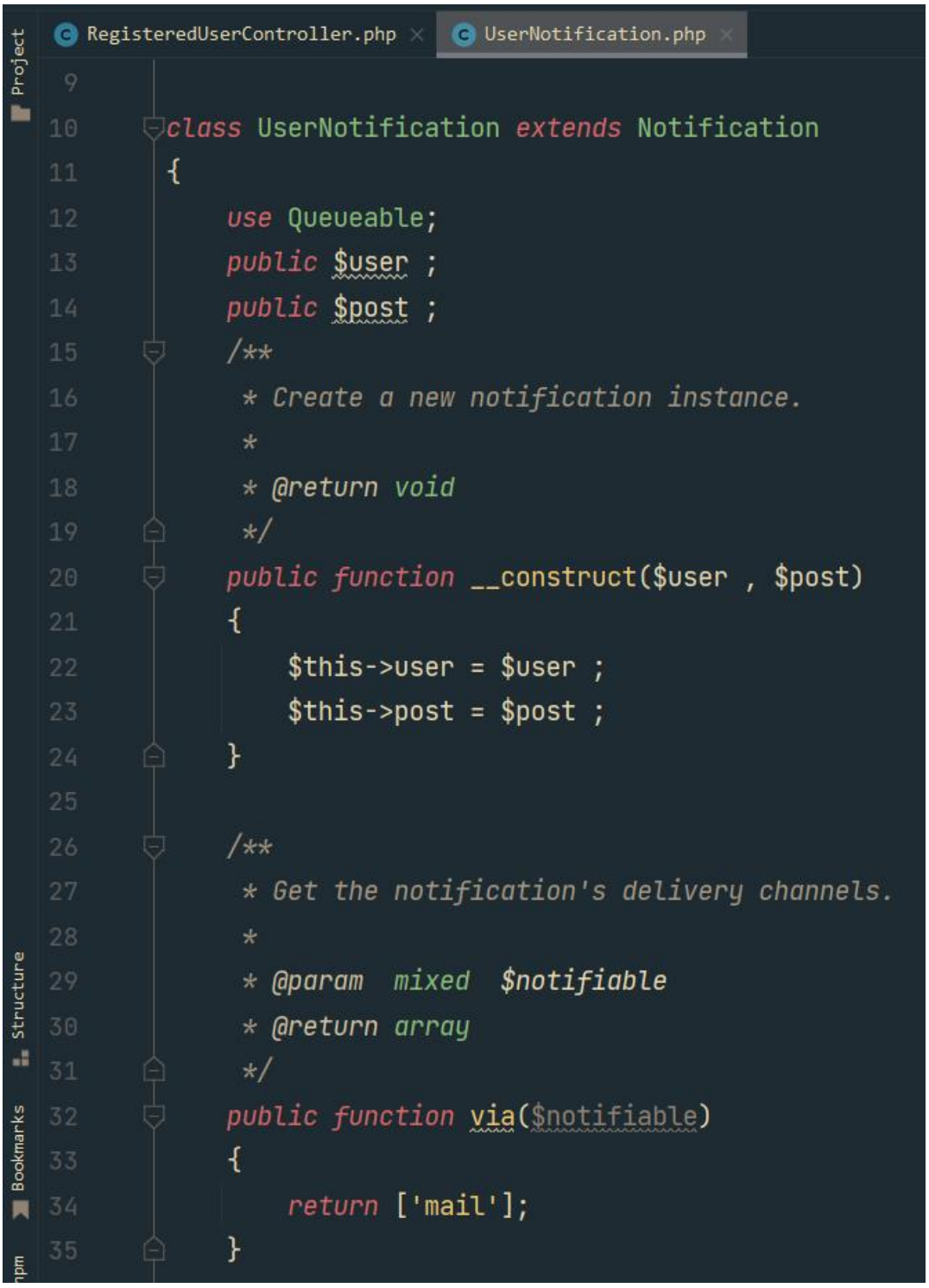
How to create a notofocation :

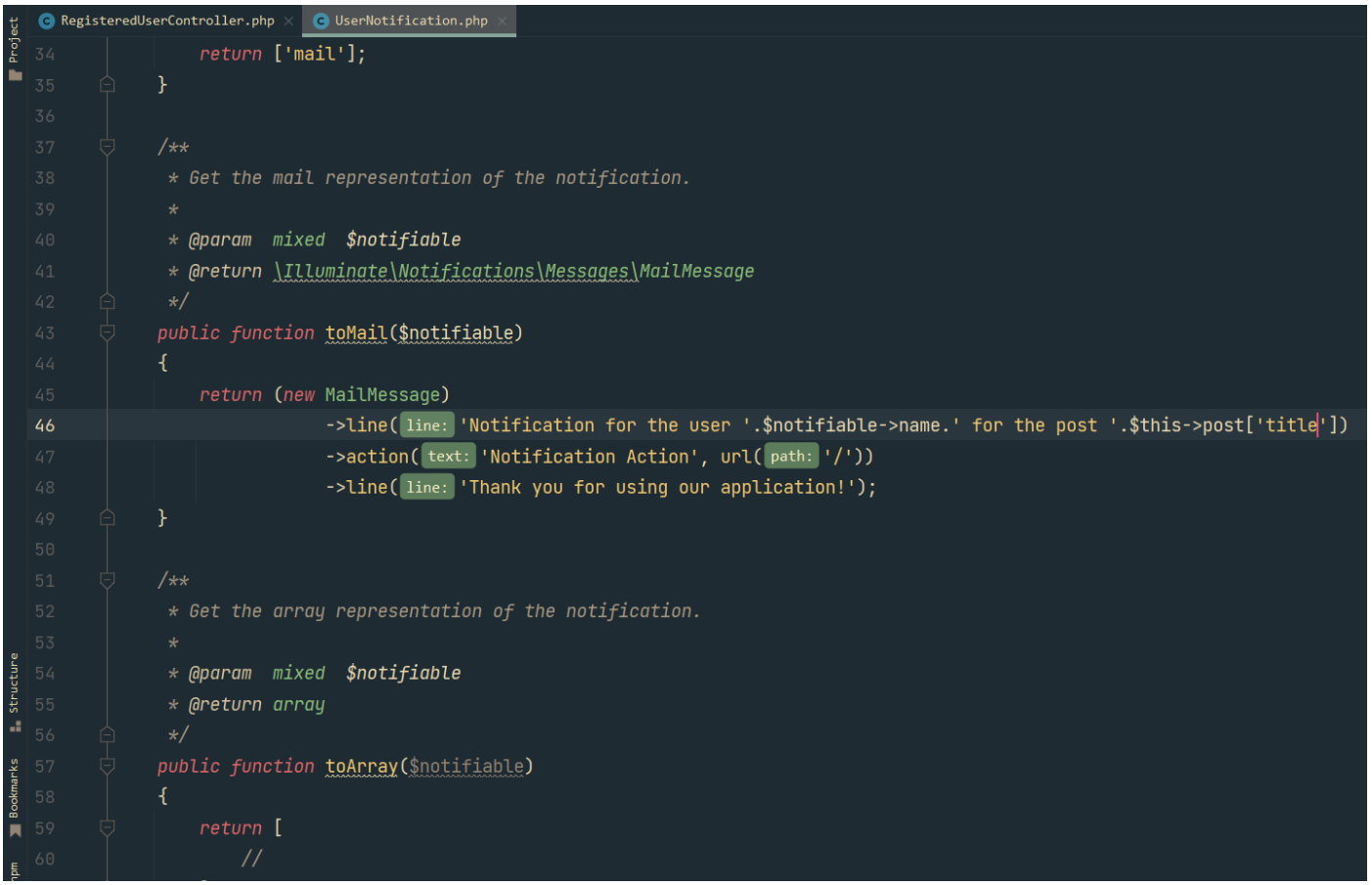
```
PS C:\Users\asus\Desktop\project gestion etudiant laravel\back-end\projectGestionEtudiant> php artisan make:notification UserNotification
```

How to make a notification : (we choose the model wich we want it to be notifiable) :

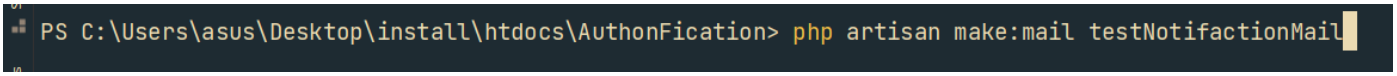


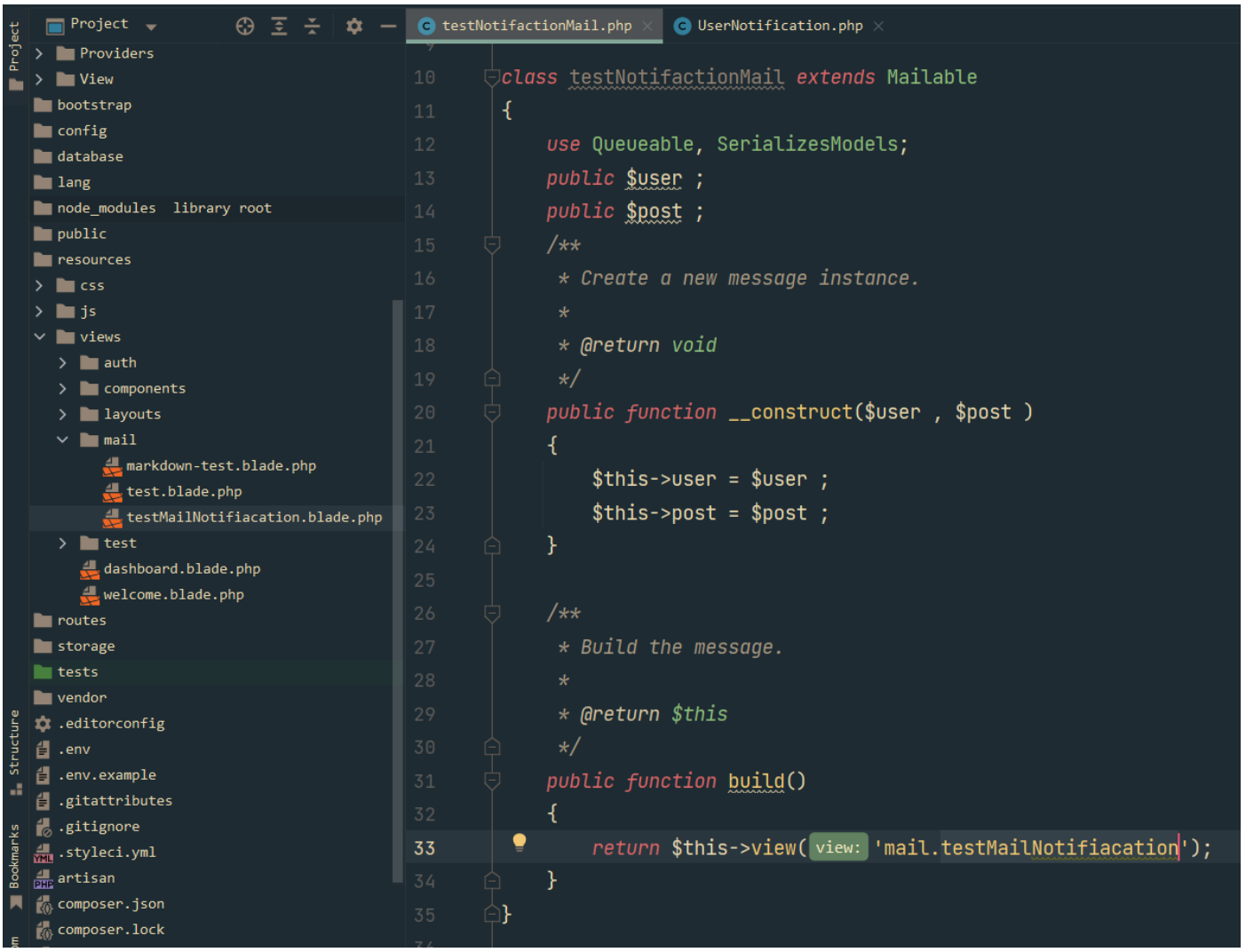
Add somthing to the mail configuration :

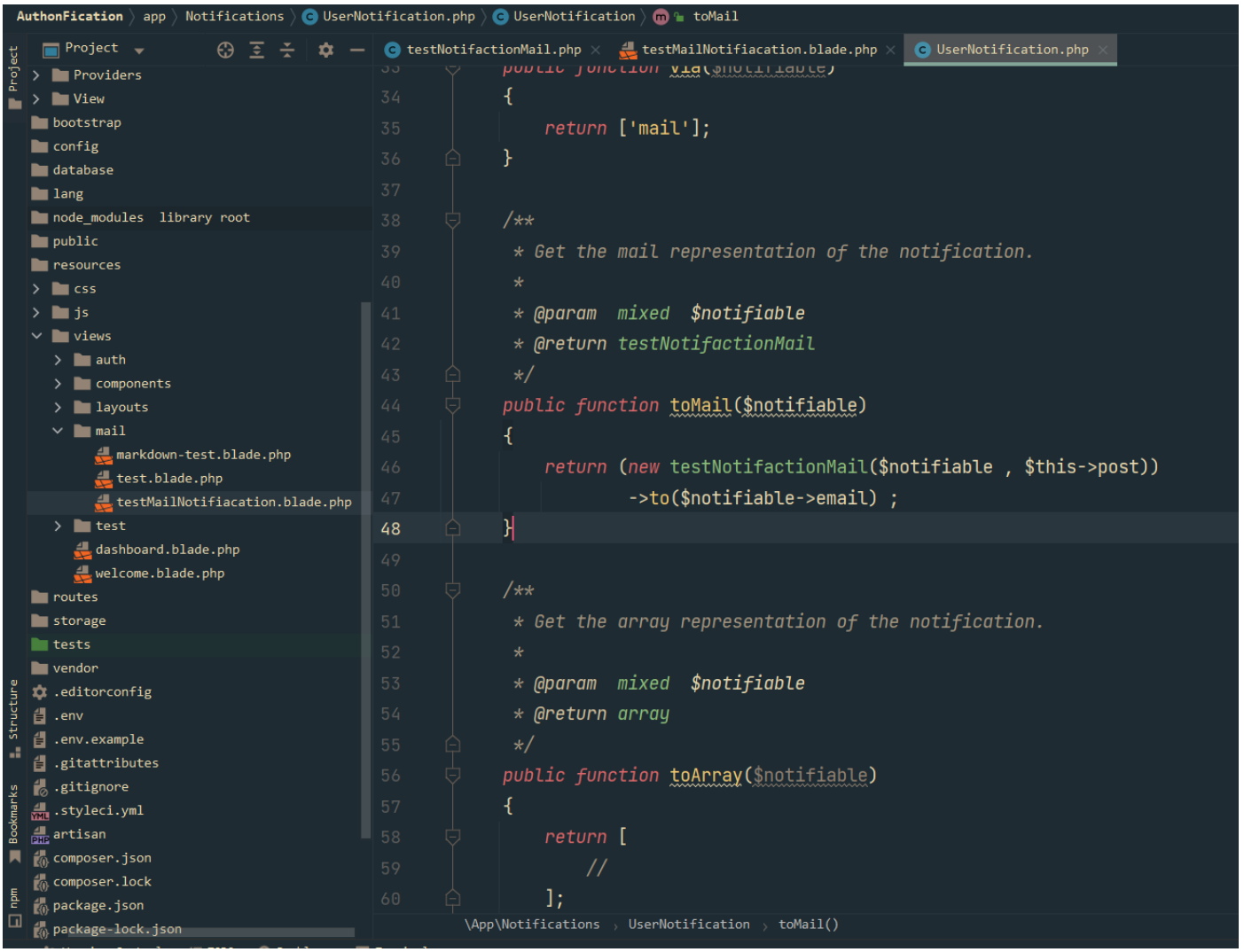

RegisteredUserController.php
UserNotification.php
Project
Structure
Bookmarks
npm
9
10 class UserNotification extends Notification
11 {
12 use Queueable;
13 public $user ;
14 public $post ;
15 /**
16 * Create a new notification instance.
17 *
18 * @return void
19 */
20 public function __construct($user , $post)
21 {
22 $this->user = $user ;
23 $this->post = $post ;
24 }
25
26 /**
27 * Get the notification's delivery channels.
28 *
29 * @param mixed $notifiable
30 * @return array
31 */
32 public function via($notifiable)
33 {
34 return ['mail'];
35 }



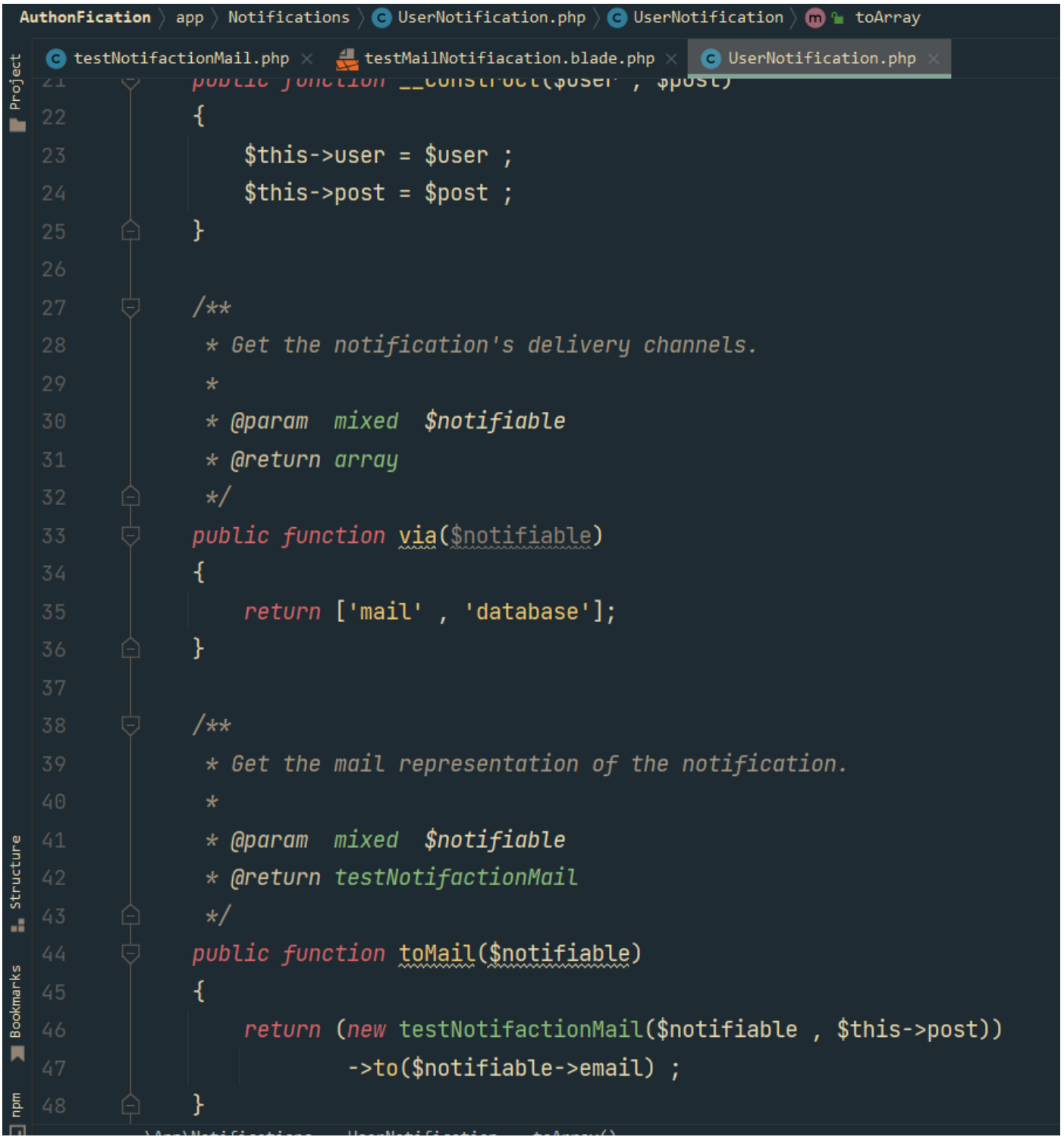
How to use our mail :



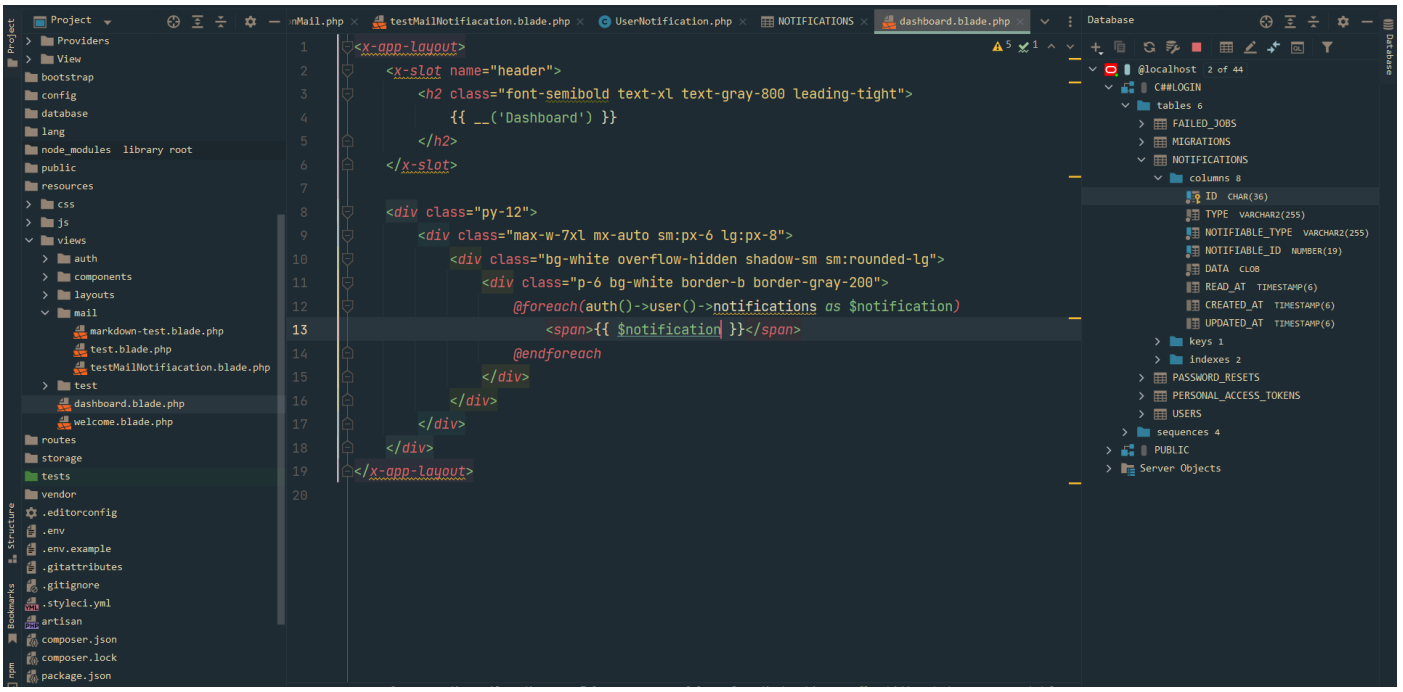

Project
Providers
View
bootstrap
config
database
lang
node_modules library root
public
resources
css
js
views
auth
components
layouts
mail
markdown-test.blade.php
test.blade.php
testMailNotifiacation.blade.php
test
dashboard.blade.php
welcome.blade.php
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
.styleci.yml
artisan
composer.json
composer.lock
testNotifactionMail.php
UserNotification.php
class testNotifactionMail extends Mailable
{
use Queueable, SerializesModels;
public $user ;
public $post ;
/**
* Create a new message instance.
*
* @return void
*/
public function __construct($user , $post)
{
$this->user = $user ;
$this->post = $post ;
}
/**
* Build the message.
*
* @return $this
*/
public function build()
{
return $this->view(view: 'mail.testMailNotifiacation');
}
}

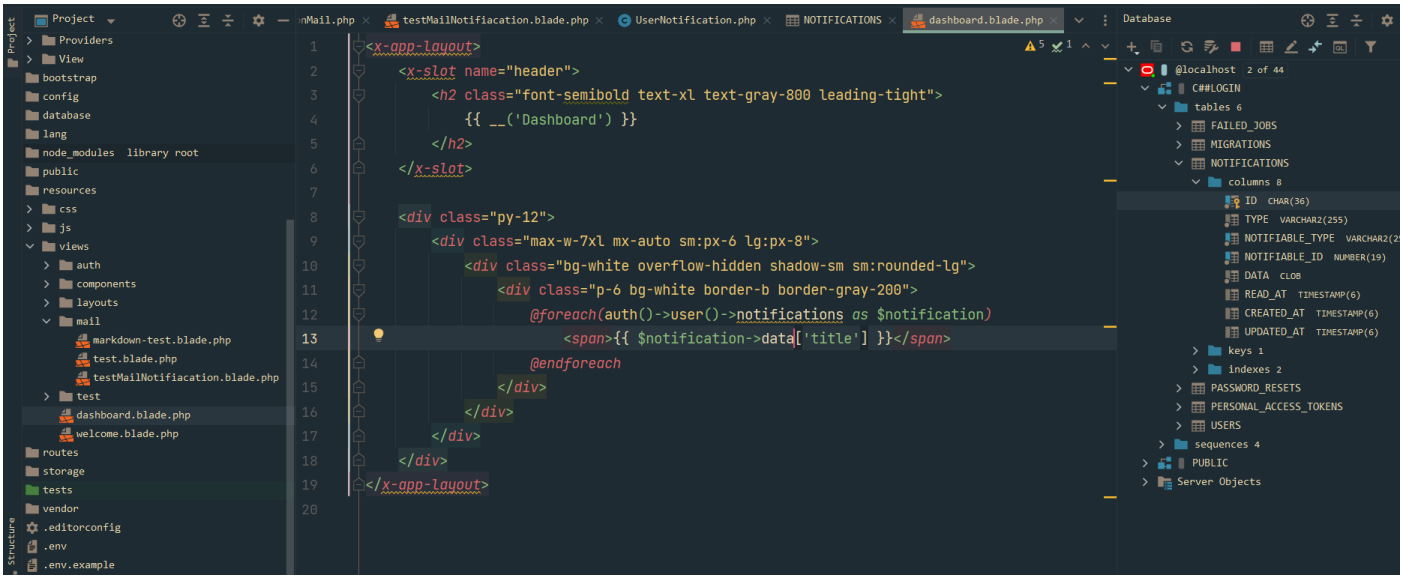


Notifications and databae :


AuthonFication app Notifications UserNotification.php UserNotification toArray
testNotifactionMail.php
testMailNotifiacation.blade.php
UserNotification.php
Project
Structure
Bookmarks
npm
21 public function __construct($user , $post)
22 {
23 $this->user = $user ;
24 $this->post = $post ;
25 }
26
27 /**
28 * Get the notification's delivery channels.
29 *
30 * @param mixed $notifiable
31 * @return array
32 */
33 public function via($notifiable)
34 {
35 return ['mail' , 'database'];
36 }
37
38 /**
39 * Get the mail representation of the notification.
40 *
41 * @param mixed $notifiable
42 * @return testNotifactionMail
43 */
44 public function toMail($notifiable)
45 {
46 return (new testNotifactionMail($notifiable , $this->post))
47 ->to($notifiable->email) ;
48 }


testNotifactionMail.php
testMailNotifiacation.blade.php
UserNotification.php
Project
tructure
* @return testNotifactionMail
*/
public function toMail($notifiable)
{
return (new testNotifactionMail($notifiable , $this->post))
->to($notifiable->email) ;
}
/**
* Get the array representation of the notification.
*
* @param mixed $notifiable
* @return array
*/
public function toArray($notifiable)
{
return [
'title' => 'my title : '.$this->post['title'] ,
'my_email' => $notifiable->email
];
}
}

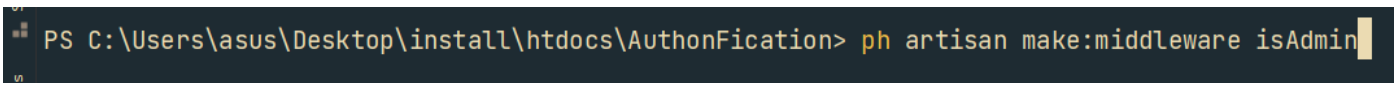


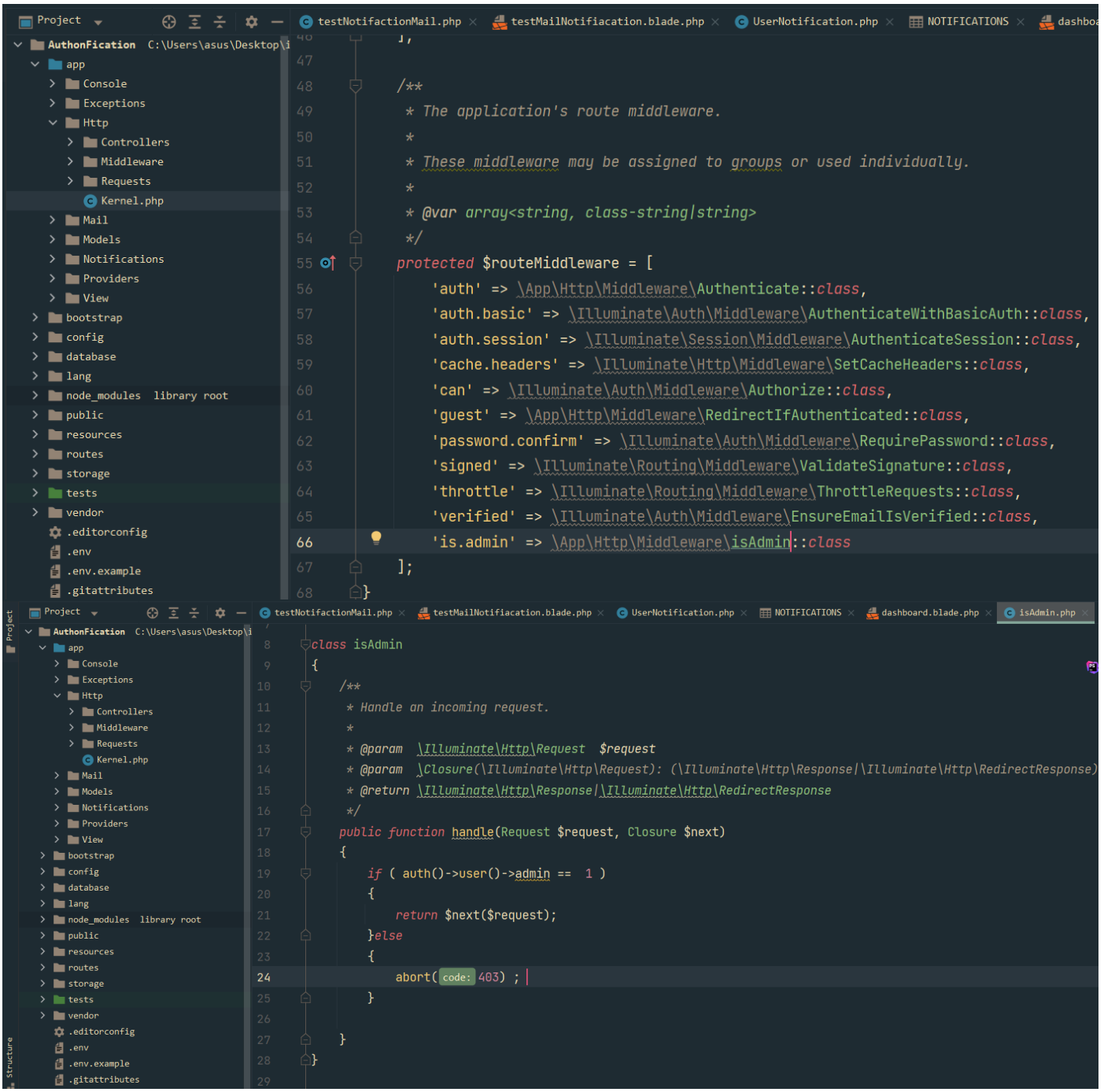


Middleware

mardi 3 mai 2022
23:17

How to create our own middleware :




Project
AuthonFication C:\Users\asus\Desktop\i
app
Console
Exceptions
Http
Controllers
Middleware
Requests
Kernel.php
Mail
Models
Notifications
Providers
View
bootstrap
config
database
lang
node_modules library root
public
resources
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
testNotifactionMail.php
testMailNotifiacation.blade.php
UserNotification.php
NOTIFICATIONS
dashboa
* The application's route middleware.
* These middleware may be assigned to groups or used individually.
* @var array<string, class-string|string>
protected $routeMiddleware = [
'auth' => \App\Http\Middleware\Authenticate::class,
'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
'can' => \Illuminate\Auth\Middleware\Authorize::class,
'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,
'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
'is.admin' => \App\Http\Middleware\isAdmin::class
];
}
dashboard.blade.php
isAdmin.php
class isAdmin
{
* Handle an incoming request.
* @param \Illuminate\Http\Request $request
* @param \Closure(\Illuminate\Http\Request): (\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)
* @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
public function handle(Request $request, Closure $next)
{
if (auth()->user()->admin == 1)
{
return $next($request);
}else
{
abort(code: 403) ;
}
}
}
Structure