

FRAMEWORK LARAVEL PHP

SOMMAIRE

Explication le Framework Laravel:

1	Intruduction	3
2	Installation de Laravel	3
3	Organisation de Laravel	4

Exemple de user management, rôle, permission and file :

1	Laravel 7 installation	6
2	Configuration le fichier .env et creation detabase	6
3	Installation les packages de composer	7
4	Création de Model	8
5	Ajouter des Middlewares	9
6	Création de l'authentification	9
7	Création le routage des pages views	10
8	Ajouter des controllers	10
9	Ajouter des fichier Blade	10
10	Création des fichiers css et js	11
11	Création seerder pour super-admin user	11
12	Exécution de l'application	12
13	Les pages créées	12
14	Les ressources inspirées.....	13
15	Conclusion	14

1. Intruduction :

Laravel est un framework web open-source écrit en PHP respectant le principe de modèle-vue-controlleur (MVC) et entièrement développé en programmation orientée objet (POO). Laravel est distribué sous licence MIT, avec ses sources hébergées sur GithUB.

Laravel à été créé par Tylor Otwell en juin 2011. En 2016 ce framework est devenue le projet PHP le mieux noté de GitHub, au pourtant reste basé sur son grand frère Symfony, pour au moins 30% de ses lignes.

Laravel fournit des fonctionnalités en termes de routage de requête, de mapping objet-relationnel, d'authentification, de vue (avec Blade), de migration de base de données , de gestion des exceptions et de test unitaire.

2. Installation :

Pour pouvoir utiliser Laravel (la version que j'ai utilisée: **7.3.0**), on a besoin d'installé PHP (version que j'ai installée: **7.4.3**), Mysql et apache2 (ou Nginx).

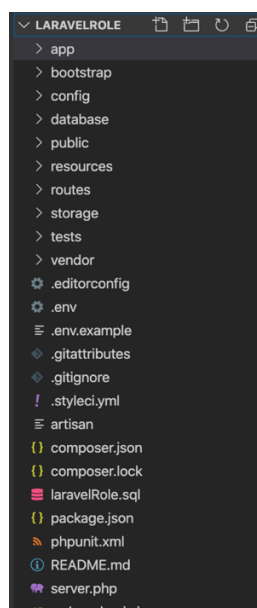
Laravel utilise **Composer** comme gestionnaire de dépendances de PHP. Il permet également de créer des projets Laravel et de télécharger le framework. Le framework Laravel est d'ailleurs un simple assemblage de plusieurs dizaines de bibliothèques. Il fonctionne sur une ligne de commande **Terminal**. Donc on commence à l'installer d'abord:

```
$ brew update  
$ brew install php  
$ brew install composer
```

Pour créer une application Laravel, il suffit juste de taper cette commande (le nom de projets laravelRole) suivante:

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

Architecture des dossiers de Laravel:



On peut vérifier que tout fonctionne bien avec l'URL <http://localhost/laravelRole/public>. Normalement on doit obtenir cette page très épurée :

Laravel

[DOCS](#)[LARACASTS](#)[NEWS](#)[BLOG](#)[NOVA](#)[FORGE](#)[VAPOR](#)[GITHUB](#)

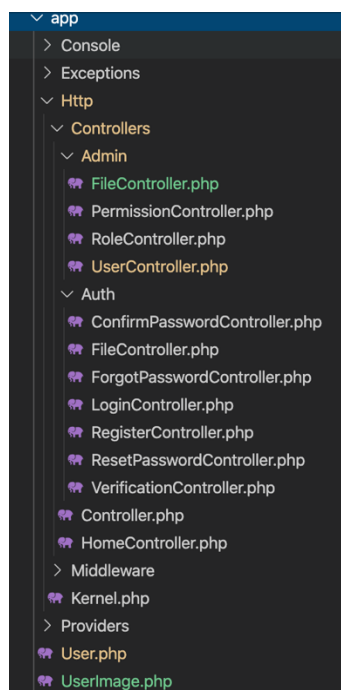
Page d'accueil de Laravel

3. Organisation de Laravel :

Maintenant que j'ai un Laravel qui fonctionne bien, et je vais voir ce qu'il contient.

Dossier app :

Ce dossier contient les éléments essentiels de l'application :



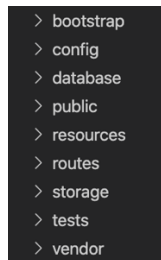
Dossier app

- **Console/Commands** : toutes les commandes en mode console, il y a au départ une commande Inspire qui sert d'exemple,
- **Http** : tout ce qui concerne la communication : contrôleurs, routes, middlewares (il y a quater middlewares de base) et requêtes,

- **Providers** : tous les fournisseurs de services (providers), il y en a déjà 5 au départ. Les providers servent à initialiser les composants.

On trouve également le fichier User.php qui est un modèle qui concerne les utilisateurs pour la base de données.

Autres dossiers :



Autre dossier

Voici une description du contenu des autres dossiers :

- **bootstrap** : scripts d'initialisation de Laravel pour le chargement automatique des classes, la fixation de l'environnement et des chemins, et pour le démarrage de l'application,
- **public** : tout ce qui doit apparaître dans le dossier public du site : images, CSS, scripts...
- **vendor** : tous les composants de Laravel et de ses dépendances,
- **config** : toutes les configurations : application, authentification, cache, base de données, espaces de noms, emails, systèmes de fichier, session...
- **database** : migrations et les populations,
- **resources** : vues, fichiers de langage et assets (par exemple les fichiers de Sass),
- **storage** : données temporaires de l'application : vues compilées, caches, clés de session...
- **tests** : fichiers de tests unitaires.

Fichiers de la racine :

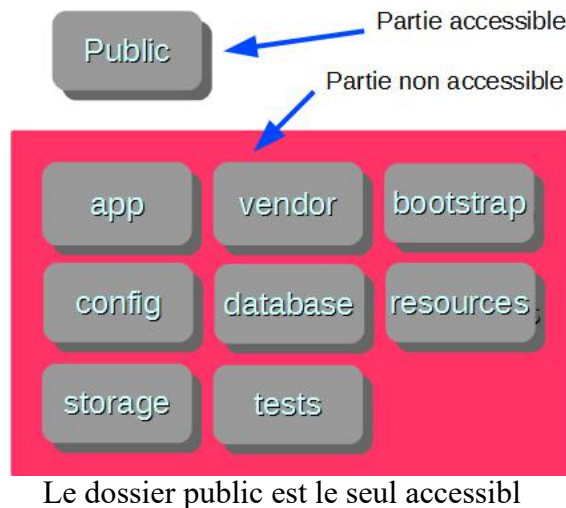
Il y a un certain nombre de fichiers dans la racine dont voici les principaux :

- **artisan** : outil en ligne de Laravel pour des tâches de gestion,
- **composer.json** : fichier de référence de Composer,

- **phpunit.xml** : fichier de configuration de phpunit (pour les tests unitaires),
- **.env** : fichier pour spécifier l'environnement d'exécution.

Accessibilité :

Pour des raisons de sécurité sur le serveur seul le dossier **public** doit être accessible



Gestion des utilisateurs, des rôles, des permission et des fichiers (User Management, Rôle management, Permission management and File) :

Un exemple concrète de comment créer une application de Gestion des utilisateurs, des rôles et des permissions en Laravel 7 :

1. Laravel 7 Installation

On commence à exécuter cette commande pour pouvoir créer un nouveau projet Laravel, on l'appelle **laravelRole** :

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

2. Configuration le fichier .env et creation database

Tout d'abord on va créer une base de donnée vierge qu'on va appeler **laravelRole** avec l'interface de phpmyadmin ou en ligne de commande **Terminal** ;

```
CREATE DATABASE laravelRole;
```

Après on va configurer notre fichier **.env** comme ça :

```

APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:hxbP0dPkXS4Ik+5mQQbxUdDUuwX6Yj59KS8vD7YkTlA=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravelRole
DB_USERNAME=latifroot
DB_PASSWORD=password

```

3. Installation les packages de composer

Maintenant, on doit installer le package Spatie pour ACL(Access Control List), de cette façon on peut utiliser sa méthode, on doit installer aussi le package de collection de formulaire. On tape ces commandes sur le Terminal :

```

$ composer require spatie/laravel-permission
$ composer require laravelcollective/html

```

Maintenant, on ouvre le fichier config/app.php et on ajoute le service provoder :

Config/app.php

```

'providers' => [
    ....
    Spatie\Permission\PermissionServiceProvider::class,
],

```

On peut également personnaliser les modifications sur le package Spatie, donc si on souhaite également apporter des modifications, on peut taper la commande ci-dessous et obtenir le fichier de configuration dans **config/permission.php** et les fichiers de migration.

```

$ php artisan vendor:publish --provider="Spatie\Permission\PermissionServiceProvider"

```

Maintenant, on a qu'à exécuter la commande de migration afin d'ajouter les tables suivantes dans la base de données:

```

$ php artisan migrate

```

Les tables qui sont générées :

users(id, name, email, password, email_verified_at, created_at, updated_at): Si l'utilisateur a un rôle de super-admin alors il a le droit d'accès à toutes les permissions autorisées par ce rôle (user-management, rôle-management ou permission-management), par contre si c'est un simple-user alors il n'aucun droit de super-admin.

roles(id, name, gruad_name, created_at, updated_at): Dans notre cas on a deux rôles principaux (super-admin et simple-user) et on peut ajouter d'autre selon nous besoin.

permissions(id, name, gruad_name, created_at, updated_at): Les permissions autorisées.

rôle_has_permissions(permission id*, rôle id*): Les permissions autorisées à un telle rôle.

4. Création de Model

Dans cette étape on va créer deux modèles un pour user, donc si c'est un nouveau projet, un modèle user existe déjà et maintenant il ne suffit juste d'ajouter un namespace pour pouvoir utiliser ses fonctionnalités et une fonction :

app/User.php :

```
use Spatie\Permission\Traits\HasRoles;
```

```
use Notifiable, HasRoles;
```

```
public function images(){  
    return $this->hasMany(UserImage::class);  
}
```

Et l'autre modèle UserImage pour faire la gestion de fichiers, pour le créer il suffit juste d'exécuter cette commande :

```
$ php artisan make :model UserImage -m
```

Avec l'option m générera le fichier de migration.

On ajoute le code suivant sur le fichier **app/UserImage** :

```
<?php  
  
namespace App;  
  
use Illuminate\Database\Eloquent\Model;  
  
class UserImage extends Model  
{  
    //  
    protected $guarded=[];  
  
    public function users(){  
        return $this->belongsTo(Users::class, 'user_id');  
    }  
}
```

On va modifier la fonction up() dans le fichier de migration suivant:

database/migrations/ 2020_04_08_082403_create_user_images_table.php


```

public function up()
{
    Schema::create('user_images', function (Blueprint $table) {
        $table->id();
        $table->integer('user_id');
        $table->string('path');
        $table->string('documentsName');
        $table->integer('sizeFile');
        $table->timestamps();
    });
}

```

Maintenant il ne reste que à exécuter la commande de migration la nouvelle tables dans la base de donnée :

```
$ php artisan migrate
```

La table qui est générée:

user_image(id, user_id, path, documentsName, sizeFile, created_at, updated_at):

cette table contient les informations de l'image ou file et elle a une clé primaire de user comme clé étrangère pour que l'utilisateur peut avoir zéro ou plusieurs image.

5. Ajouter des Middleware

Le package Spatie fournit son middleware intégré de cette façon, pour que on l'utilise facilement dans le routage des pages.

Maintenant on va ajouter ces middlewares suivants dans le fichier app/http/Kernel.php:

```

....
protected $routeMiddleware = [
    ....
    'role' => \Spatie\Permission\Middlewares\RoleMiddleware::class,
    'permission' => \Spatie\Permission\Middlewares\PermissionMiddleware::class,
    'role_or_permission' => \Spatie\Permission\Middlewares\RoleOrPermissionMiddleware::class,
]
....

```

6. Création de l'Authentification

On va faire quelque étapes pour créer auth dans Laravel 7 (avec les autres versions c'est différent):

Tout d'abord on va installer le package **laravel/ui** comme suivant :

```
$ composer require laravel/ui
```

Après on exécute cette commande pour générer auth :

```

$ php artisan ui bootstrap --auth
$ npm run dev

```

Si npm n'est pas installé il faut suffir juste de taper ces commandes suivantes :

```
$ brew update
$ brew install node
```

On tape cette commande afin de vérifier qu'il est bien installé:

```
$ npm -v
```

7. Création le routage des pages views

Dans le fichier **routes/web.php** on ajoute le code suivant :

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

Auth::routes();

Route::group(['middleware'=>['role:super-admin','auth']],function(){
    Route::resource('admin/permission', 'Admin\\PermissionController');
    Route::resource('admin/role', 'Admin\\RoleController');
    Route::resource('admin/user', 'Admin\\UserController');
});

Route::group(['middleware'=>['auth']],function(){
    Route::view('/admin','admin.dashboard');
    Route::get('admin/file/{userId}', 'Admin\\FileController@index');
    Route::get('/create', 'Admin\\FileController@create');
    Route::get('/show/{userId}', 'Admin\\FileController@show');
    Route::delete('/delete/{userId}', 'Admin\\FileController@destroy');
    Route::post('document_upload/{userId}', 'Admin\\FileController@document_upload');
});

Route::get('/home', 'HomeController@index')->name('home');
```

8. Ajouter des controllers :

Dans cette étape on va créer trois controllers pour module user, module rôle et module de permission dans le dossier **app/http/Controlers**:

Admin/UserControler.php

Admin/RoleControler.php

Admin/PermissionControler.php

Admin/FileControler.php

Le code est disponible dans le github pour pouvoir inspirer :

<https://github.com/Abdellatif-CHALAL/laravelRole>

9. Ajouter des fichiers Blade :

Dans cette étape on va créer tous les fichier view suivants dans le répertoire **resources/views**:

Layouts:

layouts/app.blade.php

Module users :

admin/user/create.blade.php
admin/user/index.blade.php
admin/user/form.blade.php
admin/user/show.blade.php
admin/user/edit.blade.php

Module roles :

admin/role/create.blade.php
admin/role/index.blade.php
admin/role/form.blade.php
admin/role/show.blade.php
admin/role/edit.blade.php

Module permissions :

admin/permission/create.blade.php
admin/permission/index.blade.php
admin/permission/form.blade.php
admin/permission/show.blade.php
admin/permission/edit.blade.php

Module files :

admin/file/create.blade.php
admin/file/index.blade.php
admin/file/show.blade.php

Autres fichier:

dashboard.blade.php

10. Création des fichiers css et js

Pour que notre application soit bien visible, il faut que on ajoute les fichiers de **CSS** et **JS** dans le dossier **public** :

public/css/app.css
public/js/app.js

11. Création seeder pour super-admin user

On va créer un nouveau seeder pour un utilisateur super-admin :

```
$ php artisan db:seed --class=PermissionTableSeeder
```

Database/seeds/PermissionTableSeeder.php

```

<?php
use Illuminate\Database\Seeder;
use App\User;
use Spatie\Permission\Models\Role;
use Spatie\Permission\Models\Permission;

class CreateAdminUserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        $user = User::create([
            'name' => 'CHALAL Abdellatif',
            'email' => 'admin@gmail.com',
            'password' => bcrypt('123456')
        ]);

        $role = Role::create(['name' => 'super-admin']);

        $permissions = Permission::pluck('id','id')->all();

        $role->syncPermissions($permissions);

        $user->assignRole([$role->id]);
    }
}

```

Maintenant, il ne reste que à taper cette commande pour ajouter le nouveau admin user qu'on vient de créer à la base de donnée:

```
$ php artisan db:seed --class=CreateAdminUserSeeder
```

12. Exécution de l'application

Maintenant, on est prêt à exécuter notre application Laravel en tapant ce lien suivant : <http://localhost/laravelRole/public>.

On peut se connecter avec ces informations d'identifications suivantes :

Email : admin@gmail.com

Password : 123456

13. Les pages créées :

User management :

Laravel	Manage Users	Manage Role	Manage permission	Upload Files	CHALAL Abdellatif ▾
---------	--------------	-------------	-------------------	--------------	---------------------

User				
+ Add New		Search... <input type="text"/>		
#	Name	Email	Actions	
1	CHALAL Abdellatif	admin@gmail.com	View	Edit Delete
1	latif	latif@gmail.com	View	Edit Delete

Rôle management :

Laravel		Manage Users	Manage Role	Manage permission	Upload Files	CHALAL Abdellatif ▾
---------	--	--------------	-------------	-------------------	--------------	---------------------

Role		+ Add New	Search...	Q
#	Name	Actions		
1	simple-user	View	Edit	Delete
1	super-admin	View	Edit	Delete


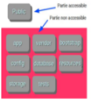
Permissions management :

Laravel		Manage Users	Manage Role	Manage permission	Upload Files	CHALAL Abdellatif ▾
---------	--	--------------	-------------	-------------------	--------------	---------------------

Permission		+ Add New	Search...	Q
#	Name	Actions		
1	create-role	View	Edit	Delete
1	delete-role	View	Edit	Delete
1	edit-role	View	Edit	Delete
1	list-role	View	Edit	Delete
1	user-manager	View	Edit	Delete

Files management :

Laravel		Manage Users	Manage Role	Manage permission	Upload Files	CHALAL Abdellatif ▾
---------	--	--------------	-------------	-------------------	--------------	---------------------

File		+ Add New File	Search...	Q
#	Name	Size	Image	Actions
1	LaravelLogo.png	54365 B		View Delete
1	img80.jpg	21127 B		View Delete

14. Les ressources inspirées :

<https://github.com/webdevmatics/webcasts>
<https://github.com/webdevmatics/ecom>
<https://github.com/Abdullahmasood553/Insert-multiple-image-using-dropzone-with-ajax>
<https://www.itsolutionstuff.com/post/laravel-6-user-roles-and-permissions-from-scratch-laravel-6-aclexample.html>

15. Conclusion :

Laravel est un framework PHP qui propose des outils pour construire une application web.

Dans le cadre de ce projet, j'ai construit une application de gestion des utilisateurs, des rôles et des permissions.

En conclusion, je dois avouer que rétrospectivement je suis satisfait de ce projet, j'ai presque atteint des nouveaux objectifs.

En effet, ce mini-projet m'a permis de comprendre mieux l'architecture de Laravel, comment mettre en pratique le model MVC(Model-View-Controller) et la maîtrise de la manœuvre de la commande php artisan, qui permet de générer le code de manière automatique.