TP : CRÉATION D'UNE MINI APPLICATION A TITRE DE COMPRÉHENSION

Création d'une plateforme de gestion des étudiants de la filière informatique.

Fonctionnalités :

- Ajout d'un étudiant
- Modification d'un étudiant
- Listing d'un étudiant
- Suppression d'un étudiant
- Attribution de note a un étudiant
- Récapitulatif des notes d'un étudiant

Durant ce TP, nous verrons comment :

- Mieux utiliser le routing
- Mieux structurer son code
- Comment créer et utiliser les contrôleurs
- Comment créer et utiliser les migrations et modèles
- Comment mettre sur pied un système d'authentification
- Comment Configurer les variables d'environnement Laravel
- Et bien d'autres aspects et astuces qui vous aiderons a évoluer de manière autonome
- Etc.…

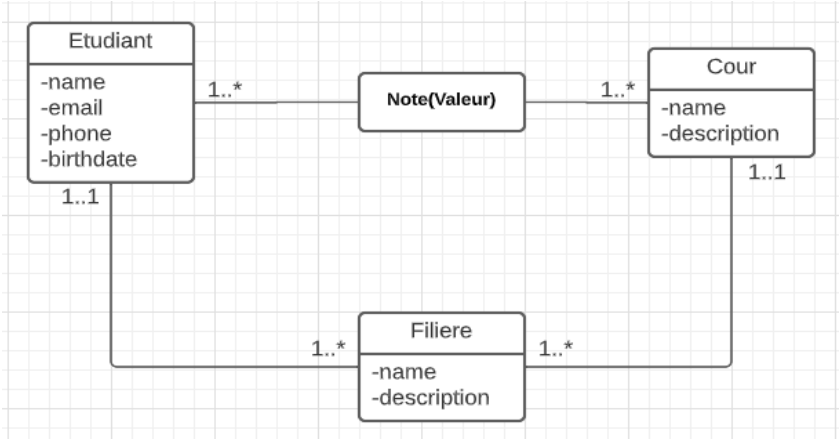
Objectifs du TP :

A la fin de ce TP, vous devez être capable de :

- Expliquer la structure d'un projet Laravel
- Créer une application Laravel
- Créer un site web en utilisant Laravel
- Ajouter des dépendances a votre projet Laravel
- Connaitre le rôle de chaque outil utiliser pour la mise ne place d'un projet Laravel (Composer, serveur de base de données, etc…)
- Mettre sur pied un système d'authentification à partir de Laravel
- Mettre sur pied un système de pagination.
- Être capable de réaliser votre projet de groupe en utilisant Laravel

TP: RESOLUTION

Model Conceptuel dedonnées :

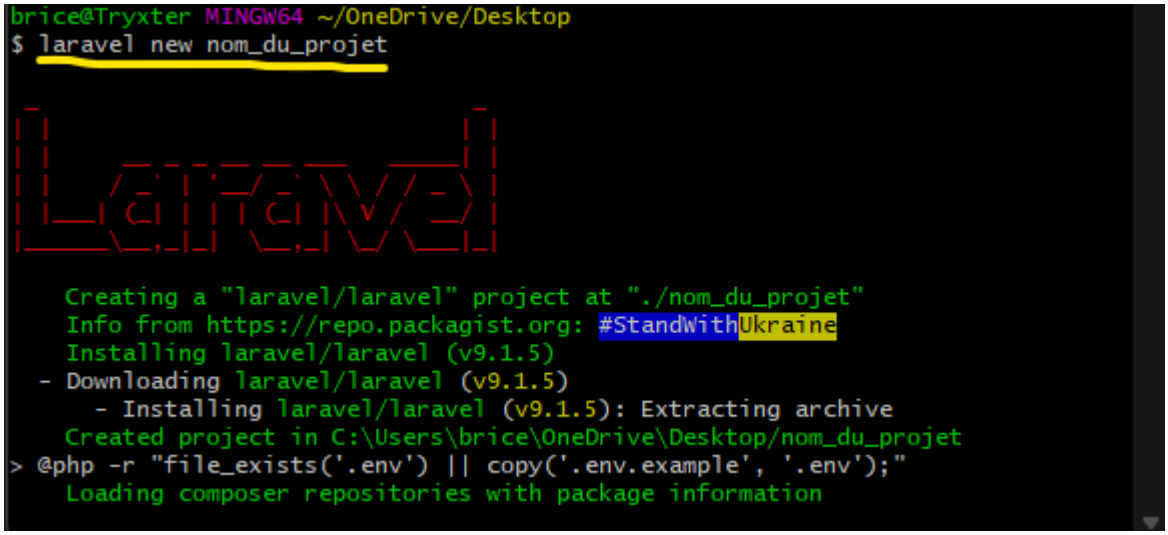


Pour la résolution de notre TP; vous devez avoir installé les outils vus dans la section 2 de notre cour. Si tel n'est pas encore le cas, veuillez l'installer immédiatement. Une fois terminé, vus êtes prêt à commencer. Nous résoudrons notre TP en plusieurs petite phase afin que ce soit plus clair et compréhensible pour tout le monde.

Création d'un projet Laravel :

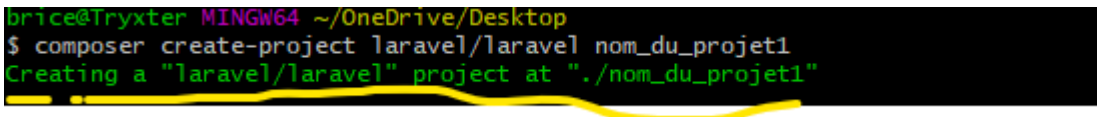
Pour faire ce TP, vous devez avoir créé un projet Laravel en utilisant la ligne de commande suivante :

laravel new nom_du_projet

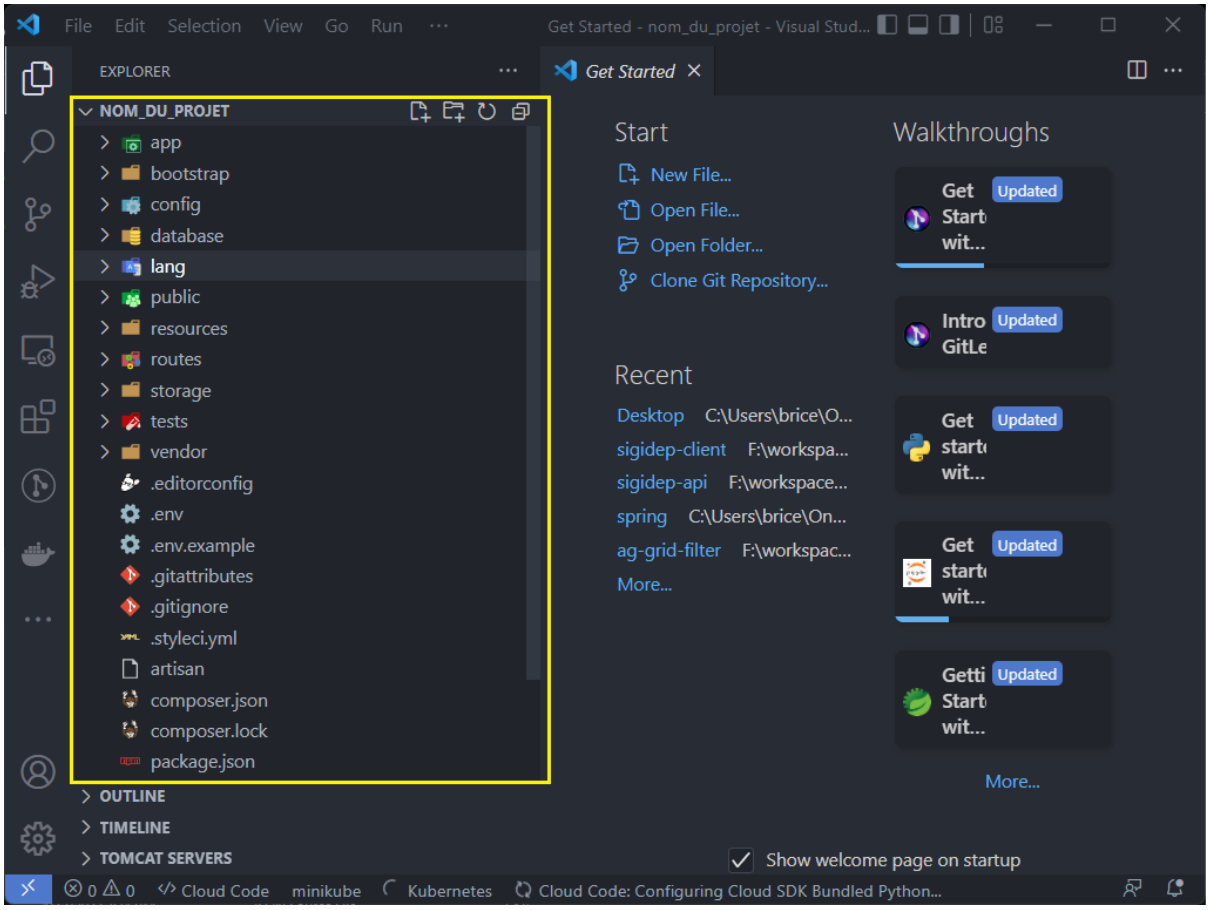


Mais cette commande n'est valable que si vous avez installé Laravel de manière globale dans composer. Sinon, utilisé plutôt

composer create-project laravel/laravel nom_du_projet



une fois terminé, ouvrez votre projet dans une IDE. Si c'est vscode, vous devrez avoir ceci :

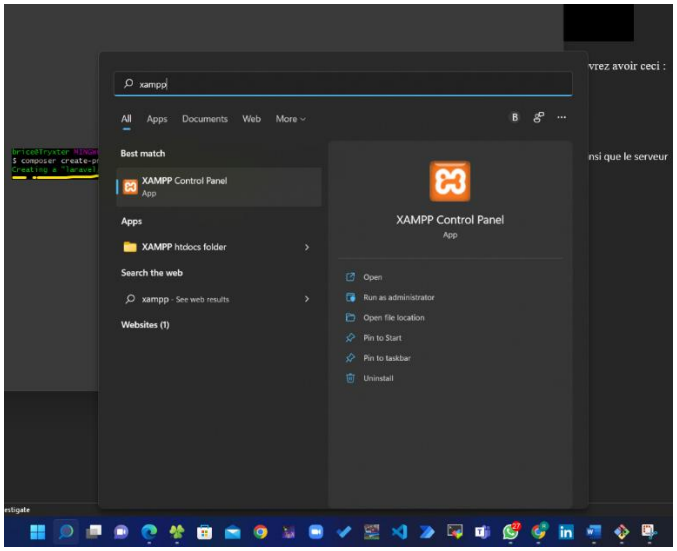


NB : pour créer un projet Laravel, il faut être connecté à internet

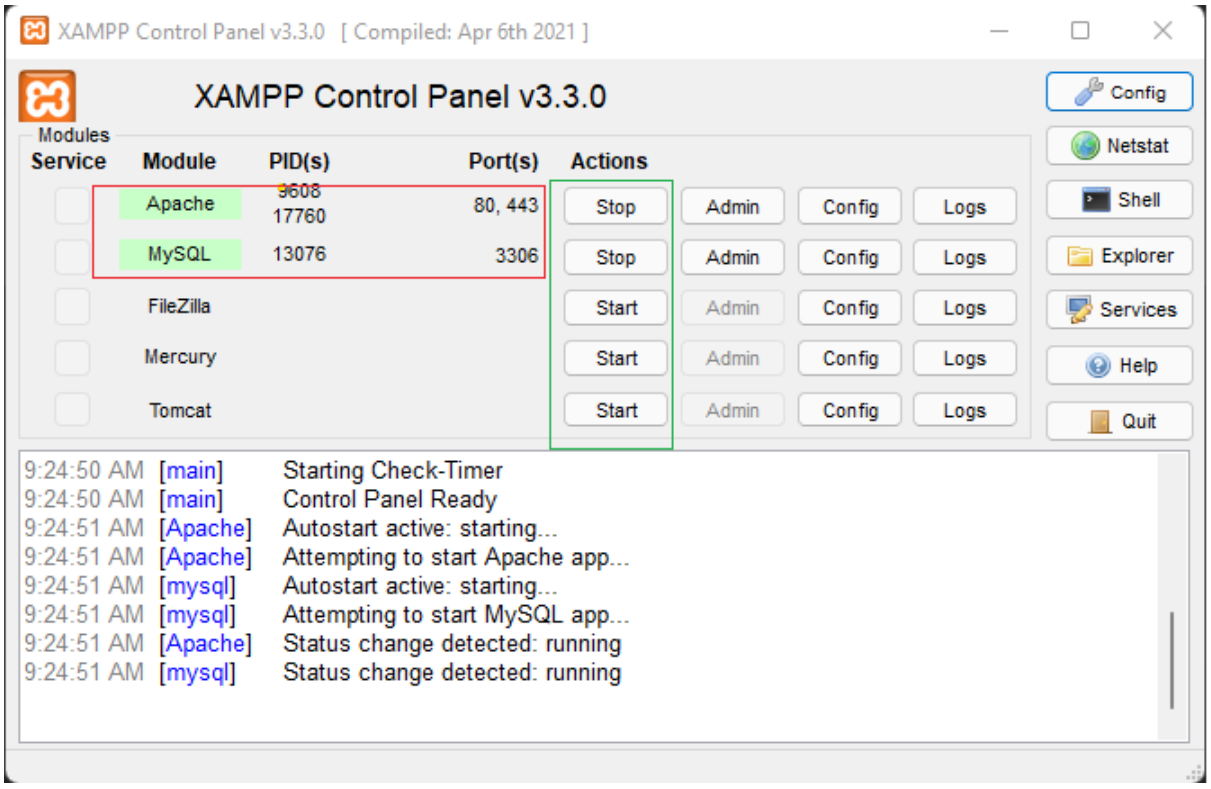
Créer notre Base de données :

Pour ceux ayant utilisé Xampp, démarrer votre serveur de base donnée ainsi que le serveur apache :

- Chercher et ouvrir xampp

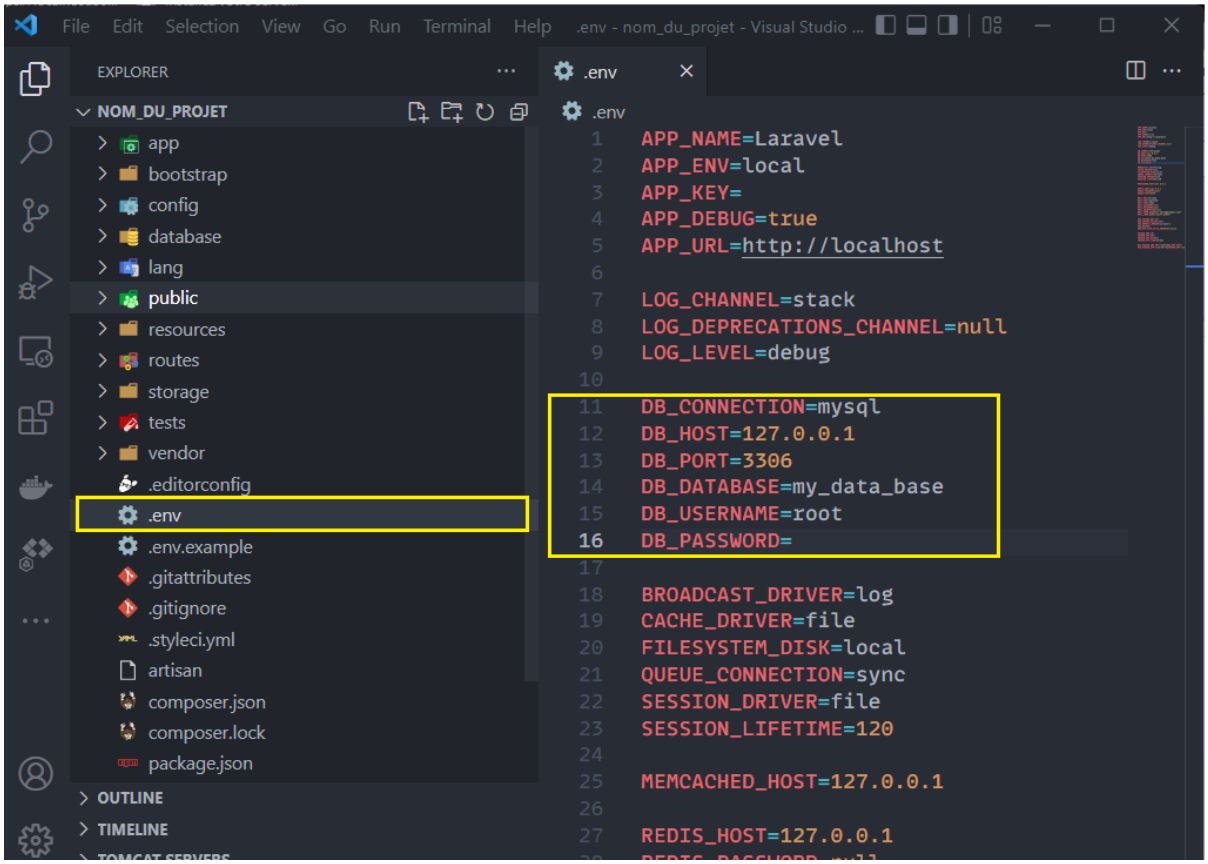


- Démarrer les différents serveurs en cliquant sur la touche start : vous obtiendrez ceci :



- Il ne vous reste qu'à créer votre base de données en ouvrant le menu **admin** qui se situe devant Apache. Une fois sur la page qui s'ouvrira, vous n'aurez qu'a cliquez sur phpMyAdmin >> nouvelle base de données >> renseigner le nom de la base de données.

Configurez votre Base de données dans votre projet Laravel :



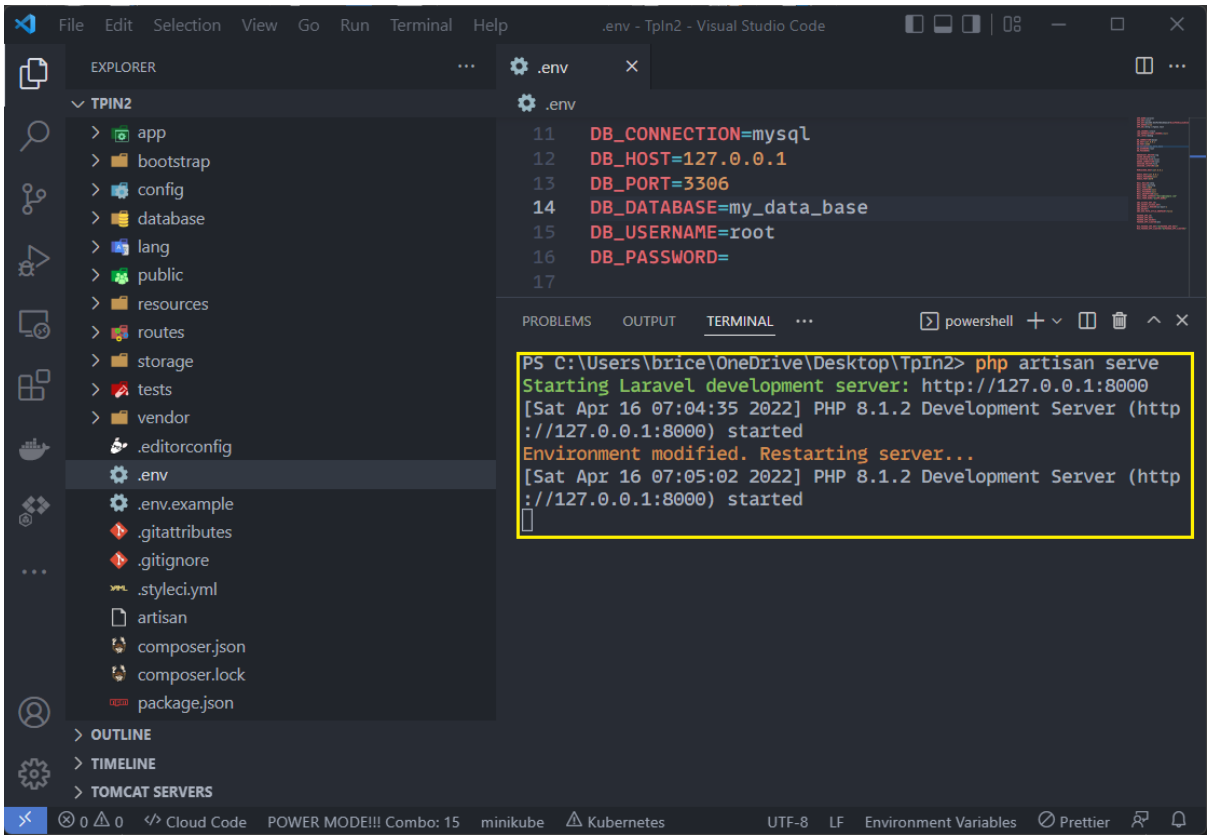
Pour se faire, ouvrir le fichier **.env** a la racine du projet, puis modifier le block encadrer en jaune.

```
DB_CONNECTION=mysql    //à ne pas modifier car nous utilisons mysql
DB_HOST=127.0.0.1      //à ne pas toucher également
DB_PORT=3306           //à ne pas toucher
DB_DATABASE=my_data_base    // nom de votre base de données crée
DB_USERNAME=root  // nom d'utilisateur de la base de donnée (par défaut root)
DB_PASSWORD=  // mot de passe de la base de donnée (par défaut vide)
```

Exécuter votre projet Laravel:

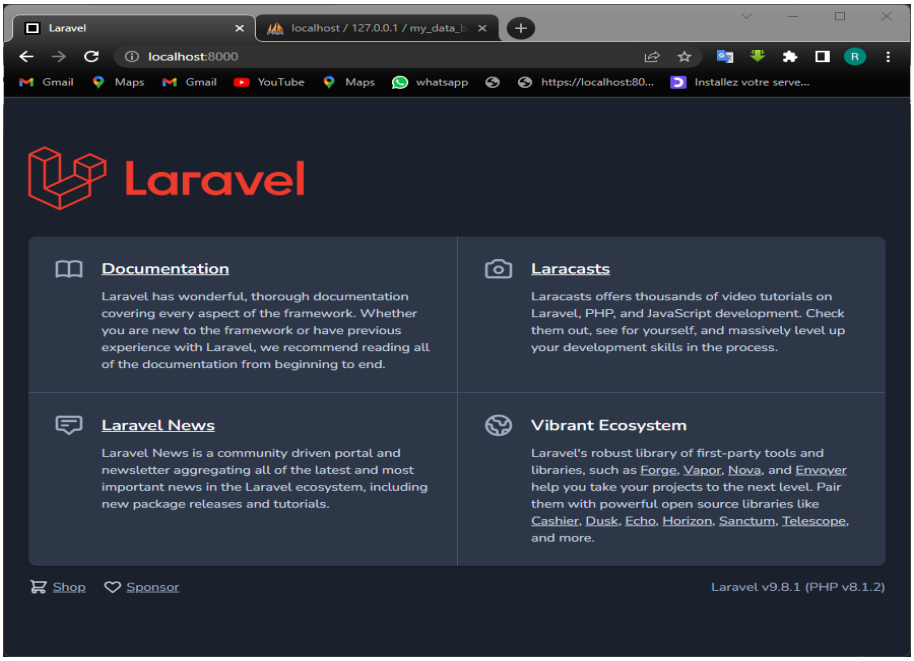
Pour exécuter votre projet Laravel utilisé la commande suivante :

php artisan serve



Puis ouvrez le navigateur et entrée l'adresse http suivante :

http://localhost:8000



Débuter avec le code.

Pour la bonne réalisation de notre TP, nous allons devoir utiliser les composants suivants :

- Les migrations : *php artisan make:migration create_nomdelamigration_table*
- Les Models : *php artisan make:model NomDuModel*
- Les Controlleurs : *php artisan make:controller NomDuControlleur*
- Les Routes : nous aurons uniquement à éditer le fichier *routes/web.php*
- Les Vues : nous allons tout simplement les créées dans le dossier *ressources*
- Les Factories : *php artisan make:factory NomDuFactory*
- Nous allons configurer un provider pour le design de notre pagination
- Installer une dépendance pour la gestion de l'authentification
- Intégrer Bootstrap pour le design de nos pages web

Notre application étant une application intégrant une base de données, nous devons créer notre base de données et la connecter avec Laravel.

NB : Cette étape est vu plus haut.

Par la suite, nous devons créer nos entités de la base de données grâce aux migrations :

Filiere : *php artisan make:migration create_filieres_table*

Ajouter ceci au contenu de la fonction **up** :

```
            $table->id();
            $table->string('name')->unique();
            $table->text('description')->nullable();
            $table->timestamps();
```

Etudiant : *php artisan make:migration create_students_table*

Ajouter ceci au contenu de la fonction **up** :

```
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->string('phone');
            $table->date('birthdate');
            $table->foreignId('filiere_id')->constrained();
            $table->timestamps();
```

Cour : : *php artisan make:migration create_cours_table*

Ajouter ceci au contenu de la fonction **up** :

```
            $table->id();
            $table->string('name')->unique();
            $table->text('description')->nullable();
            $table->foreignId('filiere_id')->constrained();
            $table->timestamps();
```

Note : : *php artisan make:migration create_notes_table*

Ajouter ceci au contenu de la fonction **up** :

```
            $table->id();

            $table->foreignId('student_id')->constrained();
            $table->foreignId('cour_id')->constrained();
            $table->double('value')->min(0)->max(20);
            $table->timestamps();
```

Une fois nos 4 entités créées, exécuter la commande suivante pour les ajouter dans notre base de données : *php artisan migrate*

⚠️ ⚠️ ⚠️ vous devez avoir votre serveur de base de données démarrer.

Une fois cette étape terminer, vous devez créer vos models.

Filiere : *php artisan make:model Filiere*

Etudiant : *php artisan make:model Student*

Cour : *php artisan make:model Cour*

Note : *php artisan make:model Note*

Remarque : le nom du model est le même que le nom de l'entité(migration) à la seul différence que le model est au singulier et l'entité (migration) au pluriel.

Si vous avez respecter cette remarque, alors votre model est automatique lié à l'entité et possède par défaut tous les attributs de l'entité.

Sinon, veuillez créer l'attribut suivant dans le model avec comme valeur le nom de l'entité qu'il représente :

protected $table = 'students' pour la table students par exemple.

Ensuite, ajouter aussi l'attribut $guarded dans votre model. Vous aurez un visuel comme suit :

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Student extends Model
{
    use HasFactory;
    protected $guarded = [];
}
```

Si vous avec respecter la remarque précédente

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Student extends Model
{
    use HasFactory;
    protected $guarded = [];
    protected $table = 'students';
}
```

Sinon.

Une fois terminer avec le model, nous devons aussi créer les controlleurs :

Étudiant : *php artisan make:controller StudentController*

Filière : *php artisan make:controller FiliereController*

Cour : *php artisan make:controller CourStudentController*

Note : *php artisan make:controller NoteController*

Continuons en créant nos diffrente vue :

Pour se faire, nous organiserons le code dans le dossier ressources/views

1. Créer un dossier *layouts* dans *views*

   Ce dossier va contenir la coquille (structure de base de nos pages web ainsi que les menus et autre) de notre projet.

   Dans ce dossier, créer les fichier *app.blade.php* et *menu.blade.php.*

   Dans le fichier *app.blade.php,* ajouter ce contenu :

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=100%, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Gestion des étudiants</title>
    <link rel="stylesheet" href="{{
asset('assets/bootstrap/css/bootstrap.min.css') }}">

</head>

<body class="d-flex h-100 bg-dark">
    <div class="cover-container d-flex w-100 h-100 p-3 mx-auto flex-column"
id="main">

        <header class="mb-auto">
            <div class="px-3 py-2 bg-dark text-white">
                <div class="container">
                    <div class="d-flex flex-wrap align-items-center justify-
content-center justify-content-lg-start">
                        <a href="/"
                            class="d-flex align-items-center my-2 my-lg-0 me-
lg-auto text-white text-decoration-none">
                            <svg class="bi me-2" width="40" height="32"
role="img" aria-label="Bootstrap">
                                <use xlink:href="#bootstrap" />
                            </svg>
                        </a>
                        @include('layouts.menu')
                    </div>
                </div>
            </div>
        </header>
        @yield('content')
    </div>
</body>

</html>
```

Dans *menu.blade.php* Ajouter ce contenu :

```
<ul class="nav col-12 col-lg-auto my-2 justify-content-center my-md-0 text-
small">
    <li>
        <a href="#" class="nav-link {{ Request::is('dashboard*') ? 'text-
secondary' : 'text-white' }}"">
            Accueil
        </a>
    </li>
    <li>
        <a href=" {{ route('student.index') }}"
            class="nav-link {{ Request::is('student*') ? 'text-secondary' :
'text-white' }}">
            Liste etudiant
        </a>
    </li>
    <li>
        <a href="{{ route('cour.index') }}"
            class="nav-link {{ Request::is('cour*') ? 'text-secondary' :
'text-white' }}"">
            Cours
        </a>
    </li>
    <li>
        <a href=" #" class="nav-link text-white">
            notes
        </a>
    </li>
</ul>
```

Une fois terminer, aller dans le fichier routes/web.php et remplacer son contenu par :

```
<?php

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!
|
*/
```

```php
Route::get('/', function () {
    return view('layouts.app');
});
```

Ensuite, visualiser votre projet en démarrant le serveur Laravel grâce à la commande :

php artisan serve