Étape 1 : Installation de Laravel 9

Avant toute chose, préparer votre environnement de travail, le projet Laravel sera dans un dossier et nous allons utiliser la ligne de commande pour les différentes opérations.

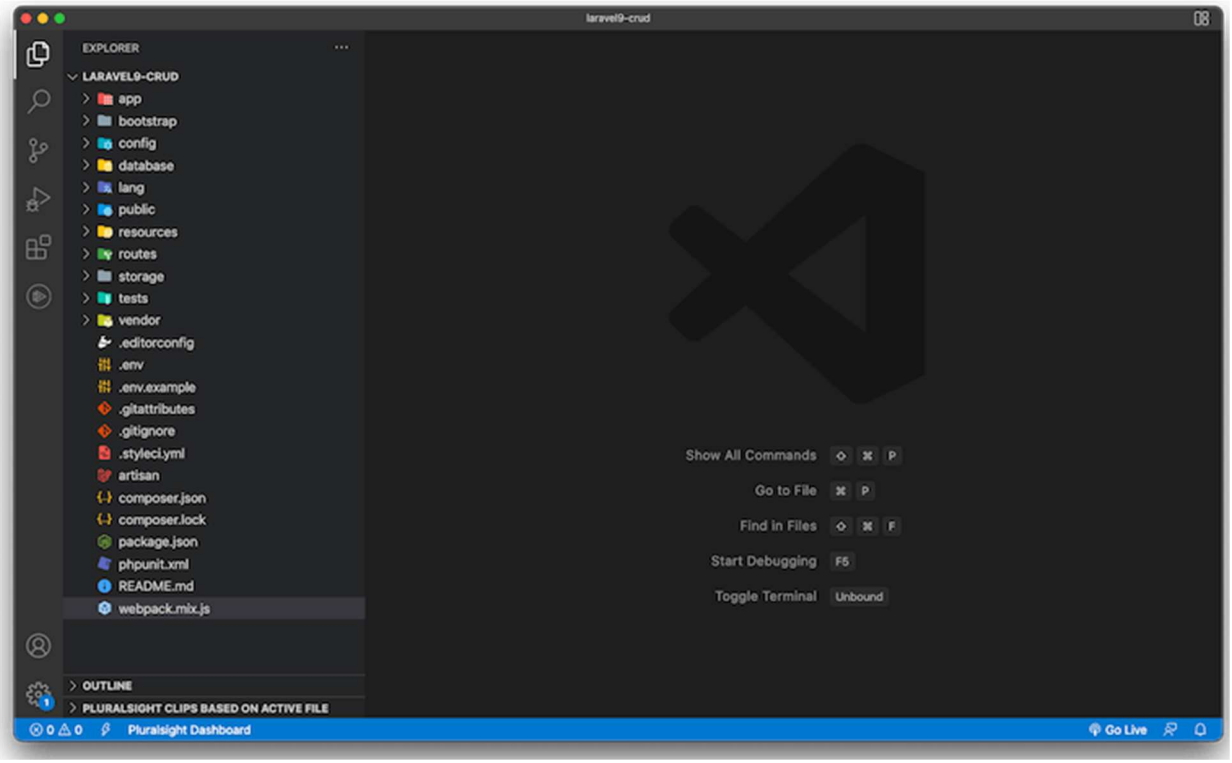
Donc placer votre terminal (invite de commande) dans le dossier de vos projets par exemple : www ou htdocs, et taper la commande suivante pour créer une nouvelle application Laravel 9 dans votre machine.

```
composer create-project --prefer-dist laravel/laravel Laravel9-crud
```

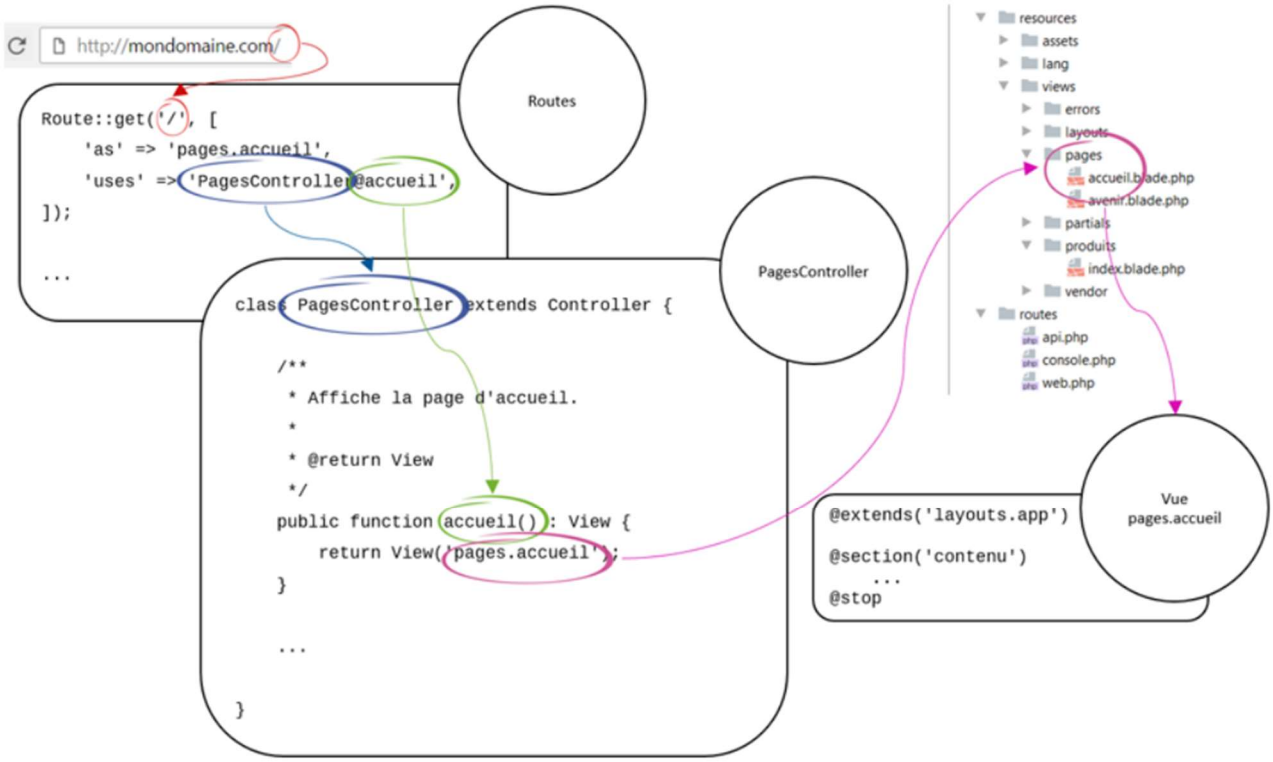
Un dossier avec le nom **Laravel9-crud** sera créer et contenant votre projet prêt à démarrer.

Apres installation de Laravel 9, pour mieux continuer ce tutoriel et bien gérer votre environnement de travail, je vous conseille d'ouvrir tout votre projet dans Vs code, afin de naviguer facilement dans les fichiers du projet.

Voici un aperçu de mon éditeur avec le projet :



Et le schéma MVC :



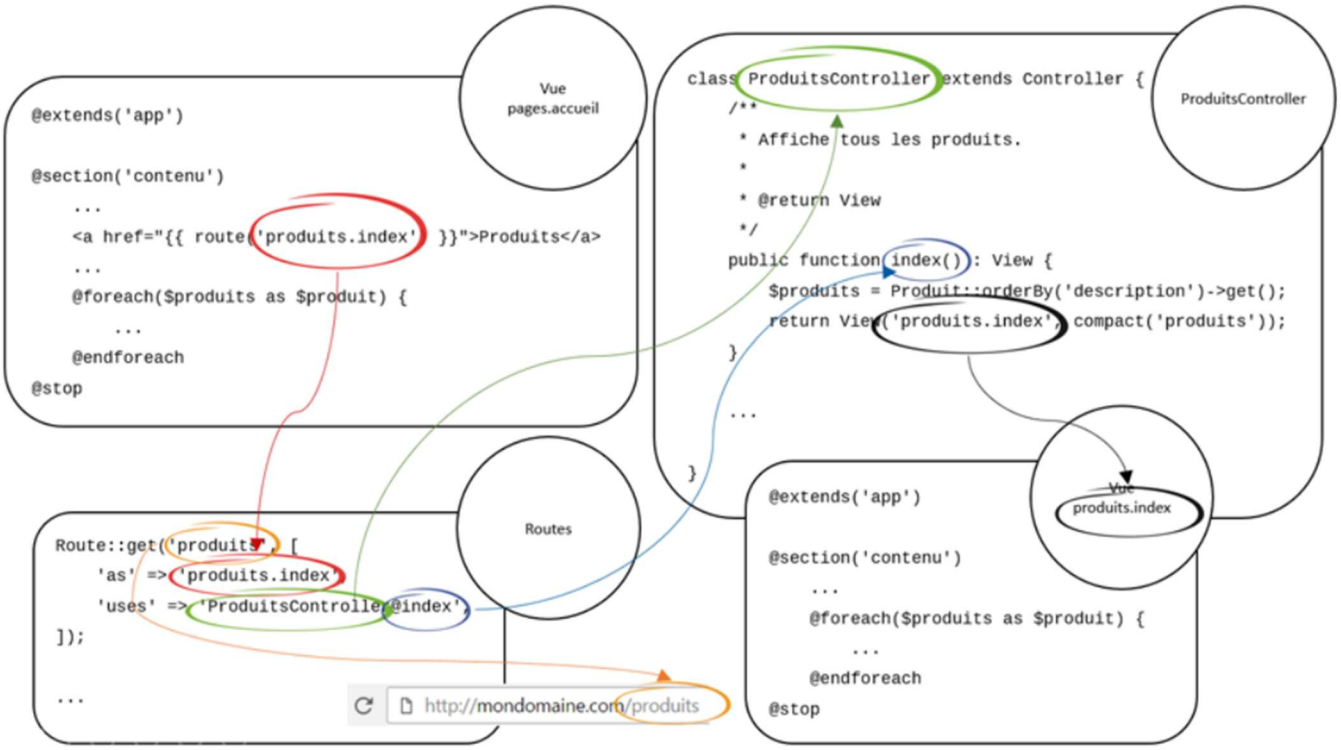
Si le fonctionnement d'un framework MVC est nouveau pour vous, il est possible que vous ayez du mal à vous y retrouver. Pour vous aider à y voir clair, voici un schéma qui montre le cheminement du traitement à partir d'un URL jusqu'à l'affichage d'une page Web.

- On y voit, en rouge, que Laravel commence par rechercher une route qui correspond à l'URL (ici : « / »).
- On y voit en bleu qu'ensuite, Laravel va appeler le contrôleur indiqué dans le « uses » de la route (ici : PagesController). Plus précisément, en vert, on voit le nom de la méthode d'action à exécuter dans ce contrôler (ici : accueil).
- En rose, on voit que Laravel va afficher la vue mentionnée dans la méthode d'action. La vue doit être placée dans un sousdossier de ressources/views (ici : pages) et porter le nom spécifié (ici : accueil).
- C'est ainsi qu'on obtient à l'écran le visuel de l'URL demandé.

Remarquez que dans ce cheminement, le nom de la route, tel qu'indiqué dans le « as » de la route, n'est pas utilisé. Il trouvera son utilité dans un autre cheminement.

Voici ci-dessus un autre exemple. Cette fois, le cheminement partira de la vue pages.accueil affichée précédemment. Prenons le cas où l'usager clique sur le lien « Produits » affiché à l'écran.

- On voit en rouge que Laravel commence par rechercher une route dont le « as » correspond au nom indiqué (ici : produits.index). Si jamais cette route n'existait pas, on obtiendrait une erreur.
- En orange, on voit que Laravel se charge de changer l'URL pour celui demandé (ici : il ajoute « produits » à l'URL de base).
- Parallèlement à cela, en vert, Laravel exécute une méthode d'action du contrôleur ProduitsController. Plus précisément, en bleu, il exécute la méthode index.
- Après avoir retrouvé les données à partir du modèle dans la variable $produits, il charge la vue produits.index. Autrement dit, il recherche index.blade.php dans le dossier ressources/views/produits.



Étape 2 : Configuration de la base de données

Maintenant il est temps de configurer notre base de données. Avant tout, rendez-vous dans votre espace PhpMyadmin et créer une base de données vide, puis ouvrez le fichier .env de votre projet et changer le nom de votre base de données, nom d'utilisateur et le mot de.

Le nom d'utilisateur et le mot de passe seront différents pour les vôtres en fonction des informations d'identification de votre base de données.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel9_crud
DB_USERNAME=root
DB_PASSWORD=root
```

Étape 3 : Création de la table en utilisant la migration

Dans ce tutoriel nous allons créer une gestion des célébrités. Pour commencer, créons le Model et la migration de notre table.

Ouvrez le terminal et taper la commande ci-dessous pour créer les fichiers du Model et de la migration.

```
php artisan make:model Personnage --migration
```

Après la commande, deux fichiers seront créés :

- Modèle Personnage.php
  databases/migrations/2022_X_X_XXXXX_create_personnages_table.php
- migration xxxxx_create_personnage_table.php
  app/Models/Personnage.php

Ouvrez donc le fichier de migration et ajuster les codes comme suis :

```php
<?php


use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;


return new class extends Migration
{

    /**
     * Run the migrations.
     *
     * @return void
     */

    public function up()
    {

        Schema::create('personnages', function (Blueprint $table) {

            $table->id();
            $table->string('nom');
            $table->text('detail');
            $table->string('company');
            $table->integer('fortune');
            $table->timestamps();

        });

    }


    /**
     * Reverse the migrations.
     *
     * @return void
     */

    public function down()
```

```
    {
        Schema::dropIfExists('personnages');
    }
};
```

Vous aurez remarqué que Laravel 9 utilise ici les **migrations anonymes** par defaut.

Puis taper la commande suivante pour que la table soit créer dans la base de données :

```
php artisan migrate
```

Étape 4 : Création du contrôleur

Taper la commande suivante pour créer un contrôleur :

```
php artisan make:controller PersonnageController
```

Ouvrez le contrôleur dans **app/Http/Controllers/PersonnageController.php** et ajuster les codes ci-après :

```
<?php

namespace App\Http\Controllers;

use App\Models\Personnage;
use Illuminate\Http\Request;

class PersonnageController extends Controller
{
    /**
     * Affiche la liste des personnages
     */
    public function index()
    {
        $personnages = Personnage::all();
        return view('personnage.index', compact('personnages'));
    }

    /**
     * return le formulaire de creation d'un personange
     */
    public function create()
    {
        return view('personnage.create');
    }
```

```php
    /**
     * Enregistre un nouveau personnage dans la base de données
     */
    public function store(Request $request)
    {

        $request->validate([
            'nom'=>'required',
            'detail'=> 'required',
            'company' => 'required',
            'fortune' => 'required'
        ]);


        $personnage = new Personnage([
            'nom' => $request->get('nom'),
            'detail' => $request->get('detail'),
            'company' => $request->get('company'),
            'fortune' => $request->get('fortune')
        ]);


        $personnage->save();
        return redirect('/')->with('success', 'Personnage Ajouté avec succès');

    }


    /**
     * Affiche les détails d'un personnage spécifique
     */

    public function show($id)
    {

        $personnage = Personnage::findOrFail($id);
        return view('personnage.show', compact('personnage'));

    }


    /**
     * Return le formulaire de modification
     */

    public function edit($id)
    {

        $personnage = Personnage::findOrFail($id);

        return view('personnage.edit', compact('personnage'));

    }


    /**
     * Enregistre la modification dans la base de données
     */
    public function update(Request $request, $id)
    {

        $request->validate([
```

```
            'nom'=>'required',
            'detail'=> 'required',
            'company' => 'required',
            'fortune' => 'required'

        ]);

        $personnage = Personnage::findOrFail($id);
        $personnage->nom = $request->get('nom');
        $personnage->detail = $request->get('detail');
        $personnage->company = $request->get('company');
        $personnage->fortune = $request->get('fortune');


        $personnage->update();

        return redirect('/')->with('success', 'Personnage Modifié avec succès');

    }


    /**
     * Supprime le personnage dans la base de données
     */
    public function destroy($id)
    {

        $personnage = Personnage::findOrFail($id);
        $personnage->delete();

        return redirect('/')->with('success', 'Personnage Modifié avec succès');

    }

}
```

Puisque notre application a des opérations crud de base, nous pouvons utiliser le contrôleur de ressources pour ce projet afin de créer un contrôleur qui contient les opérations CRUD par défaut :

php artisan make:controller PersonnageController –resource

Notez ici que j'ai ajouté le flag –resource, qui définira six méthodes à l'intérieur du PersonnageController, à savoir

- Index: (La méthode index() est utilisée pour afficher une liste de personnages).
- Create: (La méthode create() affichera le formulaire ou la vue pour créer un personnage).
- Store: (La méthode store() est utilisée pour insérer une voiture dans la base de données. Remarque : la méthode create submit les données du formulaire à la méthode store()).
- Show: (La méthode show() affichera un personnage spécifié).
- Edit: (La méthode edit() affichera le formulaire pour éditer un personnage. Le formulaire sera rempli avec les données du personnage existante).
- Update: (La méthode update() est utilisée pour mettre à jour un personnage dans la base de données. Remarque: la méthode edit() submit les données du formulaire à la méthode update()).
- Destroy: (La méthode destroy() est utilisée pour supprimer le personnage spécifié).

Vous pouvez voir que le fichier contient des opérations CRUD sous la forme de différentes fonctions. Nous utiliserons ces fonctions, une par une, pour créer des opérations specifiques. Le flag –resource appellera la

méthode internal resource() par Laravel pour générer les routes suivantes. Vous pouvez consulter la liste des routes à l'aide de la commande suivante : php artisan route:list

N'oubliez surtout pas d'ajouter ceci dans le model **app/Models/Personnage.php** :

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Personnage extends Model
{
    use HasFactory;

    protected $fillable = [
        'nom', 'company', 'fortune', 'detail'
    ];
}
```

Étape 5 : création des routes

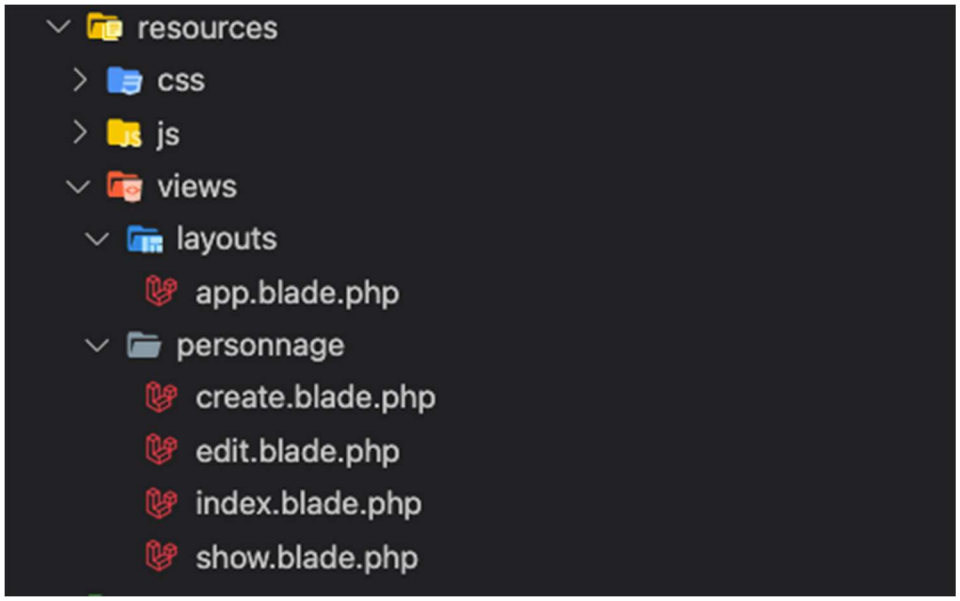
Nous allons créer des routes en utilisant le groupe des routes avec la fonction controller de Laravel 9 :

```php
<?php


use App\Http\Controllers\PersonnageController;
use Illuminate\Support\Facades\Route;


Route::controller(PersonnageController::class)->group(function () {

    Route::get('/', 'index');
    Route::get('/personnage/create', 'create');
    Route::get('/personnage/{id}', 'show');
    Route::get('/personnage/{id}/edit', 'edit');


    Route::post('/personnage', 'store');
    Route::patch('/personnage/{id}', 'update');
    Route::delete('/personnage/{id}', 'destroy');

});
```

Étape 6 : Création des interfaces avec Blade template

Finalement nous passons à la création des différentes pages dans le dossier **'ressources/views'**.

Voici les fichiers que vous allez créer :



layouts/app.blade.php

```
<!DOCTYPE html>
<html lang="fr">
   <head>

      <title>Tutoriel Laravel 9 CRUD pour débutant : insérer, Lire, modifier et supprimer</title>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
   </head>

   <body>

      <div class="container mt-5">
         @yield('content')
      </div>

   </body>

</html>
```

personnage/index.blade.php

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

@section('content')

   <div class="row">

      <div class="col-lg-11">

         <h2>Tutoriel Laravel 9 CRUD</h2>

      </div>
```

```
    <div class="col-lg-1">
        <a class="btn btn-success" href="{{ url('personnage/create') }}">Ajouter</a>
    </div>

</div>


@if ($message = Session::get('success'))

    <div class="alert alert-success">
        <p>{{ $message }}</p>
    </div>

@endif


<table class="table table-bordered">

    <tr>

        <th>No</th>
        <th>Nom</th>
        <th>Détail</th>
        <th>Company</th>
        <th>fortune</th>
        <th>Actions</th>

    </tr>

    @foreach ($personnages as $index => $personnage)

        <tr>
            <td>{{ $index }}</td>
            <td>{{ $personnage->nom }}</td>
            <td>{{ $personnage->detail }}</td>
            <td>{{ $personnage->company }}</td>
            <td>{{ $personnage->fortune }}</td>
            <td>

                <form action="{{ url('personnage/'. $personnage->id) }}" method="POST">
                    @csrf
                    @method('DELETE')

                    <a class="btn btn-info" href="{{ url('personnage/'. $personnage->id) }}">Voir</a>
                    <a class="btn btn-primary" href="{{ url('personnage/'. $personnage->id .'/edit') }}">Modifier</a>

                    <button type="submit" class="btn btn-danger">Supprimer</button>

                </form>
            </td>

        </tr>

    @endforeach
</table>

@endsection
```

personnage/create.blade.php

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


@section('content')

    <h1>Ajouter un personnage</h1>


    @if ($errors->any())

        <div class="alert alert-danger">

            <ul>
                @foreach ($errors->all() as $error)
                    <li>{{ $error }}</li>
                @endforeach

            </ul>

        </div>

    @endif

    <form action="{{ url('personnage') }}" method="POST">
        @csrf

        <div class="form-group mb-3">
            <label for="nom">Nom :</label>
            <input type="text" class="form-control" id="nom" placeholder="Entrez un nom" name="nom">
        </div>

        <div class="form-group mb-3">

            <label for="company">Company:</label>
            <input type="text" class="form-control" id="company" placeholder="Company" name="company">

        </div>

        <div class="form-group mb-3">
            <label for="fortune">Fortune ($):</label>
            <input type="number" class="form-control" id="fortune" placeholder="fortune" name="fortune">
        </div>

        <div class="form-group mb-3">
            <label for="detail">Détail:</label>
            <textarea class="form-control" id="detail" name="detail" rows="10" placeholder="Détail"></textarea>
        </div>

        <button type="submit" class="btn btn-primary">Enregister</button>

    </form>

@endsection
```

personnage/show.blade.php

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


@section('content')

    <h1>Tutoriel Laravel 9 CRUD</h1>


    <table class="table table-bordered">
```

```
        <tr>
            <th>Nom:</th>
            <td>{{ $personnage->nom }}</td>
        </tr>

        <tr>

            <th>Company:</th>
            <td>{{ $personnage->company }}</td>

        </tr>

        <tr>

            <th>détail:</th>
            <td>{{ $personnage->detail }}</td>

        </tr>

        <tr>

            <th>Fortune:</th>
            <td>$ {{ $personnage->fortune }}</td>

        </tr>

    </table>

@endsection
```

personnage/edit.blade.php

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

@section('content')

    <h1>Modifier Personnage</h1>

    @if ($errors->any())

        <div class="alert alert-danger">

            <ul>
                @foreach ($errors->all() as $error)
                    <li>{{ $error }}</li>
                @endforeach
            </ul>

        </div>

    @endif

    <form method="post" action="{{ url('personnage/'. $personnage->id) }}" >
        @method('PATCH')
        @csrf


        <div class="form-group mb-3">

            <label for="nom">Nom:</label>
            <input type="text" class="form-control" id="nom" placeholder="Entrer Nom" name="nom" value="{{ $personnage->nom }}">

        </div>
```

```
    <div class="form-group mb-3">

        <label for="company">Company:</label>
        <input type="text" class="form-control" id="company" placeholder="Entrer Company" name="company" value="{{
$personnage->company }}">

    </div>

    <div class="form-group mb-3">

        <label for="fortune">Fortune ($):</label>
        <input type="number" class="form-control" id="fortune" placeholder="fortune" name="fortune" value="{{
$personnage->fortune }}">

    </div>

    <div class="form-group mb-3">

        <label for="detail">Détail:</label>
        <textarea class="form-control" id="detail" name="detail" rows="10" placeholder="Détail">{{ $personnage->detail
}}</textarea>

    </div>

    <button type="submit" class="btn btn-primary">Enregistrer</button>

  </form>

@endsection
```

Étape 7 : Lancer l'application

Taper la commande suivante pour lancer l'application :

```
php artisan serve
```

Et aller sur le navigateur taper : **http://127.0.0.1:8000**