

**TP : Framework Laravel**

Nb : Vous devez installer laragon à partir de lien ci-dessous

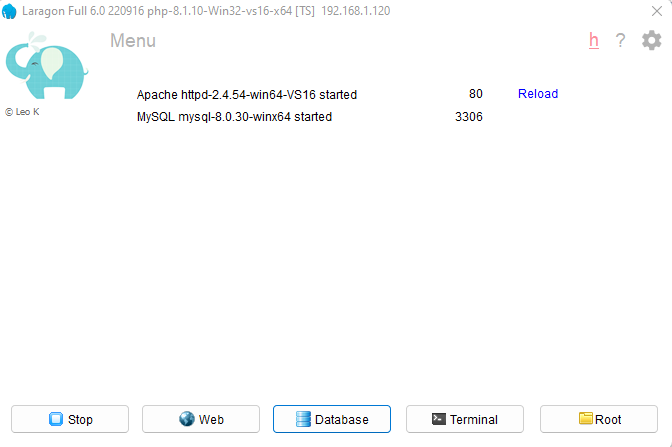
[Download | Laragon - portable, isolated, fast & powerful universal development environment for PHP, Node.js, Python, Java, Go, Ruby.](https://laragon.org/download/index.html)

--------------------------------------------------------------------------------------------------------

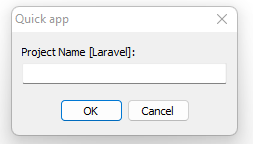
**Exercice1 :** Système de gestion d'un établissement scolaire

**1-**[**Création du projet**](https://www.tresfacile.net/tp-laravel-systeme-de-gestion-dun-etablissement-scolaire-avec-laravel-8/#create-project)

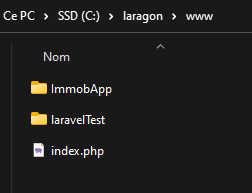
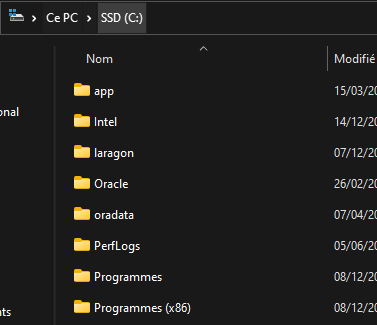
* 1. Pour commencer créer un nouveau projet Laravel 8:

****

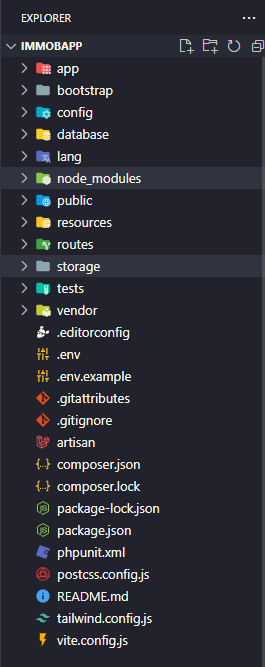
* Cliquer sur créer Quick App



* Taper le nom de votre projet : **Gestion Scolaire**
* Ouvrir votre projet qui existe au dossier **www** dans le dossier laragon qui se trouve dans votre disque C:/

****

* Ouvrir le projet dans visual studio

****

* 1. Configurer ensuite la base de donnée MySQL au niveau du fichier. env.

DB\_CONNECTION=mysql

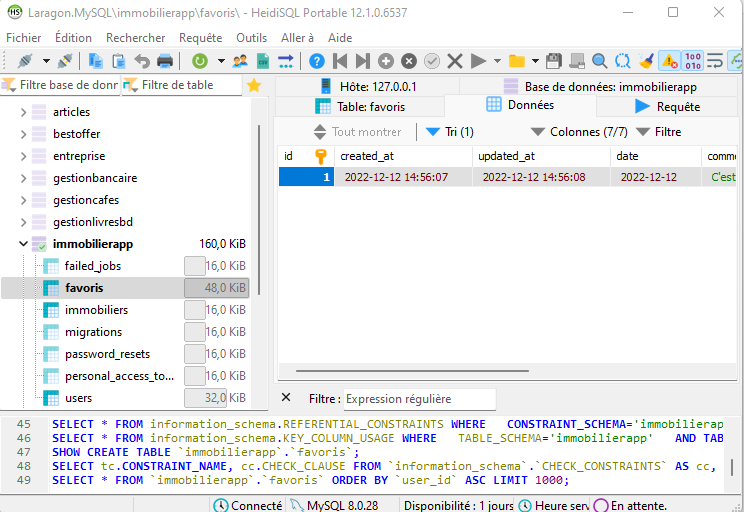
DB\_HOST=127.0.0.1

DB\_PORT=3306

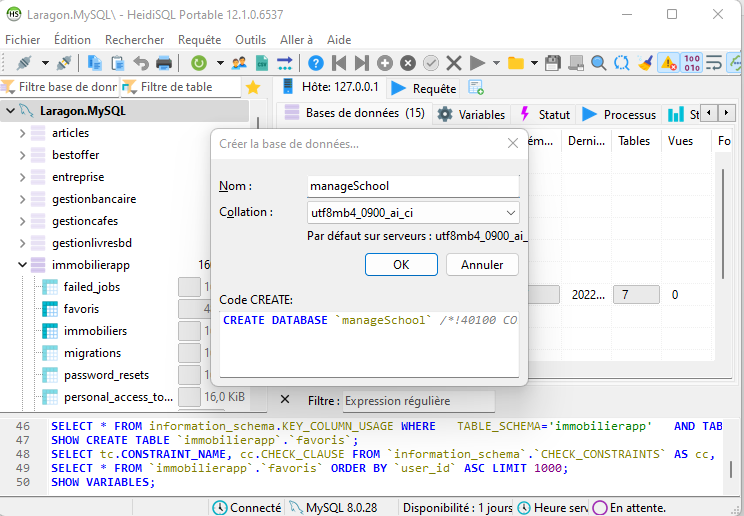
DB\_DATABASE=manageSchool

DB\_USERNAME=root

DB\_PASSWORD=



* Créer une base de donnees de nom **manageSchool**



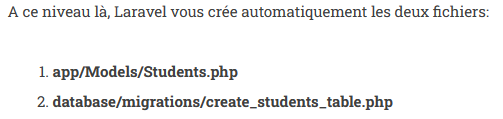
* 1. Effectuer ensuite la migration

**php artisan migrate**

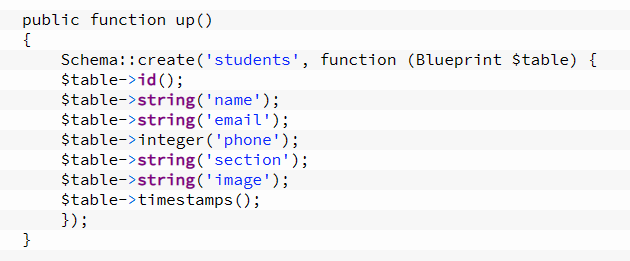
**2-**[**Création du modèle**](https://www.tresfacile.net/tp-laravel-systeme-de-gestion-dun-etablissement-scolaire-avec-laravel-8/#creation-modele)

Afin de pouvoir gérer les étudiant via un **système de CRUD**, nous devons créer un modèle Students:

**php artisan make: model Student –m**

****

Ajouter ensuite les attributs nécessaires à la gestion des étudiants au modèle Students:  
**database/migrations/create\_students\_table.php**

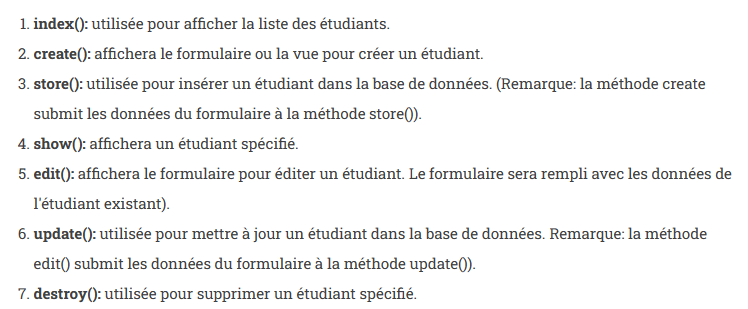
****

**3-Création du contrôleur**

Afin que notre application soit un système de CRUD valide, il faut qu'elle soit dotée d'un contrôleur. Laravel 8 crée le contrôleur automatiquement à l'aide de la commande **make:controller** et si on veut créer les méthodes au sein du contrôleur on ajoute à la fin de la commande: **--resource**:

****

Cela crée automatiquement le fichier **app/Http/Conrollers/StudentsController.php** qui contient une classe nommée **StudentsController**.  
La **commande –resources** qui a été ajoutée à la fin de la commande, avait pour rôle la **création de 7 méthodes** au sein de la classe **StudentsController**:

****

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class StudentsController extends Controller{

    /\*\*

     \* Display a listing of the resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index() {

        //

    }

    /\*\*

     \* Show the form for creating a new resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function create() {

        //

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return \Illuminate\Http\Response

     \*/

    public function store(Request $request) {

        //

    }

    /\*\*

     \* Display the specified resource.

     \*

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function show($id) {

        //

    }

    /\*\*

     \* Show the form for editing the specified resource.

     \*

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function edit($id) {

        //

    }

    /\*\*

     \* Update the specified resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function update(Request $request, $id) {

        //

    }

    /\*\*

     \* Remove the specified resource from storage.

     \*

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function destroy($id) {

        //

    }

}

**4-Définition des routes et des views.**

Les routes pour laravel8 sont gérées au niveau du fichier **route/web.php**. Afin de pouvoir ajouter correctement des routes qui pointent vers des views, on doit préalablement créer le Template de base et les views:

4.1 **Création du template de base resources/view/layout.blade.php**

1. créer ou télécharger un template bootstrap et renommer le layout.blade.php

2. mettez le fichier layout.blade.php au sein du dossier resources/view pour obtenir à la fin, le fichier resources/view/layout.blade.php (ce fichier va être utilisé comme template de base pour notre application)

3. placer ensuite les fichiers du bootstrap bootstrap.min.css dans un dossier public/css/ et bootstrap.bundle.min.js dans le dossier public/js/ (les dossiers public/css/ et public/js/ doivent être crées lorsqu'ils n'existent pas!)

4. lier ensuite les feuilles .css .js du bootstrap en ajoutant le code suivant:

**<link rel="stylesheet" href="/css/bootstrap.min.css">**

**<script src="js/bootstrap.bundle.min.js" ></script>**

5. ajouter les tags nécessaires pour faire un héritage au fichier resources/view/layout.blade.php :

**@yield('content')**

6. créer ensuite le fichier: index.blade.php au sein du dossier resources/view/ et y ajouter les balises d'héritage:

@extends('layout')

@section('content')

<!-- Content here ! -->

@endsection

7. Faite la même chose pour les fichiers, create.blade.php , edit..blade.php, update.blade.php et delete.blade.php (vous devez créer ces fichiers).

4.2 Création des routes

Pour créer une route, rien de plus simple, il suffit d'utiliser l'objet Root et d'indiquer le suffixe souhaité pour l'url de l'application et le Controller de l'application au niveau du fichier 'Routes/web.php':

use App\Http\Controllers\StudentController;

Route::resource('students' , StudentController::class);

**Remarque :**

Afin d'éviter les vulnérabilité d'attaque sous forme d'un envoie de données en masse, on doit ajouter le tableau $fillable au modèle:

app/Models/Students.php

class Students extends Model {

use HasFactory;

protected $fillable = ['name', 'email' , 'phone' , 'section' , 'image'];

}

**5- Configuration du Tailwind avec Laravel 8.**

**6-Création et affichage des données**

6.1 Création de l'objet tableau des données au niveau du contrôleur

Nous utilisons la méthode index() du contrôleur app/Http/Controllers/StudentsController.php pour pointer et afficher les données sur le fichier du template resources/view/index.blade.php

/ app/Http/Controllers/StudentsController.php

public function index()

{

// recuperation des donnees sous forme d'un tableau

$students = Students::all();

// afficher les donnees sur le template index.blade.php

return view('index', compact('students'));

}

6.2 - Création et insertion des données au niveau du contrôleur

6.2.1 - la méthode create()

La méthode create du contrôleur pointe directement vers le formulaire d'insertion des données au niveau du template qui sera rprésenté par le fichier create.blade.php

Méthode create()

public function create() {

return view('create');

}

Fichier template: create.blade.php

Le fichier de template create.blade.php est utilié pour accéder au formulaire d'insertion de données, il est accéssible via l'adresse url: <http://127.0.0.1:8000/create>

**Formulaire :**

**@extends('layout')**

**@section('content')**

**<h2 style="color:blue;">Add New Student </h2>**

**<form method="POST" action="{{ route('students.store') }}" enctype="multipart/form-data">**

**@csrf**

**<div class="form-group">**

**<label for="name">Name:</label>**

**<input type="text" class="form-control" name="name" />**

**</div>**

**<div class="form-group">**

**<label for="email">Email:</label>**

**<input type="text" class="form-control" name="email" />**

**</div>**

**<div class="form-group">**

**<label for="phone">Phone:</label>**

**<input type="text" class="form-control" name="phone" />**

**</div>**

**<div class="form-group">**

**<label for="section">Section :</label>**

**<select name="section" style="width: 100%;">**

**<!-- Les options -->**

**<option value="Math">Math</option>**

**<option value="Svt">SVT</option>**

**<option value="Physique">Physique</option>**

**<option value="Informatique">Informatique</option>**

**</select>**

**</div>**

**<div class="form-group">**

**<label for="image">Image :</label>**

**<input type="file" name="image" class="form-control">**

**</div>**

**<button type="submit" class="btn btn-success" style="float:right; width:300px;">Add Student</button>**

**</form>**

**@endsection**

6.2.2 La méthode d'insertion des données store()

Comme son nom l'indique, la méthode store() récupère les données envoyés depuis le formulaire d'insertion des données au niveau du view create.blade.php et les insère directement au sein de la table students de la base de donnée.

Méthode store()

**public function store(Request $request)**

**{**

**$validatedData = $request->validate([**

**'name' => 'required|max:255',**

**'email' => 'required|max:255',**

**'phone' => 'required',**

**'section' => 'required|max:255',**

**'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',**

**]);**

**if ($image = $request->file('image')) {**

**$destinationPath = 'image/';**

**$profileImage = date('YmdHis') . "." . $image->getClientOriginalExtension();**

**$image->move($destinationPath, $profileImage);**

**$validatedData['image'] = "$profileImage";**

**}**

**$students = Student::create($validatedData);**

**return redirect('/students')->with('success', 'Student created successfully');**

**}**

6.2.3 Template d'affichage des données index.blade.php

**@extends('layout')**

**@section('content')**

**<div class="table-wrapper">**

**<table class="fl-table">**

**<thead style="height: 5px;">**

**<tr>**

**<th> ID </th>**

**<th>Name</th>**

**<th>Email</th>**

**<th>Phone</th>**

**<th>Section</th>**

**<th>Image</th>**

**<th>Show </th>**

**<th>Update </th>**

**<th>Delete</th>**

**</tr>**

**<tr>**

**<td colspan="7"></td>**

**</tr>**

**</thead>**

**<tbody>**

**@foreach($students as $student)**

**<tr>**

**<td>{{$student->id}}</td>**

**<td>{{$student->name}}</td>**

**<td>{{$student->email}}</td>**

**<td>{{$student->phone}}</td>**

**<td style="color:brown; font-weight: bold;">{{$student->section}}</td>**

**<td><img src="/image/{{ $student->image }}" width="64" height="64"></td>**

**<td style="vertical-align:middle; ">**

**<form method="POST" align="left">**

**<a ; class="btn btn-info" href="{{ route('students.show' , $student->id) }}">Show</a>**

**</form>**

**</td>**

**<td style="vertical-align:middle; ">**

**<form method="POST" align="left">**

**<a class="btn btn-primary" href="{{ route('students.edit' , $student->id) }}">Edit</a>**

**</form>**

**</td>**

**<td style="vertical-align:middle; ">**

**<form method="POST" align="left">**

**<a class="btn btn-danger" href="{{ url('students.destroy' ,$student->id) }}">Delete</a>**

**</form>**

**</td>**

**</tr>**

**@endforeach**

**</tbody>**

**</table>**

**<div style="width:650px;">**

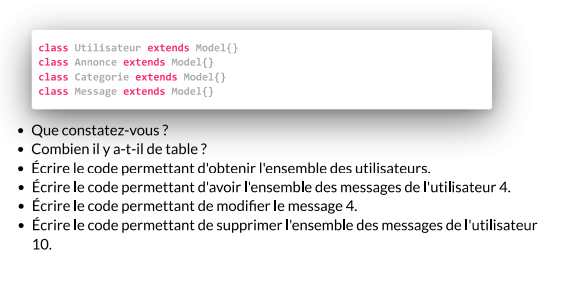
**{!! $students->links() !!}**

**</div>**

**</div>**

**@endsection**

**Exercice 2:** Gestion des annonces

****