**Laboratoire Pratique : Gestion des Données avec Laravel (Migrations, Modèles et Eloquent)**

**Objectif**

Ce laboratoire vous permettra de pratiquer les concepts clés liés aux migrations et aux modèles dans Laravel. Vous apprendrez à créer une base de données, à gérer ses tables via des migrations, et à interagir avec les données grâce à Eloquent ORM.

**Scénario du Lab**

Vous êtes chargé(e) de développer une application pour gérer une liste de contacts. L'application doit permettre :

1. La création d'une table **contacts** contenant les champs suivants :
   * **id** : Identifiant unique.
   * **name** : Nom du contact.
   * **email** : Adresse email du contact.
   * **phone** : Numéro de téléphone du contact.
   * **created\_at** et **updated\_at** : Horodatage automatique.
2. L'ajout de contacts via un formulaire.
3. La récupération et l'affichage des contacts enregistrés.

**Étapes du Lab**

**Étape 1 : Configuration de la Base de Données**

1. Créez une nouvelle base de données MySQL ou SQLite pour votre projet Laravel.
2. Configurez le fichier **.env** avec les informations suivantes :

plaintext

Copier

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=laravel\_contacts

DB\_USERNAME=root

DB\_PASSWORD=

**Étape 2 : Création de la Migration**

1. Générez une migration pour créer la table **contacts** :

bash

Copier

php artisan make:migration create\_contacts\_table --create=contacts

1. Ouvrez le fichier de migration généré dans **database/migrations** et ajoutez les colonnes nécessaires :

php

Copier

public function up()

{

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

$table->id();

$table->string('name');

$table->string('email')->unique();

$table->string('phone')->nullable();

$table->timestamps();

});

}

public function down()

{

Schema::dropIfExists('contacts');

}

1. Appliquez la migration pour créer la table :

bash

Copier

php artisan migrate

**Étape 3 : Création du Modèle**

1. Générez un modèle **Contact** associé à la table **contacts** :

bash

Copier

php artisan make:model Contact

1. Ajoutez la propriété **$fillable** dans le modèle pour autoriser l'assignation de masse :

php

Copier

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Contact extends Model

{

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

}

**Étape 4 : Création du Contrôleur**

1. Générez un contrôleur **ContactsController** :

bash

Copier

php artisan make:controller ContactsController

1. Implémentez les méthodes suivantes dans le contrôleur :
   * **index()** : Affiche la liste des contacts.
   * **create()** : Affiche le formulaire pour ajouter un nouveau contact.
   * **store(Request $request)** : Enregistre un nouveau contact dans la base de données.
   * **destroy(Contact $contact)** : Supprime un contact.

Exemple de code pour le contrôleur :

php

Copier

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\Contact;

class ContactsController extends Controller

{

public function index()

{

$contacts = Contact::all();

return view('contacts.index', compact('contacts'));

}

public function create()

{

return view('contacts.create');

}

public function store(Request $request)

{

$request->validate([

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

'email' => 'required|email|unique:contacts',

'phone' => 'nullable|string|max:20',

]);

Contact::create($request->all());

return redirect()->route('contacts.index')->with('success', 'Contact ajouté avec succès.');

}

public function destroy(Contact $contact)

{

$contact->delete();

return redirect()->route('contacts.index')->with('success', 'Contact supprimé avec succès.');

}

}

**Étape 5 : Définition des Routes**

1. Ajoutez les routes suivantes dans **routes/web.php** :

php

Copier

use App\Http\Controllers\ContactsController;

Route::get('/contacts', [ContactsController::class, 'index'])->name('contacts.index');

Route::get('/contacts/create', [ContactsController::class, 'create'])->name('contacts.create');

Route::post('/contacts', [ContactsController::class, 'store'])->name('contacts.store');

Route::delete('/contacts/{contact}', [ContactsController::class, 'destroy'])->name('contacts.destroy');

**Étape 6 : Création des Vues**

**1. Vue d'Index (resources/views/contacts/index.blade.php)**

Affiche la liste des contacts :

html

Copier

@extends('layouts.app')

@section('content')

<h1>Liste des Contacts</h1>

<a href="{{ route('contacts.create') }}" class="btn btn-primary">Ajouter un Contact</a>

<table class="table">

<thead>

<tr>

<th>Nom</th>

<th>Email</th>

<th>Téléphone</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

@foreach ($contacts as $contact)

<tr>

<td>{{ $contact->name }}</td>

<td>{{ $contact->email }}</td>

<td>{{ $contact->phone }}</td>

<td>

<form action="{{ route('contacts.destroy', $contact) }}" method="POST">

@csrf

@method('DELETE')

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

</form>

</td>

</tr>

@endforeach

</tbody>

</table>

@endsection

**2. Vue de Création (resources/views/contacts/create.blade.php)**

Formulaire pour ajouter un nouveau contact :

html

Copier

@extends('layouts.app')

@section('content')

<h1>Ajouter un Contact</h1>

<form action="{{ route('contacts.store') }}" method="POST">

@csrf

<div>

<label for="name">Nom :</label>

<input type="text" name="name" id="name" required>

</div>

<div>

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

<input type="email" name="email" id="email" required>

</div>

<div>

<label for="phone">Téléphone :</label>

<input type="text" name="phone" id="phone">

</div>

<button type="submit">Enregistrer</button>

</form>

@endsection

**Étape 7 : Tester l'Application**

1. Lancez votre serveur local :

bash

Copier

php artisan serve

1. Accédez à **http://localhost:8000/contacts** pour voir la liste des contacts.
2. Utilisez le lien "Ajouter un Contact" pour tester le formulaire.
3. Vérifiez que les contacts sont bien enregistrés dans la base de données.

**Étape 8 : Extensions Possibles**

1. **Validation Avancée** : Ajoutez des règles de validation supplémentaires pour le champ **phone**.
2. **Requête AJAX** : Implémentez l'ajout de contacts via une requête AJAX pour éviter le rechargement de la page.
3. **Fichiers Seeder** : Créez un seeder pour insérer des données de test dans la table **contacts**.

**Questions de Réflexion**

1. Que se passe-t-il si vous omettez la propriété **$fillable** dans le modèle **Contact** ?
2. Comment annuler la dernière migration appliquée ?
3. Quelle est l'utilité de la méthode **timestamps()** dans les migrations ?