PARTIE 1 : Laravel

Editeur de texte : Visual studio code

Téléchargez laragon ou xampp : après avoir installé laragon

Téléchargez composer qui est le (dependency Manager PHP) =gestionnaire des dépendances

1. **Création de projet** :

   -----/htdocs>**composer create-project –prefer-dist laravel/laravel monapp**

   -----/htdocs/monapp>**php artisan serve** (Pas obligatoire pour le moment)

   1.1. Ouvrez votre fichier **.env** puis saisissez le nom de la BD, allez dans phpmyadmin et créer le nom de la BD

```
Exemple : du fichier .env de laravel

DB_CONNECTION=mysql

DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravelIam
DB_USERNAME=root
DB_PASSWORD=
```

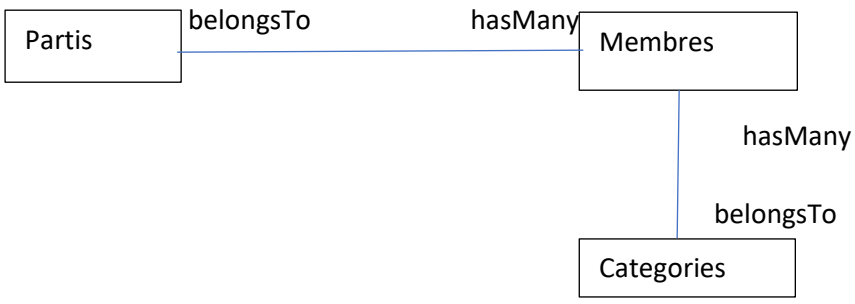
2. **Création de Controller** :

   -----monapp> **php artisan make :controller MembresController**

   -----monapp> **php artisan make :controller CategoriesController**

   -----monapp> **php artisan make :controller PartisController**

3. **Création de modèles** : (schéma de la BD)



4. Création de modèles et migration :

-----monapp> **php artisan make :model Membres - - migration**

-----monapp> **php artisan make :model Partis - - migration**

-----monapp> **php artisan make :model Categories - - migration**

5. Code de la catégories Model :

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Categories extends Model
{
    //

    Protected $fillable=array('nomp') ;
    Public static $rules=array('nomp'=>'required|min:3') ;

    Public function membres()
    {
    Return $this -> hasMany('App\Membres') ;
    }
}
```

Partis Model :

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Partis extends Model
{
    //
    Protected $fillable=array('nomp') ;
    Public static $rules=array('nomp'=>'required|min:2') ;

    Public function membres()
```

```
    {
    Return $this -> hasMany('App\Membres') ;
    }
}
```

6. **Code du membres Model** :

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Membres extends Model
{
    //
    Protected
$fillable=array('categories_id','partis_id','user_id','nom','prenom','naiss','
genre','fonction');

    Public static $rules=array('categories_id'=>'required|integer',
                               'partis_id'=>'required|integer',
                               'user_id'=>'required|bigInteger',
                               'nom'=>'required|min:2',
                               'prenom'=>'required|min:3',
                               'naiss'=>'required|min:8',
                               'genre'=>'required|min:1',
                               'fonction'=>'required|min:3'
                               ) ;

    Public function categories()
    {
    Return $this -> belongsTo('App\Categories') ;
    }

      Public function partis()
    {
    Return $this -> belongsTo('App\Partis') ;
    }
      Public function user()
```

```
    {
    Return $this -> belongsTo('App\User') ;
    }



}
```

7. **Gestion de la migration catégories** :

```php
<?php

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

class CreateCategoriesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->increments('id');
            $table->string('nom');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('categories');
    }
}
```

Gestion du fichier de migration partis :

```php
<?php

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

class CreatePartisTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('partis', function (Blueprint $table) {
            $table->increments('id');
            $table->string('nom');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('partis');
    }
}
```

Gestion du fichier de migration membres :

```php
<?php

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

```php
class CreateMembresTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('membres', function (Blueprint $table) {
         $table->increments('id');
         $table->integer('categories_id')->unsigned();
         $table->foreign('categories_id')->references('id')->on('categories');
         $table->foreign('partis_id')->references('id')->on('partis');
         $table->bigInteger('user_id')->unsigned();
         $table->foreign('user_id')->references('id')->on('users');
         $table->string('nom');
         $table->string('prenom');
         $table->date('naiss');
         $table->string('genre');
         $table->string('fonction');
         $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('membres');
    }
}
```

NB : enlevez les deux contraintes qui peuvent bloquer la génération dans le dossier de **migration «fichier users_tables_-----« : effacer unique() au niveau d'email et «fichier password_reset_---« index() au niveau de email (car la taille peut dépasser la contrainte fixée) avant de générer les migrations.**

8. **Création de la migration des tables** :

-----monapp> **php artisan migrate**

NB : si la migration refuse peut-être de passer veuillez ouvrir **App/Providers/AppServiceProvider.php**

Ajoutez un use :

```
use Illuminate\Support\Facades\Schema;
```

Puis dans la fonction register() ajouter :

```
  public function register()
    {
        //
      Schema ::defaultStringLength(191) ;
    }
```

Puis supprimer les tables avant de générer à nouveau : php artisan migrate

NB : Merci de toujours commencer votre projet avec la création de votre modèle en premier avant de gérer les Controller les pages et autres.

LARAVEL BREEZE :

Laravel Breeze est une implémentation simple et minimale de toutes les fonctionnalités d'authentification de Laravel, y compris la connexion, l'enregistrement, la réinitialisation du mot de passe, la vérification de l'e-mail et la confirmation du mot de passe.

La couche de vue par défaut de Laravel Breeze est composée de modèles Blade simples stylisés avec Tailwind CSS. Breeze fournit un excellent point de départ pour commencer une nouvelle application Laravel.

Installation :

```
composer require laravel/breeze --dev
```

Une fois que Composer a installé le package Laravel Breeze :

```
php artisan breeze:install
```

Nb : la commande npm ne marche pas sur l'éditeur visual studiocode, téléchargez le nodeJS et installer le, puis démarrer le node.js **commande prompt** dans windows.

Chargez le projet à partir de l'invite de nodeJS puis taper la commande **npm**

```
npm install

npm run dev

php artisan migrate
```

……>php artisan make :auth