Migraciones

**Crear una migración**: php artisan make:migration create\_cursos\_table

**Revertir los cambios de las migraciones** php artisan migrate:rolback

**Eliminar las migraciones**: php artisan migrate:fresh

**Editar una columna de la base de datos con las migraciones:**

php artisan make:migration add\_avatar\_to\_users\_table

**modificación de columnas**

Antes de modificar una columna, debe instalar el doctrine/dbalpaquete mediante el administrador de paquetes de Composer. La biblioteca Doctrine DBAL se usa para determinar el estado actual de la columna y para crear las consultas SQL necesarias para realizar los cambios solicitados en su columna: composer require doctrine/dbal

**Eloquent**

se utiliza tinker para probar los modelos de eloquent

**seeders**

llenar la base de datos con información falsa

ejecutar seeder php artisan de:seed

crear un seeder php artisan make:seeder CursoSeeder

para que se ejecuten un seeder creado debemos impórtalo a la clase principal para los seeder que se llama DatabaseSeeder.php con call() que pertenece a la clase Seeder

public function run()

    {

        // \App\Models\User::factory(10)->create();

        $this->call(CursoSeeder::class);

    }

php artisan migrate –seed para ejecutar todos

**factory**

se crean para llenar de información falsa en la base de datos para fines de prueba

para creación: php artisan make:factory CursoFactory

crear un factory pero asociarlo a un modelo en especifivo:

php artisan make:factory CursoFactory --model=Curso

se crea crea el factory con información que esta en la tabla

<?php

namespace Database\Factories;

use App\Models\Curso;

use Illuminate\Database\Eloquent\Factories\Factory;

class CursoFactory extends Factory

{

    protected $model = Curso::class;

    /\*\*

     \* Define the model's default state.

     \*

     \* @return array

     \*/

    public function definition()

    {

        return [

            'name' => $this->faker->sentence(),

            'descripcion' => $this->faker->paragraph(),

            'categoria' => $this->faker->randomElement(['Desarroolo Web', 'Diseño web'])

        ];

    }

}

despues se coloca ese factory en el seeder para que luego se pueda llamar ese seeder y llenar la tabla en la base de datos

<?php

namespace Database\Seeders;

use App\Models\Curso;

use Illuminate\Database\Seeder;

class CursoSeeder extends Seeder

{

    /\*\*

     \* Run the database seeds.

     \*

     \* @return void

     \*/

    public function run()

    {

        //decirle a la clase CursoSeeder que llene con el factory con el mismo nombre

        //con 50 elementos

        Curso::factory(50)->create();

    }

}

Ejecutar el seeder php artisan migrate:fresh –seed nota: este comando elimina información de la base de datos

Nota: se pueden ejecutar directamente los factory en el archivo principal de los seeder. Pero se puede seguir creando los seeder para llenar la información de la base de datos con información compleja.