 Laravel nous permet de rentrer des enregistrements de manière beaucoup plus "propre" mais surtout beaucoup plus rapide et réutilisable grâce aux **Seeders** et aux **Factories** ! Nous allons voir que les deux travaillent ensemble et qu'ils portent bien leurs noms !

1. Les Seeders

Dans l'absolu il est possible de n'utiliser que les Seeders pour peupler notre base de données de manière très basique. Car leur action est bien de "seed" (*"ensemencer"* en anglais) notre base de données.

Pour créer un nouveau seeder pour notre table cars lancez la commande :

php artisan make:seeder CarsTableSeeder

<?php

namespace Database\Seeders;

use App\Models\Car;

use Illuminate\Database\Seeder;

class CarTableSeeder extends Seeder

{

    /\*\*

     \* Run the database seeds.

     \*

     \* @return void

     \*/

    public function run()

    {

        Car::factory()->count(50)->create();

    }

}

Les Factories

("usines" en anglais) sont donc là pour nous permettre de créer des enregistrements.

On peut voir que notre UserFactory est déjà créé par défaut. Nous allons donc créer le **PostFactory**

php artisan make:factory CarFactory --model=Car

<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

class CarFactory extends Factory

{

    /\*\*

     \* Define the model's default state.

     \*

     \* @return array

     \*/

    public function definition()

    {

        return [

            'Marque' => $this->faker->sentence(6,true),

            'Prix' =>$this->faker->numberBetween($min = 1500, $max = 6000)

        ];

    }

}

Dans le fichier database\seeders\DatabaseSeeder.php

<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

    /\*\*

     \* Seed the application's database.

     \*

     \* @return void

     \*/

    public function run()

    {

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

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

    }

}

Finalement lancer la commande :

php artisan migrate --seed

## Utiliser la pagination dans Laravel

La [pagination](https://laravel.com/docs/7.x/pagination) est un moyen de repartir des données étendues sur plusieurs pages, la pagination dans Laravel est très facile à utiliser ; il est intégré au générateur de requêtes et à Eloquent ORM. La pagination Laravel couvre la limite et le décalage automatiquement.

L’implémentation de la pagination dans laravel est fluide, ajoutez la fonction de pagination laravel dans la fonction index() à l’intérieur de la classe CarController. Supprimez all() et utilisez paginate() et il prend un nombre comme argument, ce nombre définit le nombre de résultats à afficher à l’utilisateur.

De plus, vous devez importer et définir useBootstrap() dans la fonction de démarrage dans app/Providers/AppServiceProvider.php :

<?php

namespace App\Providers;

use Illuminate\Support\ServiceProvider;

use Illuminate\Pagination\Paginator;

class AppServiceProvider extends ServiceProvider

{

    /\*\*

     \* Register any application services.

     \*

     \* @return void

     \*/

    public function register()

    {

        //

    }

    /\*\*

     \* Bootstrap any application services.

     \*

     \* @return void

     \*/

    public function boot()

    {

        Paginator::useBootstrap();

    }

}

Dans la fonction index() ajouter paginate(5) au lieu de all()

Dans le fichier index.blade.php vous ajoutez :

<div class="d-flex justify-content-center">

      {{ $voitures->links() }}

</div>