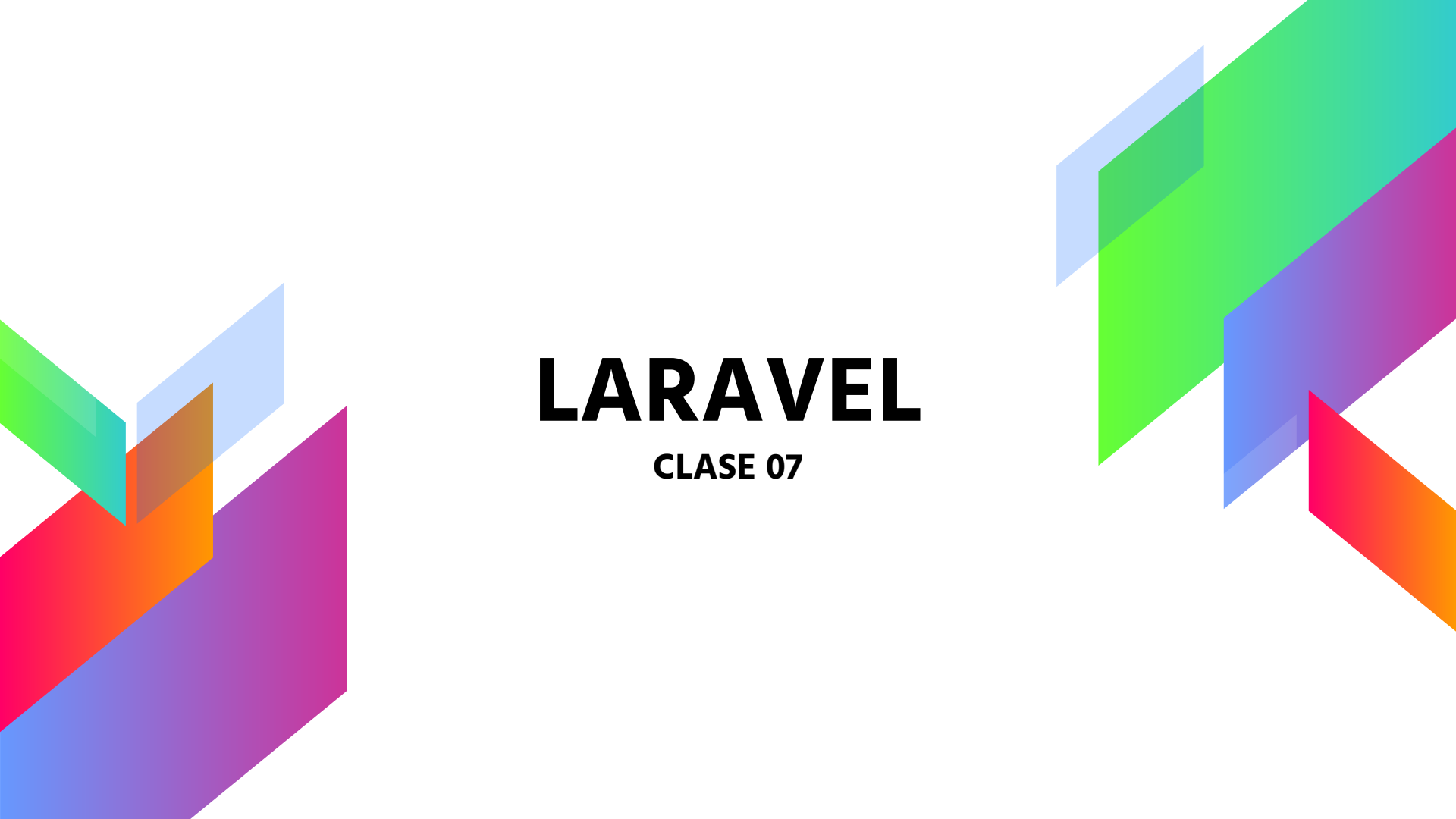


LARAVEL

CLASE 07



Abstract geometric shapes in the top-left corner, including a green triangle, a blue parallelogram, and a purple parallelogram.

MIGRATIONS

¿Qué son y cómo funcionan las migraciones?

Abstract geometric shapes in the top-right corner, including a green triangle, a blue parallelogram, and a purple parallelogram.

```
> database/migrations/2017_07_03_180000_create_flights_table.php
```

```
class CreateFlightsTable extends Migration {

    public function up() {
        Schema::create('flights', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->string('airline');
            $table->timestamps();
        });
    }


    public function down() {
        Schema::drop('flights');
    }
}
```



¿Cómo creamos una migración?

Desde la consola de comandos, ejecutamos el siguiente código:

```
php artisan make:migration nombre_de_migration  
database/migrations/  
2017_07_03_193012_nombre_de_migration.php
```



> Comandos útiles para migraciones

// Ejecuta todas las migraciones del proyecto que no hayan sido ejecutadas.

php artisan migrate

// Nos muestra el estado de cada migración. Es decir, si fue ejecutada o no.

php artisan migrate:status

// Deshace la última operación de migración.

php artisan migrate:rollback

// Deshace todas las operaciones de migración.

php artisan migrate:reset

// Deshace y vuelve a ejecutar todas las operaciones de migración.

php artisan migrate:refresh

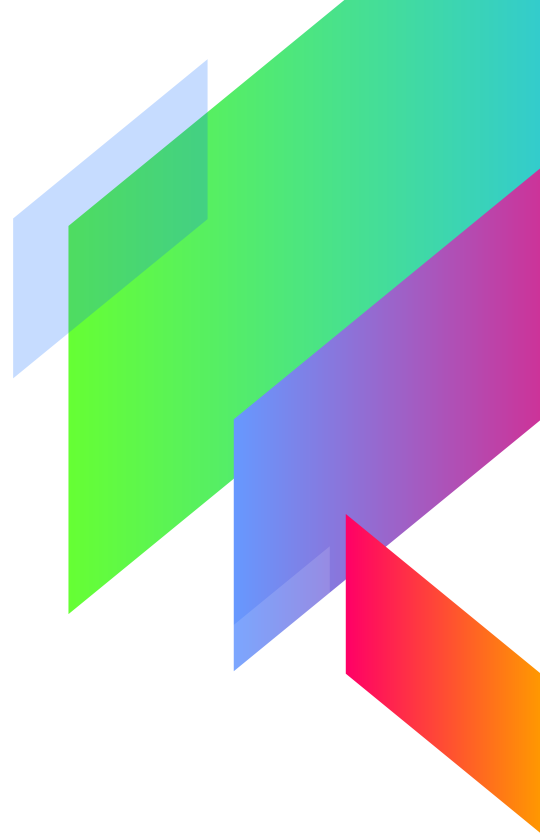
**ES MOMENTO DE
PRACTICAR !**





MODEL FACTORIES

Poblemos nuestra base de datos
con datos falsos al azar



Definiendo el modelFactory

```
$factory->define(App\Product::class, function ($faker) {  
    return [  
        'title'      => $faker->word,  
        'price'      => $faker->numberBetween(1, 900),  
        'description' => $faker->paragraph(10)  
    ];  
});  
  
// Luego, para usarlo  
$product = factory(App\Product::class)->make();  
$product->save();  
  
factory(App\Product::class)->create();  
factory(App\Product::class)->times(50)->create();
```

<https://github.com/>


```
$factory->define(App\Product::class, function ($faker) {  
    return [  
        'title'      => $faker->word,  
        'price'      => $faker->numberBetween(1, 900),  
        'description' => $faker->paragraph(10)  
    ];  
});
```

```
// Luego, para usarlo  
$product = factory(App\Product::class)->make();  
$product->save();
```

```
factory(App\Product::class)->create()  
factory(App\Product::class)->times(5)->create()
```

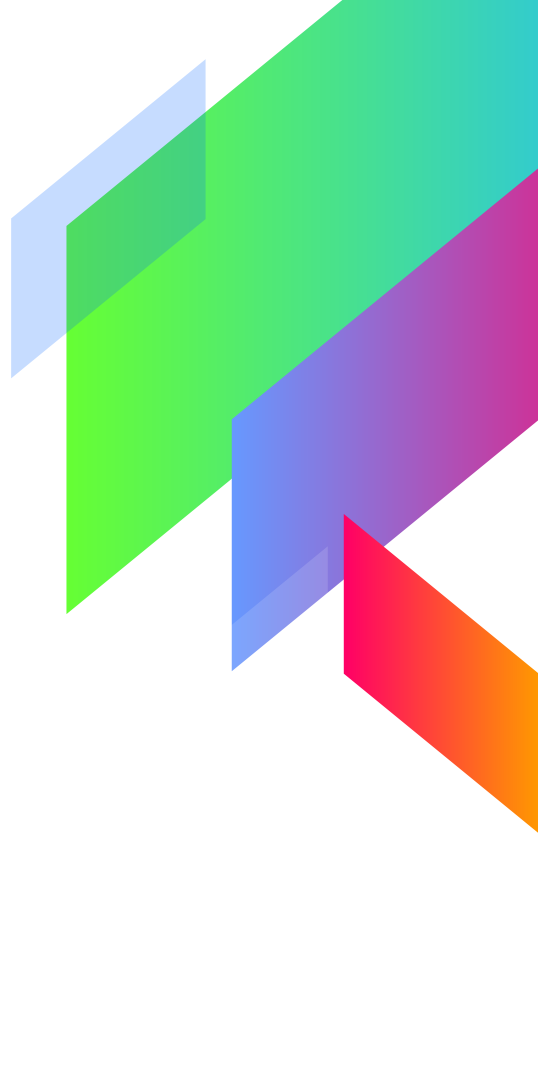
NO USAR EN CONTROLLERS

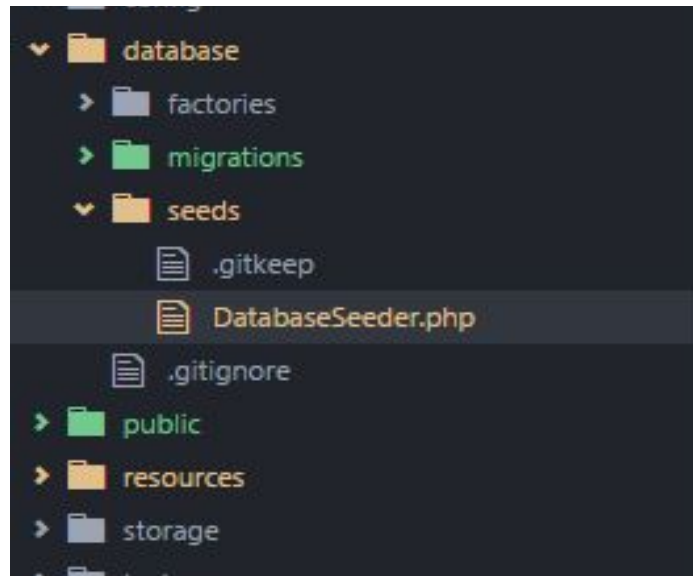
<https://github.com/>

Abstract geometric shapes in the top-left corner, including a green parallelogram, a light blue parallelogram, a brown parallelogram, and a large purple parallelogram.

SEEDERS

¿Qué son y cómo funcionan los seeders?

Abstract geometric shapes in the top-right corner, including a green parallelogram, a light blue parallelogram, a purple parallelogram, and a red parallelogram.



```
use App\Product;
use Illuminate\Database\Seeder;
use Illuminate\Database\Eloquent\Model;

class DatabaseSeeder extends Seeder
{
    public function run()
    {
        factory(Product::class)->times(50)->create();

        $actors = factory(Actor::class)->times(50)->create();
        $movies = factory(Movie::class)->times(50)->create();

        foreach ($movies as $movie) {
            // Asocio la película con 3 actores al azar
            $movie->actors()->saveMany($actors->random(3));
        }
    }
}
```

```
use App\Product;
use Illuminate\Database\Seeder;
use Illuminate\Database\Eloquent\Model;

class DatabaseSeeder extends Seeder
{
    public function run()
    {
        $directors = factory(Director::class, 10)->create();

        foreach ($directors as $director) {
            // Creo 5 películas por cada director
            factory(Movie::class, 5)->create([
                'director_id' => $director->id,
            ]);
        }
    }
}
```

Para correr nuestros seeders desde la consola



```
php artisan db:seed
```

**ES MOMENTO DE
PRACTICAR !**





I'll be back.