

# LARAVEL

AULA 07

The background features abstract geometric shapes in the corners. On the left, there are overlapping triangles in shades of green, blue, orange, and purple. On the right, there are larger, more complex shapes in shades of green, blue, purple, and orange, creating a modern, geometric aesthetic.

# MIGRATIONS

O que são e como funcionam as migrações?

<https://laravel.com/docs/5.6/migrations>

```
> 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', 50);
            $table->unsignedInteger('airline_id');
            $table->smallInteger('duration')->nullable();
            $table->timestamps();

            $table->foreign('airline_id')->references('id')->on('airlines');
        });
    }

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

```
> database/migrations/2017_07_04_180000_add_cost_to_flights_table.php
```

```
class AddCostToFlightsTable extends Migration {

    public function up() {
        Schema::table('flights', function (Blueprint $table) {
            $table->float('cost');
        });
    }

    public function down() {
        Schema::table('flights', function (Blueprint $table) {
            $table->dropColumn('cost');
        });
    }
}
```

## Como criar uma migração?

A partir da linha de comando, executar o seguinte código:

```
php artisan make:migration nome_de_migration  
database/migrations/  
2017_07_03_193012_nome_de_migration.php
```

## > Comandos úteis para migrações

```
// Executa todas as migrações do projeto que ainda não tenham sido executadas.  
php artisan migrate  
  
// Exibe o status das migrações. Ou seja, indica se foram executadas ou não.  
php artisan migrate:status  
  
// Desfaz a última operação de migração.  
php artisan migrate:rollback  
  
// Desfaz todas as operações de migração.  
php artisan migrate:reset  
  
// Desfaz e volta a executar todas as operações de migração.  
php artisan migrate:refresh
```

**HORA DE PRATICAR!**



Fazer o exercício 1

The slide features a light gray background with decorative geometric shapes in the corners. On the left, there are overlapping triangles in shades of green, blue, orange, and purple. On the right, there are similar shapes in shades of green, blue, purple, and orange. The main content is centered on the slide.

# MODEL FACTORIES

Vamos preencher nosso banco de  
dados  
com dados falsos aleatórios

<https://laravel.com/docs/5.6/seeding#using-model-factories>



## Como criar um factory?

A partir da linha de comando, executar o seguinte código:

```
php artisan make:factory nome_do_factory  
--model=Nome_Model
```

```
database/factories/  
nome_do_factory.php
```

## Definição do modelFactory

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

<https://github.com/fzaninotto/Faker>

## Definição do modelFactory

```
$factory->define(App\Product::class, function($faker) {  
    return [  
        'title'       => $faker->word,  
        'price'       => $faker->numberBetween(1, 900),  
        'description' => $faker->paragraph(5)  
    ];  
});
```

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

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

**NÃO  
USAR EM  
CONTROLLERS**

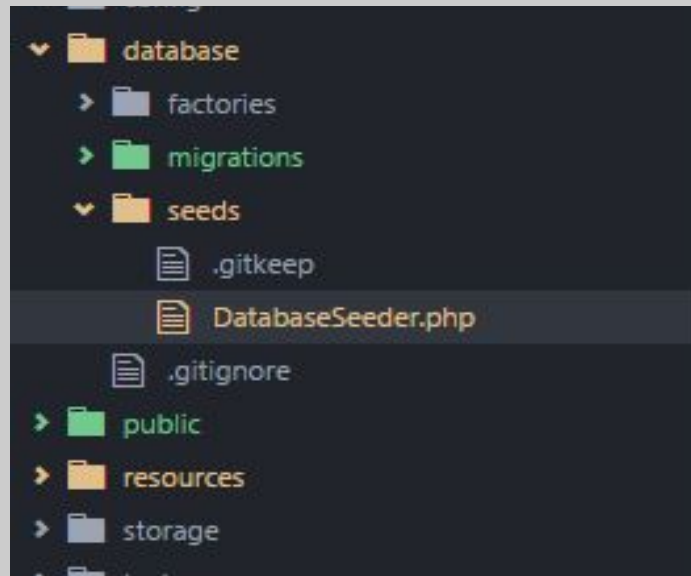
<https://github.com/fzaninotto/Faker>

The background features abstract geometric shapes in the corners. On the left, there are overlapping shapes in shades of green, blue, orange, and purple. On the right, there are similar shapes in shades of green, blue, purple, and orange. The central area is a plain light gray.

# SEEDERS

O que são e como funcionam os seeders?

<https://laravel.com/docs/5.6/seeding>



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

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

        foreach ($directors as $director) {
            // Criamos 5 filmes para cada diretor
            factory(Movie::class)->times(5)->create([
                'director_id' => $director->id,
            ]);
        }
    }
}
```

```
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) {
            // Associamos o filme a 3 atores aleatórios
            $movie->actors()->sync($actors->random(3));
        }
    }
}
```

## Como criar um seeder?


A partir da linha de comando, executar o seguinte código:

```
php artisan make:seeder nome_do_seeder
```

```
database/seeds/  
nome_do_seeder.php
```



Para executar os seeders a partir da linha de comando



```
php artisan db:seed
```

**HORA DE PRATICAR!**

## **Factories / Seeders**

Fazer o exercício 1

The background features abstract, overlapping geometric shapes in various colors including light blue, green, cyan, magenta, orange, and red. These shapes are arranged in a way that creates a sense of depth and movement, framing the central text.

**ATÉ A  
PRÓXIMA!**