LARAVEL

Parte 3

DIO CEZAR GO

MIGRATION

MIGRATION

- as migrações são como um controle de versão para seu banco de dados;
- permitem que o seu time modifique e compartilhe facilmente o esquema do banco de dados da sua aplicação;

MIGRATION

- para criar uma migração, utilize o comando do *artisan*: `make:migration`

```
php artisan make:migration create_users_table
```

MIGRATION

- isso criará uma classe localizada em *database/migrations;*
- todo arquivo de migração contém um timestamp que permite ao Laravel determinar a ordem de execução dos comandos no banco de dados;
- as opções `--table` e `--create` podem ser utilizada para indicar o nome da tabela ou se uma nova tabela estará sendo criada;

```
php artisan make:migration create_users_table --create=users
php artisan make:migration add_votes_to_users_table --table=users
```

MIGRATION - ESTRUTURA

- a classe de migração contém dois métodos: *up* e *down*;
- o método *up* é utilizado para adicionar novas tabelas, colunas, indexações em seu banco de dados;
- o método *down* deve simplesmente reverter as operações feitas em *up*;
- a criação de colunas e suas validações é bem intuitiva, mas podem estão ainda nesta lista completa: https://goo.gl/K22zTh

MIGRATION - ESTRUTURA

EXEMPLO DE DEFINIÇÃO DE UMA MIGRATION

```
<?php

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

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');
    }
}
```

MIGRATION - COMANDOS

- para executar as migrações, basta utilizar o comando:

```
php artisan migrate
```

- para dar rollback nas últimas atualizações, ou resetar todas as migrações:

```
php artisan migrate:rollback
php artisan migrate:reset
```

SEED

SEED

- muitas vezes pode ser necessário preencher suas tabelas com alguns dados, para testar as funcionalidades da sua aplicação;
- o *Laravel* possibilita isso com uma estrutura de criação de *seeds*;
- as *seeds* ficam em *database/seeds;*
- as classes podem ter o nome que você desejar, mas poderiam seguir a convenção: *UsersTableSeeder*;

SEED - ESCREVENDO

- para gerar um seeder, execute com o *artisan*: `make:seeder`

```
php artisan make:seeder UsersTableSeeder
```

SEED

- a classe seeder contém apenas um método chamado *run()*
- este método é chamado quando o comando do *artisan*: `php artisan db:seed` é chamado;
- utilize o comando em conjunto com o Query Builder para inserir dados da forma como desejar;

SEED - EXEMPLO

EXEMPLO DE DEFINIÇÃO DE UM SEED

```
<?php

use Illuminate\Database\Seeder;
use Illuminate\Database\Eloquent\Model;

class DatabaseSeeder extends Seeder{
    public function run(){
        DB::table('users')->insert([
            'name' => str_random(10),
            'email' => str_random(10).'@gmail.com',
            'password' => bcrypt('secret'),
        ]);
    }
}
```

THAT'S ALL FOLKS

MATERIAIS COMPLEMENTARES

- https://laravel.com/docs/5.5/migrations
- https://laravel.com/docs/5.5/seeding