Class three (3) - **Mass assignment**

Php artisan make:model Models\SoccerTeam –m

Serve para atribuir dados em massa

* Criou a pasta Models, por ‘boa pratica’ e o –M para criar o arquivo da Migration
* Usou o tinker

Php artisan tinker

$soccer = new \App\Models\SoccerTeam();

$soccer->name = ‘time’;

$soccer->save();

Buut, dificilmente se usa desta forma, normalmente se gera um array com vários dados e usa a funcao create();

\App\Models\SoccerTeam::create($array);

\App\Models\SoccerTeam::create(['name' => 'time1']);

Usando create, desconsidera-se tudo o que não estiver dentro do fillable da model

Voce pode instanciar a model e fazer consultas,  
$soccer = \App\Models\Soccerteam::find(1);

Muitos ORM tem mass assigment, não é exclusividade do Eloquent

### Class four (4) - Modelo de cliente para primeiro relacionamento

Vai relacionar times com clientes

Criou a model e a migration

Criou a factory

php artisan make:factory SoccerTeamFactory

### Class five (5) - Criando relacionamento many-to-one

Na migrate cliente\_clients\_table Criou o campo e a chave estrangeira  
 $table->integer('soccer\_team\_id')->unsigned();  
 $table->foreign('soccer\_team\_id')->references('id')->on('soccer\_teams');  
 // criando o campo e a chave estrangeira

Atualizou o banco com o comando:  
php artisan migrate:refresh --seed  
se você tiver seeder pronta para rodar

Porém, se o banco está em produção, use apenas   
php artisan migrate:refresh

Porquê com o seed (acho) que injeta dados fictícios

Agora, vai na Model Client.php pra descrever o relacionamento na Model, todo e qualquer relacionamento que for feito através do artisan(nas tabelas) tem que ser espelhado nas models

### Class six (6) - Manipulando relacionamento many-to-one Abriu o tinker $client = factory(\App\Models\Client::class)->states(\App\Models\Client::TYPE\_INDIVIDUAL)->make();

### Se usar o ->make(); é criado apenas em memória Se usar o ->create(); é criado apenas no banco

### Criou um time factory(\App\Models\SoccerTeam::class)->create();

### Atribuiu o time criado ao campo soccer\_team\_id do cliente que está em memória $client->soccer\_team\_id = 1;

### Salvou o cliente no banco $client->save();

### Fez uma consulta $client = \App\Models\Client::find(1)

### // o SqLite tem como padrão foreign keys off, então mudamos isso em AppServiceProvider if(config('database.default')){ \DB::connection()->getPdo()->exec('PRAGMA foreign\_keys=on;'); }

### 

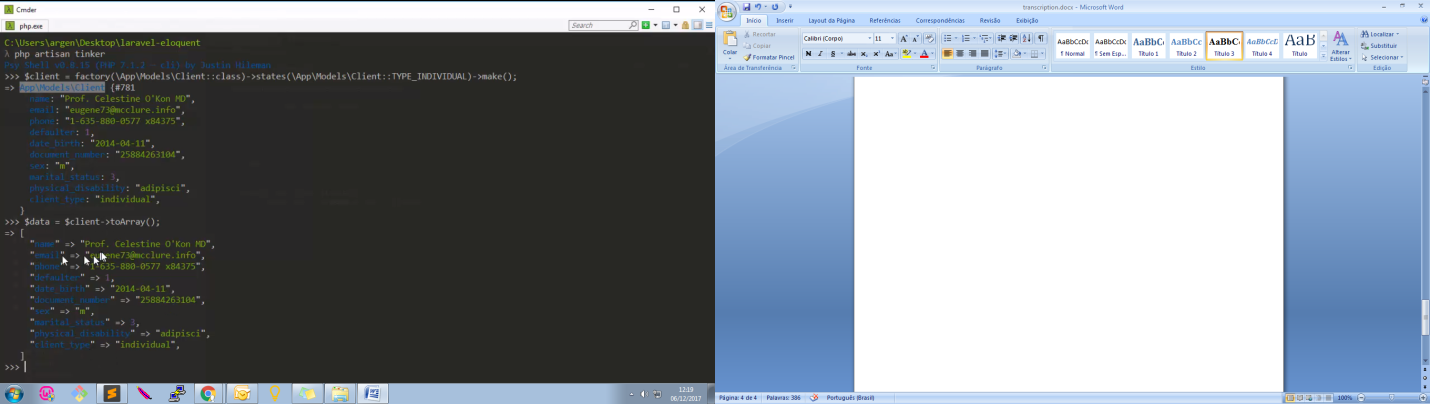
### 

### Class seven (7) - Manipulando relacionamento one-to-many

### 

### Agora está fazendo o inverso, quando eu for ver a tela de cadastro de Times, eu posso querer buscar todos os clientes que gostam daquele determinado Time, e isso fica explícito na Model

### Agora imagine se a partir de um time, eu possa cadastrar cliente, ex.: na tela do CAD de time eu tenha um botão “cadastrar cliente”, e quando clica neste botão o cliente será relacionado com este time



--

Acessou o time 3, e chamou o método create da model clients, através do método que está em time

$soccer->clients , significa colocar acessar a função clients que está na model SoccerTeam, pegar TODOS os clientes que estão ligados a este Time