

Já falamos da VIEW, que utiliza o BLADE (biblioteca de Template Engine do nosso Framework). Ele possui alguns comandos para facilitar a criação das páginas HTML do nosso site.

Já falamos do CONTROLLER, responsável por decidir o que precisa ser feito depois que a rota recebeu a requisição.

É ele quem vai, por exemplo, recuperar informações do banco de dados e encaminhar para a VIEW exibir o que precisa para o usuário.

E na última aula falamos do MODEL, que utiliza o Eloquent (biblioteca ORM do Laravel).

Ele representa uma tabela que existe no banco de dados.

Através do modelo poderemos consultar e manipular registros das tabelas do nosso banco de dados.

Mas só vimos como consultar alguma informação lá na tabela do banco de dados, não é? Usamos aquele método do Eloquent que retorna todos os registros da tabela.

```
// Localiza todos os filmes cadastrados na tabela
App\Filme::all()
```

Também vimos como buscar apenas um registro através do seu ID.

```
// Localiza todas as informações do filme com id igual a 1
App\Filme::find(1)
```

Mas existem diversos outros métodos que o Eloquent nos permite executar para, por exemplo:

```
// Pesquisar no banco de dados a partir do nome
App\Filme::where('nome', '=', 'Divertida Mente')->get()

// Montar pesquisas encadeando condições e ordenações
App\Filme::where('nome', '=', 'Divertida Mente')->where('rating', '>', '8')->get()
```

Hoje vamos ver como além de pesquisar, também criar, alterar e excluir um registro da tabela usando nossos modelos.

1. Lembra do CRUD?

Create-Read-Update-Delete

É uma sigla que identifica as operações mais comuns que você pode executar nas tabelas do seu banco de dados.

Da mesma maneira que aprendemos a fazer um “Hello World” ao estudar uma nova linguagem, “fazer um CRUD” é executar um INSERT, SELECT, UPDATE e DELETE no banco de dados usando o Laravel, por exemplo.

2. Formulários

Mas antes disso vamos lembrar de que forma um usuário consegue alterar ou criar um novo registro utilizando as páginas HTML que geramos.

É usando um FORM, não é?

E para criar um formulário no Laravel, onde fazemos isso?

No Model?

No Controller?

Ou na View?

Claro que na View =) Usando o Blade.

Montar um FORM no Laravel é exatamente como montamos antes no PHP tradicional.

Mas com dois requintes de crueldade:

```
<form method="POST" action="/filmes/delete/1">
    {{ csrf_field() }}
    {{ method_field('DELETE') }}
</form>
```

O `csrf_field()` é super necessário sempre que formos utilizar um formulário.

Ele produz uma chave (ou token) de segurança que apenas o Laravel conhece.

Isso garante que as informações recebidas pelo Framework vieram mesmo de uma página que ele mesmo criou.

E o `method__field()` especifica qual o tipo de rota que o formulário vai utilizar depois de submeter as informações.

Logo, o Laravel precisa que exista uma rota exatamente assim para processar as informações recebidas do formulário que preparou.

3. Request

Depois do formulário ser submetido ele envia as informações para a rota, que por sua vez repassa para o controlador.

O controlador recebe as informações do formulário através da classe Request.

```
public function nomeMetodo(Request $request)
{
    // Seu código aqui
}
```

Ele faz o papel dos velhos conhecidos `$_GET` e `$_POST`.

E conseguimos resgatar as informações encaminhadas pelo formulário assim:

```
// Método All - Retorna um array com todos os inputs e seus valores
$input = $request->all();

// Método Input - Retorna o valor de um só input
$input = $request->input('nome');

// Método Only - Retorna um array apenas com os inputs solicitados
$input = $request->only('nome', 'idade');
```

4. Validações

O Request tem um método muito legal chamado `validate()`.

Ele consegue exatamente validar se os campos enviados pelo formulário estão da forma como você precisa antes de gravar tudo na tabela do banco de dados.

É possível verificar se aquela string está no tamanho necessário ou torná-la obrigatória. Ou se a senha possui o mínimo de caracteres exigidos.

Olha como ele também pode conferir se aquele campo está ou não duplicado na tabela do banco de dados.

```
$request->validate([
    'nome' => 'required|unique:filmes|max:255',
    'idade' => 'required',
]);
```

Quando a validação falhar, o Laravel automaticamente volta para a view já informando onde está o problema.

Mas se tudo for validado com sucesso, tudo segue normalmente para a rota que você especificou.

Para exibir os erros de validação basta conferir lá na View a variável `$errors`, que é criada automaticamente pelo nosso Framework.

```
// Mostra os erros de validação na view
@if (count($errors) > 0)
<div class="alert alert-danger">
    <ul>
        @foreach ($errors->all() as $error)
            <li>{{ $error }}</li>
        @endforeach
    </ul>
</div>
@endif
```

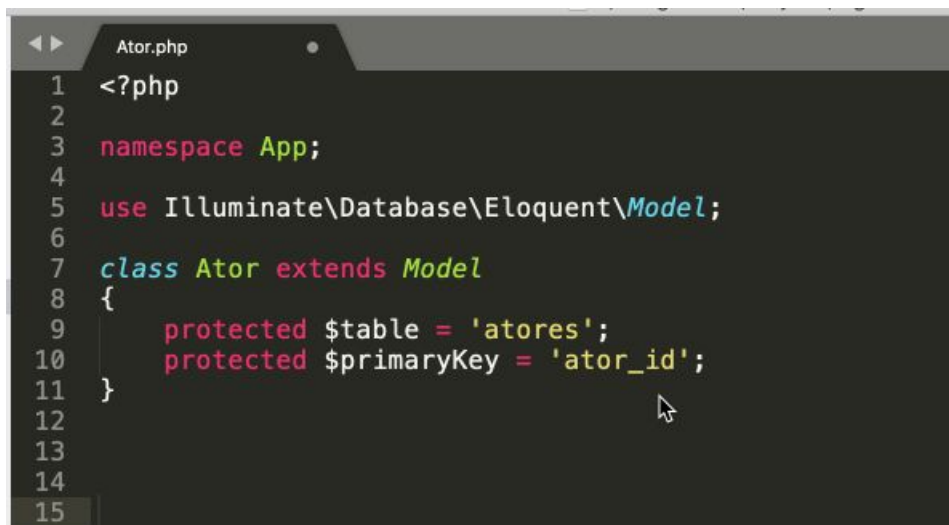
5. Configurar o Model

Primeiro precisamos lembrar do nosso Model.

Qual o nome dele?

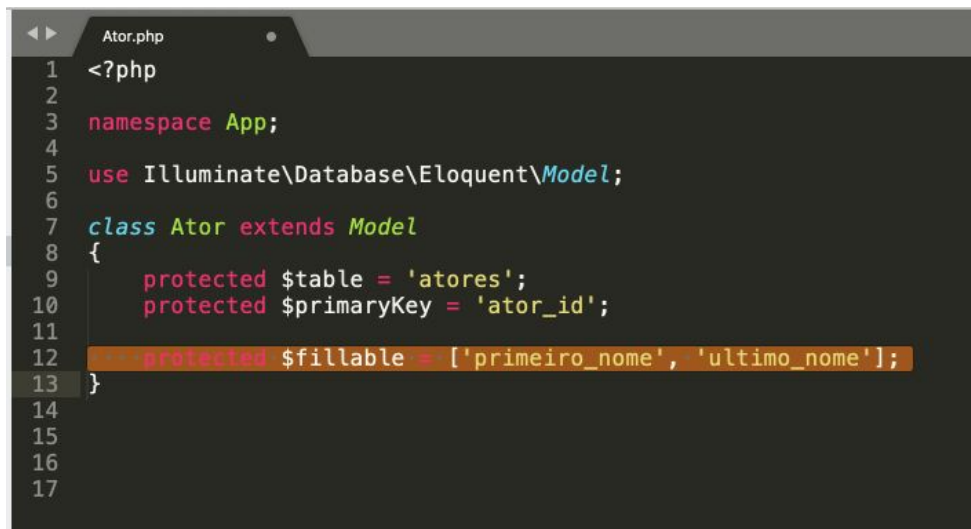
Onde ele fica mesmo?

O nosso modelo atualmente deve estar mais ou menos assim:



```
1 <?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class Ator extends Model
8 {
9     protected $table = 'atores';
10    protected $primaryKey = 'ator_id';
11 }
12
13
14
15
```

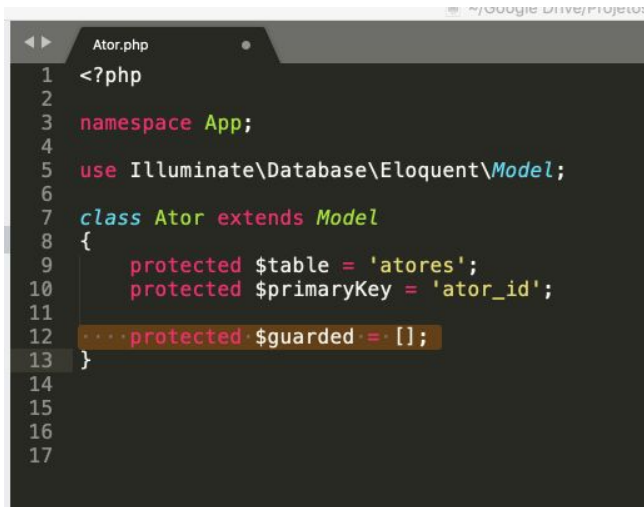
Mas para que possamos realizar inclusões e atualizações vamos também ter que incluir mais uma variável, veja só:



```
1 <?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class Ator extends Model
8 {
9     protected $table = 'atores';
10    protected $primaryKey = 'ator_id';
11
12    protected $fillable = ['primeiro_nome', 'ultimo_nome'];
13 }
14
15
16
17
```

A variável `$fillable`, dentro do nosso Model, indica para o Laravel quais são os campos da sua tabela que poderão ser alterados.

Existe também a variável `$guarded`, que funciona de forma inversa. Ela indica quais campos não poderão ser alterados, e se quiser que todos possam ser basta preenchê-la com array vazio.



```

1 <?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class Ator extends Model
8 {
9     protected $table = 'atores';
10    protected $primaryKey = 'ator_id';
11
12    ...protected $guarded = [];
13 }
14
15
16
17

```

Não se deve utilizar as duas ao mesmo tempo, ou usamos o `$fillable` ou o `$guarded`.

6. Agora no Controlador

Vai ser dentro do nosso controller que vamos colocar o código para alterar, incluir ou excluir algum registro da tabela no banco de dados.

Lembra que a rota chama uma function do controlador?

Confere então aqui abaixo os exemplos de como realizar essas operações.

Para incluir um registro na sua tabela basta:

```

$usuario = User::create([
    'nome' => $request->input('nome'),
    'idade' => $request->input('idade')
]);
$usuario->save();

```

Para atualizar um registro na sua tabela basta:

```

$usuario = Usuario::find(1);
$usuario->nome = $request->input('nome');
$usuario->idade = $request->input('idade');
$usuario->save();

```

E para excluir um registro basta:

```
$usuario = Usuario::find(1);  
$usuario->delete();
```

Para excluir vários registros de uma só vez basta:

```
// Exclui todos os filmes com a categoria igual a 5  
$filmes = Filme::where('categoria_id', '=', 5)->delete();
```

Para atualizar vários registros de uma só vez basta:

```
// Altera todos os filmes que estiverem na categoria 5 para a categoria 10  
$filmes = Filme::where('categoria_id', '=', 5)->update([  
    'categoria_id' => 10  
]);
```