

# **LARAVEL**

**AULA 03**



# MODEL

O modelo representa a lógica por trás do aplicativo, que muitas vezes coincide com a camada de dados. Ou seja, costuma ser uma classe que coincide com as tabelas do banco de dados.

## Como é possível criar um modelo?

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

```
php artisan make:model NomeModelo
```

```
// app/Filme.php
```

```
namespace App;
```

```
use Illuminate\Database\Eloquent\Model;
```

```
class Filme extends Model {
```

```
    /**
```

```
    * The attributes that aren't mass assignable
```

```
    *
```

```
    * @var array
```

```
    */
```

```
    protected $guarded = [];
```

```
    /**
```

```
    * The attributes that should be mutated to dates.
```

```
    *
```

```
    * @var array
```

```
    */
```

```
    protected $dates = ['data_de_estreia'];
```

```
}
```

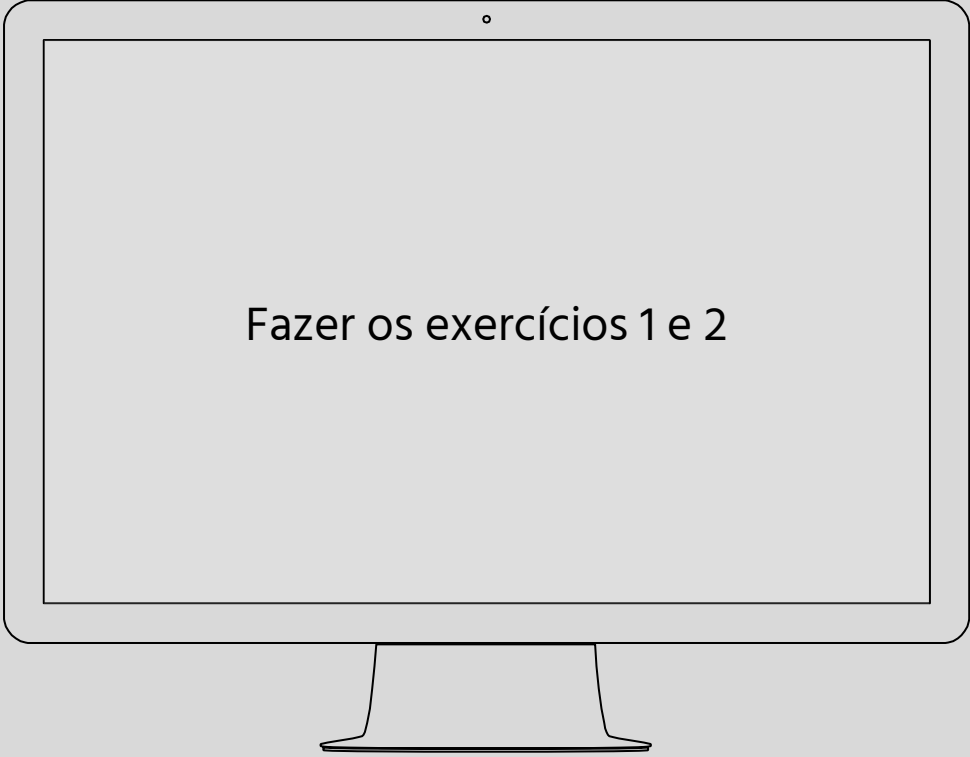
## Cuidado!

Para poder relacionar as linhas do banco de dados com objetos, Laravel utiliza determinados padrões no banco de dados:

- › Primary Key
- › Timestamps
- › Foreign keys
- › Guarded / Fillable attributes

Se o padrão não for seguido, é possível indicar isso no modelo.

**HORA DE PRATICAR!**



Fazer os exercícios 1 e 2



# ORM

Os ORMs (Object Relational Mappers) relacionam as linhas do banco de dados a objetos concretos no aplicativo. Ou seja, são responsáveis por obter os dados. O ORM de Laravel se chama **Eloquent**.

Resumindo, cada classe é correspondente a uma **tabela**. Portanto, **cada** objeto, é correspondente a uma linha dessa tabela.

## CLASSE - PHP

```
Class Filme {  
    private $id;  
    private $nome;  
    private $rating;  
    private $data_de_estreia;  
}
```

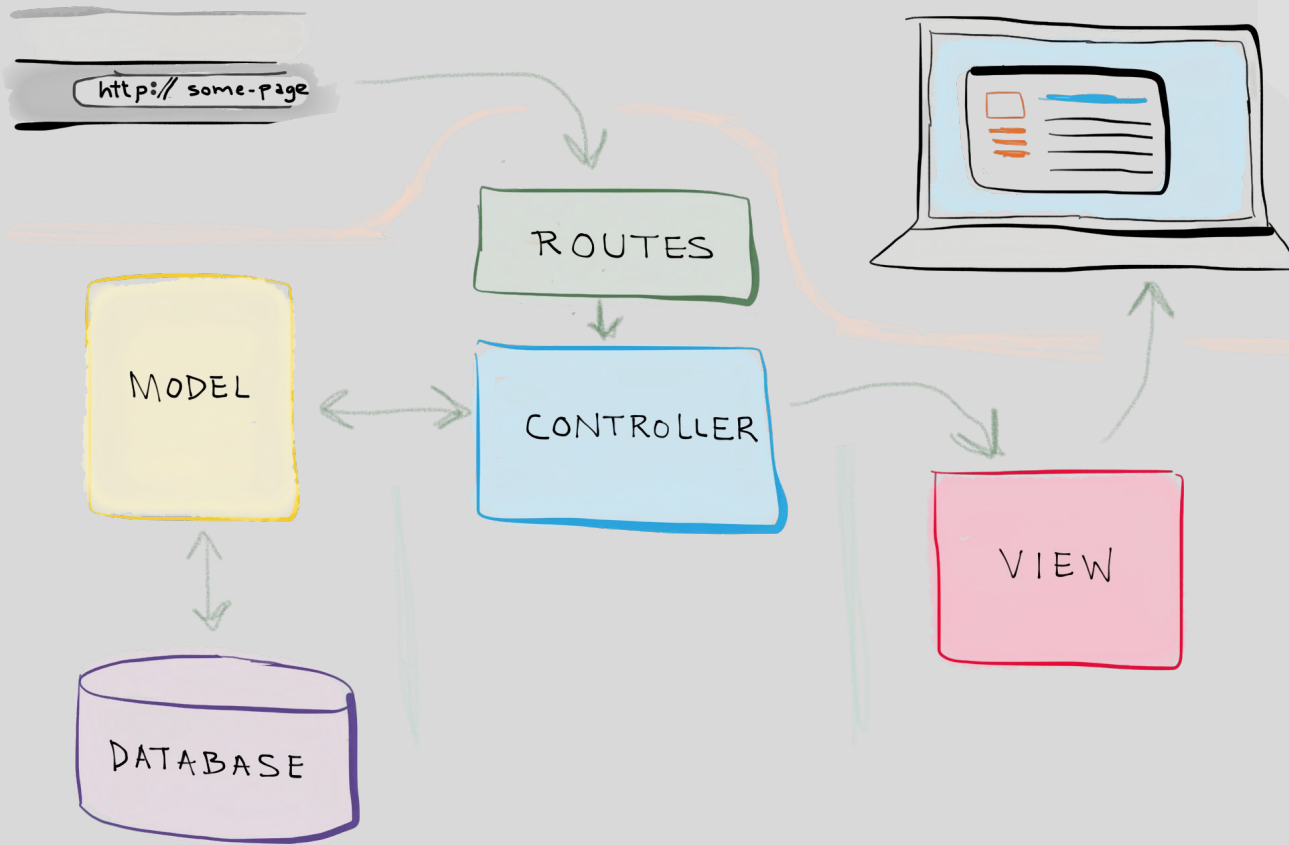
```
$filme = new Filme(1, "Toy Story",  
10, "14-03-1992");
```

## BANCO DE DADOS - SQL

Filme			
id	nome	rating	data_de_estreia
1	Toy Story	10	14-03-1995







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

# **Métodos Eloquent**

Métodos básicos de Eloquent

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

```
// Procura todos os filmes
```

```
$filmes = App\Filme::all();
```

```
// Procura um filme por ID
```

```
$filme = App\Filme::find(1);
```

```
// Procura o primeiro ou o último resultado
```

```
$filme = App\Filme::first();
```

```
$filme = App\Filme::last();
```

```
// Procura filmes por nome
```

```
$filmes = App\Filme::where('nome', 'Divertida Mente')->get();
```

```
$filmes = App\Filme::where('rating', '>', '8')->get();
```

// É possível montar queries com qualquer complexidade necessária

```
$filmes = App\Filme::where('nome', 'LIKE', 'Matrix%')  
    ->where('data_de_estreia', '<=', new DateTime('2001-02-01'))  
    ->orWhere('rating', '>', 4)  
    ->orderBy('rating', 'DESC')  
    ->take(5)  
    ->get();
```

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

# Query Builder

Gerador de consultas SQL

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

// Obter os dados de uma tabela

```
$user = DB::table('users')->get();
```

// Aplicar a cláusula select ou distinct

```
$users = DB::table('users')->select('name')->get();
```

```
$users = DB::table('users')->select('name as user_name')->get();
```

```
$users = DB::table('users')->distinct()->get();
```

// Aplicar o operador Where

```
$users = DB::table('users')->where('votes', '>', 100)->get();
```

```
$users = DB::table('users')->where('votes', '>', 100)->orWhere('name', 'John')->get();
```

```
$users = DB::table('users')->whereBetween('votes', array(1, 100))->get();
```

```
$users = DB::table('users')->whereIn('id', array(1, 2, 3))->get();
```

```
$users = DB::table('users')->whereNotIn('id', array(1, 2, 3))->get();
```

```
$users = DB::table('users')->whereNull('updated_at')->get();
```

// Aplicar order by, group by e having

```
$users = DB::table('users')  
    ->orderBy('name', 'desc')  
    ->groupBy('count')  
    ->having('count', '>', 100)  
    ->get();
```

// Aplicar offset e limit

```
$users = DB::table('users')->skip(10)->take(5)->get();
```

// Aplicar join

```
$users = DB::table('users')  
    ->join('contacts', 'users.id', '=', 'contacts.user_id')  
    ->join('orders', 'users.id', '=', 'orders.user_id')  
    ->select('users.id', 'contacts.phone', 'orders.price')  
    ->get();
```

```
$users = DB::table('users')  
    ->leftJoin('posts', 'users.id', '=', 'posts.user_id')  
    ->get();
```

## Cuidado!

Lembre que, quando usamos **Query Builder**, sempre terminamos com um método **->get()** // **->first()** // **->value()**



**HORA DE PRATICAR!**



Fazer os exercícios 3, 4 e 5

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!**