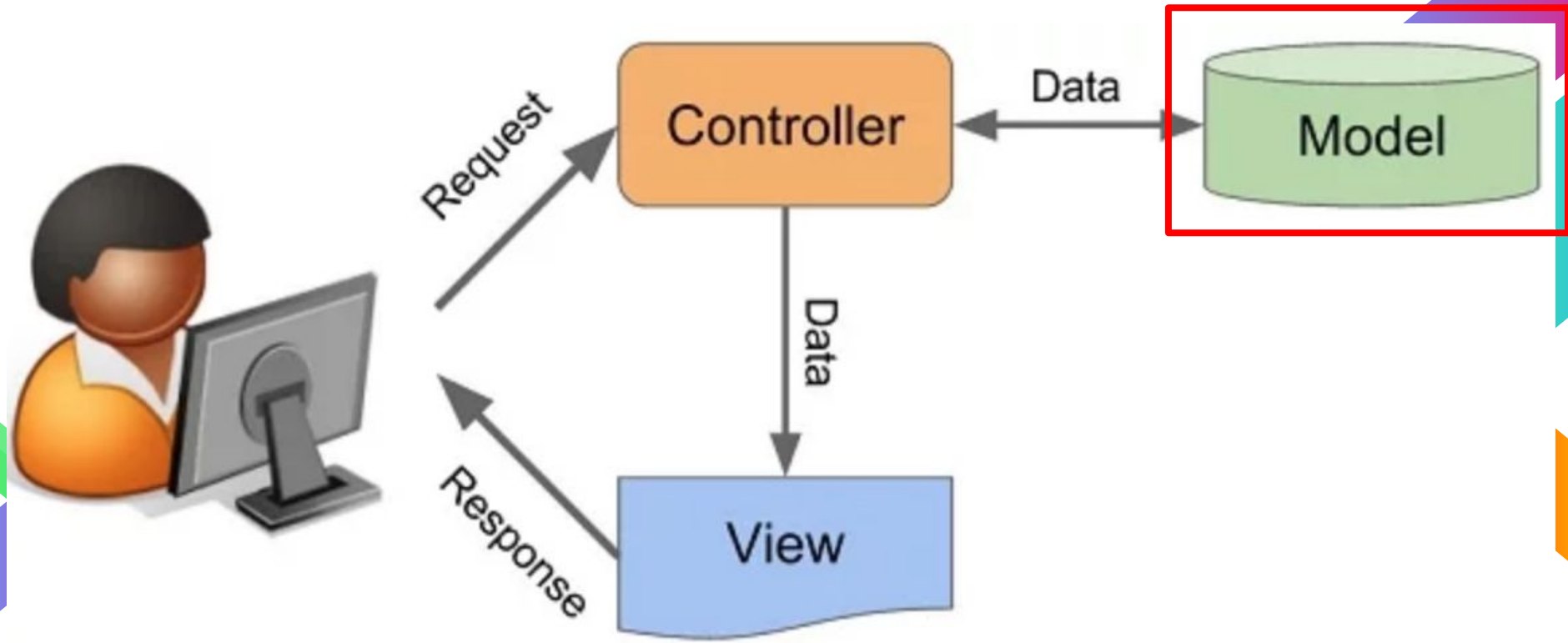
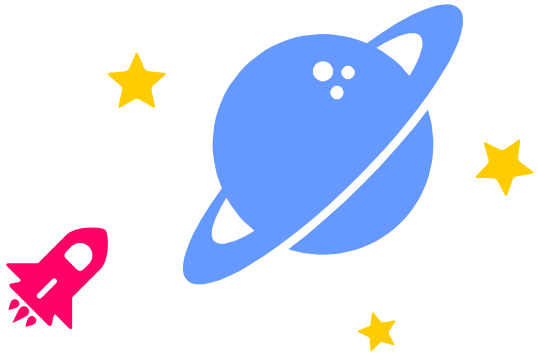
LARAVEL

CLASE 03


Request
Controller
Data
Model
Data
Response
View



MODEL

El modelo representa la lógica por debajo de nuestra aplicación que muchas veces se condice con nuestra capa de datos. Dicho de otra manera, suelen ser clases que se condicen con nuestras tablas en la base de datos.

¿Cómo creamos un modelo?

Desde la consola de comandos, ejecutamos el siguiente código:

```
php artisan make:model NombreModelo
```

Por ejemplo:

```
php artisan make:model Movie
```

```php
// app/Movie.php

    namespace App;

    use Illuminate\Database\Eloquent\Model;

    class Movie 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 = [‘relase_date’];

    }
```

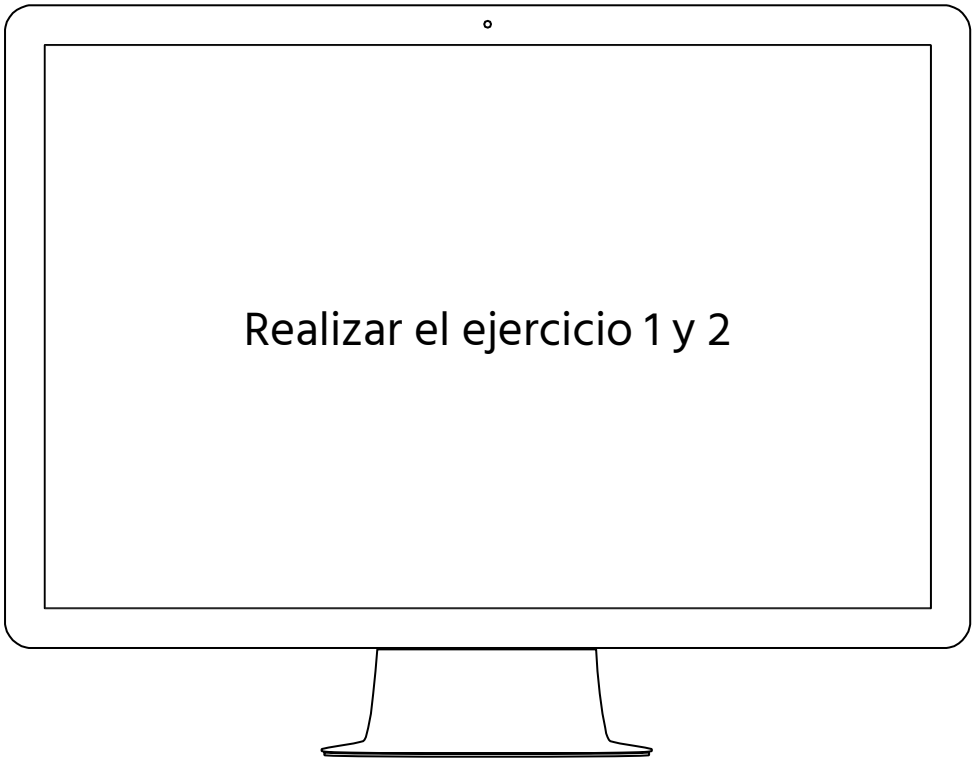
¡Cuidado!

Para poder relacionar las filas de la base de datos con objetos, Laravel asume ciertos estándares en la base de datos:

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

En caso de no seguir el estándar, se puede aclarar en el modelo.

ES MOMENTO DE
PRACTICAR !



ORM

Un ORM (Object Relational Mappers) relaciona cada una de las filas de nuestra base de datos con objetos concretos en nuestra aplicación. Es decir, es el encargado de obtener los datos. El ORM de Laravel se llama **Eloquent**.

En resumen, cada clase corresponde con una tabla Por ende, cada objeto, se corresponderá con una fila de esa tabla.

CLASE - PHP

```
Class Movie {
     private $id;
     private $title;
     private $rating;
     private $fecha_de_estreno;
}

$peli = new Movie(1, “Toy Story”,
10, “14-03-1995”);
```

BASE DE DATOS - SQL

Movie

id	title	rating	fecha_de_estreno
1	Toy Story	10	14-03-1995

Metodos Eloquent

Métodos básicos de Eloquent

CÓMO ACCEDER AL MODELO App\Movie DESDE EL CONTROLADOR:

```
//  Busco todas las películas

    $movies = App\Movie::all();

//  Busco una película por ID

    $movie = App\Movie::find(1);

//  Busco el primer o último resultado

    $movie = App\Movie::first();
    $movie = App\Movie::last();

//  Busco películas por title

    $movies = App\Movie::where('title', 'Intensamente')->get();
    $movies = App\Movie::where('rating', '>', '8')->get();
```

```
//  Podemos armar queries tan complejas como necesitemos


    $movies = App\Movie::where('title', 'LIKE', 'Matrix%')
        ->where('release_date', '<=', new DateTime('2001-02-01'))
        ->orWhere('rating', '>', 4)
        ->orderBy('rating', 'DESC') //ordenamos por rating de mayor a menor
        ->take(5) //toma los primeros 5
        ->get(); //obtenemos el resultado
```

```php
//  Usándolo en Controladores
namespace App\Http\Controllers
Class MoviesController extends Controller {
    public function index(){
        //así me traigo todas las movies de la tabla.
        $movies = \App\Movie::all();

        //usualmente esos datos los envío a una vista, de esta manera
        return view('movies.index')->with('movies', $movies);
    }
    public function show( $id ){
        //así me traigo solo la peli por el $id que recibió la ruta.
        $movie = \App\Movie::find($id);

        //usualmente esos datos los envío a una vista, de esta manera
        return view('movies.show')->with('movie', $movie);
    }
}
```

Pero vemos que estamos usando \App\Movie, eso lo podemos solucionar. ------>

```php
//  Usándolo en Controladores
namespace App\Http\Controllers

use \App\Movie;  //aquí incluyo practicamente al modelo y lo puedo usar

Class MoviesController extends Controller {

    public function index(){
        //así me traigo todas las movies de la tabla.
        $movies = Movie::all();
        //usualmente esos datos los envío a una vista, de esta manera
        return view('movies.index')->with('movies', $movies);
    }
    public function show( $id ){
        //así me traigo solo la peli por el $id que recibió la ruta.
        $movie = Movie::find($id);
        //usualmente esos datos los envío a una vista, de esta manera
        return view('movies.show')->with('movie', $movie);
    }
}
```

Tips para COMPOSER

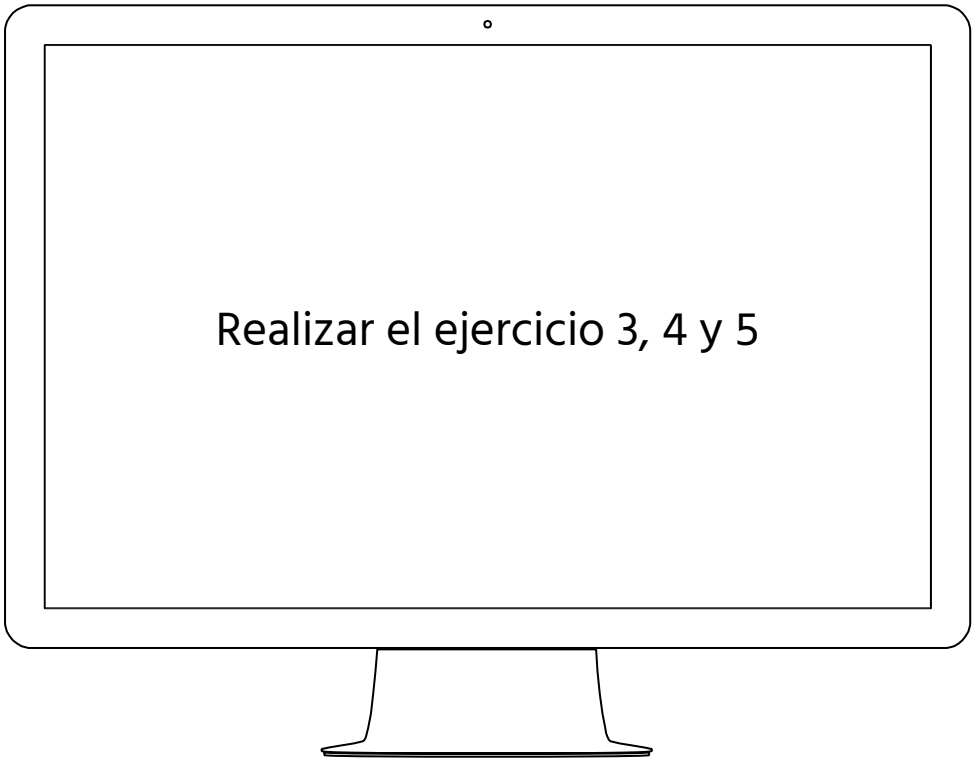
Cambiamos el nombre de una clase sea un Modelo o Controlador

Debemos ejecutar en la terminal:

```
composer dump-autload
```

Esto es porque composer almacena un repositorio de donde están todas la clases y las indexa, para luego buscarlas más facil.

ES MOMENTO DE
PRACTICAR !



¡ HASTA LA PROXIMA !