LARAVEL

CLASE 03

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
```

```php
// app/Pelicula.php

    namespace App;

    use Illuminate\Database\Eloquent\Model;

    class Pelicula 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 = [‘fecha_de_estreno’];

    }
```

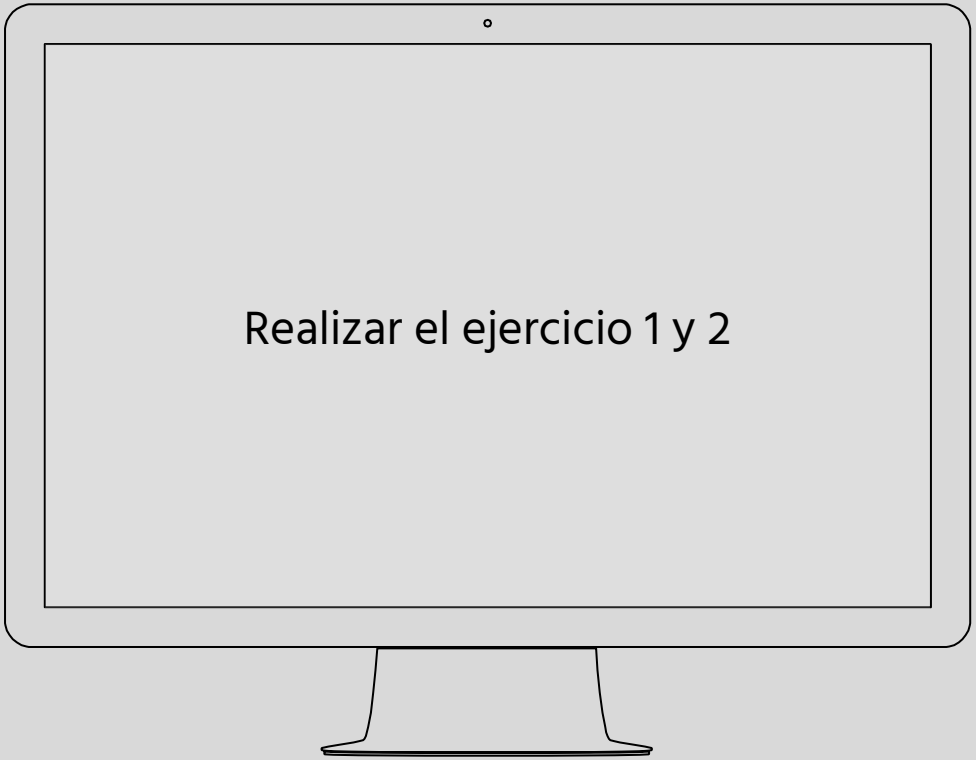
¡Cuidado!

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

- Table Name
- 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.

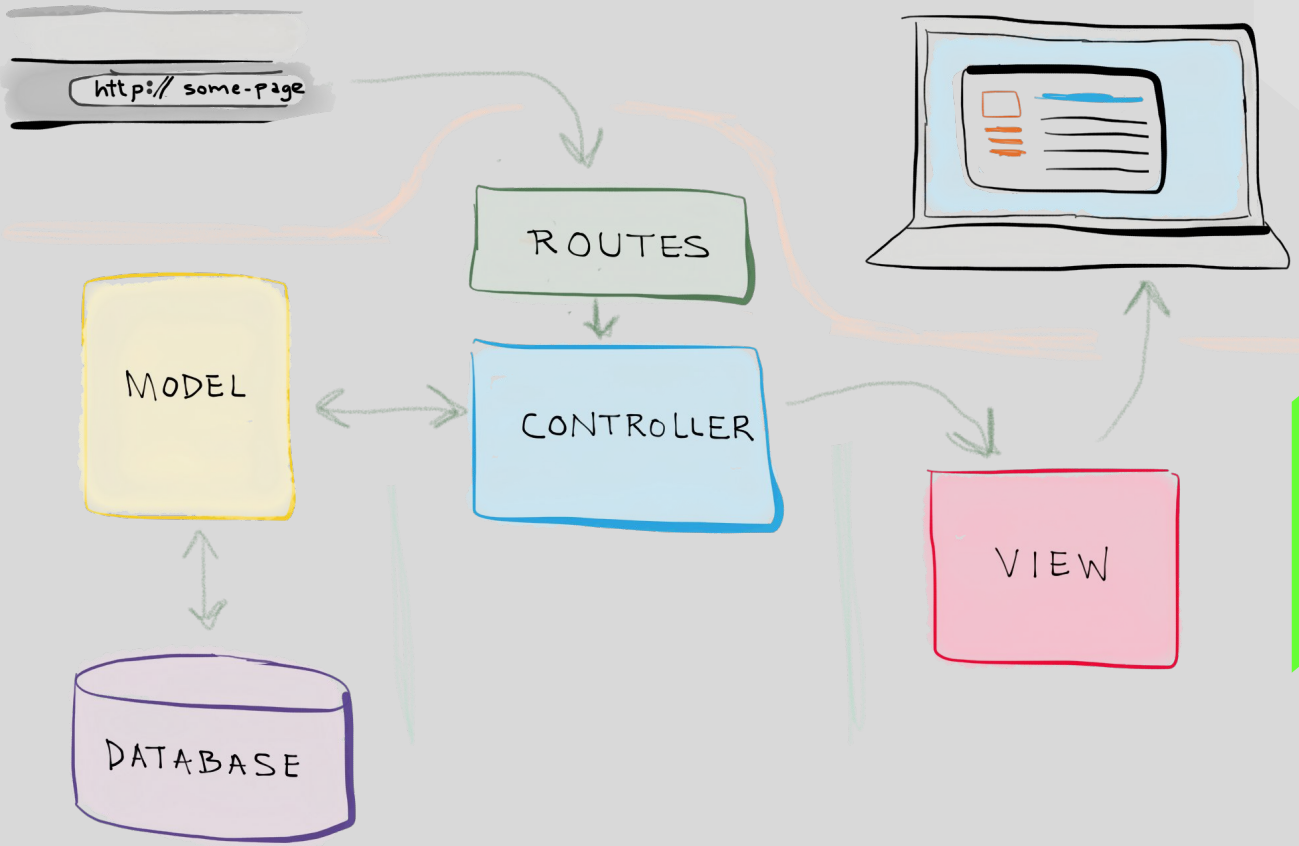
CLASE - PHP

```php
Class Pelicula {
    private $id;
    private $titulo;
    private $rating;
    private $fecha_de_estreno;
}

$peli = new Pelicula(1, “Toy Story”, 10, “14-03-1992”);
```

BASE DE DATOS - SQL

Pelicula

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


http:// some-page
ROUTES
MODEL
CONTROLLER
VIEW
DATABASE

Metodos Eloquent

Métodos básicos de Eloquent

```php
//   Busco todas las películas

     $peliculas = App\Pelicula::all();

//   Busco una película por ID

     $pelicula = App\Pelicula::find(1);

//   Busco el primer o último resultado

     $pelicula = App\Pelicula::first();
     $pelicula = App\Pelicula::last();

//   Busco películas por título

     $peliculas = App\Pelicula::where('titulo', 'Intensa Mente')->get();
     $peliculas = App\Pelicula::where('rating', '>', '8')->get();
```

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


     $peliculas = App\Pelicula::where('titulo', 'LIKE', 'Matrix%')
          ->where('fecha_de_estreno', '<=', new DateTime('2001-02-01'))
          ->orWhere('rating', '>', 4)
          ->orderBy('rating', 'DESC')
          ->take(5)
          ->get();
```

Query Builder

Generador de consultas SQL

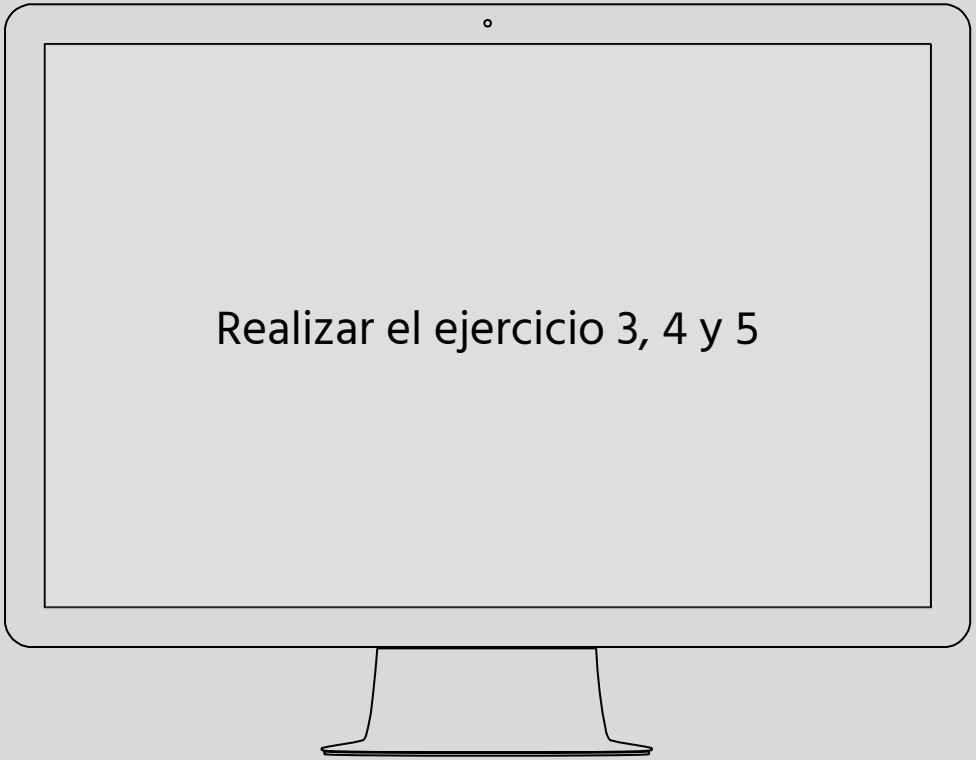
```php
//    Obtener los datos de una tabla

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

//    Aplicar la clausula select o 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 el 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();
```

```php
//   Aplicar order by, group by, y having

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

//   Aplicar offset, y 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!

Nótese que al utilizar el **Query Builder** siempre finalizamos con un método ->**get() // ->first() // ->value()**

ES MOMENTO DE
PRACTICAR !



¡ HASTA LA PROXIMA !