**Uso de Eloquent, el ORM de Laravel**

Contenido

[Razones para utilizar Eloquent 2](#_Toc128573704)

[Los modelos 2](#_Toc128573705)

[Operaciones CRUD con Eloquent 4](#_Toc128573706)

[Inserciones 4](#_Toc128573707)

[Actualizaciones 4](#_Toc128573708)

[Borrado 4](#_Toc128573709)

[Consultas con where 5](#_Toc128573710)

[Con comodines 5](#_Toc128573711)

[Consultas con whereLike 5](#_Toc128573712)

[Múltiples consultas en el mismo query 5](#_Toc128573713)

[Uso de orderBy 6](#_Toc128573714)

[Uso de groupBy 6](#_Toc128573715)

[Uso de first 6](#_Toc128573716)

[Uso de when 6](#_Toc128573717)

[Uso de withSum( ) 7](#_Toc128573718)

[Uso de withCount( ) 7](#_Toc128573719)

## Razones para utilizar Eloquent

En las aplicaciones se puede interactuar con la información de las bases de datos, directamente con el SQL del manejador de base de datos, o bien con el ORM (Object Relational Mapping) de los frameworks de desarrollo.

Con la finalidad de hacer aplicaciones que tiendan a la portabilidad, una de las características que nos harán ganar la portabilidad, es la independencia respecto al manejador de base de datos. Por ello es recomendable utilizar el ORM que tiene Laravel y que se llama Eloquent. Al menos la aplicación sería portable en la interacción con los datos.

En este texto, pondré ejemplos de las interacciones más importantes que se pueden lograr con Eloquent, a fin de tener un documento que concentre los ejemplos más utilizados, para consulta durante el desarrollo.

## Los modelos

Antes de continuar, es importante recordar que previamente ya debemos contar con nuestro esquema de base de datos construido y con los modelos que tendrán la interacción con cada relación o tabla de nuestro esquema de base de datos.

Recordemos que para crear un modelo, debemos ejecutar la directiva:

php make:model nombre\_del\_modelo

Esta directiva nos creará un archivo dentro de la carpeta app/Models de nuestro proyecto de Laravel y en nuestro modelo, debemos indicar el nombre de la tabla a la que corresponde el modelo y un arreglo con los campos “fillable”, que permitirán cualquier consulta o inserción de datos de y hacia la tabla indicada en el modelo.

A continuación, la imagen 1 muestra un ejemplo de un modelo, que posteriormente deberá ser incluido en el componente de Livewire que lo vaya a consultar, o bien, dentro del controlador de Laravel que lo vaya a utilizar.

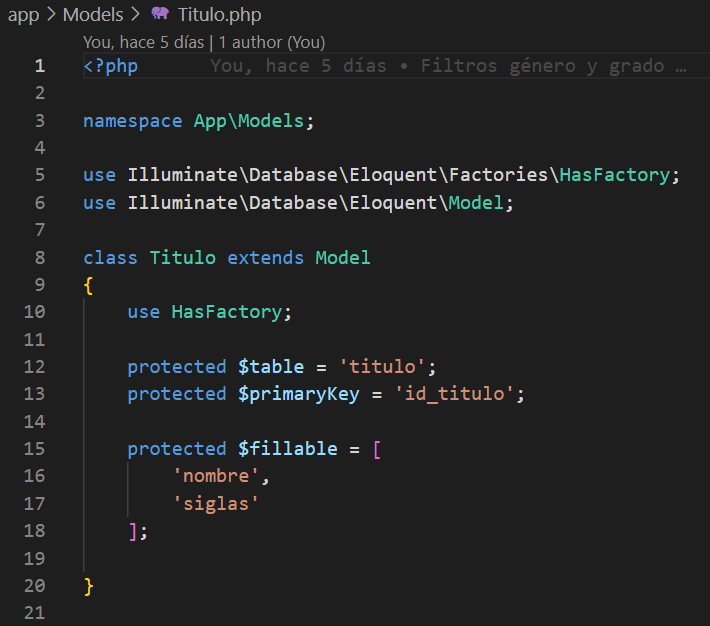


Imagen 1.

## Operaciones CRUD con Eloquent

Como sabemos, las principales operaciones con bases de datos son el CRUD (Create, Read, Update, Delete), que corresponde a la creación/inserción de datos, lectura/consulta, actualización y borrado de datos.

### Inserciones

Para realizar inserciones de información a la tabla con Eloquent, lo haremos de la siguiente manera:

Use App\Models\Titulo;

$titulo = new Titulo;

$titulo->nombre = 'Contador Público';

$titulo->siglas = 'C.P.';

$titulo->save();

### Actualizaciones

Para hacer actualizaciones en la tabla con Eloquent, está la sintaxis siguiente:

Use App\Models\Titulo;

$titulo = Titulo::find(51);

$titulo->nombre = 'Licenciado en Contaduría Pública';

$titulo->save();

### Borrado

Para eliminar registros con Eloquent, es de la siguiente manera:

Use App\Models\Titulo;

$titulo = Titulo::find(51);

$titulo->delete();

En la parte de las consultas tenemos varios ejemplos útiles de interacción con el modelo:

### Consultas con where

Use App\Models\Titulo;

$titulos = Titulo::where('nombre','Contador Público')->get();

* Es importante poner la nota que la directiva ->get() es necesario agregarla, de lo contrario la variable $titulos quedará vacía.

### Con comodines

Use App\Models\Titulo;

$titulos = Titulo::where('nombre', 'like', '%'.$this->var.'%')

->get();

### Consultas con whereLike

Use App\Models\Titulo;

$titulos = Titulo::whereLike('nombre', $this->var)

->get();

Múltiples consultas en el mismo query, utilizando orWhere

Use App\Models\Titulo;

$titulos = Titulo::where('nombre', 'like', '%'.$this->var.'%')

->orWhere('siglas', 'like', '%'.$this->var2.'%')

->get();

### Uso de orderBy

Use App\Models\Titulo;

$titulos = Titulo::whereLike('nombre', $this->var)

-> orderBy ('nombre', 'asc')

->get();

### Uso de groupBy

Use App\Models\Titulo;

$titulos = Titulo::where('nombre', $this->var)

-> groupBy ('nombre')

->get();

Uso de first (que sólo traerá el primer registro que cumpla la condición, a diferencia de get, que nos devuelve todos los registros que cumplan con la condición).

Use App\Models\Titulo;

$titulos = Titulo::whereLike('nombre', $this->var)

-> orderBy ('nombre', 'asc')

->first();

Uso de when, con una función closure (función anónima):

Use App\Models\Titulo;

$titulos = Titulo::when($this->var, function ($query, $this->var){

$query->orderBy('siglas');

})

->get();

### Uso de withSum( )

Use App\Models\Articulo;

$articulos = Articulo::select('id', 'nombre')

->withSum('productos', 'precio')

->get()

->toArray();

El resultado será la suma de los ‘precios’ por cada producto.

### Uso de withCount( )

Use App\Models\Articulo;

$articulos = Articulo::select('id', 'nombre')

->withCount('productos')

->get()

->toArray();

El resultado será el conteo de cuántos productos hay, por cada producto diferente.