Modelos con ORM Eloquent

EL ORM (Modelo Objeto-Relacional) Eloquent viene integrado en Laravel y permite trabajar con la base de datos interactuando directamente con objetos en los que se mapean las filas de las tablas, abstrayéndonos así del sistema de base de datos que vayamos a usar.

Creación de un modelo

Para crear un modelo usamos Artisan:

```
sudo php artisan make:model Autor
```

Esto nos crea un archivo en el directorio /App/Models. Lo implementamos con el siguiente contenido:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Autor extends Model
{
    use HasFactory;
    protected $table = 'autores';     //  Indicamos  que  cree  una
clase a partir de la tabla autores
    public $timestamps = false; //   La   clase   no   incluirá   los
timestamps
}
```

Inserción de una fila

Como se ha explicado anteriormente Elquent realiza una asignación entre clases y tablas, por lo que para trabajar con las filas de una tabla tendremos que tratarlas como objetos instanciados de esas clases. El código para realizar una inserción sería:

```
use App\Models\Autor;

Route::get('/', function () {
    $autor = new Autor;    // Funciona con () y sin ellos
    $autor->aut_nombre="Miguel de Cervantes";
```

```
    $autor->save();   // Método que guarda el objeto en la tabla
    echo 'Autor guardado en la tabla';
});
```

En este caso hemos puesto el código en el archivo de rutas, pero en proyectos de mayores dimensiones deberemos insertarlo en los controladores.

Recuperar una fila por su clave primaria

El código sería así:

```
use App\Models\Autor;

Route::get('/', function () {
    $autor = Autor::find('1');
    return $autor->aut_nombre;
});
```

Pero Eloquent supone que la clave primaria se llama ‘id’, en nuestro caso sería ‘aut_id’, por lo que hay que especificárselo en la definición del modelo añadiendo la línea:

```
protected $primaryKey = 'aut_id';
```

Otras operaciones

Operación	**Código**

Actualización	`use App\Models\Autor;`
	
	`Route::get('/', function () {`
	`  $autor = Autor::find(1);`
	`  $autor->aut_nombre=”Calderón de la Barca”;`
	`});`

Eliminación	`use App\Models\Autor;`
	
	`Route::get('/', function () {`
	`  $autor = Autor::find(1);`
	`  $autor->delete();`
	`});`

Eliminación mediante clave primaria	`use App\Models\Autor;`
	
	`Route::get('/', function () {`
	`  Autor::destroy(1);`
	`});`

Eliminación múltiple mediante clave primaria	`use App\Models\Autor;`
	
	`Route::get('/', function () {`
	`    Autor::destroy([3,4]);`
	`});`

Consultas

Operación	**Código**

Todas las filas	`use App\Models\Autor;`
	
	`Route::get('/', function () {`
	`    $autores=Autor::all();`
	`    var_dump($autores); // Para ver todo el array`
	`    echo $autores[0]->aut_nombre; // Para ver un elemento`
	`});`

Primera fila	`use App\Models\Autor;`
	
	`Route::get('/', function () {`
	`    $autor=Autor::first();`
	`    var_dump($autor); // Para ver todo el array`
	`    echo $autor->aut_nombre; // Para ver un elemento`
	`});`

Actualización	`use App\Models\Autor;`
	
	`Route::get('/', function () {`
	`    Autor::where('aut_id', '=', '5')`
	`            ->update([`
	`                'aut_nombre' => 'Lope de Vega'`
	`    ]);`
	`});`

Borrado	`use App\Models\Autor;`
	
	`Route::get('/', function () {`
	`    Autor::where('aut_id', '=', '5')->delete();`
	`});`

Where (consulta)	`use App\Models\Autor;`
	
	`Route::get('/', function () {`
	`    $autores=Autor::where('aut_id', '>', '5')->get();`
	`    var_dump($autores);`
	`});`

Consultas SQL

Si queremos conseguir consultas más complejas, o directamente usar SQL en Laravel, tendremos que hacer uso de las fachadas o Facades, concretamente de \Facades\DB. Las fachadas son una especie de interfaz que tiene Laravel para proporcionarnos funciones muy potentes simplificando el código.

En primer lugar debemos poner en la parte superior del archivo:

```
use Illuminate\Support\Facades\DB;
```

En el código podremos utilizar

```
$autores = DB::select("select * from autores");
```

Esto nos dejará en autores un array de objetos, que básicamente son también arrays asociativos. Podríamos trabajar con $autores de la siguiente manera:

```
foreach ($autores as $autor) {
      echo $autor->aut_nombre;
}
```

Referencias:

https://laravel.com/docs/8.x/eloquent
https://styde.net/aprende-a-usar-eloquent-el-orm-de-laravel/
https://medium.com/@LeonardoSuarezDev/3-maneras-de-consultar-a-la-base-de-datos-desde-laravel-a560787888cb