12- Comienzo del nuevo Modelo

Habiendo terminado con la modificación de departamentos, la siguiente tarea sera la creación de un nuevo modelo llamado Cargo y relacionar los dos.

Laravel tiene diferentes tipos de relaciones entre modelos (que se corresponden a los distintos tipos de relaciones de Base de Datos) pero aquí solo trabajaremos con una en particular que sea Uno a muchos (One to Many). La usaremos en dos modelos diferentes pero en principio sera **Departamento** – **Cargo**. Esta relación podría ser debatida ya que si bien un **Departamento** tendrá varios **Cargos**cada **Cargo** solo puede pertenecer a un **Departamento**.

Se podrían hacer de muchos a muchos o sea que cada **Departamento** tenga muchos **Cargos** y cada **Cargo** pueda pertenecer a muchos **Departamentos** pero examinando el objetivo de esta serie se ha decidido por lo anterior. Uno a muchos.

Modificaremos el modelo y el controlador de **Departamento**.

Ahora que tenemos las bases podremos pasar esta etapa mas rápidamente.

Primero lo básico. Modelo y migración de **Cargo**.

```
php artisan make:model Position -m
```

La **migración** contendrá lo siguiente:

```
public function up() {
    Schema::create('positions', function (Blueprint $table) {
        $table->increments('id');
        $table->string('title');
        $table->integer('departure_id')->unsigned();
        $table->foreign('departure_id')->references('id')->on('departures');
    });
}
```

Aqui estamos marcando la relación a nivel de la base de datos. Hemos creado un campo **departure_id** que se relaciona con el **id** de **departure**. También como en la migración de **Departure** hemos eliminado el **timestamp**.

El **modelo** quedaría:

```
public $timestamps = false;

protected $fillable = [
    'title',
    'departure_id'
];

public function departure() {
    return $this->belongsTo('App\Departure');
}
```

y las modificaciones de **Departure** son:

Modelo:

```
public function positions()
{
    return $this->hasMany('App\Position');
}
```

Para las relaciones podemos usar en formato de string o la constante especial **::class** está disponible a partir de PHP 5.5.0.

De las 3 operaciones del **CRUD** solo nos toca modificar una. Tanto la creación como la modificación de **Departure** se mantienen igual pero el eliminar no.

La lógica es sencilla. Si tenemos **Cargos** dentro de ese **Departamento** y eliminamos el **Departamento** los **Cargos** quedarían «huérfanos» pero en nuestro sistema eso no esta permitido ya que cada **Cargo** debe permanecer siempre a un **Departamento**.

Tenemos dos opciones. No permitir que se elimine un **Departamento** hasta que no se eliminen los **Cargos** correspondientes o eliminar automáticamente todos los **Cargos** de ese **Departamento** junto al **Departamento.**

Este sistema usa esa segunda opción por lo que el controlador de **Departure** cambiara la función **delete** como sigue

```
public function delete($id) {
    $departure = Departure::find($id);
    $departure->positions()->delete();
    $departure->delete();
}
```

El único cambio real ha sido la segunda linea que obtiene los **Cargos** relacionados y los elimina.

Para que esto empiece a funcionar debemos hacer la migración

```
php artisan migrate
```

Solo nos queda cambiar una cosa mas para que podamos centrarnos en la migración de los **Cargos**. El controlador que nos permite tener el contenido actualizado. Así lo tenemos ahora:

```
public function allQuery(Request $request) {
    if (!$request - > ajax()) return redirect('/');
    return [
            'departures' => Departure::all()
        ];
}
```

Y lo modificaremos así:

```
public function allQuery(Request $request) {
    if (!$request->ajax()) {
        return redirect('/');
    };

    return [
        'departures' => Departure::all(),
        'positions' => Position::with('departure')->get()
    ];
}
```

y en la cabecera agregaremos por supuesto

```
use App\Position;
```

Revisemos el código nuevo

```
'positions' => Position::with('departure')->get()
```

Aquí estamos devolviendo el array positions en donde hemos guardado todos los **Cargos** incluida su relación (**with('departure')**)

Lo tendremos mas claro cuando mostremos los datos.

Hay un detalle que no hemos tocado por ahora pero parece buen momento.

Tenemos este código en la vista:

```
<div class="columns margin0 text-center vertical-center personal-menu">
    <div class="column">Empleados 0</div>
    <div class="column">Departamentos 0</div>
    <div class="column">Cargo 0</div>
</div>
```

Ahora podemos cambiar el valor de **Departamentos** para tener la cantidad actualizada. Lo haremos así:

```
 <div class="column">Departamentos @{{ departures.length }}</div>
```

Simple. Y tendremos la actualización de la cantidad automáticamente a la vista todo el tiempo.

A medida que avancemos cambiaremos los otros dos valores.

Código: Github

Siguiente Post: Preparando Cargos. La magia del Select