15- Modelo Empleado

El modelo empleado es el mas completo hasta ahora aunque comparado con modelos de la vida real no tiene tanta complejidad pero en este proyecto nos acercara a unas cuantas cosas nuevas

- Control de errores de laravel (Email)
- Control de errores personalizados
- Control de errores y mensajes relacionados en el front
- Valor automático de edad sin guardar ese dato en la db
- Manejo de relaciones a dos niveles (Position – Departure)
- Manejo en el front de select relacionados.
- Un filtro primario para uno de los select
- Un componente Vue externo para la selección de fechas.

Por estas razones volveremos al sistema paso a paso. Repasaremos el modelo y el controlador de **Position**.

Para no agobiar con muchas cosas nuevas seguramente miraremos el modelo y el controlador de empleado mas de una vez, así como modificaciones progresivas en el front pero una vez terminado este modelo daremos la serie por terminada a menos que los comentarios nos pidan seguir adelante y para seguir daremos opciones en el ultimo post de posibles caminos a tomar.

Hagamos los primeros deberes: Modelo y migración, los mismos lo podemos realizar con el siguiente comando de artisan:

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

Controlador

Como ya hemos visto anteriormente podemos crear nuestro controlador de la siguiente manera:

```
php artisan make:controller EmployeeController
```

Con esto, se han creado distintos archivos en los cuales vamos a modificar lo siguiente:

En la migración

```
public function up() {
    Schema::create('employees', function (Blueprint $table) {
        $table->increments('id');
        $table->string('name');
        $table->string('lastname');
        $table->string('email')->unique();
        $table->dateTime('birthday');
        $table->integer('position_id')->unsigned();
        $table->foreign('position_id')->references('id')->on('positions');
        $table->timestamps();
    });
}
```

Para quien halla hecho migraciones aquí no hay misterios pero miremos un detalle. Tenemos email marcado como unique. Esto quiere decir que no pueden haber dos correos electrónicos iguales. En caso se intentara agregar correos electrónicos iguales salta una excepción. Para que esto no suceda haremos una validación de ello.

Modelo

```
protected $fillable = [
    'name',
    'lastname',
    'email',
    'birthday',
    'position_id'
];

/**
 * @return \Illuminate\Database\Eloquent\Relations\BelongsTo
 */
public function position() {
    return $this - > belongsTo('App\Position');
}
```

Modelo básico y simple estableciendo la relación con position. En el modelo de Position agregamos:

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

Rutas

```
Route::post('/employee/create','EmployeeController@create')->name('employeecreate');
Route::delete('/employee/delete/{id}','EmployeeController@delete')->name('employeedelete');
Route::put('/employee/update','EmployeeController@update')->name('employeeupdate');
```

y actualicemos el controlador y lo dejamos así

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Employee;

class EmployeeController extends Controller
{
    public function create(Request $request){
        //
    }

    public function delete($id){
        //
    }

    public function update(Request $request){
        //
    }
}
```

Con las funciones vacías por ahora.

Este post lo dejaremos así porque tenemos trabajo en el modal.

Código: Github

Siguiente post: Empleado. Modificando el Modal