14- Crud Cargos

Como tenemos casi todo el código colocado solo tenemos que ir rellenado funciones para que todo encaje.

La intención no es pasar por alto nada asi que si hay dudas por favor ponerlo en los comentarios.

De todas maneras solo hablaremos de las diferencias con el modelo anterior para hacerlo mas dinamico y poder dedicarle mas tiempo al tercer modelo.

Primero que nada haremos la creación. Ya abrimos el modal correctamente con el ejemplo anterior por lo que ahora seria completar la función de create del methods y del controlador.

Por supuesto estamos hablando de createPosition que tenemos creada pero vacía y la dejaremos asi:

```
createPosition() {
    if (this.titlePosition == '') {
        this.errorTitlePosition = 1;
        return;
    }
    let me = this;
    axios.post('{{route('positioncreate')}}', {
                'title': this.titlePosition,
                'departure': this.idDeparturePosition
            })
        .then(function (response) {
            me.titlePosition = '';
            me.errorTitlePosition = 0;
            me.modalPosition = 0;
            me.idDeparturePosition = 0;
            me.closeModal();
        })
        .catch(function (error) {
            console.log(error);
        });
},
```

En comparación con el alta de Departure solo enviamos un valor mas en el cual guardamos el valor del select elegido a traves de idDeparturePosition. No hay que hacer nada mas para tener el value elegido en el select.

Cuando recibimos la respuesta positiva cereamos esa variable. Lo demás es prácticamente igual.

Pongamos el código de create del controlador

```
public function create(Request $request) {
    $position = new Position();
    $position->title = $request->title;
    $position->departure_id = $request->departure;
    $position->save();
}
```

Continuemos con un pequeño cambio en la vista. Modificaremos la tabla de Cargos agregando dos lineas

```
 <th>Departamento</th>
```

```
<td>@{{ position.departure.title }}</td>
```

Cuando traemos los datos hemos usado

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

Que hemos visto que trae el departamento relacionado con cada cargo. Por supuesto eso viene en la respuesta que asignamos al array position en el front.

Accedemos a el a traves de position.departure y así veremos el departamento relacionado. Aqui usaremos solo el titulo pero mas adelante tambien sera importante el id.

Ahora hagamos eliminar. Primero crearemos una variable mas en data

```
idPosition:0
```

Luego el openModal position / delete

```
case 'delete':
    {
        this.modalGeneral = 1;
        this.titleModal = 'Eliminacion de un Cargo';
        this.messageModal = 'Confirme';
        this.modalPosition = 3;
        this.titlePosition = data['title'];
        this.idPosition = data['id'];
        this.errorTitlePosition = 0;
        this.idDeparturePosition = data['departure']['id'];
        break;
    }
```

Antes de seguir corregiremos un **BUG** de distracción. En el select del modal cambiaremos

```
:readonly="modalPosition==3"
```

por

```
:disabled="modalPosition==3"
```

Miremos los elementos de la función. Hemos asignado a la variable nueva el id del Cargo que nos interesa, el titulo al v-model del input , y al v-model del select el id del departamento relacionado.

El código de la función destroyPosition:

```
destroyPosition() {
    let me = this;
    axios.delete('{{url('/position/delete')}}'+'/'+this.idPosition)
        .then(function (response) {
            me.titlePosition = '';
            me.errorTitlePosition = 0;
            me.modalPosition = 0;
            me.idDeparturePosition = 0;
            me.closeModal();
        })
        .catch(function (error) {
            console.log(error);
        });
},
```

Una vez mas estamos prácticamente repitiendo el proceso de eliminar de departure. En el controlador el código es sencillo

```
public function delete($id) {
    Position::find($id)->delete();
}
```

Solo nos queda update pero antes agregaremos dos variable a closeModal y quedara asi

```
closeModal() {
    this.modalGeneral = 0;
    this.titleModal = '';
    this.messageModal = '';
    this.modalDeparture = 0;
    this.modalPosition = 0;
},
```

Esto corrige un problema que no había saltado. Cerrara los botones de departure y position cuando se seleccione cancelar en el Modal

Para update modificaremos el openModal position / update

```
case 'update':
    {
        this.modalGeneral = 1;
        this.titleModal = 'Modificación del Cargo';
        this.messageModal = 'Ingrese el nuevo titulo';
        this.modalPosition = 2;
        this.titlePosition = data['title'];
        this.idPosition = data['id'];
        this.errorTitlePosition = 0;
        this.idDeparturePosition = data['departure']['id'];
        break;
    }
```

Sin sorpresas por ahora. Lo siguiente updatePostition

```
updatePosition() {
    if (this.titlePosition == '') {
        this.errorTitlePosition = 1;
        return;
    }
    let me = this;
    axios.put('{{route('positionupdate')}}', {
                'id': this.idPosition,
                'title': this.titlePosition,
                'departure': this.idDeparturePosition
            })
        .then(function (response) {
            me.titlePosition = '';
            me.errorTitlePosition = 0;
            me.modalPosition = 0;
            me.idDeparturePosition = 0;
            me.idPosition = 0;
            me.closeModal();
        })
        .catch(function (error) {
            console.log(error);
        });
},
```

Lo unico nuevo es que enviamos 3 datos. El id del cargo a modificar, el nuevo titulo y la nueva relación con departamento.

El controlador quedaria asi

```
public function update(Request $request) {
    $position = Position::find($request->id);
    $position->title = $request->title;
    $position->departure_id = $request->departure;
    $position->save();
}
```

Una vez mas no tenemos sorpresas aqui.

Modificaremos el html siguiente

```
<div class="column">Cargo 0</div>
```

por

```
<div class="column">Cargo @{{ positions.length }}</div>
```

Para tener control visual de la cantidad de cargos existentes.

En un par de post hemos hecho todo el proceso de creación de migración, modelo, controlador y CRUD de position.

Eso esta muy bien pero el siguiente modelo es el mas completo y complejo. Por supuesto tambien es el mas interesante.

Código: Github

Siguiente post: Modelo Empleado