13- Preparando Cargos. La magia del Select

En este post tenemos varias cosas por hacer y como ya hemos ido paso a paso solo nos centraremos en las diferencias con el modelo anterior (las relaciones, impedir que se creen **Cargos** si no existen **Departamentos**, y el select de **Departamentos**)

Vamos con los cambios de la vista en el html. Este es el trozo de código con el que trabajaremos por ahora

```
<div class="column" v-if="menu==2">
    <div class="columns">
        <div class="column text-center">
            <h3>Cargos</h3>
        </div>
    </div>
    <div class="columns">
        <div class="column">
            Tabla Cargos
        </div>
    </div>
</div>
```

La primer parte quedara así:

```
<div class="columns">
    <div class="column text-center">
        <h3>Cargos</h3>
    </div>
    <div class="column" v-if="departures.length">
        <a class="button is-success" @click="openModal('position','create')">Agregar Cargo</a>
    </div>
    <div class="column" v-else>
        <span class="text-danger">Debe existir un departamento por lo menos</span>
    </div>
</div>
```

Aquí usamos el **v-if** para comprobar si existen **Departamentos**. Si es así mostramos el botón de agregar **Cargo** pero sino mostramos un mensaje que avisa que debe existir un **Departamento** creado.

La otra parte del html la cambiaremos así:

```
<div class="columns">
    <div class="column">
        <div v-if="!positions.length">
            No hay Cargos
        </div>
        <table v-else class="table">
            <thead>
                <th>#</th>
                <th>Titulo</th>
                <th>Eliminar</th>
                <th>Editar</th>
            </thead>
            <tbody>
                <tr v-for="position in positions">
                    <td>@{{ position.id }}</td>
                    <td>@{{ position.title }}</td>
                    <td @click="openModal('position','delete',position)">
                        <i class="fa fa-ban" aria-hidden="true"></i>
                    </td>
                    <td @click="openModal('position','update',position)">
                        <i class="fa fa-pencil" aria-hidden="true"></i>
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
</div>
```

Estamos usando una variable que aun no hemos creado **positions**. Pero lo haremos en este mismo post.

Ahora tocaremos el modal y pondremos todos los controles que necesitamos

Agregaremos estas lineas al modal

```
<p class="control" v-if="modalPosition">
    <input class="input" placeholder="Cargo" v-model="titlePosition" :readonly="modalPosition==3">
    <select class="select" :readonly="modalPosition==3" v-model="idDeparturePosition">
        <option v-for="departure in  departures" :value="departure.id">@{{ departure.title }}</option>
    </select>
</p>
<div v-show="errorTitlePosition" class="columns text-center">
    <div class="column text-center text-danger">
        El nombre del Cargo no puede estar vacío
    </div>
</div>
```

La lógica es exactamente la misma que hemos usado antes. La única cosa nueva que tenemos aqui es el select. La parte de option usa el array **departure** para poner el valor de los **Departamentos** disponibles. El **v-for** lo hemos usado antes y es facil de entender como funciona pero el **v-model** (**idDeparturePosition**) solo lo habíamos usado en los inputs.

El **v-model** es lo que nos permitirá interactuar con el select sin tener que tocar el DOM y guardara en esa variable el value de la opción elegida. Si le asignamos un valor que coincida con algún value del select se elegirá por defecto.

Ahora agregaremos los 3 botones de aceptar debajo de los 3 que ya teníamos

```
<a class="button is-success" @click="createPosition()" v-if="modalPosition==1">Aceptar</a>
<a class="button is-success" @click="updatePosition()" v-if="modalPosition==2">Aceptar</a>
<a class="button is-success" @click="destroyPosition()" v-if="modalPosition==3">Aceptar</a>
```

Agregaremos las variables que hemos usado sin crear y usaremos comentarios para tenerlas mas claras. El data queda asi:

```
data: {
    menu: 0,
    modalGeneral: 0,
    titleModal: '',
    messageModal: '',
    /***** Departure *****/
    modalDeparture: 0,
    titleDeparture: '',
    errorTitleDeparture: 0,
    departures: [],
    /********* Position **********/
    positions: [],
    modalPosition: 0,
    titlePosition: '',
    errorTitlePosition: 0,
    idDeparturePosition: 0
},
```

y crearemos 3 funciones vacías por ahora en **methods**

```
updatePosition() {},
destroyPosition() {},
createPosition() {},
```

y si pondremos código en **openModal position** / **create**

```
case "position":
    {
        switch (action) {
            case 'create':
                {
                    this.modalGeneral = 1;
                    this.titleModal = 'Creación de Cargo';
                    this.messageModal = 'Ingrese el titulo del Cargo';
                    this.modalPosition = 1;
                    this.titlePosition = '';
                    this.errorTitlePosition = 0;
                    this.idDeparturePosition = this.departures[0].id;
                    break;
                }
```

Lo interesante aqui es

```
this.idDeparturePosition = this.departures[0].id;
```

Ya hemos dicho que **idDeparturePosition** es la variable que controla el select. Como estamos creando lo que hacemos aqui es asignar el id del primer **Departamento** que tenemos en el array para que aparezca como seleccionado por defecto.

Ya que estamos dentro de Vue modificamos el valor de la respuesta automática para que quede asi:

```
allQuery() {
    let me = this;
    axios.get('{{route('allQuery')}}')
        .then(function (response) {
            let answer = response.data;
            me.departures = answer.departures;
            me.positions = answer.positions;
        })
        .catch(function (error) {
            console.log(error);
        });
},
```

En realidad solo hemos agregado

```
me.positions = answer.positions;
```

Para completar los preparativos podemos crear el controlador:

```
php artisan make:controller PositionController
```

y lo modificamos colocando el use y las funciones vacias para que quede asi

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Position;
class PositionController extends Controller {
    public function create(Request $request) {

    }
    public function delete($id) {

    }

    public function update(Request $request) {

    }
}
```

y completamos este post con las rutas correspondientes:

```
Route::post('/position/create','PositionController@create')->name('positioncreate');
Route::delete('/position/delete/{id}','PositionController@delete')->name('positiondelete');
Route::put('/position/update','PositionController@update')->name('positionupdate');
```

Tenemos la plantilla preparada para poder trabajar con el CRUD directamente.

Código: Github

Proximo post: CRUD cargos