6- Creando el modal parte I

Capitulo Anterior

Es este post trataremos de cerca una de las particularidades de **Bulma** como frameworks css.

Al ser independiente de cualquier js (muchos como bootstrap dependen de librerías js como jQuery para funcionar correctamente) hay cosas que no puede hacerse solo con css.

Bulma nos da un esquema perfecto para hacer modals pero no tiene una manera de mostrarlos y ocultarlos ya que eso depende de una acción de javascript. Sin embargo ambos estados se controlan con una clase simple (**is-active**) dejando en nuestra mano colocar o quitar la clase para mostrar o ocultar el Modal.

Antes que nada deberemos modificar el **section content.**

Donde teníamos

```
</div>
@endsection
```

entre el último **div** y el **@endsection** agregaremos el código del modal

```
 <div class="modal" :class="{'is-active' : modalGeneral}">
      <div class="modal-background"></div>
      <div class="modal-content">
           <div class="content">
                <h3 class="text-center">@{{titleModal}}</h3>
                <div class="field">
                     <label class="label">@{{messageModal}}</label>
                     <p class="control" v-if="modalDeparture!=0">
                     <input class="input" placeholder="Departamento" v-model="titleDeparture"
                               v-if="modalDeparture==1">
                    </p>
                    <div v-show="errorTitleDeparture" class="columns text-center">
                        <div class="column text-center text-danger">
                            El nombre del Departamento no puede estar vacio
                        </div>
                    </div>
                    <div class="columns button-content">
                        <div class="column">
                            <a class="button is-success" @click="createDeparture()" v-
if="modalDeparture==1">Aceptar</a>
                        </div>
                        <div class="column">
                            <a class="button is-danger" @click="closeModal()">Cancelar</a>
                        </div>
                    </div>
                </div>
            </div>
            <button class="modal-close" @click="closeModal()"></button>
        </div>
</div>
```

Miremos el código paso a paso.

La primer línea contiene todo el modal

```
<div class="modal" :class="{'is-active' : modalGeneral}">
```

Esto indica que cuando la variable **modalGeneral** tenga una valor **true** (cualquier valor mayor que cero) se le asignará la clase **is-active** al modal y por lo tanto el modal se mostrará.

La siguiente línea

```
<h3 class="text-center">@{{titleModal}}</h3>
```

Indica que en ese sitio se mostrará el contenido de la variable **titleModal.** Observemos un detalle interesante el uso de **@** delante de **{{}}** esto es para evitar que **blade** procese **titleModal** como un valor php

Un código similar pero con otra variable lo tenemos unas lineas mas abajo

```
<label class="label">@{{messageModal}}</label>
```

La primer variable la usaremos para personalizar el titulo y este otro para personalizar el mensaje antes de abrir el modal en cada situación.

La idea principal es usar un unico modal para todo el CRUD del front y así podremos indicar al usuario en que proceso esta en ese momento.

Examinemos el código siguiente

```
<p class="control" v-if="modalDeparture!=0">
   <input class="input" placeholder="Departamento" v-model="titleDeparture"
            v-if="modalDeparture==1">
</p>
```

Aquí tenemos varias cosas. Primero que nada estamos mostrando ese contenido siempre y cuando **modalDeparture** no sea **0**. Esto ahora mismo es poco coherente pero cuando hagamos el resto de CRUD's tendremos esta forma de distinguir los controles para **Departamento**, **Cargo** y **Empleado**.

v-model nos permite asignar una variable a un **<input>** de manera que el contenido se modifique dinamicamente a medida que el usuario escribe en el input.

También usamos un **v-if** con la misma variable que usamos antes para mostrar el valor en este caso. Pero esta vez verificamos que el valor sea 1. Usaremos valor 1 para crear, 2 para actualizar, 3 para borrar. En cada caso mostraremos los inputs o botones que necesitemos. Podemos dejar algunas dudas aqui ya que lo tendremos mas claro cuando avancemos en las demás opciones del CRUD.

Lo siguiente

```
<div v-show="errorTitleDeparture" class="columns text-center">
    <div class="column text-center text-danger">
         El nombre del Departamento no puede estar vacio
    </div>
</div>
```

En este apartado controlamos un mensaje de error con la ya repetida formula del **v-if**. Las variables usadas aqui para error serán lo mas semánticas posibles. **errorTitleDeparture** se explica sola.

Sigamos viendo código

```
<a class="button is-success" @click="createDeparture()" v-if="modalDeparture==1">Aceptar</a>
```

Poco a poco todo esto se nos esta haciendo familiar. Repasemos. **@onclick** es una llamada a una función (Aqui estará el código del primer uso que haremos de **Axios**). El **v-if** sigue siendo de la variable que ya usamos una vez mas asignada a un valor 1. En estos botones es cuando mas aprovecharemos los diferentes valores que asignaremos a **modalDeparture** (paciencia, lo veremos pronto muy claro)

Y finalmente un codigo que no tendra condicionante y que siempre aparecera en cada modal

```
<a class="button is-danger" @click="closeModal()">Cancelar</a>
```

Y

```
<button class="modal-close" @click="closeModal()"></button>
```

Ambas opciones llaman a la misma función. La opción de cerrar el modal.

El modal creado ahora así nos dara una serie de errores. Para que eso no suceda crearemos las variables que nos faltan en Vue y agregaremos las funciones vacías y nos quedaria así el **section script**

```
let elemento = new Vue({
    el: '.app',
    data: {
        menu: 0,
        modalGeneral: 0,
        titleModal: '',
        messageModal: '',
        modalDeparture: 0,
        titleDeparture: '',
        errorTitleDeparture: 0
    },
    methods: {
        closeModal() {}, createDeparture() {}
    },
})
```

El post se ha hecho largo pero continuaremos.

Código: Github

En el próximo post: Creando el Modal parte II