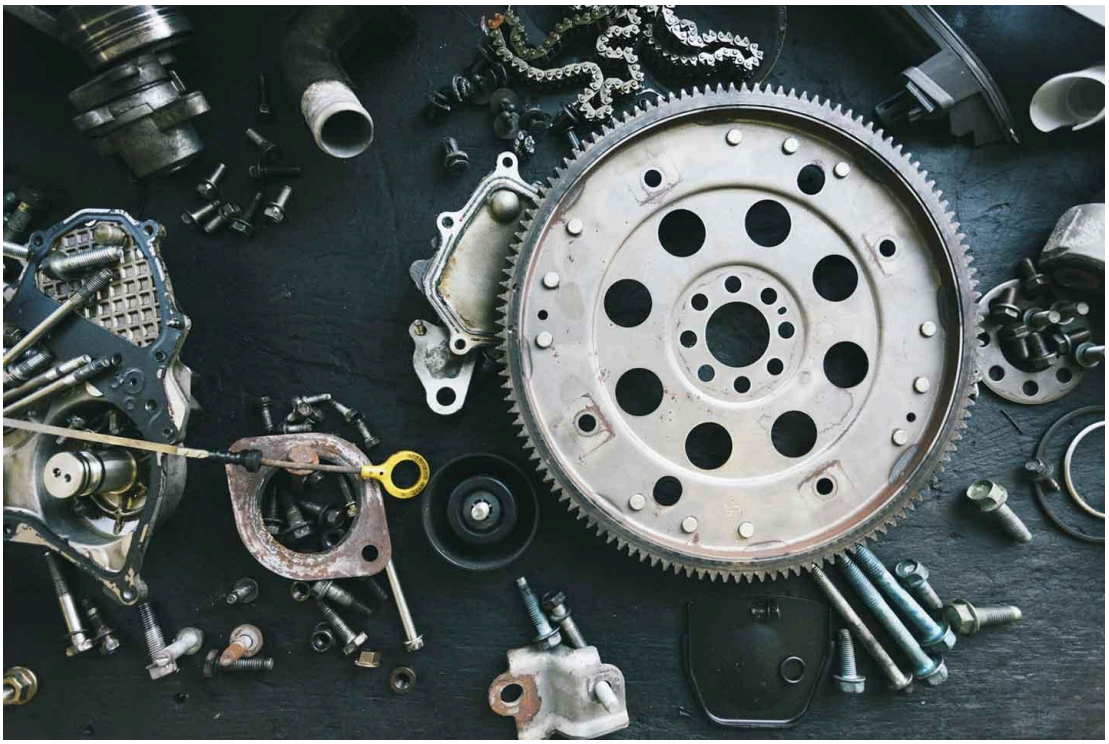
Ionic v8

Modal



Progresa
Una formación de futuro

ÍNDICE

Introducción **3**
Preparación **3**
Componente 3
Icono 3
Json 4
Ejemplo normal **4**
Visualización 4
Explicación 6
Estilos 8
Lógica 8
Ejemplo desplegable **11**
Visualización 11
Estilos 13

Introducción

Un modal es un diálogo que aparece por delante del contenido de la App, y tiene que ser cerrado por la App antes de que podamos continuar utilizando la propia App. Es útil para un componente de selección cuando hay muchas opciones de las que elegir, o cuando filtramos elementos de una lista.

Hay 2 maneras de cargar los modales:

- **Inline modals(recomendada)**: Se utiliza el ion-modal en el html, esto reduce el número de manejadores que necesitaremos para presentar el modal.
- **Controller modals(antigua)**: Se utiliza el controller en el archivo ts.

La documentación del modal está en la página:

https://ionicframework.com/docs/api/modal

Preparación

Componente

Para empezar a trabajar con el modal vamos a necesitar una página donde usarlo. Creamos la página en la **terminal**:

```
ionic g page pages/Modal --spec=false
```

Icono

Añadimos en nuestro **app.component.ts** el icono que queremos usar para el menú en la lista de iconos(addIcons).

```
addIcons({grid, logoAppleAppstore, americanFootball, beaker, star,
trash, shareSocial, caretForwardCircle, close, card, pin, car,
```

```
logoFacebook,logoTwitter,logoTiktok,logoInstagram,logoYoutube,logoGithub,add, albums, ellipsisHorizontalOutline, list, heart, call, infinite, infiniteOutline, phonePortrait});
```

JSON

Añadimos en nuestro **menu.json** el nuevo elemento que queremos que aparezca tanto en el menú como en la página de inicio.

```
,{
 "nombre": "Modal",
 "ruta": "/modal",
 "icono": "phone-portrait"
}
```

EJEMPLO NORMAL

VISUALIZACIÓN

En primer lugar vamos a mostrar en nuestro modal una serie de inputs. En nuestro **modal.page.html**:

```
<app-header titulo="Bienvenido {{usuario.nombre || ''}}">
</app-header>
<ion-content class="ion-padding">
 <h3>{{message}}</h3>
 <ion-button id="open-modal" expand="block">Open</ion-button>
 <ion-item-divider></ion-item-divider>
 <p><strong>Nombre: </strong>{{usuario.nombre}}</p>
 <p><strong>Email: </strong>{{usuario.email}}</p>
 <p><strong>Ciudad: </strong>{{usuario.ciudad}}</p>

 <ion-modal
   trigger="open-modal" (willDismiss)="onWillDismiss($event)">
```

```
    <ng-template>
      <ion-header>
        <ion-toolbar>
          <ion-buttons slot="start">
            <ion-button (click)="cancel()">Cancel</ion-button>
          </ion-buttons>
                                                <ion-title>Bienvenido
{{usuario.nombre}}!!!</ion-title>
        </ion-toolbar>
      </ion-header>
      <ion-content class="ion-padding">

                                        <form     #formulario="ngForm"
(ngSubmit)="onSubmitTemplate()">

          <ion-list>
            <ion-list-header>
              <ion-label>
                                Formulario  template:  Valid  <span
[ngClass]="{'red':          !formulario.valid,          'green':
formulario.valid}">{{  formulario.valid }}</span>
              </ion-label>
            </ion-list-header>
            <ion-item>
              <ion-input type="text"
                         name="nombre"
                         [(ngModel)]="usuario.nombre"
                         minlength="4"
                         labelPlacement="floating"
                         label="Nombre"
                         errorText="Mínimo 4 caracteres"
                         required="true"></ion-input>
            </ion-item>
            <ion-item>
              <ion-input type="email"
                         name="email"
```

```
pattern="^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2
,5})$"
                        [(ngModel)]="usuario.email"
                        label="Email"
                        labelPlacement="floating"
                        errorText="Email inválido"
                        required="true"></ion-input>
            </ion-item>
            <ion-item>
              <ion-input type="text"
                        name="ciudad"
                        [(ngModel)]="usuario.ciudad"
                        minlength="2"
                        label="Ciudad"
                        labelPlacement="floating"
                        errorText="Mínimo 2 caracteres"
                        required="true"></ion-input>
            </ion-item>
          </ion-list>
          <ion-button expand="full" type="submit"
                      [disabled]="formulario.invalid">
            Enviar
          </ion-button>
        </form>
      </ion-content>
    </ng-template>
  </ion-modal>
</ion-content>
```

Explicación

`<ion-button id="open-modal" expand="block">Open</ion-button>`

Botón que abrirá el modal.

`<ion-modal`

Etiqueta de definición del modal.

`trigger="open-modal"`

Identifica quién va a abrir el modal. En nuestro caso el botón con id “open-modal”.

`(willDismiss)="onWillDismiss($event)">`

Evento que se lanzará cuando se lance la orden de cerrar el modal.

`<ng-template>`

Plantilla que definirá el modal con estructura de una página de Ionic con ion-header e ion-content.

`<form #formulario="ngForm" (ngSubmit)="onSubmitTemplate()">`

Formulario con:

- `#formulario="ngForm"`: Identificador del formulario de tipo ngForm y con nombre formulario para poder encontrarlo desde el ts.
- `(ngSubmit)="onSubmitTemplate()"`: Función que se ejecutará cuando se envíe el formulario.

```
<span [ngClass]="{'red': !formulario.valid, 'green': formulario.valid}">{{ formulario.valid }}</span>
```

Cómo utilizar las clases de css en Angular con condicionales. Si el formulario es válido pondremos true en verde y si no es válido, false aparecerá en rojo.

`<ion-label position="floating">Nombre</ion-label>`

Nos permite poner el título del Input flotando cuando se rellene.

`[(ngModel)]="usuario.nombre"`

Uso de Template Forms de Angular para trabajar con formularios sencillos. Guardamos el dato en una variable con doble enlace de datos, si se modifica en el formulario se modifica la variable y viceversa.

```
pattern="^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$"
```

Filtro de seguridad de una dirección de email. Más información acerca de expresiones regulares (regex) en https://regexr.com/6sotl

```
<ion-button expand="full" type="submit" [disabled]="formulario.invalid">
```

Botón de tipo submit que enviará el formulario y lanzará el evento onSubmitTemplate. El botón estará deshabilitado mientras el formulario sea inválido.

Estilos

Una vez tenemos definido el html con el modal correspondiente, definimos la hoja de estilos para rojo y verde. En **modal.page.scss**:

```
.red{
 color: red;
}

.green{
 color: green;
}
```

LÓGICA

Y ya por último definimos las funciones y variables dentro de nuestro **modal.page.ts**:

```
export class ModalPage {

  /* VARIABLES */
  @ViewChild(IonModal) modal!: IonModal;
  usuario: {nombre: string; email: string; ciudad: string} =
    {
      nombre: '',
      email: '',
      ciudad: ''
    };
  message = 'Abra el modal y rellene los datos';
  /* FIN DE LAS VARIABLES */

  constructor() { }



  /* FUNCIONES */
  cancel() {
    this.modal.dismiss(null, 'cancel');
  }

  onSubmitTemplate() {
    console.log('submit');
    this.modal.dismiss(this.usuario.nombre, 'confirm');
  }
  onWillDismiss(event: Event) {
    const ev = event as CustomEvent;
    if (ev.detail.role === 'confirm') {
      this.message = `Hola, ${ev.detail.data}!`;
    }
  }
}
```

EJEMPLO DESPLEGABLE

Podemos crear un efecto de modal desplegable similar a los componentes de las aplicaciones de mapas. Para crear un modal dsplegable, debemos establecer las propiedades **breakpoints** e **initialBreakpoint** en ion-modal.

La propiedad breakpoints acepta una matriz que indica cada punto de interrupción al que la hoja puede ajustarse cuando se desliza. Una propiedad **breakpoints** de [0, 0.5, 1] indicaría que la hoja se puede deslizar para mostrar el 0% del modal, el 50% del modal y el 100% del modal. Cuando el modal se desliza al 0%, el modal se cerrará automáticamente. Tenga en cuenta que el modal no se puede cerrar con un deslizamiento si no se incluye ningún punto de interrupción 0, pero aún se puede cerrar presionando Esc o el botón de retroceso de hardware.

La propiedad **initialBreakpoint** es necesaria para que el modal de hoja sepa por qué punto de interrupción comenzar al presentarse. El valor **initialBreakpoint** también debe existir en la matriz breakpoints. Dado un valor **breakpoints** de [0, 0.5, 1], un valor **initialBreakpoint** de 0.5 sería válido ya que 0.5 está en la matriz breakpoints. Un valor **initialBreakpoint** de 0.25 no sería válido ya que 0.25 no existe en la matriz breakpoints.

La propiedad **backdropBreakpoint** se puede usar para personalizar el punto en el que el ion-backdrop comenzará a desvanecerse. Esto es útil cuando se crean interfaces que tienen contenido debajo de la hoja que debe seguir siendo interactivo. Un caso de uso común es un modal de hoja que se superpone a un mapa donde el mapa es interactivo hasta que la hoja está completamente expandida.

VISUALIZACIÓN

Vamos a añadir este nuevo modal al final de nuestro **modal.page.html**:

```
…
</ion-modal>


<ion-button id="open-modal-desplegable" expand="block">Abrir Modal
Desplegable</ion-button>


<ion-modal id="modal-desplegable" #modal
trigger="open-modal-desplegable" [initialBreakpoint]="0.25"
[breakpoints]="[0, 0.25, 0.5, 0.75]">
```

```
 <ng-template>
    <ion-content>


      <ion-toolbar>
        <ion-title>Modal</ion-title>
        <ion-buttons slot="end">
          <ion-button color="light"
(click)="modal.dismiss()">Close</ion-button>
        </ion-buttons>
      </ion-toolbar>
      <ion-searchbar placeholder="Search"
(click)="modal.setCurrentBreakpoint(0.75)"></ion-searchbar>
      <ion-list>
        <ion-item>
          <ion-avatar slot="start">
            <ion-img src="https://i.pravatar.cc/300?u=b"></ion-img>
          </ion-avatar>
          <ion-label>
            <h2>Connor Smith</h2>
            <p>Sales Rep</p>
          </ion-label>
        </ion-item>
        <ion-item>
          <ion-avatar slot="start">
            <ion-img src="https://i.pravatar.cc/300?u=a"></ion-img>
          </ion-avatar>
          <ion-label>
            <h2>Daniel Smith</h2>
            <p>Product Designer</p>
          </ion-label>
        </ion-item>
```

```
        <ion-item>
          <ion-avatar slot="start">
            <ion-img src="https://i.pravatar.cc/300?u=d"></ion-img>
          </ion-avatar>
          <ion-label>
            <h2>Greg Smith</h2>
            <p>Director of Operations</p>
          </ion-label>
        </ion-item>
        <ion-item>
          <ion-avatar slot="start">
            <ion-img src="https://i.pravatar.cc/300?u=e"></ion-img>
          </ion-avatar>
          <ion-label>
            <h2>Zoey Smith</h2>
            <p>CEO</p>
          </ion-label>
        </ion-item>
      </ion-list>
    </ion-content>
  </ng-template>
</ion-modal>
</ion-content>
```

Estilos

Y añadimos unos estilos para ver cómo podemos modificar propiedades del modal. En **modal.page.scss**:

```
.red{
  color: red;
}
```

```
.green{
 color: green;
}

#modal-desplegable {
 --border-radius: 16px;
 --box-shadow: 0 10px 15px -3px rgb(0 0 0 / 0.1), 0 4px 6px
-4px rgb(0 0 0 / 0.1);
}

#modal-desplegable::part(backdrop) {
 background: rgba(209, 213, 219);
 opacity: 1;
}

#modal-desplegable ion-toolbar {
 --background: rgb(14 116 144);
 --color: white;
}
```