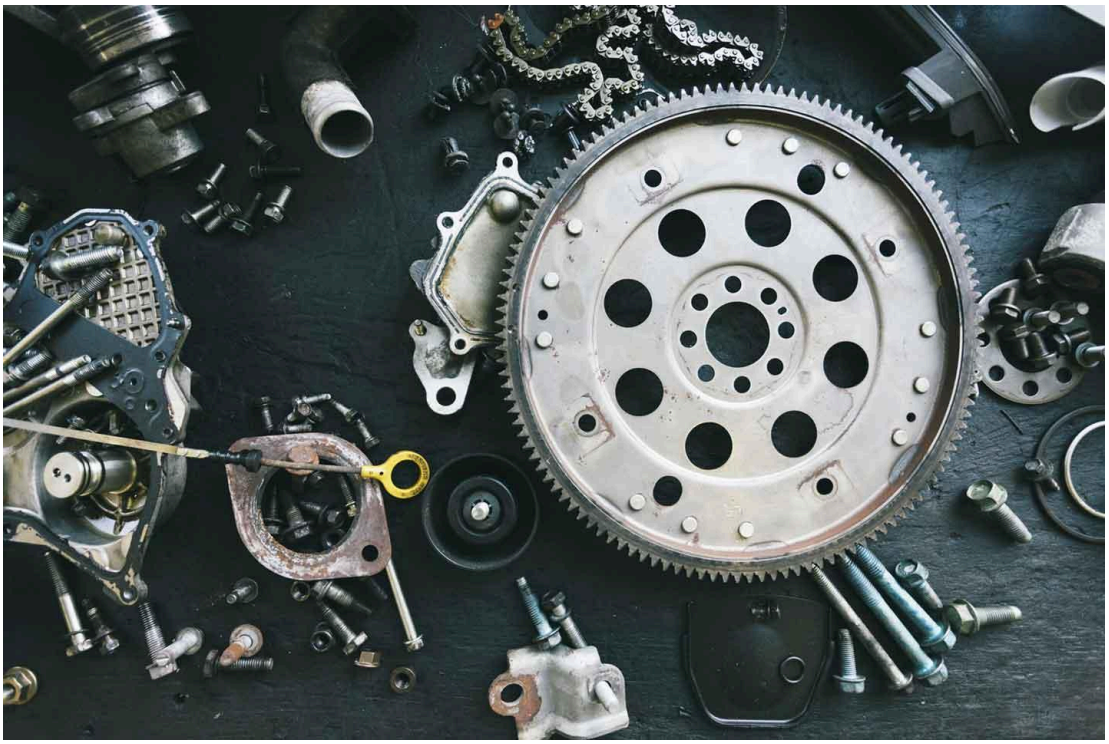
Ionic v8

Listas y Toast



Progresa
Una formación de futuro

ÍNDICE

Introducción Lista

Las listas están compuestas por múltiples filas de ítems que pueden contener texto, botones, toggles, iconos, thumbnails y mucho más. Normalmente contienen ítems con el mismo tipo de contenido, como imágenes y texto.

Las listas soportan interacciones como el desplazamiento lateral para revelar opciones o el reordenamiento de elementos de la lista.

La documentación de las listas está en la página:

https://ionicframework.com/docs/angular/list

Preparación

Componente

Para empezar a trabajar con las listas vamos a necesitar una página donde usarlas. Creamos la página en la **terminal**:

ionic g page pages/list --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, ellipsis-horizontal-outline, list});
```

JSON

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

```
,{
 "nombre": "Lists",
 "ruta": "/list",
 "icono": "list"
}
```

EJEMPLO

API

Vamos a empezar añadiendo una conexión a la API de nombres que nos proporciona jsonplaceholder. Vamos a:

https://jsonplaceholder.typicode.com/users

Esto nos proporciona un json con información de usuarios. Vamos a conectarnos desde nuestro servicio a esta falsa API. En **data.service.ts** añadimos la función para traer los usuarios:

```
export class DataService {

 constructor(private http: HttpClient) { }

 getComponentesMenu(): Observable<Componente[]> {
                                                    return
this.http.get<Componente[]>('/assets/data/menu.json');
 }

 getUsers(): Observable<Usuario[]> {
   return this.http.get<Usuario[]>(
     'https://jsonplaceholder.typicode.com/users');
```

```
  }
}
```

INTERFAZ

Y creamos la interfaz correspondiente a Usuario en nuestro archivo **interfaces.ts** dentro de la carpeta common:

```
export interface Usuario{
 id: number;
 name: string;
 username: string;
 email: string;
 address: {
   street: string;
   suite: string;
   city: string;
   zipcode: string;
   geo: {
     lat: number;
     lng: number;
   };
 };
 phone: string;
 website: string;
 company: {
   name: string;
   catchPhrase: string;
   bs: string;
 };
}
```

GUARDAR DATOS

El paso siguiente es recoger y almacenar la información que traemos de la API a través del servicio. Esto lo hacemos en el archivo Typescript de nuestra página list. En list.page.ts:

```
import {Component, inject, OnInit, ViewChild} from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import {
  IonAvatar, IonContent, IonHeader, IonIcon, IonImg, IonItem, IonItemOption, IonItemOptions, IonItemSliding, IonLabel, IonList, IonListHeader, IonReorder, IonReorderGroup, IonTitle, IonToggle, IonToolbar} from '@ionic/angular/standalone';
import {DataService} from "../../services/data.service";
import {Usuario} from "../../common/interfaces";
import {HeaderComponent} from "../../components/header/header.component";

@Component({
 selector: 'app-list',
 templateUrl: './list.page.html',
 styleUrls: ['./list.page.scss'],
 standalone: true,
   imports: [IonContent, IonHeader, IonTitle, IonToolbar, CommonModule, FormsModule, HeaderComponent, IonList, IonListHeader, IonItem, IonAvatar, IonImg, IonLabel, IonItemSliding, IonItemOptions, IonItemOption, IonIcon, IonToggle, IonReorderGroup, IonReorder]
})

export class ListPage implements OnInit {

 @ViewChild('lista', {static: false}) lista!: IonList;
 private dataService: DataService = inject(DataService);
 usuarios: Usuario[] =  [];
 habilitado: boolean = false;
```

```
  constructor() { }

  ngOnInit() {
    this.loadUsers();
  }

  call(user: Usuario) {
    console.log(user);
    this.lista.closeSlidingItems();
  }

  share(user: Usuario) {
    console.log(user);
    this.lista.closeSlidingItems();
  }

  favorite(user: Usuario) {
    console.log(user);
    this.lista.closeSlidingItems();
  }


  reordenar(event: any) {
    console.log(event);
      // En primer lugar recogemos el elemento que ha
seleccionado el usuario y lo guardo en otro array
                              const          itemMover          =
this.usuarios.splice(event.detail.from,1)[0];
    // Ahora insertamos el elemento que teníamos en el array
auxiliar para guardarlo en nuestro array
    this.usuarios.splice(event.detail.to, 0, itemMover);
    // Por último hacemos efectivo el cambio
    event.detail.complete();
  }

  private loadUsers() {
    this.dataService.getUsers().subscribe(
```

```
    {
      next:  (data: Usuario[]) => {
        this.usuarios = data;
      },
      error: err => {
        console.error(err);
      },
      complete: () => {
        console.log('Users loaded!');
      }
    }

  );
}
}
```

Explicación

```
@ViewChild('lista', {static: false}) lista: IonList;
```

Aquí queremos acceder a un elemento del html, para eso recurrimos a la función de Angular @ViewChild() que nos lo facilita. Para ello tenemos que asignarle un nombre al elemento del html que queremos usar mediante #. Una vez asignado se lo pasamos como parámetro al ViewChild.

```
this.lista.closeSlidingItems();
```

Esta función nos permite cerrar el sliding al ser llamada, de tal forma que al ponerla en las funciones del sliding, conseguimos que se cierre cuando el usuario seleccione una opción.

```
this.usuarios.splice(event.detail.to, 0, itemMover);
```

El splice nos permite modificar los elementos de un array. La función tiene 3 parámetros:

- El primero es la posición del array desde donde vamos a atacar al array.
- El segundo es el número de elementos que vamos a recortar.
- El tercero es el elemento o elementos que vamos a insertar en la posición.

VISUALIZAR DATOS

El siguiente paso es cargar la información en nuestra lista en el html. Vamos a realizar 3 listas:

- La primera será una lista normal con una imagen en forma de Avatar y texto.
- La segunda será una lista con un ion-sliding para mostrar opciones al desplazar un ítem de la lista.
- La tercera nos permitirá reorganizar la lista.

En **list.page.html**:

```
<app-header titulo="Listas - Slide Items"></app-header>

<ion-content>

 <ion-list>
   <ion-list-header>
     <h2>List 1</h2>
   </ion-list-header>
   @for (user of usuarios; track user.id){
     <ion-item>
       <ion-avatar slot="start">
         <ion-img src="assets/shapes.svg"></ion-img>
       </ion-avatar>
       <ion-label>
         <h2>{{ user.name }}</h2>
                                <p>{{    user.address.street    }},
{{user.address.city}}</p>
       </ion-label>
     </ion-item>
   }

 </ion-list>

 <ion-list #lista>
   <ion-list-header>
     <h2>List 2 - Sliding</h2>
   </ion-list-header>
```

```
  @for (user of usuarios; track user.id){
    <ion-item-sliding>
      <ion-item>
        <ion-label>
          <h3>{{ user.name }}</h3>
          <p>{{ user.email }}</p>
        </ion-label>
        <ion-label slot="end" class="ion-text-right">
          {{ user.phone }}
        </ion-label>
      </ion-item>
      <ion-item-options side="start">
                              <ion-item-option   color="danger"
(click)="favorite(user)">
          <ion-icon name="heart" slot="icon-only"></ion-icon>
        </ion-item-option>
                            <ion-item-option   color="secondary"
(click)="share(user)">
          <ion-icon name="share-social" slot="icon-only">
          </ion-icon>
        </ion-item-option>
      </ion-item-options>
      <ion-item-options side="end">
                              <ion-item-option   color="success"
(click)="call(user)">
          <ion-icon name="call" slot="icon-only"></ion-icon>
        </ion-item-option>
      </ion-item-options>
    </ion-item-sliding>
  }

 </ion-list>
 <ion-list>
   <ion-list-header>
     <h2>List 3 - Reorder</h2>
   </ion-list-header>
   <ion-toggle [(ngModel)]="habilitado"></ion-toggle>
```

```
    <ion-reorder-group [disabled]="!habilitado"
                       (ionItemReorder)="reordenar($event)">
      @for (user of usuarios; track user.id){
        <ion-item>
          <ion-avatar slot="start">
            <ion-img src="assets/shapes.svg"></ion-img>
          </ion-avatar>
          <ion-label>
            <h2>{{ user.name }}</h2>
            <p>{{ user.company.name }}</p>
          </ion-label>
          <ion-reorder slot="end"></ion-reorder>
        </ion-item>
      }

    </ion-reorder-group>
  </ion-list>
</ion-content>
```

Y añadimos los iconos de heart y call en nuestro addIcons de **app.component.ts**

```
addIcons({... , heart, call});
```

Explicación

```
 <ion-list #lista>
```

Nos permite localizar a este ion-list con el nombre de variable lista.

```
<ion-item-sliding>
```

Usamos este elemento para generar el desplazamiento lateral del ítem para poder añadir opciones.

```
<ion-item-options side="start">
```

Las opciones del sliding. El side especifica por que lado aparecerá esta opción.

Introducción Toast

Un toast es una notificación sutil usada normalmente en aplicaciones normales. Puede ser usada para proporcionar información acerca de una operación o enviar un mensaje del sistema. El toast aparece en la parte superior de la aplicación y puede ser eliminada por la propia aplicación o por interacción con el usuario.

Servicio

En primer lugar vamos a añadir nuestra nueva función para lanzar el toast en el servicio que creamos previamente para usar el toast, **toast.service.ts**:

```
async listToast(message: string) {
 const toast = await this.toastCtrl.create({
   message,
   duration: 1500,
   color: 'success'
 });
 await toast.present();
}
```

Invocación del Toast

Una vez tenemos creada nuestra lista, vamos a invocar unos toast. En **list.page.ts**:

```
import {Component, inject, OnInit, ViewChild} from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import {
  IonAvatar, IonContent, IonHeader, IonIcon, IonImg, IonItem, IonItemOption, IonItemOptions, IonItemSliding, IonLabel,
```

```
IonList, IonListHeader, IonReorder, IonReorderGroup, IonTitle,
IonToggle, IonToolbar} from '@ionic/angular/standalone';
import {DataService} from "../../services/data.service";
import {Usuario} from "../../common/interfaces";
import                    {HeaderComponent}                    from
"../../components/header/header.component";
import {ToastService} from "../../services/toast.service";

@Component({
 selector: 'app-list',
 templateUrl: './list.page.html',
 styleUrls: ['./list.page.scss'],
 standalone: true,
   imports:   [IonContent,   IonHeader,   IonTitle,   IonToolbar,
CommonModule,     FormsModule,     HeaderComponent,     IonList,
IonListHeader,    IonItem,    IonAvatar,    IonImg,    IonLabel,
IonItemSliding,    IonItemOptions,    IonItemOption,    IonIcon,
IonToggle, IonReorderGroup, IonReorder]
})
export class ListPage implements OnInit {

 @ViewChild('lista', {static: false}) lista!: IonList;
 private dataService: DataService = inject(DataService);
 private toastService: ToastService  = inject(ToastService);
 usuarios: Usuario[] =  [];
 habilitado: boolean = false;
 constructor() { }

 ngOnInit() {
   this.loadUsers();
 }

 call(user: Usuario) {
   console.log(user);
   this.toastService.listToast('Llamando a... '+user.phone);
   this.lista.closeSlidingItems();
 }
```

```
 share(user: Usuario) {
   console.log(user);
            this.toastService.listToast('Compartiendo    con...
'+user.name);
   this.lista.closeSlidingItems();
 }

 favorite(user: Usuario) {
   console.log(user);
      this.toastService.listToast('Añadiendo  a  favoritos  a...
'+user.name);
   this.lista.closeSlidingItems();
 }


 reordenar(event: any) {
   console.log(event);
      // En primer lugar recogemos el elemento que ha
seleccionado el usuario y lo guardo en otro array
                              const          itemMover          =
this.usuarios.splice(event.detail.from,1)[0];
    // Ahora insertamos el elemento que teníamos en el array
auxiliar para guardarlo en nuestro array
   this.usuarios.splice(event.detail.to, 0, itemMover);
   // Por último hacemos efectivo el cambio
   event.detail.complete();
 }

 private loadUsers() {
   this.dataService.getUsers().subscribe(
     {
       next:  (data: Usuario[]) => {
         this.usuarios = data;
       },
       error: err => {
         console.error(err);
```

```
        },
        complete: () => {
          console.log('Users loaded!');
        }
      }
    );
  }
}
```

Propiedades de Toast

header → string - Para ponerle título a nuestro Toast.

color → string → primary, secondary ... - Para asignarle color.

animated → Boolean - Para aparición y desaparición progresiva.

position → string → bottom, top, middle - Para ponerlo en la parte superior, inferior o centro.

message → string - El mensaje que mostrará el Toast.