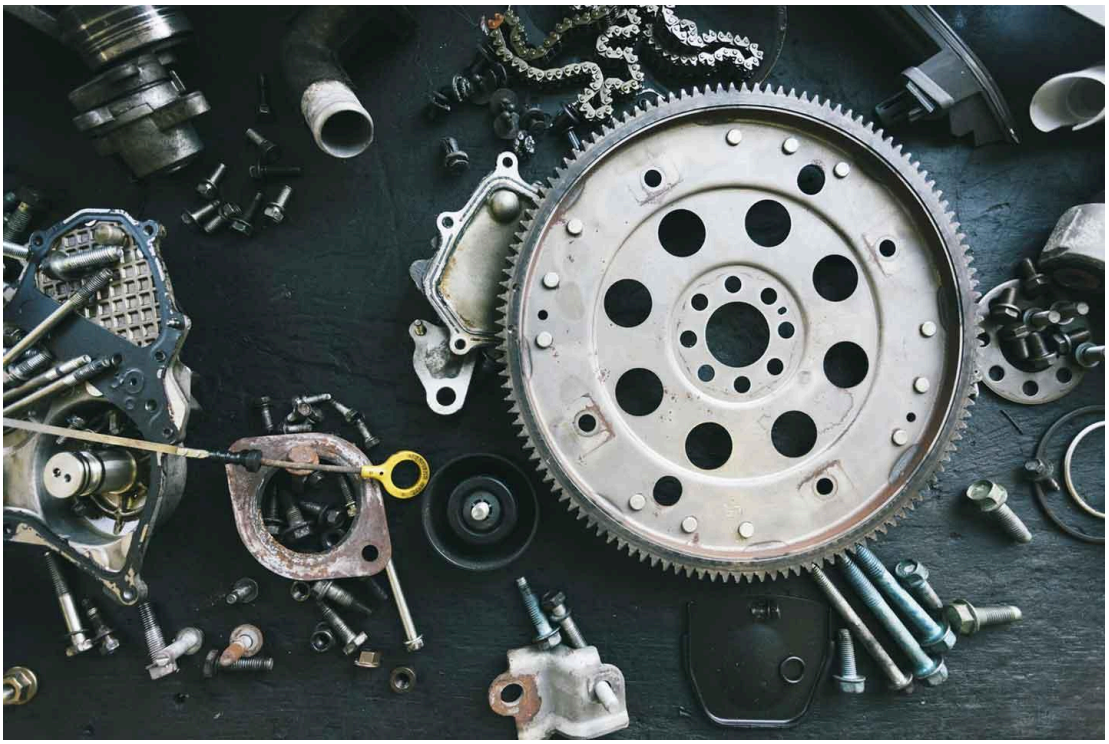
Ionic v8

Search Bar



Progresa
Una formación de futuro

ÍNDICE

Introducción

Representa un campo de texto que puede ser usado para buscar a través de una colección. Pueden mostrarse dentro de una toolbar o dentro del content principal.

La barra de búsqueda tiene que ser usada en lugar de un input sencillo para buscar listas. Un botón de borrar aparece al escribir en el campo del input y al apretarlo borrará el texto y el input se mantendrá seleccionado.

La documentación del Progress Bar está en la página:

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

Presentación

Componente

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

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

Icono

```
 addIcons({grid, … , flame, personAddOutline, search});
}
```

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

```
,{
 "nombre": "Search Bar",
 "ruta": "/search-bar",
 "icono": "search"
}
```

EJEMPLO

Para trabajar con la búsqueda vamos a necesitar un filtro que nos haga la búsqueda del elemento dentro de una lista. También necesitaremos la página donde veremos el ejemplo.

FILTRO

Vamos a crear el filtro con los pipes de Angular. En la **terminal** creamos el filtro:

```
ionic g pipe pipes/filtro
```

Y definimos nuestro filtro en **filtro.pipe.ts**:

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
 name: 'filtro'
})
export class FiltroPipe implements PipeTransform {

 transform(lista: any[],
           texto: string,
           columna: string): any[] {
```

```
    if ( texto === '') {
      return lista;
    }
    texto = texto.toLowerCase();
    return lista.filter(
        item => item[columna].toLowerCase().includes( texto));
  }
}
```

Explicación

```
transform(lista: any[], texto: string, columna: string): any[]
```

Para nuestro filtro vamos a utilizar 3 elementos:

- `lista: any[]`: Es la colección de elementos a la que le aplicaremos el filtro.
- `texto: string`: Es el texto que vamos a buscar dentro de la colección.
- `columna: string`: Es el atributo de la colección que contiene la información.

```
if ( texto === '') {return lista;}
```

Si no escriben nada al buscar, devolvemos toda la colección.

```
texto = texto.toLowerCase();
```

Pasamos a minúsculas el texto a buscar para así realizar la búsqueda sin importar minúsculas y mayúsculas.

```
return lista.filter(item => item[columna].toLowerCase().includes( texto));
```

Esta es la función que va a realizar el filtro y tiene varias partes:

- `filter`: Función que nos devuelve los elementos de la lista que cumplan la condición dentro de la función.
- `item[columna]`: De cada elemento del array queremos el atributo de la columna.
- `includes( texto)`: Es una función booleana que busca, diferenciando mayúsculas y minúsculas, el texto dado dentro de nuestro string.

SERVICIO

Ahora que ya tenemos el filtro, vamos a crear una lista de elementos para buscar en ella. Lo primero que vamos a hacer es traernos la lista de una web de fakes APIs, como hicimos para las listas. En nuestro servicio **data.service.ts** añadimos la siguiente función:

```
import {inject, Injectable} from '@angular/core';
import {HttpClient} from "@angular/common/http";
import {Observable} from "rxjs";
import {ApiMoviesResult, Componente, Usuario} from
"../common/interfaces";
import {environment} from "../../environments/environment";

@Injectable({
 providedIn: 'root'
})
export class DataService {
 private http: HttpClient = inject(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
')
 }
 loadMovies(page: number): Observable<ApiMoviesResult> {
   return this.http.get<ApiMoviesResult>(

`${environment.baseUrl}/movie/popular?api_key=${environment.apiKey}&
page=${page}`);
 }
 getAlbumes() {
   return this.http.get<any[]>('https://jsonplaceholder.typicode.com/albums');
 }
}
```

Lógica

Ahora que hemos traído la lista de albumes, vamos a cargarla en un array. También vamos a crear la función que guarde en una variable la palabra que se busque. En nuestro **search-bar.page.ts**:

```
import {Component, inject, OnInit} from '@angular/core';
import {CommonModule} from '@angular/common';
import {FormsModule} from '@angular/forms';
import {IonContent, IonHeader, IonTitle, IonToolbar} from
'@ionic/angular/standalone';
import {DataService} from "../../services/data.service";

@Component({
 selector: 'app-search-bar',
 templateUrl: './search-bar.page.html',
 styleUrls: ['./search-bar.page.scss'],
 standalone: true,
   imports: [IonContent, IonHeader, IonTitle, IonToolbar,
CommonModule, FormsModule]
})
export class SearchBarPage implements OnInit {
 private dataService: DataService = inject(DataService);
 textoBuscar = '';
 albumes: any[] = [];
 constructor() {}

 ngOnInit() {
   this.dataService.getAlbumes()
     .subscribe({
       next: (albumes: any[]) => {
         this.albumes = albumes;
       },
       error: err => {
         console.error(err);
       },
       complete: () => {
         console.log('Albumes cargados');
       }
     });
 }
```

```
  buscar(event: any) {
    this.textoBuscar = event.detail.value;
  }
}
```

Visualización

Y por último realizamos el html donde vamos a colocar nuestra barra de búsqueda y aplicaremos el filtro que hemos creado. En **search-bar.page.html**:

```
<ion-header class="no-border">
 <ion-toolbar>
   <ion-buttons slot="start">
     <ion-back-button defaultHref="/"></ion-back-button>
   </ion-buttons>
   <ion-title>Searchbar</ion-title>
 </ion-toolbar>
 <ion-searchbar animated="true" (ionInput)="buscar( $event)">
 </ion-searchbar>
</ion-header>

<ion-content class="ion-padding">
 <ion-list>
     @for (album of albumes | filtro: textoBuscar: 'title'; track
album.id){
     <ion-item>
       {{ album.title }}
     </ion-item>
   }
 </ion-list>
</ion-content>
```

Explicación

```
<ion-searchbar animated="true" (ionInput)="buscar( $event)">
```

Hay 3 elementos a destacar:

- `ion-searchbar`: Es la etiqueta de la barra de búsqueda.
- `animated="true"`: Indica que tendrá animación.
- `(ionInput)="buscar( $event)`: El evento que detecta la variación de los datos en la barra de búsqueda.

```
| filtro: textoBuscar: 'title'
```

| es la forma de utilizar un filtro. En nuestro caso tiene 3 parámetros:

- `filtro`: El nombre del filtro que vamos a aplicar.
- `textoBuscar`: La variable que contiene el texto que van escribiendo en la barra de búsqueda.
- `'title'`: El atributo donde tenemos que buscar el texto.