ÍNDICE

DESARROLLO

Vamos a comenzar el desarrollo, teniendo algunas consideraciones:

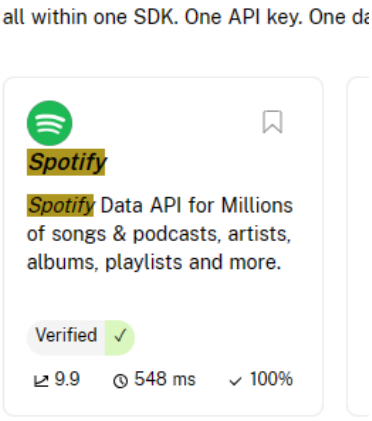
Objetivo:
Utilizar una API de Spotify para crear una tabla de favoritos.

En este proyecto involucramos tanto el frontend como el backend, y la comunicación con una API externa (Spotify). Antes de comunicarse desde la aplicacion, debemos consultar la documentación de la API de Spotify para comprender cómo realizar búsquedas y obtener información sobre las canciones.

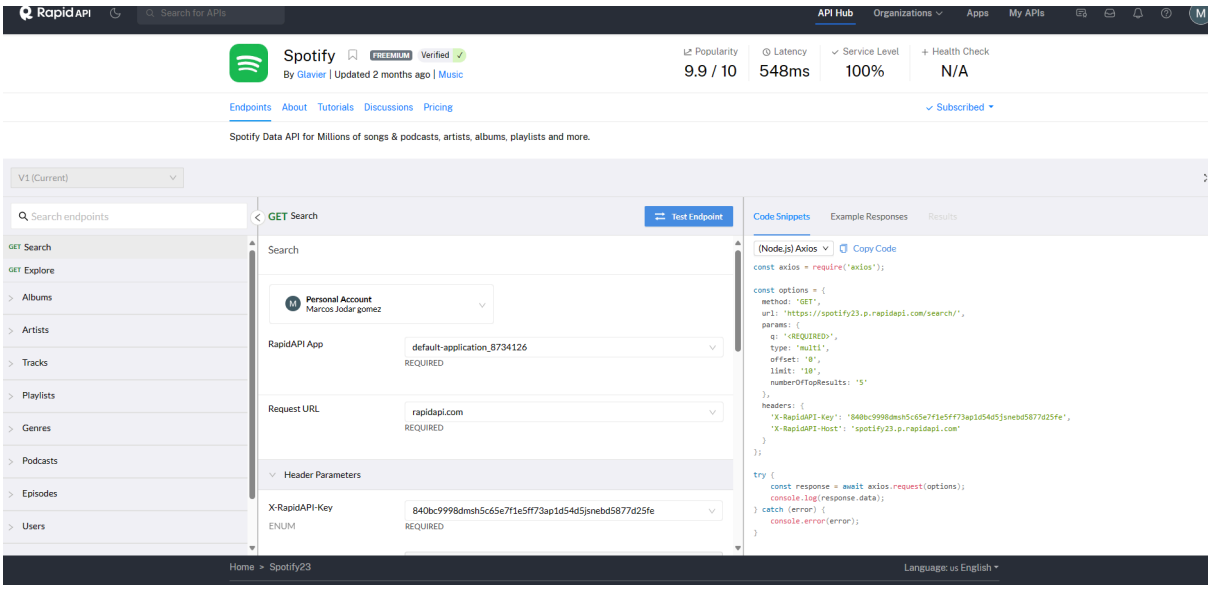
API

De donde y como hemos sacado la API de Spotify:
Debemos ir a la página Rapid API → Rapid API
Y ahí debemos buscar Spotify:

Será el primero que aparezca.

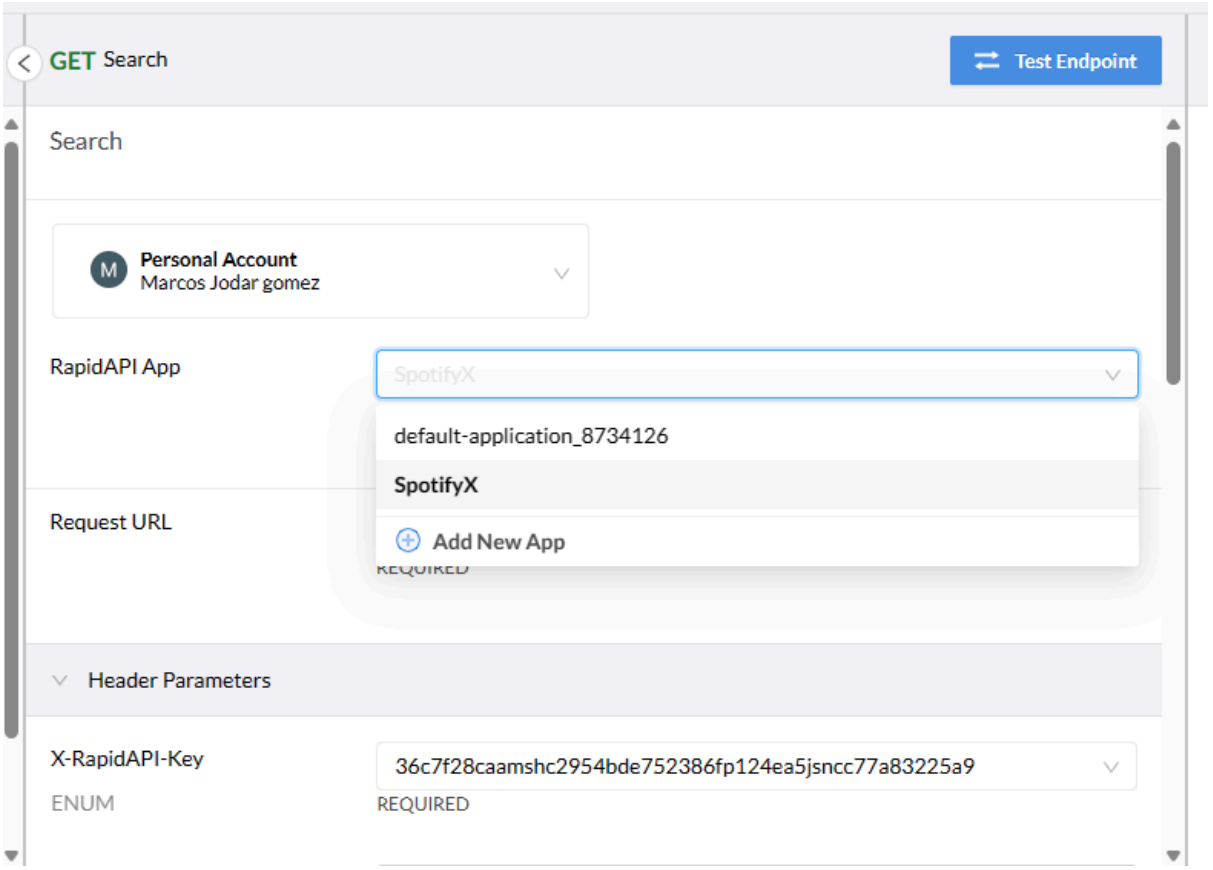


Luego nos aparecerá esta interfaz con varias opciones:



1. **Configuración de la API:**

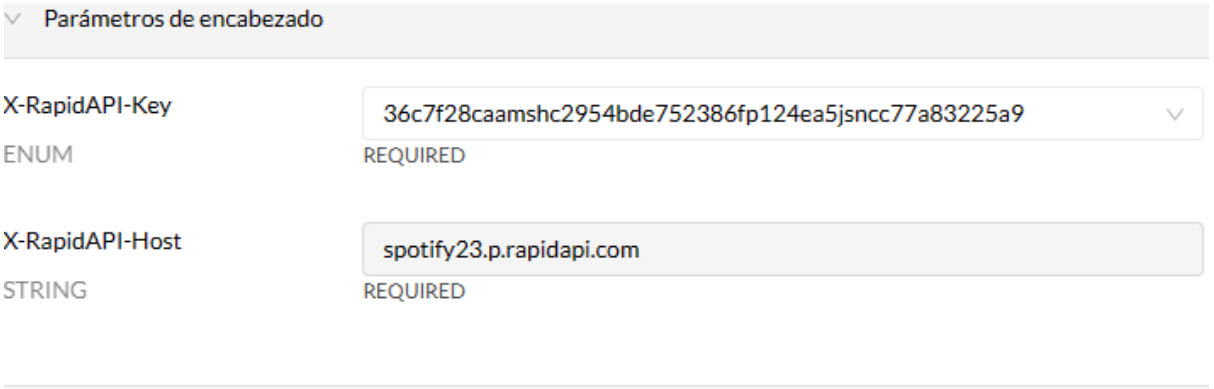
Aquí debemos ponerle un nombre a nuestro Proyecto, en mi caso es SpotifyX



Luego nos encontraremos con los encabezados de solicitud HTTP, de la API.

X-RapidAPI-Key: lo utilizamos como identificador único que nos proporciona *RapidAPI* para autenticar y autorizar las solicitudes a la API.

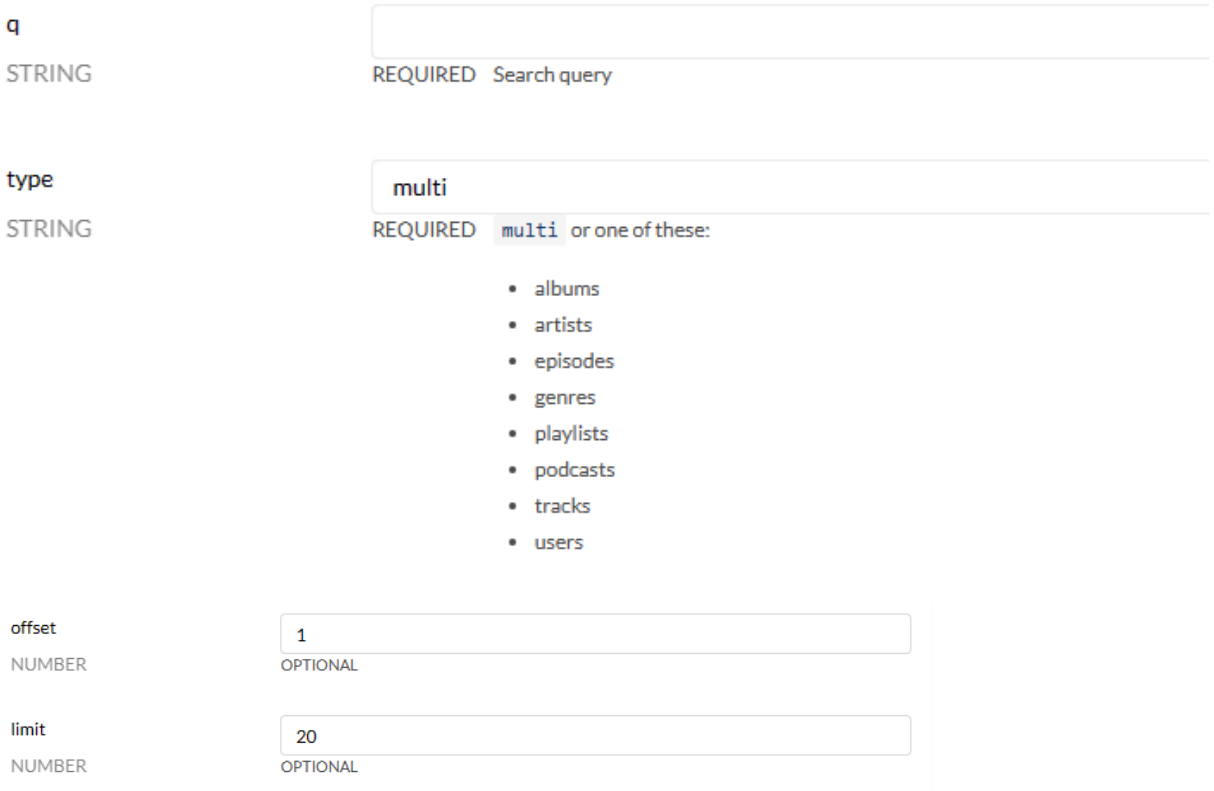
X-RapidAPI-Host: Especifica el host al que se dirige la solicitud, indicando que la solicitud se dirige al host de la API de Spotify dentro de *RapidAPI*.



Luego podemos configurar otros atributos, como:

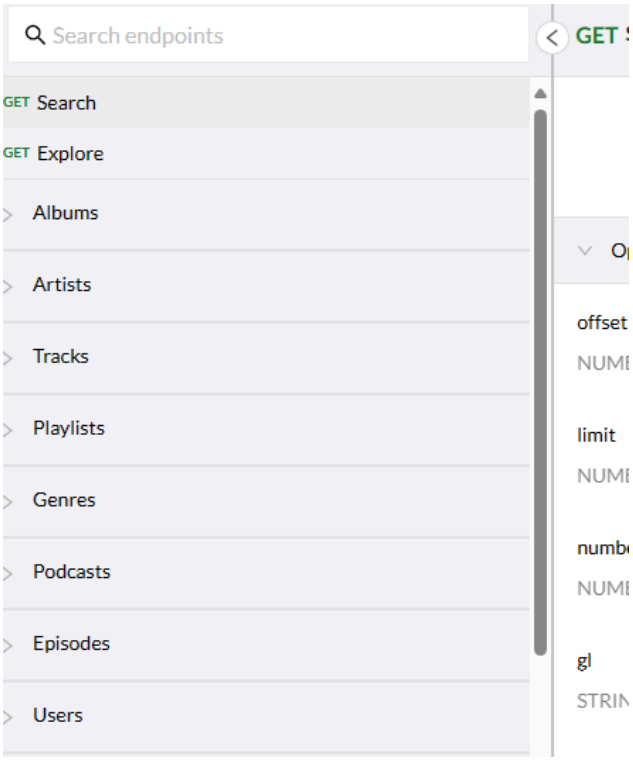
- query, que no hace falta poner nada, ya que se especificará en el codigo.
- type, para que al buscar te aparezcan, no solo canciones, si no audio libros, artistas, podcasts, etc.

Además de otros mas opcionales como offset, para saber desde donde poner el indice o límite, para saber el limite de datos



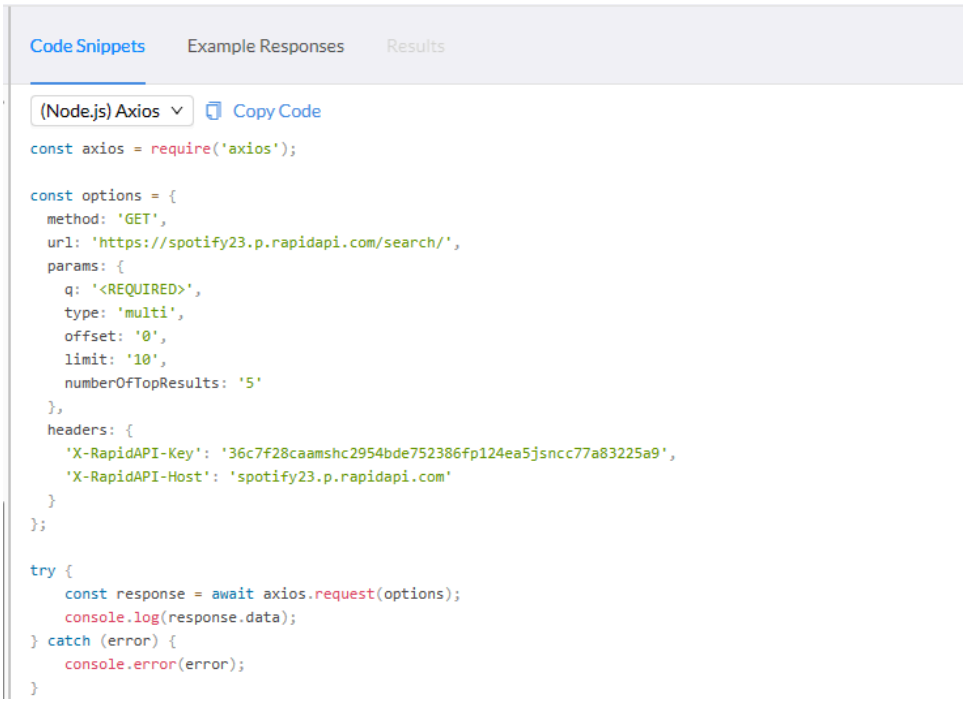
2. **Opciones de API:**

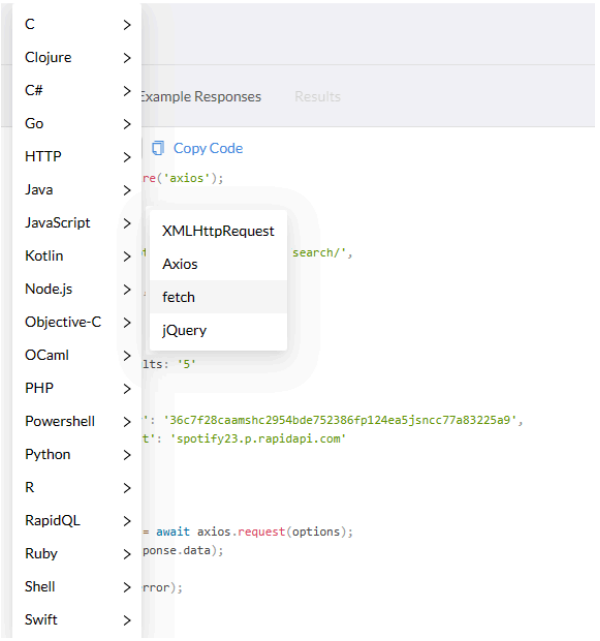
Luego tendríamos para poder elegir distintas opciones de datos, como son el Search (buscador), además de Álbumes, artistas, canciones específicas, etc.



3. Código

Por último tendríamos el código, con el url de la API y cabeceras incluidas, además de otras funciones. Pero aquí aparece como Node.js de Axios. Debemos cambiarlo:



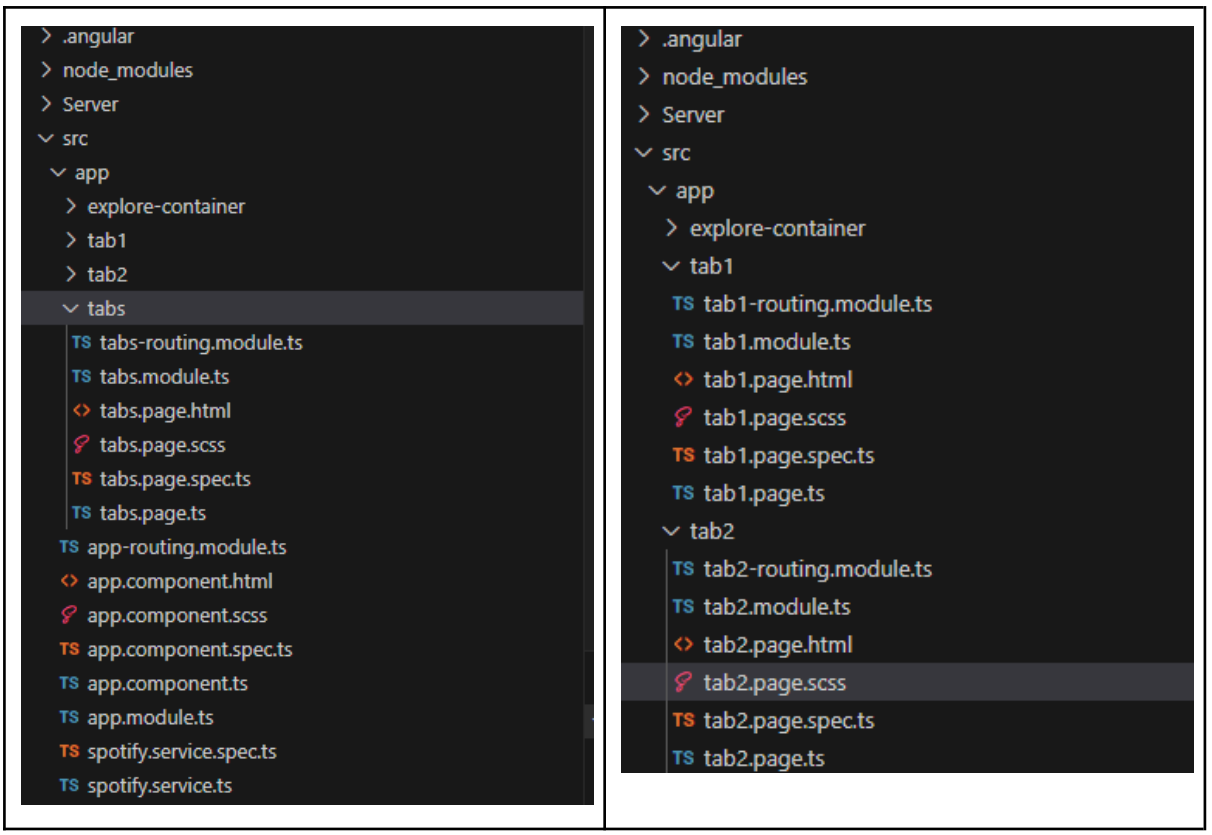


Ya tenemos el url, más las credenciales necesarias, ahora debemos aplicarlo en nuestro código:

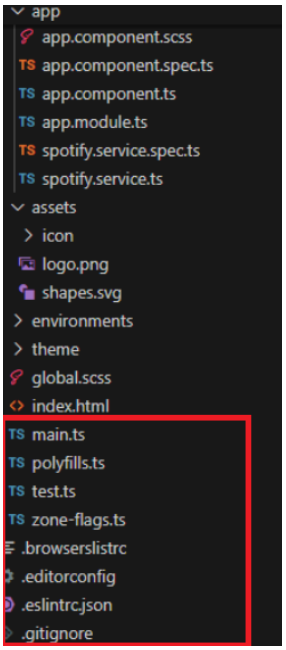
IONIC

ESTRUCTURA

Aquí es donde trabajaremos, sobre todo en tab1 y tab2, con sus clases:
tab1.module.ts - ***tab1.page.html*** - ***tab1.page.css*** - ***tab1.page.ts***



Luego también usaremos la parte de ***assets*** para añadir las fotos que necesitemos, como el logo. Y tambien ***index.html*** para configurar cosas como el icono y nombre de la pagina.



Pero las herramientas de abajo no las tocaremos.

TAB 1

Aquí haremos:
Crearemos una interfaz de usuario, donde tendremos una cabecera, con el logo, y el título de esta, en este caso es para añadir canciones a la lista de Favoritos.
Tendremos una barra de búsqueda y un botón Buscar. Si ponemos el nombre de una canción, artista o una letra, y le damos al botón buscar, se nos deberá de aparecer una lista de canciones, con las fotos del álbum. Si no añadimos nada y le damos al botón, que salte una alerta.
Cuando aparezca la lista tendrá un botón para escuchar la canción en Spotify, es decir, que tendrá un enlace. Y tendrá un botón añadir a favoritos, donde se nos añadirá a nuestra lista que tendremos en el Tab2.

HTML

```
<ion-content class="fondo">
  <div class="page-container">
    <ion-header class="colorp">
      <div>
        <ion-grid>
          <br>
          <ion-row>
            <ion-col>
              <img src="./assets/logo.png" class="logo">
            </ion-col>
            <ion-col class="p">
              <table class="pres">
                <tr>
                  <td class="a">Añadir canción a la lista Favoritos</td>
                </tr>
              </table>
            </ion-col>
          </ion-row>
        </ion-grid>
      </div>
    </ion-header>
  </div>

  <div class="fondo">
    <br>
```

```
    <br>
    <ion-card class="centro">
      <ion-label><h1>Seleccionar canción</h1></ion-label>


      <ion-searchbar showCancelButton="focus" placeholder="Nombre de la
canción" id="cancion" name="cancion" class="custom"
[(ngModel)]="cancion"></ion-searchbar>
      <button (click)="handleSearch()" class="boton btn2">Buscar</button>
      <br>
      <br>

      <div id="resultadosBusqueda" class="ctn">
        <div *ngFor="let cancion of canciones">
          <img [src]="cancion.data.albumOfTrack.coverArt.sources[0].url" />
          <h2>{{ cancion.data.name }}</h2>
          <a [href]="cancion.data.uri"><button
class="boton">ESCUCHAR</button></a>
          <ion-button (click)="anadirCanciones(cancion)" color="#1ED760"
class="boton" id="boton">Añadir a la lista</ion-button>
          <br>
          <br>
        </div>
      </div>
      <br>
    </ion-card>


    <br>

  </div>
</ion-content>
```

Codigo relevante:

- **Barra de búsqueda de canciones:** La barra de búsqueda de canciones, representada por `<ion-searchbar>`, permite a los usuarios ingresar el nombre de una canción que desean buscar en la plataforma.
- **Botón de búsqueda:** Al hacer clic en el botón de búsqueda, que está asociado al evento `(click)="handleSearch()"`, se activa la función `handleSearch()`. Esta función se encarga de iniciar la búsqueda de canciones en función del término ingresado en la barra de búsqueda.

- **Resultados de la búsqueda:** Los resultados de la búsqueda se muestran en la sección con el identificador `#resultadosBusqueda`. Para cada canción encontrada, se muestra la imagen de la portada del álbum, el nombre de la canción y un botón para escucharla en Spotify.
- **Botón "Añadir a la lista":** Cada resultado de búsqueda tiene asociado un botón que permite al usuario añadir la canción a una lista de favoritos. Este botón activa la función `anadirCanciones(cancion)` cuando se hace clic en él.

Estas funciones permiten buscar canciones, ver los resultados de la búsqueda y añadir canciones a una lista de favoritos. Ademas de escuchar las canciones

TYPESCRIPT: *tab1.page.ts*

```
import { Component, EventEmitter, Output  } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { AlertController } from '@ionic/angular';
import { Router } from '@angular/router';


@Component({
  selector: 'app-tab1',
  templateUrl: 'tab1.page.html',
  styleUrls: ['tab1.page.scss']
})
export class Tab1Page {
  cancion: string = '';
  canciones: any[] = [];
  public cancionesAgregadas: any[] = []; // Lista de canciones agregadas//
Lista de canciones agregadas


  constructor(private http: HttpClient, private alertController:
AlertController, private router: Router) {}


  async handleSearch() {
    if (this.cancion.trim() === '') {
      const alert = await this.alertController.create({
        header: 'ALERTA',
        message: 'Debes añadir una cancion.',
        buttons: ['ACEPTAR'],
      });
```

```
      await alert.present();
      return;
    }

    try {
      const url =

`https://spotify23.p.rapidapi.com/search/?q=${this.cancion}&type=multi&offset=
1&limit=20&numberOfTopResults=5`;



      const options = {
        headers: new HttpHeaders({
          'X-RapidAPI-Key':
'36c7f28caamshc2954bde752386fp124ea5jsncc77a83225a9',
          'X-RapidAPI-Host': 'spotify23.p.rapidapi.com',
        }),
      };




      const res: any = await this.http.get(url, options).toPromise();
      this.canciones = res.tracks.items;
    } catch (error) {
      console.log(`ups... error: ${error}`);
    }
  }

  // Esta función se llamará cuando hagas clic en el botón "Añadir a la lista"
  // Método para agregar una canción a la lista
  async anadirCanciones(cancion: any) {
    if (this.cancionesAgregadas.push(cancion)){
      console.log('Canción agregada a la lista:', cancion.data.name, cancion);
      this.router.navigate(['/tabs/tab2'], { state: { cancionesAgregadas:
this.cancionesAgregadas } });

    }
    // No necesitas añadir canciones aquí, ya que lo haces en el Tab2Page
  }
}
```

Manejo de la búsqueda de canciones: La función `handleSearch()` se activa cuando se busca una canción. Verifica si se ha ingresado un término de búsqueda y luego realiza una solicitud HTTP a la API de Spotify para obtener los resultados de la búsqueda.
Alerta de advertencia: Si el campo de búsqueda está vacío al intentar buscar una canción, se muestra una alerta de advertencia utilizando `AlertController` para informar al usuario que debe ingresar un término de búsqueda válido.
Añadir canciones a la lista: Cuando se hace clic en el botón *"Añadir a la lista"*, se activa la función `anadirCanciones(cancion)`. Esta función añade la canción seleccionada a la lista `cancionesAgregadas` y luego navega al **`Tab2Page`** utilizando el Router. También pasa la lista de canciones agregadas al Tab2Page utilizando el estado de la ruta.

Estas funciones permiten al usuario buscar canciones, agregarlas a una lista y navegar entre las pestañas para ver las canciones agregadas. El uso de las herramientas proporcionadas por Angular, Ionic y HttpClient permite una experiencia de usuario fluida y una interacción eficaz con la API de Spotify.

TYPESCRIPT: *tab1.module.ts*

```
import { IonicModule } from '@ionic/angular';
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';

import { HttpClientModule } from '@angular/common/http';

import { Tab1Page } from './tab1.page';

import { ExploreContainerComponentModule } from
'../explore-container/explore-container.module';
import { Tab1PageRoutingModule } from './tab1-routing.module';

@NgModule({
  imports: [
    IonicModule.forRoot(),
    CommonModule,
    FormsModule,
    HttpClientModule,
    ExploreContainerComponentModule,
    Tab1PageRoutingModule
  ],

  declarations: [Tab1Page]
})
```

```
export class Tab1PageModule {}
```

Añadimos aqui:
HttpClientModule: Proporciona capacidades de cliente HTTP para realizar solicitudes HTTP y manejar respuestas en la aplicación.
IonicModule.forRoot(): llamada de inicialización que prepara Ionic para su uso en toda la aplicación.

TAB 2

Aquí recogeremos las canciones agregadas con el botón de añadir a Favoritos y mostraremos una lista de las que hemos añadido, Puede aparecer solo o si no dando al botón mostrar. También, una vez que aparezcan las canciones, añadiremos botón para escuchar la canción, llevándonos a la página oficial de Spotify. Además del botón para eliminar canciones de la lista.
También añadiremos dos botones, uno arriba de todo y otro abajo de la página. Esto es para subir o bajar la pagina, si hay muchos elementos.

HTML

```
<ion-content class="fondo">

  <div class="page-container">

    <ion-header class="colorp">

      <div>
      <ion-grid>
        <br>
        <ion-row>
          <ion-col>
            <img src="./assets/logo.png" class="logo">
          </ion-col>

          <ion-col class="p">
            <table class="pres">
              <tr>
```

```
              <td class="a">
                FAVORITOS
              </td>
            </tr>
            </table>
          </ion-col>
        </ion-row>

      </ion-grid>
    </div>
  </ion-header>
  </div>
    <br>
    <br>

    <ion-button expand="block" (click)="scrollToBottom()" color="#1ED760"
class="boton">Bajar</ion-button>
<br>
<button (click)="actualizar()" class="btn3">Mostrar Favoritos</button>
   <br>
  <ion-card *ngFor="let cancion of cancionesAgregadas" class="ctn">
    <br>
    <img  [src]="cancion.data.albumOfTrack.coverArt.sources[0].url" />
    <ion-card-header>
      <ion-card-title class="blanco"> <h2>{{ cancion.data.name
}}</h2></ion-card-title>
      <ion-card-subtitle class="blanco"><h2>{{
cancion.data.artists.name}}</h2></ion-card-subtitle>
    </ion-card-header>

    <ion-card-content>
      {{ cancion.data.albumOfTrack.name }}
      <br>
      <br>
      <a [href]="cancion.data.uri"><button class="bn">ESCUCHAR</button></a>
      <br>

      <button (click)="eliminar(cancion)" class="btn2">ELIMINAR</button>
      <!--<button (click)="modificar()" class="btn2">MODIFICAR</button>
      -->

    </ion-card-content>
```

```
  </ion-card>




<br>
  <ion-button expand="block" (click)="scrollToTop()" color="#1ED760"
class="boton">Subir</ion-button>
</ion-content>
```

```
<ion-button expand="block" (click)="scrollToBottom()"
  color="#1ED760" class="boton">Bajar</ion-button>
```

Este botón expandible ocupa todo el ancho disponible y tiene un color de fondo verde (`#1ED760`). Al hacer clic en este botón, se invoca la función `scrollToBottom()`, hace desplazarse hacia abajo en la página.

```
<button (click)="actualizar()" class="btn3">Mostrar Favoritos</button>
```

Botón que invoca la función `actualizar()` cuando se hace clic.

```
<ion-card *ngFor="let cancion of cancionesAgregadas" class="ctn">
```

Este es un componente de tarjeta de Ionic que se repite dinámicamente para cada elemento en la matriz `cancionesAgregadas`. Cada tarjeta contiene información sobre una canción específica.
Dentro de cada tarjeta:

- **`<img [src]="cancion.data.albumOfTrack.coverArt.sources[0].url" />`**: Muestra la imagen de la portada del álbum de la canción.
- **`<ion-card-title class="blanco"><h2>{{ cancion.data.name }}</h2></ion-card-title>:`**
  Muestra el título de la canción en color blanco.

  `<ion-card-subtitle class="blanco"><h2>{{ cancion.data.artists.name}}</h2></ion-card-subtitle>:`
  Muestra el nombre del artista en color blanco.

  `<ion-card-content>...</ion-card-content>:`
  Contiene el contenido adicional de la tarjeta, como el nombre del álbum y un enlace para escuchar la canción.

  `<button (click)="eliminar(cancion)" class="btn2">ELIMINAR</button>:`
  Botón, en el que se hace clic e invoca la función `eliminar(cancion)` para eliminar la canción correspondiente de la lista.

TYPESCRIPT: *tab2.page.ts*

```typescript
import { Component, ViewChild, OnInit } from '@angular/core';
import { IonContent } from '@ionic/angular';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-tab2',
  templateUrl: 'tab2.page.html',
  styleUrls: ['tab2.page.scss']
})
export class Tab2Page {
  cancionesAgregadas: any[] = [];
  @ViewChild(IonContent) content!: IonContent;


  constructor(private route: ActivatedRoute) {}

  ngOnInit() {
    this.cancionesAgregadas = history.state.cancionesAgregadas;
    console.log('Canciones recibidas en Tab2:', this.cancionesAgregadas);
  }

  onCancionAgregada(cancion: any) {
    this.cancionesAgregadas.push(cancion);
    console.log('Canción agregada a la lista:', cancion);
  }

  eliminar(cancion: any) {
    // Encuentra el índice de la canción en la lista

      // Elimina la canción del array
      this.cancionesAgregadas.splice(cancion, 1);

  }
  actualizar(){
    window.location.reload();
  }

  scrollToBottom() {
    if (this.content) {
```

```
      this.content.scrollToBottom(500);
    }
  }

  scrollToTop() {
    if (this.content) {
      this.content.scrollToTop(500);
    }
  }
}
```

`cancionesAgregadas: any[] = []:`
variable almacena la lista de canciones agregadas en la página.

`@ViewChild(IonContent) content!: IonContent;:`
Permite acceder al componente IonContent para realizar operaciones de desplazamiento en la página.

`constructor(private route: ActivatedRoute) {}`:
El constructor inyecta el servicio ActivatedRoute, que se utiliza para obtener los parámetros de la URL.

`ngOnInit() {...}:`
Método para cuando la página se inicia. Aquí, se recuperan las canciones agregadas de la página anterior (**`Tab1Page`**) a través del historial de navegación (`history.state.cancionesAgregadas`).

`onCancionAgregada(cancion: any) {...}:`
Este método se invoca cuando se agrega una nueva canción a la lista. Agrega la nueva canción a cancionesAgregadas y muestra un mensaje en la consola.

eliminar(cancion: any) {...}:
Este método elimina una canción de la lista cancionesAgregadas según el índice proporcionado.

actualizar() {...}:
Este método recarga la página actual, lo que puede ser útil para actualizar la lista de canciones después de realizar cambios.

scrollToBottom() {...} y scrollToTop() {...}:
Estos métodos hacen que el contenido se desplace hacia abajo y hacia arriba, respectivamente, dentro del componente IonContent.

Tab2Page maneja la visualización y gestión de la lista de canciones agregadas, permitiendo agregar, eliminar y actualizar la lista, así como desplazarse por el contenido. Las canciones agregadas se obtienen de la página anterior a través del historial de navegación.

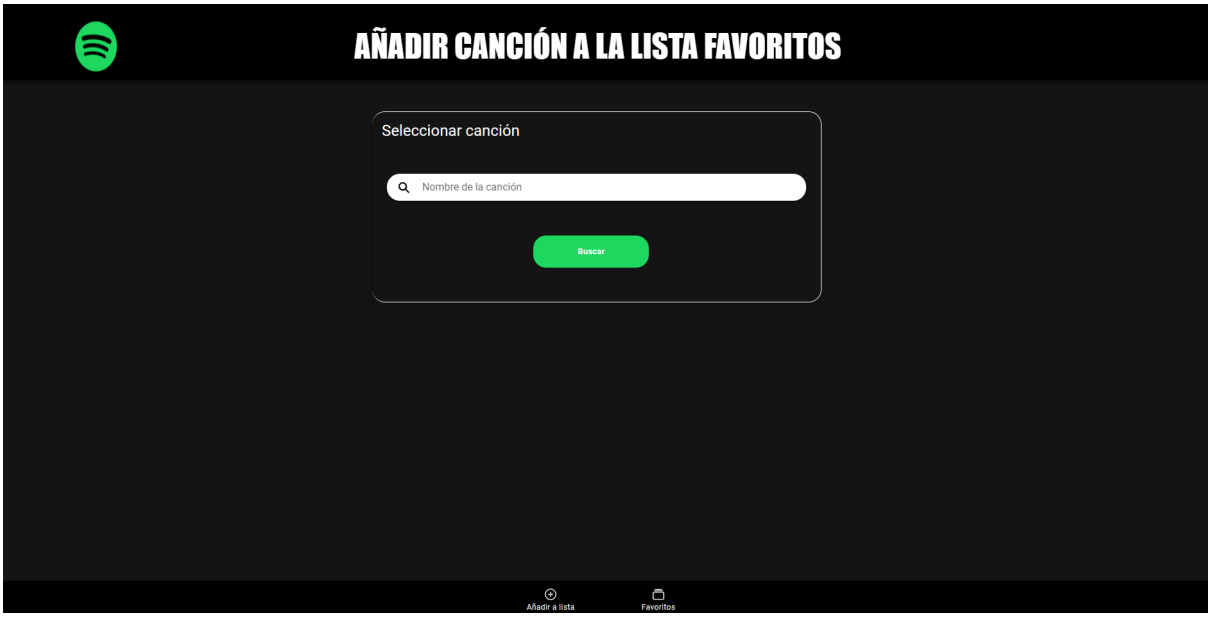
GLOBAL

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <title>SpotifyX</title><!-- NOMBRE AQUI -->

  <base href="/" />

  <meta name="color-scheme" content="light dark" />
  <meta name="viewport" content="viewport-fit=cover, width=device-width,
initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=no" />
  <meta name="format-detection" content="telephone=no" />
  <meta name="msapplication-tap-highlight" content="no" />
<!-- LOGO AQUI -->
  <link rel="icon" type="image/png" href="assets/icon/logo.png" class="logo"/>

  <!-- add to homescreen for ios -->
  <meta name="apple-mobile-web-app-capable" content="yes" />
  <meta name="apple-mobile-web-app-status-bar-style" content="black" />
</head>

<body>
  <app-root></app-root>
</body>

</html>
```

Aqui solo hemos cambiado un par de cosas, como la foto del logo y el nombre de la App

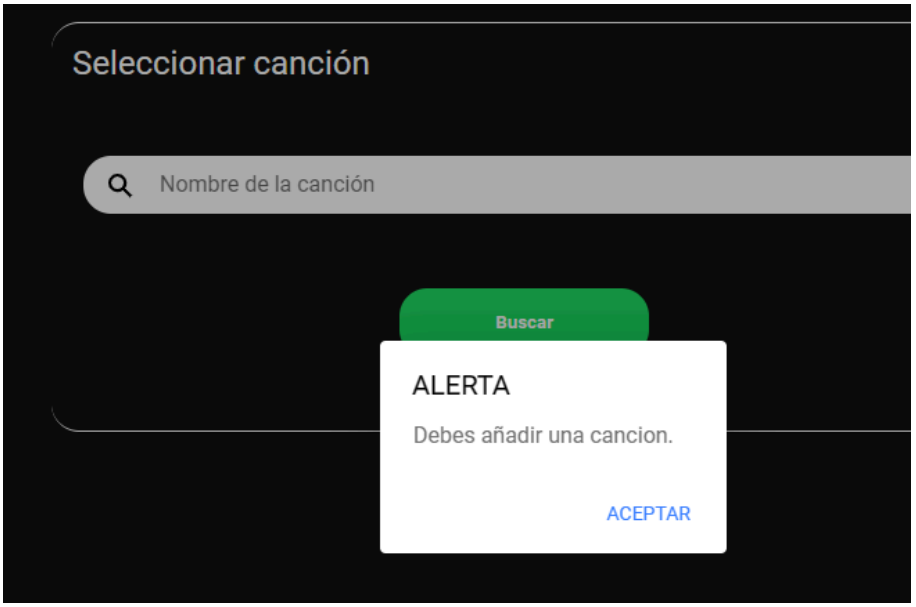
EVALUACIÓN FINAL

Si abrimos la web, con el comando ***Ionic serve*** se verá la página web:
Teniendo como primera interfaz está:

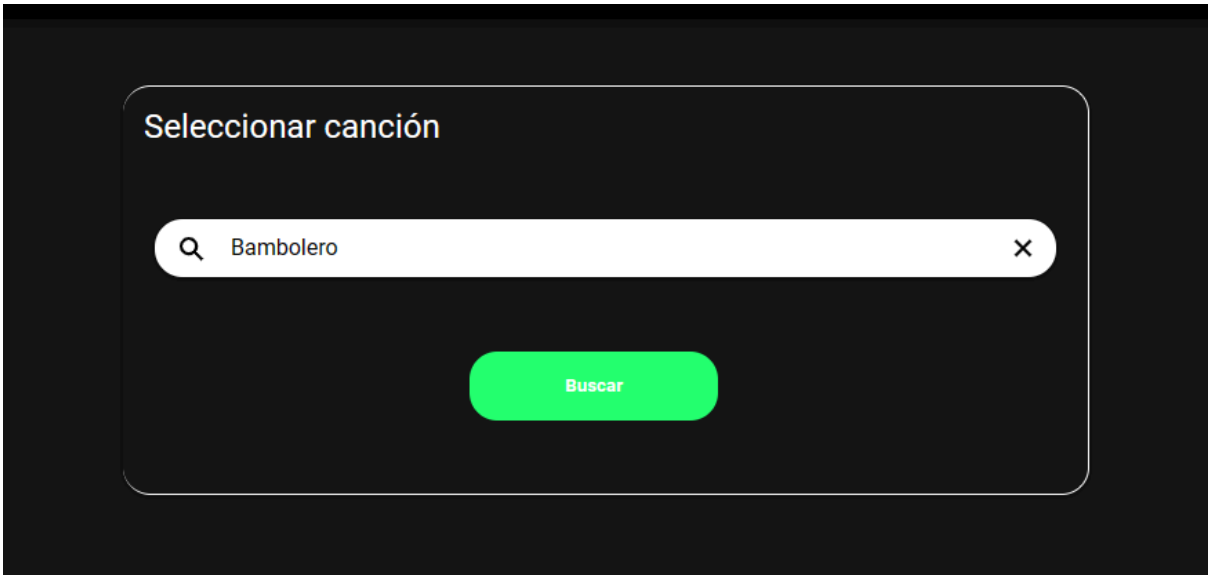


Aquí podremos añadir el nombre de la canción o artista y darle al botón buscar para uqe aparezca.

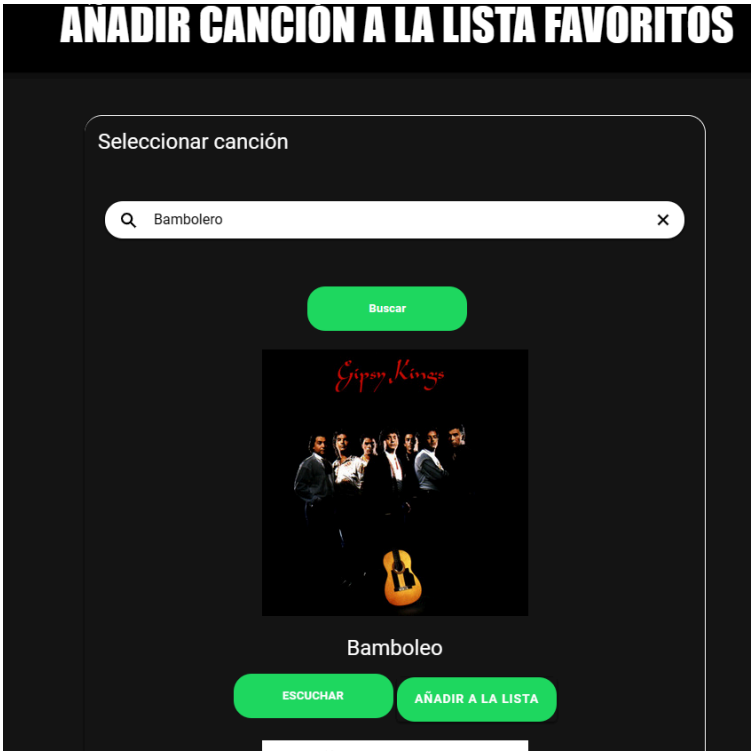
Si le damos al botón antes de tiempo, sin poner nada en la parte de buscar, se verá un mensaje de alerta, diciendo que debemos añadir una canción.



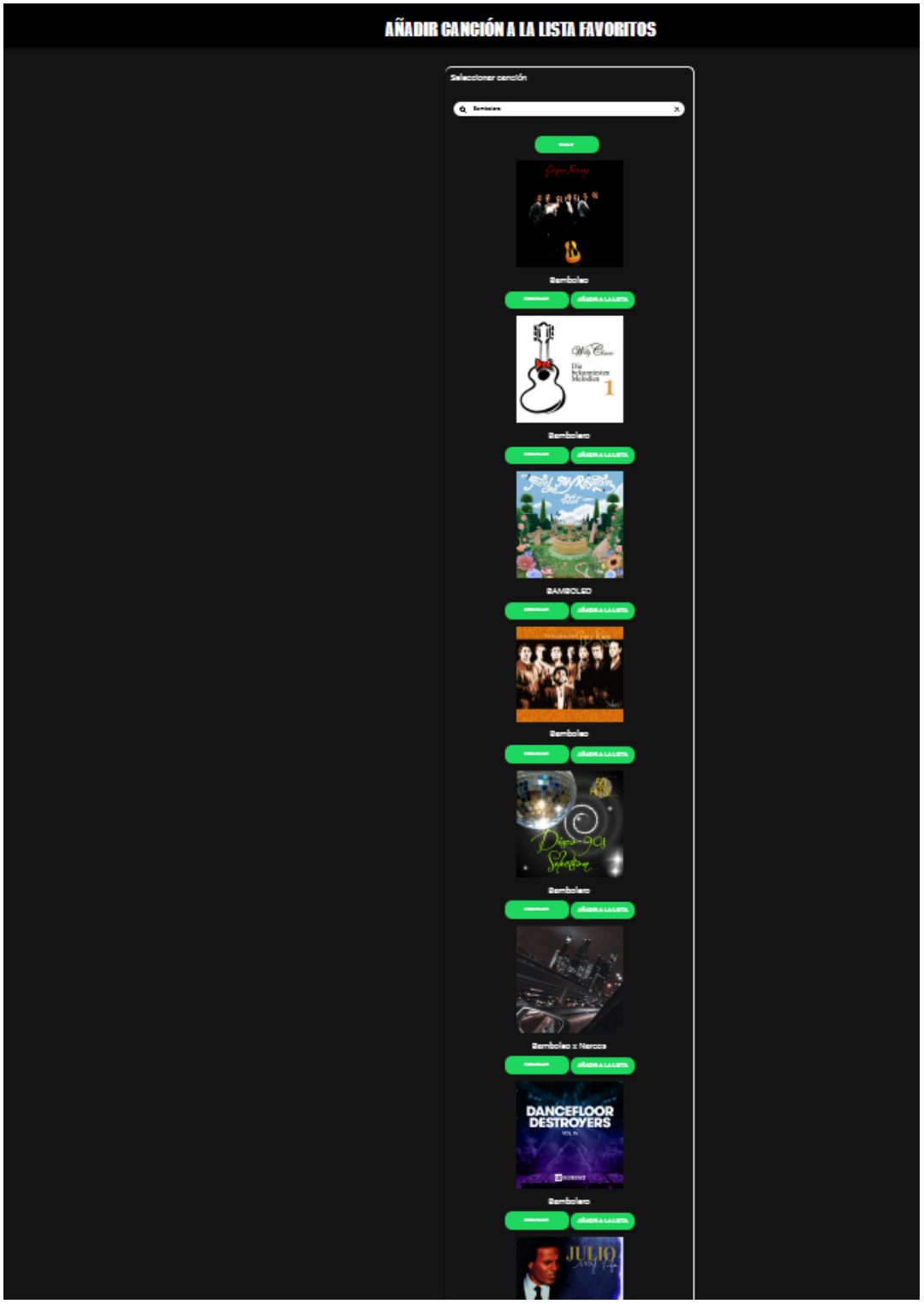
Si ponemos una canción como Bamboleo:



Nos aparecerán las canciones con sus fotos debajo.



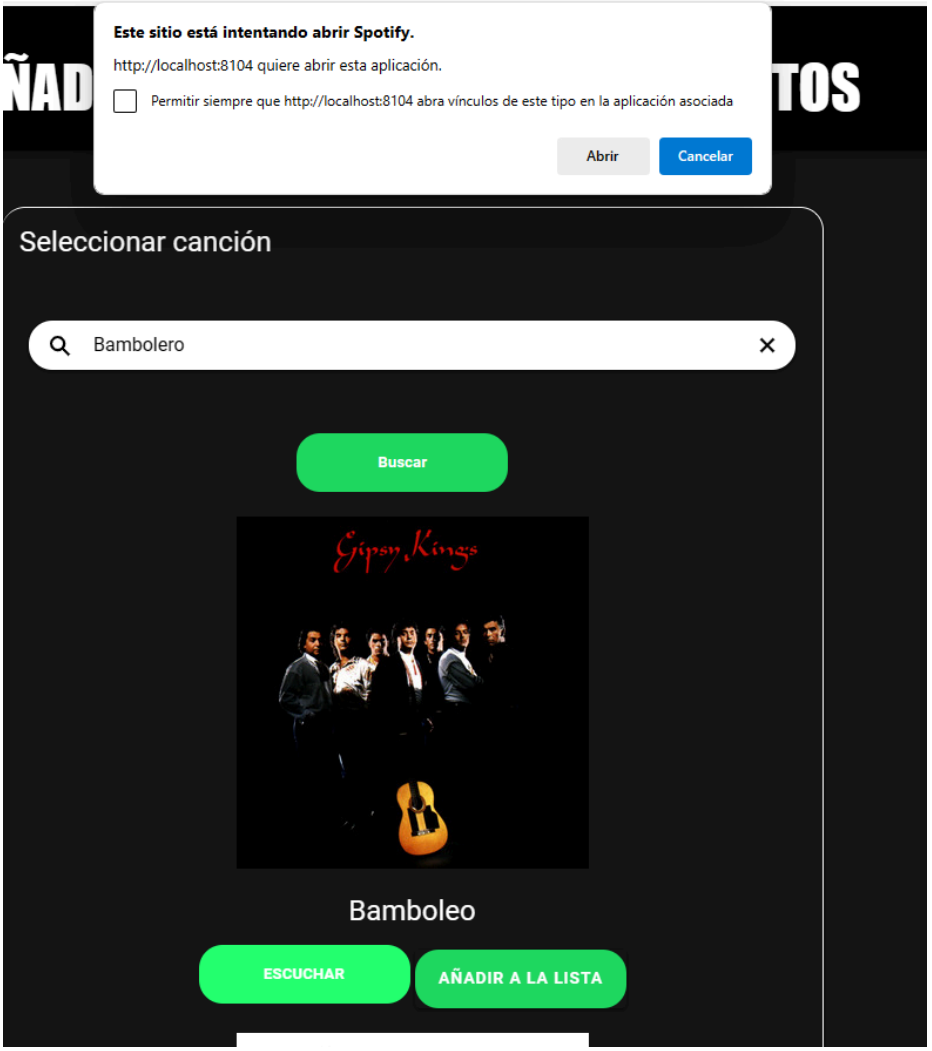
unas 20 Canciones, depende del límite que pusimos al principio, donde la configuración de la API

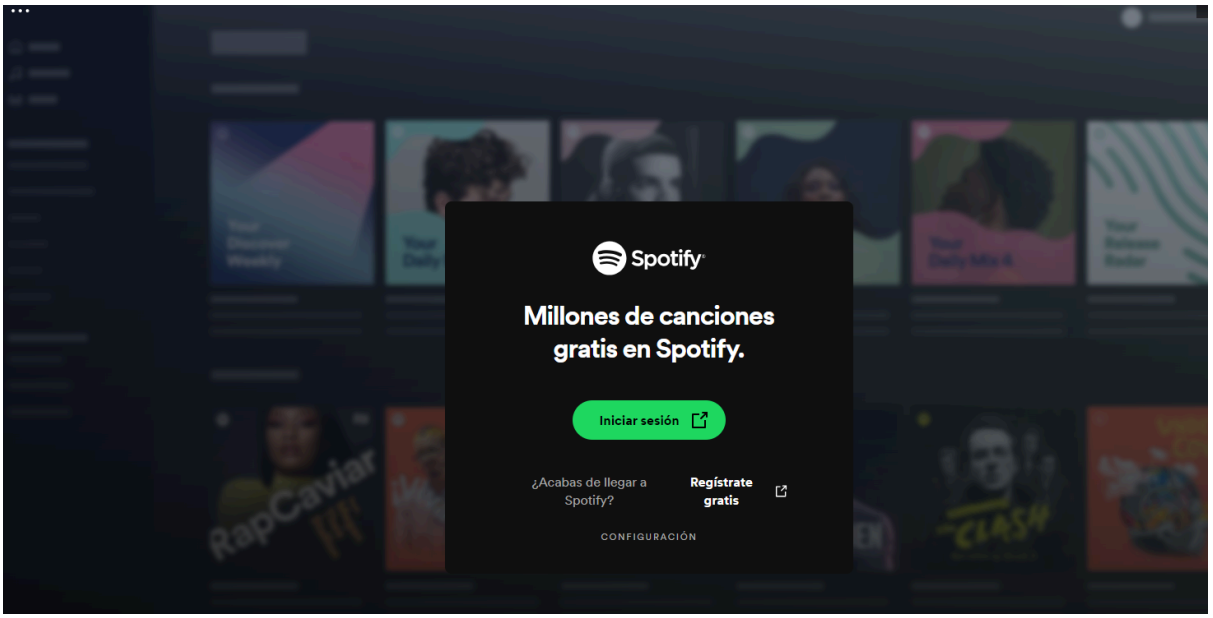


Una vez aparezcan las canciones, tenemos dos opciones:

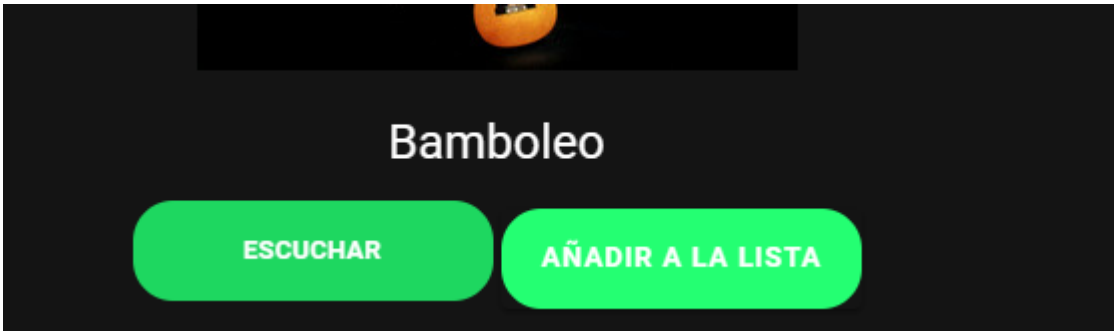
- Boton Escuchar cancion
- Boton añadir a Favoritos.

Con el bon escuchar nos pregintará siq queremos ir a Spotify y si decimos que si se abrirá.

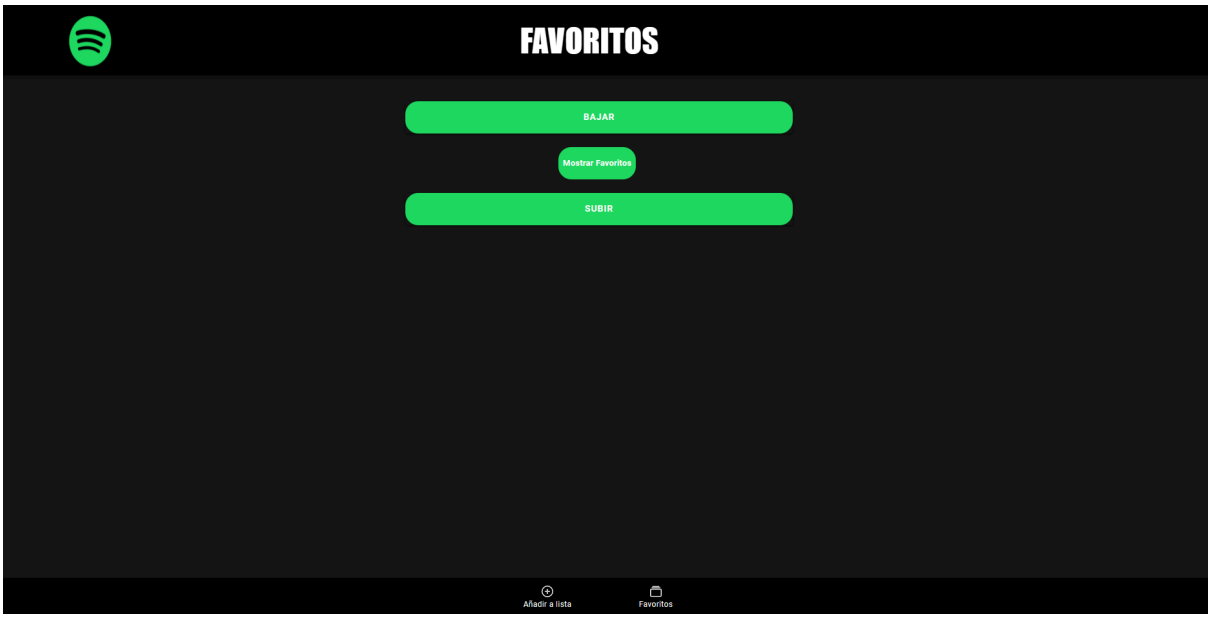

Este sitio está intentando abrir Spotify.
http://localhost:8104 quiere abrir esta aplicación.
Permitir siempre que http://localhost:8104 abra vínculos de este tipo en la aplicación asociada
Abrir
Cancelar
Seleccionar canción
Bambolero
Buscar
Gipsy Kings
Bamboleo
ESCUCHAR
AÑADIR A LA LISTA


Spotify
Millones de canciones gratis en Spotify.
Iniciar sesión
¿Acabas de llegar a Spotify?
Regístrate gratis
CONFIGURACIÓN

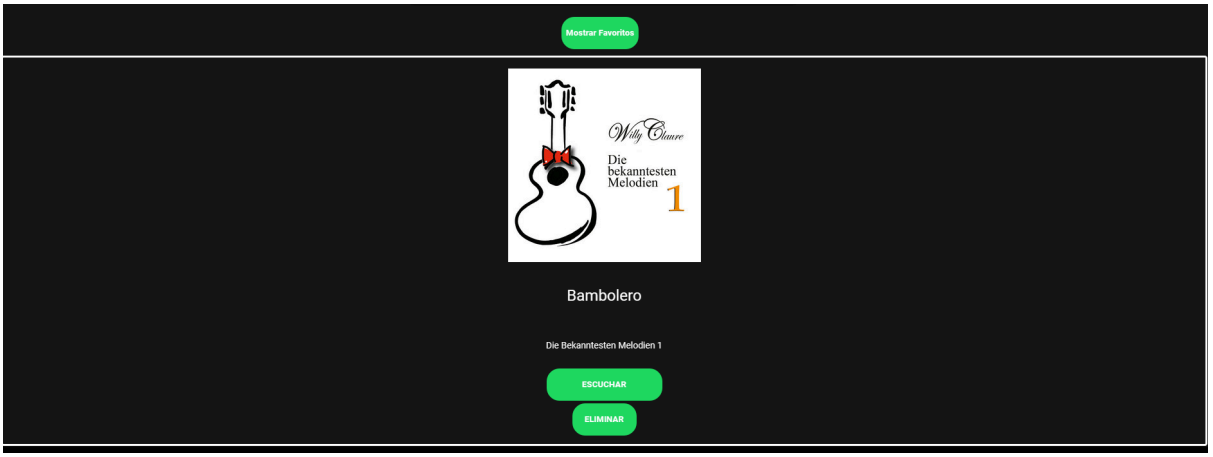
Posteriormente, el botón añadir a lista, nos llevará a la segunda interfaz.

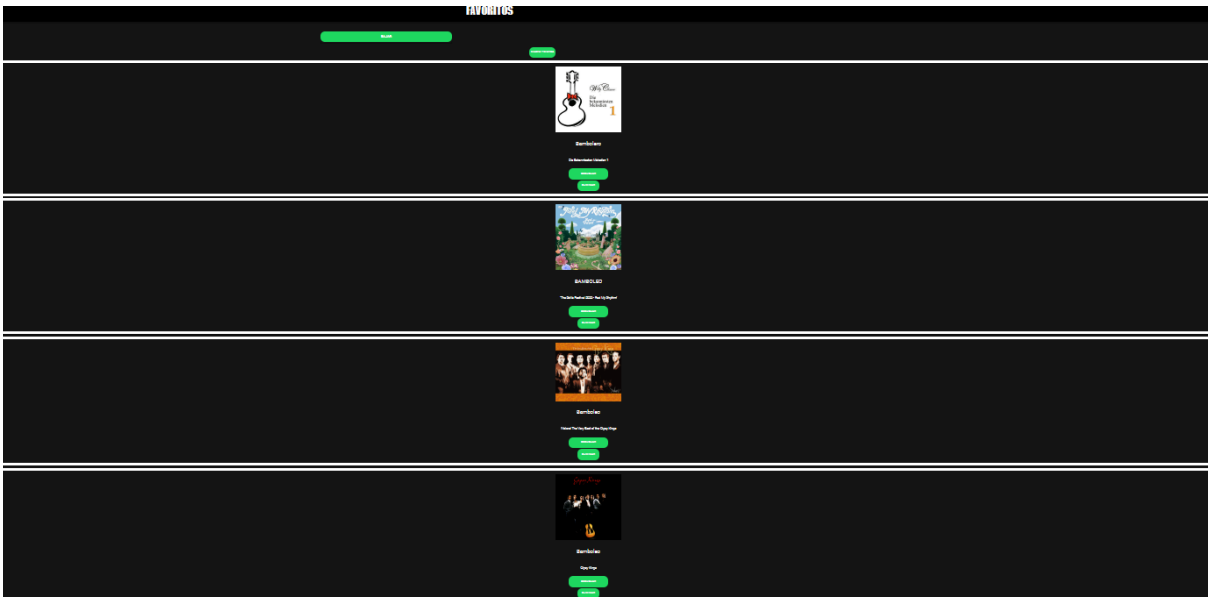


En esta interfaz se nos deberá de añadir la cancion a la lista Favoritos, asi como una Playlist.



Si le damos al botón mostrar favoritos, nos aparecerá la lista, con las canciones que hayamos añadido.

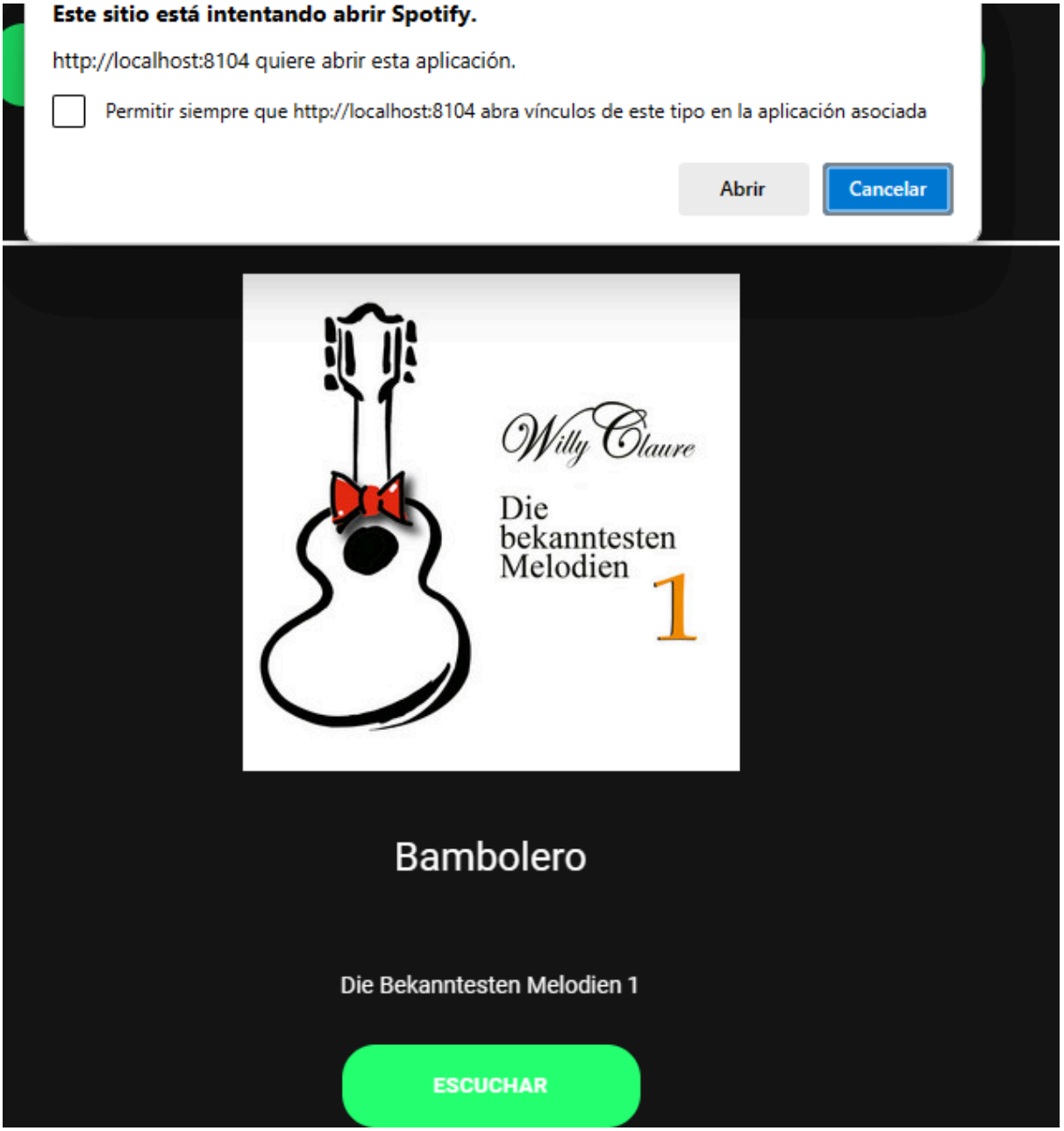




Si hemos añadido muchas canciones y queremos ir arriba del todo o abajo, tenemos los botones *BAJAR* y *SUBIR* para lograr ese objetivo.

Luego, si queremos eliminar una canción de la lista, basta con darle al botón *ELIMINAR*.

Por último tendremos la opción de escuchar la canción elegida.



Funciones varias:

Botones para subir o bajar la pestaña, estemos abajo o arriva

ERRORES

1. Error al borrar canciones, sin seguir el índice.
2. Error al añadir una canción, que nos lleva directamente a la lista de favoritos.

BIBLIOGRAFÍA

IONIC, (14-feb-2024). Components. [20 de febrero de 2024]. Disponible en:
https://ionicframework.com/docs/components

RapidAPI, (4-sep-2023). Spotify. [Consultado el 26 de Febrero de 2024]. Disponible en:
https://rapidapi.com/Glavier/api/spotify23

Youtube, (). DiegolDeve. [Consultado el 23 de Febrero de 2024]. Disponible en
https://www.youtube.com/watch?v=9cz8Gvh0J7g