Universidad Nacional Experimental de Guayana

Vicerrectorado Académico

Coordinación General de Pregrado

Ingeniería en informática

Sistemas de Bases de Datos II

Sistema de Recomendación Musical

Docente:

Clinia Cordero

Integrantes:

Anyelis Coro C.I: 30.266.262

Gipsy Anaya C.I: 27.966.466

Puerto Ordaz, Junio 2025

Planteamiento del Problema

En la actualidad, la gestión y reproducción de contenido musical en línea es una necesidad ampliamente demandada por usuarios que buscan experiencias personalizadas y accesibles desde cualquier dispositivo. Sin embargo, muchas de las aplicaciones existentes presentan limitaciones en cuanto a la flexibilidad del almacenamiento de datos y la escalabilidad para manejar grandes volúmenes de información, especialmente en entornos distribuidos.

El presente proyecto académico tiene como objetivo diseñar e implementar un sistema básico de recomendación de música que permita almacenar datos de usuarios, canciones y escuchas en una base de datos NoSQL (Cassandra), e implementar consultas OLAP simplificadas para obtener estadísticas básicas sobre el comportamiento de los usuarios. Este sistema proporcionará funcionalidades como la recomendación de canciones más escuchadas por género o en la misma ciudad del usuario, además de consultas analíticas sencillas para identificar patrones de uso.

Solución Propuesta

Para resolver esta problemática, se desarrolló una aplicación web de gestión y reproducción de música, que permite a los usuarios registrarse, iniciar sesión, buscar canciones, reproducir contenido y gestionar su perfil, utilizando una arquitectura distribuida y moderna.

Herramientas y Tecnologías Utilizadas

- **Frontend**: desarrollado en React + TypeScript, gestionado con Vite y estilizado con Tailwind CSS para ofrecer una experiencia de usuario rápida y responsiva.
- **Backend**: implementado en Python con Flask, encargado de manejar las peticiones de autenticación, gestión de usuarios y reproducción de música.
- **Base de datos NoSQL con Cassandra**: diseñada para almacenar información de usuarios y canciones, garantizando alta disponibilidad, escalabilidad horizontal y tolerancia a fallos.
- Despliegue modular que permite ejecutar tanto el backend como el frontend de manera independiente o conjunta, facilitando la administración y escalado del sistema.

Esta solución no solo permite mejorar la experiencia del usuario final, sino que también presenta una arquitectura tecnológica moderna, escalable y de alto rendimiento, adaptable a entornos académicos, empresariales o comerciales.

Objetivos

- **Objetivo General:** Desarrollar una aplicación web para la gestión y reproducción de contenido musical, integrando una arquitectura moderna basada en React para el frontend, Flask para el backend y Cassandra como base de datos NoSQL, permitiendo ofrecer una experiencia rápida, escalable y de fácil administración.

- **Objetivos a Cumplir:**

- Permitir a los usuarios registrarse y crear una cuenta.
- Permitir a los usuarios iniciar y cerrar sesión.
- Mostrar una lista de canciones disponibles.
- Permitir la búsqueda de canciones por nombre o artista.
- Permitir la reproducción de canciones desde un reproductor integrado
- Permitir a los usuarios editar su perfil.
- Gestionar la autenticación de usuarios de manera segura.
- Guardar información de usuarios y canciones en una base de datos NoSQL (Cassandra).
- Ofrecer una interfaz web accesible, responsiva y amigable.

- **Objetivos Específicos:**
  - Diseñar e implementar una base de datos distribuida utilizando Cassandra para almacenar información de usuarios y canciones.

- Desarrollar un backend en Python con Flask para gestionar las operaciones de autenticación, consulta y reproducción de música.
- Crear un frontend interactivo y responsivo con React, TypeScript y Tailwind CSS.
- Integrar un reproductor musical funcional dentro de la plataforma web.
- Permitir a los usuarios registrarse, iniciar sesión, buscar canciones, visualizar detalles y administrar su perfil.
- Garantizar la escalabilidad y modularidad de la aplicación mediante tecnologías modernas y buenas prácticas de desarrollo web.

Diccionarios de Clases y sus Usos

- **Frontend**

/src/components/

Clase/Componente	**Uso**
Navbar	Renderiza la barra de navegación principal, incluyendo accesos a las distintas secciones de la app.

Footer	Muestra un pie de página con información general o créditos.
Layout	Estructura base de las páginas, incorporando Navbar y Footer.
SongCard	Componente individual para mostrar la información de una canción (título, artista, imagen, botones de acción).
SongList	Componente que organiza y lista múltiples SongCard en pantalla.
MusicPlayer	Controlador de reproducción musical (play, pause, skip, volumen).

/src/context/

Clase/Contexto	**Uso**
AuthContext	Provee información y métodos de autenticación (login, logout, user data) a los componentes hijos de la app.

/src/pages/

Página	Uso
LoginPage	Permite al usuario iniciar sesión en la plataforma.
RegisterPage	Permite al usuario registrarse.
HomePage	Página principal que muestra las canciones disponibles.
SearchPage	Permite al usuario buscar canciones mediante palabras clave.
ProfilePage	Muestra y permite editar los datos del perfil del usuario autenticado.

- **Backend**

/backend/

Clase/Archivo	Uso
app.py	Servidor Flask principal; gestiona rutas HTTP para autenticación, consulta de canciones y administración de usuarios.

cassandra_schema.cql	Define las tablas y esquemas de la base de datos NoSQL en Cassandra.

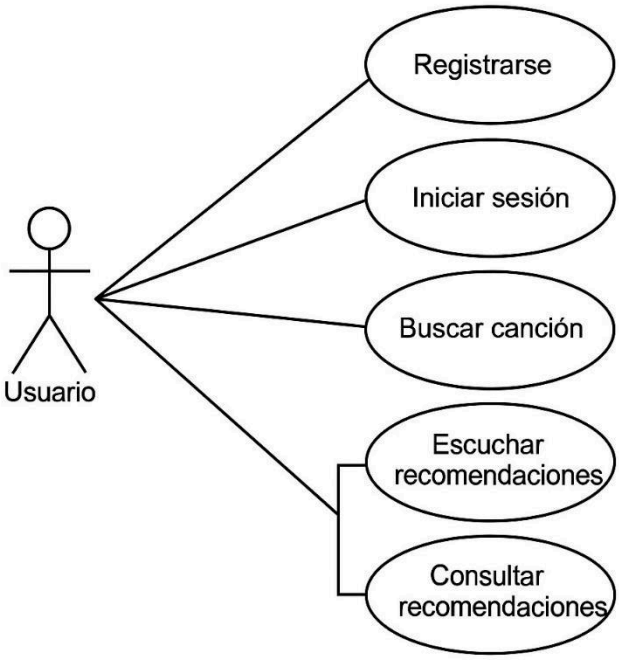
- **Bases de Datos**

cassandra_schema.cql

Tabla	**Uso**
usuarios	Almacena información de los usuarios registrados (usuario_id, nombre, ciudad).
canciones	Almacena datos de las canciones disponibles (cancion_id, título, artista, género).
escuchas	Guarda los registros de las escuchas realizadas por los usuarios, relacionando usuario, canción y fecha de escucha.

Diagramas

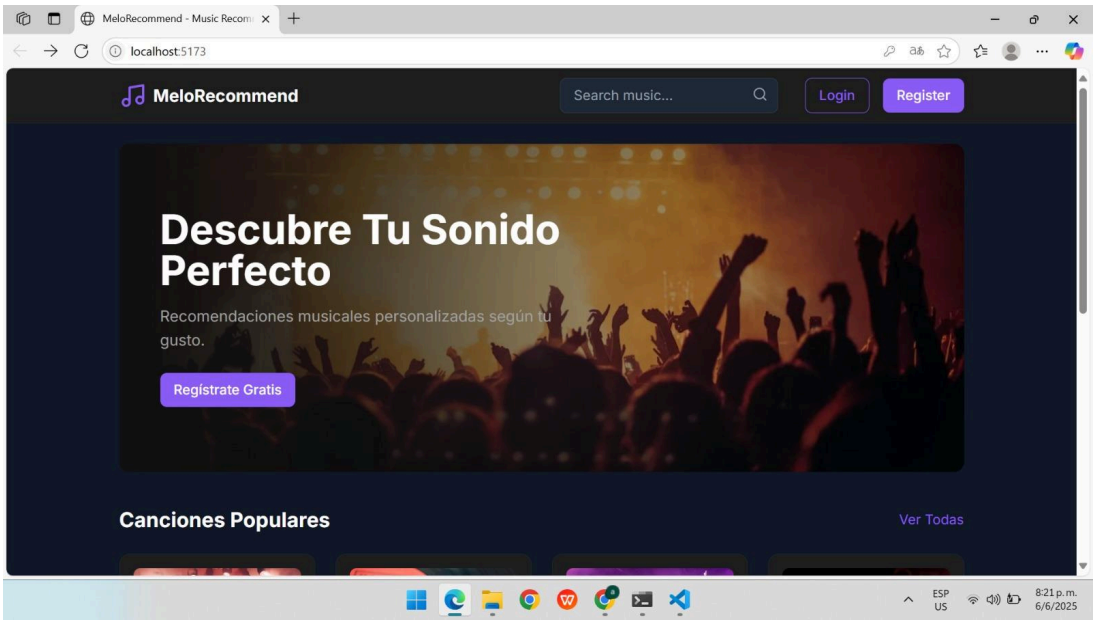
- **Diagrama de casos de uso**



- **Diagrama de flujo**

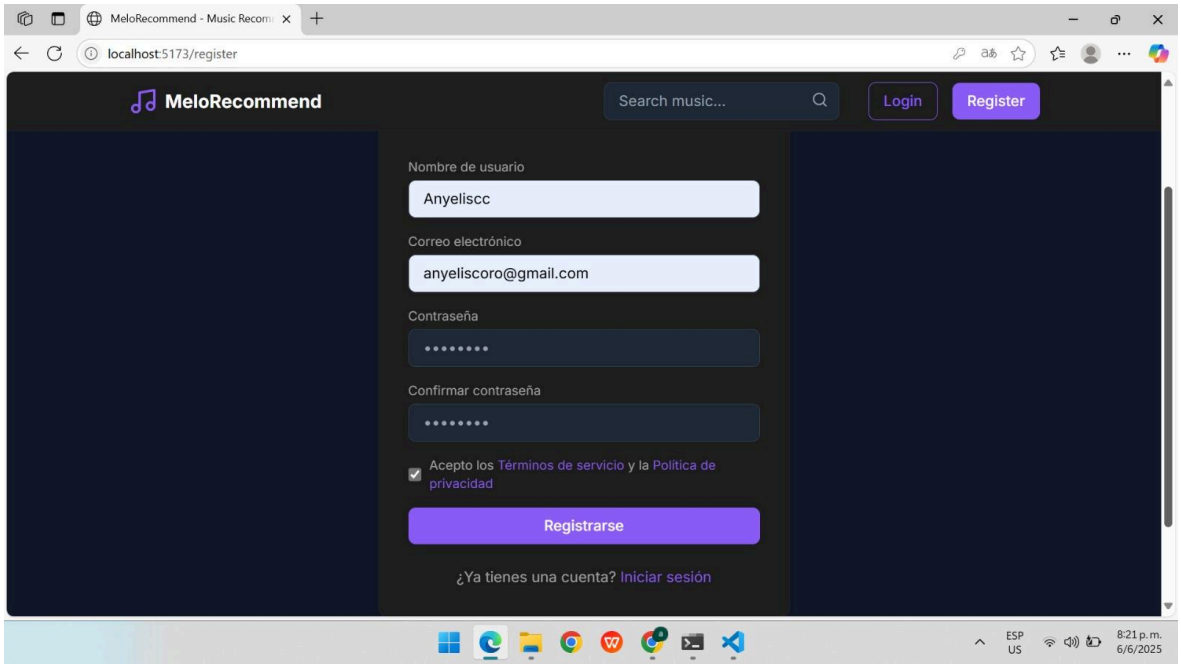
Vistas

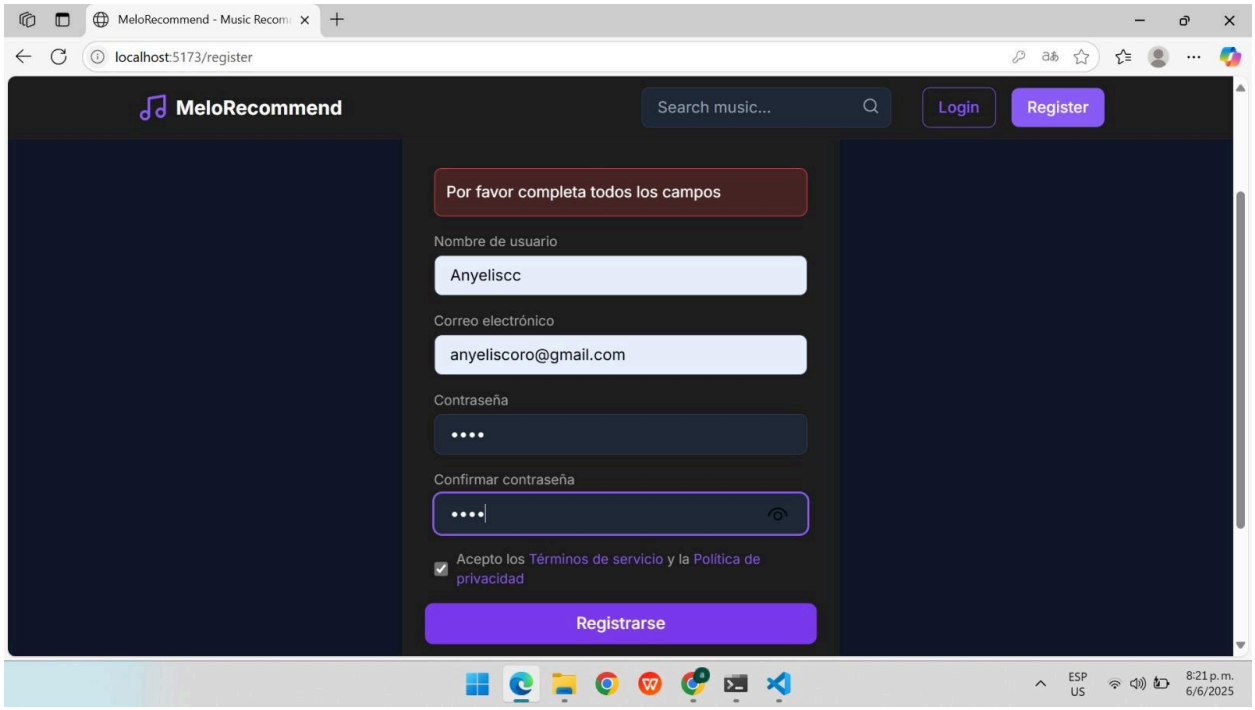
- **Página de Inicio**



Pantalla principal que se presenta al usuario al ingresar a la página, donde puede visualizar las canciones disponibles, acceder al reproductor, acceso a búsqueda de canciones en específico, canciones populares y, si lo desea, iniciar sesión.

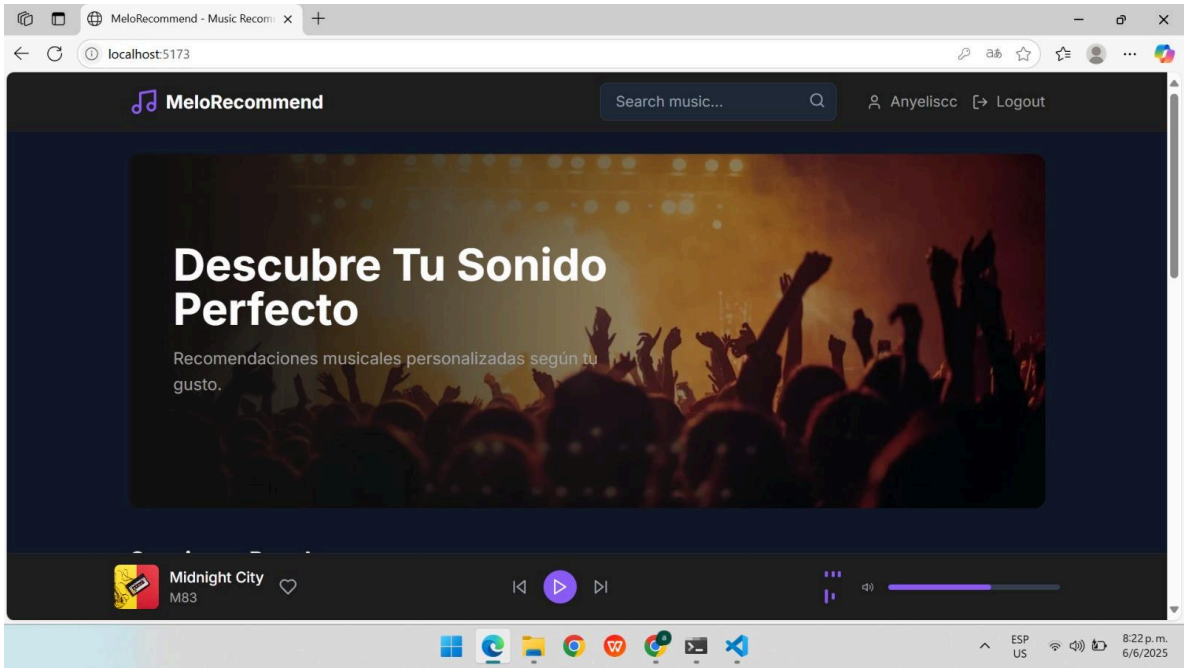
- **Crear Cuenta e Iniciar Sesión**





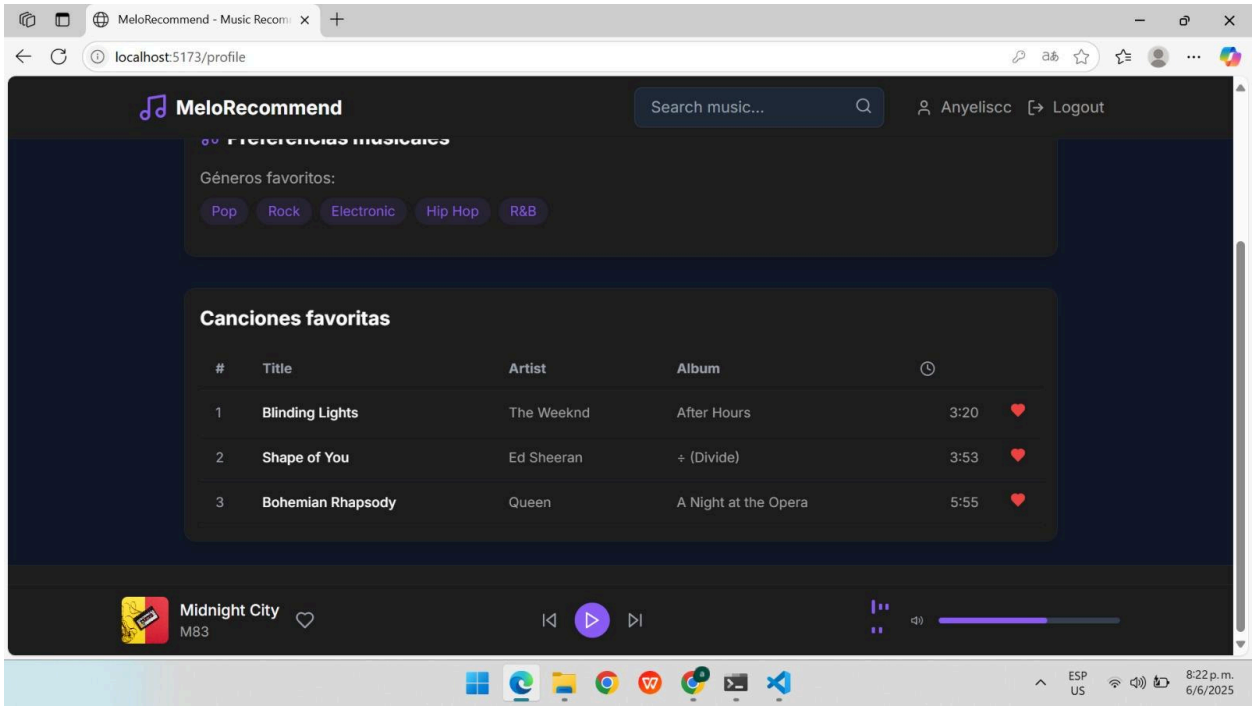
Permite a los usuarios registrarse en la página colocando todos los datos que son solicitados.

- **Inicio de Sesión**



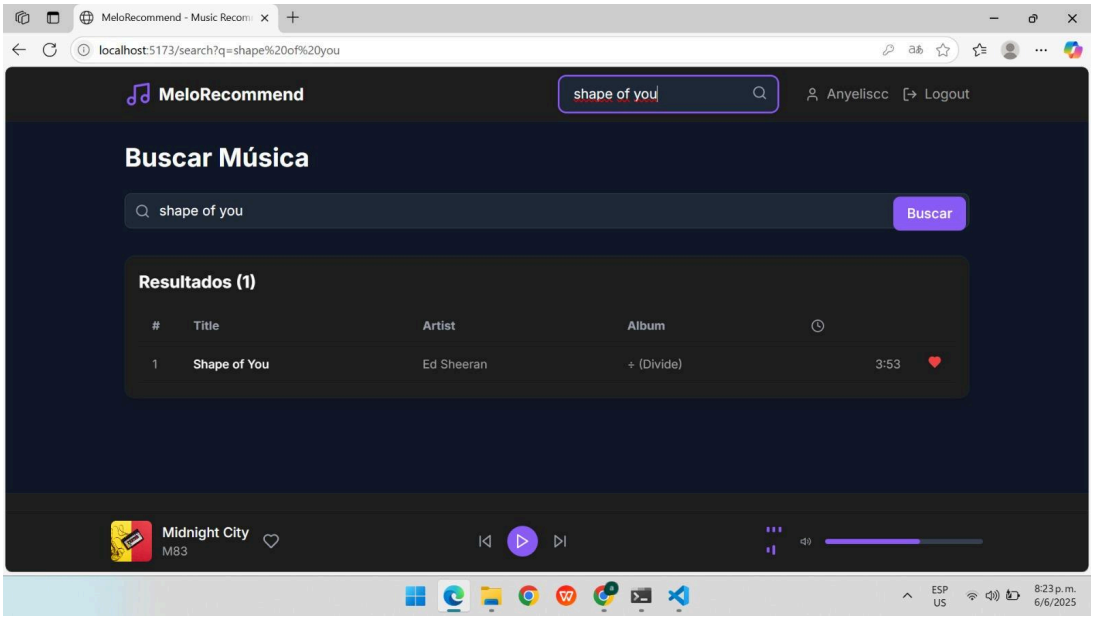
Es la vista donde se realiza efectivamente la autenticación de los datos ingresados por el usuario. Si los datos coinciden con los almacenados en la base de datos, se permite el acceso a la aplicación.

- **Apartado de Favoritos**



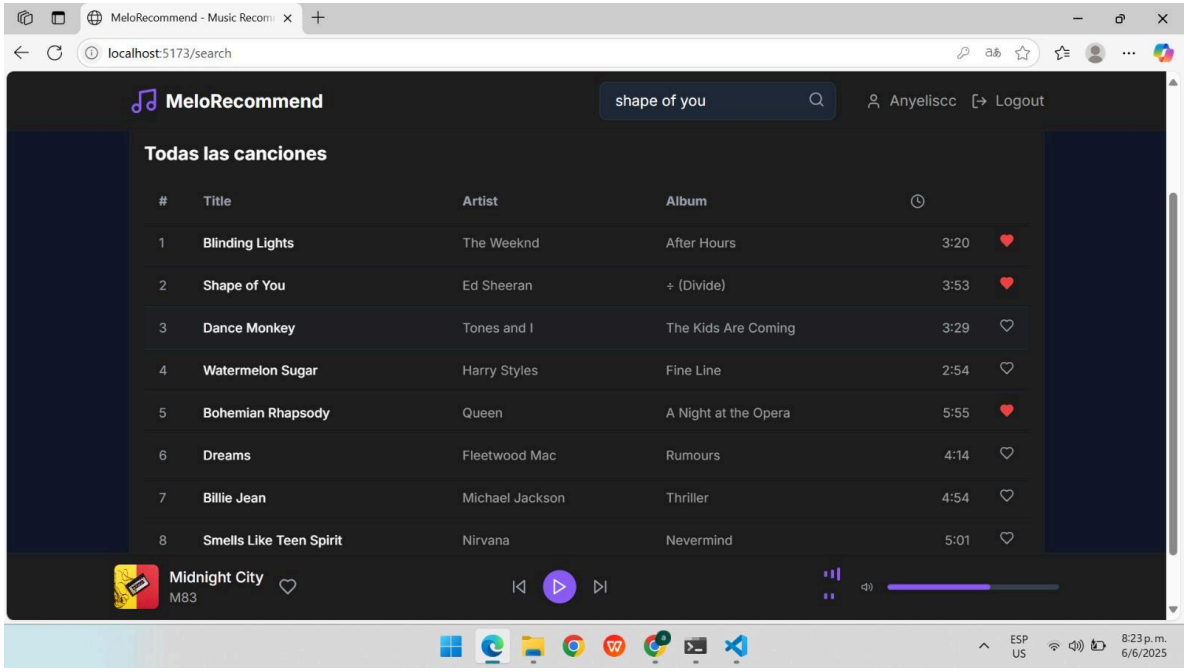
Le permite al usuario poder agrupar las canciones que más le gusten en un solo apartado, siendo así más rápido y sencillo de encontrar. Este apartado le indica al sistema que puede mostrarle al usuario distintas recomendaciones de canciones, según sus búsquedas personalizadas.

- **Búsqueda de música**



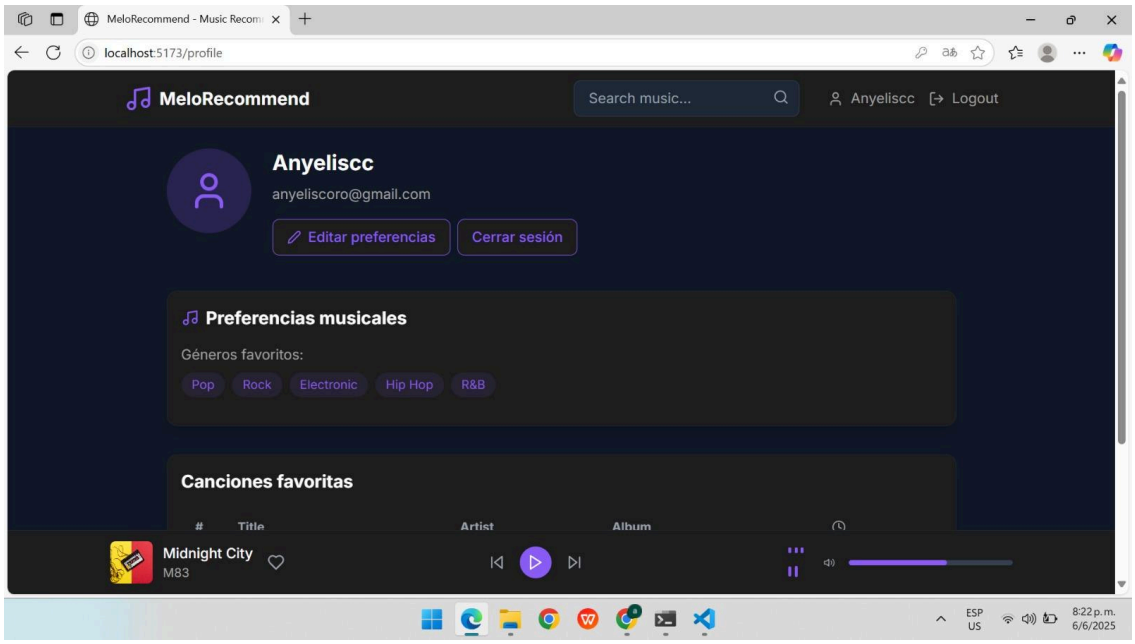
Permite al usuario consultar y filtrar canciones dentro de la página mediante palabras clave. Se puede buscar canciones por nombre, artista, género y álbum.

- **Simulación de reproducción y lista de canciones disponibles**



Simula la reproducción de una canción seleccionada por el usuario desde la lista de disponibles. Muestra el listado completo de canciones almacenadas en la base de datos, información de título, género, álbum y artista. Permite al usuario visualizar la información de cada canción y reproducirla desde allí.

- **Perfil de Usuario**



Permite al usuario visualizar y modificar sus datos personales registrados, como nombre y correo, además de visualizar sus canciones marcadas como favoritas y los géneros musicales que más suele escuchar.

Conclusiones

- Se logró implementar una aplicación web funcional de reproducción musical con autenticación de usuarios y gestión de contenido.

- La utilización de Cassandra como base de datos NoSQL permitió garantizar alta disponibilidad y un desempeño óptimo en operaciones de lectura y escritura.
- La integración de React y Flask facilitó la creación de un sistema modular, separado por capas y fácilmente escalable
- El proyecto demuestra que es posible desarrollar aplicaciones multimedia distribuidas usando tecnologías open source y modernas, sin depender de bases de datos relacionales tradicionales.