**Trabajo Práctico Integrador**



**Soporte a la gestión de datos con programación visual**

**Profesores:**

Juan Ignacio Torres

Mario Castagnino

**Grupo 2 - Integrantes:**

Angelo Faraci - 49604

Nahuel Verdera - 49602

[**Link al repositorio**](https://github.com/angelofaraci/tpi-soporte)

**Narrativa del Proyecto**

Sistema de Búsqueda y Gestión de Álbumes Musicales

Este proyecto es una aplicación web desarrollada con Django que permite a los usuarios buscar información detallada sobre álbumes musicales utilizando APIs externas. El sistema ofrece funcionalidades de gestión de usuarios, historial de búsquedas y una lista personalizada de "escuchar más tarde" para que los usuarios puedan organizar y gestionar su experiencia musical.

La aplicación surge de la necesidad de centralizar la búsqueda de información musical, proporcionando una interfaz intuitiva que permita a los usuarios encontrar álbumes, explorar su información detallada y mantener un registro personalizado de su actividad musical.

**Abstract**

El Sistema de Búsqueda y Gestión de Álbumes Musicales es una aplicación web que integra servicios de búsqueda musical con funcionalidades de gestión personal. Desarrollado con Django y utilizando APIs externas para obtener información musical, el sistema permite realizar búsquedas avanzadas de álbumes, mantener un historial de consultas, y gestionar listas personalizadas de reproducción futura. La plataforma incluye un sistema completo de autenticación de usuarios y proporciona una experiencia personalizada para cada usuario registrado.

**Requerimientos Funcionales**

RF001 - Gestión de Usuarios

* El sistema debe permitir el registro de nuevos usuarios
* El sistema debe permitir el inicio de sesión con credenciales válidas
* El sistema debe permitir el cierre de sesión seguro
* El sistema debe mantener sesiones activas de usuarios autenticados

RF002 - Búsqueda de Álbumes

* El sistema debe permitir buscar álbumes por nombre
* El sistema debe mostrar resultados detallados de búsqueda
* El sistema debe ignorar los álbumes guardados en la lista “Escuchar más tarde”
* El sistema debe integrar con APIs externas para obtener información musical
* El sistema debe mostrar información como título, artista, año, género, etc.

RF003 - Historial de Búsquedas

* El sistema debe registrar automáticamente las búsquedas de usuarios autenticados
* El sistema debe permitir visualizar el historial de búsquedas personal
* El sistema debe permitir eliminar entradas específicas del historial
* El sistema debe asociar cada búsqueda con el usuario que la realizó

RF004 - Lista "Escuchar Más Tarde"

* El sistema debe permitir agregar álbumes a una lista personal
* El sistema debe permitir eliminar álbumes de la lista personal
* El sistema debe permitir marcar álbumes como "escuchados" o "no escuchados"
* El sistema debe mostrar el estado actual de cada álbum en la lista

RF005 - Interfaz de Usuario

* El sistema debe proporcionar una interfaz web responsiva
* El sistema debe mostrar resultados de búsqueda de forma clara y organizada
* El sistema debe proporcionar navegación intuitiva entre funcionalidades

**Requerimientos No Funcionales**

RNF001 - Seguridad

* El sistema debe implementar autenticación segura de usuarios
* Las contraseñas deben estar encriptadas en la base de datos
* El sistema debe prevenir ataques de inyección SQL
* Las sesiones deben tener tiempo de expiración configurado

RNF002 - Usabilidad

* La interfaz debe ser intuitiva y fácil de usar
* El sistema debe proporcionar mensajes de error claros
* La navegación debe ser consistente en toda la aplicación

RNF003 - Disponibilidad

* El sistema debe estar disponible la mayoría del tiempo
* El sistema debe manejar errores de APIs externas correctamente
* El sistema debe proporcionar mensajes informativos cuando los servicios externos no estén disponibles

RNF004 - Compatibilidad

* El sistema debe funcionar en navegadores modernos (Chrome, Firefox, Safari, Edge)

**Stack Tecnológico**

Backend

* Framework: Django 4.x (Python)
* Base de Datos: PostgreSQL
* ORM: Django ORM
* Autenticación: Django Authentication System

Frontend

* Templates: Django Templates
* CSS: CSS personalizado
* JavaScript: JavaScript vanilla (para interacciones dinámicas)

Infraestructura

* Contenedor: Docker
* Control de Versiones: Git

APIs Externas

* APIs de servicios musicales para obtener información de álbumes (Discogs)
* Herramientas de Desarrollo (Visual Studio Code)
* Gestor de Dependencias: pip + requirements.txt
* Variables de Entorno: python-decouple
* Migraciones: Django Migrations

**Reglas de Negocio**

RN001 - Autenticación

* Solo usuarios registrados pueden acceder a funcionalidades de historial y listas personales
* Los usuarios no autenticados no pueden realizar busquedas
* Las sesiones expiran automáticamente por seguridad

RN002 - Búsquedas

* Las búsquedas se registran en un historial personal
* Las búsquedas vacías no se procesan
* El sistema limita las búsquedas por usuario para prevenir abuso

RN003 - Historial

* Cada usuario solo puede ver y gestionar su propio historial
* Las entradas del historial se pueden eliminar individualmente
* El historial tiene un límite máximo de entradas por usuario

RN004 - Lista "Escuchar Más Tarde"

* Cada usuario mantiene su propia lista personal
* No se pueden agregar álbumes duplicados a la lista
* Los álbumes pueden tener estados: "por escuchar" y "escuchado"

RN005 - Datos

* La información de álbumes se obtiene en tiempo real de APIs externas
* Los datos personales (historial, listas) se almacenan localmente
* Se mantiene la integridad referencial entre usuarios y sus datos

**Casos de Uso Principales**

CU001 - Registrar Usuario

* Actor: Usuario no registrado
* Precondición: El usuario no tiene cuenta en el sistema
* Flujo Principal:
  + El usuario accede a la página de registro
  + El usuario completa el formulario con datos válidos
  + El sistema valida la información
  + El sistema crea la cuenta y autentica al usuario
  + El usuario es redirigido a la página principal

CU002 - Buscar Álbum

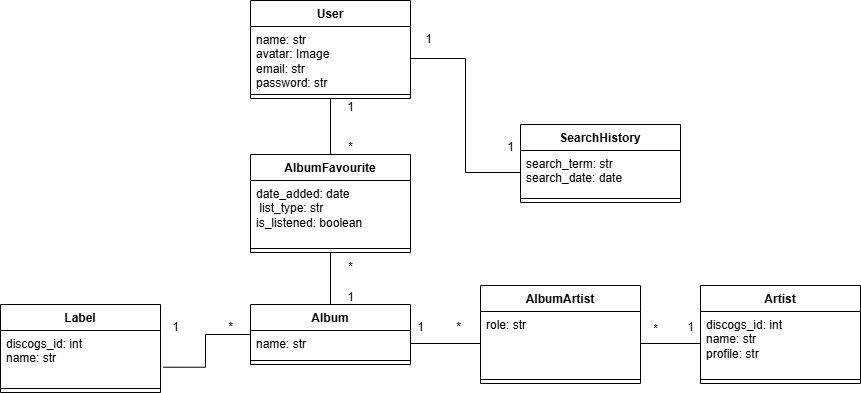
* Actor: Usuario autenticado
* Precondición: El usuario está en la página de búsqueda
* Flujo Principal:
  + El usuario ingresa el nombre del álbum a buscar
  + El usuario envía la búsqueda
  + El sistema consulta las APIs externas
  + El sistema muestra los resultados encontrados
  + Se registra la búsqueda en el historial

CU003 - Gestionar Lista "Escuchar Más Tarde"

* Actor: Usuario autenticado
* Precondición: El usuario ha iniciado sesión
* Flujo Principal:
  + El usuario busca un álbum
  + El usuario selecciona "Agregar a Escuchar Más Tarde"
  + El sistema agrega el álbum a la lista personal
  + El usuario puede ver su lista completa
  + El usuario puede marcar álbumes como escuchados o eliminarlos

CU004 - Consultar Historial

* Actor: Usuario autenticado
* Precondición: El usuario ha realizado búsquedas anteriormente
* Flujo Principal:
  + El usuario accede a su historial de búsquedas
  + El sistema muestra las búsquedas
  + El usuario puede eliminar entradas específicas
  + El sistema actualiza el historial

**Modelo de dominio**

Restricción de unicidad

* Un usuario no puede tener el mismo álbum duplicado en su lista

Reglas de Integridad

* Eliminación en cascada: Si se elimina un usuario, se eliminan sus búsquedas y listas
* Validación de unicidad en combinaciones usuario-álbum para ListenLater

**Documentación de librerías**

Django:<https://docs.djangoproject.com/en/5.2/>

Postgres: <https://www.postgresql.org/docs/>

Docker: <https://docs.docker.com/>