

Medrano Chacolla Sebastián Jorge

Sistema Propuesto para Catalogar Canciones para YouTube Music

Base de Datos II

4° Semestre

Índice

1 Introducción.....	5
1.1 Antecedentes.....	5
1.2 Definición de Procesos y Requisitos de la Fase 1.....	6
1.2.1 Configurar Perfil de Artista.....	6
1.2.1 Administrar Álbumes.....	6
1.2.3 Administrar Canciones.....	7
1.2.4 Catálogar y Búscar.....	7
1.2.2 Definición de Procesos y Requisitos de la Fase 2.....	8
1.2.3 Administrar Comentarios de Canciones.....	8
1.2.4 Curar Biblioteca Personal mediante "Me Gusta".....	8
2 Objetivos.....	8
2.1 Objetivo General de Todas las Fases.....	8
2.2 Objetivos Específicos de la Fase 1.....	9
2.2.1 Proceso de Catalogación de Canciones por detalles.....	9
2.2.2 Proceso de Asignación de Canciones a Playlists.....	9
2.3 Objetivos Específicos de la Fase 2.....	9
2.3.1 Proceso de Administrar Comentarios de Canciones.....	9
2.3.2 Proceso de Curar Biblioteca Personal mediante "Me Gusta".....	10
3 Diccionario de Datos.....	11
3.1 Descripción de Entidades y Relaciones de la Fase 1.....	11
3.1.1 La Entidad Canciones.....	11
3.1.2 La Entidad Artistas.....	11
3.1.3 La entidad redes sociales.....	12
3.1.4 La entidad álbumes.....	13
3.1.5 La entidad listas de reproducción.....	13
3.1.6 La entidad usuarios.....	14
3.1.7 La relación listas-canciones.....	15
3.1.8 La relación canciones-artistas.....	15

3.2 Descripción de Entidades y Relaciones de la Fase 2.....	16
3.2.1 La entidad comentarios (comments).....	16
3.2.2 La relación user_song_ratings.....	16
3.2.3 La Entidad Artistas (Corregida).....	17
4 Modelo Entidad-Relación.....	18
4.1 Diagrama Entidad-Relación de la fase 1.....	18
4.2 Diagrama Entidad-Relación de la fase 2.....	20
5 Modelo Relacional.....	21
5.1 Diagrama del Modelo Relacional de la fase 1.....	21
5.2 Diagrama del Modelo Relacional de la fase 2.....	22
6 Diagrama de Navegación.....	23
6.1 Diagrama de Navegación de la Fase 1.....	23
6.2 Diagrama de Navegación de la Fase 2.....	24
7 Diseño de las Vistas.....	25
8 Esquema y Scripts SQL.....	26
8.1 Esquemas y Scripts de la Base de Datos de la Fase 1.....	26
8.1.1 Script del Esquema.....	26
8.1.2 Scripts para el Proceso de Configurar Perfil de Artista.....	29
8.1.3 Scripts para el Proceso de Administrar Álbumes.....	30
8.1.4 Scripts para el Proceso de Administrar Canciones.....	31
8.1.5 Scripts para el Proceso de Catálogar y Búscar.....	32
8.1.6 Scripts para el Proceso de Asignar Canciones a Playlists.....	34
8.2 Esquema y Scripts de la Base de Datos para Fase II.....	35
8.2.1 Script del Esquema.....	35
8.2.2 Scripts para el Procesos de Administrar Comentarios de Canciones.....	37
8.2.3 Scripts para el Proceso de Curar Biblioteca Personal mediante "Me Gusta"	
.....	38
9 Intregación de Plantilla y Funcionamiento del Sistema.....	39
9.1 Menú de Inicio de Sesión y/o Registro.....	39
9.2 Página Principal (Home, Todas la Canciones).....	40

9.2.1 Página para Ver Detalles de una Canción.....	40
9.3 Página de Todos los Álbumes.....	42
9.3.1 Página para Ver Detalles Contenido de un Álbum.....	42
9.4 Página de Todos los Artistas.....	43
9.4.1 Página para Ver Detalles Contenido de un Artista.....	44
9.5 Página de Todas las Playlists.....	46
9.5.1 Página para Ver Contenido de una Playlist.....	46
9.6 Página de Canciones Favoritas.....	47
9.7 Página del Formulario para ser Artista.....	47
9.8 Página del “Studio”.....	48
9.8.1 Página para Cargar una Canción.....	48
9.8.2 Página para Administrar Canciones.....	49
9.8.3 Página para Crear un Álbum.....	51
9.8.4 Página para Administrar Álbumes.....	51
9.8.5 Página para Agregar una Red Social.....	52
10 Integración se Seguridad.....	53
10.1 Seguridad para la Contraseña del Usuario.....	53
10.1.1 Función de Registro.....	53
10.1.2 Características Importantes de Bcrypt.....	54
10.2 Seguridad para la Cookie del Identificador del Usuario.....	54
10.2.1 Función de Inicio de Sesión.....	54
10.2.2 Encriptación Autenticada (AES-GCM).....	56
11 Conclusión.....	58
11.1 Conclusión de la Fase 1.....	58
11.2 Conclusión de la Fase 2.....	58
11.3 Conclusión de la Fase 3.....	59
11.4 Conclusión de la Fase 4.....	59

Sistema Propuesto de Catálogo Musical: Caso YouTube Music

1 Introducción

1.1 Antecedentes

El nicho de la transmisión de música digital (streaming) fue dominado durante años por pioneros como Spotify y Apple Music, que establecieron el modelo de suscripción para acceder a vastos catálogos de música. Paralelamente, Google mantenía dos ofertas separadas: Google Play Music, un servicio de streaming tradicional que también permitía a los usuarios subir su propia biblioteca personal, y YouTube, que, si bien era una plataforma de video, se había convertido en el servicio de facto más grande del mundo para el consumo de música (a través de videos musicales oficiales, audio oficial, lyric videos y contenido generado por usuarios).

Existía una demanda clara por unificar estas experiencias. Los usuarios estaban divididos entre una plataforma de streaming de audio tradicional y una plataforma de video con un catálogo musical inmenso pero desorganizado. Faltaba una solución que combinara el catálogo "oficial" de las discográficas con el universo de contenido musical de YouTube.

YouTube Music nace como un proyecto para llenar este vacío, con un relanzamiento importante en 2018. La motivación principal era crear un servidor de música que fuera:

- **Exhaustivo:** Combinando el catálogo de música con licencia de Google Play Music con el interminable archivo de contenido musical de YouTube.
- **Inteligente:** Aprovechando la infraestructura de búsqueda e inteligencia artificial de Google para ofrecer recomendaciones potentes y una búsqueda semántica (permitiendo buscar canciones por letras o descripciones vagas).
- **Moderno:** Incorporando una interfaz de usuario fresca y responsive, centrada tanto en el audio como en el video, permitiendo a los usuarios cambiar sin interrupciones entre el modo de solo audio y el video musical.

- **Unificado:** Con el objetivo a largo plazo de reemplazar a Google Play Music y consolidar toda la oferta de música de Google bajo la potente y reconocida marca de YouTube.

Desde su relanzamiento, YouTube Music ha ganado rápidamente popularidad, aprovechando la base de miles de millones de usuarios de YouTube y su integración con el ecosistema de Google (como los altavoces inteligentes) para posicionarse como uno de los principales competidores en la industria del streaming.

1.2 Definición de Procesos y Requisitos de la Fase 1

1.2.1 Configurar Perfil de Artista

- **RF1. Convertirse en Artista:** El sistema debe permitir a un usuario existente tomar el rol de "artista" mediante el llenado y envío de un formulario de artista.
- **RF2. Registrar Redes Sociales:** El sistema debe permitir a los usuarios con rol de "artista" añadir enlaces a sus redes sociales en su perfil.
- **RF3. Visualizar Perfil de Artista:** El sistema debe mostrar el perfil público del artista, incluyendo su nombre, biografía y sus redes sociales.

1.2.1 Administrar Álbumes

- **RF1. Registrar Nuevo Álbum:** El sistema debe permitir a un artista registrar un nuevo álbum, especificando un nombre, año de lanzamiento y portada del álbum.
- **RF2. Editar Detalles del Álbum:** El sistema debe permitir al artista editar el nombre, año de lanzamiento y portada de un álbum existente.
- **RF3. Eliminar Álbum:** El sistema debe permitir al artista eliminar un álbum.
- **RF4. Visualizar Álbumes del Artista:** El sistema debe mostrar una lista de todos los álbumes registrados por un artista en su perfil.

1.2.3 Administrar Canciones

- **RF1. Cargar Nueva Canción:** El sistema debe permitir a un artista cargar un archivo de audio. Al cargar, el sistema debe asignarla automáticamente al artista que la sube.
- **RF2. Registrar Detalles de Canción:** El sistema debe permitir al artista registrar los detalles de una canción cargada, incluyendo: nombre, género musical, idioma y una portada (si es diferente a la del álbum).
- **RF3. Asignar Canción a Álbum:** Al registrar los detalles (o al editarlos), el sistema debe permitir al Artista asignar la canción a uno de sus álbumes existentes (creados en el Proceso 1.2.2) o marcarla como que no pertenece a “ninguno”.
- **RF4. Administrar Colaboradores:** El sistema debe permitir al Artista principal de la canción añadir a otros artistas (usuarios con rol de Artista) como colaboradores en la canción.
- **RF5. Editar Detalles de Canción:** El sistema debe permitir al Artista editar todos los detalles de una canción (nombre, género, álbum al que pertenece, colaboradores, etc.).
- **RF6. Eliminar Canción:** El sistema debe permitir al artista eliminar una canción de la plataforma.

1.2.4 Catálogar y Búscar

- **RF1. Visualización de Detalles de Canción:** El sistema debe mostrar a todos los usuarios los detalles de una canción: nombre, portada, género, idioma, artista(s) y álbum (si pertenece a uno).
- **RF2. Visualización de Contenido por Artista:** El sistema debe mostrar una lista de todas las canciones (y álbumes) asociados a un artista cuando un usuario visita su perfil.
- **RF2. Visualización de Canciones por Álbum:** El sistema debe mostrar una lista de todas las canciones que pertenezcan a un álbum.

- **RF3. Búsqueda Básica de Canciones:** El sistema debe permitir a cualquier usuario buscar canciones por nombre, nombre del artista o género musical.

1.2.2 Definición de Procesos y Requisitos de la Fase 2

1.2.3 Administrar Comentarios de Canciones

- **RF1. Añadir Comentario:** El sistema debe permitir al usuario escribir y guardar un comentario en una canción específica.
- **RF2. Ver Comentarios:** El sistema debe mostrar todos los comentarios de una canción.
- **RF3. Actualizar Comentario:** El sistema debe permitir al usuario modificar un comentario existente en una canción específica.

1.2.4 Curar Biblioteca Personal mediante "Me Gusta"

- **RF1. Marcar "Me Gusta" en una Canción:** El sistema debe permitir al usuario marcar una canción específica como "Me gusta" (o quitar esa marca).
- **RF2. Visualizar Playlist Automática "Canciones que me gustan":** El sistema debe mostrar automáticamente al usuario una lista de todas las canciones que ha marcado con "Me gusta".
- **RF3. Conteo Público de "Me Gusta":** El sistema debe ser capaz de mostrar cuántos "Me gusta" totales tiene una canción por parte de todos los usuarios.

2 Objetivos

2.1 Objetivo General de Todas las Fases

Implementar un sistema de catálogo musical integral para un servicio de música en streaming, que permita a los usuarios administrar su música de forma completa, desde la carga y catalogación de canciones, la creación y visualización de listas de reproducción personalizadas, hasta la gestión de comentarios en las canciones y la curación de su biblioteca personal mediante la función "Me gusta".

2.2 Objetivos Específicos de la Fase 1

2.2.1 Proceso de Catalogación de Canciones por detalles.

- Desarrollar e implementar una funcionalidad que permita a los usuarios cargar archivos de canciones a la plataforma de manera sencilla.
- Garantizar la visualización completa de todos los detalles de cada canción cargada.
- Implementar un sistema de búsqueda básica que permita a los usuarios encontrar canciones utilizando detalles específicos (por ejemplo, por artista o por género).

2.2.2 Proceso de Asignación de Canciones a Playlists.

- Desarrollar e implementar un sistema que permita a los usuarios crear nuevas listas de reproducción, asignándoles un nombre único.
- Asegurar que el sistema permita a los usuarios seleccionar y añadir canciones a cualquier lista de reproducción existente.
- Garantizar la visualización clara de todas las listas de reproducción creadas por el usuario.
- Implementar una funcionalidad que permita a los usuarios ver las canciones contenidas en una lista de reproducción específica.

2.3 Objetivos Específicos de la Fase 2

2.3.1 Proceso de Administrar Comentarios de Canciones.

- Desarrollar e implementar una funcionalidad que permita a los usuarios escribir y guardar comentarios en una canción específica.
- Permitir que el sistema muestre todos los comentarios asociados a una canción.
- Implementar una función que permita a los usuarios modificar sus propios comentarios existentes en una canción específica.

2.3.2 Proceso de Curar Biblioteca Personal mediante "Me Gusta".

- Desarrollar e implementar una funcionalidad que permita a los usuarios marcar una canción específica como "Me gusta" (o quitar esa marca).
- Asegurar que el sistema muestre automáticamente al usuario una lista de todas las canciones que ha marcado con "Me gusta".
- Implementar una función para que el sistema sea capaz de mostrar cuántos "Me gusta" totales tiene una canción por parte de todos los usuarios.

3 Diccionario de Datos

3.1 Descripción de Entidades y Relaciones de la Fase 1

3.1.1 La Entidad Canciones

- La canción es una entidad que representa una pieza musical individual.
- Es una entidad fuerte, ya que existe de forma independiente.
- Los atributos que tiene la canción son su identificador, título, idioma, fecha de lanzamiento, duración, género, ruta del archivo de la canción, ruta de la portada y el identificador del álbum al que pertenece.

Nombre		songs			
Tipo de Tabla	Entidad	OK	Relación		
Descripción de atributos	Clasificación de atributos	Dominio	Tipo de atributo	Longitud	Clave
id_song	Identificador	integer	Identificativo	≥ 1	Principal
title	Descriptor	varchar	Descriptivo	A - Z, a - z	
language	Descriptor	varchar	Descriptivo	A - Z, a - z	
release_date	Descriptor	timestamp	Descriptivo	0001-01- 01 9999-12- 31	
duration	Descriptor	real	Descriptivo	≥ 0	
genre	Descriptor	varchar	Descriptivo	A - Z, a - z	
song_path	Descriptor	varchar	Descriptivo	ASCII	
cover_path	Descriptor	varchar	Descriptivo	ASCII	
id_album	Identificador	integer	Identificativo	≥ 1	Foránea

3.1.2 La Entidad Artistas

- El artista es una entidad a la que pertenecen canciones y de la también se almacena información importante.

- Es una entidad fuerte.
- Los atributos que tiene el artista son su identificador, nombre, nacionalidad, biografía y fecha de debut.

artists					
Tipo de Tabla	Entidad	OK	Relación		
Descripción de atributos	Clasificación de atributos	Dominio	Tipo de atributo	Longitud	Clave
id_artist	Identificador	integer	Identificativo	>= 1	Principal
name	Descriptor	varchar	Descriptivo	A - Z, a - z	
nationality	Descriptor	varchar	Descriptivo	A – Z, a - z	
biography	Descriptor	varchar	Descriptivo	ASCII	
debut_date	Descriptor	timestamp	Descriptivo	0001-01-01 9999-12-31	.

3.1.3 La entidad redes sociales

- La red social es una entidad que almacena información sobre las plataformas sociales asociadas a un artista.
- Es una entidad débil, ya que depende de la entidad "artista".
- Los atributos que tiene la red social son su identificador, nombre, URL y el identificador del artista al que pertenece.

social_networks					
Tipo de Tabla	Entidad	OK	Relación		
Descripción de atributos	Clasificación de atributos	Dominio	Tipo de atributo	Longitud	Clave
id_red	Identificador	integer	Identificativo	>= 1	Principal
name	Descriptor	varchar	Descriptivo	A - Z, a - z	
url	Descriptor	varchar	Descriptivo	ASCII	
id_artist	Identificador	integer	Identificativo	>= 1	Foránea

3.1.4 La entidad álbumes

- El álbum es una entidad que agrupa canciones y proporciona información adicional sobre ellas.
- Es una entidad débil, ya que depende de la entidad "cancion".
- Los atributos que tiene el álbum son su identificador, nombre, fecha de lanzamiento, ruta de la portada y el identificador de la canción a la que pertenece.

albums					
Tipo de Tabla	Entidad	OK	Relación		
Descripción de atributos	Clasificación de atributos	Dominio	Tipo de atributo	Longitud	Clave
id_album	Identificador	integer	Identificativo	>= 1	Principal
name	Descriptor	varchar	Descriptivo	A – Z, a - z	
release_date	Descriptor	timestamp	Descriptivo	0001-01-01 9999-12-31	
cover_path	Descriptor	varchar	Descriptivo	ASCII	
id_artist	Identificador	integer	Identificativo	>= 1	Foránea

3.1.5 La entidad listas de reproducción

- La lista de reproducción es una entidad que permite a los usuarios organizar y agrupar canciones de forma personalizada.
- Es una entidad débil, ya que su existencia está ligada a un usuario y una canción.
- Se diferencia del álbum por su relación N:M con las canciones, lo que significa que una canción puede estar en varias listas y una lista puede contener varias canciones.
- Los atributos que tiene la lista de reproducción son su identificador, nombre, descripción, fecha de modificación, fecha de creación y el identificador del usuario al que pertenece.

Nombre		playlists			
Tipo de Tabla	Entidad	OK	Relación		
Descripción de atributos	Clasificación de atributos	Dominio	Tipo de atributo	Longitud	Clave
id_playlist	Identificador	integer	Identificativo	>= 1	Principal
name	Descriptor	varchar	Descriptivo	A - Z, a - z	
description	Descriptor	varchar	Descriptivo	ASCII	
modification_date	Descriptor	timestamp	Descriptivo	0001-01-01 9999-12-31	
creation_date	Descriptor	timestamp	Descriptivo	0001-01-01 9999-12-31	
id_user	Identificador	integer	Identificativo	>= 1	Foránea

3.1.6 La entidad usuarios

- El usuario es una entidad central en el sistema, ya que interactúa con las listas de reproducción y otras funcionalidades.
- Es una entidad fuerte, lo que significa que existe de forma independiente.
- Los atributos que tiene el usuario son su identificador, nombre, contraseña, fecha de creación y correo electrónico.

Nombre		usuarios			
Tipo de Tabla	Entidad	OK	Relación		
Descripción de atributos	Clasificación de atributos	Dominio	Tipo de atributo	Longitud	Clave
id_user	Identificador	integer	Identificativo	>= 1	Principal
name	Descriptor	varchar	Descriptivo	A - Z, a - z	
password	Descriptor	varchar	Descriptivo	ASCII	
creation_date	Descriptor	timestamp	Descriptivo	0001-01-01 9999-12-31	
email	Descriptor	varchar	Descriptivo	ASCII	

3.1.7 La relación listas-canciones

- Esta entidad de relación se utiliza para establecer la conexión N:M entre lista de reproducciones y canciones.
- Es una entidad débil, ya que su existencia depende de las entidades lista_reproduccion y cancion.
- Los atributos que tiene son los identificadores de la lista de reproducción y de la canción, que juntos forman la clave primaria compuesta.

Nombre	playlists_songs				
Tipo de Tabla	Relación		Relación	OK	
Descripción de atributos	Clasificación de atributos	Dominio	Tipo de atributo	Longitud	Clave
id_playlist	Identificador	integer	Identificativo	>= 1	Principal
id_song	Identificador	integer	Identificativo	>= 1	Principal

3.1.8 La relación canciones-artistas

- Esta entidad de canción-artista relación se utiliza para establecer la conexión N:M entre canciones y artistas.
- Es una entidad débil, ya que su existencia depende de las entidades cancion y artista.
- Los atributos que tiene son los identificadores del artista y de la canción, que juntos forman la clave primaria compuesta.

Nombre	songs_artists				
Tipo de Tabla	Relación		Relación	OK	
Descripción de atributos	Clasificación de atributos	Dominio	Tipo de atributo	Longitud	Clave
id_artist	Identificador	integer	Identificativo	>= 1	Principal
id_song	Identificador	integer	Identificativo	>= 1	Principal

3.2 Descripción de Entidades y Relaciones de la Fase 2

3.2.1 La entidad comentarios (comments)

- La entidad comments representa los comentarios que los usuarios pueden dejar en una canción específica.
- Es una entidad débil, ya que su existencia depende de la entidad songs (canciones) y users (usuarios).
- Los atributos que tiene el comentario son su identificador, el texto del comentario, la fecha de creación, la fecha de modificación, el identificador de la canción a la que pertenece y el identificador del usuario que lo realizó.

comments					
Nombre					
Tipo de Tabla	Entidad	OK	Relación		
Descripción de atributos	Clasificación de atributos	Dominio	Tipo de atributo	Longitud	Clave
id_comment	Identificador	integer	Identificativo	>= 1	Principal
comment_text	Descriptor	varchar	Descriptivo	ASCII	
creation_date	Descriptor	timestamp	Descriptivo	0001-01-01 9999-12-31	
modification_date	Descriptor	timestamp	Descriptivo	0001-01-01 9999-12-31	
id_song	Identificador	integer	Identificativo	>= 1	Foránea
id_user	Identificador	integer	Identificativo	>= 1	Foránea

3.2.2 La relación user_song_ratings

- Esta entidad de relación se utiliza para establecer la conexión N:M entre users (usuarios) y songs (canciones), registrando las interacciones de "Me gusta" de los usuarios con las canciones.
- Es una entidad débil, ya que su existencia depende de las entidades users y songs.

- Los atributos que tiene son los identificadores del usuario y de la canción, que juntos forman la clave primaria compuesta, un indicador booleano `is_liked` para saber si la canción fue marcada como "Me gusta", y la fecha de la interacción.

Nombre	user_song_ratings				
Tipo de Tabla	Relación		Relación	OK	
Descripción de atributos	Clasificación de atributos	Dominio	Tipo de atributo	Longitud	Clave
<code>id_user</code>	Identificador	integer	Identificativo	≥ 1	Principal
<code>id_song</code>	Identificador	integer	Identificativo	≥ 1	Principal
<code>is_liked</code>	Descriptor	boolean	Descriptivo	TRUE, FALSE	
<code>interaction_date</code>	Descriptor	timestamp	Descriptivo	0001-01-01 9999-12-31	

3.2.3 La Entidad Artistas (Corregida)

- Para esta segunda fase se añadió el indentificador del usuario a esta entidad, con el objetivo de que un usuario pueda pasar a ser un "artista" para empezar a añadir música tal como sucede en YouTube y YouTube Music.

Nombre	artists				
Tipo de Tabla	Entidad	OK	Relación		
Descripción de atributos	Clasificación de atributos	Dominio	Tipo de atributo	Longitud	Clave
<code>id_artist</code>	Identificador	integer	Identificativo	≥ 1	Principal
<code>name</code>	Descriptor	varchar	Descriptivo	A - Z, a - z	
<code>nationality</code>	Descriptor	varchar	Descriptivo	A – Z, a – z	
<code>biography</code>	Descriptor	varchar	Descriptivo	ASCII	
<code>debut_date</code>	Descriptor	timestamp	Descriptivo	0001-01-01 9999-12-31	

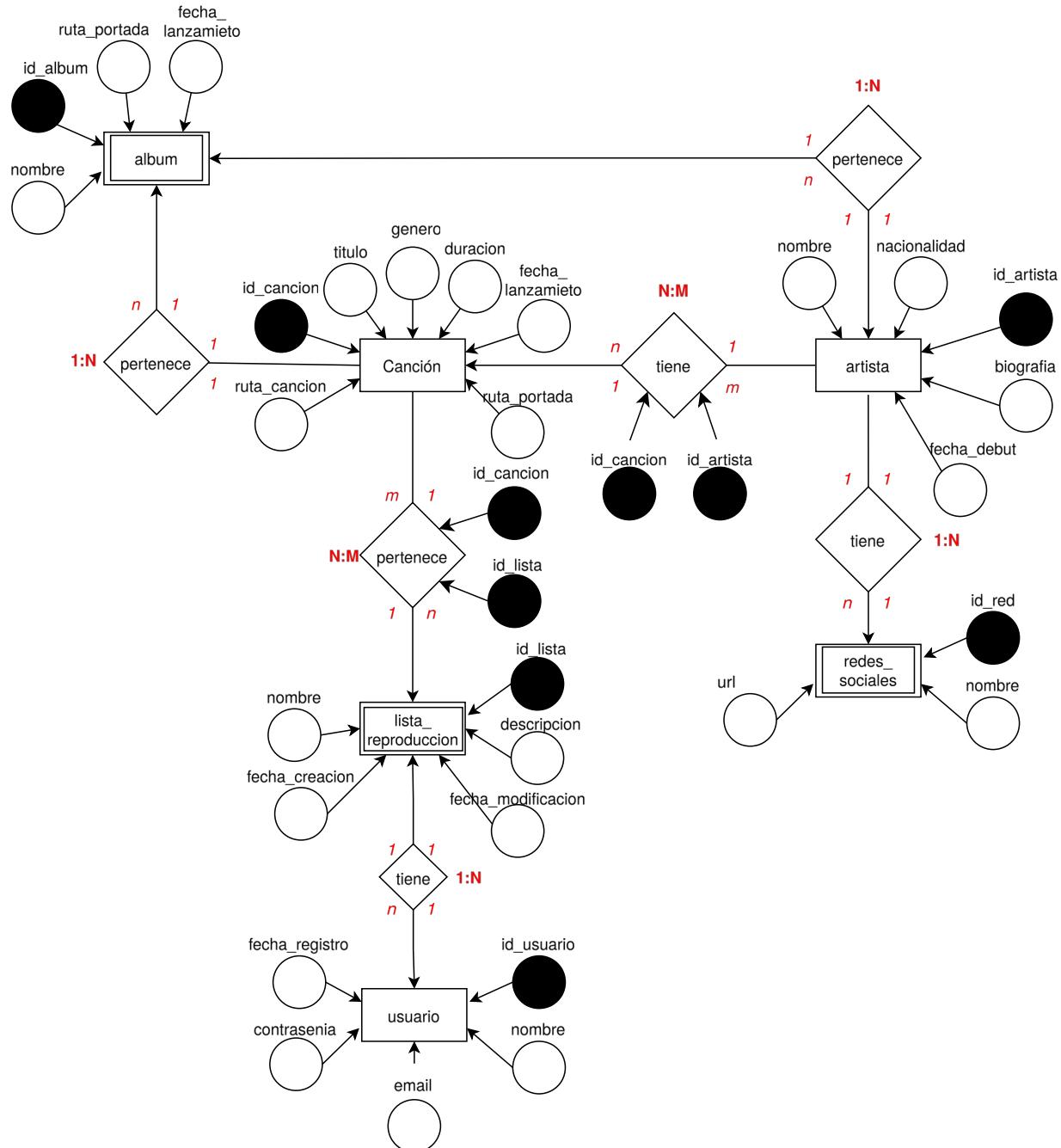
Nombre	artists				
id_user	Identificador	integer	Identificativo	>= 1	Foránea

4 Modelo Entidad-Relación

4.1 Diagrama Entidad-Relación de la fase 1

Este sería nuestro diagrama ER completo, como se puede ver ya se señala mejor como actúan las relaciones con nuestras entidades. Además que indica la cardinalidad. Para mejorar las relaciones N:M cuentas con sus respectivos identificados mientras que el resto de relaciones indican como se realizaría la interacción en el sistema.

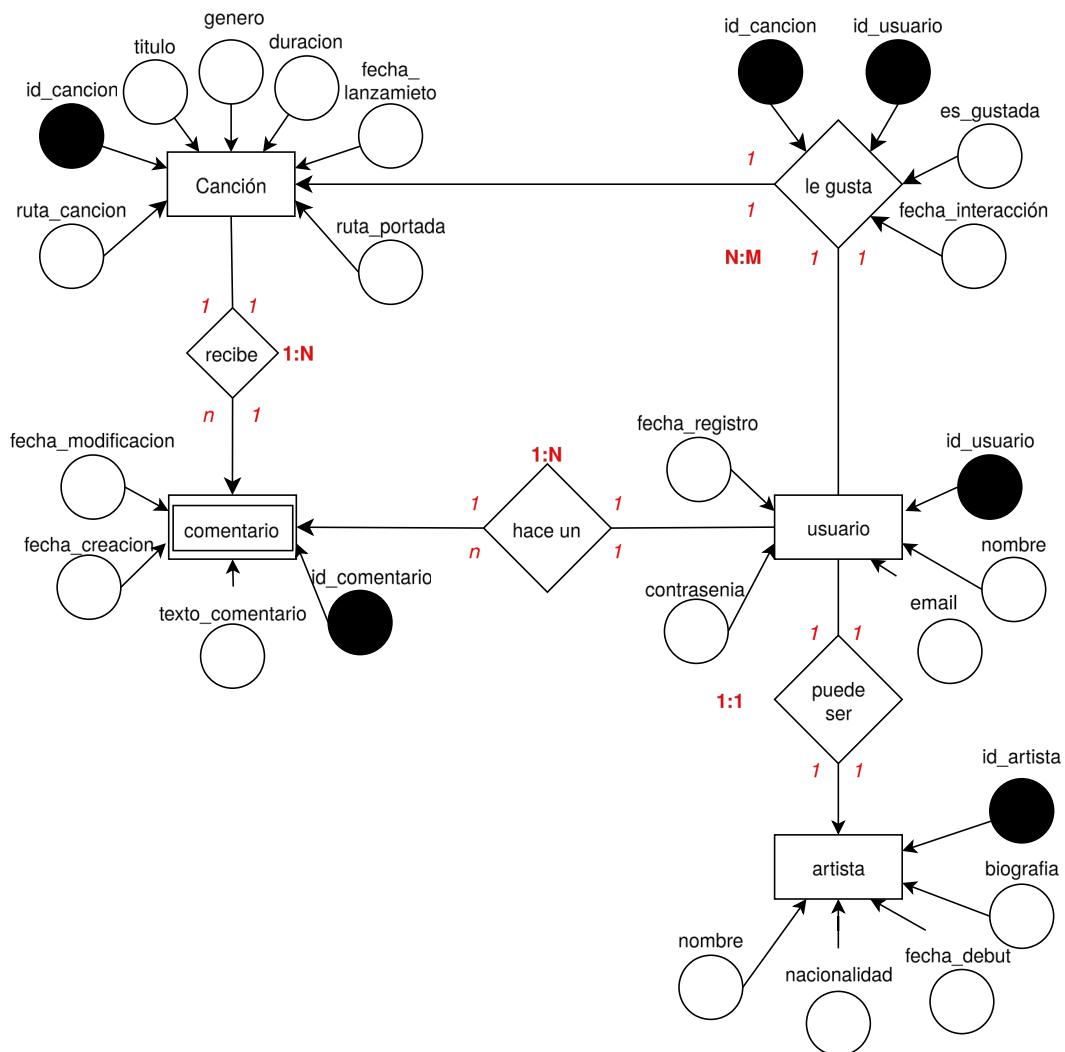
Tambien se incluyo se muestra mejor la cardinalidad y la participación que tienen todas las entidades entre si.



4.2 Diagrama Entidad-Relación de la fase 2

Este sería nuestro diagrama ER con las tablas creadas para la fase 2 del sistema propuesto, la entidad comentarios aparece entre canción y usuario como una entidad débil que depende de ambos para existir y la interacción me gusta aparece como una relación N:M que además de tener los identificados de canción y usuario, tiene el indicador para saber si es gustada y la fecha en la que se hizo la interacción.

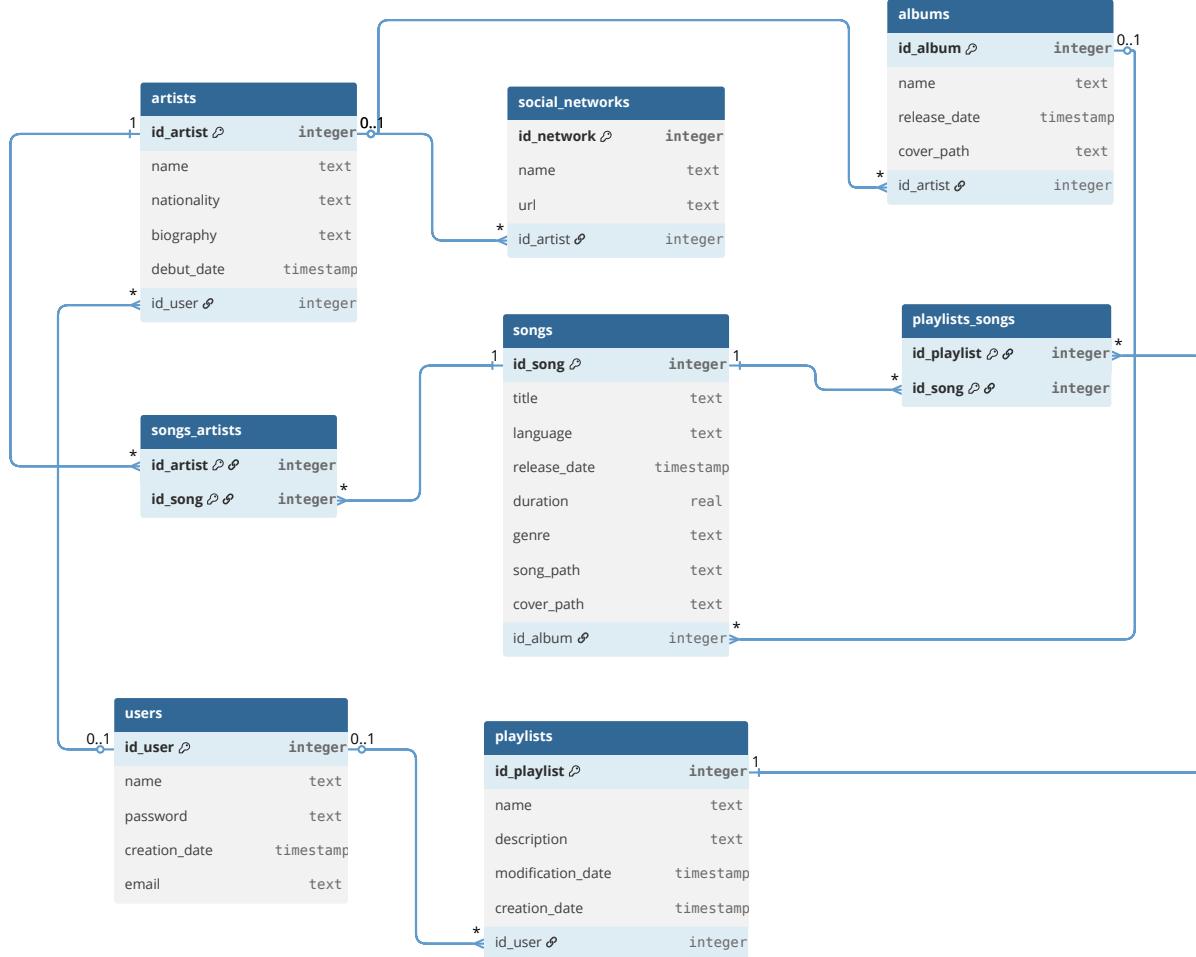
Además que las relaciones y entidades previas se mantienen, solo que en este caso para facilitar la lectura, solo se muestran las nuevas entidades y relaciones que se crearon. Y se corrigieron las relaciones de “artista”.



5 Modelo Relacional

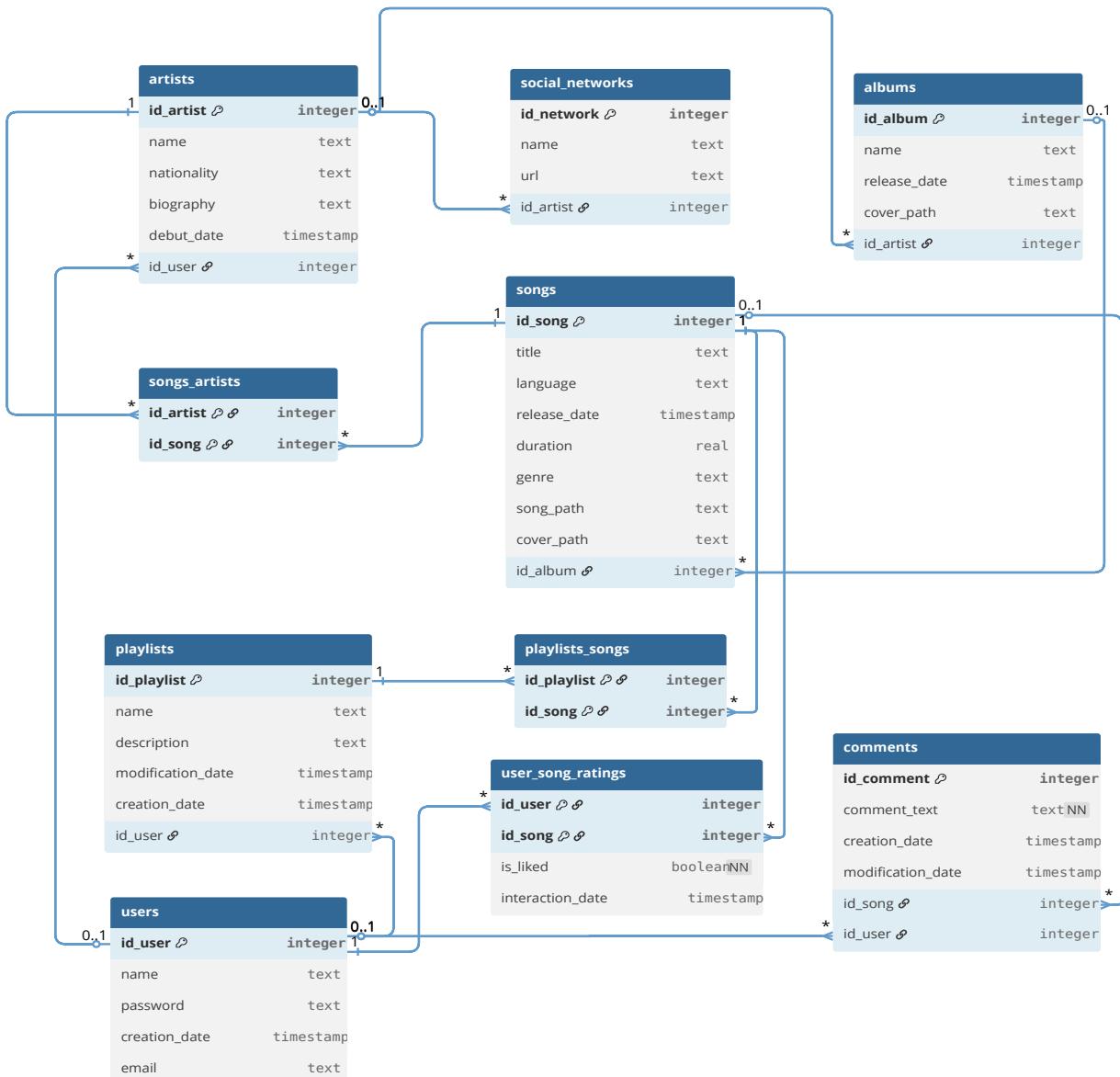
5.1 Diagrama del Modelo Relacional de la fase 1

En este diagrama se muestra como estarian nuestras tablas y esquema al momento de pasarlo a un script SQL, las relaciones N:M pasan a ser tablas.



5.2 Diagrama del Modelo Relacional de la fase 2

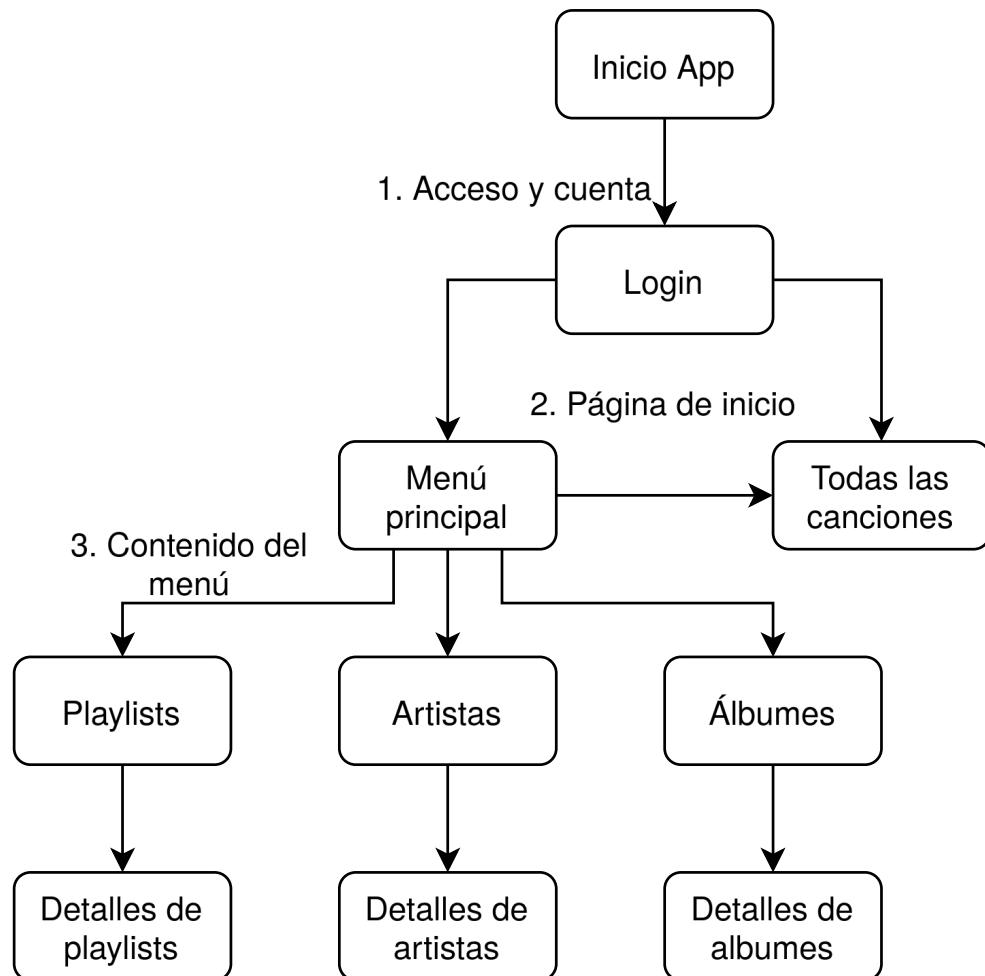
Diagrama relacional de la segunda fase donde se añaden dos tablas más que corresponden a los comentarios y la relación N:M de “me gusta”. Y se muestra la relación 1 a 1 que simboliza la conversión de usuario a artista.



6 Diagrama de Navegación

6.1 Diagrama de Navegación de la Fase 1

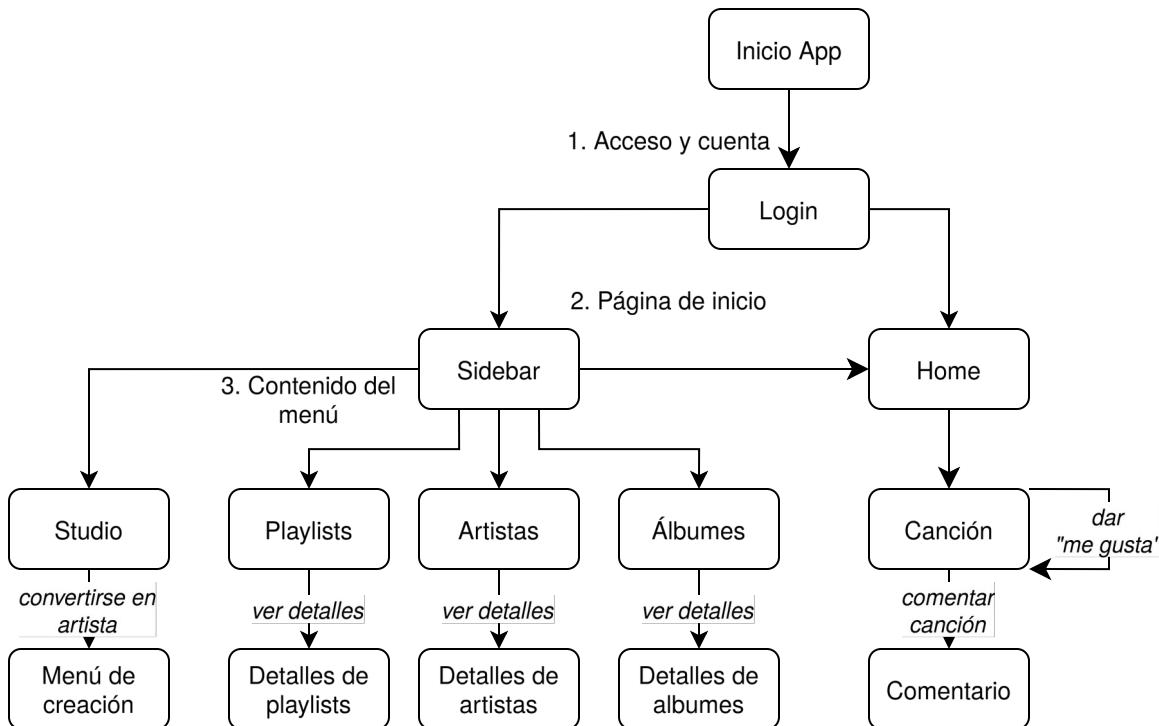
Para esta primera fase se prioriza la visualización de los detalles que almacenamos sobre las canciones, además de mostrar como el usuario interactua con las tablas.



El menú siempre está disponible

6.2 Diagrama de Navegación de la Fase 2

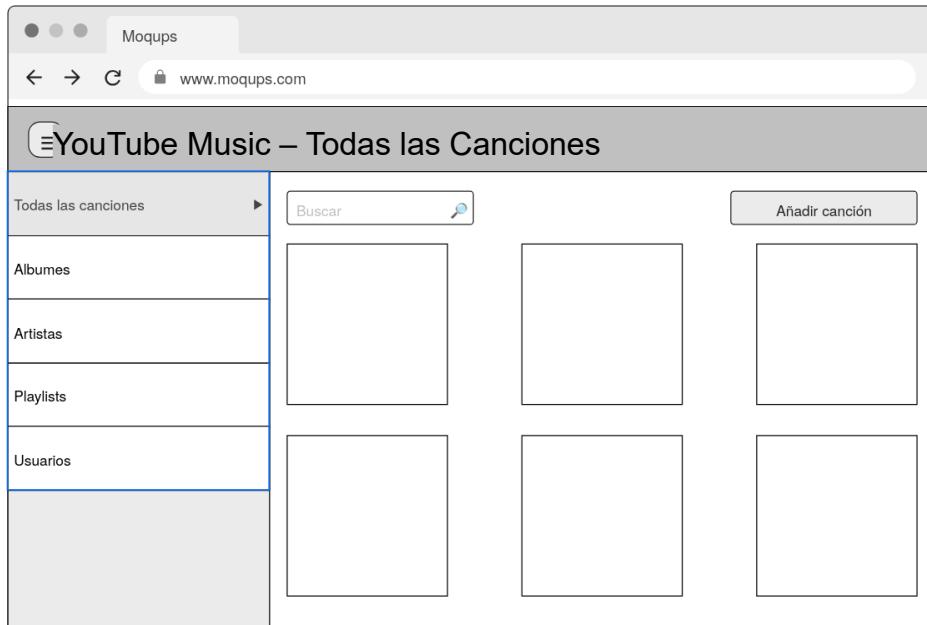
En el caso de la segunda fase se tiene en cuenta como el usuario realiza un comentario sobre una canción y como funciona la interacción del “me gusta”, además que se describe mejor como se realiza la negación dentro del sistema propuesto.



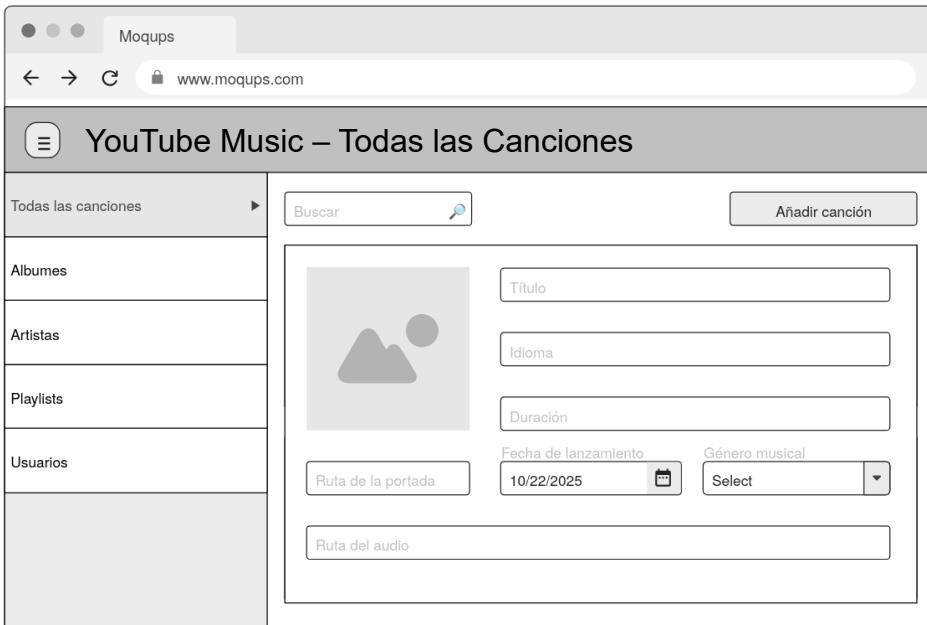
La sidebar siempre está disponible

7 Diseño de las Vistas

Vista para navegar entre canciones y plantilla para navegar.



Vista para añadir canción y plantilla para añadir.



8 Esquema y Scripts SQL

8.1 Esquemas y Scripts de la Base de Datos de la Fase 1

8.1.1 Script del Esquema

Este esquema define la estructura de una base de datos para una aplicación de música, similar a YouTube Music, gestionando usuarios, artistas, canciones y listas de reproducción.

```
PRAGMA foreign_keys = ON;
```

Esta instrucción inicial es fundamental para la integridad de los datos. Su propósito es activar la validación de las llaves foráneas ("FOREIGN KEY") en SQLite, asegurando que las relaciones entre las tablas (como un álbum y sus canciones) se mantengan consistentes.

Tabla "users"

```
CREATE TABLE users (
    id_user      INTEGER PRIMARY KEY,
    name         TEXT,
    password     TEXT,
    creation_date TIMESTAMP,
    email        TEXT
);
```

Esta tabla almacena la información esencial de los usuarios que interactúan con el sistema. Guarda un identificador único ("id_user"), su nombre, una contraseña ("password") para la autenticación, el correo electrónico y la fecha en que se creó la cuenta.

Tabla "artists"

```
CREATE TABLE artists (
    id_artist INTEGER PRIMARY KEY,
    name TEXT UNIQUE, -- El nombre artístico debería ser único
    nationality TEXT,
    biography TEXT,
    debut_date TIMESTAMP,
```

```

id_user INTEGER,
FOREIGN KEY (id_user) REFERENCES users(id_user)
);

```

El propósito de esta tabla es registrar los datos biográficos de los artistas. Incluye un identificador ("id_artist"), su nombre, nacionalidad, una biografía detallada y la fecha de su debut.

Tabla “albums”

```

CREATE TABLE albums (
    id_album    INTEGER PRIMARY KEY,
    name        TEXT,
    release_date  TIMESTAMP,
    cover_path   TEXT16
);

```

Aquí se guarda la información de cada álbum. Contiene su identificador ("id_album"), el nombre del álbum, la fecha en que fue lanzado ("release_date") y la ruta ("cover_path") donde se almacena la imagen de su portada.

Tabla “songs”

```

CREATE TABLE songs (
    id_song     INTEGER PRIMARY KEY,
    title       TEXT UNIQUE,
    language    TEXT,
    release_date  TIMESTAMP,
    duration    REAL,
    genre       TEXT,
    song_path   TEXT UNIQUE,
    cover_path   TEXT,
    id_album    INTEGER,
    FOREIGN KEY (id_album) REFERENCES album(id_album)
);

```

Esta es una tabla central que describe cada canción individualmente. Almacena su título (que debe ser único), idioma, duración, género y las rutas a sus archivos de audio ("song_path") y portada ("cover_path"). Incluye "id_album", que funciona como una

llove foránea ("FOREIGN KEY") para vincular la canción con el álbum al que pertenece (una relación 1:N).

Tabla “social_networks”

```
CREATE TABLE social_networks (
    id_network    INTEGER PRIMARY KEY,
    name          TEXT,
    url           TEXT,
    id_artist     INTEGER,
    FOREIGN KEY (id_artist) REFERENCES artist(id_artist)
);
```

Esta tabla se usa para gestionar los enlaces a las redes sociales de los artistas. Cada registro tiene el nombre de la red (ej. 'Twitter', 'Spotify'), la URL y el "id_artist" al que pertenece, permitiendo que un artista tenga múltiples redes sociales asociadas.

Tabla “playlists”

```
CREATE TABLE playlists (
    id_playlist   INTEGER PRIMARY KEY,
    name          TEXT,
    description   TEXT,
    modification_date  TIMESTAMP,
    creation_date   TIMESTAMP,
    id_user        INTEGER,
    FOREIGN KEY (id_user) REFERENCES user(id_user)
);
```

El objetivo de esta tabla es permitir a los usuarios crear sus propias listas. Guarda el nombre de la lista, una descripción, las fechas de creación y modificación, y el "id_user" que la creó. Esto la vincula directamente con la tabla "users".

Tabla “songs_artists”

```
CREATE TABLE songs_artists (
    id_artist    INTEGER,
    id_song      INTEGER,
    PRIMARY KEY (id_artist, id_song),
    FOREIGN KEY (id_artist) REFERENCES artist(id_artist),
    FOREIGN KEY (id_song) REFERENCES song(id_song)
);
```

Esta es una tabla de unión (junction table) que resuelve la relación "muchos a muchos" (N:M) entre canciones y artistas. Como una canción puede tener varios artistas (colaboraciones) y un artista puede tener muchas canciones, esta tabla almacena pares de "id_artist" y "id_song" para conectar ambos.

Tabla “playlists_songs”

```
CREATE TABLE comments (
    id_comment      INTEGER PRIMARY KEY,
    comment_text    TEXT NOT NULL,
    creation_date   TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    modification_date TIMESTAMP,
    id_song         INTEGER NOT NULL,
    id_user         INTEGER,
    FOREIGN KEY (id_song) REFERENCES songs(id_song) ON DELETE CASCADE,
    FOREIGN KEY (id_user) REFERENCES users(id_user) ON DELETE CASCADE
);
```

De forma similar a la tabla anterior, esta resuelve la relación "muchos a muchos" (N:M) entre las listas de reproducción y las canciones. Permite que una lista contenga múltiples canciones y que una misma canción pueda ser incluida en múltiples listas de diferentes usuarios.

8.1.2 Scripts para el Proceso de Configurar Perfil de Artista

RF1. Registrarse como Artista

Para cumplir con este proceso se tiene este requerimiento donde un usuario obtiene el rol de artista.

```
INSERT INTO artists (name, nationality, biography, debut_date, id_user)
VALUES (?, ?, ?, ?, ?)
```

Con ayuda de esta consulta es que se consigue esto, donde se crea una fila con los campos de la entidad “artists” y el identificador del usuario que pasará a tomar este rol (las interrogaciones representan una variable y la llave primera no se inserta explícitamente pues se hace de forma automática).

RF2. Registrar Redes Sociales

```
INSERT INTO social_networks (name, url, id_artist) VALUES (?, ?, ?)
```

Solo un usuario con rol de artista puede registrar redes sociales a su perfil, por eso esta consulta inserta una fila a la base de datos con la clave de la entidad artista que consigue un usuario tras completar su registro.

RF3. Visualizar Perfil de Artista

```
SELECT *
  FROM artists
 WHERE id_artist = ?
SELECT * FROM social_networks WHERE id_artist = ?
```

Con estas dos consultas se puede visualizar el perfil del artista junto a las redes sociales que haya registrado en su perfil.

8.1.3 Scripts para el Proceso de Administrar Álbumes

RF1. Registrar Nuevo Álbum

```
INSERT INTO albums (name, release_date, cover_path) VALUES (?, ?, ?)
```

Con esta consulta permite el registro de un nuevo álbum por un artista.

RF2. Editar Detalles del Álbum

```
UPDATE albums SET name = ?, cover_path = ? WHERE id_album = ?
```

Para poder actualizar un álbum se hace uso de esta consulta.

RF3. Eliminar Álbum

```
DELETE FROM albums WHERE id_album = ?
```

Para que el artista pueda eliminar un álbum se hace uso de esta consulta con el identificador del álbum.

RF4. Visualizar Álbumes del Artista

```
SELECT a.*, ar.name as artist_name
  FROM albums a
  JOIN artists ar ON a.id_artist = ar.id_artist
 WHERE a.id_artist = ?
```

Finalmente en base a esta consulta se obtienen todos los álbumes por artista, esta consulta sirve para la administración de álbumes, de modo que solo se muestre el contenido propio de un artista, con el fin de que se evite la edición y borrado de álbumes de otro artistas registrados en el sistema.

8.1.4 Scripts para el Proceso de Administrar Canciones

RF1. Cargar Nueva Canción

RF2. Registrar Detalles de Canción

RF3 Asignar Canción a Álbum

```
INSERT INTO songs (title, language, release_date, duration, genre, song_path,
cover_path, id_album) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
```

Esta consulta sirve la inserción en la tabla “songs” y para cargar una canción, al tener una base de datos relacional, el archivo de audio se “inserta” con su ruta en el sistema. Para poder asignar una canción a un álbum, se usa la misma consulta, en la interfáz el artista podrá definir si una canción pertenece a uno de sus álbumes o no.

Por otro lado, para asegurar que una canción se relacione automáticamente al artista en el sistema, se hace con ayuda de una consulta auxiliar

```
SELECT id_artist FROM artists WHERE id_user =
```

que nos da el identificador del artista si el usuario con una sesión iniciada en el sistema tiene ese rol.

```
INSERT INTO songs_artists (id_artist, id_song) VALUES (?, ?)
```

Con ayuda de esta consulta se realiza la inserción automática (con ayuda de backend) de la canción al perfil del artista.

RF4. Administrar Colaboradores

```
INSERT INTO songs_artists (id_artist, id_song) VALUES (?, ?)
```

Igualmente, con ayuda de esta consulta se realiza la asignación de un colaborador, aunque con ayuda de un menú desplegable en la interfaz.

```
DELETE FROM songs_artists WHERE id_artist = ? AND id_song = ?
```

Finalmente para quitar colaboradores se utiliza esta expresión.

RF5. Editar Detalles de Canción

```
UPDATE songs
SET title = ?, language = ?, duration = ?, genre = ?, song_path = ?, cover_path = ?
, id_album = ?
WHERE id_song = ?
```

Para lograr la edición se utiliza la operación UPDATE y con ayuda del backend se lográ la cambiar un archivo de audio o imagen, en la base de datos solo se actualizan sus rutas.

RF6. Eliminar Canción

```
DELETE FROM songs WHERE id_song = ?
```

Con esta expresión se logra el borrado de la canción y para asegurar su borrado en la tabla M:N “songs_artists”, es que se activo la opción ON DELETE CASCADE en el identificador de la canción y el artista (por si se borra al artista).

8.1.5 Scripts para el Proceso de Catálogar y Búscar

RF1. Visualización de Detalles de Canción

```
SELECT s.*, a.name AS album_name,
GROUP_CONCAT(ar.id_artist) AS artist_ids,
GROUP_CONCAT(ar.name) AS artist_names
FROM songs s
LEFT JOIN albums a ON s.id_album = a.id_album
LEFT JOIN songs_artists sa ON s.id_song = sa.id_song
LEFT JOIN artists ar ON sa.id_artist = ar.id_artist
WHERE s.id_song = ?
```

GROUP BY s.id_song

Esto lo logramos con ayuda de esta consulta, donde en base al identificador de la canción es que se pueden obtener todos sus detalles. Además que se muestra el nombre y identificador del álbum y del artista.

```
SELECT s.*,
GROUP_CONCAT(ar.id_artist) AS artist_ids,
GROUP_CONCAT(ar.name) AS artist_names
FROM songs s
LEFT JOIN songs_artists sa ON s.id_song = sa.id_song
LEFT JOIN artists ar ON sa.id_artist = ar.id_artist
GROUP BY s.id_song
ORDER BY id_song DESC
```

Para visualizar las todas canciones se utiliza esta consulta y en la interfaz de usuario se formatea para una mejor presentación. Con la cláusula ORDER BY... DESC se visualizan las canciones desde la última que se añadió hasta la más antigua. Esta consulta se utilizaría en el menú principal, por eso se muestran todos los artistas a los que pertenece una canción.

RF2. Visualización de Contenido por Artista

```
SELECT s.*
FROM songs s
JOIN songs_artists sa ON s.id_song = sa.id_song
WHERE sa.id_artist = ?
```

Para poder vizualizar las canciones del artista se utiliza la tabla “songs_artists” y el identificador del artista. Esta consulta se usaría dentro del perfil del artista, por eso aquí no se busca mostrar el nombre del artista,

```
SELECT a.*, ar.name as artist_name
FROM albums a
JOIN artists ar ON a.id_artist = ar.id_artist
WHERE a.id_artist = ?
```

Con esta consulta se visualizan los álbumes por artista, además de mostrar el nombre del artista al que pertenece el álbum, pues esta consulta se usaría en la pestaña de álbumes.

RF3. Visualización de Contenido por Álbum

```
SELECT s.* FROM songs WHERE s.id_album = ?
```

RF4. Búsqueda Básica por detalles

```
SELECT s.*, a.name AS album_name,
GROUP_CONCAT(DISTINCT ar.id_artist) AS artist_ids,
GROUP_CONCAT(DISTINCT ar.name) AS artist_names
FROM songs s
LEFT JOIN albums a ON s.id_album = a.id_album
LEFT JOIN songs_artists sa ON s.id_song = sa.id_song
LEFT JOIN artists ar ON sa.id_artist = ar.id_artist
GROUP BY s.id_song
HAVING s.title LIKE ?
OR s.language LIKE ?
OR s.genre LIKE ?
OR a.name LIKE ?
OR GROUP_CONCAT(DISTINCT ar.name) LIKE ?
ORDER BY s.title ASC
```

Para poder realizar una búsqueda de la canción en base a sus detalles se usa esta consulta donde es posible buscar por su título, el lenguaje, género musical y el nombre del álbum.

8.1.6 Scripts para el Proceso de Asignar Canciones a Playlists

RF1. Creación de Playlist

```
INSERT INTO playlists (name, description, creation_date, modification_date,
id_user) VALUES (?, ?, strftime('%Y-%m-%d %H:%M:%S', 'now'), strftime('%Y-%m-%d
%H:%M:%S', 'now'), ?)
```

Para empezar con este proceso, un requerimiento que tendría el sistema es la creación de una playlists, requerimiento que se cumple en base a esta consulta donde la fecha de creación y modificación se registra de forma automática.

```
UPDATE playlists SET name = ?, description = ?, modification_date = strftime('%Y-%m-%d %H:%M:%S', 'now') WHERE id_playlist = ?
```

Por ejemplo, para su edición los datos que se actualizan son el nombre y descripción, donde la fecha de modificación se actualiza de forma automática

RF2. Añadir Canción a Playlist

```
INSERT INTO playlists_songs (id_playlist, id_song) VALUES (?, ?)
```

Como existe una relación de muchos a muchos entre “songs” y “playlists”, al momento de añadir una canción a una playlist lo que se hace es añadir una fila a la tabla “playlists_songs” con ambos identificadores de las entidades.

RF3. Visualización de Playlists

```
SELECT * FROM playlists WHERE id_playlist = ?
```

Para ver los detalles de una playlist en específico se utiliza esta consulta y el identificador.

RF4. Visualización de Canciones en Playlist

```
SELECT s.*  
FROM songs s  
JOIN playlists_songs ps ON s.id_song = ps.id_song  
WHERE ps.id_playlist = ?
```

Para ver todas las canciones de una playlist en específico, se utiliza esta consulta donde se usa el identificador de una playlist y la tabla N:M “playlists_songs”.

8.2 Esquema y Scripts de la Base de Datos para Fase II

8.2.1 Script del Esquema

Tabla “Comments”

```
CREATE TABLE comments (  
    id_comment      INTEGER PRIMARY KEY,  
    comment_text    TEXT NOT NULL,
```

```

creation_date  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
modification_date TIMESTAMP,
id_song        INTEGER,
id_user        INTEGER,
    FOREIGN KEY (id_song) REFERENCES songs(id_song) ON DELETE
CASCADE,
    FOREIGN KEY (id_user) REFERENCES users(id_user) ON DELETE SET
NULL
);

```

Esta tabla está almacenando los comentarios que los usuarios realizan sobre canciones específicas. Incluye un identificador único para cada comentario (id_comment), el texto del comentario (comment_text), la fecha de creación (creation_date) y la última fecha de modificación (modification_date). Además, contiene las claves foráneas id_song para vincular el comentario a la canción correspondiente.

Tabla “user_song_ratings”

```

CREATE TABLE user_song_ratings (
    id_user      INTEGER,
    id_song      INTEGER,
    is_liked     BOOLEAN NOT NULL DEFAULT TRUE,
    interaction_date  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (id_user, id_song),
    FOREIGN KEY (id_user) REFERENCES users(id_user) ON DELETE
CASCADE,
    FOREIGN KEY (id_song) REFERENCES songs(id_song) ON DELETE
CASCADE);

```

Esta tabla se utiliza para registrar las interacciones de "Me gusta" de los usuarios con las canciones, resolviendo una relación de muchos a muchos (N:M) entre usuarios y canciones. Contiene las claves foráneas id_user y id_song, que juntas forman la clave primaria compuesta. También incluye un campo booleano is_liked para indicar si la canción ha sido marcada como "Me gusta" y interaction_date para registrar cuándo ocurrió esta interacción.

8.2.2 Scripts para el Procesos de Administrar Comentarios de Canciones

RF1. Añadir Comentario

```
INSERT INTO comments (comment_text, id_song, id_user) VALUES (?, ?, ?)
```

Con esta consulta se puede añadir un comentario a una canción en concreto. Con ayuda de las claves de la canción y el usuario.

RF2. Ver Comentarios

```
SELECT c.*, u.name as username
  FROM comments c
  JOIN users u ON c.id_user = u.id_user
 WHERE c.id_song = ?
 ORDER BY c.creation_date DESC
```

Para ver todos los comentarios por canción se utiliza esta consulta, esta visualización se hace con ayuda de la clave principal de la canción. Además que se muestra el nombre del usuario que realizó el comentario.

RF3. Actualizar Comentario

```
UPDATE comments
   SET comment_text = ?, modification_date = CURRENT_TIMESTAMP
 WHERE id_comment = ?
```

Para actualizar un comentario se hace con ayuda del identificador del comentario y automáticamente se cambia la fecha de modificación. También se recibe el texto del comentario que se editó.

```
DELETE FROM comments WHERE id_comment = ?
```

Y con esta consulta es que se puede borrar un comentario en base a su clave.

8.2.3 Scripts para el Proceso de Curar Biblioteca Personal mediante "Me Gusta"

RF1. Marcar "Me Gusta" en una Canción

```
INSERT INTO user_song_ratings (id_user, id_song) VALUES (?, ?)
```

Para cumplir con este proceso, primero se debe marcar un “me gusta” a una canción, lo cual se logrará mediante esta consulta, donde se realizar una inserción a la tabla “user_song_ratings” con los identificadores del usuario y la canción, y tras la inserción se pone por defecto el valor “is_liked” como verdadero.

RF2. Visualizar Playlist Automática "Canciones que me gustan"

```
SELECT s.* FROM songs s
JOIN user_song_ratings usr ON s.id_song = usr.id_song
WHERE usr.id_user = ? AND usr.is_liked = TRUE
```

En base a esta consulta es que es posible visualizar todas las canciones que un usuario ha marcado como “me gusta”, esto se logra con un JOIN entre la tabla “songs” y “user_song_ratings”, además que se verifica si el campo “is_liked” es verdadero, por si en algún momento se decide usar ese campo con otro propósito.

RF3. Conteo Público de "Me Gusta"

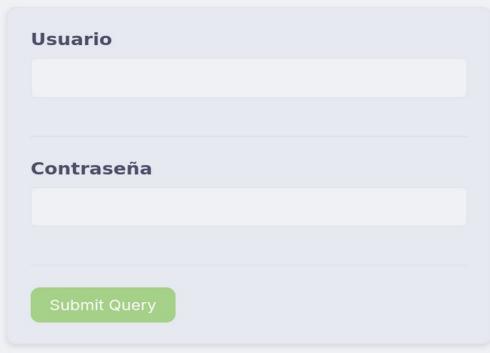
```
SELECT COUNT(*) as quantity_of_likes FROM user_song_ratings WHERE
id_song = ? AND is_liked = TRUE
```

Para ver la cantidad de “me gusta” que tiene una canción es que se usa la cláusula COUNT de todo lo que tenga la clave de la canción y donde también “is_liked” esté marcado como verdadero.

9 Integración de Plantilla y Funcionamiento del Sistema

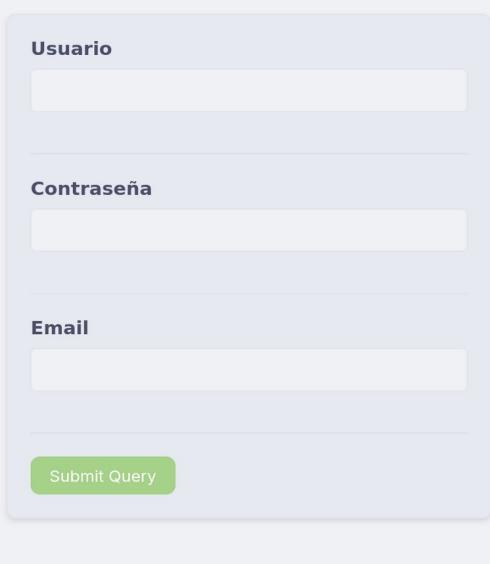
9.1 Menú de Inicio de Sesión y/o Registro

La primera página visible en el sistema es la del inicio de sesión, donde se deben ingresar el nombre de usuario y la contraseña para empezar a navegar por el resto del sistema.



The image shows a login form with a light gray background. It features two input fields: 'Usuario' (User) and 'Contraseña' (Password), each with a placeholder text area below it. Below these fields is a green rectangular button labeled 'Submit Query'. At the bottom left of the form is a small green button labeled 'Registrarse' (Register).

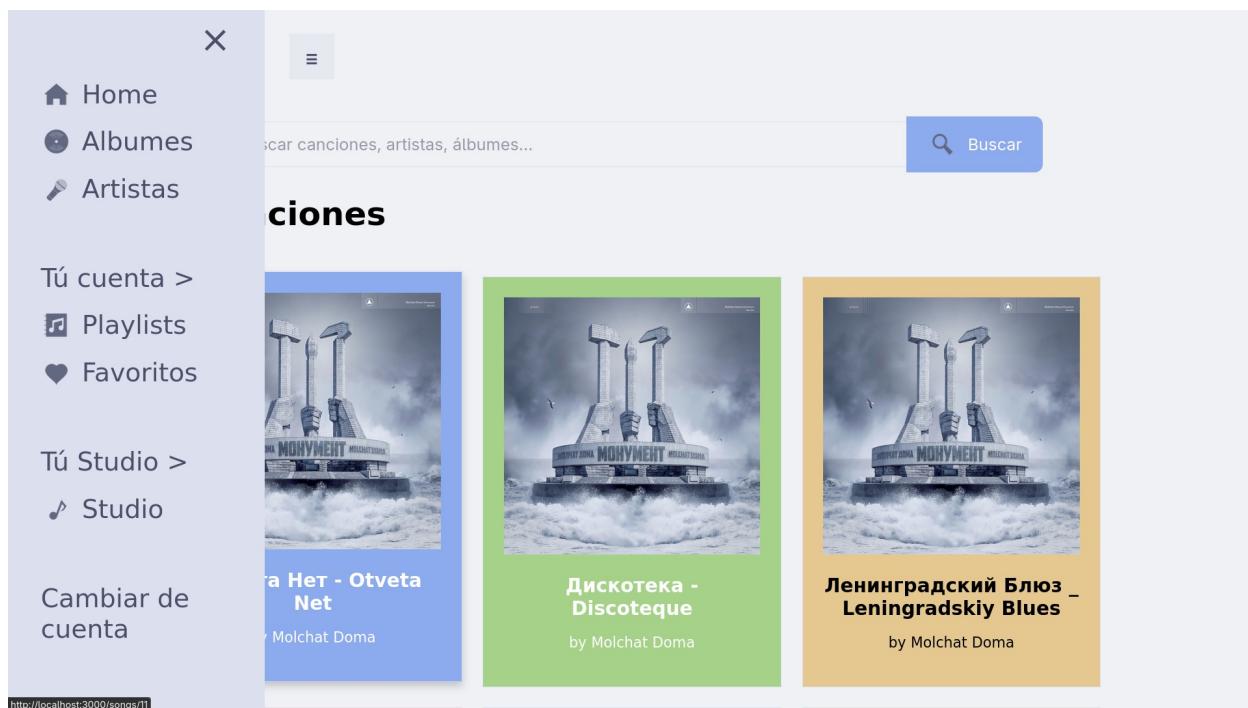
Si todavía, el usuario no creó su cuenta, puede crearla ingresando al formulario de registro con el botón “Registrarse” donde se pide que ingrese un nombre de usuario, una contraseña y su correo electrónico.



The image shows a registration form with a light gray background. It includes three input fields: 'Usuario' (User), 'Contraseña' (Password), and 'Email', each with a placeholder text area below it. Below these fields is a green rectangular button labeled 'Submit Query'. At the bottom left of the form is a small green button labeled 'Iniciar sesión' (Start session).

9.2 Página Principal (Home, Todas la Canciones)

Tras ingresar sus credenciales, el usuario podrá visualizar todas las canciones que se hayan registrado en el sistema. En esta vista, se puede realizar la **búsqueda de una canción** por sus detalles y si se encuentran canciones con el parámetro de búsqueda, estas se mostrarán en el menú. Además que al ingresar al sistema se mostrará una barra lateral desde donde se puede navegar por el resto del sistema.



9.2.1 Página para Ver Detalles de una Canción

El usuario puede ingresar a cada canción (haciendo click sobre su tarjeta o recuadro) y **visualizar sus detalles**, además de poder escucharla, **dar “me gusta”**, **realizar un comentario, borrarlo o editarlo** y **añadir una canción a una playlist**. En esta vista también es posible navegar por el resto de canciones del sistema.

きゅうくらりん/Kyu-kurarin



Idioma: Japonés
Fecha de Lanzamiento: 2025-11-12 04:34:44
Género: Jpop
Álbum: N/A
Artista: いよわ/Iyowa

Likes: 2

Selecciona una playlist

Añadir a playlist

Comentarios

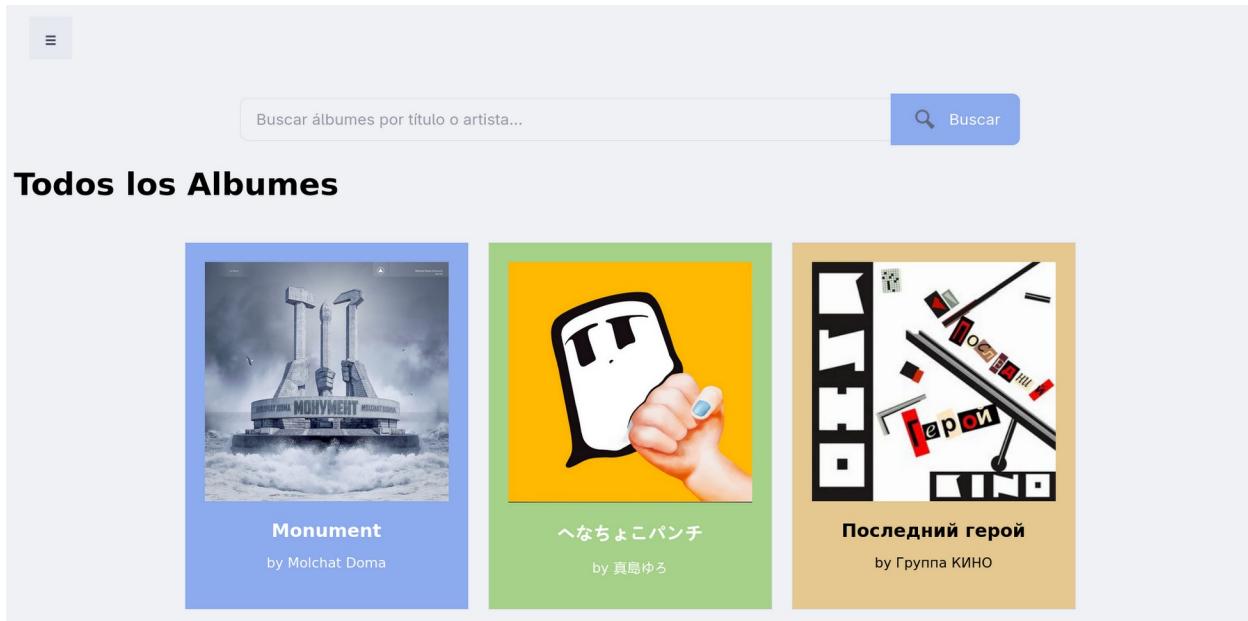
Añade un comentario... Comentar

Todas las Canciones

-  **きゅうくらりん/Kyu-kurarin**
いよわ/Iyowa
-  **『±0』**
真島ゆろ
-  **チキンブイブ**
真島ゆろ
-  **Как на войне**
Агата Кристи
-  **Atac**
Liubé
-  **Последний герой**
Группа КИНО
-  **Группа Крови**
Группа КИНО

9.3 Página de Todos los Álbumes

Dentro de esta vista, el usuario puede navegar entre todos los álbumes registrados, realizar una búsqueda por sus detalles e ingresar a uno de ellos, para ver sus detalles y contenido.



9.3.1 Página para Ver Detalles Contenido de un Álbum

Al ingresar a un álbum, se podrá **visualizar todas la canciones contenidas en él**, además de reproducir todo el contenido de un álbum, además que al ingresar a una canción, se llega a la página de sus detalles.



9.4 Página de Todos los Artistas

Esta página tiene una lógica similar a la previa, pues aquí se puede navegar por todos los artistas registrados, además de realizar una búsqueda en base a sus características registradas. El usuario debe poder acceder al contenido de cada artista

The screenshot shows a search interface with a search bar labeled "Buscar artistas por nombre..." and a "Buscar" button. Below the search bar, the title "Todos los Artistas" is displayed. The page features a grid of six artist profiles, each with a colored background and text in Spanish and English.

- Molchat Doma** (Blue box): Molchat Doma (translated as "Houses Are Silent"), founded in 2017 in Minsk, Belarus, now residing in LA, stands at the intersection of post-punk, new-wave and synth-pop.
- いよわ/Iyowa** (Green box): 曲絵マンです つくれたものを投稿します
- 真島ゆろ** (Orange box): 生きていてごめんなさい。もうちょっとと生きます。
- Агата Кристи** (Red box): Oficialnyy kanal gruppy «Агата Кристи». № 4977020034
- Liubé** (Blue box): Liubé es un grupo musical ruso formado en 1989 en la localidad de Liúbertsi, Moscú, ciudad que da nombre a la banda, también interpretan canciones militares y chansons rusas.
- Группа КИНО** (Green box): Oficialnyy kanal gruppy Кино

9.4.1 Página para Ver Detalles Contenido de un Artista

Dentro de esta página se **muestra el contenido del artista** como serían sus canciones, álbumes y redes sociales. La página principal que se muestra al ingresar, es la de todas las canciones del artista, desde donde se puede acceder a los detalles de cada una (igual a lo que sucede en la página principal).

The screenshot shows the artist profile for Molchat Doma. At the top, the artist's name is displayed in bold. Below it are three navigation links: "Canciones", "Álbumes", and "Redes Sociales". Further down are two buttons: "Reproducir Todas" and "Volver a artistas". The main section is titled "Canciones" and displays three album covers with their respective titles in Russian and English:

- Ленинградский Блюз – Leningradskiy Blues**
- Дискотека - Discoteca**
- Ответа Нет - Otvetka Net**

En otra página el usuario puede visualizar sus **álbumes registrados**, se maneja una lógica similar a la página de todos los álbumes, pues desde esta página igual se puede ingresar al contenido de un álbum.

Группа КИНО

Canciones Álbumes Redes Sociales

Álbumes

[Volver a artistas](#)

Finalmente, existe una sección del menú donde se muestran las **redes sociales registradas** del artista, además de su información registrada, como su biografía, nacionalidad o fecha de registro.

kino@yandex.ru)."/>

Группа КИНО

Canciones Álbumes Redes Sociales

Descripción

Официальный канал группы Кино

Vínculos

Instagram

instagram.com/kino.band.official

Más información

Nacionalidad: Rusia

Debut: 9/11/2025

Contacto: kino@yandex.ru

[Volver a artistas](#)

9.5 Página de Todas las Playlists

En esta página un usuario podrá **ver todas sus playlists** y también podrá **crear nuevas playlists**, además de editarlas o borrarlas.

Todas las Playlists

[Crear playlist](#)

Música japonesa

Playlist de música japonesa

[Editar](#) [Borrar](#)

Música rusa

Playlists de varios artistas rusos

[Editar](#) [Borrar](#)

9.5.1 Página para Ver Contenido de una Playlist

Similar a las vistas de artistas y álbumes, en esta página se **muestran todas las canciones de una playlist**, además que se permite editar y/o eliminar la playlist.

Música rusa

Descripción: Playlists de varios artistas rusos

Fecha de creación: 11/9/2025

Última modificación: 11/9/2025

[Reproducir Todas](#) [Volver a las playlists](#)

Canciones de la Playlist

Группа Крови

[Quitar de la playlist](#)

Последний герой

[Quitar de la playlist](#)

Атас

[Quitar de la playlist](#)

Ответа Нет - Ответа Net

[Quitar de la playlist](#)

También en esta página es donde se puede el usuario puede escuchar todas las canciones que tiene en su playlist además de navegar entre ellas.

9.6 Página de Canciones Favoritas

Aquí el usuario puede **ver una lista de reproducción de las canciones que le dio “me gusta”**, aquí se maneja la misma lógica para la búsqueda y al tocar la carta de una canción, esta te redirige a su página de detalles.

9.7 Página del Formulario para ser Artista

Si el usuario desea publicar contenido como canciones o álbumes, este debe **tomar el rol de artista** mediante el llenado de un formulario, tras este registro, el usuario puede empezar crear contenido

The screenshot shows a registration form for artists. At the top, it says "¡Conviértase en artista!". Below that, a sub-instruction reads "Para empezar a mandar canciones y álbumes a la plataforma ingrese sus datos". The form fields include "Nombre Artístico" (sebas), "Nacionalidad" (dropdown menu showing "Selecciona tu país"), "Biografía" (text area), "Fecha de Aparición" (date input showing "11 / 09 / 2025" with a calendar icon), and a green button labeled "Registrarse como Artista".

Tras llenar el formulario, el usuario con rol de artista puede empezar a crear y administrar su contenido.

9.8 Página del “Studio”

9.8.1 Página para Cargar una Canción

El usuario con rol de artista puede comenzar a **cargar canciones** a su perfil, desde aquí puede cargar el archivo de audio y la portada. Además de indicar si pertenece a uno de sus álbumes

The screenshot shows a form for adding a song. The title is "Añadir una canción". The fields include "Título" (input field), "Lenguaje" (dropdown menu showing "Albanés"), "Género Musical" (input field), "Archivo de Canción" (button labeled "Browse..." with "No file selected."), "Portada" (button labeled "Browse..." with "No file selected."), "Álbum (opcional)" (dropdown menu showing "Ninguno"), and a green "Enviar" button. At the bottom left is a blue "Volver al studio..." button.

9.8.2 Página para Administrar Canciones

Dentro del “studio”, el usuario tambien puede modificar sus los detalles de su canción y/o borrarla, esto se hace con ayuda de un formulario auxiliar. También es posible **agregar un colaborador (otro artista)** y/o eliminarlo.

真島ゆろ

Biografía: 生きていてごめんなさい。もうちょっと生きます。

Nacionalidad: Japón

Debut: 9/11/2025

Mis Canciones



Para editar una canción que el usuario con rol de artista haya cargado, el formulario auxiliar se presenta de forma que sea vean los datos anteriores y el usuario puede modificar los datos que guste y dejar el resto intacto, modificar todos o directamente no modificar ninguno.

Editar Canción

Título チチンピイブ	Lenguaje Japonés
Género Musical Jpop	
Canción Actual: /music/1762732302370_-flower.mp3	
Subir nueva canción (opcional)	
Portada Actual: 	
Subir nueva portada (opcional)	
<input type="button" value="Browse..."/> No file selected.	
Album (opcional) へなちょこパンチ	
<input type="button" value="Guardar Cambios"/>	

Si el usuario lo necesita, puede **añadir un artista o colaborador a su canción**, esto se hace tocando el botón de “añadir colaborador”, tras esta acción se despliega un formulario donde se pueden agregar otros artistas.

Canción チチンピイブ	Artista a Añadir Selecciona un artista
<input type="button" value="Añadir Colaborador"/>	

[Volver...](#)

Otra actividad de administración que puede realizar un usuario con rol de artista es eliminar un colaborador de una de sus canciones con ayuda de un botón.

Administrar Colaboradores para: チチンピイブ

Colaboradores

- 真島ゆろ
- Liubé
- いよわ/Iyowa

[Volver](#)

9.8.3 Página para Crear un Álbum

El usuario con rol de artista también puede **crear un álbum** con ayuda de un formulario, desde donde puede realizar la carga de una portada. Posteriormente le puede agregar canciones a este álbum.

Añadir álbum

Nombre	Portada
<input type="text"/>	<input type="button" value="Browse..."/> No file selected.

Enviar

[Volver al estudio...](#)

9.8.4 Página para Administrar Álbumes

Dentro de esta página, también se puede editar y/o borrar un álbum. Para editar un álbum se usa un formulario auxiliar y para borrar un álbum, el sistema pregunta al usuario si está seguro en eliminarlo.

真島ゆろ

Biografía: 生きていてごめんなさい。もうちょっと生きます。

Nacionalidad: Japón

Debut: 9/11/2025

Mis Álbumes

へなちょこパンチ

Editar **Eliminar**

[Volver al estudio...](#)

El formulario auxiliar se presenta de esta forma donde los datos anteriores del álbum que se busca editar se presentan, el usuario puede cambiar solo los puntos que desee o dejarlo sin modificar.

Editar Álbum

Nombre	Portada Actual:
へなちょこパンチ	
Subir nueva portada (opcional) <input type="button" value="Browse..."/> No file selected.	
Guardar Cambios	

9.8.5 Página para Agregar una Red Social

Finalmente existe una página desde donde un artista con rol de artista puede **agregar más redes sociales a su perfil**. Estos luego se harán visibles en su perfil, en la sección de todos los artistas.

Agregue una red social a su perfil

Nombre de la red Social	Dirección del perfil
<input type="text"/>	<input type="text"/>
Enviar	

[Volver al studio...](#)

10 Integración se Seguridad

10.1 Seguridad para la Contraseña del Usuario

10.1.1 Función de Registro

Para el registro de un nuevo usuario donde se pide su contraseña, se usa esa función:

```
async function handleSignUp(req: Request): Promise<Response> {
  try {
    const body = await req.formData();
    const user = body.get("user") as string;
    const pass = body.get("password") as string;
    const hashedPassword = await Bun.password.hash(pass);
    const email = body.get("email") as string;

    if (!user || !hashedPassword || !email) {
      return new Response(JSON.stringify({ message: 'Faltan campos obligatorios' }), { status: 400 });
    }

    await userModel.insertUser(user, hashedPassword, email);

    return new Response(null, {
      status: 302, // O 301 para redirección permanente
      headers: {
        Location: "/login",
      },
    });

  } catch (error) {
    console.error("Error al manejar la solicitud de registro:", error);
    return new Response(JSON.stringify({ message: 'Error interno del servidor' }), { status: 500 });
  }
}
```

Donde lo destacable es el uso de la función Bcrypt como algoritmo de hash. La línea await Bun.password.hash(pass), utiliza la API de contraseñas incorporada en Bun, que utiliza Bcrypt por defecto (con un costo o "work factor" predeterminado).

Bcrypt no es un simple algoritmo de hash como MD5 o SHA-256 (que usa shasum). Es una Función de Derivación de Clave (KDF) diseñada específicamente para hashear contraseñas de forma segura.

10.1.2 Características Importantes de Bcrypt

- **Usa "Salting" (Sal) Automáticamente:** Antes de hashear la contraseña, Bcrypt le añade una cadena aleatoria y única llamada "sal" (salt). Esto significa que incluso si dos usuarios tienen la misma contraseña (ej. "123456"), sus hashes almacenados en la base de datos serán completamente diferentes.
- **Esto frustra los "ataques de tabla arcoíris" (rainbow table attacks):** Es Lento (Costoso) por Diseño y a diferencia de MD5 o SHA, que son extremadamente rápidos, Bcrypt es intencionalmente lento, pues utiliza un "factor de costo" (cost factor) que determina cuántas rondas de cómputo debe realizar.
Y esta lentitud es buena para la seguridad, pues si un atacante roba tu base de datos de hashes, le tomará una cantidad de tiempo y recursos computacionales enorme intentar adivinar las contraseñas una por una (ataque de fuerza bruta).

10.2 Seguridad para la Cookie del Identificador del Usuario

10.2.1 Función de Inicio de Sesión

Dentro del sistema un usuario puede navegar por sus listas de reproducción, modificarlas, eliminarlas o añadir más canciones, además un artista puede administrar su contenido (canciones o álbumes), entonces para evitar que otro usuario manipule el contenido de otro, se hace uso de una cookie tras el login.

```
async function handleLogin(req: Request): Promise<Response> {
  try {
    const body = await req.formData();
    const name = body.get("user") as string;
    const pass = body.get("password") as string;

    // (Respuesta de error, está bien)
    if (!name || !pass) {
```

```
    return new Response(JSON.stringify({ message: 'Faltan campos obligatorios' }), {  
status: 400 });  
}  
  
const user = await userModel.getUser(name);  
  
// (Verificación de usuario, está bien)  
if (!user) {  
    return new Response(JSON.stringify({ message: 'Credenciales incorrectas' }), {  
status: 401 });  
}  
  
// (Verificación de contraseña, está bien)  
const hashedPassword = user.password;  
const verify = await Bun.password.verify(pass, hashedPassword);  
  
if (!verify) {  
    return new Response(JSON.stringify({ message: 'Credenciales incorrectas' }), {  
status: 401 });  
}  
  
// Encrypt the user ID for the cookie  
const encryptedUserId = await encrypt(user.id_user.toString());  
const userCookie = `id_user=${encryptedUserId}; Path=/; HttpOnly`;  
  
// Prepara los headers para la respuesta  
const headers = new Headers();  
headers.set('Location', "/"); // A dónde redirigir  
headers.append('Set-Cookie', userCookie); // Añade la PRIMERA cookie  
  
// Envía la redirección (302) CON todas las cookies  
return new Response(null, {  
status: 302,  
headers: headers, // Usa el objeto headers que preparamos  
});  
  
} catch (error) {  
    console.error("Error al manejar la solicitud de inicio de sesión:", error);  
    return new Response(JSON.stringify({ message: 'Error interno del servidor' }), {  
status: 500 });  
}  
}
```

La función handleLogin en lugar de solo poner el indentificador del usuario en la cookie, lo encripta. Pues si no se usara encriptación, un usuario malintencionado (o simplemente curioso) puede en su ver esto en la memoria del navegador "id_user: 1" y cambiar manualmente el valor de mi cookie de "1" a "2", de forma que logre robar la sesión de otro usuario.

10.2.2 Encriptación Autenticada (AES-GCM)

La sección código de encrypt resuelve esto usando AES-GCM. Quien hace lo siguiente:

Cifrado Simétrico (Confidencialidad)

- AES (Advanced Encryption Standard) es el algoritmo que "revuelve" los datos.
- Es simétrico, lo que significa que usa la misma clave secreta tanto para encriptar como para desencriptar.
- Oculta el dato. Por ejemplo El "id_user=123" se convierte en "id_user=a9b4c8..." (un texto hexadecimal). Ahora el usuario ya no puede ver cuál es su ID.

GCM: Sello de Seguridad (Integridad y Autenticidad)

- GCM (Galois/Counter Mode) es un "modo de operación" para AES.
- Lo que hace es añadir autenticación al cifrado. AES-GCM se conoce como un cifrado AEAD (Authenticated Encryption with Associated Data).

El IV (Vector de Inicialización)

- El "IV" es un número aleatorio que se usa solo una vez por cada encriptación.
- Asegura que si se encripta el mismo dato (ej. "1") dos veces, se obtendrán dos resultados cifrados completamente diferentes.
- Esto impide que un atacante pueda ver patrones. Si no se usara un IV, todas las cookies de usuarios con "id_user=1" serían idénticas, lo cual es una fuga de información.

Con esto se encripta el identificador del usuario y se utiliza en las consultas previas como las que ayudan para crear, editar o eliminar comentarios, crear playlists,

dar “me gusta”, etc. Gracias a la función getUserCookie que se utiliza en algunos handlers.

```
export async function getUserCookie(req: Request): Promise<number | null> {
  const encryptedId = await getCookieValue(req, 'id_user');
  if (!encryptedId) {
    return null;
  }

  const decryptedId = await decrypt(encryptedId);
  if (!decryptedId) {
    return null;
  }

  const num = parseInt(decryptedId, 10);
  return isNaN(num) ? null : num;
}
```

Y para verificar que un usuario con sesión iniciada sea artista y por lo tanto, tenga funciones como gestiones sus canciones y álbumes, es que se utiliza esta función con una consulta y getUserCookie. La cuál da el identificador del artista si el usuario tiene ese rol

```
export async function isArtist(req: Request): Promise<{ id_artist: number } | Response> {
  const db = new Database("data/music-server.db");
  const userId = await getUserCookie(req);

  if (!userId) {
    return new Response("Unauthorized", { status: 401 });
  }

  const artist = db.query<{ id_artist: number }, [number]>("SELECT id_artist FROM artists WHERE id_user = ?").get(userId);

  if (!artist) {
    return new Response("Forbidden: You are not an artist", { status: 403 });
  }

  return artist;
```

11 Conclusión

11.1 *Conclusión de la Fase 1*

En conclusión, se pudo llevar a cabo parte del objetivo general de diseñar un sistema de catálogo musical robusto y moderno, y se cumple mediante el diseño de una estructura que permite a los usuarios administrar su música de forma integral, similar a la que requeriría un servicio a gran escala como YouTube Music.

El sistema propuesto abarca el ciclo completo, desde la catalogación de una inmensa biblioteca de canciones, artistas y álbumes, hasta la creación y visualización de listas de reproducción personalizadas por millones de usuarios.

En relación con los objetivos específicos, el sistema desarrollado para el proceso de catalogación permite el almacenamiento de metadatos complejos de las canciones y asegura la visualización clara de todos sus detalles. Además, se establecen las bases para funcionalidades clave en un servicio de streaming: un sistema de búsqueda y filtrado básico, permitiendo a los usuarios encontrar y organizar música por detalles específicos como artista o género.

Además, se cumplen los objetivos del proceso de asignación a playlists. Con este diseño los usuarios pueden crear nuevas listas de reproducción, de forma que se facilita la selección y adición de canciones a las listas, y se permite una visualización clara tanto de las listas creadas como de las canciones que contienen, un pilar de la experiencia de usuario en YouTube Music.

11.2 *Conclusión de la Fase 2*

Para la segunda fase, se han cumplido los objetivos relacionados con la administración de comentarios de canciones, permitiendo a los usuarios escribir, guardar, visualizar y modificar comentarios en canciones específicas, además haciendo que solo los usuarios que crearon esos comentarios los puedan modificar, de forma que se fomente la interacción. Asimismo, se logró la curación de la biblioteca personal mediante la función "Me gusta", donde los usuarios pueden marcar canciones como favoritas, visualizar una playlist automática de estas canciones y ver el conteo público

de "Me gusta" de cada canción, lo que enriquece la experiencia de personalización y descubrimiento musical.

11.3 Conclusión de la Fase 3

Se puede concluir que si se pudo implementar la plantilla y gracias a eso se pudo presentar un sistema que cumple con los requerimientos que se tenían planificados. Gracias al uso de CSS y JavaScript se pudo presentar un sistema sencillo de usar y un menú simple de administración para los artistas.

Con ayuda de una barra lateral integrada en todas las vistas en que se pudo realizar la navegación como se tenía previsto, tratando de dar al usuario la suficiente "comodidad" para navegar por el sistema.

11.4 Conclusión de la Fase 4

En conclusión se implementaron dos medidas de seguridad donde se utiliza Bcrypt (Hashing) para verificar contraseñas. Siendo un proceso de un solo sentido, donde se guarda un hash ilegible que no puedes revertir. Su propósito es solo responder "Sí" o "No" a la pregunta "¿Es esta la contraseña correcta?".

Y se utiliza AES-GCM (Encriptación) para proteger la ID del usuario que inicio sesión en una cookie de sesión. Es un proceso de dos sentidos (cifrar y descifrar) que requiere una clave secreta. Y su propósito es ocultar información y garantizar que no haya sido manipulada.

Es gracias a estas dos medidas de seguridad que se busca garantizar mayor seguridad dentro del sistema, tanto para los usuarios con un rol "casual" como los que tiene el rol de artista.