DESAFÍO ENTREGABLE 2

SQL Flex

Profesor: Camilo Andres Redondo

Tema

- Diseño y modelado de una base de datos para una plataforma de streaming musical

Autor

- Luis Angel Umiña Navia

Fecha de presentación:

- 28/03/2025

Tabla de Contenido

Temática del Proyecto Final:

Diseño y modelado de una base de datos para una plataforma de streaming musical

1. Introducción

El presente proyecto, titulado SoundWaveDB, consiste en el diseño e implementación de una base de datos relacional para una plataforma de streaming musical. Este proyecto fue desarrollado como parte del curso de SQL en Coderhouse, con el objetivo de aplicar los conocimientos adquiridos en el manejo de estructuras de datos, relaciones entre tablas y desarrollo de objetos de base de datos avanzados como vistas, funciones, procedimientos almacenados y triggers.

La elección del modelo de negocio musical responde al crecimiento sostenido que ha tenido la industria del streaming en los últimos años, convirtiéndose en una de las principales formas de consumo de contenido digital a nivel global. SoundWaveDB busca simular cómo una plataforma de este tipo puede gestionar de forma eficiente la información de usuarios, artistas, canciones, álbumes, listas de reproducción, favoritos y eventos.

El desarrollo del proyecto incluye el modelado de la base de datos, acompañado de un conjunto de datos de prueba, consultas SQL analíticas y la creación de objetos que mejoran la integridad, eficiencia y funcionalidad de la base de datos. Todo el código se encuentra documentado y disponible en un repositorio público.

2. Objetivo del proyecto

El objetivo de esta base de datos es estructurar y optimizar la gestión de datos en una plataforma de streaming musical. Permite:

- Registrar y gestionar usuarios y sus preferencias.
- Almacenar información sobre artistas y sus géneros musicales.
- Administrar álbumes y canciones, vinculándolos con artistas.
- Gestionar listas de reproducción y su contenido.
- Registrar eventos musicales y colaboraciones entre artistas.
- Facilitar el registro de suscripciones y comentarios de los usuarios.

3. Situación Problemática

En la actualidad, las plataformas de streaming musical manejan una gran cantidad de información sobre usuarios, artistas y contenido. Contar con una base de datos bien estructurada permite organizar y gestionar esta información de manera eficiente, facilitando la experiencia de los usuarios y optimizando la administración de los datos.

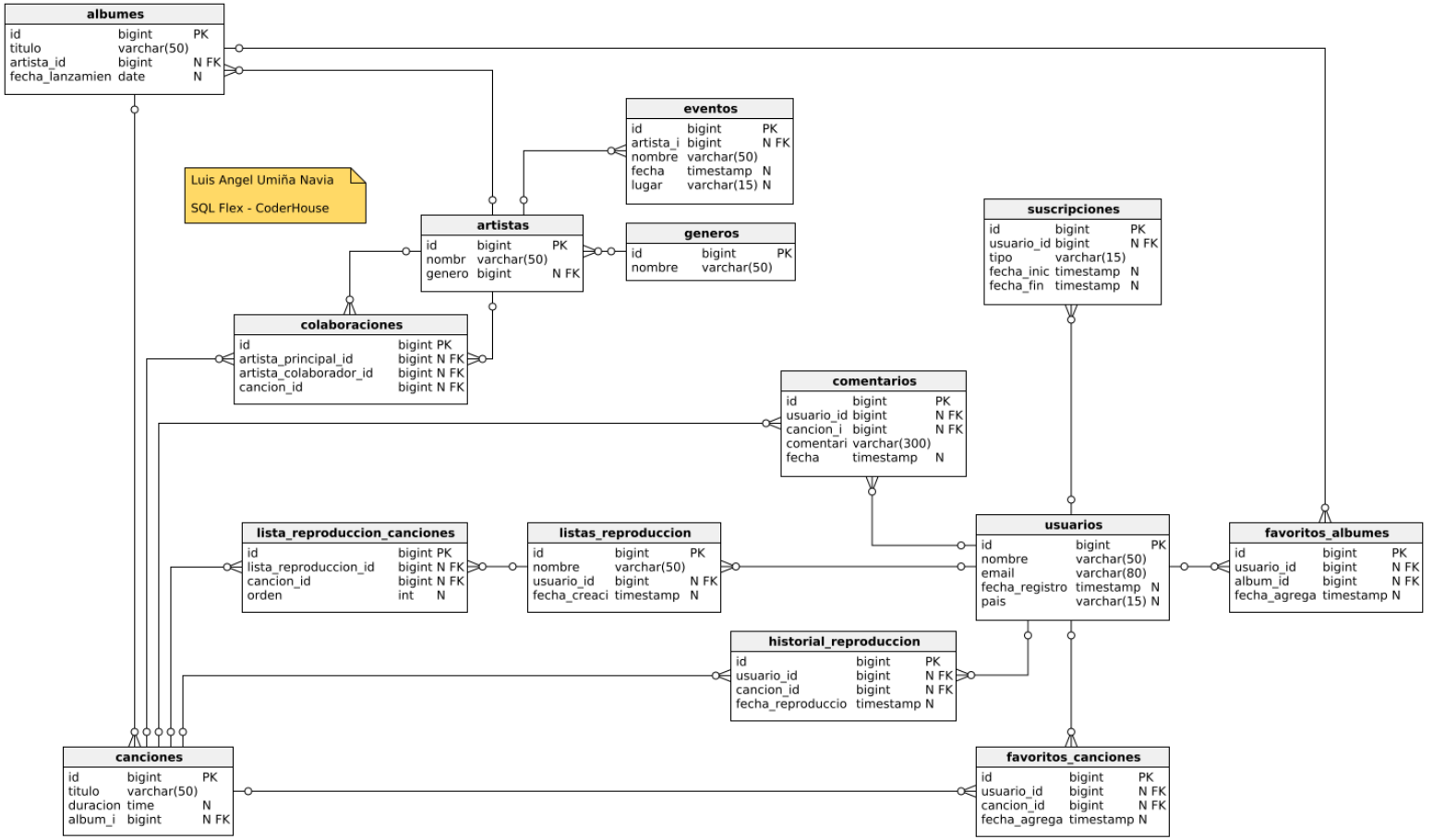
4. Modelo de Negocio

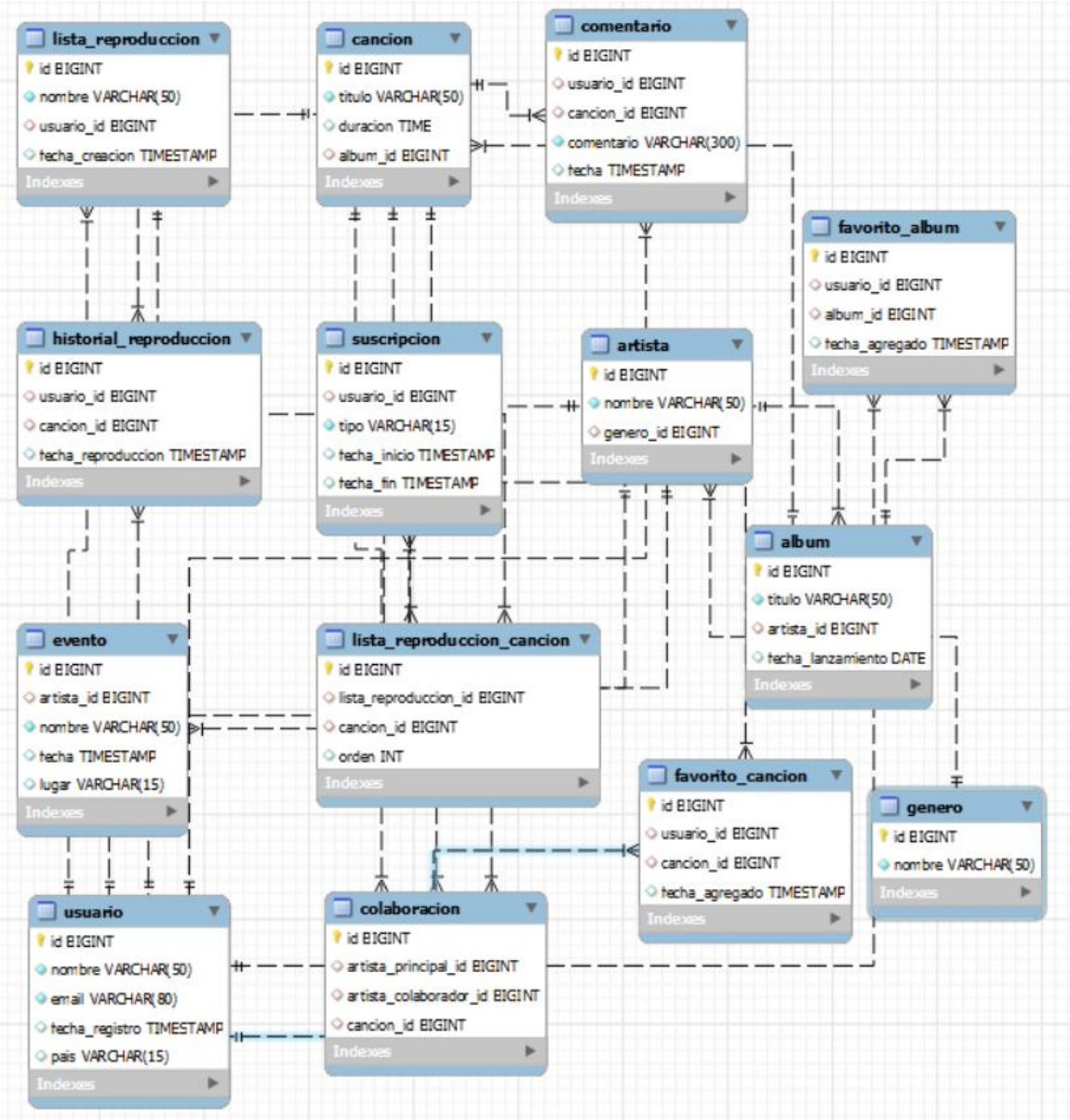
SoundWave es una plataforma de streaming musical que conecta a oyentes y artistas mediante una experiencia digital personalizada. Su modelo de negocio se basa principalmente en suscripciones freemium, publicidad dirigida y promoción de eventos musicales.

- **Propuesta de Valor:**
  - **Para usuarios:** acceso a millones de canciones, recomendaciones personalizadas, listas de reproducción y experiencias musicales sin interrupciones mediante suscripción Premium.
  - **Para artistas:** visibilidad de su contenido, análisis de su audiencia y promoción de eventos o lanzamientos.
- **Fuentes de Ingreso:**
  - **Suscripciones Premium:** acceso ilimitado y sin publicidad.
  - **Usuarios Free**: acceden gratuitamente, pero con anuncios intercalados.
  - **Publicidad personalizada**: según hábitos de escucha y país.
  - **Eventos musicales patrocinados:** gestión de conciertos, festivales y lives.
  - **Datos analíticos:** la plataforma puede vender estadísticas agregadas a agencias musicales o discográficas (por ejemplo, álbumes más reproducidos o canciones con mejor engagement).
- **Gestión de información clave:** La base de datos SoundWaveDB permite administrar toda la información crítica del negocio:
  - Perfil de usuario, país y tipo de suscripción
  - Hábitos de reproducción y preferencias
  - Álbumes, canciones y artistas
  - Colaboraciones y eventos musicales
  - Favoritos y listas de reproducción

5. Diagrama Entidad-Relación

A continuación, se detalla el diagrama Entidad-Relación desarrollado en Vertabelo, y generado por la herramienta:




lista_reproduccion
id BIGINT
nombre VARCHAR(50)
usuario_id BIGINT
fecha_creacion TIMESTAMP
Indexes
cancion
id BIGINT
titulo VARCHAR(50)
duracion TIME
album_id BIGINT
Indexes
comentario
id BIGINT
usuario_id BIGINT
cancion_id BIGINT
comentario VARCHAR(300)
fecha TIMESTAMP
Indexes
favorito_album
id BIGINT
usuario_id BIGINT
album_id BIGINT
fecha_agregado TIMESTAMP
Indexes
historial_reproduccion
id BIGINT
usuario_id BIGINT
cancion_id BIGINT
fecha_reproduccion TIMESTAMP
Indexes
suscripcion
id BIGINT
usuario_id BIGINT
tipo VARCHAR(15)
fecha_inicio TIMESTAMP
fecha_fin TIMESTAMP
Indexes
artista
id BIGINT
nombre VARCHAR(50)
genero_id BIGINT
Indexes
album
id BIGINT
titulo VARCHAR(50)
artista_id BIGINT
fecha_lanzamiento DATE
Indexes
evento
id BIGINT
artista_id BIGINT
nombre VARCHAR(50)
fecha TIMESTAMP
lugar VARCHAR(15)
Indexes
lista_reproduccion_cancion
id BIGINT
lista_reproduccion_id BIGINT
cancion_id BIGINT
orden INT
Indexes
favorito_cancion
id BIGINT
usuario_id BIGINT
cancion_id BIGINT
fecha_agregado TIMESTAMP
Indexes
genero
id BIGINT
nombre VARCHAR(50)
Indexes
usuario
id BIGINT
nombre VARCHAR(50)
email VARCHAR(80)
fecha_registro TIMESTAMP
pais VARCHAR(15)
Indexes
colaboracion
id BIGINT
artista_principal_id BIGINT
artista_colaborador_id BIGINT
cancion_id BIGINT
Indexes

6. Listado de Tablas

A continuación, se describen las tablas de nuestro modelo, indicando las claves primarias y foráneas junto con una breve descripción de cada columna.

Tabla: usuario

Contiene la información de los usuarios registrados en la plataforma.

Campo	*Tipo de Dato*	*Clave*	*Descripción*
id	BIGINT	PK	Identificador único de usuario
nombre	VARCHAR(50)		Nombre del usuario
email	VARCHAR(80)	UNIQUE	Correo electrónico
fecha_registro	TIMESTAMP		Fecha de registro en la plataforma
pais	VARCHAR(15)		País de residencia

Tabla: artista

Almacena los datos de los artistas que tienen contenido en la plataforma.

Campo	*Tipo de Dato*	*Clave*	*Descripción*
id	BIGINT	PK	Identificador único del artista
nombre	VARCHAR(50)		Nombre del artista
genero_id	BIGINT	FK	Relación con la tabla géneros

Tabla: genero

Guarda los géneros musicales asociados a los artistas y sus canciones.

Campo	*Tipo de Dato*	*Clave*	*Descripción*
id	BIGINT	PK	Identificador único del género
nombre	VARCHAR(50)		Nombre del género musical

Tabla: album

Registra los álbumes publicados por los artistas en la plataforma.

Campo	***Tipo de Dato***	***Clave***	***Descripción***
id	BIGINT	PK	Identificador único del álbum
titulo	VARCHAR(50)		Nombre del álbum
artista_id	BIGINT	FK	Relación con la tabla artistas
fecha_lanzamiento	DATE		Fecha de lanzamiento del álbum

Tabla: cancion

Almacena la información de las canciones disponibles en la plataforma.

Campo	***Tipo de Dato***	***Clave***	***Descripción***
id	BIGINT	PK	Identificador único de la canción
titulo	VARCHAR(50)		Nombre de la canción
duracion	TIME		Duración de la canción
album_id	BIGINT	FK	Relación con la tabla albumes

Tabla: lista_reproduccion

Contiene las listas de reproducción creadas por los usuarios.

Campo	***Tipo de Dato***	***Clave***	***Descripción***
id	BIGINT	PK	Identificador único de la lista
nombre	VARCHAR(50)		Nombre de la lista de reproducción
usuario_id	BIGINT	FK	Relación con la tabla usuarios
fecha_creacion	TIMESTAMP		Fecha de creación de la lista

Tabla: lista_reproduccion_cancion

Relaciona las listas de reproducción con las canciones que contienen.

Campo	*Tipo de Dato*	*Clave*	*Descripción*
id	BIGINT	PK	Identificador único
lista_reproduccion_id	BIGINT	FK	Relación con listas_reproduccion
cancion_id	BIGINT	FK	Relación con canciones
orden	INT		Posición de la canción en la lista

Tabla: historial_reproduccion

Registra las canciones reproducidas por los usuarios.

Campo	*Tipo de Dato*	*Clave*	*Descripción*
id	BIGINT	PK	Identificador único del historial
usuario_id	BIGINT	FK	Relación con la tabla usuarios
cancion_id	BIGINT	FK	Relación con la tabla canciones
fecha_reproduccion	TIMESTAMP		Fecha y hora de reproducción

Tabla: suscripcion

Almacena las suscripciones de los usuarios a distintos planes de la plataforma.

Campo	***Tipo de Dato***	***Clave***	***Descripción***
id	BIGINT	PK	Identificador único de la suscripción
usuario_id	BIGINT	FK	Relación con la tabla usuarios
tipo	VARCHAR(15)		Tipo de suscripción
fecha_inicio	TIMESTAMP		Fecha de inicio de la suscripción
fecha_fin	TIMESTAMP		Fecha de vencimiento de la suscripción

Tabla: comentario

Permite a los usuarios comentar sobre canciones en la plataforma.

Campo	***Tipo de Dato***	***Clave***	***Descripción***
id	BIGINT	PK	Identificador único del comentario
usuario_id	BIGINT	FK	Relación con la tabla usuarios
cancion_id	BIGINT	FK	Relación con la tabla canciones
comentario	VARCHAR(300)		Texto del comentario
fecha	TIMESTAMP		Fecha en que se realizó el comentario

Tabla: colaboracion

Registra las colaboraciones entre artistas en una misma canción.

Campo	***Tipo de Dato***	***Clave***	***Descripción***
id	BIGINT	PK	Identificador único
artista_principal_id	BIGINT	FK	Relación con artistas (artista principal)
artista_colaborador_id	BIGINT	FK	Relación con artistas (colaborador)
cancion_id	BIGINT	FK	Relación con canciones

Tabla: evento

Almacena eventos musicales relacionados con los artistas de la plataforma.

Campo	***Tipo de Dato***	***Clave***	***Descripción***
id	BIGINT	PK	Identificador único del evento
artista_id	BIGINT	FK	Relación con la tabla artistas
nombre	VARCHAR(50)		Nombre del evento
fecha	TIMESTAMP		Fecha y hora del evento
lugar	VARCHAR(15)		Ubicación del evento

Tabla: favorito_cancion

Guarda las canciones favoritas de los usuarios.

Campo	***Tipo de Dato***	***Clave***	***Descripción***
id	BIGINT	PK	Identificador único
usuario_id	BIGINT	FK	Relación con la tabla usuarios
cancion_id	BIGINT	FK	Relación con la tabla canciones
fecha_agregado	TIMESTAMP		Fecha en que se marcó como favorita

Tabla: favorito_album

Registra los álbumes favoritos de los usuarios.

Campo	***Tipo de Dato***	***Clave***	***Descripción***
id	BIGINT	PK	Identificador único
usuario_id	BIGINT	FK	Relación con la tabla usuarios
album_id	BIGINT	FK	Relación con la tabla albumes
fecha_agregado	TIMESTAMP		Fecha en que se marcó como favorito

7. Implementación de VISTAS

Las vistas en una base de datos son consultas predefinidas que permiten presentar datos de manera simplificada o combinada, facilitando la lectura y la reutilización de consultas complejas. En este proyecto se han implementado tres vistas:

- **vw_usuario_comentario:**
    - Esta vista une la información de las tablas usuario, comentario y cancion. Permite ver, de manera consolidada, el nombre del usuario, el comentario que realizó y el título de la canción a la que se refirió.
    - **¿Para qué sirve?:** Facilita la obtención de datos para reportes o análisis sin necesidad de escribir el JOIN completo en cada consulta. Por ejemplo, un administrador puede revisar rápidamente todos los comentarios y a qué canciones corresponden.
- **vw_artista_album:**
    - Une la tabla artista con la de album, mostrando el nombre del artista, el título del álbum y la fecha de lanzamiento.
    - **¿Para qué sirve?:** Permite consultar de forma directa la discografía de cada artista, facilitando el análisis de la producción musical y la organización de la información de lanzamiento.
- **vw_lista_canciones_compleja:**
    - Utiliza la función GROUP_CONCAT para concatenar los títulos de las canciones de cada lista, mostrando en una sola columna todos los títulos ordenados según la posición definida.
    - Incluye un subquery que cuenta la cantidad total de canciones en cada lista.
    - **¿Para qué sirve?:** Ofrece una visión global de cada lista de reproducción, permitiendo saber no solo quién creó la lista y su nombre, sino también qué canciones contiene y cuántas son. Esto es útil para reportes y análisis de las listas de reproducción, haciendo más accesible la información agregada.

8. Implementación de FUNCIONES

Las funciones en SQL son bloques de código que reciben parámetros, realizan cálculos o consultas y devuelven un valor. Se utilizan para encapsular lógica y facilitar la reutilización de código en múltiples partes del sistema

- **fn_contar_canciones_album:**
  - Esta función recibe como parámetro el ID de un álbum y retorna la cantidad de canciones asociadas a dicho álbum.
  - **¿Para qué sirve?:** Permite obtener rápidamente, mediante una llamada a la función, el número de canciones de cualquier álbum, lo que es útil para informes, estadísticas y validaciones.
- **fn_total_favoritos_usuario:**
  - Recibe el ID de un usuario y retorna el total de favoritos, sumando los registros de la tabla de favoritos de canciones y la de álbumes.
  - **¿Para qué sirve?:** Facilita el análisis de la popularidad o la actividad de un usuario en la plataforma, permitiendo conocer en un solo valor cuántos contenidos ha marcado como favorito

9. Implementación de PROCEDIMIENTOS ALMACENADOS (Stored Procedures)

Los procedimientos almacenados son secuencias de instrucciones SQL que se almacenan en la base de datos y se pueden ejecutar de forma repetida. Permiten encapsular procesos de negocio o tareas repetitivas, lo que mejora la mantenibilidad y consistencia de las operaciones.

- **sp_insertar_comentario:**
  - Este procedimiento permite insertar un nuevo comentario en la tabla comentario. Recibe como parámetros el ID del usuario, el ID de la canción y el texto del comentario.
  - **¿Para qué sirve?:** Centraliza la lógica de inserción de comentarios, lo que puede incluir validaciones o acciones adicionales (como la verificación a través de triggers) y facilita la integración con aplicaciones front-end o sistemas externos.
- **sp_listar_albums_artista:**
  - Permite listar todos los álbumes asociados a un artista, recibiendo como parámetro el ID del artista.

- **¿Para qué sirve?:** Ayuda a extraer de forma rápida y directa la discografía de un artista, lo que es especialmente útil para generar reportes, analizar la producción musical y mostrar información en interfaces de usuario.

10. Implementación de TRIGGERS

Los triggers son fragmentos de código que se ejecutan automáticamente en respuesta a ciertos eventos (como inserciones, actualizaciones o eliminaciones) en una tabla. Se utilizan para mantener la integridad de los datos y aplicar reglas de negocio de forma automática.

- **tr_usuario_before_insert:**
  - Este trigger se ejecuta antes de insertar un nuevo registro en la tabla usuario. Verifica si el campo pais está nulo o vacío y, en ese caso, asigna el valor "Desconocido".
  - **¿Para qué sirve?:** Asegura que ningún registro de usuario quede sin información básica en el campo país, manteniendo la consistencia de los datos y evitando valores nulos o inconsistentes.
- **tr_comentario_before_insert:**
  - Se ejecuta antes de insertar un comentario en la tabla comentario. Comprueba que la longitud del comentario sea de al menos 5 caracteres. Si no se cumple esta condición, se genera un error que impide la inserción.
  - **¿Para qué sirve?:** Garantiza la calidad de los datos, evitando comentarios demasiado breves que podrían no aportar información útil, lo que ayuda a mantener la integridad y el valor de los datos registrados.

11. Instrucciones sobre el Script SQL

El script de creación de la base de datos se encuentra en el archivo **SoundWaveDB_CoderHouse_DB.sql**. Este script integra la creación del schema, la definición de todas las tablas (con sus claves primarias y foráneas), la inserción de datos, y la implementación de nuevas funcionalidades: vistas complejas, funciones, procedimientos almacenados y triggers.

Se puede ejecutar en un entorno MySQL utilizando los siguientes pasos:

1. Descargar el archivo del repositorio de GitHub: https://github.com/LuisUmina/SoundWaveDB_CH.git
2. Abrir MySQL Workbench o una herramienta compatible.
3. Ejecuta el script para crear la base de datos, insertar los datos y configurar las funcionalidades adicionales.

12. Herramientas Utilizadas

Para el desarrollo del proyecto SoundWaveDB se utilizaron las siguientes herramientas:

- **MySQL Workbench:** herramienta principal para la creación del esquema de base de datos, ejecución de consultas SQL, creación de vistas, funciones, procedimientos almacenados y triggers.
- **Lenguaje SQL (MySQL):** utilizado para definir la estructura de la base de datos, insertar registros y desarrollar objetos como vistas, funciones, procedimientos y triggers.
- **GitHub:** repositorio online para almacenar y compartir los scripts del proyecto de manera organizada y accesible. Permite el control de versiones y la documentación del trabajo realizado.
- **Word:** para la redacción del informe técnico que acompaña al proyecto.

13. Conclusión

En conclusión, SoundWaveDB representa una solución integral y sólida para la gestión de datos en una plataforma de streaming musical. La incorporación de vistas complejas, funciones, procedimientos almacenados y triggers ha permitido optimizar tanto las consultas como la integridad de la información, asegurando que las relaciones entre las diferentes entidades se mantengan de manera consistente. Este proyecto refleja la aplicación de los conocimientos adquiridos en el curso y demuestra mi compromiso por desarrollar una base de datos robusta y escalable, capaz de adaptarse a futuras mejoras y necesidades.