PROYECTO MUSIVERSE

MÓDULO: INNOVACIÓN EN GESTIÓN DE DATOS

Asignaturas: Base de Datos II y Programación I

Nombres de los integrantes:

- Lucas Sebastián Brocanelli
- Marco Virinni
- Emilce Robles
- Mario Arce
- Gabriela Eliana Acosta

Fecha de entrega: 6/9/2024

PROYECTO MUSIVERSE

a. Descripción del tema elegido para el proyecto final:

El proyecto final consiste en el desarrollo de un programa que permite a los usuarios gestionar y reproducir música mediante una base de datos relacional. Los usuarios podrán buscar canciones por género, banda o nombre, crear playlists personalizadas, y marcar sus canciones favoritas. También incluye la funcionalidad de llevar un historial de reproducción con el horario en que se escuchó cada canción, así como la posibilidad de escuchar una canción aleatoria según el género musical preferido. El backend estará desarrollado en Python y la base de datos será gestionada con MySQL.

b. Objetivos principales:

1. Permitir a los usuarios gestionar su música favorita mediante la creación de playlists y marcando canciones como favoritas.
2. Implementar una función de búsqueda avanzada de canciones por nombre, banda o género.
3. Registrar un historial de reproducción que incluya la fecha y hora de cada escucha.
4. Desarrollar una función que sugiera canciones aleatorias según el género musical preferido del usuario.
5. Crear una aplicación robusta y escalable que pueda compartir datos entre diferentes usuarios en futuros sprints.

c. Metodología a utilizar:

El desarrollo del proyecto se basará en la metodología ágil, utilizando sprints para la entrega gradual de funcionalidades.

- **Análisis y diseño:** Se utilizará un DER (Diagrama Entidad-Relación) y un MR (Modelo Relacional) para la estructuración de la base de datos.
- **Desarrollo:** Python será el lenguaje de programación principal, y MySQL se encargará de la persistencia de los datos.
- **Pruebas:** Se realizarán pruebas unitarias y de integración para asegurar el correcto funcionamiento de las funcionalidades implementadas.
- **Documentación:** Se mantendrá una documentación clara y detallada de cada sprint.

d. Relevancia del proyecto en el contexto académico o profesional:

Este proyecto es relevante tanto en el contexto académico como profesional, ya que integra conocimientos de desarrollo backend, manejo de bases de datos relacionales, y gestión de proyectos ágil. Además, el desarrollo de un sistema de gestión musical es un excelente ejercicio práctico para aplicar el conocimiento adquirido en áreas como la programación orientada a objetos, consultas SQL y la implementación de algoritmos de búsqueda y recomendación. Profesionales en el área de desarrollo de software pueden beneficiarse de este proyecto, ya que simula escenarios reales de trabajo con bases de datos y proporciona experiencia en la creación de aplicaciones escalables.

Aunque el uso del programa, desde la perspectiva de un usuario final, es más interactivo y de entretenimiento.

Detalles de la Base de Datos:

1. Descripción General:

La base de datos que sustenta nuestra aplicación de música está diseñada para gestionar de forma eficiente los datos relacionados con usuarios, canciones, artistas, álbumes, géneros musicales y actividades de los usuarios, como la creación de playlists y el registro de

canciones favoritas. Esta base permite a los usuarios buscar música y realizar acciones como guardar canciones en sus listas de reproducción o favoritos, así como registrar el historial de reproducción, todo con un enfoque en la personalización.

2. Tablas:

Las tablas forman la estructura principal del sistema, cada una con un propósito específico:

- **usuario**: Almacena la información del usuario.
- **playlist**: Contiene las playlists creadas por los usuarios.
- **favoritos**: Gestiona las canciones que el usuario marca como favoritas.
- **historialreproduccion**: Registra cada reproducción realizada por un usuario.
- **canciones**: Contiene la información de las canciones.
- **playlistcanciones**: Es una tabla intermedia que conecta las playlists con las canciones.
- **album**: Contiene los álbumes a los que pertenecen las canciones.
- **artistas**: Guarda la información de los artistas que lanzan álbumes.
- **generomusical**: Almacena los géneros musicales.

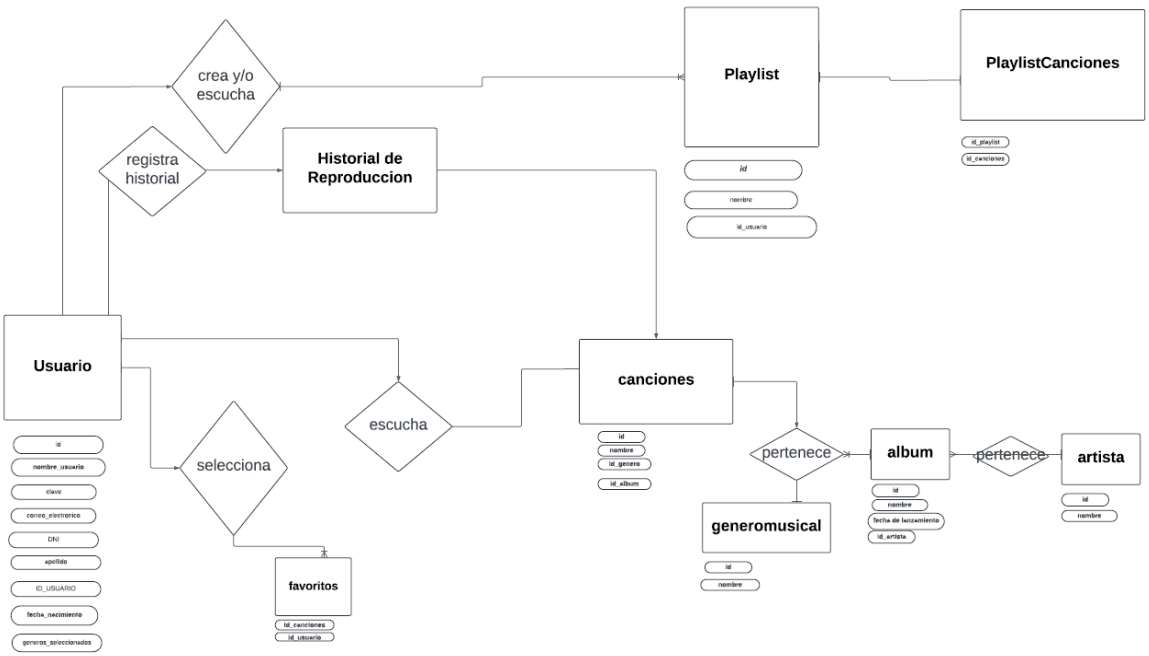
3. Relaciones:

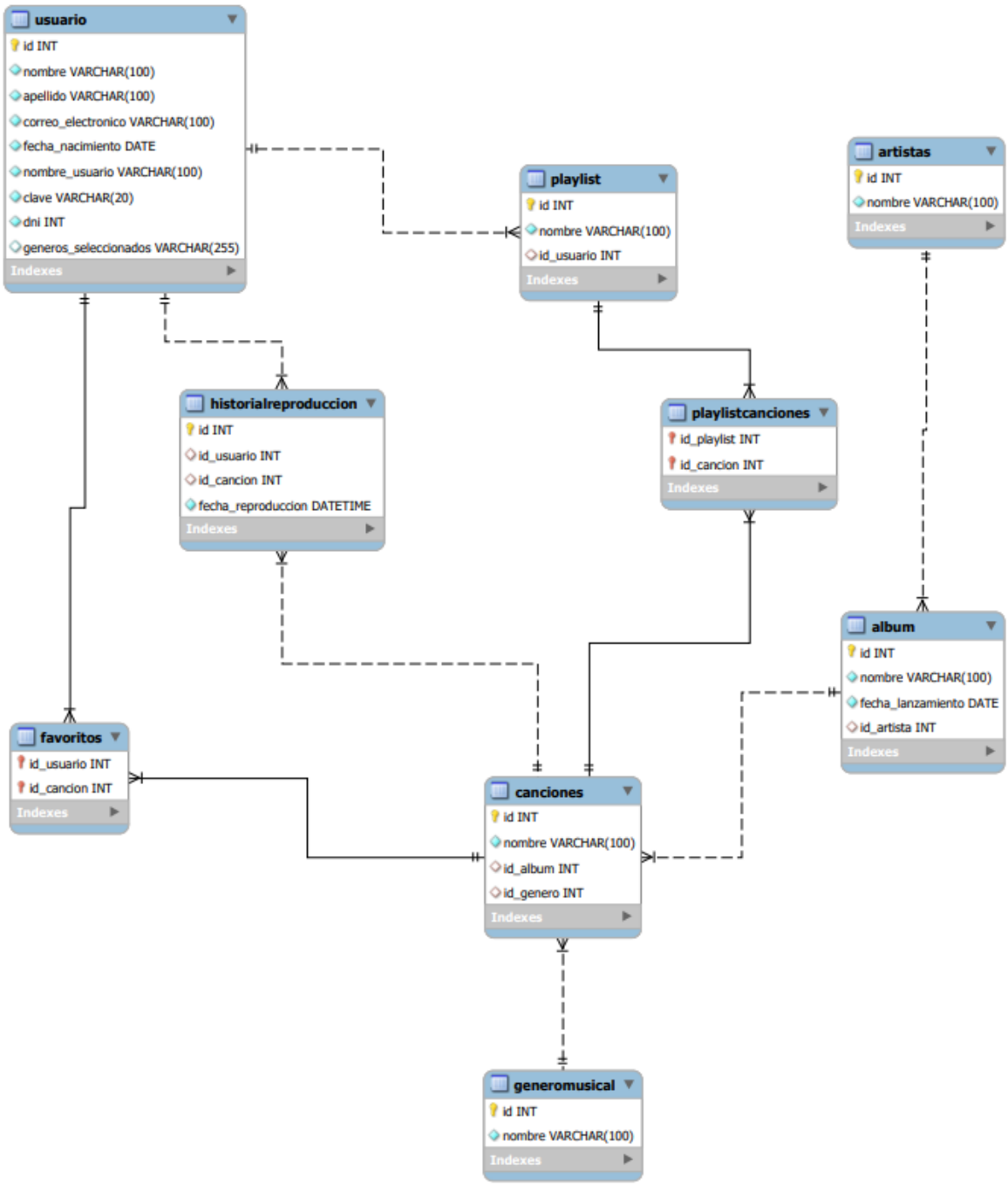
- **Usuario - Playlist (1)**:
    - **Descripción**: Un usuario puede crear varias playlists, pero cada playlist pertenece únicamente a un usuario. Esto se refleja en la clave foránea id_usuario dentro de la tabla playlist.
    - **Cardinalidad**: Un usuario tiene muchas playlists, pero cada playlist pertenece a un solo usuario.
- **Playlist - Canciones (N)**:

    - **Descripción**: Una playlist puede contener muchas canciones y, a su vez, una canción puede estar en varias playlists. La relación es gestionada mediante una tabla intermedia playlistcanciones, que tiene dos claves foráneas: id_playlist (de la tabla playlist) e id_cancion (de la tabla canciones).
    - **Cardinalidad**: Relación de muchos a muchos (N).
- **Usuario - Favoritos (1)**:
    - **Descripción**: Un usuario puede tener muchas canciones en su lista de favoritos, pero cada canción en favoritos está asociada a un solo usuario. La tabla favoritos contiene claves foráneas tanto de usuario como de canciones.
    - **Cardinalidad**: Un usuario tiene muchas canciones favoritas, pero una canción en favoritos pertenece a un único usuario dentro de esta relación.
- **Canción - Álbum (1)**:
    - **Descripción**: Un álbum puede contener muchas canciones, pero cada canción pertenece a un solo álbum. Esto se refleja en la clave foránea id_album dentro de la tabla canciones.
    - **Cardinalidad**: Un álbum tiene muchas canciones, pero una canción pertenece a un solo álbum.
- **Álbum - Artista (1)**:
    - **Descripción**: Un artista puede haber lanzado varios álbumes, pero cada álbum está asociado a un solo artista. Esto se maneja con la clave foránea id_artista dentro de la tabla album.
    - **Cardinalidad**: Un artista puede tener varios álbumes, pero un álbum es creado por un solo artista.
- **Canción - Género Musical (1)**:

- **Descripción**: Un género musical puede tener muchas canciones, pero cada canción pertenece a un único género musical. Esto se representa mediante la clave foránea id_genero en la tabla canciones.
- **Cardinalidad**: Un género tiene muchas canciones, pero una canción tiene solo un género.

- **Usuario - Historial de Reproducción (1)**:
  - **Descripción**: Un usuario puede reproducir varias canciones, y cada reproducción se registra con la canción y el momento en que fue reproducida. Esto se refleja en la tabla historialreproduccion que contiene claves foráneas tanto de usuario como de canciones.
  - **Cardinalidad**: Un usuario puede reproducir muchas canciones (con múltiples registros de cada reproducción), pero una entrada en el historial corresponde a un solo usuario.





4. Conclusión:

Este proyecto ofrece una solución eficaz para la gestión de usuarios, canciones y playlists, con una estructura de base de datos diseñada para organizar las interacciones clave de manera eficiente, lo que permite una experiencia personalizada para los usuarios. El uso de herramientas accesibles como MySQL y Python garantiza un desarrollo manejable, mientras que las funcionalidades personalizadas, como las recomendaciones basadas en el historial de

escucha, mejoran la experiencia general. Además, la democratización del acceso a la música fomenta la diversidad cultural y apoya a artistas independientes.

Sin embargo, la viabilidad de una aplicación de música en un mercado competitivo requiere una clara propuesta de valor. Competir con plataformas establecidas como Spotify y Apple Music implica ofrecer funcionalidades innovadoras y una estrategia de monetización sólida, ya sea mediante suscripciones, anuncios o colaboraciones. El éxito del proyecto dependerá no solo de la optimización del rendimiento y las consultas, sino también de identificar un nicho que permita destacarse y asegurar una excelente experiencia de usuario en un entorno en constante evolución.