ÍNDICE

1. REQUISITOS DEL SISTEMA

1.1. FASE INICIAL

A modo de introducción, explicaré de manera general en lo que se basa este proyecto. La idea es agrupar todos los posibles conocimientos del primer año de DAM (Desarrollo de Aplicaciones Multiplataforma), en un único proyecto, abarcando conocimientos como conceptos básicos de bases de datos relacionales, fundamentos de la programación orientada a objetos en Java, estructura básica de páginas web con HTML, estilización de páginas web con CSS e interactividad en la página con JavaScript, conceptos básicos de pruebas de software, y la correspondiente documentación del código, entre otros.

El proyecto se centra en desarrollar una aplicación, la cual, será una plataforma para gestionar información sobre artistas, canciones, álbumes y profesionales relacionados en la industria musical. Permitirá a los usuarios almacenar, editar y visualizar datos detallados, ofreciendo funcionalidades avanzadas como búsquedas específicas, análisis de estadísticas y comparaciones entre artistas. La aplicación estará optimizada para un alto rendimiento, seguridad y usabilidad, asegurando una experiencia fluida y accesible. Además, facilitará la integración con otros sistemas y proporcionará una interfaz intuitiva y adaptable a diferentes dispositivos. En resumen, será una herramienta robusta y versátil para organizar y analizar la información musical de manera eficiente. A continuación, detallaremos cada una de estas funcionalidades.

1.2. ANÁLISIS

A continuación, especificaré los requisitos funcionales y no funcionales, proporcionando una breve descripción de cada uno para entender sus diferencias. Los requisitos funcionales detallan las funciones y características específicas que debe cumplir el sistema, mientras que los requisitos no funcionales describen los criterios de calidad y restricciones operativas que debe seguir el sistema para garantizar su rendimiento y usabilidad.

1.2.1 REQUISITOS FUNCIONALES

1) *Gestión de Artistas*
   - Agregar, editar y eliminar artistas: Permitir la gestión completa de los registros de artistas.
   - Visualizar detalles de un artista: Mostrar toda la información de un artista, incluyendo canciones, discos, premios, conciertos, etc.
   - Buscar artistas por diferentes criterios: Nombre artístico, país, ciudad, género, estado, etc.
   - Listar artistas activos, retirados y fallecidos.
2) *Gestión de Representantes*
   - Agregar, editar y eliminar representantes.

- Visualizar detalles de un representante, incluyendo la lista de artistas representados.
- Buscar representantes por nombre, nacionalidad, etc.

3) *Gestión de Productores*

- Agregar, editar y eliminar productores.
- Visualizar detalles de un productor, incluyendo canciones producidas.
- Buscar productores por nombre, nacionalidad, etc.

4) *Gestión de Redes Sociales*

- Agregar, editar y eliminar redes sociales de los artistas.
- Visualizar detalles de las redes sociales de un artista, incluyendo número de seguidores.

5) *Gestión de Premios*

- Agregar, editar y eliminar premios.
- Visualizar detalles de un premio, incluyendo el ganador y la organización que lo entrega.
- Buscar premios por nombre, fecha, etc.

6) *Gestión de Canciones*

- Agregar, editar y eliminar canciones.
- Visualizar detalles de una canción, incluyendo artistas, álbum, productor, etc.
- Buscar canciones por título, artista, género, etc.
- Mostrar el número total de reproducciones de una canción.

7) *Gestión de Conciertos*

- Agregar, editar y eliminar conciertos.
- Visualizar detalles de un concierto, incluyendo artistas colaboradores, lugar, duración, etc.
- Buscar conciertos por artista, país, ciudad, fecha, etc.

8) *Gestión de Géneros*

- Agregar, editar y eliminar géneros.
- Visualizar detalles de un género, incluyendo su país de origen.
- Buscar géneros por nombre, país de origen, etc.

9) *Gestión de Álbumes*

- Agregar, editar y eliminar álbumes.
- Visualizar detalles de un álbum, incluyendo canciones, artistas colaboradores, géneros, etc.
- Buscar álbumes por título, artista, género, etc.
- Mostrar el número total de reproducciones de un álbum.

10) *Funcionalidades Adicionales*

- Artista con más conciertos: Función que permita identificar al artista con más conciertos realizados.

- Álbum con más y menos reproducciones: Función para encontrar los álbumes con más y menos reproducciones totales.
- Artista con más premios ganados: Identificar al artista que ha ganado más premios.
- Canción más reproducida: Función para encontrar la canción con el mayor número de reproducciones.
- Búsqueda avanzada de conciertos: Permitir búsquedas por rango de fechas, país y ciudad.
- Estadísticas de géneros musicales: Mostrar estadísticas de géneros más interpretados por diferentes artistas.
- Conciertos próximos: Listar los próximos conciertos de cada artista.
- Historial de premios de un artista: Mostrar un historial de premios ganados por un artista a lo largo de su carrera.
- Comparación entre artistas: Comparar dos o más artistas en términos de oyentes de Spotify, número de premios, canciones, discos, etc.
- Informe de carrera de un artista: Generar un informe completo sobre la carrera de un artista, incluyendo todos los conciertos dados, premios ganados, álbumes y canciones publicadas.

1.2.1 REQUISITOS NO FUNCIONALES

1) *Rendimiento*

- Tiempo de respuesta: Las consultas y operaciones deben completarse en menos de 2 segundos para mantener una experiencia de usuario fluida.
- Escalabilidad: La aplicación debe poder manejar un aumento en la cantidad de datos (por ejemplo, miles de artistas, canciones, álbumes, etc.) sin degradación significativa del rendimiento.

2) *Seguridad*

- Autenticación y autorización: Implementar un sistema de autenticación para usuarios y un control de acceso basado en roles (administradores, editores, usuarios estándar).
- Protección de datos: Encriptar datos sensibles como contraseñas, correos electrónicos y números de teléfono.
- Auditoría y logs: Mantener registros de acceso y modificaciones de datos para auditorías de seguridad.

3) *Usabilidad*

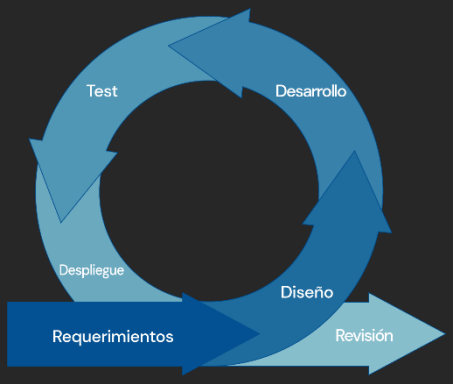
- Interfaz de usuario intuitiva: La interfaz debe ser fácil de usar y entender, con una navegación clara y coherente.
- Accesibilidad: La aplicación debe cumplir con las pautas de accesibilidad web (WCAG) para ser utilizable por personas con discapacidades.
- Compatibilidad con dispositivos móviles: La interfaz debe ser responsiva, adaptándose correctamente a diferentes tamaños de pantalla y dispositivos.

4) *Mantenimiento*
   - Documentación: Proveer documentación detallada y actualizada del código, así como manuales de usuario y guías de instalación.
   - Modularidad: El diseño de la aplicación debe ser modular para facilitar el mantenimiento y las futuras actualizaciones.
   - Pruebas automatizadas: Implementar un conjunto de pruebas automatizadas para asegurar la funcionalidad del sistema después de cambios y actualizaciones.
5) *Fiabilidad*
   - Disponibilidad: La aplicación debe tener una disponibilidad mínima del 99.9%, asegurando que esté accesible en la mayoría del tiempo.
   - Recuperación ante fallos: Implementar mecanismos de respaldo y recuperación de datos para minimizar la pérdida de información en caso de fallos del sistema.
6) *Escalabilidad y Rendimiento*
   - Carga máxima: La aplicación debe soportar al menos 1000 usuarios concurrentes sin pérdida significativa de rendimiento.
   - Optimización de consultas: Optimizar las consultas a la base de datos para asegurar tiempos de respuesta rápidos incluso con grandes volúmenes de datos.
7) *Portabilidad*
   - Compatibilidad con múltiples plataformas: La aplicación debe ser compatible con los principales sistemas operativos (Windows, macOS, Linux) y navegadores web (Chrome, Firefox, Edge, Safari).
   - Implementación en la nube: Facilitar la implementación en servicios de nube como AWS, Azure o Google Cloud para aprovechar la escalabilidad y disponibilidad que ofrecen estas plataformas.
8) *Integración*
   - Importación y exportación de datos: Permitir la importación y exportación de datos en formatos comunes (CSV, JSON, XML) para facilitar la interoperabilidad con otros sistemas.
9) *Localización e Internacionalización*
   - Soporte multilingüe: La aplicación debe soportar múltiples idiomas y permitir la fácil adición de nuevas traducciones.
   - Formatos regionales: Adaptarse a formatos de fecha, hora, y moneda según la configuración regional del usuario.

2. MODELO DE CICLO DE VIDA

Ante todos los posibles modelos de ciclo de vida que existen, como el modelo en cascada o el modelo en espiral, he decidido escoger el modelo de Desarrollo Ágil. Este modelo es una elección adecuada debido a su capacidad para adaptarse a cambios, fomentar la colaboración con los clientes, entregar valor de manera temprana, gestionar riesgos de manera continua y mantener una experiencia de usuario óptima. Estas características son esenciales para el éxito de una plataforma de información sobre artistas y música en un mercado competitivo y en constante cambio. Debido a estas ventajas, este modelo es el más apropiado para este proyecto, ya que otros modelos tienen una serie de desventajas. Por ejemplo, la falta de flexibilidad del modelo en cascada hace difícil ajustarse a las demandas cambiantes del mercado, y el enfoque más costoso y riguroso en la gestión de riesgos del Modelo en Espiral puede no ser ideal para un proyecto de esta naturaleza. Este modelo implica una planificación y evaluación exhaustiva de riesgos en cada fase, lo que puede resultar en una mayor inversión en tiempo y recursos, algo que puede ser costoso para proyectos con presupuestos ajustados, como el desarrollo de una plataforma de información musical. Además, la implementación exitosa del Modelo en Espiral requiere una planificación y gestión minuciosas, lo que puede ser más complejo y requerir una inversión de recursos adicional.

El modelo en espiral es más adecuado para proyectos de mayor complejidad, como sistemas críticos para la seguridad o la industria aeroespacial, donde la gestión de riesgos es fundamental. Para una plataforma de información sobre artistas y música, que, si bien puede ser compleja, no está en el mismo nivel de complejidad que los ejemplos anteriores, el Modelo en Espiral podría resultar más costoso y riguroso de lo necesario. En lugar de un enfoque centrado en la evaluación continua de riesgos, el Desarrollo Ágil se enfoca en la entrega temprana de valor al cliente, la adaptabilidad a cambios y la satisfacción del usuario, lo que es esencial en un mercado competitivo y en constante cambio como la información musical en línea. Esto hace que el Desarrollo Ágil sea más adecuado para proyectos de naturaleza similar y con presupuestos y recursos limitados.



3. PLANIFICACIÓN DE CODIFICACIÓN

3.1. SERVIDOR

En cuanto al servidor de mi proyecto, voy a desarrollarlo todo en local. Respecto a la base de datos, utilizaré Oracle SQL. Utilizar estas herramientas ofrece varias ventajas: trabajar en un entorno local permite un control completo sobre el desarrollo y la configuración del servidor. Oracle SQL, por su parte, es una base de datos robusta y segura, ideal para gestionar grandes volúmenes de datos y ofrecer un rendimiento fiable en entornos empresariales.

3.2. LADO DEL CLIENTE

Para esta parte, utilizaremos los lenguajes HTML, CSS y JavaScript, ya que son esenciales para el desarrollo web, y tiene gran compatibilidad con los navegadores. HTML sería imprescindible para la estructura de la web, CSS para el diseño y decoración y JavaScript para diferentes funciones, así como animaciones interactivas e interactividad con el usuario.

3.3. HERRAMIENTAS

Para el desarrollo de mi proyecto, utilizaré diversas herramientas y metodologías que me permitirán optimizar y gestionar de manera eficiente el ciclo de vida del software. En cuanto al entorno de desarrollo integrado (IDE), emplearé tanto Visual Studio Code como Eclipse. Visual Studio Code es una herramienta ligera y altamente configurable que me permitirá editar código de manera rápida y eficiente, mientras que Eclipse es un IDE robusto que ofrece herramientas avanzadas para el desarrollo de aplicaciones Java, incluyendo la optimización y refactorización del código.

En relación con el control de versiones, utilizaré Git junto con GitHub. Git me permitirá llevar un control preciso de los cambios realizados en el código, facilitando la colaboración y la integración continua. GitHub servirá como plataforma para alojar el repositorio, permitiendo el acceso remoto y la gestión de versiones del proyecto.

Para la automatización de tareas, realizaré pruebas automatizadas que garantizarán la calidad y fiabilidad del software desarrollado. Además, generaré documentación utilizando Javadoc, lo que permitirá una comprensión clara y detallada del código. Para la creación de diagramas y la generación automática de código, emplearé Visual Paradigm, una herramienta que facilita la modelización visual y la generación de código a partir de los diagramas diseñados.

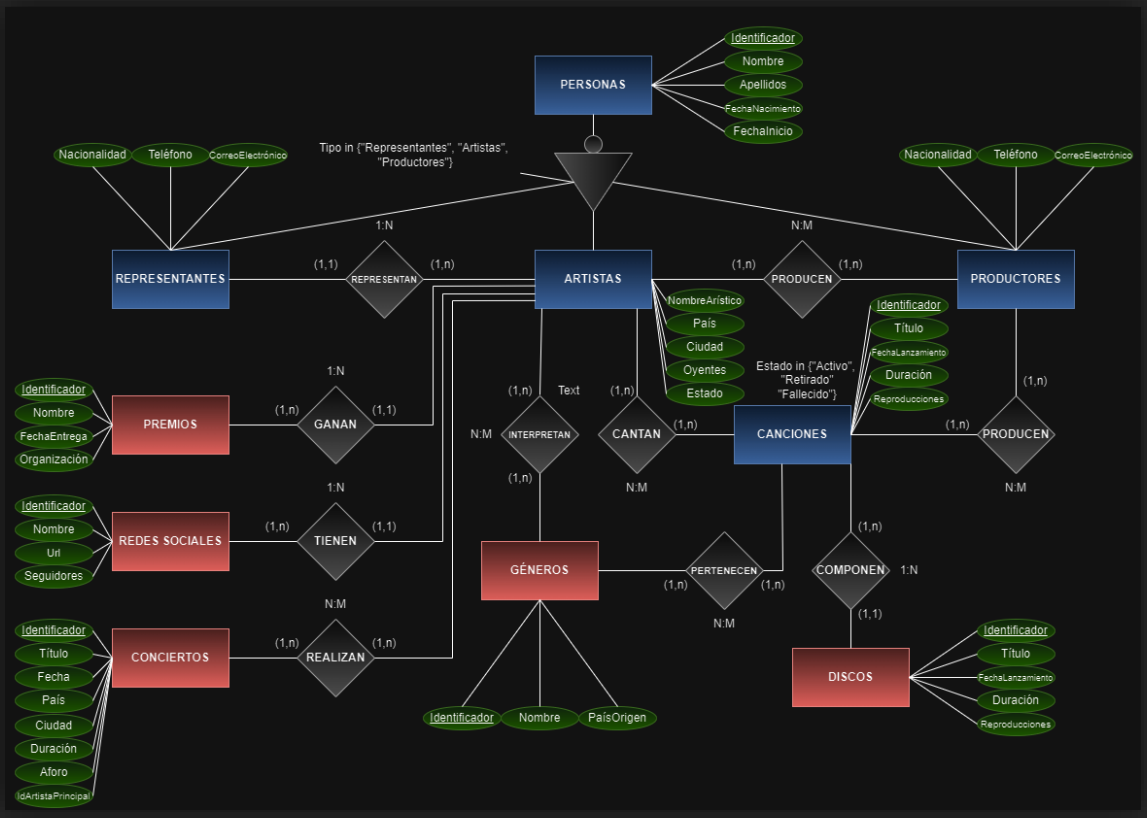
Finalmente, utilizaré las herramientas de Eclipse para optimizar y refactorizar el código, asegurando que el software sea eficiente, mantenible y de alta calidad. Esta combinación de herramientas y metodologías me permitirá desarrollar un proyecto robusto y bien documentado, con un flujo de trabajo eficiente y organizado.

4. PLANIFICACIÓN DE LAS FASES RESTANTES DEL CICLO DE VIDA

4.1. FASE DE DISEÑO

A continuación, explicaré los tres tipos de diagramas esenciales para el desarrollo del sistema. Primero, el Modelo Entidad-Relación (MER) que se utilizará para diseñar la estructura de la base de datos en Oracle SQL, permitiendo definir las entidades y sus relaciones de manera clara. Segundo, el Diagrama de Clases que se empleará para la aplicación en Java, detallando las clases, sus atributos, métodos y las relaciones entre ellas. Finalmente, el Diagrama de Casos de Uso que ilustrará la interacción de los usuarios con la aplicación de Java, describiendo las funcionalidades del sistema desde la perspectiva del usuario y facilitando la identificación de los requisitos funcionales.

4.1.1 MER



PERSONAS (Identificador, Nombre, Apellidos, FechaNacimiento, FechaInicio, Tipo)
PK: Identificador CK: Tipo in [“Representante”, “Artista”, “Productor”] NN: Nombre, Apellidos, FechaNacimiento, FechaInicio, Tipo

REPRESENTANTES (Identificador, Nacionalidad, Teléfono, CorreoElectrónico)
PK: Identificador FK: Identificador → **PERSONAS** NN: Nacionalidad, Teléfono, CorreoElectrónico

ARTISTAS (Identificador, NombreArtístico, País, Ciudad, Oyentes, Estado. IdentificadorRepresentante)
PK: Identificador FK: Identificador → **PERSONAS** IdentificadorRepresentante → **REPRESENTANTES** CK: Estado in [“Activo”, “Retirado”, “Fallecido”] NN: NombreArtístico, País, Ciudad, Oyentes, Estado, IdentificadorRepresentante

PRODUCTORES (Identificador, Nacionalidad, Teléfono, CorreoElectrónico)
PK: Identificador FK: Identificador → **PERSONAS** NN: Nacionalidad, Teléfono, CorreoElectrónico

GÉNEROS (Identificador, Nombre, PaísOrigen)
PK: Identificador NN: Nombre, PaísOrigen

DISCOS (Identificador, Título, FechaLanzamiento, Duración, Reproducciones)
PK: Identificador NN: Título, FechaLanzamiento, Duración, Reproducciones

CANCIONES (Identificador, Título, FechaLanzamiento, Duración, Reproducciones. IdentificadorDisco)
PK: Identificador CK: IdentificadorDisco → **DISCOS** NN: Título, FechaLanzamiento, Duración, Reproducciones

PREMIOS (Identificador, Nombre, FechaEntrega, Organización, IdentificadorArtista)
PK: Identificador CK: IdentificadorArtista → **ARTISTAS** NN: Nombre, FechaEntrega, Organización, IdentificadorArtista

REDES (Identificador, Nombre, Url, Seguidores, IdentificadorArtista)
PK: Identificador CK: IdentificadorArtista → **ARTISTAS** NN: Nombre, Url, Seguidores, IdentificadorArtista

CONCIERTOS (Identificador, Título, Fecha, País, Ciudad, Duración, Aforo, IdArtistaPrincipal)
PK: Identificador UK: IdArtistaPrincipal FK: IdArtistaPrincipal → **ARTISTAS** NN: Título, Fecha, País, Ciudad, Duración, Aforo

PRODUCTORES_PRODUCEN_ARTISTAS (IdentificadorProductor, IdentificadorArtista)
PK: IdentificadorProductor, IdentificadorArtista FK: IdentificadorProductor → **PRODUCTOR** IdentificadorArtista → **ARTISTA**

PRODUCTORES_PRODUCEN_CANCIONES (IdentificadorProductor, IdentificadorCancion)
PK: IdentificadorProductor, IdentificadorCancion FK: IdentificadorProductor → **PRODUCTOR** IdentificadorArtista → **CANCIONES**

ARTISTAS_CANTAN _CANCIONES (IdentificadorArtista, IdentificadorCancion)
PK: IdentificadorArtista, IdentificadorCancion FK: IdentificadorArtista → **ARTISTAS** IdentificadorArtista → **CANCIONES**

ARTISTAS_INTERPRETAN _GÉNEROS (IdentificadorArtista, IdentificadorGenero)
PK: IdentificadorArtista, IdentificadorGenero FK: IdentificadorArtista → **ARTISTAS** IdentificadorGenero → **GÉNEROS**

ARTISTAS_REALIZAN _CONCIERTOS (IdentificadorArtista, IdentificadorGenero)
PK: IdentificadorArtista, IdentificadorConcierto FK: IdentificadorArtista → **ARTISTAS** IdentificadorConcierto → **CONCIERTOS**

ACLARACIONES

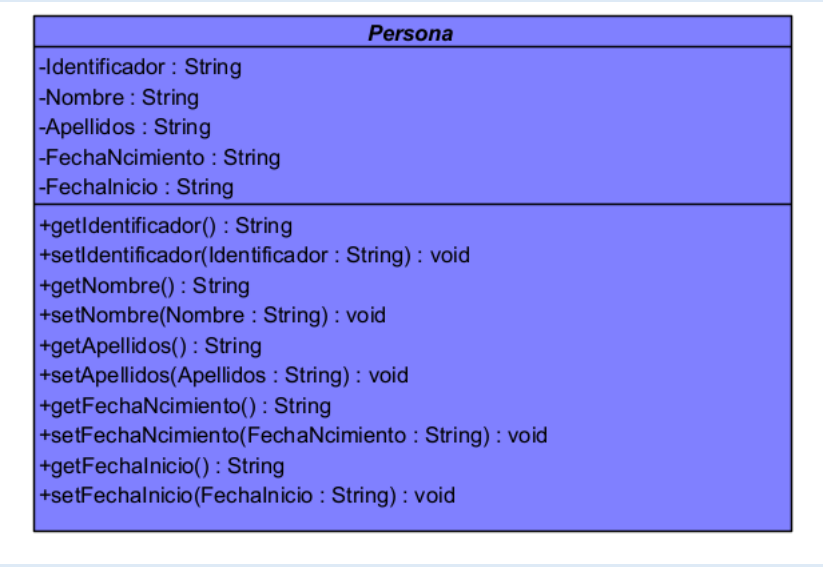
- La jerarquía es solapada, ya que se puede dar el caso que un artista sea productor de otros artistas.
- El identificador del disco en las canciones puede ser nulo, ya que hay canciones que no pertenecen a ningún disco.
- Una canción puede pertenecer a un único disco.
- Una canción puede pertenecer a más de un género a la vez, ya que hay canciones que son una mezcla de varios géneros.
- La red social que se guarda es única, ya que contiene un URL específico de un artista en específico de una plataforma en específico.
- En un concierto puede haber varios artistas ya que, a parte del artista principal, puede haber colaboraciones y que salgan a cantar canciones más de un artista.

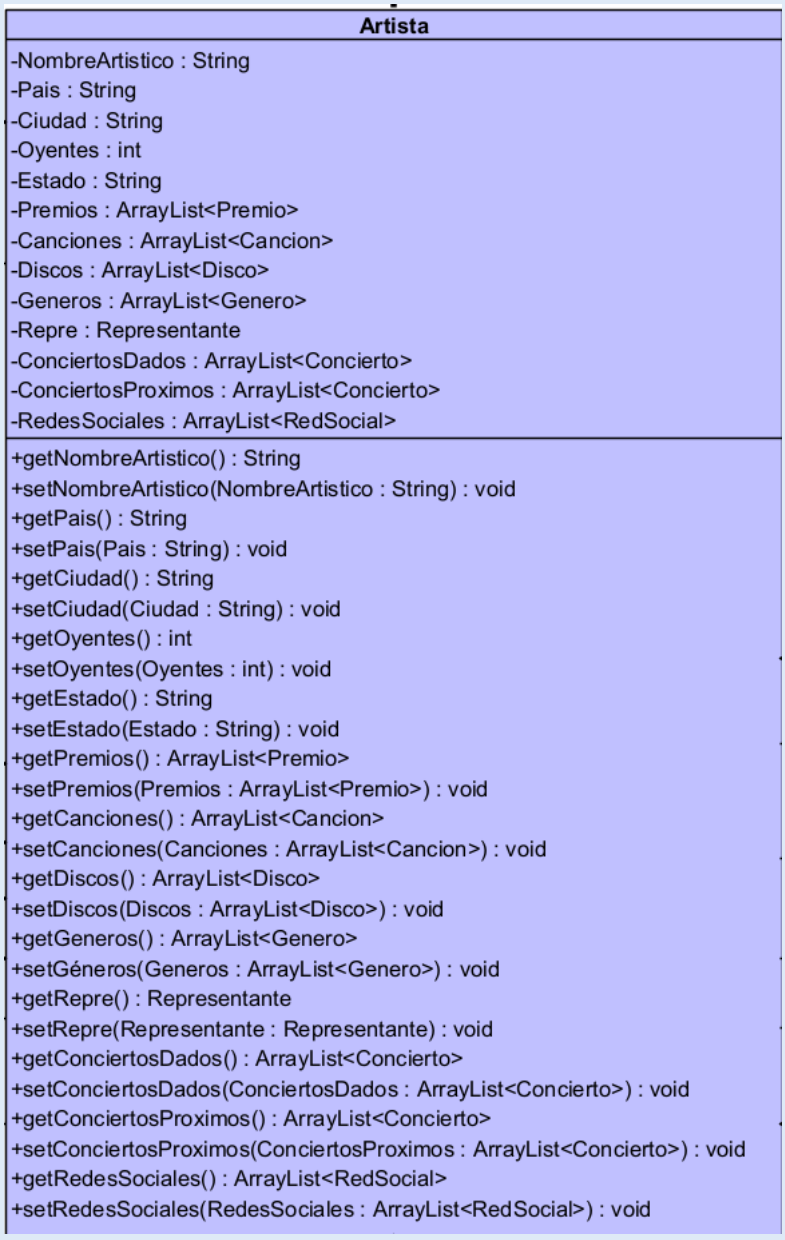
ENLACES

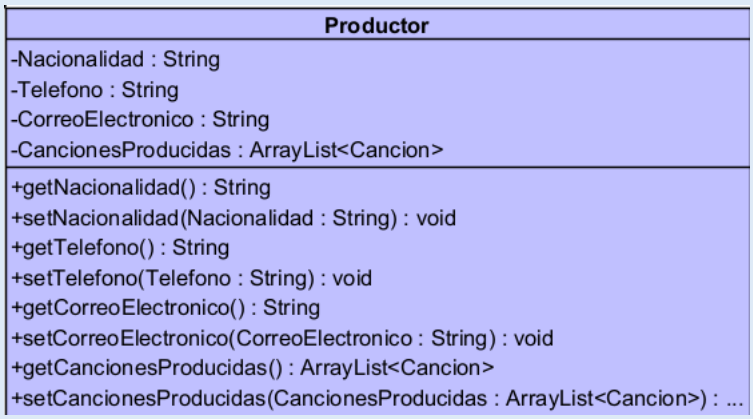
MER
https://github.com/Josecp03/Proyecto_1DAM/blob/main/BDD/MER.png
CREACIÓN DE USUARIO
https://github.com/Josecp03/Proyecto_1DAM/blob/main/BDD/Creaci%C3%B3nUsuario.sql
BASE DE DATOS
https://github.com/Josecp03/Proyecto_1DAM/blob/main/BDD/BD.sql
PRUEBAS
https://github.com/Josecp03/Proyecto_1DAM/blob/main/BDD/Pruebas.sql

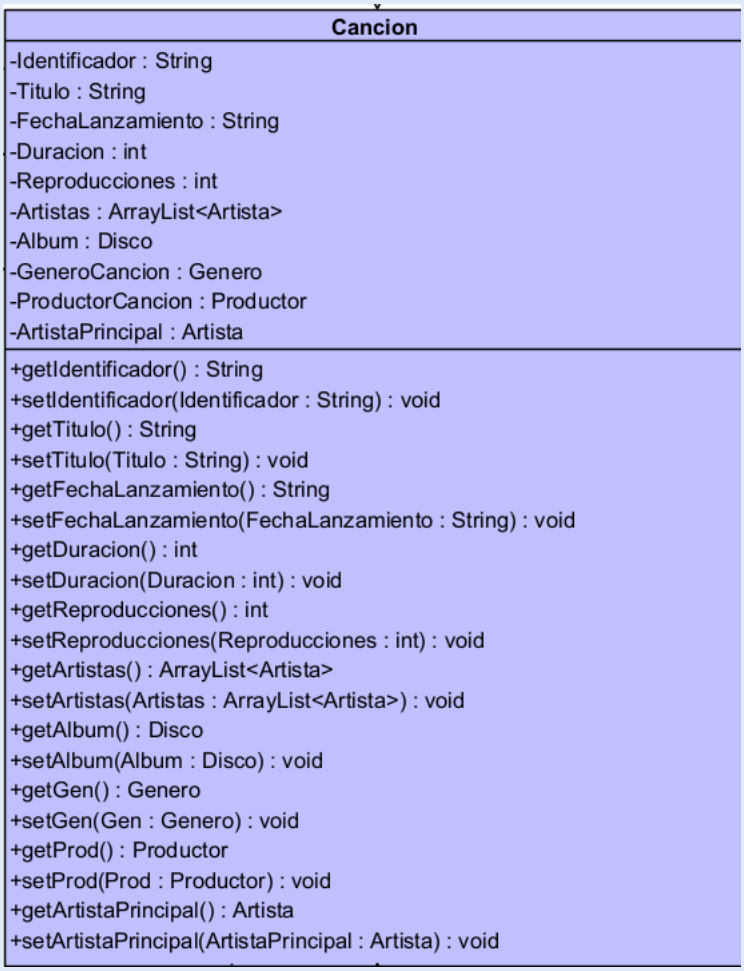
4.1.1 DIAGRAMA DE CLASES

CLASES

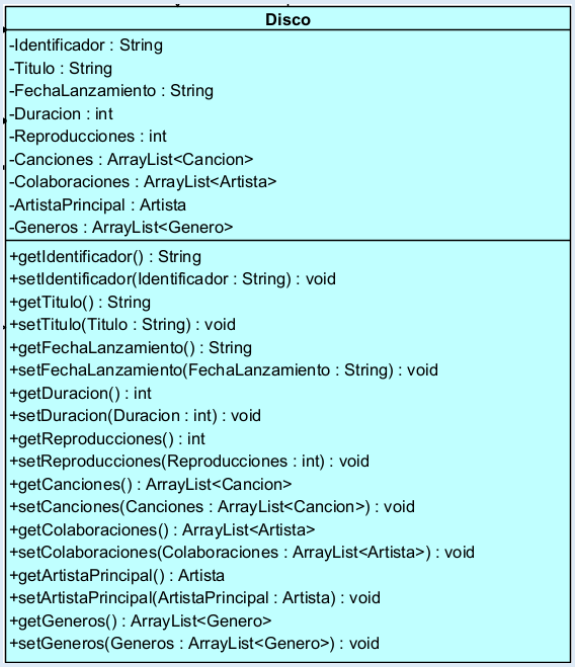


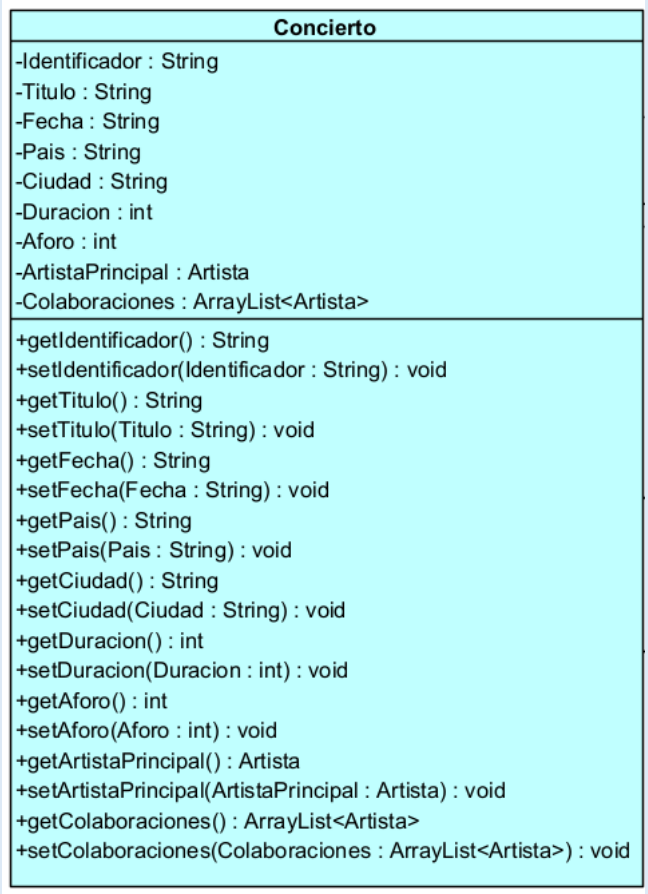

Artista
-NombreArtistico : String
-Pais : String
-Ciudad : String
-Oyentes : int
-Estado : String
-Premios : ArrayList<Premio>
-Canciones : ArrayList<Cancion>
-Discos : ArrayList<Disco>
-Generos : ArrayList<Genero>
-Repre : Representante
-ConciertosDados : ArrayList<Concierto>
-ConciertosProximos : ArrayList<Concierto>
-RedesSociales : ArrayList<RedSocial>
+getNombreArtistico() : String
+setNombreArtistico(NombreArtistico : String) : void
+getPais() : String
+setPais(Pais : String) : void
+getCiudad() : String
+setCiudad(Ciudad : String) : void
+getOyentes() : int
+setOyentes(Oyentes : int) : void
+getEstado() : String
+setEstado(Estado : String) : void
+getPremios() : ArrayList<Premio>
+setPremios(Premios : ArrayList<Premio>) : void
+getCanciones() : ArrayList<Cancion>
+setCanciones(Canciones : ArrayList<Cancion>) : void
+getDiscos() : ArrayList<Disco>
+setDiscos(Discos : ArrayList<Disco>) : void
+getGeneros() : ArrayList<Genero>
+setGéneros(Generos : ArrayList<Genero>) : void
+getRepre() : Representante
+setRepre(Representante : Representante) : void
+getConciertosDados() : ArrayList<Concierto>
+setConciertosDados(ConciertosDados : ArrayList<Concierto>) : void
+getConciertosProximos() : ArrayList<Concierto>
+setConciertosProximos(ConciertosProximos : ArrayList<Concierto>) : void
+getRedesSociales() : ArrayList<RedSocial>
+setRedesSociales(RedesSociales : ArrayList<RedSocial>) : void


Productor
-Nacionalidad : String
-Telefono : String
-CorreoElectronico : String
-CancionesProducidas : ArrayList<Cancion>
+getNacionalidad() : String
+setNacionalidad(Nacionalidad : String) : void
+getTelefono() : String
+setTelefono(Telefono : String) : void
+getCorreoElectronico() : String
+setCorreoElectronico(CorreoElectronico : String) : void
+getCancionesProducidas() : ArrayList<Cancion>
+setCancionesProducidas(CancionesProducidas : ArrayList<Cancion>) : ...

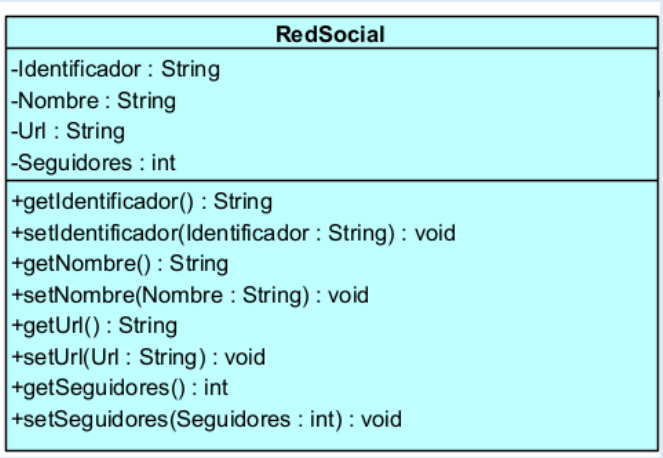

Cancion
-Identificador : String
-Titulo : String
-FechaLanzamiento : String
-Duracion : int
-Reproducciones : int
-Artistas : ArrayList<Artista>
-Album : Disco
-GeneroCancion : Genero
-ProductorCancion : Productor
-ArtistaPrincipal : Artista
+getIdentificador() : String
+setIdentificador(Identificador : String) : void
+getTitulo() : String
+setTitulo(Titulo : String) : void
+getFechaLanzamiento() : String
+setFechaLanzamiento(FechaLanzamiento : String) : void
+getDuracion() : int
+setDuracion(Duracion : int) : void
+getReproducciones() : int
+setReproducciones(Reproducciones : int) : void
+getArtistas() : ArrayList<Artista>
+setArtistas(Artistas : ArrayList<Artista>) : void
+getAlbum() : Disco
+setAlbum(Album : Disco) : void
+getGen() : Genero
+setGen(Gen : Genero) : void
+getProd() : Productor
+setProd(Prod : Productor) : void
+getArtistaPrincipal() : Artista
+setArtistaPrincipal(ArtistaPrincipal : Artista) : void


Representante
-Nacionalidad : String
-Telefono : String
-CorreoEelectronico : String
-CantantesRepresentados : ArrayList<Artista>
+getNacionalidad() : String
+setNacionalidad(Nacionalidad : String) : void
+getTelefono() : String
+setTelefono(Telefono : String) : void
+getCorreoEelectronico() : String
+setCorreoEelectronico(CorreoEelectronico : String) : void
+getAttribute()
+setAttribute(attribute) : void
+getCantantesRepresentados() : ArrayList<Artista>
+setCantantesRepresentados(CantantesRepresentados : ArrayList<Artista>) : void


Disco
-Identificador : String
-Titulo : String
-FechaLanzamiento : String
-Duracion : int
-Reproducciones : int
-Canciones : ArrayList<Cancion>
-Colaboraciones : ArrayList<Artista>
-ArtistaPrincipal : Artista
-Generos : ArrayList<Genero>
+getIdentificador() : String
+setIdentificador(Identificador : String) : void
+getTitulo() : String
+setTitulo(Titulo : String) : void
+getFechaLanzamiento() : String
+setFechaLanzamiento(FechaLanzamiento : String) : void
+getDuracion() : int
+setDuracion(Duracion : int) : void
+getReproducciones() : int
+setReproducciones(Reproducciones : int) : void
+getCanciones() : ArrayList<Cancion>
+setCanciones(Canciones : ArrayList<Cancion>) : void
+getColaboraciones() : ArrayList<Artista>
+setColaboraciones(Colaboraciones : ArrayList<Artista>) : void
+getArtistaPrincipal() : Artista
+setArtistaPrincipal(ArtistaPrincipal : Artista) : void
+getGeneros() : ArrayList<Genero>
+setGeneros(Generos : ArrayList<Genero>) : void


Concierto
-Identificador : String
-Titulo : String
-Fecha : String
-Pais : String
-Ciudad : String
-Duracion : int
-Aforo : int
-ArtistaPrincipal : Artista
-Colaboraciones : ArrayList<Artista>
+getIdentificador() : String
+setIdentificador(Identificador : String) : void
+getTitulo() : String
+setTitulo(Titulo : String) : void
+getFecha() : String
+setFecha(Fecha : String) : void
+getPais() : String
+setPais(Pais : String) : void
+getCiudad() : String
+setCiudad(Ciudad : String) : void
+getDuracion() : int
+setDuracion(Duracion : int) : void
+getAforo() : int
+setAforo(Aforo : int) : void
+getArtistaPrincipal() : Artista
+setArtistaPrincipal(ArtistaPrincipal : Artista) : void
+getColaboraciones() : ArrayList<Artista>
+setColaboraciones(Colaboraciones : ArrayList<Artista>) : void


Premio
-Identificador : String
-Nombre : String
-FechaEntrega : String
-Organizacion : String
-Ganador : Artista
+getIdentificador() : String
+setIdentificador(Identificador : String) : void
+getNombre() : String
+setNombre(Nombre : String) : void
+getFechaEntrega() : String
+setFechaEntrega(FechaEntrega : String) : void
+getOrganizacion() : String
+setOrganizacion(Organizacion : String) : void
+getGanador() : Artista
+setGanador(Ganador : Artista) : void


RedSocial
-Identificador : String
-Nombre : String
-Url : String
-Seguidores : int
+getIdentificador() : String
+setIdentificador(Identificador : String) : void
+getNombre() : String
+setNombre(Nombre : String) : void
+getUrl() : String
+setUrl(Url : String) : void
+getSeguidores() : int
+setSeguidores(Seguidores : int) : void

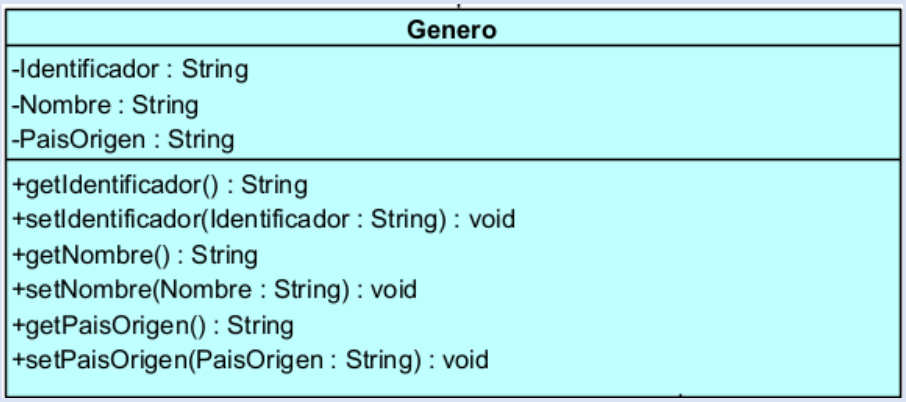
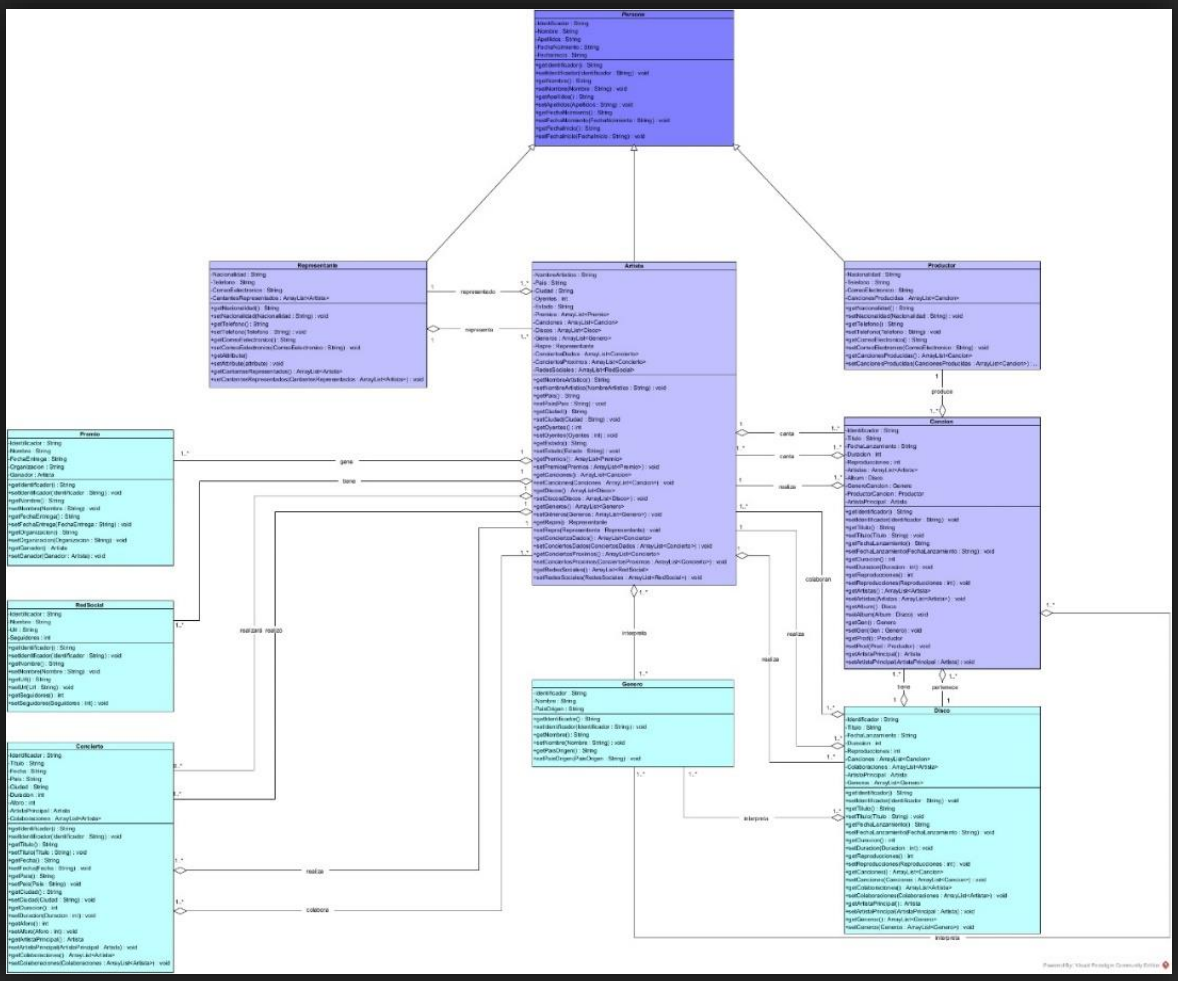

Genero
-Identificador : String
-Nombre : String
-PaisOrigen : String
+getIdentificador() : String
+setIdentificador(Identificador : String) : void
+getNombre() : String
+setNombre(Nombre : String) : void
+getPaisOrigen() : String
+setPaisOrigen(PaisOrigen : String) : void

DIAGRAMA FINAL



ENLACES

DIAGRAMA
https://github.com/Josecp03/Proyecto_1DAM/blob/main/Diagrama%20de%20Clases/UrbanArtists.vpp
IMAGEN DIAGRAMA
https://github.com/Josecp03/Proyecto_1DAM/blob/main/Diagrama%20de%20Clases/Diagrama%20de%20Clases.jpg

5. ACLARACIÓN

Considerando los actuales compromisos y limitaciones de tiempo, he llegado a la conclusión de que no me será posible continuar con el desarrollo del proyecto. A pesar del entusiasmo y la dedicación que he puesto en esta iniciativa, he constatado que el alcance del proyecto es demasiado ambicioso para el tiempo del que dispongo en este momento.