Informe final desafío II

Kevin Esteban Echeverri Carmona, Ivonne Lizeth Rosero Cardona
Facultad de Ingeniería-Universidad de Antioquia
kevin.echeverri@udea.edu.co, ivonne.rosero@udea.edu.co

1. ANÁLISIS DEL PROBLEMA

UdeATunes plantea una simulación de un entorno ampliamente conocido: las plataformas de streaming musical en formato digital. Escuchar música, podcast o audiolibros se ha convertido en una actividad cotidiana para millones de usuarios, por lo que este desafío busca modelar las principales características de dichos sistemas desde el enfoque de la programación orientada a objetos.

El problema propuesto consiste en diseñar e implementar un primer prototipo funcional de una plataforma personalizada que permita gestionar usuarios, artistas, a álbumes, canciones y listas de reproducción, así como las interacciones entre estos elementos. A través de esta implementación, se pretende fortalecer las competencias adquiridas en el cursó, como el usó de clases, relaciones entre objetos, encapsulación n, sobrecarga de operadores y manejó eficiente de estructuras dinámicas de datos.

El sistema diferencia entre tipos de usuarios (estándar y premium), cada uno con beneficios y restricciones específicas. Adema s, se requiere gestionar los archivos multimedia asociados (canciones, portadas, anuncios publicitarios) y medir el consumo de recursos durante la ejecución de cada funcionalidad, lo que permite evaluar la eficiencia del diseño propuesto.

El sistema propuesto debe contemplar múltiples características que reflejan las dinámicas reales de una plataforma de streaming musical. Entre los requerimientos más relevantes se encuentra la diferenciación entre tipos de usuarios, destacando los beneficios exclusivos de la categoría Premium, como la posibilidad de crear una lista personalizada de canciones favoritas.

Estas listas pueden ser compartidas o seguidas por otros usuarios Premium, permitiendo así la sincronización continua entre perfiles. Adema s, las listas de reproducción pueden ejecutarse en modo secuencial o aleatorio, con la opción de retroceder hasta seis canciones previas, manteniendo un registro ordenado del historial de reproducción.

Por otro lado, los usuarios estándar están sujetos a la inserción de mensajes publicitarios cada dos canciones reproducidas. Dichos anuncios se clasifican en categorías C, B y AAA, que determinan su nivel de prioridad o frecuencia de aparición. Se establece, además, la restricción de que ningún anuncio puede repetirse de forma consecutiva, la que añade complejidad al manejó aleatorio y a la lógica de priorización del sistema.

Todos estos elementos representan desafíos significativos en el diseño e implementación del prototipo, especialmente en lo referente a la gestión eficiente de datos, la sincronización entre objetos y el control de flujo lógico durante la ejecución de las diferentes funcionalidades.

La solución propuesta se fundamenta en los principios de la programación orientada a objetos (POO) y en una estructura de programación modular, con el fin de lograr un sistema organizado, escalable y de fácil mantenimiento.

Así mismo, se implementaron estructuras de datos dinámicas y arreglos gestionados con memoria dinámica para el almacenamiento y manejó eficiente de la información. Esto permite la lectura y actualización de los datos desde archivos que actúan como una base de datos permanente, garantizando persistencia entre ejecuciones y facilitando la gestión de la información del sistema.

2. DISEÑO DEL SISTEMA

El diseño del sistema propuesto para UdeATunes se basa en los principios de la programación orientada a objetos (POO), con el propósito de representar de forma estructurada y modular las entidades que intervienen en la plataforma de streaming musical.

Para este propósito, se elabora un diagrama UML que ilustra las clases principales, sus atributos, métodos y

las relaciones existentes entre ellas. Este diagrama constituye la base conceptual del sistema, ya que permite comprender como interactúan los diferentes componentes y como se organiza la información dentro del programa.

Adicionalmente, se detalla la función e importancia de cada clase, justificando su inclusión dentro del modelo y la manera en que contribuye al cumplimiento de los requerimientos planteados en el desafío.

2.1. CLASES PRINCIPALES

Fue necesario durante el desarrollo del desafío modificar e incluso añadir algunas clases y métodos a las ya propuestas en el informe preliminar, para obtener una mejor organización y desarrollo del desafío.

- **Usuarios:** Representa a un usuario registrado en la plataforma. Contiene los datos persónales básicos, el tipo de membresía (estándar o premium) y las listas de reproducción asociadas. En caso de ser un usuario Premium, puede tener una lista de canciones favoritas y seguir listas de otros usuarios.
- **Artista:** Contiene la información básica de cada artista y su catálogo de álbumes.
- **Álbumes:** Representa un álbum musical con información general y una colección de canciones.
- **Canción:** Contiene información básica de cada canción y además modela la unidad básica de reproducción
- **Créditos:** Representa los colaboradores de una canción (productores, músicos o compositores).
- **Publicidad:** Administra los mensajes publicitarios y su prioridad.
- **Gestor de publicidad:** Muestra aleatoriamente la publicidad asegurándose que no se repita consecutivamente.
- **Lista de favoritos:** Permite a los usuarios Premium almacenar, editar y ejecutar sus canciones favoritas.
- **Reproductor:** Gestiona la reproducción musical: aleatoria, secuencial, repetición y retroceso. Controla la aparición de anuncios en usuarios estándar.
- **Interfaz:** Se encarga de la interacción con el usuario, así como la carga de datos, inició de sesión, menú de opciones y medición de recursos.

Cabe resaltar que todas las clases se invocan desde un main el cual es el controlador general del programa.

2.2. BASE DE DATOS

Durante el desarrollo del desafío se construyó una base de datos que contiene la información necesaria y de manera permanente, correspondiente a todo y todos los involucrados para el funcionamiento básico de UdeATunes.

Esta base de datos se compone de un sistema sencillo de archivos en formato de texto plano, los cuales se almacenan en distintas carpetas que comparten una misma ruta de origen denominada **"data_set".**

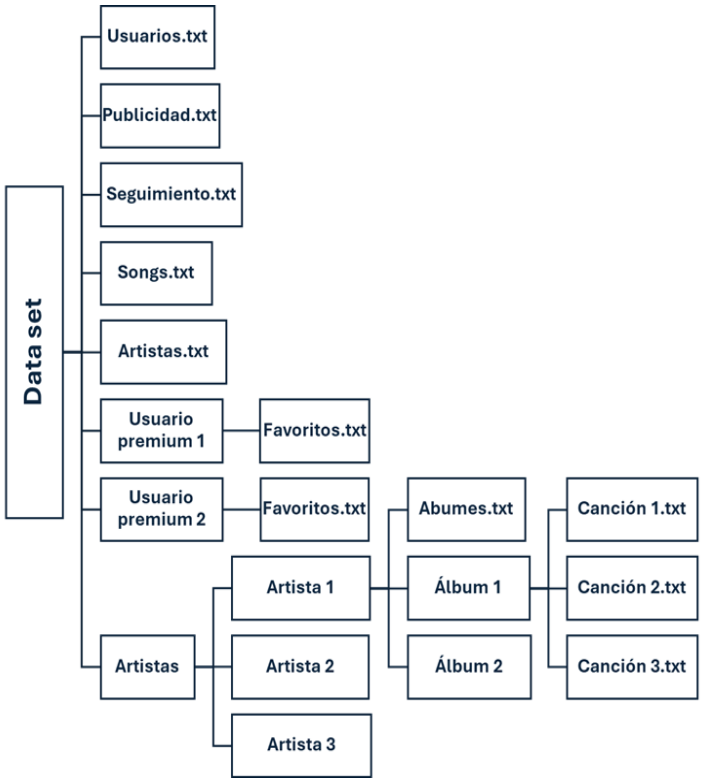

Figura 1. Jerarquía base de datos.

En la figura 1 se puede apreciar la jerarquía que compone la base de datos construida para contener la información de todos los involucrados en el funcionamiento del programa. De esta manera tenemos que:

- En el caso usuarios.txt contiene: nickname, tipo de membresía (1 para usuario con membresía y 0 para usuario estándar), ciudad, país y fecha de registro, esto para todos los usuarios registrados en

la plataforma, cabe resaltar que en esta versión 1.0 de la plataforma no es posible el registro de nuevos usuarios.

- El archivo publicidad.txt contiene todo el banco de anuncios junto con su categoría.

- El archivo seguimiento.txt contiene información de los usuarios que se siguen entre sí, de la forma seguidor|seguido.

- El archivo songs.txt contiene los "id" de todas las canciones disponibles en la plataforma.

- El archivo artistas.txt contiene id, nombre, edad, país, seguidores, posición.

- En la carpeta de cada usuario premium se encuentra el archivo favoritos.txt que contiene una lista de los "id" de sus canciones favoritas.

- En la carpeta artistas está contenida una carpeta por cada uno de los artistas disponibles.

- En cada una de las carpetas de artistas se encuentra un archivo albumes.txt que contiene información del álbum (id, genero, duración del álbum, nombre, fecha de lanzamiento, sello disquero, posición y puntuación) además hay una carpeta por cada uno de los álbumes que pertenecen a ese artista.

- En cada una de las carpetas de los álbumes se encuentran los archivos texto, uno por cada canción perteneciente a ese álbum y el cual contiene información básica de la canción como (id, nombre, duración y créditos.

De esta manera se tiene una información organizada sin tener ningún problema al momento de reproducir de manera aleatoria.

2.3. RELACIONES ENTRE CLASES

Las relaciones entre las clases se definieron de la siguiente manera.

- La clase canción tiene una relación de composición con la clase créditos ya que no existirían unos créditos si previamente no se ha creado un objeto denominado canción.

- La clase gestor tiene una relación de agregación con publicidad ya que se diseña un arreglo dinámico con el total de anuncios publicitarios, pero no se descarta tener un único anuncio, para este caso no es necesario un gestor.

- La clase reproductor actúa como controlador dentro del programa empleando las clases artista, álbumes y canción con una relación de asociación con el fin de buscar y crear instancias temporalmente a partir de un id de canción.

- La clase de interfaz tiene una relación de agregación con la clase reproductor ya que sin el reproductor no se instancia ninguna interfaz y también tiene relación de dependencia con las clases artista, álbumes, canción y publicidad.

Es necesario aclarar que se implementaron estas relaciones ya que son necesarias para el tipo de estructura sobre la que se trabajó.

2.4. DIAGRAMA DE CLASES UML DEL SISTEMA

El diseño del sistema UdeATunes se representa mediante el siguiente diagrama de clases UML, el cual refleja la estructura lógica y las relaciones entre los distintos componentes que conforman la plataforma. Este modelo busca capturar la esencia del funcionamiento de un servició de streaming musical, empleando los principios de la programación orientada a objetos sin recurrir al usó de herencia, privilegiando en su lugar la composición y la agregación para representar las dependencias entre entidades.

Cada clase fue diseñada con una responsabilidad específica: desde la gestión de usuarios, artistas, álbumes y canciones, hasta el control de reproducción y la administración de publicidad. Esta organización modular permite un manejo eficiente de los datos, facilita la escalabilidad del sistema y asegura un comportamiento coherente con los requerimientos funciónales planteados en el desafío.

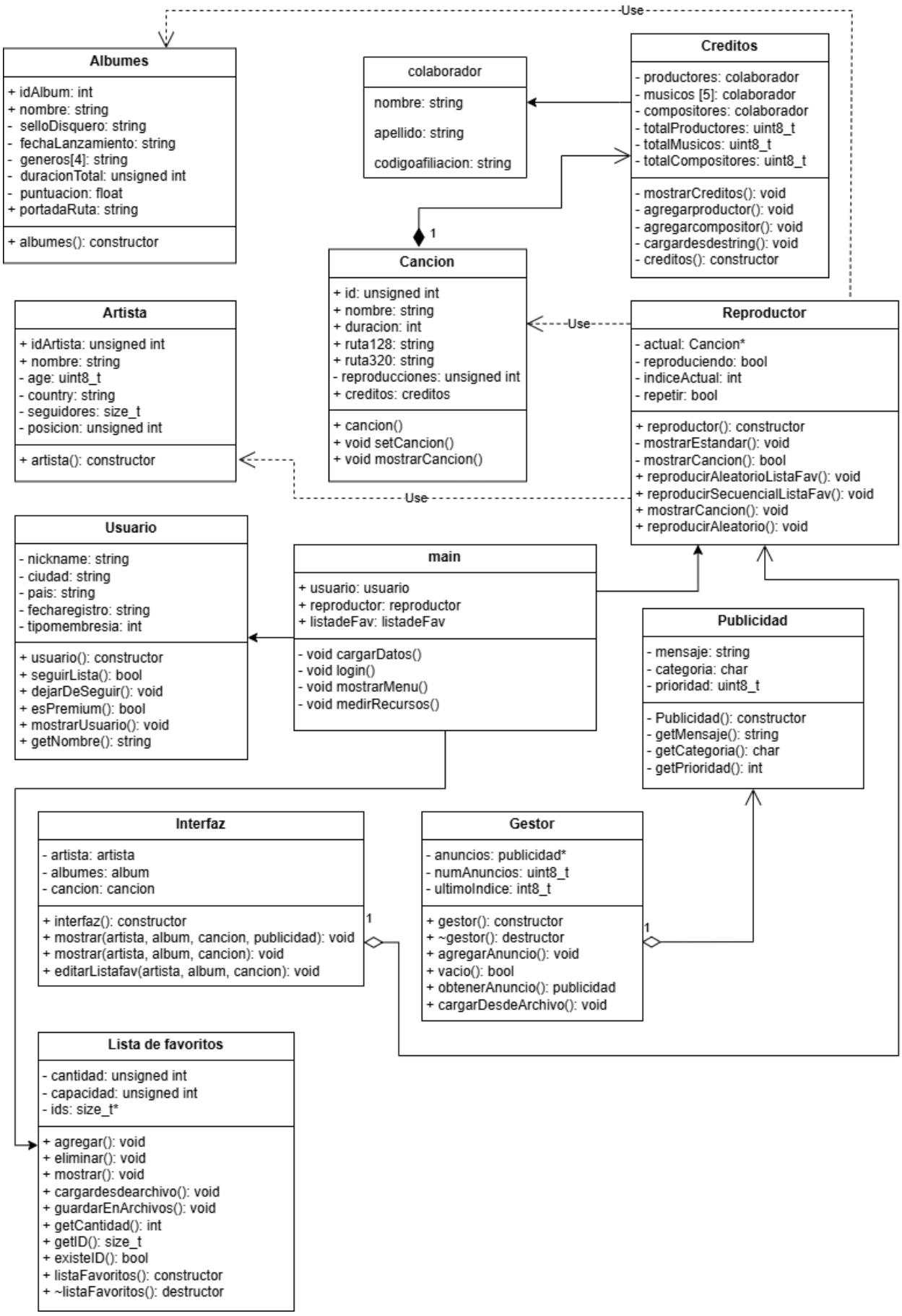

Figura 2. Diagrama de clases UML

En la figura 2 se aprecia el diagrama de clases final implementado en el desarrollo del desafío, donde se muestran las respectivas relaciones entre clases necesarios para el funcionamiento del programa.

3. METODOLOGIA Y DESARROLLO

El desarrollo del sistema UdeATunes se llevó a cabo bajo los principios fundamentales de la programación orientada a objetos (POO), empleando una metodología iterativa e incremental. Cada componente del sistema fue diseñado, implementado y verificado de manera independiente antes de su integración en el entorno general, garantizando así la coherencia funcional y la estabilidad del software.

Durante el proceso de implementación, se priorizaron aspectos como la claridad del código fuente, la modularidad de las clases y el uso eficiente de los recursos de memoria. Para este último aspecto, se realizó una selección cuidadosa de los tipos de datos ofrecidos por el lenguaje C++, utilizando de manera adecuada variables de tipo uint8_t, unsigned int, char, int y string, con el fin de optimizar el almacenamiento y el rendimiento del sistema.

Asimismo, se adoptó el uso de archivos de texto plano como mecanismo de persistencia de datos, lo que permitió mantener una estructura ligera y fácilmente manipulable sin la necesidad de incorporar sistemas gestores de bases de datos externos.

En cuanto al control de versiones y la colaboración entre los integrantes del equipo, se empleó la plataforma GitHub, implementando una estructura de trabajo basada en ramas independientes. Cada integrante contó con una rama propia para el desarrollo de sus componentes, además de una rama principal (*main*) destinada a la integración y validación final del código. Este enfoque favoreció la organización, el control de cambios y la trazabilidad del proyecto durante todo el proceso de desarrollo.

4. RESULTADOS Y FUNCIONAMIENTO

El sistema permite gestionar usuarios, artistas, álbumes, canciones y publicidad de manera integrada. Los usuarios estándar visualizan anuncios según una lógica de prioridad, mientras que los usuarios premium acceden a funciones adicionales como la creación y gestión de listas de favoritos.

La base de datos en texto plano demostró ser una solución práctica para esta primera versión del sistema, garantizando una lectura rápida y una estructura clara para el almacenamiento de la información.

Asimismo, se logró desarrollar un aplicativo funcional basado en interacción por terminal, que permite a los usuarios registrados ejecutar las diferentes acciones correspondientes a su tipo de membresía. A través de esta interfaz, el sistema ilustra de manera estructurada las operaciones disponibles para cada usuario, posibilitando la reproducción y simulación de archivos musicales, así como la ejecución de las funcionalidades asociadas a la gestión de contenidos y publicidad.

En las siguientes figuras se muestran ejemplos del entorno de interacción del sistema, donde se observa el comportamiento de la aplicación desde la terminal, así como la forma en que los usuarios pueden acceder a las distintas opciones y ejecutar las acciones correspondientes a su tipo de membresía.

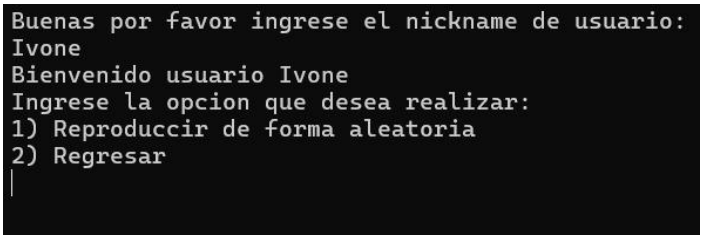

Figura 3. Opciones de interacción para el tipo de usuario estándar.

Figura 4. Visualización del anuncio en la reproducción para el tipo de usuario estándar.

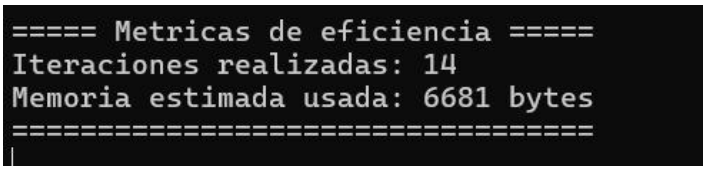

Figura 5. Visualización de las métricas de eficiencia del del programa.

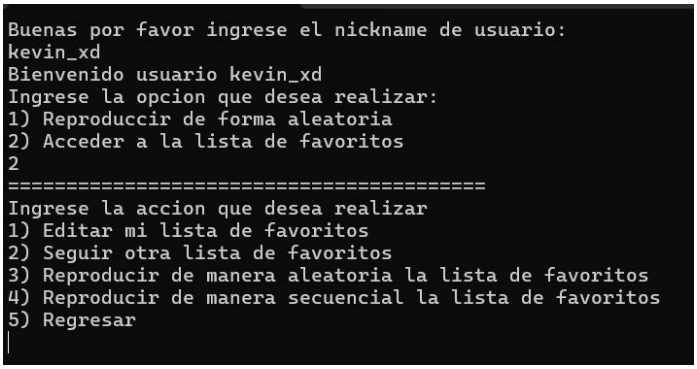

Figura 6. Opciones de interacción para el tipo de usuario premium.

Las imágenes anteriores ilustran el comportamiento operativo del sistema durante su ejecución, evidenciando el flujo de interacción entre el usuario y las distintas funcionalidades implementadas. Asimismo, permiten observar las métricas de eficiencia obtenidas, las cuales reflejan el desempeño y la correcta gestión de los recursos durante la ejecución del programa.

5. CONCLUSIONES

El desarrollo del sistema UdeATunes permitió aplicar de manera integral los conceptos fundamentales de la programación orientada a objetos, tales como la encapsulación, la composición, la agregación y la modularidad. A través de la creación de un entorno que simula una plataforma de streaming musical, se logró representar de manera fiel las relaciones e interacciones entre los distintos componentes del sistema.

La implementación de una base de datos en formato de texto plano, organizada jerárquicamente en carpetas bajo una ruta común (*data_set*), facilitó la persistencia de la información y permitió un manejo transparente de los datos sin la necesidad de sistemas de gestión de bases de datos externos.

Asimismo, el uso de estructuras dinámicas y archivos interconectados garantizó la correcta lectura, almacenamiento y actualización de la información. Esta solución resultó adecuada para un entorno académico y experimental, demostrando que es posible desarrollar sistemas funcionales y escalables utilizando herramientas simples, pero bien estructuradas.

Finalmente, el proyecto evidenció la importancia de una buena planificación y diseño en el desarrollo de software orientado a objetos, así como la necesidad de mantener una arquitectura modular que facilite futuras expansiones, tales como la incorporación de nuevas funcionalidades o la migración hacia una base de datos relacional o no relacional más avanzada.