Universidad de Antioquia
Departamento de Electrónica y Telecomunicaciones
2598521 - Informatica II

Desafio 2 - UdeATunes

Informe de Análisis y Diseño Inicial

Rigoberto Berrio Berrio
Estudiante de Ingeniería Electrónica
Universidad de Antioquia
rigoberto.berrio1@udea.edu.co

Jeisson Stevens Martinez Arevalo
Estudiante de Ingeniería en Telecomunicaciones
Universidad de Antioquia
jeisson.martinez1@udea.edu.co

15 de octubre de 2025

1. Análisis del problema

El sistema **UdeATunes** busca simular una plataforma de streaming musical con funcionalidades diferenciadas según el tipo de usuario: estándar o premium. El objetivo general es desarrollar una aplicación modular que permita reproducir música, gestionar listas de favoritos y medir el consumo de recursos durante la ejecución de cada funcionalidad.

Los principales requerimientos del sistema son:

- **Gestión de datos:** almacenamiento de información sobre usuarios, artistas, álbumes, canciones, listas de reproducción y anuncios publicitarios.
- **Diferenciación de membresía:** los usuarios premium disfrutan de ventajas como reproducción sin anuncios, audio en 320 kbps y creación de listas personalizadas.
- **Reproducción musical:** se permite la reproducción aleatoria, mostrando en pantalla la ruta del archivo `.ogg` y la portada del álbum correspondiente.
- **Publicidad para usuarios estándar:** los anuncios se muestran cada dos canciones, priorizados por categoría (C, B o AAA).
- **Listas de favoritos:** los usuarios premium pueden crear su propia lista (hasta 10 000 canciones) y seguir las listas de otros usuarios premium.
- **Métricas de desempeño:** cada ejecución registra el número de iteraciones y la memoria viva utilizada.

A partir de estos requerimientos, se definió un modelo conceptual que representa las entidades del dominio y sus relaciones funcionales.

2. Modelo conceptual del dominio

El **diagrama de clases** (Figura 1) describe las principales entidades del sistema y su interacción. El diseño separa las **entidades de negocio** (usuarios, artistas, canciones) de los **servicios funcionales** (reproductor, gestor de anuncios y métricas).

2.1. Clases principales

- **Usuario:** almacena información personal y de membresía. Contiene métodos como `isPremium()`, `getKbps()` y `puedeEscucharSinPublicidad()`.
- **Artista, Álbum y Canción:** representan la jerarquía musical. Los álbumes agrupan canciones y cada canción almacena las rutas a sus versiones `ogg128` y `ogg320`.
- **Créditos y Persona:** registran a los colaboradores de una canción (productores, músicos, compositores) con su código SAYCO.
- **ListaReproducción:** estructura que contiene las canciones favoritas de un usuario premium. Implementa los métodos `add()`, `remove()` y `size()`.
- **Reproductor:** controla la reproducción aleatoria, avance, retroceso y repetición.
- **GestorAnuncios y Anuncio:** administran los mensajes publicitarios y su prioridad.
- **Métricas:** mide las iteraciones y memoria consumida para evaluar la eficiencia de cada módulo.

2.2. Relaciones entre clases

Las relaciones modeladas reflejan la estructura jerárquica y las dependencias funcionales:

- **Artista–Álbum:** un artista puede tener varios álbumes.
- **Álbum–Canción:** relación de composición; las canciones no existen fuera del álbum.
- **Canción–Créditos–Persona:** cada canción tiene un bloque de créditos, y cada bloque puede contener múltiples personas.
- **Usuario–ListaReproducción:** un usuario premium tiene una lista propia de favoritos y puede seguir varias listas de otros usuarios.
- **Reproductor–GestorAnuncios–Métricas:** el reproductor integra los servicios de anuncios y métricas según el tipo de usuario.

3. Solución propuesta

El diseño del sistema **UdeATunes** se construyó siguiendo una arquitectura orientada a objetos, buscando mantener una alta cohesión interna en cada módulo y un bajo acoplamiento entre ellos. La solución se fundamenta en la separación clara de las entidades del dominio (usuarios, artistas, álbumes, canciones y créditos) de los servicios funcionales (reproductor, anuncios, métricas y listas de reproducción).

3.1. Estructura general

El sistema se organiza en torno a tres capas conceptuales:

- **Capa de dominio:** agrupa las clases `Usuario`, `Artista`, `Album`, `Cancion`, `Creditos` y `Persona`. Cada una modela un elemento real del ecosistema musical.
- **Capa funcional:** compuesta por las clases `Reproductor`, `GestorAnuncios` y `Metricas`, encargadas de los procesos activos del sistema: reproducción, publicidad y medición de recursos.
- **Capa de soporte:** contiene las estructuras de gestión como `ListaReproduccion` y las relaciones entre usuarios premium para manejo de listas seguidas.

3.2. Módulos principales y responsabilidades

- **Usuario:** Representa la entidad central del sistema. Registra información de cuenta (nickname, país, ciudad, tipo de membresía y fecha de inscripción). Los usuarios *premium* poseen métodos adicionales que les permiten reproducir sin anuncios, acceder a audio en 320 kbps, y crear o seguir múltiples listas de favoritos. Su relación con `ListaReproduccion` es doble: cada usuario tiene una lista propia y puede seguir varias listas de otros usuarios premium.
- **Artista, Álbum y Canción:** Estas clases mantienen relaciones jerárquicas. `Artista` agrupa uno o varios `Album`, y cada `Album` contiene múltiples `Cancion`. Cada canción incluye un método `parseIds()` que interpreta su identificador de nueve dígitos (formato 5–2–2) para determinar a qué artista y álbum pertenece. Además, las canciones guardan las rutas de sus archivos de audio (`ogg128Path`, `ogg320Path`) y la portada correspondiente al álbum.
- **Créditos y Persona:** Modelan el equipo artístico detrás de cada canción. `Creditos` agrupa tres listas: productores, músicos y compositores (de tipo `Persona`), cada uno con nombre, apellidos y un código SAYCO de 10 caracteres. Esta estructura será esencial para futuras versiones del sistema, en las que se calcularán regalías y derechos de autor.

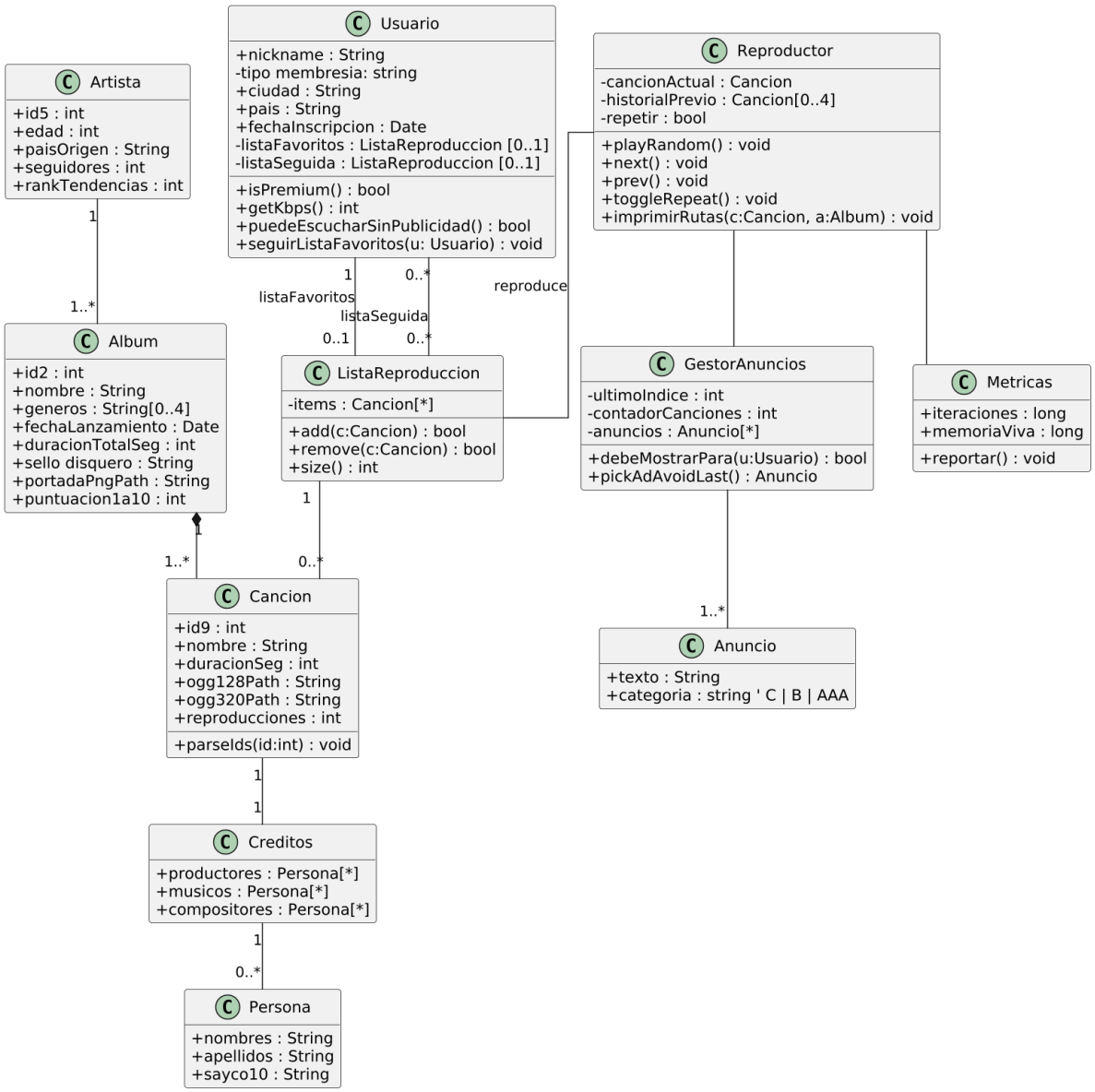

Figura 1: Diagrama de clases del sistema *UdeATunes*.

- **ListaReproduccion:** Permite almacenar y gestionar canciones, garantizando que no existan duplicados. Los usuarios premium pueden editar su lista de favoritos (añadir o quitar canciones), seguir listas ajenas y reproducir su lista completa en orden o de forma aleatoria.
- **Reproductor:** Controla la reproducción musical. Incluye funciones para iniciar reproducción aleatoria, avanzar o retroceder entre canciones, y activar el modo repetición. También registra el historial de hasta cuatro canciones previas y permite imprimir en pantalla la ruta completa de la canción y la portada correspondiente. Este módulo interactúa con `GestorAnuncios` (para los usuarios estándar) y con `Metricas` para registrar eficiencia de ejecución.
- **GestorAnuncios y Anuncio:** Administran los mensajes publicitarios mostrados a los usuarios estándar. Cada anuncio posee un texto (máximo 500 caracteres) y una categoría (C, B o AAA) con diferente prioridad de aparición. El método `pickAdAvoidLast()` garantiza que no se repita el último anuncio mostrado, mientras que `debeMostrarPara()` determina si un anuncio debe mostrarse según el tipo de

usuario y el número de canciones reproducidas.

- **Metricas:** Registra información de desempeño como el número de iteraciones ejecutadas y la memoria viva consumida durante cada funcionalidad. Su método `reportar()` consolida estos datos, permitiendo evaluar la eficiencia del sistema.

3.3. Lógica funcional integrada

El flujo general de funcionamiento se describe así:

1. El usuario inicia sesión y el sistema recupera sus datos y tipo de membresía.
2. Al iniciar la reproducción, el `Reproductor` selecciona canciones aleatoriamente. Si el usuario es estándar, se invoca `GestorAnuncios` cada dos canciones.
3. El sistema muestra en pantalla las rutas de los archivos reproducidos y la portada del álbum.
4. Las métricas de iteraciones y memoria viva se actualizan continuamente.
5. Los usuarios premium pueden editar, seguir o ejecutar sus listas de favoritos con las mismas reglas de reproducción.

3.4. Justificación del diseño

El modelo propuesto ofrece una estructura coherente y extensible:

- Mantiene la independencia entre los módulos de dominio y los de servicio.
- Facilita la implementación futura de nuevos tipos de usuarios o algoritmos de recomendación.
- Permite una evaluación clara de eficiencia, según las métricas solicitadas en el enunciado.

Este diseño servirá de base para las próximas fases del desafío, donde se incluirán las funciones de autenticación, persistencia de datos y menús de interacción.

4. Conclusión

El presente análisis establece la estructura conceptual y las relaciones principales del sistema **UdeATunes**. El diagrama de clases propuesto sienta las bases para una implementación en C++ modular, mantenible y alineada con los requerimientos del proyecto.