Informe Final- UdeATunes

Nombre:

Carlos Mario Rendón Martínez

Michell Londoño Marin

Materia:Informática II

Profesor:
Aníbal Jose Guerra Soler.

Fecha de entrega: 25 de octubre de 2025

INTRODUCCIÓN:

El objetivo del desafío es desarrollar un sistema de streaming musical llamado UdeATunes, en el que se deben gestionar usuarios, canciones, y permitir la reproducción de música de manera eficiente. El sistema debe gestionar las membresías de los usuarios (premium y estándar), ofrecer opciones de reproducción aleatoria, gestionar listas de favoritos, mostrar publicidad para usuarios estándar, y medir el consumo de recursos (memoria y tiempo de ejecución).

Diseño general

Cada clase se implementará de forma independiente, utilizando únicamente composición y punteros para la relación entre objetos. A continuación, se presenta una descripción resumida de cada clase:

- **USUARIO**:
Contiene información como nombre, tipo de membresía, ciudad, país y fecha de registro.

- **PUBLICIDAD:**
Almacena la categoría y el mensaje asociado, con distintos niveles de prioridad.

-**ARTISTA:**
Contiene un código identificador, edad, país, número de seguidores y una lista de álbumes.

-**ÁLBUM:**
Incluye código, nombre, géneros, fecha, duración total y canciones asociadas.

-CANCIÓN:
Posee un identificador único, nombre, duración, ruta del archivo y número de reproducciones.

-COLABORADOR:
Representa a las personas que participan en una canción. Se implementó con atributos como nombre, apellido, código de afiliación y tipo de colaborador.

-SISTEMA:
Administra las colecciones de usuarios, artistas, canciones y publicidades.

JUSTIFICACIÓN TÉCNICA
El desarrollo de este proyecto se basó en los principios de modularidad, encapsulación y eficiencia en memoria.

- Se usaron arreglos dinámicos con punteros en lugar de estructuras más complejas para practicar el manejo de memoria.

- Las clases se diseñaron sin herencia múltiple, priorizando la composición.

- Se evitó el uso de librerías externas para mantener el control sobre la gestión de memoria.

- El código fue escrito con estilo claro, pensando en que cada clase pueda compilarse de forma independiente sin conflictos.

También aplicamos los conceptos vistos en clase de Programación Orientada a Objetos (POO) como:

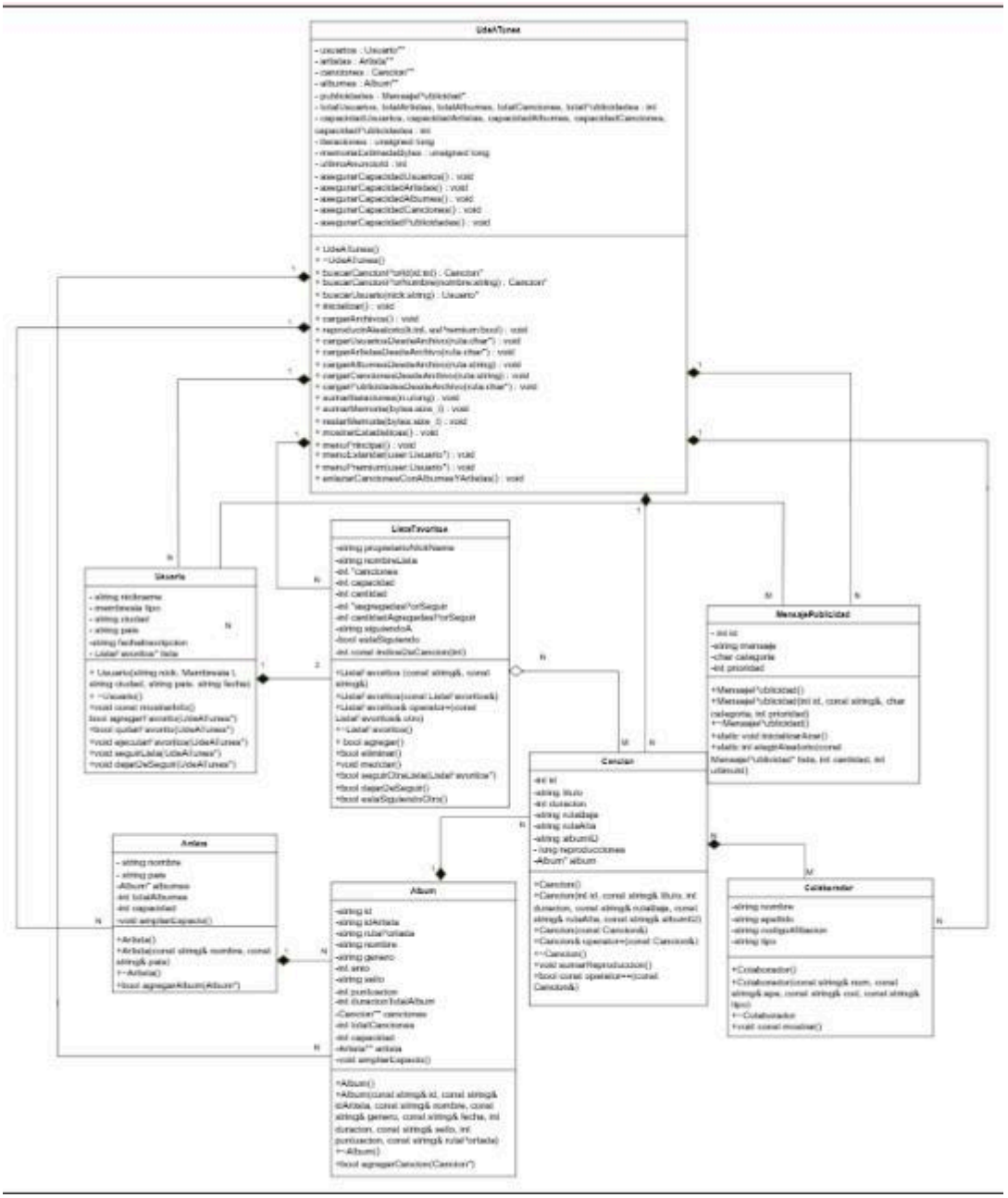
- Constructores y destructores, para controlar la creación y liberación de memoria.

- Getters y setters, para proteger los datos privados.

- Sobrecarga de métodos y operadores, que se planeó para ampliar el comportamiento de algunas clases sin duplicar código.

El diseño propuesto busca mantener la eficiencia, reducir la pérdida de memoria y mostrar el uso correcto de punteros.

DIAGRAMA DE CLASES UML



LÓGICA DE LOS SUBPROGRAMAS PRINCIPALES

-Ingreso a la plataforma: En esta parte se va a encargar de la validación de las credenciales del usuario, esto será por medio de un archivo de datos que nos va a permitir determinar el tipo de membresía que tenga el usuario. Dependiendo de qué membresía será, el sistema va a cargar el menú que le corresponde si es premium o si es estándar.

-Reproducción aleatoria: Con el proceso de la reproducción el mismo sistema se va a encargar de seleccionar las canciones de manera aleatoria mediante un rand(), esto nos va a mostrar en consola la ruta de la canción que se escogió y la portada del álbum. En la parte donde estarán los usuarios de estándar se va a mostrar dos canciones que aparecen con un mensaje publicitario.

-Gestión de favoritos (solo Premium): Cuando el usuario tiene la membresía de Premium, este va a obtener los beneficios de poder permitir agregar o eliminar las canciones de una lista dinámica sin duplicados. Si el usuario sigue en la lista de otro usuario, estos 2 se combinan con la propia.

-Medición de consumo: En esta parte vamos a mostrar cada acción del sistema, es decir, mostraremos cómo contabiliza el número de iteraciones realizadas y también se va a encargar de calcular la memoria utilizada mediante de sizeof(), esto lo vamos a aplicar a los objetos, arreglos y a los punteros.

FORMATO DE ALMACENAMIENTO DE DATOS:

-usuarios.txt: contiene nickname, tipo de membresía, ciudad, país y fecha.
-artistas.txt: código, nombre, edad, país, seguidores y posición.
-albumes.txt: código de álbum, nombre, géneros, duración total y portada.
-canciones.txt: id de canción, nombre, duración, rutas 128/320, reproducciones y créditos.
-publicidad.txt: id, texto, categoría y prioridad.
Cada línea del archivo representa un registro separado, y los datos se cargan en memoria al iniciar el programa.

RESUMEN DE PRUEBAS

Prueba	Descripción	Resultado
Ingreso de usuario	Validación de tipo premium o estándar.	Correcta
Reproducción aleatoria	Reproduce 5 canciones distintas con espera de 3 segundos.	Correcta
Reproducción aleatoria	Se muestra un anuncio cada dos canciones.	Correcta
Publicidad estándar	Se muestra un anuncio cada dos canciones.	Correcta
Lista de favoritos	Agrega y elimina canciones sin duplicar.	Correcta
Medición de recursos	Muestra iteraciones y memoria usada al final de cada ejecución.	Correcta

PROBLEMAS ENFRENTADOS

Los problemas que enfrentamos mi compañera y yo fueron varios, ya que cuanto más avanzamos haciendo este proyecto encontramos muchas dificultades como saber gestionar los punteros y sobre todo el más complejo fue aplicar lo de la memoria dinámica, especialmente cuando creamos los arreglos de los objetos dentro de cada clase compuesta. Bueno, también fue necesario corregir las fugas que se presentaba por medio de la memoria usada, el delete y delete [].

.

EVOLUCIÓN DEL PROYECTO:

Cada día que pasamos haciendo este proyecto encontramos diversos problemas, pero poco a poco investigando y resolviendo las dudas con nuestros profesores de laboratorio pudimos avanzar… Primero completamos las clases principales (Colaborador y Mensaje Publicidad), luego las multimedia (Canción, Álbum y Artista), y después los Usuarios. Luego integramos la clase principal que era UdeAtunes que se encarga del control general del sistema.

CONCLUSIÓN:

Este desafío de UdeATunes nos permitió aplicar todo lo que hemos aprendido hasta el momento sobre Poo, especialmente con el tema de manejo de clases, objetos, punteros y memoria dinámica. Además, gracias a este desafío nos permitió conectar varias clases en un sistema grande sin necesidad de usar herencia, solo usamos la composición y las relaciones entre cada objeto.