Universidad de Antioquia

Facultad de Ingeniería – Ingeniería Electrónica

Informática II

Desafío 1I

Brayan Camilo Silva Porras

Hamilton Alexander Suarez Pérez

2025-2

Informe

El proyecto UdeATunes consiste en el desarrollo de una plataforma de streaming musical que simula el funcionamiento de un sistema real, incluyendo manejo de usuarios, reproducción de canciones, listas de favoritos, medición de métricas y persistencia de datos, el trabajo se implemento en C++ aplicando principios de POO, modularidad y plantillas genéricas. La estructura del sistema se basa en las capas descritas en el pre-informe: presentación, lógica de negocio, servicios y persistencia.

Metodología

El desarrollo del proyecto se realizó de manera incremental, iniciando por la definición de las clases base y el diagrama UML, luego se implementaron las estructuras genéricas y los servicios para finalmente integrar las capas de presentación y lógica de negocio. Durante las pruebas, se verificó el funcionamiento correcto de cada módulo de forma independiente antes de realizar la integración completa del sistema. Esta metodología permitió mantener un código limpio, modular y fácil de depurar.

Arquitectura General

El sistema se organizó en cuatro etapas:

1. **Presentación:** se encarga de la interacción directa con el usuario a través de la consola. Muestra menús, mensajes y controla el flujo del programa principal, se organiza aquí para que el main quede libre por buenas prácticas de organización de código, así el main solo ejecuta la interfaz.
2. **Orquestación:** Contiene las funciones mas importantes del sistema como lo son reproductor que gestiona la reproducción de canciones, controles y la publicidad, y plataforma que coordina la autenticación de usuarios, mantiene la sesión activa y conecta con el resto de módulos.
3. **Servicios:** con la clase medidor se mide el rendimiento del sistema en tiempo de ejecución, registra las iteraciones y el consumo global de memoria ayudando así a evaluar la eficiencia del código.
4. **Persistencia de datos:** Se encarga de leer los archivos de datos que se necesitan para el código, mantenerlos en memoria durante la ejecución y guardar los cambios al finalizar

Además, se implementó una estructura genérica propia que cumple el rol de contenedor dinámico para los diferentes tipos de datos utilizados.

Descripción de las Clases Principales

Interfaz Usuario: Gestiona la interacción con el usuario por medio de menús y opciones de texto, controla el flujo principal del programa y muestra secciones como inicio de sesión, información de la aplicación y salida del sistema, incluye funciones utilitarias para limpiar pantalla, leer opciones y pausar la ejecución.

Plataforma: Actúa como el núcleo que coordina. Se encarga de ejecutar la sesión del usuario, enlazar los módulos de reproducción y gestión de datos. Mantiene la estructura modular del programa y la lógica de autenticación.

Reproductor: contiene la lógica para reproducir todo, incluye la reproducción aleatoria y manual, la reproducción de listas de favoritos y se integra con la clase medidor para contabilizar iteraciones y uso de memoria también simula la publicidad cada dos canciones.

Gestor de datos: Administra la lectura y escritura de archivos del sistema, carga la información de artistas, álbumes y canciones al inicio manteniéndolo disponible en memoria. También se encarga de la persistencia de los datos al finalizar la ejecución.

Arreglo dinámico: Estructura genérica, permite almacenar cualquier tipo de dato con una capacidad inicial de 10 elementos y un factor de crecimiento de 1.5x esta incluye operaciones de inserción, eliminación, modificación, acceso y redimensionamiento dinámico, garantizando una complejidad promedio de O(1) para acceso y agregación.

Funcionalidades Destacadas

Reproducción aleatoria con temporizador: Selecciona 5 canciones al azar y las "reproduce" durante 3 segundos cada una y muestra metadatos como nombre, álbum, ruta del archivo, y portada.

Controles de reproducción manual: Permite al usuario navegar con siguiente, anterior y repetir.

Publicidad dinámica: se muestra cada dos canciones para usuario estándar y simulando un modelo freemium real.

Favoritos: Los usuarios premium pueden guardar y reproducir su lista de favoritos en orden o aleatoriamente.

Medición de rendimiento: Durante la ejecución, el sistema registra las iteraciones y la memoria utilizada, al finalizar se muestran las métricas globales.

Estructuras de datos propias

Arreglo dinámico: implementa un contenedor flexible sin usar vectores, respetando las normas del desafío, en su diseño se incluye constructor de copia, destructor, operadores de asignación y acceso seguro.

Medidor: Usa variables estáticas para acumular datos de todo el sistema y poder calcular el costo aproximado en memoria y tiempo.

Métricas y eficiencia

El uso del medidor permite observar el número de iteraciones ejecutadas y la cantidad de memoria reservada durante la carga y reproducción. Gracias al uso de arreglo dinámico las operaciones de acceso y adición mantienen una baja complejidad en promedio y la redimensión tiene costo amortizado. El código es modular con separaciones claras de responsabilidades y buena reutilización.

Limitaciones del código

- No hay validación de errores.
- La persistencia es básica en archivos .txt.
- El temporizador de 3 segundos es simulado sin reproducción de audio.

Conclusiones

El sistema cumple los objetivos del desafío, se logró la integración coherente de las clases del UML y el código final destacando la modularidad, reutilización y claridad del diseño.

Aunque existen limitaciones técnicas propias del alcance académico, el resultado demuestra la comprensión del paradigma y del trabajo con estructuras genéricas en C++.