Informe Desafio II

Mauricio Aguas, Jharlin Castro

***Resumen*—El presente informe documenta el desarrollo de UdeATunes, una aplicación de streaming musical desarrollada en C++ que modela las funcionalidades básicas de una plataforma digital de música.**

***Palabras Claves*—Programación Orientada a Objetos, Streaming Musical, Estructuras de Datos Dinámicas, Gestión de Memoria, Clases.**

I. Objetivo General

Desarrollar un sistema de streaming musical que emule el funcionamiento de plataformas como Spotify utilizando los principios de Programación Orientada a Objetos en C++, implementando estructuras de datos dinámicas propias para la gestión eficiente de usuarios, artistas, álbumes, canciones y funcionalidades de reproducción.

II. Objetivos específicos

Diseñar e implementar un sistema de clases que modele las entidades del dominio musical (Usuario, Artista, Album, Cancion, Anuncio) aplicando conceptos de herencia para diferenciar usuarios estándar y premium.
Crear estructuras de datos dinámicas personalizadas (Lista enlazada con plantillas) que permitan el almacenamiento y manipulación eficiente de las colecciones de datos sin utilizar la STL de C++.
Implementar un sistema de reproducción aleatoria con control de flujo diferenciado según el tipo de usuario, incluyendo manejo de publicidad ponderada por categorías para usuarios estándar.
Desarrollar funcionalidades específicas para usuarios premium como gestión de listas de favoritos, seguimiento de otros usuarios y navegación avanzada en la reproducción.

III. Anlisis del problema

El problema planteado requiere modelar un servicio de streaming musical completo con las siguientes características principales:

Gestión de Usuarios Diferenciada: El sistema debe manejar dos tipos de usuarios con comportamientos distintos. Los usuarios estándar (gratuitos) tienen limitaciones como publicidad obligatoria y menor calidad de audio, mientras que los premium pagan $19,900 mensuales por beneficios adicionales como ausencia de publicidad, mayor calidad de audio y funcionalidades exclusivas.

Sistema de Contenido Jerárquico: La estructura Artista → Álbum → Canción requiere un manejo cuidadoso de relaciones, especialmente considerando el sistema de identificación de 9 dígitos donde los primeros 5 dígitos corresponden al artista, los siguientes 2 al álbum y los últimos 2 a la canción específica.

Publicidad Inteligente: La gestión de anuncios debe implementar un sistema de ponderación (C: 1x, B: 2x, AAA: 3x) sin repetición consecutiva, lo que requiere algoritmos de selección aleatoria ponderada con memoria de estado.

IV. Consideraciones de Diseño

Estructuras de Datos Propias: Se implementó una clase Lista<T> genérica usando plantillas que proporciona las operaciones básicas requeridas (inserción, eliminación, búsqueda) sin depender de contenedores STL.

Herencia y Polimorfismo: La diferenciación entre usuarios estándar y premium se abordó mediante herencia, donde UsuarioEstandar y UsuarioPremium heredan de una clase base Usuario. Esto permite el uso de polimorfismo para manejar comportamientos específicos como la calidad de audio y la gestión de favoritos.

Gestión de Memoria: El sistema utiliza punteros inteligentes y gestión manual de memoria con new y delete para controlar el ciclo de vida de los objetos y evitar fugas de memoria.

Persistencia de Datos: Se diseñó un formato de archivos de texto estructurado con separadores de punto y coma (;) para cada entidad, permitiendo carga y guardado eficiente de todos los datos del sistema.

V. Problemas de Desarollo

En esta sección siguiendo este formato de texto y tipo de letra debe de escribir su interpretación de los resultados mostrados anteriormente, además debe de colocar las respuestas a las preguntas de la guía.

Gestión de Memoria Dinámica

Problema: Fugas de memoria y referencias colgantes al manejar múltiples listas dinámicas.
Solución: Implementación de destructores apropiados y gestión cuidadosa del ciclo de vida de objetos. Se estableció una política clara de propiedad donde UdeATunes es propietario de las entidades principales y las clases derivadas mantienen solo referencias.

Compatibilidad de Tipos en Plantillas

Problema: Conflictos entre tipos size_t (unsigned) e int (signed) en comparaciones y operaciones aritméticas. **Solución**: Estandarización del uso de size_t para índices y tamaños, con conversiones explícitas cuando es necesario mantener compatibilidad con APIs existentes.

Vinculación de Entidades

Problema: Complejidad en la vinculación de IDs temporales (del archivo) con IDs finales de 9 dígitos durante la inicialización. **Solución**: Implementación de un proceso de vinculación en dos fases: primero carga con IDs temporales, luego generación y asignación de IDs definitivos durante la fase de vinculación.

Persistencia de Estado

Problema: Mantener coherencia entre datos en memoria y almacenamiento permanente, especialmente para favoritos y contadores de reproducción. **Solución**: Desarrollo de métodos guardarUsuarios() y guardarCanciones() que preservan el estado actual del sistema en formatos estructurados.

Interfaz de Usuario Consistente

Problema: Desalineación visual en interfaces de texto y manejo inconsistente de entrada del usuario. **Solución**: Creación de la clase InterfazVisual con métodos estáticos para interfaces consistentes y MenuInteractivo para gestión unificada de flujos de usuario.

VI. Evolucion de la solucion

Fase 1: Análisis y Diseño (Octubre 17-19)

El diseño inicial se centró en identificar las entidades principales y sus relaciones. Se desarrolló el diagrama UML que establece la arquitectura del sistema, definiendo claramente la jerarquía de usuarios y las responsabilidades de cada clase.

Decisiones Clave:

- Uso de herencia para diferenciación de usuarios
- Implementación de Lista<T> genérica con plantillas
- Sistema de IDs jerárquicos de 9 dígitos
- Separación clara entre lógica de negocio e interfaz
- **Fase 2: Implementación Core (Octubre 20-24)**

Se implementaron las funcionalidades básicas del sistema, comenzando con las estructuras de datos fundamentales y progresando hacia las funcionalidades específicas.

Iteraciones Principales:

1. **Estructuras Básicas**: Lista<T>, clases de entidad (Usuario, Cancion, Album, Artista)
2. **Carga de Datos**: Parseo de archivos de texto con manejo de errores
3. **Sistema de Usuarios**: Login, diferenciación por tipo, gestión de sesiones
4. **Reproducción Básica**: Selección aleatoria, interfaz de reproducción

- **Fase 3: Funcionalidades Avanzadas (Octubre 25-27)**

Integración de funcionalidades específicas para cada tipo de usuario y optimización del sistema.

Desarrollos Específicos:

- Sistema de publicidad ponderada con memoria de estado
- Gestión completa de favoritos para usuarios premium
- Navegación hacia atrás con historial limitado
- Interfaces visuales mejoradas con formato consistente
- Sistema de créditos para cálculo de regalías

Consideraciones de Eficiencia Implementadas:

Búsqueda Optimizada: Implementación de buscarPorId() en Lista<T> para búsquedas O(n) con terminación temprana.

Gestión de Memoria: Uso de punteros para evitar copias innecesarias de objetos grandes, especialmente en operaciones de vinculación.

Reutilización de Objetos: Las instancias de canciones, artistas y álbumes se crean una vez y se referencian mediante punteros en todas las operaciones posteriores.

- **Validaciones y Casos Límite**
- **Validación de Entrada**: Manejo robusto de entrada de usuario con limpieza de buffer
- **Listas Vacías**: Verificación de contenido antes de operaciones de acceso
- **IDs Inexistentes**: Manejo graceful de búsquedas fallidas
- **Memoria Insuficiente**: Verificación de asignación exitosa de memoria dinámica
- **Extensibilidad Futura**

El diseño modular permite extensiones como:

- Nuevos tipos de usuario (Artista, Administrador)
- Géneros musicales adicionales

Conclusiones

El desarrollo del proyecto consolidó conocimientos sobre gestión de memoria dinámica, uso efectivo de plantillas y implementación de polimorfismo. La experiencia de modelar un sistema completo desde el análisis hasta la implementación proporcionó perspectiva práctica sobre el diseño de software orientado a objetos.