# **Informe de Desarrollo – Proyecto UdeATunes (Desafío II)**

**Brandon Manuelle Morales Cardona**

**Juan Camilo Ramirez Lozano**

Este informe describe de forma general cómo desarrollamos el proyecto UdeATunes, correspondiente al Desafío II del curso de Informática II. El objetivo fue crear una simulación de una plataforma de streaming musical utilizando Programación Orientada a Objetos (POO) en C++, siguiendo buenas prácticas de diseño y sin depender de la biblioteca STL. Nuestra meta fue modelar un sistema que refleje el funcionamiento real de un servicio como Spotify, pero adaptado a las especificaciones del desafío y totalmente ejecutable por consola.

## Análisis del problema

El problema propuesto consistía en diseñar un programa que representara de forma completa un servicio de streaming musical. El sistema debía manejar usuarios, artistas, álbumes, canciones y mensajes publicitarios, distinguiendo entre usuarios estándar (con publicidad) y usuarios premium (sin anuncios y con beneficios adicionales). También se solicitaba medir la eficiencia del sistema a través del consumo de memoria y el número de iteraciones realizadas. Finalmente, todo debía manejarse mediante una interfaz de texto clara y funcional.

Desde el principio, identificamos que el proyecto requería una estructura modular, bien organizada y con una fuerte aplicación de los principios de POO: encapsulación, abstracción, relaciones entre clases y sobrecarga de operadores. Con esto en mente, diseñamos un conjunto de clases que interactúan entre sí para representar las entidades y procesos del sistema.

## Diseño de la solución

El diseño del sistema gira alrededor de la clase principal UdeATunes, que actúa como el núcleo del programa. Desde allí se controlan las operaciones principales como el inicio de sesión, la reproducción de canciones, la gestión de listas de favoritos y la administración de la publicidad. A continuación, se resumen las clases más importantes y su papel dentro del sistema:

• Artista: contiene la información básica de cada artista, como su código, edad y país de origen.  
• Álbum: agrupa canciones y contiene información como nombre, fecha de lanzamiento, duración y portada.  
• Canción: almacena el identificador, nombre, duración, rutas de los archivos de audio y el número de reproducciones.  
• Usuario: representa a los usuarios del sistema e incluye su nickname, tipo de membresía, ciudad, país y fecha de registro.  
• ListaFavoritos: permite a los usuarios premium guardar y seguir listas de canciones favoritas.  
• MensajePublicitario: administra los anuncios que se muestran a los usuarios estándar según una prioridad asignada.  
• UdeATunes: gestiona toda la lógica del sistema, incluyendo carga de datos, reproducción aleatoria, favoritos y métricas.

Decidimos implementar nuestras propias estructuras de datos en el archivo 'estructuras.h'. Diseñamos una lista enlazada genérica y a partir de ella derivamos estructuras como pilas y colas. Esto nos permitió trabajar con memoria dinámica de manera controlada y comprender mejor el funcionamiento interno de estas estructuras.

## Lógica de implementación

El programa principal se encuentra en 'main.cpp', desde donde se inicializa la clase UdeATunes y se muestran los menús de interacción. El flujo de ejecución del sistema es el siguiente:

1. Al iniciar, el sistema carga automáticamente los datos desde los archivos de texto ('usuarios.txt', 'artistas.txt', 'albumes.txt', 'canciones.txt' y 'publicidad.txt').  
2. El usuario puede iniciar sesión con su nickname, y dependiendo de si es estándar o premium, accede a diferentes opciones.  
3. Se ofrece la posibilidad de reproducir canciones de forma aleatoria. Los usuarios estándar verán mensajes publicitarios cada dos canciones, mientras que los premium escuchan sin interrupciones.  
4. Los usuarios premium también pueden crear y gestionar su lista de favoritos, seguir la lista de otro usuario y reproducir sus canciones favoritas en orden o aleatoriamente.  
5. Al finalizar cada acción importante, se muestran métricas sobre el rendimiento del sistema.

## Medición del consumo de recursos

Uno de los requerimientos más interesantes del desafío fue medir la eficiencia del programa. Inicialmente pensamos en hacerlo con un módulo aparte (utils\_mem), pero finalmente integramos toda la medición dentro de la clase UdeATunes para mantener una estructura más coherente. El sistema utiliza un contador de iteraciones ('iteracionesTotales') y una función ('calcularMemoria()') que estima la memoria ocupada por todos los objetos en ejecución. Cada vez que se completa una operación importante, se muestran en consola los valores actualizados de memoria e iteraciones. De esta forma, el usuario y el programador pueden observar el rendimiento del sistema en tiempo real.

## Problemas enfrentados y soluciones

Durante el desarrollo enfrentamos varios retos técnicos y de diseño. Uno de los primeros fue implementar estructuras de datos dinámicas sin usar la STL, lo cual nos obligó a construir nuestras propias versiones de listas enlazadas, pilas y colas. Otro desafío fue manejar correctamente la reproducción aleatoria y el modo de repetición, lo que resolvimos utilizando la biblioteca <random> y banderas de control internas.

También tuvimos que asegurarnos de evitar fugas de memoria, especialmente al trabajar con punteros y memoria dinámica. Para eso implementamos destructores adecuados en las clases que administran estructuras enlazadas. Por último, quisimos que el código fuera legible y modular, por lo que dividimos las responsabilidades entre archivos y clases especializadas.

## Evolución de la solución y conclusiones

El desarrollo de UdeATunes fue un proceso gradual en el que fuimos integrando cada módulo paso a paso. Primero estructuramos las clases base, luego las relaciones entre ellas, y finalmente las funcionalidades avanzadas como las listas de favoritos y la medición de métricas. Esto nos permitió depurar el código más fácilmente y mantener un diseño limpio.

En su versión final, el sistema cumple con todos los objetivos del desafío: aplica los principios de la POO, usa memoria dinámica sin STL, mide su eficiencia y reproduce canciones con comportamientos diferenciados según el tipo de usuario. En el futuro, nos gustaría ampliar el proyecto agregando persistencia de datos (guardar los cambios de favoritos) y mejorar la experiencia de usuario con una interfaz más visual.

En general, consideramos que UdeATunes fue una excelente oportunidad para poner en práctica los conocimientos adquiridos durante el curso, y nos permitió entender mejor cómo se construyen y organizan sistemas complejos en C++.