Análisis y Diseño de la Solución para el Desafío II

Cristian Bravo
Jose manuel giraldo

Contexto

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).

Análisis del Problema:

Gestión de Usuarios:

Los usuarios deben poder ser de dos tipos: **premium** o **estándar**.

Los **usuarios premium** tienen acceso a funcionalidades exclusivas como la creación de listas personalizadas de canciones favoritas, mejor calidad de audio y reproducción sin publicidad.

Los **usuarios estándar** deben ver anuncios después de cada dos canciones y escuchar música en calidad estándar.

Reproducción de Música:

El sistema debe permitir la **reproducción aleatoria** de canciones.

Los **usuarios premium** pueden controlar la reproducción de las canciones (pasar a la siguiente, retroceder, repetir).

Implementar temporizador de 3 segundos entre canciones, y la reproducción debe detenerse después de **5 canciones** para test.

Se debe mostrar la **ruta del archivo de audio** y la **portada del álbum** correspondiente.

Publicidad:

Los **usuarios estándar** deben ver anuncios publicitarios con una prioridad de visualización según la categoría del mensaje (C, B, AAA).

Medición de Recursos:

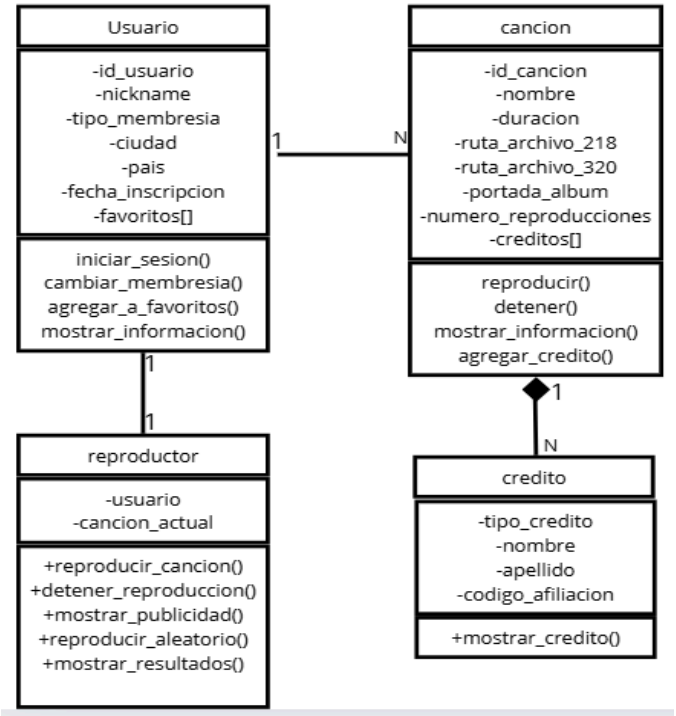
Se deben medir **el número de iteraciones** realizadas (esto incluyendo iteraciones mías, invocaciones y funciones externas) y el **uso de memoria** por parte del sistema.

Interacción con Archivos:

El sistema debe ser capaz de **leer y escribir** datos en archivos de texto, para gestionar los usuarios, canciones y mensajes publicitarios.

Diseño de la Solución

El diseño de la solución se basa en los principios de la **Programación Orientada a Objetos (POO)**. Se utilizarán **clases** para representar las entidades clave (usuarios, canciones, artistas, créditos, reproductores) y sus relaciones.



prototipo de diagrama de clases

Clases y Métodos Principales:

Usuario:

Atributos: **id_usuario**, **nickname**, **tipo_membresia**, **favoritos** (array de canciones), **ciudad**, **pais**, **fecha_inscripcion**.

Métodos:

`iniciar_sesion()`, `cambiar_membresia()`, `agregar_a_favoritos()`, `mostrar_informacion()`.

Cancion:

Atributos: **id_cancion**, **nombre**, **duracion**, **ruta_archivo_128**, **ruta_archivo_320**, **portada_album**, **creditos** (puntero a Credito).

Métodos:

`reproducir()`, `detener()`, `mostrar_informacion()`, `agregar_credito()`.

Credito:

Atributos: **tipo_credito**, **nombre**, **apellido**, **codigo_afiliacion**.

Metodos

`mostrar_credito()`.

Reproductor:

Atributos: **usuario**, **cancion_actual**.

Métodos:

`reproducir_cancion()`, `detener_reproduccion()`, `mostrar_publicidad()`, `reproducir_aleatorio()`.

Medición de Recursos

Para la medición de recursos, implementamos lo siguiente:

Medición de Iteraciones:

Iteraciones internas: Se cuentan cada vez que se invoca una función dentro de un ciclo o un bucle, como en reproducir_aleatorio, donde se realiza un bucle sobre las canciones.

Funciones externas: También se cuentan las invocaciones a funciones externas, como la lectura o escritura de archivos.

Medición del Uso de Memoria:

Memoria total: Se utiliza sizeof para calcular el tamaño de las clases y los punteros, obteniendo una **estimación aproximada** de la memoria utilizada por los objetos.

Memoria local: Calculamos el espacio utilizado por las **variables locales** y el **paso por valor** (cuando se pasan parámetros por valor a una función). El **paso por referencia** (cuando se pasan punteros) no incrementa el uso de memoria.