Imagen que contiene Patrón de fondo

El contenido generado por IA puede ser incorrecto.

**informe**

**Informática II**

**Desafío 2**

**Estudiantes:**

**Jhon Tristancho Muñoz**

**Mateo Arbeláez Cardona**

**Profesor:**

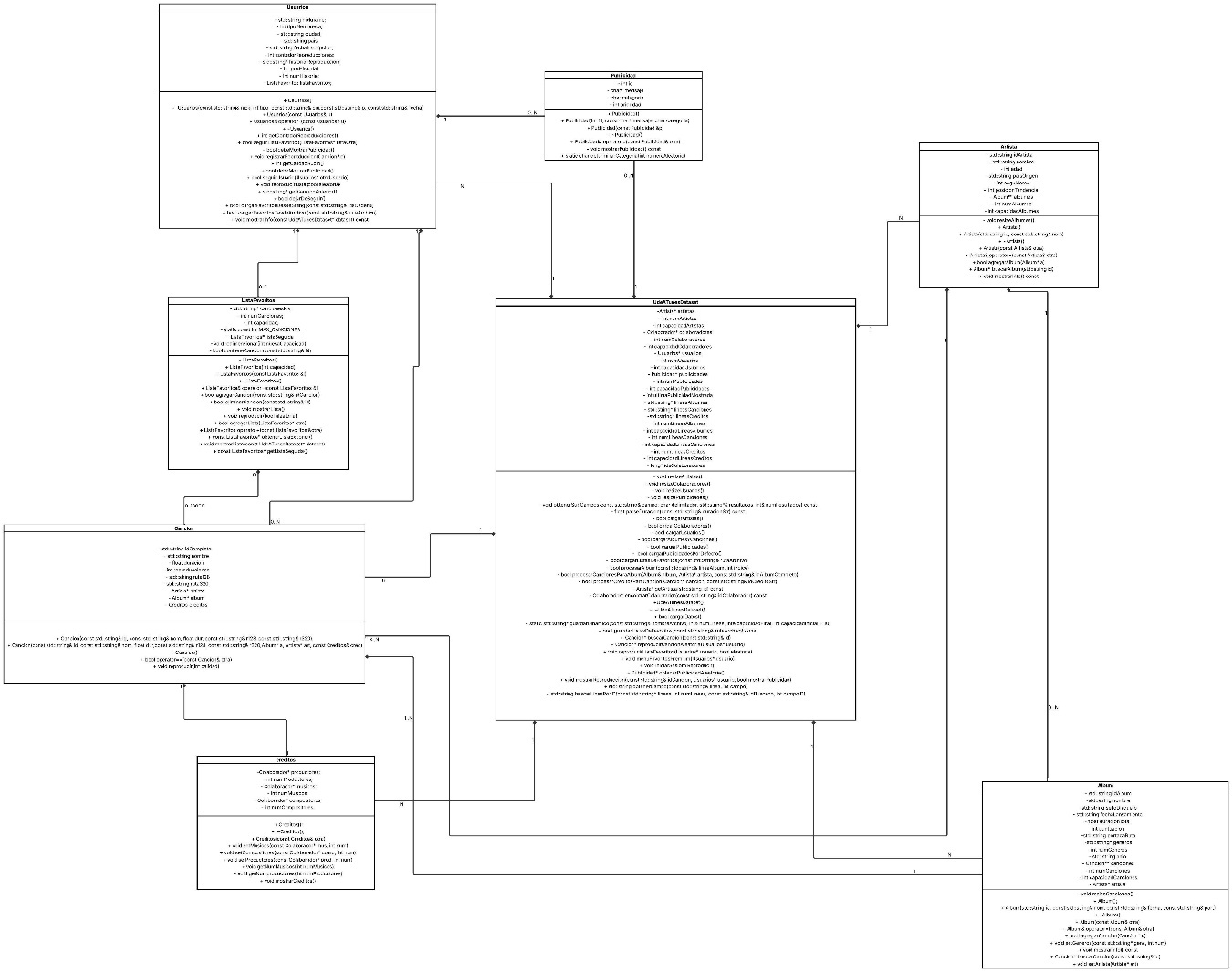
**ANÍBAL JOSÉ GUERRA SOLER**

**Desarrollo del problema:**

nuestro programa consiste en tener un login donde solamente se pondrá usuario y contraseña en el cual a la hora de iniciar sesión nosotros tendremos un archivo .txt con todos los usuarios y un respectivo identificador para saber si es premium o de paga. Para esto sobrecargamos un método para identificar el tipo de usuario.

El usuario estándar podrá buscar una canción y agregarla para escuchar nosotros buscaremos la canción en el .txt de canciones con un algoritmo de busqueda y con respecto a los anuncios vamos a tener dos .txt dependiendo de la frecuencia en que se deben de mostrar y en un random le daremos mas porcentaje de que aparezca o mas prioridad a aquellos que deben de aparecer mas.

El usuario premium podrá agregar canciones a su lista de favoritos y también seguir de manera continua la lista de favoritos de otro usuario. Esto lo lograremos manejando las listas independientemente como arreglos dinámicos separados. Para hacer que tengan la misma prioridad a la hora de aparecer canciones de las dos listas primero se manejara un random de 1 a 2 y dependiendo el que caiga es la lista que se escoge para nuevamente hacer un random de las canciones de la lista. Para mostrar en pantalla imagen de álbum, artista y canción lo haremos teniendo en cada .txt un índice que nos indique donde esta el álbum al cual pertenece la canción y este a su ves un índice de a que artista pertenece. Logrando así que podamos manejar de manera eficiente los datos de las canciones.



Por ahora, tenemos una idea de como dar solucion a los problemas planteados, seria asi:

Vamos a tener unas funciones que nos van ayudar a eso:

Principalmente hablar de la lectura y carga de archivos, leer los archivos de usuarios, artistas, álbumes, canciones y publicidad al inicio, y guardar los cambios al final. Se abren los archivos una sola vez al inicio del programa. contamos el número de registros para dimensionar los arreglos dinámicos, se leen los datos y se crean los objetos correspondientes por ejemplo un new usuario y cerramos y eso nos garantiza tener los archivos hasta el final de la ejecucion.

La estructura de datos que manejamos es la siguiente:

Tenemos los siguientes archivos: publicidad.txt , usuarios.txt, artistas.txt, canciones.txt, colaboradores.txt, listadefavoritos.txt y creditos.txt .

El principal recurso que utilizamos fue le codigo de la cancion ya que al estar seccionado en diferentes partes y que contiene informacion diferente lo utilizamos como herramienta para navegar entre la informacion y identificarla. Para cada txt manejamos un criterio de orden :

Canciones:

<ID\_CANCION>,<NOMBRE>,<DURACION>,<RUTA\_AUDIO\_128>,<RUTA\_AUDIO\_320>,<REPRODUCCIONES>,<ID\_CREDITOS>

Artistas:

<ID\_ARTISTA>,<NOMBRE>,<PAIS>,<GENERO>

Album:

<ID\_ALBUM>,<NOMBRE>,<FECHA>,<RUTA\_PORTADA>

Usuarios:

<NOMBRE\_USUARIO>,<TIPO\_USUARIO>,<CIUDAD>,<PAIS>,<FECHA\_REGISTRO>

Publicidad:

<ID\_PUBLICIDAD>,<MENSAJE>,<PRIORIDAD>

Creditos:

<ID\_CREDITOS>,<ID\_COLABORADORES>….

Colaboradores:

<ID\_COLABORADORES>,<NOMBRE>,<ROL>

Lista de favoritos:

<NOMBRE\_USUARIO>,<ID\_CANCION>….

Para el manejo de datos de estas funciones tenemos 4 metodos muy importantes.el primer metodo se llama guardarDinamico. Este se encarga de hacer que todos los txt bajarlos a arreglos dinamicos y que todas sus lineas queden en un arreglo de punteros a strings. Y este a su vez para hacer un correcto uso de la memoria tiene para cada tipo de txt un metodo de redimensionamiento en caso de que la primera reserva de memoria no sea suficiente.y hace una reserva del doble de memoria.una vez ya listo el arreglo se le entregan los parametros a cada clase para crear los objetos y liberar la memoria. Luego tenemos otro metodo que se encarga de iterar dentro del string con el fin de poder encontrar la columna que buscamos para comparar o para entregar como parametro . lo que hace es sacar la informcaion de un campo especifico a su vez tambien tenemos otra que sirve para acceder a punteros de punteros que fue un caso especifico de aplicación que tuvimos que utilizar.

Tenemos un metodod especifico para cargar los datos de cada txt y asi crear los objetos y entregarle los atributos a cada clase. En estos metodos utilizamos las funciones de redimencionamiento previamente mensionadas y tambien sus respectivas funciones para acceder a las columnas dentro del string.

Para mostrar las publicidades hicimos un random del 0 al 5 para que la ponderacion de cada publicidad sea acorde con el criterio de prioridad pero a su vez sea aleatorio.

Si cae el 0 es del primer tipo,si cae del 1 al 2 es del segundo tipo y si cae del 3 al 5 es del ultimo tipo

buscarCancionPorID(): Busca jerárquicamente en memoria usando la estructura del ID de 9 dígitos. Asi se establece, ya que nos va permitir una organización y una mejor búsqueda. Pensamos algo así:

El sistema recibe el ID, luego se extraen sus componentes, se busca primero el artista que coincida con el ID de los primeros 5 dígitos, dentro de ese artista, se localiza el álbum usando los 2 dígitos intermedios. dentro del álbum, se recorre su lista de canciones y se retorna aquella que coincida con el ID completo. Si no se encuentra en ningún nivel, la función retorna nulo.

seguirUsuario(): Combinar las listas de favoritos entre usuarios mediante operator+. Aunque todavía esta en discusión su implementación, pensamos en poder tener arreglos separados pero tenemos que evaluar diferentes opciones, mirar su gasto en memoria y procesamiento para poder decidir de mejor manera

reproducirLista(): Recorre los IDs almacenados en el arreglo, reproduciendo en orden o aleatorio.

El método recibe un parámetro booleano, si es falso las canciones se recorren secuencialmente. Si es true se generan posiciones aleatorias controladas con un número semilla. Si el usuario es estándar, se muestra publicidad cada cierto número de reproducciones.

retrocederCancion(): Permite regresar hasta 6 canciones en el historial. Pensamos que mediante se retroceda una canción se mantiene un arreglo circular historial[6] dentro del usuario. Cada vez que se reproduce una canción, se guarda en historial[posHistorial]. se incrementa posHistorial hasta llegar a 6 y dar la validación para ya no dejar al usuario retroceder mas.

mostrarPublicidadAleatoria(): Muestra anuncios aleatoriamente cada cierto número de reproducciones. Para los anuncios que tengan una mayor probabilidad de salir estamos pensando en poder una ponderación a los ciertos anuncios que deban salir con mas frecuencia.

Los principales problemas que afrontamos fue que para hacer la búsqueda en los string estábamos utililzando stol para hacer que los numero se nos volvieran long con el fin de comparar para encontrar las canciones.pero esto hacia que se perdieran los 00 de el índice de la canción y no lograba encontrar concidencias por lo tanto comparamos los string directamente.