Imagen que contiene Patrón de fondo

El contenido generado por IA puede ser incorrecto.

**Informe final**

**Informática II**

**Desafío 2**

**Estudiante:**

**Jhon Alexander Tristancho Muñoz**

**Profesor:**

**ANÍBAL JOSÉ GUERRA SOLER**

**Desarrollo del problema:**

El principal objetivo de este desafio era crear un servicio de streaming musical UdeATunes de prototipo 1.0, teniedo presente el criterio de eficiencia para su desarrollo, pero considerando las limitaciones ya que va ser simplemente un prototipo.

Mi programa consiste en tener un login donde solamente se pondrán el nickname del usuario, en el cual a la hora de iniciar sesión tendré 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.

Para evitar el constante abrir y cerrar archivos, yo decidí hacer reservas en memoria en base a un poco de investigacion, como normalmente cuantas canciones tiene un album, como en estadistica cuantos albumnes maneja un artista y cuantos colaboradores en produccion tiene una cancion.

El usuario estándar solo podrá reproducir música de manera aleatoria que cada dos canciones tendrá que ver anuncios, con respecto a los anuncios vamos a tener un .txt que maneje todos los anuncios que se vayan a cargar en el sistema, teniendo en cuenta que pueden tener hasta 500 caracteres y 50 mensajes publicitarios.

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, y a la hora de que un usuario vaya a seguir a otro, resolvi ese problema asignándole un apuntador en ejecución a cada lista de favoritos de los usuarios, y digamos si aparece de esta manera:  
  
Logotipo

El contenido generado por IA puede ser incorrecto.

Usuario Lista de canciones de favoritos Usuario al que sigue

**Esto fue de gran ayuda para la solución de la lista de favoritos.**

Esto me fue de mucha ayuda, ya que al momento de ver si el usuario seguía a alguien, solo fue tomar un puntero a la lista de favoritos del otro usuario, y quedaría juntarla con la de mi usuario para ver si tenia alguna repetida. En los casos donde no hubiera lista seguida, simplemente puntero a nulo.

**La estructura de los txt:**

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

1. Canciones:

Ejemplo:   
000010101 (este primero es el id de la cancion),Miradas (este es el nombre),3:45(este la duracion) ,/users/storage/(sub-ruta X variable)/claudia\_lorelle/audio/miradas\_128.ogg (este la ruta para los estandar) ,/users/storage/(sub-ruta X variable)/claudia\_lorelle/audio/miradas\_320.ogg(este la ruta para los premium) ,245 (este el numero reproducciones),1001 (y este id de creditos)

1. Artistas:

00001 (id del artista) ,Claudia\_Lorelle (nombre de la artista),Colombia (pais de donde es la artista),Pop(y genero que canta)

1. Album:

0000101(id del album),Almas\ rotas(nombre del album),2022(año en el que se creo),/users/storage/(sub-ruta X variable)/claudia\ lorelle/image/almas\ rotas.png

(y ruta de la portada)

1. Usuarios:

user123,1,Medellin,Colombia,2025-01-15

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

1. Publicidad:

1,50% de descuento en tu primera suscripcion premium!,A

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

1. Creditos:

1001,2001,2002,2003,2004

<ID\_CREDITOS>,<ID\_COLABORADORES>….

1. Colaboradores:

2001,Pedro Gomez,Productor

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

**Funcionalidades Implementadas.**

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.

Tenemos un metodo 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 la publicidad use un metodo inteligente según su ponderacion en base a su prioridad, donde las publicidades con mayor prioridad tienen más probabilidades de ser elegidas. Primero calcula un peso total sumando todas las prioridades disponibles, luego genera un número aleatorio dentro de ese rango para determinar qué publicidad seleccionar mediante un sistema de acumulación progresiva, lo que me asegura que muy probablemente las de mayor publicidad salagan, ademas toco implementar una validacion para evitar mostrar la misma publicidad de forma consecutiva, saltando a la siguiente cuando se detecta repetición, garantizando así variedad en la experiencia del usuario mientras respeta la importancia relativa de cada anuncio.

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.

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

El método recibe un parámetro tipo int , si es 0 las canciones se recorren secuencialmente. Si es 1 se generan posiciones aleatorias controladas con un rand() que nos sirve para generar números pseudoaleatorios.

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.

**Dificultades:**

Principalmente el abandono de mi compañero, que aparte de haberme abandonado me dejo bastante desorganizado el proyecto porque trabajo sobre el main en GitHub.

Los principales problemas que afronte fue que para hacer la búsqueda en los string estaba 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.

Se me complico la parte de que un usuario podía seguir la lista de otro usuario, ya que al momento de que la siguiera se me estaban remplazando las canciones originales y ese historial se perdia, pero lo pude solucionar cambiando la lógica y mejoro la eficiencia la cuestión del puntero.

Consideracion:

Principalmente se dejo listo todo lo pedido en el documento, hasta que por ahora no se iba a utilizar e iba a hacer solo para futuro, como los colaboradores, se dejo implementada esa clase.

**Anexo el diagrama de clase por aparte en la documentación para el orden.**