Universidade do Minho
Escola de Engenharia

Relatório Fase 2

Implementação de Plataforma de Streaming de Música

Diogo Moreira, Nuno Mendes, Rafael Silva

Universidade do Minho - Licenciatura em Engenharia Informática

7 de janeiro de 2025

INTRODUÇÃO

Este trabalho prático tem como principal objetivo o desenvolvimento de uma plataforma de streaming de músicas implementada em linguagem C, que realiza a gestão e análise de dados relacionados a músicas, artistas, usuários, álbuns e histórico. O projeto está dividido em duas fases, sendo esta segunda focada precisamente no tratamento desses dados, na resposta a 3 novas queries e na elaboração de um programa interativo.

Neste exercício, foi-nos pedida uma especial atenção para os conceitos de modularização e encapsulamento, competências essenciais para o correto funcionamento do programa. Assim, a estrutura do código foi organizada de forma modular e com atenção ao encapsulamento, separando claramente as interfaces (.h) e implementações (.c) de cada componente.

SISTEMA

Arquitetura Geral

A arquitetura do sistema de streaming de música foi planeada de forma a assegurar a modularidade e a eficiência na manipulação de dados, abordando cada etapa do processamento de maneira organizada e estruturada. Cada funcionalidade do sistema foi isolada em módulos com responsabilidades bem definidas, o que facilita o desenvolvimento e a colaboração entre elementos do grupo. Essa divisão permite que os módulos sejam construídos, testados e mantidos de forma independente, assegurando que melhorias ou correções possam ser feitas sem impactar todo o sistema.

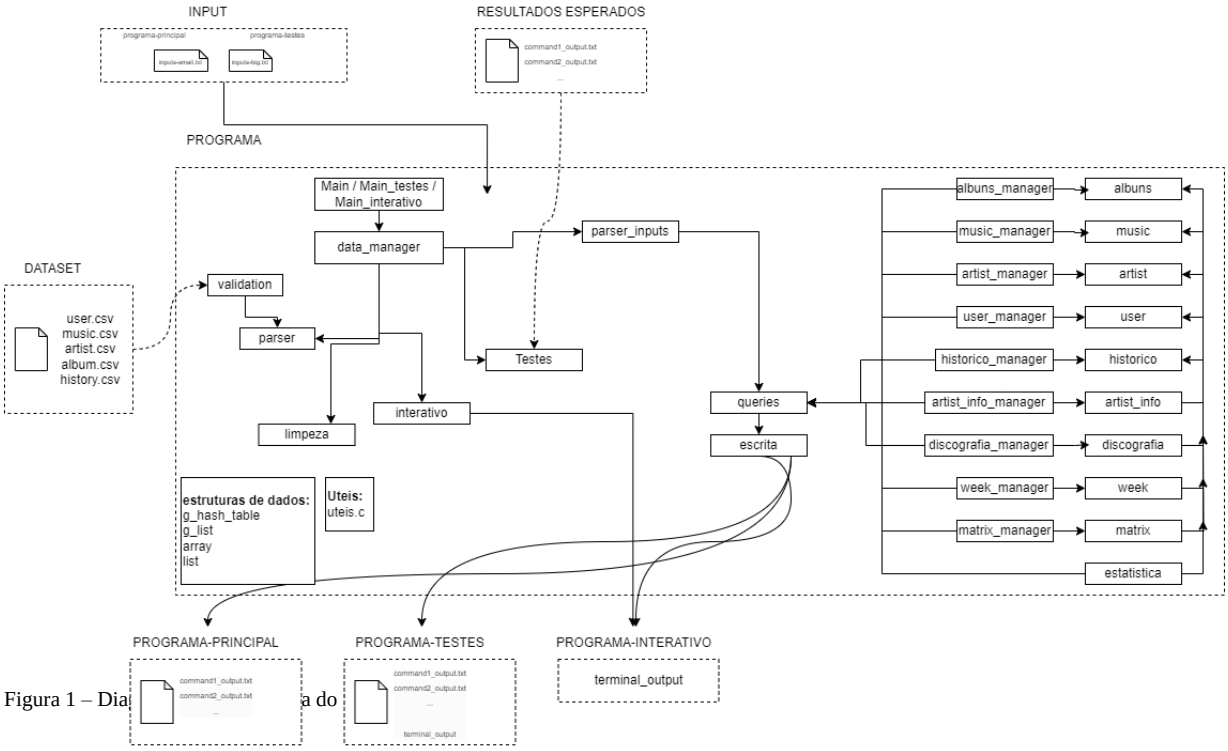

Figura 1 – Dia a do

Módulos Principais

Módulos de Entidade e Manager

A separação das entidades principais — artistas, usuários, músicas, álbuns e histórico — em módulos independentes associados a um módulo manager, permite uma clara distinção de responsabilidades e facilita a

organização dos dados. Cada um desses módulos encapsula as funcionalidades e estruturas relacionadas a uma entidade específica, o que ajuda a evitar dependências desnecessárias e aumenta a coesão do sistema. Essa abordagem permite que o sistema trate os dados de cada entidade de maneira especializada.

Os módulos de entidade definem as estruturas fundamentais e funções mais simples e genéricas associadas aos dados.

Importância dos Módulos Entidade:

1. Servem como a fundação do modelo de dados, descrevendo claramente os atributos disponíveis. É onde são definidas as estruturas e atributos de usuários, artistas etc.
2. Fornecem funções simples e reutilizáveis, como criação e destruição de instâncias de uma entidade (Ex: criação de um álbum, getters básicos).
3. A lógica mais complexa é transferida para o manager, mantendo o módulo base simples e focado na definição das entidades.

Já os módulos manager, em geral, desempenham o papel de intermediar a interação entre as estruturas de dados e as funções de mais alto nível da aplicação. Eles encapsulam a lógica necessária para manipular as estruturas de dados de forma eficiente.

Importância dos Módulos Manager:

1. Gerenciam as operações relacionadas à criação, inicialização, manipulação e limpeza de recursos (hash tables, árvores, listas, etc.)
2. Reduzem a duplicação de código, permitindo que todas as manipulações de uma estrutura de dados sigam as mesmas regras
3. Como a lógica de manipulação está centralizada, erros ou inconsistências são mais fáceis de rastrear e corrigir.
4. O uso de estruturas globais é abstraído pelo manager. Isso evita acessos diretos a essas variáveis em outros módulos, protegendo contra manipulações incorretas.

Módulos de I/O

A criação de módulos específicos para entrada e saída de dados (como os módulos parser, parser_input e escrita) foi motivada pela necessidade de separar as operações de manipulação de arquivos dos demais processos do sistema. Essa divisão permite que a leitura e formatação dos dados aconteçam de forma isolada, garantindo que os dados estejam prontos e no formato correto antes de serem utilizados por outros módulos. Além disso, essa modularidade facilita testes e depuração, já que problemas relacionados à entrada de dados podem ser diagnosticados e resolvidos de forma independente.

Módulo de Utilidade

O módulo de utilidade foi criado para centralizar funções comuns que são usadas em diversos módulos do sistema, evitando duplicação de código e mantendo a consistência. Ao agrupar operações auxiliares e genéricas, como manipulação de strings, criamos uma base de suporte para os demais módulos, que podem acessar essas funções sem precisar redefini-las. Essa divisão facilita a manutenção e torna o código mais limpo, pois qualquer ajuste nessas funções pode ser feito em um único lugar, propagando automaticamente as mudanças para todas as partes do sistema que dependem delas.

Módulos de Gestão do Sistema

Os módulos de gestão foram divididos em vários módulos, sendo o principal o módulo queries. Este módulo é dedicado a processar uma consulta específica, simplificando o desenvolvimento e a manutenção dessas funcionalidades. Essa divisão também possibilita que novos módulos de querie sejam adicionados futuramente, caso surjam novas necessidades.

Com a finalidade de melhorar o tempo de execução e tratamento de dados, e garantir desempenho e clareza, módulos tais como: estatisticas, matrix, discografia, week, etc., foram criados com responsabilidades específicas. Esses módulos são dedicados à criação e manipulação de estruturas de dados otimizadas e à implementação de funções auxiliares necessárias às queries para processar consultas.

Outros Módulos

Além dos módulos de gestão e consulta, outros módulos desempenham papéis cruciais para o funcionamento completo do sistema, garantindo robustez, extensibilidade e facilidade de uso. Entre eles, destacam-se o main principal (o ponto de entrada principal do programa), main testes e seus módulos correspondentes (onde se dá o processo de verificação e validação de funcionalidades do sistema), além do main interativo (oferece uma pequena interface interativa para o usuário, permitindo que ele explore e execute queries), módulos de limpeza (onde se encontram as funções responsáveis por libertar os recursos), e o módulo de validação (onde é feita a validação dos dados, não permitindo dados inválidos de serem armazenados nas estruturas de dados).

DISCUSSÃO

Análise Crítica

Ao desenvolver o sistema, um ponto crucial foi a escolha das estruturas de dados adequadas para as queries.

Q1 - Listar o resumo de um utilizador ou artista, consoante o identificador recebido por argumento.

- Objetivo: A partir de um ID, de artista ou usuário, devolver um resumo de informações, “name;type;country;num_albums_individual;total_recipe”, respetivo ao Artista e “email;first_name;last_name;age;country”, respetivo ao Usuário.

- Estruturas e Resolução: Recurso a hash tables das entidades para obter dados simples (O(1)) e recurso a hash tables auxiliares, em conjunto com Glist (O(N)) (para obter todas as chaves de um certo tipo) para calcular total_recipe e num_albums_individual.

Q2 - Quais são os top N artistas com maior discografia?

- Objetivo: A partir de um número n de utilizadores (com opção de adicionar um filtro), a querie deve devolver os n artistas com maior discografia.
- Estruturas e Resolução: Recurso a hash tables das entidades para obter dados simples (O(1)) e recurso a hash tables auxiliares, em conjunto com Glist (O(N)) para calcular o tempo de discografia.

Q3 - Quais são os géneros de música mais populares numa determinada faixa etária?

- Objetivo: Gerar uma lista com os géneros mais populares (mais likes) numa determinada faixa etária (através de um max e um min).
- Estruturas e Resolução: Tabelas hash, GList e Arrays para calcular géneros mais populares.

Q4 - Qual é o artista que esteve no top 10 mais vezes?

- Objetivo: Identificar os 10 artistas com o maior tempo de reprodução através do histórico (ocasionalmente com um filtro de intervalo de datas a considerar)
- Estruturas e Resolução: Uso de tabelas de hash auxiliares para buscar dados e GList para calcular os tempos.

Q5 - Recomendação de utilizadores com gostos parecidos

- Objetivo: Gerar usuários com gostos de géneros parecidos, através de uma matriz com o número de vezes que cada user ouviu um certo género.
- Estruturas e Resolução: Matriz bidimensional (gêneros x utilizadores) e tabelas hash para mapear IDs a índices na matriz, com recurso à GList para indexação.

Q6 - Resumo anual para um utilizador

- Objetivo: Analisar o resumo temporal de um utilizador num ano específico (ocasionalmente com um parâmetro N – N artistas mais ouvidos pelo utilizador).
- Estruturas e Resolução: Hash tables auxiliares e GLists para cálculos intermédios.

Testes e Resultados

CONCLUSÃO

Concluímos que a implementação modular e o uso de encapsulamento foram essenciais para o sucesso desta fase do projeto, facilitando o desenvolvimento, a colaboração e a manutenção do sistema. Os testes realizados revelaram diferenças significativas no desempenho entre as queries, destacando a necessidade de ajustes para otimizar o uso de memória e tempo de execução, especialmente para a query q3. Futuras Melhorias: Um ponto a ser aprimorado seria a implementação de estruturas de dados próprias e otimizadas para o nosso caso, uma vez que a dependência da GLib apresentou alguns problemas de desempenho. Em suma, o trabalho nesta fase foi um exercício valioso em modularidade e eficiência, e os aprendizados servem como base sólida para as próximas etapas do projeto.