Universidade do Minho

Relatório da Fase 2

Laboratórios de Informática III

Grupo 66

André Filipe Pereira Ribeiro (a104436)

David Silva e Costa (a104615)

Rodrigo Oliveira Fernandes (a104175)

Janeiro 2024

Índice

1-Introdução

O presente trabalho prático, desenvolvido no âmbito da unidade curricular de Laboratórios de Informática III, teve como objetivo principal o desenvolvimento de um sistema de gestão e análise de dados para uma plataforma de streaming de música. Este projeto visou não só a implementação de funcionalidades específicas, mas também o desenvolvimento de competências fundamentais na programação estruturada em C.

O sistema desenvolvido permite processar e analisar dados relativos a músicas, artistas, utilizadores, álbuns e histórico de utilização, oferecendo diversas funcionalidades de consulta através de diferentes modos de execução. A implementação focou-se particularmente na organização modular do código e na aplicação de princípios de encapsulamento, elementos cruciais para garantir a manutenibilidade e escalabilidade do sistema.

Entre os aspectos mais relevantes do nosso trabalho, destacamos:

A implementação de uma arquitetura modular robusta, com clara separação entre a gestão de dados, processamento de queries e interface com o utilizador;

A utilização eficiente de estruturas de dados da biblioteca GLib, nomeadamente hash tables, que permitem um acesso rápido e eficiente aos dados;

O desenvolvimento de um sistema de validação abrangente, tanto a nível sintático como lógico, garantindo a integridade dos dados processados;

A criação de três modos de execução distintos (principal, interativo e testes), proporcionando diferentes formas de interação com o sistema;

A implementação de um conjunto de queries que permitem análises complexas sobre os dados, como a análise de popularidade de géneros musicais por faixa etária e o cálculo de métricas de performance dos artistas.

Além das funcionalidades base, o programa inclui características particularmente interessantes como a capacidade de processar grandes volumes de dados de forma eficiente, a gestão adequada de memória para evitar fugas de memória (memory leaks), e um sistema robusto de tratamento de erros que permite identificar e registar adequadamente entradas inválidas.

Nos capítulos seguintes, apresentaremos em detalhe a arquitetura do sistema, discutiremos as decisões técnicas tomadas e analisaremos o desempenho do programa em diferentes cenários de utilização.

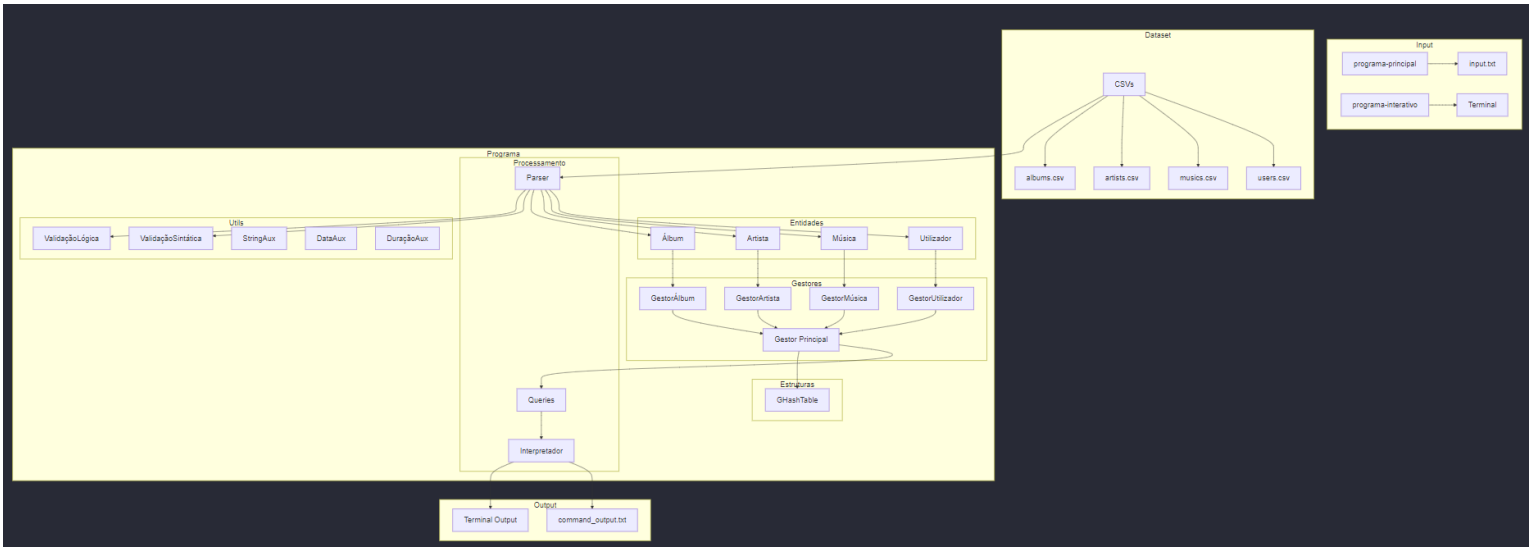
2-Alterações relativas à primeira fase

Na evolução do projeto para a segunda fase, o foco principal centrou-se na melhoria da qualidade geral do código. Foi dada especial atenção ao encapsulamento dos dados, melhorando a organização e a proteção das estruturas internas dos módulos. Adicionalmente, implementámos a gestão de álbuns, integrando esta nova funcionalidade de forma coerente com a arquitetura existente. A documentação do código foi também significativamente melhorada, facilitando a sua manutenção e compreensão. Estas alterações, embora subtis, representam um avanço importante na qualidade e robustez do sistema.

3-Sistema

A arquitetura do nosso sistema foi cuidadosamente planeada para garantir uma separação clara de responsabilidades e um alto nível de modularização. O sistema está organizado em vários módulos fundamentais, cada um com uma função específica e bem definida:

Arquitetura Geral



A arquitetura do sistema divide-se nos seguintes componentes principais:

Módulos de Gestão (Managers)

Gestor Principal: Atua como ponto central de coordenação, agregando e gerindo os gestores específicos para cada tipo de entidade

GestorArtista: Responsável pela gestão dos dados dos artistas

GestorUtilizador: Gere a informação dos utilizadores

GestorMusica: Controla os dados relacionados com músicas

GestorAlbum: Responsável pela gestão dos álbuns

Módulos de Entidades

Artista: Encapsula toda a informação relativa a artistas

Utilizador: Gere os dados dos utilizadores

Musica: Contém a informação das músicas

Album: Mantém os dados dos álbuns

ArtistaResultado: Estrutura auxiliar para processamento de queries específicas

Módulos de Processamento

Parser: Responsável pela leitura e interpretação dos ficheiros CSV

Interpretador: Processa e executa as queries solicitadas

Validação: Divide-se em validação sintática e lógica dos dados

Módulos Auxiliares

Handlers: Gestão de ficheiros e operações de I/O

Utilitários: Funções auxiliares para manipulação de strings, datas e durações

Justificação das Escolhas Arquiteturais

A arquitetura foi desenhada seguindo princípios fundamentais de engenharia de software:

Encapsulamento: Cada módulo encapsula os seus dados e operações, expondo apenas uma interface bem definida através dos ficheiros .h. Isto permite uma clara separação entre a interface e a implementação.

Modularidade: A separação em módulos específicos permite:

- Manutenção mais simples
- Testabilidade independente de cada componente
- Reutilização de código
- Facilidade na implementação de novas funcionalidades

Estruturas de Dados: Utilizamos principalmente hash tables da biblioteca GLib para armazenamento eficiente dos dados, permitindo:

- Acesso rápido aos elementos
- Gestão eficiente de memória

Gestão de Memória: Implementamos um sistema robusto de gestão de memória com:

- Funções específicas para alocação e libertação de recursos
- Utilização consistente de funções de clone para evitar problemas de aliasing
- Gestão adequada de strings e arrays dinâmicos

Esta arquitetura permite não só cumprir os requisitos funcionais do projeto, mas também facilita a manutenção e extensão do sistema, como demonstrado pela fácil integração do módulo de álbuns na segunda fase do projeto.

4-Discussão

Modularização e Encapsulamento

O projeto foi desenvolvido com um forte foco na modularização e encapsulamento, seguindo boas práticas de engenharia de software. Cada entidade do sistema (Utilizador, Música, Artista e Álbum) foi implementada como um módulo independente, com a sua própria interface (.h) e implementação (.c).

O encapsulamento foi garantido através de:

- Estruturas de dados privadas nos ficheiros .c
- Interfaces públicas bem definidas nos ficheiros .h
- Funções de acesso (getters e setters) para manipular os dados
- Gestores específicos para cada tipo de entidade

Estruturas de Dados

A escolha principal para armazenamento de dados foi a GHashTable da biblioteca GLib, justificada por:

- Eficiência nas operações de busca
- Gestão automática de colisões
- Suporte integrado para funções de hash e comparação

Análise de Desempenho

Máquina: *Windows 11, 8GB RAM, AMD Ryzen 7 3750H @ 2300Mhz*

Memória utilizada: 2234 MB

Tempos de execução:

Q1: 0.000065 s (média de 120 queries)

Q2: 0.004849 s (média de 120 queries)

Q3: 0.000004 s (média de 60 queries)

Elapsed time: 276.514455 s

Pontos Fortes

A modularização eficiente foi um dos principais pontos fortes do nosso projeto. A clara separação entre os diferentes componentes do sistema permitiu não só uma melhor organização do código, mas também facilitou significativamente o processo de desenvolvimento colaborativo.

A gestão de memória foi implementada de forma robusta, com especial atenção à prevenção de memory leaks. Cada módulo possui as suas próprias funções de alocação e libertação de memória, e implementámos uma política consistente de clonagem de dados para evitar problemas de aliasing. O uso das funções de gestão de memória da GLib também contribuiu para uma gestão mais segura e eficiente.

A interface do utilizador, tanto no modo principal como no interativo, foi desenvolvida priorizando a clareza e funcionalidade. O modo interativo oferece feedback claro e gestão de erros adequada, enquanto o modo principal processa eficientemente os comandos em batch.

Limitações e Possíveis Melhorias

A generalização de certas partes do código também apresenta oportunidades de melhoria. Por exemplo, as funções de validação poderiam ser reimplementadas usando um sistema mais genérico baseado em regras, permitindo maior flexibilidade na adição de novos tipos de validação sem necessidade de modificar o código existente. Similarmente, o sistema de parsing poderia beneficiar de uma abordagem mais genérica que facilitasse a adição de novos tipos de dados.

Uma última área de potencial melhoria seria a implementação de testes unitários mais abrangentes. Embora tenhamos desenvolvido testes funcionais para as queries, um conjunto mais completo de testes unitários para cada módulo individual aumentaria ainda mais a robustez do sistema.

Conclusão

O desenvolvimento deste trabalho prático permitiu consolidar conhecimentos essenciais em programação estruturada na linguagem C, com especial foco na modularização, encapsulamento e gestão eficiente de memória. A criação de um sistema de streaming de música funcional não só exigiu a implementação de conceitos fundamentais de programação, mas também desafiou-nos a resolver problemas complexos relacionados com manipulação de dados, validação e otimização.

Embora tenhamos conseguido implementar as funcionalidades da primeira fase com sucesso, é importante destacar que, apesar de termos tentado desenvolver as queries adicionais da segunda fase (Q4, Q5, Q6), o tempo disponível revelou-se insuficiente para concluir estas tarefas, visto que somos estudantes do 3.º ano da licenciatura, estávamos sobrecarregados com inúmeros trabalhos e responsabilidades. A complexidade acrescida destas funcionalidades, combinada com a exigência de um planeamento detalhado e execução cuidadosa, tornou inviável a sua implementação dentro do prazo.

Ainda assim, os resultados obtidos destacam-se pela estrutura bem organizada do projeto, uso eficaz de ferramentas como o Valgrind para evitar memory leaks, e a implementação de queries robustas e interativas. No entanto, identificámos algumas áreas que poderiam ser melhoradas, como o refinamento de certas estruturas de dados para maior eficiência e a introdução de funcionalidades adicionais, como um recomendador mais avançado.

Em suma, este projeto foi uma oportunidade valiosa para aplicar conceitos de engenharia de software em um contexto prático, desenvolvendo não apenas competências técnicas, mas também capacidades de trabalho em equipa e gestão de projetos colaborativos.