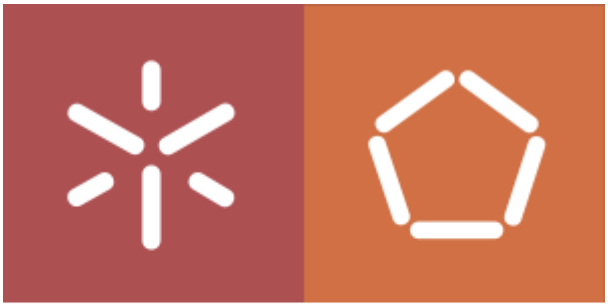


Universidade do Minho
Escola de Engenharia

Laboratórios de Informática III

Fase 2

A104433, Francisco Jorge Salgado de Castro

ciscozin

A107361, Miguel Rocha Diegues

MiguelDiegues24

A99432, Nuno Marco Pacheco da Silva

Kouraan

Índice

Síntese da fase 1

Na Fase 1, desenvolvemos a estrutura base do projeto, implementando conceitos fundamentais de modularidade e encapsulamento. Foram utilizadas diferentes estruturas de dados para representar artistas, músicas e utilizadores, empregando structs específicas para cada tipo de entidade.

No que diz respeito à validação, criámos módulos dedicados para assegurar a integridade dos dados durante o parsing de ficheiros CSV. Os dados eram processados por parsers genéricos e específicos, que chamavam funções de validação nos setters correspondentes.

Para gestão dos dados, implementámos hash tables com auxílio da GLib, garantindo consultas rápidas. Também foram desenvolvidas três queries principais: consulta a dados de utilizadores, identificação de artistas com discografias mais longas e análise de géneros musicais mais populares dentro de intervalos etários.

A Fase 1 revelou desafios na gestão de memória e na eliminação de leaks, mas conseguimos atingir os objetivos estabelecidos, criando uma base sólida para expandir na Fase 2.

Introdução à fase 2

Na Fase 2 do projeto, expandimos significativamente o sistema desenvolvido na Fase 1, incorporando novas funcionalidades e dados para aumentar a complexidade e a versatilidade do programa. Nesta etapa, foram adicionados dois novos módulos principais, álbuns e histórico, que aumentam a riqueza das consultas possíveis e permitem análises mais detalhadas. Além disso, o sistema agora suporta um volume maior de dados e executa queries adicionais, promovendo a robustez e eficiência.

Outro ponto de destaque é a implementação de três executáveis distintos: programa-principal, programa-interativo e programa-testes, cada um com funcionalidades específicas. O programa-principal mantém o foco em processar um conjunto pré-definido de queries a partir de um ficheiro de entrada. O programa-interativo introduz uma interface de linha de comandos para permitir ao utilizador explorar os dados dinamicamente. O programa-testes, por sua vez, avalia a corretude e o desempenho das implementações, comparando os resultados obtidos com os esperados.

Foram introduzidas três novas queries (Q4, Q5 e Q6), além de melhorias nas queries anteriores, como a atualização da Q1 para incluir informações sobre álbuns individuais e receitas totais. A validação dos ficheiros CSV também foi ampliada, com novos critérios sintáticos e lógicos, garantindo a qualidade dos dados processados.

Por fim, a modularização e o encapsulamento foram refinados para melhorar a organização do código e o isolamento entre módulos, seguindo as melhores práticas de desenvolvimento em C. Estas melhorias visam não só a eficiência do sistema, mas também a sua manutenibilidade e clareza. A qualidade do código foi elevada com a introdução de documentação padrão Doxygen, detalhando todas as funções e estruturas.

Modularidade e Encapsulamento

Modularidade

1. Módulo de Estruturas de Dados

Este módulo é responsável por definir as estruturas que organizam e armazenam os dados do sistema. No nosso trabalho temos o artists.h, musics.h, albums.h, users.h e os seus respetivos .c. Cada estrutura é composta por campos específicos como IDs, nomes etc.

2. Módulo de Entidade do Sistema

Este módulo organiza a lógica específica de cada tipo de entidade. Assim sendo fazem parte deste modulo os respetivos managers dos ficheiros abordados acima. Estes módulos gerenciam as hashtables e armazenam os dados de cada entidade. Por exemplo, o artistsmanager processa os artistas a partir das linhas CSV, valida os dados e insere os em uma hashtable.

3. Módulo de Gestão do Sistema

Este módulo centraliza a coordenação e a interação entre diferentes entidades do sistema.

No caso temos o database_manager.h/c, que atua como um gerenciador central, unificando os diferentes módulos de entidades. Possui funções como create_database_manager para inicializar todos os gerenciadores.

4. Módulo de IO

Este módulo lida com a leitura de dados de arquivos, escrita de resultados, e interação com o usuário. Estamos a falar do files.h/c, do parser.h/c e do interactive.h/c. O parser interpreta os dados de arquivos CSV, separando-os por campos e validando-os antes de inseri-los nas estruturas do sistema. O files fornece funções auxiliares para manipulação de arquivos como abertura, escrita e registo de erros. O interactive implementa o modo interativo, permitindo que o usuário insira comandos diretamente pelo terminal.

5. Módulo de Utilidade

Este módulo fornece as funções genéricas e reutilizáveis que suposrtam a lógica principal do sistema, vale admitir que na fase 1 faltáva-nos este módulo em específico. No caso adicionamos o utils.h/c que contém funções auxiliares usadas em diversos

módulos, como manipulação de strings (remove_quotes) ou cálculos. Temos támbem o validation.h/c que se trata do módulo de validação e verificação dos dados antes de inseri-los nas estruturas.

Encapsulamento

O encapsulamento consiste em restringir o acesso direto aos dados, garantindo que apenas o módulo responsável possa manipulá-los diretamente.

Vantagens do Encapsulamento no nosso projeto:

1. Flexibilidade

Mudanças internas nas estruturas de dados podem ser feitas sem impacto nos módulos externos, já que estes acessam os dados apenas por meio das interfaces públicas.

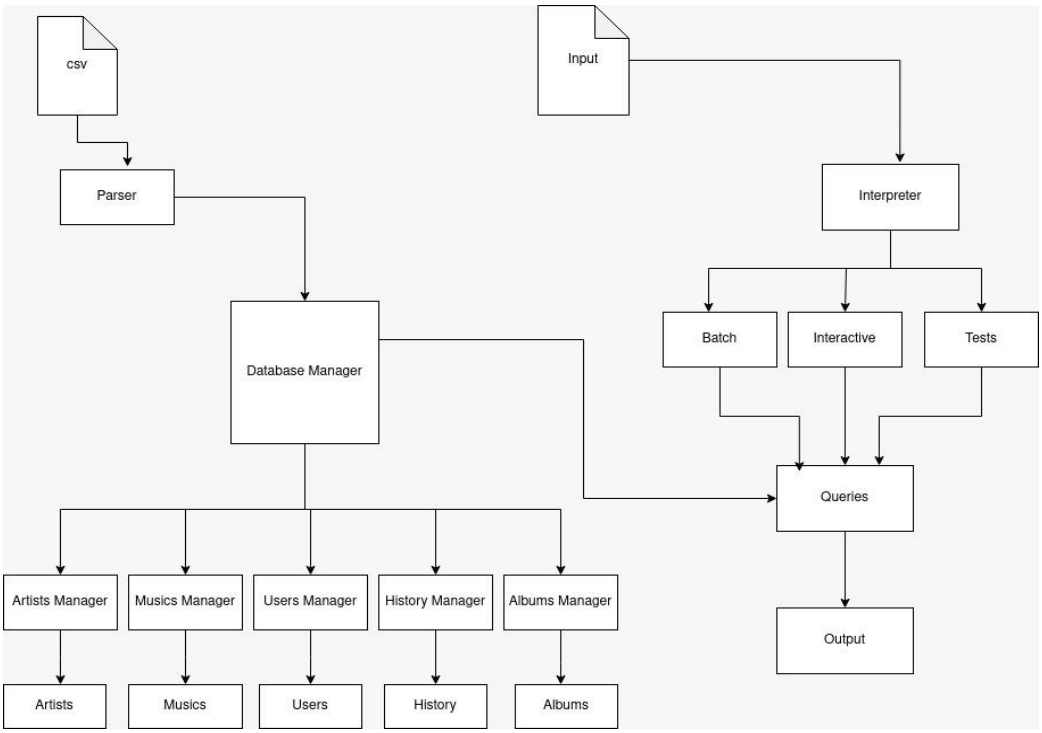
2. Robustez

Qualquer problema com a manipulação dos dados pode ser isolado e resolvido dentro do módulo responsável, reduzindo erros.

3. Manuntenção

A possibilidade de modificar a implementação interna das estruturas sem alterar a interface pública simplifica a evolução do sistema, sendo que isto nos ajudou bastante a melhorar alguns problemas que tinhamos na fase 1.

Arquitetura



A arquitetura do trabalho é robusta, organizada e segue boas práticas de desenvolvimento. Ela facilita a manutenção, expansão e depuração. No fundo está o que estivemos a explicar acima sobre modularidade e encapsulamento num desenho.

Queries

- **Query 1**

  Esta query tem como objetivo retornar informações detalhadas de um utilizador ou artista com base no identificador fornecido. A função funciona da seguinte forma, se um user_id for fornecido, a função procura o utilizador na hashtable de users, caso contrário, busca-se o artista na hashtable de artists. Para utilizadores o código recupera e imprime informações como email, nome, idade e país. Para artistas, são recuperados o nome, tipo, país, número de álbuns individuais e a receita total.

- **Query 4**

- **Query 5**

- **Query 6**

Validação dos datasets

A validação dos datasets, serve para garantir a integridade dos dados, prevenir erros no sistema e registar as inconsistências.

Temos presentes dois tipos de validação, a validação sintática e a validação lógica, a sintática certifica-se que os campos têm o formato correto, datas, emails, duração de músicas etc. A lógica verifica a consistência dos dados em relação ao contexto do sistema, por exemplo o campo artist_id deve referenciar um artista existente e válido, entre outros exemplos.

Programa de testes e Modo Interativo

Programa de testes

O programa de testes é uma ferramenta cujo seu objetivo principal é validar a implementação das queries e avaliar o

desempenho e a utilização de memória. Os resultados obtidos são comparados com os resultados esperados e caso haja diferenças o programa informa, mede o tempo médio de execução de cada query o tempo total do programa e avalia a quantidade de memória usada. O programa recebe 3 argumentos, caminho para os arquivos csv, arquivo de comandos com as queries a serem executadas e o diretório contendo os resultados esperados.

Modo Interativo

Por outro lado, o modo interativo oferece uma interface para o usuário explorar os dados e executar queries no momento. O programa solicita o caminho para os arquivos csv de entrada e carrega os dados, permite ao usuário inserir comandos diretamente no terminal especificando a query e seus parâmetros, garante que os comandos e argumentos inseridos pelo usuário sejam válidos, por fim os resultados das queries são exibidos no terminal.

Ausência de leaks e erros de memória

A gestão de memória foi algo que na primeira fase nos deu algum trabalho de modo a não possuir memory leaks, porém conseguimos que os unicos leaks de memória fossem aqueles pertencentes à biblioteca Glib. No entanto mesmo assim, na

primeira fase pecamos nos erros de memória que não conseguimos tirar. No entanto nesta fase foi algo que conseguimos resolver com bastante mais facilidade.

Qualidade do código

A documentação serve como um guia para ajudar na interpretação de funções, descrevendo a funcionalidade das funções, estruturas e módulos. Por outras palavras facilita a compreensão da lógica do trabalho.

O código está documentado através do padrão que nos foi atribuído sendo este o Doxygen. De forma breve, as funções tem um resumo (brief) os parametros (param) e o que a função devolve (return).

Observações

Na nossa opinião, o trabalho apresenta uma boa base, com forte atenção à modularidade e encapsulamento. A validação e a extinção de leaks de memória também é algo que tivemos cuidado em ser exímios. No entanto é notório os pequenos ajustes a quais o trabalho necessita. O maior problema é ocupar uma grande quantidade de MB e demorar um vasto tempo para ser processado, devido a isso conseguimos fazer o trabalho apenas para o dataset regular. Conseguimos concluir com isso que para o programa demorar menos tempo poderíamos ter evitado processos repetidos, otimizar loops, usufruir da

execução paralela, isto é, várias partes do código a serem produzidas ao mesmo tempo. Para este usar menos memória deveríamos ter cuidado nas alocações desnecessárias e gerir o tamanho inicial das hashtables para evitar realocações frequentes. Vale também apontar a dificuldade evidente que possuímos nestas queries da fase 2, que tinham uma dificuldade acrescida. Sendo assim temos como pontos fracos o excesso de memória utilizada, o tempo de processamento e as falhas nas queries da segunda fase. No entanto, sentimos que este trabalho proporcionou novas aprendizagens relacionadas como por exemplo à modularidade, organização de codigo, encapsulamento, gestão de memória entre muitos outros.