



Universidade do Minho
Escola de Engenharia

Laboratórios de Informática III

Fase 1

Grupo 58

Ano letivo 2024/2025

Samuel Gibson Da Cunha Figueiredo Lobato - A106907

Lucas Rafael da Cunha Franco Robertson - A89467

Marco Rocha Ferreira - A106857

Conteúdo

Introdução.....	3
Sistema.....	4
Diagrama de arquitetura.....	4
Gestores.....	4
Entidades.....	5
Motivação para a modularidade.....	5
Queries.....	5
Query 1.....	5
Query 2.....	6
Query 3.....	6
Discussão.....	8
Conclusão.....	9

Introdução

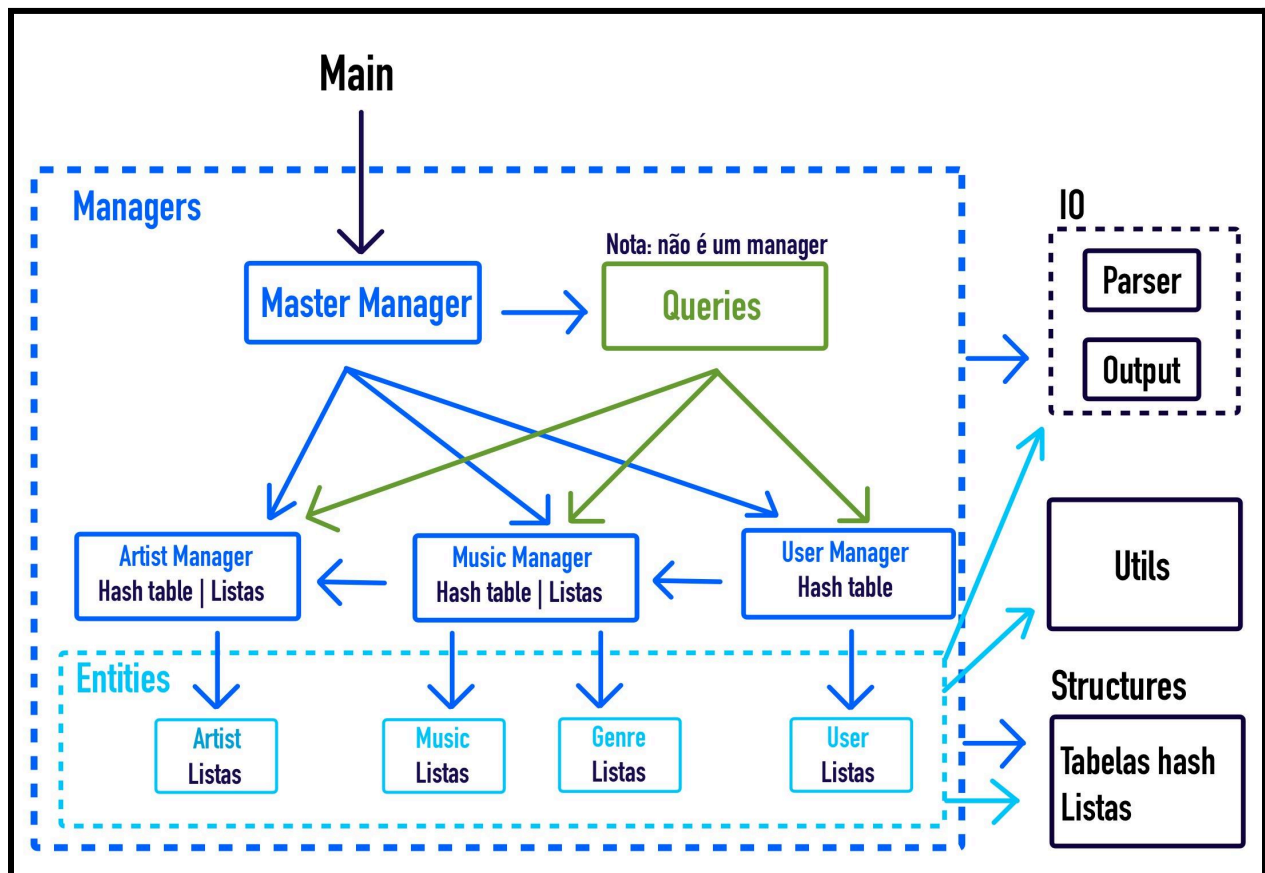
Este relatório configura-se como um material de apoio para o projeto realizado no âmbito da UC Laboratórios de Informática III, pelo grupo 58. O intuito deste trabalho resume-se a:

- Desenvolver um conhecimento mais profundo acerca da linguagem de programação C, com ênfase na aplicação do conceito de modularidade no código e no uso de estratégias de encapsulamento;
- Aprender a usar ferramentas que permitem melhorar a gestão de memória, com o principal objetivo de evitar *memory leaks*, que tornam o programa inseguro e instável.

A manipulação de forma organizada de uma quantidade razoável de dados num curto período de tempo e a utilização de uma quantidade de memória bastante adequada são características do nosso programa que merecem ser destacadas, e que serão exploradas nas secções a seguir.

Sistema

Diagrama de arquitetura



Gestores

- Master Manager: comporta os três gestores abaixo descritos e controla a execução do programa;

- Artist Manager: Inclui a tabela de hash dos artistas e um array dos mesmos ordenado segundo a discografia total de cada um;
- Music Manager: Inclui a tabela de hash das músicas e um array dos géneros musicais;
- User Manager: Inclui a tabela de hash dos utilizadores.

Cada um dos três gestores de entidades contém a lógica de armazenamento das mesmas, sendo que a validação lógica é executada aqui (só após ser validada é que uma entidade é efetivamente armazenada). Mais concretamente, o Music Manager valida a lista de artistas de cada música e o User Manager valida as *liked musics* de cada utilizador.

Nota: as três tabelas de hash estão organizadas segundo os IDs das entidades correspondentes.

Entidades

- Artists;
- Musics;
- Genre;
- User.

Cada entidade é instanciável e contém os dados presentes nos ficheiros CSV. No que toca a funções, todos estes módulos disponibilizam *getters* e *setters* para que o acesso aos dados seja controlado, i.e., para que outros módulos não alterem os dados das entidades. Por fim, cada módulo contém ainda as funções de validação sintática relativas à sua entidade.

Queries

Este módulo tem uma estrutura geral que se ramifica em cada tipo de query e é única, ou seja, dependendo do tipo de query a ser processado, os campos são atualizados com a informação relevante dessa query. Depois, na execução de uma nova query, os campos anteriores são sinalizados como inválidos, e assim sucessivamente.

IO

Parser

Este módulo serve para leitura de ficheiros, instanciando estruturas Parser que contêm um apontador para um ficheiro e um campo que indica o número de bytes lidos no último acesso a esse ficheiro. Este último campo tem como propósito verificar se foi, efetivamente, lida alguma coisa e também navegar no ficheiro (recorrendo à função `fseek`).

Output

Contrariamente, o módulo Output serve para escrita, quer para o *standard output*, quer para ficheiros. Neste último caso, são instanciadas estruturas Output que contêm um apontador para o ficheiro (criando-o caso não exista).

Utils

De momento, o módulo Utils contém apenas uma função usada por vários outros módulos para validar os campos de lista dos ficheiros CSV.

Testagem

Este módulo contém funções específicas ao programa de testes, cujo objetivo é comparar os resultados obtidos com os que eram esperados, imprimindo para o terminal os ficheiros e a linha dentro deste onde ocorreram os erros.

Discussão

Configurações dos computadores

	PC 1	PC 2	PC 3
Processador	Intel® Core™ i7-1165G7	Intel® Core™ i7-1355U	AMD Ryzen 7 5700U
Frêquencia do CPU	4.70 GHz	5 GHz	4.3 GHz
Capacidade da cache	12 MB	12 MB	8 MB
Números de <i>cores</i>	4	10	8
Números de <i>threads</i>	8	12	16

Análise de desempenho

	Q 1 ¹ (ms)	Q 2 ¹ (ms)	Q 3 ¹ (ms)	Armazena mento ² (s)	Resposta a Queries ³ (s)	Free ⁴ (s)	Total ⁵ (s)
PC 1	0,001035	0,00754	0,001926				
PC 2	0.00072	0.004921	0.001134	1.741639	0.001403	0.127251	1.8702 94
PC 3	0.002820	0.016153	0.003418	3.120427	0.003622	0.209787	3.6111 5

¹ As colunas “Q1”, “Q2” e “Q3”, fazem referência ao tempo médio de execução das queries.

² Tempo de armazenamento, validação e organização dos dados.

³ Tempo total da lógica de resposta.

⁴ Tempo de libertação da memória usada.

⁵ Tempo de execução total.

Conclusão

Após muitas horas de trabalho, conseguimos arquitetar um algoritmo o qual, embora não possua uma interface de usuário, por muito pouco não se configura um trabalho profissional. Um código capaz de operar uma imensa quantidade de dados em um tempo relativamente curto e, graças ao encapsulamento, sem deturpar a informação original. Ademais, o produto final não admite, na medida do possível, vazamentos de memória, porquanto para a execução do programa há funções predefinidas de bibliotecas externas às quais não temos acesso a sua algoritmia (e por conseguinte não conseguimos neutralizar os erros de alocação de memória).

Francamente, dentre os maiores desafios para a conclusão da 1ª fase destaca-se:

- Dificuldade em projetar a arquitetura de sistemas exatamente como a equipa docente exigia: Diversas foram as vezes em que funções tiveram de ser realocadas para pastas completamente distintas da onde haviam sido inicialmente inicializa-das. Para a 2ª fase, é credível que a arquitetura nesse ponto já esteja organizada de forma mais definitiva.
- Complicações com o conceito de encapsulamento: Após certificação do funcionamento das queries, foi apresentado à produção deste código a necessidade de “encapsular” o programa. Isso exigiu uma certa remodulação dos alicerces do trabalho.

O projeto requerido permitiu aos integrantes do grupo revisar a sua base de programação, de forma a identificar déficits pessoais em conceitos fundamentais de computação, e consequentemente corrigi-los.

Por fim, após diversas reestruturações, informa-se que as barreiras à implementação das nossas ideias foram derrubadas com sucesso.