Relatório 1ª Fase | Laboratórios de Informática III

Grupo 78 | 2024/2025

Afonso Carpinteiro (A106909)

Bruno Carvalho (A106809)

Índice

1. Introdução

Este relatório tem como objetivo apresentar informações relativas à 1ª Fase do Projeto da Unidade Curricular "Laboratórios de Informática III", integrada no 2º ano da Licenciatura em Engenharia Informática, realizada no ano letivo de 2024/2025, na Universidade do Minho.

O referido projeto consiste na implementação de uma base de dados, utilizando-se, para o efeito, ficheiros *.csv* fornecidos pelos docentes, constituídos por dados relativos a uma aplicação de músicas (artistas, utilizadores e as próprias músicas). O principal objetivo é armazenar estes dados e implementar métodos de pesquisa e associações entre estes.

Neste projeto são aplicados conceitos basilares como a modularidade e o encapsulamento.

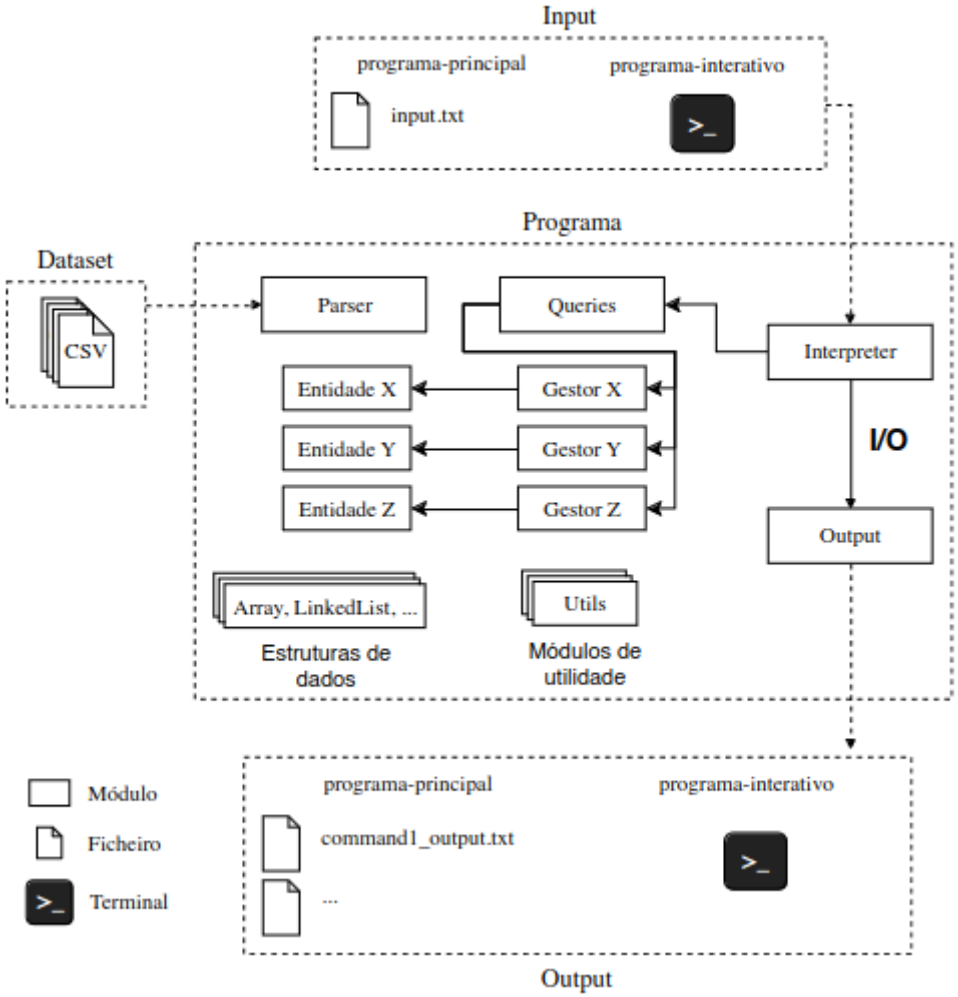
Na 1ª fase deste projeto, foi-nos pedido que implementássemos o *parsing*, a validação (sintática e lógica) dos dados de entrada, o programa principal, um programa de testes, assim como três *queries*.

2. Desenvolvimento

Neste projeto, foram implementados o parsing de dados e o programa-principal, sendo possível executar as queries sobre os dados de forma sequencial, estando esses pedidos guardados num ficheiro *.txt*, cujo caminho é recebido como argumento. Assim, é-nos requerido não só a leitura e interpretação dos dados presentes nos ficheiros, como também o seu armazenamento em estruturas de dados por nós escolhidas. Para auxiliar o nosso trabalho, foi-nos permitido que usássemos a biblioteca *glib*.

2.1. Estrutura Aplicada

No desenvolvimento deste trabalho, utilizou-se a seguinte estrutura (fornecida pelos docentes) presente no enunciado do projeto:



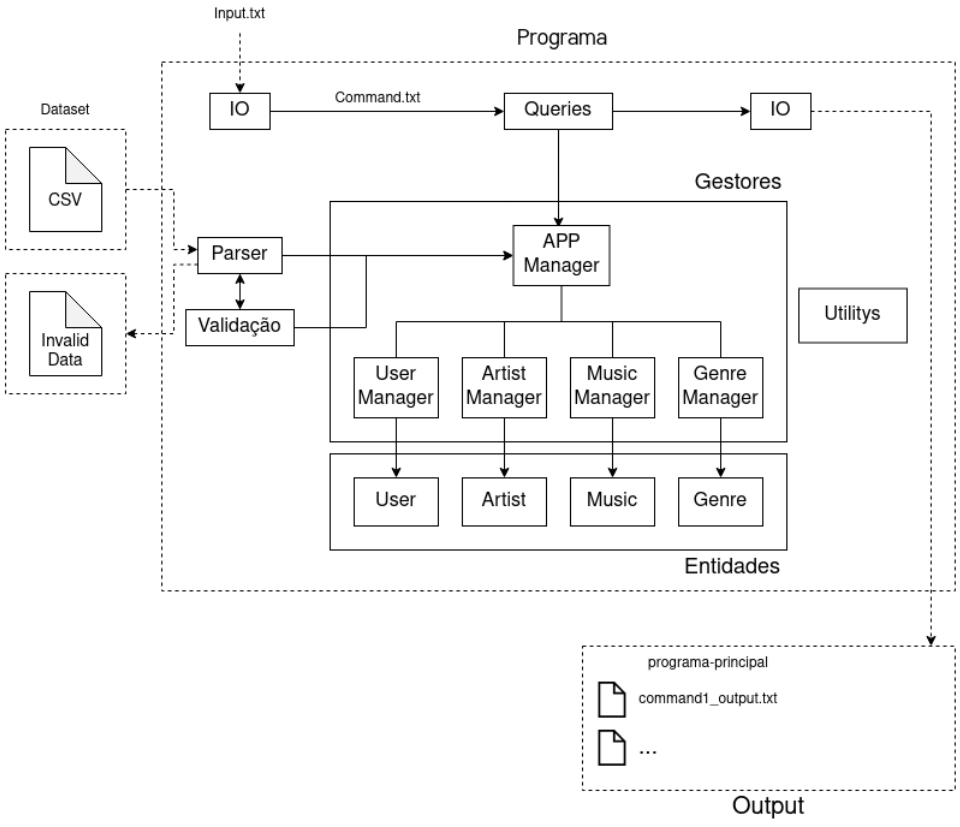
2.2. Funcionamento do projeto

A aplicação, através do número de argumentos recebidos, seleciona o seu modo de funcionamento. Nesta fase, não foi implementado o programa interativo, logo, este não será analisado nesta secção.

Através do primeiro argumento, o caminho para os ficheiros *.csv* do *dataset*, é criada e preenchida as bases de dados que contêm cada uma das entidades. A leitura de dados do ficheiro é realizada através de um *parser* genérico (comum a todas as entidades) que inclui funções que fazem a validação e a inserção dos dados nas estruturas de dados. Se a validação dos dados falhar, os mesmos são escritos num ficheiro de erros.

Posteriormente, é realizado o parsing dos comandos que obtém o número da query a ser chamada e os argumentos da query. Estas utilizam funções de listagem e obtenção de dados disponibilizadas pelos gestores das entidades e, no caso de uma query necessitar de gestores de diferentes entidades, é utilizado o módulo APP Manager, responsável por interligar diferentes gestores.

Através de duas funções de escrita em ficheiro (uma de escrita de strings e outra de escrita de int's), as queries vão escrevendo as respostas diretamente no ficheiro de output. Ao longo de todo o programa, são realizados todos os *free's* necessários de forma a garantir que, no final da execução do mesmo, não existem *memory leaks*.



2.3. Queries

2.3.1. Query 1

Apresenta vários dados relativos a um utilizador. De todas as queries, esta foi a mais fácil de implementar, visto que, recebendo o identificador (*id*) do utilizador, bastava procurar esse utilizador na Tabela de *Hash* (*GHashTable*) onde tinha sido, anteriormente (caso fosse sintaticamente e logicamente válido) guardado, utilizando como *key* da Tabela o id do utilizador. De seguida, são impressos no ficheiro de output os dados relativos ao utilizador.

2.3.2. Query 2

Apresenta os top N artistas, podendo ainda, no input, receber um filtro país. Se assim for, apresenta os top N artistas desse mesmo país. Para esta query aproveitamos a validação lógica das músicas. Verificando-se que certa música era válida, adicionou-se o valor da duração da mesma a cada um dos artistas da música. Na função da query em si, criou-se um *array* com todos os *id's* de artistas (nesta altura, já validados). Para tal fez-se uso de uma função da *GLib* que coloca num *array* por ela criado todas as *keys* de uma Tabela de Hash (neste caso, a tabela de Hash dos artistas). De seguida organizou-se o *array* por ordem decrescente de discografia e depois é/são impresso(s) no ficheiro de output o(s) top N artista(s). A maior dificuldade foi o facto do filtro país poder existir ou não, tendo, no entanto, sido prontamente identificada uma forma de resolver tal situação.

2.3.3. Query 3

Apresenta os géneros de música mais populares numa determinada faixa etária. Para esta query, decidiu-se criar uma entidade "géneros" e um gestor para o mesmo em que guarda os géneros num *array*. Aproveitamos a validação lógica dos utilizadores para criar e armazenar os *likes* do género pois quando vamos validar um utilizador é necessário verificar se as músicas que o utilizador gosta são válidas logo esta validação tem acesso ao gestor das músicas. Assim, enquanto verificamos as músicas guardamos os seus *id's* num array e, caso o utilizador seja válido, verificamos o género de cada uma procurando a música no gestor usando a função *music_manager_lookup* e adicionamos o *like* ao género dessa música. Quanto ao *like* adicionado, a estrutura da entidade género é constituida por uma *string* com o nome do género, um *array* de inteiros onde a posição corresponde à idade e o valor na posição corresponde aos *likes* do

género por utilizadores daquela idade. Por isso é também passada a idade do utilizador à função que adiciona *likes* ao género e ainda a estrutura possui um inteiro a que chamamos *soma_likes* que é utilizado na função da *Query* 3 e que, por isso, é inicializada a zero. A função da *Query* 3, recebe o gestor de géneros e, em cada género, soma os *likes* da faixa etária dada pelo comando de *input* e guarda este valor no campo previamente descrito denominado *soma_likes*. A seguir, o *array* é organizado por ordem decrescente a partir da *soma_likes* e depois basta imprimir no ficheiro de output a informação de cada género começando no início do *array*. A maior dificuldade nesta query foi como aproveitar a validação lógica do utilizador para armazenar os *likes* de cada género de forma a que cada *like* fique "ligado" a uma certa idade.

2.4. Análise de Desempenho

O programa foi executado cinco (5) vezes tendo sido registados, na tabela infra, os valores médios de cada query e de execução total, assim como o pico de memória utilizada. Os dois dispositivos utilizados foram os seguintes:

• Dispositivo 1 (Lenovo Ideapad 320-15ISK)

• Intel® Core™ i3-6006U @ 2.00GHz | 12,0 GiB RAM DDR4 2133MHz (Ubuntu 22.04.4 LTS)

• Dispositivo 2 (Lenovo Ideapad 100-15IBD)

• Intel® Core™ i3- 5005U @ 2.00GHz | 6,0 GiB RAM DDR3 1600MHz (Linux Lite 7)

	Dispositivo 1	Dispositivo 2
Query 1	0,002068s	0,016105s
Query 2	0,004477s	0,011396s
Query 3	0,009632s	0,001941s
Tempo de Execução Total	11,774221s	11,937068s
Pico de memória utilizada	192.998 KB	192.717 KB

Tabela 1: Tempo de execução médio das 3 queries, execução total e máximo de memória utilizada

2.5. Otimizações e Limitações

No que toca a otimizações, acreditamos que podíamos fazer uso de estruturas de dados que tivessem melhor compatibilidades com as queries, principalmente a *query* 2 e a *query* 3. Apesar destas possíveis otimizações, consideramos que já obtivemos ótimos resultados tanto no tempo de execução como na memória.

2.6. Dificuldades sentidas

A maior dificuldade sentida durante o desenvolvimento desta 1ª fase do projeto foram a resolução de *memory leaks* nem sempre óbvias, tendo de "correr" o *valgrind* várias vezes, o que acabava por ser um pouco penoso.

3. Conclusão

A realização desta 1ª fase do projeto foi, sem dúvida, desafiante, tendo sido, desde o início marcada por vários percalços. No entanto, consideramos que fizemos um ótimo trabalho, tendo conseguido concluir corretamente todas as tarefas obrigatórias nesta fase.

A nossa dedicação e organização de trabalho contribuiu também para o bom desempenho do grupo. Não menos importante, foi a aprendizagem de "novos" conceitos como a modularidade e o encapsulamento, assim como um maior conhecimento sobre a linguagem de programação *C*.

Acreditamos que ainda existem aspetos a melhorar e que será certamente um dos nossos objetivos para a 2ª fase do projeto.