



**Universidade do Minho**  
Escola de Engenharia

# Relatório Projeto-LI3-Fase2-Grupo 33

João Sousa – A106900

José Rocha - A106887

Vasco Cota - A106886

7 de janeiro de 2025

# Índice

## 1 - Introdução

## 2 – Arquitetura e Estruturas

### 2.1 – Alterações para a nova fase

### 2.2 - Arquitetura

### 2.3 - Queries

## 3 – Modo interativo

### 3.1 – Requisitos

- **3.1.1 - Gerais**
- **3.1.2 - Funcionamento**
- **3.1.3 - Interface**
- **3.1.4 - Adicionais**

### 3.2 -Interface

## 4 – Programa de testes

## 5 – Desempenho do programa

### 5.1 - Valgrind

## 6 – Principais dificuldades

## 7 – Conclusão

# 1 Introdução

A segunda fase deste projeto visou o desenvolvimento da implementação das nossas funções de carregamento de dados (parser) e de execução de comandos (executar\_comandos). Os focos principais desta fase foram a implementação de um modo interativo, criando uma interface com o utilizador mais intuitiva para a realização das queries e a manipulação dos dados.

Nesta segunda fase também houve um aumento do dataset, tendo sido acrescentados duas novas entidades (álbuns e histórico).

Para além disso também foram acrescentadas três novas queries e uma modificação a uma da fase anterior criando um maior leque de informações sobre os dados. O aumento da complexidade traz o desafio de manter um programa modular de forma a garantir uma eficiência na gestão dos dados fornecidos.

## 2 Arquitetura e Estruturas

### 2.1- Alterações para a nova fase

Durante o desenvolvimento do projeto fizemos ajuste estruturais e de redesignação de modo a otimizar o programa.

Começamos por simplificar as estruturas de modo a garantir um melhor desempenhos e uma melhor qualidade estrutural.

Para além disso realizamos alterações no processo de guardar os dados que nos eram fornecidos de maneira a simplificar processos utilizados anteriormente como por exemplo adicionar novos dados as estruturas que iriam ser necessários para a execução das queries (ex.: nalbuns nos artistas, nreproduções nas musicas, etc.).

Estas alterações permitiram não só uma simplificação da estrutura global mas também para facilitar a introdução de novas expansões.

### 2.2 Arquitetura

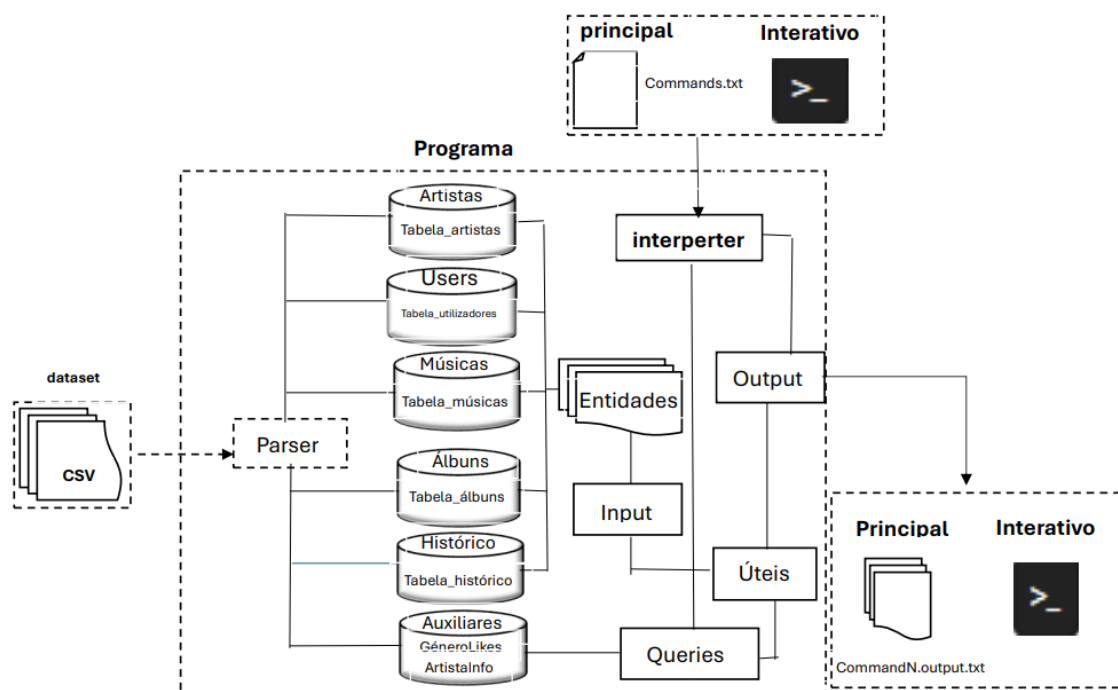
Na construção desta fase projeto, tivemos como objetivo realizar um desenvolvimento modular, tendo melhorado o que havíamos feito na fase anterior e incorporando um modo interativo. Dividindo o Projeto em várias fases permitindo uma modularização onde cada fase tem os seus propósitos.

Quanto á **modularidade** cada campo opera de forma independente facilitando a gestão, os updates e a interpretação das suas funções. Ao ter uma estrutura modular permite-nos ter uma maior flexibilidade na realização de

alterações sem ter de efetuar alterações em outras partes do programa. Esta técnica permite uma maior tranquilidade na realização de novas tarefas pois não se compromete a estrutura global do programa.

Outro aspeto importante para esta fase foi o **encapsulamento** das estruturas, onde cada módulo encapsula as suas operações libertando apenas interações necessárias para operações exteriores, Para além de garantir a integridade dos dados o encapsulamento também simplifica a leitura e a compreensão do programa.

A figura (2.2.1) apresentada abaixo demonstra de forma simplificada a estrutura de funcionamento do projeto.



## 2.3-Queries

Este projeto propõe a realização de 6 Queries sendo três referentes a esta nova fase, duas á fase anterior e outra já presente na fase anterior mas com alterações para a nova fase.

**Querye 1:** Listar o resumo de um utilizador ou de um artista consoante o identificador recebido por argumento.

Esta querye irá receber como input o id de um utilizador ou de um artista. A nossa função irá verificar se se trata de um id de um utilizador ou de um artista. Caso verifique que se trata de um utilizador, através da função `buscar_user`, a função

listar\_resumo\_utilizador irá imprimir os dados email; 1º nome, apelido, idade (calculada pela função calcular\_idade) e o país para o ficheiro. Caso não seja utilizador irá verificar se é artista, pela função buscar\_artista, caso seja a função listar\_resumo\_artista irá imprimir os dados nome, tipo, país, nº albúms e receita (estes últimos dois são calculadas aquando do carregamento de albúms onde irá ser incrementado o nº de albúms do artista caso o álbum lhe pertença já a receita irá ser calculada no carregamento do histórico onde se irá ver a quem pertence a música do histórico e em caso de ser um artista individual a receita é incrementada somando o valor da receita por stream do artista, caso seja um grupo a receita do grupo será incrementada pelo valor da receita por stream do grupo e depois a receita dos constituintes do grupo será incrementada pelo valor da receita por stream do grupo a dividir pelo nº de constituintes)

## **Querie 2: Quais são os top N artistas com maior discografia**

Nesta querie o objetivo é dar o top N artistas com maior discografia podendo ainda ser colocado um filtro de país, ou seja o top N artistas com maior discografia de um país. Para isso a função output\_artists irá percorrer a tabela dos artistas e em caso de existência do filtro para o país irá só verificar os artistas deste país guardando a lista “filtered\_artists” após isto irá ordenar esta lista em ordem decrescente do tempo da discografia de cada artista (Este campo é calculado ao carregar músicas verifica a lista dos artistas da música e incrementa a discografia ao adicionar a duração da música) através da função compare\_artists. Após isto através de um ciclo for irá imprimir as informações dos primeiros n artistas desta lista (i.e. o nome. O seu tipo (individual ou grupo), a duração da discografia (formato hh:mm:ss) e o país)

## **Querie 3: Quais são os géneros de música mais populares numa determinada faixa etária.**

A query recebe como argumento as idades mínima e máxima da faixa etária e deve dar como output a lista dos géneros com maior popularidade para essa faixa etária. De modo a facilitar a execução criamos a struct “GeneroLikes” que guarda o género e o número de likes que as músicas do género tem. A função principal “querie3” inicializa a struct dos géneros e irá percorrer a tabela dos artistas e calcula a idade deste se ele tiver idade entre a faixa etária irá verificar o género das músicas que este deu like e incrementa o valor de likes daquele género. Terminando por imprimir por ordem decrescente o género e o respetivo nº de likes.

## **Querie 4: Qual é o artista que esteve no top 10 mais vezes.**

Nesta query é nos pedido para indicar qual é o artista que esteve presente mais vezes no top 10 sendo possível receber um intervalo de tempo para analisar sobre esse período. Este top tem por conta o tempo de audição durante o período estipulado. Para resolver esta querie o nosso programa irá definir o intervalo de tempo a verificar caso não exista irá ser utilizado um intervalo geral de modo a abordar todo o histórico. Após isso agrupamos o período por semanas para calcular qual foi o top dessa

semana. Após serem agrupados os tops pela semana iremos verificar qual o artista esteve presente mais vezes nesse top durante o período estipulado. Em caso de empate entre dois artistas o que irá desempatar será o id ficando o que tem menor id na frente. (Esta query não está implementada na execução do programa).

### **Query 5: Recomendação do utilizador com gostos parecidos.**

Para esta query nós devíamos construir uma matriz onde cada linha representa um user e cada coluna um género e o valor de cada célula correspondia ao número de likes que o utilizador tinha em músicas daquele género. Também nos foi fornecida uma biblioteca para usar na implementação desta query. Nós recebíamos dois argumentos o 1º era o id do user a qual deveria ser dado os id dos users semelhantes e o 2º argumento era o número de users semelhantes a serem recomendados. No nosso programa a matriz é criada e após isso iremos calcular a similaridade entre os users. Após verificar a similaridade atribuindo um parâmetro de semelhança iremos organizar num array por ordem de semelhança e iremos imprimir o número de utilizadores recomendados que nos é pedido. (Esta query não está implementada na execução do programa).

### **Query 6: Resumo anual para um utilizador:**

Para esta query recebemos dois argumentos, o utilizador a qual o resumo será feito e o segundo será o ano a que o resumo irá fazer referência. Pode ainda ser recebido um terceiro argumento N que no resultado deverá ser impresso os N artistas mais ouvidos pelo utilizador naquele ano. No final devemos imprimir o tempo total de audição daquele user, o número de músicas ouvidas, o artista mais ouvido, o dia que reproduziu mais músicas, o género mais escutado, o álbum favorito, a hora do dia que ouviu mais músicas e caso tenha o argumento N o top artistas. Para resolver este problema começamos por obter o histórico do ano correspondente daquele user e guardar numa lista. Após isso incrementamos o tempo de audição, o nr de músicas distintas caso aquela música ainda não tiver sido ouvida antes, também iremos verificar a lista dos artistas daquela música e alterar a informação daquele artista na sua struct "Artistainfo" que contém a duração e o nº de músicas que o user ouviu daquele artista. De seguida verificamos o género da música e incrementamos o valor do número de vezes que ouviu músicas daquele género assim como o tempo de audição do álbum a que a música pertence será incrementado pelo tempo de audição do histórico. Para terminar a análise ao histórico iremos verificar a dia e a hora e incrementar na tabela. Após todo o histórico analisado iremos verificar qual foi o artista, a música, o álbum e o género mais ouvidos e também qual foi a dia e a hora que houve maior audição. Caso seja pedido também listamos o top artistas daquele User.

## **3-Modo Interativo**

Neste ponto iremos abordar de forma detalhada o modo interativo, um dos focos principais desta fase. Neste modo nós desenvolvemos vários menus interativos de forma intuitiva para o utilizador para poderem aceder a realização das Queries.

### **3.1 Requisitos**

O programa interativo deve seguir os seguintes requisitos:

#### **3.1.1 Gerais**

- O programa deve ser um executável que não receba argumentos na linha de comando.
- Deve disponibilizar um menu interativo via terminal.
- O menu deve permitir a execução de comandos (queries) sobre os dados.
- O programa deve ser projetado para não sofrer crashes ou comportamentos erráticos com comandos ou argumentos inválidos

#### **3.1.2 Funcionamento**

- **Seleção do Dataset:**
  - Perguntar ao utilizador o caminho do dataset a processar.
  - Caso o utilizador não forneça um caminho, deve ser usado um caminho padrão relativo à pasta de execução.
- **Execução de Queries:**
  - Solicitar ao utilizador que introduza a query a executar e os respetivos argumentos.
  - Validar os dados introduzidos pelos utilizadores para evitar erros.

#### **3.1.3 Interativos**

- Deve facilitar a navegação e compreensão dos comandos disponíveis.

#### **3.1.4 Adicionais**

- Caso o caminho não seja especificado deverá ser usado um default.

### 3.2 Interface

```
joserocha@MacBook-Pro-de-Jose-2 trabalho-pratico X ./Programa-interativo
== Programa Interativo ==
Insira o caminho dos dados: |
```

```
== Programa Interativo ==
Insira o caminho dos dados: small/dataset/som_eros
Carregando dados do diretório: small/dataset/som_eros
Carregando dados de usuários, músicas, artistas e álbuns...
Carregamento de artistas concluído.
Iniciando o carregamento de álbuns...
Carregamento de álbuns concluído.
Iniciando o carregamento de músicas...
Carregamento de músicas concluído.
Iniciando o carregamento de utilizadores...
Carregamento de utilizadores concluído.
Iniciando o carregamento do histórico...
Carregamento do histórico concluído.
Dados carregados com sucesso!
```

```
== Menu Interativo ==
1. Resumo de Utilizador/Artista (Query 1)
2. Top N Artistas com Maior Discografia (Query 2)
3. Géneros Populares por Faixa Etária (Query 3)
4. Artista com Mais Presenças no Top 10 (Query 4)
5. Recomendação de Utilizadores (Query 5)
6. Resumo Anual para um Utilizador (Query 6)
0. Sair
Escolha uma opção: 7
Opção inválida. Escolha novamente.
Escolha uma opção: 1
Insira o ID do Utilizador/Artista ou escreva 'menu' para voltar ao menu: A0009188

Resumo do Artista:
MC Mark;individual;Hungary;0;0.00
Pressione Enter para voltar ao menu...|
```

```
== Menu Interativo ==
1. Resumo de Utilizador/Artista (Query 1)
2. Top N Artistas com Maior Discografia (Query 2)
3. Géneros Populares por Faixa Etária (Query 3)
4. Artista com Mais Presenças no Top 10 (Query 4)
5. Recomendação de Utilizadores (Query 5)
6. Resumo Anual para um Utilizador (Query 6)
0. Sair
Escolha uma opção: |
```

Figura 3.2.1 exemplo de interface do programa interativo.

## 4-Programa de testes

O programa de testes foi parte essencial para a depuração da correta realização das funções do programa verificando a validade das respostas. Desde o tempo para a execução de cada query quer para a análise do uso da memória. Podendo assim realizar uma análise abrangente do desempenho do programa.

Os testes seguem o modelo da modularidade e do encapsulamento avaliando cada query individualmente sem deixar que algum fator externo interfira. Assim permite uma maior facilidade na identificação e correção de problemas.

A estrutura do programa de testes segue a sugestão dada pelos professores no enunciado para verificar a execução sendo possível a realização dos testes com diferentes tipos de dados dando uma maior exatidão sobre o desempenho de cada query.

De forma resumida este programa de testes recebe um caminho para o dataset, outro para o input.txt dos comandos e o caminho para os ficheiros com os resultados esperados. Este programa executa os comandos e compara os resultados obtidos com os esperados indicando para cada query quais os comandos a que esta realiza e apresenta diferenças com os esperados. Para além disso indica o tempo para realizar cada query, o tempo total de execução e a memória usada.

## 5 – Desempenho

A constante análise do desempenho representa uma parte crucial para um desenvolvimento equilibrado e eficiente do programa. Nesta secção iremos abordar o comportamento do sistema em várias máquinas diferentes.

Iremos mostrar dados relativamente ao tempo médio de execução de cada querie, o tempo total de execução do programa e a memória utilizada comparando com os diferentes datasets. Esta comparação permite-nos tirar conclusões sobre a influência de cada fase no programa.

A tabela 5.0.1 apresenta os valores tanto do tempo de execução do programa como da memória usada em máquinas com especificações diferentes.

	Máquina 1	Máquina 2	Máquina 3
Modelo	MacBook Pro 2022	Asus Tuf Dash F15	Asus VivoBook X509MA
CPU	Apple M2	Intel i7	Intel i5
RAM	8GB	16GB	8GB
Sistema Operativo	MacOS	Ubuntu	Ubuntu
Compilador	Apple clang version 16.0.0	Gcc 11.4.0	Gcc 11.4.0
Memória Usada (small)	ERRO	0.92GB	0.919GB
Tempo execução	4.346s	4.501s	22.422s
Memória Usada (big)	ERRO	5.097GB	5.097GB
Tempo execução	147.728s	83.554s	449.997s

Os valores de memória usada no caso da maquina 1 estão denotados como error devido ao nº especificado pela máquina ser muito discrepante do valor real. (673GB e 2677GB valores impossíveis de ser o que realmente acontece)

Após análise da tabela é de notar que os computadores com hardware com melhor desempenho têm um comportamento semelhante em relação ao tempo de execução o mesmo não acontece com a máquina com pior desempenho que gera uma discrepância nestes quesitos apesar de em termos de memória serem valores semelhantes.

A seguinte tabela faz uma análise do tempo de execução do programa e do tempo médio em cada querie.

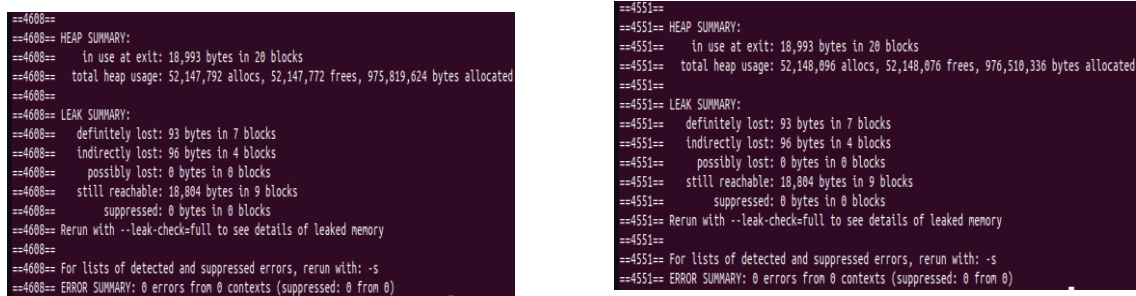
	small	big
Q1	0.591ms	35ms
Q2	0.071ms	0.5ms
Q3	0.037ms	0.215ms
Q6	0.082ms	0.154ms
Total	4.4s	115s

Os baixos valores no tempo médio de execução das queries mostra a eficiência do programa na resolução das queries.(Tempos Q4 e Q5 não presentes pois não são executadas)

### 5.1 - Valgrind

Outra ferramenta que foi essencial para a realização desta fase foi o Valgrind que permitiu que mantivéssemos a eficiência da gestão de memória tanto no programa principal como no programa de testes tendo reduzido de forma

significativa nesta fase os valores de memory leaks. Como mostra a imagem abaixo (figura 5.1.1)



```
==4608== HEAP SUMMARY:
==4608==   in use at exit: 18,993 bytes in 20 blocks
==4608== total heap usage: 52,147,792 allocs, 52,147,772 frees, 975,819,624 bytes allocated
==4608== LEAK SUMMARY:
==4608==   definitely lost: 93 bytes in 7 blocks
==4608==   indirectly lost: 96 bytes in 4 blocks
==4608==   possibly lost: 0 bytes in 0 blocks
==4608==   still reachable: 18,804 bytes in 9 blocks
==4608==   suppressed: 0 bytes in 0 blocks
==4608== Rerun with --leak-check=full to see details of leaked memory
==4608==
==4608== For lists of detected and suppressed errors, rerun with: -s
==4608== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

==4551== HEAP SUMMARY:
==4551==   in use at exit: 18,993 bytes in 20 blocks
==4551== total heap usage: 52,148,096 allocs, 52,148,076 frees, 976,510,336 bytes allocated
==4551== LEAK SUMMARY:
==4551==   definitely lost: 93 bytes in 7 blocks
==4551==   indirectly lost: 96 bytes in 4 blocks
==4551==   possibly lost: 0 bytes in 0 blocks
==4551==   still reachable: 18,804 bytes in 9 blocks
==4551==   suppressed: 0 bytes in 0 blocks
==4551== Rerun with --leak-check=full to see details of leaked memory
==4551==
==4551== For lists of detected and suppressed errors, rerun with: -s
==4551== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

**Figura 5.1.1 Resultado do valgrind tanto no programa de teste quer no programa principal**

A utilização ativa desta ferramenta permitiu garantir uma maior estabilidade no desenvolvimento do programa. Concluindo, com esta abordagem permitiu-nos criar um sistema praticamente livre de memory leaks.

## 6 – Principais dificuldades

### 6.1 - Desenvolvimento de queries e gestão de memória

A principal dificuldade no desenvolvimento desta nova fase foi a implementação das queries. Os principais problemas deviam-se ao gerenciamento da memória e da eficiência do programa para uma correta execução das mesmas. Estes problemas ficaram agravados devido á diferença dos sistemas. Fazendo com que várias vezes o programa no sistema MacOS devdo ao diferente modo de funcionamento gerasse problemas nos outros sistemas sem que fosse detetado no mesmo. Tornando, assim, mais complexo o desenvolvimento de um programa eficiente.

## 7 - Conclusão

Em jeito de conclusão no desenrolar deste projeto estivemos num processo constante de aprimoramento dos nossos conhecimentos e das nossas capacidades naquilo que diz respeito ao desenvolvimento dos aspetos principais deste projeto.

Desde o entendimento da importância da modularidade e do encapsulamento para a construção de um sistema eficaz e eficiente, ao gerenciamento da memória desde perceber o que realmente era necessário para a implementação do programa, á eliminação de módulos desnecessários, a criação de novas estruturas e métodos para facilitar os processos como a criação de um modo interativo que gera uma abordagem amigável e simples com o utilizador. Permitindo haver várias experiências intuitivas com o programa.

Apesar de não estar perfeito, este projeto teve uma grande importância para nós pois permitiu desenvolver uma grande aprendizagem e preparando-nos com bases mais sólidas para o desenvolvimento de novos projetos.