Universidade do Minho

Relatório 1ª Fase | Laboratórios de Informática III | Grupo 37 | 2024/2025

Francisco Lage (A106813) Ricardo Neves (A106850) Rui Faria (A106899)

Índice

1. Introdução

O projeto tem como objetivo o desenvolvimento de um **sistema para analisar dados de um serviço de streaming de música**, que permite a consulta, análise e interação com um conjunto de dados de utilizadores, músicas, artistas e preferências. A plataforma foi desenvolvida em linguagem **C**, com foco na eficiência no processamento e na leitura de grandes volumes de dados.

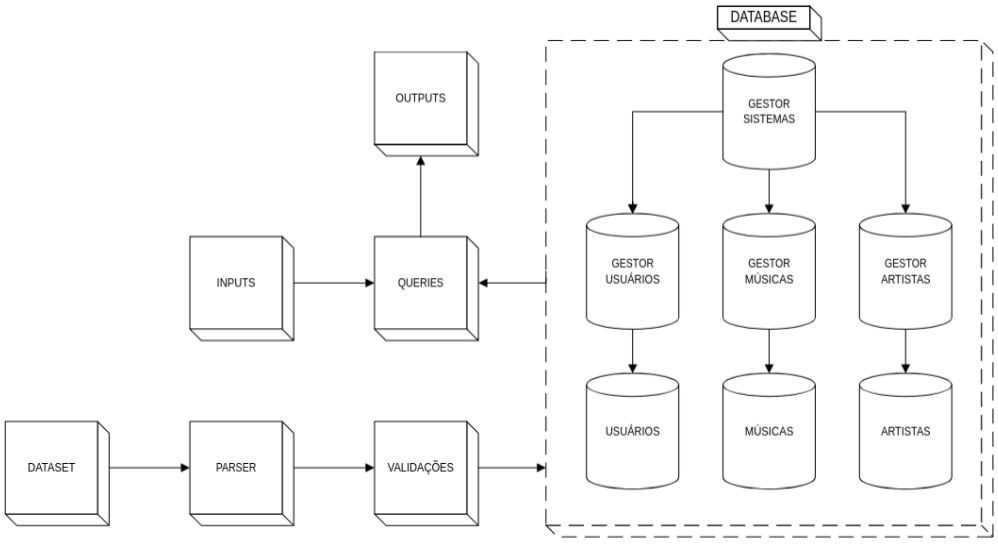
A execução do sistema envolve a leitura de arquivos *CSV*, que contêm informações sobre utilizadores, músicas, artistas e um ficheiro *inputs.txt*, que contém os comandos de consulta que o sistema deve processar. Os dados são então manipulados e analisados com base em *queries*, cujos resultados são escritos em arquivos de saída, utilizando o ***writer*** para garantir que os dados sejam formatados corretamente e armazenados de forma eficaz.

2. Objetivos do Projeto

O objetivo principal do projeto é implementar um sistema de análise de **streaming de música** com a capacidade de:

- **Processar dados de utilizadores, artistas e músicas** a partir de arquivos CSV.
- **Executar queries** complexas, que envolvem dados relacionados a utilizadores e suas preferências musicais.
- **Escrever os resultados das consultas** em arquivos de saída formatados de forma adequada, utilizando a funcionalidade de **writers**.
- **Gerir erros de execução e falhas** no processamento, filtrando informações inválidas.

3. Arquitetura do Sistema



4. Funcionamento do Sistema

4.1. Processamento de Arquivos *CSV* e *inputs.txt*

O sistema começa com a leitura dos arquivos de entrada, nomeadamente os arquivos **CSV** e o *inputs.txt*. O arquivo CSV contém informações sobre os utilizadores, músicas e artistas, estruturadas em várias colunas. A leitura e processamento desses arquivos são feitos de forma eficiente, utilizando funções que interpretam cada linha de acordo com o tipo de dado que ela representa. Esses dados são então armazenados nas estruturas internas do sistema, como as *hashtables*.

O arquivo *inputs.txt* contém os **comandos** de consulta que o sistema deverá processar. Cada linha deste arquivo corresponde a uma query específica que o sistema deve interpretar. O comando começa com um ***token*** que determina qual query será executada (por exemplo, 1 para a obtenção de dados do utilizador). Com base neste token, o sistema chama a função correspondente para processar a consulta.

4.2. Uso de Writers para Gravação de Resultados

O sistema utiliza *writers* para gravar os resultados das consultas(queries) em arquivos de saída formatados adequadamente. Cada tipo de consulta possui um formato de saída específico, que é respeitado pelo writer correspondente.

Por exemplo, ao processar uma consulta sobre os **artistas mais populares**, o sistema escreve os resultados num ficheiro com a estrutura:

name 1;type 1;discography duration 1;country 1

name 2;type 2;discography duration 2;country 2

...

4.3. Execução das Queries

O sistema implementa um conjunto de consultas que permitem recuperar informações específicas a partir de dados armazenados. Cada consulta é invocada conforme os comandos presentes no arquivo *inputs.txt*, com o objetivo de processar rapidamente grandes volumes de dados.

A **Query 1** permite recuperar as informações de um utilizador com base no seu *username*. Para isso, o sistema faz uso de uma *Hashtable* (*GHastable*), que armazena os dados dos utilizadores de forma eficiente. A chave de busca, que é o *username*, mapeia diretamente para a posição na tabela, permitindo que a consulta seja realizada de forma rápida. Uma vez localizada a entrada correspondente, as informações do utilizador, como *nome*, *e-mail* e *tipo de subscrição*, são formatadas conforme o especificado e escritas no ficheiro de saída. Se o utilizador não for encontrado, o sistema gera um ficheiro de saída vazio.

A **Query 2** tem como objetivo recuperar os **top N artistas com maior discografia**. Para isso, o sistema processa os dados dos artistas, levando em consideração um possível filtro de país. Caso o filtro de país seja fornecido, apenas os artistas desse país são considerados na consulta. Se não houver filtro, todos os artistas são avaliados.

O processo inicia com a obtenção dos dados necessários, utilizando as estruturas de dados como tabelas hash para garantir um acesso eficiente aos artistas e às suas informações. É obtida a partir dos valores da hashtable dos artistas, uma lista de artistas(GList),que é, então, ordenada pela duração total das suas músicas. Os N artistas com maior duração de discografia são selecionados.

Esses dados (nome do artista, tipo, duração total da discografia e país) são gravados em um ficheiro *CSV* de saída. Caso o filtro de país seja aplicado, apenas os artistas correspondentes são registrados.

Por fim, a **Query 3** busca identificar os gêneros musicais mais populares entre os utilizadores de uma faixa etária específica. O sistema utiliza uma *GHashtable* para contar os likes por género, e uma lista ordenada para garantir que os géneros mais populares sejam destacados. O resultado é formatado e escrito no ficheiro de saída, ordenado conforme os critérios definidos (ordem decrescente de likes e, em caso de empate, por ordem alfabética).

Cada consulta é projetada para garantir rapidez e eficiência na recuperação e processamento dos dados, utilizando estruturas de dados otimizadas, como a *Hashtables*, e estratégias de formatação precisas para gerar os ficheiros de saída conforme o esperado.

5. Desenvolvimento do Sistema

5.1. Definição de Funcionalidades

O sistema permite que os utilizadores interajam com o banco de dados de música, executando consultas sobre os seus dados pessoais, sobre artistas e sobre as músicas que mais gostam. As principais funcionalidades do sistema incluem:

- **Consulta a Dados de Utilizadores**:

- Recuperação de informações sobre utilizadores com base no **username**.
- **Consulta de Likes**: Exibição das músicas mais curtidas por cada utilizador.
- **Consulta de Artistas**: Obtenção dos artistas mais populares, com base na sua discografia e país de origem.

5.2. Estrutura do Banco de Dados

O banco de dados foi estruturado com tabelas separadas para utilizadores, artistas, músicas e gêneros musicais.

5.3. Gestão de Erros e Validação de Dados

O sistema possui um mecanismo de **validação de dados** localizado na pasta *validacao*, que garante a integridade das entradas de utilizadores, artistas e músicas. Cada tipo de dado é validado com base em critérios específicos, como formato de e-mail para utilizadores, dados de país para artistas e consistência nos campos das músicas.

Caso seja detectado um erro, o sistema gera um ***log de erro*** correspondente, que é gravado em arquivos como *user_errors.csv, artist_errors.csv e music_errors.csv*, e movido para a pasta

resultados. Nas ***queries***, se ocorrer um erro (ex: utilizador não encontrado), o ficheiro de saída é gerado vazio, evitando resultados inválidos e mantendo a estabilidade do sistema.

6. Desafios e Soluções Implementadas

Durante o desenvolvimento do sistema, enfrentamos desafios significativos relacionados à gestão eficiente de dados e ao processamento rápido de grandes volumes de informações. A utilização de tabelas hash e listas duplamente ligadas foi crucial para garantir a agilidade na manipulação dos dados. Outro desafio que tivemos de superar foi a necessidade de apresentarmos um código modular e bem encapsulado.

7. Conclusões

Este projeto demonstrou a capacidade de desenvolver um sistema de análise de **streaming de música** eficiente, utilizando boas práticas de programação, como o encapsulamento, a modularidade e conceitos sólidos de estruturação de dados. O sistema foi projetado para lidar com grandes volumes de dados, sendo capaz de executar *queries complexas* com agilidade e garantir a integridade dos resultados. A utilização de tabelas hash, listas duplamente ligadas e outras estruturas eficientes permitiu um processamento rápido e escalável, fundamental para a experiência de uso.

Um dos maiores desafios enfrentados foi relacionado com a **gestão de memória**, particularmente devido aos ***leaks de memória*** que surgiram no componente do *Reader*. Esses leaks causaram uma série de dificuldades durante o desenvolvimento, consumindo tempo e esforço significativos para a depuração. O impacto desses problemas foi mitigado através da adoção de **estratégias alternativas**, como o uso de *string split* para o processamento de entradas.

No geral, o projeto serviu como uma valiosa experiência no desenvolvimento de sistemas robustos e eficientes, proporcionando aprendizagens significativas sobre gestão de memória e a importância de testar e validar adequadamente cada componente do sistema. A experiência adquirida será essencial para otimizar o código na próxima fase do projeto.