Universidade do Minho
Escola de Engenharia

Relatório Fase 1

Implementação de Plataforma de Streaming de Música

Diogo Moreira, Nuno Mendes, Rafael Silva

Universidade do Minho - Licenciatura em Engenharia Informática

11 de novembro de 2024

Universidade do Minho
Escola de Engenharia

INTRODUÇÃO

Este trabalho prático tem como principal objetivo o desenvolvimento de uma plataforma de streaming de músicas implementada em linguagem C, que realiza a gestão e análise de dados relacionados a músicas, artistas e usuários. O projeto está dividido em duas fases, sendo esta primeira focada precisamente no tratamento desses dados. O sistema deve ser capaz de carregar dados fornecidos em arquivos .csv, armazená-los em estruturas de dados apropriadas e responder a um conjunto de queries pré-definidas.

Neste exercício, foi-nos pedida uma especial atenção para os conceitos de modularização e encapsulamento, competências essenciais para o correto funcionamento do programa. Assim, a estrutura do código foi organizada de forma modular e com atenção ao encapsulamento, separando claramente as interfaces (.h) e implementações (.c) de cada componente.

Universidade do Minho
Escola de Engenharia

SISTEMA

Arquitetura Geral

A arquitetura do sistema de streaming de música foi planeada de forma a assegurar a modularidade e a eficiência na manipulação de dados, abordando cada etapa do processamento de maneira organizada e estruturada. Cada funcionalidade do sistema foi isolada em módulos com responsabilidades bem definidas, o que facilita o desenvolvimento e a colaboração entre elementos do grupo. Essa divisão permite que os módulos sejam construídos, testados e mantidos de forma independente, assegurando que melhorias ou correções possam ser feitas sem impactar todo o sistema.

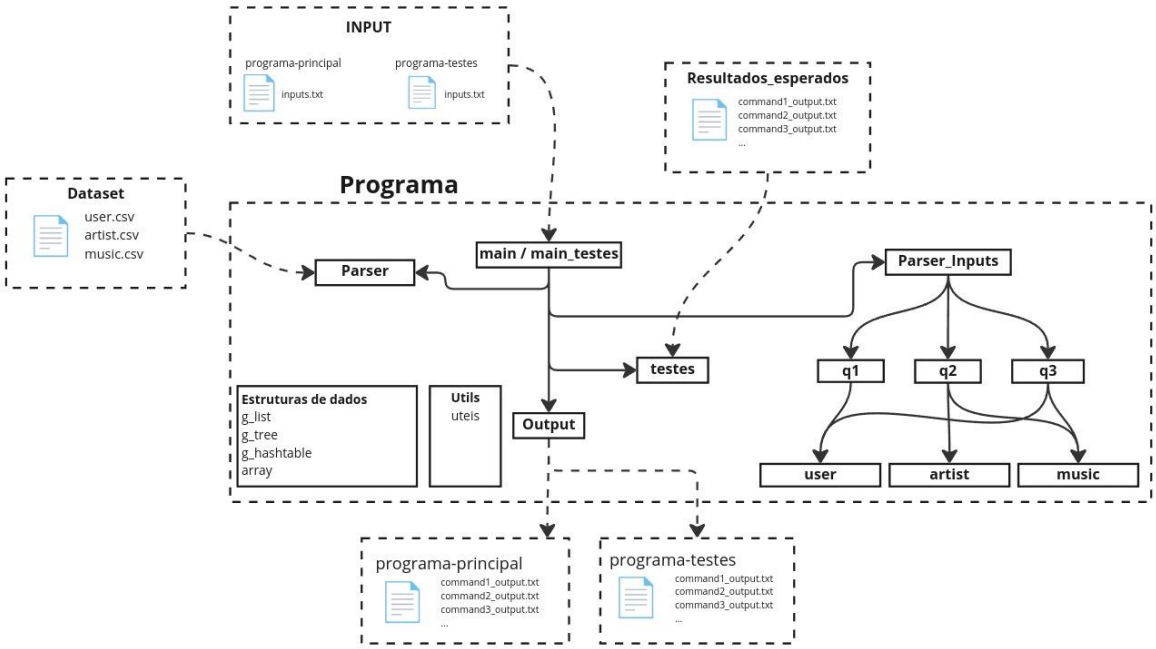

Figura 1 – Diagrama da Arquitetura do Sistema

Módulos Principais

Módulos de Entidade

A separação das entidades principais — artistas, usuários e músicas — em módulos independentes permite uma clara distinção de responsabilidades e facilita a organização dos dados. Cada um desses módulos encapsula as funcionalidades e estruturas relacionadas a uma entidade específica, o que ajuda a evitar dependências desnecessárias e aumenta a coesão do sistema. Essa abordagem permite que o sistema trate os dados de cada entidade de maneira especializada.

Módulos de I/O

A criação de módulos específicos para entrada e saída de dados (como os módulos parser e parser_input) foi motivada pela necessidade de separar as operações de manipulação de arquivos dos demais processos do sistema. Essa divisão permite que a leitura e formatação dos dados aconteçam de forma isolada, garantindo que os dados estejam prontos e no formato correto antes de serem utilizados por outros módulos. Além disso, essa modularidade facilita testes e depuração, já que problemas relacionados à entrada de dados podem ser diagnosticados e resolvidos de forma independente.

Módulo de Utilidade

O módulo de utilidade foi criado para centralizar funções comuns que são usadas em diversos módulos do sistema, evitando duplicação de código e mantendo a consistência. Ao agrupar operações auxiliares e genéricas, como manipulação de strings, criamos uma base de suporte para os demais módulos, que podem acessar essas funções sem precisar redefini-las. Essa divisão facilita a manutenção e torna o código mais limpo, pois qualquer ajuste nessas funções pode ser feito em um único lugar, propagando automaticamente as mudanças para todas as partes do sistema que dependem delas.

Módulos de Gestão do Sistema

Os módulos de gestão foram divididos em três módulos principais de queries (q1, q2 e q3) e um módulo de teste de modo facilitar a execução de funcionalidades específicas. Cada módulo de querie é dedicado a processar uma consulta específica, simplificando o desenvolvimento e a manutenção dessas funcionalidades. Essa divisão também possibilita que novos módulos de querie sejam adicionados futuramente, caso surjam novas necessidades.

O módulo testes, por outro lado, foi criado para gerenciar a verificação da integridade e exatidão dos resultados, comparando-os com os resultados esperados e fornecendo feedback imediato. Essa abordagem promove um desenvolvimento orientado a testes e simplifica o processo de validação, uma vez que permite identificar e corrigir rapidamente qualquer divergência nos resultados.

DISCUSSÃO

Análise Crítica

Ao desenvolver o sistema, um ponto crucial foi a escolha das estruturas de dados adequadas para as queries. Optamos por utilizar a biblioteca GLib, explorando estruturas como hash tables, listas e árvores para obter um desempenho equilibrado entre tempo de execução e consumo de memória. No entanto, o uso da GLib apresentou desafios, pois a sua implementação nem sempre otimiza o consumo de recursos para todas as queries, especialmente para a query q3, que exige uma análise mais complexa.

Testes e Resultados

A seção de testes foi essencial para comparar os resultados gerados pelo sistema com os resultados esperados, o que nos permitiu identificar erros com precisão. O monitoramento dos tempos de execução para cada query também nos ajudou a identificar pontos de melhoria em termos de desempenho. Nos testes realizados em diferentes ambientes de máquina (Figuras 2, 3 e 4), notamos que a query q3 teve um tempo de execução consideravelmente maior em comparação com as queries q1 e q2. Isto se deve à maior complexidade da q3 e reflete uma área a ser melhorada em relação à eficiência das estruturas de dados.

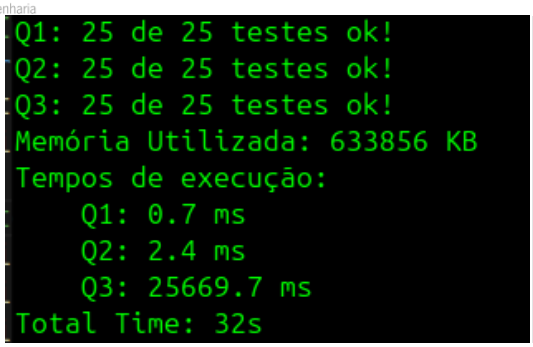

Figura 2 – Ambiente de Teste A106841

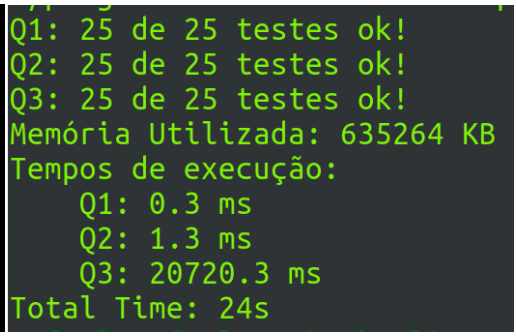

Figura 3 – Ambiente de Teste A107289

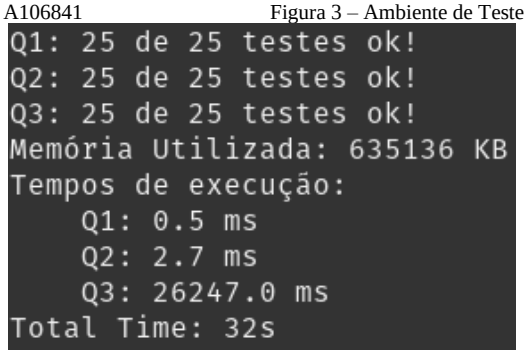

Figura 4 – Ambiente de Teste A106875

Estes foram os resultados obtidos em cada uma das máquinas. Embora existam algumas diferenças de tempos entre máquinas, o essencial, e é perceptivel nos três, é que há uma diferença significativa entres as queries. Tratando-se da querie de mais dificil implementação, a q3 acaba por ser onde o maior tempo de execução se dá. Não conseguimos um bom aproveitamento das estruturas de dados, sendo um ponto a melhorar no programa. Todavia, a q1 e a q2 não levam quase tempo nenhum de execução.

CONCLUSÃO

Concluímos que a implementação modular e o uso de encapsulamento foram essenciais para o sucesso desta fase do projeto, facilitando o desenvolvimento, a colaboração e a manutenção do sistema. Os testes realizados revelaram

diferenças significativas no desempenho entre as queries, destacando a necessidade de ajustes para otimizar o uso de memória e tempo de execução, especialmente para a query q3. Futuras Melhorias: Um ponto a ser aprimorado seria a implementação de estruturas de dados próprias e otimizadas para o nosso caso, uma vez que a dependência da GLib apresentou alguns problemas de desempenho. Em suma, o trabalho nesta fase foi um exercício valioso em modularidade e eficiência, e os aprendizados servem como base sólida para as próximas etapas do projeto.