



**Universidade do Minho**  
Escola de Engenharia  
Licenciatura em Engenharia Informática

## **Unidade Curricular de Redes de Computadores**

Ano Letivo de 2023/2024

# **Trabalho Prático LI3**

## **Fase 2**

## **Grupo 35**

Flávio David Rodrigues Sousa (A100715)  
André Miguel Alves de Carvalho (A100818)  
Sandro José Rodrigues Coelho (A105672)

25 de janeiro de 2024

**L**  
**I3**

# Índice

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Desenvolvimento</b>	<b>2</b>
2.1	Estruturas de dados utilizadas	2
<b>3</b>	<b>Módulos</b>	<b>4</b>
3.1	Structs	4
3.2	Utilitys	6
3.3	Parser	6
3.4	Catalogs	7
3.5	Menu	8
3.6	Modes	8
3.7	Interpreter	9
3.8	Queries	9
<b>4</b>	<b>Queries</b>	<b>11</b>
4.1	Query 1	11
4.2	Query 2	11
4.3	Query 3	12
4.4	Query 4	12
4.5	Query 5	12
4.6	Query 6	13
4.7	Query 7	13
4.8	Query 8	13
4.9	Query 9	14
4.10	Query 10	14
<b>5</b>	<b>Tempo de execução e mediana</b>	<b>15</b>
<b>6</b>	<b>Paginação</b>	<b>20</b>
<b>7</b>	<b>Menu Interativo</b>	<b>21</b>
<b>8</b>	<b>MakeFile</b>	<b>25</b>
<b>9</b>	<b>Dificuldades Encontradas</b>	<b>26</b>
<b>10</b>	<b>Conclusão</b>	<b>27</b>

# 1 Introdução

Este documento representa a fase 2 do projeto da unidade curricular de LI3 (Laboratórios de Informática III) do ano letivo 2023/24. Nesta fase, foi avaliada a eficiência e a estruturação do projeto, destacando certos elementos, tal como a modularização, encapsulamento, reutilização de código, o desempenho geral do programa e a adequação das estruturas e algoritmos usados pelo mesmo.

Será também aproveitado o uso da batch assim como a criação e uso de um modo interativo e mais "*user-friendly*".

As *queries* resolvidas serão executadas por comandos pré-definidos pelos docentes da unidade curricular, de modo a poder testar todos os aspetos do código presente, armazenando por fim os resultados de cada comando em ficheiros de texto.

Por fim, estes ficheiros são testados na plataforma online da unidade curricular (Plataforma de Teste), de modo a descobrir se existe algum erro com as *queries*, ou com o modo com que entregam os seus outputs.

## 2 Desenvolvimento

Para realizar o trabalho de forma mais eficaz, decidimos dividir o código, pois seria mais fácil de controlar o que estaria a ser realizado sobre cada parte do projeto.

Com essa ideia em mente, foi possível dividir o desenvolvimento do *parser* para um aluno em particular enquanto ou outros dois alunos realizavam as *queries* escolhidas.

Após abordarmos a fase de planeamento do trabalho, começamos com a base do mesmo, verificando a integridade de todos os dados que seriam manipulados pelo programa, de modo a que seja possível validar cada documento.

Com isso, foi implementado o *parser*, sendo uma das partes mais importantes do programa, permitindo a análise e entendimento dos dados fornecidos. Esta etapa envolveu a criação eficiente de um algoritmo para processar e interpretar todos os dados que o atravessam, garantindo assim uma transição rápida e eficaz.

Em seguida, foi necessário criar as funções responsáveis por manipular as *queries* criadas, de modo a que fosse possível obter os resultados necessários para os comandos fornecidos. Durante este processo, percebemos a necessidade de criar estruturas adicionais para otimizar certas *queries*. Neste caso, foram usadas árvores e outras estruturas que serão abordadas neste relatório.

Ao mesmo tempo, incorporamos um modo de execução *batch*, facilitando a experiência dos utilizadores.

Por fim, terminamos o desenvolvimento das restantes *queries* bem como o modo interativo para com o utilizador. Esta é a etapa final, conseguindo trazer todo o potencial existente das *queries* para o programa. É de notar que, também foi desenvolvido um modelo de paginação para o modo interativo.

### 2.1 Estruturas de dados utilizadas

Para a realização deste projeto decidimos utilizar quatro tipos de estruturas base, sendo elas: *HashTables*, *Árvores*, *Listas Ligadas* e *Arrays*, cuja motivação de utilização será abordada e justificada ao longo do relatório.

De forma breve, as *Hashtables* armazenam os dados que são *parsed* e são as principais estru-

turas utilizadas. O principal motivo do seu uso deve-se ao fator tempo e ao fator eficiência, uma vez que, esta estrutura tem um tempo de procura constante para qualquer *key* solicitada. Cada *key* aponta para uma *struct* que contém todas as informações necessárias.

De mesma forma, as árvores são utilizadas nas *queries* 2, 4 e 5, uma vez que, surgiu a necessidade de organizar valores consoante datas e, acreditamos que, esta estrutura será mais eficiente nesta fase.

Por último, as *Listas Ligadas* apenas são utilizadas na *query* 2, uma vez que, era pedido que, caso nenhum parâmetro que especificasse o tipo fosse passado à função esta teria de apresentar tanto os voos como as reservas de um utilizador em ordem, então para facilitar a comparação entre as datas dos voos e das reservas, as árvores foram convertidas em listas ligadas.

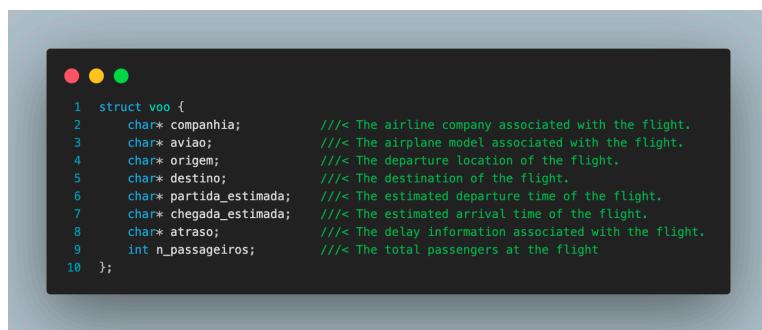
# 3 Módulos

Um dos objetivos deste projeto é promover o encapsulamento e a modularização do código, resultando na criação de vários módulos. Esses módulos foram desenvolvidos para permitir o uso das funções em qualquer consulta que necessite de acesso a essas funcionalidades. Além disso, foram gerados arquivos .h associados aos arquivos .c correspondentes, os quais apenas invocam as funções contidas nesses arquivos .c. Essa abordagem visa evitar o acesso direto às informações por fatores externos.

## 3.1 Structs

Relativamente às estruturas, as bases para armazenamento da informação de cada linha individualizada, desenvolvemos quatro fundamentais, sendo elas:

- **Voo**(Figura 3.1): Esta inclui informação sobre a companhia, o avião, a origem, o destino, a partida estimada, a chegada estimada, o atraso de um voo, que é calculada automaticamente por uma função, e um número de passageiros que é inicializado a 0 e calcula posteriormente.



```
1 struct voo {
2     char* companhia;           /////< The airline company associated with the flight.
3     char* aviao;               /////< The airplane model associated with the flight.
4     char* origem;              /////< The departure location of the flight.
5     char* destino;             /////< The destination of the flight.
6     char* partida_estimada;   /////< The estimated departure time of the flight.
7     char* chegada_estimada;   /////< The estimated arrival time of the flight.
8     char* atraso;              /////< The delay information associated with the flight.
9     int n_passageiros;        /////< The total passengers at the flight
10    };

```

Figura 3.1: Estrutura Voo

- Passageiro(Figura 3.2): Esta inclui informação sobre o passageiro.



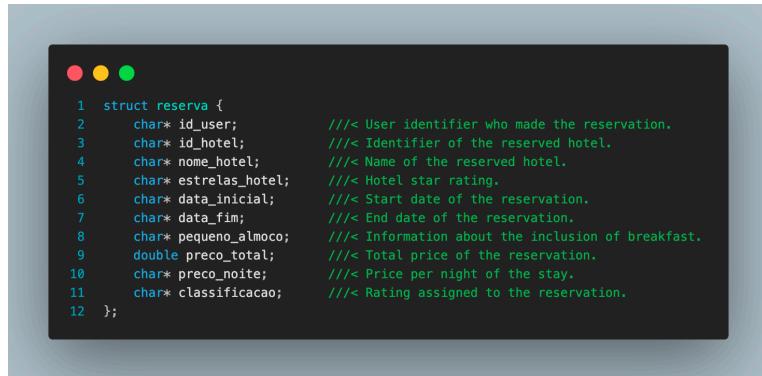
```

1 struct passageiro {
2     char* data;           ///< Passenger data
3     struct passageiro* next;    ///< Next Passenger Struct
4 };

```

Figura 3.2: Estrutura Passageiro

- Reserva(Figura 3.3): Esta inclui informação sobre o *ID* do utilizador e do hotel, nome do hotel, estrelas, data inicial e final de uma reserva, se a reserva contém pequeno-almoço incluído, o preço total de uma reserva que é calcula automaticamente na altura do parsing, o preço de uma reserva por noite e a classificação entregue para as reservas.



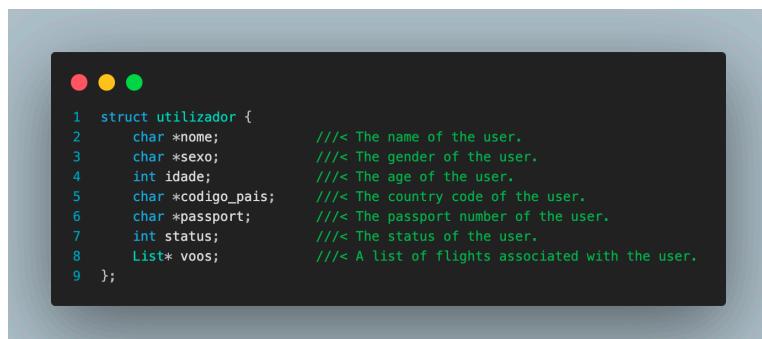
```

1 struct reserva {
2     char* id_user;        ///< User identifier who made the reservation.
3     char* id_hotel;       ///< Identifier of the reserved hotel.
4     char* nome_hotel;     ///< Name of the reserved hotel.
5     char* estrelas_hotel; ///< Hotel star rating.
6     char* data_inicial;  ///< Start date of the reservation.
7     char* data_fim;       ///< End date of the reservation.
8     char* pequeno_almoco; ///< Information about the inclusion of breakfast.
9     double preco_total;  ///< Total price of the reservation.
10    char* preco_noite;   ///< Price per night of the stay.
11    char* classificacao; ///< Rating assigned to the reservation.
12 };

```

Figura 3.3: Estrutura Reserva

- Utilizador(Figura 3.4): Esta inclui informação sobre o nome do utilizador, o sexo, a idade que é calculada automaticamente na altura do parsing, o código do país, o passaporte e o status de atividade do utilizador.



```

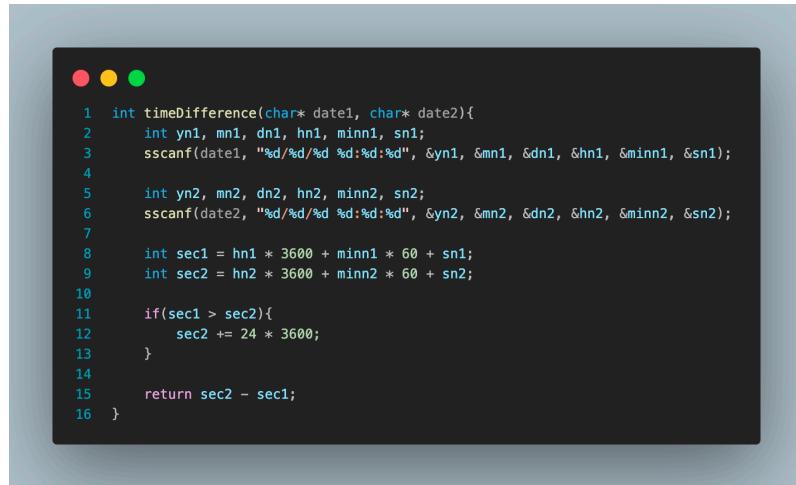
1 struct utilizador {
2     char *nome;           ///< The name of the user.
3     char *sexo;           ///< The gender of the user.
4     int idade;            ///< The age of the user.
5     char *codigo_pais;   ///< The country code of the user.
6     char *passport;      ///< The passport number of the user.
7     int status;           ///< The status of the user.
8     List* voos;          ///< A list of flights associated with the user.
9 };

```

Figura 3.4: Estrutura Utilizador

## 3.2 Utilities

Este módulo define algumas funções gerais que são usadas em diferentes partes do programa e não são necessariamente classificadas como função específica de algum outro módulo. Neste caso, é servido de exemplo funções responsáveis por calcular diferenças de datas e horas tal como a função presente na Figura 3.5.



```
1 int timeDifference(char* date1, char* date2){  
2     int yn1, mn1, dn1, hn1, minn1, sn1;  
3     sscanf(date1, "%d/%d/%d %d:%d:%d", &yn1, &mn1, &dn1, &hn1, &minn1, &sn1);  
4  
5     int yn2, mn2, dn2, hn2, minn2, sn2;  
6     sscanf(date2, "%d/%d/%d %d:%d:%d", &yn2, &mn2, &dn2, &hn2, &minn2, &sn2);  
7  
8     int sec1 = hn1 * 3600 + minn1 * 60 + sn1;  
9     int sec2 = hn2 * 3600 + minn2 * 60 + sn2;  
10  
11    if(sec1 > sec2){  
12        sec2 += 24 * 3600;  
13    }  
14  
15    return sec2 - sec1;  
16 }
```

Figura 3.5: Função timeDifference

## 3.3 Parser

Com o objetivo de garantir a versatilidade do *parser*, isto é, permitir que seja uma função capaz de operar independente do contexto, senti-mos a necessidade de o tornar uma função de ordem superior, o que garante o requisito anterior. Sendo assim, o *parser* é responsável por processar cada linha de um arquivo dado e enviar, para uma determinada função, responsável por operar com a linha anteriormente processada. O nosso *parser* foi então elaborado com os seguintes atributos presentes na Figura 3.6.



```
1 void parser(char *nome, int tamanho, void (*function)(char *nome1, char **tokens, char *linha, GHashTable *hash_recebe, GHashTable *hash1), GHashTable *hash, GHashTable *hash_01);  
2
```

Figura 3.6: Parser

## 3.4 Catalogs

Com objetivo de gerir e armazenar a nossa informação de forma mais eficiente, criamos um catálogo responsável por cada estrutura anteriormente explorada.

Existem assim quatro catálogos essenciais, em que cada um possui uma função capaz de criar o próprio catálogo. Sendo assim, esses catálogos são os seguintes:

- catalog\_users(Figura 3.7): Contém uma *HashTable* onde a key é o *ID* do utilizador, e o value é um apontador para uma estrutura *utilizador* com informação do respetivo utilizador.



```
● ● ●
1 struct catalog_users {
2     GHashTable* users_hash; // Hash table containing user information.
3 };
```

Figura 3.7: Catalog\_Users

- catalog\_reservations(Figura 3.8): Contém uma *HashTable* onde a key é o *ID* da reserva, e o value é um apontador para uma estrutura *reserva* com informação da respetiva reserva.



```
● ● ●
1 struct catalog_reservations {
2     GHashTable* reservations_hash; // Hash table containing reservation information.
3 };
```

Figura 3.8: Catalog\_Reservations

- catalog\_flights(Figura 3.9): Contém uma *HashTable* onde a key é o *ID* do voo, e o value é um apontador para uma estrutura *voo* com informação do respetivo voo.



```
● ● ●
1 struct catalog_flights {
2     GHashTable* flights_hash; // Hash table containing flight information.
3 };
```

Figura 3.9: Catalog\_Flights

- catalog\_passengers: Este catálogo apenas existe para que sejam atualizados as informações sobre os passageiros para uso de outros catálogos.

Por fim, criamos um catálogo capaz de criar e gerir os catálogos existentes, conseguindo assim um controlo maior sobre a informação proveniente dos mesmos. (Figura 3.10)



```

1 struct catalogs {
2     Catalog_users* u_catalog;
3     Catalog_flights* f_catalog;
4     Catalog_reservations* r_catalog;
5 };

```

Figura 3.10: Catalog

## 3.5 Menu

Este módulo implementa um simples menu capaz de gerir o programa. Este exibe ao utilizador as informações disponíveis e permite para o mesmo várias opções, que mudam baseado no input do utilizador. Será demonstrado e explorado na Secção 7 deste relatório.

## 3.6 Modes

Neste módulo, são apresentados dois modos de tratamento de pedidos, que são os seguintes:

- Batch(Figura 3.11): Este modo recebe os catálogos disponíveis no programa, assim como o ficheiro de input disponibilizado, entregando essa informação ao interpretador.



Figura 3.11: Batch

- Interactive(Figura 3.12): Este modo recebe os catálogos disponíveis no programa e espera os inputs entregues pelo próprio menu, analisando esse input e entregando o mesmo para o interpretador.



Figura 3.12: Interactive

## 3.7 Interpreter

Este módulo é responsável por, dado um input, determinar e enviar o pedido para a *query* correta com os respetivos argumentos solicitados. Além disso, foi implementada uma componente versátil no *interpreter*, isto é, tanto em modo *batch* como em modo *interactive* este módulo é invocado e responsável pelo direcionamento do pedido.

## 3.8 Queries

A implementação das *queries* foi uma parte do projeto que necessitou de atenção extra, devido à complexidade que a maioria delas tinha, sendo essenciais para o funcionamento do programa, estas representam os pedidos dos utilizadores, obtendo assim as informações pedidas. Com isto, foi possível verificar a importância dos pedidos, assim como a importância da estratégia

usada para que o problema pudesse ser resolvido da forma mais eficaz, fornecendo resultados coerentes e rigorosos.

# 4 Queries

## 4.1 Query 1

A primeira *query* consiste em "*Listar o resumo de um utilizador, voo, ou reserva, consoante o identificador recebido por argumento.*", este entrega as informações destinadas ao argumento entregue pelo utilizador, dependendo do identificador recebido.

Se o identificador começar com "0000", a função irá trata-lo como um voo, entregando as informações relacionadas com ele, tal como a companhia, avião, origem, destino, hora de partida estimada, hora de chegada estimada, número de passageiros e o tempo de atraso.

Se o identificador começar com "Book", a função irá trata-lo como uma reserva, entregando as suas informações, tal como ID do hotel, nome do hotel, quantidade das estrelas do hotel, data de começo, data de fim da reserva, se a reserva inclui pequeno-almoço, quantidade de noites passadas e preço total da mesma.

Se o identificador não for nenhum dos anteriores, será tratado como um utilizador, entregando assim as informações referentes ao mesmo, tal como, nome, sexo, idade, código do país, passaporte, numero de voos, numero de reservas e total gasto.

Por fim, a função verifica se *type* da função, dependendo do seu valor, as funções serão escritas no formato correto para o ficheiro de saída. Além disso, é alocada memória sempre que é armazenada alguma informação, logo, é necessário libertar essa memória de modo a não haver *memory leaks* no programa.

## 4.2 Query 2

A segunda *query*, consiste em "*Listar os voos ou reservas de um utilizador, se o segundo argumento for flights ou reservations, respetivamente, ordenados por data (da mais recente para a mais antiga).* Caso não seja fornecido um segundo argumento, apresentar voos e reservas, juntamente com o tipo."

Esta função utiliza funções auxiliares para criar e organizar as listas de voos e reservas, imprimindo os seus resultados no formato desejado para o ficheiro de saída.

Com as funções auxiliares criadas, é possível identificar os voos e reservas que se encontram associados ao utilizador, permitindo assim a criação de árvores ordenadas através da data e do *ID*.

Com isto, são usadas outras funções de modo a organizar as árvores por data, realizando a comparação entre as mesmas.

### 4.3 Query 3

A terceira *query*, consiste em "*Apresentar a classificação média de um hotel, a partir do seu identificador*".

A função percorre as reservas, e verifica se pertencem ao hotel desejado, se assim for, soma as classificações e a quantidade de classificações, criando assim a média da mesma no fim.

Caso não haja nenhuma reserva para o hotel, é escrita uma linha vazia no ficheiro de saída, caso contrário, calcula a média e escreve-a no ficheiro de saída.

### 4.4 Query 4

A quarta *query*, consiste em "*Listar as reservas de um hotel, ordenadas por data de início (da mais recente para a mais antiga). Caso duas reservas tenham a mesma data, deve ser usado o identificador da reserva como critério de desempate (de forma crescente)*".

A função utiliza uma função auxiliar de modo a construir uma árvore capaz de armazenar as reservas associadas a um certo hotel.

As várias funções auxiliares presentes, desempenham os seus papéis para a funcionalidade da árvore, tal como a criação, manipulação, libertação, contagem e ordenação das árvores criadas.

No fim, são revistas todas as reservas presentes, identificando aquelas associadas ao hotel que se encontra escolhido pelo utilizador, ordenando-as assim por data e *ID*, imprimindo as informações no formato desejado.

### 4.5 Query 5

A quinta *query*, consiste em "*Listar os voos com origem num dado aeroporto, entre duas datas, ordenados por data de partida estimada (da mais antiga para a mais recente)*".

Para situações em que dois voos possuem a mesma data, a ordenação deve ser feita através

do *ID* de voo em ordem crescente.

São usadas funções auxiliares, de modo a construir uma árvore que contém todos os voos que respeitam os critérios estabelecidos pelo utilizador, nomeadamente a origem e as datas, com isso, as funções permitem identificar aqueles voos com a origem desejada e os que se encontram no intervalo de datas especificado pelo utilizador.

Em seguida, a árvore é percorrida pela ordem estabelecida e imprime as informações no formato desejado.

As funções auxiliares presentes, desempenham os seus devidos papéis para a funcionalidade das árvores, tal como a *query* anterior, resultando na criação, manipulação, libertação, contagem e ordenação das árvores criadas.

## 4.6 Query 6

A sexta *Query* tem como objetivo listar os principais N aeroportos com o maior número de passageiros para um ano específico.

Esta utiliza uma *HashTable* de voos de modo a calcular a contagem de passageiros de cada aeroporto durante o ano especificado. Essas informações são então armazenadas numa estrutura chamada "Airport". De seguida, a estrutura será ordenada de acordo com o número de passageiros, usando assim o nome do aeroporto como um critério de desempate caso não seja possível ordenar por número de passageiros. Os resultados são então formatados e impressos da maneira que assim se deseja.

## 4.7 Query 7

Não foi implementada.

## 4.8 Query 8

A oitava *query*, consiste em "*Apresentar a receita total de um hotel a partir de um identificador, entre um período de tempo especificado, considerando o preço por noite de cada reserva, de todas as noites de reservas entre as datas entregues.*".

A função itera a hash table de reservations, e para cada reserva presente, verifica se as mesmas se encontram no período de tempo especificado, verificando se o *ID* do Hotel coincide com o *ID* que foi dado como *input*. Se ambas condições forem verificadas, a função calcula a *revenue* do Hotel.

São usadas várias funções auxiliares de modo a que seja possível extrair a informação necessária sobre as reservas, obtendo o preço total da reserva para o período de tempo especificado e uma função para comparar as datas de modo a perceber a quantidade de noites para realizar a *revenue* total da reserva atual.

## 4.9 Query 9

A nona *query*, consiste em "*Listar todos os utilizadores cujo nome começa com o prefixo passado por argumento, ordenados por nome de forma crescente.*". Caso ambos nomes sejam iguais, é organizado através do *ID* do utilizador.

A função compara ambos nomes de modo a organizar ambos através de ordenação sensível a cada idioma. Durante a execução, a função compara os nomes através do prefixo de cada nome, se ambos prefixos forem iguais, é verificada a letra seguinte, até achar uma letra que seja diferente da que está a ser comparada, ordenando assim os nomes, caso contrário, se os nomes forem iguais, é necessário comparar os *IDs* dos utilizadores e ordenar os nomes através deles.

Por fim, os resultados são ordenados usando a função *qsort* e a função criada para comparação personalizada, formatando e escrevendo os resultados no ficheiro de saída.

## 4.10 Query 10

Não foi implementada.

# 5 Tempo de execução e mediana

No decorrer da nossa avaliação do desempenho do programa, realizamos testes abrangentes num MacBook Pro equipado com o processador M1 da Apple, contando com 8GB de memória RAM e operando no sistema operativo Sonoma na versão 14.2.1. Foi também utilizada o programa-testes presente no nosso projeto para avaliação das três situações de seguida exploradas:

- **Situação 1:** (Figura 5.1) Utilizamos o dataset small com o respetivo input small

```
Command 1 - Passed the test in 0.00427100 seconds!
Command 2 - Passed the test in 0.00004900 seconds!
Command 3 - Passed the test in 0.00377800 seconds!
Command 4 - Passed the test in 0.00004400 seconds!
Command 5 - Passed the test in 0.00003800 seconds!
Command 6 - Passed the test in 0.00003800 seconds!
Command 7 - Passed the test in 0.00003800 seconds!
Command 8 - Passed the test in 0.00006000 seconds!
Command 9 - Passed the test in 0.00005300 seconds!
Command 10 - Passed the test in 0.00005300 seconds!
Command 11 - Passed the test in 0.00003900 seconds!
Command 12 - Passed the test in 0.00004000 seconds!
Command 13 - Passed the test in 0.00006000 seconds!
Command 14 - Passed the test in 0.00005300 seconds!
Command 15 - Passed the test in 0.00005700 seconds!
Command 16 - Passed the test in 0.00004000 seconds!
Command 17 - Passed the test in 0.00004300 seconds!
Command 18 - Passed the test in 0.00003500 seconds!
Command 19 - Passed the test in 0.00004000 seconds!
Command 20 - Passed the test in 0.00003900 seconds!
Command 21 - Passed the test in 0.00194400 seconds!
Command 22 - Passed the test in 0.00004300 seconds!
Command 23 - Passed the test in 0.00196100 seconds!
Command 24 - Passed the test in 0.00005300 seconds!
Command 25 - Passed the test in 0.00216000 seconds!
Command 26 - Passed the test in 0.00217300 seconds!
Command 27 - Passed the test in 0.00220100 seconds!
Command 28 - Passed the test in 0.01342300 seconds!
Command 29 - Passed the test in 0.01273000 seconds!
Command 30 - Passed the test in 0.01249100 seconds!
Command 31 - Passed the test in 0.00016100 seconds!
Command 32 - Passed the test in 0.00016300 seconds!
Command 33 - Passed the test in 0.00161000 seconds!
Command 34 - Did not pass the test because of an error in line 1
Command 35 - Did not pass the test because of an error in line 1
Command 36 - Did not pass the test because of an error in line 1
Command 39 - Did not pass the test because of an error in line 1
Command 40 - Did not pass the test because of an error in line 1
Command 41 - Passed the test in 0.00251900 seconds!
Command 42 - Passed the test in 0.00245600 seconds!
Command 43 - Passed the test in 0.00236100 seconds!
Command 45 - Passed the test in 0.00068600 seconds!
Command 51 - Passed the test in 0.000378100 seconds!
Command 52 - Passed the test in 0.000378100 seconds!
Command 53 - Passed the test in 0.00382600 seconds!
Command 54 - Passed the test in 0.00004500 seconds!
Command 55 - Passed the test in 0.00004600 seconds!
Command 56 - Passed the test in 0.00004900 seconds!
Command 57 - Passed the test in 0.00004000 seconds!
Command 58 - Passed the test in 0.00008100 seconds!
Command 59 - Passed the test in 0.00010100 seconds!
Command 60 - Passed the test in 0.00006300 seconds!
Command 61 - Passed the test in 0.00003800 seconds!
Command 62 - Passed the test in 0.00004300 seconds!
Command 63 - Passed the test in 0.00003800 seconds!
Command 64 - Passed the test in 0.00006500 seconds!
Command 65 - Passed the test in 0.00007300 seconds!
Command 66 - Passed the test in 0.00004500 seconds!
Command 67 - Passed the test in 0.00004400 seconds!
Command 68 - Passed the test in 0.00004200 seconds!
Command 69 - Passed the test in 0.00196100 seconds!
Command 70 - Passed the test in 0.00008500 seconds!
Command 71 - Passed the test in 0.00193300 seconds!
Command 72 - Passed the test in 0.00004600 seconds!
Command 73 - Passed the test in 0.00191100 seconds!
Command 74 - Passed the test in 0.00004300 seconds!
Command 75 - Passed the test in 0.00004300 seconds!
Command 76 - Passed the test in 0.00215200 seconds!
Command 77 - Passed the test in 0.00239700 seconds!
Command 78 - Passed the test in 0.01333000 seconds!
Command 79 - Passed the test in 0.01223600 seconds!
Command 80 - Passed the test in 0.01317700 seconds!
Command 81 - Passed the test in 0.000080100 seconds!
Command 82 - Passed the test in 0.00019000 seconds!
Command 83 - Passed the test in 0.00177700 seconds!
Command 84 - Did not pass the test because of an error in line 3
Command 85 - Did not pass the test because of an error in line 3
Command 86 - Did not pass the test because of an error in line 3
Command 87 - Did not pass the test because of an error in line 2
Command 89 - Did not pass the test because of an error in line 2
Command 91 - Passed the test in 0.00238200 seconds!
Command 92 - Passed the test in 0.00235300 seconds!
Command 93 - Passed the test in 0.00231400 seconds!
Command 95 - Passed the test in 0.00061200 seconds!
CPU Time: 0.49430400 seconds
Memory usage: 33783808 KB
```

Figura 5.1: Resultados da Situação 1

Nesta situação apuramos os seguintes resultados:

Totalidade de Comandos:95

Nº Comandos Executados com sucesso: 74

Média de tempo por querie:0.00520 segundos

Tempo de CPU: 0,5 segundos

Memória Ocupada: 1.34GB

## Motivo de erros: Queries sem implementação e erros na query 8

- **Situação 2:** (Figura 5.2) Utilizamos o dataset large com o respetivo input large

```

Command 1 - Passed the test in 0.71444300 seconds!
Command 2 - Did not pass the test because of an error in line 1
Command 3 - Passed the test in 0.00007500 seconds!
Command 4 - Did not pass the test because of an error in line 1
Command 5 - Passed the test in 0.00013000 seconds!
Command 6 - Passed the test in 0.00009900 seconds!
Command 7 - Passed the test in 0.00009900 seconds!
Command 8 - Did not pass the test because of an error in line 1
Command 9 - Passed the test in 1.45538000 seconds!
Command 10 - Passed the test in 0.00008000 seconds!
Command 11 - Did not pass the test because of an error in line 6
Command 12 - Did not pass the test because of an error in line 1
Command 13 - Passed the test in 0.00011600 seconds!
Command 14 - Did not pass the test because of an error in line 1
Command 15 - Passed the test in 0.68169800 seconds!
Command 16 - Passed the test in 0.00006000 seconds!
Command 17 - Passed the test in 0.00018100 seconds!
Command 18 - Passed the test in 0.00009900 seconds!
Command 19 - Did not pass the test because of an error in line 1
Command 20 - Passed the test in 0.00017700 seconds!
Command 21 - Passed the test in 0.66967400 seconds!
Command 22 - Passed the test in 0.69068900 seconds!
Command 23 - Passed the test in 0.00013300 seconds!
Command 24 - Did not pass the test because of an error in line 1
Command 25 - Passed the test in 0.00010100 seconds!
Command 26 - Passed the test in 0.00012900 seconds!
Command 27 - Did not pass the test because of an error in line 1
Command 28 - Passed the test in 0.00008700 seconds!
Command 29 - Did not pass the test because of an error in line 1
Command 30 - Passed the test in 1.40652500 seconds!
Command 31 - Did not pass the test because of an error in line 1
Command 32 - Did not pass the test because of an error in line 1
Command 33 - Passed the test in 1.43201100 seconds!
Command 34 - Passed the test in 0.60785900 seconds!
Command 35 - Did not pass the test because of an error in line 1
Command 36 - Passed the test in 0.00011700 seconds!
Command 37 - Passed the test in 0.00009900 seconds!
Command 38 - Passed the test in 0.00006500 seconds!
Command 39 - Did not pass the test because of an error in line 2
Command 40 - Did not pass the test because of an error in line 1
Command 42 - Did not pass the test because of an error in line 17
Command 43 - Passed the test in 0.66820600 seconds!
Command 44 - Did not pass the test because of an error in line 1
Command 45 - Did not pass the test because of an error in line 6
Command 46 - Passed the test in 0.00011200 seconds!
Command 47 - Did not pass the test because of an error in line 1
Command 48 - Passed the test in 0.00013700 seconds!
Command 49 - Passed the test in 0.31066400 seconds!
Command 50 - Passed the test in 0.00008300 seconds!
Erro ao abrir o ficheiro result: No such file or directory
Command 51 - Did not pass the test because of an error in line -1
Command 52 - Did not pass the test because of an error in line 1
Command 53 - Passed the test in 0.00009700 seconds!
Command 54 - Passed the test in 0.00008700 seconds!
Command 55 - Passed the test in 0.00007500 seconds!
Command 56 - Passed the test in 0.00007600 seconds!
Command 57 - Passed the test in 0.64756100 seconds!
Command 58 - Passed the test in 0.00012300 seconds!
Command 59 - Did not pass the test because of an error in line 1
Command 60 - Passed the test in 1.35955300 seconds!
Command 61 - Did not pass the test because of an error in line 5
Command 62 - Did not pass the test because of an error in line 1
Command 63 - Did not pass the test because of an error in line 1
Command 64 - Passed the test in 0.00009900 seconds!
Command 65 - Passed the test in 1.30044380 seconds!
Command 66 - Passed the test in 1.31703700 seconds!
Command 67 - Passed the test in 0.00227900 seconds!
Command 68 - Passed the test in 1.40050300 seconds!
Command 69 - Passed the test in 0.68563600 seconds!
Command 70 - Passed the test in 1.38733200 seconds!
Command 71 - Passed the test in 0.67741100 seconds!
Command 72 - Passed the test in 1.30404300 seconds!
Command 73 - Passed the test in 0.00010900 seconds!
Erro ao abrir o ficheiro result: No such file or directory
Command 74 - Did not pass the test because of an error in line 1
Command 75 - Did not pass the test because of an error in line 1
Command 76 - Did not pass the test because of an error in line 1
Command 77 - Did not pass the test because of an error in line 1
Command 78 - Did not pass the test because of an error in line 1
Command 79 - Did not pass the test because of an error in line 1
Command 80 - Passed the test in 0.00010800 seconds!
Command 81 - Passed the test in 0.00008900 seconds!
Command 82 - Did not pass the test because of an error in line 1
Command 83 - Passed the test in 0.00008900 seconds!
Command 84 - Passed the test in 1.29866300 seconds!
Command 85 - Did not pass the test because of an error in line 1
Command 86 - Did not pass the test because of an error in line 7
Command 88 - Did not pass the test because of an error in line 1
Command 89 - Passed the test in 0.00008600 seconds!
Command 90 - Passed the test in 1.34502200 seconds!
Command 91 - Passed the test in 0.28076200 seconds!
Command 93 - Passed the test in 0.24085300 seconds!
Command 94 - Passed the test in 0.00018600 seconds!
Command 95 - Did not pass the test because of an error in line 16
Command 96 - Passed the test in 0.00010500 seconds!
Command 97 - Passed the test in 2.92296300 seconds!
Command 98 - Did not pass the test because of an error in line 1
Command 99 - Passed the test in 0.00009200 seconds!

```

```

Command 100 - Passed the test in 0.00008700 seconds!
Command 101 - Passed the test in 0.00008700 seconds!
Command 102 - Passed the test in 0.65585700 seconds!
Command 103 - Did not pass the test because of an error in line 4
Erro ao abrir o ficheiro result: No such file or directory
Command 104 - Did not pass the test because of an error in line -1
Command 105 - Passed the test in 0.00013800 seconds!
Command 106 - Passed the test in 0.00010000 seconds!
Erro ao abrir o ficheiro result: No such file or directory
Command 107 - Did not pass the test because of an error in line -1
Command 108 - Passed the test in 0.00015300 seconds!
Erro ao abrir o ficheiro result: No such file or directory
Command 109 - Did not pass the test because of an error in line -1
Command 110 - Did not pass the test because of an error in line 4
Command 112 - Did not pass the test because of an error in line 1
Command 113 - Did not pass the test because of an error in line 1
Command 114 - Passed the test in 0.00013800 seconds!
Command 115 - Did not pass the test because of an error in line 3
Command 116 - Passed the test in 0.00010100 seconds!
Command 117 - Passed the test in 0.00009400 seconds!
Command 119 - Passed the test in 0.00007300 seconds!
Command 120 - Passed the test in 1.44010500 seconds!
Command 121 - Passed the test in 0.00010200 seconds!
Command 122 - Did not pass the test because of an error in line 1
Command 123 - Passed the test in 0.00013800 seconds!
Command 124 - Did not pass the test because of an error in line 1
Command 125 - Passed the test in 0.68496200 seconds!
Erro ao abrir o ficheiro result: No such file or directory
Command 126 - Did not pass the test because of an error in line -1
Command 127 - Did not pass the test because of an error in line 1
Command 128 - Passed the test in 0.00012800 seconds!
Command 129 - Passed the test in 1.36416700 seconds!
Command 131 - Did not pass the test because of an error in line 1
Command 132 - Passed the test in 0.00015300 seconds!
Command 133 - Did not pass the test because of an error in line 1
Command 134 - Did not pass the test because of an error in line 1
Command 135 - Passed the test in 0.00012800 seconds!
Command 136 - Passed the test in 2.23103500 seconds!
Command 137 - Passed the test in 0.60099900 seconds!
Command 138 - Passed the test in 0.70579300 seconds!
Command 139 - Passed the test in 0.70107200 seconds!
Command 140 - Did not pass the test because of an error in line 1
Command 141 - Passed the test in 0.65701600 seconds!
Command 142 - Passed the test in 1.31543900 seconds!
Command 143 - Passed the test in 0.68072800 seconds!
Command 144 - Did not pass the test because of an error in line 1
Command 145 - Passed the test in 0.65804500 seconds!
Command 146 - Passed the test in 1.37221300 seconds!

Command 147 - Passed the test in 0.65199800 seconds!
Command 148 - Passed the test in 0.66540700 seconds!
Command 149 - Did not pass the test because of an error in line 1
Command 150 - Passed the test in 0.00009800 seconds!
Command 151 - Did not pass the test because of an error in line 1
Command 152 - Passed the test in 0.00016500 seconds!
Command 153 - Did not pass the test because of an error in line 1
Erro ao abrir o ficheiro result: No such file or directory
Command 154 - Did not pass the test because of an error in line -1
Command 155 - Did not pass the test because of an error in line 6
Command 156 - Passed the test in 0.00010300 seconds!
Command 157 - Passed the test in 0.67933900 seconds!
Command 158 - Did not pass the test because of an error in line 1
Command 159 - Passed the test in 0.69182100 seconds!
Command 160 - Passed the test in 0.28069000 seconds!
Command 161 - Passed the test in 1.38019900 seconds!
Command 162 - Passed the test in 0.00010400 seconds!
Command 163 - Passed the test in 0.00010300 seconds!
Command 164 - Did not pass the test because of an error in line 1
Command 165 - Passed the test because of an error in line 1
Command 166 - Passed the test in 0.00016400 seconds!
Command 167 - Passed the test in 0.00008800 seconds!
Command 168 - Passed the test in 0.00033300 seconds!
Command 169 - Passed the test in 0.00008500 seconds!
Command 170 - Passed the test in 0.68711900 seconds!
Command 171 - Passed the test in 0.00012200 seconds!
Command 172 - Did not pass the test because of an error in line 1
Command 173 - Did not pass the test because of an error in line 1
Command 174 - Did not pass the test because of an error in line 1
Command 175 - Passed the test in 0.63982400 seconds!
Command 176 - Passed the test in 1.33358800 seconds!
Command 177 - Passed the test in 0.00011700 seconds!
Command 178 - Passed the test in 0.00021200 seconds!
Command 179 - Passed the test in 0.67842300 seconds!
Command 180 - Passed the test in 0.66173500 seconds!
Command 181 - Passed the test in 1.32423400 seconds!
Command 182 - Passed the test in 0.00012200 seconds!
Command 183 - Did not pass the test because of an error in line 1
Command 184 - Passed the test in 1.36245700 seconds!
Command 185 - Passed the test in 0.00012300 seconds!
Command 186 - Did not pass the test because of an error in line 1
Command 187 - Passed the test in 0.65778400 seconds!
Command 188 - Passed the test in 0.28833300 seconds!
Command 189 - Did not pass the test because of an error in line 1
Command 190 - Did not pass the test because of an error in line 1
Erro ao abrir o ficheiro result: No such file or directory
Command 191 - Did not pass the test because of an error in line -1
Command 192 - Passed the test in 0.00013800 seconds!

```

Figura 5.2: Resultados da Situação 2

Nesta situação apuramos os seguintes resultados:

Totalidade de Comandos:500

Nº Comandos Executados com sucesso: 128

Média de tempo por querie:0.38728 segundos

Tempo de CPU: Sem valor (segmentation fault)

Memória Ocupada: Sem valor (segmentation fault)

Motivo de erros: Foi detetado um *segmentation fault* no comando 196 que bloqueou todo o programa; erros semelhantes aos da situação 1

- **Situação 3:** (Figura 5.3) Utilizamos o dataset large com o input small

```

Command 1 - Passed the test in 0.00011200 seconds!
Command 2 - Passed the test in 0.00004200 seconds!
Command 3 - Passed the test in 0.00004000 seconds!
Command 4 - Passed the test in 0.00003900 seconds!
Command 5 - Passed the test in 0.00003800 seconds!
Command 6 - Passed the test in 0.00004000 seconds!
Command 7 - Passed the test in 1.61637500 seconds!
Command 8 - Did not pass the test because of an error in line 1
Command 9 - Did not pass the test because of an error in line 1
Command 10 - Did not pass the test because of an error in line 1
Command 11 - Passed the test in 0.00003300 seconds!
Command 12 - Passed the test in 0.00004700 seconds!
Command 13 - Did not pass the test because of an error in line 1
Command 14 - Did not pass the test because of an error in line 1
Command 15 - Did not pass the test because of an error in line 1
Command 16 - Passed the test in 0.00005500 seconds!
Command 17 - Passed the test in 0.00006500 seconds!
Command 18 - Passed the test in 0.00006600 seconds!
Command 19 - Passed the test in 0.00004300 seconds!
Command 20 - Passed the test in 0.00004000 seconds!
Command 21 - Passed the test in 0.00006200 seconds!
Command 22 - Passed the test in 0.00005400 seconds!
Command 23 - Passed the test in 0.00005200 seconds!
Command 24 - Passed the test in 0.00004000 seconds!
Command 25 - Did not pass the test because of an error in line 1
Command 26 - Did not pass the test because of an error in line 1
Command 27 - Did not pass the test because of an error in line 1
Command 28 - Did not pass the test because of an error in line 1
Command 29 - Did not pass the test because of an error in line 1
Command 30 - Did not pass the test because of an error in line 1
Command 31 - Did not pass the test because of an error in line 1
Command 32 - Did not pass the test because of an error in line 1
Command 33 - Did not pass the test because of an error in line 1
Command 34 - Did not pass the test because of an error in line 1
Command 35 - Did not pass the test because of an error in line 1
Command 36 - Did not pass the test because of an error in line 1
Command 37 - Did not pass the test because of an error in line 1
Command 38 - Did not pass the test because of an error in line 1
Command 39 - Did not pass the test because of an error in line 1
Command 40 - Did not pass the test because of an error in line 1
Command 41 - Did not pass the test because of an error in line 1
Command 42 - Did not pass the test because of an error in line 1
Command 43 - Did not pass the test because of an error in line 1
Command 44 - Did not pass the test because of an error in line 1
Command 45 - Did not pass the test because of an error in line 1
Command 51 - Passed the test in 0.00006600 seconds!
Command 52 - Passed the test in 0.00005100 seconds!
Command 53 - Passed the test in 0.00004300 seconds!
Command 54 - Passed the test in 0.00004000 seconds!
Command 55 - Passed the test in 0.00004300 seconds!
Command 56 - Passed the test in 0.00004400 seconds!
Command 57 - Passed the test in 1.50279000 seconds!
Command 58 - Did not pass the test because of an error in line 2
Command 59 - Did not pass the test because of an error in line 2
Command 60 - Did not pass the test because of an error in line 2
Command 61 - Passed the test in 0.00010600 seconds!
Command 62 - Passed the test in 0.00009400 seconds!
Command 63 - Did not pass the test because of an error in line 2
Command 64 - Did not pass the test because of an error in line 2
Command 65 - Did not pass the test because of an error in line 2
Command 66 - Passed the test in 0.00007900 seconds!
Command 67 - Passed the test in 0.00007700 seconds!
Command 68 - Passed the test in 0.00006900 seconds!
Command 69 - Passed the test in 0.00004900 seconds!
Command 70 - Passed the test in 0.00004400 seconds!
Command 71 - Passed the test in 0.00004500 seconds!
Command 72 - Passed the test in 0.00004700 seconds!
Command 73 - Passed the test in 0.00004200 seconds!
Command 74 - Passed the test in 0.00004700 seconds!
Command 75 - Did not pass the test because of an error in line 2
Command 76 - Did not pass the test because of an error in line 2
Command 77 - Did not pass the test because of an error in line 2
Command 78 - Did not pass the test because of an error in line 2
Command 79 - Did not pass the test because of an error in line 2
Command 80 - Did not pass the test because of an error in line 2
Command 81 - Did not pass the test because of an error in line 2
Command 82 - Did not pass the test because of an error in line 2
Command 83 - Did not pass the test because of an error in line 2
Command 84 - Did not pass the test because of an error in line 3
Command 85 - Did not pass the test because of an error in line 3
Command 86 - Did not pass the test because of an error in line 3
Command 87 - Did not pass the test because of an error in line 2
Command 88 - Did not pass the test because of an error in line 2
Command 89 - Did not pass the test because of an error in line 2
Command 90 - Did not pass the test because of an error in line 2
Command 91 - Did not pass the test because of an error in line 2
Command 92 - Did not pass the test because of an error in line 2
Command 93 - Did not pass the test because of an error in line 2
Command 94 - Did not pass the test because of an error in line 2
CPU Time: 110.46399400 seconds
Memory usage: 2592014336 KB

```

Figura 5.3: Resultados da Situação 3

Nesta situação apuramos os seguintes resultados:

Totalidade de Comandos:95

Nº Comandos Executados com sucesso: 36

Média de tempo por *querie*:0.06536 segundos

Tempo de CPU: 111 segundos

Memória Ocupada: 2.414GB

Motivo de erros: Erros semelhantes aos da situação 1

Em suma, e como forma de agrupar toda esta análise feita segue-se, na Figura 5.4, um gráfico com todos os resultados apurados.

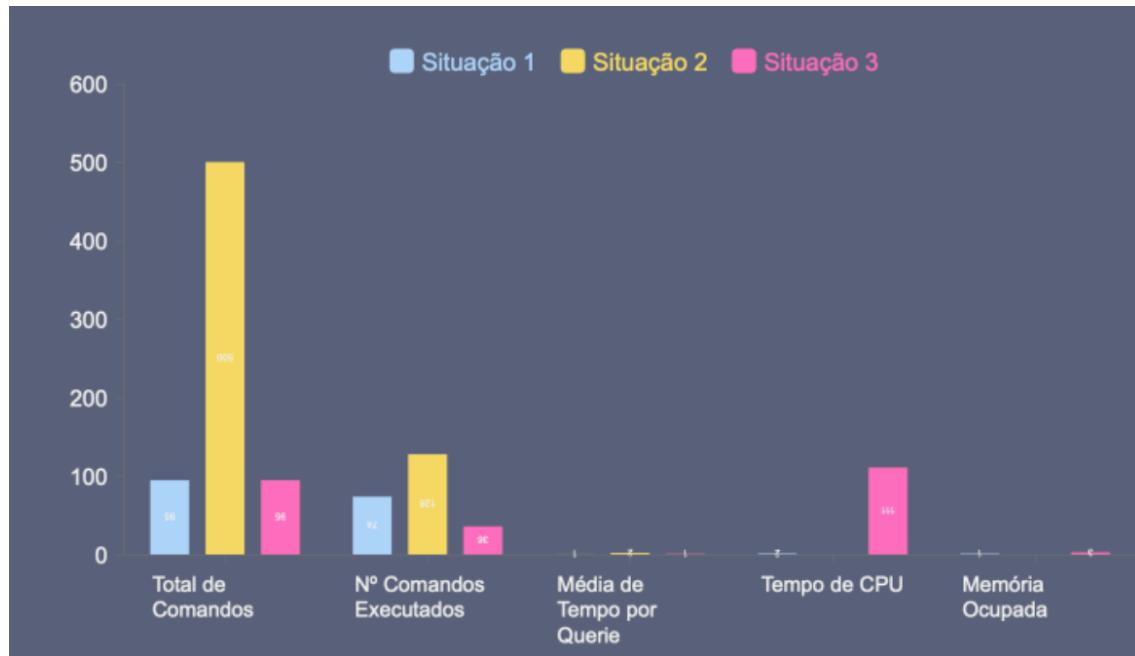


Figura 5.4: Gráfico com os Resultados

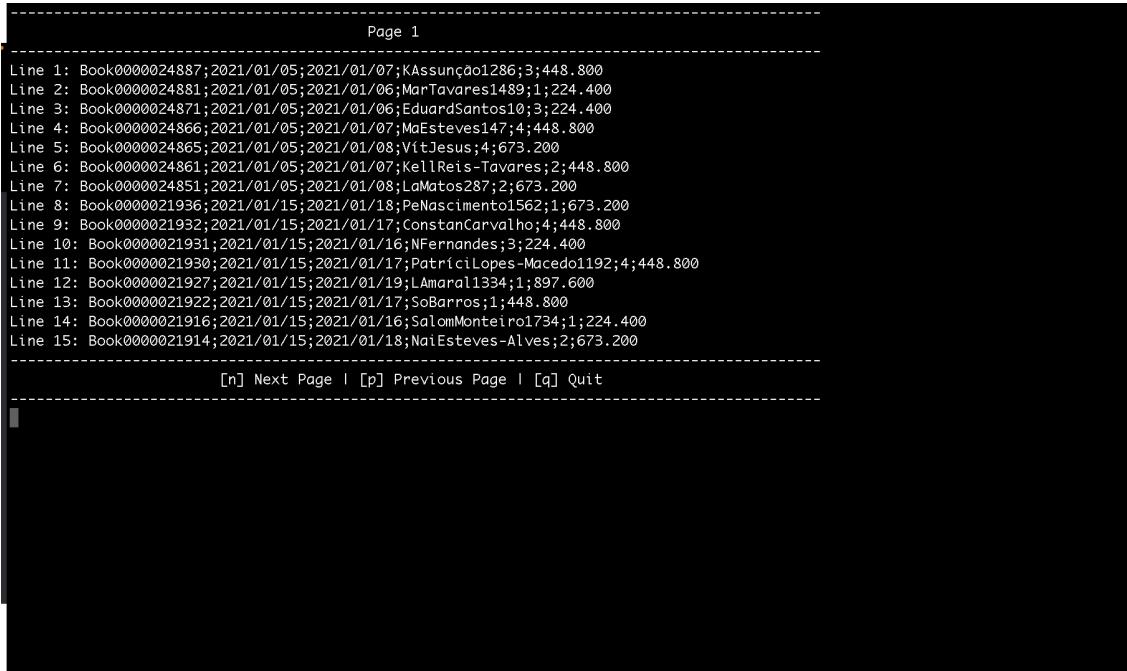
# 6 Paginação

A implementação de paginação foi incorporada às queries 4, 5 e 6 devido à magnitude dos seus resultados. Em cada página, são apresentados no máximo 15 elementos, proporcionando a capacidade de avançar para a próxima página ou retroceder para a página anterior.

Para as *queries* 4 e 5, que operam sobre estruturas de árvores, foi necessário armazenar os nós da árvore num array, associando cada nó a um número de página. Conforme o utilizador navega entre as páginas, o programa calcula o índice do primeiro elemento da página e imprime as 15 linhas correspondentes.

Para a *query* 6, que já opera sobre um *array*, apenas foi necessário implementar o controlo de páginas.

Sendo assim, conseguimos implementar, com sucesso, a paginação no nosso programa. Na Figura 6.1 é apresentado um output paginado referente à *query* 4 com o seguinte input *HTL1001*:



```
-----  
Page 1  
-----  
Line 1: Book0000024887;2021/01/05;2021/01/07;KAssunçao1286;3;448.800  
Line 2: Book0000024881;2021/01/05;2021/01/06;MarTavares1489;1;224.400  
Line 3: Book0000024871;2021/01/05;2021/01/06;EduardSantos10;3;224.400  
Line 4: Book0000024866;2021/01/05;2021/01/07;MaEsteves147;4;448.800  
Line 5: Book0000024865;2021/01/05;2021/01/08;VitJesus;4;673.200  
Line 6: Book0000024861;2021/01/05;2021/01/07;KellReis-Tavares;2;448.800  
Line 7: Book0000024851;2021/01/05;2021/01/08;LaMatos287;2;673.200  
Line 8: Book0000021936;2021/01/15;2021/01/18;PeNascimento1562;1;673.200  
Line 9: Book0000021932;2021/01/15;2021/01/17;ConstanCarvalho;4;448.800  
Line 10: Book0000021931;2021/01/15;2021/01/16;NFernandes;3;224.400  
Line 11: Book0000021930;2021/01/15;2021/01/17;PatrícioLopes-Macedo1192;4;448.800  
Line 12: Book0000021927;2021/01/15;2021/01/19;LAmara11334;1;897.600  
Line 13: Book0000021922;2021/01/15;2021/01/17;SoBarros;1;448.800  
Line 14: Book0000021916;2021/01/15;2021/01/16;SalomMonteiro1734;1;224.400  
Line 15: Book0000021914;2021/01/15;2021/01/18;NaiEsteves-Alves;2;673.200  
-----  
[n] Next Page | [p] Previous Page | [q] Quit
```

Figura 6.1: Paginação

## 7 Menu Interativo

O menu interativo foi uma etapa da implementação bastante interessante e muito intuitiva. O fator liberdade de criação foi, sem dúvida, algo que nos deu vontade em criar uma interface fácil e o mais versátil para todos os futuros utilizadores. De forma breve, vamos iniciar uma atividade fictícia e analisar o percurso de um novo utilizador:

- Início da aplicação onde é perguntado se deseja alterar o *path* para os dataset ou não (Figura 7.1):

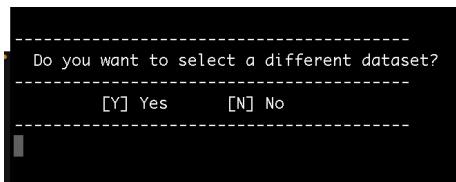


Figura 7.1: Início

- Menu inicial onde permite o utilizador de avançar ou sair da aplicação (Figura 7.2):

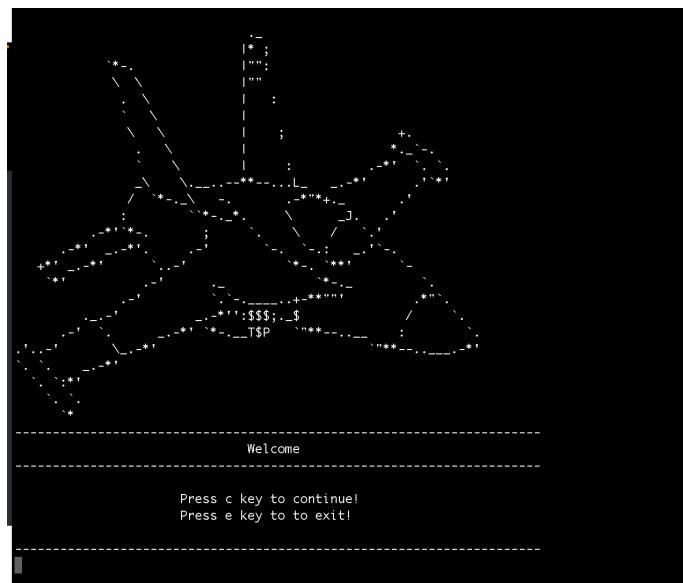


Figura 7.2: Menu

- Lista de *queries* com a devida explicação; aguarda a digitação do utilizador (Figura 7.3):

```

Welcome
-----
Press c key to continue!
Press e key to exit!
-----
c

What queries do you want to run?

1 | Listar o resumo de um utilizador, voo, ou reserva, consoante o identificador recebido por argumento
2 | Listar os voos ou reservas de um utilizador
3 | apresentar a classificação média de um hotel, a partir do seu identificador
4 | Listar as reservas de um hotel, ordenadas por data de inicio
5 | Listar os voos com origem num dado aeroporto, entre duas datas, ordenados por data de partida estimada
6 | Listar o top N aeroportos com mais passageiros, para um dado ano
7 | Listar o top N aeroportos com a maior mediana de atrasos
8 | Apresentar a receita total de um hotel entre duas datas (inclusive), a partir do seu identificador.
9 | Listar todos os utilizadores cujo nome começa com o prefixo passado por argumento, ordenados por nome (de forma crescente)
10 | Apresentar várias métricas gerais da aplicação
0 | Back

```

Figura 7.3: Menu de Queries

- Para a *querie* solicitada é pedido os restantes atributos (Figura 7.4):

```

Welcome
-----
Press c key to continue!
Press e key to exit!
-----
c

What queries do you want to run?

1 | Listar o resumo de um utilizador, voo, ou reserva, consoante o identificador recebido por argumento
2 | Listar os voos ou reservas de um utilizador
3 | apresentar a classificação média de um hotel, a partir do seu identificador
4 | Listar as reservas de um hotel, ordenadas por data de inicio
5 | Listar os voos com origem num dado aeroporto, entre duas datas, ordenados por data de partida estimada
6 | Listar o top N aeroportos com mais passageiros, para um dado ano
7 | Listar o top N aeroportos com a maior mediana de atrasos
8 | Apresentar a receita total de um hotel entre duas datas (inclusive), a partir do seu identificador.
9 | Listar todos os utilizadores cujo nome começa com o prefixo passado por argumento, ordenados por nome (de forma crescente)
10 | Apresentar várias métricas gerais da aplicação
0 | Back
1

-----
Please enter the arguments for your query
Please separate them with a space

```

Figura 7.4: Restante input

- Se for uma resposta curta é impressa de forma direta (Figura 7.5); caso seja uma resposta longa é iniciada uma paginação permitindo, ao utilizador, navegar pelas totalidades de páginas (Figura 7.6):

```
Welcome
-----
Press c key to continue!
Press e key to exit!

-----
c

What queries do you want to run?
1 | Listar o resumo de um utilizador, voo, ou reserva, consoante o identificador recebido por argumento
2 | Listar os voos ou reservas de um hotel, a partir do seu identificador
3 | apresentar a classificação média de um hotel, a partir do seu identificador
4 | Listar as reservas de um hotel, ordenadas por data de inicio
5 | Listar os voos com origem num dado aeroporto, entre duas datas, ordenados por data de partida estimada
6 | Listar o top N aeroportos com mais passageiros, para um dado ano
7 | Listar o top N aeroportos com a maior mediana de atrasos
8 | Apresentar a receita total de um hotel entre duas datas (inclusive), a partir do seu identificador.
9 | Listar todos os utilizadores cujo nome começa com o prefixo passado por argumento, ordenados por nome (de forma crescente)
10 | Apresentar várias métricas gerais da aplicação
0 | Back

1

-----
Please enter the arguments for your query
Please separate them with a space
-----
JessiTavares910
User: Jéssica Tavares;F;63;PT;ZE466021;11;5;2365.200
Do you want to select another query? (y/n): 
```

Figura 7.5: Resposta curta

```
-----
Page 1
-----
Line 1: Book0000024887;2021/01/05;2021/01/07;KAssuncao1286;3;448.800
Line 2: Book0000024881;2021/01/05;2021/01/06;MarTavares1489;1;224.400
Line 3: Book0000024871;2021/01/05;2021/01/06;EduardSantos10;3;224.400
Line 4: Book0000024866;2021/01/05;2021/01/07;MaEsteves147;4;448.800
Line 5: Book0000024865;2021/01/05;2021/01/08;VtJesus;4;673.200
Line 6: Book0000024861;2021/01/05;2021/01/07;KellReis-Tavares;2;448.800
Line 7: Book0000024851;2021/01/05;2021/01/08;LuMatos287;2;673.200
Line 8: Book0000021936;2021/01/15;2021/01/18;PeNascimento1562;1;673.200
Line 9: Book0000021932;2021/01/15;2021/01/17;ConstanCarvalho;4;448.800
Line 10: Book0000021931;2021/01/15;2021/01/16;Nfernandes3;224.400
Line 11: Book0000021930;2021/01/15;2021/01/17;PatricleLopes-Macedo1192;4;448.800
Line 12: Book0000021927;2021/01/15;2021/01/19;LAmoril1334;1;897.600
Line 13: Book0000021922;2021/01/15;2021/01/17;SobBarros;1;448.800
Line 14: Book0000021916;2021/01/15;2021/01/16;SalomMonteiro1734;1;224.400
Line 15: Book0000021914;2021/01/15;2021/01/18;NatEsteves-Alves;2;673.200
-----
[n] Next Page | [p] Previous Page | [q] Quit
```

Figura 7.6: Resposta Longa

- Por último, é perguntado ao utilizador se deseja fazer mais pedidos levando-o para o menu principal (Figura 7.2); caso contrário é encerrado o programa (Figura 7.7);

```

-----
----- Welcome -----
-----
Press c key to continue!
Press e key to exit!

c

What queries do you want to run?
1 | Listar o resumo de um utilizador, voo, ou reserva, consoante o identificador recebido por argumento
2 | Listar os voos ou reservas de um utilizador
3 | apresentar a classificação media de um hotel, a partir do seu identificador
4 | Listar as reservas de um hotel, ordenadas por data de inicio
5 | Listar os voos com origem num dado servico, entre duas datas, ordenados por data de partida estimada
6 | Listar o top N aeroportos com mais passageiros, para um dado ano
7 | Listar o top N aeroportos com a maior mediana de atrasos
8 | Apresentar a receita total de um hotel entre duas datas (inclusive), a partir do seu identificador.
9 | Listar todos os utilizadores cujo nome começa com o prefixo passado por argumento, ordenados por nome (de forma crescente)
10 | Apresentar várias métricas gerais da aplicação
0 | Back

0

Do you want to select another query? (y/n): n
Exiting the program. Goodbye!
flavio@Flavios-MacBook-Pro ~$ 

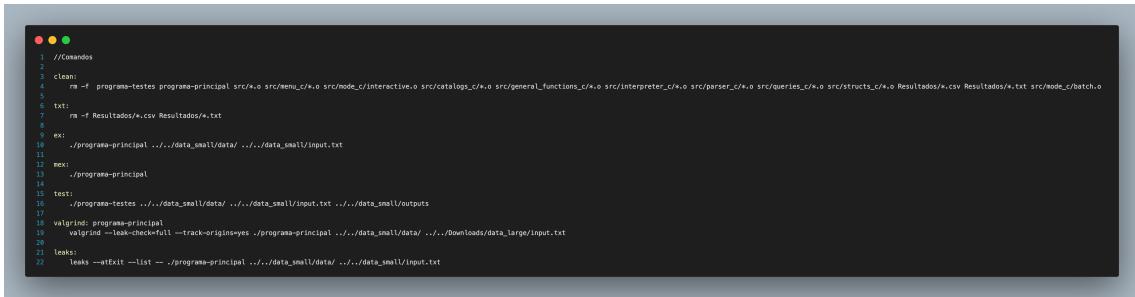
```

Figura 7.7: Permite ao utilizador fazer mais pedidos

## 8 MakeFile

A nossa MakeFile é bastante simples e intuitiva.

De forma breve, temos a compilação de todo o programa, bem como do programa-testes. Adicionalmente, temos alguns comandos extras (Figura 8.1) utilizados por nós com vista a facilitar a execução e análise de casos especiais.



```
1 //Comandos
2
3 clean:
4     rm -f programas-principal src/*.o src/menu_c/*.o src/mode_c/interactive.o src/catalogs_c/*.o src/general_functions_c/*.o src/interpreter_c/*.o src/parser_c/*.o src/queries_c/*.o src/structs_c/*.o Resultados/*.txt src/mode_c/batch.o
5
6 txt:
7     rm -f Resultados/*.csv Resultados/*.txt
8
9 ex:
10    ./programa-principal ../../data_small/data/ ../../data_small/input.txt
11
12 mex:
13    ./programa-principal
14
15 test:
16    ./programa-testes ../../data_small/data/ ../../data_small/input.txt ../../data_small/outputs
17
18 valgrind: programa-principal
19     valgrind --leak-check=full --track-origins=yes ./programa-principal ../../data_small/data/ ../../Downloads/data_large/input.txt
20
21 leaks:
22     leaks --atExit --list -- ./programa-principal ../../data_small/data/ ../../data_small/input.txt
```

Figura 8.1: Comandos disponíveis na MakeFile

## 9 Dificuldades Encontradas

Durante esta segunda fase do projeto a maior dificuldade encontrada foi a diminuição de memória utilizada, que infelizmente não conseguimos completar esse objetivo.

Outra dificuldade encontrada com este projeto foi conseguir completar todas as *Queries* disponibilizadas devido ao foco em outros elementos do projeto, tal como, a estruturação do projeto com as alterações que foram propostas na primeira fase, a correção de *memory leaks* encontradas durante a totalidade do projeto, dificuldades na resolução de certas *Queries* e na paginação das árvores existentes.

# 10 Conclusão

Com o fim deste projeto, foi possível perceber a importância do trabalho em equipa e a colaboração com os professores da unidade curricular.

Infelizmente, o projeto não ficou completo na sua totalidade, sendo possível observar a falta das queries 7, 10 e o uso com sucesso do *dataset large*, em que causava problemas de memória excedida.

Em conclusão, o grupo tem a noção que o projeto evoluiu bastante desde a 1<sup>a</sup> fase e que embora não estando integralmente completo, o projeto alcançou quase todos os resultados pretendidos. Tudo isto se deve a trabalho árduo e é uma prova da capacidade e do potencial dos elementos do grupo e que servirá de exemplo para o futuro. Espera-se que esta evolução e não só o atingir de quase todos os objetivos como também a capacidade de empenho que o grupo depositou neste trabalho tragam resultados positivos.

Em suma, apesar de esta fase não ter corrido como esperado devido aos problemas que surgiram de forma tardia no manuseamento da memória, no geral o balanço desta fase foi positivo. Acreditamos ter-nos centralizado nos dois maiores objetivos desta Unidade Curricular. Com isto, pretendemos dizer que, a modularização e o encapsulamento foram dois objetivos estritamente cumpridos e bem adquiridos. Acrescentamos que, o modo interativo foi, da mesma forma, alcançado com sucesso.

No entanto, e com uma visão futurista, segue-se algumas alterações que o grupo teria feito no projeto:

- Desenvolver uma árvore ordenada por data, por exemplo, em que cada nó apontaria para uma estrutura, ou seja, cada catálogo teria, para além da *GHashTable* uma árvore ordenada por datas - esta estratégia evitaria a criação de árvores no processamento de queries, bem como iterações em *HashTables*.
- Diminuir o consumo de memória utilizado pelo programa