



UNIVERSIDADE DO MINHO
LICENCIATURA EM ENGENHARIA INFORMÁTICA

Laboratórios de Informática III

2023/2024

Fase 2

Realizado por:

Ana Alves	A95128
Simão Antunes	A100597
Gonçalo Brandão	A100663

25 de janeiro de 2024

Conteúdo

1	Introdução	3
2	Arquitetura do projeto	3
2.1	Encapsulamento e modularidade	4
2.2	Estrutura de dados	5
2.3	Parsing dos dados	5
2.4	Validação dos ficheiros de dados	5
2.5	Queries	6
3	Leitura e Tratamento de Dados	6
4	Modo Interativo	7
5	Teste	8
5.1	Memory Leaks	8
5.2	Testes de Desempenho	8
6	Conclusão	12

Lista de Figuras

1	Sugestão da arquitetura para a aplicação a desenvolver	3
2	arquitetura para a aplicação a desenvolver	4
3	Estrutura Users	4
4	Gets da estrutura Users	5
5	Gets da estrutura Flights	5
6	Catálogo de estatísticas	6
7	função para o parser	6
8	menu: página inicial	7
9	menu: paginação	7
10	desempenho das queries no Ubuntu 22.04 (i7-10thGen)	9
11	desempenho das queries no Ubuntu 22.04 (AMD Ryzen 7 4800hs)	10
12	desempenho das queries no Ubuntu 22.04 Fedora 39 (AMD Ryzen 5 5600H)	11

1 Introdução

Para a segunda parte deste projeto, foi necessária uma reestruturação do projeto até então executado. Este relatório contém os passos e decisões tomados na realização da mesma. Desta forma, foi necessário ser capaz de criar coleções e estruturas de dados de forma a conseguir implementar estratégias eficientes e rápidas de realizar as queries propostas. Para este efeito, recorreremos à biblioteca da linguagem C, GLib. Este relatório contém os passos e decisões tomados na realização das duas fases do projeto. Inicialmente é apresentada uma descrição detalhada de como realizamos o encapsulamento e garantimos modularidade, seguida de uma explicação da implementação do parsing dos ficheiros, bem como a forma como fizemos o a validação dos mesmos. Posteriormente temos uma explicação das estratégias implementadas no desenvolvimento das queries e do modo interativo proposto. Além disso, apresentamos os nossos resultados obtidos de forma a verificar a rapidez de cada query implementada, finalizando com uma breve conclusão.

2 Arquitetura do projeto

A arquitetura do trabalho foi baseada na arquitetura sugerida no enunciado disponibilizado. Desta forma, conseguimos entender o conceito de encapsulamento e modularidade e destacar os principais componentes, interações e dependências, o que é crucial para tomar decisões informadas sobre a implementação.

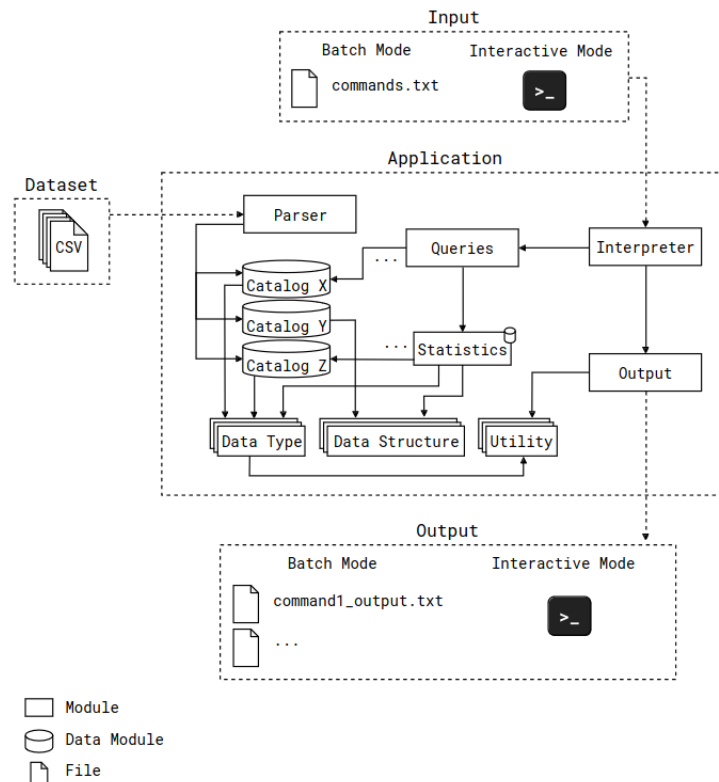


Figura 1: Sugestão da arquitetura para a aplicação a desenvolver

2.1 Encapsulamento e modularidade

De forma a garantir um bom encapsulamento e modularidade no trabalho, o mesmo foi estruturado para esse efeito, criando módulos, onde as dependências apenas são dos níveis abaixo em relação aos mais acima, nunca no mesmo "nível" ou de cima para baixo.

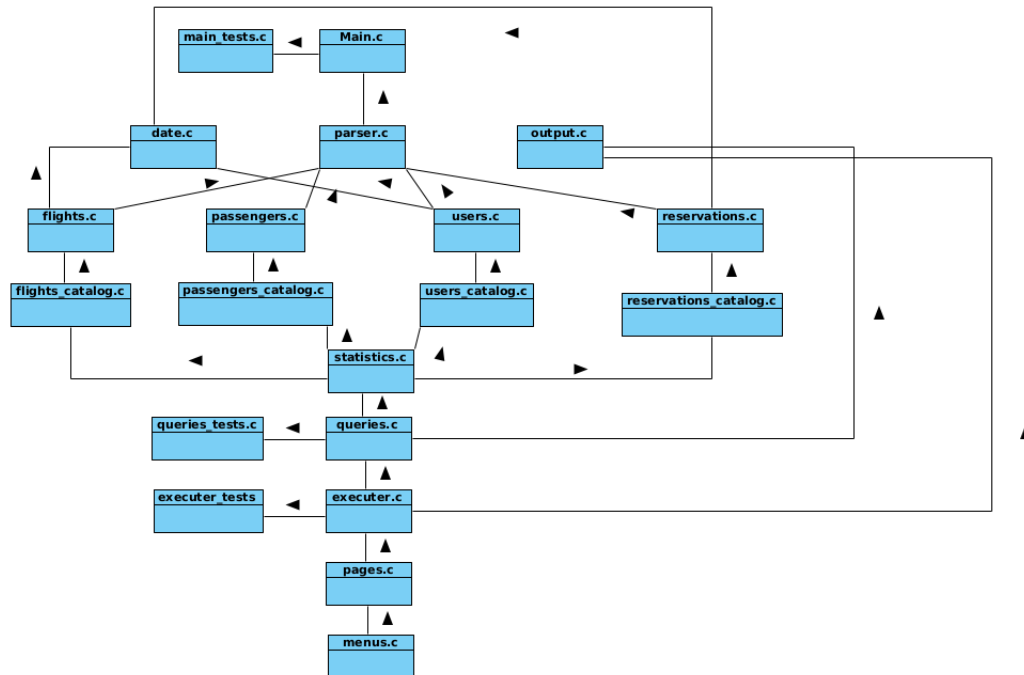


Figura 2: arquitetura para a aplicação a desenvolver

Posto isto, é nestes novos módulos que são implementadas coleções específicas, e todas as operações sobre estas, sendo estas estruturas privadas e as operações referidas públicas. Ou seja, apenas as funções no interior do módulo podem aceder a estas estruturas. No entanto, é sempre garantida a opacidade entre módulos através de cópias dos valores. Isto ocorre, por exemplo, com a estrutura User e as respetivas funções get que são as únicas com acesso à estrutura mencionada, devolvendo sempre cópias. Este encapsulamento está demonstrado na imagem que se segue. É de salientar que, no diagrama, estão apenas presentes os módulos gerais. Deste modo, de forma a tornar a imagem mais perceptível.

```
typedef struct users {
    char * id;
    char * name;
    Date * birth_date;
    char sex;
    char * passport;
    char * country_code;
    Date * account_creation;
    int account_status;
} User;
```

Figura 3: Estrutura Users

```

char * get_user_id(User * user);

char * get_user_name(User * user);

Date * get_user_birth_date(User * user);

char get_user_sex(User * user);

char * get_user_passport(User * user);

char * get_user_country_code(User * user);

Date * get_user_account_creation(User * user);

int get_user_account_status(User * user);

```

Figura 4: Gets da estrutura Users

2.2 Estrutura de dados

```

typedef struct flights {
    int id;
    char * airline;
    char * plane_model;
    int total_seats;
    char * origin;
    char * destination;
    Date * schedule_departure_date;
    Date * schedule_arrival_date;
    Date * real_departure_date;
    Date * real_arrival_date;
} Flight;

```

Figura 5: Gets da estrutura Flights

A nível das estruturas de dados foram tomadas decisões de modo a diminuir o tamanho que elas ocupam. Para tal, a estratégia que utilizamos passou por utilizar o tipo de dados com menor tamanho possível. Como no caso do id e no total seats, foram definidos como int uma vez que consideramos mais eficiente do que um char.

2.3 Parsing dos dados

Para fazermos o parsing dos ficheiros CSV, em vez de termos uma função de parsing específica para cada ficheiro, de modo a reutilizarmos código, criámos uma função geral para parsing de ficheiros. Deste modo, a função recebe como argumento uma função que vai ser específica de cada ficheiro. Posto isto, optamos inicialmente por criar um array de strings, composto pelos tokens, obtidos a partir de cada linha de dados dos ficheiros, os quais são de seguida tratados pela função passada como argumento.

2.4 Validação dos ficheiros de dados

De forma a fazer a validação de entradas de dados durante o parsing dos ficheiros CSV é passada como argumento uma função específica que tem, no seu corpo, uma função responsável pela validação dos dados de cada ficheiro. Assim, verifica-se token a token se o seu campo é ou não nulo e, no caso de estar preenchido, se está conforme o pressuposto. Caso seja válido, é então criada a coleção específica do ficheiro,

i.e Users, Flights, Passengers e Reservations, e adicionada à sua própria estrutura de dados, contida num catálogo exclusivo.

2.5 Queries

Para responder às queries foram criadas novas estruturas para servir como valor de return das várias funções e também para melhorar o tempo de resposta. As estruturas de dados definidas no catálogo das estatísticas são apenas a GHashTable e o GPttrArray. A GHashTable foi escolhida para recolher informação nas queries, pois como para uma dada key a informação tem de ir sendo atualizada, é a estrutura mais rápida para detetar e devolver o campo que queremos atualizar. Já o GPttrArray foi a estrutura escolhida para que se pudesse ordenar as coleções que lhe são inseridas. Todas as funções que podem aceder ao catálogo das estatísticas são implementadas em statistics.c respeitando sempre a opacidade e encapsulamento. Estas estruturas facilitam na resposta às queries uma vez que estar a percorrer um catálogo sempre que é feita uma interrogação é ineficiente. Dessa forma, o nosso objetivo era percorrer os catálogos necessários apenas uma vez e criar estruturas adequadas para responder a cada query. No entanto, essa implementação levou a um maior uso de memória, o que impediu que o programa fosse executado com o large-dataset.

```
typedef struct cs{
    GHashTable * users;
    GHashTable * flights;
    GHashTable * reservations;
    GHashTable * hotels;
    GPttrArray * hreservations;
} CatalogoStats;
```

Figura 6: Catálogo de estatísticas

3 Leitura e Tratamento de Dados

De modo a conseguir responder às queries propostas de forma eficiente, primeiro organizamos o grande volume de informação presente nos flights, users, passengers e reservations. Desta forma, criamos catalogos diferentes para cada tipo de dados com a ajuda da biblioteca GLib. Para efectuar o parser, utilizamos uma função geral capaz de trabalhar qualquer estrutura de dados.

```
void parser(char * path, void (* buildStruct)(char **, void *, void *, void *), void * structure1, void * structure2, void * structure3) {
    FILE * file = fopen(path, "r");
    if (!file) {
        perror("Erro ao abrir o ficheiro");
        return;
    }
    char * line = NULL;
    ssize_t read;
    size_t len;
    while ((read = getline(&line, &len, file)) != -1) {
        char * line_copy = strdup(line);
        char * formatted_line = strsep(&line_copy, "\n");
        char ** tokens = parseLine(formatted_line);
        buildStruct(tokens, structure1, structure2, structure3);
        free(formatted_line);
        free(tokens);
    }
    free(line);
    fclose(file);
}
```

Figura 7: função para o parser

4 Modo Interativo

Para o modo interativo, foi criada uma coleção (Paginacao) que possui uma estrutura de dados com o intuito de armazenar a informação à medida que se vai formando um output nas queries, sendo escolhido o GPtrArray (dados), para além disso são utilizados também caracteres de modo a facilitar a paginação. Deste modo, no fim de uma query, a coleção encontra-se preenchida ao que se segue um modo interativo que permite ao utilizador ver até 40 valores de output simultaneamente, permitindo avançar e recuar de página, ir diretamente para a última página e voltar para a primeira. Enquanto o utilizador quiser, pode continuar a executar queries até o mesmo pretender encerrar o programa.

```
Sistema de gestão
e consulta de viagens

Grupo 34:
- Ana João Alves
- Gonçalo Brandão
- Simão Antunes

Welcome!
What queries do you want to run?

1. Display User, Flight or Reservation Summary (1 <ID>)
2. List User Flights, Reservations or Both (2 <ID> [flights|reservations])
3. Display Average Hotel Rating (3 <ID>)
4. List Reservations by Hotel (4 <ID>)
5. List Flights by Origin Airport (5 <Name> <Begin_date> <End_date>)
6. List Top Airports for a Year (6 <Year> <N>)
7. List Top Airports by Median Delay (7 <N>)
8. Display Total Revenue by Hotel (8 <ID> <Begin_date> <End_date>)
9. List Users by Name Prefix (9 <Prefix>)
10. Display General Metrics (10 [year | month])

Press E key to to exit!

Enter your option: █
```

Figura 8: menu: página inicial

```
Page 1/11

01 - Book0000030197;2023/07/21;2023/07/24;GabrieGomes;4;673.200
02 - Book0000030199;2023/07/21;2023/07/23;MartRibeiro1463;1;448.800
03 - Book0000030217;2023/07/21;2023/07/22;AdriRamos278;2;224.400
04 - Book0000027509;2023/07/18;2023/07/20;IMatos1421;2;448.800
05 - Book0000027516;2023/07/18;2023/07/19;RuMda1404;2;224.400
06 - Book0000027521;2023/07/18;2023/07/22;GoOliveira1587;1;897.600
07 - Book0000027525;2023/07/18;2023/07/19;MigueBorges1734;4;224.400
08 - Book0000027528;2023/07/18;2023/07/20;GTorres1871;2;448.800
09 - Book0000027529;2023/07/18;2023/07/22;EdgFaria-Mota1014;3;897.600
10 - Book0000027532;2023/07/18;2023/07/19;JuliMaia1685;2;224.400
11 - Book0000027539;2023/07/18;2023/07/21;JLoureiro;1;673.200
12 - Book0000027541;2023/07/18;2023/07/20;ÉriSoares1545;4;448.800
13 - Book0000027554;2023/07/18;2023/07/22;BrVicente-Alves;4;897.600
14 - Book0000027556;2023/07/18;2023/07/21;RuMendes;1;673.200
15 - Book0000027557;2023/07/18;2023/07/20;SarAzevedo729;2;448.800
16 - Book0000027559;2023/07/18;2023/07/21;BeMatias;3;673.200
17 - Book0000027565;2023/07/18;2023/07/19;SalTavares159;3;224.400
18 - Book0000027569;2023/07/18;2023/07/22;JoeTavares;1;897.600
19 - Book0000027575;2023/07/18;2023/07/22;RafaeEsteves-Simões1918;2;897.600
20 - Book0000029217;2023/06/17;2023/06/18;JoanMagalhães-Alves;3;224.400
21 - Book0000029219;2023/06/17;2023/06/19;NCoelho-Costa;4;448.800
22 - Book0000029222;2023/06/17;2023/06/19;MadaDomingues;4;448.800
23 - Book0000029224;2023/06/17;2023/06/20;CCorreia;4;673.200
24 - Book0000029227;2023/06/17;2023/06/19;RHenriques1210;4;448.800
25 - Book0000029239;2023/06/17;2023/06/19;FerMacedo720;1;448.800
26 - Book0000029240;2023/06/17;2023/06/21;Antorres-Campos;3;897.600
27 - Book0000029242;2023/06/17;2023/06/21;CésaMendes1953;3;897.600
28 - Book0000029246;2023/06/17;2023/06/19;SofAnjos;2;448.800
29 - Book0000018530;2023/06/09;2023/06/10;WilMarques;2;224.400
30 - Book0000018536;2023/06/09;2023/06/13;CaetanAnjos;3;897.600
31 - Book0000018545;2023/06/09;2023/06/11;IDomingues;2;448.800
32 - Book0000018550;2023/06/09;2023/06/11;VMagalhães;4;448.800
33 - Book0000018555;2023/06/09;2023/06/10;CAssunção896;2;224.400
34 - Book0000018556;2023/06/09;2023/06/11;PGomes324;3;448.800
35 - Book0000018557;2023/06/09;2023/06/12;PaSousa;1;673.200
36 - Book0000018579;2023/06/09;2023/06/12;LHiranda310;1;673.200
37 - Book0000018582;2023/06/09;2023/06/12;DaMarques109;3;673.200
38 - Book0000018583;2023/06/09;2023/06/12;Loureço1019;2;673.200
39 - Book0000000951;2023/05/10;2023/05/14;Kelly-VerMarques;2;897.600
40 - Book0000000953;2023/05/10;2023/05/13;LBatista727;2;673.200

[N] Next Page | [P] Previous Page | [F] First Page | [L] Last Page | [Q] Quit
█
```

Figura 9: menu: paginação

5 Teste

De forma geral, a plataforma disponibilizada pelos docentes para testar o código tornou-se bastante útil na detenção e correção de erros ou código incompleto. Com base nos resultados obtidos, o código foi reestruturado diversas vezes e com o auxílio do Valgrind estes foram resolvidos.

5.1 Memory Leaks

Na primeira fase do trabalho, o método utilizado gerava imensos erros e memory leaks devido às dependências de dados entre estruturas de dados que tornavam difícil libertar a memória, problema que foi corrigido através de uma reestruturação do mesmo. Ao longo do trabalho fomos sempre tentando prevenir a ocorrência das mesmas através da criação de funções para fazer free tanto das estruturas de dados como das coleções. Para além disso, sempre que foi necessária a utilização de memória dinâmica, a implementação do código foi pensada de modo a poder libertar a memória alocada assim que possível. Atualmente o trabalho tem apenas as memory leaks associadas à utilização da biblioteca GLib, que são 0,019 MB.

5.2 Testes de Desempenho

Para esta parte do trabalho foram utilizadas as bibliotecas `time.h` para obter o tempo de execução e `sys/resource.h` para obter o maximum memory resident size das queries e do programa. Deste modo, temos uma função que compara os ficheiros criados pelas queries implementadas com os ficheiros com o output pretendido, lendo linha a linha os dois ficheiros e comparando as strings. Esta função vai então verificar se a query executou dentro do tempo estipulado e se os ficheiros são iguais, originando o output consoante os resultados obtidos.

Sistema Operativo	Processador	RAM(GB)	Memória (KB)	Tempo de Execução (s)
Ubuntu 22.04	i7- 10thGen	12	41600	1.111206
			41680	1.129149
			41472	1.140693
Fedora 39	AMD Ryzen 5 5600H	8	40704	0.695691
			40704	0.737732
			40832	0.796419
Ubuntu 22.04	AMD Ryzen 7 4800hs	32	41632	0.729595
			41604	0.699054
			41620	0.704132

Tabela 1: Desempenho do programa

```

-----
Here are the results!
-----
Command1 - Passed the test in 0.000127 seconds!
Command2 - Passed the test in 0.000015 seconds!
Command3 - Passed the test in 0.000060 seconds!
Command4 - Passed the test in 0.000013 seconds!
Command5 - Passed the test in 0.000011 seconds!
Command6 - Passed the test in 0.000010 seconds!
Command7 - Passed the test in 0.000012 seconds!
Command8 - Passed the test in 0.000071 seconds!
Command9 - Passed the test in 0.000045 seconds!
Command10 - Passed the test in 0.000036 seconds!
Command11 - Passed the test in 0.000006 seconds!
Command12 - Passed the test in 0.000006 seconds!
Command13 - Passed the test in 0.000047 seconds!
Command14 - Passed the test in 0.000042 seconds!
Command15 - Passed the test in 0.000027 seconds!
Command16 - Passed the test in 0.000006 seconds!
Command17 - Passed the test in 0.000006 seconds!
Command18 - Passed the test in 0.000006 seconds!
Command19 - Passed the test in 0.016286 seconds!
Command20 - Passed the test in 0.000023 seconds!
Command21 - Passed the test in 0.007528 seconds!
Command22 - Passed the test in 0.000006 seconds!
Command23 - Passed the test in 0.010454 seconds!
Command24 - Passed the test in 0.000006 seconds!
Command25 - Passed the test in 0.000027 seconds!
Command26 - Passed the test in 0.000030 seconds!
Command27 - Passed the test in 0.000030 seconds!
Command28 - Passed the test in 0.005353 seconds!
Command29 - Passed the test in 0.004927 seconds!
Command30 - Passed the test in 0.005281 seconds!
Command31 - Passed the test in 0.003740 seconds!
Command32 - Passed the test in 0.003531 seconds!
Command33 - Passed the test in 0.004021 seconds!
Command39 - Passed the test in 0.026345 seconds!
Command40 - Passed the test in 0.023229 seconds!
Command41 - Passed the test in 0.023381 seconds!
Command42 - Passed the test in 0.023058 seconds!
Command43 - Passed the test in 0.023350 seconds!
Command51 - Passed the test in 0.000030 seconds!
Command52 - Passed the test in 0.000005 seconds!
Command53 - Passed the test in 0.000034 seconds!
Command54 - Passed the test in 0.000006 seconds!
Command55 - Passed the test in 0.000005 seconds!
Command56 - Passed the test in 0.000006 seconds!
Command57 - Passed the test in 0.000005 seconds!
Command58 - Passed the test in 0.000030 seconds!
Command59 - Passed the test in 0.000028 seconds!
Command60 - Passed the test in 0.000048 seconds!
Command61 - Passed the test in 0.000007 seconds!
Command62 - Passed the test in 0.000007 seconds!
Command63 - Passed the test in 0.000037 seconds!
Command64 - Passed the test in 0.000026 seconds!
Command65 - Passed the test in 0.000026 seconds!
Command66 - Passed the test in 0.000007 seconds!
Command67 - Passed the test in 0.000008 seconds!
Command68 - Passed the test in 0.000006 seconds!
Command69 - Passed the test in 0.010004 seconds!
Command70 - Passed the test in 0.004305 seconds!
Command71 - Passed the test in 0.006428 seconds!
Command72 - Passed the test in 0.000006 seconds!
Command73 - Passed the test in 0.011097 seconds!
Command74 - Passed the test in 0.000007 seconds!
Command75 - Passed the test in 0.000033 seconds!
Command76 - Passed the test in 0.000023 seconds!
Command77 - Passed the test in 0.000022 seconds!
Command78 - Passed the test in 0.000009 seconds!
Command79 - Passed the test in 0.006315 seconds!
Command80 - Passed the test in 0.006049 seconds!
Command81 - Passed the test in 0.003971 seconds!
Command82 - Passed the test in 0.003009 seconds!
Command83 - Passed the test in 0.005072 seconds!
Command89 - Passed the test in 0.023051 seconds!
Command90 - Passed the test in 0.023025 seconds!
Command91 - Passed the test in 0.023311 seconds!
Command92 - Passed the test in 0.023274 seconds!
Command93 - Passed the test in 0.023375 seconds!
-----
Memory usage: 41536 KB
Elapsed time: 0.713444 seconds
-----

```

Figura 10: desempenho das queries no Ubuntu 22.04 (i7-10thGen)

```

-----
Here are the results!
-----
Command1 - Passed the test in 0.000127 seconds!
Command2 - Passed the test in 0.000015 seconds!
Command3 - Passed the test in 0.000006 seconds!
Command4 - Passed the test in 0.000013 seconds!
Command5 - Passed the test in 0.000011 seconds!
Command6 - Passed the test in 0.000010 seconds!
Command7 - Passed the test in 0.000012 seconds!
Command8 - Passed the test in 0.000071 seconds!
Command9 - Passed the test in 0.000045 seconds!
Command10 - Passed the test in 0.000036 seconds!
Command11 - Passed the test in 0.000006 seconds!
Command12 - Passed the test in 0.000006 seconds!
Command13 - Passed the test in 0.000047 seconds!
Command14 - Passed the test in 0.000042 seconds!
Command15 - Passed the test in 0.000027 seconds!
Command16 - Passed the test in 0.000006 seconds!
Command17 - Passed the test in 0.000006 seconds!
Command18 - Passed the test in 0.000006 seconds!
Command19 - Passed the test in 0.016286 seconds!
Command20 - Passed the test in 0.000023 seconds!
Command21 - Passed the test in 0.007528 seconds!
Command22 - Passed the test in 0.000006 seconds!
Command23 - Passed the test in 0.010454 seconds!
Command24 - Passed the test in 0.000006 seconds!
Command25 - Passed the test in 0.000027 seconds!
Command26 - Passed the test in 0.000030 seconds!
Command27 - Passed the test in 0.000030 seconds!
Command28 - Passed the test in 0.005353 seconds!
Command29 - Passed the test in 0.004927 seconds!
Command30 - Passed the test in 0.005281 seconds!
Command31 - Passed the test in 0.003740 seconds!
Command32 - Passed the test in 0.003531 seconds!
Command33 - Passed the test in 0.004021 seconds!
Command34 - Passed the test in 0.026345 seconds!
Command35 - Passed the test in 0.023229 seconds!
Command36 - Passed the test in 0.023381 seconds!
Command37 - Passed the test in 0.023058 seconds!
Command38 - Passed the test in 0.023350 seconds!
Command39 - Passed the test in 0.000038 seconds!
Command40 - Passed the test in 0.000005 seconds!
Command41 - Passed the test in 0.000034 seconds!
Command42 - Passed the test in 0.000006 seconds!
Command43 - Passed the test in 0.000005 seconds!
Command44 - Passed the test in 0.000005 seconds!
Command45 - Passed the test in 0.000006 seconds!
Command46 - Passed the test in 0.000006 seconds!
Command47 - Passed the test in 0.000005 seconds!
Command48 - Passed the test in 0.000005 seconds!
Command49 - Passed the test in 0.000006 seconds!
Command50 - Passed the test in 0.000006 seconds!
Command51 - Passed the test in 0.000006 seconds!
Command52 - Passed the test in 0.000006 seconds!
Command53 - Passed the test in 0.000006 seconds!
Command54 - Passed the test in 0.000006 seconds!
Command55 - Passed the test in 0.000005 seconds!
Command56 - Passed the test in 0.000006 seconds!
Command57 - Passed the test in 0.000005 seconds!
Command58 - Passed the test in 0.000006 seconds!
Command59 - Passed the test in 0.000006 seconds!
Command60 - Passed the test in 0.000006 seconds!
Command61 - Passed the test in 0.000007 seconds!
Command62 - Passed the test in 0.000007 seconds!
Command63 - Passed the test in 0.000007 seconds!
Command64 - Passed the test in 0.000006 seconds!
Command65 - Passed the test in 0.000006 seconds!
Command66 - Passed the test in 0.000007 seconds!
Command67 - Passed the test in 0.000008 seconds!
Command68 - Passed the test in 0.000006 seconds!
Command69 - Passed the test in 0.010004 seconds!
Command70 - Passed the test in 0.004305 seconds!
Command71 - Passed the test in 0.006428 seconds!
Command72 - Passed the test in 0.000006 seconds!
Command73 - Passed the test in 0.011097 seconds!
Command74 - Passed the test in 0.000007 seconds!
Command75 - Passed the test in 0.000033 seconds!
Command76 - Passed the test in 0.000023 seconds!
Command77 - Passed the test in 0.000022 seconds!
Command78 - Passed the test in 0.000009 seconds!
Command79 - Passed the test in 0.006315 seconds!
Command80 - Passed the test in 0.006049 seconds!
Command81 - Passed the test in 0.003971 seconds!
Command82 - Passed the test in 0.003009 seconds!
Command83 - Passed the test in 0.005072 seconds!
Command84 - Passed the test in 0.023051 seconds!
Command85 - Passed the test in 0.023025 seconds!
Command86 - Passed the test in 0.023311 seconds!
Command87 - Passed the test in 0.023274 seconds!
Command88 - Passed the test in 0.023375 seconds!
-----
Memory usage: 41536 KB
Elapsed time: 0.713444 seconds
-----

```

Figura 11: desempenho das queries no Ubuntu 22.04 (AMD Ryzen 7 4800hs)

```

-----
Here are the results!
-----
Command1 - Passed the test in 0.000237 seconds!
Command2 - Passed the test in 0.000033 seconds!
Command3 - Passed the test in 0.000045 seconds!
Command4 - Passed the test in 0.000040 seconds!
Command5 - Passed the test in 0.000029 seconds!
Command6 - Passed the test in 0.000028 seconds!
Command7 - Passed the test in 0.000025 seconds!
Command8 - Passed the test in 0.000050 seconds!
Command9 - Passed the test in 0.000041 seconds!
Command10 - Passed the test in 0.000035 seconds!
Command11 - Passed the test in 0.000026 seconds!
Command12 - Passed the test in 0.000023 seconds!
Command13 - Passed the test in 0.000035 seconds!
Command14 - Passed the test in 0.000036 seconds!
Command15 - Passed the test in 0.000047 seconds!
Command16 - Passed the test in 0.000023 seconds!
Command17 - Passed the test in 0.000023 seconds!
Command18 - Passed the test in 0.000025 seconds!
Command19 - Passed the test in 0.010486 seconds!
Command20 - Passed the test in 0.004227 seconds!
Command21 - Passed the test in 0.006104 seconds!
Command22 - Passed the test in 0.000035 seconds!
Command23 - Passed the test in 0.011192 seconds!
Command24 - Passed the test in 0.000051 seconds!
Command25 - Passed the test in 0.000041 seconds!
Command26 - Passed the test in 0.000047 seconds!
Command27 - Passed the test in 0.000037 seconds!
Command28 - Passed the test in 0.005117 seconds!
Command29 - Passed the test in 0.004783 seconds!
Command30 - Passed the test in 0.005112 seconds!
Command31 - Passed the test in 0.003315 seconds!
Command32 - Passed the test in 0.003226 seconds!
Command33 - Passed the test in 0.004281 seconds!
Command39 - Passed the test in 0.019048 seconds!
Command40 - Passed the test in 0.018697 seconds!
Command41 - Passed the test in 0.018702 seconds!
Command42 - Passed the test in 0.019450 seconds!
Command43 - Passed the test in 0.018728 seconds!
Command51 - Passed the test in 0.000075 seconds!
Command52 - Passed the test in 0.000032 seconds!
Command53 - Passed the test in 0.000041 seconds!
Command54 - Passed the test in 0.000030 seconds!
Command55 - Passed the test in 0.000023 seconds!
Command56 - Passed the test in 0.000024 seconds!
Command57 - Passed the test in 0.000029 seconds!
Command58 - Passed the test in 0.000040 seconds!
Command59 - Passed the test in 0.000045 seconds!
Command60 - Passed the test in 0.000041 seconds!
Command61 - Passed the test in 0.000024 seconds!
Command62 - Passed the test in 0.000024 seconds!
Command63 - Passed the test in 0.000043 seconds!
Command64 - Passed the test in 0.000043 seconds!
Command65 - Passed the test in 0.000058 seconds!
Command66 - Passed the test in 0.000028 seconds!
Command67 - Passed the test in 0.000023 seconds!
Command68 - Passed the test in 0.000023 seconds!
Command69 - Passed the test in 0.011363 seconds!
Command70 - Passed the test in 0.004302 seconds!
Command71 - Passed the test in 0.006195 seconds!
Command72 - Passed the test in 0.000030 seconds!
Command73 - Passed the test in 0.011129 seconds!
Command74 - Passed the test in 0.000033 seconds!
Command75 - Passed the test in 0.000073 seconds!
Command76 - Passed the test in 0.000042 seconds!
Command77 - Passed the test in 0.000034 seconds!
Command78 - Passed the test in 0.009034 seconds!
Command79 - Passed the test in 0.009575 seconds!
Command80 - Passed the test in 0.009107 seconds!
Command81 - Passed the test in 0.003588 seconds!
Command82 - Passed the test in 0.003113 seconds!
Command83 - Passed the test in 0.004659 seconds!
Command89 - Passed the test in 0.018421 seconds!
Command90 - Passed the test in 0.017519 seconds!
Command91 - Passed the test in 0.017931 seconds!
Command92 - Passed the test in 0.017671 seconds!
Command93 - Passed the test in 0.017489 seconds!
-----
Memory usage: 40576 KB
Elapsed time: 0.002197 seconds
-----

```

Figura 12: desempenho das queries no Ubuntu 22.04 Fedora 39 (AMD Ryzen 5 5600H)

Em termos de desempenho, observamos que nosso programa se destaca ao lidar com conjuntos de dados pequenos, apresentando resultados em menos de milissegundos na maioria dos casos. Essa eficiência inicial é promissora, sugerindo uma implementação otimizada para cenários menos desafiadores.

No entanto, é crucial considerar como esse desempenho se escala quando confrontado com conjuntos

de dados maiores. O comportamento eficiente em datasets pequenos não foi diretamente proporcional.

Além disso, ao lidar com dados em maior escala, notamos um aumento significativo no consumo de memória. O aumento no consumo de memória está intimamente ligado aos algoritmos utilizados no nosso programa. Tornou-se evidente que, ao lidar com volumes de dados mais substanciais, certos algoritmos podem não estar otimizados para gerenciar eficientemente os recursos disponíveis, resultando em um uso significativo de memória.

Testar o programa em diferentes tamanhos de conjunto de dados foi fundamental para garantir uma eficiência consistente em diversas escalas, como é visível a semelhança dos resultados nas diferentes máquinas.

6 Conclusão

Consideremos que o trabalho foi bastante desafiante principalmente com a nova implementação do dataset large. Mesmo após várias alterações da estrutura do código, não correu como planeávamos uma vez que para um elevado número de dados o trabalho ultrapassa o limite de memória.

Por outro lado, consideramos que conseguimos implementar um código que contempla boas práticas de encapsulamento, modularidade e opacidade. No primeiro, alcançamos um bom encapsulamento tendo sempre em conta uma hierarquia entre módulos, em que apenas existem dependências de baixo para cima e nunca no sentido inverso ou no mesmo "nível". Em relação à modularidade, garantimos que, apenas é possível gerir um determinado módulo com as funções do mesmo e, para além disso, tornámos os detalhes de implementação dos módulos privados e públicos apenas as funções necessárias a outros módulos. Já no que toca à opacidade, entendemos que esteja assegurada uma vez que nunca são fornecidos os valores reais dos dados, mas sim cópias ou clones dos mesmos.