Departamento de Informática

Escola de Engenharia

Universidade do Minho

Relatório Fase 2 (Final)

Laboratórios de Informática III

Licenciatura em Engenharia Informática

2023/2024

Grupo 47

Ana Cerqueira, a104188

Nuno Silva, a104089

Eduardo Faria, a104353

25/01/2024

Índice

Introdução

No âmbito da disciplina "Laboratórios de Informática III", fomos incumbidos do desenvolvimento de um sistema para organizar dados relacionados a viagens. Este sistema envolve a manipulação de informações sobre voos e respetivos passageiros, utilizadores e reservas cujas informações são inseridas em Hash Tables de modo a ficarem disponíveis de forma organizada e otimizada para uso durante o programa.

Nesta segunda fase, após feita a estrutura e organização inicial do trabalho, trabalhamos no sentido de aperfeiçoar alguns pontos e desenvolver os restantes tópicos propostos.

Modularidade e encapsulamento

Este foi um dos principais pontos desenvolvidos pelo nosso grupo durante esta segunda fase. Após recebermos o *feedback* dos professores docentes no sentido de melhorar este tópico, mudámos a estrutura do nosso programa de forma a satisfazer melhor este aspeto.

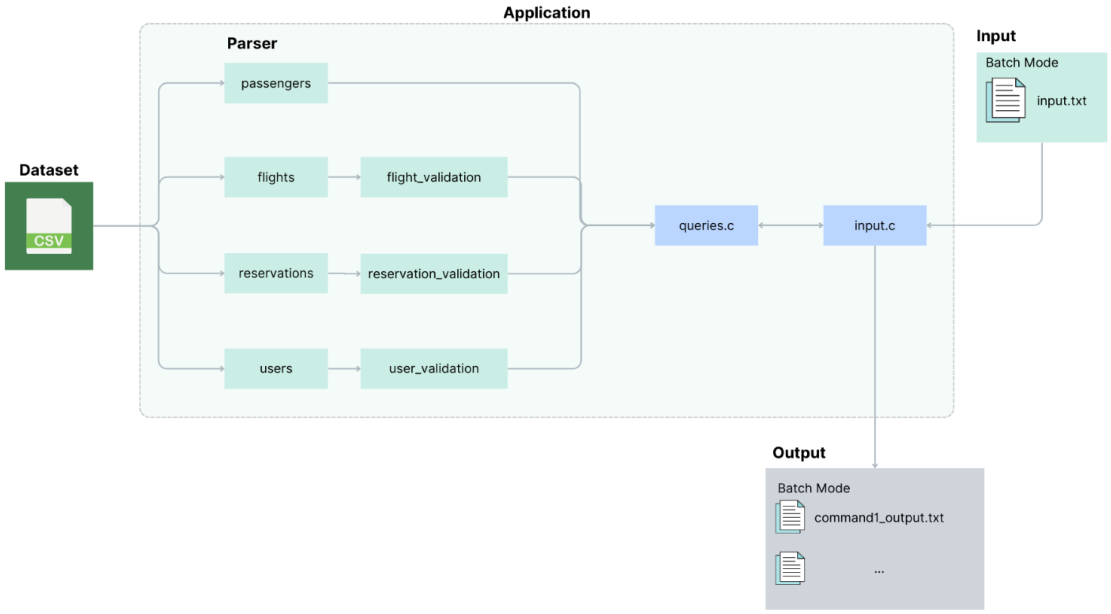
A primeira modificação que realizamos foi tornar as structs opacas. Isto permite que a implementação interna de uma estrutura de dados seja alterada sem afetar o código que a utiliza. Isso ajuda a evitar dependências desnecessárias entre as diferentes partes do programa e promove uma clara separação entre a interface pública e a implementação privada.

Por outro lado, modificamos algumas funções, nomeadamente as que envolviam as queries, de modo que o input seja tratado num ficheiro, a execução noutro e posteriormente o output num ficheiro distinto.

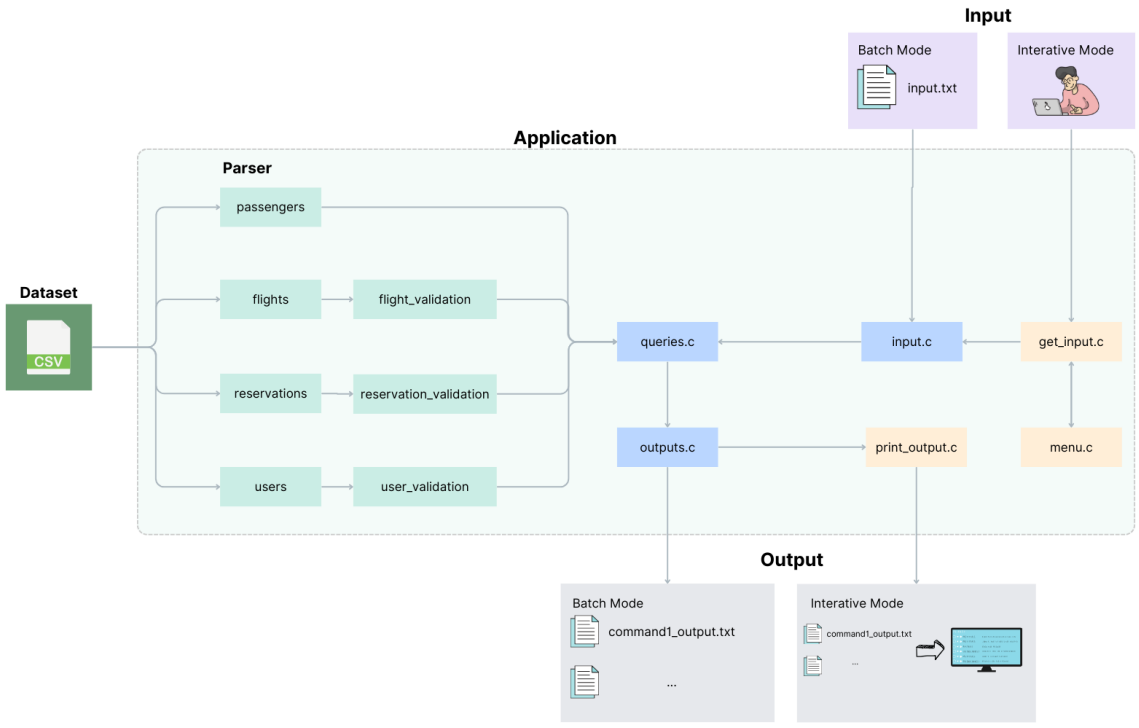
Outras alterações incluíram a simplificação de diversas funções mediante a criação de versões que utilizam funções auxiliares. Para isso, reunimos certas porções de código repetidas em diversas situações e criámos funções gerais, de forma que estas possam substituir os segmentos que eram repetidos diversas vezes. Para além disso, procedeu-se também à fragmentação de funções extensas em múltiplas partes menores.

Após implementar uma série de alterações, desde a fragmentação de arquivos até a simplificação de funções, acreditamos que a nossa aplicação está agora significativamente mais simples e intuitiva. Apesar do aumento da complexidade, também devido à implementação do modo interativo, o seu funcionamento tornou-se muito mais linear.

Implementação inicial:



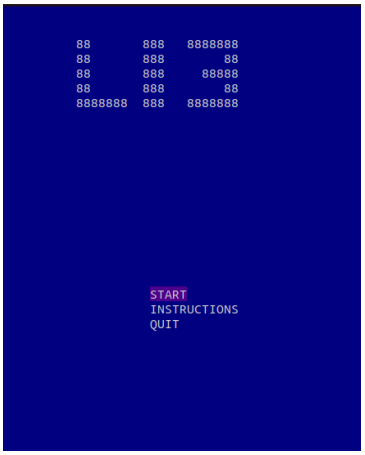
Implementação final:



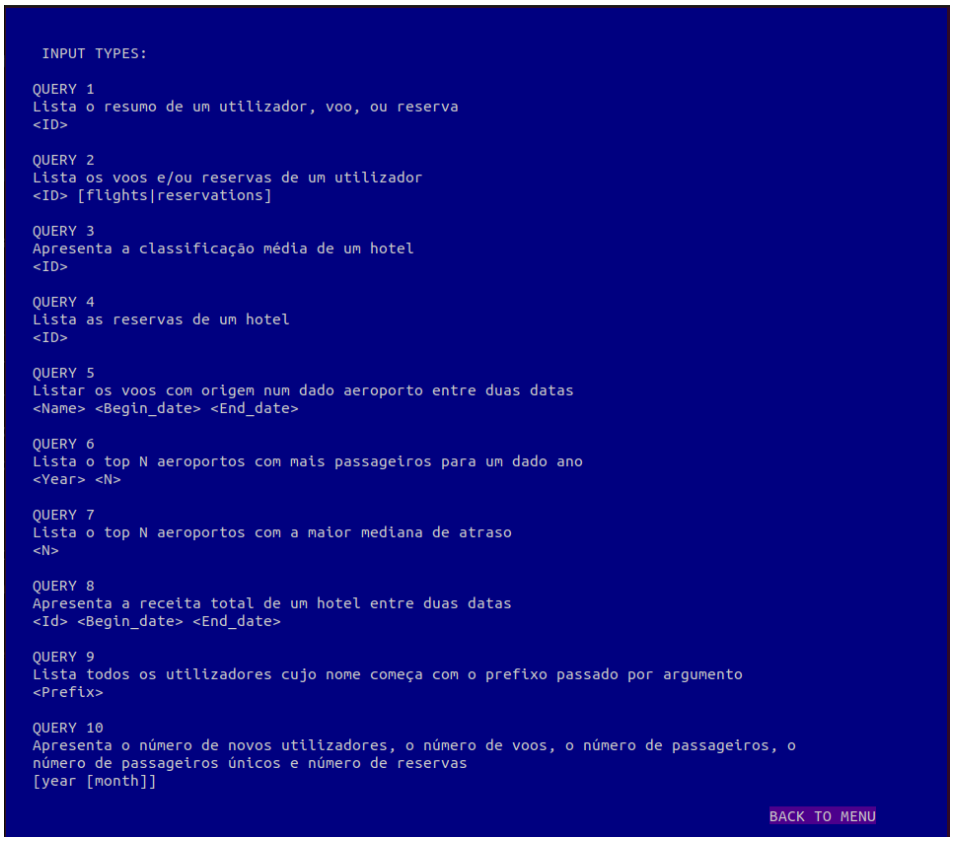
Modos de execução

Nesta segunda fase, para além do modo *batch*, era pedida a implementação de um modo interativo.

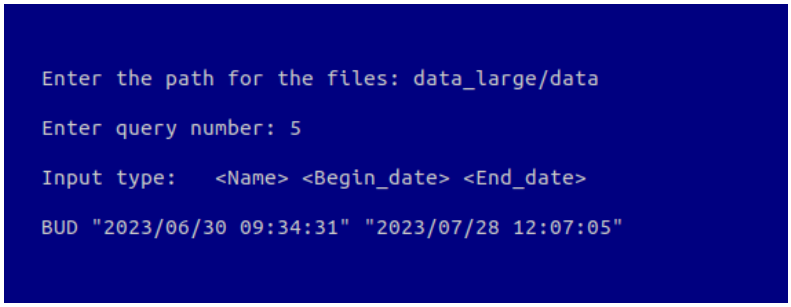
O nosso grupo optou pela utilização da biblioteca *ncurses* com a qual já estávamos familiarizados, devido ao projeto do semestre passado. Com ela criámos um menu simples onde são disponibilizadas as instruções para os inputs das queries, um botão para começar a execução do programa e outro para sair.



Quando selecionada a 2º opção (INSTRUCTIONS), é apresentada a seguinte lista que apresenta uma breve explicação do output de cada query e qual deve ser a forma do input.



Quando selecionada a opção 'START' o programa pede para o utilizador introduzir o *path* para os ficheiros com os dados. Isto só será pedido a primeira vez que o programa é executado, considerando sempre os mesmos ficheiros de dados para os próximos inputs. É depois pedido ao utilizador que ceda o número da query que se pretende chamar e devolvido o formato que o seu input deve ter de forma a poder chamá-la corretamente.



Após isto, serão apresentados os resultados correspondentes à chamada.

Para além destes atributos básicos, acreditámos ainda ser importante realçar algumas funcionalidades adicionais que o nosso grupo implementou, tais como:

- O scroll para outputs grandes em resposta à query;
- O print de mensagens de erro para inputs inválidos, tornando a utilização mais intuitiva;
- O display do input do usuário em tempo real na janela ncurses permitindo ao user escrever e apagar livremente fazendo com que a utilização do programa seja bastante simples.

Queries

Para esta fase final do projeto apenas tínhamos de solucionar as queries 5, 6, 7 e 10. De seguida, apresentamos a explicação individual do raciocínio usado na resolução de cada uma.

Query 5

O objetivo da `func_answer_query_5` é listar os voos com origem em um determinado aeroporto, entre duas datas específicas, ordenados por data de partida estimada, da mais recente para a mais antiga.

A função realiza esta tarefa por meio de uma busca na *`flights_hash_table`* de voos que têm a origem desejada e estão dentro do intervalo de datas fornecido. Ela começa por inicializar uma lista de estruturas `FlightInfo` chamada `list_of_flights`, que será utilizada para armazenar as informações dos voos encontrados. Depois, a função itera sobre os voos presentes na `Hash Table` e verifica se cada voo atende aos critérios estabelecidos: origem igual à fornecida e datas de partida dentro do intervalo especificado. Para os voos que atendem aos requisitos, a função insere-os ordenadamente na lista `list_of_flights`. A ordenação é realizada com base na data de partida estimada dos voos.

A função retorna a string `output5`, que contém as informações formatadas dos voos que atendem aos critérios da query.

Query 6

Nesta questão era pedida a listagem do top N aeroportos com mais passageiros, para um dado ano.

A função realiza essa tarefa através da iteração na `flights_hash_table`. Ela começa por iniciar uma lista de estruturas `Out6` chamada `info`, que será utilizada para armazenar informações sobre o número total de passageiros para cada aeroporto (origem e destino). Posteriormente, a função itera sobre os voos presentes na `Hash Table` e, durante a iteração, verifica se cada voo pertence ao ano especificado. Para cada voo que atende ao ano, a função verifica se o aeroporto de origem e destino já está presente na lista `info`. Se já estiver, incrementa o número total de passageiros para esse aeroporto; caso contrário, adiciona o aeroporto à lista. Após a iteração, a função cria uma lista `ordered_info` ordenada pelo número total de passageiros em cada aeroporto, por ordem decrescente, ou pelo nome dos aeroportos, por ordem crescente, caso haja empate no número de passageiros. Isto é feito utilizando a função `func_insert_passengers_in_order`.

A função retorna a string `output6`, que contém as informações formatadas dos top N aeroportos com mais passageiros para o ano especificado.

Query 7

Esta query tem como objetivo fornecer a lista dos top N aeroportos com a maior mediana de atrasos. Esses atrasos são determinados pela diferença entre a data estimada e a data real de partida para voos com origem nesse aeroporto.

A função correspondente, `func_answer_query_7`, realiza a tarefa por meio da iteração pela `flights_hash_table`. Ela começa por inicializar uma lista de estruturas `AirportInfo` chamada `list_of_airports`, que será utilizada para armazenar informações sobre a mediana de atrasos para cada aeroporto (origem). A função itera sobre os voos presentes na `Hash Table` e, durante a iteração, verifica se cada voo pertence à origem desejada e, em seguida, insere a informação do atraso na lista correspondente. Após a iteração, a função calcula a mediana dos atrasos para cada aeroporto e cria uma lista `ordered_list_of_airports` ordenada por essa mediana, por ordem decrescente.

A função retorna a string `output7`, que contém as informações formatadas dos top N aeroportos com a maior mediana de atrasos.

Query 10

Nesta query solicitava-se a apresentação de várias métricas gerais da aplicação. As métricas a consideradar são:

- número de novos utilizadores registados;
- número de voos;
- número de passageiros;
- número de passageiros únicos;
- número de reservas.

A função 'func_answer_query_10' analisa os dados fornecidos e gera estatísticas sobre o número total de criações de contas, voos, passageiros, passageiros únicos e reservas. Ela verifica o parâmetro 'token' e, caso este seja nulo, a função calculará as estatísticas para todo o período disponível (todos os anos que possuem registos). Por outro lado, caso o 'token' contenha apenas o ano, serão calculadas as estatísticas de cada mês (com registos) desse determinado ano. Por fim, se o 'token' incluir não apenas o ano, mas também o mês, a função calculará as estatísticas de cada dia (com registos) desse determinado mês, nesse mesmo ano.

Posteriormente itera sobre os possíveis anos, meses ou dias e calcula o número total de criações de contas, voos, passageiros, passageiros únicos e reservas. Com base nos cálculos realizados, a função chama funções específicas (func_output_10_year, func_output_10_month, ou func_output_10_day) para formatar e agregar as informações na *string* de saída.

Por fim, retorna a string output10, que contém as informações formatadas sobre as métricas gerais da aplicação para o período especificado.

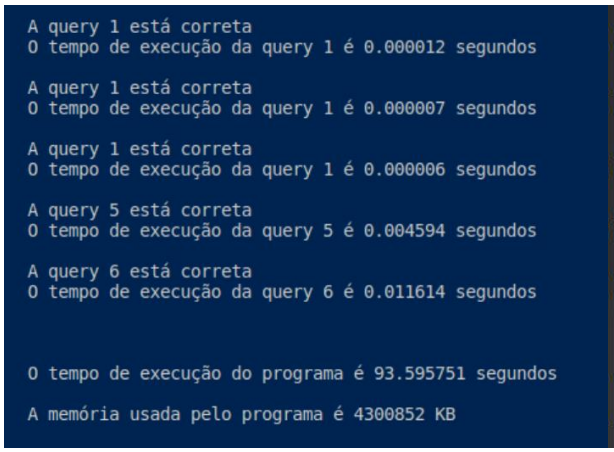
Testes funcionais e de desempenho

Nesta fase é também considerada uma componente de desempenho, onde são desenvolvidos testes que validam e avaliam o funcionamento do nosso programa. Para isso, foram criados testes de modo a avaliar o funcionamento de cada query.

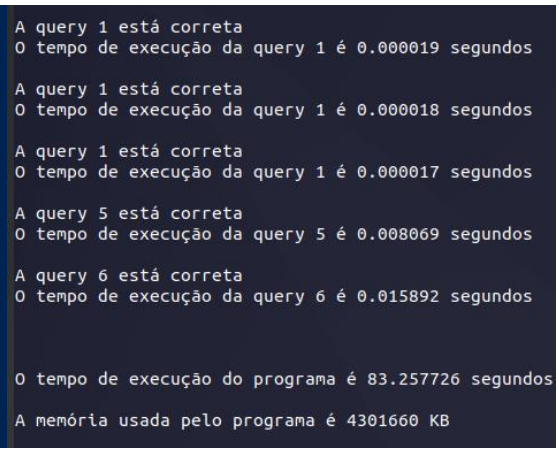
Este programa compara cada resultado esperado com o resultado real da query executada, indicando se o teste passou ou não. Caso o resultado obtido seja diferente do esperado, indica-se a linha onde a primeira incongruência foi encontrada. Para além da validação, indica-se o tempo de execução de cada query, bem como o tempo de execução geral e ainda a memória usada pelo programa.

Numa perspetiva de autoavaliação e reflexão sobre o nosso trabalho, passamos agora a analisar os dados obtidos através destes testes. Para isso, vamos considerar dois computadores (A e B) e o dataset large concedido pelos docentes.

Output A:



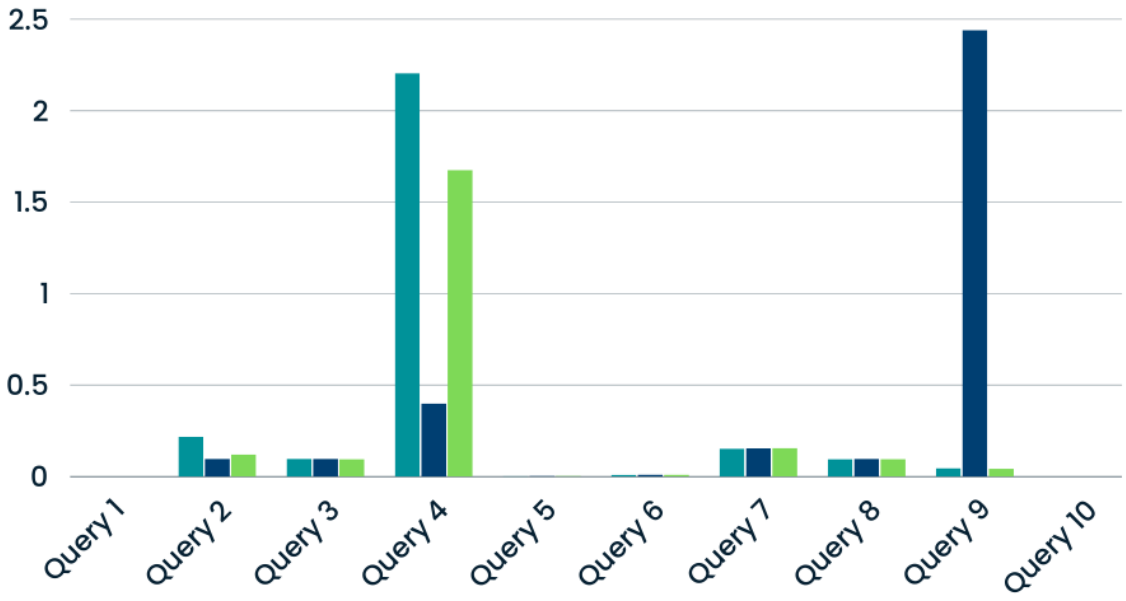
Output B:



Numa análise mais superficial, podemos verificar que apesar do programa utilizar mais memória no computador B, o seu tempo de execução é menor neste dispositivo. Podemos ver ainda que o tempo de execução das queries que temos como exemplo é menor em A apesar do seu tempo de execução total ser superior. As diferenças nestes parâmetros podem ocorrer devido:

- **Configuração de Hardware:** Mesmo que o programa seja o mesmo, as configurações de hardware podem causar impacto no desempenho. O computador B pode ter um processador mais rápido, uma GPU mais potente, ou outras características que tornam a execução mais eficiente.
- **Configurações do Sistema Operacional:** As configurações do sistema operacional em cada computador podem influenciar o desempenho. Por exemplo, a alocação de recursos, configurações de gestão de energia e políticas de cache podem variar.
- **Carga de Trabalho do Sistema:** Outros processos em execução nos computadores podem afetar o desempenho. Se o computador A estiver a realizar tarefas mais intensas em termos de CPU ou memória, pode haver um impacto no tempo de execução do programa.

Apresentamos agora um gráfio onde é exibido o tempo de execução de cada query. Para o obter, foram retirados 3 tempos de execução aleatórios dos resultados do computador A.



Numa primeira análise, verificámos que as queries 1, 5, 6 e 10 apresentam tempos de execução bastante pequenos relativamente às restantes. Em relação à query 1, o seu baixo tempo de execução deve-se à forma como os dados utilizados por ela estão guardados nas *hash tables*. Como a procura nas *hash tables* é feita de acordo com as chaves, obtemos uma procura bastante otimizada (de tempo constante) e o seu tempo de execução é bastante pequeno. Para as outras queries o baixo tempo de execução pode ocorrer por diferentes fatores, desde as quantidades reduzidas de dados com que têm de trabalhar até operações mais simples que tem de realizar.

Vemos ainda variações significativas de tempos de execução dentro da mesma query, como é o caso da query 4 e 9. Estas variações podem ocorrer devido a quantidades variadas de dados a serem tratados durante a sua execução, resultantes de diferentes inputs. Isto porque elas têm de apresentar o seu resultado de forma ordenada e manipular grandes ou pequenas quantidades de dados, o que torna o processo de ordenação mais demorado ou menos demorado, respetivamente.

Conclusão

A realização deste projeto não foi de todo um processo fácil. Foi necessária muita pesquisa e ajuda/esclarecimentos por parte de colegas e docentes. Apesar das dificuldades, acreditámos ter desenvolvido competências importantes para o nosso percurso formativo.

Após a conclusão deste desafio fica o sentimento de dever cumprido, sendo que demos o nosso melhor, ficando, por isso, profundamente satisfeitos com toda a aprendizagem durante este processo.