Universidade do Minho
Escola de Engenharia

Relatório do Projeto de LI3 - Fase 2

Grupo 41

Trabalho realizado por:
Ana Oliveira a104536
André Campos a104618
Beatriz Peixoto a104170

Braga, janeiro de 2024

Índice

Introdução.....3
Arquitetura..... 3
Código.....5
Programa Principal.....5
Módulos..... 5
Queries..... 6
Query 1 - Resumo de User/Voo/Reserva.....6
Query 2 - Voos e Reservas de um User..... 6
Query 3 - Classificação média de um Hotel.....6
Query 4 - Reservas de um Hotel.....7
Query 5 - Listar Voos de um Aeroporto..... 7
Query 6 - Top N Aeroportos com mais passageiros.....7
Query 7 - Top N Aeroportos com a maior Mediana.....7
Query 8 - Receita Total de um Hotel.....7
Query 9 - Listar os Users que contêm o mesmo Prefixo.....7
Query 10 - Métricas gerais da Aplicação.....7
Testes funcionais e de desempenho..... 8
Atualizações..... 8
Conclusão..... 9

Introdução

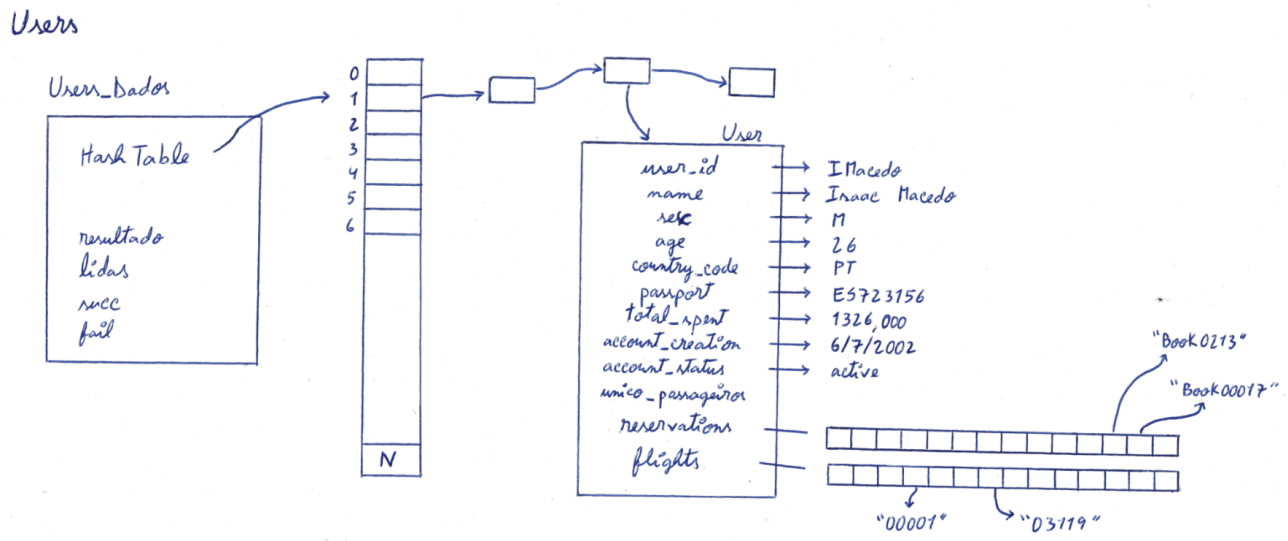
O presente projeto foi realizado no âmbito da Unidade Curricular de Laboratórios de Informática III, integrada no plano curricular do 2º ano, 1º semestre, do ano letivo de 2023/2024, do Curso de Licenciatura em Engenharia Informática, da Universidade do Minho.

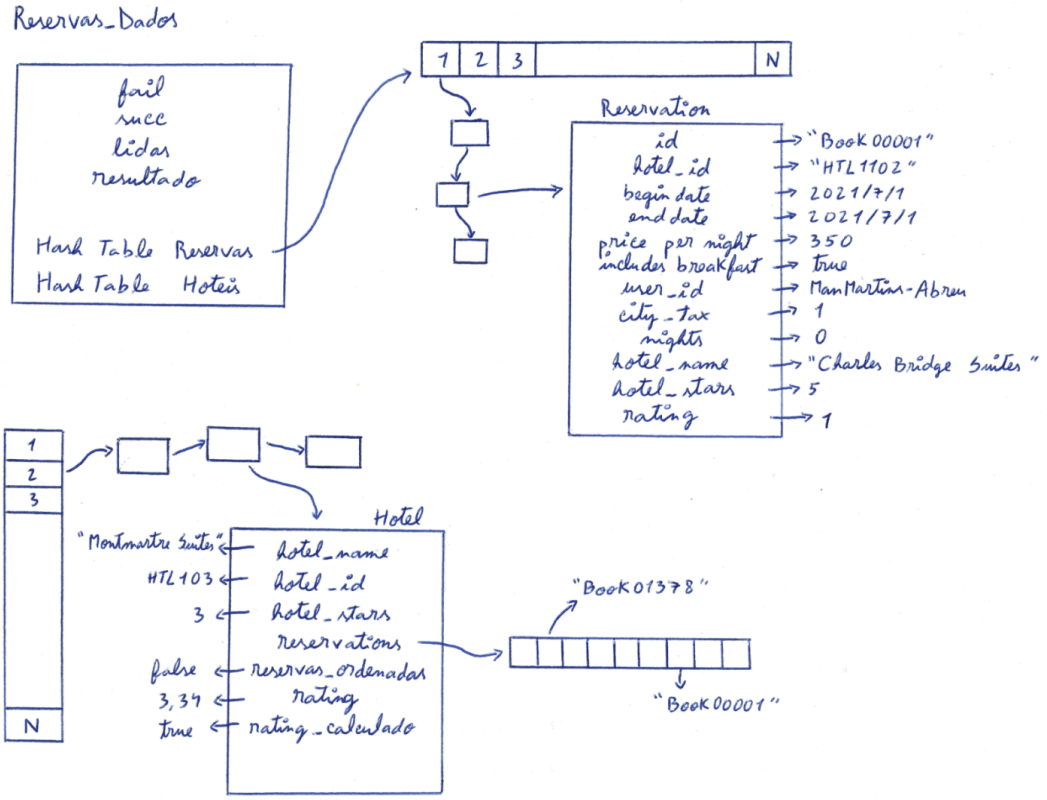
A avaliação, nesta segunda fase do projeto, irá focar-se na execução das restantes queries, a integração do modo interativo, testes de memória, testes funcionais e a implementação de estratégias de modularidade e encapsulamento do código, e também a melhoria de aspetos já analisados na fase 1. Além disto, foi, por parte dos docentes, implementado um dataset de tamanho significativamente superior.

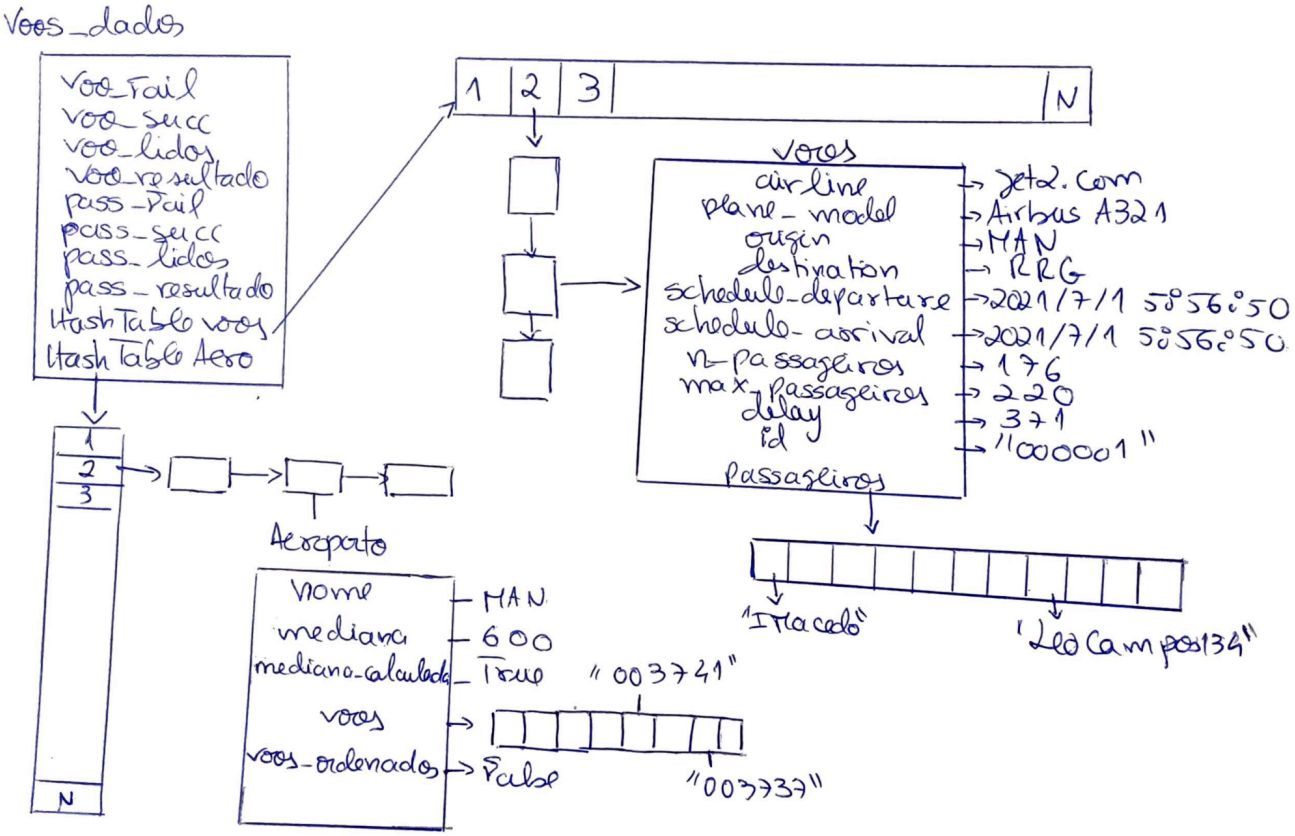
A realização deste projeto em grupo, tem como objetivos a consolidação das aprendizagens de C que nos foram lecionadas ao longo do semestre, bem como a utilização de ferramentas cruciais para o desenvolvimento de projetos na linguagem suprarreferida.

Relativamente à estrutura deste relatório, o mesmo terá início na abordagem da arquitetura atualizada do projeto, a partir da qual iremos analisar o código do mesmo, seguido do programa principal e de um pequeno excerto dedicado aos módulos utilizados**.** Posteriormente, será refletido o código relativo ao modo interativo, seguido da resolução das queries restantes. Por fim, encerraremos este relatório em conjunto com o projeto de Laboratórios de Informática III, através de uma pequena conclusão crítico-reflexiva sobre o trabalho que foi realizado ao longo deste semestre, no âmbito desta unidade curricular.

Arquitetura




Reservas_Dados
fail
succ
lidas
resultado
Hash Table Reservas
Hash Table Hotéis
1 2 3 N
Reservation
id → "Book00001"
hotel_id → "HTL1102"
begin date → 2021/7/1
end date → 2021/7/1
price per night → 350
includes breakfast → true
user_id → ManMartins-Abreu
city_tax → 1
nights → 0
hotel_name → "Charles Bridge Suites"
hotel_stars → 5
rating → 1
Hotel
"Montmartre Suites" ← hotel_name
HTL103 ← hotel_id
3 ← hotel_stars
reservations
false ← reservas_ordenadas
3,34 ← rating
true ← rating_calculado
"Book01378"
"Book00001"


Voos_dados
voo_fail
voo_succ
voo_lidos
voo_resultado
pass_fail
pass_succ
pass_lidos
pass_resultado
Hash Table voos
Hash Table Aero
1 2 3 N
Voos
airline → jet2.com
plane_model → Airbus A321
origin → MAN
destination → RRG
schedule_departure → 2021/7/1 5:56:50
schedule_arrival → 2021/7/1 5:56:50
n_passageiros → 176
max_passageiros → 220
delay → 371
id → "1000001"
Passageiros
"Ithaca6"
"Leo Campos134"
Aeroporto
nome → MAN
mediana → 600
mediana_calculada → True
voos → "003741"
voos_ordenados → False
"003737"

Código

Programa Principal

O programa principal tem 4 fases: Inicialização, Leitura de Dados, Queries e Finalização.

Inicialização: o programa começa por declarar uma string que será o caminho para os ficheiros de entrada. Se o número de argumentos é 3 (modo batch), então a string será o 1 argumento. Se o número de argumentos é 1 (modo interativo), então a string será definida pelo utilizador. Já se o número de argumentos não forem nem 1 nem 3, aparece uma mensagem no terminal a indicar que o número de argumentos está incorreto.

Leitura de Dados: após obtermos o caminho, procedemos à incorporação dos ficheiros de entrada. Começamos pelo Users, pois não necessitam de outros dados. Depois incorporamos as Reservations, dado que elas necessitam dos Users, depois os Flights e por fim os Passengers, já que estes necessitam dos Flights e Users. No fim iremos obter 3 estruturas de dados principais: Users, Voos e Reservas. Em todas essas estruturas, teremos também o número de dados lidos, dados válidos, dados inválidos e por fim o resultado, que indica se os dados foram lidos do local indicado pelo caminho, foram lidos pela pasta default ou se nem lidos foram. Após a leitura dos dados, todos os dados inválidos são escritos em ficheiros .csv e são guardados na pasta "Resultados".

Queries: Após a incorporação dos dados, iremos resolver as queries. No modo batch, um ficheiro de texto (2 argumento) é lido e cada linha é lida, executada e o seu resultado é escrito num ficheiro de texto e guardado na pasta "Resultados". No modo interativo, o usuário escreve a query que pretende realizar, sendo esta lida, executada e o seu resultado exibido no ecrã.

Finalização: Após a execução das queries, o programa termina, mas antes é necessário "limpar" todos os dados usados pelo programa, como também é necessário terminar o ecrã do modo interativo. Após a limpeza, é apresentada uma mensagem de despedida com os créditos dos desenvolvedores e o programa é encerrado.

Módulos

Aqui estão todos os módulos que este programa utiliza:

ArrayDin: Funções sobre uso e manipulação de array dinâmicos do tipo genérico
Batch: Funções que realiza a leitura do ficheiro de texto onde as queries estão escritas
Datas: Funções sobre uso e manipulação do Tipo "Datas" usados nas Reservas e Voos
Ficheiros: Funções que usam e manipulam ficheiros, lendo e escrevendo ficheiros
HashTable: Funções sobre uso e manipulação de HashTable (Chaining) do tipo genérico
Modo_Interativo: Funções sobre o uso do modo interativo
Queries: Funções para a detecção e execução das queries
Reservas: Funções sobre uso e manipulação de Reservas e Hoteis
Testes: Funções auxiliares para o programa de testes
Users: Funções sobre uso e manipulação de Users
Voos: Funções sobre uso e manipulação de Voos e Aeroportos

Modo Interativo

Neste modo, que não recebe argumentos, é primeiramente inicializado um ecrã de título, com o texto “Bem vindo ao modo interativo”. De seguida, é pedido ao utilizador que introduza o caminho para o dataset, sendo esta variável guardada. Tendo o caminho para o dataset, o programa principal procede à leitura de dados provenientes deste mesmo dataset. Na fase de queries do programa principal, é inicializada a função correspondente a um menu, com as opções de queries. Ao ser selecionada a query desejada, esta função chama uma nova função correspondente à query escolhida, que irá fazer ao utilizador perguntas relacionadas com esta (como, por exemplo, se deseja flag, qual o id do user, etc). Assim, procede a formar a string que será utilizada quando invoca a função interpreterQueries, que lê essa mesma string, juntamente com os dados de users, voos e reservas, e invoca a função da query especificada na string. Por fim, tendo executado a interpreterQueries, e portanto possuindo o output desejado, resta imprimir este mesmo no terminal.

(A paginação do output é executada simplesmente imprimindo as linhas equivalentes ao tamanho do terminal, e continuando para as n linhas seguintes (este n sendo o tamanho do terminal) ou retrocedendo para as n anteriores, com o auxílio das setas para cima e para baixo). Apesar deste ter sido o funcionamento desejado, devido a erros e falta de tempo, a paginação não foi colocada a funcionar a 100%, dando problemas quando se trata de outputs longos.

Dá-se assim por terminado o modo interativo e o programa principal procede à fase da limpeza.

Queries

Query 1 - Resumo de User/Voo/Reserva

Usamos as funções que verificam se existe algum user/voo/reserva com aquele ID, e se existir, apresentar o seu resumo. Começamos primeiro por verificar se o ID corresponde a um voo ou reserva, já que pela forma que estes dados estão distribuídos pela HashTable, apresentariam tempo de execução O(1), e por fim verificamos se o ID pertence a um user, pois o tempo de execução é maior.

Query 2 - Voos e Reservas de um User

Verificamos se o user existe, e se existir, cria um array dinâmico inicialmente vazio. Dependendo do argumento opcional, adicionamos ou não os voos e as reservas do user a esse array dinâmico. Por fim, usamos uma função de ordenação (quickSort) para ordenar o array dinâmico.

Query 3 - Classificação média de um Hotel

Durante o processamento das reservas, para além de as adicionarmos num array dinâmico de reservas do Hotel, adicionamos todos os ratings dessas reservas. Quando esta query é chamada, primeiro verifica se o

rating médio do hotel foi calculado. Se foi, devolve a classificação média, se não foi, dividimos o somatório de ratings das reservas do hotel pelo número de reservas que o hotel tem.

Query 4 - Reservas de um Hotel

Verificamos se o hotel existe, e se existir, usamos uma função chamada ordenaReservasHotel que ordena as reservas do referido hotel e obtemos o array dinâmico que esse hotel tem que contém todas as suas reservas ordenadas.

Query 5 - Listar Voos de um Aeroporto

Verificamos se o aeroporto existe, e se existe, obtemos um array dinâmico que contém todos os voos desse aeroporto. Criamos um array dinâmico, fazemos uma procura linear nos voos do aeroporto e insere-se nesse array dinâmico apenas os voos que estão entre as datas dadas pela query. Após essa procura, usamos a função quickSort para ordenar esse array, dos voos mais recentes aos mais antigos.

Query 6 - Top N Aeroportos com mais passageiros

Criamos um array dinâmico e inserimos todos os aeroportos juntamente com o número de passageiros que aquele aeroporto tem no ano especificado pelo argumento da query. Após isso usamos o quickSort para ordenar esse array dinâmico. No fim só exibimos os primeiros N elementos desse array dinâmico, sendo N o número dado pelo argumento da query.

Query 7 - Top N Aeroportos com a maior Mediana

Criamos um array dinâmico e inserimos todos os aeroportos juntamente com a mediana de atrasos dos voos desse aeroporto. Após isso usamos o quickSort para ordenar esse array dinâmico. No fim só exibimos os primeiros N elementos desse array dinâmico, sendo N o número dado pelo argumento da query.

Query 8 - Receita Total de um Hotel

Verificamos se o hotel existe, e se existe, obtemos um array dinâmico que contém todas as reservas desse hotel. Após isso, calculamos o somatório das receitas totais de cada reserva desse hotel entre as datas especificadas pelos argumentos. Após isso, exibimos esse somatório.

Nota: caso uma reserva não esteja dentro das datas, a sua receita total será 0

Query 9 - Listar os Users que contêm o mesmo Prefixo

Criamos um array dinâmico e fazemos uma procura linear nos Users e inserimos no array dinâmico apenas os Users cujo nome tem o prefixo dado pelo argumento da query. Após isso, ordena esse array dinâmico com o quickSort e exibe esse array.

Query 10 - Métricas gerais da Aplicação

Criamos uma HashTable onde cada elemento será uma data. Essas datas podem ser anos, meses do mesmo ano ou dias no mesmo mês, dependendo do argumento opcional. São feitas 3 procuras lineares nos dados dos Users, Reservas e Voos. Primeiro obtemos a data desse dado, se a data não existe, cria uma data na HashTable. Se a data existe (ou após a sua criação), atualiza o número de dados nessa data aumentando 1 unidade. *(Exemplo: User foi criado em 1/1/2000 e as métricas são os anos. Na data "Ano 2000", o número de users será incrementado)*

Após essas procuras, colocamos os elementos da HashTable num array dinâmico e ordenamos esse array. Fazemos o processamento de todos os elementos do array dinâmico onde obtemos a quantidade de Users, Reservas, Voos e Passageiros aconteceram naquela data.

Para obter a quantidade de passageiros únicos, definimos o bool "unico_passageiro" como falso, vamos a todos os voos que aconteceram naquela data e iremos definir o bool "unico_passageiro" de cada passageiro como verdadeiro. Por fim obtemos a quantidade de Users cujo bool "unico_passageiro" é verdadeiro e exibimos esse número como sendo o número de passageiros únicos naquela data.

Testes funcionais e de desempenho

O programa de testes recebe 3 argumentos: o caminho para o dataset com os ficheiros de entrada, o caminho para o ficheiro de comandos e, por fim, o caminho para a pasta com os ficheiros de output esperados.

Este programa verifica se existem diferenças entre os outputs obtidos e os esperados. Para isso, chama a função verificacaoOutputs, que recebe 2 argumentos: o caminho para os outputs esperados e o caminho para o ficheiro de comandos. Nessa função, são criados dois arrays dinâmicos, onde num deles são guardados os caminhos até cada ficheiro de comando esperado e no outro são guardados os caminhos correspondentes a cada ficheiro de resultado obtido.

De seguida, é chamada a função comparaFich, que recebe como argumentos um elemento de cada array, ou seja, o caminho até ao output esperado do comando X e o caminho até ao resultado obtido para o comando X. Nessa função, ambos os ficheiros são analisados linha a linha e caso seja detetada alguma diferença, a função retorna uma mensagem indicando a linha do comando X em que se encontra o erro. Caso contrário, retorna uma mensagem a informar que não foram encontradas incongruências entre os dois ficheiros.

Além disso, também é apresentado o tempo de execução de cada query para os vários comandos.

Para analisar o tempo de execução das queries, executamos o programa de testes em 2 máquinas diferentes e apontamos os valores do tempo de execução médio de cada query. Por outras palavras, como cada query é executada mais de uma vez na lista de comandos, determinamos o valor médio desses tempos de execução para cada query. Com esses dados, elaboramos 2 gráficos, um para cada máquina, onde se pode observar os diferentes desempenhos para as queries.

Atualizações

Da fase 1 para a fase 2 foram realizadas algumas alterações cruciais, entre elas as seguintes:

- Encapsulamento
- Utilização de estruturas de tipo genérico
- As árvores binárias foram substituídas por HashTables, pois as árvores binárias demoravam bastante tempo
- Removida ordenação dos dados
- Uso de parsers genéricos em vez de parsers específicos
- Funções das queries e do modo batch foram organizadas e separadas, estando muito melhor apresentado
- São exibidas mais informações sobre o modo batch e o processamento dos dados
- Query 3 executa mais depressa
- Funções sobre estatística e funções auxiliares foram removidas pois, ou quebravam encapsulamento, ou funções eram apenas usadas por um módulo *(por exemplo: o cálculo da mediana só é usado no módulo dos Voos)*
- Estrutura dos dados alterada para gastar menos memória
- Voos agora possuem uma lista com todos os seus passageiros
- Incorporação de todos os aeroportos, e não apenas aqueles onde os voos decolam
- Removida LimpaLeaks

Conclusão

Ao longo deste projeto foram desenvolvidas novas aprendizagens, particularmente, ao nível de trabalho em grupo, programação em C, desenvolvimento de projetos com uso da biblioteca, *ncurses*, uso adequado de memória (como a verificação de memory leaks), a importância que certas técnicas e estruturas podem fazer no desempenho dos programas, como o tempo necessário para a execução do mesmo, e compreensão dos conceitos de encapsulamento e modularidade, os quais nos eram desconhecidos até agora.

Como já se esperava, com o desenvolvimento do projeto, surgiram certas dificuldades, tais como: a gestão de memória, técnicas de encapsulamento e também a navegação na biblioteca *ncurses*. Contudo, tentamos sempre interpretar estas dificuldades como pontos a melhorar.