Universidade Do Minho
Licenciatura Em Engenharia Informática

LI3

Grupo 51

Diogo Ferreira (A104266) David Figueiredo (A104360)

Bernardo Moreira (A104349)

Ano Letivo 2023/2024

Conteúdo

Capítulo 1

Introdução e desafios

Este projeto consistiu no desenvolvimento de um Sistema de Gestão de utilizadores, voos e reservas na linguagem de programação C. Sendo um projeto em grande escala foi necessário a utilização de estruturas de dados eficientes para armazenar e consultar grandes quantidades de informação. As estruturas de dados utilizadas no projeto foram criadas por nós ou pertencem à biblioteca GLib, no caso as Hash Tables.

O projeto encontra-se dividido em duas fases sendo que a primeira fase concentra-se mais na implementação do parsing dos dados e no modo batch. Este modo envolve a consulta de dados armazenados num ficheiro de texto que é dado como argumento na execução do programa.

Capítulo 2

Módulos e estruturas de dados

2.1 Utilizadores

Para armazenar todos os utilizadores fornecidos nos ficheiros de texto foi utilizada Hash Table, inserindo cada utilizador na Hash Table deixando como chave, para ir procurar futuramente esse utilizador, o seu id.

2.2 Reservas

Para armazenar todas as reservas fornecidas pelos ficheiros de texto, a estratégia utilizada foi semelhante à dos utilizadores, usando na mesma Hash Table, mas agora a chave utilizada foi o id da reserva.

2.3 Voos

Para armazenar todos os voos fornecidos pelos ficheiros de texto, foi também utilizada Hash Table, deixando como chave o id de voo para futuros acessos à Hash Table.

2.4 Passageiros

Para armazenar os passageiros foi utilizada uma estratégia diferente dos restantes dados, uma vez que para estes foi utilizado um array de structs.

2.5 Queries

Para esta primeira fase do projeto foram escolhidas para resolução as queries 1, 2, 3, 4, 5 e 8.

A **query 1** consiste em listar o resumo de um utilizador, voo, ou reserva, consoante o identificador recebido por argumento. A query deverá retornar as seguintes informações em cada caso:

- Utilizador

nome;sexo;idade;código_do_país;passaporte;número_voos;número_reservas;total_gasto (name;sex;age;country_code;number_of_-ights;number_of_reservations;total_spent)

- Voo

companhia;avião;origem;destino;partida_est;chegada_est;número_passageiros;tempo_atraso(airline;plane_model;origin;destination;schedule_departure_date;schedule_arrival_date;passengers;delay)

- Reserva

id_hotel;nome_hotel;estrelas_hotel;data_início;data_fim;pequeno_almoço;número_de_noites;preço_total(hotel_id;hotel_name;hotel_stars;begin_date;end_date;includes_breakfast;nights;total_price)

A **query 2** tem como objetivo 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 seu tipo (flight ou reservation).

Comando
2 <ID> [flights|reservations]
Output
id;date[;type]
id;date[;type]
...

A **query 3** consiste em apresentar a classificação média de um hotel, a partir do seu identificador (id).

A **query 4** deve listar as reservas de um hotel, ordenadas pela sua data (da mais recente para a mais antiga). Caso duas reservas tenham a mesma data, deve ser usado o id da reserva como critério de desempate (o id menor deve aparecer primeiro).

A **query 5** listar os voos com origem num dado aeroporto, entre duas datas, ordenados por data de partida estimada (da mais antiga para a mais recente). Caso dois voos tenham a mesma data, o seu id deve ser usado como critério de desempate (o id menor deve aparecer primeiro).

A **query 8** deve apresentar a receita total gerada por um hotel entre duas datas (inclusive), a partir do seu identificador. As receitas do hotel devem levar em consideração apenas o preço por noite entre as datas fornecidas.

2.6 Outputs

Na realização dos outputs foi tido em conta que estes devem ser escritos num ficheiro de texto e que podem ter dois formatos diferentes. O formato desejado no output é especificado em cada input fornecido sendo que a sua ausência faz com que o output deva ser apresentado da seguinte forma:

```
F000000123;2023/10/06;flight
R000000456;2023/10/02;reservation
F000000121;2023/10/01;flight
```

caso no input seja apresentado o caracter 'F' o formato deverá ser:

```
--- 1 ---
id: F000000123
date: 2023/10/06
type: flight

--- 2 ---
id: R000000456
date: 2023/10/02
type: reservation

--- 3 ---
...
```

Capítulo *3*

Dificuldades sentidas

Este trabalho não é de fácil execução sendo que a maior dificuldade nesta primeira fase foi o armazenamento e consulta das informações, bem como a implementação da modularidade e do encapsulamento.

Capítulo *3*

Conclusões

Concluindo, apesar das dificuldades que foram surgindo ao longo do projeto, acreditamos que tentamos ao máximo realizar um bom projeto que infelizmente não apresentou os resultados esperados pelos elementos envolvidos na realização do projeto. Sentimo-nos preparados para na segunda fase aprimorar o trabalho realizado na primeira fase e também ultrapassar os desafios que foram apresentados na segunda fase do projeto.