Departamento de Informática
Licenciatura em Engenharia Informática
Laboratórios de Informática III

Universidade do Minho
Escola de Engenharia

Trabalho Prático
Laboratórios de Informática III
Fase 1

Grupo 82

Filipa Oliveira da Silva A104167

Maria Cleto Rocha A104441

Mário Rafael Figueiredo da Silva A104182

Braga, novembro 2023

Índice

Introdução

Este projeto está a ser desenvolvido no âmbito da unidade curricular de Laboratórios de Informática III no ano letivo 2023/2024, unidade esta que pretende dar a consolidação aos alunos de conhecimentos essenciais da linguagem C, e de Engenharia de Software, mais precisamente modularidade e encapsulamento.

O projeto está dividido em duas fases. A primeira fase consiste em realizar o *parsing* dos dados e o modo *batch*. Este modo consiste em executar várias *queries* sobre os dados de forma sequencial, estando esses pedidos armazenados num ficheiro de texto cujo caminho é recebido como argumento.

Assim, o objetivo é não só interpretar os dados dos ficheiros recebidos, mas também armazená-los em estruturas de dados que facilitem o seu acesso.

Na segunda fase ser-nos-á pedido para completarmos o resto dos requisitos do programa. Programa este que visa torná-lo capaz de realizar todas as *queries*, testes funcionais e de performance, verificação do conjunto de dados fornecidos e criação de um modo interativo.

Desenvolvimento

Como é bastante notável, este projeto não é fácil, mas mesmo assim esforçamo-nos para entregar o melhor trabalho possível. Começamos, então, por organizar a arquitetura de como seria a ligação entre os diferentes módulos e a maneira como estão relacionados entre si, resultando no seguinte:

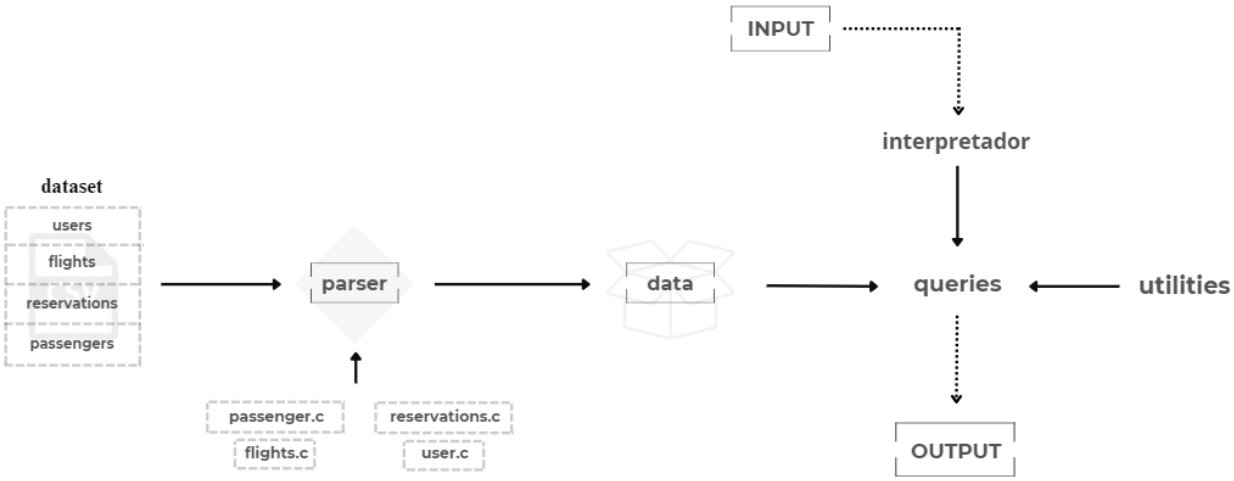

Figura 1: Diagrama da arquitetura utilizada no projeto

Deparamo-nos com vários desafios, especialmente porque foi difícil decidir por onde iríamos começar. Tentamos, por isso, perceber qual seria a estrutura mais adequada para armazenar os dados, acabando por optar por *structs* e *arrays dinâmicos*, uma vez que é aquilo com que estamos mais familiarizados a trabalhar. Assim, criamos uma *struct* para cada um dos parâmetros: *users*, *reservations*, *flights* e *passengers*; e tratamos do *parsing* dos dados. Tendo em conta a modularidade e o facto de recebermos ficheiros CSV para serem lidos, decidimos criar uma função geral "*parser"* que chama funções doutros ficheiros associados ao *parsing* de cada um dos parâmetros dados. Depois, tratamos de guardar cada erro existente em cada um dos ficheiros em outros CSV, e os dados prontos a serem lidos em *arrays*.

Depois disto, tínhamos predefinido realizar seis *queries*, mas não conseguimos alcançar esse objetivo e, por isso, tendo em conta a simplicidade das *queries* de modo a ajudar-nos, decidimos tentar resolver as seguintes:

Query 1: consiste em *"listar o resumo de um utilizador, voo, ou reserva, consoante o identificador recebido por argumento"* – onde o identificador corresponde ao *user_id*, para os *users* e *reservas*, e o *id* para os *voos*. Assim, começamos por criar uma função que identifique qual o *id* a que o input se refere. Depois, é chamada a própria função da *query* que vai separar os valores, imprimindo-os num ficheiro de texto.

Query 3: consiste em *"apresentar a classificação média de um hotel, a partir do seu identificador",* isto é, recebemos o *id* de um determinado hotel como input e, de seguida, somamos todas as *ratings* associadas a esse hotel dividindo pelo número de avaliações, de forma a calcular a sua média. Como esta *query* é um pouco mais simples de desenvolver, não foi necessário criar nenhuma função auxiliar.

Query 4: consiste em *"listar as reservas de um hotel, ordenadas por data de início (da mais recente para a mais antiga)"*, sendo que é utilizado o identificador da reserva como critério de desempate no caso em que duas reservas têm a mesma data. Para isto, criamos inicialmente três funções, uma para calcular o número de noites que constituem a reserva, outra para calcular o preço total da estadia, e a terceira para organizar as reservas da mais recente para a mais antiga, tal como pede no enunciado, e para tratar o caso de empate.

Query 9: consiste em *"listar todos os utilizadores cujo nome começa com o prefixo passado por argumento, ordenados por nome (de forma crescente)"*, sendo que é utilizado o identificador do utilizador como critério de desempate no caso de dois utilizadores terem o mesmo nome. Nesta *query* temos também em conta que utilizadores que tenham o seu parâmetro *estado* como "inativo" não são considerados. Assim, criamos uma função auxiliar para comparar os utilizadores e verificar se tem o mesmo prefixo, e que organiza os dados tendo em conta a ordem pedida. Na própria *query*, fazemos as devidas verificações relativas aos utilizadores.

Dificuldades sentidas

Acreditamos que a maior dificuldade sentida foi conseguir implementar a modularidade e o encapsulamento, especialmente utilizando a linguagem C. Apesar disso, e de não termos conseguido cumprir completamente com o objetivo, estamos bastante orgulhosos com o nosso empenho e dedicação e, para além de incompleto, com o resultado do nosso projeto, pois como é a primeira vez a tratar deste tipo de temas, compreendemos que não haja tanta facilidade e sucesso no produto final. Esperamos, por isso, conseguir melhorar os aspetos onde não nos saímos tão bem na segunda fase e, assim, entregar um trabalho com melhor execução.

Conclusão

Resumindo, apesar das dificuldades e dos desafios que nos foram colocados, sentimo-nos preparados para o que aí vem e, como tal, para concluir este projeto e aperfeiçoar o que fizemos até agora, sendo que isto já nos deu conhecimento sobre aspetos importantes a ter em conta na programação, como os conceitos de modularidade e encapsulamento, que são fundamentais para um código de qualidade.