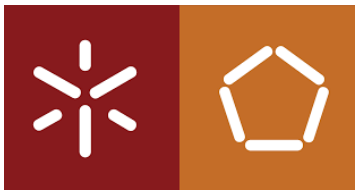


Relatório 1ª Fase | Laboratórios de Informática III
Grupo 1 | 2023/2024

Hélder Gomes (A104100) **João Lobo (A104356)** **Rita Camacho (A104439)**

Índice

1. Introdução

O presente relatório apresentará informações relativas à 1ª Fase do Projeto Final da Unidade Curricular Laboratórios de Informática III, pertencente ao 2º Ano da Licenciatura em Engenharia Informática, realizada no ano letivo 2023/2024, na Universidade do Minho.

O projeto final referido consiste na implementação de uma base de dados em memória, utilizando-se ficheiros *.csv* fornecidos pelos professores, constituídos por dados relativos a viagens (voos, reservas, passageiros, etc.), sendo o objetivo principal armazená-los e implementar métodos de pesquisa e associações entre estes.

São aplicados neste trabalho conceitos fundamentais como modularidade, encapsulamento, reutilização e abstração de dados.

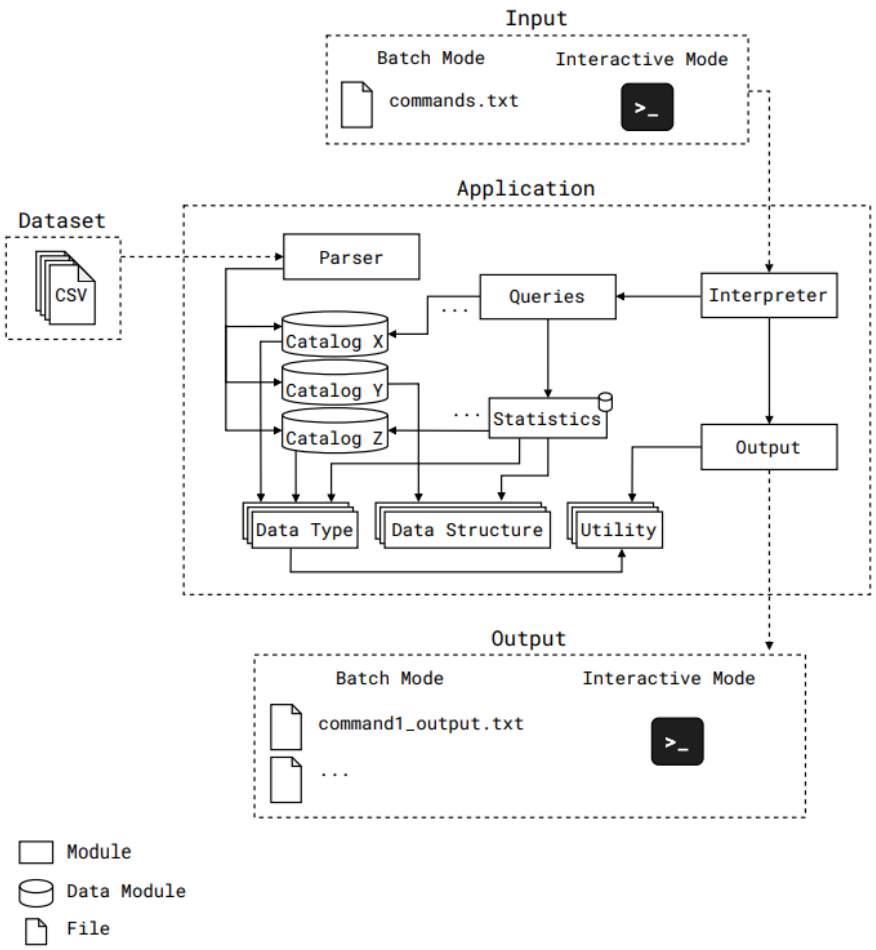
Na 1ª fase agora concluída, era necessário implementar o *parsing* e validação dos dados de entrada e o modo *batch*, bem como a realização de 6 *queries* das 10 propostas pela equipa docente.

2. Desenvolvimento

No projeto atual foi implementado o *parsing* de dados de entrada e o modo *batch*, sendo possível executar as *queries* sobre os dados sequencialmente, encontrando-se esses pedidos guardados num ficheiro de texto cujo caminho é recebido como argumento. Assim, requer não só a leitura e interpretação dos dados presentes nos ficheiros, como o seu respetivo armazenamento em estruturas de dados flexíveis e de eficiente acesso. Para tal, foi permitida a utilização da biblioteca *glib*, fornecendo implementações pré-definidas de estruturas de dados úteis.

2.1. Estrutura Aplicada

No desenvolvimento do trabalho, utilizou-se a seguinte estrutura proposta pelos docentes:



2.2. Funcionamento do Projeto

A aplicação, através do número de argumentos recebidos, seleciona o seu modo de funcionamento. Nesta fase foi apenas implementado o modo *batch*, sendo apenas este analisado nesta secção.

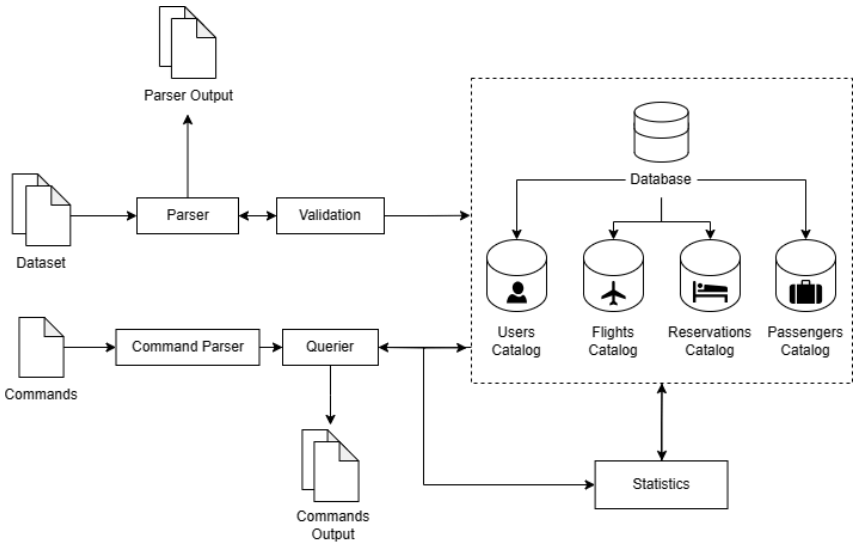
No início da aplicação, dentro dos catálogos do módulo *database*, são imediatamente criadas todas as estruturas necessárias para armazenar os dados que, posteriormente, serão preenchidas pelos próximos passos.

Através do primeiro argumento (caminho para os ficheiros *.csv* do *dataset*) é populada a base de dados que contém os catálogos de cada entidade. A leitura de dados do ficheiro é realizada através de um parser genérico que recebe apontadores para funcões que fazem o tratamento, validação e inserção dos dados nos catálogos. Caso a validação dos dados falhe, é chamada uma função de output de erros, também passada como argumento ao *parser*.

Seguidamente, é realizado o *parsing* dos comandos que obtém o índice da *query* a ser chamada, o modo de *output* dos dados e os argumentos da *query*. As *queries* utilizam funções de listagem e obtenção de dados disponibilizadas pelos catálogos e, no caso de *queries* que tratem de entidades com associações, utilizam o módulo *statistics*, responsável pela gestão e criação de estruturas de dados que interligam entidades.

Através de um módulo de *writing* genérico com modos de *output* tipo *csv* e tabular (*flag* F), com funções que recebem um número variável de argumentos, respetiva formatação e nome dos campos, as *queries* vão escrevendo as *"respostas"* num buffer de ficheiro, transferido para o disco no final da execução do comando.

Após a execução do programa, é feita uma limpeza da memória, eliminando todas as estruturas previamente criadas pelos módulos e, consequentemente, eliminados os dados do *dataset*, evitando *memory leaks*.



2.3. *Queries*

Das 10 *queries* apresentadas, foram escolhidas as 6 seguintes para realizar na 1ª fase:

2.3.1. *Query* 1

Lista o resumo de um utilizador, voo, ou reserva, consoante o identificador recebido por argumento. Como as entidades foram gravadas em memória através de tabelas de *Hash* (*GHashTable*), a implementação desta *query* foi trivial, visto que os IDs entre entidades têm formatações diferentes, tornando a escolha e procura acessíveis dentro das estruturas de dados utilizadas. A maior dificuldade foi com dados relativos a associações entre várias entidades, como passageiros, reservas de um utilizador, etc. No entanto, após a implementação do módulo *statistics*, este problema resolveu-se.

2.3.2. *Query* 2

Lista 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, apresenta voos e reservas, juntamente com o tipo (night ou reservation). Em caso de empate, ordena pelo identificador (de forma crescente). Através do módulo *statistics*, que utiliza *arrays* de apontadores (*GPtrArray*) dos catálogos, recolheu-se associações inter utilizadores e entidades, sendo apenas necessário filtrar e ordenar os dados após isso.

2.3.3. *Query* 3

Apresenta a classificação média de um hotel, a partir do seu identificador. Foi realizada uma tabela de *Hash* (*GHashTable*) intermédia, sendo a chave o ID do hotel e o valor uma *array* de apontadores (*GPtrArray*), para tornar a procura das avaliações mais simples e eficiente.

2.3.4. *Query* 4

Lista as reservas de um hotel, ordenadas por data de início (da mais recente para a mais antiga). Caso duas reservas tenham a mesma data, deve ser usado o identificador da reserva como critério de desempate (de forma crescente). Foi realizada uma tabela de *Hash* (*GHashTable*) intermédia, sendo a chave o ID do hotel e o valor uma *array* de apontadores (*GPtrArray*), para tornar a procura das reservas referentes a um hotel mais eficaz e rápida.

2.3.5. *Query* 5

Lista 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 identificador do voo deverá ser usado como critério de desempate (de forma crescente). Para facilitar a procura e *sorting* dos voos escolhemos utilizar uma *array* de apontadores para as entidades (*GPtrArray*), já que a tabela de *Hash* criada inicialmente não seria uma boa solução para este problema, sabendo que, por natureza, estas estruturas não são construídas para executar processos iterativos sobre os seus elementos.

2.3.6. *Query* 9

Lista todos os utilizadores cujo nome começa com o prefixo passado por argumento, ordenados por nome (lexicograficamente). Caso dois utilizadores tenham o mesmo nome, deverá ser usado o seu identificador como critério de desempate (de forma crescente). Utilizadores inativos não deverão ser considerados pela pesquisa. Foi utilizada uma *array* de apontadores (*GPtrArray*), para tornar a procura dos nomes mais rápida e eficaz, visto que a tabela de *Hash* previamente implementada tem um tempo de iteração muito lento e ineficiente.

2.4. Otimizações e Limitações

As únicas possíveis otimizações pensadas a serem futuramente implementadas são diminuição de memória utilizada e do tempo de execução, embora consideremos que já obtemos ótimos resultados a esse nível. No que toca à diminuição de memória utilizada, poderíamos ter removido alguns dados nas *structs* para já inutilizados, mas consideramos que, por uma questão de exigência da sua existência por parte dos docentes, não faria muito sentido, especialmente se os mesmos vierem a ser utilizados em futuras *queries*.

2.5. Dificuldades Sentidas

Algumas dificuldades que fomos enfrentando ao longo da realização do projeto foram a resolução de *memory leaks* nem sempre óbvias e contínua adaptação de estruturas de dados ao longo da criação de novas *queries*.

3. Conclusão

Concluindo, mesmo tendo enfrentado alguns desafios, o desenvolvimento da 1ª Fase foi realizado da melhor forma possível, encontrando-se concluídas corretamente todas as tarefas obrigatórias atribuídas à mesma.

Acreditamos que a nossa organização nos permitiu obter uma base estável para a 2ª fase do projeto, tendo sido possível graças à nossa dedicação, aprendizagem de novos conceitos como gestão de memória, encapsulamento e modularidade, e conhecimento cada vez mais aprofundado da linguagem de programação *C*.

No entanto, existem ainda possíveis melhorias que poderemos aplicar, sendo um dos nossos objetivos a ter em conta durante o desenvolvimento da 2ª fase.