Universidade do Minho
Escola de Engenharia

Relatório do Projeto de LI3 - 1ª Fase

Grupo 41

Trabalho realizado por:

André Campos a104618

Beatriz Peixoto a104170

Ana Oliveira a104536

novembro de 2023

Índice

Introdução

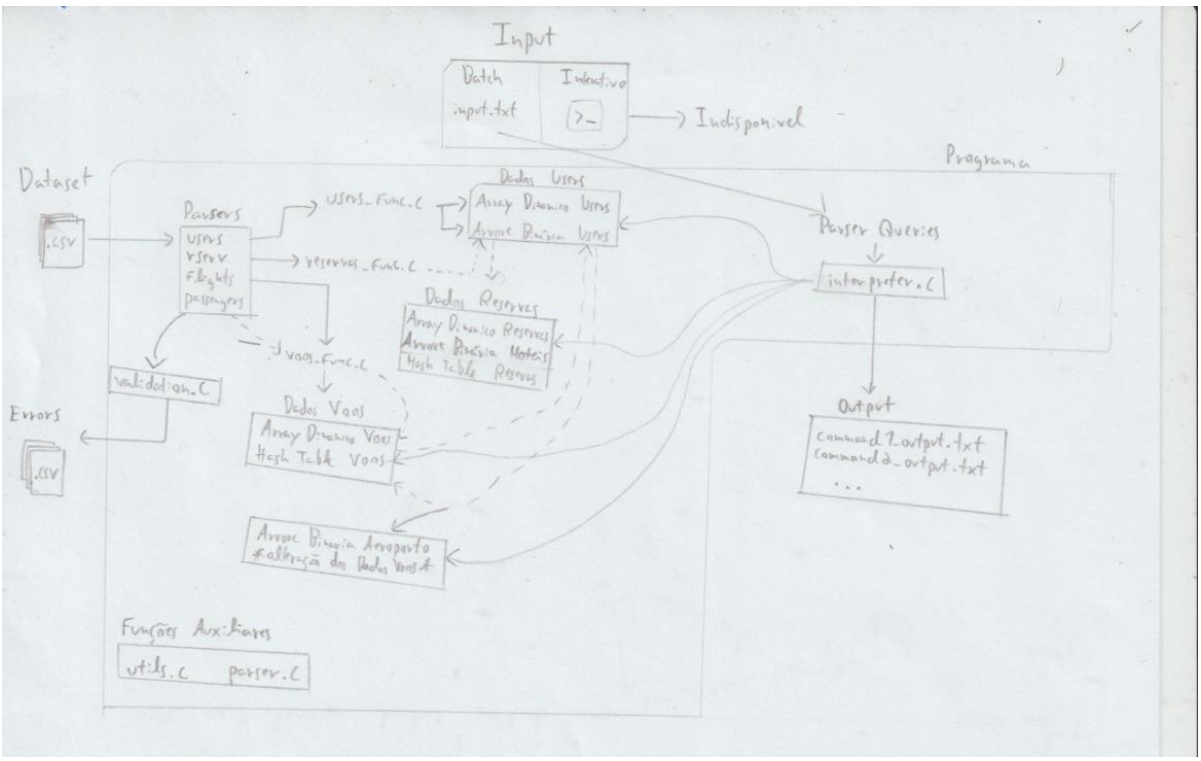
No âmbito da UC de Laboratórios de Informática III, do 2º ano da licenciatura em Engenharia Informática, foi realizado o projeto aqui apresentado, com o objetivo de consolidar aprendizagens de C e utilização de ferramentas cruciais para o desenvolvimento de projetos em C.

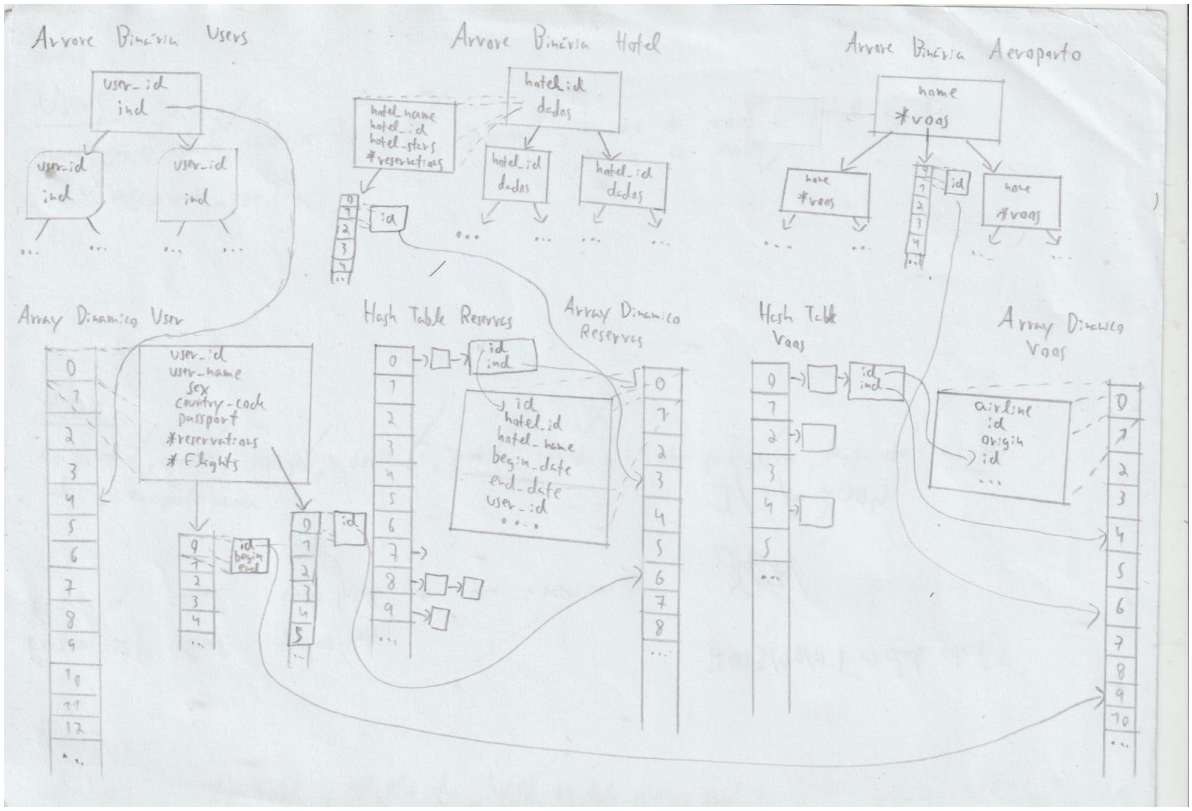
Nesta primeira fase do projeto, a avaliação irá focar no parsing dos ficheiros de entrada, modo batch, 6 das 10 queries apresentadas, validação do dataset e, por fim, ausência de memory leaks na execução do programa.

Este relatório irá começar por abordar a estrutura/arquitetura do projeto, onde iremos, de seguida, discutir o código partindo do programa principal. Depois disso, debatendo as funções relativas aos users, reservas e voos, falaremos também das funções de validação, seguido da resolução das queries.

Por fim, iremos refletir sobre as nossas dificuldades ao longo do desenvolvimento deste projeto, terminando com aspetos do nosso código que pretendemos tentar melhorar na 2ª fase.

Arquitetura





Código

Programa Principal

É no ficheiro “programa-principal.c” que se encontra a função main deste projeto.

Esta função recebe 2 argumentos, “n_args”, um inteiro que corresponde ao número de argumentos que o utilizador insere no terminal, e “args”, correspondente a uma lista de strings com cada argumento.

O programa começa por indicar ao utilizador como colocar os argumentos de forma a aceder a cada modo de execução (“Modo Iterativo: 0 argumentos\nModo Batch: <ficheiros .csv das entradas> <ficheiro .txt das queries>\n”).

De seguida, utiliza um “if” para descobrir se o número de argumentos introduzidos (n_args) foi igual a 1, sendo este remetente ao modo iterativo.

Se este “if” se cumpria, como o modo iterativo não se encontra disponível, o programa apenas imprimia um aviso de que eesse modo não se encontra disponível, aconselhando o utilizador a tentar o modo batch.

Caso este “if” não se cumpra, utiliza-se um outro “if” que verifica se o “n_args” é igual a 3, desta vez remetente ao modo batch, que procedia à sua execução. Assim sendo, inicializam-se 3 novas variáveis referentes ao número de dados: lidos, incorporados no programa e rejeitados do programa, todas de valor 0. A função procede a percorrer todos os dados e aplica o parser, imprimindo de seguida o total de dados lidos, dados incorporados e dados rejeitados. Agora que apenas se encontram na execução os dados válidos, são lidas as queries e aplica-se o parserQueries, que toma como argumento a lista de queries, abre o

ficheiro destas, lê cada uma e chama a função interpretaQuerys, que examina a query, descobre de qual se trata e chama a sua função correspondente. Logo após, prodece a limpar todas as árvores e arrays correspondentes aos dados e imprime uma mensagem de encerramento do programa.

Por fim, no caso de nenhum dos if's anteriores se cumprir, significa que o número de argumentos apresentados não corresponde ao batch nem ao iterativo, imprimindo, portanto, uma mensagem de erro.

Users_func.c, Reservas_func.c e Voos_func.c

Estes ficheiros são relativos à criação das estruturas de armazenamento e processamento destes dados, ou seja, dos utilizadores, das reservas e dos voos.

Validação

Na função validação.c/.h, foram criadas várias funções que auxiliam à validação de um utilizador, reserva ou voo.

Essas funções são: data_validation e hora_validation que, tal como o nome indica validam uma data e hora, respetivamente, recorrendo à criação das structs data e hora, que separam a string da data ou hora em 3 inteiros correspondentes a, no caso da data, ano, mês e dia; existe também a user_validation, que pega na struct do utilizador e verifica as condições necessárias para que este seja válido (por vezes com o auxílio de outras funções como a quebraHora e quebraData, assim como as funções data_validation e hora_validation, referidas anteriormente); além disso, temos também a reservation_validation e a flight_validation que funcionam de forma semelhante à user_validation e verificam as condições necessárias para se considerar válida a reserva e o voo, respetivamente.

Queries

Query 1

A função query1, a partir de um user id, flight id ou id de reserva, vai listar toda a informação sobre esse utilizador, voo ou reserva, respetivamente. Assim sendo, ela procura o id do user. Se este dá um nº positivo ou 0, esse user existe, sendo este nº o indice do user no array dinâmico e imprime este user; se der um outro nº, significa que esse user não existe, e procede a utilizar a mesma lógica para procurar o id nos voos e de seguida reservas.

Query 2

Nesta função, copia-se para um array as reservas e os voos do user pedido. De seguida ordena-se o array consoante a data. Por fim, dá-se print do array ordenado. Caso no argumento opcional apareça voos, não coloca nesse array as reservas ou vice-versa.

Query 3

Consegue-se o rating de todas as reservas de um hotel, sendo este correspondente à avaliação dos users, e faz-se a média destas.

Query 4

Na query4, são listadas as reservas de um hotel, ordenadas por data de início. Dá como output o id do hotel, a data de entrada e de saída, o id do utilizador, o rating do utilizador e o preço total que ele pagou pela reserva.

Query 7

Esta função calcula a mediana dos atrasos de cada aeroporto e lista esses aeroportos por ordem decrescente de atrasos.

Query 9

Na query9, são ordenados os utilizadores que possuem todos um certo prefixo, pelo nome. Caso tenham o mesmo nome, ordenam-se pelo id.

Dificuldades e Limitações

Iremos agora referir algumas dificuldades que encontramos ao longo do desenvolvimento deste projeto.

Uma das maiores dificuldades foi, de facto organizar e dividir as tarefas entre os membros do grupo. Além disso, ocorreram-nos alguns erros incomuns, os quais tivemos alguma dificuldade a resolver.

Aspetos a Desenvolver/Melhorar na 2ª Fase

Iremos ter em conta as dificuldades mencionadas anteriormente e tentar melhorar sobre isto, particularmente no que toca à organização do grupo. Além disso, temos a noção de que algum código deve ser revisado e simplificado.