Projeto Laboratórios de informática

1º Fase

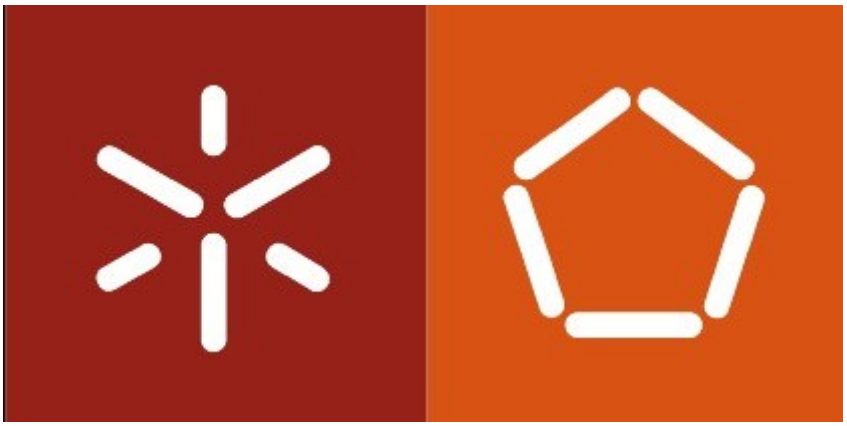
Desenvolvido pelo grupo 45:

Afonso Pedreira (104537)

Dário Guimarães (A104344)

Hugo Rauber (A104534)

Licenciatura em Engenharia informática



Departamento de informática

Universidade do Minho

1. Introdução

O presente relatório refere-se à primeira fase do desenvolvimento do projeto proposto, que visa a criação de um programa de gestão e consulta de dados relacionados a

utilizadores, voos e reservas. Esta fase inicial concentra-se na criação da arquitetura da aplicação, na implementação das funcionalidades básicas e na validação dos dados, conforme especificado nos requisitos fornecidos.

A proposta visa fornecer uma plataforma robusta e funcional que permita aos usuários consultar e extrair informações essenciais dos conjuntos de dados disponibilizados, bem como garantir a integridade e validade das informações processadas.

Durante esta etapa inicial, foi dado ênfase à definição de uma arquitetura modular, seguindo os princípios de modularidade e encapsulamento discutidos nas diretrizes do projeto. Além disso, foram implementadas algumas das funcionalidades iniciais de consulta, cujos resultados foram submetidos aos testes dados, visando garantir a sua exatidão e adequação aos requisitos estabelecidos.

Este relatório, dividido em capítulos, aborda detalhes sobre a estrutura da aplicação, as funcionalidades implementadas até o momento, os testes realizados para validar o correto funcionamento das consultas(etc). Além disso, são discutidos os módulos desenvolvidos, ressaltando a aplicação dos conceitos de modularidade.

Este relatório representa uma etapa inicial do projeto, oferecendo uma visão abrangente do progresso alcançado até o momento e servindo como base para a próxima fase de desenvolvimento, onde novas funcionalidades e otimizações serão implementadas.

2. Estrutura da Aplicação

Este capítulo destina-se a oferecer uma visão detalhada da organização e arquitetura do sistema desenvolvido. Este capítulo é fundamental para compreender como é que os vários módulos e componentes acabam por se interligar para criar a funcionalidade global do programa.
A estrutura da aplicação é fundamentada na modularidade , princípio essencial para criar um sistema organizado, de fácil manutenção e extensível. Neste projeto, esta estrutura materializa-se em diversos ficheiros e pastas dentro do código-fonte.

Passa-se a descrever brevemente como se organizam as pastas em questão e a sua estrutura:

A pasta **src** abriga a maior parte do código, com cada ficheiro dedicado a funcionalidades específicas. Os ficheiros estão divididos em pastas, facilitando a compreensão e manutenção do sistema.

Os ficheiros de cabeçalho (**include**) definem as estruturas de dados essenciais para o funcionamento do programa. Essas estruturas representam entidades como utilizadores, voos, reservas, entre outras, conforme descritas no enunciado do projeto.

Os **catalogs** são preenchidos e organizados pelos ficheiros **catalogs filler** e **catalogs creator**, com a finalidade de armazenar e gerenciar informações provenientes dos ficheiros **CSV.** O **file parser** é responsável por analisar e interpretar os dados desses ficheiros, preenchendo as estruturas definidas e populando os catálogos.

O **interpreter** processa os comandos fornecidos, interpretando cada comando e direcionando a linha lida para as funções específicas de cada query.

As funções relacionadas a cada **query** estão separadas em ficheiros individuais para manter a organização e facilitar a manutenção. Estes ficheiros contêm as lógicas necessárias para executar cada operação de acordo com a especificação do enunciado.

A pasta **utils** contém funções utilitárias compartilhadas entre diferentes partes do projeto, promovendo a reutilização de código e evitando a redundância.

Há ficheiros específicos para lidar com a escrita de erros (**write errors**) e a geração dos resultados (**write output**). O ficheiro **values parser** é responsável por validar e adicionar valores às estruturas de dados apropriadas.

O ficheiro **main** é o ponto de entrada do programa, coordenando a execução, chamando as funções adequadas e orquestrando o fluxo do sistema.

Essa estrutura modularizada e bem organizada do código permite uma implementação mais clara e compreensível, além de facilitar a identificação e correção de problemas, caso surjam. A separação de responsabilidades em diferentes ficheiros e pastas contribui para um código mais legível, facilitando o desenvolvimento e a manutenção do software.

3. Funcionalidades Implementadas

Aqui, abordar-se-ão, as **Funcionalidades Implementadas.** É essencial descrever as operações e capacidades que o programa oferece ao utilizador com base nos requisitos do projeto. Isso inclui detalhar as funcionalidades específicas que foram desenvolvidas para atender às necessidades enunciadas.

As funcionalidades implementadas podem ser organizadas conforme a especificação das queries e operações exigidas. Cada uma dessas funcionalidades está associada a uma query específica, e pode ser explicada da seguinte forma:

- **(Q1) Listar informações de um Utilizador, Voo ou Reserva** : Permite obter um resumo detalhado de um utilizador, voo ou reserva com base no identificador fornecido, excluindo utilizadores com status "inactive".
- **(Q2) Listar Voos ou Reservas de um Utilizador** : Permite listar voos e/ou reservas de um usuário, ordenando-os por data ou tipo.
- **(Q3) Apresentar Classificação Média de um Hotel** : Obtém a classificação média de um hotel a partir do seu identificador.
- **(Q4) Listar Reservas de um Hotel** : Lista as reservas de um hotel ordenadas por data de início.
- **(Q5) Listar Voos de um Aeroporto entre Datas** : Apresenta voos com origem num determinado aeroporto entre datas específicas.
- **(Q6) Listar Top Aeroportos com mais Passageiros:** Gera uma lista dos principais aeroportos com mais passageiros num determinado ano.
- **(Q7) Listar Top Aeroportos com a Maior Mediana de Atrasos** : Identifica os principais aeroportos com a maior mediana de atrasos nos voos.
- **(Q8) Apresentar Receita Total de um Hotel entre Datas** : Calcula a receita total de um hotel entre datas específicas com base nas reservas.
- **(Q9) Listar Utilizadores por Prefixo de Nome** : Lista todos os utilizadores cujo nome começa com um determinado prefixo, ordenando-os por nome.
- **(Q10) Apresentar Métricas Gerais da Aplicação** : Fornece várias métricas gerais, incluindo número de novos utilizadores, voos, passageiros, passageiros únicos e reservas, agregados por ano, mês ou dia.

4. Validação e Testes Funcionais

No capítulo de **Validação e Testes Funcionais**, descreve-se como os dados foram validados, assegurando que estão corretos e aderem aos padrões estabelecidos. Além disso, os testes funcionais foram necessários para garantir que as funcionalidades implementadas operavam como esperado. Assim, aqui está uma descrição sobre isto:

- **Validação dos Dados**:
- **Tratamento de Dados Inválidos**:
- **Testes Funcionais**:
- **Estratégia de Testes**:
- **Output de Resultados de Testes**:
- **Medição de Desempenho dos Testes**:

5. Módulos de Utilidade

O projeto desenvolvido adota uma estrutura modular que facilita a gestão e reutilização de funcionalidades comuns, com uma implementação direta dos módulos de utilidade. A separação de tarefas em módulos distintos permite uma abordagem mais organizada e coesa no código.

- **Catálogos e Hashtables:**
    - Os catálogos e hashtables funcionam como estruturas de dados principais para armazenar e gerenciar informações relacionadas a utilizadores, voos, reservas e outros elementos fundamentais do sistema. Essas estruturas, implementadas de acordo com as especificações fornecidas, facilitam o acesso eficiente e a manipulação dos dados.
    -
- **Parsers e Criadores de Catálogos:**
    - Os parsers de ficheiros são responsáveis por ler os dados dos ficheiros CSV e efetuar o processamento inicial, garantindo a correta interpretação dos valores e o preenchimento das estruturas de dados adequadas. Os criadores de catálogos utilizam essa informação para construir e organizar os catálogos e hashtables necessários para a execução das queries.

- **Interpretador e Funcionalidades de Queries:**
    - O interpretador analisa os comandos recebidos e direciona para as respetivas funcionalidades associadas a cada query. Cada funcionalidade de query é encapsulada em módulos separados, garantindo a separação lógica e permitindo a expansão ou modificação de funcionalidades de forma independente.

- **Escrita e Armazenamento de Resultados:**
    - Os módulos responsáveis por escrever os resultados em ficheiros são cruciais para a correta apresentação das saídas geradas pelo programa. Eles organizam os resultados de acordo com os padrões estabelecidos e armazenam-nos na estrutura de pastas apropriada.

- **Utilitários Compartilhados:**
    - Os utilitários (utils) guardam funções comuns ou recorrentes, promovendo a reutilização de código e simplificando o desenvolvimento. Estas funções abrangem operações como validação de valores, manipulação de strings, entre outras, utilizadas em vários pontos do projeto.

Embora ainda não tenha sido implementado o encapsulamento completo, a estrutura modular clara evidencia a modularidade do código, facilitando o entendimento,

manutenção e expansão do projeto. A próxima fase visa implementar o encapsulamento para melhorar ainda mais a estruturação do código-fonte.