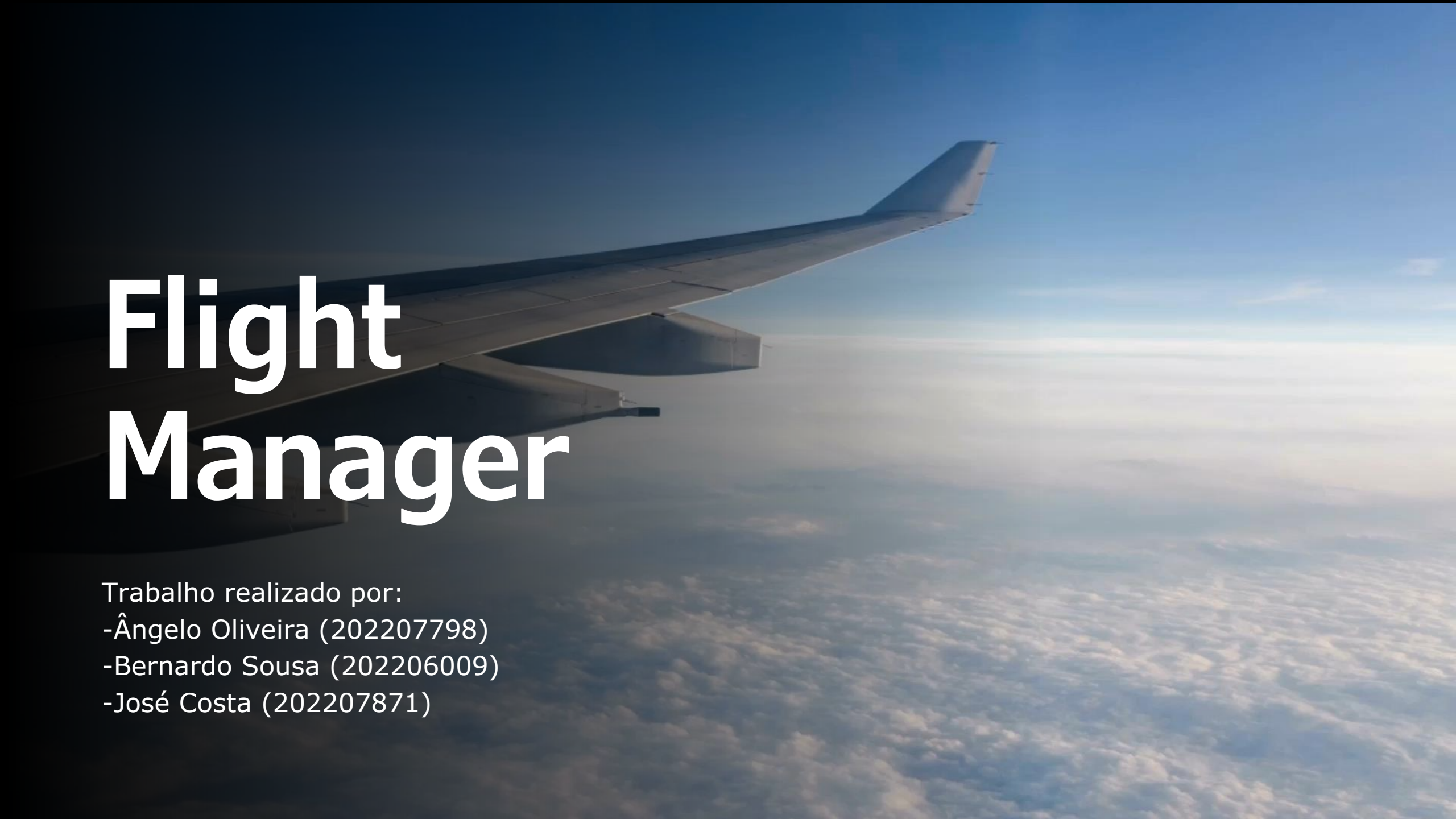

Flight Manager

Trabalho realizado por:
-Ângelo Oliveira (202207798)
-Bernardo Sousa (202206009)
-José Costa (202207871)

Classes

- Graph : Responsável por definir a estrutura algébrica usada como base para a elaboração do projeto. Altamente influenciada pela definição de grafo dada em aula.

- Airport: Classe que serve de informação aos vértices do grafo. Possante de atributos como código, nome, cidade, país e coordenadas de localização.

- Airline: Parâmetro de voo, permitindo diferenciar voos com origens e destinos idênticos que seria de outra forma impossível. Papel fulcral na elaboração de filtros, sendo a principal opção de escolha para o utilizador para personalizar a sua viagem. Definido por nome, callsign, código e país de origem.

Classes

Logic: Usada como principal responsável pela elaboração dos métodos que compõem a aplicação, desde estatísticas à elaboração de um itinerário de viagem. Atributos compostos por um grafo, mapa de aeroportos e mapa de Airlines (nomes para códigos).

LoadingFunctions: Ponto de partida da nossa aplicação, trata do processamento dos dados fornecidos nos ficheiros .csv lendo-os e organizando-os nas estruturas por nós escolhidas(unordered set).

UI: Interface servil do utilizador para a navegação do programa.

LoadingFunctions

- Classe possante de dois unordered_sets que contem os aeroportos e as Airlines.

- Loading das funções feito a partir de 3 funções principais sendo cada uma responsável por processar os aeroportos, Airlines e voos presentes nos csv.

- A leitura dos csv é feita, partindo cada linha em tokens que correspondem aos atributos de cada classe, construindo progressivamente um novo objeto da classe em questão, sendo depois inserido nas estruturas respetivas.

```
void LoadingFunctions::LoadAirports(Graph<Airport>& g) {
    std::ifstream file("dataset/airports.csv");
    if (!file.is_open()) {
        std::cerr << "Failed to open the CSV file." << std::endl;
    }

    std::string line;
    getline(file, line);

    while (getline(file, line)) {
        std::istringstream lineStream(line);
        std::vector<std::string> tokens;
        std::string token;

        while (getline(lineStream, token, ',')) {
            tokens.push_back(token);
        }

        Airport airport = Airport(tokens[0] , tokens[1] , tokens[2] , tokens[3] , stod(tokens[4]))

        if(!g.addVertex(airport)){
            std::cerr << "Failed to add vertex to graph" << std::endl;
        }

        airports.insert(airport);
```

```
    if (!file.is_open()) {
        std::cerr << "Failed to open the CSV file." << std::endl;
    }

    std::string line;
    getline(file, line);

    while (getline(file, line)) {
        std::istringstream lineStream(line);
        std::vector<std::string> tokens;
        std::string token;

        while (getline(lineStream, token, ',')) {
            tokens.push_back(token);
        }

        std::string temp = tokens[3];
        NormalizeString(temp);

        Airline airline = Airline(tokens[0] , tokens[1] , tokens[2] , temp);

        airlines.insert(airline);
    }
    file.close();
```

Descrição do Grafo

- Vertex -> Informação correspondente a um aeroporto;
- Vetor de arestas, isto é, viagens entre aeroportos;
- Campos visited e processing para suporte a algoritmos de pesquisa no grafo;
- Campos low e num usados em procura de pontos de articulação;
- Indegree correspondente ao numero de voos que nele tem fim.

```
class Vertex {
    T info;                // contents
    vector<Edge<T> > adj;  // list of outgoing edges
    bool visited;          // auxiliary field
    bool processing;       // auxiliary field
    int low;               // auxiliary field
    int indegree;          // auxiliary field
    int num;               // auxiliary field
    Vertex<T>* parent;

    void addEdge(Vertex<T> *dest, double w, string airline);
    bool removeEdgeTo(Vertex<T> *d);
```

- Edge -> Representa um voo;
- Vértice equivalente a ao destino do voo;
- Peso representativo da distancia;
- Airline responsável pelo voo em questão.

```
template <class T>
class Edge {
    Vertex<T> * dest;      // destination
    double weight;         // edge weight
    string airline;        // airline code
public:
    Edge(Vertex<T> *d, double w,string air
    Vertex<T> *getDest() const;
    void setDest(Vertex<T> *dest);
    double getweight() const;
    void setweight(double weight);
    std::string getAirline()const ;
    void setAirline(string airline);
    friend class Graph<T>;
    friend class Vertex<T>;
```

Descrição do Grafo

- Grafo -> Estrutura algébrica responsável pela organização da rede de voos;
- Vetor de referencias aos vértices que os constituem;
- Presente nele grande parte das funções que servem de base à elaboração das estatisicas apresentadas na aplicação.

```
template <class T>
class Graph {
    vector<Vertex<T> *> vertexSet;  // vertex set
    stack<Vertex<T>> stack_;        // auxiliary field
    list<list<T>> list_sccs;        // auxiliary field

    void dfsVisit(Vertex<T> *v,  vector<T> & res) const;
    bool dfsIsDAG(Vertex<T> *v) const;
public:
    Vertex<T> *findVertex(const T &in) const;
    int getNumVertex() const;
    int calculateDiameter() const;
    bool addVertex(const T &in);
    bool removeVertex(const T &in);
    bool addEdge(const T &sourc, const T &dest, double w,
    bool removeEdge(const T &sourc, const T &dest);
    vector<Vertex<T> * > getVertexSet() const;
    vector<T> dfs() const;
    vector<T> dfs(const T & source) const;
    vector<T> bfs(const T &source) const;
    void bfsDifferent(T &source) const;
    vector<T> topsort() const;
    bool isDAG() const;
    void calculateIndegrees() const;
```

Funcionalidades Implementadas

A nível de global:

-Número Total de Aeroportos O(1)
-Número Total de Voos O(V)

A nível de aeroporto:

Número de destinos possíveis, sob a forma de aeroporto, cidade, país O(V + E)

Top k aeroportos que constituem o maior trafego aéreo; O(V*log(V) + V*E)

Aeroportos essenciais para o funcionamento da rede aérea.O(V+E)

A nível de cidade/airline:

-Número Total de voos de uma certa airline. O(V+E)

Número Total de de voos que envolvam uma cidade O(V)

A nível de país:

Número Total de países alcançáveis a partir de um certo aeroporto ou cidade O(V + E)

Funcionalidades Implementadas

Consulta de voos

Por cidade

Por aeroporto

Por airline

Funcionalidades Implementadas

- Planejamento de viagens

Objetivo: Encontrar o melhor voo para o utilizador, considerando este o com menos número de escalas

Diferentes interpretações de local de partida e de destino:
Aeroporto;
País;
Cidade;
Coordenadas.

Filtros de preferência para o utilizador:
As Airlines que queira USAR na sua viagem;
As Airlines que queira EVITAR na sua viagem.

Para que seja possível disponibilizar as diferentes opções de partida/chegada, são procurados os aeroportos que fazem mais sentido ser consideradas conforme a escolha

UI

- Objetivo principal: providenciar uma interface “user-friendly” que permita a qualquer pessoa, mesmo que leiga ao objetivo do trabalho, usufruir do mesmo.
- Inclui um menu principal, a partir do qual damos as opções de consultar voos, estatísticas, planear uma viagem ou cessar o uso da aplicação.

```
:menu_start() {
op;
<< "################################################" << endl
<< "@@@@@  @@@@@  @@@@   @@@@@   @@@   @@@@   @@@@@  " << endl
<< "@   @    @    @@ @@  @@ @@  @   @  @@ @@    @    " << endl
<< "@@@@@    @    @@@@   @@@@@  @   @  @@@@    @    " << endl
<< "@   @    @    @@ @@  @@     @   @  @@ @@    @    " << endl
<< "@   @  @@@@@  @@ @@  @@      @@@   @@ @@    @    " << endl
<< "################################################" << endl
<< "Welcome to the Air Travel Flight Management System, what w
<< "A. Proceed to the application" << endl
<< "B. Close the application" << endl
<< "Insert the letter: " ;

:main_menu(){
 op;
 << "What would you like to know?" << endl;
 << "A. Consult Flights" << endl
 << "B. Consult Flight Statistics" << endl
 << "C. Plan a Trip" << endl
 << "D. Exit the Application" << endl
 << "Insert your choice:";
```

UI

- Uma vez escolhida a opção do menu principal, é então pedido ao utilizador que especifique, dentro de cada opção possível, para especificar o que deseja

```
void UI::statistics_menu(){
    char op;
    cout << "What statistics would you like to see?" << endl
         << "A. Global Statistics" << endl
         << "B. Airport Statistics" << endl
         << "C. Airline/City Statistics" << endl
         << "D. Country Statistics" << endl
         << "E. Consult longest possible flights" << endl
         << "Insert your choice:";
```

```
void UI::flight_consultation() {
    char op;
    cout << "How would you like to search for your flight? " << endl
         << "A. Consult number of flights out of an airport and from how many different airlines;"
         << "B. Consult number of flights per city/airline " << endl
         << "Insert your choice: ";
```

```
void UI::trip_planner(){
    char op;
    cout << "How would you like to chose your starting point?" << endl
         << "A. Origin Airport (You may use the name or the code of the airport)" << endl
         << "B: Origin City (We'll pick all airports in the city)" << endl
         << "C. Origin Country (We'll pick all airports in the country)" << endl
         << "D. Origin Coordinates (We'll pick the closest airport/s to the given coordinates)"
         << "Insert your option:";
```

Destaque de Funcionalidades

- O trabalho por nós realizado apresenta variadas funcionalidades, possibilitando aos utilizadores uma experiencia confortável, eficiente e esclarecedora.

- Na nossa opinião conjunta, consideramos que a parte que mais sentimos vontade de destacar é a possibilidade de aplicar filtro na elaboração de voos.

- Consideramos que as funções com esse fim ficaram bem realizadas, e a elaboração das mesmas foi , em conjunto com as restantes componentes, sem duvida, uma oportunidade de aprendizagem na manipulação desta nova estrutura, o grafo.

Dificuldades Sentidas

- Um dos maiores entraves que sentimos quanto à realização do projeto principalmente face ao anterior, foi a questão de termos que o fazer numa altura mais atribulada a nível de testes e exames.

- Apesar de não termos achado o trabalho excessivamente exigente, o facto de sermos forçados a trabalhar com uma nova estrutura causou algum desconforto no inicio do projeto. Foi sendo dissipado no passar dos dias passados a trabalhar.

- Ademais, a organização de informação na interface do utilizador também se mostrou ser algo complicado, tentar conciliar toda a informação que tínhamos disponíveis com manter o menu algo leve, navegável e intuitivo não foi tão imediato como esperávamos.