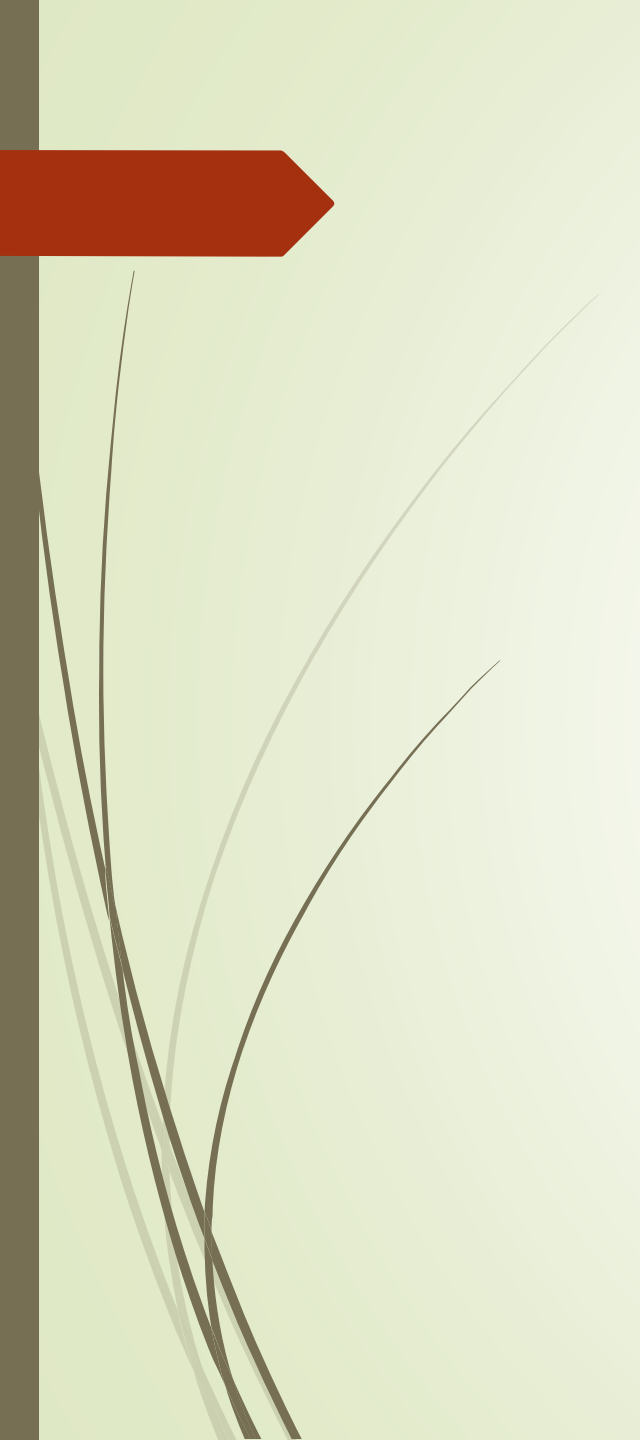


# Air Travel Flight Management System

David Carvalho [up202208654](#)  
Leonardo Magalhães [up202208726](#)  
Tiago Pinto [up202206280](#)



**“Ser programador é ser paciente e persistente na busca de novos conhecimentos.”**

*Igor Barros*

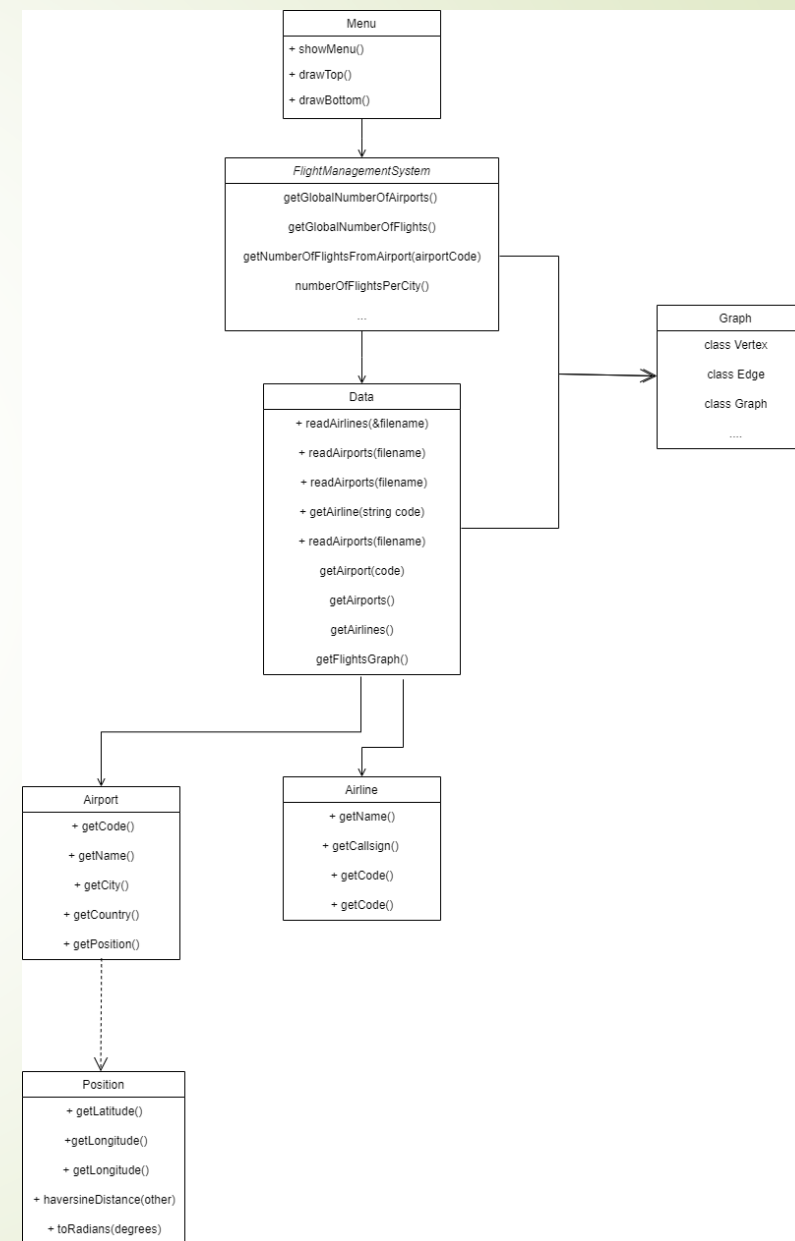
# Índice

- Introdução
- Diagrama de Classes
- Leitura do Dataset
- Estrutura do Grafo Utilizado
- Estruturas Utilizadas
- Funcionalidades e Algoritmos Implementados
- Interface com o Utilizador
- Destaque de Funcionalidades
- Principais Desafios e Esforço do Grupo
- Conclusão

# Introdução

- No contexto da disciplina de Algoritmos e Estruturas de Dados, empenhámo-nos na criação de um projeto que demonstra o nosso compromisso com a aplicação prática de conceitos avançados nesta área.
- Optámos por uma seleção criteriosa de algoritmos e estruturas de dados, bem como pela abordagem da resolução de problemas através da técnica "Divide and Conquer".
- Estamos confiantes de que o nosso projeto corresponderá e atingirá os objetivos propostos.

# Diagrama de Classes



# Leitura do Dataset

## ► Airlines Dataset:

Utilizamos o método `readAirlines` para ler o ficheiro "airlines.csv", usando a estrutura de dados `unordered_map` "Airlines" com informações como código, nome, callsign e país de cada companhia aérea.

## ► Airports Dataset:

A leitura do ficheiro "airports.csv" é realizada pelo método `readAirports`, preenchendo o `unordered_map` 'airports' com detalhes como código, nome, cidade, país, latitude e longitude para cada aeroporto.

# Leitura do Dataset

## ► Flights Dataset:

O método `createFlightsGraph` constrói um grafo de voos com base no ficheiro "flights.csv", considerando informações como origem, destino, companhia aérea e distância entre aeroportos.



# Estrutura do Grafo Utilizado

- O grafo que desenvolvemos é composto por vértices que representam os aeroportos e arestas que representam os voos entre esses aeroportos.

- Vértices (Aeroportos):

Cada vértice no grafo corresponde a um aeroporto.

As propriedades associadas a cada vértice incluem código do aeroporto, nome, cidade, país, latitude e longitude.

```
class Vertex {  
    string info;           ///< airport code  
    vector<Edge> adj;      ///< list of outgoing edges  
    bool visited;         ///< auxiliary field  
    bool processing;      ///< auxiliary field  
    int inDegree;         ///< auxiliary field  
    int outDegree;        ///< auxiliary field  
    int num;              ///< auxiliary field  
    int low;              ///< auxiliary field  
}
```



# Estrutura do Grafo Utilizado

- Arestas (Voos):

As arestas conectam os vértices e representam os voos entre aeroportos.

- Peso das Arestas:

As arestas têm um peso associado, representando a distância entre os aeroportos. Esse peso pode ser usado para otimizar rotas.

```
class Edge {  
    Vertex * dest;      // destination vertex  
    float distance;     // edge distance  
    string airline;
```



# Estrutura do Grafo Utilizado

- Este grafo fornece uma estrutura eficiente para modelar e analisar a conexão entre aeroportos, permitindo a implementação de algoritmos para planeamento de rotas, busca de voos otimizados e outras funcionalidades relevantes para um sistema de gestão de voos aéreos

```
class Graph {  
    vector<Vertex *> vertexSet;    // vertex set  
    int _index_;                  // auxiliary field  
    stack<Vertex> _stack_;        // auxiliary field  
    list<list<string>> _list_sccs_; // auxiliary field
```

# Estruturas Utilizadas

```
class Airport {  
private:  
    std::string code;           ///< The code of the airport.  
    std::string name;          ///< The name of the airport.  
    std::string city;          ///< The city where the airport is located.  
    std::string country;       ///< The country where the airport is located.  
    Position position;         ///< The geographic position of the airport.  
}
```

```
class Airline {  
private:  
    std::string code;           ///< The code of the airline.  
    std::string name;          ///< The name of the airline.  
    std::string callsign;       ///< The callsign of the airline.  
    std::string country;       ///< The country of the airline.  
}
```

# Estruturas Utilizadas

```
struct Route {  
    std::string source;  
    std::string target;  
    std::vector<std::string> airlines;  
  
    bool operator<(const Route& r) const {  
        if (source != r.source)  
            return source < r.source;  
        if (target != r.target)  
            return target < r.target;  
        return airlines < r.airlines;  
    }  
    bool operator==(const Route& r) const {  
        return source == r.source && target == r.target && airlines == r.airlines;  
    }  
};
```

# Funcionalidades e Algoritmos Implementados

```
int getGlobalNumberOfAirports() const;
int getGlobalNumberOfFlights() const;
int getNumberOfFlightsFromAirport(const std::string& airportCode) const;
int getNumberOfAirlinesFromAirport(const std::string& airportCode) const;
void numberOfFlightsPerCity() const;
void numberOfFlightsPerAirline() const;
int getNumberOfCountriesFromAirport(const std::string& airportCode) const;
int getNumberOfCountriesFromCity(const std::string& city, const std::string &country) const;
void numberOfReachableDestinationsFromAirport(const std::string &airportCode) const;
void numberOfReachableDestinationsFromAirportWithStops(const std::string &airportCode, int maxStops) const;
void getMaxTripWithStops();
int calcStopsBFS(Vertex *source, vector<std::pair<std::string, std::string>> &aux);
void getTopAirportWithMostTraffic(int k) const;
unordered_set<string> getEssentialAirports() const;
void printRoute(const Route& route) const;
```

# Funcionalidades e Algoritmos Implementados

```
vector<vector<Route>> findBestFlightOptions(const std::string& source, const std::string& destination) const;
void findBestFlightOptionsByAirportName(const std::string &source, const std::string &destination) const;
void findBestFlightOptionsByCity(const std::string &sourceCity, const std::string &sourceCountry, const std::string &destinationCity, const std::string &destinationCountry) const;
void findBestFlightOptionsByCoordinates(double latitude, double longitude, const std::string &destination) const;
void findBestFlightOptionsByAirportCodeToCityName(const std::string &source, const std::string &destinationCity, const std::string &destinationCountry) const;
void findBestFlightOptionsByAirportNameToCityName(const std::string &sourceName, const std::string &destinationCity, const std::string &destinationCountry) const;
void findBestFlightOptionsByAirportCodeToCoordinates(const string &source, double latitude, double longitude) const;
void findBestFlightOptionsByAirportNameToCoordinates(const string &sourceName, double latitude, double longitude) const;
void findBestFlightOptionsByCityToAirportCode(const string &sourceCity, const string &sourceCountry, const string &destinationCode) const;
void findBestFlightOptionsByCityToAirportName(const string &sourceCity, const string &sourceCountry, const string &destinationName) const;
void findBestFlightOptionsByCityToCoordinates(const string &sourceCity, const string &sourceCountry, double latitude, double longitude) const;
void findBestFlightOptionsByCoordinatesToAirportName(double latitude, double longitude, const string &destinationName) const;
void findBestFlightOptionsByCoordinatesToCity(double latitude, double longitude, const string &destinationCity, const string &destinationCountry) const;
void findBestFlightOptionsByCoordinatesToCoordinates(double sourceLatitude, double sourceLongitude, double destinationLatitude, double destinationLongitude) const;
```





# Funcionalidades e Algoritmos Implementados

```
void findBestFlightOptionsByAirportName(const std::string &source, const std::string &destination, const std::vector<std::string> &selectedAirlines) const;
void findBestFlightOptionsByCity(const std::string &sourceCity, const std::string &sourceCountry, const std::string &destinationCity, const std::string &destinationCountry, const std::vector<std::string> &selectedAirlines) const;
void findBestFlightOptionsByCoordinates(double latitude, double longitude, const std::string &destination, const std::vector<std::string> &selectedAirlines) const;
void findBestFlightOptionsByAirportCodeToCityName(const std::string &source, const std::string &destinationCity, const std::string &destinationCountry, const std::vector<std::string> &selectedAirlines) const;
void findBestFlightOptionsByAirportNameToCityName(const std::string &sourceName, const std::string &destinationCity, const std::string &destinationCountry, const std::vector<std::string> &selectedAirlines) const;
void findBestFlightOptionsByAirportCodeToCoordinates(const std::string &source, double latitude, double longitude, const std::vector<std::string> &selectedAirlines) const;
void findBestFlightOptionsByAirportNameToCoordinates(const std::string &sourceName, double latitude, double longitude, const std::vector<std::string> &selectedAirlines) const;
void findBestFlightOptionsByCityToAirportCode(const std::string &sourceCity, const std::string &sourceCountry, const std::string &destinationCode, const std::vector<std::string> &selectedAirlines) const;
void findBestFlightOptionsByCityToAirportName(const std::string &sourceCity, const std::string &sourceCountry, const std::string &destinationName, const std::vector<std::string> &selectedAirlines) const;
void findBestFlightOptionsByCityToCoordinates(const std::string &sourceCity, const std::string &sourceCountry, double latitude, double longitude, const std::vector<std::string> &selectedAirlines) const;
void findBestFlightOptionsByCoordinatesToAirportName(double latitude, double longitude, const std::string &destinationName, const std::vector<std::string> &selectedAirlines) const;
void findBestFlightOptionsByCoordinatesToCity(double latitude, double longitude, const std::string &destinationCity, const std::string &destinationCountry, const std::vector<std::string> &selectedAirlines) const;
void findBestFlightOptionsByCoordinatesToCoordinates(double sourceLatitude, double sourceLongitude, double destinationLatitude, double destinationLongitude, const std::vector<std::string> &selectedAirlines) const;
vector<vector<Route>> findBestFlightOptions(const std::string &source, const std::string &destination, const std::vector<std::string> &selectedAirlines) const;
```



# Funcionalidades e Algoritmos Implementados

```
vector<vector<Route>> findBestFlightOptionsWithFewestAirlines(const string &source, const string &destination) const;
static vector<Route> minimizeAirlines(const vector<Route>& routes);

void findBestFlightOptionsWithFewestAirlinesByAirportNameToAirportName(const string &sourceName, const string &destinationName) const;
void findBestFlightOptionsWithFewestAirlinesByAirportCodeToCity(const string &sourceCode, const string &destinationCity, const string &destinationCountry) const;
void findBestFlightOptionsWithFewestAirlinesByAirportNameToCity(const string &sourceName, const string &destinationCity, const string &destinationCountry) const;
void findBestFlightOptionsWithFewestAirlinesByAirportCodeToCoordinates(const string &source, double latitude, double longitude) const;
void findBestFlightOptionsWithFewestAirlinesByAirportNameToCoordinates(const string &sourceName, double latitude, double longitude) const;
void findBestFlightOptionsWithFewestAirlinesByCity(const string &sourceCity, const string &sourceCountry, const string &destinationCity, const string &destinationCountry) const;
void findBestFlightOptionsWithFewestAirlinesByCityToAirportCode(const string &sourceCity, const string &sourceCountry, const string &destinationCode) const;
void findBestFlightOptionsWithFewestAirlinesByCityToAirportName(const string &sourceCity, const string &sourceCountry, const string &destinationName) const;
void findBestFlightOptionsWithFewestAirlinesByCityToCoordinates(const string &sourceCity, const string &sourceCountry, double latitude, double longitude) const;
void findBestFlightOptionsWithFewestAirlinesByCoordinatesToAirportCode(double latitude, double longitude, const string &destination) const;
void findBestFlightOptionsWithFewestAirlinesByCoordinatesToAirportName(double latitude, double longitude, const string &destinationName) const;
void findBestFlightOptionsWithFewestAirlinesByCoordinatesToCity(double latitude, double longitude, const string &destinationCity, const string &destinationCountry) const;
void findBestFlightOptionsWithFewestAirlinesByCoordinatesToCoordinates(double sourceLatitude, double sourceLongitude, double destinationLatitude, double destinationLongitude) const;
```

# Interface com o Utilizador

## Visualização do Menu Inicial

```
-----  
|===== Menu =====|  
|-----|  
| 1. Get from airports |  
| 2. Statistics        |  
| 3. Best flight option|  
| 4. Personalized preferences|  
| 5. Smallest distance between two airports|  
| Q. Exit              |  
|-----|  
|=====|  
|-----|  
Choose an option:|
```

# Interface com o Utilizador

## Visualização de Dados sobre Aeroporto

```
-----  
|===== Menu =====|  
|-----|  
| 1. Global number of airports |  
| 2. Number of flights/airlines out of airport |  
| 3. Number of countries flown from airport |  
| 4. Number of reachable destinations from airport |  
| 5. Number of destinations from airport with stops |  
| 6. Top airports with most traffic |  
| 7. Essential airports |  
| Q. Exit |  
|-----|  
|=====|  
|-----|  
Choose an option:|
```

# Interface com o Utilizador

## Visualização de Estatísticas

```
-----  
|===== Menu =====|  
|-----|  
| 1.  Get global number of flights      |  
| 2.  Get number of flights per city    |  
| 3.  Get number of flights per airline |  
| 4.  Get number of countries flown from city |  
| 5.  Get max trip with stops           |  
| Q.  Exit                             |  
|-----|  
|=====|  
|-----|  
Choose an option:|
```

# Interface com o Utilizador

Pesquisar Melhor Opção de Voo

```
-----
|===== Menu =====|
|-----|
| 1. Find best flight option from airport |
| 2. Find best flight option from city   |
| 3. Find best flight option from coordinates |
| Q. Exit                               |
|-----|
|=====|
|-----|
Choose an option:|
```

```
-----
|===== Menu =====|
|-----|
| 1. To airport |
| 2. To city    |
| 3. To coordinates |
| Q. Exit      |
|-----|
|=====|
|-----|
Choose an option:
```

```
-----
|===== Menu =====|
|-----|
| 1. Code |
| 2. Airport Name |
| Q. Exit |
|-----|
|=====|
|-----|
Choose an option:|
```

# Interface com o Utilizador

Pesquisar Voo De acordo Com Preferências

```
-----  
|===== Menu =====|  
|-----|  
| 1. Best flight option with selected airlines |  
| 2. Best flight option with fewest airlines  |  
| Q. Exit                                     |  
|-----|  
|=====|  
|-----|  
Choose an option:|
```



# Destaque de Funcionalidades

- Sentimos um grande orgulho na eficiente leitura e processamento dos dados, resultando na criação de um grafo dinâmico de voos com informações cruciais. O controlo da complexidade temporal evidencia a nossa atenção meticulosa aos detalhes.
- Neste projeto, orgulhamo-nos também de ter feito todos os parâmetros pedidos.
- Para além disso, ainda incluímos uma funcionalidade extra que mede a menor distância entre dois aeroportos e indica qual o caminho de menor distância (em km).



# Principais Desafios e Esforço do Grupo

- Enfrentámos desafios iniciais na interpretação dos dados, que foram superados com esforço conjunto. A implementação precisa do grafo dinâmico e a otimização da complexidade temporal exigiram colaboração e a escolha cuidada de algoritmos, destacando a nossa capacidade de superar obstáculos no desenvolvimento deste sistema.
- Todos os elementos contribuíram para os muitos âmbitos deste projeto e o esforço de cada um foi equilibrado e essencial para o resultado desenvolvido e apresentado.

# Conclusão

- No âmbito do projeto proposto, dedicamo-nos totalmente à aplicação de conceitos avançados de algoritmos e estruturas de dados de forma eficaz e eficiente.
- Empenhámo-nos profundamente na análise de algoritmos, selecionando cuidadosamente as estruturas de dados mais apropriadas para assegurar o desempenho eficaz do projeto.
- Procurámos, igualmente, otimizar a complexidade temporal e espacial do código, conduzindo análises de complexidade e realizando testes de desempenho.



# Obrigado!