




# Projeto 2 AED

---

- Realizado por:
- Francisco Magalhães - up202007945
- Lucas Faria - up202207540
- Rodrigo Sousa -up202207292



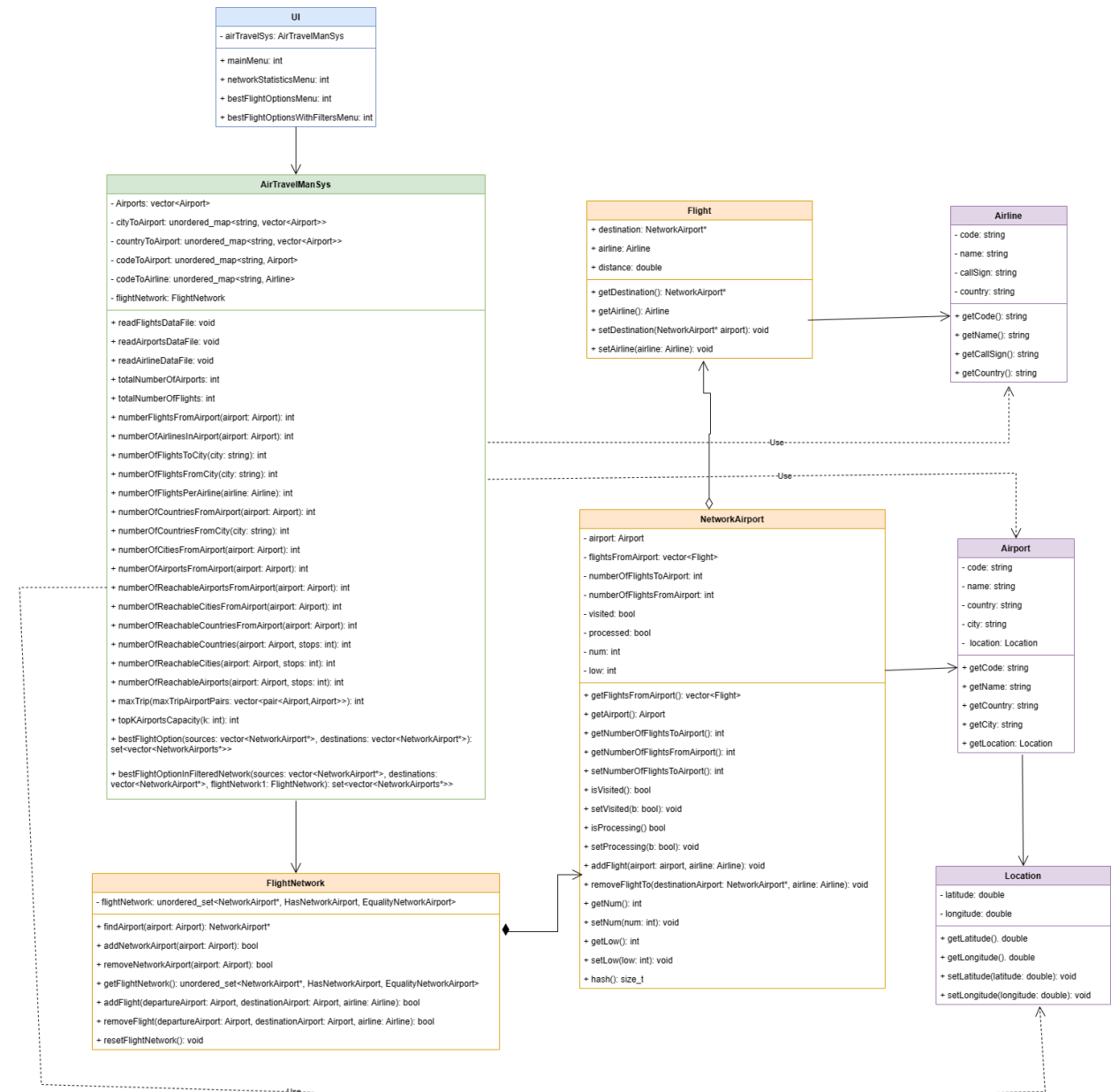
# Introdução

---

- Neste trabalho desenvolvemos um sistema de gerenciamento de viagens aéreas. Através deste sistema, o utilizador consegue saber algumas das estatísticas mais importantes relativas aos voos, além de conseguir encontrar a melhor opção de rotas para as suas viagens.
- Neste trabalho são implementados vários dos algoritmos e estruturas de dados abordados durante as aulas de forma a obter o melhor sistema possível.

# Diagrama UML de Classes

- **UI:** UI.h, UI.cpp;
- **AirTravelManSys:** AirTravelManSys.h, AirtravelManSys.cpp;
- **FlightNetwork:** FlightNetwork.h, FlightNetwork.cpp;
- **NetworkAirport:** FlightNetwork.h, NetworkAirport.cpp;
- **Flight:** FlightNetwork.h, Flight.cpp;
- **Airport:** Airport.h, Airport.cpp;
- **Airline:** Airline.h; Airline.cpp;
- **Location:** Location.h, Location.cpp.



# Leitura do Dataset

---

- Primeiramente, é lido o ficheiro com as Airlines.
- São guardadas numa Hash Table(`unordered_map`) mapeadas pelo seu código.
- Em segundo, é lido o ficheiro com os aeroportos.
- É preenchido o grafo com os nós formados pelos Aeroportos. Diversas Hash Tables auxiliares são criadas para ajudar em futuras funções.
- Por último, é lido o ficheiro que contém os voos. São acrescentados ao grafo como arestas entre dois aeroportos. Além disso, a cada voo é associado a uma Airline.

# Utilização de Grafos

---

- Foi utilizado um grafo neste trabalho formado por **3 classes**:
  - **FlightNetwork** – corresponde à classe **Graph** utilizada nas aulas. A grande diferença vem do facto de o vetor com os nós ter sido substituído por uma Hash Table(`unordered_set`), o que permite tempos de busca constantes.
  - **NetworkAirport** – corresponde à classe **Vertex** utilizada nas aulas. A estrutura é bastante similar mas com os nomes das variáveis e métodos adequados à funcionalidade do grafo.
  - **Flight**- Corresponde à classe **Edge** utilizada nas aulas. A estrutura também é bastante similar apesar de os nomes das variáveis e métodos usados terem nomes mais adequados à funcionalidade do grafo.

# Utilização de Grafos

- UML de classes do grafo:



# Funcionalidades

---

- As funcionalidades podem ser divididas em **3 tipos**:
  - **Estatísticas da Network;**
  - **Procura das melhores opções de voos;**
  - **Procura da melhores opções de voos filtrados por Airlines.**

# Funcionalidades – Estatísticas da Network

---

- **Nº de voos e aeroportos** – Complexidade  $O(1)$  para os aeroportos e  $O(N)$  para os voos;
- **Nº de voos que partem de um aeroporto e quantas Airlines diferentes atuam nesse aeroporto** - Complexidade  $O(1)$  para aeroportos  $O(n^2)$  para Airlines.
- **Número de voos por cidade e Airline** – Complexidade  $O(n)$  para cidades e  $O(n^2)$  para Airlines.
- **Número de diferentes países para os quais um Aeroporto ou Cidade têm voo direto** - Complexidade  $O(n \log(N))$  para aeroportos e  $O(n^2 \log(N))$  para cidades.
- **Número de destinos que é possível chegar em voos diretos desde de um aeroporto** – Complexidade  $O(n \log(N))$  para aeroportos e cidades e  $O(n^2 \log(N))$  para países.



# Funcionalidades – Estatísticas da Network

---

- **Número de destinos que é possível chegar de um aeroporto com numero ilimitado de paragens** – Complexidade  $O(n^2 \log(N))$  para cidades e países e  $O(n^2)$  para aeroportos. Usam algoritmos BFS para executar esta tarefa.
- **Número de destinos que é possível chegar num número limitado de paragens -** Complexidade  $O(n^2 \log(N))$  para cidades e países e  $O(n^2)$  para aeroportos. Usam algoritmos BFS para executar esta tarefa.
- **Distancia máxima entre um par de aeroportos** – Complexidade  $O(n^3)$ . Usa algoritmo BFS para auxiliar na procura.
- **Top aeroportos tendo em conta a sua capacidade aeroportuária-** Complexidade  $O(n \log(N))$ .
- **Aeroportos essenciais para a Network (articulation points)** – Complexidade  $O(n^2 \log(N))$ . Usa o algoritmo de Tarjan adaptado com recurso a um DFS.

# Funcionalidades – Procura das Melhores Opções de Voos

---

- A procura da melhor opção de voo foi dividida em **três partes**:
  - **Converter um input do utilizador** (localização, cidade, aeroporto) num conjunto de aeroportos que podem servir de local de partida e chegada. Esta parte foi realizada com uma complexidade  $O(1)$  para cidades e códigos de aeroportos,  $O(n)$  para nomes de aeroportos e  $O(n\log(n))$  para localizações.
  - **Encontrar a menor distância** entre o conjunto possível de aeroportos de partida e chegada. Foi usado um algoritmo BFS com complexidade  $O(n^2)$ .
  - **Descobrir o conjunto de caminhos possíveis** (nodes por onde é necessário passar). Foi usado um algoritmo BFS com complexidade  $O(n^2)$  que no pior dos casos é  $O(n^3)$  quando encontra um caminho.
- No geral, esta funcionalidade conta com uma complexidade que pode variar entre  $O(n^2)$  e  $O(n^4)$  dependendo do numero de aeroportos de origem e chegada são providos à função.

# Funcionalidades – Procura das Melhores Opções de Voos Filtradas por Airlines

---

- A procura da melhor opção de voo filtradas por Airline segue a mesma lógica que a procura sem filtros mas acrescenta um passo no início:
  - **Converte a network original numa nova filtrada por:**
    - **Airlines desejadas** – operação efetuada numa com uma complexidade  $O(N^2)$ ;
    - **Airlines indesejadas** – operação efetuada numa com uma complexidade  $O(N^2)$ ;

# Interface de Utilizador

---

- A interface de utilizador (classe UI) é composta por um conjunto de menus e submenus que permitem ao utilizador interagir com o sistema. Existem **4 menus principais** divididos depois em alguns submenus:
  - **Menu Principal** – primeiro menu com o qual o utilizador interage. Permite aceder a outros menus.
  - **Menu das Estatísticas da Network** – menu no qual o utilizador pode escolher qual função relacionada com as estatísticas da network quer executar. Reencaminha a execução para submenus que tratam de executar cada função de acordo com as suas características.
  - **Menu das melhores opções de voos** – menu onde o utilizador pode escolher um local de partida e um local de destino e obter as melhores opções de voos existentes.
  - **Menu da melhores opções de voos filtradas** - menu onde o utilizador pode escolher um local de partida, um local de destino e um conjunto de filtros (companhias aéreas indesejadas ou desejadas) e obter as melhores opções de voos existentes.

# Interface de Utilizador - Demonstração

---

Vamos ver alguns  
exemplos da utilização da  
Interface de utilizador

---

# Funcionalidades em Destaque

---

- Existem pelo menos **três funcionalidades** que gostaríamos de salientar:
  - A procura em tempo constante de aeroportos e cidades sem comprometer a estrutura do grafo;
  - A rápida procura pelas melhores opções de voos apesar do tamanho do DataSet;
  - O menu permitir uma fácil utilização e estar organizado de uma forma lógica.

# Dificuldades no desenvolvimento

---

- Podemos considerar que existiram principalmente **duas grandes dificuldades**:
  - Encontrar uma forma de ter tempos de busca constantes para aeroportos e cidades sem afetar a estrutura e funcionalidade do grafo;
  - Conseguir executar os melhores algoritmos com melhores complexidades nas funcionalidades mais complexas como a procura de aeroportos essenciais ou a busca pelas melhores opções de voos.
- Estas dificuldades foram superadas com a cooperação dos vários elementos do grupo para encontrar as melhores soluções possíveis.

# Conclusão



Neste trabalho tivemos a oportunidade de desenvolver um sistema de gestão de viagens aéreas.



Tivemos a oportunidade de aplicar na prática todos os conceitos apreendidos.



Assim, podemos concluir que este foi um trabalho proveitoso para a aprendizagem nesta unidade curricular.