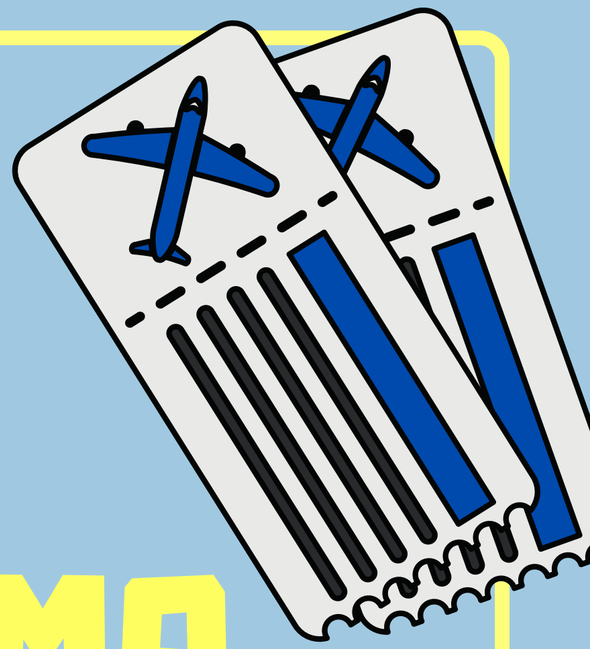
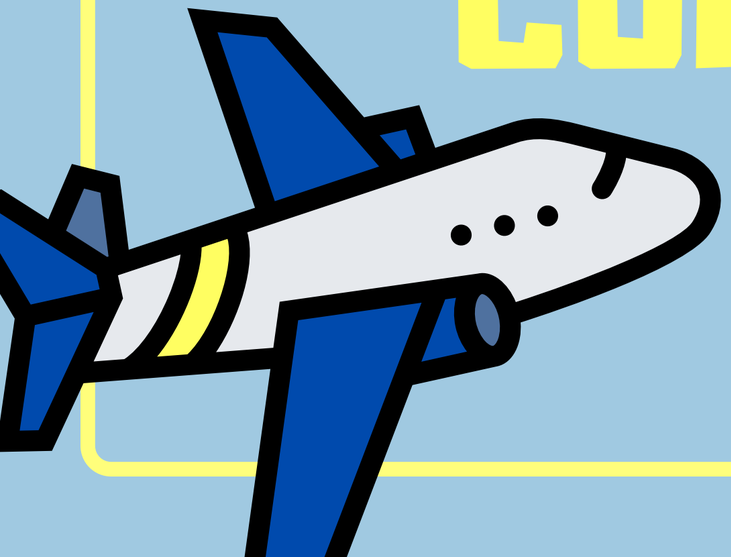


GESTÃO DE INFORMAÇÃO DE UMA COMPANHIA AÉREA



Turma 6 Grupo 60
Isabel Amaral - up202006677
Milena Gouveia - up202008862
Sofia Moura - up201907201

Descrição do problema

Implementar um sistema de gestão de informação inovador e personalizado às necessidades de uma companhia aérea.

O sistema tem como objetivo guardar e gerir informação relativa aos aviões, voos, passageiros e bagagens relacionados com a companhia.

Os aviões estão sujeitos a serviços de manutenção e limpeza, cada um atribuído a um funcionário. Ficam também registados os serviços que já foram realizados.

Os passageiros adquirem o seu bilhete, tendo em conta a ocupação dos lugares do voo que desejam, indicando se transportam bagagem ou não.

Dependendo da escolha do passageiro, a bagagem pode ser recolhida e transportada para o avião automaticamente através do check-in automático.

A companhia aérea dá aos passageiros, acesso a um conjunto de informações sobre os transportes terrestres mais próximos e a localização dos mesmos para cada aeroporto.

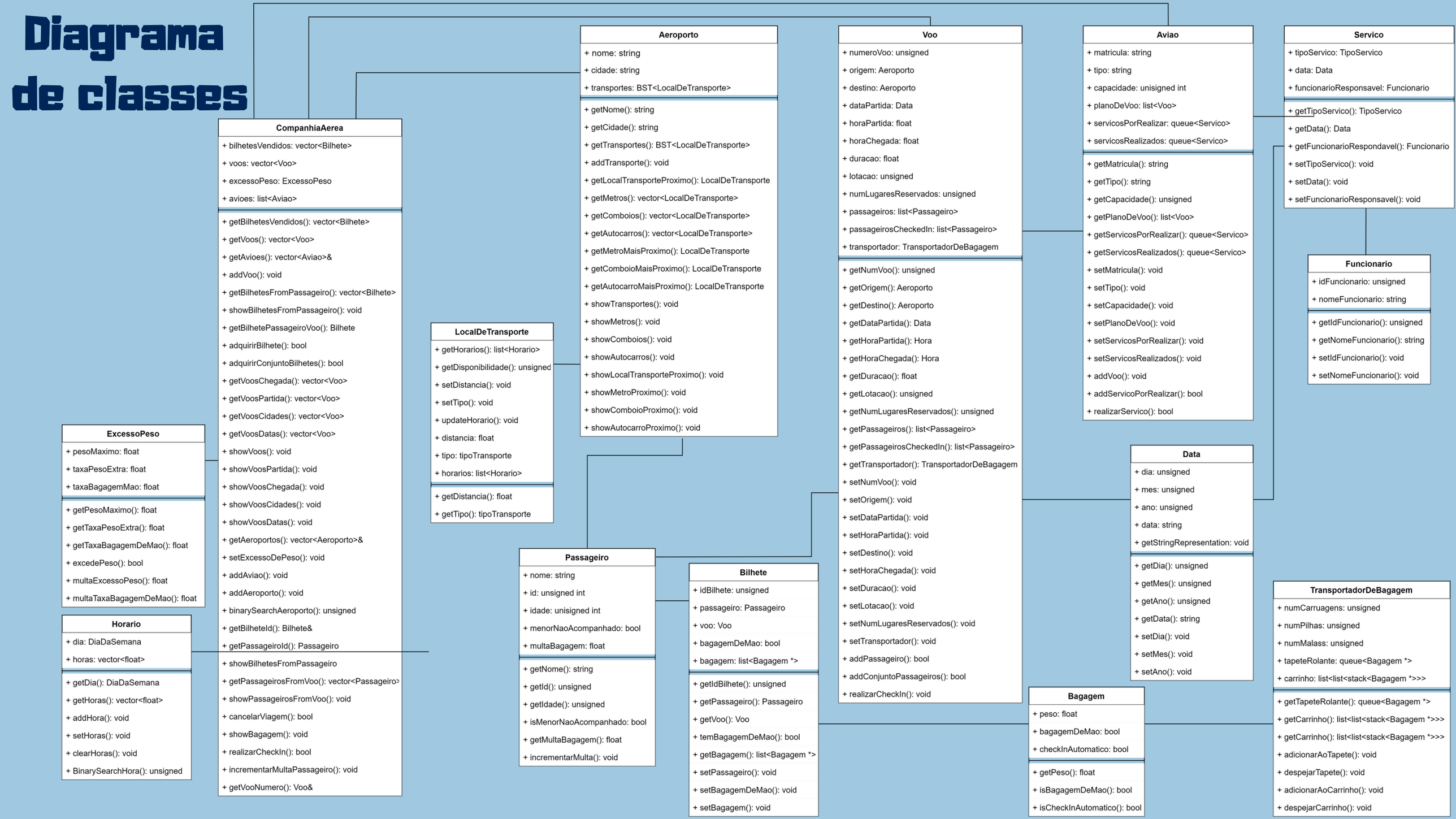
Descrição da solução

Para resolver o problema proposto procedemos à criação de uma classe principal, **CompanhiaAerea**, que tem como atributos listas e vetores de objetos das restantes classes que considerámos fundamentais:

- **Aviao** - armazena informação sobre avião, voos que vai realizar e serviços pelos quais já passou ou de que necessita
- **Aeroporto** - guarda informações sobre o aeroporto e os locais próximos para transporte público de passageiros
- **Voo** - inclui as informações do voo, dos passageiros que adquiriram bilhete para o mesmo e do transporte das suas bagagens.
- **Bilhete** - guarda informações sobre o bilhete, o passageiro que o adquiriu, a sua bagagem e o voo correspondente

Para além destas, criamos classes secundárias que, apesar de não serem utilizadas diretamente pela **CompanhiaAerea**, são importantes na modelação das classes listadas acima (**LocalDeTransporte**, **Passageiro**, **Bagagem**, **Servico**, **Horario**, **Funcionario**, **TransportadorDeBagagem**, **Data** e **ExcessoDePeso**).

Diagrama de classes



Estrutura de ficheiros

Para armazenar a informação relativa as classes Aeroporto, Aviao, Bilhete, Bagagem, ExcessoDePeso, LocalDeTransporte, Passageiro, Servico, TransportadorDeBagagem e Voo foram utilizados ficheiros de texto.

Para ler a informação, optamos por criar uma nova classe com apenas essa funcionalidade. A classe **LoadData** tem implementados métodos "load" que criam uma variável do tipo ifstream, onde será feita a abertura do ficheiro e lidas as linhas de texto do mesmo até atingir o seu fim.
A primeira linha de cada ficheiro contém o número de objetos que decidimos criar da classe correspondente, enquanto que cada uma das restantes correspondem a um atributo que permite construir um objeto da classe. As linhas com "***" servem de separação para cada objeto e são ignoradas).

ex: ficheiro aeroportos.txt, correspondente à classe Aeroporto

```
5-> número de aeroportos criados
***
Aeroporto Francisco Sa Carneiro-> nome
Porto-> cidade
***
Aeroporto Humberto Delgado
Lisboa
***
Aeroporto Joao Paulo II
S. Miguel
***
Aeroporto Internacional de Maputo Mavalane
Maputo
***
Aeroporto Internacional John F. Kennedy
Nova Iorque
```

Há algumas exceções em que os ficheiros têm uma estrutura ligeiramente diferente - ex: em locaisDeTransporte.txt decidimos incluir antes de cada linha referente ao conjunto de horários para ler uma linha adicional que definia o número de horas que deviam ser lidas.

Lista de funcionalidades implementadas

- **CRUD - OK**

  - Create: em todas as classes temos construtores que criam objetos através da leitura dos ficheiros

  - Read: nas classes que consideramos necessárias, temos vários métodos que permitem o display da informação sobre os objetos destas classes. Para muitas delas consideramos útil implementar o overload do operador << (ex: CompanhiaAerea::showVoos, CompanhiaAerea::showBagagem, CompanhiaAerea::showPassageirosFromVoo)

  - Update: a maioria das classes tem métodos que permitem a atualização dos valores dos atributos das instâncias (ex: CompanhiaAerea::addVoo, CompanhiaAerea::addAviao, CompanhiaAerea::realizarCheckIn)

  - Delete: existem métodos que removem objetos das estrutras de dados onde estão a ser guardados (ex: CompanhiaAerea::cancelarViagem, Aviao::realizarServico)

Lista de funcionalidades implementadas

- **Listagem - OK**

É possível ao utilizador fazer a listagem dos voos agendados pela companhia aérea e visualizar os passageiros com bilhete para um determinado voo.
Também é dada ao utilizador a possibilidade de ver as viagens para o qual tem bilhete comprado e para cada bilhete as suas bagagens.

- **Pesquisa - OK**

O utilizador consegue pesquisar os aeroportos, os voos com partida e/ou chegada numa determinada cidade, voos que se vão realizar em determinadas datas e locais de transporte próximos a um determinado aeroporto.

- **Funcionalidades extra**

Decidimos implementar algumas funcionalidades extra tais como a realização de check-in de um bilhete e o cancelamento de uma viagem, desde que o check-in para esta ainda não tivesse sido efetuado. Além disto, implementamos também a atribuição de multas durante o check-in para passageiros com excesso de bagagem ou com bagagem de mão sem os seus bilhetes o permitirem.

Destaque de funcionalidade

Das várias funcionalidades implementadas neste trabalho, consideramos que a melhor é a possibilidade de visualizar os vários locais de transporte nas proximidades de um aeroporto segundo diferentes critérios de ordenação.

```
11. Visualizar dados de todos os locais de transporte
12. Visualizar dados de todos os locais de Metro
13. Visualizar dados de todos os locais de Comboio
14. Visualizar dados de todos os locais de Autocarro
15. Visualizar dados do local de transporte mais proximo
16. Visualizar dados do local de transporte de Metro mais proximo
17. Visualizar dados do local de transporte de Comboio mais proximo
18. Visualizar dados do local de transporte de Autocarro mais proximo
0. Voltar a pagina anterior.

ESCOLHA UMA OPCAO:11

1. Distancia Ascendente - Disponibilidade Descendente - Tipo
2. Disponibilidade Descendente - Distancia Ascendente - Tipo
3. Tipo - Distancia Ascendente - Disponibilidade Descendente
```

Na figura ao lado é possível observar as várias formas de pesquisa para os locais de transporte na vizinhança de um determinado aeroporto e as 3 formas formas de listagem que a nossa implementação permite.

O utilizador pode, assim, escolher se pretende visualizar os locais de transporte disponíveis de acordo com um critério de distância ascendente (primeiro os que se encontram mais próximos), disponibilidade descendente (primeiro os locais que possuem mais horários disponíveis) ou por tipo (primeiro metro, depois comboio e por fim autocarro).

Algoritmos relevantes

- **Algoritmos de ordenação**: maioritariamente em vetores

Com o objetivo de manter algumas das nossas estruturas de dados ordenadas (ex: bilhetesVendidos e voos, ambos atributos da classe CompanhiaAerea), usamos o algoritmo **sort** definido na biblioteca algorithm da STL. Para este fim, tivemos de efetuar o overload do operador < para as classes cujos objetos pretendíamos ordenar. Além disso, temos um exemplo de uma lista ordenada (horarios, atributo da classe LocalDeTransporte), para a qual tivemos de usar o método **sort** definido na biblioteca list da STL.

De forma a permitir que os locais de transporte fossem apresentados ao utilizador segundo diferentes critérios por ele escolhidos, tivemos de criar funções de ordenação específicas para cada uma das opções, exceto a que já estava implementada através do operador <.

- **Algoritmos de pesquisa:**

- Binary Search: De modo a manter a ordem de algumas das nossas estruturas de dados (ex: aeroportos, atributo da classe CompanhiaAerea), decidimos usar este algoritmo com o objetivo de preservar a ordem correta ao inserir um novo elemento e evitar que elementos repetidos fossem inseridos.

- Linear Search: Também usamos este algoritmo para pesquisa em estruturas não ordenadas e em estruturas ordenadas mas cujo critério de ordenação não coincidia com aquele que pretendíamos usar na pesquisa.

Exemplos de execução

- Visualização dos bilhetes adquiridos e bagagem

```
BEM-VINDO A NOSSA COMPANHIA AEREA
1. Pesquisar Voos
2. Realizar Check-in
3. As minhas Reservas
4. Transportes: Locais e Horarios
5. Companhia: Pesquisa e Listagens
0. Sair.

ESCOLHA UMA OPCAO:3

Insira o seu numero de identificacao:2
 NUMERO DE BILHETE: 3

Voo Numero: 11121
Origem: Aeroporto Francisco Sa Carneiro- Porto
Destino: Aeroporto Joao Paulo II- S. Miguel
Data: 30-1-2022 Partida: 12.55h Chegada: 14.20h Duracao: 2.25 horas
Joana Falcao nao tem direito a levar bagagem de mao
Pode levar 2 mala(s) no total.
```

```
1. Ver bagagem
2. Ver multa
3. Cancelar uma viagem
0. voltar

ESCOLHA UMA OPCAO:1
 Insira o numero do bilhete:3
 Joana Falcao leva 2 mala(s) no total.
1.
Bagagem de porao de peso 17.60
2.
Bagagem de mao de peso 11.40

1. Pesquisar Voos
2. Realizar Check-in
3. As minhas Reservas
4. Transportes: Locais e Horarios
5. Companhia: Pesquisa e Listagens
0. Sair.

ESCOLHA UMA OPCAO:
```

Exemplos de execução

- Pesquisa de um local de transporte

```
BEM-VINDO A NOSSA COMPANHIA AEREA
1. Pesquisar Voos
2. Realizar Check-in
3. As minhas Reservas
4. Transportes: Locais e Horarios
5. Companhia: Pesquisa e Listagens
0. Sair.

ESCOLHA UMA OPCAO:4
1- Aeroporto Francisco Sa Carneiro
2- Aeroporto Humberto Delgado
3- Aeroporto Internacional John F. Kennedy
4- Aeroporto Internacional de Maputo Mavalane
5- Aeroporto Joao Paulo II

Escolha uma das opcoes relativas ao nome do aeroporto (0 para voltar a pagina anterior):1

11. Visualizar dados de todos os locais de transporte
12. Visualizar dados de todos os locais de Metro
13. Visualizar dados de todos os locais de Comboio
14. Visualizar dados de todos os locais de Autocarro
15. Visualizar dados do local de transporte mais proximo
16. Visualizar dados do local de transporte de Metro mais proximo
17. Visualizar dados do local de transporte de Comboio mais proximo
18. Visualizar dados do local de transporte de Autocarro mais proximo
0. Voltar a pagina anterior.

ESCOLHA UMA OPCAO:12
```

Exemplos de execução

- Pesquisa de um local de transporte

```
1. Distancia Ascendente - Disponibilidade Ascendente - Tipo
2. Disponibilidade Ascendente - Distancia Ascendente - Tipo
3. Tipo - Distancia Ascendente - Disponibilidade Ascendente


ESCOLHA EM QUE ORDEM DESEJA QUE OS DADOS SEJAM APRESENTADOS:2
Tipo de transporte: Metro
Distancia ao aeroporto: 70 metros
Horarios:
Dias Uteis: 0.00  1.00  2.30  3.40  4.55  5.20  6.50  7.28  8.23  9.12  10.20  11.03  12.08  13.16  14.18  15.27  16.37
 17.19  18.18
Sabados: 0.30  1.30  2.30  7.30  8.30  9.30  10.30  11.30  12.30  16.30  17.30  18.30  19.30  20.03  21.30  22.30  23.30


Tipo de transporte: Metro
Distancia ao aeroporto: 60.00 metros
Horarios:
Dias Uteis: 0.30  1.30  2.30  3.30  4.30  5.30  6.30  7.30  8.30  9.30  10.30  11.30  12.30  13.30  14.30  15.30  16.30
 17.30  18.30  19.30  20.03  21.30  22.30  23.30
```

Exemplos de execução

- Visualização de voos e compra de um bilhete

```
BEM-VINDO A NOSSA COMPANHIA AEREA
1. Pesquisar Voos
2. Realizar Check-in
3. As minhas Reservas
4. Transportes: Locais e Horarios
5. Companhia: Pesquisa e Listagens
0. Sair.

ESCOLHA UMA OPCAO:1

6. Imprimir todos os voos
7. Visualizar voos com partida numa determinada cidade
8. Visualizar voos com chegada numa determinada cidade
9. Visualizar voos entre duas cidades selecionadas
10. Visualizar voos realizados em determinadas datas
0. Voltar a pagina anterior
```

```
ESCOLHA UMA OPCAO:7
 1- Porto
2- Lisboa
3- Nova Iorque
4- Maputo
5- S. Miguel

Insira a cidade de partida:1

Deseja selecionar alguma data? [Escolha 'S'-Sim ou 'N'- Nao ou 0-Voltar]
N
---------------
Voo Numero: 11121
Origem: Aeroporto Francisco Sa Carneiro- Porto
Destino: Aeroporto Joao Paulo II- S. Miguel
Data: 30-1-2022 Partida: 12.55h Chegada: 14.20h Duracao: 2.25 horas
---------------
Voo Numero: 11122
Origem: Aeroporto Francisco Sa Carneiro- Porto
Destino: Aeroporto Joao Paulo II- S. Miguel
Data: 30-1-2022 Partida: 20.55h Chegada: 22.20h Duracao: 2.25 horas
```

Exemplos de execução

- Visualização de voos e compra de um bilhete

```
1. Comprar um bilhete
0. Voltar
ESCOLHA UMA OPCAO:1

INSIRA O NUMERO DO VOO PARA O QUAL DESEJA COMPRAR BILHETE:11121
 INSIRA O NUMERO DE PESSOAS: (0 para voltar):1
 Deseja levar bag
agem de mao?
1. Sim
2. Nao
1

Nome:Sofia
 No. de identificacao:1
 Idade:20
 Quantas bagagens de porao deseja levar?
1
Peso da bagagem de mao:10
 Peso da bagagem de porao:25
 COMPRA REALIZADA COM SUCESSO!
```

Esforço e principais dificuldades

Dificuldades

- Planeamento da informação que devíamos guardar em cada classe e as relações entre cada uma delas
- Decidir quais getters deveriam retornar os objetos pretendidos como referência ou uma cópia do mesmo.
- Não foi possível implementar a atualização de dados nos ficheiros mas a mesma pode ser verificada através do menu, recorrendo às opções de visualização.

Esforço de cada elemento

- Numa fase inicial dividimos as várias classes que tínhamos delineado entre todos os elementos do grupo. Posteriormente, à medida que foi surgindo a necessidade de implementarmos novos métodos auxiliares e relações entre as várias classes começamos todas a contribuir para todas as classes.
- Numa fase final em que começamos a testar o nosso código mais extensivamente, todos os elementos do grupo mostraram-se disponíveis para ajudar na resolução dos diversos erros que foram sendo detetados por cada um dos elementos.