U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

COMPANHIA AÉREA

PROJETO 1 - AED - 2º ANO - L.EIC

Grupo G6:
Angy da Cruz (202007253)
Daniela Aguiar (201910371)
Tomás Martins (201704976)

PROBLEMA

Uma nova companhia aérea acabou de entrar no mercado e requer a implementação de um sistema de gestão de informação inovador e personalizado, que responda às suas necessidades. A função primordial do sistema será **guardar** e **gerir informação** relativa a aviões, voos, passageiros e bagagens.

SOLUÇÃO E ALGORITMOS

A nossa solução fundamenta-se nas instruções fornecidas inicialmente, as quais fomos modificando para obter uma melhor estrutura do programa.

O projeto está à volta das diferentes classes e dos seus métodos que tratam diferentes dados e situações, além disto, temos outras estruturas para apoiar a apresentação da escrita e de leitura dos ficheiros.

O Aeroporto é a nossa *classe mãe* que agrupa os diferentes aviões em vetores.

Cada Avião tem a informação de ele mesmo como a matrícula e a sua capacidade, também guardar numa lista os voos que realiza e num queue o serviço designado para cada um dos aviões. Os Voos são caracterizados por o número de voo, data de partida, duração e a capacidade que tem, e vai guardar num vetor os passageiros que compraram bilhete para o voo.

A classe Serviços tem os dados sobre o tipo de serviço que pode ser de manutenção ou de limpeza, a data em que é realizada o serviço e o funcionário responsável.

Todo passageiro tem o número de bilhetes que compraram (pode ser um ou mais bilhetes) e se o passageiro pretende levar consigo bagagem ou não. Na classe Bagagem é indicado se o check-in foi automático ou não.

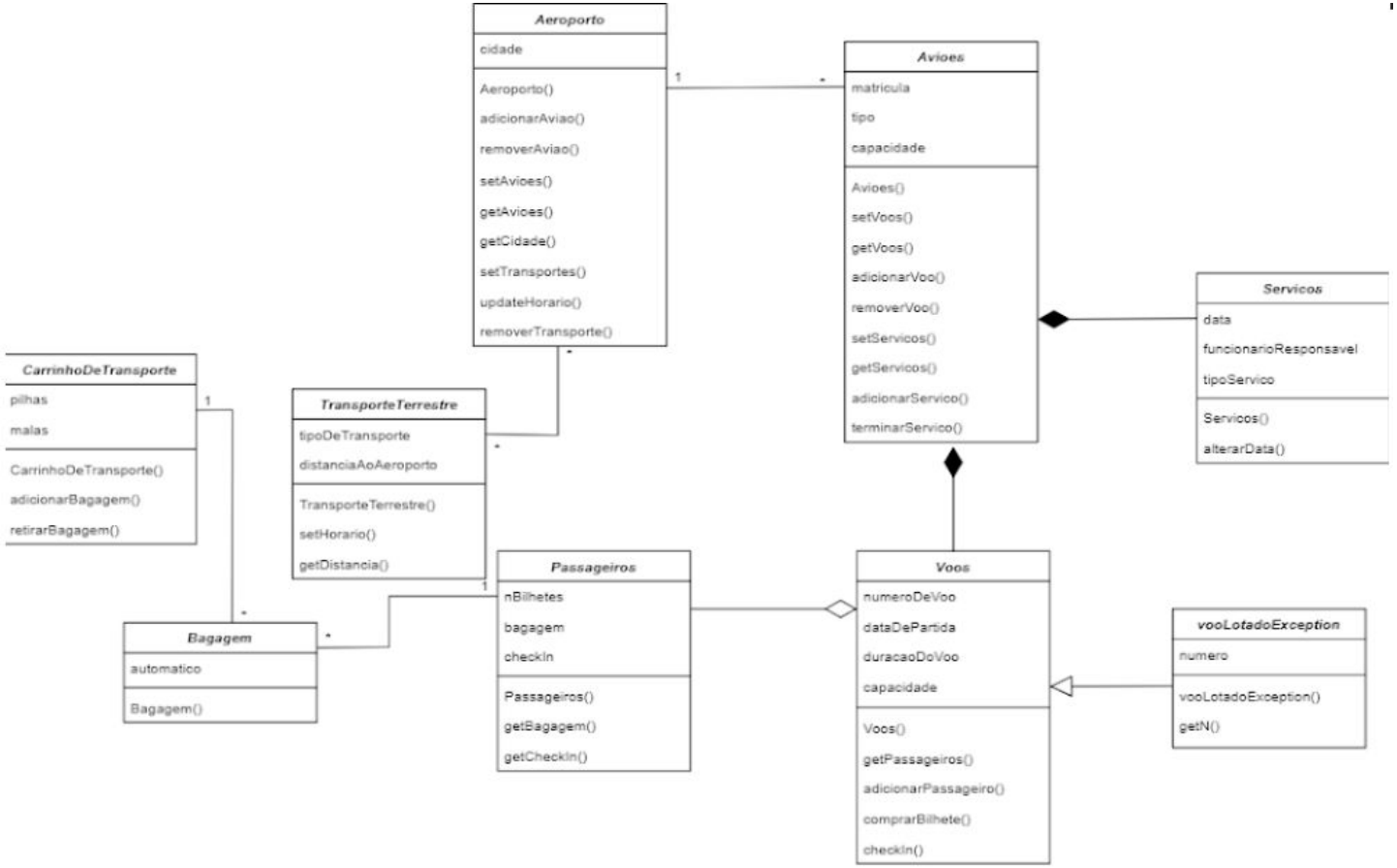
Para transportar a bagagem temos o Carrinho de Transporte que guarda a quantidade de pilhas onde é são colocadas uma certa quantidade de malas, esta informação é guardada en stacks que a sua vez contêm informação em forma de lista e queue.

A companhia aérea fornece aos passageiros informação dos Transportes Terrestres que é guardada numa árvore de pesquisa binária com o tipo de transporte, a distância ao aeroporto e uma lista com os horários.

Sendo que o programa permite operações como pesquisa e adição, edição e remoção de elementos dos vetores, foram desenvolvidas exceções para poder lidar com os casos onde a informação inserida por o utilizador é inválida e assim informar sobre o problema.

O projeto é feito através de um conjunto de métodos que vai facilitar ao utilizador ter uma interação mais eficiente e fluida com o programa, sendo apresentado ao utilizador as instruções necessárias para uma correta utilização do mesmo.

DIAGRAMA DE CLASSES



ESTRUTURA DE FICHEIROS

+.idea
->.gitignore
->.name
->CompanhiaAeria.iml
->misc.xml
->vcs.xml
+cmake-build-debug
+CMakeFiles
+3.20.2
+CompilerIdC
->CMakeCCompilerId.c
->a.exe
+CompilerIdCXX
->CMakeCXXCompilerId.cpp
->a.exe
->CMakeCCompiler.cmake
->CMakeCXXCompiler.cmake
->CMakeDetermineCompilerABI_C.bin
->CMakeDetermineCompilerABI_CXX.bin
->CMakeRCCompiler.cmake
->CMakeSystem.cmake
+CompanhiaAeria.dir
+codigo
->Aeroporto.cpp.obj
->Avioes.cpp.obj
->Bagagem.cpp.obj
->CarrinhoDeTransporte.cpp.obj
->Passageiros.cpp.obj
->Servicos.cpp.obj
->TransporteTerrestre.cpp.obj
->Voos.cpp.obj
->main.cpp.obj
->CXX.includecache
->DependInfo.cmake
->build.cmake
->cmake_dean.cmake
->depend.internal
->depend.make
->flags.make
->link.txt
->linklibs.rsp
->objects.a
->objects1.rsp
->progress.make
+Testing/Temporary
->LastTest.log
->CMakeCache.txt
->CMakeDoxyfile.in
->CMakeDoxygenDefaults.cmake
->CompanhiaAeria.cbp
->CompanhiaAeria.exe
->Doxifile
->Makefile
->cmake_install.cmake
+codigo
->Aeroporto.cpp
->Aeroporto.h
->Avioes.cpp
->Avioes.h
->Bagagem.cpp
->Bagagem.h
->CarrinhoDeTransporte.cpp
->CarrinhoDeTransporte.h
->Passageiros.cpp
->Passageiros.h
->Servicos.cpp
->Servicos.h
->TransporteTerrestre.cpp
->TransporteTerrestre.h
->Voos.cpp
->Voos.h
->bst.h
->ficheiro.txt
->main.cpp
+documentação
+html
->_aeroporto_8cpp.html
->_aeroporto_8h.html
->_aeroporto_8h_source.html
->_avioes_8cpp.html
->_avioes_8h.html
->_avioes_8h_source.html
->_bagagem_8cpp.html
->_bagagem_8h.html
->_bagagem_8h_source.html
->_carrinho_de_transporte_8cpp.html
->_carrinho_de_transporte_8h.html
->_carrinho_de_transporte_8h_source.html
->_passageiros_8cpp.html
->_passageiros_8h.html
->_passageiros_8h_source.html
->_servicos_8cpp.html
->_servicos_8h.html
->_servicos_8h_source.html
->_transporte_terrestre_8cpp.html
->_ transporte_terrestre_8h.html
->_ transporte_terrestre_8h_source.html
->_voos_8cpp.html
->_voos_8h.html
->_voos_8h_source.html
->annotated.html
->bc_s.png
->bdwn.png
->class_aeroporto-members.html
->class_aeroporto.html
->class_avioes-members.html
->class_avioes.html
->class_bagagem-members.html
->class_bagagem.html
->class_carrinho_de_transporte-members.html
->class_carrinho_de_transporte.html
->class_passageiros-members.html
->class_passageiros.html
->class_servicos-members.html
->class_servicos.html
->class_transporte_terrestre-members.html
->class_transporte_terrestre.html
->class_voos-members.html
->class_voos.html
->classes.html
->classvoo_lotado_exception-members.html
->classvoo_lotado_exception.html
->closed.png
->dir_4d7be947362df13686123d6563352e9b.html
->doc.png
->doxygen.css
->doxygen.svg
->dynsections.js
->files.html
->folderclosed.png
->folderopen.png
->functions.html
->functions_func.html
->globals.html
->globals_func.html
->image.jpg
->index.html
->jquery.js
->main_8cpp.html
->menu.js
->menudata.js
->nav_f.png
->nav_g.png
->nav_h.png
->open.png
->splitbar.png
->sync_off.png
->sync_on.png
->tab_a.png
->tab_b.png
->tab_h.png
->tab_s.png
->tabs.css
->Doxyfile
->image.jpg
->CMakeLists.txt

FUNCIONALIDADES

Com o nosso código conseguimos:

1. Adicionar e remover num vetor os aviões que chegam e saem do aeroporto.
2. Adicionar e retornar todos os aviões que existem no aeroporto naquele momento.
3. Retornar a cidade onde o aeroporto se situa
4. Retornar e agrupar todos os transportes terrestre que estão na proximidade do aeroporto.
5. Remover transportes da lista.
6. Verificar se dois aviões são os mesmos.
7. Adicionar e retornar numa lista todos os voos que vai realizar um avião (plano De Voo).
8. Adicionar e remover voos no plano de voos.

9. Retornar uma lista com os serviços que são designado para cada avião.
10. Adicionar um serviço à lista dos serviços.
11. Retirar da fila dos serviços e por na lista de serviços completados.
12. Retornar se o check-in da bagagem de um passageiro foi automático ou não.
13. Adicionar a bagagem de um passageiro ao carrinho de transporte.
14. Retirar da primeira pilha do carrinho de transporte a última mala.
15. Retorna se o passageiro leva bagagem ou não.
16. Alterar as datas dos serviços.
17. Listar os horários dos transporte.
18. Modificar a lista dos horários dos transportes.
19. Retornar a distância que há entre a parada do transporte e o aeroporto.

20. Retornar o tipo de transporte terrestre.

21. Verificar se dois voos são iguais

22. Retornar num vector todos os passageiros que compraram bilhetes para o voo.

23. Adicionar passageiros a um voo.

24. Indicar o n.º de bilhetes que o passageiro quer comprar e se deseja levar bagagem ou não.

FUNCIONALIDADE (DESTAQUE)

A funcionalidade que nós podemos destacar neste projeto está na classe Aeroporto e é a funcionalidade de **Adicionar e Remover aviões**:

Para todo avião que está aterrar no aeroporto vai ter que ser adicionado ao fim do vetor dos aviões do aeroporto.

Quando um avião está a descolar do aeroporto vai ser removido do vetor, o programa o que faz é ler todo o vetor do início até o fim em procura do avião que queremos remover, se encontrar removemos e se não encontramos é porque já foi removido.

DIFICULDADES E ESFORÇO INDIVIDUAL

A nossa maior dificuldade foi a má gestão do nosso próprio tempo, sendo que talvez isso nos tenha inhibido de implementar determinadas soluções que gostaríamos. Além disso, tivemos algumas dificuldades na planificação do projeto, visto que às vezes havia indecisão sobre qual era a melhor forma de declarar atributos, classes e métodos.

As decisões sobre o projeto foram tomadas em grupo, sempre estivemos em comunicação, fizemos reuniões em discord para fazer “live coding” onde todos aportavam ideias e foi feita a maior parte do trabalho. A Daniela fez a base do programa,com as classes principais e os métodos bases, também realizou a documentação do Doxygen. O Tomás adicionou códigos e métodos ao projeto, e alguns dos ficheiros. A Angy adicionou códigos e fez a apresentação com o diagrama de classes e a estrutura de ficheiros. Neste projeto todos estiveram disposto para ajudar em qualquer coisa que fosse precisa.