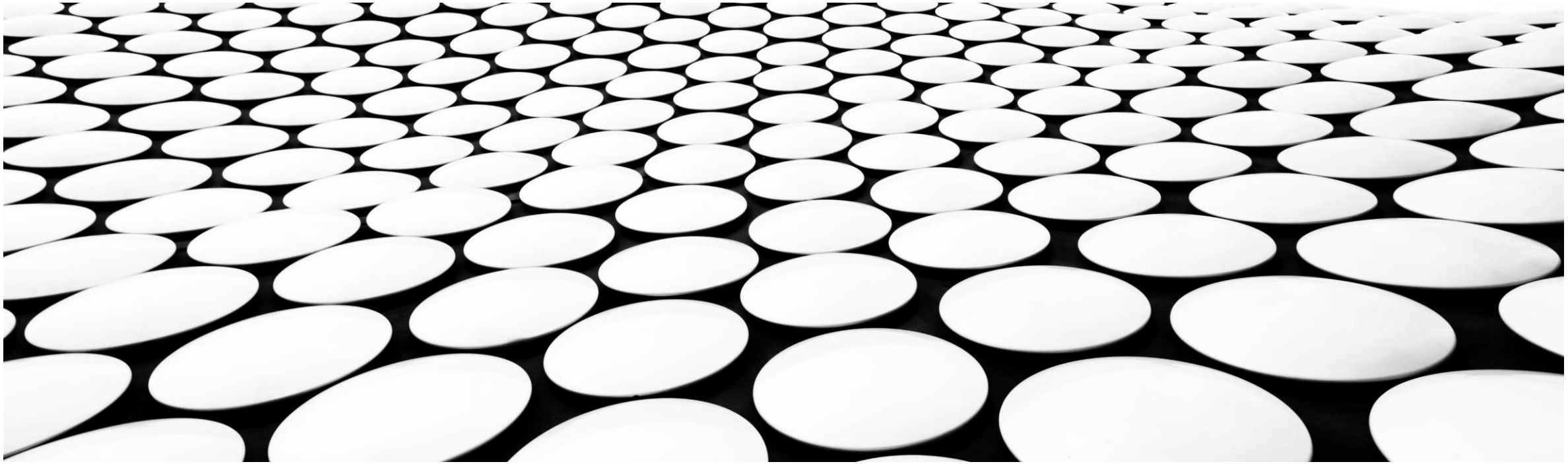
GESTÃO DE INFORMAÇÃO NUMA COMPANHIA AÉREA

TRABALHO 1 DE ALGORITMOS E ESTRUTURA DE DADOS

Autores (G14)
Fábio Cunha Morais (up202008052)
Francisco Miguel Alcobia Maia Prada (up202004646)

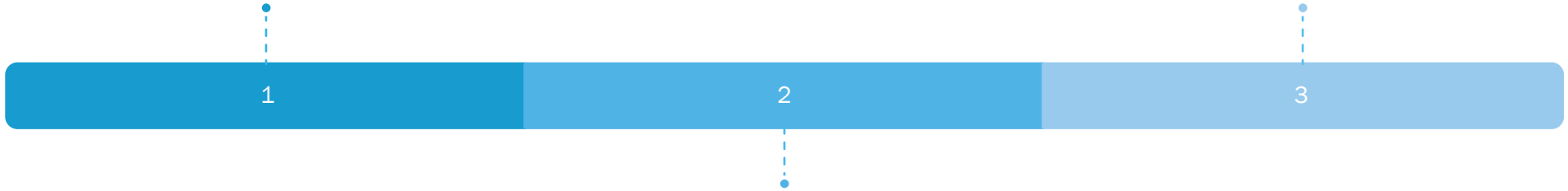


U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

PROBLEMA A RESOLVER

Gerir os aspetos principais que definem um aeroporto (aviões, voos, companhias aéreas, partidas e chegadas, etc)

Relacionar as estruturas de modo a obter uma base de dados funcional



Criar classes e respetivos atributos pertinentes de modo a gerir a informação sobre Aviões, Voos, Clientes, Companhias, etc.

SOLUÇÃO

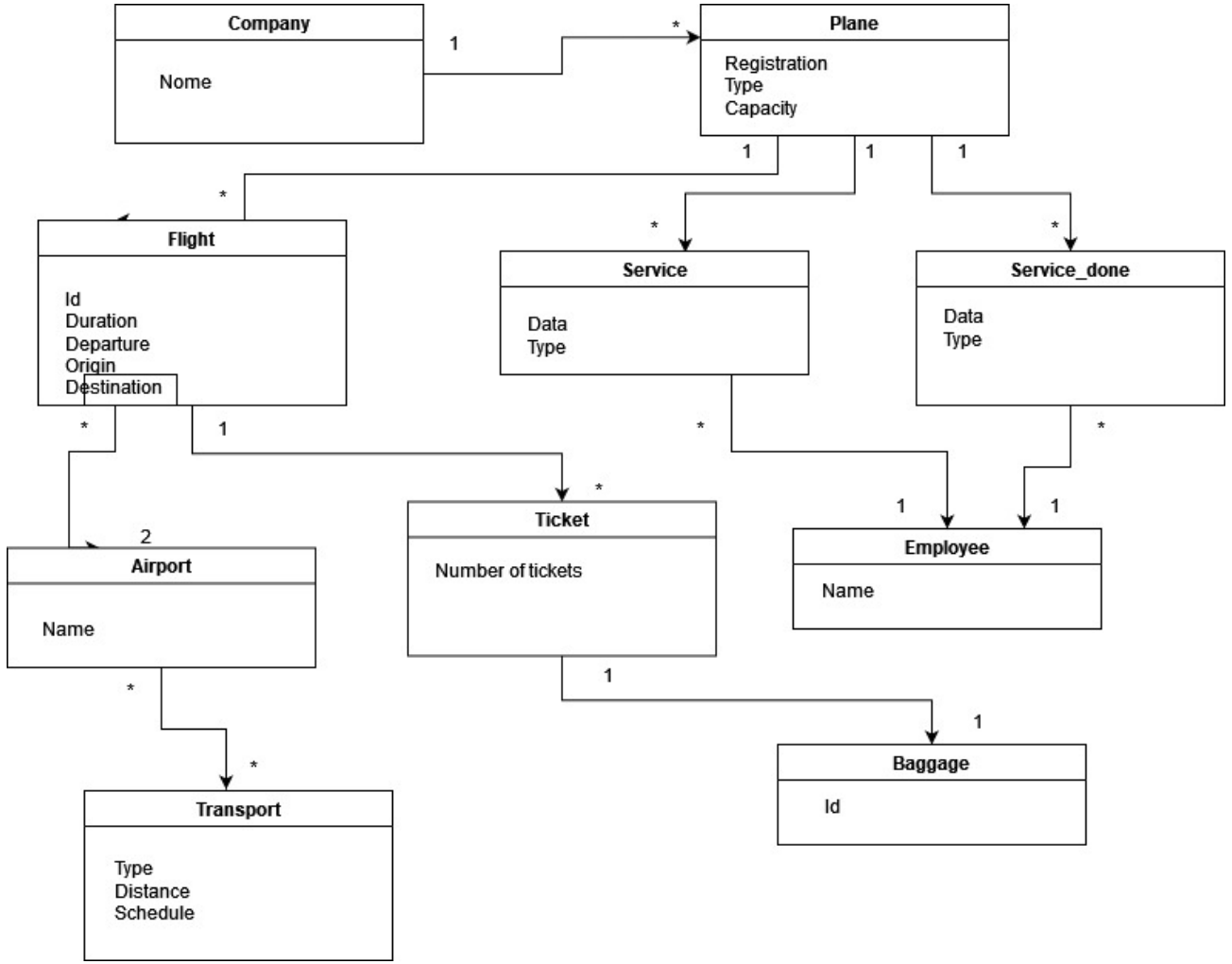
- De modo a resolver o problema dado pelo enunciado, decidimos, primeiramente, dividir o projeto entre duas interfaces: uma utilizada por qualquer companhia aérea X e outra de modo a ser utilizada para qualquer utilizador Y.
- Cada uma destas interfaces, bastante distintas uma da outra, dividem o problema e a sua solução em dois. Desta maneira, tratam ambas de problemas distintos mas que, após várias relações entre as classes usadas em cada uma delas, acabam por se complementar. Desta forma, possibilitamos uma maior e mais versátil gestão e alteração dos dados do projeto.

SOLUÇÃO ...

- **Interface da companhia:**
  - Gere os seus aviões e fornece informação acerca deles;
  - Usada para controlar voos e tráfego aéreo de cada avião;
  - Utiliza operações CRUD;
  - Assenta sobre a manipulação de ficheiros texto que são lidos no início possibilitando, assim, todos os métodos existentes e que deve ser “updated”, de modo a guardar alterações.

- **Interface do cliente:**
  - Compra bilhetes;
  - Gere a disponibilidade de bilhetes;
  - Gere a bagagem;
  - Informa acerca dos transportes perto do aeroporto;
  - Utiliza operações CRUD.

UML


Company
Nome
1
*
Plane
Registration
Type
Capacity
1
1
1
*
*
*
Flight
Id
Duration
Departure
Origin
Destination
Service
Data
Type
Service_done
Data
Type
*
1
*
*
*
*
2
Airport
Name
Ticket
Number of tickets
1
1
Employee
Name
*
1
*
1
Transport
Type
Distance
Schedule
Baggage
Id

ESTRUTURA DE FICHEIROS

Classes	baggage.h	cart.h	company.h	employees.h	flights.h	menu.h	planes.h	services.h	tickets.h	transport.h	treadmill.h
baggage.cpp	inclui										
cart.cpp	inclui	inclui									
company.cpp			inclui		inclui		inclui				
employees.cpp				inclui							
flights.cpp			inclui		inclui		inclui		inclui		
menu.cpp			inclui		inclui	inclui			inclui	inclui	
planes.cpp			inclui		inclui		inclui	inclui			
services.cpp				inclui				inclui			
tickets.cpp					inclui		inclui		inclui		
transport.cpp										inclui	
treadmill.cpp	inclui										inclui
main.cpp			inclui			inclui			inclui	inclui	

FUNCIONALIDADES IMPLEMENTADAS

- <u>Foram implementadas as seguintes funcionalidades para os aviões:</u>
  - Create através de um ficheiro de texto- **Completa**
  - Mostrar os aviões ordenados na consola- **Completa**
  - Adicionar aviões a um vetor mantendo-o sorted- **Completa**
  - Remover aviões do vetor- **Completa**
  - Atualizar ficheiro de texto- **Completa**
- <u>Foram implementadas as seguintes funcionalidades para os voos:</u>
  - Create através de um ficheiro de texto- **Completa**
  - Mostrar os voos de todos os aviões na consola- **Completa**
  - Adicionar o voo com as características fornecidas pela companhia a um avião escolhido por esta- **Parcial**
  - Remover voo com o id fornecido (percorre todos os voos de todos os aviões)- **Parcial**
  - Atualizar ficheiro de texto- **Parcial**

FUNCIONALIDADES IMPLEMENTADAS

- <u>Foram implementadas as seguintes funcionalidades para os serviços:</u>
  - Create através de um ficheiro de texto- **Parcial**
  - Efetuar o primeiro serviço da fila do avião escolhido- **Parcial**
  - Mostrar último serviço realizado- **Completa**
  - Adiciona serviço no fim da fila do avião escolhido- **Parcial**
  - Atualizar ficheiro de texto- **Parcial**
- <u>Foram implementadas as seguintes funcionalidades para a bagagem:</u>
  - Adicionar bagagem ao tapete rolante (fila)- **Parcial**
  - Remover bagagem do tapete rolante (deve ser seguida de adicionar ao carrinho)- **Parcial**
  - Adicionar bagagem ao primeiro sítio disponível do carrinho- **Parcial**
  - Remover bagagem pedida se ela estiver no topo duma das stacks- **Parcial**

FUNCIONALIDADES IMPLEMENTADAS

- <u>Foram implementadas as seguintes funcionalidades para os transportes:</u>
  - Create através de um ficheiro de texto- **Completa**
  - Ler o ficheiro, consoante os parâmetros requeridos- **Completa**
  - Implementar operadores de modo a pesquisar por elementos na BST- **Completa**
  - Implementar operadores de modo a inserir elementos na BST- **Parcial**
  - Criação de uma BST para cada aeroporto- **Parcial**
  - Mostrar os transportes ordenados na consola- **Completa**

DESTAQUE DE FUNCIONALIDADE

Existe uma opção “update” que permite atualizar o ficheiro de texto, tendo em conta a informação que está no programa, sendo que esta tem de ser acionada pela companhia e não ocorre automaticamente. Isto leva a uma maior segurança se, por exemplo, por lapso, se eliminar um avião. Mesmo assim, o ficheiro continuará guardado sem quaisquer alterações que possam prejudicar a gestão de cada companhia.

PRINCIPAIS DIFICULDADES ENCONTRADAS E PRESTAÇÃO

- A maior dificuldade foi, provavelmente, gerir a ligação entre as diferentes classes. Optamos por um sistema de ligação circular entre classes como companhia, aviões, voos, etc. Porém, não foi possível estabelecer certas ligações, tais como: a ligação entre a bagagem e o voo ou a ligação entre o transporte e o aeroporto.
- Outra dificuldade encontrada foi a de gerir o "Path" dos diferentes ficheiros de texto, sendo necessário adaptar o código ao computador em que este é corrido.
- Para além disto, existe um bug na interface o qual não foi possível resolver. Se o user escolher a opção de sair de um dos "sub-menus" e decidir escolher outro no main menu, o programa escreve a bst do nosso projeto no terminal e pergunta ao user se quer sair.
- Finalmente, foi nos impossível fazer documentação no doxygen. Devido a este facto, resolvemos optar por uma documentação mais básica no código em si.
- Quanto à prestação de cada elemento no grupo, devido ao tempo disponível para a realização e com a condição de sermos apenas uma dupla, vimo-nos decididos a aplicar um enorme esforço no desenvolvimento do projeto. Por vezes, um criava ficheiros de raíz e as suas bases e o outro complementava-as, e outras, acontecia o chamado vice-versa.