Faculdade de Engenharia da Universidade do Porto

GESTÃO DE INFORMAÇÃO EM UMA COMPANHIA AÉREA

AED – Grupo 92

Elementos do Grupo:

Diogo Tomáz Câmara 2019051662@fe.up.pt
Mário Pedro Soares Pinto Mota Ferreira up201907727@fe.up.pt

Dezembro de 2021

Índice

Descrição do problema

Uma companhia aérea pretende implementar um sistema de gestão de informação. O sistema deve guardar e gerir informação relativa a aviões, voos, passageiros e bagagens. Um avião possui ainda um plano de voo, isto é, a lista de voos que realiza. Os aviões estão sujeitos a serviços de manutenção e limpeza, cuja data de realização é determinada antecipadamente. Deve também guardar informação sobre os serviços já realizados.

Um passageiro pode adquirir um bilhete para determinado voo, desde que existam lugares disponíveis. Na aquisição do bilhete deve ser considerada a inclusão de bagagem ou não. Considere a possibilidade de aquisição de um conjunto de bilhetes, para viagens em grupo. Quantos aos check-ins o sistema guarda informação relativa ao transporte das malas até aos aviões correspondentes, sendo as malas transportadas por tapetes rolantes e carrinhos. Para auxílio aos passageiros, a companhia aérea mantém informação sobre os locais de transporte terrestre nas proximidades de cada aeroporto.

Descrição da solução

O primeiro passo no desenvolvimento do programa foi a implementação de todas as classes necessárias para o desenvolvimento do nosso programa. Foram então implementadas as classes aviões, voos, passageiros, aeroporto, data, menu, pessoa, serviço, mala, trabalhador, tapete e local transporte . Ao mesmo tempo foram implementadas classes com a função de registo de forma a guardar informação sobre todos os objetos das classes anteriores, sendo esta informação guardada em vetores, listas , queue e stacks, dependendo qual das formas de armazenamento se adequa mais ao problema.

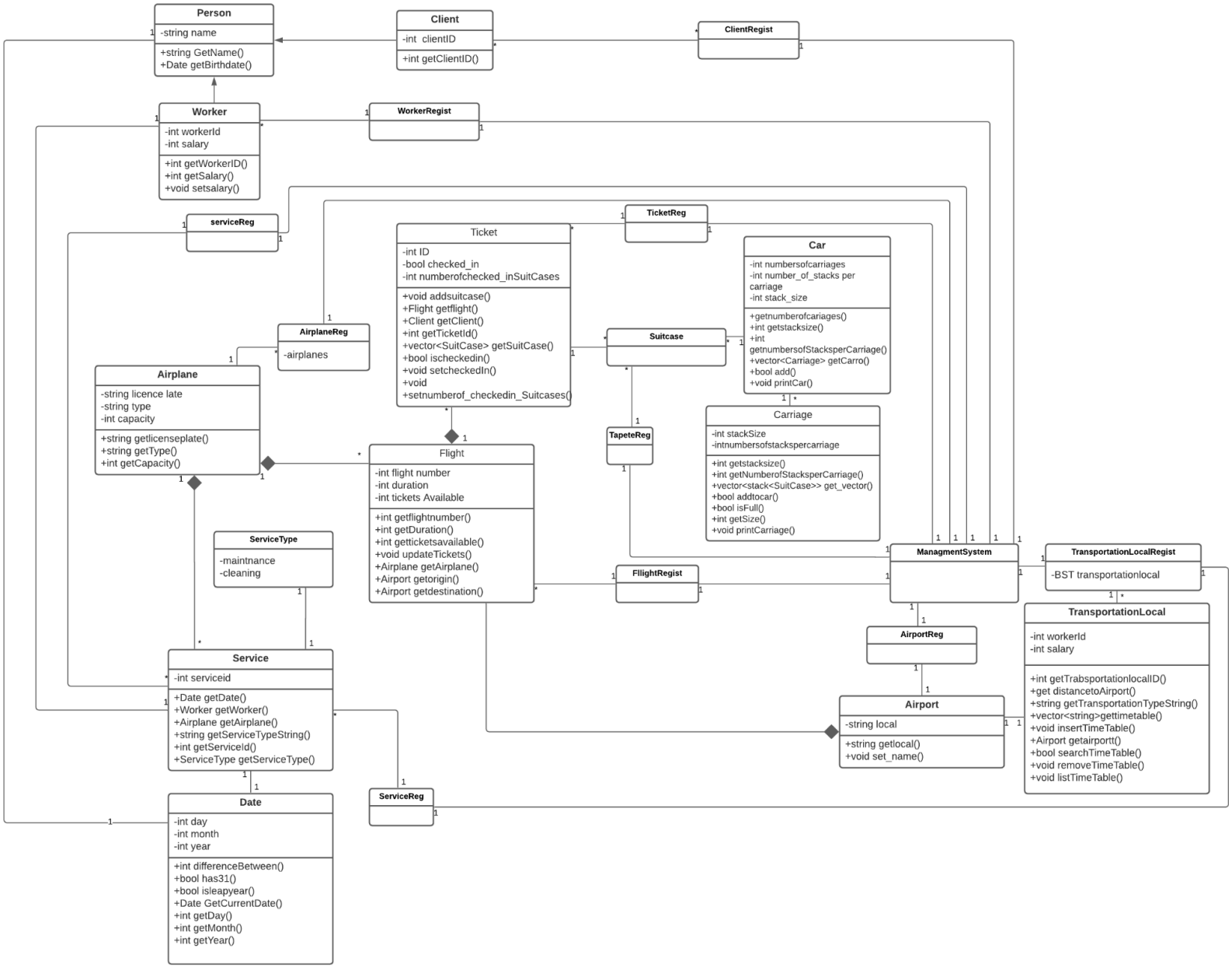
Toda esta informação é guardada em ficheiros .txt. Sempre que o programa é inicializado todos os ficheiros .txt são lidos, sendo cada linha de um ficheiro responsável pela criação de um objeto pertencentes a uma “class” do programa. Sempre que é criado um objeto , este objeto é guardado num “vector”, “lista”, “stack” ou “queue” do registo responsável por guardar informação de objetos deste tipo.

De forma a facilitar a manipulação de todos estes dados foi criada a “class” menu que se encarrega de pedir os dados ao utilizador do programa, consoante a opção selecionada.

Foram criados métodos de forma que sempre que o programa for fechado, a informação de todos os “containers” é lida e guardada em ficheiros .txt de maneira que as novas informações ou alterações na informação inicial não sejam perdidas.

Após a implementação de toda a estrutura principal do programa, foi implementado o menu e toda uma interface que o utilizador utiliza para interagir com o programa. Foram criados sub-menus de maneira que o utilizador possa interagir com todas as funcionalidades existentes.

Diagrama de classes



Estrutura de ficheiros

- **Airports.txt -** Lista de aeroportos utilizados pela companhia aérea guardando apenas a sigla do aeroporto.
- **clients.txt -** Lista de clientes registados nesta companhia guardando os seguintes dados: id, nome e data de nascimento.
- **flights.txt -** Lista de todos os voos efetuados por esta companhia aérea guardando os seguintes dados: número de voo, duração, aeroporto de origem e aeroporto de destino e número de bilhetes disponíveis.
- **plane.txt -** Lista de todos os aviões pertencentes à empresa guardando os seguintes dados: matrícula, tipo e capacidade.
- **service.txt -** Lista de todos os serviços realizados aos aviões da empresa guardando os seguintes dados: data, tipo de serviço,id do serviço, funcionário responsável e matrícula do avião.
- **tickets.txt -** Lista de todos os bilhetes vendidos pela companhia guardando os seguintes dados: id do voo, id do cliente, id do bilhete, número de malas, estado do check-in .
- **transports.txt -** Lista de todos os métodos de transporte perto dos aeroporto utilizados pela companhia guardando os seguintes dados: Aeroporto , id do transporte, distância até ao aeroporto e tipo de método de transporte. Seguidamente guarda o horário do transporte, sob a forma de, por exemplo, 0900, 1010, 1100, para respetivamente 9h00, 10h10, 11h00.
- **workers.txt -** Lista de todos os trabalhadores da companhia guardando os seguintes dados: nome , data de nascimento, id do funcionário e salário.

Funcionalidades implementadas

Foram implementadas no programa diversas funcionalidades do tipo CRUD, listagens, pesquisas e outras funcionalidades que achamos necessárias para o correto funcionamento do programa.

Create

- **Add airport -** cria um novo aeroporto e adiciona-o ao vetor registo de aeroportos.
- **Add plane -** criar um novo avião e adiciona-o ao vetor registo de aviões.
- **Add flight -** criar um novo voo e adiciona-o ao vetor de registo de voos.
- **Add client -** cria um novo cliente e adiciona-o ao vetor de registos de clientes.
- **Add worker -** cria um novo trabalhador e adiciona-o ao vetor de registos de trabalhadores.
- **Add service -** cria um novo serviço e adiciona-o ao vetor de registos de serviços.
- **Add Transportation Local -** cria um novo local de paragem de um transporte público e adiciona-a ao vetor de registos de transportes.

Read

As funcionalidades de leitura foram implementadas sempre que o programa é inicializado , lendo todos os ficheiros .txt do programa e preenchendo os “containers” com a informação lida destes mesmos ficheiros.

Delete

- **Delete airport -** remove um aeroporto do vetor registo de aeroportos.
- **Delete plane -** remove um avião do vetor registo de aviões.
- **Delete flight -** remove um voo do vetor registo de voos.
- **Delete client -** remove um cliente do vetor registo de clientes.
- **Delete worker -** remove um trabalhador do vetor registo de trabalhadores.
- **Delete service -** remove um serviço do vetor registo de serviços.

- **Delete Transportation Local -** remove um local de paragem de um transporte público do vetor registo de transportes.

Update

- **CheckIn -** Faz check in de um bilhete de um cliente, alterando o estado de um *bool* que representa o estado do check in.
- **Add Time To Time Table -** acrescenta uma nova hora ao horário
- **Remove Time From Time Table -** remove uma hora do horário

Listagens

Foram implementadas diversas listagens como:

- **List Airports**
- **List Planes**
- **List Flights**
- **List Clients**
- **List Workers**
- **List Services**
- **List Transportation Locals**
- **List Timetable**
- **List Tickets**

Foram implementadas diversas outras funcionalidades como diversos “sorts” e “searches” ao longo do desenvolvimento do programa como também funcionalidades específicas deste mesmo programa tais como :

- **Buy ticket -** efetuar a compra de um bilhete para um determinado voo.
- **Transfer Luggage to the Car -** efetua a transferências das malas de uma “queue”(tapete) para um determinado número de “stacks”.
- **Print Tapete Size -** imprime a quantidade de malas na queue(tapete).
- **Check luggage -** verifica se o cliente já fez check in , colocando a sua bagagem na na queue(tapete).
- **Sort planes -** ordena os aviões presentes no vetor aviões por ordem de capacidade.

Destaque de funcionalidade

Em geral, foi concretizada a implementação de todas as funcionalidades pedidas no enunciado do projeto. Damos especial destaque à forma como organizamos todos os dados necessários e à forma como gerimos as dependências entre classes.

Principais dificuldades e esforço de cada membro

A principal dificuldade encontrada, mesmo apesar de sermos apenas dois alunos, foi a junção de código visto que mesmo fazendo uma estruturação *à priori*, cada um de nós tem uma maneira diferente de estruturação e de redigitação de código tendo levado a algumas dificuldades na junção de partes do trabalho.

Quanto ao empenho, sentimos que ambos os membros investiram um aproximado número de horas e que se esforçaram de maneira similar.