**Gestão de Aeroporto**

QuAIRy *- Flying with class*

Bases de Dados

Ano Letivo 2018/2019

Turma 6 Grupo 6

Daniel Ferreira Brandão, up201705812, up201705812@fe.up.pt

Henrique José Santos, up201706898, up201706898@fe.up.pt

Pedro Miguel Moás, up201705208, up201705208@fe.up.pt

17 de março de 2019

Índice

[Tema do trabalho 3](#_Toc3735036)

[Implementação 4](#_Toc3735037)

[Classes e Atributos 6](#_Toc3735038)

[Diagrama de classes UML 7](#_Toc3735039)

Tema do trabalho

Este projeto tem como objetivo criar uma base de dados que conterá a informação necessária para gerir um aeroporto. Isto inclui os seus funcionários e passageiros, como também informações sobre as viagens que chegam e saem do respetivo aeroporto, para que seja possível geri-las.

Implementação

Começando pela classe **Pessoa**, sobre esta interessa saber informações básicas como nome, NIF e data de nascimento, mas para detalhes mais complexos criaram-se especializações desta, as classes **Funcionário**, que guarda o seu salário e categoria, e **Passageiro**, que apenas contém o seu número de passaporte, no caso de haver problemas de verificação.

A parte central é, naturalmente, a classe **Viagem**, que guarda a data de partida, data de chegada, duração, e se é uma partida ou chegada, dependendo do sentido da viagem. A data de chegada poderia ser determinada através dos outros atributos, mas dependeria da diferença de fuso horário entre aeroportos, sendo necessário aceder a várias outras classes sempre que fosse preciso calcular esse atributo.

Naturalmente, uma **Viagem** tem um **Avião** associado (que por sua vez pode ter várias viagens), cuja classe guarda o seu ID. Um **Avião** pertencerá a uma **Companhia Aérea** (que, certamente, poderá ter vários aviões), e está ligado a uma classe **Modelo** **Avião**, que guarda todos os modelos de aviões conhecidos, para não repetir informação no caso de haver vários com o mesmo modelo, assim como as suas capacidades.

Continuando, também é importante que haja uma gestão das bagagens que passam pelo Aeroporto. Assim, uma **Viagem** pode estar ligada a mais do que uma **Mala**. Sobre esta, guarda-se o seu ID (único) e o peso. Naturalmente, esta classe não corresponde ao objeto físico em si, pois se um passageiro trouxer a mesma mala para dois voos diferentes, certamente que o Aeroporto guardará dois IDs diferentes. Esta classe tem sempre um passageiro associado, que pode ter outras malas associadas. Para além da **Mala**, as viagens de chegada estão ainda sempre associadas a um **Terminal de Bagagem**, onde serão despachadas as suas malas, a uma hora de recolha guardada. Como seria de esperar, estes podem estar ligados a várias viagens.

Para completar as classes **Viagem** e **Passageiro**, também é importante notar que não só uma **Viagem** tem vários passageiros, como um **Passageiro** pode ter várias viagens, pois fica registado na base de dados do Aeroporto no caso provável que retorne. Para cada ligação entre um **Passageiro** e uma **Viagem**, guarda-se informação numa classe de associação Bilhete, com a informação sobre a Classe (Executiva – 0, Primeira Classe – 1, Económica – 2), o número de lugar no avião, se já fez Check-In, e se já embarcou, pois é sempre importante saber onde se encontra um passageiro, no caso de haver um eventual problema. Por isso, guardam-se dois booleanos para guardar essa informação.

Agora, relativamente à classe **Funcionário**, este está ligado a um **Local de Trabalho**, que é uma generalização de **Pista** (desolagem / aterragem), **Porta** (embarque / desembarque), **Terminal de Bagagem**, e **Balcão**. Esta última é uma generalização (completa e disjunta) de **Balcão de Apoio** e **Balcão de Check-in**. A classe **Local de Trabalho** é uma generalização incompleta e disjunta, pois existem outros tipos de locais de trabalho, mas cujos detalhes não são tão cruciais. No entanto, esses funcionários ficam registados na mesma, por uma questão de fiabilidade e consistência. Sobre as classes **Pista** e **Porta**, ambas podem ter várias viagens associadas, mas uma **Viagem** apenas tem uma de cada. Por exemplo, no caso de partida do aeroporto, uma viagem está associada a uma Porta de Embarque. Um Balcão de Check-in está associado a uma ou mais companhias aéreas, e estas podem ter vários balcões também. Um **Funcionário** pode ainda ter um chefe, ou ser chefe de vários funcionários.

Por fim, existem ainda classes **País** e **Cidade**, que são usadas para não repetir informação em muitas classes desnecessariamente. Uma pessoa tem um país, e um aeroporto pertence a uma cidade. Naturalmente que uma cidade pertence a um país, e que um país pode ter uma ou mais cidades.

Classes e Atributos

**Aeroporto**

* Nome
* Local

**Avião**

* ID

**Balcão de apoio**

* Hora de abertura
* Hora de fecho

**Balcão de Check-in**

* Número

**Bilhete**

* Classe
* Número de Lugar
* FezCheckin
* Embarcou

**Cidade**

* Nome

**Companhia Aérea**

* Nome
* Contacto

**Funcionário**

* Salário
* Categoria

**Local de Trabalho**

* ID

**Mala**

* ID
* Peso

**Modelo Avião**

* Nome
* Capacidade

**País**

* Nome

**Passageiro**

* Número de passaporte

**Pessoa**

* Nome
* Data de Nascimento
* Contacto
* NIF
* /idade

**Pista**

* Número

**Porta**

* Setor
* Número
* Embarque

**Recolha de bagagem**

* Hora
* Data

**Terminal de bagagem**

* Número

**Viagem**

* Data de partida
* Data de chegada
* Duração
* Partida

Diagrama de classes UML

O seguinte modelo conceptual, que contém as diversas classes, atributos e associações, também se encontra anexado em formato pdf.

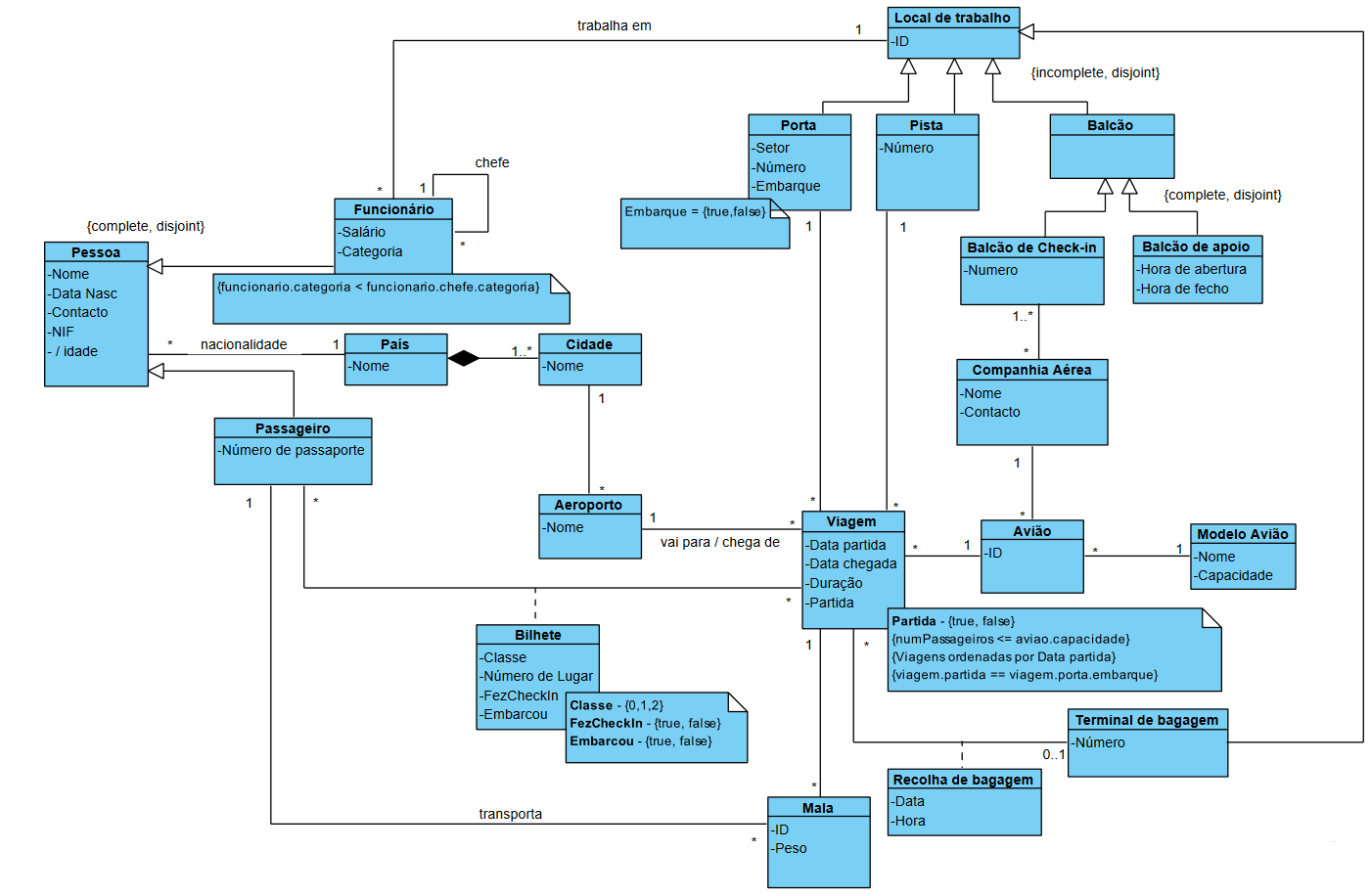


Figura 1 - Diagrama de classes UML

Esquema relacional

**Person** (PersonID, SSN, PersonName, BirthDate, PhoneNumber, Country->Country)

1. {PersonID}-> {SSN, PersonName, BirthDate, PhoneNumber, Country->Country}
2. {SSN}-> {PersonID}
3. {PhoneNumber}->{PersonID}

**Employee** (Person->Person, Salary, NIF, Workplace->Workplace)

1. {Person->Person}->{Salary , NIF , Workplace->Workplace}
2. {NIF}->{Person->Person}

**Passenger** (Person->Person, IDnumber)

1. {Person->Person}->{IDnumber}
2. {IDnumber}->{Person->Person}

**IsBoss** (BossID->Employee, BossedID->Employee)

**Country** (CountryID, CountryName)

1. {CountryID} -> {CountryName}

**City** (CityID, CityName, Country->Country)

1. {CityID}->{ CityName, Country->Country}
2. { CityName, Country->Country}->{CityID}

**Airport** (AirportCode, City->City, AirportName)

1. {AirportCode}->{City->City,AirportName}
2. {City->City,AirportName}->{AirportCode}

**Trip**(TripID)

**Departure** (Trip->Trip, DepartureDate, DepartureTime, ArrivalDate, ArrivalTime, DurationHours, Gate->Gate, Runway->Runway, Airplane->Airplane, Airport->Airport)

1. {Trip->Trip}->{ DepartureDate, DepartureTime, ArrivalDate, ArrivalTime, DurationHours, Gate->Gate, Runway->Runway, Airplane->Airplane, Airport->Airport}
2. {DepartureDate, DepartureTime, Runway->Runway}->{Trip->Trip}

**Arrival** (Trip->Trip, DepartureDate, DepartureTime, ArrivalDate, ArrivalTime, DurationHours, Gate->Gate, Runway->Runway, Airplane->Airplane, Airport->Airport, LuggageBelt->LuggageBelt, DropoffDate, DropoffTime)

1. {Trip->Trip}-> {DepartureDate, DepartureTime, ArrivalDate, ArrivalTime, DurationHours, Gate->Gate, Runway->Runway, Airplane->Airplane, Airport->Airport, LuggageBelt->LuggageBelt, DropoffDate, DropoffTime}
2. {ArrivalDate, ArrivalTime, Runway->Runway}->{Trip->Trip}

**Class** (ClassID, ClassName)

1. {ClassID}->{ClassName}
2. {ClassName}->{ClassID}

**Ticket** (TicketID, Passenger->Passenger, Departure->Departure, Arrival->Arrival, SeatRow, SeatLetter, HasCheckedIn, HasBoarded, HasEnteredBoardingZone, Class->Class)

1. {TicketID}->{Passenger->Passenger, Departure->Departure, Arrival->Arrival, SeatRow, SeatLetter, HasCheckedIn, HasBoarded, HasEnteredBoardingZone, Class->Class}
2. {Passenger->Passenger, Departure->Departure, Arrival->Arrival}->{TicketID}

**Luggage** (LuggageID, Weight, Trip->Trip, Passenger->Passenger)

1. {LuggageID}->{Weight, Trip->Trip, Passenger->Passenger}

**Airplane** (AirplaneID, AirplaneName, Airline->Airline, AirplaneModel->AirplaneModel)

1. {AirplaneID}->{AirplaneName, Airline->Airline, AirplaneModel->AirplaneModel}
2. {AirplaneName, Airline->Airline}->{AirplaneID}

**Airline** (AirlineID, AirlineName, PhoneNumber)

1. {AirplaneID} -> {AirlineName,PhoneNumber}
2. {PhoneNumber}->{AirlineID}

**AirplaneModel** (ModelID, ModelName, SeatsPerRow, NumRows, Capacity)

1. {ModelID}->{ ModelName, SeatsPerRow, NumRows, Capacity }

**Workplace** (WorkplaceID, WorkplaceName)

1. {WorkplaceID}->{WorkplaceName}

**Gate** (Workplace->Workplace, GateName, IsBoardingGate)

1. {Workplace->Workplace}->{GateName, IsBoardingGate}

**Runway** (Workplace->Workplace, RunwayNum)

1. {Workplace->Workplace}->{RunwayNum}
2. {RunwayNum}->{Workplace->Workplace}

**Desk** (Workplace->Workplace)

**CheckInDesk** (Desk->Desk, CheckInNum)

1. {Desk->Desk}->{CheckInNum}
2. {CheckInNum}->{Desk->Desk}

**HelpDesk** (Desk->Desk, OpenTime, CloseTime)

1. {Desk->Desk}->{OpenTime,CloseTime}

**LuggageBelt** (Workplace->Workplace, BeltNum)

1. {Workplace->Workplace}->{BeltNum}
2. {BeltNum}->{Workplace->Workplace}

**HasDesk** (Airline->Airline, CheckInDesk->CheckInDesk)

Restrições

**Person:**

SSN is the PRIMARY KEY

Name and BirthDate are obligatory fields (NOT NULL)

PhoneNumber is UNIQUE

**Passenger:**

IDnumber is the PRIMARY KEY

**Ticket:**

SeatRow and SeatLetter are a composite PRIMARY KEY

HasCheckedIn, HasEnteredBoardingZone and HasBoarded have a default value of FALSE, and are NOT NULL

**Class:**

Name is the PRIMARY KEY, and can assume the values “First Class”, “Economy”, “Premium Economy” or “Business”

**Trip:**

Departure