Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2020/2021

Voar Além Mar

Bárbara Ferreira Teixeira A89610

Carlos Miguel Luzia de Carvalho A89605,

João Pedro da Santa Guedes A89588

Luís Pedro Oliveira de Castro Vieira A89601

Novembro de 2020

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Voar Além Mar

Bárbara Ferreira Teixeira A89610,

Carlos Miguel Luzia de Carvalho A8960,

João Pedro da Santa Guedes A89588,

Luís Pedro Oliveira de Castro Vieira A89601

Novembro de 2020

Resumo

O presente relatório foi realizado no âmbito da Unidade Curricular de Base de Dados, descrevendo a criação de uma base de dados para uma empresa imaginária de venda de bilhetes para voos online, a VAM, Voar Além Mar. Escolhemos implementar esta base de dados por se tratar de uma situação real e cada vez mais presente no mundo atual, especialmente tendo em conta os últimos desenvolvimentos na situação pandémica que o mundo se encontra a enfrentar, conseguindo ser devidamente fundamentada, tal como iremos demonstrar e concluir durante o relatório.

Na primeira parte do relatório, que diz respeito à definição do sistema, contextualizamos e fundamentamos a implementação da base de dados relacional, analisando também a sua viabilidade para a empresa em questão, a VAM.

Na segunda parte, apresentamos os requisitos levantados, classificados e distribuídos como de descrição, exploração e controlo.

Posteriormente, apresentamos o modelo conceptual para a base de dados, tendo sempre em conta os requisitos levantados anteriormente. Além disso, identificamos e classificamos as entidades e os relacionamentos existentes entre as mesmas. Por fim, mas mais importante, a validação do modelo de dados com o utilizador, o que nos permitiu avançar para a seguinte.

Como dito no ponto acima, a fase seguinte diz respeito à modelação lógica da base de dados para a VAM, o qual validamos através das interrogações colocadas pelo utilizador e validando os relacionamentos estabelecidos entre as entidades.

Na quinta e última fase, referente à elaboração do modelo físico de base de dados, selecionamos o sistema responsável pela gestão de base de dados relacional e traduzimos o esquema lógico para o sistema escolhido, terminando com a tradução das interrogações do utilizador em SQL.

Área de Aplicação: Desenho e arquitetura de Sistemas de Bases de Dados
Palavras-Chave: Bases de Dados, Bases de Dados Relacionais, Levantamento de Requisitos, Modelo Conceptual, Modelo Lógico, Modelo Físico, brModelo, MySQL, MySQL Workbench.

Índice

Índice de Figuras

Índice de Tabelas

1. Definição de Sistema

A VAM (Voar Além Mar) é uma empresa responsável por efetuar serviços de venda de bilhetes de avião e necessita de implementar um sistema de base de dados para conseguir de melhor forma fazer uma gestão de clientes e viagens disponíveis. A base de dados deverá guardar e relacionar a informação sobre os serviços existentes de forma a que, a cada momento, seja possível obter uma perspetiva sobre a forma como o negócio se está a desenvolver. Com essa finalidade, o sistema deve estar munido de um conjunto de meios que permita saber, por exemplo, quantos bilhetes foram vendidos para uma determinada viagem, o valor faturado num determinado período de tempo, entre outras informações.
O desenvolvimento do sistema será faseado em diversas etapas, tais como: definição do sistema, levantamento e análise de requisitos, modelação conceptual, modelação lógica, implementação física,

1.1. Contexto de aplicação do sistema

A VAM (Voar Além Mar) é uma empresa fundada em 1990, originalmente Francesa, oriunda de Toulouse. Começou por atender a pequenos/curtos voos na cidade dos aviões e com o desenvolvimento da tecnologia e da internet foi pioneira na venda de bilhetes online, porém nunca abandonou a venda de bilhetes presencial. Atualmente é uma empresa com serviços usados por toda a parte do globo e tenciona melhorar o seu processamento de dados de forma a poder gerenciar melhor as viagens que oferece aos seus clientes.

Com o avanço dos tempos e da procura, cada vez mais, virtual do seu produto, tem tido alguns problemas com a atual forma de albergar os seus dados e informações.

Sendo já uma empresa de renome internacional e tendo já passado 30 anos desde a sua fundação, a VAM tem agora como objetivo abandonar totalmente a venda de bilhetes físicos e unir-se 100% com a evolução global. Posto isto, e querendo passar todas as suas funções para um ponto de vista eletrônico necessita assim de implementar um Sistema Base de Dados funcional e prático.

1.2. Fundamentação da implementação da base de dados

Com o aumento da globalização, qualidade de vida e mesmo esperança média de vida é natural que o uso de serviços de empresas como a VAM (Voar Além Mar) venha sucessivamente a aumentar uma vez que as pessoas têm cada vez mais meios e posses para realizar viagens e também está cada vez mais intrínseca a prática de viagens de trabalho, graças como já referido anteriormente á globalização.

No ano de 2020 a Humanidade encontrou-se perante uma situação pandêmica e por este motivo aliado à grande procura de voos da população numa fase anterior à pandemia a VAM considerou esta altura ser a melhor para abandonar totalmente a venda de bilhetes física, passando assim a vender exclusivamente bilhetes online. Esta medida traz vantagens do ponto de vista econômico uma vez que permitirá reduzir, justificadamente, o número de postos de trabalho mantidos pela empresa, em específico, os que estão associados à venda física de bilhetes, assim como eliminar os gastos inerentes à
exploração e manutenção dos pontos de venda. Por outro lado é uma medida que visa a proteger a segurança dos clientes e funcionários relativamente a postos de venda física não existindo assim contacto interpessoal especialmente nesta primeira fase pós-pandêmica , vem também ajudar no tratamento de dados que ao passar totalmente a uma forma eletrônica vem simplificar todo o processo, e ainda irá permitir aos clientes fazer um melhor acompanhamento das suas viagens, o dinheiro gasto, origem, destino, duração.

Posto isto, a implementação de um sistema de gestão de base de dados torna-se uma medida necessária para a empresa melhor controlar a venda de bilhetes das suas viagens e para processar e armazenar as informações relativas a estas.

1.3. Análise da viabilidade do processo

A VAM, Viajar Além Mar, com a sua vasta história e presença a nível internacional, pretende pronunciar-se como a empresa pioneira a adotar um sistema de bilhetes completamente digital, procurando para isso a melhor solução possível para poder manter a qualidade e excelência de serviço que sempre tem oferecido ao longo de todos estes anos. Para tal é preciso analisar e justificar este projeto para que seja algo viável e com futuro.

A análise da viabilidade deste projeto evidencia-se não só a nível de recursos humanos e espaços físicos, a nível financeiro no que diz respeito à implementação da base de dados mas também no levantamento de vantagens económicas que surgirão da mesma. Do ponto de vista financeiro, sobressalta o custo inicial da implementação da base de dados, algo que será recuperado a curto prazo, uma vez que a adoção deste novo sistema de compra de bilhetes permite reduzir despesas a nível de recursos humanos, nomeadamente em número de funcionário. Por outro lado, ao utilizar um sistema 100% digital, o erro humano é drasticamente diminuído, as longas e demoradas filas de espera verão o seu término passando a existir a

possibilidade de compra de bilhetes por diversos clientes de forma simultânea. Desta forma, a VAM não só se encontra a acompanhar a constante modernização e evolução do mundo tecnológico, como poderá ser alvo de maior atenção, atraindo assim mais clientes.

No que consta quanto aos custos associados à operacionalidade da base de dados não serão superiores aos custos associados à manutenção do atendimento físico, não havendo necessidade de um novo investimento a curto prazo.

A VAM sendo uma empresa de renome mantém registos de todas as viagens que apresentou e para as quais foram vendidas bilhetes portanto, aquando da implementação da base de dados, não serão inseridos dados de viagens prévias, o que faz com que o povoamento inicial seja diminuto. Como tal, o tempo necessário à implementação será mínimo e necessitará de um número reduzido de recursos humanos. Após implementação, também a sua manutenção requererá um número reduzido de recursos humanos.

Dadas as justificações acima enunciadas, chegamos à conclusão que a construção e implementação deste sistema de venda de bilhetes de forma digital é uma mais valia para a empresa, sendo deveras viável.

2. Levantamento e Análise de Requisitos

No capítulo anterior procuramos responder ao porquê da necessidade da implementação de um Sistema Base de Dados. Compreendendo a necessidade deste é depois necessário perceber que tipo de informações este deve guardar para ser funcional e cumprir os seus objetivos. Para isso é fundamental analisar e recolher informação sobre o método de operação desta empresa, pondo nos no lugar de cliente e de administrador do sistema.

2.1. Método de levantamento e de análise de requisitos adotado

Para o processo de levantamento de requisitos recorremos a métodos como a análise de documentos, observação direta e entrevista a pessoas da área.

Relativamente à análise de documentos, procuramos analisar detalhadamente em formato digital e em papel, todas as informações relativas a este, considerando assim informação útil para cliente ou para a empresa.

Relativamente à observação direta pesquisamos em várias agências e sites de viagens o modo como abordam os clientes ao efetuarem a compra de um bilhete, toda a informação que lhes é pedida e ou fornecida.

Por fim, referindo me à entrevista, procuramos entrevistar clientes assíduos nas viagens aéreas, tanto clientes mais ligados à vertente empresarial , como ligados à vertente turística, e assim procuramos saber que melhorias aos sistemas já existentes eles sugerem e que informação eles gostariam de ver representada, ou ter acesso.

2.2. Requisitos levantados

Através da informação que fomos capazes de recolher, procedemos à definição dos requisitos que a base de dados deverá suportar. Procuramos dar uma maior estruturação à informação obtida, definindo o papel de cada uma das componentes que formam o negócio da empresa.

De todo este processo resultou a diferenciação entre o posto de vista do cliente e a perspetiva do administrador, concluindo também que os objetivos relativos à utilização da base de dados de cada um são também divergentes.

2.2.1 Requisitos de descrição

Cliente

Os clientes devem estar registados no sistema para poderem ser identificados e comprar bilhetes. Os dados guardados sobre cada cliente incluem o seu nome, idade, contactos (telemóvel, email), morada, nacionalidade, o seu Número de Identificação Fiscal (NIF) e a password que permite o acesso à aplicação.

Relativamente ao NIF, sendo um número único vai ser usado para identificação do cliente na aplicação. Tendo efetuado o login na aplicação, o cliente poderá comprar bilhetes a qualquer altura desde que seja para um voo que ainda não tenha começado.

Bilhete

Um bilhete é adquirido por um cliente que pretende realizar um determinado voo. Assim, a um bilhete está associado um e um só voo e a apenas um cliente. Os dados associados a cada bilhete são: o preço, um número único gerado pelo sistema (ID) sendo o número identificativo do bilhete, data de aquisição, Gate do aeroporto (Porta de embarque) e o lugar no avião dividido entre o tipo de classe em que embarca (econômica ou executiva) e o número do lugar.

Voo

Cada voo guarda o ID do voo, o número do voo (considerámos que este é reiniciado passado um determinado intervalo de tempo), hora de partida, hora de chegada, data de partida e o número de bilhetes vendidos ,sendo que uma viagem é realizada num determinado avião o que faz com que o número máximo de bilhetes comprados seja limitado ao avião que realizará a viagem.

Avião

Um avião pode realizar várias viagens. A informação guardada para cada avião passa pelo registo do seu número de identificação, pelo número máximo de lugares, identificados pela sua classe, pelo peso máximo que pode transportar (tara), um multivalorado que representa os lugares existentes num dado avião e a companhia a que pertence.

Aeroporto

Um voo é efetuado entre dois aeroportos, sendo que um aeroporto pode ser origem ou destino de várias viagens. Os dados guardados para o aeroporto são o nome, uma localização (que inclui o seu país, localidade e código-postal) e um número que permite identificá-lo.

2.2.2 Requisitos de exploração

Do ponto de vista do cliente deve ser possível:

1. Ver o histórico dos seus voos num dado período.
2. Consultar o montante gasto num dado período.
3. Visualizar as informações associadas ao seu bilhete.
4. Pesquisar os voos disponíveis num dado período.
5. Consultar a lista de lugares livres para um voo.
6. Consultar os voos disponíveis num determinado aeroporto.
7. Consultar a lista de voos existentes.
8. Consultar a lista de aeroportos existentes.

Do ponto de vista do administrador da aplicação deve ser possível:

1. Consultar os voos realizados por uma determinada companhia num dado período.
2. Saber quais os passageiros que viajaram entre dois aeroportos num dado período.
3. Saber quais os passageiros que estiveram presentes num voo.
4. Verificar quantos bilhetes foram vendidos num dado período.
5. Calcular o valor total faturado num dado período.

2.2.3 Requisitos de controlo

Do ponto de vista de um cliente, este deve conseguir:

1. Inserir os dados para efetuar o registo.
2. Inserir dados para a compra de um novo bilhete.
3. Atualizar a informação inserida aquando do registo.

Do ponto de vista do administrador da aplicação, deve ser possível:

1. Inserir um novo voo.
2. Inserir dados sobre um novo avião, assim como os lugares existentes neste.
3. Adicionar novos aeroportos.
4. Atualizar os dados relativos a um voo.

2.3. Análise geral dos requisitos

Os requisitos de exploração registados demonstram ser pormenorizados relativamente ao funcionamento da VAM , Voar Além Mar, uma vez que englobam e descrevem todas as componentes que constituem o ambiente em que a empresa se encontra inserida e detalham as interações entre estes.

Com os requisitos de exploração e controle, torna-se evidente o papel de cada um dos utilizadores da base de dados e o objetivo final da mesma.

Concluímos então que os requisitos registados identificam corretamente os dados que serão guardados na base de dados e o quais os objetivos a cumprir com o funcionamento e utilização do sistema.

3. Modelo Conceptual

Uma vez concluída a definição dos requisitos e tendo já realizado uma análise detalhada dos mesmos, o próximo passo na implementação do sistema de base dados da VAM, Voar Além Mar, é a modelação conceptual do mesmo. Este, é independente dos detalhes da implementação e procura somente representar de forma fidedigna o modelo de informação usado pela empresa. Para facilitar a definição e interpretação do modelo, o mesmo é acompanhado de documentação, composta por um diagrama de ER e um dicionário de dados sobre as entidades, relações e atributos identificados.

3.1. Apresentação da abordagem de modelação realizada

A abordagem de modelação que decidimos aplicar na idealização e realização do modelo é uma abordagem adaptada da metodologia de Connolly & Begg (2005). Neste caso, procedemos da seguinte forma:

1. Identificação dos tipos de entidades existentes.
2. Identificação dos tipos de relacionamento.

3. Identificação e associação de atributos a cada tipo de entidade ou relacionamento.
4. Determinação do domínio dos atributos
5. Determinação das chaves candidatas, primárias e estrangeiras
6. Consideração do uso de detalhe ou generalização de entidades
7. Validação do modelo de dados

Tendo isto em conta, foi possível definir o modelo conceptual da maneira que vamos apresentar nas secções que se seguem.

3.2. Identificação e caracterização das entidades

Para a realização de um modelo conceptual, é necessário definir as várias entidades existentes, recorrendo para tal aos vários requisitos levantados. A primeira entidade identificada foi o Cliente, pois é esta entidade que vai efetuar as compras de bilhetes de avião e, por isso, é a entidade com mais dados guardados na base de dados. Depois do Cliente ser identificado, é fácil identificar mais duas entidades: Bilhete e Voo, sendo que o Bilhete é aquilo que o Cliente vai comprar para que este possa viajar num determinado Voo. De seguida, identificamos o Aeroporto e o Avião, porque a realização de um Voo necessita diretamente de um Avião, e um Voo inicia e termina num determinado Aeroporto.

Tabela 1 - Entidades

Entidade	Descrição	Aliases	Ocorrência
Cliente	Utlizadores da aplicação da empresa ou utilizadores dos serviços desta.	Utilizador	Cada cliente pode ou não comprar bilhetes para uma ou masi viagens à escolha
Bilhete	Termo geral que discreve a reserva de um utilizador para uma determinada viagem	Reserva	O bilhete é sempre associado a um cliente e a um e só um voo. Uma viagem possui um número limitado de bilhetes.
Voo	Serviço oferecido pela empresa, viagem entre duas localizações.	-	A um voo estão associados vários bilhetes.
Avião	Meio usado para realizar o voo	-	Um avião possui vários lugares dividindo-se este entre classe econômica ou executiva
Aeroporto	Ponto físico onde os voos se iniciam ou terminam	-	Um aeroporto pode ser ponto de origem ou destino de um voo.

3.3. Identificação e caracterização dos relacionamentos

Estando identificadas as entidades é necessário evidenciar como é que estas se relacionam, algo que fazemos analisando, novamente, os requisitos levantados anteriormente.

Identificámos as seguintes relações:

Cliente -> Bilhete

Esta relação representa a compra de um bilhete por parte do cliente. O cliente pode comprar vários bilhetes, ou pode nunca ter comprado um bilhete. Mas cada bilhete comprado relaciona-se apenas com um cliente, aquele que o comprou. Logo, estamos perante um relacionamento de 0 para N, sendo que o cliente pode ou não comprar um determinado bilhete (participação opcional) e o bilhete é apenas registado na base de dados se já foi comprado (participação obrigatória).

Bilhete -> Viagem

Cada bilhete relaciona-se com um único voo (participação obrigatória), no entanto um voo relaciona-se com vários ou nenhum bilhete (participação opcional), pois para um determinado voo podem ser comprados vários bilhetes ou nenhum. Consequentemente, obtemos um relacionamento de N para 1.

Voo -> Aeroporto

Cada voo tem dois relacionamentos com o aeroporto porque cada voo parte de um aeroporto e chega a um aeroporto. No entanto, um aeroporto pode ser a origem e destino de vários voos. Logo, ambos estes relacionamentos são de N para 1. A participação de ambos estes relacionamentos é obrigatória.

Voo -> Avião

Cada voo é realizado por um avião, e esta participação é obrigatória porque se um voo vai ser efectuado, será efectuado por um avião, logo este relacionamento existirá em todos os voos. Pelo contrário, um avião pode realizar vários voos, desde que estes não sejam efectuados em simultâneo e pode não efetuar nenhum voo, logo a sua participação é opcional.

Tabela 2 - Relacionamentos

Entidade	Cardinalidade	Relacionamento	Cardinalidade	Entidade
Cliente	1..1	Possui	0..n	Bilhete
Bilhete	1..n	Associado a	1..1	Voo
Voo	1..n 1..n 1..n	Parte de Chega a Feito por	1..1 1..1 1..1	Aeroporto Aeroporto Avião

3.4. Identificação e caracterização da associação dos atributos com as entidades e relacionamentos

Depois de identificarmos as entidades e os seus relacionamentos, é necessário identificar as informações referentes a cada entidade que pretendemos guardar no nosso sistema. Estes atributos são facilmente identificados analisando os requisitos levantados.

Agora apresenta-se o dicionário de dados relativo aos atributos estabelecidos (Dividido pelas várias entidades):

Tabela 3 - Atributos da entidade Bilhete

Entidade	Atributo	Descrição	Data Type	Null	Multivalorado	Derivado	Composto
Bilhete	ID	Identificador do bilhete	INT	Não	Não	Não	Não
	Data	Data de aquisição	DATE	Não	Não	Não	Não
	Gate	Porta de embarque	VARCHAR(3)	Não	Não	Não	Não
	Número	Número do bilhete	VARCHAR(3)	Não	Não	Sim	Não
	Classe	Classe escolhida	VARCHAR(30)	Não	Não	Não	Não
	Preço	Preço do bilhete	DECIMAL(6,2)	Não	Não	Não	Não

Tabela 4 - Atributos da entidade Aeroporto

Entidade	Atributo	Descrição	Data Type	Null	Multivalorado	Derivado	Composto
Aeroporto	ID	Identificador do aeroporto	INT	Não	Não	Não	Não
	Localização	Localização do aeroporto		Não	Não	Não	Sim
	País		VARCHAR(45)	Não	Não	Não	Não
	Localidade		VARCHAR(45)	Não	Não	Não	Não
	Código-Postal		VARCHAR(20)	Não	Não	Não	Não
	Nome	Nome do aeroporto	VARCHAR(100)	Não	Não	Não	Não

Tabela 5 - Atributos da entidade Avião

Entidade	Atributo	Descrição	Data Type	Null	Multivalorado	Derivado	Composto
Avião	ID	Identificador do avião	INT	Não	Não	Não	Não
	Nome	Nome do avião	VARCHAR(20)	Não	Não	Não	Não
	Numero máximo de passageiros Classe Económica	Número de lugares que o avião disponibiliza para a classe económica	TIME	Não	Não	Não	Não
	Numero máximo de passageiros Classe Executiva	Número de lugares que o avião disponibiliza para a classe executiva	TIME	Não	Não	Sim	Não
	Numero máximo de Passageiros	Número de lugares disponíveis no avião	TIME	Não	Não	Sim	Não
	Tara	Peso máximo que o avião transporta	DATE	Não	Não	Não	Não
	Companhia	Companhia aérea	INT	Não	Não	Não	Não
	Lugar	Lugares do avião		Não	Sim	Não	Sim
	ID		INT	Não	Não	Não	Não
	Classe		VARCHAR(45)	Não	Não	Não	Não
	Número		VARCHAR(3)	Não	Não	Não	Não

Tabela 6 - Atributos da entidade Cliente

Entidade	Atributo	Descrição	Data Type	Null	Multivalorado	Derivado	Composto
Cliente	NIF	Número de contribuinte do cliente	DOUBLE	Não	Não	Não	Não
	Nome	Nome do cliente	VARCHAR(45)	Não	Não	Não	Não
	Idade	Idade do cliente	INT	Não	Não	Não	Não
	Contacto	Contactos do cliente		Não	Não	Não	Sim
	Telemóvel		VARCHAR(20)	Não	Não	Não	Não
	Email		VARCHAR(45)	Não	Não	Não	Não
	Morada	Morada do cliente	VARCHAR(100)	Não	Não	Não	Não
	Nacionalidade	País de nascimento do cliente	VARCHAR(15)	Não	Não	Não	Não
	Password	Password associada ao email	VARCHAR(45)	Não	Não	Não	Não

Tabela 7 - Atributos da entidade Voo

Entidade	Atributo	Descrição	Data Type	Null	Multivalorado	Derivado	Composto
Voo	ID	Identificador do voo	INT	Não	Não	Não	Não
	Nº de voo	Número do voo	VARCHAR(20)	Não	Não	Não	Não
	Hora de Partida	Hora a que o voo inicia	TIME	Não	Não	Não	Não
	Hora de Chegada	Hora a que o voo termina	TIME	Não	Não	Sim	Não
	Duração	Duração do voo	TIME	Não	Não	Não	Não
	Data de Partida	Data do voo	DATE	Não	Não	Não	Não
	Nº de bilhetes vendidos	Número de bilhetes vendidos deste voo	INT	Não	Não	Não	Não

3.5. Detalhe ou generalização de entidades

Relativamente às entidades por nós referidas, uma vez que representam todos objetos diferentes não houve necessidade de recorrer a detalhe ou generalização destas. Não foi necessário recorrer ao detalhe de entidades pois as ocorrências das entidades de um mesmo tipo terão características semelhantes, acontecerão sobre contextos idênticos e, acima de tudo, representarão objetos também eles idênticos.

Não foi também necessário recorrer à generalização dado que embora entidades com um ou outro atributo igual entre si, a grande maioria do conjunto de atributos para cada entidade varia, assim sendo não consideramos necessário recorrer a generalização.

3.6. Apresentação e explicação do diagrama ER

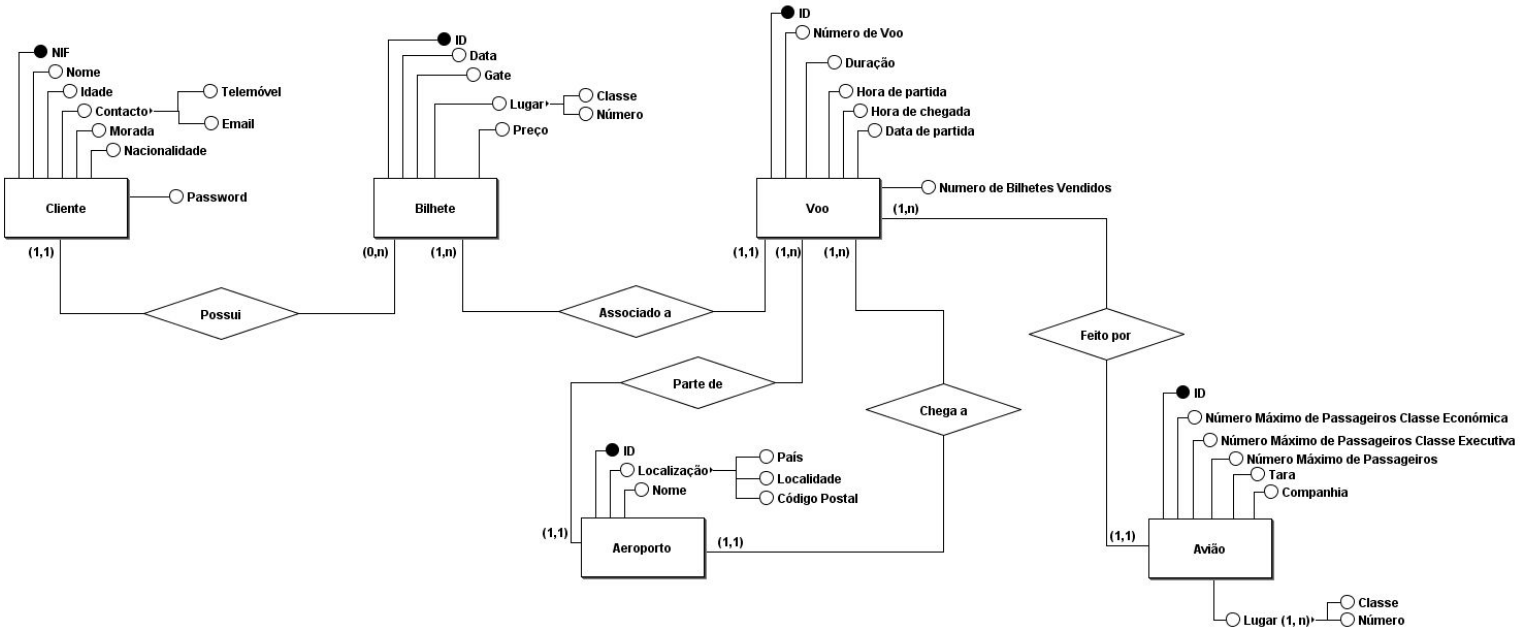


Figura 1- Diagrama ER

Em conformidade com as entidades e atributos acima mencionados, assim como os relacionamentos estabelecidos entre estas, também acima referidos temos então o modelo conceptual. Posto isto resta explicar a escolha das chaves primárias de cada entidade.

. Relativamente ao **Cliente** identificámos 3 possíveis chaves candidatas, o nif , o email ou o número de telefone porém uma vez que por cada registo na aplicação é usado um email e telemóvel estes se encontram associados a uma só conta, poderíamos usar este atributo como chave da entidade **Cliente**, contudo o cliente pode mudar de email ou número de telemóvel o que afasta estes atributos da seleção para chave primária. Sendo assim escolhemos o NIF como a chave primária de **Cliente** porque embora seja um número bastante grande a Voar Além Mar é uma empresa internacional com milhões de utilizadores, para além disso é um número único.

Na entidade **Bilhete** apenas o seu número número identificador serve como chave candidata, sendo automaticamente selecionado para chave primária.

Relativamente à entidade **Voo** identificámos duas possíveis chaves candidatas: o ID do voo ou o número de voo. Porém escolhemos o ID como chave primária uma vez que considerámos que o número de voo poderia eventualmente repetir-se

Na entidade **Aeroporto** ambos os três principais atributos poderiam ser consideradas chaves primárias porém optamos pelo número de identificação uma vez que este é atribuído pelo sistema e é menos suscetível a alterações consoante o tempo.

Por último, a entidade **Avião** apenas pode ser identificada pelo número de identificação.

3.7. Validação do modelo de dados produzido

Dado por concluído o processo de modelação conceptual da base de dados, torna-se necessário validar o modelo produzido. Para tal devemos ser capazes de satisfazer cada um dos requisitos previamente apresentados.

No caso do cliente este deverá ser capaz de ver o histórico dos seus voos num dado período, consultar o montante gasto num dado período, visualizar as informações associadas ao seu bilhete, pesquisar os voos disponíveis num dado período, consultar a lista de lugares livres para um voo, consultar os voos disponíveis num determinado aeroporto, consultar a lista de voos existentes e consultar a lista de aeroportos existentes. Assim:

- ➔ Para podermos consultar o histórico dos seus voos num dado período necessitamos das entidades Voo e Bilhete. Uma vez que cada Bilhete é associado a um Voo, e que cada Bilhete é comprado por um Cliente, este pode consultar o histórico das viagens que já foram por si realizadas num dado período de tempo, visto que cada Voo tem uma data de partida, consultando os bilhetes que adquiriu para esse período.
- ➔ Visto que cada Cliente pode aceder aos bilhetes que adquiriu, e que estes são relativos a Voos já realizados, é possível verificar o montante gasto num dado período somando os preços associados a cada Bilhete.
- ➔ Como cada Cliente pode consultar o preço, o lugar no avião, o gate a que se deve dirigir para embarcar, o número do Voo, a companhia que o realiza, uma vez que são atributos seus, e a data e hora de partida bem como a hora de chegada dado serem atributos do Voo relacionados com o Voo. Visto que cada Voo tem origem num Aeroporto e um outro como Destino, é possível também consultar a origem e destino do Voo relativo a um Bilhete.
- ➔ Cada Voo tem uma data de partida, e portanto é possível pesquisar os voos disponíveis num dado período de tempo.
- ➔ Cada Voo é feito por um Avião, e cada um deste último tipo tem uma lista de lugares. Como cada Bilhete tem um lugar associado, é possível averiguar quais os lugares livres para um dado Voo, averiguando os lugares de um Avião com o lugar atribuído a cada um dos bilhetes relativos a esse Voo. Assim os

lugares que ficarem de fora desta intersecção serão os lugares livres para o Voo requerido.

- ➔ Visto que cada Voo tem origem num Aeroporto, é possível listar os voos que daí partem.
- ➔ Existindo uma entidade Voo, podemos consultar todos os voos disponíveis.
- ➔ Existindo uma entidade Aeroporto, podemos consultar todos os aeroportos disponíveis.

Já no caso do administrador, este deve ser capaz de consultar os voos realizados por uma determinada companhia num dado período, saber quais os passageiros que viajaram entre dois aeroportos num dado período, saber quais os passageiros que estiveram presentes num voo, verificar quantos bilhetes foram vendidos num dado período e calcular o valor total faturado num dado período. Desta forma:

- ➔ Como cada Voo é realizado por um dado Avião, e tem uma data e hora de partida bem como hora de chegada, e visto que o Avião possui informação sobre a companhia, então é possível averiguar os voos realizados por uma determinada companhia num dado período.
- ➔ Cada Bilhete comprado por um Cliente está associado a um Voo, e este último possui data e hora de partida bem como hora de chegada, estando também associados a um Aeroporto de partida e outro de destino. Desta forma, conseguimos saber quais os clientes que realizaram um certo voo entres dois aeroportos num dado período de tempo.
- ➔ Como cada Bilhete comprado por um Cliente está associado a um Voo, é possível averiguar quais os passageiros que estiveram presentes num voo.
- ➔ Cada Bilhete tem uma data associada, neste caso de aquisição, logo é possível verificarmos quantos bilhetes foram comprados num dado período.
- ➔ Cada Bilhete tem um preço e uma data associada. Desta maneira podemos calcular o valor total faturado num dado período, bastando apenas somar o valor do preço de todos os bilhetes adquiridos num dado período.

Dadas as justificações acima enunciadas, consideramos que o modelo de dados apresentado satisfaz todas as necessidades relativas à base de dados.

4. Modelação Lógica

Após a conceptualização do modelo do nosso sistema de base de dados ter sido validada, prosseguimos a definição do modelo lógico. A sua construção foi orientada pelo modelo de dados relacional.

No presente capítulo, procede-se à descrição do processo de construção deste modelo, sendo o resultado apresentado de seguida. Sobre o resultado final são explicadas as devidas validações, nomeadamente através da validação com interrogações do utilizador e validação com as transações estabelecidas. Por fim, é enunciada e resumida a revisão do modelo obtido.

4.1. Construção e validação do modelo de dados lógico

De forma a identificarmos e estabelecermos as relações necessárias para que o modelo lógico fosse capaz de representar as entidades, relacionamentos e atributos necessários, analisamos o noss
o modelo conceptual por fases de forma a fazermos a transição do modelo conceptual para uma modelação lógica, derivando todas essas relações para o modelo lógico. A ordem foi a seguinte:

1. Entidades Fortes;
2. Entidades Fracas;
3. Relacionamentos binários um para muitos (1-N);
4. Relacionamentos binários um para um (1-1);
5. Relacionamentos recursivos um para um (1-1);
6. Relacionamentos superclasse/subclasse;
7. Relacionamentos binários muitos para muitos (N-M);
8. Relacionamentos complexos;
9. Atributos multivalor.

1. Tendo em consideração que uma entidade forte é uma entidade que tem atributos suficientes para formar uma chave primária, sem depender de outras entidades, concluímos que todas as entidades que identificamos são entidades fortes. Assim, no

modelo lógico, foi criada uma relação que inclui todos os atributos simples dessa entidade, identifica qual a sua chave primária, e as chaves alternativas, se existirem para essa entidade.

Entidade: Cliente (NIF, Nome, Idade, Telemóvel , Email, Morada, Nacionalidade, Password)
Chave primária: NIF

Entidade: Bilhete (id, Data, Gate, Lugar, Classe, Preço)
Chave primária: id

Entidade: Voo (id, Numero_de_Voo, Origem, Destino, Hora_de_Partida , Hora_de_Chegada, Data_de_Partida, Numero_de_bilhetes_Vendidos)
Chave primária: id

Entidade: Avião (id, Nome, Numero_maximo_de_passageiros_classe_economica, Numero_maximo_de_passageiros_classe_executivo, Numero_maximo_de_passageiros, Tara, Companhia)
Chave primária: id

Entidade: Aeroporto (id, País, Localidade, Codigo-Postal, Nome)
Chave primária: id

2. Como especificado anteriormente, não identificamos nenhuma entidade fraca.
3. Neste tipo de relacionamento, a entidade com multiplicidade N inclui um novo atributo, que é a chave primária da entidade com multiplicidade 1, como chave estrangeira. Foram identificadas, no nosso trabalho, 5 relações de 1 para N:

Bilhete (id, Data, Gate, Lugar, Classe, Preço)
Chave primária: id
Chaves estrangeiras: Cliente_NIF , Voo_id (O que significa que existe uma relação Cliente-Bilhete de 1 para N, e uma relação Bilhete-Voo de N para 1)

Voo (id, Numero_de_Voo, Origem, Destino, Hora_de_Partida , Hora_de_Chegada, Data_de_Partida, Numero_de_bilhetes_Vendidos)
Chave primária: id
Chaves estrangeiras: Origem_id, Aviao_id , Destino_id (O que significa que existe uma relação Voo-Avião de N para 1 e duas relações Voo-Aeroporto de N para 1)

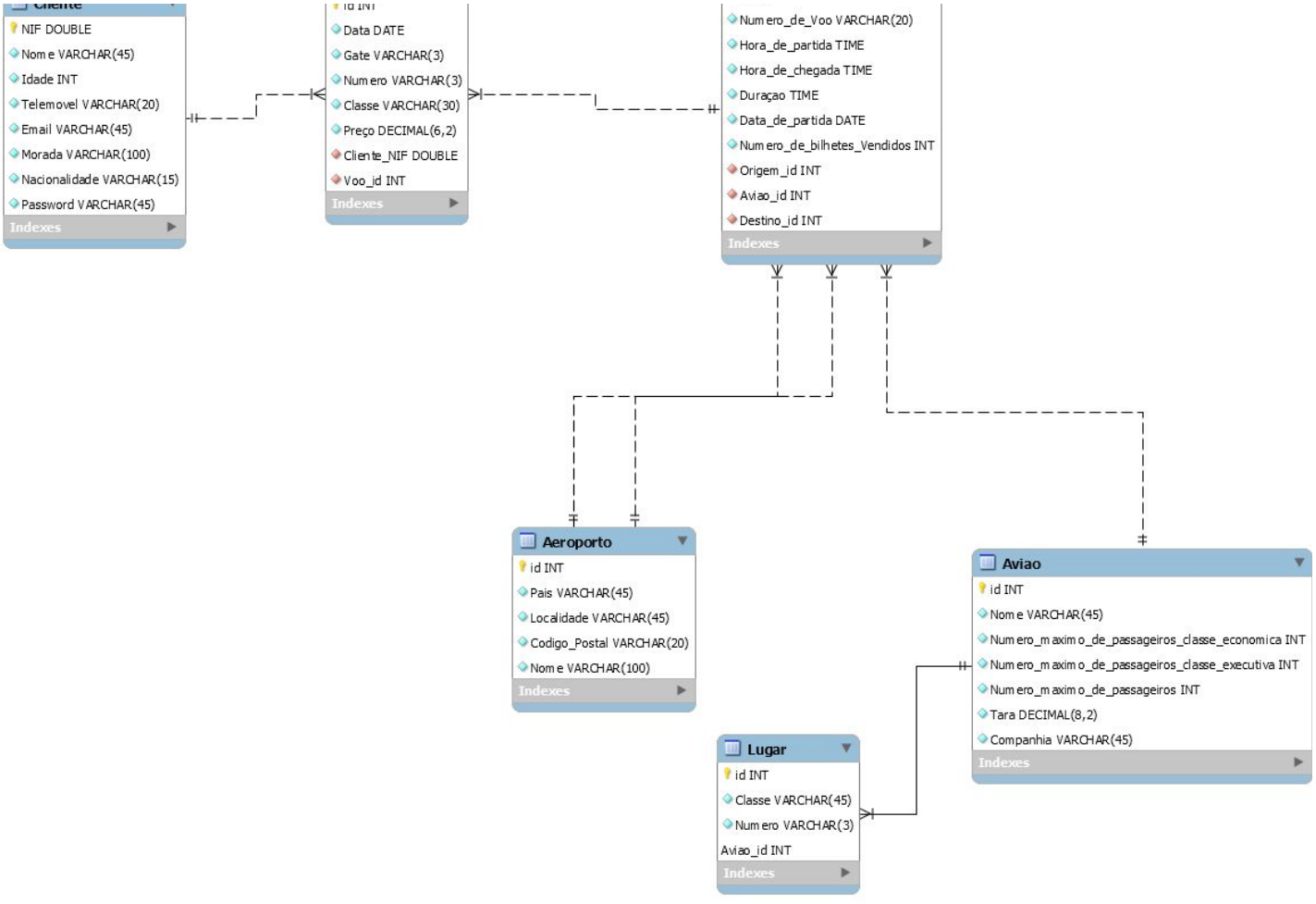
Até ao nono passo, não foram encontrados relacionamentos desses tipos no nosso modelo.

9. Este tipo particular de atributo pode assumir diferentes valores para a mesma entidade. Foi encontrado um atributo deste tipo na entidade Avião, pelo facto de um avião incluir vários lugares diferentes. Para que seja possível identificar a que avião é que este lugar pertence, este atributo vai criar uma relação e vai incluir a chave do avião a que pertence:

Lugar (id, Classe, Numero)
Chave primária: id
Chave estrangeira: Aviao_id

4.2. Desenho do modelo lógico

Figura 2 - Diagrama do Modelo Lógico



4.3. Validação do modelo com interrogações do utilizador

Para verificar se o modelo que construímos responderia aos requisitos do utilizador, tivemos de recorrer à sua validação através de interrogações ao mesmo. Posto isto relativamente aos clientes usuais da Voar Além Mar verificámos a resposta do modelo lógico às questões apresentadas da seguinte forma:

1.1 - Consigo ver o histórico das minhas viagens ?

Selecionamos da tabela dos clientes o Cliente que pretendemos e juntamos o resultado com a tabela dos bilhetes.

1.2 - E o meu histórico num dado período de tempo?

Realizamos o mesmo processo da questão anterior e de seguida selecionamos da tabela das viagens apenas aquelas que estão no intervalo que pretendemos e, por fim, juntamos o resultado desta etapa com o da primeira, obtendo desta forma o histórico das viagens dum cliente num dado período.

2 - Posso ver quanto gastei em voos num determinado período de tempo?

Efetuamos o mesmo processo referido em 1.2 e após já termos a junção das tabelas iniciais, somamos o valor do preço dos bilhetes desse cliente, obtendo assim o montante gasto num intervalo de tempo.

*

3 - Posso ver as informações do voo associadas ao meu bilhetes?

Para isto selecionamos o bilhete a consultar na tabela dos bilhetes e juntamos o resultado com a tabela dos voos, depois com a tabela dos aeroportos obtemos as informações que pretendemos.

4 - Posso pesquisar voos num dado período de tempo entre duas localizações?

Inicialmente temos que selecionar da tabela de voos as que estão naquele período de tempo, depois selecionamos os dois aeroportos na localização pretendida, convergindo as duas tabelas obtemos os voos disponíveis no intervalo de tempo.

5 - Consigo consultar a lista de lugares livres para um voo?

Cruzando a informação retirada da tabela de voos, relativamente ao voo que queremos consultar com a tabela dos aviões, relativamente ao avião que vai proceder ao voo e a tabela de lugares desse avião obtemos a lista de todos os lugares, ao subtrair a tabela obtida com a junção dos voos com o bilhete obtemos a lista de lugares livres.

6 - Posso consultar os voos que partem de um determinado aeroporto numa certa data?

Inicialmente, selecionamos o aeroporto que queremos da tabela respectiva e juntamos com a tabela de voos. Da tabela resultante, selecionamos as viagens com a data que pretendemos.

7 - Consigo consultar a lista de voos existentes ?

Basta consultar a tabela de voos

8 - Consigo consultar a lista de aeroportos existentes?

Basta consultar a tabela de aeroportos.

Com intuito de procurar uma validação do modelo também por parte da administração da VAM, fomos também interrogar estes.

1 - Consigo consultar os voos realizados por uma determinada companhia num dado período de tempo?

Selecionando a tabela de voos e a tabela dos aviões, conseguimos descobrir os voos realizados por uma determinada companhia num dado período de tempo.

2- Consigo saber quais os passageiros que viajaram entre dois aeroportos num dado período?

Selecionando a tabela de voos e a tabela de bilhetes e cruzando com os aeroportos pretendidos da tabela de aeroportos, conseguimos obter os passageiros que viajaram entre esses aeroportos num dado período.

3 - Consigo saber quais os passageiros que estiveram presentes num voo?

Selecionando o voo pretendido na tabela de voos e convergindo esse voo com a tabela de bilhetes a ele respetiva, obtemos os passageiros/clientes que participaram neste voo.

4 - Consigo verificar quantos bilhetes foram vendidos num dado período?

Selecionando na tabela dos bilhetes todos aqueles cuja data de aquisição está no intervalo pretendido e contamos quantos são.

5 - Consigo calcular o valor total faturado num dado período?

Selecionando na tabela dos bilhetes todos aqueles cuja data de aquisição está no intervalo pretendido e calculando o somatório de todos os preços referentes aos bilhetes no intervalo.

4.4. Revisão do modelo lógico produzido

Nesta fase final da modelação lógica da base de dados, foi necessário rever e validar a mesma com o objetivo de garantir que toda a estrutura criada satisfaz os requisitos desejados para a mesma. Como podemos ver nos pontos acima enunciados, através do modelo lógico apresentado fomos capazes de perceber os tipos de entidades existentes, a forma como os relacionamentos entre elas as afetam e demonstrámos ser capazes de responder às questões colocadas pelo utilizador.

Assim, após esta revisão, consideramos que o modelo lógico é válido e viável uma vez que cumpre todos os requisitos pré-estabelecidos, considerando-o assim validado.

5. Implementação Física

A última fase do processo de construção do sistema de base de dados passou pela sua implementação física, seguida do povoamento, exploração e monitorização da informação armazenada. Começamos por traduzir o esquema lógico, apresentado no capítulo anterior, para o SGBD escolhido. Uma vez implementado o esquema físico, procedemos à tradução das interrogações do utilizador e das transações estabelecidas para SQL. Por conseguinte, definimos índices e vistas de utilização. Por fim, implementamos mecanismos de segurança com o objetivo de restringir o acesso dos utilizadores a certas partes da base de dados.

Neste capítulo, para além de uma descrição detalhada de cada um dos processos referidos anteriormente, é também apresentada uma estimativa do espaço que a base de dados ocupará em disco, e é descrita a revisão do sistema implementado junto da empresa.

5.1. Seleção do sistema de gestão de base de dados

A fim de desenvolver o esquema físico da base de dados, tivemos, inicialmente, de escolher qual o SGBD a utilizar. Uma vez que nos encontramos no domínio do modelo de dados relacional, optamos por usar o MySQL pois além de ser aquele com o qual nos encontramos mais familiarizados, disponibiliza também mecanismos de controle de concorrência, que permite evitar futuros problemas no contexto da aplicação.

5.2. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL

No processo de construção do esquema físico começamos por identificar as relações base, produzidas referentes às definições apresentadas no esquema lógico. Assim temos:

Relação Cliente

Domínio **Nome**: string de comprimento variável, comprimento 45
Domínio **Idade**: inteiro
Domínio **Telemóvel**: string de comprimento variável, comprimento 20
Domínio **Email**: string de comprimento variável, comprimento 45
Domínio **Morada**: string de comprimento variável, comprimento 100
Domínio **Nacionalidade**: string de comprimento variável, comprimento 15
Domínio **Número** Identificação Fiscal: double
Domínio **Password**: string de comprimento variável, comprimento 45

Cliente(

nome	Nome	NOT NULL,
idade	Idade	NOT NULL,
telemóvel	Telemóvel	NOT NULL,
email	Email	NOT NULL,
morada	Morada	NOT NULL,
nacionalidade	Nacionalidade	NOT NULL,
nif	Número Identificação Fiscal	NOT NULL,
password	Password	NOT NULL,

CHAVE PRIMÁRIA (NIF)
CHAVE ALTERNATIVA (telemóvel)
CHAVE ALTERNATIVA (email));

Transpondo para código SQL:

```
-- -----------------------------------------------------
-- Table `IngressosDeAvioes`.`Cliente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `IngressosDeAvioes`.`Cliente` (
  `NIF` DOUBLE NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Idade` INT NOT NULL,
  `Telemovel` VARCHAR(20) NOT NULL,
  `Email` VARCHAR(45) NOT NULL,
  `Morada` VARCHAR(100) NOT NULL,
  `Nacionalidade` VARCHAR(15) NOT NULL,
  `Password` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`NIF`))
ENGINE = InnoDB;
```

Figura 3 - Código SQL correspondente à criação da tabela Cliente

Relação Bilhete

Domínio **ID** Bilhete: inteiro
Domínio **Data**: temporal
Domínio **Gate**: string de comprimento variável, comprimento 3
Domínio **Numero**: string de comprimento variável, comprimento 3
Domínio **Classe**: string de comprimento variável, comprimento 30
Domínio **Número**: inteiro
Domínio **Preço**: valor monetário, decimal(6,2)
Domínio **Cliente_NIF**: double
Domínio **Voo_id**: inteiro

Bilhete(

id	ID Bilhete	NOT NULL,
data	Data Aquisição	NOT NULL,
gate	Porta de embarque	NOT NULL
classe	Classe	NOT NULL,
número	Número	NOT NULL,
preco	Preço	NOT NULL,

cliente_nif	Nif do Cliente	NOT NULL,
voo_id	Voo	NOT NULL,

CHAVE PRIMÁRIA (id),
CHAVE ESTRANGEIRA (cliente) REFERÊNCIA Cliente(id_cliente)
ON UPDATE CASCADE
ON DELETE NO ACTION,
CHAVE ESTRANGEIRA (voo_id) REFERENCIA Voo(id_voo)
ON UPDATE CASCADE
ON DELETE NO ACTION);

Transpondo para código SQL:

```
-- -----------------------------------------------------
-- Table `IngressosDeAvioes`.`Bilhete`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `IngressosDeAvioes`.`Bilhete` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `Data` DATE NOT NULL,
  `Gate` VARCHAR(3) NOT NULL,
  `Numero` VARCHAR(3) NOT NULL,
  `Classe` VARCHAR(30) NOT NULL,
  `Preço` DECIMAL(6,2) NOT NULL,
  `Cliente_NIF` DOUBLE NOT NULL,
  `Voo_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Bilhete_Cliente_idx` (`Cliente_NIF` ASC) VISIBLE,
  INDEX `fk_Bilhete_Voo1_idx` (`Voo_id` ASC) VISIBLE,
  CONSTRAINT `fk_Bilhete_Cliente`
    FOREIGN KEY (`Cliente_NIF`)
    REFERENCES `IngressosDeAvioes`.`Cliente` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Bilhete_Voo1`
    FOREIGN KEY (`Voo_id`)
    REFERENCES `IngressosDeAvioes`.`Voo` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 4 - Código SQL correspondente à criação da tabela Bilhete

Relação Voo

Domínio **ID**: inteiro
Domínio **Número de Voo**: string de comprimento variável, comprimento 20
Domínio **Hora Partida**: temporal
Domínio **Hora Chegada**: temporal
Domínio **Duração**: temporal
Domínio **Data Partida**: temporal
Domínio **Origem_id**: inteiro
Domínio **Avião_id**: inteiro
Domínio **Destino_id**: inteiro

Viagem(

id	ID Voo	NOT NULL,
hora_de_partida	Hora de Partida	NOT NULL,
hora_de_chegada	Hora de Chegada	NOT NULL,
duração	Duração	NOT NULL,
data_de_partida	Data de Partida	NOT NULL,
Origem_id	Aeroporto Origem	NOT NULL,
Aviao_id	Avião	NOT NULL,
Destino_id	Aeroporto Destino	NOT NULL,

CHAVE PRIMÁRIA (id),
CHAVE ESTRANGEIRA (aviao_id) REFERENCIA Avião(id)
ON UPDATE CASCADE
ON DELETE NO ACTION,
CHAVE ESTRANGEIRA (Origem_id) REFERÊNCIA Aeroporto(id)
ON UPDATE CASCADE
ON DELETE NO ACTION,
CHAVE ESTRANGEIRA (Destino_id) REFERÊNCIA Aeroporto(id)
ON UPDATE CASCADE
ON DELETE NO ACTION);

Transpondo para código SQL:

```
CREATE TABLE IF NOT EXISTS `IngressosDeAvioes`.`Voo` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `Numero_de_Voo` VARCHAR(20) NOT NULL,
  `Hora_de_partida` TIME NOT NULL,
  `Hora_de_chegada` TIME NOT NULL,
  `Duraçao` TIME NOT NULL,
  `Data_de_partida` DATE NOT NULL,
  `Numero_de_bilhetes_Vendidos` INT NOT NULL,
  `Origem_id` INT NOT NULL,
  `Aviao_id` INT NOT NULL,
  `Destino_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Voo_Aeroporto1_idx` (`Origem_id` ASC) VISIBLE,
  INDEX `fk_Voo_Aviao1_idx` (`Aviao_id` ASC) VISIBLE,
  INDEX `fk_Voo_Aeroporto2_idx` (`Destino_id` ASC) VISIBLE,
  CONSTRAINT `fk_Voo_Aeroporto1`
    FOREIGN KEY (`Origem_id`)
    REFERENCES `IngressosDeAvioes`.`Aeroporto` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Voo_Aviao1`
    FOREIGN KEY (`Aviao_id`)
    REFERENCES `IngressosDeAvioes`.`Aviao` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Voo_Aeroporto2`
    FOREIGN KEY (`Destino_id`)
    REFERENCES `IngressosDeAvioes`.`Aeroporto` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 5 - Código SQL correspondente à criação da tabela Voo

Relação Aeroporto

Domínio **ID**: inteiro
Domínio **Pais**: string de comprimento variável, comprimento 45
Domínio **Localidade**: string de comprimento variável, comprimento 45
Domínio **Código Postal**: string de comprimento variável, comprimento 20
Domínio **Nome**: string de comprimento variável, comprimento 100

Aeroporto(

id	ID Aeroporto	NOT NULL,
país	País	NOT NULL,
Localidade	Localidade	NOT NULL
Código Postal	Código Postal	NOT NULL
nome	Nome	NOT NULL,

CHAVE PRIMÁRIA (id)
CHAVE ALTERNATIVA(nome);
CHAVE ALTERNATIVA(Localidade);

Transpondo para código SQL, temos:

```
-- -----------------------------------------------------
-- Table `IngressosDeAvioes`.`Aeroporto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `IngressosDeAvioes`.`Aeroporto` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `Pais` VARCHAR(45) NOT NULL,
  `Localidade` VARCHAR(45) NOT NULL,
  `Codigo_Postal` VARCHAR(20) NOT NULL,
  `Nome` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

Figura 6 - Código SQL correspondente à criação da tabela Aeroporto

Relação Avião

Domínio **ID**: inteiro
Domínio **Nome**: string de comprimento variável, comprimento 45
Dominio **Numero_maximo_passageiros_classe_economica**: inteiro
Domínio **Numero_maximo_passageiros_classe_executiva**: inteiro
Domínio **Numero_maximo_passageiros**: inteiro
Dominio **Tara** : decimal(8,2)
Domínio **Companhia** : string de comprimento variável, comprimento 45
Avião(

id	ID	NOT NULL,
nome	Nome	NOT NULL,
nªmaximo eco	..	NOT NULL,
nªmaximo exec	..	NOT NULL,
nªmaximo	..	NOT NULL,
tara	Peso do avião carregado	NOT NULL,
companhia	Companhia Aérea	NOT NULL,

CHAVE PRIMÁRIA (id));

Transpondo para código SQL, temos:

```
-- -----------------------------------------------------
-- Table `IngressosDeAvioes`.`Aviao`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `IngressosDeAvioes`.`Aviao` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(45) NOT NULL,
  `Numero_maximo_de_passageiros_classe_economica` INT NOT NULL,
  `Numero_maximo_de_passageiros_classe_executiva` INT NOT NULL,
  `Numero_maximo_de_passageiros` INT NOT NULL,
  `Tara` DECIMAL(8,2) NOT NULL,
  `Companhia` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

Figura 7 - Código SQL correspondente à criação da tabela Avião

Relação Lugar

Domínio **ID**: inteiro
Domínio **Classe**: string de comprimento variável, comprimento 45
Domínio **Numero**: string de comprimento variável, comprimento 3
Domínio **Avião_id**: inteiro

Lugar(

id	ID	NOT NULL;
classe	Classe	NOT NULL,
numero	Número	NOT NULL,
Avião_id	ID Avião	NOT NULL,

CHAVE PRIMÁRIA (id),
CHAVE ESTRANGEIRA (avião_id) REFERENCIA Avião(id)
ON UPDATE CASCADE
ON DELETE NO ACTION);

Transpondo para código SQL, temos:

```
-- -----------------------------------------------------
-- Table `IngressosDeAvioes`.`Lugar`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `IngressosDeAvioes`.`Lugar` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `Classe` VARCHAR(45) NOT NULL,
  `Numero` VARCHAR(3) NOT NULL,
  `Aviao_id` INT NOT NULL,
  PRIMARY KEY (`id`, `Aviao_id`),
  INDEX `fk_Lugar_Aviao1_idx` (`Aviao_id` ASC) VISIBLE,
  CONSTRAINT `fk_Lugar_Aviao1`
    FOREIGN KEY (`Aviao_id`)
    REFERENCES `IngressosDeAvioes`.`Aviao` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 8 - Código SQL correspondente à criação da tabela Lugar

5.3. Tradução das interrogações do utilizador para SQL

De seguida passaremos a apresentar alguns exemplos de código SQL que permitem obter resposta às questões que o utilizador colocou, expressas nos requisitos de exploração.

- **Consultar a lista de lugares livres para um voo.**

```
CREATE PROCEDURE lugaresLivres(IN id_voo INT)
BEGIN
    SELECT Lugar.Numero,Lugar.Classe FROM Lugar
    LEFT JOIN Aviao ON Lugar.Aviao_id = Aviao.id
    LEFT JOIN Voo ON Voo.Aviao_id=Aviao.id
    WHERE Voo.id=id_voo AND Lugar.Numero NOT IN (SELECT Bilhete.Numero FROM Bilhete WHERE Bilhete.voo_id=id_voo);
END //
```

Figura 9 - Código SQL para calcular os lugares livres para um voo

- **Consultar o montante gasto num dado período.**

```
DELIMITER $$
CREATE PROCEDURE montanteGasto
    (IN user DOUBLE, dia1 DATE, dia2 DATE)
BEGIN
SELECT SUM(b.Preço) FROM Bilhete b
WHERE b.Cliente_NIF = user AND
      dia1 <= b.Data AND
      dia2 >= b.Data;
END$$
```

Figura 10 - Código SQL para calcular o montante gasto num determinado período de tempo

- **Consultar os voos disponíveis num determinado aeroporto.**

```
DELIMITER //

CREATE PROCEDURE voosDeUmAeroporto(IN id_aep INT)
BEGIN
    SELECT * FROM Voo v WHERE v.origem_id = id_aep;
END //
```

Figura 11 - Código SQL que verifica quais os voos disponíveis num determinado aeroporto

- **Consultar os voos realizados por uma determinada companhia num dado período.**

```
DELIMITER //

CREATE PROCEDURE voosFeitosPorUmaCompanhiaNumPeríodo
    ( IN comp VARCHAR(45), IN dt_i DATE, IN dt_f DATE)
BEGIN
    SELECT v.id AS ID_VOO FROM Voo AS v INNER JOIN Aviao AS a
        on v.Aviao_id = a.id
            WHERE a.Companhia = comp
            AND v.Data_de_partida >= dt_i
            AND v.Data_de_partida <= dt_f;
END //voosFeitosPorUmaCompanhiaNumPeríodo
```

Figura 12 - Código SQL que determina os voos realizados por uma companhia num dado período

- **Saber quais os passageiros que estiveram presentes num voo.**

```
DELIMITER //

CREATE PROCEDURE passageirosNumVoo(IN id_voo INT)
BEGIN
    SELECT c.Nome,c.NIF,b.Cliente_NIF FROM Bilhete b, Cliente c
    WHERE b.Voo_id = id_voo AND b.Cliente_NIF = c.NIF;
END //
```

Figura 13 - Código SQL que indica os passageiros presentes num determinado voo

- **Verificar quantos bilhetes foram vendidos num dado período.**

```
DELIMITER //

CREATE FUNCTION bilhetesVendidosNumPeriodo(dt_inicio DATE,dt_fim DATE)
    RETURNS INT DETERMINISTIC
BEGIN
    RETURN (SELECT sum(v.Numero_de_bilhetes_Vendidos) AS Total_Bilhetes FROM Voo v
    WHERE v.Data_de_partida >= dt_inicio AND v.Data_de_partida <= dt_fim);
END //
```

Figura 14 - Código SQL que verifica quantos bilhetes foram vendidos num dado período

- **Calcular o valor total faturado num dado período.**

```
DELIMITER //

CREATE FUNCTION totalFaturadoNumPeríodo(dt_i DATE, dt_f DATE)
    RETURNS FLOAT DETERMINISTIC
BEGIN
    RETURN(SELECT sum(b.Preço) FROM Bilhete b
        WHERE b.Data >= dt_i AND b.Data <= dt_f);
END //

SELECT totalFaturadoNumPeríodo("2015-01-01","2015-12-31");
```

Figura 15 - Código SQL que calcula o valor faturado num dado período

5.4. Escolha, definição e caracterização de índices em SQL

De forma a minimizar o tempo de resposta às queries feitas pelo cliente, decidimos utilizar índices, que são estruturas de dados que nos irão permitir aceder a um ficheiro/arquivo mais facilmente.
Como a tabela que mais é acedida e procurada nas queries do cliente é a dos voos, decidimos que, como a data de partida de um voo é muitas vezes necessária para a conclusão de uma query, esta coluna deveria ser acedida com mais facilidade. Logo, concedemos-lhe o seguinte índice:

```
CREATE INDEX idx_data_de_partida ON Voo(Data_de_partida);
```

Figura 16 - Criação do índice sobre a coluna Data de partida

Utilizando a mesma lógica que utilizamos para a tabela dos voos, notamos que na tabela dos bilhetes a data de aquisição era uma coluna frequentemente procurada pelo cliente e , dessa forma, decidimos conceder-lhe um índice:

```
CREATE INDEX idx_data_aquisicao ON Bilhete(Data);
```

Figura 17 - Criação do índice sobre a coluna Data

5.5. Estimativa do espaço em disco da base de dados e taxa de crescimento anual

Passaremos a apresentar uma estimativa do espaço que a base de dados em estudo ocupará no disco. Para prever qualquer cenário possível, a análise feita ao crescimento da base de dados teve por base uma análise ao seu pior caso, ou seja, assumimos que todos os voos realizados estão esgotados.

Para tornar mais fácil calcular a estimativa pretendida, determinamos inicialmente quanto é que cada tipo de entidade ocuparia em disco. Desta forma obtivemos as seguintes tabelas:

Tabela 8 - Tamanho de cada entrada na tabela do Cliente e do Bilhete

Relação	Atributo	Data Type	Tamanho	Total
Cliente	Nome	VARCHAR(45)	46 Bytes	287 Bytes
	Idade	INT	4 Bytes	
	Telemovel	VARCHAR(20)	20 Bytes	
	Email	VARCHAR(45)	46 Bytes	
	Morada	VARCHAR(100)	101 Bytes	
	Nacionalidade	VARCHAR(15)	16 Bytes	
	NIF	DOUBLE	8 Bytes	
	Password	VARCHAR(45)	46 Bytes	
Bilhete	id	INT	4 Bytes	63 Bytes
	Data	DATE	3 Bytes	
	Gate	VARCHAR(3)	4 Bytes	
	Numero	VARCHAR(3)	4 Bytes	
	Classe	VARCHAR(30)	31 Bytes	
	Preço	DECIMAL(6,2)	5 Bytes	
	Cliente_NIF	DOUBLE	8 Bytes	
	Voo_id	INT	4 Bytes	

Tabela 9 - Tamanho de cada entrada na tabela do Voo e do Aeroporto

Relação	Atributo	Data Type	Tamanho	Total
Voo	id	INT	4 Bytes	26 Bytes
	hora_de_partida	TIME	3 Bytes	
	hora_de_chegada	TIME	3 Bytes	
	duração	TIME	3 Bytes	
	data_de_partida	DATE	3 Bytes	
	origem_id	INT	4 Bytes	
	aviao_id	INT	4 Bytes	
	destino_id	INT	4 Bytes	
Aeroporto	id	INT	4 Bytes	218 Bytes
	Pais	VARCHAR(45)	46 Bytes	
	Localidade	VARCHAR(45)	46 Bytes	
	Codigo_Postal	VARCHAR(20)	21 Bytes	
	Nome	VARCHAR(100)	101 Bytes	

Tabela 10 - Tamanho de cada entrada na tabela do Avião e do Lugar

Relação	Atributo	Data Type	Tamanho	Total
Avião	ID	INT	4 Bytes	113 Bytes
	Nome	VARCHAR(45)	46 Bytes	
	Numero_maximo_passageiros_classe_economica	INT	4 Bytes	
	Numero_maximo_passageiros_classe_executiva	INT	4 Bytes	
	Numero_maximo_passageiros	INT	4 Bytes	
	Tara	DECIMAL(8,2)	5 Bytes	
	Companhia	VARCHAR(45)	46 Bytes	
Lugar	id	INT	4 Bytes	58 Bytes
	Classe	VARCHAR(45)	46 Bytes	
	Numero	VARCHAR(3)	4 Bytes	
	Aviao_id	INT	4 Bytes	

Apesar da entidade Cliente ser a que requer maior quantidade de memória, não será a que ocupará mais espaço, uma vez que a quantidade mais significativa de dados será gerada pelo Bilhete, uma vez que cada lugar num voo estará associado a um bilhete.

Tendo o Anexo 1 como base, podemos verificar que existem cerca de 38 voos diários, ocupando portanto 38*26 = **988 bytes/dia**. Dado que neste momento existem 4 aviões registados na base de dados com um máximo de 300 lugares, o espaço necessário para representar todos os lugares existentes é (300*58)*4 = **69600 bytes**. Assim, a cada dia serão adicionadas 38*300 = **11400** entradas relativas à tabela Bilhete, resultando num total de 11400*63 = **718 200 bytes/dia**.

Somando tudo, e prevendo a circulação de 13 112 453 passageiros, baseado no número do ano de 2019 de passageiros que circularam pelo Aeroporto do Porto, teremos então 13 112 453 * 287 + 718 200 *365 + 988*365 + 69600 = **3839,25 MBytes/ano**.

De notar que a VAM não se encontraria somente responsável pelos voos no aeroporto do Porto, como de muitos outros, por isso será de esperar que o espaço ocupado em disco seja ainda maior.

5.6. Definição e caracterização das vistas de utilização em SQL (alguns exemplos)

Ainda que para o utilizador pareça só mais uma tabela como as outras, uma *view* é uma tabela dinâmica, resultante de várias operações entre diversas tabelas ou mesmo outras *views*, o que providencia diversas vantagens. Como tal decidimos tirar proveito deste fator, apresentando alguns exemplos.

```
CREATE VIEW VoosDisponiveis AS
SELECT Numero_de_Voo AS "Número de Voo",Hora_de_partida AS "Hora de Partida"
,Hora_de_chegada AS "Hora de Chegada",
Duraçao,
Data_de_partida AS "Data de Partida",
AO.nome AS "Aeroporto Origem" ,
AD.nome AS "Aeroporto Destino"
FROM Voo
INNER JOIN Aeroporto AS AO ON (AO.id=Voo.Origem_id )
INNER JOIN Aeroporto AS AD ON (AD.id=Voo.Destino_id )
WHERE (Current_date()=Voo.Data_de_partida AND Current_time()<Voo.Hora_de_Partida) OR (Current_date()<Voo.Data_de_partida)
```

Figura 18 - View que mostra a informação completa de todos os voos disponíveis

```
CREATE VIEW TOP3BilhetesMaisBaratos AS
SELECT Bilhete.Classe,Bilhete.Preço, AO.nome AS "Aeroporto Orgiem", AD.nome AS "Aeroporto Destino"
FROM Bilhete
INNER JOIN Voo ON Bilhete.Voo_id=Voo.id
INNER JOIN Aeroporto AS AO ON (AO.id=Voo.Origem_id )
INNER JOIN Aeroporto AS AD ON (AD.id=Voo.Destino_id )
ORDER BY Bilhete.Preço ASC
LIMIT 3;
```

Figura 19 - View que mostra a informação completa sobre os os top 3 bilhetes mais baratos

5.7. Revisão do sistema implementado

Dada por concluída a última etapa da construção do sistema pretendido pela VAM, Voar Além Mar, e dados os pontos acima enunciados, revelámos ser capazes de cumprir com todos os requisitos, respondendo às perguntas dos utilizadores.

Assim, considerando que cumpre os requisitos apresentados, concluímos que o modelo implementado é viável, apesar de ocupar espaço desnecessário no disco, algo que pode ser facilmente resolvido no futuro.

6. Conclusões e Trabalho Futuro

No desenvolvimento do SBD foi visível o aumento de produtividade e a maior facilidade na construção deste graças ao facto de termos encarado e aprofundado com o cuidado necessário cada uma das fases, sendo que, embora a criação de um SBD seja bastante iterativa, ter esta metodologia de trabalho faseada tornou-se uma grande mais valia observável na diminuição de erros afetando, assim, positivamente a evolução do mesmo. Com o faseamento do projeto foi possível aceitarmos os pilares do sistema sendo estes, principalmente, a definição do problema e o modelo conceptual. Relativamente à definição do problema foi talvez o ponto mais importante para o posterior desenvolvimento do SBD, sendo que quanto melhor conhecermos o funcionamento e o *modus operandi* da empresa, mais facilitada e fiel será a concessão e reprodução do sistema, sendo que melhor serão idealizados os requisitos levantados, elevando assim a qualidade dos mesmos e melhor será a possível resposta dada às interrogações dos possíveis utilizadores.

Deste estudo detalhado da definição do problema surge então um forte modelo conceptual capaz de traduzir bem o problema. Com o modelo conceptual bem definido nestas duas primeiras fases deriva o modelo lógico de onde posteriormente deriva o modelo físico. Para obter este modelo conceptual forte, tivemos de repetir várias vezes a elaboração deste, sendo que com o decorrer da reflexão sobre o mesmo acabávamos por nos aperceber de algo em falta e mesmo ter de analisar novamente os objetivos esperados do SBD.

Como futuro trabalho, poderíamos procurar diminuir um pouco o espaço utilizado pelo sistema base de dados, uma vez que tendo o mesmo já implementado apercebemo-nos que talvez tenhamos sido um pouco negligentes com a questão da memória ocupada por alguns

parâmetros. Poderíamos também aprofundar ainda mais as funcionalidades do sistema, sendo assim possível dar mais informação aos administradores.

Referências

Lista de Siglas e Acrónimos

BD	Base de Dados
NIF	Número de identificação fiscal
VAM	Voar Além Mar
SBD	Sistema Base de Dados
SGBD	Sistema de Gestão de Base de Dados
...	...

Anexos

I. Lista de Voos

Dia	Hora	Terminal	N° do Voo	Destino	Companhia aérea	Estado
04/12/2020	07:00	T1	FR 3048	Krakow	RYANAIR	Partiu 06:57
04/12/2020	07:00	T1	FR 9135	Paris, Beauvais	RYANAIR	Partiu 07:10
04/12/2020	07:00	T1	TP 1931	Lisboa	TAP PORTUGAL	Partiu 07:04
04/12/2020	07:15	T1	EC 7585	Madeira	EASYJET EUROPE	Partiu 07:23
04/12/2020	08:30	T1	FR 1537	Frankfurt	RYANAIR	Partiu 09:06
04/12/2020	08:30	T1	TP 1713	Madeira	TAP PORTUGAL	Partiu 08:38
04/12/2020	08:35	T1	FR 1387	Brussels, Charleroi	RYANAIR	Partiu 08:48
04/12/2020	10:05	T1	FR 7472	Eindhoven	RYANAIR	Partiu 10:08
04/12/2020	11:20	T1	TP 1941	Lisboa	TAP PORTUGAL	Partiu 11:24
04/12/2020	11:35	T1	EC 1458	Geneva	EASYJET EUROPE	Partiu 11:49
04/12/2020	11:40	T1	KL 1712 +2	Amsterdam	KLM	Partiu 11:45
04/12/2020	11:40	T1	LX 2065	Zurich	SWISS INTERNATIONAL	Partiu 12:31
04/12/2020	12:10	T1	FR 6532	Marseille	RYANAIR	Partiu 12:21
04/12/2020	12:25	T1	LH 1177	Frankfurt	LUFTHANSA	Partiu 12:39
04/12/2020	12:35	T1	TO 3409	Paris, Orly	TRANSAVIA FRANCE	Partiu 15:16
04/12/2020	13:00	T1	FR 4564	Paris, Vatry	RYANAIR	Partiu 13:00
04/12/2020	14:15	T1	HV 6002	Amsterdam	TRANSAVIA	Partiu 14:16
04/12/2020	14:50	T1	TO 3451	Paris, Orly	TRANSAVIA FRANCE	Previsto 16:20
04/12/2020	15:45	T1	FR 8348	London, Stansted	RYANAIR	Partiu 16:36
04/12/2020	15:55	T1	FR 1471	Ponta Delgada	RYANAIR	Partiu 15:52
04/12/2020	16:30	T1	EC 7583	Luxembourg	EASYJET EUROPE	Partiu 16:34
04/12/2020	16:35	T1	FR 2862	Luxembourg	RYANAIR	Closed
04/12/2020	16:40	T1	IB 8665	Madrid	IBERIA	Boarding
04/12/2020	17:25	T1	FR 5478	Karlsruhe	RYANAIR	
04/12/2020	17:30	T1	FR 6037	Terceira	RYANAIR	

Dia	Hora	Terminal	N° do Voo	Destino	Companhia aérea	Estado
04/12/2020	17:50	T1	**EC 1460**	Geneva	EASYJET EUROPE	
04/12/2020	17:50	T1	**FR 2929**	Brussels	RYANAIR	
04/12/2020	18:40	T1	**FR 7474**	Paris, Beauvais	RYANAIR	
04/12/2020	18:50	T1	**TP 1370**	London, Heathrow	TAP PORTUGAL	
04/12/2020	19:15	T1	**LG 3768**	Luxembourg	LUXAIR	
04/12/2020	19:20	T1	**DS 1134**	Basel	EASYJET SWITZERLAND	
04/12/2020	19:50	T1	**UX 1142**	Madrid	AIR EUROPA	
04/12/2020	20:55	T1	**TP 1955**	Lisboa	TAP PORTUGAL	
04/12/2020	21:40	T1	**FR 5486**	Faro	RYANAIR	
04/12/2020	22:10	T1	**S4 175**	Ponta Delgada	AZORES AIRLINES	

Anexo 1 - Lista de voos diários a partir do Porto