



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

Unidade Curricular de Bases de Dados

Ano Lectivo de 2016/2017

Base de dados de Reserva de viagens

André Rodrigues Freitas (A74619)

Cesário Miguel Pereira Perneta (A73883)

João Miguel Freitas Palmeira (A73864)

Sofia Manuela Gomes de Carvalho (A76658)

Novembro, 2016

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Base de dados de Reserva de viagens

André Rodrigues Freitas (A74619)

Cesário Miguel Pereira Perneta (A73883)

João Miguel Freitas Palmeira (A73864)

Sofia Manuela Gomes de Carvalho (A76658)

Novembro, 2016

Resumo

O projeto apresentado insere-se na unidade curricular de Bases de Dados e tem como principal objetivo a realização e futura implementação de um sistema de base de dados, capaz de gerir a informação acerca de uma companhia ferroviária.

O desenvolvimento foi feito desde o levantamento de requisitos, passando pela modulação conceptual e lógica do problema até ao desenvolvimento físico.

As várias secções deste documento apresentam o que foi descrito anteriormente de forma detalhada.

Área de Aplicação: Desenho e arquitetura de Sistemas de Bases de Dados.

Palavras-Chave: Base de Dados, MySQL, Comboios Lusitanos, Cliente, Reserva, *views, procedures*.

Índice

1. Introdução	1
1.1. Contextualização	1
1.2. Apresentação do Caso de Estudo	1
1.3. Motivação e Objetivos	2
1.3.1 Motivação	2
1.3.2 Objetivos	2
1.4. Estrutura do Relatório	2
2. Requisitos	4
3. Caracterização dos Perfis de Utilização	5
3.1. Administrador	5
3.2. Cliente	5
4. Modelo Conceptual	6
4.1. Descrição dos tipos de Entidades	7
4.2. Descrição dos tipos de Atributos	8
4.3. Descrição dos tipos de Relacionamentos	12
4.4. Validação do modelo através de transações do Utilizador	14
4.4.1 Transações	14
5. Modelo Lógico	15
5.1. Passagem do Modelo Conceptual para o Modelo Lógico	16
5.1.1 Entidades e Atributos	16
5.1.2 Atributos Multi-Valor	16
5.1.3 Atributos Compostos	16
5.1.4 Relacionamentos 1:N (binários)	16
5.1.5 Relacionamentos 1:1	16
5.1.6 Relacionamentos Complexos	17
5.1.7 Tabelas	17
5.2. Validação do modelo através da normalização	19
5.2.1 Primeira Forma Normal (1FN)	19
5.2.2 Segunda Forma Normal (2FN)	21
5.2.3 Terceira Forma Normal (3FN)	21
5.3. Validação do modelo através das transações do utilizador	22
5.3.1 Inserir Cliente	22
5.3.2 Inserir Reserva	23
5.3.3 Inserir Bilhete	24
5.3.4 Consultar Viagem	25
5.4. Restrições de Integridade	26

5.5. Modelo Lógico e Utilizador	28
6. Modelo Físico	29
6.1. Tradução e implementação do modelo lógico no SGBD	29
6.1.1 Criação das Tabelas e dos Relacionamentos	30
6.2. Povoação da Base de Dados	37
6.3. Transações	45
6.3.1 Inserir novo cliente	45
6.3.2 Alterar nome e password	45
6.3.3 Inserir um novo endereço num cliente	46
6.3.4 Inserir nova reserva	46
6.3.5 Inserir novo comboio	47
6.3.6 Inserir nova viagem	48
6.4. Organização de Ficheiros	48
6.5. Índices	48
6.6. Vistas de Utilizadores	48
6.6.1 Vistas de Administradores	49
6.6.2 Vistas de Clientes	51
6.7. Criação de Utilizador	54
6.8. Estimativa do espaço ocupado em disco pela base de dados	54
6.8.1 Tamanho ocupado pelos atributos da tabela Cliente	54
6.8.2 Tamanho ocupado pelos atributos da tabela Endereço	55
6.8.3 Tamanho ocupado pelos atributos da tabela Reserva	56
6.8.4 Tamanho ocupado pelos atributos da tabela Bilhete	56
6.8.5 Tamanho ocupado pelos atributos da tabela Viagem	57
6.8.6 Tamanho ocupado pelos atributos da tabela Comboio	57
6.8.7 Tamanho ocupado pelos atributos da tabela Lugar	58
7. Conclusão	59

Anexos

I. Procedure 1	63
II. Procedure 2	64
III. Procedure 3	65
IV. Procedure 4	66
V. Procedure 5	67
VI. Procedure 6	68
VII. Procedure 7	69
VIII. Procedure 8	70
IX. Procedure 9	71
X. Procedure 10	72
XI. Procedure 11	73

XII. Procedure 12	74
XIII. Procedure 13	75
XIV. Procedure 14	76
XV. Procedure 15	77
XVI. Procedure 16	78
XVII. Procedure 17	79

Índice de Figuras

Figura 1 – Modelo Conceptual dos Comboios Lusitanos.	6
Figura 2 – Relacionamento entre cliente e reserva.	12
Figura 3 – Relacionamento entre reserva e bilhete.	12
Figura 4 – Relacionamento entre bilhete e viagem.	13
Figura 5 – Relacionamento entre viagem e comboio.	13
Figura 6 – Modelo Lógico.	15
Figura 7 – Cliente.	17
Figura 8 – Reserva.	17
Figura 9 – Bilhete.	18
Figura 10 – Viagem.	18
Figura 11 – Comboio.	18
Figura 12 – Endereço (1FN).	19
Figura 13 – Lugar (1FN).	20
Figura 14 – Cliente (1FN).	20
Figura 15 – Bilhete (1FN).	20
Figura 16 – Inserir Cliente.	22
Figura 17 – Inserir Reserva.	23
Figura 18 – Inserir Bilhete.	24
Figura 19 – Inserir Viagem.	25
Figura 20 – Tabela Cliente.	30
Figura 21 – Tabela Endereço.	31
Figura 22 – Tabela Reserva.	32
Figura 23 – Tabela Bilhete.	33
Figura 24 – Tabela Viagem.	34
Figura 25 – Tabela Comboio.	35
Figura 26 – Tabela Lugar.	36
Figura 27 – Cliente (Povoamento).	37
Figura 28 – Endereço (Povoamento).	38
Figura 29 – Reserva (Povoamento).	39
Figura 30 – Bilhete (Povoamento).	40
Figura 31 – Viagem (Povoamento).	41
Figura 32 – Comboio (Povoamento).	42
Figura 33 – Lugar (Povoamento).	44
Figura 34 – Transação 1.	45
Figura 35 – Transação 2.	45

Figura 36 – Transação 3.	46
Figura 37 – Transação 4.	46
Figura 38 – Transação 5.	47
Figura 39 – Transação 6.	48
Figura 40 – View 1 (Admin).	49
Figura 41 – View 2 (Admin).	49
Figura 42 – View 3 (Admin).	49
Figura 43 – View 4 (Admin).	49
Figura 44 – View 5 (Admin).	49
Figura 45 – View 6 (Admin).	49
Figura 46 – View 7 (Admin).	49
Figura 47 – View 8 (Admin).	50
Figura 48 – View 9 (Admin).	50
Figura 49 – View 10 (Admin).	50
Figura 50 – View 11 (Admin).	50
Figura 51 – View 12 (Admin).	50
Figura 52 – View 13 (Admin).	50
Figura 53 – View 14 (Admin).	51
Figura 54 – View 15 (Admin).	51
Figura 55 – View 16 (Admin).	51
Figura 56 – View 17 (Admin).	51
Figura 57 – View 1 (Cliente).	51
Figura 58 – View 2 (Cliente).	52
Figura 59 – View 3 (Cliente).	52
Figura 60 – View 4 (Cliente).	52
Figura 61 – View 5 (Cliente).	52
Figura 62 – View 6 (Cliente).	52
Figura 63 – View 7 (Cliente).	53
Figura 64 – View 8 (Cliente).	53
Figura 65 – Administrador.	54
Figura 66 – UtilizadoresDeUmDadosSexo.	63
Figura 67 – UtilizadoresDeUmDadosSexoComRestriçãoDeData.	64
Figura 68 – DataPartidaCustoComDescontoAtravésBilhetesDados.	65
Figura 69 – SaberDadosDeClienteDeCertaCidade.	66
Figura 70 – NºdeBilhetesVendidosNumIntervaloDeTempo.	67
Figura 71 – NºdosBilhetesVendidosNumIntervaloDeTempo.	68
Figura 72 – MontanteAPagarPelaReservaDeUmaDataidR.	69
Figura 73 – LucroTotalDeDadaViagem.	70

Figura 74 – NumeroReservasFeitasPorDadoNome.	71
Figura 75 – NumeroBilhetesCompradosPorDadoNome.	72
Figura 76 – ComboiosEBilhetesAssociadosAUmDadoldv.	73
Figura 77 – ComboiosEBilhetesAssociadosAUMaDadaOrigemOuDestino.	74
Figura 78 – ComboiosDeDeterminadaViagem.	75
Figura 79 – NºdeLugaresOcupadosNumaDadaViagem.	76
Figura 80 – NºdeLugaresLivresNumaDadaViagem.	77
Figura 81 – DestinosDeUmaDadaViagem.	78
Figura 82 – OrigensDeUmDadoDestino.	79

Índice de Tabelas

Tabela 1 – Descrição das Entidades.	7
Tabela 2 – Descrição das Entidades.	10
Tabela 3 – Atributos Multi-Valor.	10
Tabela 4 – Atributos Compostos.	11
Tabela 5 – Descrição dos Relacionamentos.	12
Tabela 6 – Descrição dos Relacionamentos no Modelo Lógico.	27
Tabela 7 – Espaço ocupado pelos atributos da Tabela Cliente.	55
Tabela 8 – Espaço ocupado pelos atributos da Tabela Endereço.	55
Tabela 9 – Espaço ocupado pelos atributos da Tabela Reserva.	56
Tabela 10 – Espaço ocupado pelos atributos da Tabela Bilhete.	56
Tabela 11 – Espaço ocupado pelos atributos da Tabela Viagem.	57
Tabela 12 – Espaço ocupado pelos atributos da Tabela Comboio.	57
Tabela 13 – Espaço ocupado pelos atributos da Tabela Lugar.	58

1. Introdução

1.1. Contextualização

A empresa CL Comboios Lusitanos decidiu criar um sistema de reservas para satisfazer as necessidades dos seus clientes de modo a que possam garantir o seu lugar na viagem pretendida. A ideia surgiu após alguns dos clientes terem chamado à atenção das dificuldades em comprar bilhete apenas momentos antes da partida do respetivo comboio, pois surgia a situação da lotação já estar esgotada ou de haver uma grande fila de espera nas bilheteiras. Ao lembrar alguns dias do trabalho, verificaram que tais situações aconteciam de facto muitas vezes.

1.2. Apresentação do Caso de Estudo

A criação do projeto para reservas de viagens via Web ficou assim bem sustentada, o que fez com que a empresa o achasse viável. Decidiu-se então começar a implementação deste projeto. Esta empresa de reserva de viagens em comboios nacionais e internacionais permite que os seus clientes façam várias reservas. Para o fazerem, precisam de estar autenticados através de um email e password, sendo que para se registarem devem fornecer, para além, destes dois parâmetros, o nome, endereço (localidade, código-postal e rua), sexo, data de nascimento, contactos (telefone, telemóvel) e NIF. Após o fazerem, acedem às viagens disponíveis, tanto aos seus horários como à lotação atual de determinada viagem. Ao fazerem a reserva, ficará registada no email correspondente ao utilizador e terá associada a data em que foi feita e um número identificador desta. Por exemplo, se forem reservados 5 bilhetes numa reserva, correspondem a 5 lugares diferentes. Ao escolher os bilhetes desejados, que podem ser vários por reserva, terá de ser indicada a classe pretendida e as datas de partida e chegada de cada viagem. Cada viagem tem associada a si uma origem, um destino, o horário (de partida e de chegada) e um comboio onde esta será efetuada.

Os comboios têm associados a si uma lotação: 6 lugares para a classe executiva e 9 lugares para a classe económica.

A cada reserva é atribuído um número de identificação único. Contudo, se esta for feita num qualquer dia anterior ao da viagem, o cliente usufrui de um desconto de 10% sobre o valor do bilhete.

1.3. Motivação e Objetivos

1.3.1 Motivação

Com o desafio lançado na UC, decidimos passar para a criação de uma base de dados que não se adeque apenas às necessidades do cliente, mas que forneça um serviço personalizado e otimizado para a empresa. Deparamo-nos com inúmeras possibilidades de entidades e relacionamentos, confirmando assim a complexidade de um trabalho deste género.

Para tal, utilizamos o modelo relacional, lecionado na disciplina de Bases de Dados, pois é uma metodologia que permite uma resposta rápida aos pedidos de informação e melhor gestão de informação, facilitando a estruturação da mesma.

1.3.2 Objetivos

Os objetivos deste trabalho serão principalmente fundamentar os conhecimentos adquiridos na UC de Base de Dados, aplicando-os num contexto de uma situação real.

Para modelar a base de dados, é necessário haver uma preparação e planificação sobre informações dos principais componentes de uma companhia ferroviária.

Com o desenvolvimento desta Base de Dados para a gestão de reservas de viagens, quer nacionais quer internacionais de comboios, pretendemos modelar um protótipo de um sistema que permita visualizar diversas informações relativamente a uma companhia ferroviária. Esta empresa será ficcional não sendo assim utilizados nenhum dados reais.

1.4. Estrutura do Relatório

Neste relatório é descrita detalhadamente a conceção do trabalho prático, ou seja, a modelação conceptual, lógica e física da base de dados.

Utilizaremos a metodologia do livro *Database Systems – A Practical Approach to Design, Implementation and Management*, 4^a Edição (Conolly e Begg, 2005) para justificar e validar todos os modelos enunciados anteriormente.

Numa primeira fase é descrito o problema em causa através da sua contextualização e apresentação do caso de estudo.

Na segunda iremos apresentar os requisitos para o trabalho.

Numa terceira parte, as possíveis caracterizações dos possíveis perfis do administrador da plataforma e do respetivo utilizador.

Na quarta, apresentamos o modelo conceptual detalhadamente seguindo as fases enunciadas na metodologia.

Na quinta fase, traduzimos o esquema conceptual num modelo lógico, apresentando o seu aperfeiçoamento até alcançar um estado que permita a sua validação. Apresentamos, também, as possíveis transações feitas da parte do utilizador.

Na parte seis, é desenvolvida a explicação do modelo físico, sendo apresentada a tradução do modelo lógico e a implementação do modelo físico da base de dados.

Por fim, apresentamos a conclusão e alguns anexos.

2. Requisitos

Neste tópico, é necessário abordar a base de dados na perspetiva de utilizador. Para isso, todos os elementos do grupo se reuniram, trocaram ideias e relataram experiências em plataformas semelhantes de modo a contribuir com as suas perspetivas de utilizador.

Decidimos então que o funcionamento da plataforma requer uma entidade cliente que é composta por diferentes atributos. Para que o cliente se autentique na plataforma para ter acesso a todas as funcionalidades da mesma é preciso que o utilizador forneça o seu email e uma palavra-passe. Ao registar-se deve indicar os seus dados pessoais tais como o nome, o NIF, o sexo, a data de nascimento, os contactos e os seus endereços.

Este necessitará, na plataforma, de ter acesso a funcionalidades como o número total de lugares livres de um comboio para uma viagem, comboio esse que terá de ser definido por um número, por uma lotação máxima e por cada lugar disponível. Deverá ter acesso a cada horário de cada viagem ou até verificar qual o comboio de cada viagem e, mais uma vez, a viagem que será definida por um número identificativo da viagem em questão, a sua origem e destino, o horário de partida e de chegada.

Requer ainda uma entidade reserva onde o cliente tenha acesso a informações das viagens disponíveis, comboios disponíveis e lugares disponíveis em cada comboio de cada viagem através dos relacionamentos com a entidade bilhete, viagem e comboio.

Na mesma entidade, e como o cliente tem a possibilidade de participar em várias viagens, logo poderá efetuar uma ou mais reservas de viagens ou poderá reservar uma viagem para uma determinada data e num determinado lugar do comboio.

Por último, a entidade bilhete terá essencialmente como atributos o número de bilhete, o lugar do bilhete, a data de chegada e de partida, o custo normal e o custo com desconto.

3. Caracterização dos Perfis de Utilização

Através deste projeto pretendemos disponibilizar uma base de dados que contenha todos os dados acerca dos seus clientes e das suas reservas de forma explícita e intuitiva onde seja possível ser acedido pelo administrador. É possível ainda permitir ao utilizador realizar as suas reservas podendo aceder a dados relativamente a viagens e comboios.

3.1. Administrador

Tem como função gerir todos os regtos de utilizadores e de reservas que existem, através do MySQL. Este deverá criar, atualizar ou eliminar os dados que necessitem sofrer qualquer tipo de alteração na Base de Dados.

Email, NIF, Nome, Sexo, DataNascimento, Password, Telefone, Telemovel, idC, Lotacao, idV, Origem, Destino, HorarioPartida, HorarioChegada, CodigoPostal, Localidade, Rua, idR (id da Reserva), Data, Carruagem, Classe, Numero (de lugar), são todos os campos que o administrador pode e deve alterar caso seja necessário.

3.2. Cliente

Este utilizador é o principal elemento e centro objetivo do desenvolvimento desta plataforma. O cliente neste sistema poderá consultar as origens e os destinos de cada viagem, horários das mesmas, os comboios das mesmas e, mais importante, efetuar quantas reservas pretender das viagens que deseja e poderá reservar a quantidade de bilhetes desejada.

Relativamente aos bilhetes verificar o horário de um bilhete e saber quais são os lugares ocupados em cada comboio numa viagem e a sua respetiva quantidade.

Consequentemente, tem a possibilidade de saber também o número total de lugares livres num comboio, assim como quais os lugares que correspondem a esta condição.

Para além destas funcionalidades, o cliente poderá ainda alterar os seus dados pessoais.

4. Modelo Conceptual

Iremos agora apresentar o modelo conceptual do trabalho que foi concebido após uma longa análise detalhada dos requisitos do mesmo e após uma reflexão entre os elementos do grupo, chegou-se à conclusão que a melhor solução seria esta base de dados proposta:

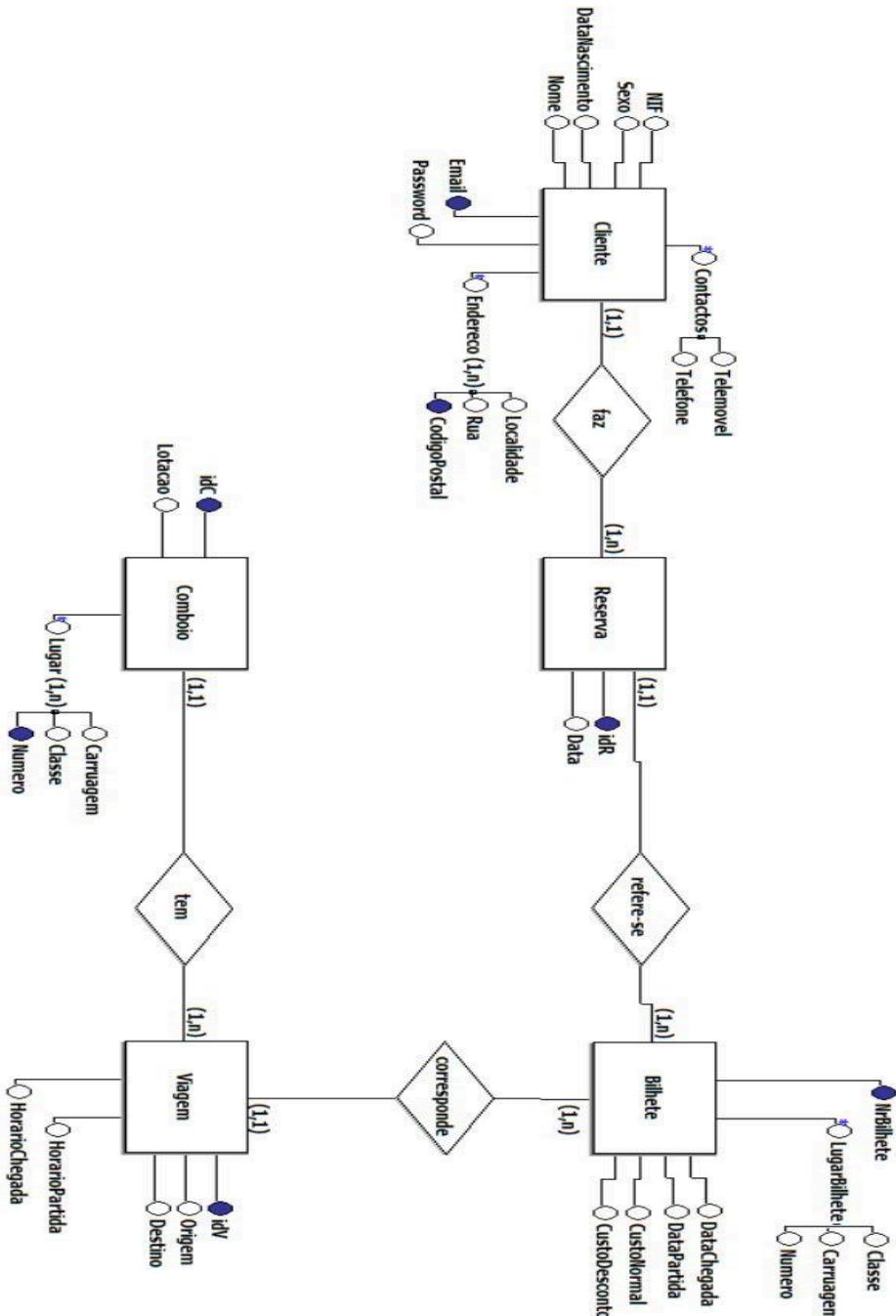


Figura 1 – Modelo Conceptual dos Comboios Lusitanos.

4.1. Descrição dos tipos de Entidades

No total existem 5 entidades compostas por diferentes atributos, interligadas por chaves estrangeiras que ligam entidades e permitem integridade referencial entre elas.

Entidades	Descrição	Ocorrências
Reserva	Entidade que contém a informação da reserva	Existem várias por cliente
Bilhete	Entidade que contém a informação do bilhete	Existem um ou mais por reserva
Cliente	Entidade que contém a informação completa do cliente	Existe uma por cliente
Viagem	Entidade que contém a informação sobre a viagem ou viagens da reserva	Existem várias por reserva
Comboio	Entidade que contém a informação do comboio para cada viagem	Existe uma por reserva

Tabela 1 – Descrição das Entidades.

4.2. Descrição dos tipos de Atributos

Este modelo é composto por diferentes tipos de atributos e nele estão contidos atributos normais, do tipo composto, do tipo multi-valor e ainda atributos que identificam a entidade (as chaves primárias).

Entidade	Atributos	Descrição	Composto	Chave Primária	Chave Estrangeira	Multi-Valor	NULL	Tipos de dados
Cliente	Email	Identifica a entidade cliente	Não	Sim	Sim	Não	Não	VARCHAR(45)
	NIF	Número de identificação fiscal do cliente	Não	Não	Não	Não	Não	INT
	Nome	Nome do cliente	Não	Não	Não	Não	Não	VARCHAR(45)
	Sexo	Sexo do cliente	Não	Não	Não	Não	Não	VARCHAR(45)
	DataNascimento	Data de nascimento do cliente	Não	Não	Não	Não	Não	DATE
	Password	Palavra-passe da conta do cliente	Não	Não	Não	Não	Não	VARCHAR(45)
	Endereco	Endereço do cliente (localidade, rua e código postal)	Sim	Não	Não	Sim	Não	INT, VARCHAR(45), VARCHAR(45)
Reserva	Contactos	Contactos do cliente (telemóvel e telefone)	Sim	Não	Não	Não	Não	INT, INT
	idR	Identifica a reserva	Não	Sim	Sim	Não	Não	INT
	Data	Data da reserva	Não	Não	Não	Não	Não	DATE

	NrBilhete	Identifica o bilhete	Não	Sim	Não	Não	Não	INT
Bilhete	DataPartida	Data de partida da viagem	Não	Não	Não	Não	Não	DATE
	DataChegada	Data de chegada da viagem	Não	Não	Não	Não	Não	DATE
	CustoNormal	Custo normal do bilhete	Não	Não	Não	Não	Não	DECIMAL(5,2)
	CustoDesconto	Custo do bilhete com desconto	Não	Não	Não	Não	Não	DECIMAL(5,2)
	LugarBilhete	Caracterização do lugar do bilhete no comboio	Sim	Não	Não	Não	Não	INT, INT, VARCHAR(45)
Viagem	idV	Identifica a viagem	Não	Sim	Sim	Não	Não	INT
	Origem	Origem da viagem	Não	Não	Não	Não	Não	VARCHAR(45)
	Destino	Destino da viagem	Não	Não	Não	Não	Não	VARCHAR(45)
	HorarioPartida	Horário de partida do comboio	Não	Não	Não	Não	Não	TIME
Comboio	HorarioChegada	Horário de chegada do comboio	Não	Não	Não	Não	Não	TIME
	idC	Identifica o comboio	Não	Sim	Sim	Não	Não	INT
	Lotacao	Lotação do comboio	Não	Não	Não	Não	Não	INT

	Lugar	Caracterização dos lugares do comboio	Sim	Não	Não	Sim	Não	INT, VARCHAR(45), INT
--	-------	---------------------------------------	-----	-----	-----	-----	-----	-----------------------------

Tabela 2 – Descrição das Entidades.

Atributos Multi-Valor:

Entidade	Atributo	Descrição	Associados	Chave Primária	Chave Estrangeira	NULL	Tipos de dados
Cliente	Endereço	Endereços do cliente	CodigoPostal	Sim	Não	Não	INT
			Localidade	Não	Não	Não	VARCHAR(45)
			Rua	Não	Não	Não	VARCHAR(45)
			Cliente_Email	Não	Sim	Não	VARCHAR(45)
Comboio	Lugar	Lugares dos comboios	Carruagem	Não	Não	Não	ENUM('1','2')
			Classe	Não	Não	Não	ENUM('económica','executiva')
			Numero	Sim	Não	Não	INT
			Comboio_idC	Sim	Sim	Não	INT

Tabela 3 – Atributos Multi-Valor.

Atributos Compostos:

Entidade	Atributo	Descrição	Associados	Chave Primária	Chave Estrangeira	NULL	Tipos de dados
Cliente	Contactos	Contactos do cliente	Telemovel	Não	Não	Não	INT
			Telefone	Não	Não	Não	INT
Bilhete	LugarBilhete	Lugar do bilhete	CarruagemB	Não	Não	Não	ENUM('1','2')
			ClasseB	Não	Não	Não	ENUM('económica','executiva')
			NumeroB	Não	Não	Não	INT

Tabela 4 – Atributos Compostos.

4.3. Descrição dos tipos de Relacionamentos

Neste modelo da nossa base de dados existem diferentes tipos de relacionamentos que são descritos na tabela seguinte.

Entidade	Multiplicidade	Relação	Multiplicidade	Entidade
Cliente	1..1	Faz	1..N	Reserva
Reserva	1..1	Refere-se	1..N	Bilhete
Bilhete	1..N	Corresponde	1..1	Viagem
Viagem	1..N	Tem	1..1	Comboio

Tabela 5 – Descrição dos Relacionamentos.

Cliente → Reserva



Figura 2 – Relacionamento entre cliente e reserva.

Reserva → Bilhete

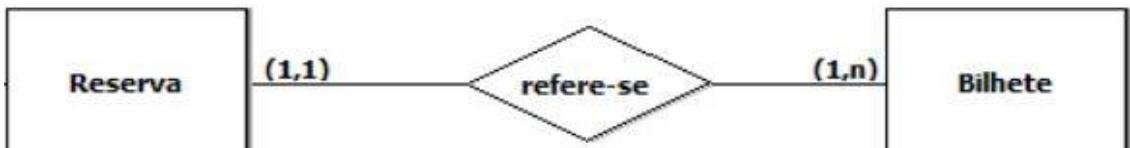


Figura 3 – Relacionamento entre reserva e bilhete.

Bilhete → Viagem



Figura 4 – Relacionamento entre bilhete e viagem.

Viagem → Comboio

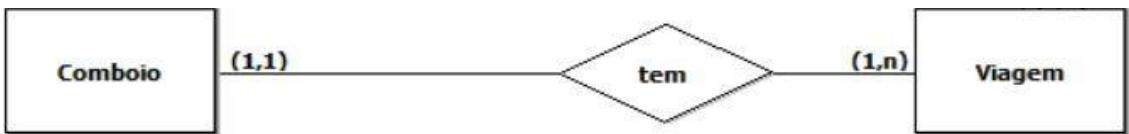


Figura 5 – Relacionamento entre viagem e comboio.

4.4. Validação do modelo através de transações do Utilizador

Realizaremos agora uma validação do esquema conceptual a partir da análise de algumas transações pedidas pelo cliente (neste caso os elementos do grupo), ao modelo. O objetivo principal nesta fase é verificar se, com este modelo conceptual, será possível efetuar essas mesmas transações.

4.4.1 Transações

Um cliente pode realizar mais do que uma reserva, ou seja, o relacionamento entre a entidade cliente e a entidade reserva terá de ser de 1:N, tal como se verifica no esquema conceptual.

Tal como o cliente, uma reserva pode referir-se a vários bilhetes o que implica uma relação entre a reserva e o bilhete de 1:N, pois o cliente pode apenas reservar um bilhete para uma determinada viagem bem como reversar vários bilhetes para a mesma viagens ou ainda vários bilhetes para diferentes viagens.

Por isso, cada bilhete vai corresponder apenas a uma viagem o que implicar um relacionamento de N:1 entre a entidade bilhete e a entidade viagem, uma vez que cada bilhete vai corresponder a uma viagem.

Em relação à última transação, é necessário que cada viagem tenha um comboio associado e várias viagens podem ser realizadas pelo mesmo comboio, logo, e como está referido no esquema conceptual, a viagem tem um relacionamento de N:1 com a entidade comboio

5. Modelo Lógico

A partir do modelo anterior, elaboramos o esquema lógico:

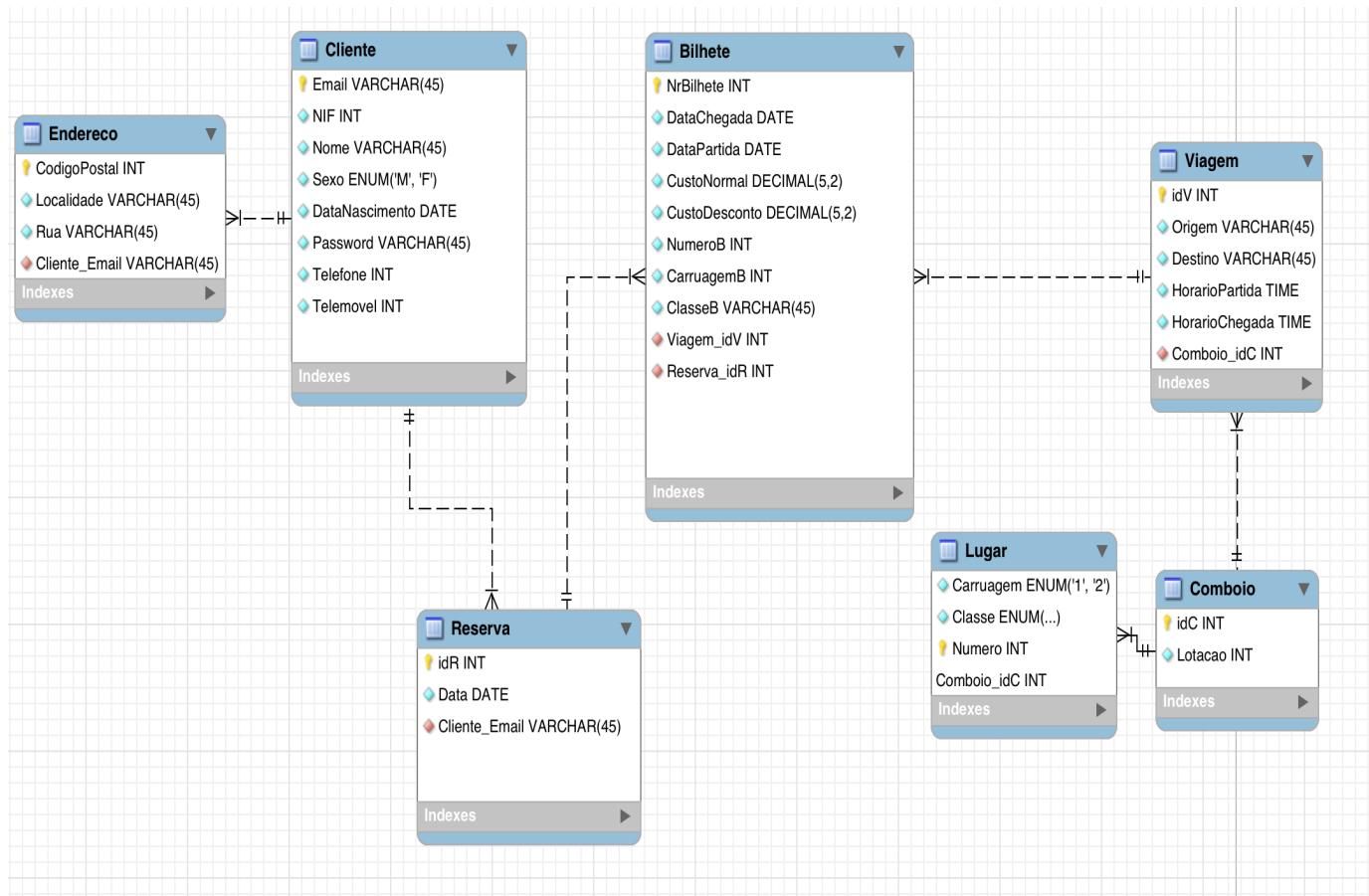


Figura 6 – Modelo Lógico.

5.1. Passagem do Modelo Conceptual para o Modelo Lógico

5.1.1 Entidades e Atributos

Em cada entidade foi criada uma relação com os atributos. Atributos estes que, tal como no Modelo Conceptual não possuem os espaços e a acentuação.

5.1.2 Atributos Multi-Valor

Existem dois atributos multi-valor e cada um origina uma relação com o mesmo nome do atributo. Em ambas as relações, existe uma chave primária e uma chave estrangeira proveniente da relação, que também é a chave primária da mesma, à qual pertencem os atributos. No atributo multi-valor Lugar a chave estrangeira também é primária.

5.1.3 Atributos Compostos

No modelo conceptual estão presentes dois atributos compostos: Contactos (na entidade Cliente) e LugarBilhete (na entidade Bilhete). Os seus atributos são colocados na respetiva tabela e não criando uma tabela extra.

5.1.4 Relacionamentos 1:N (binários)

Todos os relacionamentos deste modelo são deste tipo. Para cada relacionamento, foi colocado uma cópia da chave primária de uma das entidades envolvidas na outra entidade do par do relacionamento como sendo chave estrangeira.

5.1.5 Relacionamentos 1:1

Não existem relacionamentos deste tipo neste modelo.

5.1.6 Relacionamentos Complexos

Um relacionamento deste tipo é um relacionamento igual ou superior a três, isto é, quando três ou mais entidades são envolvidas no relacionamento. No entanto, segundo o modelo relacional (modelo onde nos baseamos para o desenvolvimento do trabalho) é necessário decompor estes casos de maneira a encontrar uma entidade intermédia e que as entidades fiquem ligadas com relacionamentos de grau dois. Portanto, não encontramos nenhum relacionamento desse tipo neste modelo.

5.1.7 Tabelas

Através destas entidades e destes relacionamentos surgem as tabelas correspondentes que contêm os diferentes tipos de informação.

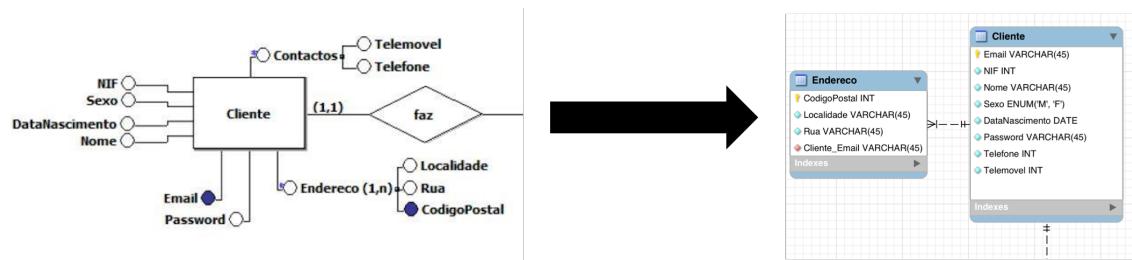


Figura 7 – Cliente.

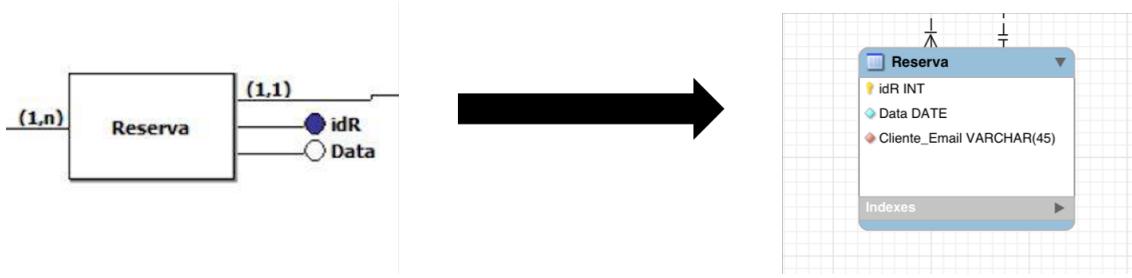


Figura 8 – Reserva.

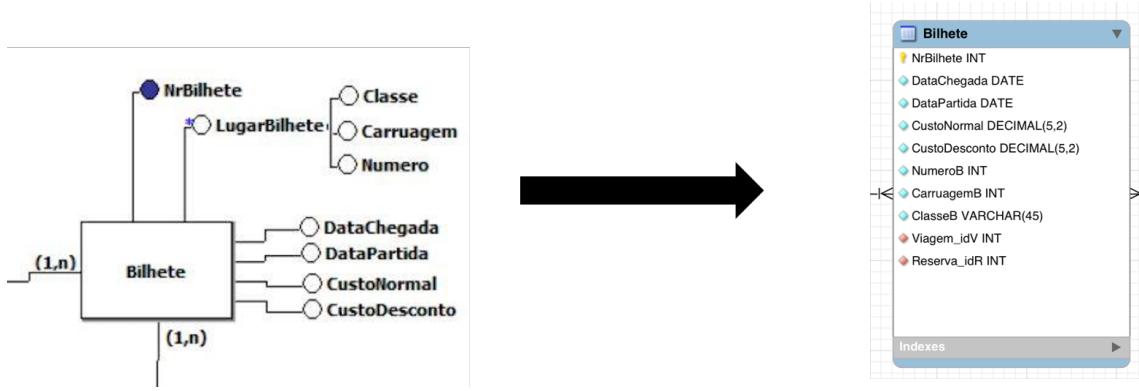


Figura 9 – Bilhete.

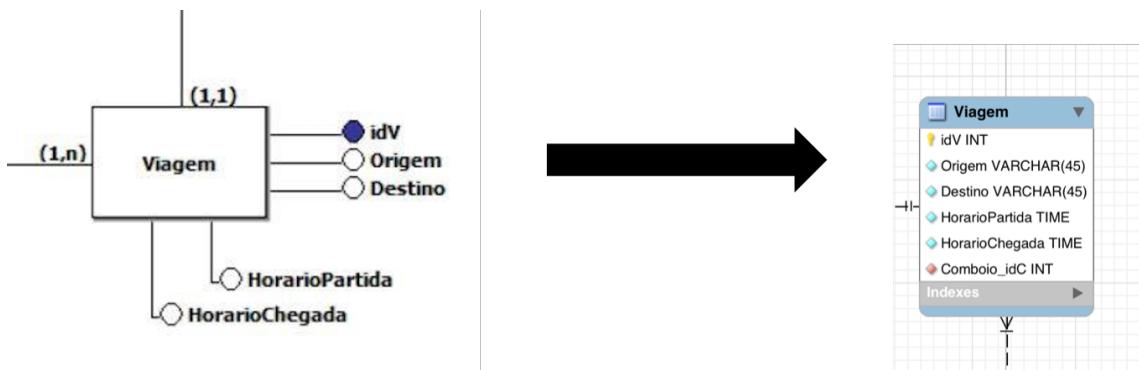


Figura 10 – Viagem.



Figura 11 – Comboio.

5.2. Validação do modelo através da normalização

Por vezes, quando realizamos a passagem do modelo conceptual para o modelo lógico são formadas algumas anomalias que tornam o modelo inválido e o sistema da base de dados inconsistente. Para que tal não aconteça, realizaremos o processo de normalização que converte cada entidade gradualmente para “formas normais”, desde a primeira forma normal (1FN) até à terceira forma normal (3FN).

Este método tem como principal objetivo decifrar reações com redundância de dados, relações com base nas suas chaves e dependências nos seus atributos, podendo assim eliminar possíveis anomalias que se encontrem na nossa base de dados.

5.2.1 Primeira Forma Normal (1FN)

Nesta primeira fase (1FN) é necessário identificar os possíveis elementos de informação repetidos nas tabelas, isto é, verificar se todos os valores de todos os atributos são atómicos, ou seja, se não é possível decompô-los.

Para que esta forma seja respeitada também não é possível existir subgrupos de atributos repetidos.

Atributos Multi-Valor:

Como será possível visualizar, nas figuras seguintes serão apresentados os atributos MV onde na passagem do modelo conceptual para o modelo lógico foi necessária a criação de uma tabela para cada um desses atributos, confirmando assim que é respeitada a 1FN.

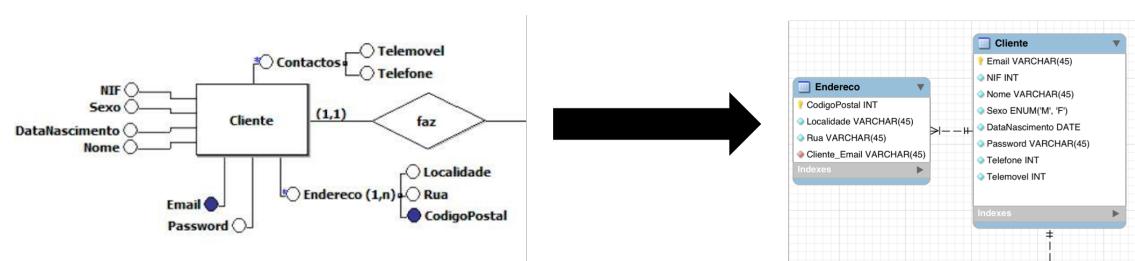


Figura 12 – Endereço (1FN).



Figura 13 – Lugar (1FN).

Atributos Compostos:

Na tradução do esquema conceptual para o esquema lógico, todos os atributos compostos foram colocados nas tabelas das respetivas entidades, respeitando mais uma vez a 1FN.

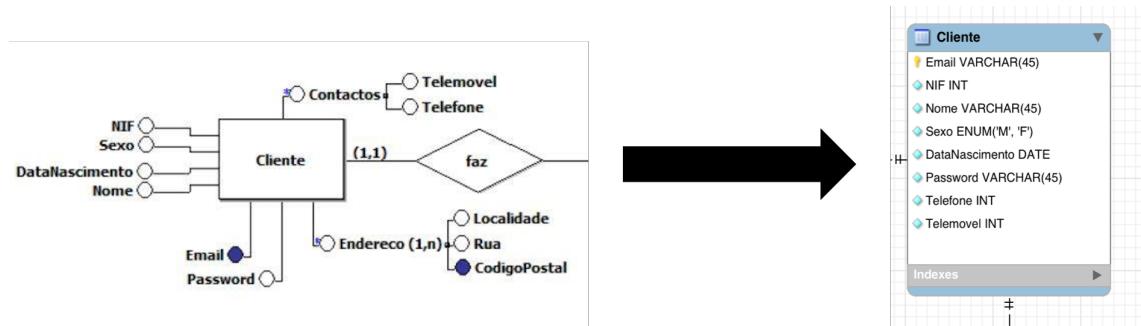


Figura 14 – Cliente (1FN).



Figura 15 – Bilhete (1FN).

5.2.2 Segunda Forma Normal (2FN)

Nesta fase, e após completar a primeira forma normal, é necessário eliminar redundâncias e analisar dependências parciais, ou seja, em cada relação todos os atributos que não pertencem à chave primária dependem de toda esta mesma chave e não apenas parcialmente dela.

Sendo assim, iniciamos a correção do modelo apenas nas chaves primárias compostas, uma vez que as chaves primárias simples já respeitam a 2FN.

Neste modelo existe apenas uma chave primária composta no atributo MV Lugar, pois era essencial identificar de forma unívoca o lugar num determinado comboio. Logo, como existem vários comboios e a numeração dos seus lugares é igual em todos os comboios, era preciso mais uma chave primária para definir o comboio em questão, pois o lugar referente já é definido pela outra chave primária.

Em suma, o modelo já respeita a normalização da segunda forma normal.

5.2.3 Terceira Forma Normal (3FN)

Nesta última fase, para além de ser fundamental que o modelo esteja na 2FN, é preciso verificar que os atributos que não pertence à chave não dependem de nenhum atributo que também não pertencem à chave.

Caso seja necessário efetuar a passagem a 3FN, separa-se os atributos que dependem de outro atributo não dependente à chave, decompondo a relação em duas (ou mais) relações.

No nosso caso, não é preciso, visto que o modelo já se encontra normalizado para a 3FN.

Concluímos, após o processo de normalização, que o nosso modelo está normalizado até a terceira forma normal e assim sendo é um modelo válido, logo não existem redundâncias de informação nem falta de consistência na base de dados.

5.3. Validação do modelo através das transações do utilizador

Neste tópico, vamos analisar as transações realizadas no modelo lógico de modo a validar o mesmo. Para isso, definimos várias transações que vamos agora apresentar e averiguar se são todas respeitadas no esquema lógico.

Todas as seguintes transações têm como objetivo principal a criação de uma reserva por parte do cliente.

5.3.1 Inserir Cliente

A primeira transação refere-se à inserção de um novo cliente na base de dados. Tal como já foi referido, o utilizador precisa de se registar na plataforma para poder realizar uma reserva.

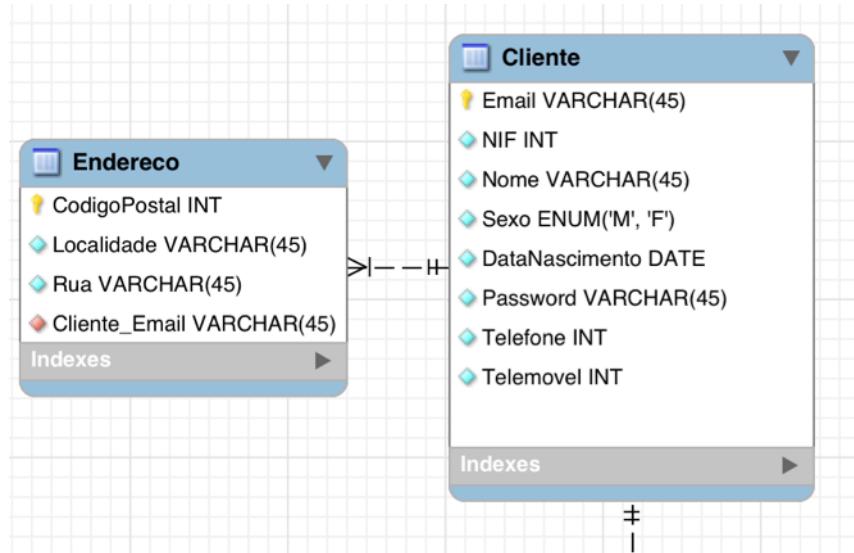


Figura 16 – Inserir Cliente.

5.3.2 Inserir Reserva

Ao inserir a nova reserva o cliente precisa de estar autenticado e esta entidade tem de conter as informações de todos os bilhetes da reserva.

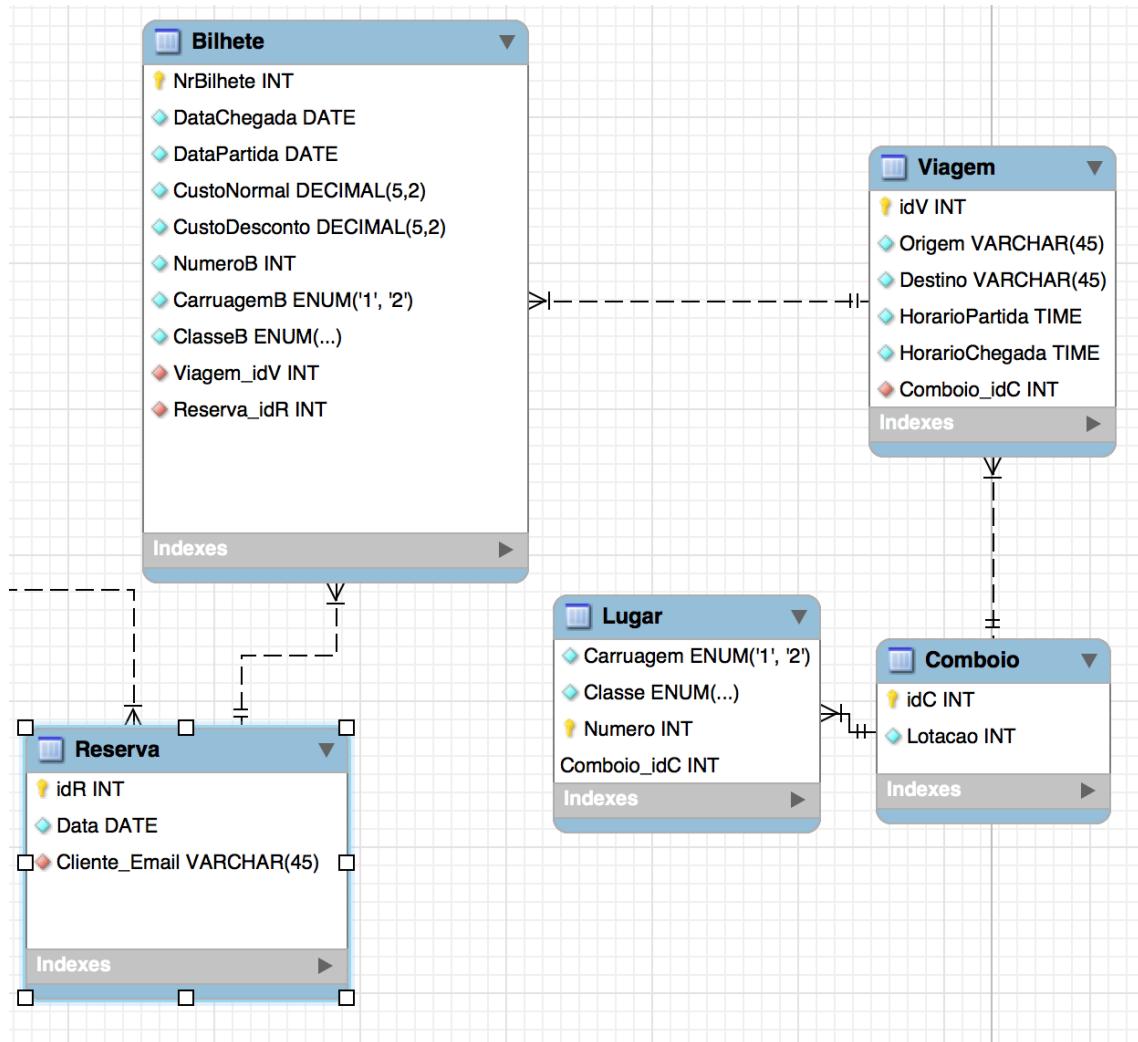


Figura 17 – Inserir Reserva.

5.3.3 Inserir Bilhete

A inserção de um bilhete tem as informações das datas, do custo e do lugar do cliente no comboio da viagem correspondente.

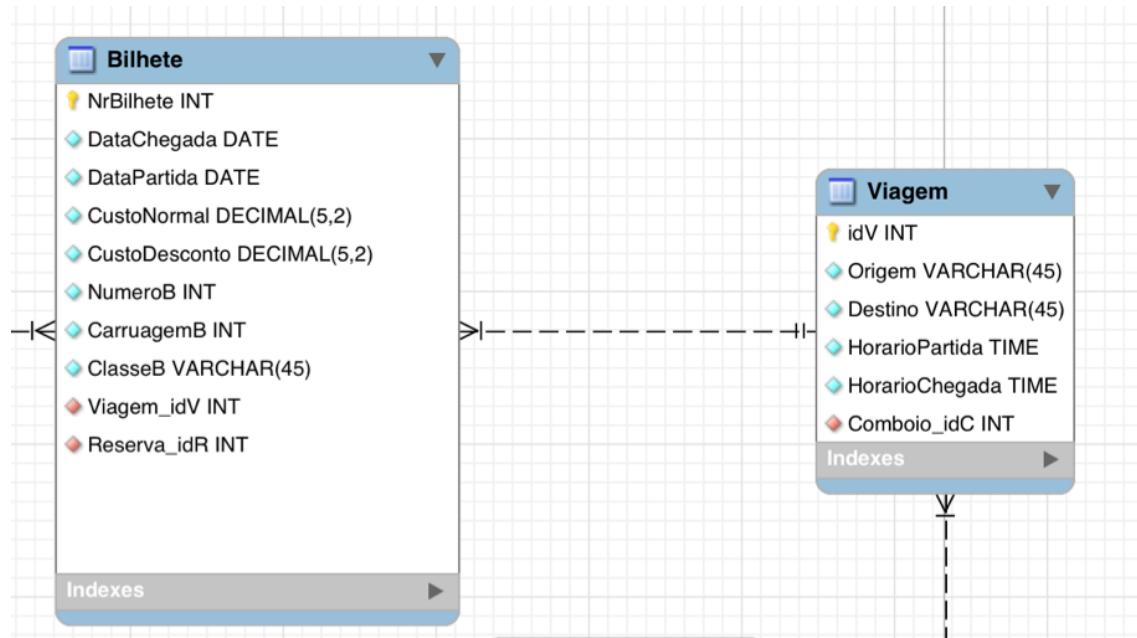


Figura 18 – Inserir Bilhete.

5.3.4 Consultar Viagem

Cliente tem acesso a um leque finito de viagens propostas na plataforma, de modo a consultar o id de cada viagem para posteriormente colocar na sua reserva.

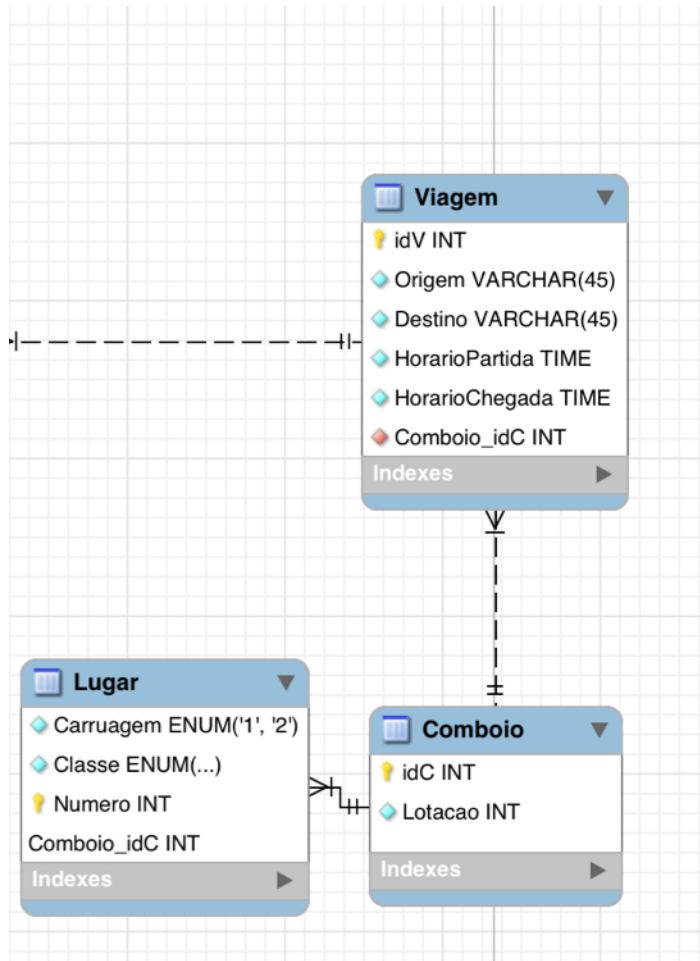


Figura 19 – Inserir Viagem.

5.4. Restrições de Integridade

As restrições de integridade são regras que devem ser impostas para que a base de dados não fique incompleta, pouca exata ou inconsistente.

Necessidade de valores

Existem atributos que têm de ter sempre valor válido, ou seja, que não podem ser nulos. No nosso modelo lógico, optamos por ter todos os valores não nulos, tornando assim os diversos campos de preenchimento obrigatório. No entanto, enquanto que alguns deles, como os horários, origens e destinos nas viagens têm de ser de facto não nulos, outros como a data de nascimento no cliente poderiam até não ser introduzidos se tivesse sido tomada essa opção.

Alguns exemplos de atributos que nunca poderiam ser nulos por serem essenciais:

- **Nome:** é um atributo da entidade cliente que ajuda a identificá-lo, logo não pode ser nulo.
- **NumeroB:** é um atributo da entidade bilhete que identifica o lugar reservado para essa viagem, logo também tem de ter valor não nulo.
- **Lotacao:** atributo da entidade comboio que identifica os lugares totais de cada comboio, sendo, por isso, essencial para determinar os lugares livres disponíveis, podendo ser não nulo.

Restrições de domínio dos atributos

Todos os atributos têm um domínio, ou seja, um conjunto de valores que podem tomar. Estes parâmetros encontram-se bem definidos na secção 4.2 onde é feita a descrição dos tipos de atributos. A título de exemplo, o NIF do cliente toma o tipo de valor INT visto ser um conjunto de inteiros e a password tem o tipo VARCHAR(45) visto ser um conjunto de caracteres.

Multiplicidade

Na passagem do modelo conceitual para o lógico, os atributos multi-valor geraram uma nova tabela com relacionamento de 1:N. Os relacionamentos de N:M geraram também uma nova tabela de modo a criar relacionamentos de 1:N, no entanto, não temos relacionamentos de N:M no nosso modelo conceitual.

A seguir, encontra-se a tabela de relacionamentos do modelo lógico após serem verificadas estas restrições de multiplicidade:

Entidade	Multiplicidade	Multiplicidade	Entidade
Endereço	N	1	Cliente
Cliente	1	N	Reserva

Reserva	1	N	Bilhete
Bilhete	N	1	Viagem
Viagem	N	1	Comboio
Comboio	1	N	Lugar

Tabela 6 – Descrição dos Relacionamentos no Modelo Lógico.

Integridade de entidade

A chave primária de uma entidade nunca pode assumir valores nulos, algo que pode ser verificado mais uma vez através da secção 4.2. Concretamente, convém também referir que esta situação também se deve verificar quando há chaves primárias compostas como acontece na tabela do lugar.

Integridade referencial

Para existir uma chave estrangeira, terá de existir a chave primária respetiva noutra tabela.

Alterações nas chaves primárias devem causar alterações nas chaves estrangeiras respetivas.

No nosso modelo, consideramos obrigatório que todas as chaves estrangeiras fossem não nulas. Por exemplo, na entidade reserva é indispensável que a chave estrangeira email exista, pois não podemos ter uma reserva sem estar associada a um cliente e à sua chave primária.

O segundo problema de integridade referencial consiste em definir as condições que se têm de verificar quando um candidato a chave estrangeira é inserido, atualizado ou apagado. Para isto, vamos analisar diversos casos para as entidades viagem e comboio:

- **Caso 1: inserir um registo na relação filho** – verifica-se se a chave estrangeira na relação filho tem alguma correspondência na relação pai. Ou seja, neste caso, verificamos se a chave estrangeira Comboio_idC do novo elemento da entidade Viagem está definida como valor da chave primária idC da entidade Comboio.
- **Caso 2: eliminar um registo da relação filho** – a remoção de um registo da relação filho não afeta a integridade, o que neste caso verifica-se visto que poderíamos eliminar um registo da entidade Viagem sem causar problemas de integridade.
- **Caso 3: atualização do registo da chave estrangeira numa relação filho** – o registo atualizado na relação filho terá de ter correspondência na relação pai tal como no caso 1. Ou seja, atualizando algum registo de Comboio_idC na entidade Viagem, terá de existir esse novo valor na chave primária de Comboio.
- **Caso 4: inserir um registo na relação pai** – uma inserção na relação pai não afeta a integridade referencial, no entanto, para que a informação esteja atualizada podemos

fazer alterações na relação filho. Neste caso, inserir um novo Comboio não afetaria a integridade, mas seria um Comboio sem qualquer Viagem associada.

- **Caso 5: eliminar um registo da relação pai** – apagar um registo da relação pai pode afetar a integridade referencial visto que é possível que haja um registo na relação filho que dependa do apagado fazendo com que haja um registo na relação filho sem correspondência na relação pai. Se apagarmos um registo da entidade Comboio associado a algum registo da entidade Viagem, vai ser perdida a entidade referencial visto que poderemos passar a ter uma Viagem sem Comboio associado, algo que não faz sentido no nosso modelo. A estratégia que adotamos foi a NO ACTION, ou seja, se existir um Comboio associado a uma Viagem, esse Comboio não poderá ser apagado.
- **Caso 6: atualização do registo da chave primária numa relação pai** – atualizar um registo numa relação pai pode causar perda de integridade pelas mesmas razões do caso 5 visto que pode fazer com que haja um registo na relação filho que perca a correspondência na relação pai. Mais uma vez, adotamos a estratégia NO ACTION, ou seja, se o Comboio estiver associado a alguma Viagem, a sua chave primária não poderá ser modificada.

Restrições Gerais

Não consideramos restrições do “mundo real”, tais como possíveis impedimentos de um comboio fazer mais do que um certo viagens, entre outros.

5.5. Modelo Lógico e Utilizador

Nesta fase, e devido a inexistência de um utilizador em concreto, não podemos considerar este tópico nesta etapa deste projeto.

6. Modelo Físico

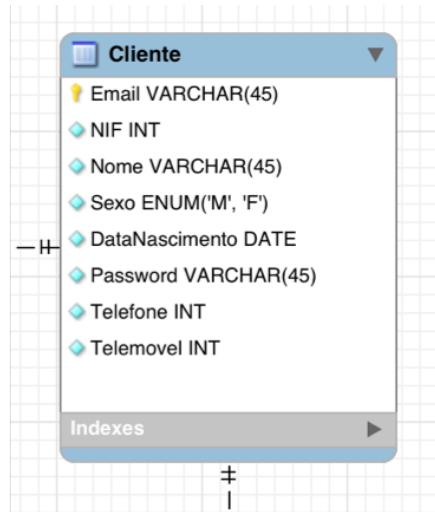
Nesta última secção realizamos a tradução do modelo lógico para o modelo físico. Este modelo irá ser implementado no SGBD.

Para realizar esta implementação utilizamos o MySQL Workbench, que é um programa desenvolvido pela Oracle. No Workbench criaremos as tabelas já apresentadas anteriormente no esquema lógico, *queries* e transações que serão bastante úteis para o bom funcionamento da plataforma bem como permitirão facilitar a interação com o utilizador.

6.1. Tradução e implementação do modelo lógico no SGBD

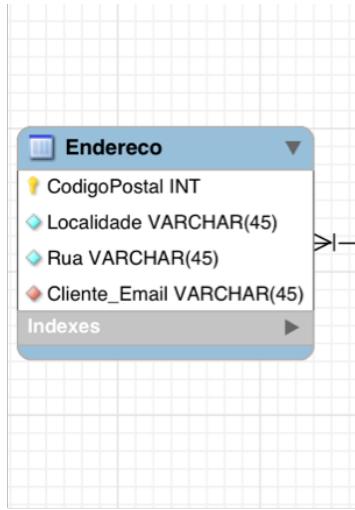
Esta é a primeira fase da modelação física e neste processo é necessário traduzir o esquema lógico apresentado na secção anterior. Para isso, utilizamos a ferramenta do Workbench denominada por *Forward Enginner* que a partir do modelo anterior gera o código em SQL para as tabelas correspondentes e ainda para os relacionamentos entre elas.

6.1.1 Criação das Tabelas e dos Relacionamentos



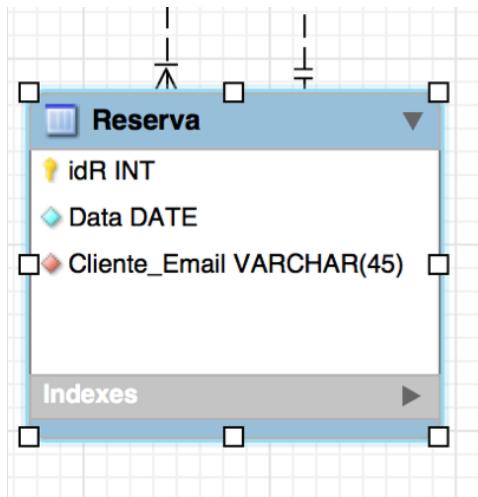
```
-- Table `ComboiosLusitanos`.`Cliente`  
--  
CREATE TABLE IF NOT EXISTS `ComboiosLusitanos`.`Cliente` (  
  `Email` VARCHAR(45) NOT NULL,  
  `NIF` INT NOT NULL,  
  `Nome` VARCHAR(45) NOT NULL,  
  `Sexo` ENUM('M', 'F') NOT NULL,  
  `DataNascimento` DATE NOT NULL,  
  `Password` VARCHAR(45) NOT NULL,  
  `Telefone` INT NOT NULL,  
  `Telemovel` INT NOT NULL,  
  PRIMARY KEY (`Email`))  
ENGINE = InnoDB;
```

Figura 20 – Tabela Cliente.



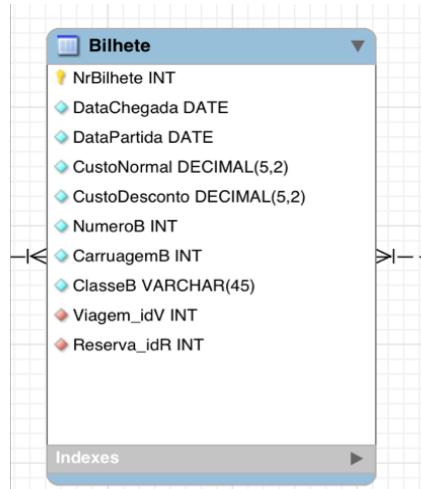
```
-- Table `ComboiosLusitanos`.`Endereco`
-----
CREATE TABLE IF NOT EXISTS `ComboiosLusitanos`.`Endereco` (
  `CodigoPostal` INT NOT NULL,
  `Localidade` VARCHAR(45) NOT NULL,
  `Rua` VARCHAR(45) NOT NULL,
  `Cliente_Email` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`CodigoPostal`),
  CONSTRAINT `fk_Endereco_Cliente1`
    FOREIGN KEY (`Cliente_Email`)
    REFERENCES `ComboiosLusitanos`.`Cliente` (`Email`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 21 – Tabela Endereço.



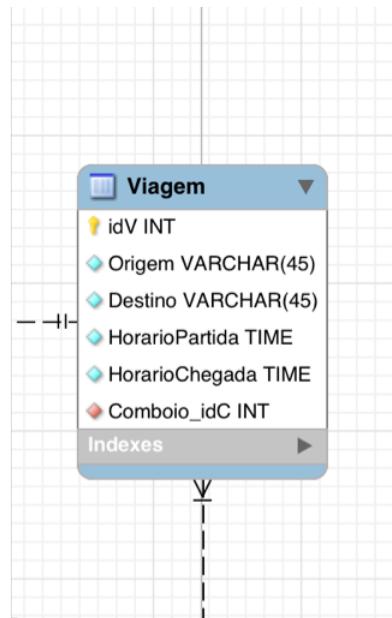
```
-- Table `ComboiosLusitanos`.`Reserva`
-----
CREATE TABLE IF NOT EXISTS `ComboiosLusitanos`.`Reserva` (
  `idR` INT NOT NULL,
  `Data` DATE NOT NULL,
  `Cliente_Email` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idR`),
  INDEX `fk_Reserva_Cliente1_idx` (`Cliente_Email` ASC),
  CONSTRAINT `fk_Reserva_Cliente1`
    FOREIGN KEY (`Cliente_Email`)
    REFERENCES `ComboiosLusitanos`.`Cliente` (`Email`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 22 – Tabela Reserva.



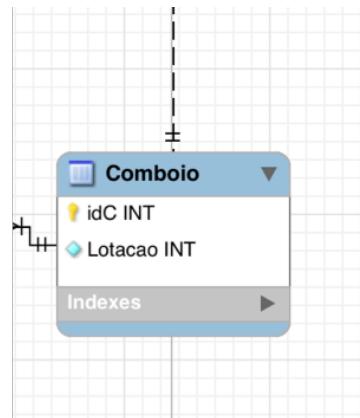
```
-- Table `ComboiosLusitanos`.`Bilhete`
-----
CREATE TABLE IF NOT EXISTS `ComboiosLusitanos`.`Bilhete` (
  `NrBilhete` INT NOT NULL,
  `DataChegada` DATE NOT NULL,
  `DataPartida` DATE NOT NULL,
  `CustoNormal` DECIMAL(5,2) NOT NULL,
  `CustoDesconto` DECIMAL(5,2) NOT NULL,
  `NumeroB` INT NOT NULL,
  `CarruagemB` ENUM('1', '2') NOT NULL,
  `ClasseB` ENUM('económica', 'executiva') NOT NULL,
  `Viagem_idV` INT NOT NULL,
  `Reserva_idR` INT NOT NULL,
  PRIMARY KEY (`NrBilhete`),
  INDEX `fk_Bilhete_Viagem1_idx` (`Viagem_idV` ASC),
  INDEX `fk_Bilhete_Reserva1_idx` (`Reserva_idR` ASC),
  CONSTRAINT `fk_Bilhete_Viagem1`
    FOREIGN KEY (`Viagem_idV`)
    REFERENCES `ComboiosLusitanos`.`Viagem` (`idV`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Bilhete_Reserva1`
    FOREIGN KEY (`Reserva_idR`)
    REFERENCES `ComboiosLusitanos`.`Reserva` (`idR`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 23 – Tabela Bilhete.



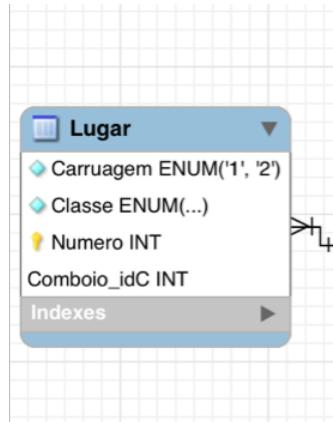
```
-- Table `ComboiosLusitanos`.`Viagem`
-----
CREATE TABLE IF NOT EXISTS `ComboiosLusitanos`.`Viagem` (
  `idV` INT NOT NULL,
  `Origem` VARCHAR(45) NOT NULL,
  `Destino` VARCHAR(45) NOT NULL,
  `HorarioPartida` TIME NOT NULL,
  `HorarioChegada` TIME NOT NULL,
  `Comboio_idC` INT NOT NULL,
  PRIMARY KEY (`idV`),
  INDEX `fk_Viagem_Combocio1_idx`(`Comboio_idC` ASC),
  CONSTRAINT `fk_Viagem_Combocio1`
    FOREIGN KEY (`Comboio_idC`)
    REFERENCES `ComboiosLusitanos`.`Comboio`(`idC`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 24 – Tabela Viagem.



```
-- Table `ComboiosLusitanos`.`Comboio`  
-----  
CREATE TABLE IF NOT EXISTS `ComboiosLusitanos`.`Comboio` (  
  `idC` INT NOT NULL,  
  `Lotacao` INT NOT NULL,  
  PRIMARY KEY (`idC`))  
ENGINE = InnoDB;
```

Figura 25 – Tabela Comboio.



```
-- Table `ComboiosLusitanos`.`Lugar`
-----
CREATE TABLE IF NOT EXISTS `ComboiosLusitanos`.`Lugar` (
  `Carruagem` ENUM('1', '2') NOT NULL,
  `Classe` ENUM('económica', 'executiva') NOT NULL,
  `Numero` INT NOT NULL,
  `Comboio_idC` INT NOT NULL,
  PRIMARY KEY (`Numero`, `Comboio_idC`),
  CONSTRAINT `fk_Lugar_Comboio1`
    FOREIGN KEY (`Comboio_idC`)
    REFERENCES `ComboiosLusitanos`.`Comboio` (`idC`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 26 – Tabela Lugar.

6.2. Povoação da Base de Dados

```
INSERT INTO Bilhete
(NrBilhete, DataChegada, DataPartida, CustoNormal, CustoDesconto, NumeroB, CarruagemB, ClasseB, Viagem_idV, Reserva_idR)
VALUES
(1, '2016-11-19', '2016-11-19', 5, 4.5, 7, 2, 'económica', 11, 4),
(2, '2016-11-19', '2016-11-19', 9, 8.1, 8, 2, 'económica', 21, 6),
(3, '2016-12-19', '2016-12-19', 24, 21.6, 10, 2, 'económica', 31, 18),
(4, '2017-01-19', '2017-01-19', 45, 40.5, 11, 2, 'económica', 41, 9),
(5, '2016-11-20', '2016-11-21', 50, 45, 4, 1, 'executiva', 51, 25),
(6, '2017-02-19', '2017-02-19', 47, 42.3, 9, 2, 'económica', 61, 12),
(7, '2016-11-22', '2016-11-22', 34, 30.6, 6, 1, 'executiva', 71, 7),
(8, '2016-11-23', '2016-11-23', 20, 18, 3, 1, 'executiva', 81, 4),
(9, '2016-12-11', '2016-12-11', 25, 22.5, 5, 1, 'executiva', 91, 2),
(10, '2016-11-29', '2016-11-29', 62, 55.8, 1, 1, 'executiva', 101, 7),
(11, '2017-03-19', '2017-03-19', 14, 12.6, 7, 2, 'económica', 111, 1),
(12, '2017-04-10', '2017-04-10', 32, 28.8, 10, 2, 'económica', 121, 3),
(13, '2016-11-19', '2016-11-19', 17, 15.3, 11, 2, 'económica', 131, 9),
(14, '2017-03-10', '2017-03-10', 12, 10.8, 12, 2, 'económica', 141, 11),
(15, '2017-01-09', '2017-01-09', 25, 22.5, 13, 2, 'económica', 151, 20),
(16, '2016-11-25', 8, 7.2, 14, 2, 'económica', 161, 7),
(17, '2016-11-30', '2016-11-30', 9, 8.1, 15, 2, 'económica', 171, 10),
(18, '2016-11-27', '2016-11-27', 4, 3.6, 8, 2, 'económica', 181, 17),
(19, '2016-11-24', '2016-11-24', 24, 21.6, 10, 2, 'económica', 191, 19),
(20, '2016-12-08', '2016-12-08', 27, 24.3, 12, 2, 'económica', 201, 18),
(21, '2017-03-01', '2017-03-01', 5, 4.5, 14, 2, 'económica', 12, 16),
(22, '2016-11-19', '2016-11-19', 9, 8.1, 6, 1, 'executiva', 22, 18),
(23, '2016-12-07', '2016-12-07', 24, 21.6, 3, 1, 'executiva', 32, 21),
(24, '2017-01-29', '2017-01-29', 45, 40.5, 9, 2, 'económica', 42, 22),
(25, '2016-12-21', '2016-12-21', 50, 45, 1, 1, 'executiva', 52, 4),
(26, '2017-02-09', '2017-02-09', 47, 42.3, 11, 2, 'económica', 62, 14),
(27, '2016-11-23', '2016-11-23', 34, 30.6, 15, 2, 'económica', 72, 5),
(28, '2017-01-23', '2017-01-23', 20, 18, 4, 1, 'executiva', 82, 23),
(29, '2016-12-11', '2016-12-11', 25, 22.5, 2, 1, 'executiva', 92, 4),
(30, '2016-12-30', '2016-12-30', 62, 55.8, 8, 2, 'económica', 102, 22),
(31, '2017-04-19', '2017-04-19', 14, 12.6, 5, 1, 'executiva', 112, 24),
(32, '2017-04-03', '2017-04-03', 32, 28.8, 12, 2, 'económica', 122, 16),
(33, '2016-12-01', '2016-12-01', 17, 15.3, 9, 2, 'económica', 132, 5),
(34, '2017-01-10', '2017-01-10', 12, 10.8, 2, 1, 'executiva', 142, 24),
(35, '2017-01-27', '2017-01-27', 25, 22.5, 5, 1, 'executiva', 152, 15),
(36, '2016-11-27', '2016-11-27', 8, 7.2, 4, 1, 'executiva', 162, 8),
(37, '2017-01-30', '2017-01-30', 9, 8.1, 1, 1, 'executiva', 172, 23),
(38, '2016-12-02', '2016-12-02', 4, 3.6, 3, 1, 'executiva', 182, 8),
(39, '2016-12-06', '2016-12-06', 24, 21.6, 7, 2, 'económica', 192, 13),
(40, '2016-12-09', '2016-12-09', 27, 24.3, 14, 2, 'económica', 202, 25);
```

Figura 27 – Cliente (Povoamento).

```

INSERT INTO Endereco
(CodigoPostal, Localidade, Rua, Cliente_Email)
VALUES
(4700123, 'Braga', 'Rua Pacheco Dias', 'maria@gmail.com'),
(4720257, 'Vila Verde', 'Rua Quinta Avenida', 'leocunha88@hotmail.com'),
(2386146, 'Vila Franca de Xira', 'Rua Gomes da Silva', 'matilde@gmail.com'),
(1234845, 'Viana do Castelo', 'Travessa da Avela', 'bia130894@hotmail.com'),
(2135342, 'Santarém', 'Rua Romário de Albuquerque', 'carolina@gmail.com'),
(4898523, 'Viana do Castelo', 'Rua de Cima', 'marianna0909@hotmail.com'),
(3843834, 'Estremoz', 'Rua Rebelo Santos', 'analopec@gmail.com'),
(4710456, 'Braga', 'Rua da Alegria', 'inesgoncalves@hotmail.com'),
(2098534, 'Lisboa', 'Rua de Baixo', 'margarida@gmail.com'),
(1221124, 'Melgaço', 'Rua das Castanhas', 'sofifreitas@hotmail.com'),
(7253213, 'Beja', 'Rua da Primavera', 'larasilva@gmail.com'),
(1224213, 'Valença', 'Rua dos Pardais', 'laurinhapereira@hotmail.com'),
(8342242, 'Portimão', 'Rua da Sardinha', 'franciscasacunha@gmail.com'),
(4653123, 'Marco de Canavezes', 'Rua Jorge Sousa', 'jucunha@hotmail.com'),
(2134124, 'Coimbra', 'Rua do Estudante', 'alicemaravilha@gmail.com'),
(4632564, 'Vila Nova de Gaia', 'Rua Nova', 'claridade@hotmail.com'),
(7812234, 'Albufeira', 'Rua da Falésia', 'diana@gmail.com'),
(9342890, 'Lagoa', 'Rua da Praia', 'lua@hotmail.com'),
(1321653, 'Valença', 'Rua de Espanha', 'mdln1996@gmail.com'),
(4521234, 'Terras de Bouro', 'Rua da Velha', 'malfadasilva@hotmail.com'),
(8762922, 'Faro', 'Rua da Solidão', 'birogeria@gmail.com'),
(4209853, 'Porto', 'Rua do Almirante José Silva Dias', 'maralves@hotmail.com'),
(2134321, 'Lisboa', 'Rua Nova Angola', 'constapereira@gmail.com'),
(3421321, 'Mealhada', 'Rua do Leitão', 'iris123@gmail.com'),
(2134732, 'Lisboa', 'Rua da Luz', 'martimhanha@hotmail.com'),
(7631213, 'Évora', 'Rua de Roma', 'joaofonseca@gmail.com'),
(5213774, 'Chaves', 'Rua dos Antepassados', 'rodrigomaradona@hotmail.com'),
(5783323, 'Bragança', 'Rua dos Apaixonados', 'santiagoferro@gmail.com'),
(5232234, 'Chaves', 'Rua dos Entrelaçados', 'francottih@hotmail.com'),
(6234523, 'Vila Real', 'Rua Duarte Carmo II', 'afonsocunha@gmail.com'),
(2345632, 'Seixal', 'Rua dos Mágicos', 'tomasmuller@hotmail.com'),
(2343123, 'Seixal', 'Rua Cosme Damião', 'miguelnogueira@gmail.com'),
(4351312, 'Marco de Canavezes', 'Rua dos Desafogados', 'guilhermecabral@hotmail.com'),
(7354633, 'Beja', 'Rua Grande', 'gabriel@gmail.com'),
(2345999, 'Sintra', 'Rua da Liberdade', 'duartebelo@hotmail.com'),
(6573435, 'Vila Real', 'Rua da Beira Interior', 'goncalosilva@gmail.com'),
(3987909, 'Arouca', 'Rua do Esconderijo', 'pedrodias@hotmail.com'),
(5475767, 'Bragança', 'Rua da Camélia Vermelha', 'tiagoteixeira@gmail.com'),
(4715380, 'Braga', 'Rua dos Abades', 'rafasilva@hotmail.com'),
(4653987, 'Amares', 'Rua da Amendoeira', 'diogomendes@gmail.com'),
(4423134, 'Barcelos', 'Rua dos Pecadores', 'lourenco98@hotmail.com'),
(6211546, 'Vila Real', 'Avenida Trás-os-Montes', 'lucas@gmail.com'),
(8323000, 'Portimão', 'Rua dos Pescadores', 'dinis@hotmail.com'),
(4780344, 'Braga', 'Avenida Central', 'salvador@gmail.com'),
(3679312, 'Aveiro', 'Rua do Rio', 'vicentexpto@hotmail.com'),
(2004199, 'Lisboa', 'Rua dos Campeões', 'guga@gmail.com'),
(2100564, 'Amadora', 'Rua De Moçambique', 'simaosabrosa79@hotmail.com'),
(8888777, 'Albufeira', 'Rua do Camarão', 'josecidplatina@gmail.com'),
(4700222, 'Braga', 'Rua Abade Loureira', 'davidfonseca1995@hotmail.com');

```

Figura 28 – Endereço (Povoamento).

```

INSERT INTO Reserva
(idR, Data, Cliente_Email)
VALUES
(1, '2016-11-11', 'josecidplatina@gmail.com'),
(2, '2016-11-11', 'vicentexpto@hotmail.com'),
(3, '2016-11-11', 'davidfonseca1995@hotmail.com'),
(4, '2016-11-11', 'sofifreitas@hotmail.com'),
(5, '2016-11-12', 'rodrigomaradona@hotmail.com'),
(6, '2016-11-12', 'lua@hotmail.com'),
(7, '2016-11-12', 'salvador@gmail.com'),
(8, '2016-11-13', 'pedrodias@hotmail.com'),
(9, '2016-11-13', 'sarasampaio@hotmail.com'),
(10, '2016-11-13', 'tiagoteixeira@gmail.com'),
(11, '2016-11-13', 'guga@gmail.com'),
(12, '2016-11-14', 'martimantha@hotmail.com'),
(13, '2016-11-14', 'duartebelo@hotmail.com'),
(14, '2016-11-15', 'biirogeria@gmail.com'),
(15, '2016-11-15', 'margarida@gmail.com'),
(16, '2016-11-15', 'guilhermecabral@hotmail.com'),
(17, '2016-11-16', 'santiagoferro@gmail.com'),
(18, '2016-11-16', 'constapereira@gmail.com'),
(19, '2016-11-16', 'lucas@gmail.com'),
(20, '2016-11-16', 'inesgoncalves@hotmail.com'),
(21, '2016-11-17', 'carolina@gmail.com'),
(22, '2016-11-17', 'iris123@gmail.com'),
(23, '2016-11-17', 'franctotti@hotmail.com'),
(24, '2016-11-18', 'analopes@gmail.com'),
(25, '2016-11-19', 'tomasmuller@hotmail.com');

```

Figura 29 – Reserva (Povoamento).

```

INSERT INTO Bilhete
(NrBilhete, DataChegada, DataPartida, CustoNormal, CustoDesconto, NumeroB, CarruagemB, ClasseB, Viagem_idV, Reserva_idR)
VALUES
(1, '2016-11-19', '2016-11-19', 5, 4.5, 7, 2, 'económica', 11, 4),
(2, '2016-11-19', '2016-11-19', 9, 8.1, 8, 2, 'económica', 21, 6),
(3, '2016-12-19', '2016-12-19', 24, 21.6, 10, 2, 'económica', 31, 18),
(4, '2017-01-19', '2017-01-19', 45, 40.5, 11, 2, 'económica', 41, 9),
(5, '2016-11-20', '2016-11-21', 50, 45, 4, 1, 'executiva', 51, 25),
(6, '2017-02-19', '2017-02-19', 47, 42.3, 9, 2, 'económica', 61, 12),
(7, '2016-11-22', '2016-11-22', 34, 30.6, 6, 1, 'executiva', 71, 7),
(8, '2016-11-23', '2016-11-23', 20, 18, 3, 1, 'executiva', 81, 4),
(9, '2016-12-11', '2016-12-11', 25, 22.5, 5, 1, 'executiva', 91, 2),
(10, '2016-11-29', '2016-11-29', 62, 55.8, 1, 1, 'executiva', 101, 7),
(11, '2017-03-19', '2017-03-19', 14, 12.6, 7, 2, 'económica', 111, 1),
(12, '2017-04-10', '2017-04-10', 32, 28.8, 10, 2, 'económica', 121, 3),
(13, '2016-11-19', '2016-11-19', 17, 15.3, 11, 2, 'económica', 131, 9),
(14, '2017-03-10', '2017-03-10', 12, 10.8, 12, 2, 'económica', 141, 11),
(15, '2017-01-09', '2017-01-09', 25, 22.5, 13, 2, 'económica', 151, 20),
(16, '2016-11-25', '2016-11-25', 8, 7.2, 14, 2, 'económica', 161, 7),
(17, '2016-11-30', '2016-11-30', 9, 8.1, 15, 2, 'económica', 171, 10),
(18, '2016-11-27', '2016-11-27', 4, 3.6, 8, 2, 'económica', 181, 17),
(19, '2016-11-24', '2016-11-24', 24, 21.6, 10, 2, 'económica', 191, 19),
(20, '2016-12-08', '2016-12-08', 27, 24.3, 12, 2, 'económica', 201, 18),
(21, '2017-03-01', '2017-03-01', 5, 4.5, 14, 2, 'económica', 12, 16),
(22, '2016-11-19', '2016-11-19', 9, 8.1, 6, 1, 'executiva', 22, 18),
(23, '2016-12-07', '2016-12-07', 24, 21.6, 3, 1, 'executiva', 32, 21),
(24, '2017-01-29', '2017-01-29', 45, 40.5, 9, 2, 'económica', 42, 22),
(25, '2016-12-21', '2016-12-21', 50, 45, 1, 1, 'executiva', 52, 4),
(26, '2017-02-09', '2017-02-09', 47, 42.3, 11, 2, 'económica', 62, 14),
(27, '2016-11-23', '2016-11-24', 34, 30.6, 15, 2, 'económica', 72, 5),
(28, '2017-01-23', '2017-01-23', 20, 18, 4, 1, 'executiva', 82, 23),
(29, '2016-12-11', '2016-12-11', 25, 22.5, 2, 1, 'executiva', 92, 4),
(30, '2016-12-30', '2016-12-30', 62, 55.8, 8, 2, 'económica', 102, 22),
(31, '2017-04-19', '2017-04-19', 14, 12.6, 5, 1, 'executiva', 112, 24),
(32, '2017-04-03', '2017-04-03', 32, 28.8, 12, 2, 'económica', 122, 16),
(33, '2016-12-01', '2016-12-01', 17, 15.3, 9, 2, 'económica', 132, 5),
(34, '2017-01-10', '2017-01-10', 12, 10.8, 2, 1, 'executiva', 142, 24),
(35, '2017-01-27', '2017-01-27', 25, 22.5, 5, 1, 'executiva', 152, 15),
(36, '2016-11-27', '2016-11-27', 8, 7.2, 4, 1, 'executiva', 162, 8),
(37, '2017-01-30', '2017-01-30', 9, 8.1, 1, 1, 'executiva', 172, 23),
(38, '2016-12-02', '2016-12-02', 4, 3.6, 3, 1, 'executiva', 182, 8),
(39, '2016-12-06', '2016-12-06', 24, 21.6, 7, 2, 'económica', 192, 13),
(40, '2016-12-09', '2016-12-09', 27, 24.3, 14, 2, 'económica', 202, 25);

```

Figura 30 – Bilhete (Povoamento).

```

INSERT INTO Viagem
(idV, Origem, Destino, HorarioPartida, HorarioChegada, Comboio_idC)
VALUES
(11, 'Braga', 'Porto', 121500, 130500, 1),
(21, 'Porto', 'Lisboa', 133000, 161000, 1),
(31, 'Lisboa', 'Madrid', 104500, 192000, 1),
(41, 'Lisboa', 'Barcelona', 120000, 220000, 4),
(51, 'Barcelona', 'Paris', 154000, 234000, 4),
(61, 'Barcelona', 'Milão', 141500, 235000, 10),
(71, 'Braga', 'Faro', 111500, 182000, 2),
(81, 'Lisboa', 'Portimão', 161000, 193000, 5),
(91, 'Faro', 'Sevilha', 092000, 120000, 2),
(101, 'Paris', 'Londres', 100000, 143000, 4),
(111, 'Braga', 'Vigo', 081500, 104500, 7),
(121, 'Braga', 'Vila Real', 095000, 115000, 9),
(131, 'Porto', 'Bragança', 091000, 122000, 6),
(141, 'Lisboa', 'Beja', 084500, 101500, 5),
(151, 'Beja', 'Lagos', 103000, 125000, 5),
(161, 'Porto', 'Viana do Castelo', 112500, 121500, 3),
(171, 'Viana do Castelo', 'Vigo', 123000, 135000, 3),
(181, 'Valência', 'Vigo', 093000, 110000, 8),
(191, 'Chaves', 'Lisboa', 090500, 154000, 5),
(201, 'Madrid', 'Barcelona', 105500, 163000, 10),
(12, 'Porto', 'Braga', 132500, 143500, 1),
(22, 'Lisboa', 'Porto', 183000, 235000, 1),
(32, 'Madrid', 'Lisboa', 193000, 045500, 1),
(42, 'Barcelona', 'Lisboa', 222000, 080000, 4),
(52, 'Paris', 'Barcelona', 050500, 171000, 4),
(62, 'Milão', 'Barcelona', 000500, 091000, 10),
(72, 'Faro', 'Braga', 183500, 014000, 2),
(82, 'Portimão', 'Lisboa', 194000, 235000, 5),
(92, 'Sevilha', 'Faro', 121000, 153000, 2),
(102, 'Londres', 'Paris', 144000, 181000, 4),
(112, 'Vigo', 'Braga', 110000, 133000, 7),
(122, 'Vila Real', 'Braga', 120000, 140000, 9),
(132, 'Bragança', 'Porto', 123000, 154000, 6),
(142, 'Beja', 'Lisboa', 103000, 130000, 5),
(152, 'Lagos', 'Beja', 130000, 152000, 5),
(162, 'Viana do Castelo', 'Porto', 123000, 132000, 3),
(172, 'Vigo', 'Viana do Castelo', 140000, 152000, 3),
(182, 'Vigo', 'Valência', 111500, 124500, 8),
(192, 'Lisboa', 'Chaves', 160000, 223500, 5),
(202, 'Barcelona', 'Madrid', 164000, 221500, 10);

```

Figura 31 – Viagem (Povoamento).

```
INSERT INTO Comboio
(idC, Lotacao)
VALUES
(1, 15),
(2, 15),
(3, 15),
(4, 15),
(5, 15),
(6, 15),
(7, 15),
(8, 15),
(9, 15),
(10, 15);
```

Figura 32 – Comboio (Povoamento).

```

INSERT INTO Lugar
(Carruagem, Classe, Numero, Comboio_idC)
VALUES
(1, 'executiva', 1, 1), (1, 'executiva', 2, 1), (1, 'executiva', 3, 1), (1, 'executiva', 4, 1), (1, 'executiva', 5, 1), (1, 'executiva', 6, 1), (2, 'económica', 7, 1), (2, 'económica', 8, 1), (2, 'económica', 9, 1), (2, 'económica', 10, 1), (2, 'económica', 11, 1), (2, 'económica', 12, 1), (2, 'económica', 13, 1), (2, 'económica', 14, 1), (2, 'económica', 15, 1), (1, 'executiva', 1, 2), (1, 'executiva', 2, 2), (1, 'executiva', 3, 2), (1, 'executiva', 4, 2), (1, 'executiva', 5, 2), (1, 'executiva', 6, 2), (2, 'económica', 7, 2), (2, 'económica', 8, 2), (2, 'económica', 9, 2), (2, 'económica', 10, 2), (2, 'económica', 11, 2), (2, 'económica', 12, 2), (2, 'económica', 13, 2), (2, 'económica', 14, 2), (2, 'económica', 15, 2), (1, 'executiva', 1, 3), (1, 'executiva', 2, 3), (1, 'executiva', 3, 3), (1, 'executiva', 4, 3), (1, 'executiva', 5, 3), (1, 'executiva', 6, 3), (2, 'económica', 7, 3), (2, 'económica', 8, 3), (2, 'económica', 9, 3), (2, 'económica', 10, 3), (2, 'económica', 11, 3), (2, 'económica', 12, 3), (2, 'económica', 13, 3), (2, 'económica', 14, 3), (2, 'económica', 15, 3), (1, 'executiva', 1, 4), (1, 'executiva', 2, 4), (1, 'executiva', 3, 4), (1, 'executiva', 4, 4), (1, 'executiva', 5, 4), (1, 'executiva', 6, 4), (2, 'económica', 7, 4), (2, 'económica', 8, 4), (2, 'económica', 9, 4), (2, 'económica', 10, 4), (2, 'económica', 11, 4), (2, 'económica', 12, 4), (2, 'económica', 13, 4), (2, 'económica', 14, 4), (2, 'económica', 15, 4), (1, 'executiva', 1, 5), (1, 'executiva', 2, 5), (1, 'executiva', 3, 5), (1, 'executiva', 4, 5), (1, 'executiva', 5, 5), (1, 'executiva', 6, 5), (2, 'económica', 7, 5), (2, 'económica', 8, 5), (2, 'económica', 9, 5), (2, 'económica', 10, 5), (2, 'económica', 11, 5), (2, 'económica', 12, 5), (2, 'económica', 13, 5), (2, 'económica', 14, 5), (2, 'económica', 15, 5), (1, 'executiva', 1, 6), (1, 'executiva', 2, 6), (1, 'executiva', 3, 6), (1, 'executiva', 4, 6), (1, 'executiva', 5, 6), (1, 'executiva', 6, 6), (2, 'económica', 7, 6), (2, 'económica', 8, 6), (2, 'económica', 9, 6), (2, 'económica', 10, 6), (2, 'económica', 11, 6), (2, 'económica', 12, 6), (2, 'económica', 13, 6), (2, 'económica', 14, 6), (2, 'económica', 15, 6),

```

```

(1, 'executiva', 1, 7),
(1, 'executiva', 2, 7),
(1, 'executiva', 3, 7),
(1, 'executiva', 4, 7),
(1, 'executiva', 5, 7),
(1, 'executiva', 6, 7),
(2, 'económica', 7, 7),
(2, 'económica', 8, 7),
(2, 'económica', 9, 7),
(2, 'económica', 10, 7),
(2, 'económica', 11, 7),
(2, 'económica', 12, 7),
(2, 'económica', 13, 7),
(2, 'económica', 14, 7),
(2, 'económica', 15, 7),
(1, 'executiva', 1, 8),
(1, 'executiva', 2, 8),
(1, 'executiva', 3, 8),
(1, 'executiva', 4, 8),
(1, 'executiva', 5, 8),
(1, 'executiva', 6, 8),
(2, 'económica', 7, 8),
(2, 'económica', 8, 8),
(2, 'económica', 9, 8),
(2, 'económica', 10, 8),
(2, 'económica', 11, 8),
(2, 'económica', 12, 8),
(2, 'económica', 13, 8),
(2, 'económica', 14, 8),
(2, 'económica', 15, 8),
(1, 'executiva', 1, 9),
(1, 'executiva', 2, 9),
(1, 'executiva', 3, 9),
(1, 'executiva', 4, 9),
(1, 'executiva', 5, 9),
(1, 'executiva', 6, 9),
(2, 'económica', 7, 9),
(2, 'económica', 8, 9),
(2, 'económica', 9, 9),
(2, 'económica', 10, 9),
(2, 'económica', 11, 9),
(2, 'económica', 12, 9),
(2, 'económica', 13, 9),
(2, 'económica', 14, 9),
(2, 'económica', 15, 9),
(1, 'executiva', 1, 10),
(1, 'executiva', 2, 10),
(1, 'executiva', 3, 10),
(1, 'executiva', 4, 10),
(1, 'executiva', 5, 10),
(1, 'executiva', 6, 10),
(2, 'económica', 7, 10),
(2, 'económica', 8, 10),
(2, 'económica', 9, 10),
(2, 'económica', 10, 10),
(2, 'económica', 11, 10),
(2, 'económica', 12, 10),
(2, 'económica', 13, 10),
(2, 'económica', 14, 10),
(2, 'económica', 15, 10);

```

Figura 33 – Lugar (Povoamento).

6.3. Transações

Neste tópico vamos analisar as transações criadas na nossa Base de Dados.

Para a criação destas transações tivemos em conta a simplicidade das mesmas, isto é, não criamos transações bastante complexas que justificassem o surgimento de impedimentos na execução de quaisquer *query* da nossa plataforma.

Assumimos também que a afluência à plataforma estaria entre os parâmetros normais, logo não existiram períodos de grande afluência.

6.3.1 Inserir novo cliente

```
-- transação que permite inserir um novo cliente
delimiter $$

CREATE PROCEDURE Inserir_Cliente(IN EmailC VARCHAR(45), IN NIFC INT, IN NomeC VARCHAR(45), IN SexoC ENUM('M','F'), IN DataN DATE, IN Pass VARCHAR(45), IN Telf INT, IN Telm INT)
BEGIN
START TRANSACTION;
INSERT INTO Cliente
(Email, NIF, Nome, Sexo, DataNascimento, Password, Telefone, Telemovel)
VALUES (EmailC, NIFC, NomeC, SexoC, DataN, Pass, Telf, Telm);

SELECT * FROM Cliente WHERE email = EmailC;

ROLLBACK;

END $$
CALL Inserir_Cliente('ola@gmail.com', 99999999, 'Joaquim', 'M', '1984-11-15', 'lol', 253234999, 967948888);
```

Figura 34 – Transação 1.

6.3.2 Alterar nome e password

```
-- transação que permite alterar a password e o nome de um cliente
delimiter $$

CREATE PROCEDURE Alterar_Dados(IN EmailC VARCHAR(45), IN NomeC VARCHAR(45), IN Pass VARCHAR(45))
BEGIN
START TRANSACTION;
UPDATE Cliente
SET nome = NomeC
WHERE email=emailC;
UPDATE Cliente
SET password = Pass
WHERE email=emailC;
SELECT * FROM Cliente WHERE email =emailC;

ROLLBACK;

END $$
CALL Alterar_Dados('maria@gmail.com', 'Maria', 'olaola');
```

Figura 35 – Transação 2.

6.3.3 Inserir um novo endereço num cliente

```
-- transação que permite inserir um novo endereços de um cliente
delimiter $$ 
CREATE PROCEDURE Inserir_Endereco(IN EmailC VARCHAR(45), IN CodP INT, IN Loc VARCHAR(45), IN R VARCHAR(45))
BEGIN
    START TRANSACTION;
        UPDATE Endereco
        SETCodigoPostal = CodP
        WHERE Cliente_Email=EmailC;
        UPDATE Endereco
        SET Localidade = Loc
        WHERE Cliente_Email=EmailC;

    SELECT * FROM Endereco WHERE Cliente_Email=emailC;

    ROLLBACK;
END $$ 
CALL Inserir_Endereco('maria@gmail.com', '100000000', 'Vila Nova', 'Rua Nova');
```

Figura 36 – Transação 3.

6.3.4 Inserir nova reserva

```
-- transação que permite a um cliente reservar uma viagem para uma determinada data e num determinado lugar do comboio
delimiter $$ 
CREATE PROCEDURE Reserva_vitagem(IN idReserva INT, IN numeroBilhete INT, IN custoN DECIMAL(5,2), IN custoD DECIMAL(5,2), IN EmailC VARCHAR(45), IN DataC DATE,
IN DataP DATE, IN idViagem INT, IN nLugar INT)
BEGIN
    START TRANSACTION;

    INSERT INTO Reserva
    (idR, Data, Cliente_Email)
    VALUES
    (idReserva, CURDATE(), EmailC);

    SELECT *
    FROM Reserva
    WHERE cliente_email = EmailC;

    INSERT INTO Bilhete
    (NrBilhete, DataChegada, DataPartida, CustoNormal, CustoDesconto, NumeroB, CarruagemB, ClasseB, Viagem_idV, Reserva_idR)
    VALUES
    (numeroBilhete, DataC, DataP, custoN, custoD, nLugar, CASE WHEN nLugar between 1 AND 6 THEN 1 ELSE 2 END,
    CASE WHEN nLugar between 1 AND 6 then 'executiva' else 'económica' end, idViagem, idReserva);

    SELECT *
    FROM Bilhete
    WHERE Reserva_idR = idReserva;

    ROLLBACK;
END $$ 
CALL Reserva_vitagem(28, 80, 25, 22.5, 'pedrodias@hotmail.com', CURDATE(), CURDATE(), 91, 1);
```

Figura 37 – Transação 4.

6.3.5 Inserir novo comboio

```
-- inserir novo comboio e associá-lo a uma viagem existente
delimiter $$

CREATE PROCEDURE Insere_comboio(IN idComboio INT, IN LotacaoC INT, IN idViagem INT)
BEGIN
    START TRANSACTION;

    SELECT * FROM Viagem WHERE idv = idViagem;

    INSERT INTO Comboio
    (idC, Lotacao)
    VALUES
    (idComboio, LotacaoC);

    INSERT INTO Lugar
    (Carruagem, Classe, Numero, Comboio_idC)
    VALUES
    (1, 'executiva', 1, idComboio),
    (1, 'executiva', 2, idComboio),
    (1, 'executiva', 3, idComboio),
    (1, 'executiva', 4, idComboio),
    (1, 'executiva', 5, idComboio),
    (1, 'executiva', 6, idComboio),
    (2, 'económica', 7, idComboio),
    (2, 'económica', 8, idComboio),
    (2, 'económica', 9, idComboio),
    (2, 'económica', 10, idComboio),
    (2, 'económica', 11, idComboio),
    (2, 'económica', 12, idComboio),
    (2, 'económica', 13, idComboio),
    (2, 'económica', 14, idComboio),
    (2, 'económica', 15, idComboio);

    UPDATE Viagem
    SET Comboio_idC = idComboio
    WHERE idV = idViagem;

    SELECT * FROM Viagem WHERE comboio_idc = idComboio;

    ROLLBACK;

END $$

CALL Insere_comboio(11, 15, 21);
```

Figura 38 – Transação 5.

6.3.6 Inserir nova viagem

Figura 39 – Transação 6.

```
-- transação que permite inserir uma viagem para um comboio que já existe
delimiter $$

CREATE PROCEDURE Inserir_Viagem(IN idViagem INT, IN OrigemV VARCHAR(45), IN DestinoV VARCHAR(45), IN HPartida TIME, IN HChegada TIME, Comboio INT)
BEGIN
    START TRANSACTION;
    INSERT INTO Viagem
    (idV, Origem, Destino, HorarioPartida, HorarioChegada, Comboio_idC)
    VALUES
    (idViagem, OrigemV, DestinoV, HPartida, HChegada, Comboio);

    SELECT * FROM Viagem WHERE idV=idViagem;

    ROLLBACK;

END $$

CALL Inserir_Viagem(221, 'Coimbra', 'Viana do Castelo', '130000', '143000', 9);
```

6.4. Organização de Ficheiros

Neste modelo não alteramos a organização dos ficheiros na BD, ou seja, utilizamos a configuração por defeito do MySQL Workbench, a configuração em B-Tree.

6.5. Índices

Quanto a este tópico, para além dos índices criados no Workbench no momento da geração do código SQL através do *Forward Engineer*, não foram criados quaisquer outros índices.

6.6. Vistas de Utilizadores

Neste esquema físico decidimos atribuir diferentes vistas para guardar algumas tabelas úteis que poderão ser consultadas frequentemente pelos vários tipos de utilizadores.

6.6.1 Vistas de Administradores

```
-- bilhete com custo mínimo
CREATE VIEW BilheteMinimo AS
SELECT MIN(CASE WHEN R.Data<B.DataPartida THEN B.CustoDesconto ELSE B.CustoNormal END) AS BilheteMaisBarato
FROM Bilhete AS B INNER JOIN Reserva AS R;
```

Figura 40 – View 1 (Admin).

```
-- bilhete com custo máximo
CREATE VIEW BilheteMáximo AS
SELECT MAX(CASE WHEN R.Data<B.DataPartida THEN B.CustoDesconto ELSE B.CustoNormal END) AS BilheteMaisBarato
FROM Bilhete AS B INNER JOIN Reserva AS R;
```

Figura 41 – View 2 (Admin).

```
-- 4 bilhetes com menor custo
CREATE VIEW 4BilhetesMínimos AS
SELECT DISTINCT nrbilhete, (CASE WHEN R.Data<B.DataPartida THEN B.CustoDesconto ELSE B.CustoNormal END) AS custo
FROM Bilhete AS B INNER JOIN Reserva AS R
ORDER BY custo
LIMIT 4;
```

Figura 42 – View 3 (Admin).

```
-- junção da tabela Cliente com os seus Endereços
CREATE VIEW ClienteEOSeuEndereço AS
SELECT *
FROM Cliente AS C INNER JOIN Endereco AS E
ON C.email=E.cliente_email;
```

Figura 43 – View 4 (Admin).

```
-- criação de uma tabela auxiliar com bilhetes da classe executiva
CREATE VIEW Normal AS
SELECT nrbilhete, DataChegada, DataPartida, CustoNormal AS Custo, numeroB, carruagemB, classeB, Viagem_idV, Reserva_idR
FROM Bilhete
WHERE ClasseB='executiva';
```

Figura 44 – View 5 (Admin).

```
-- criação de uma tabela auxiliar com bilhetes da classe económica
CREATE VIEW Barato AS
SELECT nrbilhete, DataChegada, DataPartida, CustoNormal AS Custo, numeroB, carruagemB, classeB, Viagem_idV, Reserva_idR
FROM Bilhete
WHERE ClasseB='económica';
```

Figura 45 – View 6 (Admin).

```
-- custo real bilhete
CREATE VIEW CustoBilhete AS
SELECT (CASE WHEN R.Data<B.DataPartida THEN B.CustoDesconto ELSE B.CustoNormal END) AS CustoReal, CustoNormal, idR, Cliente_Email, Data, DataPartida
FROM Reserva AS R INNER JOIN Bilhete AS B
WHERE R.idR=B.Reserva_idR;
```

Figura 46 – View 7 (Admin).

```
-- montante de cada reserva
CREATE VIEW MontanteReserva AS
SELECT (SUM(CASE WHEN R.Data<B.DataPartida THEN B.CustoDesconto ELSE B.CustoNormal END)) AS CustoPorReserva, idR, Cliente_Email
FROM Reserva AS R INNER JOIN Bilhete AS B
ON R.idr=B.reserva_idr
INNER JOIN Cliente AS C
ON R.cliente_email = C.email
GROUP BY idr;
```

Figura 47 – View 8 (Admin).

```
-- montante a pagar por cada cliente pelas suas reservas
CREATE VIEW CustoReservasCliente AS
SELECT (SUM(CASE WHEN R.Data<B.DataPartida THEN B.CustoDesconto ELSE B.CustoNormal END)) AS CustoPorCliente, Nome, Cliente_Email
FROM Reserva AS R INNER JOIN Bilhete AS B
ON R.idr=B.reserva_idr
INNER JOIN Cliente AS C
ON R.cliente_email = C.email
GROUP BY email;
```

Figura 48 – View 9 (Admin).

```
-- lucro total de cada viagem
CREATE VIEW LucroViagem AS
SELECT (SUM(CASE WHEN R.Data<B.DataPartida THEN B.CustoDesconto ELSE B.CustoNormal END)) AS LucroDeCadaViagem, idV, comboio_idc
FROM Reserva AS R INNER JOIN Bilhete AS B
ON R.idr=B.reserva_idr
INNER JOIN Viagem AS V
ON B.Viagem_idV=V.idV
GROUP BY idv;
```

Figura 49 – View 10 (Admin).

```
-- lucro total de todas as viagens
CREATE VIEW LucroViagens AS
SELECT (SUM(CASE WHEN R.Data<B.DataPartida THEN B.CustoDesconto ELSE B.CustoNormal END)) AS LucroTotal
FROM Reserva AS R INNER JOIN Bilhete AS B
ON R.idr=B.reserva_idr
INNER JOIN Viagem AS V
ON B.Viagem_idV=V.idV;
```

Figura 50 – View 11 (Admin).

```
-- numero total de utilizadores que compram bilhete
CREATE VIEW NúmeroUtilizadoresCompramBilhete AS
SELECT COUNT(*)
FROM (SELECT DISTINCT email
      FROM Cliente AS C INNER JOIN Reserva AS R
      ON C.email=R.Cliente_Email) AS E;
```

Figura 51 – View 12 (Admin).

```
-- id reservas feitas por um utilizador
CREATE VIEW ReservasDeUtilizador AS
SELECT idR, Cliente_Email
FROM Reserva AS R INNER JOIN Cliente AS C
WHERE R.Cliente_Email=C.Email
ORDER BY email;
```

Figura 52 – View 13 (Admin).

```
-- numero de reservas feitas por cada utilizador
CREATE VIEW NúmeroReservasUtilizador AS
SELECT nome, COUNT(idr)
  FROM Reserva AS R INNER JOIN Cliente AS C
  ON R.cliente_email=C.email
 GROUP BY nome;
```

Figura 53 – View 14 (Admin).

```
-- quantidade de bilhetes comprados por cada utilizador
CREATE VIEW QuantidadeBilhetesUtilizador AS
SELECT nome, COUNT(nrbilhete)
  FROM Bilhete AS B INNER JOIN Reserva AS R
  ON B.reserva_idr=R.idr
 INNER JOIN Cliente AS C
  ON R.cliente_email=C.email
 GROUP BY nome;
```

Figura 54 – View 15 (Admin).

```
-- bilhetes comprados por cada utilizador e origem e destino de cada um dos bilhetes
CREATE VIEW BilhetesCompradosUtilizador AS
SELECT nome, nrbilhete, origem, destino
  FROM Viagem AS V INNER JOIN Bilhete AS B
  ON V.idv=B.viagem_idv
 INNER JOIN Reserva AS R
  ON B.reserva_idr=R.idr
 INNER JOIN Cliente AS C
  ON R.cliente_email=C.email;
```

Figura 55 – View 16 (Admin).

```
-- comboio destinado a cada bilhete e origem e destino da viagem correspondente a esse bilhete
CREATE VIEW ComboioBilhete AS
SELECT nrbilhete, comboio_idc, origem, destino
  FROM Bilhete AS B INNER JOIN Viagem AS V
  ON B.viagem_idv=V.idv
 ORDER BY nrbilhete;
```

Figura 56 – View 17 (Admin).

6.6.2 Vistas de Clientes

```
-- viagens e comboios respetivos
CREATE VIEW ViagemComboio AS
SELECT idv, comboio_idc, origem, destino FROM Viagem;
```

Figura 57 – View 1 (Cliente).

```
-- todas viagens com origens e destinos correspondentes
CREATE VIEW Viagens AS
SELECT idv, nrbilhete, comboio_idc, origem, destino
  FROM Bilhete AS B INNER JOIN Viagem AS V
  ON B.viagem_idv=V.idv
 ORDER BY idv;
```

Figura 58 – View 2 (Cliente).

```
-- viagens e horários respetivos
CREATE VIEW ViagensHorários AS
SELECT idv, horariopartida, horariochegada, origem, destino
  FROM Viagem;
```

Figura 59 – View 3 (Cliente).

```
-- horário correspondente a cada bilhete
CREATE VIEW BilheteHorário AS
SELECT nrbilhete, horariopartida, horariochegada, origem, destino, datapartida, datachegada, idv
  FROM Viagem AS V INNER JOIN Bilhete AS B
  ON V.idv = B.viagem_idv
 ORDER BY nrbilhete;
```

Figura 60 – View 4 (Cliente).

```
-- lugares ocupados em cada comboio numa viagem
CREATE VIEW LugaresOcupadosViagem AS
SELECT idc, numerob, carruagemb, classeb, nrbilhete, idv, origem, destino
  FROM Comboio AS C INNER JOIN Viagem AS V
  ON C.idc = V.comboio_idc
    INNER JOIN Bilhete AS B
    ON V.idv = B.viagem_idv
 ORDER BY idc;
```

Figura 61 – View 5 (Cliente).

```
-- numero de lugares ocupados em cada comboio numa viagem
CREATE VIEW NúmeroLugaresOcupadosViagem AS
SELECT idc, count(numerob), idv, origem, destino
  FROM Comboio AS C INNER JOIN Viagem AS V
  ON C.idc = V.comboio_idc
    INNER JOIN Bilhete AS B
    ON V.idv = B.viagem_idv
 GROUP BY idc;
```

Figura 62 – View 6 (Cliente).

```
-- numero de lugares livres num comboio de uma dada viagem
CREATE VIEW NúmeroLugaresLivresViagem AS
SELECT idc, idv, (Lotacao-COUNT(numero)) AS LugLiv, DataPartida, HorarioPartida, DataChegada, HorarioChegada, Origem, Destino
FROM Comboio AS C INNER JOIN Viagem AS V
ON C.idc = V.comboio_idc
INNER JOIN Bilhete AS B
ON V.idv = B.viagem_idv
GROUP BY NrBilhete
Order By idC;
```

Figura 63 – View 7 (Cliente).

```
-- lugares livres em cada comboio
CREATE VIEW LugaresLivresComboio AS
SELECT idc, nrbilhete, numerob, origem, destino, datapartida, datachegada, idv, numero, carruagem
FROM Lugar AS L INNER JOIN Comboio C
ON L.comboio_idc = C.idc
INNER JOIN Viagem AS V
ON C.idc = V.comboio_idc
INNER JOIN Bilhete AS B
ON V.idv = B.viagem_idv
WHERE numerob <> numero;
```

Figura 64 – View 8 (Cliente).

6.7. Criação de Utilizador

```
-- Criação do utilizador 'bossBD'  
CREATE USER 'bossBD'@'localhost';  
SET PASSWORD FOR 'bossBD'@'localhost' = 'euamome1234';  
  
SELECT *  
  FROM mysql.user  
 WHERE User = 'bossBD';  
  
-- permissão de acesso a todos os objetos de todas as bases de dados  
GRANT ALL ON comboioslusitanos.* TO 'bossBD'@'localhost';  
  
SHOW GRANTS FOR 'bossBD'@'localhost';  
  
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'bossBD'@'localhost';
```

Figura 65 – Administrador.

6.8. Estimativa do espaço ocupado em disco pela base de dados

Ao construir uma base de dados é necessário ter em consideração o espaço ocupado pela mesma, pois é importante que possamos desenvolver um modelo que seja extensível e onde tenhamos a capacidade de desenvolver ainda mais o modelo de modo a suportar novos requisitos com o menos efeito possível sobre os utilizadores existentes. Por isso, é essencial o cálculo de uma estimativa realista de ocupação de dados tanto da implementação existente como de futuras implementações.

Vamos realizar então o cálculo de espaço ocupado por cada tabela do modelo, mais concretamente, por cada atributo que as compõem.

6.8.1 Tamanho ocupado pelos atributos da tabela Cliente

Entidade	Atributos	Tipo de Dados	Tamanho
Cliente	Email	VARCHAR(45)	45 bytes
	NIF	INT	4 bytes

	Nome	VARCHAR(45)	45 bytes
	Sexo	ENUM('M','F')	4 bytes
	DataNascimento	DATE	4 bytes
	Password	VARCHAR(45)	45 bytes
	Telefone	INT	4 bytes
	Telemovel	INT	4 bytes

Tabela 7 – Espaço ocupado pelos atributos da Tabela Cliente.

Espaço estimado por registo = $45 + 4 + 45 + 4 + 4 + 45 + 4 + 4 = 155$

Espaço estimado total inicial (50 registos iniciais) = $155 * 50 = 7750$

Prevê-se que este número aumente no futuro com o aumento de regtos na plataforma e consequentemente o espaço ocupado pelo resto das tabelas também aumente gradualmente.

6.8.2 Tamanho ocupado pelos atributos da tabela Endereço

Entidade	Atributos	Tipo de Dados	Tamanho
Endereço	CodigoPostal	INT	4 bytes
	Localidade	VARCHAR(45)	45 bytes
	Rua	VARCHAR(45)	45 bytes

Tabela 8 – Espaço ocupado pelos atributos da Tabela Endereço.

Espaço estimado por registo = $4 + 45 + 45 = 94$

Espaço estimado total inicial (50 registos iniciais) = 4700

6.8.3 Tamanho ocupado pelos atributos da tabela Reserva

Entidade	Atributos	Tipo de Dados	Tamanho
Reserva	idR	INT	4 bytes
	Data	DATE	4 bytes

Tabela 9 – Espaço ocupado pelos atributos da Tabela Reserva.

Espaço estimado por registo = $4 + 4 = 8$

Espaço estimado total inicial (50 registos iniciais) = $8 * 25 = 200$

6.8.4 Tamanho ocupado pelos atributos da tabela Bilhete

Entidade	Atributos	Tipo de Dados	Tamanho
Bilhete	NrBilhete	INT	4 bytes
	DataChegada	DATE	4 bytes
	DataPartida	DATE	4 bytes
	CustoNormal	DECIMAL(5,2)	8 bytes
	CustoDesconto	DECIMAL(5,2)	8 bytes
	NumeroB	INT	4 bytes
	CarruagemB	INT	4 bytes
	ClasseB	VARCHAR(45)	45 bytes

Tabela 10 – Espaço ocupado pelos atributos da Tabela Bilhete.

Espaço estimado por registo = $4 + 4 + 4 + 8 + 8 + 4 + 4 + 45 = 81$

Espaço estimado total inicial (40 registos iniciais) = $81 * 40 = 3240$

6.8.5 Tamanho ocupado pelos atributos da tabela Viagem

Entidade	Atributos	Tipo de Dados	Tamanho
Viagem	idV	INT	4 bytes
	Origem	VARCHAR(45)	45 bytes
	Destino	VARCHAR(45)	45 bytes
	HorarioPartida	TIME	5 bytes
	HorarioChegada	TIME	5 bytes

Tabela 11 – Espaço ocupado pelos atributos da Tabela Viagem.

Espaço estimado por registo = $4 + 45 + 45 + 5 + 5 = 104$

Espaço estimado total inicial (40 registos iniciais) = $104 * 40 = 4160$

6.8.6 Tamanho ocupado pelos atributos da tabela Comboio

Entidade	Atributos	Tipo de Dados	Tamanho
Comboio	idC	INT	4 bytes
	Lotocao	INT	4 bytes

Tabela 12 – Espaço ocupado pelos atributos da Tabela Comboio.

Espaço estimado por registo = $4 + 4 = 8$

Espaço estimado total inicial (10 registos iniciais) = $8 * 10 = 80$

6.8.7 Tamanho ocupado pelos atributos da tabela Lugar

Entidade	Atributos	Tipo de Dados	Tamanho
Lugar	Carruagem	ENUM('1','2')	4 bytes
	Classe	ENUM('económica','executiva')	16 bytes
	Numero	INT	4 bytes

Tabela 13 – Espaço ocupado pelos atributos da Tabela Lugar.

Espaço estimado por registo = $4 + 16 + 4 = 24$

Espaço estimado total inicial (150 regtos iniciais) = $24 * 150 = 3600$

7. Conclusão

Em suma, a realização deste trabalho prático permitiu aprofundar os nossos conhecimentos sobre os conteúdos lecionados nas aulas e consolidamos competências relacionadas com o processo de criação e desenvolvimento de um Sistema de Gerenciamento de Bases de Dados. Com a apresentação do tema a desenvolver por parte do docente da unidade curricular, executamos o levantamento e análise de requisitos necessários para a base de dados, tendo reconhecido que haviam várias entidades fundamentais para a correta implementação de um sistema de reservas de viagens em comboios nacionais e internacionais. Após a definição das diversas entidades, foram selecionados os respetivos atributos e os relacionamentos entre as entidades. Com todos estes, e após várias confirmações com o docente, construiu-se o modelo conceptual.

Seguidamente, na modelação lógica procedeu-se à transformação do modelo conceptual para o modelo lógico, usando a ferramenta *Workbench*.

Por fim, foi implementado o modelo físico, onde povoamos a base de dados, entre outros.

Para concluir, estamos cientes da existência de alterações possíveis a pequenos detalhes que permitiriam encaixar novas possibilidades de desenvolvimento e corrigir alguns erros de idealização e modulação. Por exemplo, as classes económica e executiva têm os mesmos preços de bilhete, algo que não acontece na realidade, mas que não foi identificado nos nossos requisitos em tempo útil para que fosse implementado.

Referências

- [1] Thomas M. Connolly and Carolyn E. Begg, *Database Systems: A practical approach to Design, Implementation and Management*, 6th Ed. New York, USA: Pearson, 2014.

Lista de Siglas e Acrónimos

MV Multi-Valor

BD Base de dados

SGDB Sistema de Gestão de Base de Dados

SQL *Structured Query Language*

Anexos

Os anexos abaixo apresentados incluem informação adicional sobre os *procedures* para complementar este tópico abordado.

I. Procedure 1

```
delimiter $$  
CREATE PROCEDURE UtilizadoresDeUmDadosSexo (IN sex CHAR(1))  
BEGIN  
    SELECT Email, nome, DataNascimento  
    FROM cliente  
    WHERE sexo = sex;  
END $$  
CALL UtilizadoresDeUmDadosSexo ("M");
```

Figura 66 – UtilizadoresDeUmDadosSexo.

II. Procedure 2

```
delimiter $$  
CREATE PROCEDURE UtilizadoresDeUmDadosSexoComRestriçãoDeData (IN sex CHAR(1), dat1 Date)  
BEGIN  
    SELECT Email, nome, DataNascimento  
    FROM cliente  
    WHERE sexo = sex AND DATE(datanascimento) <> dat1;  
END $$  
CALL UtilizadoresDeUmDadosSexoComRestriçãoDeData ("F", '1994-07-02');
```

Figura 67 – UtilizadoresDeUmDadosSexoComRestriçãoDeData.

III. Procedure 3

```
delimiter $$  
CREATE PROCEDURE DataPartidaCustoComDescontoAtravésDeBilhetesDados (IN a1 Int, a2 Int, a3 Int, a4 Int, a5 Int)  
BEGIN  
    SELECT datapartida, custodesconto, nrbilhete  
    FROM bilhete  
    WHERE nrbilhete IN (a1,a2,a3,a4,a5)  
    ORDER BY nrbilhete DESC;  
END $$  
CALL DataPartidaCustoComDescontoAtravésDeBilhetesDados (9,2,19,10,8);
```

Figura 68 – DataPartidaCustoComDescontoAtravésBilhetesDados.

IV. Procedure 4

```
delimiter $$  
CREATE PROCEDURE SaberDadosDeClienteDeCertaCidade (IN cid CHAR(20))  
BEGIN  
    SELECT nome, codigopostal, rua, localidade  
    FROM endereco AS B INNER JOIN cliente AS C  
    WHERE B.Cliente_Email=C.Email AND localidade = cid;  
END $$  
CALL SaberDadosDeClienteDeCertaCidade ("Braga");
```

Figura 69 – SaberDadosDeClienteDeCertaCidade.

V. Procedure 5

```
-- número de bilhetes vendidos num intervalo de tempo
delimiter $$

CREATE PROCEDURE NºdeBilhetesVendidosNumIntervaloDeTempo (IN dat1 DATE, dat2 DATE)
BEGIN
    SELECT COUNT(nrbilhete)
    FROM Bilhete AS B INNER JOIN Reserva AS R
        ON B.Reserva_idR=R.idR
        WHERE R.Data between dat1 and dat2
        ORDER BY nrbilhete;
END $$

CALL NºdeBilhetesVendidosNumIntervaloDeTempo ('2016-11-11', '2016-11-12');
```

Figura 70 – NºdeBilhetesVendidosNumIntervaloDeTempo.

VI. Procedure 6

```
-- número de cada bilhete vendido num intervalo de tempo
delimiter $$

CREATE PROCEDURE NºdosBilhetesVendidosNumIntervaloDeTempo (IN dat1 DATE, dat2 DATE)
BEGIN
    SELECT nrbilhete
    FROM Bilhete AS B INNER JOIN Reserva AS R
    ON B.Reserva_idR=R.idR
    WHERE R.Data between dat1 and dat2
    ORDER BY nrbilhete;
END $$

CALL NºdosBilhetesVendidosNumIntervaloDeTempo ('2016-11-11', '2016-11-12');
```

Figura 71 – NºdosBilhetesVendidosNumIntervaloDeTempo.

VII. Procedure 7

```
-- montante a pagar por uma dada reserva
delimiter $$

CREATE PROCEDURE MontanteAPagarPelaReservaDeUmaDadaidR (IN idr1 INT)
BEGIN
    SELECT (SUM(CASE WHEN R.Data < B.DataPartida THEN B.CustoDesconto ELSE B.CustoNormal END)) AS CustoDeUmaReserva, Nome, Cliente_Email
    FROM Reserva AS R INNER JOIN Bilhete AS B
    ON R.idr=B.reserva_idr
    INNER JOIN Cliente AS C
    ON R.cliente_email = C.email
    WHERE idr = idr1;
END $$
CALL MontanteAPagarPelaReservaDeUmaDadaidR (11);
```

Figura 72 – MontanteAPagarPelaReservaDeUmaDataidR.

VIII. Procedure 8

```
-- lucro total de uma viagem
delimiter $$

CREATE PROCEDURE LucroTotalDeDadaViagem (IN idv2 INT)
BEGIN
    SELECT (SUM(CASE WHEN R.Data<B.DataPartida THEN B.CustoDesconto ELSE B.CustoNormal END)) AS LucroDeUmaViagem, idV, comboio_idc
    FROM Reserva AS R INNER JOIN Bilhete AS B
    ON R.idr=B.reserva_idr
    INNER JOIN Viagem AS V
    ON B.Viagem_idV=V.idV
    WHERE idV = idv2;
END $$
CALL LucroTotalDeDadaViagem (11);
```

Figura 73 – LucroTotalDeDadaViagem.

IX. Procedure 9

```
-- Numero de reservas feitas por um dado utilizador ( -- para a transação em que utilizador verifica quantas reservas efetuou)
delimiter $$

CREATE PROCEDURE NumeroReservasFeitasPorDadoNome (IN nout CHAR (20))
BEGIN
    SELECT nome, COUNT(idr)
    FROM Reserva AS R INNER JOIN Cliente AS C
    ON R.cliente_email=C.email
    WHERE nome = nout;
END $$

CALL NumeroReservasFeitasPorDadoNome ("Sofia");
```

Figura 74 – NumeroReservasFeitasPorDadoNome.

X. Procedure 10

```
-- numero de bilhetes comprados por um dado utilizador
delimiter $$

CREATE PROCEDURE NumeroBilhetesCompradosPorDadoNome (IN utBilh CHAR (20))
BEGIN
    SELECT nome, COUNT(nrbilhete)
    FROM Bilhete AS B INNER JOIN Reserva AS R
    ON B.reserva_idr=R.idr
    INNER JOIN Cliente AS C
    ON R.cliente_email=C.email
    WHERE nome = utBilh;
END $$

CALL NumeroBilhetesCompradosPorDadoNome ("José");
```

Figura 75 – NumeroBilhetesCompradosPorDadoNome.

XI. Procedure 11

```
-- comboios e bilhetes associados a uma dada viagem
delimiter $$

CREATE PROCEDURE ComboiosEBilhetesAssociadosAUmDadoIdv (IN idvv INT)
BEGIN
    SELECT idv, nrbilhete, comboio_idc, origem, destino
    FROM Bilhete AS B INNER JOIN Viagem AS V
    ON B.viagem_idv=V.idv
    WHERE idv=idvv;
END $$

CALL ComboiosEBilhetesAssociadosAUmDadoIdv (11);
```

Figura 76 – ComboiosEBilhetesAssociadosAUmDadoIdv.

XII. Procedure 12

```
delimiter $$  
CREATE PROCEDURE ComboiosEBilhetesAssociadosAUmaDadaOrigemOuDestino (IN cbav1 CHAR(20), cbav2 CHAR(20))  
BEGIN  
    SELECT idv, nrbilhete, comboio_idc, origem, destino  
    FROM Bilhete AS B INNER JOIN Viagem AS V  
    ON B.viagem_idv=V.idv  
    WHERE origem = cbav1 OR destino = cbav2  
    ORDER BY origem;  
END $$  
CALL ComboiosEBilhetesAssociadosAUmaDadaOrigemOuDestino ("Braga", "Madrid");
```

Figura 77 – ComboiosEBilhetesAssociadosAUmaDadaOrigemOuDestino.

XIII. Procedure 13

```
delimiter $$  
CREATE PROCEDURE ComboiosDeDeterminadaViagem (IN cdo CHAR(20), cdd CHAR(20))  
BEGIN  
    SELECT idv, comboio_idc, origem, destino  
    FROM Viagem  
    WHERE origem = cdo AND destino = cdd;  
END $$  
Call ComboiosDeDeterminadaViagem ("Braga", "Porto")
```

Figura 78 – ComboiosDeDeterminadaViagem.

XIV. Procedure 14

```
-- numero de lugares ocupados num dado comboio numa viagem
delimiter $$

CREATE PROCEDURE N°deLugaresOcupadosNumaDadaViagem (IN ido INT)
BEGIN
    SELECT idc, idv, count(numerob) AS NumeroDeOcupados, DataPartida, HorarioPartida, DataChegada, HorarioChegada, Origem, Destino, Lotacao
    FROM Comboio AS C INNER JOIN Viagem AS V
    ON C.idc = V.comboio_idc
    INNER JOIN Bilhete AS B
    ON V.idv = B.viagem_idv
    WHERE idc = ido;
END $$

CALL N°deLugaresOcupadosNumaDadaViagem (5);
```

Figura 79 – N°deLugaresOcupadosNumaDadaViagem.

XV. Procedure 15

```
-- numero de lugares livres num dado comboio numa viagem (para a transação do numero de lugares livres de um comboio para uma viagem)
delimiter $$

CREATE PROCEDURE NºdeLugaresLivresNumaDadaViagem (IN id INT)
BEGIN
    SELECT idc, idv, ((lotacao)-count(numero)) AS LugLiv, DataPartida, HorarioPartida, DataChegada, HorarioChegada, Origem, Destino, Lotacao
    FROM Comboio AS C INNER JOIN Viagem AS V
    ON C.idc = V.comboio_idc
    INNER JOIN Bilhete AS B
    ON V.idv = B.Viagem_idv
    WHERE C.idc=id;
END $$

CALL NºdeLugaresLivresNumaDadaViagem (8);
```

Figura 80 – NºdeLugaresLivresNumaDadaViagem.

XVI. Procedure 16

```
-- Saber os destinos de uma certa origem
delimiter $$

CREATE PROCEDURE DestinosDeUmaDadaViagem (IN part CHAR(20))
BEGIN
    SELECT Viagem.Origem, Viagem.Destino
        FROM Viagem
       WHERE Viagem.Origem = part;
END $$

CALL DestinosDeUmaDadaViagem ("Barcelona");
```

Figura 81 – DestinosDeUmaDadaViagem.

XVII. Procedure 17

```
-- Saber de onde partem os comboios para um certo destino
delimiter $$

CREATE PROCEDURE OrigensDeUmDadoDestino (IN dest CHAR(20))
BEGIN
    SELECT Viagem.Origem, Viagem.Destino
        FROM Viagem
       WHERE Viagem.Destino = dest;
END $$

CALL OrigensDeUmDadoDestino ("Braga");
```

Figura 82 – OrigensDeUmDadoDestino.