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

Unidade Curricular de Bases de Dados

Ano Lectivo de 2018/2019

Gestão de um sistema de autoestradas

João Nunes – A82300

Luís Braga – A82088

Luís Martins – A82298

Flávio Martins – A65277

Novembro, 2018

Data de Recepção	
Responsável	
Avaliação	
Observações	

Gestão de um sistema de autoestradas

João Nunes – A82300

Luís Braga – A82088

Luís Martins – A82298

Flávio Martins – A65277

Novembro, 2018

Resumo

Este relatório é o resultado do trabalho elaborado no âmbito da Unidade Curricular de Base de Dados, no qual o grupo chegou a um consenso e escolheu o tema “Gestão de um sistema de autoestradas”. Ao longo deste relatório serão apresentados todos os passos que tivemos de efetuar desde a criação do Modelo Conceptual até à implementação do Modelo Físico.

Inicialmente é apresentada uma breve introdução juntamente com uma pequena contextualização do nosso projeto, o caso de estudo em questão bem como a motivação do projeto e os seus principais objetivos, também é analisada a viabilidade do projeto.

De seguida, é feita uma análise de requisitos, onde são expostas as entidades que serão essenciais para a construção do projeto.

Na apresentação do Modelo Conceptual são descritas as entidades e relacionamento envolvidos, bem como os respetivos atributos. Para cada entidade, é apresentada informação sobre a sua descrição e ocorrência sendo possível portanto, identificar as chaves primárias bem como as chaves estrangeiras. Para cada atributo é apresentado o domínio de valores possíveis.

Na transição do Modelo Conceptual para Lógico é abordado novamente os relacionamentos e o domínio de atributos, mas desta vez de forma mais aprofundada. É abordada a normalização das tabelas, onde é justificada a normalização do nosso modelo até à terceira forma normal, o modelo também é validado segundo as transações estabelecidas, interrogações do utilizador e com o utilizador em si.

Passando para o Modelo Físico, é apresentada a tradução do Modelo Lógico para um esquema que já pode ser realmente usado como um SGBD, todo ele em código SQL, nesta fase são elaboradas as respostas às interrogações identificadas. É feita também uma estimativa dos requisitos de espaço em disco que poderá ser opcupado pela base de dados. Termina com uma análise de transações e regras de acesso sendo o modelo revisto uma última vez pelo utilizador.

De seguida é efetuada a migração da base de dados relacional, concebida anteriormente, para uma base de dados não relacional (MongoDB), é apresentado o esquema sobre o qual os dados irão ser inseridos após a migração, é apresentado também o código responsável pela migração e as interrogações elaborados no MongoDB.

Por último, são apresentadas as conclusões do trabalho realizado, apontando os seus pontos fortes e fracos e são também mencionadas algumas sugestões para estender o trabalho realizado num futuro próximo.

Área de Aplicação: Desenho e Arquitetura de Sistemas de Base de Dados no âmbito de uma aplicação responsável por gerir o funcionamento geral de um sistema de autoestradas.

Palavras-Chave: Base de Dados, Base de Dados Relacionais, Análise de Requisitos, Modelo Conceptual, Modelo Lógico, Modelo Físico, brModelo, MySQL Workbench, SQL, Base de Dados Não Relacional, MongoDB.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1. Contexto da aplicação do sistema

Neste novo século em que vivemos, a mudança deixou de ser um processo lento e previsível, passando a ser um fenónemo do dia-a-dia, instantâneo e imprevisível, capaz de apanhar desprevenida qualquer oganização, independentemente da sua dimensão, posicionamento ou reputação. Esta mudança de paradigmas leva ao surgimento de novos desafios e oportunidades, cujo aproveitamento têm por base três competências chave: capacidade de inovar, capacidade de controlar o funcionamento da empresa e o controlo de custos.

Tendo isto como base a GreenWay, concessionária de uma rede de autoestradas em Portugal, devido a um rápido acréscimo de automóveis a circular nas suais vias, aliado a um rápido crescimento da indústria e população, verificou um número maior de utentes a circular nas suas vias.

Como tal, decidiu criar uma base de dados para registar e controlar toda a informação referente aos sistemas de autoestradas, nomedeadamente sobre os seus utentes, mas também calcular balanços financeiros consoante a sua fonte de rendimento, os pagamentos na portagem, e os seus custos, podendo estes ser de manuntentenção ou despesas salariais.

1.2. Fundamentação da aplicação da base de dados

A empresa GreenWay necessita de guardar os seus dados de uma forma rápida e eficiente, de modo a que a procura e gestão de dados seja o mais rápida possível e, dada o enquadramento da mesma, necessita de guardar os registos dos seus utentes.

O registo irá guardar o troço de autoestrada que determinado utente percorreu, sendo o estrato depois calculado a partir da distância percorrida entre a entrada e a saída juntamente com a classe do veículo do utente e a categoria da autoestrada. O sistema também terá que ter a ambivalência de ajudar no que toca a parte financeira das autoestradas, irá ser possível calcular os ganhos apartir de todos os pagamentos efetuados nas portagens e as despesas que estão associadas à manuntenção da autoestrada juntamente com, por exemplo, a folha salarial dos funcionários.

A GreenWay, é um granda empresa no panorama nacional e como tal apenas a admistração e os funcionários das portagens têm acesso a uma interface onde a informação é apresentada de uma forma clara e concisa. A informação apresentada poderá variar desde os registos dos veículos no sistema até à folha salarial da empresa.

1.3. Motivação e Objectivos

Na realização deste projeto foi proposta ao grupo a implementação de uma BD com base num tema de trabalho escolhido pelo mesmo. Este tema apresenta alguma complexidade a nível da implementação da SBD tornando o desafio do projeto ainda mais aliciante. Guardar registos de circulação no sistema é sempre sinónimo de perca de algum tempo por parte dos funcionários a trabalhar em cada uma das portagens, com o desenvolvimento deste trabalho pretendemos simplificar toda esta parte do trabalho por parte dos funcionários, apresentando um sistema duplamente rápido e eficiente.

Logo, o desenvolvimento deste projeto pretende simplificar todo o processo desde que o utente entre no sistema até ao momento onde ele sai do sistema. Os registos, são guardados apartir do momento em que sai da autoestrada, onde é apresentada toda a informação relevante à viagem do utente da autoestrada. Para além dos registos serem mantidos ordenados cronologicamente, também se pretende que se faça um balanço contabilístico ao final de cada mês de forma a apresentar os ganhos/prejuízos ao final de cada mês.

Assim sendo, foi construída uma base de dados simples, sempre focada na realidade e nos problemas que surgem frequentemente neste tipo de sistemas, tornando depois a tarefa de gestão de dados e de contabilidade da empresa ainda mais fácil, poupando tempo aos funcionários da empresa.

1.4. Estrutura do Relatório

Após ter sido apresentado o caso de estudo escolhido e a motivação por detrás dele, nos seguintes capítulos irá ser exposta a análise e a construção do Modelo Conceptual da base de dados, a transição deste para o seu esquema Lógico e por último a implementação do esquema Físico. Estas etapas serão abordadas nos próximos capítulos do relatório.

Na análise do caso de estudo dá-se a conhecer uma breve descrição das entidades e dos relacionamentos entre elas, bem como os atribuitos que as caractereizam. A maneira como um utilizador irá interagir com a informação da base de dados é descrita sob a forma de possíveis operações de manipulação e consulta, tudo isto através de exemplos práticos.

Logo depois, é exposta a análise do Modelo Conceptual da base de dados, onde são apresentadas de forma tabular as entidades e os relacionamentos envolvidos, bem como, os respetivos atributos. Para cada entidade, é apresentada informação sobre a sua descrição e ocorrência, enquanto que, para os relacionamentos é caracterizada a sua multiplicidade entre as entidades envolvidos. Para os atributos, é identificado o tipo de dados e o seu tamanho, bem como se estes podem ser nulos, multivalorados ou derivados, após isto, determina-se também o domínio de valores possíveis. Apresenta-se para cada entidade as chaves candidatas, onde é identificada a chave primária bem como as chaves alternativas. Segue ainda a descrição das transações de informação que um funcionário e um utente poderão realizar. Por fim, fechando esta secção é apresentada a representação gráfica do Modelo Conceptual.

Passando para o Modelo Lógico, primeiramente é feita a especificação das entidades e diferentes tipos de relacionamentos entre elas, identificando-se agora as chaves estrangeiras. É aborada a normalização das tabelas, onde é justificada a normalização do nosso modelo até à terceira forma normal.

De seguida, é apresentado um retrato do Modelo Lógico, proveniente do *MySQL Workbench,* onde é possível identificar os vários relacionamentos anteriormente descritos. É verificada a integridade do modelo, identificando informação sobre os atributos, nomeadamente se são de preenchimento obrigatório e quais as restrições ao domínio de valores possíveis de atribuir. Estuda-se a multiplicidade dos relacionamentos através da contextualização das chaves primárias e estrangeiras. Justifica-se, também, a integridade referencial entre entidades, ou seja, apresenta-se as regras que mantêm a consistência de informação através das tabelas que partilham uma chave estrangeira. Para finalziar esta secção, é apresntado um pequeno estudo sobre a viabilidade do crescimento futuro da base de dados, onde é sugerida a implementação de novas funcionalidades.

Finalmente, é apresentado o Modelo Físico da base de dados. A descrição detalhada de atributos e realcionamentos já segue, desta forma, uma semântica aproximadada da linguagem SQL. É feita uma estimativa dos requisitos do espaço em disco que poderá ser ocupado pela base de dados. Termina com uma análise de registos bem como uma definição das vistas dos utilizadores e regras de acesso. Por último, são apresentadas as conclusões do trabalho realizado, bem como alguns anexos importantes.

1.5. Análise da viabilidade do projeto

Este projeto consiste na implementação de um Sistema de Base de Dados (SBD), mais concretamente um Sistema de Base de Dados Relacional (SBRD).

Uma base de dados relacional é composta por um cojunto de tabelas e de associações entre elas. A associação entre os dados é a vantagem dos modelos relacionais, uma vez que cada tabela criada com este sistema poderá ter mais que uma finalidade e os seus dados poderão ser vistos de formas e formatos diferentes.

Relativamente à implementação desta base de dados é de destacar as seguintes três vantagens mais relevantes:

1. **Resposta rápida aos pedidos de informação:** como os dados estão integrados numa única estrutura, sendo esta a base de dados, a resposta a questões mais complexas podeá ser processada numa maineira mais rápida.
2. **Acesso múltiplo:** o *software* de gestão de base de dados permite que os dados sejam acedidos de diversas maneiras onde, por exemplo, os dados poderão ser visualizados através de pesquisas sobre qualquer um dos campos da tabela.
3. **Flexibilidade:** devido à independência entre dados e programa, qualquer alteração num desses elementos não implica modificações a grande escala no outro.

A maior, e a mais grave desvantagem recai sobre o valor associado ao sistema de gestão da base de dados. O custo de trabalhar com um *software* altamente sofisticado requer um desenho e desenvolvimento com bastante cautela e ponderação, pelo que, qualquer má conceptualização poderá resultar numa construção de uma base de dados ineficaz e com consequências nefastas para uma organização.

2. Levantamento e análise de requisitos

Concluída esta etapa incial e partinda da informação apresentada nos capítulos anteriores, procede-se a fase de análise de requisitos, sendo este, uma das fases mais importantes para a elaboração da base de dados.

2.1. Metodologia e análise de requisitos adotado

A proposta de requisitos é fruto da reflexão e análise crítica por parte do grupo de trabalho, tendo sempre em mente a concretização das requisições e imposições inerentes ao caso de estudo em questão.

2.2. Requisitos de descrição

- O sistema deverá guardar o registo de um dado veículo na sua passagem na autoestrada, bem como os custos associados desta mesma autoestrada.
- A cada registo do veículo no sistema deverá ser guardado o troço de autoestrada que o utente percorreu, o montante pago pelo utente, a data inicial e final que corresponde a data em que começou a circular na autoestrada e a data em que acabou de circular. Bem como o identificador único do registo e o identificador do veículo em questão.
- Cada troço possui dois pontos que o delimitam, que correspondem à entrada e saída e ambos são descritos, um identificador e a distância entre os dois pontos para calcular o montante a pagar.
- A autoestrada irá possuir um código único, uma descrição e uma taxa para efeito do pagamento a ser efetuado pelos utentes.
- O veículo irá ser identificado no sistema por um identificador único, possui uma matrícula e possui também a classe deste mesmo, para ser usado no montante a pagar pelo utente, e o nome do utente sobre o qual o veículo está registado.
- A despesa da autoestrada tem um identificador único, e uma data em que a despesa foi feita. A despesa possui também um valor monetário associado a esta bem como uma descrição sobre o tipo de despesa que foi feita, poderá ser por exemplo manuntenção periódica do piso da estrada.

2.3. Requisitos de exploração

- Apresentar o número de veículos que circularam numa dada autoestrada num dado intervalo de tempo;
- Apresentar o total faturado, os gastos e o balanço num dado intervalo de tempo;

- Apresentar a matrícula e nome do proprietário assim como o total gasto dos top 5 veículos que mais gastaram no sistema;
- Fazer um top 10 dos veículos que mais quilómetros percorreram desde sempre, apresentando as suas matrículas e distância total percorrida;
- Apresentar todos os registos ordenados por classe de carro;
- Apresentar o total gasto em despesas que tenham uma dada palavra na descrição;
- Apresentar o total ganho e total gasto em todas as autoestradas.

2.4. Requisitos de controlo

O sistema deverá propocionar vários modos de acesso, de modo a encapsular informações ou capacidades de alteração de determinados utilizadores. Deste modo o sistema deve permitir:

- A utilizadores não administrativos, ou seja a funcionários das portagens, deverão registar a passagem do veículo pelo sistema. Devem inserir a matrícula do veículo, a classe do veículo de modo a no final puder ser calculado o montante, e receber por fim esse valor calculado;
- Os utilizadores administrativos, encarregues da parte administrativa da empresa, têm acesso a todos registos passados no sistema, bem como, à parte financeira da empresa, com os ganhos e prejuízos;
- A administração, também possui o acesso a manuntenção da autoestrada, podendo marcar uma manuntenção.

2.5. Análise geral dos requisitos

Face aos requisitos levantados, o objetivo é que haja um bom funcionamento da base de dados, não só a nível de armazenamento da informação mas também ao nível de consulta e de gestão da mesma. A integridade dos dados também é fundamental para um bom funcionamento do sistema, pelo que é necesário estabelecer as diferenças de acesso entre os vários utilizadores.

3. Modelação Conceptual

Finalizada a análise dos requisitos e dado o contexto e necessidade do sistema de base de dados a implementar, segue-se a modelação conceptual, escrita em notação CHEN, com a finalidade de obter uma identificação clara e concisa sobre os componentes pertinentes a esta mesma, como por exemplo, as entidades, os relacionamentos entre as entidades, atributos e chaves para cada entidade.

A validação do modelo com os requisitos descritos será apresentada posteriormente ao processo acima descrito.

3.1. Abordagem da modelação realizada

Em primeiro lugar, o grupo de trabalho procedeu para a fase de tradução dos requisitos nos seus respectivos componentes atómicos, por exemplo, as entidades.

Posteriormente, analisou-se as várias associações entre as entidades identificadas bem como os seu relacionamentos.

Por último, posto isto tudo, associou-se os vários atributos a estas entidades.

3.2. Identificação e caraterização das entidades

As entidades identificadas e utilizadas na modelação conceptual do trabalho são as seguintes:

Nome da entidade	Descrição	Também conhecido por	Ocorrências

Autoestrada	Termo usado para descrever uma estrada especialmente destinada ao trânsito rápido de automóveis.		Cada autoestrada possui um ou mais troços, terá também zero ou mais despesas a ser efetuadas nela.
Troco	Termo usado para descrever um pedaço de estrada.	secção	Cada troço tem zero ou mais registos associados
Registo	Termo utilizado para descrever algo que possui campos de informações.	referência	Cada registo, irá possuir um troço e irá possuir apenas um veículo a ele associado.
Veículo	Meio de transporte, que auxilia a tarefa de condução.	automóvel, carro, viatura	Cada veículo pode estar associado a vários registos.
Despesas	Termo utilizado para descrever a ação ou efeito de manter e de conservar.	gasto	Cada despesa possui apenas 1 autoestrada associada.

Tabela 1 – Dicionário de dados de entidades

3.3. Identificação e caraterização dos relacionamentos

Os relacionamentos presentes no modelo do grupo de trabalho são como se segue:

Entidade 1	Relacionamento	Entidade 2	Multiplicidade
Autoestrada	possui	Despesas	1-n
Autoestrada	Composta por	Troço	1-n
Troco	tem	Registo	1-n
Veículo	tem	Registo	1-n

Tabela 2 - Dicionário sobre os dados dos relacionamentos

3.4. Identificação e caraterização das associações dos atributos com as entidades e relacionamentos

Nesta secção são apresentados os atributos presentes no modelo conceptual:

Entidade	Atributo	Descrição	Tipo de dados	Null	Multivalor
Autoestrada	codigo	Identifica unicamente a autoestrada	INT	Não	Não
	Descricao	Breve sínopse da autoestrada	VARCHAR(32)	Não	Não
	Taxa	Taxa da autoestrada baseada na prioridade, varia entre zero e um	DECIMAL(3,2)	Não	Não
Troco	id_Troco	Identifica unicamente o troço da autoestrada	INT	Não	Não

	Des_Entrada	Descrição da portagem de entrada do troço	VARCHAR(45)	Não	Não
	Des_Saida	Descrição da portagem de saída do troço	VARCHAR(45)	Não	Não
	Distancia	Compriment o total to troço da autoestrada	DECIMAL(10,2)	Não	Não
Registo	id_Registo	Identifica unicamente o registo	INT	Não	Não
	Data_inicial	A data da entrada do veículo na autoestrada	DATETIME	Não	Não
	Data_final	A data da saída do veículo da autoestrada	DATETIME	Não	Não
	Montante	Total a pagar pelo utente da autoestrada	DECIMAL(6,2)	Não	Não
Veículo	id_Veiculo	Identifica unicamente o veículo	INT	Não	Não
	matricula	A matrícula do veículo	VARCHAR(8)	Não	Não
	Categoria	A classe do veículo, varia entre zero e um	ENUM(‘1’,’2’,’3’,’4’,’5’)	Não	Não
	Proprietario	O nome do dono do veículo	VARCHAR(128)	Não	Não

Despesas	id_Despesa	Identificar unicamente a despesa	INT	Não	Não
	Data	A data inicial da	DATE	Não	Não
	Valor	O valor da despesa	DECIMAL(10,2)	Não	Não
	Descricao	Descrição sobre o tipo de despesa	VARCHAR(256)	Não	Não

Tabela 3 - Dicionário de dados sobre os atributos presentes no modelo conceptual

3.5. Detalhe ou generalização das entidades

Nesta secção irá se apresentar uma descrição de cada uma das entidades identificadadas pelo grupo de trabalho.

- **Autoestrada** – Uma vez que a empresa GreenWay, necessita de um sistema de gestão de autoestradas, naturalmente terá de existir a entidade de autoestrada, uma vez que esta empresa alberga um vasto leque de autoestradas.
- **Troço** – Uma autoestrada é composta por vários troços, por exemplo Lisboa-Vila Franca de Xira, estes troços possuem uma portagem à entrada e outra à saída e irão também possuir uma extensão, ou seja, a distância em quilómetros da primeira portagem de entrada até a última portagem de saída desse tal troço.
- **Registo** – O registo é das principais entidades do sistema, uma vez que é necessário gerir o funcionamento das autoestradas, sendo, portanto, necessário gerir o sistema no que toca à parte burocrática. Associado ao registo existe a data incial, onde se guarda a primeira passagem do veículo na primeira portagem e a data final que é quando o veículo chega à segunda portagem. A matrícula do veículo também é guardada no registo.
- **Veículo** – Os veículos habitam o sistema de autoestradas, são os utentes destas mesmas, a necessidade de esta entidade estar no sistema é trivial. Cada veículo irá possuir uma classe, que irá influenciar o montante a pagar no final, um proprietário, o nome da pessoa

sobre o qual o veículo está registado, uma matrícula e o id do carro que funciona como identificador único da entidade.

- **Despesa** – As autoestradas, como qualquer outra via, estão sujeitas a despesas pontuais e periódicas, de modo a manter ou melhorar a qualidade para os utentes. Daí a necessidade desta entidade, nela será guardada um identificador único de despesa, uma data em que foi feita a despesa, o valor desta mesma e por fim uma breve descrição sobre a despesa em causa.

3.6. Apresentação e explicação do diagrama ER

Após uma análise extensa acerca dos requisitos e a respetiva tradução nas entidades e das relações entre estas, o modelo conceptual encontra-se pronto para ser elaborado, e como tal elaborou-se o seguinte modelo:

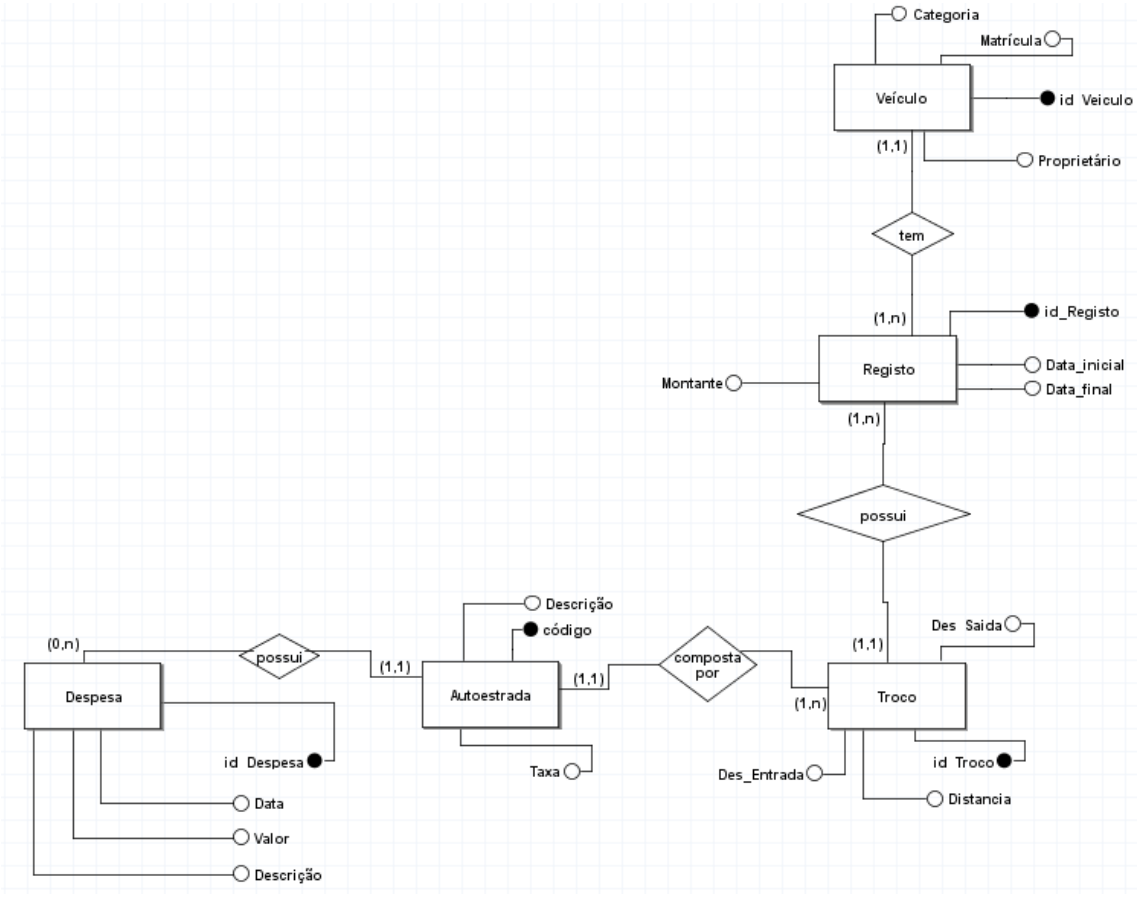

Figura 1 - Modelo Conceptual do Sistema de Autoestradas GreenWay

O modelo conceptual foi elaborado seguindo a notação CHEN, usando como ferramenta de construção o programa *brModelo.* As entidades são identificadas pelos retângulos, as relações pelos losângulos e os atributos estão associados a cada entidade e são representados pelos círculos. A multiplicidade de cada relacionamento está também identificada à entrada de cada entidade. Posto isto, a identificação de todos os componentes fulcrais do sistema desenvolvido é de fácil compreensão.

Os atributos e realções já foram discutidos nos pontos anteriores, contudo, de uma maneira geral o sistema de gestão de autoestradas irá possuir as entidades da despesa, a própria autoestrada, o troço (no modelo aparece como troco), o registo e o veículo, cada uma destas entidades irá possuir um identificador único para efeito de gestão e de procura e até de diferenciação. Os restantes atributos são específicos e únicos a cada entidade conforme o seu papel no sistema do grupo de trabalho.

Relativamente às relações, de modo a adaptar à realidade e imposições da empresa, uma autoestrada irá possuir zero ou mais despesas, o custo de operação de uma autoestrada é gigange, sendo necessário representar isso através desta relação. Uma autoestrada também é composta por um ou mais troços, uma vez que, como qualquer outra via, uma autoestrada necessita de ter um ponto de entrada e de saída, sendo que cada troço irá representar esse par de entrada e de saída, por exemplo, Lisboa-Barreiro (entrada portagem em Lisboa, saída

portagem em Barreiro). Os troços da autoestrada irão também possuir um todo conjunto de registos associados, um ou mais registos. Por fim, os registos irão também estar associados a apenas 1 veículo, uma vez que, cada pessoa transita no sistema em apenas 1 veículo de cada vez.

3.7. Validação do modelo de dados com o utilizador

Uma vez terminada a fase de modelação conceptual, o grupo de trabalho reuniu-se com a administração da empresa GreenWay onde se apresentou o modelo. O modelo reunia todos os requisitos, proprocionando as funcionalidades e capacidade de gestão e de consulta que a administração da emprese pretendia, e como tal foi aprovado.

4. Modelação lógica

O passo seguinte passa pela conversão do modelo conceptual para o respetivo modelo lógico, onde irá suportar os requisitos da base de dados anteriormente referidos.

Neste capítulo encontra-se detalhado todo este processo, por fim é novamente feita outra validação do modelo lógico obtido.

4.1 Construção e validação do modelo de dados lógico

Nesta secção, irão ser apresentados os passos tomados e necessários para a derivação do modelo lógico a partir dos componentes do respetivo modelo conceptual.

4.1.1 Dervivação das relações para o modelo lógico

De modo a derivar as relações para o modelo lógico, é necessário ter em atenção as 5 entidades identificadas no modelo conceptual e as suas respetivas relações.

A partir das seguintes entidades: Autoestrada, Troco, Registo, Veículo e Despesa e os seus relacionamentos, resultam nas seguintes tabelas no modelo lógico:

- Autoestrada(**codigo**, Descricao, Taxa);

O codigo é a chave primária representado por um INT, a Descricao é um VARCHAR(32) e a Taxa um DECIMAL(3,2).

- Troco(**id_Troco**, Des_Entrada, Des_Saida, Distancia, *Autoestrada_codigo*);

O id_Troco é a chave primária representado por um INT, a Des_Entrada e a Des_Saida são ambos representados por um VARCHAR(45), a Distancia é representado por um DECIMAL(10,2) e a Autoestrada_codigo é uma chave estrangeira resultante do relacionamento com a entidade da Autoestrada.

- Registo(**id_Registo**, Data_inicial, Data_final, Montante, *Troco_id* ,*Veiculo_idC*);

O id_Registo é a chave primária representado por um INT, a Data_incial e a Data_final são ambos DATETIME, o Montante é um DECIMAL(6,2). O Troco_id é resultante da relação com a entidade do Troco e portanto é uma chave estrangeira, e o Veiculo_id é também outra chave estrangeira devido à relação com a entidade do Veículo.

- Veículo(**id_Veiculo**, matricula, Proprietario, Categoria);

O id_Veiculo é a chave primária representado por um INT, a matricula é representada por um VARCHAR(8) e o Proprietario por um VARCHAR(128), a categoria é representada por um ENUM('1','2','3','4','5').

- Despesa(**id_Despesa**, Data, Valor, Descricao, *Autoestrada_codigo*);

O id_Despesa é a chave primária representado por um INT, a Data é representada por um DATE, o Valor por um DECIMAL(10,8) e a Descricao por um VARCHAR(256). O Autoestrada_codigo é a chave estrangeira devido ao relacionamento entre esta entidade e a entidade da Autoestrada.

É de realçar que não houve a necessidade de criar tabelas auxiliares de relação uma vez que não foi identificado nenhum relacionamento n-m.

Também não houve necessidade de criar tabelas com o intuito de eliminar atributos multivalorados, uma vez que, na concepção do modelo, não houve necessidade de utilizar atributos multivalorados.

4.1.2 Derivação dos relacionamentos para o modelo lógico

Uma vez completada a parte de derivar as relaçoes para tabelas no modelo lógico, segue-se derivação dos relacionamentos para o modelo lógico, onde, se pode verificar os seguintes relacionamentos:

- **Relacionamento de um para muitos (1:N)**

Nos relacionamentos deste tipo, existe uma distinção entre uma entidade “pai” e uma entidade “filho” que deverá ser identificada. Na entidade “filho” estão presentes chaves estrangeiras que correspendem à chave primária da entidade “pai”, como se pode observar nos seguintes realcionamentos levantados a partir do modelo conceptual da base de dados:

  - ✓ **Autoestrada para Manuntencao:**

  Autoestrada(codigo[PK]) -> Despesa(id_Despesa[PK],Autoestrada_codigo[FK]);

  - ✓ **Autoestrada para Troco:**

  Autoestrada(codigo[PK]) -> Troco(id_Troco[PK],Autoestrada_codigo[FK]);

  - ✓ **Troco para Registo:**

  Troco(id_Troco[PK])-> Registo(id_Registo[PK],Troco_id[FK]);

  - ✓ **Veiculo para Registo:**

  Veiculo(id_Veiculo[PK]) -> Registo(id_Registo[PK],Veiculo_id[FK]);

- **Relacionamento de um para um (1:1)**

Foi dado como sucesso, uma vez que não existe relacionamentos de um para um no modelo conceptual.

- **Relacionamento de muitos para muitos (N:N)**

Tais relacionamentos também não existem no modelo conceptual, logo, este passo foi dado como sucesso.

- **Relacionamentos recursivos:**

Este passo foi dado com sucesso, devido à inexistência de relacionamentos recursivos no modelo conceptual desenvolvido.

4.1.3 Análise das restrições de integridade

Estas restrições aplicadas são essenciais, uma vez que, garantem a exatidão e consistência dos dados num sistema de base de dados relacional. Ou seja, desta maneira, está

assegurado que os dados representam assertivamente a realidade modelada. Portanto, é imperativo que certos tipos de restrições sejam respeitados, tais como:

- **Integridade de entidade**
  A chave primária de uma entidade não pode pode conter atributos com valor nulo.
- **Integridade referencial**
  Esta restrição garante que um valor da chave estrangeira de uma relação "filho" seja o mesmo da chave primária da relação "pai" associada.
- **Restrições de domínio do atributo**
  A qualquer atributo é necessário associar, obrigatoriamente, uma gama de valores válida. Esta restrição verifica-se através de uma análise do dicionário de dados.
- **Multiplicidade**
  Indica a cardinalidade dos relacionamentos entre as duas entidades da base de dados. Na transição do modelo conceptual para o modelo lógico surgem tantas novas relações quantos relacionamentos binários n para m que puderem ser identificados no modelo conceptual.

Ora, em cada atributo de cada tabela o grupo teve o cuidado de ter isto tudo em atenção. Cada chave, seja esta primária ou estrangeira, e cada atributo não pode ser NULL, ou seja, obrigatoriamente necessita de possuir algum valor, exemplificando o comportamento de cada tabela temos:

Autoestrada (codigo,Descricao,Taxa)
Chave primária: codigo (NOT NULL, Unique Index, Auto Incremental)

Despesa (id_Despesa,Data,Valor,Descricao,*Autoestrada_codigo*)
Chave primária: id_Despesa (NOT NULL, Unique Index, Auto Incremental)
Chave estrangeira: Autoestrada_codigo (NOT NULL) **refere a** Autoestrada(codigo)

Troco(id_Troco,Des_Entrada,Des_Saida,Distancia,*Autoestrada_codigo*)
Chave primária: id_Troco (NOT NULL, Unique Index, Auto Incremental)
Chave estrangeira: Autoestrada_codigo (NOT NULL) **refere a** Autoestrada(codigo)

Veiculo(id_Veiculo,Matricula,Proprietario,Categoria)
Chave primária: id_Veiculo (NOT NULL, Unique Index, Auto Incremental)

Registo(id_Registo,Data_inicial,Data_final,Montante,*Veiculo_id*,*Troco_id*)
Chave primária: id_Registo (NOT NULL, Unique Index, Auto Incremental)
Chave estrangeira: Veiculo_id (NOT NULL) **refere a** Veiculo(id_Veiculo)
Troco_id (NOT NULL) **refere a** Troco(id_Troco)

Ou seja, efetuando uma análise no comportamento de cada chave, é fácil verificar que o modelo lógico construído cumpre todas as restrições de integridade.

4.2. Desenho do modelo lógico

O modelo lógico foi elaborado utilizando a ferramenta do *MySQL Workench*, onde, se traduziu o modelo conceptual no modelo lógico.

Para tal, as entidades identificadas no modelo conceptual foram convertidas nas respetivas tabelas, com os mesmos atributos e os mesmos tipos apresentados no dicionário de dados. A cardinalidade das ligações identificadas no modelo conceptual também se manteram no modelo lógico, uma vez que todo os sistema de base de dados necessita de estar interligado.

Logo, o modelo lógico elaborado irá ser o seguinte:

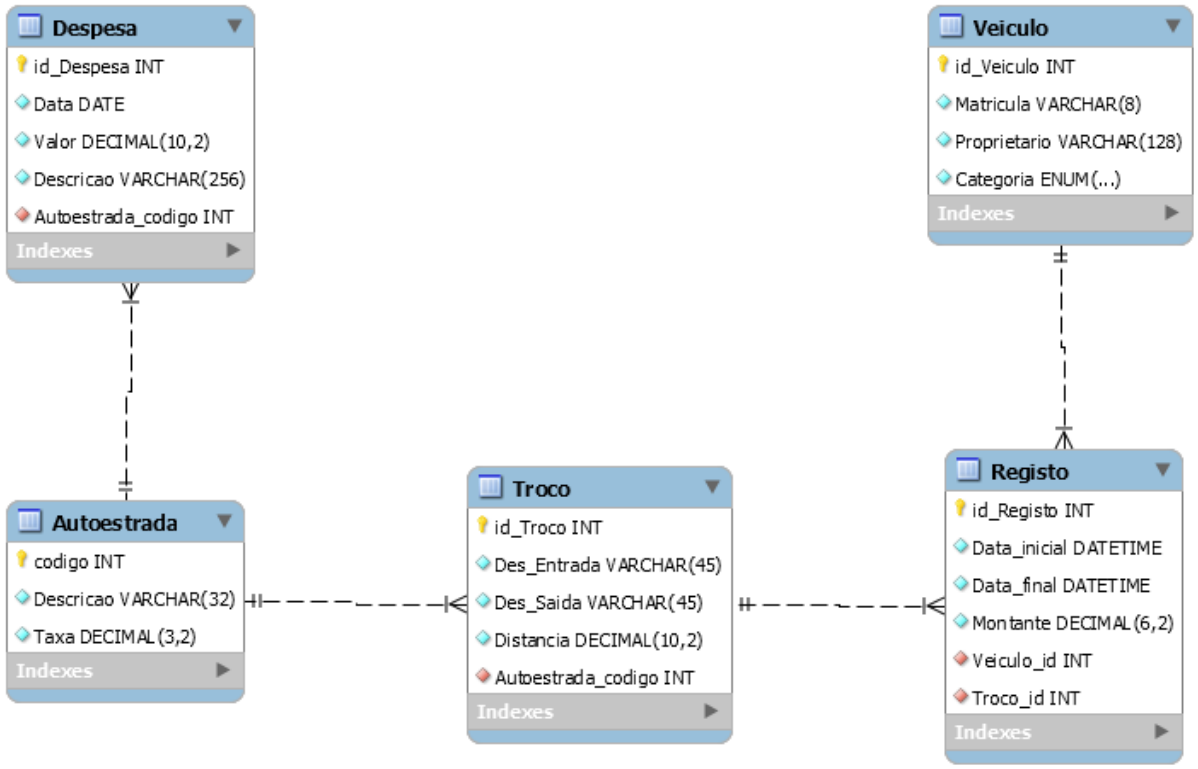

Figura 2 - Modelo lógico do sistema de gestão de autoestradas

4.3. Validação do modelo através da normalização

A base de dados à qual o modelo lógico dará origem necessita de ser robusta e sem problemas de utilização, sendo necessário para tal garantir que o modelo é válido através da normalização.

Como tal, são verificadas as três primeiras formas normais, pelo que será suficiente para garantir que são evitados quaisqueres problemas de redundância de dados.

De seguida, é apresentada uma tabela onde se encontram as tabelas apresentadas no modelo lógico juntamente com os seus atributos.

Nome da tabela	Determinante	Dependência
Autoestrada(A)	codigo(a1)	Descricao (a2) Taxa (a3)
Veiculo(B)	id_Veiculo (b1)	Matricula (b2) Proprietario (b3) Categoria (b4)
Troco(C)	id_Troco(c1)	Des_Entrada(c2) Des_Saida(c3) Distancia(c4) Autoestrada_codigo[FK](a1)
Registo(D)	id_Registo(d1)	id_Registo(d2) Data_inicial(d3) Data_final(d4) Montante(d5) Troco_id[FK](c1) Veiculo_id[FK](b1)
Despesas(E)	id_Despesa(e1)	Data(e2) Valor(e3) Descricao(e4) Autoestrada_codigo[FK](a1)

Tabela 4 - Dependências e determinantes do modelo lógico

Contudo, de forma a validar o modelo através da normalização é preciso primeiro entender as três primeiras formas normais, resumidamente:

- **1ª Forma Normal –** Na primeira forma normal, o objetivo é identificar os elementos de dados repetidos na tabela, ou seja, os principais candidatos a ter uma tabela própria, verificando o modelo e analisando a tabela 4 é possível verificar que já se encontra normalizado segundo a mesma.
- **2ª Forma Normal –** Após verificada a primeira forma normal, na segunda forma normal é necessário verificar nas várias tabelas por chaves primárias compostas, ou seja algo que ainda seja definido por mais do que um atributo. As tabelas que apenas se encontram definidas por uma só chave já se encontram de acordo com esta forma, para as outros verificou-se as chaves primárias e a sua importância na tabela e os atributos que não são funcionalmente dependentes da mesma. Aplicando esta abordagem, o modelo também se encontra na segunda forma normal.

- **3ª Forma Normal -** Esta fase consiste em remover as dependências transitivas que foram identificadas nas tabelas do modelo lógico, tentando colocar estas numa nova relação. Analisado o modelo e a tabela 4, esta fase está concluída.

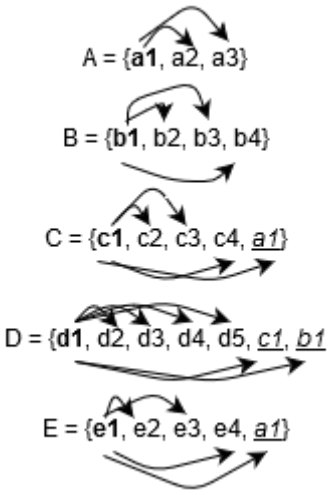

Figura 3 - Diagrama de dependências

Como é possível verificar pelo diagrama de dependências da figura três e pelo que já foi dito em relação as três formas normais, o modelo encontra-se validado uma vez que cumpre a 1ª a 2ª e a 3ª forma normal.

Logo, o modelo não possui portanto qualquer tipo de redundância ou inconsistência de dados, por consequência conseguimos inferir que o modelo é fidedigno.

4.4. Validação do modelo com interrogações do utilizador

Uma vez feita a conversão do modelo conceptual para o respetivo modelo lógico, utilizando para tal um todo conjunto de regras e normas necessárias para obter um modelo de dados válido e adequado às necessidades e satisfação do seu utilizador, ou seja a empresa GreenWay. O modelo disponibiliza todas as funcionalidades que a empresa necessita, sendo assim cumprido os requisitos e imposições estabelecidos no início do projeto.

4.5. Validação do modelo com as transações estabelecidas

A base de dados necessita de suportar as transações mais importantes e mais frequentes, logo, é necessário efetuar um estudo de como estas se irão comportar.

Ora, por exemplo, quando um funcionário da portagem necessita de registar a passagem de um veículo no sistema e receber o pagamento, o funcionário terá de tirar partido da base de dados de modo a guardar o registo deste veículo no sistema e de modo a receber o cálculo do montante automaticamente. O funcionário terá que inserir a data inicial e a data final do veículo na autoestrada, que corresponde ao intervalo do tempo no veículo no troço, o id do troço no qual trabalha e a identificação do veículo.

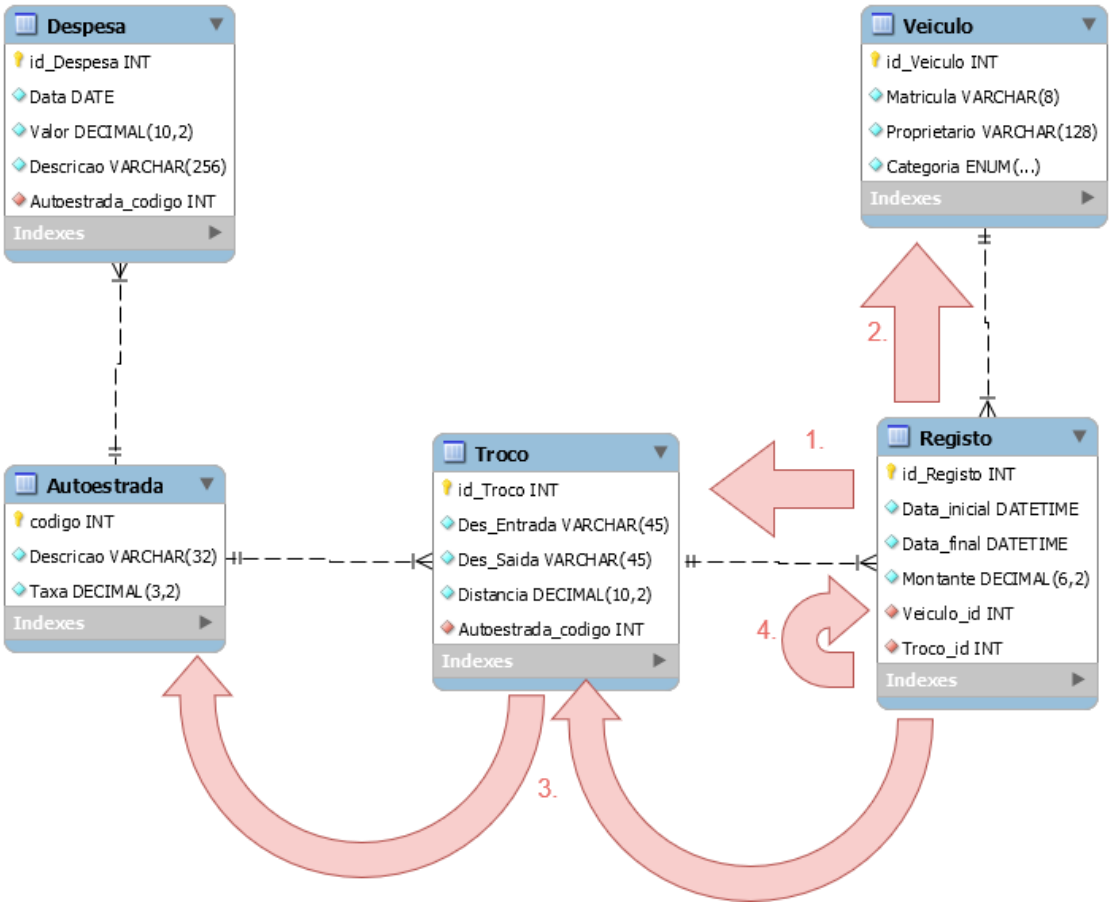

Figura 4 - representação gráfica transação que insere registo

De modo a inserir um registo no sistema, o funcionário tem que inserir a data inicial e final, e o id do veículo e do troço, de seguida o montante é calculado automaticamente, em (1) com o id do troço retira-se a Distancia, que irá dar o número de km perocrridos, emn(2) é retirada com o id do veículo a categoria do carro, (3) com o id do troço, é retirado o código da autoestrada que é utilizado para retirar a taxa da autoestrada, em (4) é calculado o montante utilizando para tal a fórmula:

*Montante = Distancia * Taxa * (1+(1/Categoria))*

Sendo depois adicionado o registo guardado também em 4.

4.6. Validação do modelo lógico com o utilizador

O grupo de trabalho reuniu-se com a empresa GreenWay, os utilizadores da base de dados, e após algum tempo, a administração da empresa confirmou que garante os requisitos pretendidos.

5. Implementação Física

Uma vez completado o modelo lógico, segue-se a implementação física da base de dados num SGBD, partindo do modelo lógico previamente desenvolvido.

5.1. Seleção do sistema de gestão de base de dados

Tendo em consideração o percurso académico do grupo, a ferramenta de gestão de base de dados cujo grupo tem mais proximidade foi o *MySQL*, razão pela qual foi a preterida em relação as outras.

A formulação do esquema físico utilizando o *MySQL* é feita com muito pouco esforço e com qualidade garantida de modo a que no final, seja produzido um produto eficiente.

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

O esquema lógico de modo a ser traduzido para o respetivo esquema físico, no SGBD utilizado pelo grupo, bastou utilizar a ferramenta do *Foward Engineering*, disponibilizado por este tal SGBD.

O primeiro passo de tradução foi portanto dado com tremenda facilidade, sendo que a script de criação foi também gerada automaticamente onde temos:

```
CREATE SCHEMA IF NOT EXISTS `autoestrada` DEFAULT CHARACTER SET utf8 ;
USE `autoestrada` ;
```

Figura 5 - Criação do *workplace*

A figura 5 apresenta a criação inicial do *workplace* também conhecido por *schema* que irá servir como base para as consequentes tabelas de dados.

```
CREATE TABLE IF NOT EXISTS `autoestrada`.`Autoestrada` (
  `codigo` INT NOT NULL AUTO_INCREMENT,
  `Descricao` VARCHAR(32) NOT NULL,
  `Taxa` DECIMAL(3,2) NOT NULL,
  PRIMARY KEY (`codigo`),
  UNIQUE INDEX `codigo_UNIQUE` (`codigo` ASC) VISIBLE)
ENGINE = InnoDB;
```

Figura 6 - Criação da tabela autoestrada

A figura 6 apresenta a criação da tabela autoestrada.

```
CREATE TABLE IF NOT EXISTS `autoestrada`.`Veiculo` (
  `id_Veiculo` INT NOT NULL AUTO_INCREMENT,
  `Matricula` VARCHAR(8) NOT NULL,
  `Proprietario` VARCHAR(128) NOT NULL,
  `Categoria` ENUM('1', '2', '3', '4', '5') NOT NULL,
  PRIMARY KEY (`id_Veiculo`),
  UNIQUE INDEX `idC_UNIQUE` (`id_Veiculo` ASC) VISIBLE)
ENGINE = InnoDB;
```

Figura 7 - Criação da tabela veiculo

A figura 7 apresenta a criação da tabela veiculo.

```
CREATE TABLE IF NOT EXISTS `autoestrada`.`Troco` (
  `id_Troco` INT NOT NULL AUTO_INCREMENT,
  `Des_Entrada` VARCHAR(45) NOT NULL,
  `Des_Saida` VARCHAR(45) NOT NULL,
  `Distancia` DECIMAL(10,2) NOT NULL,
  `Autoestrada_codigo` INT NOT NULL,
  PRIMARY KEY (`id_Troco`),
  UNIQUE INDEX `id_Troco_UNIQUE` (`id_Troco` ASC) VISIBLE,
  INDEX `fk_Troco_Autoestrada1_idx` (`Autoestrada_codigo` ASC) VISIBLE,
  CONSTRAINT `fk_Troco_Autoestrada1`
    FOREIGN KEY (`Autoestrada_codigo`)
    REFERENCES `autoestrada`.`Autoestrada` (`codigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 8 - Criação da tabela troco

A figura 8 apresenta o script de criação da tabela troco, respetivo ao troço da autoestrada.

```
CREATE TABLE IF NOT EXISTS `autoestrada`.`Registo` (
  `id_Registo` INT NOT NULL AUTO_INCREMENT,
  `Data_inicial` DATETIME NOT NULL,
  `Data_final` DATETIME NOT NULL,
  `Montante` DECIMAL(6,2) NOT NULL,
  `Veiculo_id` INT NOT NULL,
  `Troco_id` INT NOT NULL,
  PRIMARY KEY (`id_Registo`),
  INDEX `fk_Registo_Veiculo1_idx` (`Veiculo_id` ASC) VISIBLE,
  UNIQUE INDEX `idR_UNIQUE` (`id_Registo` ASC) VISIBLE,
  INDEX `fk_Registo_Troco1_idx` (`Troco_id` ASC) VISIBLE,
  CONSTRAINT `fk_Registo_Veiculo1`
    FOREIGN KEY (`Veiculo_id`)
    REFERENCES `autoestrada`.`Veiculo` (`id_Veiculo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Registo_Troco1`
    FOREIGN KEY (`Troco_id`)
    REFERENCES `autoestrada`.`Troco` (`id_Troco`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 9 - Criação da tabela registo

A figura 9 é referente ao script de criação da tabela correspondente ao registo.

```
CREATE TABLE IF NOT EXISTS `autoestrada`.`Despesa` (
  `id_Despesa` INT NOT NULL AUTO_INCREMENT,
  `Data` DATE NOT NULL,
  `Valor` DECIMAL(10,2) NOT NULL,
  `Descricao` VARCHAR(256) NOT NULL,
  `Autoestrada_codigo` INT NOT NULL,
  PRIMARY KEY (`id_Despesa`),
  INDEX `fk_Manuntencao_Autoestrada1_idx` (`Autoestrada_codigo` ASC) VISIBLE,
  UNIQUE INDEX `idM_UNIQUE` (`id_Despesa` ASC) VISIBLE,
  CONSTRAINT `fk_Manuntencao_Autoestrada1`
    FOREIGN KEY (`Autoestrada_codigo`)
    REFERENCES `autoestrada`.`Autoestrada` (`codigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

USE `autoestrada` ;
```

Figura 10 - Criação da tabela despesa

A figura 7 é referente ao script de criação da tabela despesa.

Todas as tabelas criadas a partir de todos estes scripts serão povoadas através de um todo outro conjunto de scripts.

```
call autoestrada.insereAutoestrada('A2(Braga-P.Barca',0.09);
call autoestrada.insereAutoestrada('A3(Coimbra-T.Novas)',0.2);
call autoestrada.insereAutoestrada('A4(T.Novas-Lisboa)',0.15);
call autoestrada.insereAutoestrada('A5(S.Gabriel-C.Ourique)',0.32);
```

Figura 11 - Povoamento da tabela autoestrada

A figura 11 corresponde ao scipt de povoiação da tabela da autoestrada.

```
call autoestrada.insereVeiculo('07-AB-54','Tiago Ferreira',1);
call autoestrada.insereVeiculo('34-74-CD','Joaquim Vilas Boas',1);
call autoestrada.insereVeiculo('99-87-CP','Luisinho Martins',5);
call autoestrada.insereVeiculo('BC-65-74','Luis Porto Machado',2);
call autoestrada.insereVeiculo('BV-31-22','Joana Martins',4);
call autoestrada.insereVeiculo('PL-21-74','Flávio dos Dias',3);
call autoestrada.insereVeiculo('87-LO-21','João Diaz de Teixeira',1);
call autoestrada.insereVeiculo('90-24-KL','Inês de Paiva',4);
call autoestrada.insereVeiculo('JP-69-24','José Peixoto',3);
call autoestrada.insereVeiculo('76-DA-48','Chico Lomba',1);
call autoestrada.insereVeiculo('05-GJ-45','Rebeca Goncalves Sousa',5);
call autoestrada.insereVeiculo('MB-48-36','Luiza Fernandes Cardoso',2);
call autoestrada.insereVeiculo('98-TY-01','Thomas Zeeb',3);
call autoestrada.insereVeiculo('95-31-AP','Emmanuel Gavrilov',4);
call autoestrada.insereVeiculo('59-RM-99','Chiyoe Goto',1);
call autoestrada.insereVeiculo('PX-52-34','Erick Barbosa Rodrigues',2);
call autoestrada.insereVeiculo('23-AY-76','Estevan Oliveira Goncalves',5);
call autoestrada.insereVeiculo('QE-78-07','Vitória Cavalcanti Fernandes',1);
call autoestrada.insereVeiculo('99-99-RE','Anna Cardoso Azevedo',3);
call autoestrada.insereVeiculo('PO-02-01','Rafaela Sousa',4);
call autoestrada.insereVeiculo('56-20-PR','Miguel Antunes',2);
call autoestrada.insereVeiculo('98-NM-11','Luís Mendes',5);
call autoestrada.insereVeiculo('38-HJ-02','Hugo Sousinha Bossanova',1);
call autoestrada.insereVeiculo('GR-04-20','Maria Joana',3);
call autoestrada.insereVeiculo('69-GT-15','Cátia Gonçalves',4);
call autoestrada.insereVeiculo('SW-00-24','Hugo Vaquero',2);
```

Figura 12 - Povoar tabela veículo

Na figura 12 está representado o script de povoamento da tabela correspondente ao veículo.

```
call autoestrada.InsereTroco('Primeira Portagem(Braga)', 'Segunda Portagem(Avidos)', 23, 1);
call autoestrada.InsereTroco('Segunda Portagem(Avidos)', 'Primeira Portagem(Braga)', 23, 1);
call autoestrada.InsereTroco('Segunda Portagem(Avidos)', 'Terceira Portagem(Porto)', 34.4, 1);
call autoestrada.InsereTroco('Terceira Portagem(Porto)', 'Segunda Portagem(Avidos)', 34.4, 1);
call autoestrada.InsereTroco('Terceira Portagem(Porto)', 'Quarta Portagem(V.N.Gaia)', 15, 1);
call autoestrada.InsereTroco('Quarta Portagem(V.N.Gaia)', 'Terceira Portagem(Porto)', 15, 1);
call autoestrada.InsereTroco('Quarta Portagem(V.N.Gaia)', 'Quinta Portagem(Lourosa)', 21, 1);
call autoestrada.InsereTroco('Quinta Portagem(V.N.Gaia)', 'Quarta Portagem(Lourosa)', 21, 1);
```

Figura 13 – Povoar primeiro troço autoestrada

```
call autoestrada.InsereTroco('Primeira Portagem(Braga)', 'Segunda Portagem(Guimaraes)', 24.9, 2);
call autoestrada.InsereTroco('Segunda Portagem(Guimaraes)', 'Primeira Portagem(Braga)', 24.9, 2);
call autoestrada.InsereTroco('Segunda Portagem(Guimaraes)', 'Terceira Portagem(Fafe)', 20, 2);
call autoestrada.InsereTroco('Terceira Portagem(Fafe)', 'Segunda Portagem(Guimaraes)', 20, 2);
call autoestrada.InsereTroco('Terceira Portagem(Fafe)', 'Quarta Portagem(P.Lanhoso)', 25, 2);
call autoestrada.InsereTroco('Quarta Portagem(P.Lanhoso)', 'Terceira Portagem(Fafe)', 25, 2);
call autoestrada.InsereTroco('Quarta Portagem(P.Lanhoso)', 'Quinta Portagem(Vila Verde)', 22, 2);
call autoestrada.InsereTroco('Quinta Portagem(Vila Verde)', 'Quarta Portagem(P.Lanhoso)', 22, 2);
call autoestrada.InsereTroco('Quinta Portagem(Vila Verde)', 'Sexta Portagem(P.Barca)', 21.8, 2);
call autoestrada.InsereTroco('Sexta Portagem(P.Barca)', 'Quinta Portagem(Vila Verde)', 21.8, 2);
```

Figura 14 - Povoar segundo troço autoestrada

```
call autoestrada.InsereTroco('Primeira Portagem(Coimbra)', 'Segunda Portagem(Espinhal)', 28.8, 3);
call autoestrada.InsereTroco('Segunda Portagem(Espinhal)', 'Primeira Portagem(Coimbra)', 28.8, 3);
call autoestrada.InsereTroco('Segunda Portagem(Espinhal)', 'Terceira Portagem(Pombal)', 40.3, 3);
call autoestrada.InsereTroco('Terceira Portagem(Pombal)', 'Segunda Portagem(Espinhal)', 40.3, 3);
call autoestrada.InsereTroco('Terceira Portagem(Pombal)', 'Quarta Portagem(Leiria)', 27, 3);
call autoestrada.InsereTroco('Quarta Portagem(Leiria)', 'Terceira Portagem(Pombal)', 27, 3);
call autoestrada.InsereTroco('Quarta Portagem(Leiria)', 'Quinta Portagem(Fatima)', 30.2, 3);
call autoestrada.InsereTroco('Quinta Portagem(Fatima)', 'Quarta Portagem(Leiria)', 30.2, 3);
call autoestrada.InsereTroco('Quinta Portagem(Fatima)', 'Sexta Portagem(Torres Novas)', 26.1, 3);
call autoestrada.InsereTroco('Sexta Portagem(Torres Novas)', 'Quinta Portagem(Fatima)', 26.1, 3);
```

Figura 15 - Povoar terceiro troço autoestrada

```
call autoestrada.InsereTroco('Primeira Portagem(Torres Novas)', 'Segunda Portagem(Santarem)', 39.4, 4);
call autoestrada.InsereTroco('Segunda Portagem(Santarem)', 'Primeira Portagem(Torres Novas)', 39.4, 4);
call autoestrada.InsereTroco('Segunda Portagem(Santarem)', 'Terceira Portagem(A.Cima)', 26.7, 4);
call autoestrada.InsereTroco('Terceira Portagem(A.Cima)', 'Segunda Portagem(Santarem)', 26.7, 4);
call autoestrada.InsereTroco('Terceira Portagem(A.Cima)', 'Quarta Portagem(V.F.Xira)', 26.3, 4);
call autoestrada.InsereTroco('Quarta Portagem(V.F.Xira)', 'Terceira Portagem(A.Cima)', 26.3, 4);
call autoestrada.InsereTroco('Quarta Portagem(V.F.Xira)', 'Quinta Portagem(Lisboa)', 30.8, 4);
call autoestrada.InsereTroco('Quinta Portagem(Lisboa)', 'Quarta Portagem(V.F.Xira)', 30.8, 4);
```

Figura 16 - Povoar quarto troço autoestrada

```
call autoestrada.InsereTroco('Primeira Portagem(S.Gabriel)', 'Segunda Portagem(Alcabideche)',2.59,5);
call autoestrada.InsereTroco('Segunda Portagem(Alcabideche)','Primeira Portagem(S.Gabriel)',2.59,5);
call autoestrada.InsereTroco('Segunda Portagem(Alcabideche)','Terceira Portagem(Estoril)',2.69,5);
call autoestrada.InsereTroco('Terceira Portagem(Estoril)','Segunda Portagem(Alcabideche)',2.69,5);
call autoestrada.InsereTroco('Terceira Portagem(Estoril)','Quarta Portagem(S.Domingos de Rana)',4.68,5);
call autoestrada.InsereTroco('Quarta Portagem(S.Domingos de Rana)','Terceira Portagem(Estoril)',4.68,5);
call autoestrada.InsereTroco('Quarta Portagem(S.Domingos de Rana)','Quinta Portagem(Queijas)',6.02,5);
call autoestrada.InsereTroco('Quinta Portagem(Queijas)','Quarta Portagem(S.Domingos de Rana)',6.02,5);
call autoestrada.InsereTroco('Quinta Portagem(Queijas)','Sexta Portagem(C.Ourique)',8.06,5);
call autoestrada.InsereTroco('Sexta Portagem(C.Ourique)','Quinta Portagem(Queijas)',8.06,5);
```

Figura 17 - Povoar quinto troço autoestrada

```
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 16:50:14', 1, 1);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:00:59', 1, 3);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 14:34:25', 1, 5);
call autoestrada.InsereRegisto('2017-05-18 13:40:30', '2017-05-18 13:59:04', 1, 7);
call autoestrada.InsereRegisto('2017-05-15 16:25:00', '2017-05-15 16:50:14', 2, 2);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:00:14', 2, 4);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:08:14', 2, 6);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 13:50:14', 2, 8);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 17:10:14', 3, 9);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 15:58:14', 3, 11);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:00:14', 3, 13);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:32:14', 3, 15);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 16:50:14', 4, 17);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 15:58:14', 4, 10);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 14:57:14', 4, 12);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:17:14', 4, 14);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 17:15:14', 5, 16);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:17:14', 5, 18);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:34:14', 5, 19);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:16:14', 5, 21);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 17:03:14', 6, 23);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:10:14', 6, 25);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:23:14', 6, 27);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:37:14', 6, 28);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 16:50:14', 7, 20);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 15:52:14', 7, 22);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:50:14', 7, 24);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:04:14', 7, 26);
```

Figura 18 - Povoar registo (1)

```
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 17:05:14', 8, 29);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 15:57:14', 8, 30);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 14:57:14', 8, 31);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:23:14', 8, 32);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 17:12:14', 9, 33);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:01:14', 9, 33);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:17:14', 9, 34);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 13:58:14', 9, 35);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 17:17:14', 10, 36);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:56:14', 10, 37);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:36:14', 10, 38);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 15:02:14', 10, 39);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 16:59:14', 11, 40);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 17:19:14', 11, 41);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:12:14', 11, 42);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:37:14', 11, 43);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 16:43:14', 12, 44);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:12:14', 12, 45);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:50:14', 12, 46);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:11:14', 12, 1);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 17:23:14', 13, 2);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:12:14', 13, 3);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 14:56:14', 13, 4);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 13:59:14', 13, 5);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 16:53:14', 14, 6);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 15:58:14', 14, 7);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:08:14', 14, 8);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:09:14', 14, 9);
```

Figura 19 - Povoar registo (2)

```
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 17:12:14', 15, 10);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 17:00:14', 15, 11);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 14:52:14', 15, 12);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:56:14', 15, 13);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 17:27:14', 16, 14);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 15:52:14', 16, 15);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 16:03:14', 16, 16);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:03:14', 16, 17);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 17:47:14', 17, 18);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 15:50:14', 17, 19);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 14:40:14', 17, 20);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:10:14', 17, 21);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 17:20:14', 18, 22);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 15:40:14', 18, 23);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:20:14', 18, 24);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 13:40:14', 18, 25);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 18:00:14', 19, 26);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:00:14', 19, 27);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:01:14', 19, 28);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 13:38:14', 19, 29);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 16:50:14', 20, 30);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 15:50:14', 20, 31);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:10:14', 20, 32);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 13:50:14', 20, 33);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 17:20:14', 21, 34);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:05:14', 21, 35);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:10:14', 21, 36);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:00:14', 21, 37);
```

Figura 20 - Povoar registo (3)

```
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 17:00:14', 22, 38);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:30:14', 22, 39);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 14:50:14', 22, 40);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:00:14', 22, 41);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 16:40:14', 23, 42);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:00:14', 23, 43);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 15:25:14', 23, 44);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:00:14', 23, 45);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 16:50:14', 24, 46);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:00:14', 24, 1);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 14:35:14', 24, 15);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:00:14', 24, 42);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 16:40:14', 25, 24);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:20:14', 25, 25);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 14:41:14', 25, 46);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:50:14', 25, 37);
call autoestrada.InsereRegisto('2017-05-15 16:20:00', '2017-05-15 16:40:14', 26, 36);
call autoestrada.InsereRegisto('2017-05-16 15:20:00', '2017-05-16 16:20:14', 26, 20);
call autoestrada.InsereRegisto('2017-05-17 14:20:00', '2017-05-17 14:41:14', 26, 30);
call autoestrada.InsereRegisto('2017-05-18 13:20:00', '2017-05-18 14:50:14', 26, 12);
```

Figura 21 - Povoar registo (4)

```
call autoestrada.insereDespesa('2017-04-18', 1000, 'Limpeza da autoestrada', 1);
call autoestrada.insereDespesa('2017-04-20', 1000, 'Limpeza da autoestrada', 2);
call autoestrada.insereDespesa('2017-04-22', 1000, 'Limpeza da autoestrada', 3);
call autoestrada.insereDespesa('2017-04-24', 1000, 'Limpeza da autoestrada', 4);
call autoestrada.insereDespesa('2017-04-26', 1000, 'Limpeza da autoestrada', 5);
call autoestrada.insereDespesa('2017-04-30', 13000, 'Salarios de funcionarios', 1);
call autoestrada.insereDespesa('2017-04-30', 13000, 'Salarios de funcionarios', 2);
call autoestrada.insereDespesa('2017-04-30', 13000, 'Salarios de funcionarios', 3);
call autoestrada.insereDespesa('2017-04-30', 14000, 'Salarios de funcionarios', 4);
call autoestrada.insereDespesa('2017-04-30', 21000, 'Salarios de funcionarios', 5);
```

Figura 22 - Povoar despesa

5.3. Tradução das interrogações do utilizador para SQL

De seguida, serão apresentadas as queries correspondentes às interrogações propostas pelo utilizador, ou seja, serão apresentadas as soluções dos requisitos de exploração.

```
CREATE PROCEDURE `query1` (IN auto INT, IN data_in Date, IN data_fin Date)
BEGIN
    SELECT COUNT(*) AS total_veiculos FROM
    (
    SELECT *
    FROM registo INNER JOIN troco ON troco_id = id_troco
        WHERE Autoestrada_codigo = auto && Data_final >= data_in && Data_final<= data_fin
        GROUP BY veiculo_id
    ) AS Lista_veiculos_a1_intervalo;
END
```

Figura 23 - Query número 1

Na figura 23, está apresentada a query correspondente ao primeiro requisito de exploração, que irá devolver o número total de veículos que circularam na autoestrada A1 entre um intervalo de tempo escolhido e na autoestrada pretendida.

Para tal, basta fazer um count do número de vezes que aparece, de seguida faz-se um *SELECT* juntando tabela do registo com a do troco, utilizando para tal um *INNER JOIN*, verificando se o código da autoestrada é o pretendido, e se encontra-se no dado intervalo de tempo, agrupando todas estas ocorrências por id do veículo.

No final é feito o *count* do número de veículos, sendo retornado o número de veículos que era pretendido retornar.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `query2`(IN ano INT)
BEGIN
    SELECT total_ganho, total_gasto, (total_ganho-total_gasto) as balanco FROM (
        SELECT *
        FROM (
        SELECT SUM(Montante) as total_ganho
            FROM registo
            WHERE YEAR(Data_final) = ano) AS total_ganho
        JOIN (
        SELECT SUM(Valor) as total_gasto
            FROM despesa
            WHERE Year(Data) = ano) AS total_gasto ) as subqueries;
END
```

Figura 24 - Query número 2

Na figura 24, é apresentada a solução para o segundo requisito de exploração, cujo objetivo é apresentar o valor total faturado e o total gasto no ano pretendido pelo utilizador, ou seja, apresentar o balanço das contas.

De modo a resolver esta query, foi feito dois *SELECTS* distintos, um da tabela do registo, onde foi retirado a soma dos Montantes do ano pretendido, e foi feito o *JOIN*, com a soma dos valores da tabela da despesa, sendo apresentado no final na forma de uma tabela, o total ganho e o total gasto na autoestrada no escolhido.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `query3`()
BEGIN
    SELECT v.Matricula, v.Proprietario, SUM(r.Montante) as Total_Gasto
        FROM veiculo v, registo r
            WHERE r.Veiculo_id = v.id_Veiculo
            GROUP BY r.Veiculo_id
            ORDER BY SUM(r.Montante) DESC
            LIMIT 5;
END
```

Figura 25 - Query número 3

É apresentada na figura 25 a solução para o terceiro requisito de exploração, onde se pretende apresentar a matrícula e nome do proprietário assim como o total gasto dos 5 veículos que mais gastaram no sistema.

Para se responder a tal query, é necessário efetuar um *SELECT* e retirar a matrícula e o proprietário da tabela do Veiculo, e a soma dos montantes da tabela do registo, sendo necessário comparar o id do veículo do registo com o id do veículo do carro de modo a manter a integridade referencial.

De seguida, os valores são agrupados pelo id do veículo do registo e ordenados pela soma do montante gasto sendo limitado aos 5 primeiros.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `query4`()
BEGIN
    SELECT v.matricula, SUM(t.distancia) as distancia_total
        FROM registo r, troco t, veiculo v
            WHERE r.Troco_id = t.id_Troco && r.veiculo_id = v.id_veiculo
            GROUP BY r.Veiculo_id
            ORDER BY SUM(t.distancia) DESC
            LIMIT 10;
END
```

Figura 26 – Query número 4

Na figura 26, está apresentada a solução para o quarto requisito de exploração. O problema consiste em apresentar as matrículas dos 10 veículos que mais distância percorreram no sistema, explicitando essa tal distância.

A resposta a essa interrogação consiste em novamente, fazer um *SELECT* retirando a matrícula e a soma das distâncias do troço (valore que aparecem na tabela resultante), utilizando a tabela do registo, do troco e do veiculo. Primeiramente é necessário verificar se o id do troço do registo corresponde ao id do troço e se o id do veículo do registo corresponde ao id do veículo, de seguida, os dados são agrupados pelo id do veículo sendo ordenados pela soma da distância do troço e limitados aos 10 primeiros, para obter o tal top 10 dos veículos.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `query5`()
BEGIN
    SELECT r.id_Registo, r.Data_inicial, r.Data_final, r.Montante, r.Veiculo_id, r.Troco_id
    FROM registo r , veiculo v
        WHERE r.Veiculo_id = v.id_Veiculo
        ORDER BY v.categoria;
END
```

Figura 27 - Query número 5

Nesta interrogação, é pretendido que os registos sejam ordenados por categoria do veículo logo, é feito o *SELECT* de todos os elementos do veículo sendo que serão usadas tanto a tabela do registo como a do veículo (para poder retirar a categoria), de seguida, só é necessário verificar se o id do veículo no registo corresponde ao id do veículo em caso afirmativo é adicionada a tabela da solução que irá ser ordenada por categoria.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `query6`(IN descr VARCHAR(10))
BEGIN
    SELECT SUM(Valor) AS Custo_total FROM despesa
    WHERE POSITION(descr IN Descricao) != 0;
END
```

Figura 28 - Query número 6

Na query 6, é pretendido apresentar o valor total das despesas que contêm a palavra limpeza na sua descrição, efetivamente calculando o valor total das despesas do tipo limpeza. Para tal acontecer, só é necessário fazer o *SELECT* do valor da tabela da despesa fazendo o *SUM*, de modo a dar o valor total, e na descrição da despesa, caso seja zero é porque a palavra "Limpeza" não está lá, logo verfica-se para diferente de zero, retirando todos os valores da despesa que na descrição possuem a palavra "Limpeza".

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `query7`(IN data_in Date, IN data_fin Date)
BEGIN
    SELECT t1.autoestrada_codigo, total_ganho, total_gasto
    FROM (
    SELECT autoestrada_codigo, SUM(Montante) as total_ganho
        FROM registo INNER JOIN troco ON id_troco = troco_id
        WHERE Data_final BETWEEN data_in AND data_fin
        GROUP BY autoestrada_codigo) t1
    INNER JOIN (
    SELECT autoestrada_codigo, SUM(Valor) as total_gasto
        FROM despesa
        WHERE Data BETWEEN data_in AND data_fin
        GROUP BY autoestrada_codigo) t2 ON t1.autoestrada_codigo = t2.autoestrada_codigo;

END
```

Figura 29 - Query número 7

Nesta última interrogação, o intuito é apresentar o total faturado e o total gasto em todas as autoestradas do sistema num determiando intervalo de tempo. A resolução passa por efetuar dois *SELECTS*, o primeiro para retirar a soma dos Montantes do registo (faturado) e o segundo para retirar a soma dos valores da despesa. É necessário também verificar os intervalos de tempo tanto a retirar do registo como da despesa, e agrupar segundo os códigos, de modo a informação ser apresentada de uma forma coerente.

5.4. Tradução das transações estabelecidas para SQL

Uma transação é uma propragação de uma ou mais mudanças na base de dado sempre que se cria, apaga ou muda-se nos dados de uma tabela está se a fazer uma transação nessa tabela, como tal, o grupo elaborou algumas transações base tais como:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `insereAutoestrada`(IN DescricaoAuto VARCHAR(32), IN TaxaAuto decimal(3,2))
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;

    START TRANSACTION;

    INSERT INTO autoestrada (Descricao, Taxa) VALUES
    (DescricaoAuto,TaxaAuto);

    IF erro THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Autoestrada existente no sistema!';
        ROLLBACK;
    ELSE
        COMMIT;
    END IF;
END
```

Figura 30 - Transação insere autoestrada

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `InsereRegisto`(IN Data_inicial DATETIME,IN Data_final DATETIME,
                            IN Veiculo_id INT, IN Troco_id INT)
BEGIN
    DECLARE montante DECIMAL(6,2);
    DECLARE dist DECIMAL(10,2);
    DECLARE classeV INT;
    DECLARE taxa decimal(3,2);
    DECLARE erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro=1;

    START TRANSACTION;

    SET dist = (SELECT Distancia FROM Troco WHERE Troco_id = id_Troco);
    SET classeV = (SELECT Categoria FROM Veiculo WHERE id_Veiculo = Veiculo_id);
    SET taxa = (SELECT a.taxa FROM Autoestrada a, Troco t WHERE t.id_Troco = Troco_id AND t.Autoestrada_codigo = a.codigo);

    SET montante =  dist * taxa * (1+(1/classeV));

    INSERT INTO registo (Data_inicial,Data_final,Montante,Veiculo_id, Troco_id)
        VALUES
        (Data_inicial,Data_final,montante,Veiculo_id, Troco_id);


    IF erro THEN
        ROLLBACK;
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Erro de inserção!';
    ELSE
        COMMIT;
    END IF;
END
```

Figura 31 - Transação insere registo

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `insereDespesa`(IN data_Despesa DATE,IN valor_Despesa decimal(10,2),IN Descricao_Despesa VARCHAR(256),
                            IN codigo_Auto_Despesa INT)
BEGIN

        DECLARE erro BOOL DEFAULT 0;
        DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;

        START TRANSACTION;

        INSERT INTO despesa(Data,Valor,Descricao,Autoestrada_codigo)
        VALUES(data_Despesa,valor_Despesa,Descricao_Despesa,codigo_Auto_Despesa);

        IF erro THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Despesa existente no sistema!';
        ROLLBACK;
    ELSE
        COMMIT;
    END IF;
END
```

Figura 32 - Transação insere despesa

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `InsereTroco`(IN Entrada VARCHAR(45), IN Saida VARCHAR(45),
                                IN Distancia DECIMAL(10,2), IN Autoestrada INT)
BEGIN

        DECLARE erro BOOL DEFAULT 0;
        DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;

        START TRANSACTION;

        INSERT INTO Troco (Des_Entrada, Des_Saida, Distancia, Autoestrada_codigo)
        VALUES (Entrada, Saida, Distancia, Autoestrada);

        IF erro THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Portagem existente no sistema!';
        ROLLBACK;
    ELSE
        COMMIT;
    END IF;

END
```

Figura 33 - Transação insere troço

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `insereVeiculo`(IN matriculaAuto VARCHAR(8),IN proprietarioAuto VARCHAR(128),IN categoriaAuto INT)
BEGIN
        DECLARE erro BOOL DEFAULT 0;
        DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;

        START TRANSACTION;

        INSERT INTO veiculo(matricula,Proprietario,Categoria)
            VALUES (matriculaAuto,proprietarioAuto, categoriaAuto);

        IF erro THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Veiculo existente no sistema!';
        ROLLBACK;
    ELSE
        COMMIT;
    END IF;
END
```

Figura 34 - Transação insere veículo

5.5. Estimativa dos requisitos de espaço em disco e taxa de crescimento anual

Tendo em conta os domínios definidos e descritos na sua secção apropriada, ou seja, a secção destinada aos atributos que compõem as entidades no sistema, e na ilustração do povoamento no sistema, através de scripts é possível estimar o tamanho ocupado em disco, o seguinte script tem esse mesmo intuito:

```
/* Espaço Ocupado - Individual */
SELECT table_name AS 'Tabela',
    (data_length + index_length) / 1024 AS Tam_kbytes
        FROM information_schema.TABLES
        WHERE table_schema = 'autoestrada'
        ORDER BY Tam_kbytes DESC;
```

Figura 35 - Script de estimativa de espaço ocupado em cada uma das tabelas

Sendo que o resultado de execução do script apresentado na figura 10, é a seguinte tabela:

Tabela	Tam_kbytes
registo	64.0000
despesa	48.0000
troco	48.0000
autoestrada	32.0000
veiculo	32.0000

Figura 36 - Valor em kbytes ocupado em cada tabela

De forma a analisar com mais detalhe o tamanho ocupado pela aplicação no seu todo, elaborou-se um novo script responsável por calcular o tamanho total gasto pela base de dados:

```
/* Espaço Ocupado - Total */
SELECT table_schema,
    SUM((data_length + index_length)) / 1024 AS Tam_kbytes
    FROM information_schema.TABLES
    WHERE table_schema = 'autoestrada';
```

Figura 37 - Script que calcula o tamanho total ocupado

Onde podemos verificar após a execução do script que o tamanho total ocupado é o seguinte:

TABLE_SCHEMA	Tam_kbytes
autoestrada	224.0000

Figura 38 - Tamanho total ocupado

Portanto, o tamanho ocupado em disco é de 224.0000 kbytes, sendo que se prevé uma taxa de crescimente de 200% nos próximos 2 anos.

5.6. Definição e caraterização dos mecanismos de segurança em SQL

Sendo uma base de dados de um um ramo de extrema importância, autoestradas, é importantíssimo assegurar a sua integridade. Desse modo, foi necessário estabelecer uma hieraquia entre utilizadores, utilizadores esses identificados nos requisitos de controlo. Os utilizadores do tipo administrador têm acesso a todos os mecanismos associados a base de dados e a todos os métodos desenvolvidos. Os funcionários apenas têm acesso no que toca ao inserir um veículo na base de dados e a inserir registo podendo também consultar os registos, os utentes apenas podem consultar os seus registos nas autoestradas.

```
CREATE USER 'administrador'@'localhost' IDENTIFIED BY '123456';
CREATE USER 'funcionario'@'localhost' IDENTIFIED BY '123456';
CREATE USER 'cliente'@'localhost' IDENTIFIED BY '123456';

GRANT EXECUTE ON PROCEDURE consultaRegisto TO 'administrador'@'localhost';
GRANT EXECUTE ON PROCEDURE insereAutoestrada TO 'administrador'@'localhost';
GRANT EXECUTE ON PROCEDURE insereDespesa TO 'administrador'@'localhost';
GRANT EXECUTE ON PROCEDURE InsereRegisto TO 'administrador'@'localhost';
GRANT EXECUTE ON PROCEDURE InsereTroco TO 'administrador'@'localhost';
GRANT EXECUTE ON PROCEDURE InsereVeiculo TO 'administrador'@'localhost';
GRANT EXECUTE ON PROCEDURE query1 TO 'administrador'@'localhost';
GRANT EXECUTE ON PROCEDURE query2 TO 'administrador'@'localhost';
GRANT EXECUTE ON PROCEDURE query3 TO 'administrador'@'localhost';
GRANT EXECUTE ON PROCEDURE query4 TO 'administrador'@'localhost';
GRANT EXECUTE ON PROCEDURE query5 TO 'administrador'@'localhost';
GRANT EXECUTE ON PROCEDURE query6 TO 'administrador'@'localhost';
GRANT EXECUTE ON PROCEDURE query7 TO 'administrador'@'localhost';

GRANT EXECUTE ON PROCEDURE InsereVeiculo TO 'funcionario'@'localhost';
GRANT EXECUTE ON PROCEDURE InsereRegisto TO 'funcionario'@'localhost';
GRANT EXECUTE ON PROCEDURE consultaRegisto TO 'funcionario'@'localhost';

GRANT EXECUTE ON PROCEDURE consultaRegisto TO 'cliente'@'localhost';
```

Figura 39 - Permissões relativas aos vários utilizadores

Desta maneira, está assegurado que os vários utilizadores do sistema, apenas têm acesso as funcionalidades atríbuidas a eles próprios.

5.7. Revisão do sistema implementado com o utilizador

Completada a implementação física, o grupo reuniu-se mais uma vez com a administração da empresa, desta vez com vista a completar o trabalho incumbido ao grupo. O sistema satisfaz todos os requisitos e imposições da administração, sendo rápido e eficaz, e como tal a base de dados será implementada num espaço de, no máximo, seis meses.

6. Paradigma NoSQL

6.1 Opção por um sistema NoSQL no GreenWay

Na segunda fase do projeto, foi facultada a proposta de implementar uma BD NoSQL com base no trabalho anteriormente elaborado, gestão de um sistema de autoestradas, apresentando então alguma complexidade no que toca a implementação do SBD. Trata-se portanto, de uma base de dados resultante de um processo de migração da base de dados relacional anteriormente realizada. Foi construída então uma base de dados simples mas eficaz com o intuito de mais uma vez simplificar o processo de gestão de dados bem como o seu armazenamento.

Como foi dito anteriormente, foi elaborada uma base de dados em MongoDB (NoSQL) para a empresa responsável por uma gestão de um sistema de autoestradas GreenWay com base na base de dados elaborada anteriormente.

Esta escolha por uma base de dados não relacional contudo, necessita de ser fundamentada de modo a entender as vantagens que esta traz em relação com a base de dados efetuada na primeira fase do trabalho.

Os Sistemas de Base de Dados relacionais têm dominado o mercado, tornando-se o padrão para a maioria dos Sistemas de Gestão de Base de Dados (SGBD), devido as suas caraterísticas tais como preservar a integridade através da consistência da base de dados e durabilidade. Porém, devido a um elevado crescimento do volume de dados o que implica uma maior dificuldade de conciliar o tipo de modelo com a procura. Têm surgido alternativas uma vez que a utilização destas têm se tornado problemática e não tão eficiente e como tal surgiram as base de dados não relacionais que apresentam uma maior flexibilidade, uma vez que não possuem uma estrutura rígida permitindo com que as alterações na estrutura de aramazenamento destas seja menos demorada que nas base de dados relacionais.

As Base de Dados NoSQL vêm de certa forma a tentar solucionar os diversos problemas relacionados com a escalabilidade, performance e disponibilidade das base de dados relacionais. Promovem uma alternativa com alta capacidade de amarazenamento, velocidade e disponibilidade procurando livrar-se de certas diretrizes que regem a criação de uma base de dados relacional.

Estas, usam diversos modelos de dados entre eles os documentos e grafos por exemplo, o SGBD utilizado, tal como foi referido anteriormente, será o MongoDB, será elaborada uma migração com o objetivo de obter um Sistema de Gestão de Base de Dados capaz de aceitar um elevado tráfego de utilizadores e ao mesmo tempo capaz de assegurar um bom nível de performance.

6.2 MongoDB

Uma das soluções mais utlizadas de maneira a armazenar os dados em base de dados não relacionais são os modelos de dados orientados a documentos, onde cada registo e os seus dados são considerados um documento.

´ O MongoDB, armazena os dados em documentos JSON com um esquema dinâmico, ou seja, os dados armazenados não têm de seguir um esquema rígido possuindo um conjunto de vantagens e desvantagens sendo apresentadas de seguida algumas.

Vantagens

- Os dados como não são estruturados sendo mais facilmente armazenados, as inserções nesta base de dados orientada a documentos apresenta um desempenho consideravelmente melhor que numa base de dados relacional uma vez que não apresenta nenhum *schema* não sendo necessário este ser regido por uma série de considerações.
- A inexistência de relacionamento entre documentos implica também uma inexistência de transações e joins. Nos documentos de modo a colmatar esta desvantagem, possibilta-se todas as informações necessárias. Assim sendo, as consultas são mais simples e rápidas.

Desvantagens

- Como não existem relacionamento entre os documentos, a integridade nestes sistemas não existe pelo que é possível existir bastantes dados repetidos.

Logo, em suma, a escolha do MongoDB deve-se devido à sua inovação no que trouxe no âmbito do armazenamento de estruturas de dados, facilidade de modelação dos objetos nucleares à aplicação juntamente com a sua escalabilidade, com um todo conjunto de funções a ele associado.

6.3. Estrutura base para o sistema de dados NoSQL

De modo a implementar a base de dados em MongoDB baseada na base de dados anteriormente concebida, criou-se apenas uma coleção Autoestrada como se pode ver na figura 40. A autoestrada possui um código, uma descrição e uma taxa, possuirá também uma lista com as despesas associadas, possuindo uma data da despesa, um valor e uma descrição. Possuirá também uma lista com troços os quais possuem um id do troço, uma descrição acerca da entrada e a saída e a distância no qual o troço se encontra na autoestrada. Por fim, possuirá uma lista com os registos no qual juntou-se os atributos da entidade veículo juntamente com o registo ou seja, possuirá uma data inicial e uma data final, um montante, uma matrícula, proprietário e categoria.

6.4. Estrutura do documento

Autoestrada

{

“Código” : código da autoestrada,

“Descrição” : breve descrição textual da autoestrada,

“Taxa” : taxa aplicada na circulação de uma autoestrada,

“Despesas” : [

{

“Data” : data em que foi efetuada a despesa,

“Valor” : quantia associada a uma dada despesa,

“Descrição” : breve descrição textual acerca da despesa efetuada na autoestrada,

}

],

“Troços” : [

{

“id_troco” : código do troço,

“Des_Entrada” : descrição acerca da portagem inicial,

“Des_Saida” : descrição acerca da portagem final,

“Distancia” : extensão de um dado troço,

“Registos” : [

{

“Data_inicial” : data de entrada de um dado utente no troço da autoestrada,

“Data_final” : data de saída de um dado utente no troço da autoestrada,

“Montante” : valor a pago de um dado utente,

“Matricula” : matricula de um dado veículo que circulou na autoestrada,

“Proprietário” : nome do proprietário do veículo,

“Categoria” : categoria do veículo para efeito do cálculo do montante,

}

],

}

],

}

Portanto, cada campo deste documento é baseado na base de dados anteriormente feita, ou seja, cada campo irá ter por base a sua respetiva tabela e atributos da base de dados relacional, de modo a que, o processo de migração ocorra o mais facilmente possível e que a base de dados não relacional seja o mais fiél possível.

É de salvaguardar que, como cada registo está associado a apenas um veículo na base de de dados anterior, decidiu-se juntar o veículo com o registo na *collection*, sendo que cada campo do registo irá também possuir os campos do veículo, novamente baseados nos atributos destas duas entidades da base de dados relacional.

6.5. Migração de dados do MySQL para MongoDB

A migração da BD relacional (MySQL) para uma BD não relacional (MongoDB) pode passar por várias opções no que toca a maneira como os dados são importados. O grupo optou por codificar esta passagem dos dados da BD presentes no MySQL e armazenar no MongoDB utilizando a linguagem de programação Java, devido à sua flexibilidade equanto linguagem de programação (possui extensas librarias), e também é uma das linguagems que o grupo de trabalho se sente mais confortável. Para conectar ambos os SGBDs é necessário o uso de drivers para assim ser facultada a recolha e inserção de dados.

Na extração dos dados da BD relacional e na inserção da BD não relacional, é necessário ter em conta o esquema delineado anteriormente em relação à BD no MongoDB, atavés deste esquema, está delineado a forma de construção e inserção na *Collection* no seu respetivo documento. Esta extração de dados da base de dados relacional consiste em apenas uma operação, esta operação corresponde à única *Collection* delineada pelo grupo de trabalho (“Autoestrada”). Para tal, foi necessário combinar os dados das tabelas do MySQL, através de operações como o *SELECT* e o seu *INNER JOIN*, sendo obtida a collection Autoestrada.

Quanto ao carregamento, os dados são recolhidos de forma a preencher a *Collection*, o processo de inserção em si no MongoDB é bastante simples, limita-se partindo dos dados recolhidos no passo anterior, a formar o documento com o formato pretendido e inserir o mesmo na BD, sendo neste passo garantida a criação do esquema da collection da Autoestrada.

6.6. Implementação do processo de migração de dados

Na migração de uma BD MySQL para uma BD MongoDB, é necessário ter em conta que os relacionamentos entre as tabelas deixam de existir no segundo caso, contudo os dados necessitam na mesma de ser armazenadas de tal maneira que, mesmo sem relações, este corresponda a todas as necessidades anteriormente explicitadas na primeira fase do projeto, uma vez que o propósito da empresa ainda se mantém.

Foi necessário reflexão no que toca ao esquema da BD não relacional, só foi elaborada uma *Collection*, uma vez que a criação de uma única collection, que alberga a informação toda, não iria permitir que os dados fossem repetidos demasiadas vezes, que é uma das desvantagens da utilização das base de dados não relacionais. Isto levou à criação da *Collection* Autoestrada, que para além de possuir a sua própria informação, possui também informação acerca das despesas efetuadas nela, possuindo também informação acerca dos troços que a compoêm e entradas também sobre a circulação nos troços (registos).

Como tal, foi criado o seguinte código de modo a incutir as funcionalidades anteriormente expostas.

```
public static void autoestrada(ResultSet rs) throws SQLException{
    while (rs.next()) {
    int a_codigo = rs.getInt("Codigo");
    String a_descricao = rs.getString("Descricao");
    float a_taxa = rs.getFloat("Taxa");
    System.out.println(a_codigo + "\t" + a_descricao + "\t" + a_taxa);

    List<BasicDBObject> desp = new ArrayList<BasicDBObject>();
    List<BasicDBObject> troc = new ArrayList<BasicDBObject>();

    BasicDBObject ae = new  BasicDBObject("_id", a_codigo)
                .append("descricao", a_descricao)
                                    .append("taxa", a_taxa)
                .append("despesas", desp)
                .append("trocos", troc);

                collection.insert(ae);
}
```

Figura 41 - Código migração da autoestrada

Na figura 41, está apresentado o código responsável pela migração da tabela da autoestrada da base de dados relacional, são extraídos os campos dessa tabela relativo ao código, descrição e taxa, sendo estes depois colocados num *BasicDBObject* de modo a ser adicionados à coleção no MongoDB.

```java
public static void despesa(ResultSet rs) throws SQLException{
    while(rs.next()) {
    int d_id = rs.getInt("id_Despesa");

    java.sql.Date d_sql_data = rs.getDate("Data");
    java.util.Date d_data = new java.util.Date(d_sql_data.getTime());

    float d_valor = rs.getFloat("Valor");
    String d_descricao = rs.getString("Descricao");
    int d_ae_codigo = rs.getInt("Autoestrada_codigo");
    System.out.println(d_id + "\t" + d_data + "\t" + d_valor + "\t" + d_descricao + "\t" + d_ae_codigo);

    BasicDBObject despesa = new BasicDBObject("despesas" ,
                            new BasicDBObject()
                                .append("data", d_data)
                                .append("valor",d_valor)
                                .append("descricao",d_descricao));

    BasicDBObject id = new  BasicDBObject()
                            .append("_id", d_ae_codigo);

    BasicDBObject ats = new BasicDBObject()
                            .append("$addToSet", despesa);
    collection.update(id,ats);
}
```

Figura 42 - Código da migração da despesa

Na figura anterior, está respresentado o código responsável pela transição da tabela da despesa para o MongoDB, novamente são retirados todos os campos dessa tabela, ou seja, o valor, a descrição e o código da autoestrada cuja despesa é relativa, e é colocado tal como no anterior num *BasicDBObject* para ser inserido na collection.

```java
public static void troco(ResultSet rs) throws SQLException{
    while(rs.next()) {
    int t_id = rs.getInt("id_Troco");
    String t_des_entrada = rs.getString("Des_entrada");
    String t_des_saida = rs.getString("Des_saida");
    float t_distancia = rs.getInt("Distancia");
    int t_ae_codigo = rs.getInt("Autoestrada_codigo");
    List<BasicDBObject> reg = new ArrayList<BasicDBObject>();
    System.out.println(t_id + "\t" + t_des_entrada + "\t" + t_des_saida + "\t" + t_distancia + "\t" + t_ae_codigo);

            String query = "SELECT r.id_Registo, r.Data_inicial, r.Data_final, r.Montante, r.Troco_id, r.Veiculo_id, v.id_Veiculo, v.Matricula, v.Proprietario, v.Categoria "
                         + "From autoestrada.Registo r, autoestrada.Veiculo v "
                         + "where r.Veiculo_id = v.id_Veiculo and r.Troco_id = " + t_id;

            Statement stmt2 = con.createStatement();
            ResultSet rs2 = stmt2.executeQuery(query);

            while(rs2.next()){
                java.sql.Date d_sql_data = rs2.getDate("r.Data_Inicial");
                Time t = rs2.getTime("r.Data_Inicial");
                java.util.Date d_data = new java.util.Date(d_sql_data.getTime());
                d_data.setHours(t.getHours());
                d_data.setMinutes(t.getMinutes());
                d_data.setSeconds(t.getSeconds());
                java.sql.Date d_sql_data2 = rs2.getDate("r.Data_Final");
                Time t2 = rs2.getTime("r.Data_Final");

                java.util.Date d_data2 = new java.util.Date(d_sql_data2.getTime());

                d_data2.setHours(t2.getHours());
                d_data2.setMinutes(t2.getMinutes());
                d_data2.setSeconds(t2.getSeconds());
                BasicDBObject registo = new BasicDBObject( new BasicDBObject().append("Data_inicial", d_data)
                                                               .append("Data_final", d_data2)
                                                               .append("Montante", rs2.getFloat("r.Montante"))
                                                               .append("Matricula", rs2.getString("v.Matricula"))
                                                               .append("Proprietario", rs2.getString("v.Proprietario"))
                                                               .append("Categoria", rs2.getInt("v.Categoria")));
                reg.add(registo);
            }
            reg = reg.stream().sorted((d1,d2)-> ((java.util.Date)d1.get("Data_inicial")).compareTo((java.util.Date)d2.get("Data_final"))).collect(Collectors.toList());

    BasicDBObject troco = new BasicDBObject("trocos" ,
                                            new BasicDBObject()
                                            .append("_id", t_id)
                                            .append("entrada", t_des_entrada)
                                            .append("saida",t_des_saida)
                                            .append("distancia",t_distancia)
                                            .append("registos", reg));
    BasicDBObject id = new  BasicDBObject()
                .append("_id", t_ae_codigo);

    BasicDBObject ats = new BasicDBObject()
                .append("$addToSet", troco);
```

Figura 43 - Código da migração do troço, do registo e veículo

Na figura supracitada, é apresentado o código desenvolvido de modo a efetuar a migração dos dados das tabelas relativas ao troço, registo e do veículo, onde, tal como no anterior, são extraídos todos os campos relativos a estas tabelas, com a ajuda de uma query simples, escrita em SQL, que seleciona todos os campos destas três tabelas de modo a ajudar na extração dos dados de cada um destes campos , de seguida é apenas necessário acrescentar estes campos ao *BasicDBObject* de modo a inserir na collection. É de referir também, que os registos na collection encontram-se ordenados por data.

```
public static void main(String[] args) throws ClassNotFoundException,SQLException {

    MongoClient mongoClient = new MongoClient("localhost", 27017);

    DB database = mongoClient.getDB("autoestrada");

    collection = database.getCollection("Autoestrada");

    Class.forName("com.mysql.jdbc.Driver");

    Properties p = new Properties();
    p.put("user","administrador");
    p.put("password","123456");
    con = DriverManager.getConnection(CONNECTION,p);
    Statement stmt = null;

    System.out.println("-----Autoestradas-----\n\ncodigo\tdescricao\ttaxa");
    String query =  "SELECT * FROM autoestrada.Autoestrada";
    stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery(query);
            collection.drop();

            autoestrada(rs);
    query = "SELECT * FROM autoestrada.Despesa";
    rs = stmt.executeQuery(query);
    System.out.println("\n\n-----Despesas-----\n\nid\tdata\tvalor\tdescricao\tAutoestrada_codigo");
    HashMap<Integer,List<BasicDBObject>> listaDespesas = new HashMap<Integer,List<BasicDBObject>>();

    despesa(rs);

    query = "SELECT * FROM autoestrada.Troco";
    rs = stmt.executeQuery(query);
    System.out.println("\n\n-----Trocos-----\n\nid\tentrada\tsaida\tdistancia\tAutoestrada_codigo");

    troco(rs);

    con.close();
}
```

Figura 44 - Código da main

Na main, são efetuadas as conecções tanto ao MongoDB como ao SQL, de modo a aceder a ambas as bases de dados, sendo depois executadas as funções anteriormente apresentadas, da figura 41 a 43, de modo a completar a migração de dados do SQL para o MongoDB.

6.7. Interrogações no MongoDB

Relativamente as questões identificadas anteriormente, o grupo de trabalho desenvolveu um conjunto de queries em MongoDB baseadas nas queries desenvolvidas em SQL, de modo a ajustar esta base de dados à neceidade da empresa. A lógica por detrás destas queries e o funcionamento é exatamente igual às queries SQL apenas adaptada à linguagem utilizada no MongoDB.

```
Criação da query1:
db.system.js.save({
                _id: "query1",
                    value : function(num_aut,d_in,d_fin){return db.Autoestrada.distinct(
                                                                                        "trocos.registos.Matricula",
                                                                                        {
                                                                                            _id: num_aut,
                                                                                            "trocos.registos.Data_inicial":{"$gte": new Date(d_in)},
                                                                                            "trocos.registos.Data_final":{"$lte": new Date(d_fin)}
                                                                                        }
                                                                                        ).length;}
                })
```

Figura 45 – Função responsável por responder à query 1

```
> query1(1,"2017-01-01","2017-12-31")
6
```

Figura 46 - Resultado da query 1

```
db.system.js.save(
        {_id: "query2",
            value : function(ano){
    var resultado = []
    var balanco = 0
    var despesas = 0
    var ganhos = 0

    db.Autoestrada.aggregate([{$unwind: "$despesas"},
                              {$group: {_id:{$year: "$despesas.data"} , total_gasto: {$sum:"$despesas.valor"}}},
                              {$match: {_id: ano}}])
                              .forEach(function(u){despesas = u.total_gasto})

    db.Autoestrada.aggregate([{$unwind: "$trocos"},{$unwind: "$trocos.registos"},
                              {$group: {_id: {$year: "$trocos.registos.Data_final"}, total_ganho:{$sum:"$trocos.registos.Montante"}}},
                              {$match: {_id: ano}}])
                              .forEach(function(u){ganhos = u.total_ganho})

    balanco = ganhos-despesas

    var d = []
    d.push("Gastos")
    d.push(despesas)

    var g = []
    g.push("Ganhos")
    g.push(ganhos)

    var b = []
    b.push("Balanco")
    b.push(balanco)

    resultado.push(d)
    resultado.push(g)
    resultado.push(b)

    return resultado
}})
```

Figura 47 - Função responsável por responder à query 2

```
> query2(2016)
[ [ "Gastos", 0 ], [ "Ganhos", 0 ], [ "Balanco", 0 ] ]
>
>
>
>
>
> query2(2017)
[
        [
                "Gastos",
                79000
        ],
        [
                "Ganhos",
                557.4100013971329
        ],
        [
                "Balanco",
                -78442.58999860287
        ]
]
>
>
>
>
> query2(2020)
[ [ "Gastos", 0 ], [ "Ganhos", 0 ], [ "Balanco", 0 ] ]
>
```

Figura 48 - Resultado da query 2

```
Criação da query3:
db.system.js.save({
                _id: "query3",
                value: function(){
                    return db.Autoestrada.aggregate([
                        {$unwind : "$trocos"},
                        {$unwind : "$trocos.registos"},
                        {
                            $group : {
                                _id : {
                                    "matricula" : "$trocos.registos.Matricula",
                                    "proprietario" : "$trocos.registos.Proprietario"
                                },
                                total : { $sum : "$trocos.registos.Montante" }
                            }
                        },
                        {
                            $sort : { total : -1 }
                        },
                        {
                            $limit : 5
                        },
                        {
                            $project: {
                                _id : 0,
                                matricula : "$_id.matricula",
                                proprietario : "$_id.proprietario",
                                total : 1
                            }
                        }
                    ]);
                    }

})
```

Figura 49 - Função responsável por responder à query 3

```
> db.loadServerScripts()
> query3()
{ "total" : 49.44000053405762, "matricula" : "87-LO-21", "proprietario" : "João Diaz de Teixeira" }
{ "total" : 45.39999961853027, "matricula" : "QE-78-07", "proprietario" : "Vitória Cavalcanti Fernandes" }
{ "total" : 32.730000019073486, "matricula" : "23-AY-76", "proprietario" : "Estevan Oliveira Goncalves" }
{ "total" : 32.239999771118164, "matricula" : "BV-31-22", "proprietario" : "Joana Martins" }
{ "total" : 30.770000457763672, "matricula" : "PL-21-74", "proprietario" : "Flávio dos Dias" }
```

Figura 50 - Resultado da query 3

```
Criação da query4:
db.system.js.save({
                _id: "query4",
                value: function(){
                    return db.Autoestrada.aggregate([
                                    {$unwind : "$trocos"},
                                    {$unwind : "$trocos.registos"},
                                    {
                                        $group : {
                                            _id : "$trocos.registos.Matricula",
                                            total : { $sum : "$trocos.distancia" },
                                        }
                                    },
                                    {
                                        $sort : { total : -1 }
                                    },
                                    {
                                        $limit : 10
                                    },
                                    {
                                        $project: {
                                            _id : 0,
                                            matricula : "$_id",
                                            total : 1
                                        }
                                    }
                                ]);
                }
})
```

Figura 51 - Função responsável por responder à query 4

```
> db.loadServerScripts()
> query4()
{ "total" : 171, "matricula" : "99-87-CP" }
{ "total" : 168, "matricula" : "BV-31-22" }
{ "total" : 164, "matricula" : "95-31-AP" }
{ "total" : 149, "matricula" : "PX-52-34" }
{ "total" : 147, "matricula" : "59-RM-99" }
{ "total" : 135, "matricula" : "23-AY-76" }
{ "total" : 134, "matricula" : "PL-21-74" }
{ "total" : 130, "matricula" : "99-99-RE" }
{ "total" : 125, "matricula" : "BC-65-74" }
{ "total" : 124, "matricula" : "34-74-CD" }
>
```

Figura 52 - Resultado da query 4

```
Criação da query5:
db.system.js.save({
            _id:"query5",
            value: function(){
                return db.Autoestrada.aggregate([
                                {$unwind : "$trocos"},
                                {$unwind : "$trocos.registos"},
                                {
                                    $sort : { "trocos.registos.Categoria" : 1 }
                                },
                                {
                                    $project: {
                                        _id : 0,
                                        data_inicial : "$trocos.registos.Data_inicial",
                                        data_final : "$trocos.registos.Data_final",
                                        montante : "$trocos.registos.Montante",
                                        categoria : "$trocos.registos.Categoria"
                                    }
                                }
                            ]);
            }
})
```

Figura 53 - Função responsável por responder à query 5

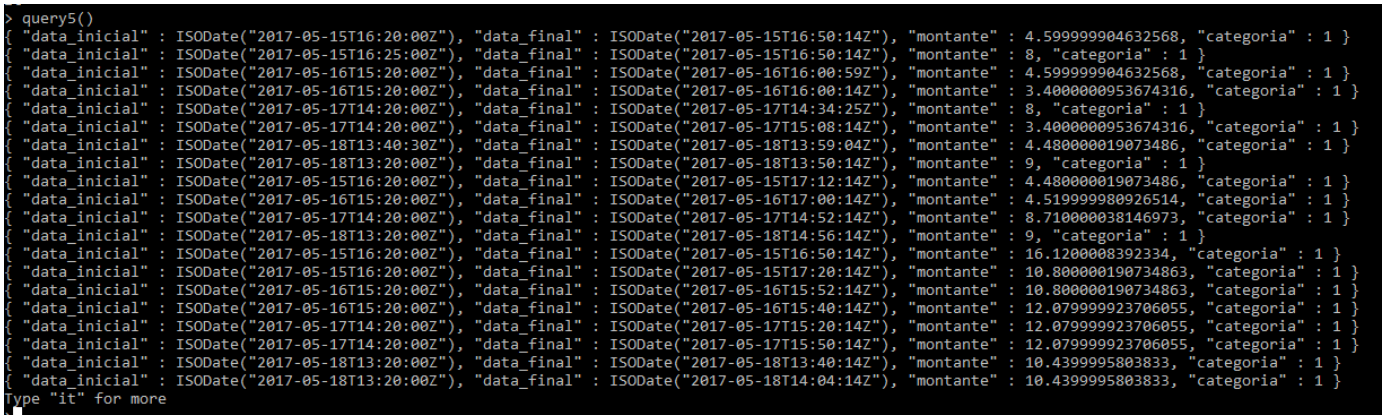

Figura 54 - Resultado da query 5

```
Criação da query6:
db.system.js.save({
            _id: "query6",
            value: function(descr){ return db.Autoestrada.aggregate([
                                                                    { $unwind: "$despesas"},
                                                                    { $match: {"despesas.descricao": {$regex: descr}}},
                                                                    {
                                                                        $group: {
                                                                            _id : null,
                                                                            Custo_total : { $sum: "$despesas.valor"}
                                                                        }
                                                                    },
                                                                    {
                                                                        $project : {
                                                                        _id : 0,
                                                                        Custo_total : 1
                                                                        }
                                                                    }
                                                                ]);
            }
})
```

Figura 55 - Função responsável por responder à query 6

```
> db.loadServerScripts()
> query6("Limpeza")
{ "Custo_total" : 5000 }
>
```

Figura 56 - Resultado da query 6

```
db.system.js.save(
        {_id: "query7",
            value : function(ano){
    var resultado = []

    var ids = db.Autoestrada.find({},{_id:1})

    ids.forEach(function(u){resultado.push(query7Aux(u._id, ano))})

    return resultado
```

Figura 57 - Função responsável por responder à query 7

```
db.system.js.save(
        {_id: "query7Aux",
            value : function(id, ano){
    var resultado = []
    var balanco = 0
    var despesas = 0
    var ganhos = 0

    db.Autoestrada.aggregate([{$unwind: "$despesas"},
                              {$group: {_id: {id: "$_id",ano: {$year: "$despesas.data"}} , total_gasto: {$sum:"$despesas.valor"}}},
                              {$match: {"_id.ano": ano}},
                              {$match: {"_id.id": id}}])
                              .forEach(function(u){despesas = u.total_gasto})

    db.Autoestrada.aggregate([{$unwind: "$trocos"},
                              {$unwind: "$trocos.registos"},
                              {$group: {_id:  {id: "$_id",ano: {$year: "$trocos.registos.Data_final"}}, total_ganho:{$sum:"$trocos.registos.Montante"}}},
                              {$match: {"_id.ano": ano}},
                              {$match: {"_id.id":id}}])
                              .forEach(function(u){ganhos = u.total_ganho})

    balanco = ganhos-despesas

    var auto = []
    auto.push("autoestrada")
    auto.push(id)

    var d = []
    d.push("Gastos")
    d.push(despesas)

    var g = []
    g.push("Ganhos")
    g.push(ganhos)

    var b = []
    b.push("Balanco")
    b.push(balanco)

    resultado.push(auto)
    resultado.push(d)
    resultado.push(g)
    resultado.push(b)

    return resultado
}})
```

Figura 58 - Função auxiliar criada para ajudar na resposta à query 7

Figura 59 - Resultado da query 7

7. Conclusões e trabalho futuro

Em conclusão, o balanço que o grupo de trabalho pode fazer sobre o panorma do trabalho é positivo, onde todos os passos no desenvolvimente de uma base de dados foram seguidos à risca. Dede a recolha de requisitos, passando pelo modelo conceptual, modelo lógico e a implementação de um modelo físico num SGBD.

A etapa inicia, que consiste em levantar requisitos junto da empresa revelou-se como sendo fulcral, uma vez que serviu como o génesis para a construção dos consequentes modelos. No entanto, ao longo da concepção desses tais modelos, existiram várias reavaliações dos requistos e foram concebidos vários modelos, houve uma certa dificuldade no que tocou à interpretação do problema em questão. No que toca ao modelo conceptual numa primeira fase decorreu tudo como previsto, foi apenas na tradução para o modelo lógico que começaram a ocorrer alguns imprevistos, especialmente no que toca da relação do troço com o registo. Numa primeira fase o grupo de trabalho ao invés de identificar o troço, identificou a entidade da portagem o que fez que houvesse uma relação dupla com o registo, de modo a guardar os ids das portagens iniciais e finais, o que fez com que na transformação para o modelo lógico não pudesse ser validado por normalização, não cumpre a 3ª forma normal. Como tal, o grupo identificou uma nova entidade, troço, que substituiu a antiga tanto no modelo conceptual como no físico, a partir deste momento o trabalho prosseguiu como esperado. No que toca ao modelo físico, houve algumas dificuldades relativamente à utilização do SGBD escolhido pelo grupo, o *MySQL*, devido à inexperiência dos elementos relativamente à ferramenta, para além disto, houve também dificuldade relativamente aos procedimentos mais complexos, como por exemplo, mecanismos de segurança, onde o grupo não conseguiu efetuar, e até mesmo no que toca as permissões dos utilizadors, que poderiam ter sido melhor desenhadas. As permissões poderiam ser refinadas no que toca ao nível dos procedimentos e não das tabelas. No que toca as transações o grupo também poderia ter aprofundado melhor esta parte, poderiam ter sido feitas mais transações no que toca à alteração ou até mesmo remoção, visto que traria mais funcionalidades à aplicação.

Na segunda fase do projeto, finda a apresentação e a explicitação das etapas que, na sua ação conjunta, permitem uma completa e correta migração dos dados do modelo relacional para o modelo não relacional e ainda as posteriores interrogações contudo, todo este processo levantou sérios problemas ao grupo de trabalho, problemas esses que foram ultrapassados, tendo construído uma base de dados não relacional final competente, eficaz e escalável.

No que toca ao processo em si da construção da base de dados não relacional, como foi efetuada uma migração de uma base de dados já existente, o grupo não se prendeu com problemas relacionados com a contextualização e definição do problema, uma vez que estes foram tratados aquando da concepção do sistema segundo uma abordagem relacional, de seguida foi necessário conhecer melhor o paradigma não relacional, ou seja, o conceito de documento, collection, e todas as outras funcionalidades que estes providenciam, o que provou como sendo um entrave para o grupo, devido à inexperiência em trabalhar com base de dados

não relacionais. Contudo, com o devido *Self-learning* eventualmente todo este processo se tornou mais fácil, existindo mesmo assim dificuldades na transição das queries escritas em SQL para MongoDB.

No entanto, a transição da base de dados relacional para uma base de dados não relacional em MongoDB, provou-se como sendo um passo simples, foi escrito um código na linguagem Java com o intuito de efetuar o parse dos dados, sendo depois esses inseridos na collection no MongoDB.

Tendo em conta o que foi dito, mesmo assim, o grupo considera que elaborou um sistema de base de dados relacional e não relacional capaz e conceptualmente projetado para uma expensão futura e preparado para sustentar com rigor e correção os requisitos subjacentes à gestão de dados de um sistema de autoestradas tanto no paradigma relacional como no paradigma não relacional.

Referências

- Connolly, T. And Begg, C. (2002). Database systems. 1st ed. Harlow, England: Addison-Wesley.
- https://docs.mongodb.com/manual/

Lista de Siglas e Acrónimos

BD	Base de Dados
SGBD	Sistema de Gestão de Base de Dados