Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

Unidade Curricular de Bases de Dados

Ano Letivo de 2022/2023

Auto VrumVrum

Francisco Pinto Lameirão A97504
João Machado Gonçalves A97321
Lara Beatriz Pinto Ferreira A95454
Matilde Maria Ferreira de Sousa Fernandes A95319

2 de fevereiro de 2024

BD

Data de Receção	
Responsável	
Avalição	
Observações	

Auto VrumVrum

Francisco Pinto Lameirão A97504
João Machado Gonçalves A97321
Lara Beatriz Pinto Ferreira A95454
Matilde Maria Ferreira de Sousa Fernandes A95319

2 de fevereiro de 2024

Resumo

Neste relatório, foi desenvolvido um sistema de base de dados no âmbito da unidade curricular de Bases de Dados, com o foco principal na análise, planeamento, modelação, arquitetura e implementação de sistemas deste tipo. O objetivo do sistema é implementar uma base de dados para um stand, capaz de suportar grandes volumes de informação com facilidade de acesso, compreensão e organização. Para garantir uma estrutura eficiente do sistema, o processo foi dividido em três etapas distintas.

Começamos por estabelecer os requisitos essenciais que a nossa base de dados deveria suportar. Em seguida, elaboramos um modelo conceptual, identificando as entidades-chave e os relacionamentos entre elas, com base nos requisitos definidos anteriormente. Após a conclusão da modelação conceptual, passamos para a segunda fase.

Na segunda fase, procedemos à construção do modelo lógico a partir do modelo conceptual. Realizamos uma análise e validação detalhadas, utilizando técnicas de normalização e considerando as interrogações do utilizador. Somente após verificar que o modelo lógico estava pronto, avançamos para a terceira etapa.

A terceira fase consistiu em gerar um modelo físico no MySQL Workbench, baseado no modelo lógico previamente desenvolvido. Essa etapa garantiu que as modelações anteriores convergissem para a criação de uma base de dados correta e segura. Além disso, criamos um script em Python para importar ficheiros para a base de dados e possibilitar a análise de dados através do Power BI.

Após a conclusão dessas fases, consideramos o projeto como finalizado.

Área de Aplicação:

Palavras-Chave: Bases de Dados, Bases de Dados Relacionais, Modelo Concetual, Modelo Lógico, Modelo Físico, MySQL, MySQL Workbench, brModelo, Entidades, Relacionamentos, Atributos, Requisitos, Índices SQL, Vistas SQL.

Índice

Lista de Figuras

Lista de Tabelas

1 Definição do Sistema

1.1 Contextualização

A "AutoVrumVrum" é um stand de carros localizado no concelho de Vila Verde fundado pelo empreendedor Xavier Mota em 2009, devido ao seu grande amor por automóveis. O stand oferece um leque variado em termos de tipos de carros, tanto em preço como formato.

Os funcionários que trabalham no stand são vendedores ou compradores, ou seja, ou são responsáveis pela venda de carros a clientes ou a compra de carros a fornecedores. Ambos os tipos de funcionários também estão responsáveis pelo agendamento e a realização de Test-Drives por parte de clientes interessados.

Inicialmente, o stand teve bastante sucesso, pois o Xavier era conhecido por tratar muito bem a sua clientela mais leal, oferecendo prendas e descontos exclusivos aos clientes regulares dos seus estabelecimentos.

Nos últimos anos, Xavier tem estado mais ocupado com os seus outros projetos, por isso parou de ter tanto controlo sobre o funcionamento do dia-a-dia do stand, confiando nos seus funcionários para manterem o negócio rentável.

1.2 Fundamentação

Desde que tem estado mais ausente, Xavier reparou que tem tido mais queixas da qualidade do serviço e muitos dos seus clientes (especialmente os que compravam mais regularmente automóveis) tinham decidido ir a um dos seus competidores.

Enquanto investigava a razão por trás deste declínio, Xavier reparou que havia uma falta de organização grave em relação aos registos das compras, vendas e carros do stand, pois estes ainda eram feitos em livros de registo, que não era um problema tão grande quando o stand era mais pequeno, mas desde que Xavier se ausentou do seu trabalho tem-se verificado cada vez mais erros por parte dos funcionários. Outro problema que surgiu com a ausência do Xavier foi a falta do contacto que anteriormente tinha com os seus clientes

1.3 Motivação e Objetivos

Após ser informado dos vários problemas que o seu stand se deparava com, Xavier decidiu que era necessário modernizar a forma como se guardava os registos da loja. Assim, propôs o desenvolvimento de um projeto que envolvesse um sistema que suportasse o seu negócio.

O Xavier quer, através deste sistema, poder saber quais são os seus clientes mais habituais sem ter de estar constantemente presente no stand. Também quer reduzir ao máximo os erros que os seus funcionários podem cometer ao fazer registos, visto que isto já resultou em várias queixas por parte de clientes.

O nosso objetivo final é disponibilizar uma ferramenta de gestão interna, que permita ao stand uma melhor organização e acesso à informação relativa a compras, vendas, carros, etc. . A base de dados desenvolvida irá possibilitar uma melhor análise estatística que poderá influenciar futuras decisões de negócio por parte de Xavier.

1.4 Análise da Viabilidade do Processo

Para podermos analisar a viabilidade deste projeto, teremos de ter em conta o custo dos recursos necessários para a sua realização e as vantagens que o mesmo traria.

Apesar de haver a necessidade de um investimento inicial, é expectável que este seja retornado através da redução de custos associados a funcionários. Em termos práticos, o projeto irá reduzir a quantidade de erros humanos nas operações dia a dia, resultando assim numa melhor experiência para os clientes da AutoVrumVrum.

Visto que o povoamento inicial da base de dados não irá incluir vendas e compras feitas anteriormente (excluindo as relacionadas a carros atualmente no stand), não será muito dispendioso em termos de recursos humanos.

Os custos de manutenção da base de dados serão, em princípio, bastante reduzidos visto que não será muito complexa e terá um número pequeno de registos (comparativamente a, por exemplo, um restaurante). Também devido à sua simplicidade, a manutenção terá custos humanos pequenos.

Após esta análise, concluímos que as vantagens que a implementação da base de dados traria seriam muito maiores que as suas desvantagens. Concluímos assim que este projeto é viável.

1.5 Recursos e Equipa de Trabalho

Em termos de recursos humanos, necessitaremos da Equipa de Desenvolvimento, o CEO Xavier Mota e os Funcionários do Stand. Também será necessário garantir que pelo menos alguns dos Funcionários do Stand conseguem fazer manutenção básica na base de dados, caso haja algum problema trivial.

Em termos de recurso materiais, será necessário um servidor (físico ou na *cloud*) que tenha os recursos necessários para hospedar a base de dados. Também será necessário um Sistema de Gestão de Base de Dados (SGDB). Para este projeto, decidimos utilizar o MySQL. Os modelos que iremos elaborar ao longo do projeto serão feitos no MySQL Workbench e no brmodelo

Para o sistema de Recolha de Dados e o de Painéis de Análise iremos criar um programa usando a linguagem Python e o Microsoft Power BI Desktop, respetivamente.

1.6 Plano de Execução do Projeto

Uma vez que este projeto exige um planeamento rígido de todas as etapas do seu desenvolvimento, recorremos a um diagrama de Gantt para ilustrar todo este processo.

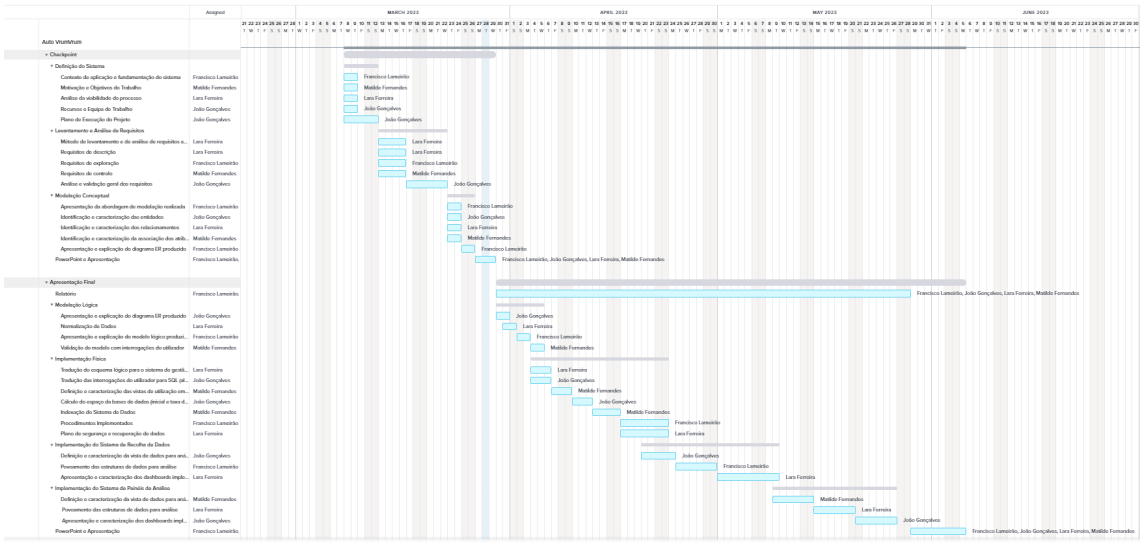

Figura 1.1: Plano de Execução do Projeto

2 Levantamento e Análise de Requisitos

2.1 Estratégia de Levantamento de Requisitos

Para obter os requisitos necessários, adotamos uma estratégia de levantamento de requisitos que envolveu a convocação de várias reuniões com o gerente do stand. O principal objetivo dessas reuniões foi obter informações detalhadas sobre o funcionamento do stand e esclarecer quaisquer dúvidas que surgissem ao longo do processo.

Durante as reuniões, abordamos diversas questões importantes para o desenvolvimento do sistema de base de dados do stand. Exploramos tópicos como a gestão do stock, o processo de venda e compra de veículos, as interações com os clientes, os procedimentos de pós-venda e as necessidades de relatórios, análises e estatísticas.

Tentamos compreender as necessidades específicas do gerente do stand e obter um panorama completo do fluxo de trabalho e dos requisitos do sistema, solicitamos exemplos práticos para melhor compreender os processos envolvidos. Além disso, também exploramos as restrições relacionados à infraestrutura do stand, como a disponibilidade de recursos de hardware.

Obtemos informações claras e detalhadas para orientar o desenvolvimento do sistema de base de dados do stand, garantindo que o sistema atenda de forma precisa e eficiente ás necessidades e expectativas do gerente do stand.

2.2 Requisitos de Descrição

AutoVrumVrum

D

Processo de Desenvolvimento do Sistema de Bases de Dados

Levantamento de Requisitos Versão: 1.00/2023

Documento de Requisitos de Descrição março de 2023

Nº	Data e Hora	Descrição	Área/Vista	Fonte
RD1	15/03/2023 16:31	Cada cliente deve ser registado com um número sequencial (valor único) com 6 digitos.	Clientes	Stand
RD2	15/03/2023 16:33	É necessário armazenar dados do cliente nomeadamente nome, sexo, data de nascimento, morada, telefone, email, nº de contribuinte (NIF).	Clientes	Stand
RD3	16/03/2023 19:02	Um carro tem de ser identificado pelo id, marca, modelo, ano, kilometros, cilindrada, combustivel, preço e estado.	Carro	Stand
RD4	16/03/2023 22:08	Cada venda ao cliente deve ser registada com um número sequencial (valor único), data da venda, id do cliente, id do funcionário associado, id do veículo vendido e valor da venda.	Vendas	Stand
RD5	16/03/2023 22:08	É necessário armazenar dados do funcionário nomeadamente id, nome, sexo, data de nascimento, morada, telefone, email, nº de contribuinte (NIF) e tipo de funcionário.	Funcionários	Stand
RD6	22/03/2023 18:31	Os tests drives devem conter o nome do cliente, nome do funcionário, data de ocorrência, hora de ocorrência, id do carro.	Test Drive	Stand
RD7	22/03/2023 18:31	É necessário armazenar dados do fornecedor nomeadamente nome da empresa, morada, telemóvel e tipo.	Fornecedor	Stand
RD8	22/03/2023 18:31	Cada compra a um fornecedor deve ser registada com um número sequencial (valor único), data da compra, id do veículo comprado e valor da compra, NIF .	Compras	Stand
RD9	22/03/2023 18:31	O sistema deve ser capaz de classificar os funcionários do stand em três tipos diferentes: comprador, vendedor e proprietário do Stand.	Funcionários	Stand

Figura 2.1: Requisitos de Descrição

2.3 Requisitos de Manipulação

AutoVrumVrum — M

Processo de Desenvolvimento do Sistema de Bases de Dados

Levantamento de Requisitos — Versão: 1.00/2023

Documento de Requisitos de Manipulação — março de 2023

Nº	Data e Hora	Descrição	Área	Fonte	Analista
RM1	16/03/2023 19:07	Deve ser possível obter a lista de todos os carros disponiveis no stand.	Carros	Stand	
RM2	18/03/2023 22:28	Deve ser possível obter a lista de todos os carros vendidos no stand.	Carros	Stand	
RM3	22/03/2023 18:32	Deve ser possível obter a lista de todos os funcionarios do stand.	Funcionário	Stand	
RM4	22/03/2023 18:32	Deve ser possível obter a lista de todos os clientes do stand.	Cliente	Stand	
RM5	22/03/2023 18:32	Deve ser possível obter a lista de todos os carros disponiveis de uma certa marca no stand.	Carros	Stand	
RM6	22/03/2023 18:32	Deve ser possível obter o histórico de compras de um dado cliente.	Cliente	Stand	
RM7	22/03/2023 18:33	Deve ser possível inserir Carros, Clientes, Funcionários, Fornecedores, Vendas, Test Drives e Compras.	Carros/Clientes/Funcionários/Fornecedor/Vendas/Test Drive/Compras	Stand	
RM8	22/03/2023 18:33	Aceder ao histórico de test drives do cliente.	Test Drive/Cliente	Stand	
RM9	22/03/2023 18:33	Aceder a lista das marcas mais vendidas até ao momento.	Carros	Stand	
RM10	22/03/2023 18:33	Aceder as receitas mensais do stand até ao momento.	Vendas	Stand	
RM11	22/03/2023 18:47	Deve ser possível obter a lista de todos os fornecedores do stand.	Fornecedor	Stand	
RM12	22/03/2023 18:47	Deve ser possível obter o histórico de compras feitas a um dado fornecedor.	Compras/Fornecedor	Stand	

Figura 2.2: Requisitos de Manipulação

2.4 Requisitos de Controlo

AutoVrumVrum — C

Processo de Desenvolvimento do Sistema de Bases de Dados

Levantamento de Requisitos — Versão: 1.00/2023

Documento de Requisitos de Controlo — março de 2023

Nº	Data e Hora	Descrição	Área/Vista	Fonte	Analista
RC1	22/03/2023 18:45	Apenas os funcionários vendedores podem registar vendas.	Funcionário/Vendas	Stand	
RC2	22/03/2023 18:45	Apenas os funcionários compradores e vendedores podem adicionar, editar e remover os carros disponiveis no stand.	Funcionário/Carros	Stand	
RC3	22/03/2023 18:45	Apenas os funcionários vendedores podem adicionar, editar e remover clientes do stand.	Funcionário/Clientes	Stand	
RC4	22/03/2023 18:45	Apenas o proprietário pode adicionar, editar e remover funcionários de todos os tipos	Funcionário	Stand	
RC5	22/03/2023 18:45	Apenas o funcionário vendedor pode agendar Test drives	Funcionário/Test drives	Stand	
RC6	23/03/2023 18:45	Apenas os funcionários compradores podem adicionar, editar e remover fornecedores	Funcionário/Fornecedores	Stand	

Figura 2.3: Requisitos de Controlo

2.5 Validação dos requisitos estabelecidos

Após a fase de levantamento de requisitos, é de extrema importância proceder à validação dos mesmos, a fim de assegurar a sua precisão, relevância e viabilidade. A validação dos requisitos estabelecidos constitui um passo fundamental para garantir que o sistema atenda de forma adequada ás necessidades e expectativas do gerente do stand.

Nesse contexto, foi realizada uma nova reunião com o gerente, com o propósito de verificar se os requisitos capturados refletem de maneira precisa as necessidades e os objetivos do projeto. Durante essa reunião, em conjunto com o gerente, procedemos à análise minuciosa dos requisitos, avaliando se estes eram claros, compreensíveis e livres de contradições ou ambiguidades.

Após uma análise cuidadosa, não foram identificadas quaisquer falhas ou problemas nos requisitos estabelecidos. Dessa forma, não foi necessário realizar alterações ou ajustes após a reunião.

A validação bem-sucedida dos requisitos proporciona uma base sólida para o desenvolvimento do sistema de base de dados, garantindo que seja desenvolvido de acordo com as expectativas e objetivos do gerente do stand.

3 Modelação Conceptual

3.1 Apresentação da abordagem de modelação realizada

Após o levantamento dos requisitos necessários, a nossa equipa iniciou a modelação conceptual. Uma representação visual clara e coerente, tem bastante importância para guiarmos o desenvolvimento do sistema de base de dados que queremos realizar.

Esta etapa é fundamental em qualquer projeto, pois proporciona uma visão abrangente do sistema e de como as diferentes entidades se relacionam entre si. Além disso, o modelo conceptual permite-nos antecipar e identificar problemas antes mesmo de começar a implementação. Através do modelo, é possível simular diferentes cenários e analisar o impacto de possíveis alterações,o que nos permite fazer decisões informadas.

É essencial que o modelo conceptual aborde as entidades, suas relações e atributos de acordo com os requisitos levantados anteriormente. Dessa forma, podemos descrever o comportamento do sistema e obter uma representação clara da estrutura da base de dados do stand de automóveis. Essa representação visual facilita a gestão das informações e contribui para um sistema mais eficiente.

Com um modelo bem elaborado, podemos garantir uma implementação eficiente e uma boa gestão das informações do sistema de base de dados do stand.

3.2 Identificação e caracterização das entidades

Entidades	Descrição	Ocorrência
Cliente	Informações pessoais acerca dos clientes do stand	Os clientes têm um registo único que é obrigatório para realizarem compras e fazerem Test drives
Funcionário	Termo geral responsável por manter o funcionamento do stand	Dependendo do tipo trabalha como vendedor ou comprador
Fornecedor	Termo geral responsável pelo fornecimento de carros	Fornece os carros para o stand
Vendas	Representa a venda dos carros realizadas aos clientes	Cada venda tem um id que é atribuído no momento do seu registo
Compra	Representa as compras de carros realizadas ao fornecedor	Cada compra tem um id que é atribuído no momento do seu registo
Carro	Informações acerca dos carros dos stand	Os carros têm um registo único que é obrigatório para pertecerem ao stand
Test Drive	Representa um Test drive realizado ou por realizar de um cliente	Cada Test drive tem um id que é atribuído no momento do seu registo

Tabela 3.1: Identificação das entidades

3.3 Identificação e caracterização dos relacionamentos

Entidades	Multiplicidade	Relacionamento	Multiplicidade	Entidade
Cliente	(1,1)	pede	(0,N)	Test Drive
Funcionário	(1,1)	agenda	(0,N)	Test Drive
Fornecedor	(1,1)	vende	(1,N)	Carro
Venda	(0,N)	é feita	(1,1)	Cliente
Compra	(1,1)	inclui	(1,N)	Carro
Funcionário	(1,1)	realiza	(0,N)	Venda
Funcionário	(1,1)	realiza	(0,N)	Compra
Compra	(0,N)	é feita	(1,1)	Fornecedor
Venda	(1,1)	inclui	(1,1)	Carro
Test Drive	(1,N)	pode ter	(1,1)	Carro

Tabela 3.2: Relacionamentos entre entidades

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

Entidades	Atributos	Domínio	Opcional
Cliente	ID	INT	Não
	Nome	VARCHAR(75)	Não
	Genero	VARCHAR(1)	Não
	Data de Nascimento	DATE	Não
	Contactos - Telemovel	INT	Não
	Contactos - Email	VARCHAR(90)	Não
	NIF	INT	Não
	Morada	VARCHAR(75)	Não
Funcionário	ID	INT	Não
	Nome	VARCHAR(75)	Não
	Genero	VARCHAR(50)	Não
	Data de Nascimento	DATE	Não
	NIF	INT	Não
	Tipo	ENUM(...)	Não
	Contactos - Telemovel	INT	Não
	Contactos - Email	VARCHAR(90)	Não
Fornecedor	ID	INT	Não
	Nome	VARCHAR(75)	Não
	Tipo	ENUM(...)	Não
	NIF	INT	Não
	Contactos - Telemovel	INT	Não
	Contactos - Email	VARCHAR(90)	Não
Carro	ID	INT	Não
	Marca	VARCHAR(50)	Não
	Modelo	VARCHAR(75)	Não
	Ano	INT	Não
	Kilometros	INT	Não
	Cilindrada	INT	Não
	Combustivel	ENUM(...)	Não
	Preco	INT	Não
	Estado	VARCHAR(75)	Não
Compra	ID	INT	Não
	Data Compra	DATE	Não
	Preco	INT	Não

Tabela 3.3: Identificação dos Atributos de cada entidade

Entidades	Atributos	Domínio	Opcional
Venda	ID	INT	Não
	Data Venda	DATE	Não
	Preco	INT	Não
Test Drive	ID	INT	Não
	Data Ocorrencia	DATE	Não

Tabela 3.4: Identificação dos Atributos de cada entidade

3.5 Apresentação e explicação do diagrama conceptual produzido

Seguindo a metodologia para a criação de um SGBD, criamos um modelo conceptual que suportasse os requisitos previamente levantados. Após a definição pormenorizada de todas as entidades existentes no sistema, dos relacionamentos entre elas e dos seus respetivos atributos, foi criado o seguinte modelo conceptual.

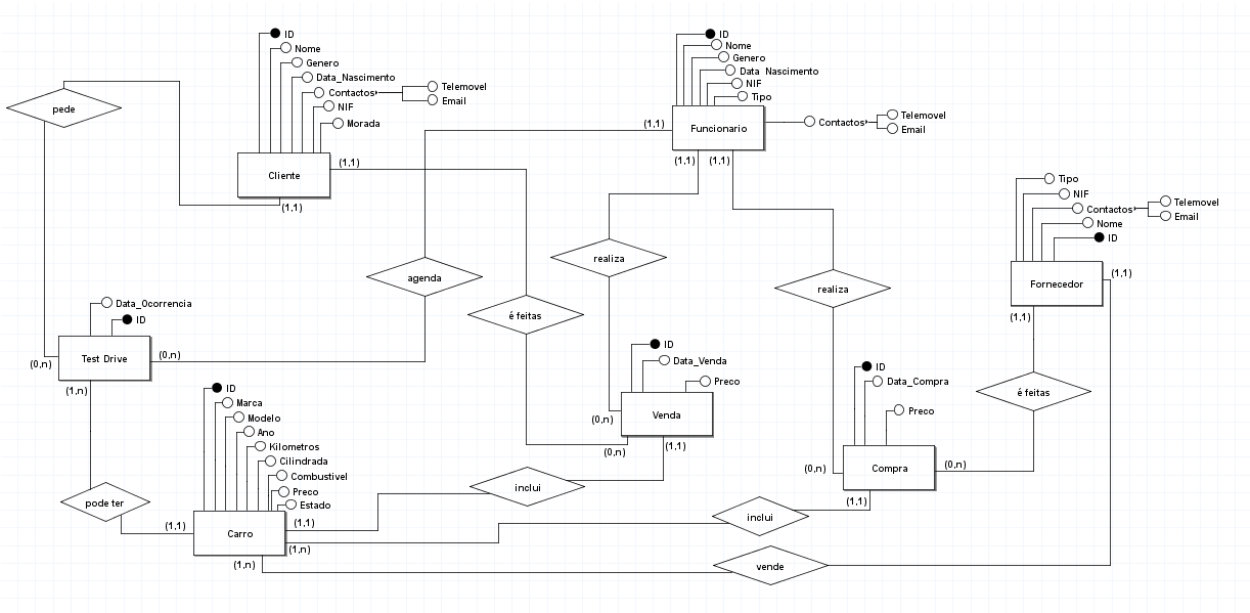

Figura 3.1: Modelo Conceptual

No modelo conceptual, cada entidade é representada por um retângulo, onde são listados os atributos que a caracterizam. Os relacionamentos entre as entidades são mostrados através de linhas que as conectam, com losangos no meio que esclarecem a natureza do relacionamento e a cardinalidade em cada extremidade da linha. Essa representação gráfica permite uma visualização clara e precisa das entidades envolvidas, seus atributos e as interações entre elas.

4 Modelação Lógica

4.1 Construção e validação do modelo de dados lógico

A construção do modelo de dados lógico foi realizada com base no modelo conceptual previamente elaborado. Nesta etapa, as entidades foram convertidas em tabelas, que contêm os atributos correspondentes a cada entidade. As relações entre as entidades são estabelecidas por meio das chaves estrangeiras, que referenciam as chaves primárias de outras tabelas.

Durante a validação do modelo de dados lógico, verificamos cuidadosamente a consistência e a integridade das tabelas, assegurando que todos os requisitos levantados estivessem adequadamente representados e que não houvesse redundâncias ou inconsistências no modelo.

Assim, após feita a conversão obtemos 13 tabelas, das quais 6 são derivadas de atributos compostos.

Atributos	Tipo	Chave
ID	INT	Chave Primária
Nome	VARCHAR(75)	-
Genero	VARCHAR(50)	-
Data_de_Nascimento	DATE	-
NIF	INT	-
Morada	VARCHAR(75)	-

Tabela 4.1: Tabela Cliente

Atributos	Tipo	Chave
Email	VARCHAR(75)	Chave Primária
Cliente_ID	INT	Chave Estrangeira

Tabela 4.2: Tabela do Email_Cliente

Atributos	Tipo	Chave
Telemovel	INT	Chave Primária
Cliente_ID	INT	Chave Estrangeira

Tabela 4.3: Tabela Telemovel_Cliente

Atributos	Tipo	Chave
ID	INT	Chave Primária
Nome	VARCHAR(75)	-
Genero	VARCHAR(50)	-
Data_de_Nascimento	DATE	-
NIF	INT	-
Tipo	ENUM('Comprador', 'Vendedor')	-

Tabela 4.4: Tabela Funcionário

Atributos	Tipo	Chave
Email	VARCHAR(75)	Chave Primária
Funcionario_ID	INT	Chave Estrangeira

Tabela 4.5: Tabela Email_Funcionario

Atributos	Tipo	Chave
Telemovel	INT	Chave Primária
Funcionario_ID	INT	Chave Estrangeira

Tabela 4.6: Tabela Telemovel_Funcionario

Atributos	Tipo	Chave
ID	INT	Chave Primária
Nome	VARCHAR(75)	-
Tipo	ENUM('Individual', 'Empresa')	-
NIF	INT	-

Tabela 4.7: Tabela Fornecedor

Atributos	Tipo	Chave
Email	VARCHAR(75)	Chave Primária
Fornecedor_ID	INT	Chave Estrangeira

Tabela 4.8: Tabela Email_Fornecedor

Atributos	Tipo	Chave
Telemovel	INT	Chave Primária
Fornecedor_ID	INT	Chave Estrangeira

Tabela 4.9: Tabela Telemovel_Fornecedor

Atributos	Tipo	Chave
ID	INT	Chave Primária
Marca	VARCHAR(50)	-
Modelo	VARCHAR(75)	-
Ano	INT	-
Kilometros	INT	-
Cilindrada	INT	-
Combustivel	ENUM('Gasolina', 'Gasoleo', 'Hibrido', 'Eletrico', 'Gas')	-
Preco	INT	-
Estado	VARCHAR(75)	-
Fornecedor_ID	INT	Chave Estrangeira

Tabela 4.10: Tabela Carro

Atributos	Tipo	Chave
ID	INT	Chave Primária
Data_Compra	DATE	-
Preco	INT	-
Funcionario_ID	INT	Chave Estrangeira
Fornecedor_ID	INT	Chave Estrangeira
Carro_ID	INT	Chave Estrangeira

Tabela 4.11: Tabela Compra

Atributos	Tipo	Chave
ID	INT	Chave Primária
Data_Venda	DATE	-
Preco	INT	-
Cliente_ID	INT	Chave Estrangeira
Funcionario_ID	INT	Chave Estrangeira
Carro_ID	INT	Chave Estrangeira

Tabela 4.12: Tabela Venda

Atributos	Tipo	Chave
ID	INT	Chave Primária
Data_Ocorrencia	DATE	-
Cliente_ID	INT	Chave Estrangeira
Funcionario_ID	INT	Chave Estrangeira
Carro_ID	INT	Chave Estrangeira

Tabela 4.13: Tabela Test_Drive

4.2 Normalização de Dados

Após desenvolvermos o modelo lógico, decidimos avaliar a normalização do mesmo. Assim, conseguimos reduzir a redundância de dados caso o modelo viole algum dos princípos das formas que decidimos usar.

Ao analisar o nosso modelo, verificamos que todos os atributos presentes são atómicos. Todos os atributos multivalorados presentes no modelo conceptual têm as suas próprias tabelas, respeitando assim a Primeira Forma Normal (1FN).

Analisando os valores presentes nas diferentes entidades do modelo, podemos verificar que são todos unicamente dependentes da chave primária da entidade. Como verificamos anteriormente que o modelo obedecia às regras da 1FN, conseguimos admitir que o modelo está na Segunda Forma Normal (2FN).

Assim, validamos o nosso modelo segundo a teoria da normalização (até a 2FN). Teoricamente, isto resultará num melhor desempenho e integridade para a base de dados.

4.3 Modelo lógico produzido

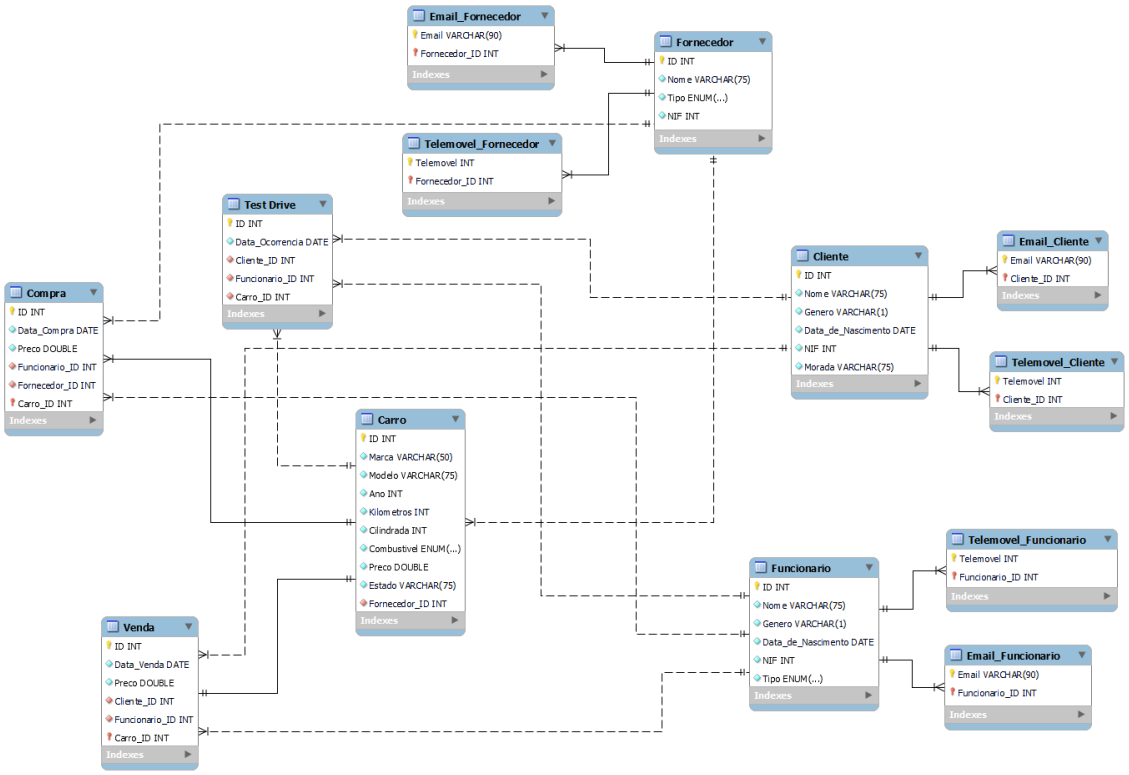

Figura 4.1: Modelo Lógico

4.4 Validação do modelo com interrogações do utilizador

Após concluído o esboço inicial das tabelas que compõem o nosso modelo lógico, onde identificamos as chaves primárias e estrangeiras de cada tabela, procedemos à representação gráfica dessas tabelas. Essa representação foi feita de forma imediata, mantendo a mesma multiplicidade entre os relacionamentos e entidades nas tabelas.

Em seguida, validamos o modelo lógico por meio da normalização, o que nos permitiu concluir que nosso esboço inicial seria válido. Além disso, a validação com através de questões feitas ao gerente desempenhou um papel essencial para confirmar que nosso modelo lógico atendia aos requisitos de exploração necessários para a implementação física de uma base de dados para o Stand.

Dessa forma, com base na validação e nas respostas fornecidas pelo gerente, consideramos o nosso modelo como sendo o modelo lógico final. Ele está apto a atender ás necessidades de exploração e pode ser utilizado como base para a implementação física da base de dados.

5 Implementação Física

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

Para traduzir o esquema lógico para o MySQL, utilizamos a ferramenta MySQL Workbench. Nessa ferramenta, convertemos o modelo lógico que concebemos anteriormente em código SQL usando a opção "Converter para Físico". Esse comando gerou o código necessário para criar a base de dados com base no esquema lógico, o que simplificou o processo de criação da base de dados.

No entanto, foi necessário fazer alguns ajustes no código gerado para garantir que as tabelas estivessem de acordo com os requisitos específicos do nosso projeto. Isso incluiu a adição de restrições, definição de chaves primárias e estrangeiras, e ajustes nos tipos de dados dos atributos.

```
CREATE TABLE IF NOT EXISTS `autovrumvrum`.`Fornecedor` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(75) NOT NULL,
  `Tipo` ENUM('Individual', 'Empresa') NOT NULL,
  `NIF` INT NOT NULL,
  CONSTRAINT chk_nif_forn CHECK(LENGTH(nif)=9),
  PRIMARY KEY (`ID`))
ENGINE = InnoDB;
```

Figura 5.1: Código SQL correspondente a criação da tabela Fornecedor

5.2 Tradução das interrogações do utilizador para SQL

Com base no modelo lógico, na criação e no povoamento da base de dados, realizamos consultas SQL para atender às perguntas e necessidades do gerente. As queries em SQL permitiram extrair informações relevantes da base de dados, aplicar filtros e combinar dados de diferentes tabelas para obter resultados específicos.

```sql
DROP PROCEDURE IF EXISTS carros_disponiveis_por_marca;

DELIMITER //
CREATE PROCEDURE carros_disponiveis_por_marca (IN marca VARCHAR(50))
    BEGIN
        SELECT carro.Marca, carro.Modelo, carro.Ano, carro.Kilometros, carro.Cilindrada, carro.Combustivel, carro.Preco, carro.Estado
        FROM Carro
        WHERE Carro.ID NOT IN (SELECT Venda.Carro_ID FROM Venda WHERE Venda.Carro_ID)
        AND Carro.Marca = marca;
    END //
DELIMITER //

CALL carros_disponiveis_por_marca( marca: 'bmw');
```

Figura 5.2: Código SQL para obter os carros disponíveis por marca

```sql
DROP PROCEDURE IF EXISTS historico_testDrives;

DELIMITER //
CREATE PROCEDURE historico_testDrives(IN cliente_id INT)
    BEGIN
        SELECT cliente.Nome, carro.Marca, carro.Modelo, carro.Ano, carro.Kilometros,
               carro.Cilindrada, carro.Combustivel, testDrive.Data_Ocorrencia, funcionario.Nome AS 'Funcionario'
        FROM TestDrive
        INNER JOIN Carro ON Carro.ID = TestDrive.Carro_ID
        INNER JOIN Cliente ON Cliente.ID = TestDrive.Cliente_ID
        INNER JOIN Funcionario ON Funcionario.ID = TestDrive.Funcionario_ID
        WHERE TestDrive.Cliente_ID = cliente_id;
    END //
DELIMITER //

CALL historico_testDrives( cliente_id: 1);
```

Figura 5.3: Código SQL para obter o historio de test drives de um cliente

```sql
DROP PROCEDURE IF EXISTS ObterMarcasMaisVendidas;

DELIMITER //
CREATE PROCEDURE ObterMarcasMaisVendidas()
BEGIN
    SELECT Carro.Marca, COUNT(*) AS Quantidade
    FROM Carro
    INNER JOIN Venda ON carro.ID = Venda.Carro_ID
    GROUP BY Carro.Marca
    ORDER BY Quantidade DESC, Carro.Marca;
END //
DELIMITER //

CALL ObterMarcasMaisVendidas();
```

Figura 5.4: Código SQL para obter as marcas mais vendidas

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

No desenvolvimento de um SGBD um fator importante são as *views*, estas são o resultado de uma ou mais operações relacionais sobre a base de dados. Uma *view* pode ser considerada como uma tabela que advém da consulta de outras tabelas, sendo por vezes designada de tabela virtual.

As vistas permitem simplificar e evitar escrever consultas frequentes, escondendo também certos detalhes que não têm interesse ou não dizem respeito a um dado utilizador. Este mecanismo providencia uma série de vantagens com a sua utilização e como tal decidimos tirar proveito das mesmas com os seguintes exemplos.

```sql
DROP VIEW IF EXISTS viewCarrosDisponiveis;

DELIMITER //
CREATE VIEW viewCarrosDisponiveis AS
    SELECT Carro.Marca, Carro.Modelo, Carro.Ano, Carro.Kilometros, Carro.Cilindrada, Carro.Combustivel, Carro.Preco, Carro.Estado
    FROM Carro
    WHERE Carro.ID NOT IN (SELECT Venda.Carro_ID FROM Venda WHERE Venda.Carro_ID);
DELIMITER //

SELECT * FROM viewCarrosDisponiveis;
```

Figura 5.5: View que mostra a informação relativa aos carros disponíveis

```
DROP VIEW IF EXISTS viewCarrosVendidos;

DELIMITER //
    CREATE VIEW viewCarrosVendidos AS
        SELECT Carro.Marca, Carro.Modelo, Carro.Ano, Carro.Kilometros, Carro.Cilindrada, Carro.Combustivel, Carro.Preco, Carro.Estado
        FROM Carro
        WHERE Carro.ID IN (SELECT Venda.Carro_ID FROM Venda WHERE Venda.Carro_ID);
DELIMITER //

SELECT * FROM viewCarrosVendidos;
```

Figura 5.6: View que mostra a informação relativa aos carros vendidos

```
DROP VIEW IF EXISTS viewClientes;

DELIMITER //
CREATE VIEW viewClientes AS
    SELECT Cliente.Nome, Cliente.Genero, Cliente.Data_de_Nascimento, Cliente.NIF, Cliente.Morada
    FROM Cliente;
DELIMITER //

SELECT * FROM viewClientes;
```

Figura 5.7: View que mostra a informação relativa aos clientes

5.4 Cálculo do espaço da bases de dados (inicial e taxa de crescimento anual)

Tendo em consideração o tamanho ocupado por datatypes como Int, String, Float, entre outros, conseguimos calcular o espaço ocupado em disco por cada atributo pertencente a cada tabela, obtendo então uma estimativa da ocupação total em disco da nossa base de dados.

Atributos	Tipo	Tamanho(bytes)
ID	INT	4
Nome	VARCHAR(75)	77
Genero	VARCHAR(50)	52
Data_de_Nascimento	DATE	3
NIF	INT	4
Morada	VARCHAR(75)	77
Total	-	**217**

Tabela 5.1: Cálculo dos bytes associados à entidade Cliente.

Atributos	Tipo	Tamanho(bytes)
Email	VARCHAR(75)	77
Cliente_ID	INT	4
Total	-	**81**

Tabela 5.2: Cálculo dos bytes associados à entidade Cliente Email_Cliente

Atributos	Tipo	Tamanho(bytes)
Telemovel	INT	4
Cliente_ID	INT	4
Total	-	**8**

Tabela 5.3: Cálculo dos bytes associados à entidade Telemovel_Cliente

Atributos	Tipo	Tamanho(bytes)
ID	INT	4
Nome	VARCHAR(75)	77
Genero	VARCHAR(50)	52
Data_de_Nascimento	DATE	3
NIF	INT	4
Tipo	ENUM('Comprador', 'Vendedor')	1.125
Total	-	**141.125**

Tabela 5.4: Cálculo dos bytes associados à entidade Funcionário

Atributos	Tipo	Tamanho(bytes)
Email	VARCHAR(75)	77
Funcionario_ID	INT	4
Total	-	**81**

Tabela 5.5: Cálculo dos bytes associados à entidade Email_Funcionario

Atributos	Tipo	Chave
Telemovel	INT	4
Funcionario_ID	INT	4
Total	-	**8**

Tabela 5.6: Cálculo dos bytes associados à entidade Telemovel_Funcionario

Atributos	Tipo	Tamanho(bytes)
ID	INT	4
Marca	VARCHAR(50)	52
Modelo	VARCHAR(75)	77
Ano	INT	4
Kilometros	INT	4
Cilindrada	INT	4
Combustivel	ENUM(...)	1.5
Preco	INT	4
Estado	VARCHAR(75)	77
Fornecedor_ID	INT	4
Total	-	**231.5**

Tabela 5.7: Cálculo dos bytes associados à entidade Carro

Atributos	Tipo	Tamanho(bytes)
ID	INT	4
Data_Compra	DATE	3
Preco	INT	4
Funcionario_ID	INT	4
Fornecedor_ID	INT	4
Carro_ID	INT	4
Total	-	**23**

Tabela 5.8: Cálculo dos bytes associados à entidade Compra

Atributos	Tipo	Tamanho(bytes)
ID	INT	4
Data_Venda	DATE	3
Preco	INT	4
Cliente_ID	INT	4
Funcionario_ID	INT	4
Carro_ID	INT	4
Total	-	**23**

Tabela 5.9: Cálculo dos bytes associados à entidade Venda

Atributos	Tipo	Tamanho(bytes)
ID	INT	4
Data_Ocorrencia	DATE	3
Cliente_ID	INT	4
Funcionario_ID	INT	4
Carro_ID	INT	4
Total	-	**19**

Tabela 5.10: Cálculo dos bytes associados à entidade Test_Drive

Atributos	Tipo	Tamanho(bytes)
ID	INT	4
Nome	VARCHAR(75)	77
Tipo	ENUM('Individual', 'Empresa')	1,125
NIF	INT	4
Total	-	**86,125**

Tabela 5.11: Cálculo dos bytes associados à entidade Fornecedor

Atributos	Tipo	Tamanho(bytes)
Email	VARCHAR(75)	77
Fornecedor_ID	INT	4
Total	-	**81**

Tabela 5.12: Cálculo dos bytes associados à entidade Email_Fornecedor

Atributos	Tipo	Tamanho(bytes)
Telemovel	INT	4
Fornecedor_ID	INT	4
Total	-	**8**

Tabela 5.13: Cálculo dos bytes associados à entidade Telemovel_Fornecedor

Após o somatório do total de todas as tabelas teríamos que, o tamanho total da nossa base de dados seria, sem povoamento, **1007,75 bytes**.

No entanto, para uma estimativa real, tendo em conta o povoamento do nosso modelo faria um total de 217 * 15 + 81 * 15 + 8 * 15 + 141.125 * 10 + 81 * 10 + 8 * 10 + 231.5 * 28 + 23 * 28 + 23 * 19 + 19 * 29 + 86.125 * 8 + 81 * 8 + 8 * 8 = **16406.25 bytes**.

5.5 Indexação do Sistema de Dados

A indexação desempenha um papel fundamental na otimização do desempenho e da eficiência das consultas num sistema de base de dados. Ao inserir numa tabela Tuplos, os índices são automaticamente criados para as colunas que possuem a restrição PRIMARY KEY. No nosso caso, os índices são criados para os campos de ID, os quais são auto-incrementados.

Os índices são essenciais por diversos motivos. Primeiramente, eles garantem a unicidade dos valores de uma coluna, evitando a inserção de registos duplicados e mantendo a consistência dos dados. Além disso, os índices aceleram a busca de registos específicos, permitindo que as consultas sejam executadas de forma mais rápida.

```
`ID` INT NOT NULL AUTO_INCREMENT,
PRIMARY KEY (`ID`))
```

Figura 5.8: Código SQL correspondente a indexação

5.6 Procedimentos Implementados

Criamos um ficheiro *inserts.sql*, onde criamos procedimentos com o objetivo de fornecer uma estrutura organizada e reutilizável para a inserção de dados nas tabelas. Cada procedimento corresponde a uma tabela específica e possui os parâmetros necessários para receber os valores dos registos a serem inseridos.

```
DELIMITER //
CREATE PROCEDURE insere_fornecedor (IN Nome VARCHAR(75), IN Tipo ENUM('Individual', 'Empresa'), IN NIF INT)
    BEGIN
        DECLARE exit handler for SQLEXCEPTION
            BEGIN
                SHOW ERRORS LIMIT 1;
                SHOW WARNINGS;
                ROLLBACK;
            END;

        DECLARE exit handler for SQLWARNING
            BEGIN
                SHOW ERRORS LIMIT 1;
                SHOW WARNINGS;
                ROLLBACK;
            END;

        START TRANSACTION;
            INSERT INTO Fornecedor
                (Nome, Tipo, NIF)
                VALUES
                (Nome, Tipo, NIF);
        COMMIT;
END //
```

Figura 5.9: Código SQL correspondente a inserção de um registo na tabela Fornecedor

```
DELIMITER //
CREATE PROCEDURE insere_cliente (IN Nome VARCHAR(75), IN Genero VARCHAR(1) , IN Data_de_Nascimento DATE, IN NIF INT, IN Morada VARCHAR(75))
    BEGIN
        DECLARE exit handler for SQLEXCEPTION
            BEGIN
                SHOW ERRORS LIMIT 1;
                SHOW WARNINGS;
                ROLLBACK;
            END;

        DECLARE exit handler for SQLWARNING
            BEGIN
                SHOW ERRORS LIMIT 1;
                SHOW WARNINGS;
                ROLLBACK;
            END;
        START TRANSACTION;
            INSERT INTO Cliente
                (Nome, Genero, Data_de_Nascimento, NIF, Morada)
                VALUES
                (Nome, Genero, Data_de_Nascimento, NIF, Morada);
        COMMIT;
END //
```

Figura 5.10: Código SQL correspondente a inserção de um registo na tabela Cliente

6 Implementação do Sistema de Recolha de Dados

6.1 Apresentação e modelo do sistema

O sistema de recolha de dados possibilita a obtenção de dados de diversas fontes e o armazenamento dos mesmos numa base de dados, permitindo que sejam posteriormente utilizados para análises, relatórios e outras atividades. Este sistema traz diversos benefícios para o stand. Ele permite a centralização dos dados, facilitando a integração e a análise das informações. Além disso, automatizar o processo de recolha e inserção de dados economiza tempo e reduz erros humanos, melhorando a eficiência operacional.

No nosso sistema, a primeira etapa é estabelecer conexão com a nossa base de dados que contém as tabelas que desejamos povoar. Em seguida, realizamos o parsing dos arquivos CSV, extraímos os dados e inserimos-los nas tabelas correspondentes através da execução de procedimentos SQL presentes no ficheiro *inserts.sql*.

6.2 Implementação e funcionamento do sistema de recolha

Nesta fase do projeto, utilizamos os procedimentos armazenados definidos no ficheiro *inserts.sql* para inserir os registos dos arquivos CSV nas tabelas da base de dados.

Para a implementação do sistema começamos por criar uma função chamada *execute_procedure_from_csv* que recebe: uma **connection** e um **cursor** que são os objetos de conexão e cursor respetivamente que permitem interagir com a base de dados; um **filename** que corresponde ao nome do arquivo CSV a ser lido; um **procedure_name** que corresponde ao nome da *store_procedure* que será executada; um **num_arguments** que corresponde ao número de argumentos esperados pela *store_procedure*.

O ficheiro CSV é aberto em modo de leitura e de seguida é criado um objeto *csv.reader* com base no ficheiro aberto, que permite ler as linhas do mesmo. A primeira linha do ficheiro é ignorada para evitar que o cabeçalho seja interpretado como argumentos. De seguida é criado um loop que itera cada linha do ficheiro CSV, dentro do loop, os valores dos argumentos para a procedure são extraídos da linha atual do CSV. A *store_procedure* é executada utilizando o cursor e os argumentos obtidos que envia a instrução para a base de dados executar a *store_procedure* com os argumentos fornecidos.

Após a execução da *store_procedure* para cada linha do arquivo CSV, as alterações são confirmadas na base de dados para garantir que as inserções realizadas foram efetivamente aplicadas.

Caso ocorra algum erro durante a execução da função, é exibida uma mensagem de erro, a informar que houve um problema na execução da *store_procedure*.

```python
# Funcao que executa uma stored procedure com os argumentos de um ficheiro CSV
def execute_procedure_from_csv(connection, cursor, filename, procedure_name, num_arguments):
    try:
        with open(filename, 'r') as file:
            csvreader = csv.reader(file)
            next(csvreader)  # da skip a primeira linha do ficheiro (cabecalho)

            for row in csvreader:
                # da skip a linhas vazias
                arguments = [row[i] for i in range(num_arguments)]

                # executa a stored procedure com os argumentos da linha atual
                cursor.callproc(procedure_name, arguments)

            # faz commit das alteracoes
            connection.commit()
            print("Stored procedure executed successfully.")
    except Exception as e:
        print("Error executing stored procedure:", e)
```

Figura 6.1: Código Python correspondente a função *execute_procedure_from_csv*

Após estar definida a função *execute_procedure_from_csv*, a conexão com a base de dados é estabelecida utilizando a função *sql.connect* e é criado um cursor a partir da conexão estabelecida, que permite executar comandos SQL na base de dados. Se a conexão for bem sucedida função *execute_procedure_from_csv* é chamada várias vezes com diferentes ficheiros CSV e as correspondentes *store_procedure*, cada vez que é executada a *execute_procedure_from_csv* a tabela correspondente ao nome do ficheiro CSV que foi lido é povoada.

```
try:
    connection = sql.connect(
        host="127.0.0.1",
        database="autovrumvrum",
        user="root",
        password="lapata2002"
    )

    if connection.is_connected():
        db_Info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_Info)
        cursor = connection.cursor()
        cursor.execute("select database();")
        record = cursor.fetchone()
        print("You're connected to database: ", record)

        execute_procedure_from_csv(connection, cursor, "./CSV/fornecedor.csv", "insere_fornecedor", 3)
        execute_procedure_from_csv(connection, cursor, "./CSV/funcionario.csv", "insere_funcionario", 5)
        execute_procedure_from_csv(connection, cursor, "./CSV/clientes.csv", "insere_cliente", 5)
        execute_procedure_from_csv(connection, cursor, "./CSV/carros.csv", "insere_carro", 9)
        execute_procedure_from_csv(connection, cursor, "./CSV/compra.csv", "insere_compra", 5)
        execute_procedure_from_csv(connection, cursor, "./CSV/venda.csv", "insere_venda", 5)
        execute_procedure_from_csv(connection, cursor, "./CSV/testdrive.csv", "insere_TestDrive", 4)
        execute_procedure_from_csv(connection, cursor, "./CSV/telemovelCliente.csv", "insere_tlm_Cliente", 2)
        execute_procedure_from_csv(connection, cursor, "./CSV/emailCliente.csv", "insere_email_Cliente", 2)
        execute_procedure_from_csv(connection, cursor, "./CSV/emailFornecedor.csv", "insere_email_Fornecedor", 2)
        execute_procedure_from_csv(connection, cursor, "./CSV/telemovelFornecedor.csv", "insere_tlm_Fornecedor", 2)
        execute_procedure_from_csv(connection, cursor, "./CSV/telemovelFuncionario.csv", "insere_tlm_Funcionario", 2)
        execute_procedure_from_csv(connection, cursor, "./CSV/emailFuncionario.csv", "insere_email_Funcionario", 2)
        execute_procedure_from_csv(connection, cursor, "./CSV/emailFuncionario.csv", "insere_email_Funcionario", 2)

except Error as e:
    print("Error while connecting to MySQL", e)
finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("MySQL connection is closed")
```

Figura 6.2: Código Python correspondente a recolha de dados

7 Implementação do Sistema de Painéis de Análise

7.1 Definição e caracterização da vista de dados para análise

Xavier pretende tomar decisões relacionadas à empresa com base em dados presentes na vista de dados. Assim, decidimos analisar todas as entidades do nosso projeto, fornecendo assim a maior quantidade de informação possível para que Xavier esteja sempre a par do estado da loja. Iremos analisar métricas maioritariamente económicas pois são a melhor forma de avaliar o funcionamento de um stand de carros. Utilizamos o Power BI Desktop da Microsoft para fazer esta análise, pois este proporciona uma ligação com a base de dados que desenvolvemos e permite fazer as várias análises que pretendiamos.

7.2 Povoamento das estruturas de dados para análise

Como referido anteriormente, o Power BI Desktop consegue fazer uma ligação com a base de dados, retirando os dados para análise diretamente dela. Assim, garantimos que os dados analisados são consistentes com os guardados na base de dados e automatiza o processo de povoamento.

É importante referir que a componente do MySQL que é suposto possibilitar a ligação com o Power BI (o conector NET) não parece estar a funcionar atualmente. Ao fazer uma pesquisa na Internet, observamos que várias pessoas já tinham encontrado o mesmo problema e a única solução que encontramos era utilizar uma versão anterior do conector.

Para evitar erros devido ao uso de software desatualizado, decidimos utilizar o ODBC (Open DataBase Connectivity). Assim, conseguimos fazer a ligação entre o Power BI e a base de dados sem utilizar o conector que nos estava a dar problemas.

Não utilizamos as tabelas relativas a emails/números pois não achamos estes atributos relevantes para a análise de dados pretendida. Também foi necessário identificar atributos monetários (p.e. o preço do carro) como tal, para o Power BI os apresentar como tal.

7.3 Apresentação e caracterização dos *dashboards* implementados

Para a análise de dos dados, decidimos implementar uma *dashboard* para cada tipo de "serviço"que o stand disponibiliza (compra, venda e *Test Drive*). Em cada *dashboard*, colocamos gráficos que mostrassem dados que fossem de interesse ao Xavier.

Tivemos de guardar cada *dashboard* num ficheiro à parte, pois o Power BI estava a ter alguns problemas com ambiguidade de ligações. Apesar de haver uma função "*USERELATIONSHIP*"em DAX (*Data Analysis eXpressions*) que possivelmente resolveria este problema, não conseguimos usá-la no nosso modelo

No *dashboard* relativo a Vendas, decidimos apresentar um gráfico com o total gasto por cada cliente no stand (como foi pedido pelo Xavier), um gráfico linear com o valor das vendas feitas por data, um gráfico circular com o valor das vendas feitas por marca do carro e um gráfico de barras com o valor das vendas organizado pelo tipo de combustível do carro vendido.

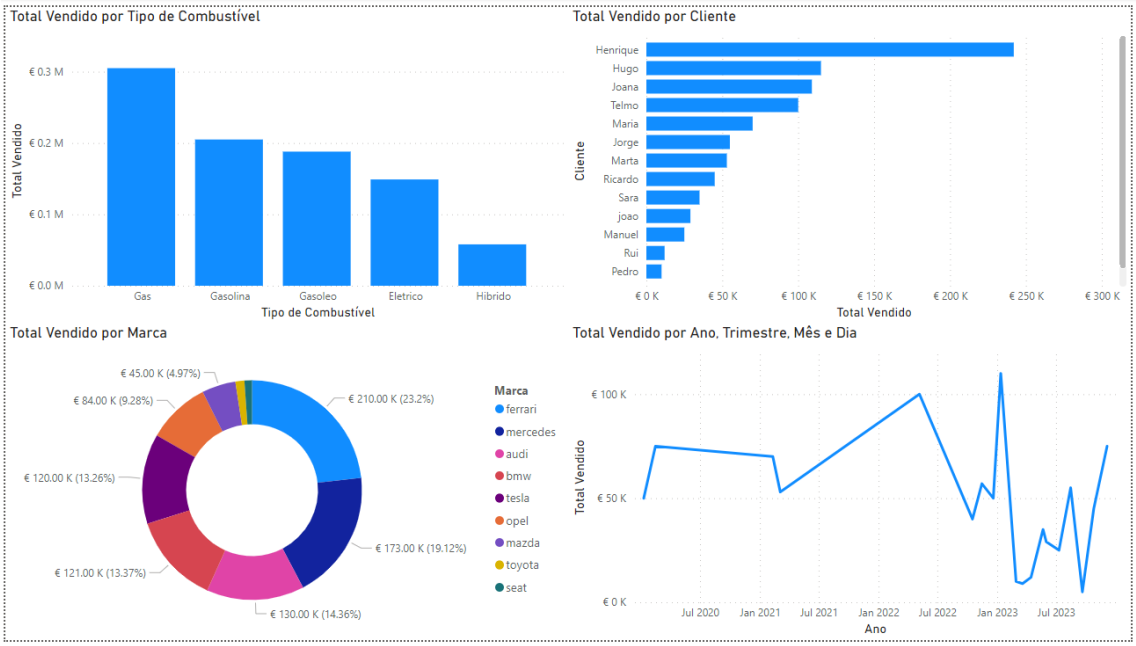

Figura 7.1: *Dashboard* relativo a Vendas

No *dashboard* relativo a Compras, decidimos apresentar um gráfico circular que representa a quantidade de Fornecedores de cada tipo, um gráfico de barras com o valor do total comprado a cada Fornecedor, um gráfico circular que representa a proporção de carros novos e usados entre os que foram adquiridos e um gráfico e outro gráfico circular com o valor do total comprado, dividido pelo marca do carro comprado.

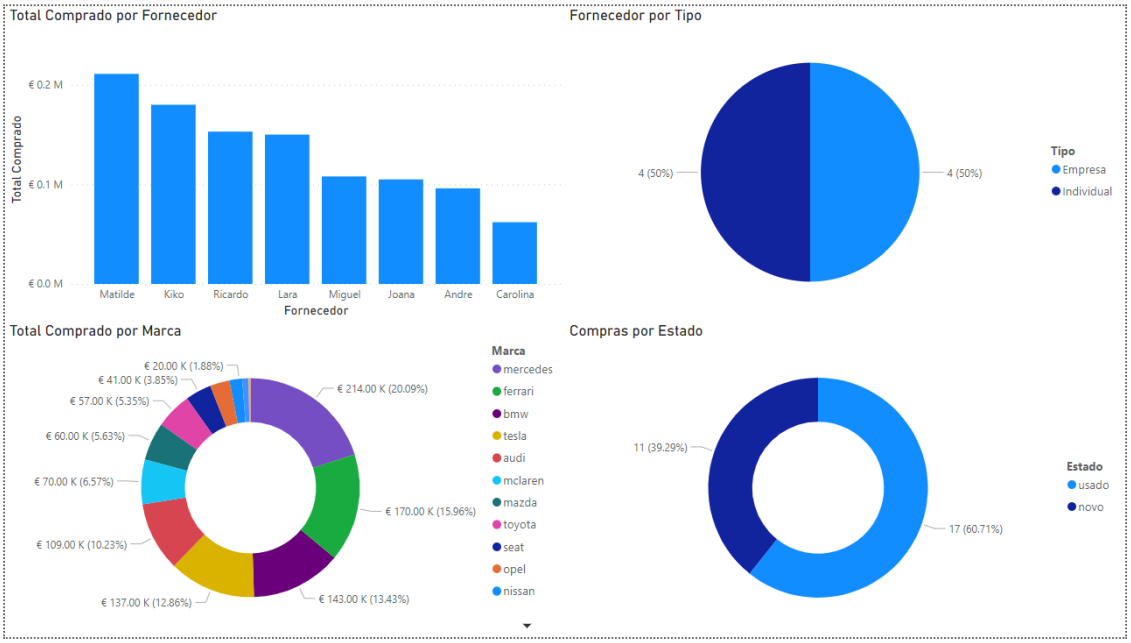

Figura 7.2: *Dashboard* relativo a Compras

No *dashboard* relativo a Test Drives, escolhemos apresentar a quantidade de Test Drives por cliente através de um gráfico de barras horizontal e a quantidade de Test Drives por marca, a quantidade de Test Drives por género do cliente e a quantidade de Test Drives por marca do carro através de gráficos circulares.

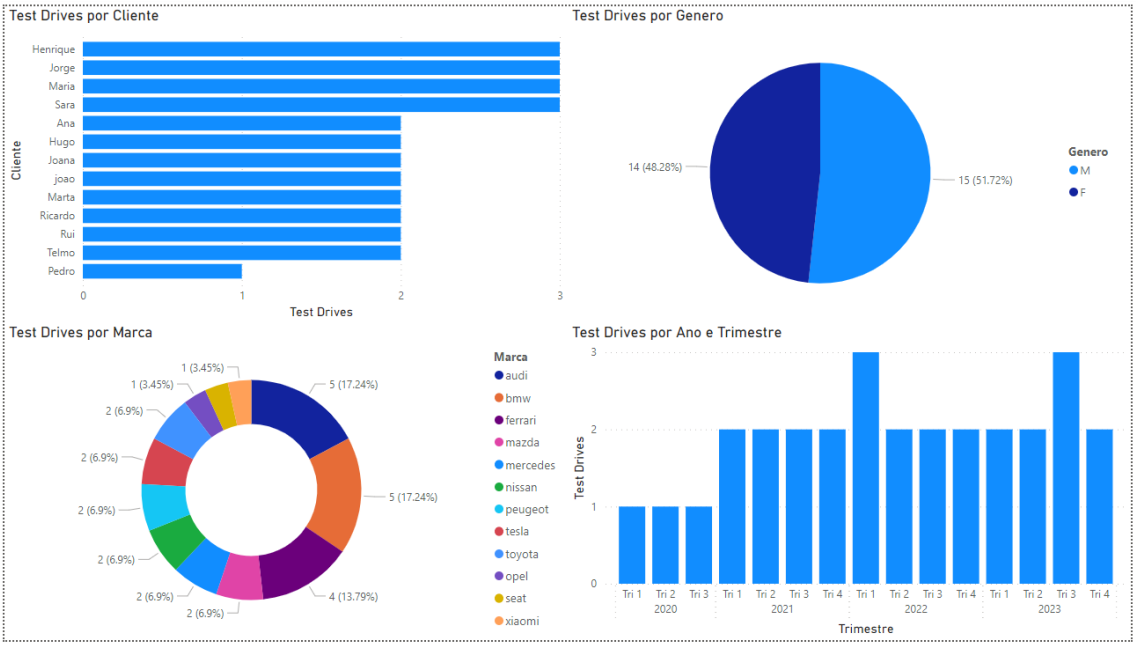

Figura 7.3: *Dashboard* relativo a Test Drives

8 Conclusões e Trabalho Futuro

8.1 Reflexão

Ao longo do desenvolvimento deste projeto, deparamo-nos com várias dificuldades devido à inexperiência dos membros do grupo relativamente às várias ferramentas que utilizamos e a certos componentes teóricos.

Por exemplo, depois de concluirmos a implementação física da base de dados, reparamos que, ao contrário do que pensávamos, a base de dados não estava na 3FN, pois o atributo *Marca* da tabela *Carro* é dependente do atributo *Modelo*. Mas criar uma nova tabela com a ligação entre *Modelo* e *Marca* para reduzir a redundância de dados implicaria a alteração de praticamente todo o trabalho que tínhamos desenvolvido e não achamos isto realista, pois só reparamos neste facto pouco antes do prazo de entrega do trabalho.

Apesar de haverem alguns problemas parecidos a este com o projeto, achamos que o resultado final cumpre todos os objetivos que fomos definindo ao longo da elaboração do trabalho. Os requisitos pedidos pelo cliente foram obedecidos e conseguimos implementar os vários sistemas que nos foram pedidos.

8.2 Trabalho Futuro

Como referido anteriormente, achamos que as principais razões pelas dificuldades que encontramos foram a falta de prática com as ferramentas utilizadas e a má gestão de tempo. Com a experiência que adquirimos enquanto elaboramos este trabalho e tendo mais atenção na gestão do tempo, achamos que conseguiríamos fazer uma base de dados mais eficiente e "*feature-rich*"do que a que foi elaborada neste trabalho.

Lista de Siglas e Acrónimos

BD Bases de Dados

SGDB Sistema de Gestão de Bases de Dados

DAX Data Analysis Expressions

1FN Primeira Forma Normal

2FN Segunda Forma Normal

3FN Terceira Forma Normal