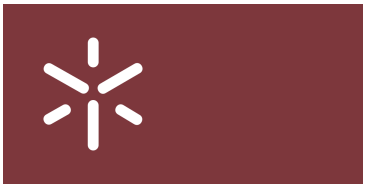

Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática
Mestrado Integrado em Engenharia Informática

Unidade Curricular de Base de Dados

Ano Letivo de 2024/2025

Base de Dados para uma cadeia de Stands Automóveis

João Costa a100707
Sandro Coelho a105672
Tiago Carneiro a93207

Grupo 33
Junho, 2025

BD

Data de receção	
Responsável	
Avaliação	
Observações	

Base de Dados para uma cadeia de Stands Automóveis

João Costa (a100707)
Sandro Coelho (a105672)
Tiago Carneiro (a93207)

Junho, 2024

Resumo

A empresa P.M. Veículos, fundada em 27 de outubro de 1999 no Porto por Pedro Monteiro, iniciou o seu negócio de aluguer de automóveis. Devido ao seu sucesso, decidiu abrir outra stand em Lisboa e recentemente, outra stand em Portimão com uma orientação mais focada em carros desportivos e de competição. Devido a registo de informações ser realizada em papel em cada stand, problemas de atraso de entrega para o departamento contabilístico são inevitáveis, além de preocupações com a segurança desses registos. Dessa forma, Pedro Monteiro decidiu implementar uma organização eficiente dos dados de todos os stands através de uma Base de Dados para todas as suas informações.

Na segunda fase do projeto, foi concretizada a implementação física da base de dados, seguindo o modelo lógico previamente desenvolvido. Esta etapa envolveu a criação do esquema da base de dados em MySQL, o desenvolvimento de um sistema de povoamento com dados iniciais e a introdução de mecanismos para facilitar a utilização e análise dos dados, incluindo vistas, índices, procedimentos, funções e gatilhos. Para além disso, foi também concebido e implementado um sistema de migração de dados, capaz de integrar informação proveniente de diferentes fontes – SQL, CSV e JSON – através de um script Python, garantindo uma integração estruturada e escalável com a base de dados central.

Áreas de Aplicação: Base de Dados

Palavras-Chave: Base de Dados, Modelação ER, MySQL, Gestão de Stands Automóveis, Álgebra Relacional, Requisitos

Índice

Lista de Figuras

Lista de Tabelas

1. Definição do Sistema

1.1. Contexto

A empresa *P.M. Veículos* foi fundada a 27 de outubro de 1999, no Porto por Pedro Monteiro. O fundador começou por abrir um stand de aluguer de carros no Porto, próximo do aeroporto. Como um apaixonado por carros desde muito novo, uma paixão que herdou do seu pai, achou que o que gostaria de fazer para o resto da vida envolveria de alguma maneira veículos motorizados.

Ao longo dos anos visto ao sucesso que o seu stand obteve este achou por bem expandir o seu negócio e abrir um novo stand em Lisboa, mais uma vez localizada perto do aeroporto.

Na sua mais recente expansão este decidiu optar por um caminho diferente e escolheu abrir um stand em Portimão com a ideia de que esta seria destinado ao aluguer de carros desportivos e de competição que poderiam ser conduzidos tanto na estrada como na autódromo do Algarve.

Neste momento, a informação da *P.M. Veículos* apresenta-se guardada num livro individual a cada stand, o que resulta em atrasos na entrega de documentos para o departamento contabilístico do Sr. Pedro Monteiro. Cada livro apresenta-se como sendo responsabilidade do gerente de cada stand, sendo eles quem introduzem informação para os mesmos. Para além disso, a segurança destes livros é de extrema importância, estando esta de momento nas mãos dos gerentes, algo que não agrada ao Sr. Pedro Monteiro.

Atualmente a empresa possui então três stands e um total de 14 funcionários. Cada um dos stands possui um gerente bem como 3 funcionários responsáveis pelo atendimento ao cliente. Para além disso no stand de Portimão existem ainda 2 funcionários extras responsáveis pela gestão e manutenção dos veículos.

Com a inauguração do mais recente stand Pedro Monteiro acredita ser necessário fazer uma organização mais centralizada e eficiente dos dados portanto achou boa ideia a implementação de uma Base de Dados.

1.2. Motivação e Objetivos do Trabalho

A necessidade deste trabalho provêm dos desafios projetados pela *P.M. Veículos* no que diz respeito à centralização e gestão eficiente dos dados após a mais recente inauguração de um novo stand em Portimão.

Ata da reunião de objetivos a alcançar

10 de março de 2025

Participantes : - Autores do projeto - Pedro Monteiro - Gerentes dos Stands - Contabilidade

Nesta primeira reunião, os gerentes dos stands chamaram a atenção para as dificuldades associadas à reintrodução de dados de clientes habituais, um processo que representa um desperdício significativo de tempo. Esta questão tem um impacto direto na eficiência da equipa e acaba por contribuir para atrasos na entrega de documentos essenciais para a contabilidade.

Perante esta situação, a contabilidade sublinhou a importância de melhorar o fluxo de entrega de documentação, uma vez que é fundamental cumprir os prazos legais. O sistema atualmente em uso, sobretudo após a inauguração do novo stand, revela-se incapaz de acompanhar as exigências operacionais, criando entraves à produtividade e ao cumprimento das obrigações fiscais.

Além disso, o Sr. Pedro Monteiro manifestou preocupação relativamente à segurança dos dados. O recurso a registos físicos continua a representar um risco, dada a vulnerabilidade a perdas e furtos, tornando imperativa a implementação de um sistema seguro e protegido contra acessos não autorizados.

Fim da Reunião

Como descrito anteriormente, a empresa enfrenta atualmente diversos desafios relacionados com a descentralização da informação e os métodos antiquados de registo utilizados nos seus três stands. A utilização de livros físicos individuais para cada stand não só dificulta a comunicação entre equipas e departamentos, como também compromete a segurança e a integridade dos dados, especialmente no que toca à entrega atempada de documentação contabilística ao Sr. Pedro Monteiro.

A reintrodução constante de dados de clientes habituais representa uma tarefa redundante, que afeta negativamente a produtividade dos funcionários e contribui para atrasos nos processos administrativos. Esta situação agrava-se com a recente expansão para Portimão, onde a operação mais especializada exige um controlo mais rigoroso dos veículos e da informação associada.

A contabilidade, por sua vez, sublinhou a dificuldade em manter um fluxo eficaz de documentos entre os stands e o escritório central, salientando a importância de garantir o cumprimento dos prazos legais e fiscais. Para além disso, a dependência de documentos físicos levanta sérias preocupações em relação à segurança dos dados, um ponto que o Sr. Pedro Monteiro frisou como sendo prioritário para o futuro da empresa.

Diante destes desafios, o principal objetivo deste projeto é modernizar e centralizar a gestão de informação, garantindo simultaneamente a segurança, integridade e acessibilidade dos dados.

Na sequência da reunião onde foram discutidos os objetivos a alcançar, seguem-se os mesmos:

- **Centralização da Informação**: Desenvolver um sistema de base de dados que permita reunir, num único local digital, todos os registos relevantes de clientes, contratos de aluguer, veículos disponíveis e respetiva manutenção, eliminando a necessidade de livros físicos por stand.
- **Reutilização de Dados**: Implementar funcionalidades que permitam recuperar rapidamente informações de clientes recorrentes, poupando tempo aos funcionários e reduzindo a margem de erro na introdução de dados.
- **Melhoria da Comunicação Interna**: Facilitar a troca de informação entre stands e o departamento de contabilidade, assegurando um fluxo de dados contínuo, organizado e acessível de forma segura.
- **Segurança e Integridade dos Dados**: Garantir que toda a informação está protegida contra perdas, furtos ou acessos indevidos através de mecanismos de autenticação e backups regulares.
- **Eficiência Operacional**: Aumentar a eficiência global da empresa, permitindo um acompanhamento mais rigoroso das operações de aluguer, da gestão de frota e da entrega de documentação, reforçando a qualidade do serviço prestado.

Estes objetivos foram definidos com base nos desafios levantados durante a reunião e servirão como diretrizes principais para o desenvolvimento do novo sistema de gestão de dados da (nome da empresa), com vista à modernização e profissionalização da sua estrutura organizacional.

1.3. Análise da Viabilidade do Projeto

Para verificar se realmente devemos proceder à criação do sistema de base de dados foi necessário analisar a sua viabilidade. Começaremos por observar os benefícios da criação da mesma:

- **Acesso centralizado**: Qualquer stand poderá consultar informações de aluguer em tempo real
- **Segurança da informação**: Controlo de acesso à informação e possível backup dos dados.
- **Melhor experiência do cliente**: Processo de aluguer será mais rápido e eficiente.

Após verificarmos os benefícios que a criação de uma base de dados irá trazer para esta empresa é necessário perceber se estes benefícios irão justificar os custos associados à criação da base de dados.

Uma vez que a base de dados irá ser criada por um grupo de 4 alunos ao longo de 4 meses o seu custo total irá ser 0€. No entanto, apesar de não existir um custo de desenvolvimento, irá sempre existir um custo associado à necessidade da existência de hardware bem como os custos para manter o mesmo. Assumindo que é utilizado um servidor local para armazenar e processar dados então iremos ter um custo inicial de aproximadamente 1500€ e para além disso podemos prever um custo anual de 900€ por ano relativo ao consumo de eletricidade.

Analisando os benefícios esperados é expectável que para além de uma melhor eficiência do tratamento dos dados da empresa também se torna uma experiência mais agradável para os seus clientes o que certamente irá levar a um crescimento de popularidade da empresa e consequentemente a quantidade de clientes trazendo ainda mais lucro para a empresa.

Ou seja, para além de benefícios relativos a gestão de dados e melhor experiência tanto de funcionários como de clientes também podemos esperar um aumento de pelo menos 10% no lucro da empresa o que implica um aumento de aproximadamente 100000€ por ano o que supera os custos associados à criação e manutenção da base da dados.

Podemos então concluir que para além de fatores não quantificáveis como a maior facilidade para a gestão de dados da empresa também irão existir benefícios monetários o que justifica a viabilidade da criação da base de dados.

1.4. Recursos e Equipa de trabalho

O desenvolvimento do sistema de gestão de base de dados envolveu a intervenção de várias pessoas.

A decisão do proprietário da empresa, Pedro Monteiro, de implementar uma base de dados foi essencial para o sucesso do projeto. A sua disponibilidade para comunicar com a equipa de desenvolvimento e expressar claramente os seus requisitos, juntamente com a participação ativa dos gerentes de cada stand, foram fatores determinantes.

Para além dos responsáveis pela orientação do projeto também foi fulcral o empenho demonstrado pelos 4 alunos encarregues de idealizar e desenvolver a base de dados.

A colaboração eficiente entre todas as partes envolvidas resultou na criação de uma base de dados funcional e bem estruturada, garantindo o sucesso do projeto.

1.5. Plano de execução do Projeto

O primeiro passo de todo este projeto foi a idealização de como iria ser realizado.

Para tal foi construido um diagrama de GANTT(Figura 1) com a intenção de definir espaços de tempo para a realização de cada sub tarefa.

Fase:	Subtópico:	Início:	Fim:	Duração
Definição do Sistema	Contextualização e fundamentação	2/25/2025	3/1/2025	4
	Objetivos e Viabilidade	2/27/2025	3/1/2025	2
	Recursos e equipa	2/28/2025	3/1/2025	1
Definição de Requisitos	Definição do Método	3/3/2025	3/11/2025	8
	Levantamento de Requisitos	3/4/2025	3/11/2025	7
	Análise de Requisitos	3/7/2025	3/11/2025	4
	Organização de Requisitos	3/8/2025	3/11/2025	3
	Validação	3/10/2025	3/11/2025	1
Modelação Conceptual	Definição da abordagem de modelação	3/17/2025	3/24/2025	7
	Identificação e caracterização das entidades	3/18/2025	3/24/2025	6
	Identificação e caracterização dos relacionamentos	3/19/2025	3/24/2025	5
	Identificação e caracterização dos atributos	3/20/2025	3/24/2025	4
	Construção do diagrama ER	3/21/2025	3/24/2025	3
	Validação do modelo conceptual	3/23/2025	3/24/2025	1
Modelação Lógica	Validação do modelo através da normalização	3/24/2025	3/31/2025	7
	Validação do modelo com transações do utilizador	3/26/2025	3/31/2025	5
	Construção do modelo	3/28/2025	3/31/2025	3
	Validação do modelo lógico	3/31/2025	4/1/2025	1

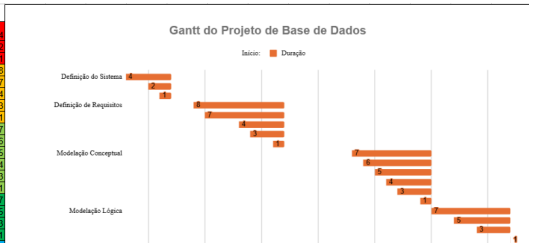

Figura 1: Diagrama de GANTT inicial.

O plano acabou por não ser executado exatamente como definido inicialmente. Provou-se um desafio principalmente nas primeiras semanas organizar um período de tempo ideal para todos os envolvidos reunirem e discutir as tarefas a realizar.

Por esse motivo só foi possível ao grupo reunir pela primeira vez no dia 10 de março, segunda-feira, o que estava inicialmente planeado como sendo uma semana onde os membros do grupo não iriam trabalhar para este projeto devido à existência de outros trabalhos a realizar. A partir desta data o grupo reuniu-se online todas as segundas de forma a discutir o estado atual do projeto, trabalhar em conjunto e discutir tarefas a serem feitas ao longo da semana.

Na segunda fase do projeto não foi possível ao grupo reunir tantas vezes como seria desejado uma vez que todos tinham de se focar em outros projetos e na fase de avaliação que se aproximava. Mesmo assim todos dispensaram algum do seu tempo para desenvolver o projeto.

2. Levantamento e Análise de Requisitos

2.1. Método de Levantamento dos Requisitos

Após a primeira reunião foi preciso tomar uma decisão relativamente a como iriam ser levantados os requisitos necessários para o desenvolvimento da base de dados.

Inicialmente foi realizada uma reunião entre os criadores do sistema e o dono da empresa bem como os gerentes de cada stand para mais facilmente identificar o que estes pretendiam do sistema.

Ata da reunião de levantamento de requisitos

15 de março de 2025

Participantes : - Autores do projeto - Pedro Monteiro - Gerentes dos Stands

Nesta reunião era pretendido levantar os requisitos necessários para o desenvolvimento da base de dados que permite uma gestão eficiente de clientes, veículos e alugueres da empresa.

Durante esta reunião os responsáveis por criar a base de dados tentaram colocar questões simples aos funcionários e dono da empresa de forma a reunirem informação sucinta e essencial para o desenvolvimento do sistema. Algumas das informações retiradas desta reunião foi quais seriam as diferentes entidades que iriam constituir o sistema bem como quais os dados associados a estas. Outra ponto fulcral a retirar desta reunião foi o modo como a empresa opera, tanto de os dados que armazenam e da forma como os tratam.

Fim da Reunião

Para além disso também foi disponibilizado aos criadores do sistema alguns registos físicos que eram utilizados pela empresa para melhor perceber quais informações a empresa precisa de manter sobre os seus clientes.

Estes diferentes métodos permitiu identificar as necessidades e expectativas dos utilizadores e levou por fim à construção do documento final(Anexo1).

2.2. Organização dos Requisitos Levantados

Através da reunião realizada e da análise de documentos providos pela empresa foi possível organizar os requisitos levantados em três tipos: Descrição, Manipulação e Controlo.

Os Requisitos de Descrição representam todos os requisitos que caracterizam entidades, definem relações entre estas, ou seja, aqueles que permitem estruturar a base de dados.

Os Requisitos de Manipulação descrevem procedimentos que o sistema irá fazer, como por exemplo uma listagem ou um cálculo.

Os Requisitos de Controlo são aqueles que descrevem restrições sobre quem pode fazer certas operações sobre a base de dados.

2.2.1. Requisitos de Descrição

Os requisitos de descrição levantados foram os seguintes:

RD01 -> Um cliente tem um identificador único, nome, data de nascimento, nacionalidade, morada que é constituída por código postal, rua, localidade, NIF, email e telefone.

RD02 -> Cada stand tem um identificador único, data de inauguração, telefone, morada que é constituída por código postal, rua e localidade.

RD03 -> Cada stand tem uma lista de veículos associada, assim como uma lista de funcionários.

RD04 -> Cada veículo tem um id, marca, modelo, matrícula, tipo (normal ou de competição), estado atual de aluguer, preço do mesmo e histórico.

RD05 -> Um funcionário tem um id, nome, um cargo e horas trabalhadas.

RD06 -> Um aluguer tem um id, uma data de início e fim, preço total e estado do veículo após a sua devolução.

RD07 -> Um cliente pode fazer vários alugueres.

RD08 -> Um veículo pertence apenas a um stand.

RD09 -> Apenas um funcionário é responsável por um aluguer.

RD10 -> Um aluguer só pode ter um veículo associado.

RD11 -> Um aluguer só pode ter um cliente.

RD12 -> Um veículo pode estar associado a mais do que um aluguer.

RD13 -> Um funcionário só pode trabalhar num stand.

Os Requisitos de Descrição apresentados fornecem uma visão detalhada das entidades, atributos e relações essenciais ao funcionamento da empresa.

2.2.2. Requisitos de Manipulação

Os requisitos de manipulação levantados foram os seguintes:

RM01 -> Deve ser possível listar veículos disponíveis, indisponíveis e total.

RM02 -> Deve ser possível listar funcionários.

RM03 -> Deve ser possível listar stands.

RM04 -> Deve ser possível listar clientes.

RM05 -> Deve ser possível calcular preço do aluguer.

RM06 -> Deve ser possível visualizar alugueres de um determinado cliente.

RM07 -> Deve ser possível calcular a média de alugueres por dia de um stand.

RM08 -> Deve ser possível listar os alugueres dos quais um funcionário foi responsável.

RM09 -> Deve ser possível consultar as horas trabalhadas por um funcionário.

RM10 -> Deve ser possível listar veículos por tipo.

RM11 -> Deve ser possível listar os clientes por nacionalidade.

RM12 -> Deve ser possível listar os clientes por localidade.

Os Requisitos de Manipulação apresentados garantem que o sistema oferecerá mecanismos eficazes para consulta, análise e operação sobre os dados armazenados.

2.2.3. Requisitos de Controlo

Os requisitos de controlo levantados foram os seguintes:

RC01 -> O Sr. Pedro Monteiro deve ser capaz de aceder a todos os dados.

RC02 -> Os Gerentes de cada stand devem ser capazes de ver e editar os documentos de cada um dos seus stands.

RC03 -> Os funcionários apenas podem aceder a alguns dos seus próprios dados, bem como informação dos alugueres em que estão envolvidos.

RC04 -> O preço associado a um carro apenas pode ser alterado pelo dono.

Os Requisitos de Controlo são fundamentais para garantir a segurança, confidencialidade e integridade da informação armazenada na base de dados. Estes requisitos definem claramente os níveis de acesso e as permissões de cada tipo de utilizador, assegurando que apenas as pessoas autorizadas possam visualizar ou modificar determinados dados sensíveis.

2.3. Revisão e Validação dos Requisitos

Após a criação do ficheiro relativo ao levantamento de requisitos e equipa responsável por criar o projeto reunião para rever o que foi feito. Após a correção de alguns erros foi então marcada uma reunião com o dono da empresa para se fazer uma última validação.

Nesta ultima reunião os requisitos foram aprovados. Então com esta etapa terminada podemos passar para a modelação conceptual.

3. Modelação Conceptual

3.1. Apresentação da Abordagem de Modelação Realizada

Após o levantamento dos requisitos foi necessário esquematizar e organizar a informação levantada. Procedeu-se então à construção de um modelo conceptual.

Para a criação do mesmo o grupo reuniu-se para escolher qual o software a ser utilizado para o desenvolvimento do diagrama e chegou-se à conclusão que a ferramenta a utilizar seria o **BrModelo**, uma vez que é uma ferramenta open source, utiliza uma adaptação da notação de Chen utilizada nas aulas e para além disso permite a conversão do modelo conceptual para um modelo lógico.

3.2. Identificação e Caracterização das Entidades

O primeiro passo para o desenvolvimento do modelo conceptual foi a identificação das possíveis entidades. Analisando os requisitos levantados foi claro perceber que as entidades necessárias ao sistema seriam as enumeradas de seguida:

1.**Stand**
O stand representa cada um dos locais onde a empresa se localiza. Possui um identificador único, data de inauguração, telefone e morada constituída por código postal, rua e localidade.

2.**Veículo**
O veículo identifica cada um dos carros que a empresa possui para alugar. Um veículo tem associado a si um identificador único, marca, modelo, matrícula, o seu tipo, estado atual de aluguer, preço e histórico.

3.**Funcionário**
A entidade funcionário identifica todos os colaboradores da empresa, tanto gerentes como vendedores. Cada funcionário tem um identificador único, nome, cargo, horas trabalhadas.

4.**Cliente**
O cliente refere-se à entidade mais importante para a empresa, as pessoas que irão usufruir da mesma. Cada cliente tem um identificador único, nome, data de nascimento, nacionalidade, morada, NIF, email e telefone.

5.**Aluguer**
A entidade aluguer identifica o serviço prestado pela empresa. Foi decidido que aluguer deveria de ser uma entidade e cada um deles ter um identificador único uma vez que a informação única de cada aluguer irá ser necessária no futuro. Cada aluguer tem um id, data de inicio e fim, preço total e estado.

As entidades identificados podem ser observados na seguinte tabela:

Entidade	Descrição	Requisito
“Cliente”	“Pessoa que solicita os serviços da Stand. Entrega registos de dados importantes para a empresa.”	“RD01”
“Stand”	“Local de armazenamento e aluguer de veículos. Possui registos de dados de localização e de disponibilidade.”	“RD02”
“Veículo”	“Automóvel disponível para aluguer. Possui registos sobre o mesmo.”	“RD04”
“Funcionário”	“Colaborador na empresa em questão. Possui registos sobre o mesmo e dados sobre o próprio salário.”	“RD05”
“Aluguer”	“Contrato temporário de posse de um veículo através de pagamentos. Possui dados sobre o veículo envolvido, o cliente e informações sobre o contrato.”	“RD06”

Tabela 1: Tabela de entidades

3.3. Identificação e Caracterização dos Relacionamentos

Como para as identidades, os relacionamentos também foram identificados através da análise dos requisitos levantados. Após a identificação das identidades é necessário perceber quais as associações entre elas. Um ponto importante a verificar nestes relacionamentos é qual a sua cardinalidade e participação.

- Stand **tem** Veículo. Este relacionamento define uma ligação entre os stands e a frota de veículos que este possui. Esta associação tem cardinalidade **1:N** e participação **T:T**, isto é, um stand pode possuir vários veículos e um veículo só pode pertencer a um stand, todos os stands têm pelo menos um veículo e um veículo tem de pertencer a um stand.
- Stand **tem** Funcionário. Associação entre um stand e os seus colaboradores. Este relacionamento tem cardinalidade **1:N** e participação **T:T**, ou seja, um stand tem vários funcionários enquanto que um funcionário só pode trabalhar num stand, para além disso todos os funcionários tem de estar associados a um stand assim como todos os stands têm de ter um funcionário.
- Funcionário **é responsável** Aluguer. Este relacionamento associa um funcionário ao aluguer que irá processar. Esta associação tem cardinalidade **1:N** e participação **P:T**, portanto um funcionário pode ser responsável por vários alugueres enquanto que um aluguer tem apenas um funcionário, todos os alugueres têm de ter um funcionário responsável, no entanto um funcionário pode ainda não ter sido responsável por nenhum aluguer.
- Cliente **faz** Aluguer. Este relacionamento define uma ligação entre o cliente e o serviço prestado pela empresa. Este relacionamento tem cardinalidade **1:N** e participação **P:T**, isto é, um cliente pode fazer vários alugueres e cada aluguer pertence apenas a um único cliente, um cliente pode ainda não ter feito nenhum aluguer, mas todos os alugueres estão associados a um cliente.
- Aluguer **envolve** Veículo. Este relacionamento liga cada aluguer ao veículo envolvido no mesmo. Esta associação tem cardinalidade **1:1** e participação **T:P**, ou seja, cada aluguer envolve um único veículo e cada veículo só pode estar associado a um aluguer ativo de cada vez, todos os alugueres envolvem um veículo mas nem todos os veículos estão associados a um aluguer.

Os relacionamentos identificados podem ser observados na seguinte tabela:

Entidade	Relacionamento	Cardinalidade	Participação	Entidade	Requisitos
Stand	tem	1:N	T:T	Veículo	RD03, RD08
Stand	tem	1:N	T:T	Funcionário	RD03, RD13
Funcionário	é responsável	1:N	P:T	Aluguer	RD09
Cliente	faz	1:N	P:T	Aluguer	RD07, RD11
Aluguer	envolve	1:1	T:P	Veículo	RD10, RD12

Tabela 2: Tabela de relacionamentos

3.4. Identificação e Caracterização dos Atributos das Entidades

Mais uma vez, a identificação e caracterização dos atributos das entidades e relacionamentos é feita através da análise dos requisitos levantados, uma vez que não foram levantados atributos relativos a relacionamentos iremos apenas abordar os atributos das entidades.

Obter os atributos é bastante simples e requer apenas a análise dos requisitos levantados, tomemos como exemplo o RD01:

RD01 -> Um cliente tem um identificador único, nome, data de nascimento, nacionalidade, morada que é constituída por código postal, rua, localidade, NIF, email e telefone.

Este requisito de descrição descreve claramente que um cliente tem um conjunto de características, um "identificador único", "nome", "morada", "NIF", "email" e "telefone". De notar que a morada é constituída por código postal, rua e localidade, logo podemos concluir que a morada é um atributo composto enquanto que o identificador único(id) é um atributo chave e os restantes são atributos simples. A única decisão a tomar é qual o domínio de cada atributo, ou seja o que define o seu tipo de dados. Neste caso do cliente iremos ter INT, VARCHAR e DATE.

Procedeu-se então à criação de tabelas de atributos para cada uma das entidades aplicando a lógica usada no exemplo anterior.

Os atributos associados ao cliente podem ser observados na seguinte tabela:

Atributo	Descrição	Domínio	Nulo	Exemplo	Requisito
Id	Identificador sequencial do Cliente	INT	N	26	RD01
Nome	Nome completo do cliente	VARCHAR(64)	N	Max Emillian Verstappen	RD01
Data_nasc	Data de nascimento do cliente	DATA	N	30-9-1997	RD01
Nacionalidade	Nacionalidade do cliente	VARCHAR(50)	N	Países Baixos	RD01
NIF	Número identificação fiscal do cliente	INT(9)	N	528317964	RD01
Email	Endereço de correio eletrónico do cliente	VARCHAR(75)	N	maxverstappen33 @gmail.com	RD01
Telefone	Número de telefone do cliente	VARCHAR(15)	N	+351 963897296	RD01
Morada	Endereço do cliente	-	N	-	RD01
Cod_Postal	Código postal	VARCHAR(10)	N	2337-999	RD01
Rua	Rua do cliente	VARCHAR(20)	N	Rua de Morioh	RD01
Localidade	Cidade do cliente	VARCHAR(20)	N	Suzuka	RD01

Tabela 3: Tabela de atributos de Cliente

Os atributos associados ao Stand podem ser observados na seguinte tabela:

Atributo	Descrição	Domínio	Nulo	Exemplo	Requisito
Id	Identificador sequencial do stand	INT	N	3	RD02
Data_inauguração	Data em que o stand foi inaugurado	DATA	N	12-12-2012	RD02
Telefone	Contacto do stand	INT(9)	N	252180799	RD02
Morada	Endereço do stand	-	N	-	RD02
Cod_Postal	Código postal	VARCHAR(10)	N	4470-995	RD02
Rua	Rua do stand	VARCHAR(20)	N	Rua da Caralinda	RD02
Localidade	Cidade do stand	VARCHAR(20)	N	Porto	RD02

Tabela 4: Tabela de atributos de um Stand

Os atributos associados a um veículo podem ser observados na seguinte tabela:

Atributo	Descrição	Domínio	Nulo	Exemplo	Requisito
Id	Identificador único de um veículo	INT	N	1633	RD04
Matricula	Matricula de um veículo	VARCHAR(8)	N	06-08-TM	RD04
Modelo	Modelo do veículo	VARCHAR(20)	N	Tumbler	RD04
Marca	Marca do veículo	VARCHAR(20)	N	Toyota	RD04
Tipo	Qual o tipo do carro(standard ou desportivo)	VARCHAR(10)	N	desportivo	RD04
Estado	Qual o estado atual do veículo(disponível ou alugado)“	VARCHAR(10)	N	alugado	RD04
Preço	Qual o preço do veículo por dia	DECIMAL(5,2)	N	279.99	RD04
Histórico	Anotações sobre o véiculo, se já esteve envolvido em algum acidente, se já apresentou algum tipo de problema mecânico	VARCHAR(255)	N	Acidente leve em 2022	RD04

Tabela 5: Tabela de atributos de um veículo

Os atributos associados a um funcionário podem ser observados na seguinte tabela:

Atributo	Descrição	Domínio	Nulo	Exemplo	Requisito
Id	Identificador único de um funcionário	INT	N	26	RD05
Nome	Nome completo de um funcionário	VARCHAR(64)	N	Beatriz Gomes Silva	RD05
Cargo	Qual o cargo que o funcionário exerce	VARCHAR(20)	N	Gerente	RD05
Horas_trabalhadas	A quantidade de horas que o funcionário trabalhou	INT	N	112	RD05

Tabela 6: Tabela de atributos de um funcionário

Os atributos associados a um aluguer podem ser observados na seguinte tabela:

Atributo	Descrição	Domínio	Nulo	Exemplo	Requisito
Id	Identificador único de um aluguer	INT	N	1478	RD06
Data_inicio	Data em que o carro foi alugado	DATA	N	15-01-2024	RD06
Data_fim	Data de final de aluguer do carro	DATA	N	23-01-2024	RD06
Preço_total	Custo total do aluguer	Decimal(6,2)	N	123.77	RD06
Estado	Estado em que o veículo foi devolvido	VARCHAR(40)	N	Sem problemas	RD06

Tabela 7: Tabela de atributos de um aluguer

3.5. Apresentação e Explicação do Diagrama ER Produzido

Após a conclusão dos passos anteriores procedeu-se à esquematização do modelo conceptual.

Como mencionado anteriormente foi utilizado o BrModelo para a criação do diagrama ER. Inicialmente foram criadas todas as entidades, de seguida foi feito os relacionamentos entre cada uma delas e no final foram atribuídos os atributos respetivos a cada uma delas.

Algumas considerações a ter sobre o modelo construído é que o software utilizado tem a sua adaptação da notação de Chen. Sendo as mais notáveis o facto que uma chave primária está identificada através de um círculo preenchido, para além disso a cardinalidade e participação são representadas através de um tuplo (participação, cardinalidade) como por exemplo na relação Stand tem Funcionário temos cardinalidade **1:N** e participação **T:T** que no modelo conceptual é representado como visto na seguinte figura:

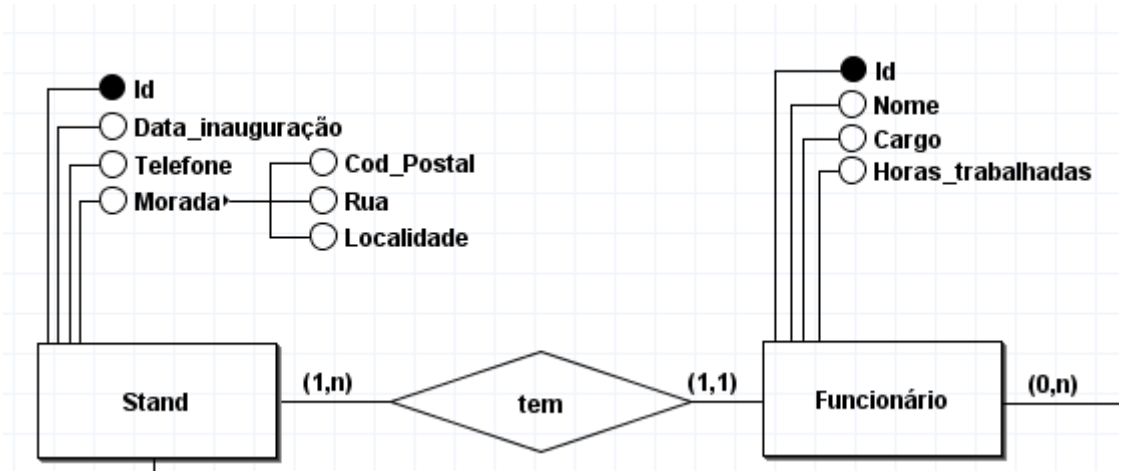

Figura 2: Exemplo de relação.

O diagrama ER produzido pode ser visto na figura seguinte.

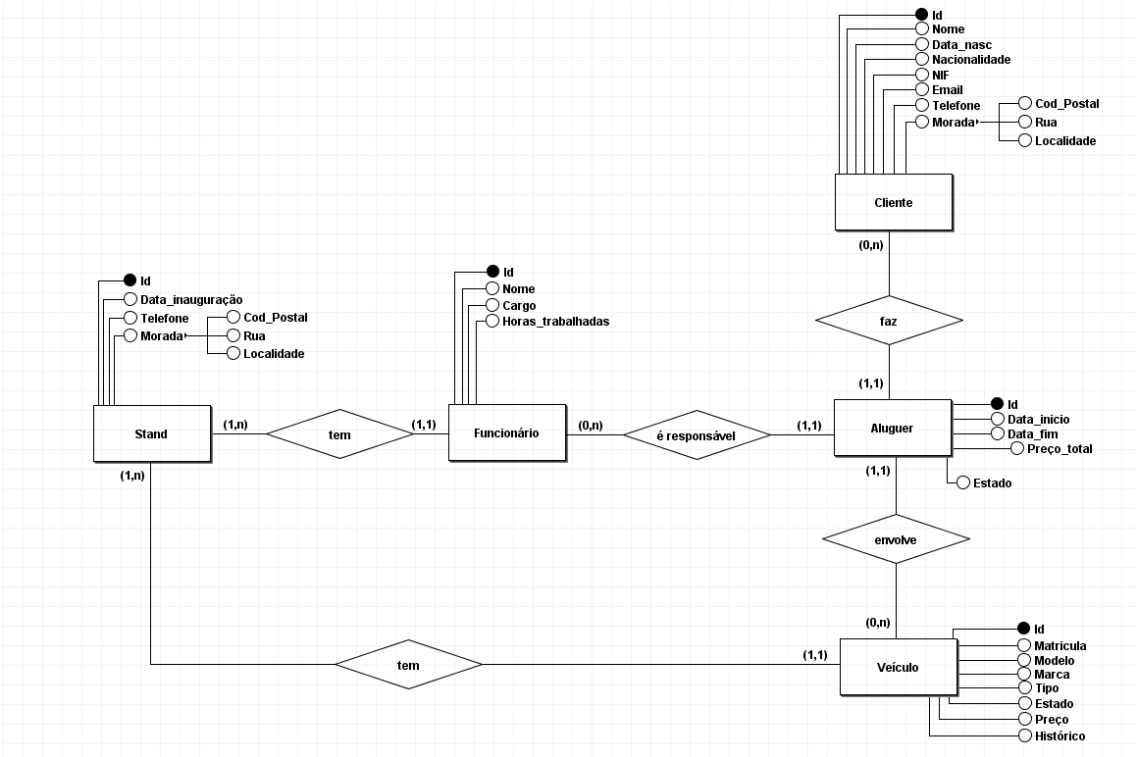

Figura 3: Modelo conceptual construído.

Após a construção do modelo conceptual este foi apresentado ao Sr Monteiro para a validação do mesmo. Uma vez que este acho por bem o modelo construído não foi necessário fazer nenhuma alteração ao mesmo e procedeu-se então ao passo seguinte.

4. Modelação Lógica

4.1. Construção do Modelo de Dados Lógico

Após a validação do modelo conceptual procedeu-se à criação do modelo de dados lógico. Para tal decidiu-se utilizar 2 abordagens, uma vez que o modelo conceptual foi criado através do software brModelo e este apresenta uma função que transforma o modelo conceptual no modelo lógico decidiu-se tirar proveito desta funcionalidade. Para além disso foi ainda criado um modelo lógico manualmente utilizando o MySQLWorkench o que irá permitir no futuro simplificar a implementação física.

4.2. Apresentação e Explicação do Modelo Lógico

Como referido anteriormente um dos modelos lógicos criados foi utilizando o software brModelo, portanto vamos abordar qual o processo utilizado para criar o modelo feito no MySQLWorkbench e no final comparar os dois modelos.

O primeiro passo para a construção do modelo lógico foi a criação de todas as entidades identificadas no modelo conceptual como tabelas. Cada uma destas tabelas tinha inicialmente os atributos simples da sua entidade, cada um deles com um tipo de dados apropriado, chave primária e atributos não nulos identificados como tal, bem como a decomposição dos seus atributos compostos nas entidades em que tal caso estava presente. De seguida foram foram estabelecidas as relações entre as tabelas através da adição de chaves estrangeiras respeitando a cardinalidade e participação presente no modelo conceptual.

Iremos agora apresentar o dicionário de dados de cada uma das tabelas criadas.

Atributo	**Domínio**	**Nulo**	**Chave estrangeira**
Id	INT	N	N
Nome	VARCHAR(64)	N	N
Data_nasc	DATE	N	N
Nacionalidade	VARCHAR(30)	N	N
NIF	INT(9)	N	N
Email	VARCHAR(75)	N	N
Telefone	VARCHAR(15)	N	N
Cod_Postal	VARCHAR(10)	N	N
Rua	VARCHAR(20)	N	N
Localidade	VARCHAR(20)	N	N

Tabela 8: Tabela do Cliente

A tabela “Cliente” teve origem na entidade “Cliente”. Os seus atributos simples deram origem a atributos na tabela, o atributo composto “Morada”, como é constituído por atributos simples deu origem aos atributos “Cod_Postal”, “Rua” e “Localidade” na tabela. As duas chaves identificadas como candidatas foram o “Id” e o “NIF”, uma vez que o Id é um número sequencial atribuído pela empresa este foi escolhido como a chave primária.

Atributo	Domínio	Nulo	Chave estrangeira
Id	INT	N	N
Data_inauguracao	DATE	N	N
Telefone	INT(9)	N	N
Cod_postal	VARCHAR(10)	N	N
Rua	VARCHAR(20)	N	N
Localidade	VARCHAR(20)	N	N

Tabela 9: Tabela do Stand

A tabela “Stand” teve origem na entidade “Stand”. Todos os atributos simples foram transportados diretamente para a tabela. Tal como em cliente o atributo composto “Morada” foi decomposto nos campos “Cod_Postal”, “Rua” e “Localidade”. Sendo o campo “Id” o identificador único atribuido a cada stand pela empresa este foi definido como chave primária.

Atributo	Domínio	Nulo	Chave estrangeira
Id	INT	N	N
Matricula	VARCHAR(8)	N	N
Modelo	VARCHAR(20)	N	N
Marca	VARCHAR(20)	N	N
Tipo	VARCHAR(10)	N	N
Estado	VARCHAR(10)	N	N
Preco	DECIMAL(5,2)	N	N
Historico	TEXT	S	N
StandId	INT	N	S(“Stand”)

Tabela 10: Tabela do Veiculo

A tabela “Veículo” teve origem através da entidade “Veículo”. Mais uma vez os seus atributos simples foram transportados diretamente para a tabela. Os candidatos a chave primária seriam o “Id” e a “Matricula”, mais uma vez optou-se por escolher o identificador único exclusivo à empresa como chave primária. O atributo StandId trata-se de uma chave estrangeira referente a um Stand o que permite identificar a que Stand pertence o Veículo de acordo com o relacionamento Stand tem Veículo.

Atributo	Domínio	Nulo	Chave estrangeira
Id	INT	N	N
Nome	VARCHAR(64)	N	N
Cargo	VARCHAR(20)	N	N
Horas_trabalhardas	INT	N	N
StandId	INT	N	S(“Stand”)

Tabela 11: Tabela de Funcionário

A tabela “Funcionario” teve origem na entidade “Funcionario”. Todos os seus atributos simples foram transportados diretamente para a tabela. Neste caso o único candidato a chave primária era o “Id” e portanto foi escolhido como tal. “StandId” trata-se da chave estrangeira referente a um Stand onde o funcionário trabalha.

Atributo	Domínio	Nulo	Chave estrangeira
Id	INT	N	N
Data_inicio	DATE	N	N
Data_fim	DATE	N	N
Preco_total	DECIMAL(6,2)	N	N
Estado	VARCHAR(40)	N	N
ClienteId	INT	N	S(“Cliente”)
FuncionarioId	INT	N	S(“Funcionario”)
VeiculoId	INT	N	S(“Veiculo”)

Tabela 12: Tabela de Aluguer

A tabela “Aluguer” deriva da entidade do mesmo nome. Todos os seus atributos simples foram deram origem a atributos na tabela. A chave primária escolhida foi o “Id”, um número sequencial para cada aluguer. Os campos “ClienteId”, “FuncionarioId” e “VeiculoId” são chaves estrangeiras que referenciam respetivamente o cliente que fez o aluguer, o funcionário responsável e o veículo alugado.

Os modelos criados podem ser vistos nas figuras seguintes:

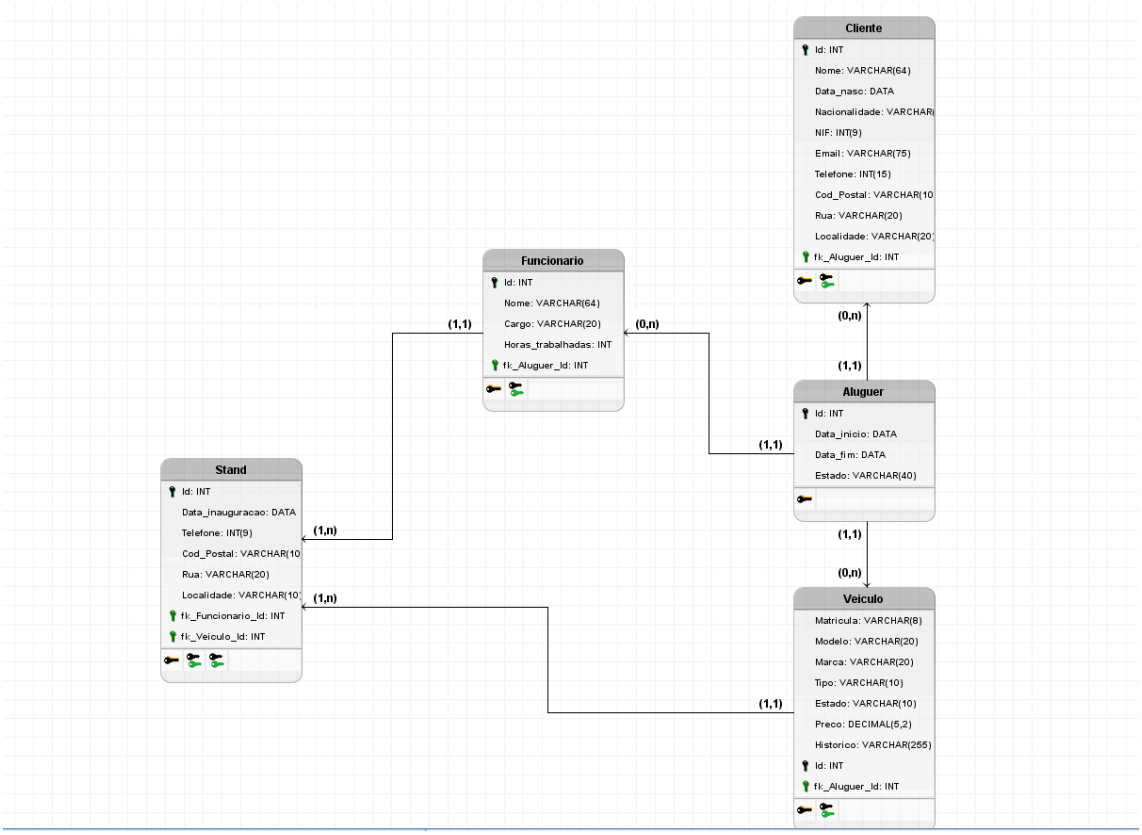

Figura 4: Modelo criado automaticamente pelo brModelo

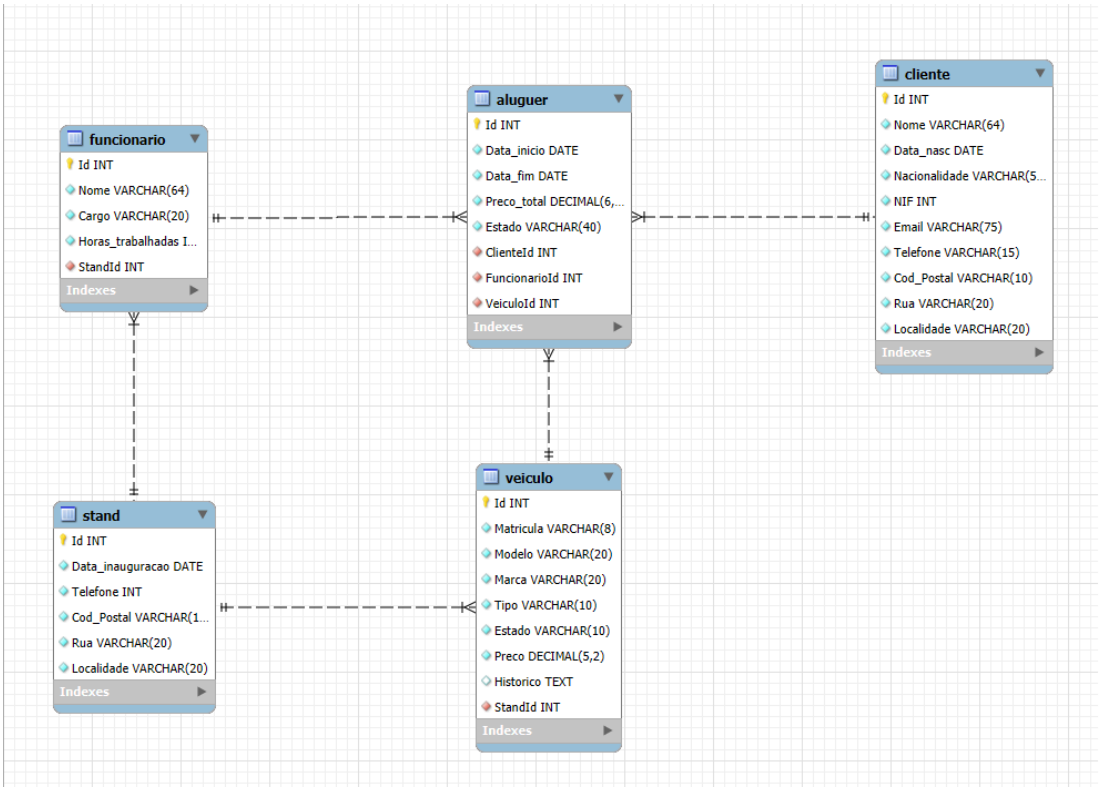

Figura 5: Modelo criado no MySQLWorkbench

É de notar que existem diferenças nos modelos. Por exemplo podemos observar na tabela cliente do modelo criado automaticamente pelo brModelo que este possui uma chave estrangeira referente a aluguer. Esta relação implica que um cliente estaria associado a um aluguer o que vai contra os requisitos levantados, o objetivo é que o cliente possa fazer vários alugueres e um aluguer apenas possa pertencer a um cliente. Portanto seria mais lógico o aluguer ter uma chave estrangeira referente ao cliente que fez o aluguer tal como feito no modelo criado manualmente. Podemos aplicar a mesma lógica na relação entre "Veículo" "Aluguer" e "Funcionário" "Aluguer".

Decidiu-se então prosseguir com o modelo criado manualmente no MySQLWorkbench.

4.3. Normalização de Dados

Após a construção do modelo lógico é essencial verificar se o modelo estava normalizado.

Analisando o modelo criado podemos verificar que este se encontra na Primeira Forma Normal uma vez que todos os seus atributos são atómicos. De seguida podemos observar que não existem dependências parciais, isto é, cada atributo de uma tabela depende unicamente da sua chave primária, logo o modelo encontra-se na Segunda Forma Normal. Por fim podemos reparar que nenhum atributo depende de outro atributo não chave, portanto não existem dependências transitivas o que nos permite afirmar que o modelo encontra-se também na Terceira Forma Normal.

Uma vez que todas as tabelas do modelo lógico foram analisadas e se encontram na Terceira Forma Normal podemos concluir que o modelo está normalizado até à 3FN.

4.4. Validação do Modelo com Interrogações do Utilizador

Nesta fase, o objetivo é garantir que o modelo de dados lógico é capaz de responder às necessidades identificadas nos requisitos. Para tal, foi criado um conjunto de perguntas(queries), através do uso de Álgebra relacional, para cada um dos requisitos de manipulação.

As queries implementadas podem ser vistas de seguida:

- RM01 - Listar os veículos disponíveis, indisponíveis e totais?

$$\sigma_{\text{Estado='disponivel'}}(\text{Veiculo})$$

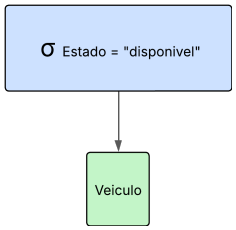

Figura 6: Árvore da interrogação do RM01 para veiculos disponiveis.

$$\sigma_{\text{Estado='indisponivel'}}(\text{Veiculo})$$

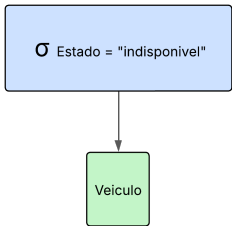

Figura 7: Árvore da interrogação do RM01 para veiculos indisponiveis.

$$\pi_{\text{Id, Matricula, Modelo, Marca, Tipo, Estado, Preco, Historico, StandId}}(\text{Veiculo})$$

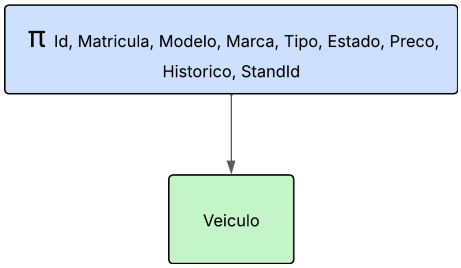

Figura 8: Árvore da interrogação do RM01 para os veiculos totais.

- RM02 - Listar funcionários

$$\pi_{\text{Id, Nome, Cargo, Horario\_trabalho, StandId}}(\text{Funcionario})$$

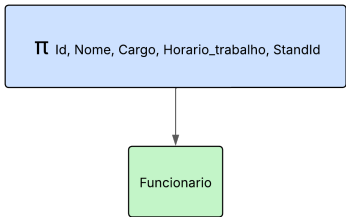

Figura 9: Árvore da interrogação do RM02.

- RM03 - Listar stands

$$\pi_{\text{Id, Data\_inauguracao, Telefone, Cod\_Postal, Rua, Localidade}}(\text{Stand})$$

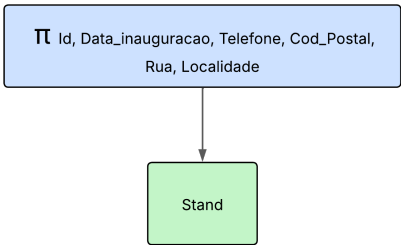

Figura 10: Árvore da interrogação do RM03.

- RM04 - Listar clientes

$$\pi_{\text{Id, Nome, Data\_nasc, Nacionalidade, NIF, Email, Telefone}}(\text{Cliente})$$

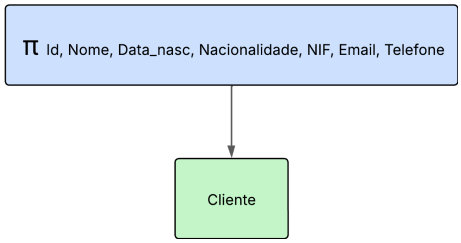

Figura 11: Árvore da interrogação do RM04.

- RM05 - Calcular preço do aluguer

$$\pi_{\text{Id, (Data\_fim - Data\_inicio)} \times \text{Preco} \rightarrow \text{PrecoCalc}}(\text{Aluguer} \bowtie \text{Veiculo})$$

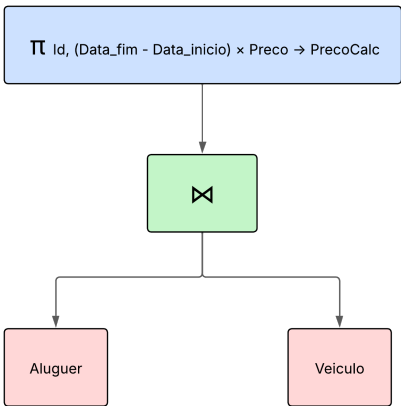

Figura 12: Árvore da interrogação do RM05.

- RM06 - Visualizar alugueres de um cliente c

$$\sigma_{\text{ClienteId} = \text{c}}(\text{Aluguer})$$

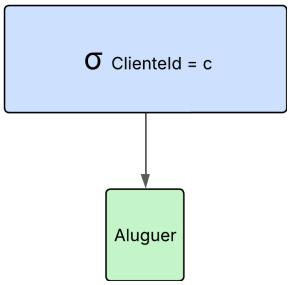

Figura 13: Árvore da interrogação do RM06.

- RM07 - Média de alugueres por dia de um stand s

$$AS = (\sigma_{StandId\ =\ s}(Veiculo)) \bowtie Aluguer$$

$$R = \gamma_{\ Data\_inicio\ ;\ COUNT(*)} \rightarrow \ NumAlug(AS)$$

$$\gamma_{\ ;\ AVG(NumAlug)} \rightarrow Media(R)$$

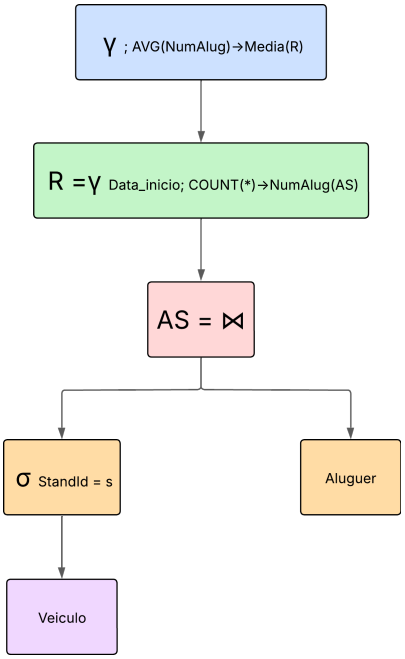

Figura 14: Árvore da interrogação do RM07.

- RM08 - Alugueres por funcionário f

$$\sigma_{FuncionarioId\ =\ f}\ (Aluguer)$$

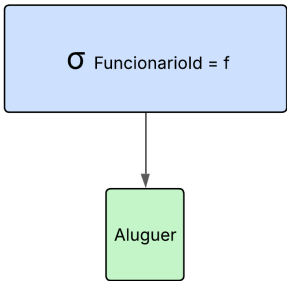

Figura 15: Árvore da interrogação do RM08.

- RM09 - Consultar horas trabalhas por um funcionário f

$$\pi_{\text{Nome, Horas\_trabalhadas}}(\sigma_{\text{Id = f}}(\text{Funcionario}))$$

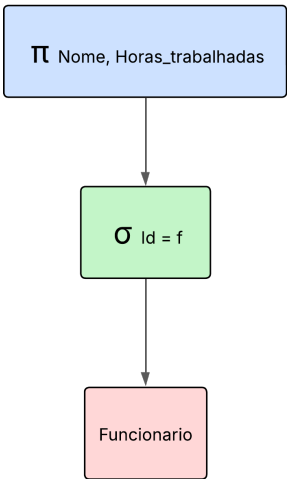

Figura 16: Árvore da interrogação do RM09.

- RM10 - Veículos por tipo t

$$\sigma_{\text{Tipo = t}}\ (\text{Veiculo})$$

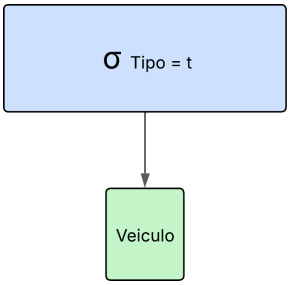

Figura 17: Árvore da interrogação do RM10.

- RM11 - Clientes por nacionalidade n

$$\sigma_{\text{Nacionalidade = n}}\ (\text{Cliente})$$

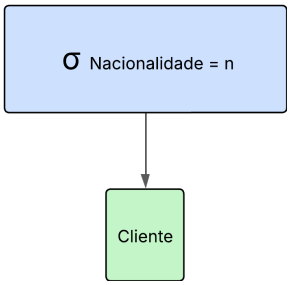

Figura 18: Árvore da interrogação do RM11.

- RM12 - Clientes por localidade l

$$\sigma_{\text{Localidade} = \text{l}} \ (\text{Cliente})$$

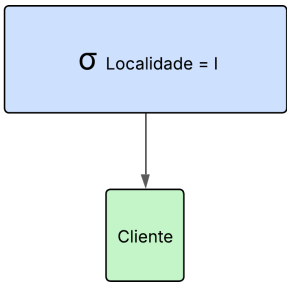

Figura 19: Árvore da interrogação do RM12.

Através do uso de Álgebra Relacional verificou-se possível a implementação de todas as interrogações e, assim, o modelo lógico foi considerado validado e pronto a ser implementado fisicamente.

5. Implementação Física

5.1. Apresentação e explicação da base de dados implementada

Após a conclusão das fases anteriores podemos agora começar a implementação física da base de dados. Para tal, foi construído um script SQL com o objetivo de criar todas as estruturas definidas no modelo lógico. Iremos agora de seguida abordar os diferentes passos que levaram à criação da BD.

O primeiro passo foi a criação da própria base de dados que foi feita da seguinte forma:

```
CREATE DATABASE IF NOT EXISTS PM_Veículos
CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

Figura 20: Criação da base de dados

Nesta instrução é criada a BD "PM_Veículos" caso esta ainda não exista no sistema. A base de dados é configurada segundo os parâmetros *CHARACTER SET utf8mb4*, para que seja possível armazenar no sistema um conjunto de caracteres mais abrangentes, tornando assim o sistema mais robusto.O parêmtro *COLLATE utf8mb4_unicode_ci* define a ordenação dos textos como insensível a maiúsculas/minúsculas e acentos o que permite clareza na forma como comparações e ordenações são feitas na BD.

Com a base de dados criada, torna-se agora possível definir as instruções de criação das tabelas e restantes objetos que irão compor o sistema.

Para iniciar a definição das tabelas provenientes do modelo lógico, é importante começar pelas tabelas independentes — ou seja, aquelas que não possuem chaves estrangeiras e, portanto, não dependem de nenhuma outra tabela para serem criadas. Portanto as primeiras tabelas a serem definidas serão Stand e Cliente.

A tabela Stand foi estruturada para armazenar a informação relativa aos diferentes pontos físicos da empresa. Esta tabela inclui um conjunto de colunas que identificam os dados essenciais de cada stand. O campo Id representa o identificador único de cada stand, sendo definido como chave primária e não podendo assumir valores nulos. A coluna Data_inauguração corresponde à data em que o stand foi inaugurado, sendo do tipo DATE e obrigatória. O contacto telefónico é armazenado na coluna Telefone, representado por um número inteiro de nove dígitos. O Cod_Postal, onde é registado o código postal do stand (um VARCHAR de tamanho 10). A Rua, que guarda o nome da rua (também VARCHAR, com limite de 20 caracteres).Por fim, Localidade, que indica a cidade onde o stand se encontra, igualmente definida como VARCHAR(20) e de preenchimento obrigatório.

A criação da tabela foi feita com a seguinte instrução SQL:

```sql
CREATE TABLE IF NOT EXISTS Stand (
    Id INT PRIMARY KEY AUTO_INCREMENT,
    Data_inauguração DATE NOT NULL,
    Telefone INT(9) NOT NULL,
    Cod_Postal VARCHAR(10) NOT NULL,
    Rua VARCHAR(20) NOT NULL,
    Localidade VARCHAR(20) NOT NULL
);
```

Figura 21: Criação da tabela Stand

A tabela Cliente foi concebida para armazenar os dados pessoais e de contacto de cada cliente registado no sistema. O campo Id funciona como identificador único de cada cliente, sendo definido como chave primária e do tipo inteiro, não permitindo valores nulos. O nome completo do cliente é registado na coluna Nome, utilizando um campo do tipo VARCHAR com um limite de 64 caracteres e também de preenchimento obrigatório. A coluna Data_nasc guarda a data de nascimento do cliente, sendo do tipo DATE e obrigatória. A nacionalidade é armazenada na coluna Nacionalidade, com um limite de 50 caracteres e sem permitir valores nulos. O campo NIF corresponde ao número de identificação fiscal do cliente, sendo obrigatório e representado como um número inteiro de nove dígitos. O endereço de correio eletrónico é registado na coluna Email, utilizando VARCHAR(75) e sendo de preenchimento obrigatório. O número de telefone do cliente é guardado na coluna Telefone, com um limite de 15 caracteres, permitindo assim o armazenamento de prefixos internacionais. A morada completa do cliente é representada pelas colunas Cod_Postal, Rua e Localidade, todas definidas como campos do tipo VARCHAR, respetivamente com 10, 20 e 20 caracteres de limite, e todas de preenchimento obrigatório.

A criação da tabela Cliente foi feita com a seguinte instrução SQL:

```sql
CREATE TABLE IF NOT EXISTS Cliente (
    Id INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(64) NOT NULL,
    Data_nasc DATE NOT NULL,
    Nacionalidade VARCHAR(50) NOT NULL,
    NIF INT(9) NOT NULL,
    Email VARCHAR(75) NOT NULL,
    Telefone VARCHAR(15) NOT NULL,
    Cod_Postal VARCHAR(10) NOT NULL,
    Rua VARCHAR(20) NOT NULL,
    Localidade VARCHAR(20) NOT NULL
);
```

Figura 22: Criação da tabela Cliente

A tabela Funcionario foi criada com o objetivo de armazenar a informação relativa aos trabalhadores da empresa. O campo Id funciona como identificador único de cada funcionário e é definido como chave primária, sendo um número inteiro obrigatório. O nome do funcionário é registado na coluna Nome, utilizando um campo VARCHAR com um limite de 64 caracteres, também obrigatório. O cargo ocupado pelo funcionário é indicado na coluna Cargo, representado por um VARCHAR de até 20 caracteres e sem permitir valores nulos. As horas totais de trabalho atribuídas a cada funcionário são registadas na coluna Horas_trabalhadas, utilizando o tipo INT e sendo de preenchimento obrigatório. Por fim, a coluna StandId identifica o stand onde o funcionário trabalha, funcionando como chave estrangeira que referencia o campo Id da tabela Stand. Esta ligação permite manter a integridade relacional entre os funcionários e os stands aos quais estão associados.

A criação da tabela Funcionario foi feita com a seguinte instrução SQL:

```sql
CREATE TABLE IF NOT EXISTS Funcionario (
    Id INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(64) NOT NULL,
    Cargo VARCHAR(20) NOT NULL,
    Horas_trabalhadas INT NOT NULL,
    StandId INT NOT NULL,
    FOREIGN KEY (StandId) REFERENCES Stand(Id)
);
```

Figura 23: Criação da tabela Funcionario

A tabela Veiculo tem como finalidade armazenar toda a informação relevante sobre os veículos disponíveis nos diversos stands. O campo Id é o identificador único de cada veículo, definido como chave primária da tabela e representado por um número inteiro. A matrícula do veículo é guardada no campo Matricula, sendo este um valor do tipo VARCHAR com um limite de 8 caracteres e obrigatório. O modelo e a marca do veículo são registados respetivamente nas colunas Modelo e Marca, ambas do tipo VARCHAR com um tamanho máximo de 20 caracteres e de preenchimento obrigatório. O campo Tipo identifica se se trata de um veículo normal ou de competição, utilizando um VARCHAR(10) sem permitir valores nulos. A coluna Estado armazena a situação atual do veículo (por exemplo, disponível ou alugado), também com um VARCHAR(10) obrigatório. O preço de aluguer do veículo é indicado na coluna Preço, utilizando o tipo DECIMAL(5,2), que permite representar valores monetários com duas casas decimais. O campo Histórico é do tipo TEXT e armazena, caso exista, anotações relevantes sobre o veículo. Finalmente, o campo StandId liga o veículo ao stand ao qual pertence, funcionando como chave estrangeira que referencia o campo Id da tabela Stand, garantindo assim a integridade relacional entre veículos e os respetivos stands.

A criação da tabela Veiculo foi feita com a seguinte instrução SQL:

```sql
CREATE TABLE IF NOT EXISTS Veiculo (
    Id INT PRIMARY KEY AUTO_INCREMENT,
    Matricula VARCHAR(8) NOT NULL,
    Modelo VARCHAR(20) NOT NULL,
    Marca VARCHAR(20) NOT NULL,
    Tipo VARCHAR(10) NOT NULL,
    Estado VARCHAR(10) NOT NULL,
    Preço DECIMAL(5,2) NOT NULL,
    Histórico TEXT,
    StandId INT NOT NULL,
    FOREIGN KEY (StandId) REFERENCES Stand(Id)
);
```

Figura 24: Criação da tabela Veiculo

A tabela Aluguer é responsável por armazenar todas as informações relativas aos processos de aluguer de veículos. Cada aluguer é identificado de forma única através do campo Id, que atua como chave primária da tabela e é representado por um número inteiro não nulo. A coluna Data_início regista a data de início do aluguer, enquanto a Data_fim regista a data de fim, sendo ambas do tipo DATE e de preenchimento obrigatório. O valor total a ser pago pelo cliente é guardado na coluna Preço_total, definida como DECIMAL(6,2), permitindo representar o preço. O campo Estado descreve a condição do veículo após a sua devolução (por exemplo, “bom estado” ou “danificado”), sendo um VARCHAR(40) obrigatório.

Além das informações descritivas do aluguer, esta tabela estabelece ligações com outras três entidades fundamentais do sistema: o cliente que realizou o aluguer (ClienteId), o funcionário responsável pelo processo (FuncionarioId) e o veículo alugado (VeículoId). Estas três colunas funcionam como chaves estrangeiras, estabelecendo ligações respetivamente com os campos Id das tabelas Cliente, Funcionario e Veiculo.

A criação da tabela Aluguer foi feita com a seguinte instrução SQL:

```
CREATE TABLE IF NOT EXISTS Aluguer (
    Id INT PRIMARY KEY AUTO_INCREMENT,
    Data_início DATE NOT NULL,
    Data_fim DATE NOT NULL,
    Preço_total DECIMAL(6,2) NOT NULL,
    Estado VARCHAR(40) NOT NULL,
    ClienteId INT NOT NULL,
    FuncionarioId INT NOT NULL,
    VeículoId INT NOT NULL,
    FOREIGN KEY (ClienteId) REFERENCES Cliente(Id),
    FOREIGN KEY (FuncionarioId) REFERENCES Funcionario(Id),
    FOREIGN KEY (VeículoId) REFERENCES Veiculo(Id)
);
```

Figura 25: Criação da tabela Aluguer

5.2. Criação de utilizadores da base de dados

A base de dados "PM_Veiculos" será acedida por diferentes perfis de utilizadores, cada um com responsabilidades e permissões distintas. Para fazer esta gestão foram criados roles no MySQL, o que permite agrupar permissões e depois associá-las facilmente aos utilizadores do sistema. Os perfis definidos foram o proprietário, gerente e funcionário.

A criação das roles foi feita segundo as seguintes instruções SQL:

```
CREATE ROLE IF NOT EXISTS
    'proprietario'@'localhost',
    'gerente'@'localhost',
    'funcionario'@'localhost';
```

Figura 26: Criação das Roles

Após a definição de cada perfil é necessário atribuir as permissões a cada um deles. Estas permissões devem ter por base os requisitos de controlo levantados anteriormente. O primeiro requisito identifica o Sr. Pedro Monteiro, o proprietário da empresa como capaz de aceder a todos os dados da BD. O RC02 define os gerentes como sendo capazes de ver e editar todos os documentos associados ao seu stand, no entanto, é de realçar que o RC04 define que apenas o proprietário pode alterar o preço de um veículo. Por fim o RC03 revela que os funcionários apenas podem ver alguns dos seus dados e dos alugueres dos quais foram responsáveis.

As permissões foram atribuídas como pode ser visto na seguinte imagem:

```sql
-- Permissoes

-- RC01:
GRANT ALL PRIVILEGES ON PM_Veículos.* TO 'proprietario'@'localhost';

-- RC02:
GRANT SELECT, INSERT, UPDATE ON PM_Veículos.Cliente TO 'gerente'@'localhost';
GRANT SELECT, INSERT, UPDATE ON PM_Veículos.Funcionario TO 'gerente'@'localhost';
GRANT SELECT, INSERT, UPDATE ON PM_Veículos.Aluguer TO 'gerente'@'localhost';

-- Gerente altera tudo, menos o preço dos veículos
GRANT SELECT(Id, Matricula, Modelo, Marca, Tipo, Estado, Histórico, StandId),
      UPDATE(Matricula, Modelo, Marca, Tipo, Estado, Histórico, StandId),
      INSERT(Id, Matricula, Modelo, Marca, Tipo, Estado, Histórico, StandId)
ON PM_Veículos.Veiculo TO 'gerente'@'localhost';

-- RC03:
-- Funcionários podem ver a propria informação e os dados dos alugueres
GRANT SELECT(Nome, Cargo, Horas_trabalhadas) ON PM_Veículos.Funcionario TO 'funcionario'@'localhost';
GRANT SELECT(Estado, Data_início, Data_fim, Preço_total) ON PM_Veículos.Aluguer TO 'funcionario'@'localhost';

-- RC04:
-- Apenas o proprietário pode alterar o preço dos veículos
GRANT UPDATE(Preço) ON PM_Veículos.Veiculo TO 'proprietario'@'localhost';
```

Figura 27: Permissões das Roles

Com os perfis e respetivas permissões devidamente atribuídas, foi então possível proceder à criação dos utilizadores correspondentes. Primeiro foi criado o utilizador pedro.monteiro que se trata do dono da empresa, também se criou o utilizador ana.costa, gerente do stand do Porto e por fim foi criado o utilizador carlos.mendes, um dos funcionários da empresa. Finalmente foram atribuídas os respetivos roles a cada um desses utilizadores.

Este processo foi feito através do conjunto de instruções do seguinte excerto:

```sql
CREATE USER IF NOT EXISTS
    'pedro.monteiro'@'localhost' IDENTIFIED BY 'adminpm2025',
    'ana.costa'@'localhost' IDENTIFIED BY 'gerenteporto2025',
    'carlos.mendes'@'localhost' IDENTIFIED BY 'funcporto2025';

-- Atribuir as ROLES
GRANT 'proprietario'@'localhost' TO 'pedro.monteiro'@'localhost';
SET DEFAULT ROLE 'proprietario'@'localhost' TO 'pedro.monteiro'@'localhost';

GRANT 'gerente'@'localhost' TO 'ana.costa'@'localhost';
SET DEFAULT ROLE 'gerente'@'localhost' TO 'ana.costa'@'localhost';

GRANT 'funcionario'@'localhost' TO 'carlos.mendes'@'localhost';
SET DEFAULT ROLE 'funcionario'@'localhost' TO 'carlos.mendes'@'localhost';
```

Figura 28: Criação dos Utilizadores

5.3. Povoamento da base de dados

Nesta etapa irão finalmente ser inseridos dados na BD. Uma vez que um dos tema a ser abordado futuramente é a "Implementação do sistema de migração de dados" neste ponto iremos apenas focar no primeiro método de povoamento utilizado que foi a inserção manual de dados através de instruções SQL. Esta abordagem foi escolhida por ser mais adequada para um número reduzido de registos e por permitir controlo total sobre os dados introduzidos e funcionarão como um conjunto de registos de teste que permitem validar o funcionamento da base de dados. Para a tabela stand foram inseridos apenas 3 entradas uma vez que existem apenas 3 stands, nas restantes foram inseridos entre 5 a 10 registos cada.

Os dados foram inseridos utilizando instruções como as representas na seguinte imagem:

```sql
-- Inserção em Stand
INSERT INTO Stand (Id, Data_inauguração, Telefone, Cod_Postal, Rua, Localidade) VALUES
(1, '2012-12-12', 252180799, '4470-995', 'Rua da Caralinda', 'Porto'),
(2, '2015-09-15', 212345678, '4000-002', 'Avenida Central', 'Lisboa'),
(3, '2020-01-03', 213456789, '8000-003', 'Estrada do Sul', 'Portimão');

-- Inserção em Cliente
INSERT INTO Cliente (Id, Nome, Data_nasc, Nacionalidade, NIF, Email, Telefone, Cod_Postal, Rua, Localidade) VALUES
(1, 'João Silva', '1985-04-23', 'Portuguesa', 123456789, 'joao.silva@email.com', '919999999', '1100-001', 'Rua da Alegria', 'Lisboa'),
(2, 'Maria Santos', '1990-10-12', 'Portuguesa', 987654321, 'maria.santos@email.com', '929888888', '4200-002', 'Rua do Sol', 'Porto'),
(3, 'Carlos Neves', '1978-06-30', 'Portuguesa', 112233445, 'carlos.neves@email.com', '938777777', '3000-003', 'Rua das Oliveiras', 'Coimbra'),
(4, 'Ana Lopes', '1995-02-18', 'Brasileira', 223344556, 'ana.lopes@email.com', '967666666', '1500-004', 'Rua das Acácias', 'Lisboa'),
(5, 'Rui Martins', '1980-12-01', 'Portuguesa', 334455667, 'rui.martins@email.com', '968555555', '5000-005', 'Rua do Norte', 'Braga'),
(6, 'Sofia Almeida', '1992-11-22', 'Portuguesa', 445566778, 'sofia.almeida@email.com', '938444444', '6000-006', 'Rua do Mar', 'Setúbal'),
(7, 'Tiago Rocha', '1988-07-17', 'Portuguesa', 556677889, 'tiago.rocha@email.com', '927333333', '7000-007', 'Rua das Palmeiras', 'Évora');

-- Inserção em Funcionario
INSERT INTO Funcionario (Id, Nome, Cargo, Horas_trabalhadas, StandId) VALUES
(1, 'Carlos Mendes', 'Vendedor', 160, 1),
(2, 'Ana Costa', 'Gerente', 170, 2),
(3, 'Bruno Teixeira', 'Vendedor', 150, 1),
(4, 'Inês Pereira', 'Assistente', 140, 3),
(5, 'Fábio Lima', 'Vendedor', 155, 2),
(6, 'Clara Matos', 'Gerente', 165, 3);
```

Figura 29: Exemplo de inserção de registos

O conjunto de instruções SQL criados originalmente para povoar a BD foram essenciais para validar a integridade da mesma.

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

De forma a determinar a dimensão inicial da base de dados, assim como a realizar uma estimação da sua taxa de crescimento anual, será então necessário ter em consideração os tipos de dados que se encontram armazenados em cada tabela, assim como espaço reservado para os dados do MySQL. Segue então uma análise de forma a calcular o espaço necessário para o início da base de dados e para o seu futuro crescimento anual.

Assumindo que cada registo não se repete nas tabelas, seguimos com as seguintes especificações:

- A tabela Cliente: Consiste de Id (INT, 4B), Nome (VARCHAR(64), 33B), Data_Nascimento (DATE, 3B), Nacionalidade (VARCHAR(50), 26B), NIF (INT, 4B), Email (VARCHAR(75), 39B), Telefone (VARCHAR(15), 9B), Código_Postal (VARCHAR(10), 6B), Rua (VARCHAR(20), 11B), Localidade (VARCHAR(20), 11B), em que equivale a um total de 146 Bytes.
- A tabela Aluguer: Consiste de Id (Int, 4B), Data_Início (DATE, 3B), Data_Fim (DATE, 3B), Preço_Total (DECIMAL(6,2), 4B), Estado (VARCHAR(40), 21B), Cliente_Id (INT, 4B), FuncionarioId (INT, 4B), Veiculo_Id(INT ,4B), em que equivale a um total de 47 Bytes.

- A tabela Stand: Consiste de Id (Int, 4B), Data_Inauguração (DATE, 3B), Telefone (INT, 4B), Código_Postal (VARCHAR(10), 6B), Rua (VARCHAR(20), 11B), Localidade (VARCHAR(20), 11B), em que equivale a um total de 39 Bytes.
- A tabela Funcionário: Consiste de Id (INT, 4B), Nome (VARCHAR(64), 33B), Cargo (VARCHAR(20), 11B), Horas_Trabalhadas (INT, 4B), StandId (INT, 4B), em que equivale a um total de 56 Bytes.
- A tabela Veículo: Consiste de Id (INT ,4B), Matrícula (VARCHAR(8), 5B), Modelo (VARCHAR(20), 11B), Marca (VARCHAR(20), 11B), Tipo (VARCHAR(10), 6B), Estado (VARCHAR(10), 6B), Preço (DECIMAL(5,2), 3B), Histórico (TEXT, 102B), StandId (INT, 4B), em que equivale a um total de 172 Bytes.

De forma a calcular a taxa de crescimento anual, seguimos as expectativas seguintes:

- Para a tabela de clientes, estimamos cerca de 50 novos clientes por ano, assumindo que tem um novo cliente a cada 7 dias.
- Para a tabela de Aluguer, estimamos cerca de 100 novos registos por ano, em que assumimos que em média, 2 alugueres serão efetuados pelos clientes ativos.
- Para a tabela do Stand, estimamos que cerca de 2 novos registos por ano, considerando que a abertura das stands depende do sucesso e a um ritmo lento.
- Para a tabela de Funcionário, estimamos cerca de 5 registos por ano, considerando a abertura de stands e a necessidade de trabalhadores para os mesmos.
- Para a tabela de Veículo, estimamos cerca de 20 novos registos por ano, considerando a necessidade de novos veículos para combater a demanda dos mesmos devido às novas stands e demanda de clientes.

5.5. Definição e caracterização de vistas de utilização em SQL

Nesta secção são apresentadas várias vistas criadas com o objetivo de simplificar o acesso e a análise dos dados por parte dos diferentes utilizadores do sistema. As vistas são especialmente úteis para encapsular lógica de consulta complexa, fornecer uma camada de abstração sobre os dados reais e reforçar a segurança, ao permitir acesso restrito a colunas ou linhas específicas.

Cada uma das vistas apresentadas foi pensada para responder a necessidades reais de consulta. As instruções SQL seguintes ilustram algumas das vistas consideradas mais úteis.

```sql
-- Veículos disponíveis (RM01)
CREATE VIEW Veiculos_Disponiveis AS
SELECT * FROM Veiculo
WHERE Estado = 'Disponível';

-- Veículos indisponíveis (RM01)
CREATE VIEW Veiculos_Indisponiveis AS
SELECT * FROM Veiculo
WHERE Estado != 'Disponível';

-- Funcionários por stand (RM02)
CREATE VIEW Funcionarios_Stands AS
SELECT f.Id, f.Nome, f.Cargo, f.Horas_trabalhadas, s.Localidade AS Stand_Localidade
FROM Funcionario f
JOIN Stand s ON f.StandId = s.Id;

-- Clientes por localidade (RM12)
CREATE VIEW Clientes_Por_Localidade AS
SELECT Localidade, COUNT(*) AS Total_Clientes
FROM Cliente
GROUP BY Localidade;

-- Alugueres por cliente (RM06)
CREATE VIEW Alugueres_Cliente AS
SELECT a.Id AS AluguerId, c.Nome AS Cliente, v.Marca, v.Modelo, a.Data_início, a.Data_fim, a.Preço_total
FROM Aluguer a
JOIN Cliente c ON a.ClienteId = c.Id
JOIN Veiculo v ON a.VeículoId = v.Id;
```

Figura 30: Exemplo de Views

Por exemplo poderá ser muito útil um funcionário verificar quais são os veículos que se encontram disponíveis neste momento, portanto a vista Veículos_Disponíveis será bastante útil. Ou então por outro lado temos a vista Funcionarios_Stand que poderá ser útil de um ponto de vista de recurso humanos para a consulta dos gerentes ou do próprio dono da empresa.

5.6. Tradução das interrogações do utilizador para SQL

Durante a fase de levantamento dos requisitos, identificaram-se as principais necessidades às quais a base de dados terá de responder. Para isso, foram elaboradas expressões em álgebra relacional capazes de atender a essas necessidades.

Com as consultas em álgebra relacional já definidas, resta agora convertê-las para SQL. A seguir, apresenta-se o resultado dessa conversão, seguindo a mesma ordem em que os requisitos foram originalmente especificados.

Entrando agora em detalhe sobre cada um dos requisitos:

```sql
-- ================================================
-- RM01 – Listar veículos disponíveis, indisponíveis e totais
-- ================================================
-- 1. Veículos disponíveis
SELECT *
  FROM Veiculo
 WHERE Estado = 'Disponível';

-- 2. Veículos indisponíveis
SELECT *
  FROM Veiculo
 WHERE Estado <> 'Disponível';

-- 3. Todos os veículos
SELECT *
  FROM Veiculo;
```

Figura 31: RM01 - Listar veículos disponíveis, indisponíveis e total

Este requisito é respondido através de três consultas distintas, a primeira selecciona todos os veículos cujo Estado = 'Disponível', a segunda devolve os veículos cujo Estado <> 'Disponível' e a terceira apresenta todos os registos da tabela Veiculo.

```
-- ================================================
-- RM02 - Listar funcionários
-- ================================================
SELECT Id,
       Nome,
       Cargo,
       Horas_trabalhadas,
       StandId
  FROM Funcionario;


-- ================================================
-- RM03 - Listar stands
-- ================================================
SELECT Id,
       Data_inauguração,
       Telefone,
       Cod_Postal,
       Rua,
       Localidade
  FROM Stand;
```

Figura 32: RM02 - Listar funcionários | RM03 - Listar stands.

Na imagem anterior podemos observar os requisitos RM02 e RM03 implementados em SQL.

O primeiro efetua um SELECT nos campos Id, Nome, Cargo, Horas_trabalhadas e StandId da tabela Funcionario, obtendo informação de cada colaborador.

Já o segundo realiza um SELECT em Stand, devolvendo Id, Data_inauguração, Telefone, Cod_Postal, Rua e Localidade para apresentar informação detalhada de cada Stand.

```
-- ================================================
-- RM04 - Listar clientes
-- ================================================
SELECT Id,
       Nome,
       Data_nasc,
       Nacionalidade,
       NIF,
       Email,
       Telefone
  FROM Cliente;
```

Figura 33: RM04 - Listar clientes

Este requisito interroga a tabela Cliente, seleccionando Id, Nome, Data_nasc, Nacionalidade, NIF, Email e Telefone, permitindo apresentar os dados principais sobre todos os clientes.

```sql
-- ================================================
-- RM05 – Calcular preço do aluguer
-- ================================================
-- Para cada aluguer, multiplica o número de dias pelo preço diário do veículo
SELECT A.Id AS AluguerId,
       DATEDIFF(A.Data_fim, A.Data_início) * V.Preço AS PrecoCalc
  FROM Aluguer AS A
  JOIN Veiculo AS V
    ON A.VeículoId = V.Id;
```

Figura 34: RM05 - Calcular preço do aluguer

Na imagem anterior podemos observar um JOIN entre Aluguer e Veiculo, calculando a diferença em dias entre Data_fim e Data_início, multiplicando esse valor pelo Preço diário do veículo e devolvendo o valor total (PrecoCalc) para cada aluguer.

```sql
-- ================================================
-- RM06 – Visualizar alugueres de um cliente c
-- ================================================
SELECT *
  FROM Aluguer
 WHERE ClienteId = 2;
```

Figura 35: RM06 - Alugueres de um determinado cliente

Aqui a tabela Aluguer é filtrada com WHERE ClienteId = ?, devolvendo apenas os registos de aluguer associados ao cliente cujo identificador é passado como parâmetro.

```sql
-- ================================================
-- RM07 – Média de alugueres por dia de um stand s
-- ================================================
-- 1) Junta veículos do stand s com os respetivos alugueres
-- 2) Agrupa por dia de início e conta nº de alugueres
-- 3) Calcula a média desses valores
SELECT AVG(R.NumAlug) AS MediaAlugueresPorDia
  FROM (
         SELECT A.Data_início,
                COUNT(*) AS NumAlug
           FROM Aluguer AS A
           JOIN Veiculo AS V
             ON A.VeículoId = V.Id
          WHERE V.StandId = 1
          GROUP
             BY A.Data_início
       ) AS R;
```

Figura 36: RM07 - Calcular a média de alugueres por dia de um stand

Este requisito utiliza uma subconsulta que faz junção entre Aluguer e Veiculo, filtra pelo StandId = ?, agrupa por Data_início e conta o número de alugueres por dia. De seguida, calcula a média desses totais diários através da função AVG.

```sql
-- ================================================
-- RM08 - Alugueres por funcionário f
-- ================================================
SELECT *
  FROM Aluguer
 WHERE FuncionarioId = 1;
```

Figura 37: RM08 - Listar os alugueres dos quais um funcionário foi responsável

Aqui filtramos a tabela Aluguer com WHERE FuncionarioId = ?, devolvendo todos os alugueres pelos quais um determinado funcionário foi responsável.

```sql
-- ================================================
-- RM09 - Consultar horas trabalhadas por um funcionário f
-- ================================================
SELECT Nome,
       Horas_trabalhadas
  FROM Funcionario
 WHERE Id = 4;
```

Figura 38: RM09 - Consultar as horas trabalhadas por um funcionário

Para responder ao requisito RM09, seleccionamos o Nome e Horas_trabalhadas da tabela Funcionario com WHERE Id = ?, mostrando apenas as horas acumuladas pelo colaborador em questão.

```sql
-- ================================================
-- RM10 - Veículos por tipo t
-- ================================================
SELECT *
  FROM Veiculo
 WHERE Tipo = 'Desportivo';
```

Figura 39: RM10 - Listar veículos por tipo

Ao aplicar a condição WHERE Tipo = ?, esta consulta seleciona exclusivamente os veículos que correspondem ao tipo especificado e devolve todos os registos resultantes dessa filtragem.

```sql
-- ================================================
-- RM11 - Clientes por nacionalidade n
-- ================================================
SELECT *
  FROM Cliente
 WHERE Nacionalidade = 'Portuguesa';
```

Figura 40: RM11 - Listar os clientes por nacionalidade

Utilizando WHERE Nacionalidade = ?, a instrução recupera somente os clientes que partilham a nacionalidade indicada e exibe todos os registos que satisfaçam esse critério.

```
-- ================================================
-- RM12 - Clientes por localidade 1
-- ================================================
SELECT *
  FROM Cliente
 WHERE Localidade = 'Braga';
```

Figura 41: RM12 - Listar os clientes por localidade

Filtrando a tabela Cliente com WHERE Localidade = ?, este requisito retorna apenas os clientes residentes na localidade informada e apresenta o conjunto completo desses registos.

5.7. Indexação do Sistema de Dados

A indexação é uma técnica fundamental para otimizar o desempenho de uma base de dados, permitindo reduzir significativamente o tempo necessário para aceder a determinados registos. Um índice funciona de forma semelhante ao índice de um livro o que permite aceder mais rapidamente aos registos desejados sem ser necessário percorrer toda a tabela. Embora a base de dados desenvolvida para este projeto ainda não contenha um volume expressivo de dados, foram definidos alguns índices que, tendo em conta o modelo e as consultas mais prováveis, poderão vir a ser extremamente úteis à medida que o sistema cresce e ganha escala.

Um dos índices considerados foi o índice idx_veiculo_matricula. A matrícula de um veículo é um identificador único e, por essa razão, é muito provável que seja utilizada frequentemente em pesquisas ou validações. A existência deste índice permite acelerar a localização de um veículo específico sem necessidade de varrer toda a tabela Veiculo.

Foi também ponderada a criação do índice idx_veiculo_estado, tendo em conta que o estado de um veículo — por exemplo, se está “Disponível” ou “Alugado” — será certamente alvo de filtros frequentes em diversas consultas, como por exemplo para listar apenas os veículos prontos a alugar. Este índice poderá assim melhorar de forma significativa o desempenho dessas operações.

Adicionalmente, definiu-se o índice idx_aluguer_funcionario, que visa otimizar as consultas que recuperam todos os alugueres realizados por um determinado funcionário. Esta informação pode ser relevante tanto para fins administrativos como para análise de desempenho individual.

Apesar de os índices ainda não terem sido implementados, uma vez que os ganhos de desempenho neste momento seriam marginais, as respetivas instruções SQL já foram preparadas e encontram-se documentadas. A sua aplicação poderá ser feita de forma simples e rápida caso se verifique essa necessidade no futuro.

A criação dos índices ponderados foi feita segundo o seguinte conjunto de instruções:

```
CREATE INDEX idx_veiculo_matricula ON Veiculo(Matricula);

CREATE INDEX idx_veiculo_estado ON Veiculo(Estado);

CREATE INDEX idx_aluguer_funcionario ON Aluguer(FuncionarioId);
```

Figura 42: Índices criados

5.8. Implementação de procedimentos, funções e gatilhos

Nesta secção foram implementados três mecanismos importantes no contexto de bases de dados relacionais: um procedimento armazenado, uma função e um gatilho. Estas estruturas aumentam o poder expressivo da base de dados, permitindo encapsular lógica de negócio diretamente no sistema de gestão de base de dados, promovendo consistência, segurança e desempenho.

Começando pelo procedimento armazenado, foi criada uma rotina denominada Realizar_Aluguer que tem como objetivo automatizar o processo de criação de um novo aluguer. Este procedimento recebe como parâmetros os identificadores do cliente, do veículo e do funcionário, bem como as datas de início e fim do aluguer. Internamente, calcula o preço total com base no valor diário do veículo e na duração do aluguer, e insere automaticamente o registo na tabela Aluguer. Para garantir a integridade de todo o processo, este procedimento utiliza uma transação explícita que assegura que, em caso de erro em qualquer uma das operações, nenhuma alteração será efetuada. Esta abordagem garante consistência e fiabilidade, evitando por exemplo situações em que um aluguer seja criado sem um cálculo correto do preço ou com referência a um veículo inexistente.

O procedimento armazenado foi criado através da seguinte instrução:

```
-- ------------------------------------------------
-- Procedimento armazenado para alugar um veículo
-- ------------------------------------------------
DELIMITER $$
CREATE PROCEDURE Realizar_Aluguer (
    IN p_clienteId INT,
    IN p_funcionarioId INT,
    IN p_veiculoId INT,
    IN p_data_inicio DATE,
    IN p_data_fim DATE
)
BEGIN
    DECLARE v_preco_dia DECIMAL(10,2);
    DECLARE v_duracao INT;
    DECLARE v_preco_total DECIMAL(10,2);

    START TRANSACTION;

    -- Obter preço diário do veículo
    SELECT Preço INTO v_preco_dia
    FROM Veiculo
    WHERE Id = p_veiculoId;

    -- Calcular duração
    SET v_duracao = DATEDIFF(p_data_fim, p_data_inicio);

    -- Calcular preço total
    SET v_preco_total = v_duracao * v_preco_dia;

    -- Inserir aluguer
    INSERT INTO Aluguer (Data_início, Data_fim, Preço_total, Estado, ClienteId, FuncionarioId, VeículoId)
    VALUES (p_data_inicio, p_data_fim, v_preco_total, 'Normal', p_clienteId, p_funcionarioId, p_veiculoId);

    COMMIT;
END$$
DELIMITER ;
```

Figura 43: Procedimento de realização de aluguer

Complementarmente, foi criada uma função chamada Calcular_Duracao, que recebe duas datas e retorna o número de dias entre elas. Esta função é útil para normalizar o cálculo da duração de alugueres, podendo ser reutilizada em várias consultas e procedimentos que necessitem dessa lógica de forma consistente.

A função foi criada segundo a seguinte instrução:

```
-- ---------------------------------------------
-- Função para calcular a duração de um aluguer
-- ---------------------------------------------
DELIMITER $$

CREATE FUNCTION Calcular_Duracao(data_inicio DATE, data_fim DATE)
RETURNS INT
DETERMINISTIC
BEGIN
    RETURN DATEDIFF(data_fim, data_inicio);
END$$

DELIMITER ;
```

Figura 44: Função de calcular duração

Por fim, foi implementado um gatilho Atualizar_Estado_Veiculo associado à inserção de novos registos na tabela Aluguer. Este gatilho atualiza automaticamente o estado do veículo correspondente para "Indisponível" assim que um novo aluguer é efetuado. Deste modo, evita-se que o mesmo veículo seja alugado em simultâneo por mais do que um cliente, assegurando integridade lógica no sistema.

O gatilho foi criado através da seguinte instrução:

```
-- ---------------------------------------------
-- Atualizar o estado do veiculo ao ser alugado
-- ---------------------------------------------
DELIMITER $$

CREATE TRIGGER Atualizar_Estado_Veiculo
AFTER INSERT ON Aluguer
FOR EACH ROW
BEGIN
    UPDATE Veiculo
    SET Estado = 'Alugado'
    WHERE Id = NEW.VeículoId;
END$$

DELIMITER ;
```

Figura 45: Gatilho que atualiza o estado de um veículo ao ser alugado

Estes três mecanismos demonstram como é possível automatizar, validar e controlar ações críticas dentro do sistema de gestão de alugueres de veículos, reduzindo a necessidade de ações manuais e prevenindo erros comuns em operações de escrita.

6. Implementação do sistema de migração dados

Para garantir a integração de dados provenientes de diferentes origens na base de dados central, foi desenvolvido um sistema de migração que processa e insere informação a partir de três fontes distintas: um ficheiro SQL relacional, um conjunto de ficheiros CSV e um ficheiro JSON. Esta abordagem permite simular um cenário realista em que os dados podem estar dispersos por múltiplos formatos e sistemas.

A primeira fonte corresponde a um conjunto de instruções SQL previamente utilizadas para povoar a base de dados durante as fases iniciais do projeto. Este script, preparado manualmente, contém comandos INSERT que inserem registos nas tabelas principais, como Stand, Cliente, Funcionário, Veículo e Aluguer. Por ser uma fonte relacional, a sua estrutura coincide com a do modelo da base de dados central, facilitando a inserção direta dos dados.

A segunda fonte é composta por dois ficheiros CSV: um para a tabela Cliente e outro para a tabela Funcionário. Cada ficheiro contém um pequeno conjunto adicional de registos, que complementa os dados inseridos previamente através do SQL. A escolha do formato CSV prende-se com a sua ampla adoção em sistemas de gestão de dados e com a facilidade de manipulação e visualização em ferramentas como folhas de cálculo. A leitura e inserção destes dados é realizada através da biblioteca pandas em Python, que permite transformar os dados em comandos SQL compatíveis com o SGBD MySQL.

A terceira fonte de dados inclui dois ficheiros JSON, correspondentes às tabelas Veículo e Aluguer. Este formato é frequentemente utilizado em APIs modernas e aplicações web, permitindo representar estruturas de dados de forma hierárquica e legível. Os ficheiros JSON preparados para este projeto contêm novos registos a serem integrados na base de dados central. A sua leitura foi feita recorrendo à biblioteca json, sendo os dados processados e inseridos linha a linha através de comandos INSERT.

O processo de migração foi centralizado num único script Python, responsável por ler os dados de cada uma das fontes, estabelecer ligação com a base de dados MySQL e executar os comandos de inserção. Para reforçar a segurança e flexibilidade do sistema, a palavra-passe da base de dados é lida diretamente do terminal, utilizando a biblioteca getpass, evitando que fique exposta no código-fonte.

O código python do processo de migração de dados pode ser dividido em 5 partes.

Inicialmente é feita a conexão à base de dados como pode ser visto na imagem seguinte:

```
# === 1. Conexão com a base de dados MySQL ===
user = input("Utilizador MySQL: ")
password = getpass.getpass("Password MySQL: ")
conn = mysql.connector.connect(
    host="localhost",
    user=user,
    password=password,
    database="pm_veículos"
)
cursor = conn.cursor()
```

Figura 46: Conexão à base de dados

De seguida é feita a execução do script SQL:

```python
# === 2. Executar instruções do ficheiro povoamentoPM.sql ===
with open("dados/PovoamentoPM.sql", "r", encoding="utf-8") as f:
    sql_script = f.read()
    for statement in sql_script.split(';'):
        if statement.strip():
            cursor.execute(statement)
    conn.commit()
    print("Script SQL executado com sucesso.")
```

Figura 47: Povoamento através de script SQL

Posteriormente insere-se os dados provenientes dos ficheiro CSV:

```python
# === 3. Inserir dados de ficheiros CSV ===
def inserir_csv_em_tabela(caminho, tabela):
    df = pd.read_csv(caminho)
    cols = ", ".join(df.columns)
    placeholders = ", ".join(["%s"] * len(df.columns))
    sql = f"INSERT INTO {tabela} ({cols}) VALUES ({placeholders})"
    for row in df.itertuples(index=False):
        cursor.execute(sql, tuple(row))
    conn.commit()
    print(f"Dados inseridos do CSV em '{tabela}'")

inserir_csv_em_tabela("dados/clientes.csv", "Cliente")
inserir_csv_em_tabela("dados/funcionarios.csv", "Funcionario")
```

Figura 48: Povoamento através de CSV

Finalmente insere-se os dados provenientes dos ficheiros JSON:

```python
# === 4. Inserir dados de ficheiros JSON ===
def inserir_json_em_tabela(caminho, tabela):
    with open(caminho, "r", encoding="utf-8") as f:
        dados = json.load(f)
    if not dados:
        return
    colunas = dados[0].keys()
    placeholders = ", ".join(["%s"] * len(colunas))
    sql = f"INSERT INTO {tabela} ({', '.join(colunas)}) VALUES ({placeholders})"
    for linha in dados:
        cursor.execute(sql, tuple(linha.values()))
    conn.commit()
    print(f"Dados inseridos do JSON em '{tabela}'")

inserir_json_em_tabela("dados/veiculos.json", "Veiculo")
inserir_json_em_tabela("dados/alugueres.json", "Aluguer")
```

Figura 49: Povoamento através de JSON

No final fecha-se a conexão à BD.

A escolha da linguagem Python para esta tarefa justifica-se pela sua versatilidade, simplicidade e riqueza em bibliotecas especializadas, como pandas para leitura de ficheiros CSV e json para ficheiros estruturados. O resultado é um processo automatizado, fiável e extensível, que permite integrar dados heterogéneos na base de dados relacional de forma coerente e segura.

7. Conclusões e Trabalho Futuro

7.1. Conclusão

O desenvolvimento do sistema de base de dados para a empresa de aluguer de veículos foi concluído com sucesso, abrangendo todas as fases essenciais: desde a análise de requisitos, modelação conceptual, transformação para modelo lógico relacional, implementação física da base de dados, até à migração e integração de dados provenientes de múltiplas fontes.

A construção do modelo relacional, devidamente normalizado até à Terceira Forma Normal (3FN), assegurou a integridade e a consistência dos dados, evitando redundâncias. A validação do modelo através de interrogações em Álgebra Relacional confirmou que este satisfaz os requisitos definidos.

A migração de dados foi executada com sucesso utilizando um script Python, capaz de integrar dados de um ficheiro SQL (modelo relacional), dois ficheiros CSV e dois ficheiros JSON. Esta tarefa reforçou a flexibilidade do sistema, demonstrando a capacidade de integração com diferentes formatos de dados — uma realidade comum em ambientes empresariais.

Durante o processo de desenvolvimento foram identificados vários aspetos positivos, como a clareza do modelo, a modularidade das fases, e a automatização da migração.

7.2. Trabalho Futuro

No futuro, é recomendável expandir o script de migração de dados de forma a incluir mecanismos de validação e controlo de erros, permitindo verificar automaticamente a integridade e a consistência dos dados provenientes de ficheiros CSV ou JSON antes da sua inserção na base de dados. Além disso, seria vantajoso implementar um sistema de backups automáticos que permita realizar cópias de segurança regulares da base de dados, garantindo a recuperação dos dados em caso de falha ou perda de informação.

8. Lista de Siglas e Acrónimos

BD - Base de dados.

P.M. Veículos - Pedro Monteiro Veículos, Pode Mandar.

9. Anexos

O anexo1 é um ficheiro do google sheets, enviado juntamente com este relatório de nome “Requisitos_BD”.

O anexo2 trata-se de uma pasta comprimida com todos os scripts SQL criados bem como os ficheiros e código python utilizada para o povoamento da base de dados.