

**Universidade do Minho**  
Escola de Engenharia  
Licenciatura em Engenharia Informática

## **Unidade Curricular de Bases de Dados**

Ano Letivo de 2024/2025

### **Belo Cars - Grupo 27**

**Afonso Martins**  
a106931

**Luis Felício**  
a106913

**Goncalo Castro**  
a107337

**Jorge Barbosa**  
a106799

Junho, 2025

# BD

Data da Receção	
Responsável	
Avaliação	
Observações	

**Belo Cars - Grupo 27**

**Afonso Martins**  
a106931

**Luis Felício**  
a106913

**Goncalo Castro**  
a107337

**Jorge Barbosa**  
a106799

Junho, 2025

## Resumo

Este relatório documenta o desenvolvimento de um Sistema de Gestão de Base de Dados para uma empresa especializada no aluguer de automóveis e veículos comerciais — a *Belo Cars*. O projeto nasce da necessidade de modernizar e otimizar os processos internos da empresa, assegurando maior segurança na gestão de dados, eficiência operacional e controlo administrativo mais rigoroso.

Numa fase inicial, o foco recaiu sobre a análise e modelação do sistema, contemplando a contextualização do problema, a definição dos requisitos e a construção dos modelos Conceptual e Lógico. Esta fase incluiu ainda a elaboração dos respetivos esquemas e descrições que fundamentam a estrutura da base de dados proposta.

Numa segunda fase, avançou-se para a implementação física do sistema em ambiente MySQL, que abrangeu a criação das tabelas, utilizadores e permissões, bem como a definição de vistas, índices, *queries*, procedimentos, funções, e o povoamento da base de dados com dados representativos. Foi ainda realizada uma análise detalhada do espaço de armazenamento utilizado e do potencial crescimento da base de dados ao longo do tempo.

O sistema resultante tem como objetivo proporcionar uma solução fiável, escalável e eficiente para a gestão de alugueres, contribuindo para a digitalização e melhoria contínua dos processos administrativos da *Belo Cars*.

**Área de Aplicação:** Desenho e Arquitetura de Sistemas de Bases de Dados.

**Palavras-Chave:** Sistema de Base de Dados, Belo Cars, Modelação Conceptual, Modelação Lógica, Aluguer, Implementação Física

# Índice

<b>1. Definição do Sistema</b>	<b>1</b>
1.1. Contextualização	1
1.2. Fundamentação	1
1.3. Objetivos	2
1.4. Viabilidade	3
1.5. Equipa de Trabalho	3
1.6. Recursos	3
1.7. Plano de Execução	4
1.8. Revisão e Aprovação	5
1.9. Notas Adicionais	6
<b>2. Levantamento e Análise de Requisitos</b>	<b>7</b>
2.1. Definição do Método	7
2.2. Levantamento	10
2.2.1. Vistas de Utilização	11
2.2.2. O Documento de Requisitos	11
2.3. Análise	12
2.4. Organização	12
2.5. Validação	14
<b>3. Modelação Conceptual</b>	<b>15</b>
3.1. Apresentação da Abordagem de Modelação Realizada	15
3.2. Identificação e Caracterização das Entidades	15
3.2.1. Cliente	15
3.2.2. Aluguer	16
3.2.3. Veículo	16
3.2.4. Funcionário	17
3.2.5. Stand	18
3.3. Identificação e Caracterização dos Relacionamentos	18
3.3.1. Cliente → Aluguer : 1 → n	19
3.3.2. Veículo → Aluguer : 1 → n	20
3.3.3. Veículo → Stand : 1 → n	20
3.3.4. Stand → Funcionario : 1 → n	21
3.3.5. Funcionario → Aluguer : 1 → n	21
3.4. Identificação e Caracterização dos Atributos das Entidades e dos Relacionamentos	22
3.5. Apresentação e Explicação do Modelo Conceptual Produzido	23
3.6. Validação do esquema	23
<b>4. Modelação Lógica</b>	<b>24</b>
4.1. Construção e Validação do Modelo de Dados Lógico	24
4.1.1. Derivação de Relações do Modelo de Dados Local	24
4.1.1.1. Cliente	24
4.1.2. Aluguer	25
4.1.3. Veiculo	26
4.1.4. Funcionário	26
4.1.5. Stand	27
4.2. Apresentação do Modelo Lógico Final Produzido	28

4.3. Normalização de Dados .....	28
4.4. Validação do Modelo recorrendo à Álgebra Relacional .....	29
<b>5. Implementação Física .....</b>	<b>33</b>
5.1. Apresentação e explicação da base de dados implementada .....	33
5.1.1. Tabelas do Modelo Físico .....	34
5.1.1.1. Criação da tabela Localizacao .....	34
5.1.1.2. Criação da tabela Stand .....	34
5.1.1.3. Criação da tabela Contactos_Stand .....	34
5.1.1.4. Criação da tabela Cliente .....	35
5.1.1.5. Criação da tabela Contactos_Cliente .....	35
5.1.1.6. Criação da tabela Funcionario .....	35
5.1.1.7. Criação da tabela Contactos_Funcionario .....	36
5.1.1.8. Criação da tabela Veiculo .....	36
5.1.1.9. Criação da tabela Aluguer .....	36
5.2. Criação de utilizadores da base de dados .....	37
5.2.1. Definição de Grupos de Utilizadores ( <i>Roles</i> ) .....	37
5.2.2. Atribuição de Permissões por <i>Role</i> .....	37
5.2.3. Criação de Utilizadores Convidados ( <i>Guests</i> ) .....	38
5.2.4. Criação de Funcionários ( <i>Staff</i> ) .....	38
5.2.5. Criação do Administrador ( <i>Admin</i> ) .....	38
5.2.6. Atribuição de <i>Roles</i> por Defeito .....	38
5.3. Povoamento da base de dados .....	38
5.3.1. Tabela Funcionário .....	39
5.3.2. Tabela Cliente .....	39
5.3.3. Tabela Stand .....	39
5.3.4. Tabela Veículo .....	39
5.3.5. Tabela Aluguer .....	40
5.3.6. Tabela Localização .....	40
5.3.7. Tabela Contactos_Stand .....	40
5.3.8. Tabela Contactos_Cliente .....	40
5.3.9. Tabela Contactos_Funcionario .....	41
5.4. Cálculo do espaço da base de dados (inicial e taxa de crescimento anual) .....	41
5.4.1. Tamanho da base de dados inicial .....	41
5.4.2. Tamanho da base de dados no futuro (crescimento anual) .....	42
5.5. Definição e caracterização de vistas de utilização em SQL .....	44
5.6. Indexação do Sistema de Dados .....	45
5.7. Implementação de procedimentos, funções e gatilhos .....	46
<b>6. Implementação do Sistema de Migração de Dados .....</b>	<b>50</b>
6.1. CSV .....	50
6.1.1. Importação de Bibliotecas e Configuração Inicial .....	50
6.1.2. Conexão com a Base de Dados .....	50
6.1.3. Extração dos dados guardados em ficheiro .....	51
6.1.4. Padronização de dados .....	51
6.1.5. Ordem de introdução das tabelas na base de dados .....	52
6.1.6. Introdução dos registos na base de dados .....	52
6.1.7. Função geral de introdução dos dados .....	52
6.1.8. Função principal do programa .....	53
6.1.9. Características do sistema .....	53
6.2. JSON .....	53
6.2.1. Importação de Bibliotecas e Configuração Inicial .....	53
6.2.2. Estabelecimento da Conexão com a Base de Dados .....	54
6.2.3. Definição da Diretoria dos Ficheiros Fonte .....	54
6.2.4. Mapeamento das Funções de Inserção .....	54
6.2.5. Função Principal de Carregamento e Inserção .....	54

6.2.6.	Função Principal do Programa .....	55
6.2.7.	Características do Sistema .....	55
6.3.	PostgreSQL .....	56
6.3.1.	Importação de Bibliotecas e Configuração de Logging .....	56
6.3.2.	Classe <i>DatabaseMigrator</i> - Inicialização .....	56
6.3.3.	Estabelecimento de Conexões .....	56
6.3.4.	Obtenção de Lista de Tabelas .....	57
6.3.5.	Obtenção do Esquema das Tabelas .....	57
6.3.6.	Conversão de Tipos de Dados .....	57
6.3.7.	Criação de Tabelas MySQL .....	58
6.3.8.	Criação de Tabelas MySQL .....	58
6.3.9.	Migração de Dados por Lotes .....	59
6.3.10.	Orquestração da Migração Completa .....	60
6.3.11.	Gestão de Conexões .....	60
6.3.12.	Função Principal com Configurações Seguras .....	60
6.3.13.	Ficheiro de Configuração (.env) .....	61
6.3.14.	Características do Sistema de Migração .....	61
<b>7.</b>	<b>Conclusões Críticas .....</b>	<b>63</b>
7.1.	Dificuldades Sentidas e Estratégias de Superação .....	63
7.2.	Alterações em relação à 1ª Fase .....	63
7.3.	Análise do Diagrama de Gantt Real vs Planeado .....	63
7.4.	Trabalho Futuro .....	64
<b>8.</b>	<b>Referência Bibliográfica .....</b>	<b>66</b>
	<b>Lista de Siglas e Acrónimos .....</b>	<b>67</b>
<b>Anexos</b>	<b>.....</b>	<b>68</b>
Anexo 1	Requisitos Gerais Pt.1 .....	68
Anexo 2	Requisitos Gerais Pt.2 .....	68
Anexo 3	Tabela de Requisitos de Descrição .....	69
Anexo 4	Tabela de Requisitos de Manipulação .....	70
Anexo 5	Tabela de Requisitos de Controlo .....	71
Anexo 6	Primeira Página do Questionário que serviu de inquérito aos funcionários da Belo Cars . . .	72
Anexo 7	Segunda Página do Questionário que serviu de inquérito aos funcionários da Belo Cars . . .	72
Anexo 8	Terceira Página do Questionário que serviu de inquérito aos funcionários da Belo Cars . . .	72
Anexo 9	Quarta Página do Questionário que serviu de inquérito aos funcionários da Belo Cars . . . .	72
Anexo 10	Quinta Página do Questionário que serviu de inquérito aos funcionários da Belo Cars . . . .	73
Anexo 11	Primeira Página da Ata de uma das Reuniões com a Sofia Moreau .....	73
Anexo 12	Segunda Página da Ata de uma das Reuniões com a Sofia Moreau .....	74
Anexo 13	Modelo Conceptual Final .....	74
Anexo 14	Modelo Lógico Final .....	75
Anexo 15	Esquema Físico gerado no MySQL Workbench .....	75
Anexo 16	Tamanho relativo a cada tabela (incluindo índices) .....	76
Anexo 17	Diagrama de Gantt Planeado .....	76
Anexo 18	Diagrama de Gantt Final (preenchido ao longo do desenvolvimento do projeto) .....	77

## Lista de Figuras

Figura 1	Diagrama de Gantt .....	5
Figura 2	Primeira Página da Ata de uma das Reuniões com a Sofia Moreau .....	7
Figura 3	Segunda Página da Ata de uma das Reuniões com a Sofia Moreau .....	8
Figura 4	Primeira Página do Questionário que serviu de inquérito aos funcionários da Belo Cars .....	9
Figura 5	Segunda Página do Questionário que serviu de inquérito aos funcionários da Belo Cars .....	9
Figura 6	Terceira Página do Questionário que serviu de inquérito aos funcionários da Belo Cars .....	9
Figura 7	Quarta Página do Questionário que serviu de inquérito aos funcionários da Belo Cars .....	9
Figura 8	Quinta Página do Questionário que serviu de inquérito aos funcionários da Belo Cars .....	9
Figura 9	Primeira Página da Tabela de Requisitos Gerais .....	12
Figura 10	Segunda Página da Tabela de Requisitos Gerais .....	12
Figura 11	Tabela de Requisitos de Descrição .....	13
Figura 12	Tabela de Requisitos de Manipulação .....	13
Figura 13	Tabela de Requisitos de Controlo .....	14
Figura 14	Caracterização geral das entidades .....	15
Figura 15	Entidade Cliente - Modelo Conceptual .....	16
Figura 16	Entidade Aluguer - Modelo Conceptual .....	16
Figura 17	Entidade Veículo - Modelo Conceptual .....	17
Figura 18	Entidade Funcionário - Modelo Conceptual .....	17
Figura 19	Entidade Stand - Modelo Conceptual .....	18
Figura 20	Caracterização geral dos relacionamentos .....	18
Figura 21	Relacionamento Cliente -> Aluguer - Modelo Conceptual .....	19
Figura 22	Relacionamento Veículo -> Aluguer - Modelo Conceptual .....	20
Figura 23	Relacionamento Veículo -> Stand - Modelo Conceptual .....	20
Figura 24	Relacionamento Stand -> Funcionario - Modelo Conceptual .....	21
Figura 25	Relacionamento Funcionário -> Aluguer - Modelo Conceptual .....	21
Figura 26	Caracterização dos atributos das Entidades: Cliente, Veículo e Aluguer .....	22
Figura 27	Caracterização dos atributos das Entidades: Funcionário e Stand .....	22
Figura 28	Modelo Conceptual Final .....	23
Figura 29	Conversão da Entidade "Cliente" - Modelo Lógico .....	25
Figura 30	Conversão da Entidade "Aluguer" - Modelo Lógico .....	25
Figura 31	Conversão da Entidade "Veiculo" - Modelo Lógico .....	26
Figura 32	Conversão da Entidade "Funcionario" - Modelo Lógico .....	27
Figura 33	Conversão da Entidade "Stand" - Modelo Lógico .....	28
Figura 34	Modelo Lógico Final .....	28
Figura 35	Expressão em Álgebra Relacional do RM03 .....	29
Figura 36	Ávore lógica da expressão do RM03 .....	30
Figura 37	Expressão em Álgebra Relacional do RM05 .....	30
Figura 38	Ávore lógica da expressão do RM05 .....	30
Figura 39	Expressão em Álgebra Relacional do RM06 .....	31
Figura 40	Ávore lógica da expressão do RM06 .....	31
Figura 41	Expressão em Álgebra Relacional do RM07 .....	31
Figura 42	Ávore lógica da expressão do RM07 .....	32
Figura 43	Tabela resultado da execução da Querie no RelaX - exemplo populado ilustrativo .....	32
Figura 44	Esquema Físico gerado no MySQL Workbench .....	33
Figura 45	Tamanho relativo a cada tabela (incluindo índices) .....	41

Figura 46	Insercao de dados no postgres para Teste .....	55
Figura 47	Migração de dados do postgres para o MySQL .....	55
Figura 48	Insercao de dados no postgres para Teste .....	61
Figura 49	Migração de dados do postgres para o MySQL .....	61
Figura 50	Diagrama de Gantt Planeado .....	64
Figura 51	Diagrama de Gantt Final (preenchido ao longo do desenvolvimento do projeto) .....	64



# 1. Definição do Sistema

## 1.1. Contextualização

Em pleno outono do ano de 1970, nasce a “*Belo Cars*”, uma empresa inovadora e confiável, especializada na gestão de alugueres de carros e veículos comerciais. No dia 26 de outubro de 1980, o seu fundador, Leonidas Moreau, com o crescimento contínuo da *Belo Cars*, quis ampliar a sua atividade e agora possui três *stands* estrategicamente localizados em Lisboa, Braga e Viseu. Cada *stand* foi projetado para atender diferentes perfis de clientes e necessidades regionais, proporcionando maior acessibilidade e conveniência. Movido pela paixão pelo setor automóvel e pela crescente necessidade de serviços de aluguer eficientes e acessíveis, Leonidas idealizou a *Belo Cars* para oferecer soluções personalizadas tanto para clientes individuais quanto para empresas de maior dimensão. Desde então, a empresa expandiu-se e consolidou-se, tornando-se uma referência no setor de aluguer de veículos. Atualmente, a CEO, Sofia Moreau, filha de Leonidas, lidera a empresa, que conta com uma frota diversificada composta por veículos comerciais e de passageiros, além de uma equipa de mais de 200 colaboradores dedicados a garantir um serviço eficiente e de qualidade. Os pagamentos são realizados de forma segura e transparente, tanto de forma física como por intermediação de um banco (seja por transferência bancária ou cartão multibanco), garantindo praticabilidade e confiabilidade para os clientes. Antes da entrega do veículo é feito um orçamento detalhado, e o pagamento total é cobrado no ato imediatamente antes da retirada do automóvel. Para garantir um serviço ágil e organizado, a empresa conta com um departamento interno especializado na gestão de frota e clientes, otimizando os processos de reserva, manutenção e entrega dos veículos. A *Belo Cars* destaca-se pela sua experiência no aluguer de veículos, oferecendo desde carros de passeio para viagens e turismo, até veículos comerciais para transporte de mercadorias. Com esta introdução recente de um sistema digital avançado de reservas e rastreio, a empresa reforça o seu compromisso com a inovação e a satisfação dos clientes, e no futuro próspero do negócio.

## 1.2. Fundamentação

Ao longo dos últimos 50 anos desde a fundação da *Belo Cars*, a empresa tem evoluído constantemente para otimizar as suas operações e garantir um serviço de aluguer eficiente e seguro. No entanto, grande parte da sua gestão de reservas, contratos e informações de clientes e frota era realizada manualmente e com registos físicos em papel, exigindo que os clientes se deslocassem presencialmente às suas instalações para efetuar reservas, assinar contratos. Esse método tradicional, além de consumir tempo e recursos, tornava os processos mais suscetíveis a erros administrativos, atrasos e dificuldades na organização dos dados, e no seu posterior tratamento e consulta. Com o crescimento da empresa e a expansão dos seus serviços, tornou-se evidente a necessidade de um sistema moderno e centralizado para armazenar, processar e organizar todas as operações de aluguer de forma mais rápida, segura e eficiente. Sem um Sistema de Base de Dados (SBD) adequado, a empresa enfrentava desafios como: a duplicação e difícil manutenção dos registos, dificuldades no rastreio/manutenção da totalidade da frota, desperdício de esforço manual(humano), inconsistências em registos de pagamentos, entre outros. Além disso, o modelo anterior de gestão limitava a experiência do cliente, que não conseguia aceder facilmente a informações sobre reservas, contratos ou histórico de alugueres sem a deslocação a uma das sucursais físicas. A implementação de um SBD relacional permitirá que a *Belo Cars* ofereça um serviço mais transparente e acessível, com a possibilidade de procuras e listagens em tempo real, reservas online e um controlo mais

eficiente sobre a frota de automóveis. Este novo sistema não só modernizará a infraestrutura tecnológica da empresa, como também garantirá uma maior segurança na gestão de pagamentos, contratos e dados sensíveis, reduzindo riscos de fraudes ou adulterações de dados indesejáveis e melhorando a experiência dos clientes e a produtividade da equipa administrativa, permitindo até uma futura expansão ainda maior do nome da empresa.

### 1.3. Objetivos

A implementação do Sistema de Base de Dados (SBD) pela *Belo Cars* é orientada por uma série de objetivos estratégicos, todos projetados para modernizar a gestão da empresa e aprimorar a experiência do cliente. Abaixo estão os principais objetivos:

- **Aumentar a segurança e a confiabilidade dos dados:**

Um dos principais objetivos da adoção de um Sistema de Base de Dados é garantir a segurança e a integridade das informações relacionadas à frota, clientes e transações. Com um sistema digital seguro, a empresa elimina riscos associados ao extravio de documentos físicos, erros administrativos e acessos não autorizados, assim como infortúnios físicos que possam acontecer aos dados em papel.

Numa perspetiva de Proteção de Dados, a centralização dos dados numa plataforma robusta reduz significativamente o risco de perda de informações, assegurando que apenas utilizadores autorizados possam aceder a contratos, reservas e informações financeiras sensíveis.

- **Melhorar a eficiência operacional e a gestão da frota:**

Com um SBD relacional, a *Belo Cars* poderá automatizar processos administrativos e melhorar a gestão da sua frota, permitindo um controlo mais preciso da disponibilidade dos veículos, do histórico de alugueres e pagamentos. Isso agiliza o atendimento aos clientes, reduzindo o tempo de espera e garantindo que os veículos estejam sempre disponíveis e em perfeitas condições para o uso.

- **Melhorar a experiência do cliente:**

A introdução de um sistema digital de reservas permitirá aos clientes consultar a disponibilidade, realizar reservas online, acessar contratos e efetuar pagamentos de forma mais rápida e segura. Essa modernização não só melhora a transparência e a praticidade do serviço, como também aumenta a satisfação e a fidelização dos clientes.

Além disso, a análise dos dados armazenados no SBD possibilita um conhecimento mais aprofundado sobre as preferências e hábitos dos clientes, permitindo à empresa oferecer promoções personalizadas e serviços mais personalizados às preferências de cada envolvido.

- **Organizar e otimizar o modelo de negócio:**

A implementação do SBD também visa padronizar e otimizar os processos operacionais da *Belo Cars*, garantindo um fluxo de trabalho mais eficiente e uma melhor comunicação interna entre todos os setores da empresa. Com uma Base de Dados centralizada, a empresa pode gerir reservas, pagamentos e manutenções de forma integrada, reduzindo falhas operacionais e facilitando a tomada de decisões estratégicas. Deste modo também reorganiza o fluxo de recursos humanos associados a este tipo de tarefas para outras que tragam mais benefícios ao futuro da empresa.

Além disso, o sistema permitirá uma análise detalhada do desempenho da empresa e auxílio em Relatórios, fornecendo dados essenciais para melhorar a alocação de recursos e impulsionar o crescimento sustentável da *Belo Cars* a longo prazo.

## 1.4. Viabilidade

A CEO, Sofia Moreau, reconhece a importância da modernização e da digitalização dos processos da *Belo Cars* para acompanhar o crescimento tecnológico da empresa e melhorar a experiência dos clientes. A implementação de um Sistema de Base de Dados (SBD) permitirá à empresa otimizar a gestão de alugueres e operações internas, garantindo diversos benefícios ao nível dos processos internos, tais como:

- **Segurança e confidencialidade reforçadas:** A digitalização dos dados elimina a necessidade de ficheiros físicos, diminuindo o risco de perda, roubo ou danos. Além disso, o controlo de acessos garantirá que apenas colaboradores autorizados possam visualizar e modificar informações sensíveis à empresa.
- **Proteção de dados pessoais:** O novo sistema impedirá acessos não autorizados a informações de clientes e contratos, cumprindo regulamentações de privacidade e aumentando a confiança dos clientes no serviço prestado.
- **Otimização da gestão da frota:** O sistema proporcionará um controlo detalhado sobre a disponibilidade especificações dos veículos, controlo de funcionários e clientes e histórico de alugueres, reduzindo falhas e aumentando a organização da empresa como um todo.
- **Automação de processos administrativos:** A reserva, contratação e pagamento serão simplificados e centralizados, reduzindo o tempo de espera dos clientes e minimizando erros administrativos.
- **Aumento da margem de lucro:** Com processos mais eficientes, espera-se um crescimento significativo na taxa de conversão de alugueres e uma redução de custos operacionais, impulsionando a rentabilidade da *Belo Cars*.
- **Competitividade no mercado:** A implementação do SBD fortalecerá a posição da *Belo Cars* no setor, proporcionando um serviço mais ágil e tecnologicamente avançado em relação à concorrência.

A implementação deste sistema será um marco na evolução da empresa, consolidando-a como uma referência em inovação no setor de aluguer de veículos.

## 1.5. Equipa de Trabalho

Para a implementação da Base de Dados para a *Belo Cars* ficou delineada uma equipa que consiste nos seguintes envolvidos:

- **Pessoal Externo** – Engenheiros Informáticos como Gonçalo Castro, Afonso Martins, Jorge Barbosa e Luís Felício são responsáveis por todo o desenvolvimento técnico desde a Contextualização e Modelação até à Implementação Física do SGB.
- **Pessoal Interno** – Sofia Moreau (CEO), Funcionários dos *Stand's*, Pessoal de Administração Interna, Apoio ao cliente, etc... Estas pessoas são a fonte principal de informação e requisitos, pois são os intervenientes e finais utilizadores do sistema que pretendemos desenvolver.

## 1.6. Recursos

Fora os Recursos Humanos listados no tópico anterior, denotam-se de seguida os recursos que seriam precisos para o bom progresso do projeto que temos em mãos:

- Computadores pessoais suficientemente capazes de executar o 'MySQL Workbench' e o 'MySQL Community Server' que serão as ferramentas mais desafiadoras a nível de computação.
- Garantir o acesso a ferramentas de software adequadas para a realização da Modelação e Implementação, tais como, 'BrModelo' e 'MySQL Workbench'.

- Servidores físicos ou em *cloud*, para alojar o sistema produzido. Se for em instalações locais, exige investimento em *racks*, climatização e UPS; ou como alternativa: serviços de *cloud computing* (AWS, Azure, Google Cloud).

Note-se que: Ainda que este projeto não inclua a vertente de *Deploy* do sistema, referenciada no último ponto, consideramos importante a referência a este tipo de recursos para uma utilização efetiva e bem sucedida num futuro próspero da Base de Dados da *Belo Cars*.

## 1.7. Plano de Execução

Para garantir uma implementação eficaz deste Sistema de Base de Dados, Sofia Moreau e a administração da *Belo Cars*, em parceria com uma equipa de engenheiros de *software* e especialistas em gestão de dados, desenvolveram um plano detalhado da execução do projeto. Esse plano segue um cronograma estruturado baseado num Diagrama de Gantt, que contempla todas as fases essenciais do projeto.

O projeto está estruturado em cinco grandes fases principais, cada uma com tarefas específicas, distribuídas no tempo entre 17 de fevereiro de 2025 e 31 de maio de 2025. A seguir, descreve-se o plano de execução com base no diagrama:

### 1. Definição do Sistema (De 17/02/2025 a 05/03/2025)

Objetivo: Estudar o contexto e justificar a necessidade do sistema.

Tarefas:

- Contextualização
- Fundamentação
- Objetivos
- Viabilidade
- Recursos
- Equipa de Trabalho
- Plano de Execução
- Revisão e Aprovação

### 2. Definição de Requisitos (De 06/03/2025 a 25/03/2025)

Objetivo: Definir os requisitos técnicos e funcionais do sistema e organizá-los.

Tarefas:

- Definição do Método
- Levantamento
- Análise
- Organização
- Validação

### 3. Modelação Conceptual (De 26/03/2025 a 07/04/2025)

Objetivo: Representar os dados e relações através de um modelo conceptual.

Tarefas:

- Identificação das Entidades
- Identificação dos Relacionamentos
- Definição de Atributos

- Elaboração do Esquema Conceptual
- Validação do Esquema

4. Modelação Lógica (De 08/04/2025 a 17/04/2025)

Objetivo: Traduzir o modelo conceptual para um modelo lógico relacional.

Tarefas:

- Derivação das Relações
- Elaboração do Esquema Lógico Inicial
- Validação do Esquema Final

5. Implementação Física (De 19/04/2025 a 31/05/2025)

Objetivo: Implementar fisicamente a base de dados e realizar operações de população e definição de queries.

Tarefas:

- Criação do Sistema de Base de Dados
- Criação do Esquema Físico
- Povoamento Inicial
- Execução de Queries
- Exemplos de execução de Transação de Venda

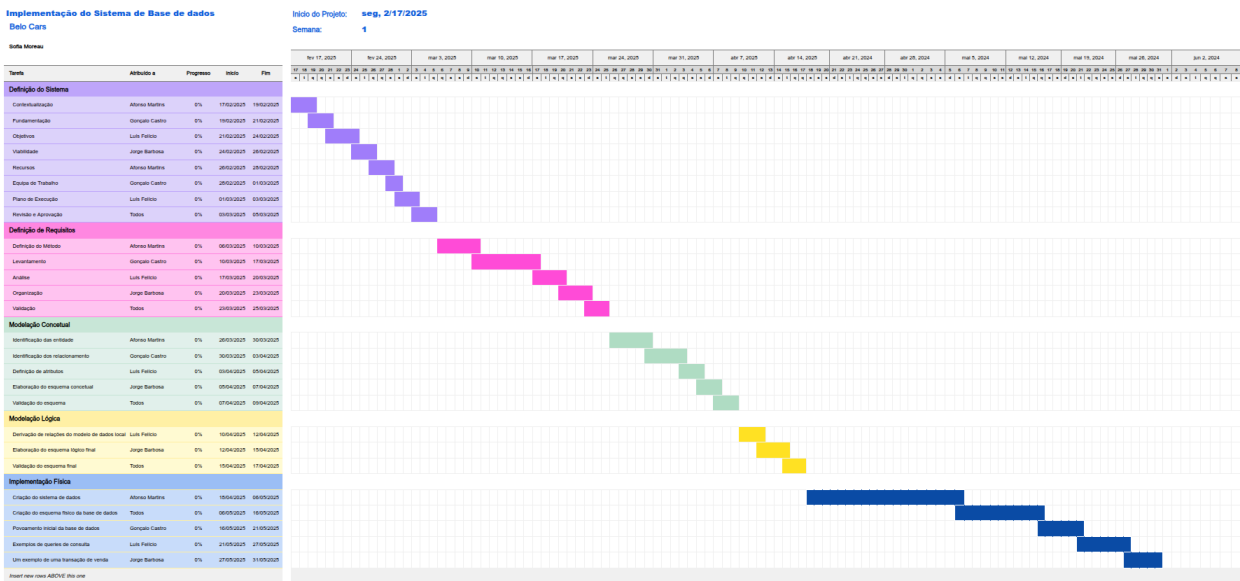


Figura 1: Diagrama de Gantt

1.8. Revisão e Aprovação

A equipa gestora da *Belo Cars*, juntamente com os especialistas em tecnologia, reviu a estrutura do novo sistema e validou a sua viabilidade. Durante reuniões estratégicas com os interessados, foram feitos ajustes ao plano de execução para garantir que a implementação ocorresse de forma eficiente e sem impactar negativamente as operações da empresa através do diagrama de Gantt elaborado. Com a aprovação final, o projeto foi oficialmente iniciado e sua implementação está em progresso.

## 1.9. Notas Adicionais

Dado que o sistema armazenará informações críticas de clientes e operações, serão adotadas medidas rigorosas de segurança, incluindo autenticação multi-fator, criptografia de dados e *backups* regulares. Estas precauções garantirão a integridade e confidencialidade das informações, prevenindo acessos indevidos e possíveis falhas técnicas no desenvolvimento dos trabalhos. Além disso, a utilização destas técnicas garantirá que mesmo em caso de intrusão, os dados permaneçam protegidos e inacessíveis para indivíduos não autorizados.

## 2. Levantamento e Análise de Requisitos

### 2.1. Definição do Método

No caso da *Belo Cars*, para proceder ao levantamento de requisitos e garantir uma implementação eficiente do Sistema de Base de Dados (SBD), foram definidos os seguintes métodos:

#### 1ª Reuniões:

- Reunião com a CEO, Sofia Moreau, e a administração da *Belo Cars* para discutir os objetivos da modernização e as necessidades específicas da empresa.
- Reuniões com gestores de frota e colaboradores dos três *stands* (Lisboa, Braga e Viseu) para compreender os desafios operacionais e as expectativas em relação ao novo sistema.
- Reuniões com os clientes para entender as suas preferências, dificuldades e necessidades no processo de aluguer de veículos.

De seguida fica um exemplo de uma das reuniões efetuadas no âmbito deste projeto:

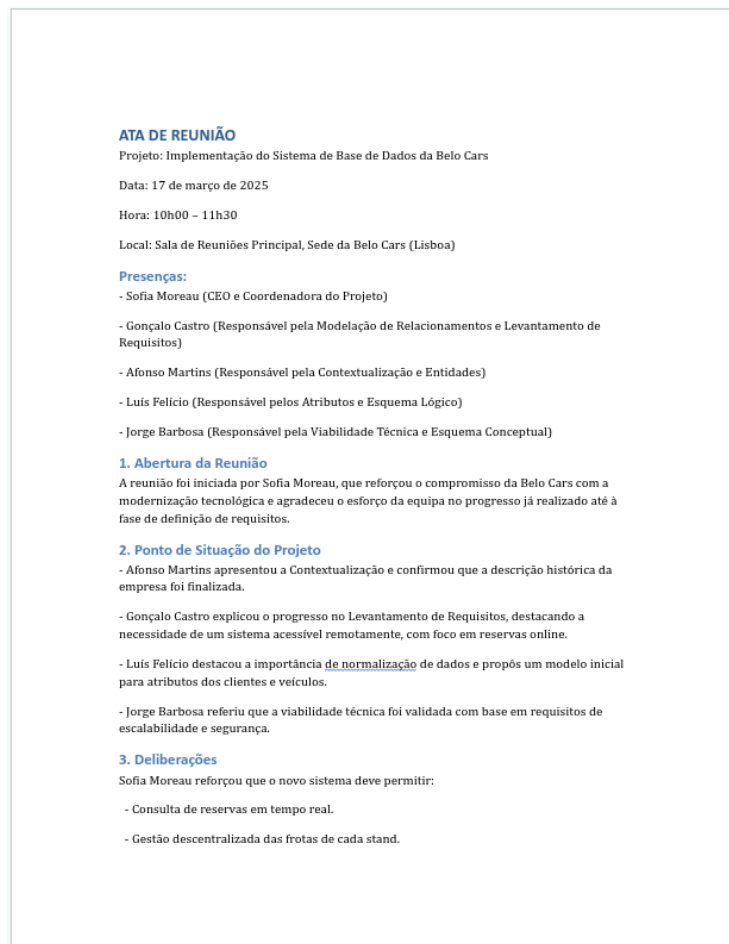


Figura 2: Primeira Página da Ata de uma das Reuniões com a Sofia Moreau

- Relatórios automáticos para análise de desempenho e previsões de utilização.		
Foi decidido que o modelo conceptual deve estar completo até 7 de abril de 2025, incluindo todos os relacionamentos e regras de negócio.		
A equipa de TI irá validar a infraestrutura atual e propor melhorias até ao final de março.		
<b>4. Ações a Desenvolver</b>		
Tarefa	Responsável	Prazo
Concluir levantamento funcional	Gonçalo Castro	25/03/2025
Apresentar rascunho do modelo ER	Jorge Barbosa	01/04/2025
Rever atributos e normalização	Luís Felício	02/04/2025
Validar compatibilidade técnica	Afonso Martins	28/03/2025
<b>5. Encerramento</b>		
A reunião foi encerrada às 11h30 com palavras de motivação da CEO:		
"A Belo Cars nasceu de uma ideia simples: mobilidade com confiança. Este projeto é mais do que uma atualização — é a base do nosso futuro digital." — Sofia Moreau		
<b>Próxima Reunião:</b>		
Data Provisória: 27 de março de 2025		
Objetivo: Validação final do modelo conceptual e integração de requisitos técnicos.		

Figura 3: Segunda Página da Ata de uma das Reuniões com a Sofia Moreau

## 2º Análise da Documentação:

- Revisão dos processos administrativos atuais relacionados à gestão de reservas, contratos, pagamentos e manutenção da frota.
- Análise dos documentos utilizados na empresa, como contratos de aluguer, relatórios de manutenção, faturas e comprovativos de pagamento.
- Avaliação das atuais políticas de segurança e proteção de dados existentes para garantir conformidade com regulamentações de privacidade.

## 3º Observação dos Processos:

- Acompanhamento da rotina de trabalho nos escritórios da Belo Cars para identificar ineficiências e oportunidades de melhoria na gestão de alugueres.
- Observação do fluxo de atendimento ao cliente e das interações entre diferentes departamentos da empresa.
- Identificação de processos que podem ser automatizados para aumentar a eficiência e reduzir o “erro humano”.

## 4º Inquéritos:

- Elaboração de inquéritos para clientes avaliarem a sua satisfação com os serviços atuais e indicarem funcionalidades desejáveis para o novo sistema.
- Aplicação de questionários aos colaboradores para compreender dificuldades operacionais e recolher sugestões de melhorias.
- Identificação das necessidades específicas de diferentes tipos de clientes, como turistas, empresas e transportadoras.



**Questionário Interno — Recolha de Opiniões e Sugestões (Funcionários *Belo Cars*)**

Este formulário tem como objetivo recolher a opinião dos colaboradores sobre o funcionamento da empresa e obter sugestões para melhorar os processos internos, a comunicação e a experiência de trabalho.

[Seguinte](#) [Limpar formulário](#)

Nunca envie palavras-passe através dos Google Forms.

Este conteúdo não foi criado nem aprovado pela Google. - [Termos de Utilização](#) - [Política de privacidade](#)

Este formulário parece suspeito? [Relatório](#)

Google Formulários

Figura 4: Primeira Página do Questionário que serviu de inquérito aos funcionários da Belo Cars

**Questionário Interno — Recolha de Opiniões e Sugestões (Funcionários *Belo Cars*)**

\* Indica uma pergunta obrigatória

**Seção 1: Dados Gerais (opcionais)**

Nome  
A sua resposta

Departamento em que trabalha \*  
Selecione

Função / Cargo atual \*  
A sua resposta

Tempo na Empresa \*  
☐ Menos de 1 ano  
☐ 1 a 3 anos  
☐ 4 a 6 anos  
☐ Mais de 6 anos

[Anterior](#) [Seguinte](#) [Limpar formulário](#)

Figura 5: Segunda Página do Questionário que serviu de inquérito aos funcionários da Belo Cars

**Questionário Interno — Recolha de Opiniões e Sugestões (Funcionários *Belo Cars*)**

\* Indica uma pergunta obrigatória

**Avaliação dos Processos Atuais**

Como avalia os atuais processos de reserva e gestão de veículos? \*

1 2 3 4 5  
Muito Ineficiente ☐ ☐ ☐ ☐ ☐ Muito Eficiente

Quais são os maiores desafios que enfrenta no dia a dia do seu trabalho? \*

A sua resposta

Será que os recursos disponíveis são suficientes para executar bem o seu trabalho? \*

☐ Sim  
☐ Em parte  
☐ Não  
☐ Não sei / Prefiro não responder

[Anterior](#) [Seguinte](#) [Limpar formulário](#)

Figura 6: Terceira Página do Questionário que serviu de inquérito aos funcionários da Belo Cars

**Questionário Interno — Recolha de Opiniões e Sugestões (Funcionários *Belo Cars*)**

\* Indica uma pergunta obrigatória

**Sugestões e Inovação**

Que melhorias gostaria de ver implementadas no novo sistema de gestão (base de dados, reservas, etc...)? \*

A sua resposta

Há algum processo específico que considera desnecessário ou que poderia ser simplificado? \*

A sua resposta

Tem sugestões para melhorar a comunicação entre departamentos?

A sua resposta

Outras observações ou sugestões gerais

A sua resposta

[Anterior](#) [Seguinte](#) [Limpar formulário](#)

Figura 7: Quarta Página do Questionário que serviu de inquérito aos funcionários da Belo Cars

**Questionário Interno — Recolha de Opiniões e Sugestões (Funcionários *Belo Cars*)**

\* Indica uma pergunta obrigatória

**Seção Final**

Gostaria de ser contactado para esclarecer melhor as suas respostas ou aprofundar as suas ideias? \*

☐ Sim  
☐ Não

Se respondeu 'Sim' acima, indique o seu email de contacto

A sua resposta

[Anterior](#) [Enviar](#) [Limpar formulário](#)

Figura 8: Quinta Página do Questionário que serviu de inquérito aos funcionários da Belo Cars

### 5º Análise de Soluções Existentes:

- Pesquisa e avaliação de sistemas de base de dados utilizados por outras empresas do setor de aluguer de veículos.
- Identificação das melhores práticas e funcionalidades essenciais para o SBD da Belo Cars.
- Análise de viabilidade técnica e económica para garantir que a solução escolhida seja eficiente e escalável.

## 2.2. Levantamento

No processo de desenvolvimento do sistema de base de dados da *Belo Cars*, foi realizado o levantamento de requisitos, seguido pela organização e validação dos mesmos por áreas de trabalho.

Nesta fase crucial do desenvolvimento, encontram-se seis áreas distintas que desempenham papéis fundamentais no funcionamento da empresa:

- **Clientes:** Esta área abrange todos os dados relacionados aos clientes da *Belo Cars*, incluindo informações de contacto, nome, data de nascimento e o seu NIF para efeitos de faturação.
- **Veículos:** Aqui, reunimos informações sobre a frota de veículos da empresa. Isso inclui detalhes como modelo, ano de fabricação, disponibilidade para aluguer, tipo (Veículo de Passageiros ou de Mercado-rias), entre outras especificações.
- **Alugueres:** Esta área é essencial para o acompanhamento e gestão de todas os alugueres realizados. Os requisitos desta categoria incluem o registo de datas de reserva, veículos e clientes associados, funcionários responsáveis e termos do contrato.
- **Financeiro:** Esta área abrange todos os aspetos financeiros, tais como pagamentos dos clientes, valores diários de aluguer e outros relatórios financeiros.
- **Recursos Humanos:** São consideradas todas as informações relativas aos colaboradores da *Belo Cars*, incluindo contratação, atribuição de tarefas, gestão de horários, em relação a todos os Funcionários de todos os estabelecimentos.

No âmbito de um projeto de um sistema de base de dados, as diversas áreas da empresa podem ser consideradas como diferentes vistas de utilização do sistema. Cada área representa uma vista única, através da qual os utilizadores interagem com o sistema e exploram a informação pertinente às suas funções específicas.

A equipa decidiu iniciar o processo de levantamento de requisitos pela área dos Clientes, dada a importância de compreender as necessidades e preferências dos clientes da empresa, estes que são o remetente final do serviço e o ponto fulcral no sucesso de uma empresa. Posteriormente, abordar-se-ão as outras áreas, garantindo uma visão abrangente de todas as partes envolvidas no funcionamento da empresa.

Como planeado, foram realizadas várias reuniões com Sofia Moreau e a equipa de desenvolvimento. Durante essas reuniões, foram discutidos detalhadamente os requisitos específicos de cada área, bem como as necessidades operacionais da *Belo Cars*.

Além das reuniões, os analistas realizaram uma análise metódica da documentação existente na empresa, com especial atenção aos registos dos clientes, detalhes de alugueres anteriores, disponibilidade da frota. Foram guardadas cópias de diversos documentos relevantes e associadas a cada requisito devidamente identificado, garantindo uma referência clara para futuras etapas do processo de desenvolvimento.

Para complementar as análises realizadas, os analistas também observaram de perto as operações diárias da empresa, tais como a gestão de reservas, interação com clientes e manutenção da frota. Com isto, obtiveram uma compreensão mais detalhada dos fluxos de trabalho da *Belo Cars* e identificaram possíveis melhorias.

Foram também elaborados pequenos inquéritos distribuídos a alguns clientes, colaboradores e funcionários da empresa, solicitando a sua opinião sobre a qualidade de serviço prestado, bem como sugestões de melhoria.

À medida que os trabalhos de levantamento de requisitos avançavam, os analistas anotavam meticolosamente todas as informações reunidas num documento de requisitos. Assim, garantiam que todas as necessidades e requisitos fossem adequadamente considerados e integrados no sistema final.

#### **2.2.1. Vistas de Utilização**

Inicialmente, é essencial haver uma abordagem para cada uma destas vistas de utilização de forma independente e autónoma. Cada área deve ser cuidadosamente examinada e desenvolvida em separado. Este processo permite uma compreensão detalhada das necessidades e requisitos específicos de cada área, garantindo que todos os elementos essenciais sejam identificados e integrados no sistema de base de dados de forma eficaz.

Posteriormente, à medida que as vistas são desenvolvidas e refinadas individualmente, é necessário conciliar tudo. Este processo envolve a integração harmoniosa das diferentes vistas de utilização, de modo que seja criado um esquema global coeso e abrangente para a base de dados.

Através deste processo, é possível configurar o sistema de forma a ter em conta as diversas perspetivas de utilização dos diferentes utilizadores. Ao utilizar um único esquema, o sistema é capaz de fornecer uma experiência consistente e integrada, garantindo que todas as áreas da agência tenham acesso à informação relevante de forma eficiente e eficaz.

#### **2.2.2. O Documento de Requisitos**

Os analistas registaram, para cada requisito recolhido, no documento de requisitos da seguinte forma:

- a data do seu levantamento;
- texto do requisito, fornecendo o máximo de detalhes possível sobre as informações relacionadas com os dados envolvidos;
- área de aplicação - a vista de utilização;
- quem forneceu o requisito;
- quem realizou o levantamento.

**Documento de Recolha de Requisitos**

Desenvolvimento de um Sistema de Base de Dados

Março de 2025

**Levantamento Geral:**

Levantamento de Requisitos					
Nº	Data	Descrição	Área	Fonte	Analista
1	4 de março	Cada veículo contém um identificador único(matrícula), marca, modelo, ano de fabrico, preço diário, o consumo, o nº de passageiros que transporta, o tipo de combustível, o tipo (Mercadoria ou Passageiros) e estado atual (se está disponível ou não)	Veículos	Sofia Moreau	Gonçalo Castro
2	4 de março	Se um Veículo for do tipo mercadoria deverá ainda possuir a tara (peso quando vazio) e a carga máxima	Veículos		Afonso Martins
3	4 de março	Cada cliente denota um identificador único, nome, data de nascimento, contacto (email e telemóvel) e nº de contribuinte (NIF)	Clientes	Sofia Moreau	Gonçalo Castro
4	4 de março	Cada Aluguer possui um identificador único (número sequencial), data de início, data de fim, valor da transação e o método de pagamento	Aluguer	Sofia Moreau	Luís Felício
5	4 de março	Cada funcionário possui um identificador único, nome, contactos (email e telefone), departamento e a sua função no departamento	Funcionários	Sofia Moreau	Jorge Barbosa
6	4 de março	Os dados acerca dos carros podem ser consultados por qualquer pessoa	Veículos	Sofia Moreau	Gonçalo Castro
7	4 de março	Os dados acerca dos clientes podem ser acedidos pelo próprio ou por um Funcionário mediante a inserção do identificador	Clientes	Sofia Moreau	Afonso Martins
8	4 de março	Os dados dos funcionários podem ser acedidos pelo próprio ou por um superior (admin)	Funcionários	Sofia Moreau	Luís Felício
9	9 de março	Os veículos podem ser filtrados por qualquer uma das especificações: marca, modelo, ano de fabrico, preço diário, o consumo, o nº de passageiros, o tipo de combustível, o tipo do veículo e disponibilidade	Veículos	Sofia Moreau	Jorge Barbosa
10	9 de março	Os funcionários também possuem autorização para editar todos os alugueres mediante a introdução do ID	Funcionários / Alugueres	Sofia Moreau	Gonçalo Castro

Figura 9: Primeira Página da Tabela de Requisitos Gerais

11	9 de março	Os alugueres podem ser filtrados por: intervalo de tempo (dado uma data de início e data de fim), intervalo de valores de pagamentos e por método de pagamento.	Alugueres	Sofia Moreau	Luís Felício
12	9 de março	Cada Aluguer tem de estar associado a um e um só cliente, mas um cliente pode ter 0 ou vários alugueres	Alugueres / Clientes	Sofia Moreau	Jorge Barbosa
13	10 de março	Cada Aluguer possui um e um só veículo, mas um veículo pode estar em 0 ou mais alugueres.	Alugueres / Veículos	Sofia Moreau	Gonçalo Castro
14	10 de março	Cada aluguer tem de ter associado um funcionário, um funcionário pode ter vários alugueres ou não possuir qualquer aluguer associado	Alugueres / Funcionários	Sofia Moreau	Afonso Martins
15	10 de março	Cada Stand inclui um identificador único, uma localização (que possui endereço e código postal) e contactos (email e telefone)	Stands	Sofia Moreau	Luís Felício
16	10 de março	Um Stand tem de ter um ou mais veículos a ele associados, mas um veículo só pode pertencer a um stand	Stands / Veículos	Sofia Moreau	Jorge Barbosa
17	25 de março	Os carros podem ser transferidos para outro stand pelos funcionários	Veículos / Funcionários	Sofia Moreau	Gonçalo Castro
18	25 de março	Tem de ser possível a listagem de todos os alugueres efetuados por um cliente mediante a introdução do seu id (incluindo o id, a marca e o modelo do veículo associado) e, opcionalmente, um intervalo de datas limite	Alugueres / Clientes	Sofia Moreau	Afonso Martins
19	25 de março	Listar os alugueres supervisionados por um dado funcionário, dado o id do funcionário e, opcionalmente, um intervalo de datas	Alugueres / Funcionários	Sofia Moreau	Luís Felício
20	25 de março	Consulta de todos os veículos de um dado stand que estão disponíveis para alugar	Veículos / Stands	Sofia Moreau	Jorge Barbosa
21	25 de março	Calcular o valor da receita num dado dia, associado a todos as transações efetuadas nesse dia	Alugueres	Sofia Moreau	Afonso Martins
22	25 de março	Listar por ordem decrescente os funcionários que mais efetuaram alugueres para monitorizar o desempenho	Funcionários / Alugueres	Sofia Moreau	Gonçalo Castro
23	25 de março	Cada stand pode empregar um ou mais funcionários. Cada funcionário está associado apenas e unicamente a um stand	Stand / Funcionários	Sofia Moreau	Afonso Martins
24	3 de maio	Desenvolvimento de um sistema de migração de dados, sendo que as 3 localizações possuem formatos diferentes (JSON, CSV e Postgres)	Migração	Sofia Moreau	Luís Felício

Figura 10: Segunda Página da Tabela de Requisitos Gerais

## 2.3. Análise

Após o levantamento de requisitos é necessário avaliar a sua viabilidade e fazer uma revisão geral minuciosa de forma a garantir a sua completude e correção, verificando a existência de erros, inconsistências, redundância ou ambiguidades, assegurando assim, que o projeto avança com uma base sólida e coesa.

Esta revisão é essencial para assegurar que o sistema de base de dados tenha uma base sólida e coesa, proporcionando uma estrutura robusta para o desenvolvimento da mesma.

## 2.4. Organização

Os analistas iniciaram o processo organizando os requisitos levantados em três categorias principais para o futuro sistema, nomeadamente:

- **Descrição:** abrange requisitos relacionados com a estruturação dos dados na base de dados, como definição de tabelas, atributos, domínios e restrições;
- **Manipulação:** contém os requisitos referentes à interação com os dados, incluindo tudo o que seja povoamento ou exploração de dados, desde de simples *queries* até procedimentos, funções ou gatilhos;
- **Controlo:** trata da gestão dos utilizadores e das suas permissões no sistema de base de dados.

Para cada uma das categorias foi criada uma tabela específica:

- Documento de Requisitos de Descrição:

### Organização e Categorização:

Requisitos de Descrição						
Nº		Data	Descrição	Área	Fonte	Revisor
RD01	1	4 de março	Cada veículo contém um identificador único(matrícula), marca, modelo, ano de fabrico, preço diário, o consumo, o nº de passageiros que transporta, o tipo de combustível, o tipo (Mercadoria ou Passageiros) e estado atual (se está disponível ou não)	Veículos	Sofia Moreau	Gonçalo Castro
RD02	2	4 de março	Se um Veículo for do tipo mercadoria deverá ainda possuir a tara (peso quando vazio) e a carga máxima	Veículos		Afonso Martins
RD03	3	4 de março	Cada cliente denota um identificador único, nome, data de nascimento, contacto (email e telemóvel) e nº de contribuinte (NIF)	Cientes	Sofia Moreau	Gonçalo Castro
RD04	4	4 de março	Cada Aluguer possui um identificador único (número sequencial), data de início, data de fim, valor da transação e o método de pagamento	Aluguer	Sofia Moreau	Luís Felício
RD05	5	4 de março	Cada funcionário possui um identificador único, nome, contactos (email e telefone), departamento e a sua função no departamento.	Funcionários	Sofia Moreau	Jorge Barbosa
RD06	15	10 março	Cada Stand inclui um identificador único, uma localização (que possui endereço e código postal) e contactos (email e telefone)	Stands	Sofia Moreau	Luís Felício

Figura 11: Tabela de Requisitos de Descrição

- Documento de Requisitos de Manipulação:

Requisitos de Manipulação						
Nº	Data	Descrição	Área	Fonte	Revisor	
RM01	9	9 de março	Os veículos podem ser filtrados por qualquer uma das especificações: marca, modelo, ano de fabrico, preço diário, o consumo, o nº de passageiros, o tipo de combustível, o tipo do veículo e disponibilidade	Veículos	Sofia Moreau	Jorge Barbosa
RM02	11	9 de março	Os <u>alugueres</u> podem ser filtrados por: intervalo de tempo (dado uma data de início e data de fim), intervalo de valores de pagamentos e por método de pagamento.	Alugueres	Sofia Moreau	Luís Felício
RM03	18	25 março	Tem de ser possível a listagem de todos os alugueres efetuados por um cliente mediante a introdução do seu id (incluindo o id, a marca e o modelo do veículo associado) e, opcionalmente, um intervalo de datas limite	Alugueres / Clientes	Sofia Moreau	Afonso Martins
RM04	19	25 março	Listar os alugueres supervisionados por um dado funcionário, dado o id do funcionário e, opcionalmente, um intervalo de datas	Alugueres / Funcionários	Sofia Moreau	Luís Felício
RM05	20	25 março	Consulta de todos os veículos de um dado stand que estão disponíveis para alugar	Veículos / Stands	Sofia Moreau	Jorge Barbosa
RM06	21	25 março	Calcular o valor da receita num dado dia, associado a todas as transações efetuadas nesse mesmo dia	Alugueres	Sofia Moreau	Afonso Martins
RM07	22	25 março	Listar por ordem decrescente os funcionários que mais efetuaram alugueres para monitorizar o desempenho	Funcionários / Alugueres	Sofia Moreau	Gonçalo Castro
RM08	24	3 de maio	Desenvolvimento de um sistema de migração de dados tendo em conta que as 3 localizações possuem formatos diferentes (JSON, CSV e Postgres)	Migração	Sofia Moreau	Luís Felício

Figura 12: Tabela de Requisitos de Manipulação

- Documento de Requisitos de Controlo:

Requisitos de Controlo					
Nº	Data	Descrição	Área	Fonte	Revisor
RC01	6 4 de março	Os dados acerca dos carros podem ser consultados por qualquer pessoa (Convidado)	Veículos	Sofia Moreau	Gonçalo Castro
RC02	7 4 de março	Os dados acerca dos clientes podem ser acedidos por um Funcionário mediante a inserção do identificador	Clientes	Sofia Moreau	Afonso Martins
RC03	8 4 de março	Os dados dos funcionários podem ser acedidos pelo próprio ou por um superior (admin)	Funcionários	Sofia Moreau	Luís Felício
RC04	10 9 de março	Os funcionários também possuem autorização para editar todos os alugueres mediante a introdução do ID	Funcionários / Alugueres	Sofia Moreau	Gonçalo Castro
RC05	12 9 de março	Cada Aluguer tem de estar associado a um e um só cliente, mas um cliente pode ter 0 ou vários alugueres	Alugueres / Clientes	Sofia Moreau	Jorge Barbosa
RC06	13 10 março	Cada Aluguer possui um e um só veículo, mas um veículo pode estar em 0 ou mais alugueres.	Alugueres / Veículos	Sofia Moreau	Gonçalo Castro
RC07	14 10 março	Cada aluguer tem de ter associado um funcionário, um funcionário pode ter vários alugueres ou não possuir qualquer aluguer associado	Alugueres / Funcionários	Sofia Moreau	Afonso Martins
RC08	16 10 março	Um Stand tem de ter um ou mais veículos a ele associados, mas um veículo só pode pertencer a um stand	Stand / Veículos	Sofia Moreau	Jorge Barbosa
RC09	17 25 março	Os carros podem ser transferidos para outro stand pelos funcionários	Veículos / Funcionários	Sofia Moreau	Gonçalo Castro
RC10	23 25 de março	Cada stand pode empregar um ou mais funcionários. Cada funcionário está associado apenas e unicamente a um stand	Stand / Funcionários	Sofia Moreau	Afonso Martins

Figura 13: Tabela de Requisitos de Controlo

Durante o processo de análise e organização, os analistas trabalharam em cada documento, utilizando o *Google Docs* para registar e organizar os requisitos de forma eficiente. Essa abordagem permitiu uma melhor gestão e acompanhamento de cada tipo de requisito, facilitando o processo de revisão, além de garantir que todas as necessidades fossem adequadamente documentadas.

## 2.5. Validação

Após uma revisão detalhada, a equipa de analistas conduziu uma reunião abrangente com todos os intervenientes do projeto para validar os requisitos levantados. Cada requisito foi minuciosamente examinado e discutido, permitindo a identificação e correção de quaisquer detalhe que pudesse surgir. No final, todos os intervenientes aprovaram os requisitos, assegurando a sua precisão e alinhamento com as necessidades da *Belo Cars*.

## 3. Modelação Conceptual

### 3.1. Apresentação da Abordagem de Modelação Realizada

A elaboração do esquema conceptual para o sistema de base de dados da *Belo Cars* é uma etapa crucial no processo de desenvolvimento, pois esta define a estrutura do sistema de base de dados, incluindo as entidades principais, os seus relacionamentos e atributos essenciais.

O processo de modelação conceptual adotado para desenvolver o modelo foi desenvolvido utilizando a **notação de Chen (1974, 2002)** combinada com elementos da notação **Min-Max de Abrial (1974)**, adaptados à ferramenta **BR Modelo**, esta oferece uma representação precisa e visualmente intuitiva das entidades, relacionamentos e atributos do sistema.

### 3.2. Identificação e Caracterização das Entidades

Designação	Descrição	Sinónimo	Ocorrência
Cliente	Indivíduos ou entidades que alugam veículos. Possui um identificador único (ID).	Consumidor, Locatário	Cada cliente identificado pelo seu ID.
Aluguer	Registo de um aluguer efetuado por um cliente, com valor e datas.	Locação, Arrendamento	Cada aluguer associado a um cliente e veículo.
Veículo	Veículo disponível para aluguer, com características como matrícula e tipo.	Automóvel, Carro	Cada veículo identificado pela sua matrícula.
Stand	Local físico onde os veículos estão disponíveis e funcionários trabalham.	Filial, Loja	Cada stand identificado pelo seu ID.
Funcionário	Pessoa que trabalha na empresa de aluguer, com função e departamento.	Colaborador, Empregado	Cada funcionário identificado pelo seu ID.

Figura 14: Caracterização geral das entidades

#### 3.2.1. Cliente

A entidade **Cliente** representa um indivíduo que contrata os serviços da *Belo Cars* para o aluguer de veículos. Como elemento essencial do negócio, é fundamental manter registos precisos de todos os clientes da empresa.

Segundo o requisito **RD03**, um cliente é composto pelos seguintes atributos:

- **Id** – identificador único do cliente;

- **Nome** – nome completo do cliente;
- **Contacto** – número de telefone do cliente e formas de contacto adicionais:
  - **Telemóvel** - número de telemóvel para contacto direto
  - **Email** - endereço de email principal para comunicações
- **Data de Nascimento** - data de nascimento do cliente
- **NIF** - Número de Identificação Fiscal do cliente

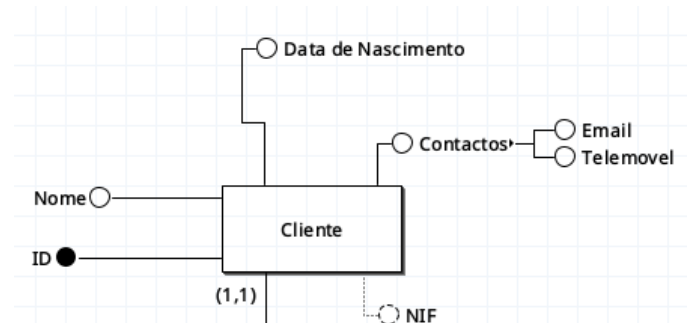


Figura 15: Entidade Cliente - Modelo Conceptual

### 3.2.2. Aluguer

A entidade **Aluguer** representa o registo de uma solicitação de aluguer de veículo feita por um cliente na *Belo Cars*. Cada aluguer contém informações essenciais para garantir um processo eficiente e organizado.

Segundo o requisito **RD04**, um aluguer é composto pelos seguintes atributos:

- **ID** – identificador único do Aluguer;
- **Data de Início** – data prevista para o início do aluguer;
- **Data de Fim** – data prevista para a devolução do veículo;
- **Valor** – custo total do aluguer;
- **Método de Pagamento** – forma de pagamento escolhida pelo cliente.

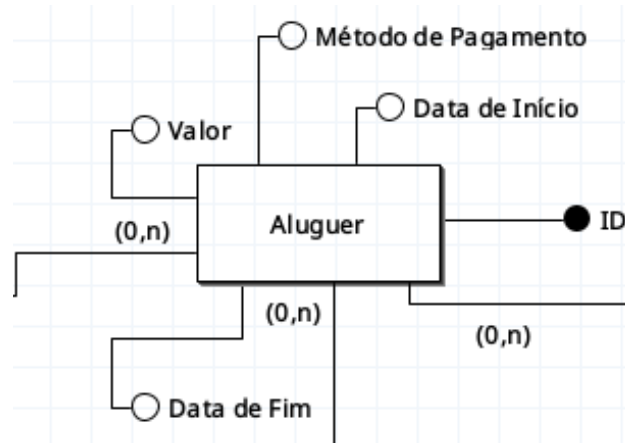


Figura 16: Entidade Aluguer - Modelo Conceptual

Poderá ser um Aluguer válido o seguinte: com o identificador numérico **“342”** foi feito com início a **12 de março de 2025** e término a **19 de março de 2025**, com um valor total de **350€** e método de pagamento por **cartão de débito**.

### 3.2.3. Veículo

A entidade **Veículo** representa um automóvel para aluguer na *Belo Cars*. Cada veículo possui informações detalhadas para garantir uma gestão eficiente da frota e um processo de aluguer organizado.

Segundo os requisitos **RD01** e **RD02**, cada Veículo registado no sistema possui os seguintes atributos:

- **Matrícula** – identificador único do veículo;



- **Marca** – fabricante do veículo;
- **Modelo** – modelo específico do veículo;
- **Ano** – ano de fabricação do veículo;
- **Tipo** – categoria do veículo;
- **Preço Diário** – valor cobrado por dia de aluguer;
- **Nº de Passageiros** – capacidade máxima de ocupantes;
- **Disponível** – indica se o veículo está disponível para aluguer;
- **Tara** – peso do veículo sem carga;
- **Carga Máxima** – peso máximo que o veículo pode transportar;
- **Consumo** – consumo médio de combustível;
- **Transmissão** – tipo de caixa de velocidades (manual ou automática);
- **Combustível** – tipo de combustível consumido pelo veículo.

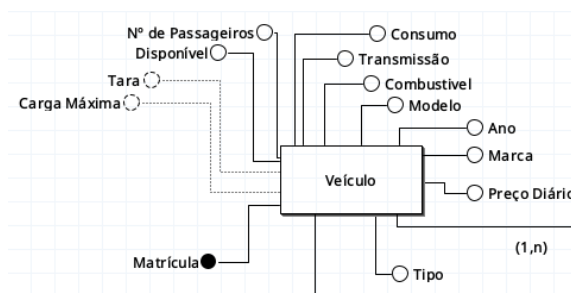


Figura 17: Entidade Veículo - Modelo Conceptual

Segue-se um exemplo hipotético que poderá representar um Veículo pertencente à *Belo Cars* na Base de Dados: com a matrícula “AA-22-BB” corresponde a um **Mercedes-Benz** (Marca) modelo **Classe A** (Modelo), fabricado no ano de **2022** (Ano). É um veículo do tipo **Compacto** com preço diário de **85€**. Tem capacidade para **5 passageiros**, como é um carro de passageiros não possui informações sobre peso máximo e tara. O consumo médio é de **5,2 L/100km**, possui transmissão **automática** e utiliza **gasolina** como combustível. Atualmente, encontra-se **indisponível** para aluguer pois já se encontra reservado.

### 3.2.4. Funcionário

A entidade **Funcionário** representa o registo de um funcionário que trabalha na *Belo Cars*. Cada funcionário contém informações essenciais para a gestão de recursos humanos e organização da empresa.

Segundo o requisito **RD05**, um funcionário é composto pelos seguintes atributos:

- **ID** – identificador único do funcionário;
- **Nome** – nome completo do funcionário;
- **Departamento** – setor da empresa onde o funcionário exerce suas funções;
- **Função** – cargo ou papel desempenhado pelo funcionário;
- **Contacto** – informações para comunicação com o funcionário:
  - **Email** – endereço de correio eletrónico profissional;
  - **Telefone** – número de telefone para contacto.

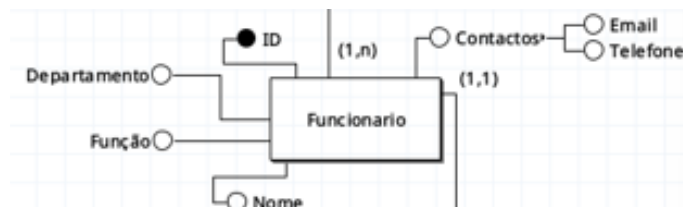


Figura 18: Entidade Funcionário - Modelo Conceptual

Segue-se um exemplo hipotético de um funcionário registado no sistema a ser implementado. O funcionário com o identificador numérico “125” chama-se **João Silva**, trabalha no departamento de **Vendas** e exerce a função de **Consultor Comercial**. Pode ser contactado através do email **joao.silva@belocars.pt** ou pelo telefone **+351 923 456 789**.

### 3.2.5. Stand

A entidade **Stand** representa a estrutura onde os funcionários exercem as suas funções e também acolhe os veículos para aluguer.

Segundo os requisitos **RD06** :

- **ID** – identificador único do Stand;
- **Localização** – cidade onde o Stand está situado;
- **Endereço** – morada completa do Stand;
- **Código Postal** – código postal correspondente ao endereço;
- **Contacto** – informações para comunicação com o Stand:
  - **Email** – endereço de correio eletrónico do Stand;
  - **Telefone** – número de telefone para contacto.

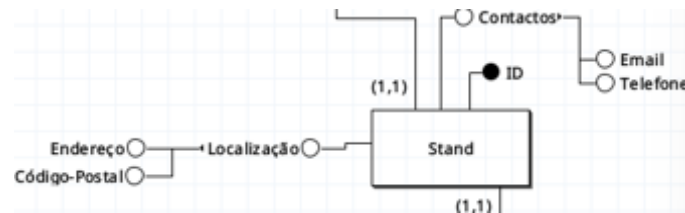


Figura 19: Entidade Stand - Modelo Conceptual

Segue-se um exemplo hipotético de um Stand no sistema a ser implementado: com o identificador numérico “1” possui a sua localização em **Braga**, com endereço na **Avenida da Liberdade, 123** e código postal **4710-249**. Para contacto, está disponível o email **braga@belocars.pt** e o telefone **+351 253 123 456**.

## 3.3. Identificação e Caracterização dos Relacionamentos

Entidade	Relacionamento	Cardinalidade	Participação	Entidade
Cliente	Efetuou	1:N	T:P	Aluguer
Veículo	Associado	1:N	T:P	Aluguer
Funcionário	Supervisionou	1:N	T:P	Aluguer
Veículo	Pertence	1:N	T:T	Stand
Stand	Emprega	N:1	T:T	Funcionário

Figura 20: Caracterização geral dos relacionamentos

### 3.3.1. Cliente → Aluguer : 1 → n

Segundo os requisitos do sistema **RC06**, cada Cliente pode solicitar e efetuar vários alugueres de veículos, sendo que este pode ter zero ou mais alugueres associados. Cada Aluguer tem a ele apenas e unicamente um Cliente associado. Isto permite visualizar todo o histórico de alugueres de cada cliente individualmente.

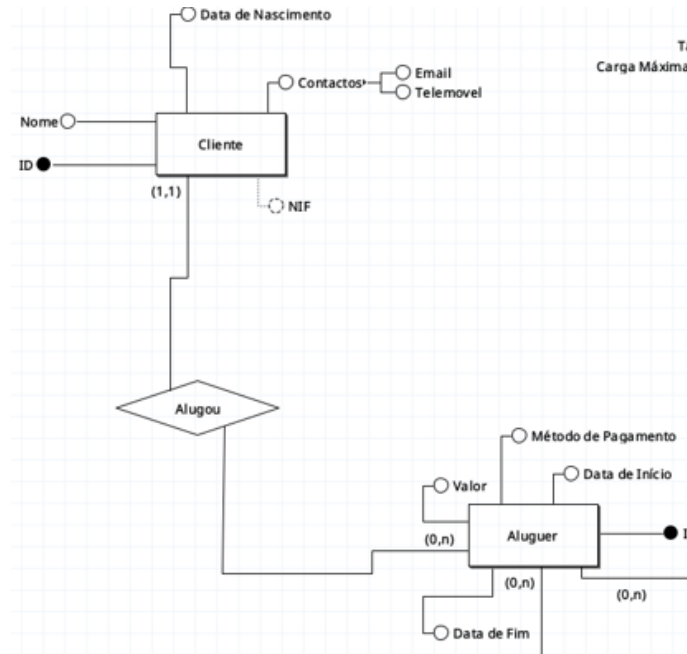


Figura 21: Relacionamento Cliente -> Aluguer - Modelo Conceptual

### 3.3.2. Veículo → Aluguer : 1 → n

Segundo o **RC07**, cada veículo da frota pode ser reservado múltiplas vezes ao longo do tempo, sendo que um Veículo pode ter zero ou mais alugueres associados. Cada Aluguer tem que ter pelo menos um Veículo atribuído. Isto possibilita o controlo da disponibilidade e ocupação de cada veículo da frota.

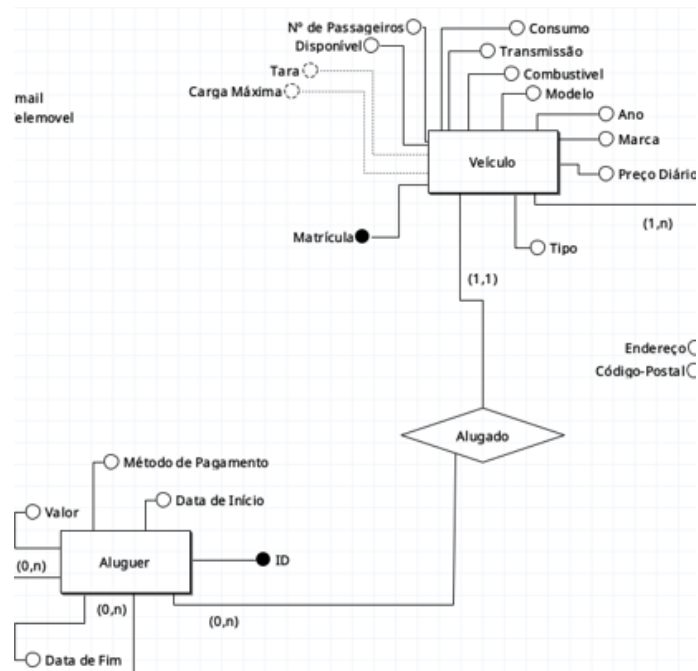


Figura 22: Relacionamento Veículo -> Aluguer - Modelo Conceptual

### 3.3.3. Veículo → Stand : 1 → n

Segundo o **RC09**, cada Veículo deve estar associado a um Stand. Um Veículo pertence a um único Stand, enquanto que um stand está associado a um ou mais veículos. Isso simplifica o processo de gestão e a alocação dos veículos na empresa.

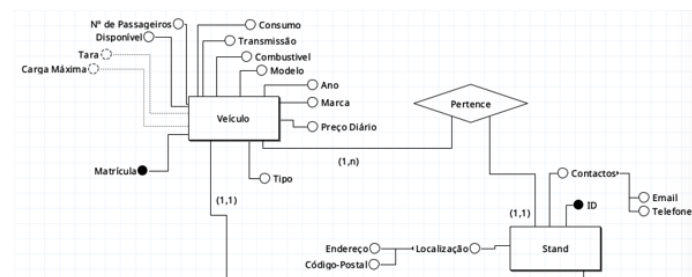


Figura 23: Relacionamento Veículo -> Stand - Modelo Conceptual

### 3.3.4. Stand → Funcionario : 1 → n

Segundo **RC11**, cada Stand pode empregar vários funcionários, sendo que este deve ter pelo menos um Funcionário associado. Cada Funcionário está associado apenas e unicamente a um Stand. Isto permite visualizar todos os funcionários de cada stand individualmente.

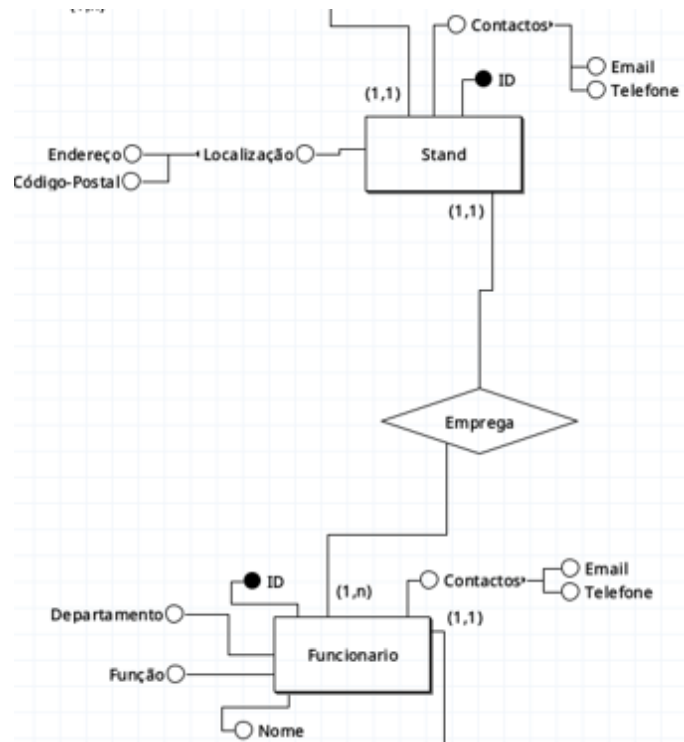


Figura 24: Relacionamento Stand -> Funcionario - Modelo Conceptual

### 3.3.5. Funcionario → Aluguer : 1 → n

Segundo o **RC08**, cada Funcionário pode supervisionar vários alugueres, sendo que cada Aluguer deve ter apenas e unicamente um funcionário responsável pela sua supervisão. Isto permite controlar todos os alugueres realizados sob a responsabilidade de cada funcionário.

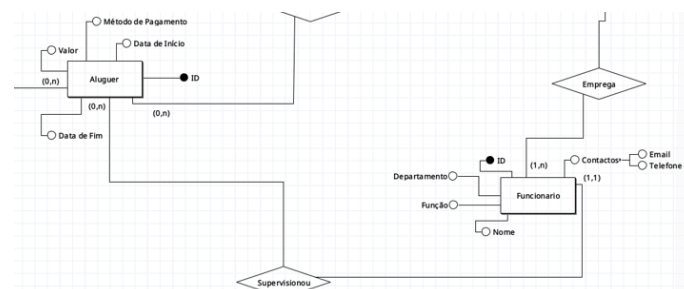


Figura 25: Relacionamento Funcionário -> Aluguer - Modelo Conceptual

### 3.4. Identificação e Caracterização dos Atributos das Entidades e dos Relacionamentos

Entidade	Atributo	Tipo de atributo	Descrição	Domínio e tamanho	Opcional	Exemplo
Ciente	ID	Chave	Identificador único	INT	N	124
Ciente	Nome	Simples	Primeiro e último nome	VARCHAR(50)	N	João Silva
Ciente	Data de Nascimento	Simples	Data de Nascimento	DATE	N	20/02/1985
Ciente	NIF	Simples	Número de contribuinte	INT	S	267326748
Ciente	Contactos	Composto	Contactos do cliente	VARCHAR(50)	S	joao@gmail.com
	- Email		Endereço de email	VARCHAR(16)	S	963456482
	- Telefone		Número de Telefone	VARCHAR(16)	S	
Aluguer	ID	Chave	Identificador do aluguer	INT	N	5432
Aluguer	Valor	Derivado	Quantidade a pagar pelo aluguer	DECIMAL(10,2)	N	1273,50
Aluguer	DataInicio	Simples	Data de início do aluguer	DATE	N	20/08/2024
Aluguer	DataFim	Simples	Data de fim do aluguer	DATE	N	27/08/2024
Aluguer	MetodoPagamento	Simples	Forma como o cliente pagou o aluguer	VARCHAR(50)	N	Multibanco
Veiculo	Matricula	Chave	Matricula que identifica o carro	VARCHAR(16)	N	AB-30-FV
Veiculo	NPassageiros	Simples	Número de passageiros que o veículo pode transportar	INT	N	5
Veiculo	Tara	Simples	Peso do veículo sem carga em Kg	INT	S	1000
Veiculo	CargaMaxima	Simples	Peso que o veículo pode carregar, somando a tara	INT	S	3500
Veiculo	Transmissao	Simples	Tipo de transmissão do veículo	VARCHAR(16)	N	Manual
Veiculo	Combustivel	Simples	Combustível usado pelo veículo	VARCHAR(16)	N	Diesel
Veiculo	Consumo	Simples	Média do consumo do carro /100 Km	Decimal(4,1)	N	7,4
Veiculo	Modelo	Simples	Modelo do veículo	VARCHAR(32)	N	Série 3
Veiculo	Marca	Simples	Marca do veículo	VARCHAR(16)	N	BMW
Veiculo	Ano	Simples	Ano de fabrico do veículo	INT	N	2020
Veiculo	Disponivel	Simples	Se está disponível para ser alugado	BOOLEAN	N	TRUE
Veiculo	PrecoDiario	Simples	Preço do aluguer do carro por dia	DECIMAL(10,2)	N	50,25
Veiculo	Tipo	Simples	Tipo de veículo (Pesado, passageiros)	VARCHAR(16)	N	Pesado

Figura 26: Caracterização dos atributos das Entidades: Cliente, Veículo e Aluguer

Entidade	Atributo	Tipo de atributo	Descrição	Domínio e tamanho	Opcional	Exemplo
Stand	ID	Chave	Identificador único	INT	N	124
Stand	NIF	Simples	Primeiro e último nome	VARCHAR(50)	N	João Silva
Stand	Contactos	Composto	Formas de contactar o stand			
	- Email		Endereço de Email	VARCHAR(50)	S	belocarsviseu@gmail.com
	- Telefone		Número de Telefone	VARCHAR(16)	S	254634834
Stand	Localizacao	Composto	Localização o stand			
	- Endereco		Rua, cidade, número da porta	VARCHAR(50)	N	Rua do Comércio, nº 45, Viseu
	-CodigoPostal		Código postal do stand	VARCHAR(16)	N	3500-116
Funcionario	ID	Chave	Identificador do funcionário	INT	N	53472
Funcionario	Departamento	Simples	Departamento onde está inserido o funcionário	VARCHAR(16)	N	Recursos Humanos
Funcionario	Funcao	Simples	Função que exerce na empresa	VARCHAR(16)	N	Diretor
Funcionario	Nome	Simples	Primeiro e último nome	VARCHAR(50)	N	Luís Pinto
Funcionario	Contactos	Composto	Formas de contactar o funcionário			
	- Email		Endereço de Email	VARCHAR(50)	S	luis@gmail.com
	- Telefone		Número de Telefone	VARCHAR(16)	S	937647828

Figura 27: Caracterização dos atributos das Entidades: Funcionário e Stand

### 3.5. Apresentação e Explicação do Modelo Conceptual Produzido

O processo iniciou-se por analisar os requisitos levantados, identificando as entidades principais e os relacionamentos entre as mesmas. Neste contexto, o **ALuguer** destaca-se como uma entidade central deste caso de estudo, pois é a componente principal de muitas operações da empresa, incluindo a atribuição de funcionários, atribuição dos veículos, entre outros.

Foi adotada a estratégia de desenvolver um esquema conceptual único com todas as vistas de utilização identificadas, obtendo assim uma abordagem abrangente permitindo uma compreensão integrada do sistema como um todo. Sendo possível garantir uma consistência e coesão do modelo.

Ao integrar todas as entidades num único esquema conceptual, a visualização das relações entre elas fica mais acessível, bem como as interdependências importantes.

De seguida, segue-se o modelo conceptual final na sua totalidade:

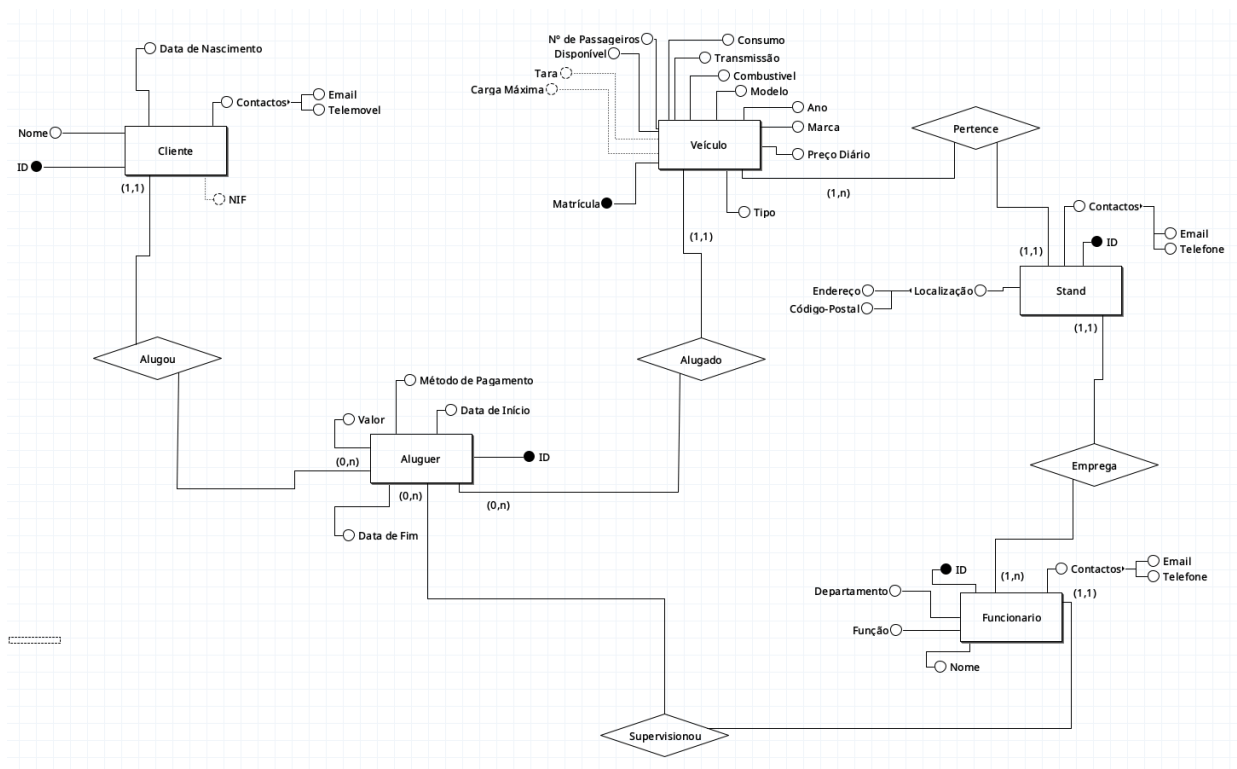


Figura 28: Modelo Conceptual Final

### 3.6. Validação do esquema

Na fase de validação do modelo conceptual da base de dados da *Belo Cars* é essencial rever minuciosamente todo o trabalho realizado, com especial atenção ao esquema desenvolvido e a documentação de suporte. A equipa de analistas está toda envolvida neste processo de revisão, examinando cada componente do esquema e a sua documentação associada.

Cada analista confere se o esquema produzido corresponde às necessidades específicas da empresa tendo em consideração os requisitos recolhidos anteriormente nesta fase.

Após algumas correções de detalhes identificadas durante o processo de revisão, os intervenientes aprovaram o esquema conceptual produzido, podendo avançar para a próxima etapa do processo de desenvolvimento.

## 4. Modelação Lógica

### 4.1. Construção e Validação do Modelo de Dados Lógico

O processo de conversão para o modelo lógico da base de dados envolveu a tradução dos elementos do modelo conceptual (desenvolvido na fase anterior) para tabelas e relacionamentos lógicos mais próximos da fase de Implementação.

Durante este processo, cada entidade identificada no modelo conceptual foi traduzida para uma tabela correspondente no modelo lógico. Os atributos de cada entidade foram reproduzidos em atributos de tabela, garantindo uma representação confiável das características de cada elemento do modelo conceptual. Além disso, os relacionamentos entre as entidades foram traduzidos em chaves estrangeiras ou novas tabelas dependendo da sua cardinalidade.

O método iniciou-se com as entidades que apresentam menos dependências, avançando gradualmente para aquelas com mais dependências, conforme se verifica na explicação do modelo lógico produzido.

Esta estratégia foi escolhida para garantir uma progressão lógica na definição das entidades e dos seus atributos. Ao abordar inicialmente as entidades com menor número de dependências, foi estabelecida uma base sólida para o modelo lógico, facilitando a integração e o desenvolvimento das entidades mais trabalhosas.

Durante este processo, utilizou-se a ferramenta *BrModelo*, pois oferece uma interface intuitiva e eficaz para a criação do esquema lógico.

#### 4.1.1. Derivação de Relações do Modelo de Dados Local

Durante o processo de conversão para o modelo lógico, seguiu-se uma abordagem adequada e estruturada. Inicialmente, identificou-se as entidades fundamentais do sistema, dando prioridade àquelas com menos dependências. Cada etapa de conversão envolveu a derivação de relações, normalização de dados e validação do esquema produzido, assegurando a consistência dos dados.

##### 4.1.1.1. Cliente

A tabela “Cliente” foi criada através da entidade *CLiente* como parte do processo de modelação lógica da base de dados, destinada à representação e registo de indivíduos que contratam os serviços da empresa.

O atributo *ID* foi definido como chave primária para garantir a identificação única de cada cliente, representado com um círculo preto no diagrama conceptual. Além disso, foi incluído o atributo simples *Nome* para manter em registo o nome completo do cliente.

Durante a conversão do modelo conceptual para o modelo lógico, observa-se que os atributos da entidade original foram mapeados diretamente na tabela *Cliente*. No entanto, os atributos *Email* e *Telemóvel*, que estão agrupados sob o conceito de *Contactos* no modelo conceptual, foram incorporados na tabela “*Contactos\_Cliente*” no modelo lógico, conforme mostrado na parte superior direita do diagrama.

A tabela “Cliente” também mantém outros atributos importantes como *Data de Nascimento* e o *NIF* (Número de Identificação Fiscal). A relação (1,1) indica que cada registo na entidade *CLiente* está associado a exatamente um registo correspondente na implementação lógica.



O diagrama ilustra claramente a transição do modelo conceptual (à esquerda) para o modelo lógico de dados (à direita), mostrando como os conceitos abstratos foram transformados em estruturas concretas na base de dados, mantendo a integridade das relações e atributos.

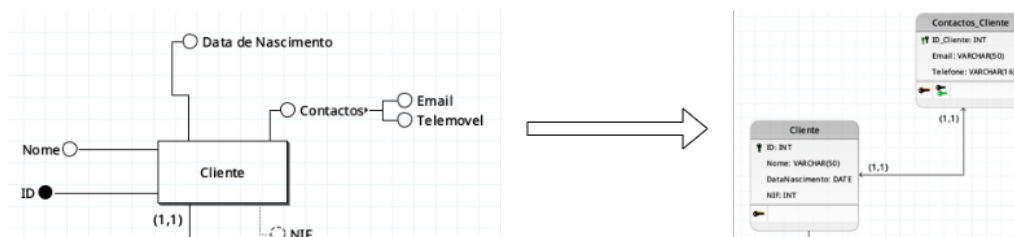


Figura 29: Conversão da Entidade "Cliente" - Modelo Lógico

#### 4.1.2. Aluguer

A tabela "Aluguer" foi criada através da entidade Aluguer como parte do processo de modelação lógica da base de dados, destinada à representação e registo das operações de aluguer de Veículos.

O atributo *ID* foi definido como chave primária para garantir a identificação única de cada operação de aluguer, representado com um círculo preto no diagrama conceptual. Este identificador único permite rastrear cada transação individualmente no sistema.

Durante a conversão do modelo conceptual para o modelo lógico, observa-se que os atributos da entidade original foram mapeados diretamente na tabela Aluguer. Os atributos simples incluem *Valor*, que regista o montante financeiro da operação, *Método de Pagamento* que indica a forma como o pagamento foi efetuado, e os campos temporais *Data de Início* e *Data de Fim* que delimitam o período do aluguer.

A cardinalidade (0,n) indica que um Aluguer pode estar associado aos Veículos ou pode não estar associado a nenhum. Na implementação lógica, observamos que a tabela "Aluguer" mantém relações com as entidades *ID\_Cliente*, *Matrícula* e *ID\_Funcionário*, sugerindo que cada registo de Aluguer está associado a um Cliente específico, um Veículo identificado por matrícula e o Funcionário responsável pela operação.

O diagrama ilustra claramente a transição do modelo conceptual (à esquerda) para o modelo lógico de dados (à direita), demonstrando como a estrutura abstrata da entidade Aluguer foi transformada numa tabela concreta na base de dados, preservando todos os atributos e relacionamentos necessários para o funcionamento do sistema de gestão de alugueres.

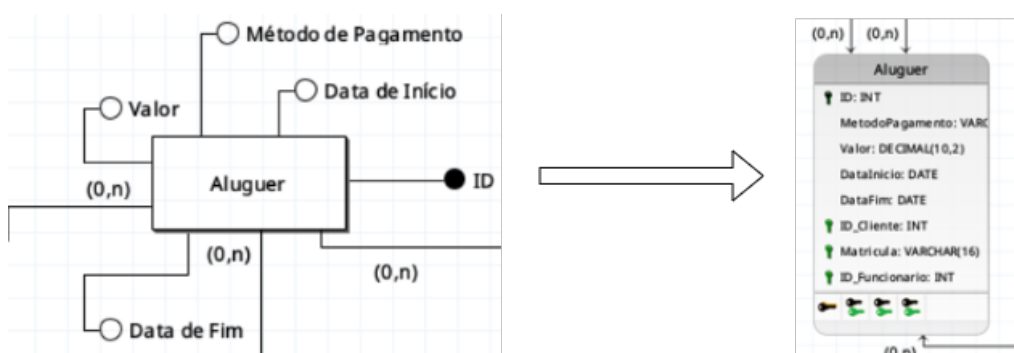


Figura 30: Conversão da Entidade "Aluguer" - Modelo Lógico

### 4.1.3. Veículo

A tabela “Veículo” foi criada através da entidade Veículo como parte do processo de modelação lógica da base de dados, destinada à representação e registo dos veículos para aluguer na empresa.

O atributo *Matrícula* foi definido como chave primária para garantir a identificação única de cada veículo, representado com um círculo preto no diagrama conceptual. Esta escolha é lógica uma vez que a matrícula é um identificador único e oficial de cada veículo.

Durante a conversão do modelo conceptual para o modelo lógico, observa-se que os atributos da entidade original foram mapeados diretamente na tabela “Veículo”. Os atributos incluem características técnicas como *Nº de Passageiros*, *Tara*, *Carga Máxima* e *Disponível* (que indica se o veículo está disponível para aluguer), além de *Transmissão* e *Combustível* que especificam detalhes mecânicos importantes.

Outros atributos relevantes incluem *Modelo*, *Ano*, *Marca* e *Tipo*, que caracterizam o veículo comercialmente, além do *Preço Diário* e *Consumo* que são informações essenciais para o cálculo de custos e identificação.

A cardinalidade (1,n) no diagrama indica que cada Veículo está associado a pelo menos um registo de Aluguer na implementação lógica. Podemos observar que na tabela resultante, todos os atributos foram implementados com tipos de dados apropriados: *VARCHAR* para textos como Marca e Modelo, *INT* para valores numéricos inteiros como Ano e Nº de Passageiros, *DECIMAL* para valores monetários como Preço Diário e *BOOLEAN* para valores lógicos como Disponível.

O diagrama ilustra claramente a transição do modelo conceptual (à esquerda) para o modelo lógico de dados (à direita), demonstrando como todos os atributos foram preservados e adequadamente tipificados na implementação da base de dados, mantendo as características essenciais para o sistema de gestão da frota de Veículos.

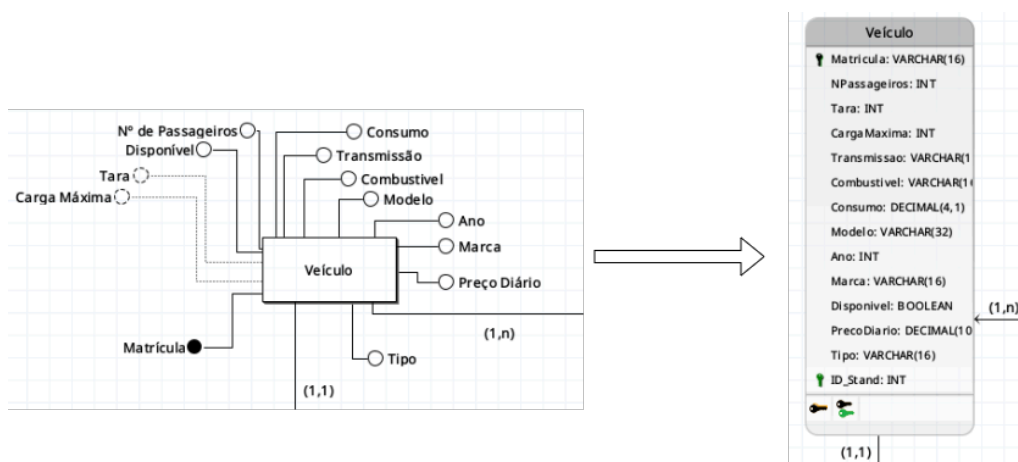


Figura 31: Conversão da Entidade "Veículo" - Modelo Lógico

### 4.1.4. Funcionário

A tabela “Funcionário” foi criada através da entidade Funcionário como parte do processo de modelação lógica da base de dados, destinada à representação e registo dos colaboradores da empresa responsáveis pelos serviços de aluguer.

O atributo *ID* foi definido como chave primária para garantir a identificação única de cada funcionário, representado com um círculo preto no diagrama conceptual. Este identificador único permite rastrear cada colaborador individualmente no sistema.

Durante a conversão do modelo conceptual para o modelo lógico, observa-se que os atributos da entidade original foram mapeados para a tabela “Funcionário”. Os atributos simples incluem *Nome*, *Departamento* e *Função*, que registam informações fundamentais sobre o colaborador dentro da estrutura organizacional.

Os atributos multi-valor *Email* e *Telefone*, que estão agrupados sob o conceito de *Contactos* no modelo conceptual, foram implementados através da criação de uma tabela separada “Contactos\_Funcionario” no

modelo lógico, conforme mostrado na parte inferior direita do diagrama. Esta abordagem permite associar dois tipos de contactos a um único funcionário.

As cardinalidades (1,n) presentes nos relacionamentos indicam que cada funcionário está associado a pelo menos um Stand. Na tabela resultante, os atributos foram implementados com tipos de dados apropriados: *INT* para o identificador, *VARCHAR* para *strings* como Nome, Departamento e Função.

O diagrama ilustra claramente a transição do modelo conceptual (à esquerda) para o modelo lógico de dados (à direita), demonstrando como a estrutura abstrata da entidade Funcionário foi transformada em tabelas concretas na base de dados, preservando todos os atributos e relacionamentos necessários para o funcionamento do sistema de gestão de Recursos Humanos da empresa.

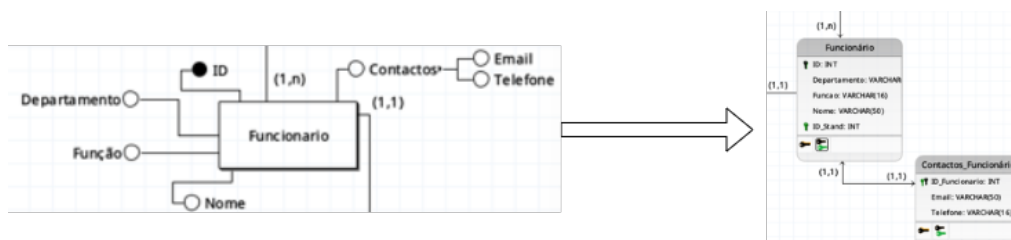


Figura 32: Conversão da Entidade "Funcionario" - Modelo Lógico

#### 4.1.5. Stand

A tabela “Stand” foi criada através da entidade Stand como parte do processo de modelação lógica da base de dados, destinada à representação e registo dos locais físicos onde os veículos são disponibilizados para aluguer.

O atributo *ID* foi definido como chave primária para garantir a identificação única de cada Stand, representado com um círculo preto no diagrama conceptual. Este identificador único permite referenciar cada sucursal individualmente no sistema.

Durante a conversão do modelo conceptual para o modelo lógico, observa-se que os atributos foram organizados em tabelas distintas para melhor estruturação dos dados. O atributo composto *Localização*, que inclui *Endereço* e *Código Postal* no modelo conceptual, foi implementado como uma tabela separada “Localização” no modelo lógico. Esta tabela mantém uma relação com a tabela principal “Stand” através do campo *ID\_Stand*.

Os atributos multi-valor *Email* e *Telefone*, que estão agrupados sob o conceito de *Contactos* no modelo conceptual, foram implementados numa tabela separada “Contactos\_Stand” no modelo lógico, conforme mostrado na parte superior direita do diagrama. Esta abordagem permite associar dois tipos de contactos a um único stand.

Na implementação lógica, os atributos foram tipificados apropriadamente: *INT* para identificadores, *VARCHAR* para textos como *Endereço*, *Email*, *Telefone* e *Código Postal*.

O diagrama ilustra claramente a transição do modelo conceptual (à esquerda) para o modelo lógico de dados (à direita), demonstrando como a entidade “Stand” foi decomposta em três tabelas relacionadas: “Stand” (tabela principal), “Localização” (para dados de morada) e “Contactos\_Stand” (para informações de contacto). Esta estrutura normalizada otimiza o armazenamento de dados e facilita a manutenção das informações relacionadas aos *Stands* da empresa.

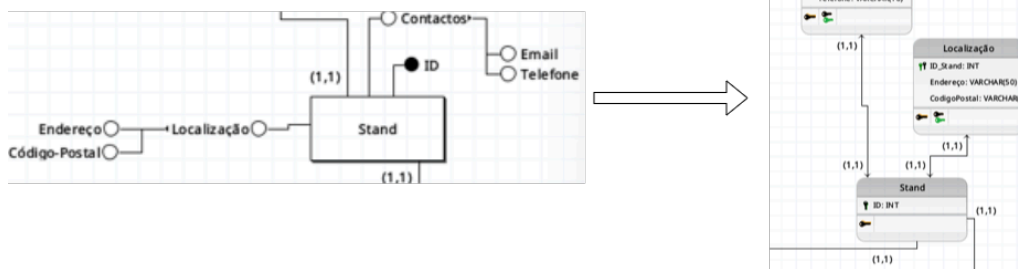


Figura 33: Conversão da Entidade "Stand" - Modelo Lógico

## 4.2. Apresentação do Modelo Lógico Final Produzido

Na imagem seguinte, encontra-se o modelo lógico da base de dados desenvolvido:

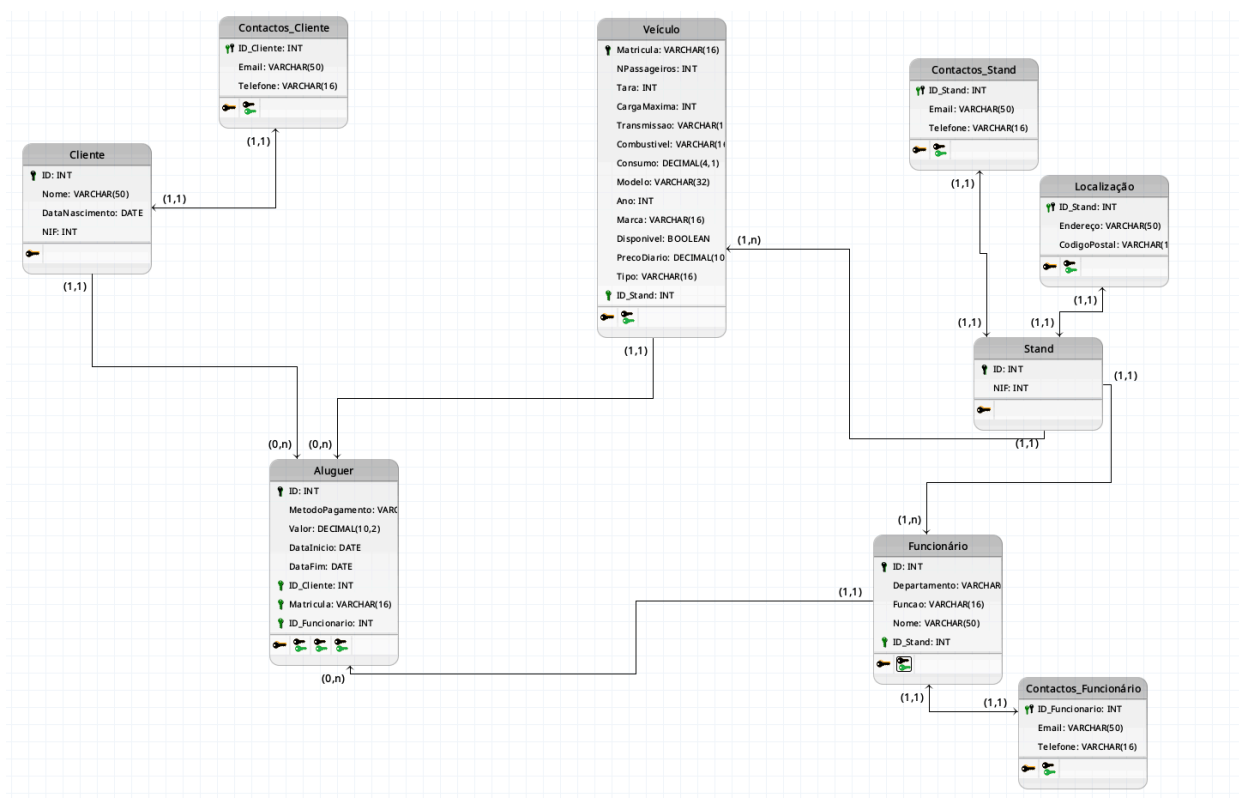


Figura 34: Modelo Lógico Final

## 4.3. Normalização de Dados

Neste processo de conceptualização e desenvolvimento de uma Base de Dados deve sempre ser tido em conta o conceito de Normalização de Dados, que nada mais é do que a manutenção e a garantia dos aspetos fundamentais que estão na base de qualquer sistema que implemente uma Base de Dados - a asseguaração da **Integridade dos Dados** e **Redundância Controlada**. Para isso estão descritas várias regras

que regem esta estrutura dos Modelos de Base de Dados Relacionais, regras estas a que chamamos **Formas Normais (FN)**.

Desta forma, queremos demonstrar nesta secção que a nossa proposta satisfaz as três primeiras Formas Normais - a 1FN, 2FN e 3FN. Depois de uma análise cuidada do nosso modelo podemos afirmar que:

- Todas as tabelas do modelo **têm uma chave primária bem definida, garantindo que cada registo é único**. Isto também assegura que **cada campo armazena apenas um valor por registo**, cumprindo assim os requisitos da Primeira Forma Normal (1FN), ou seja, não existem atributos com múltiplos valores em nenhuma tabela. Exemplos de tabelas que comprovam este conceito com a utilização de um Id único, são: Cliente, Aluguer, Veículo, Stand, Funcionário.
- Além disso, observa-se que todos os atributos de cada tabela **dependem totalmente da sua chave primária**. Isso significa que os critérios da Segunda Forma Normal (2FN) estão satisfeitos, garantindo que não existem dependências parciais entre os atributos e a chave primária. Exemplos desta regra são: Contactos\_Cliente, Contactos\_Funcionário, Contactos\_Stand, Localizacao, que utilizam chaves estrangeiras para se conectarem à tabela principal em vez de fazerem parte da própria tabela, criando dependências parciais.
- Por fim, verificamos também que todos os atributos não-chave de cada tabela dependem **apenas e diretamente** da chave primária e **não de outro atributo não-chave**. Isso satisfaz os critérios da Terceira Forma Normal (3FN), eliminando dependências transitivas. Exemplos incluem Cliente, Stand, Aluguer, Veículo, onde todos os campos dependem unicamente da respetiva chave primária.

Conclui-se que o modelo cumpre os requisitos da Primeira, Segunda e Terceira Formas Normais. Todas as tabelas apresentam uma chave primária que garante a unicidade dos registos, evitam atributos multi-valorados, e não possuem dependências parciais ou transitivas. Desta forma, o modelo apresenta uma estrutura relacional coerente, consistente e bem normalizada até à 3FN.

## 4.4. Validação do Modelo recorrendo à Álgebra Relacional

Nesta secção pretende-se afirmar a correção dos Requisitos de Manipulação estipulados anteriormente, recorrendo à Álgebra Relacional. Note-se que foram selecionados apenas alguns dos Requisitos de Manipulação para manter uma exposição pertinente e sem repetição de formulações ou estratégias semelhantes.

**RM03:** *Tem de ser possível a listagem de todos os alugueres efetuados por um cliente mediante a introdução do seu id (incluindo também o id, a marca e o modelo do veículo associado) e, opcionalmente, um intervalo de datas limite*

Para satisfazer este requisito, aplica-se uma operação de seleção sobre a tabela Aluguer, filtrando os registos pelo ID\_Cliente fornecido e, opcionalmente, por um intervalo de datas (DataInicio e DataFim). Em seguida, executa-se uma junção com a relação Veiculo, através do atributo ID\_Veiculo, para obter os detalhes do veículo associado a cada aluguer. Por fim, utiliza-se a projeção para apresentar apenas os campos relevantes: ID\_Aluguer, Nome do Cliente, Marca, Modelo.

- **Expressão:**

$$\pi_{\text{Aluguer.ID, Nome, Marca, Modelo}} \left( \sigma_{(\text{ID\_Cliente} = 1 \text{ and } \text{DataInicio} \geq 20250401 \text{ and } \text{DataFim} \leq 20250410)} \left( \left( \text{Aluguer} \bowtie \text{ID\_Cliente} = \text{Cliente.ID} \left( \text{Cliente} \right) \right) \bowtie \text{Matricula} = \text{Veiculo.Matricula} \left( \text{Veiculo} \right) \right) \right)$$

5 ms

Figura 35: Expressão em Álgebra Relacional do RM03

- **Árvore:**

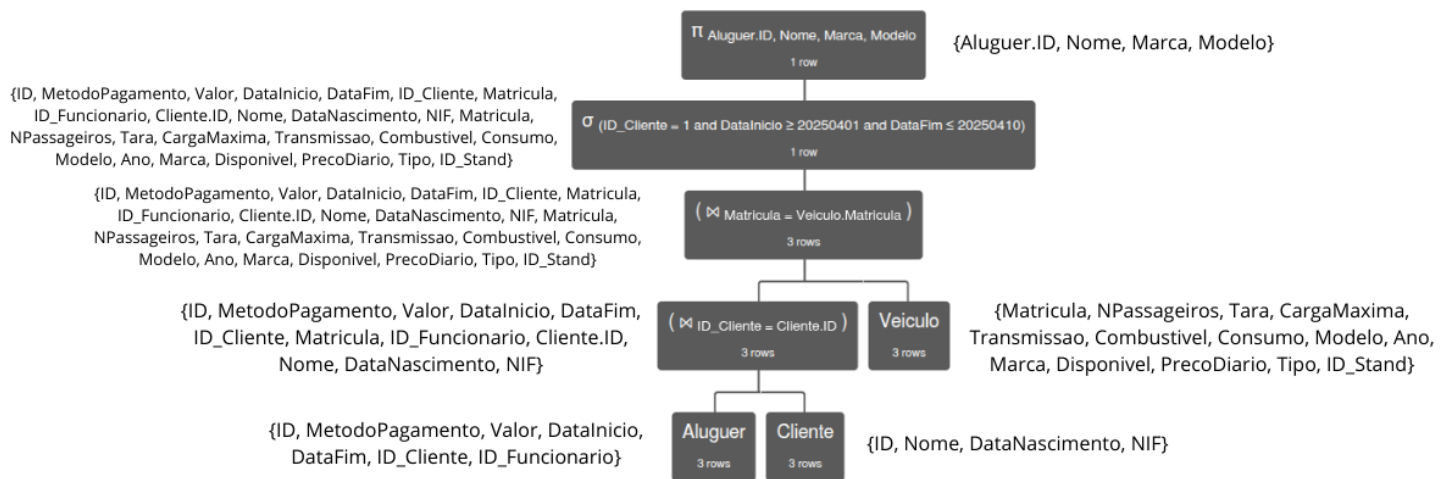


Figura 36: Ávore lógica da expressão do RM03

**RM05:** Consulta de todos os veículos de um dado stand que estão disponíveis para alugar

Para cumprir este requisito, inicia-se com uma seleção sobre a tabela Veiculo, utilizando o ID\_Stand correspondente ao stand pretendido. Esta operação é combinada com a verificação do estado de Disponibilidade do veículo (por exemplo, atributo Disponível=true). Através desta filtragem, obtêm-se apenas os veículos pertencentes ao stand indicado que estão atualmente disponíveis para aluguer. Uma projeção final pode ser aplicada para devolver apenas os atributos relevantes, como ID\_Veiculo, Marca e Modelo.

- **Expressão:**

$\Pi_{ID, Matricula, Modelo, Marca, PrecoDiario} \sigma_{ID\_Stand = 1 \text{ and } Disponível} ( (Stand) \bowtie_{ID = ID\_Stand} Veiculo )$

Figura 37: Expressão em Álgebra Relacional do RM05

- **Árvore:**

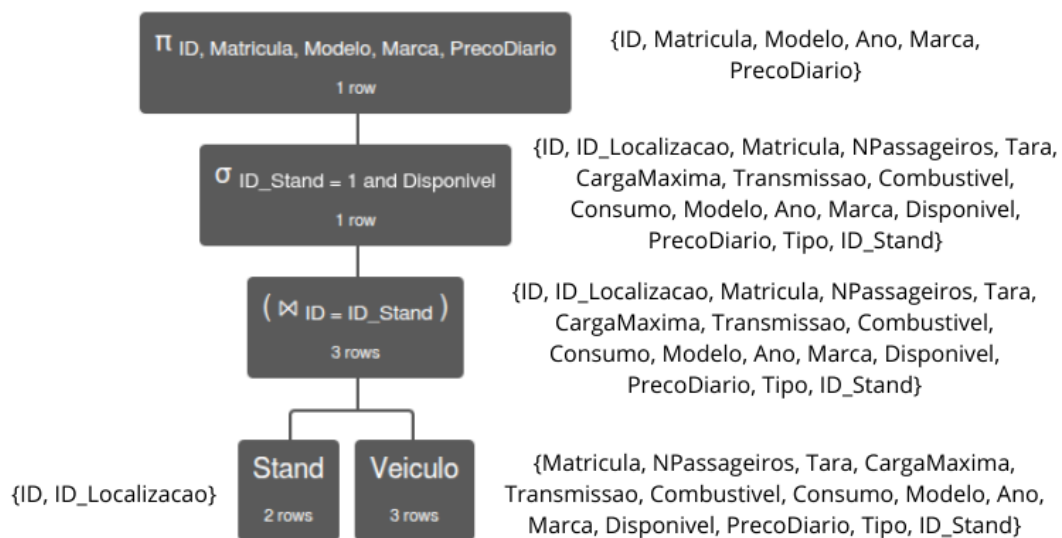


Figura 38: Ávore lógica da expressão do RM05

**RM06:** Calcular o valor da receita num dado dia, associado a todos as transações efetuadas nesse mesmo dia

Para calcular a receita total de um determinado dia, aplica-se uma seleção sobre a tabela Aluguer, filtrando os registos pela data indicada (DataInicio = data\_desejada). De seguida, utiliza-se uma operação de agrupamento e agregação ( $\gamma$ ) para calcular a soma dos valores do atributo Preço, resultando no total da receita associada a todas as transações efetuadas nesse dia.

- **Expressão:**

$$\sigma_{\text{DataInicio} = 20250401} \left( \gamma_{\text{DataInicio}; \text{SUM(Valor)} \rightarrow \text{ReceitaTotal}} (\text{Aluguer}) \right)$$

3 ms

Figura 39: Expressão em Álgebra Relacional do RM06

- **Árvore:**

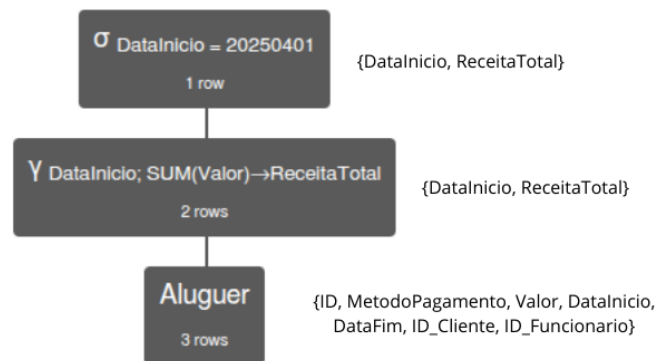


Figura 40: Árvore lógica da expressão do RM06

**RM07:** Listar por ordem decrescente os funcionários que mais efetuaram alugueres para monitorizar o desempenho

Para monitorizar o desempenho dos funcionários com base no número de alugueres realizados, inicia-se com uma junção entre as tabelas Funcionario e Aluguer, através do atributo ID\_Funcionario. Em seguida, aplica-se uma operação de agregação ( $\gamma$ ) para contar o número de alugueres por funcionário. Por fim, utiliza-se a ordenação decrescente ( $\tau$ ) sobre o total de alugueres para listar os funcionários por ordem de desempenho.

- **Expressão:**

$$\pi_{\text{total\_alugueres}, \text{Funcionario.ID}, \text{Funcionario.Nome}, \text{Funcionario.ID\_Stand}} \left( \tau_{\text{total\_alugueres desc}} \left( \gamma_{\text{ID\_Funcionario}; \text{COUNT(ID)} \rightarrow \text{total\_alugueres}} (\text{Aluguer}) \bowtie_{\text{ID\_Funcionario} = \text{ID}} (\text{Funcionario}) \right) \right)$$

3 ms

Figura 41: Expressão em Álgebra Relacional do RM07

- **Árvore:**

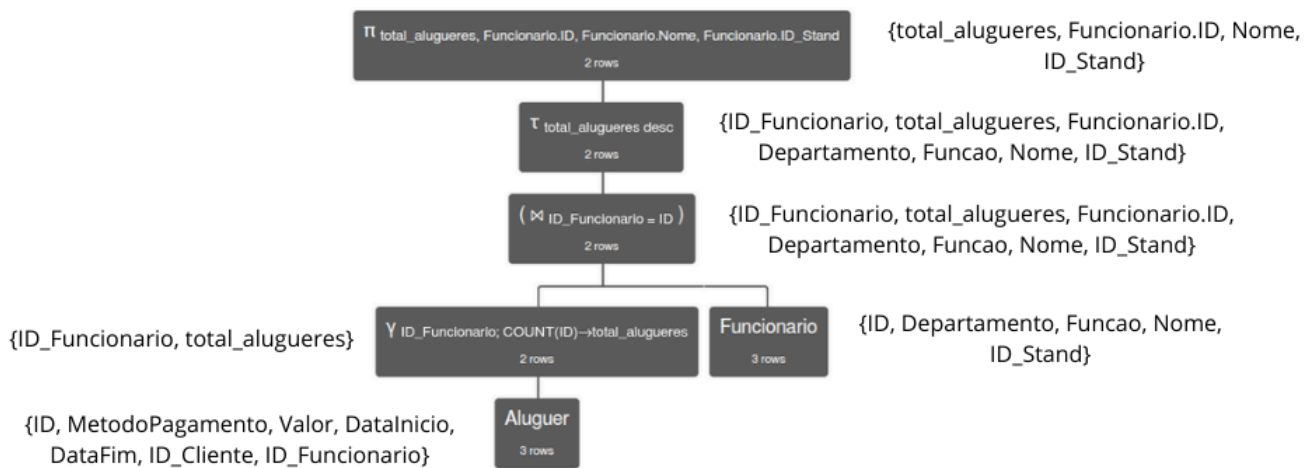


Figura 42: Ávore lógica da expressão do RM07

Para demonstrar como ficaria o ‘*Top Funcionários*’, fica aqui um exemplo ilustrativo da mesma *querie* executada na Calculadora de Álgebra Relacional *RelaX*, que retrata o *top* decrescente dos Funcionários que supervisionaram mais alugueres:

<b>total_alugueres</b>	<b>Funcionario.ID</b>	<b>Funcionario.Nome</b>	<b>Funcionario.ID_Stand</b>
2	1	'carlos_magalhaes'	1
1	2	'marta_costa'	1

Figura 43: Tabela resultado da execução da Querie no RelaX - exemplo populado ilustrativo

Em suma, a aplicação da Álgebra Relacional permitiu formalizar e estruturar logicamente os requisitos de manipulação propostos, garantindo uma representação rigorosa e expressiva das operações sobre os dados. Através do uso combinado de operadores como seleção, projeção, junção, agregação e ordenação, foi possível satisfazer integralmente os cenários definidos, assegurando a extração precisa da informação pretendida e proporcionando uma base sólida para a futura implementação das consultas em SQL no sistema final.



## 5. Implementação Física

### 5.1. Apresentação e explicação da base de dados implementada

A implementação do esquema lógico na ferramenta **MySQL Workbench** envolveu a criação de diversas tabelas, cada uma projetada para armazenar informações específicas relacionadas com a empresa de aluguer de veículos, como planeado até ao momento, tanto no Modelo Conceitual como no Modelo Lógico.

O processo de implementação seguiu uma abordagem de baixo para cima, seguindo o modelo lógico apresentado nas imagens, começando com tabelas que possuíam menos dependências e avançando para aquelas com mais relações com outras tabelas (Por exemplo a Tabela ALuguer tem mais dependências que a Tabela Funcionário). Isto garantiu que as chaves estrangeiras pudessem ser referenciadas corretamente à medida que as tabelas eram criadas.

Cada tabela foi cuidadosamente projetada para refletir as entidades e relacionamentos identificados durante a fase de modelação/planeamento do sistema de base de dados.

O esquema físico resultante foi produzido como um conjunto de tabelas interligadas, cada uma com a sua própria estrutura de colunas e restrições de chave, que irão ser evidenciadas mais à frente. Este esquema proporciona uma base sólida para armazenar e manipular os dados da *Belo Cars* de aluguer de veículos de forma eficiente e organizada, com o principal objetivo de responder aos requisitos estabelecidos no início do projeto.

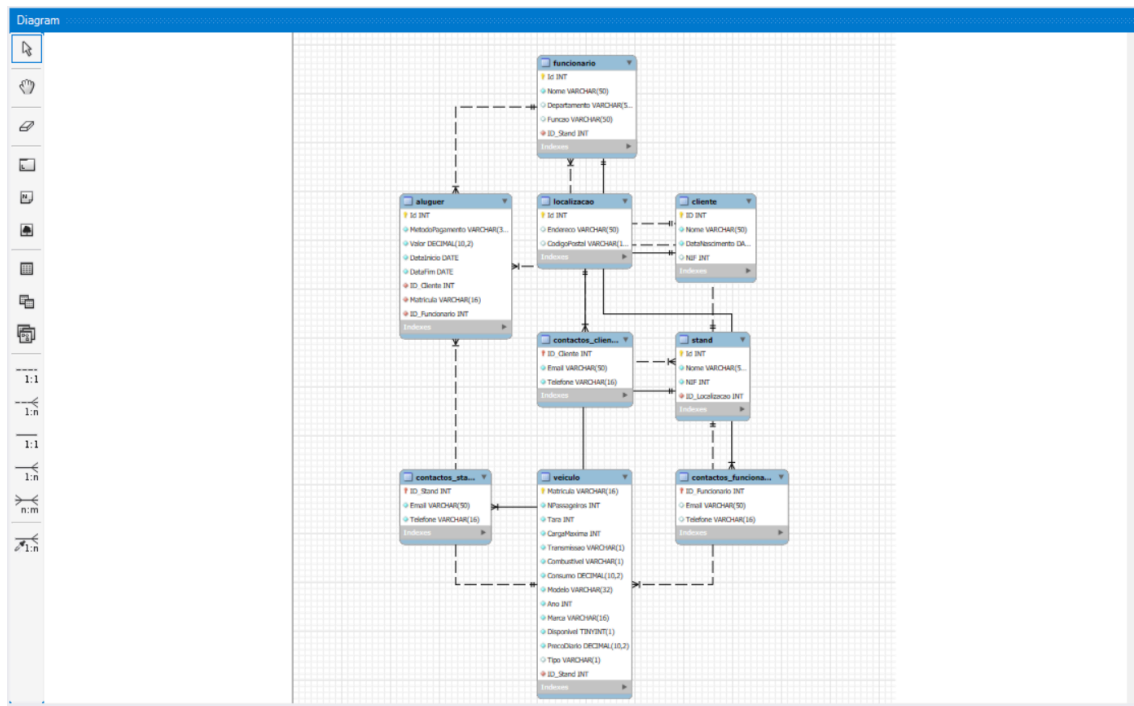


Figura 44: Esquema Físico gerado no MySQL Workbench

### 5.1.1. Tabelas do Modelo Físico

#### 5.1.1.1. Criação da tabela Localizacao

A tabela “Localizacao” armazena os endereços físicos dos stands da empresa. Cada localização é identificada por um ID único, contendo também o endereço e o código postal.

```
-- =====
-- Tabela: Localizacao
-- Descrição: Armazena os endereços físicos associados aos Stands
-- =====
CREATE TABLE IF NOT EXISTS Localizacao (
  Id INT,
  Endereco VARCHAR(50),
  CodigoPostal VARCHAR(10),
  PRIMARY KEY (Id)
);
```

#### 5.1.1.2. Criação da tabela Stand

A tabela “Stand” representa os stands de veículos da empresa, estando cada um associado a uma localização. Contém também o nome e NIF de cada stand.

```
-- =====
-- Tabela: Stand
-- Descrição: Representa os stands de veículos da empresa
-- Cada stand está associado a uma localização (1:1)
-- =====
CREATE TABLE IF NOT EXISTS Stand (
  Id INT NOT NULL,
  Nome VARCHAR(50) NOT NULL,
  NIF INT NOT NULL UNIQUE,
  ID_Localizacao INT NOT NULL UNIQUE,
  PRIMARY KEY (Id),
  CONSTRAINT fk_Stand_Localizacao
    FOREIGN KEY (ID_Localizacao)
    REFERENCES Localizacao(Id)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);
```

#### 5.1.1.3. Criação da tabela Contactos\_Stand

A tabela “Contactos\_Stand” contém os contactos (email e telefone) de cada stand. Está ligada à tabela Stand com remoção em cascata para garantir consistência.

```
-- =====
-- Tabela: Contactos_Stand
-- Descrição: Contactos associados a cada Stand (1:1)
-- Remoção em cascata para consistência de dados
-- =====
CREATE TABLE IF NOT EXISTS Contactos_Stand (
  ID_Stand INT NOT NULL,
  Email VARCHAR(50) NOT NULL UNIQUE,
  Telefone VARCHAR(16) NOT NULL UNIQUE,
  PRIMARY KEY (ID_Stand),
  CONSTRAINT fk_ContactosStand_Stand
    FOREIGN KEY (ID_Stand)
    REFERENCES Stand(Id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

#### 5.1.1.4. Criação da tabela Cliente

A tabela “Cliente” regista os dados dos clientes que alugam veículos. Inclui o nome, data de nascimento e NIF de cada cliente.

```
-- =====
-- Tabela: Cliente
-- Descrição: Regista os dados dos Clientes que alugam Veículos
-- =====
CREATE TABLE IF NOT EXISTS Cliente (
    ID INT NOT NULL,
    Nome VARCHAR(50) NOT NULL,
    DataNascimento DATE NOT NULL,
    NIF INT UNIQUE,
    PRIMARY KEY (ID)
);
```

#### 5.1.1.5. Criação da tabela Contactos\_Cliente

A tabela “Contactos\_Cliente” guarda os contactos de cada cliente. Está ligada à tabela Cliente e suporta remoção em cascata.

```
-- =====
-- Tabela: Contactos_Cliente
-- Descrição: Contactos associados a Clientes (1:1)
-- Remoção em cascata garante integridade
-- =====
CREATE TABLE IF NOT EXISTS Contactos_Cliente (
    ID_Cliente INT NOT NULL,
    Email VARCHAR(50) UNIQUE NOT NULL,
    Telefone VARCHAR(16) UNIQUE NOT NULL,
    PRIMARY KEY (ID_Cliente),
    CONSTRAINT fk_ContactosCliente_Cliente
        FOREIGN KEY (ID_Cliente)
        REFERENCES Cliente(ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

#### 5.1.1.6. Criação da tabela Funcionario

A tabela “Funcionario” regista os dados dos funcionários dos stands, incluindo o departamento, função e a associação a um stand.

```
-- =====
-- Tabela: Funcionario
-- Descrição: Contém os dados dos Funcionários dos Stands
-- Cada Funcionário está associado a um Stand (N:1)
-- =====
CREATE TABLE IF NOT EXISTS Funcionario (
    Id INT NOT NULL,
    Nome VARCHAR(50) NOT NULL,
    Departamento VARCHAR(50),
    Funcao VARCHAR(50),
    ID_Stand INT NOT NULL,
    PRIMARY KEY (Id),
    CONSTRAINT fk_Funcionario_Stand
        FOREIGN KEY (ID_Stand)
        REFERENCES Stand(Id)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
);
```

#### 5.1.1.7. Criação da tabela Contactos\_Funcionario

A tabela “Contactos\_Funcionario” armazena os contactos dos funcionários, ligados diretamente à tabela Funcionario.

```
-- =====
-- Tabela: Contactos_Funcionario
-- Descrição: Contactos associados a Funcionários (1:1)
-- =====
CREATE TABLE IF NOT EXISTS Contactos_Funcionario (
    ID_Funcionario INT NOT NULL,
    Email VARCHAR(50) UNIQUE,
    Telefone VARCHAR(16) UNIQUE,
    PRIMARY KEY (ID_Funcionario),
    CONSTRAINT fk_ContactosFuncionario_Funcionario
        FOREIGN KEY (ID_Funcionario)
        REFERENCES Funcionario(Id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

#### 5.1.1.8. Criação da tabela Veiculo

A tabela “Veiculo” contém os dados dos veículos disponíveis para aluguer, incluindo atributos técnicos e ligação ao stand.

```
-- =====
-- Tabela: Veiculo
-- Descrição: Contém os dados dos Veículos para aluguer
-- Inclui restrições para validação de dados
-- =====
CREATE TABLE IF NOT EXISTS Veiculo (
    Matricula VARCHAR(16) NOT NULL,
    NPassageiros INT NOT NULL CHECK (NPassageiros > 0),
    Tara INT,
    CargaMaxima INT,
    Transmissao VARCHAR(1) NOT NULL CHECK (Transmissao IN ('M','A')), -- Manual/Automática
    Combustivel VARCHAR(1) NOT NULL CHECK (Combustivel IN ('G','D','E','H')), -- Gasolina/
    Diesel/Elétrico/Híbrido
    Consumo DECIMAL(10,2) NOT NULL CHECK (Consumo >= 0),
    Modelo VARCHAR(32) NOT NULL,
    Ano INT NOT NULL,
    Marca VARCHAR(16) NOT NULL,
    Disponivel BOOLEAN NOT NULL DEFAULT FALSE,
    PrecoDiario DECIMAL(10,2) NOT NULL CHECK (PrecoDiario >= 0),
    Tipo VARCHAR(2) CHECK (Tipo IN ('P','M','PM')), -- Passageiros, Mercadorias, Ambos
    ID_Stand INT NOT NULL,
    PRIMARY KEY (Matricula),
    CONSTRAINT fk_Veiculo_Stand
        FOREIGN KEY (ID_Stand)
        REFERENCES Stand(Id)
);
```

#### 5.1.1.9. Criação da tabela Aluguer

A tabela “Aluguer” regista os alugueres efetuados. Cada registo associa um cliente, um veículo e um funcionário responsável, com as datas de aluguer, valor e método de pagamento.

```
-- =====
-- Tabela: Aluguer
-- Descrição: Regista os Alugueres efetuados
-- Cada Aluguer está ligado a um Cliente, Funcionário e Veículo
-- =====
```

```
CREATE TABLE IF NOT EXISTS Aluguer (
    Id INT NOT NULL,
    MetodoPagamento VARCHAR(32),
    Valor DECIMAL(10,2) NOT NULL DEFAULT 0.00 CHECK (Valor >= 0),
    DataInicio DATE NOT NULL,
    DataFim DATE NOT NULL,
    ID_Cliente INT NOT NULL,
    Matricula VARCHAR(16) NOT NULL,
    ID_Funcionario INT NOT NULL,
    PRIMARY KEY (Id),
    CONSTRAINT fk_Aluguer_Cliente
        FOREIGN KEY (ID_Cliente)
        REFERENCES Cliente(Id),
    CONSTRAINT fk_Aluguer_Veiculo
        FOREIGN KEY (Matricula)
        REFERENCES Veiculo(Matricula),
    CONSTRAINT fk_Aluguer_Funcionario
        FOREIGN KEY (ID_Funcionario)
        REFERENCES Funcionario(Id)
);
```

## 5.2. Criação de utilizadores da base de dados

Esta secção foi combinada num só ficheiro (*SQL Script*) que inclui a **definição e criação de grupos** diferentes de intervenientes, as **permissões associadas a cada grupo** e por fim a criação efetiva de alguns **exemplos de Utilizadores** associados ao caso de estudo da empresa.

### 5.2.1. Definição de Grupos de Utilizadores (*Roles*)

Foram definidos três grupos principais de utilizadores no sistema: **guest**, **staff** e **admin**. Estes grupos refletem diferentes níveis de acesso e responsabilidade na gestão dos dados da empresa, que iremos detalhar de seguida.

```
-- Criação de grupos de Utilizadores (por cargo)
CREATE ROLE IF NOT EXISTS guest;
CREATE ROLE IF NOT EXISTS staff;
CREATE ROLE IF NOT EXISTS admin;
```

### 5.2.2. Atribuição de Permissões por *Role*

Cada grupo de utilizadores recebe um conjunto específico de permissões de acordo com o seu nível de acesso. Estas permissões determinam as operações que cada grupo pode realizar sobre as tabelas da base de dados. O **guest** é o grupo de utilizadores por defeito e por isso o mais limitado - tem apenas acesso a ver à informação dos veículos; a **staff** tem permissões mais alargadas que o anterior - consegue adicionar, atualizar e remover alugueres, clientes e veículos; e o **admin** seria o grupo mais privilegiado - tendo acesso completo a todas as tabelas, para controlo de todas os registos, além de possibilidade de garantir permissões a outros utilizadores.

```
-- CONVIDADO pode apenas consultar carros
GRANT SELECT ON BeloCars.Carro TO guest;

-- FUNCIONÁRIO tem permissões mais alargadas (mas não totais)
GRANT SELECT, INSERT, UPDATE, DELETE ON BeloCars.Aluguer TO staff;
GRANT SELECT, UPDATE ON BeloCars.Cliente TO staff;
GRANT SELECT, INSERT, UPDATE ON BeloCars.Carro TO staff;

-- ADMIN tem permissões completas
GRANT ALL PRIVILEGES ON BeloCars.* TO admin WITH GRANT OPTION;
```

### 5.2.3. Criação de Utilizadores Convidados (*Guests*)

Os utilizadores do tipo **guest** representam visitantes/convidados que apenas podem consultar os veículos disponíveis para aluguer. Foi criado um Utilizador a título de exemplo: 'utilizador\_guest'@'localhost'.

```
-- Utilizadores tipo VISITANTE
CREATE USER IF NOT EXISTS 'utilizador_guest'@'localhost' IDENTIFIED BY 'guest123';
GRANT guest TO 'utilizador_guest'@'localhost';
```

### 5.2.4. Criação de Funcionários (*Staff*)

Os utilizadores do tipo **staff** correspondem a funcionários dos stands da *BeloCars*, com permissões para gerir dados de alugueres, veículos e clientes (respetivamente as operações de adição, remoção e atualização de registos associados a essas tabelas). A título de exemplo foram criados os acessos para os funcionários func1 e func2, que poderiam ser substituídos pelos nomes dos funcionários correspondentes.

```
-- Utilizadores tipo FUNCIONÁRIO
CREATE USER IF NOT EXISTS 'func1'@'localhost' IDENTIFIED BY 'func123';
CREATE USER IF NOT EXISTS 'func2'@'localhost' IDENTIFIED BY 'func456';
GRANT staff TO 'func1'@'localhost';
GRANT staff TO 'func2'@'localhost';
```

### 5.2.5. Criação do Administrador (*Admin*)

Para uma gestão eficiente da base de dados da *Belo Cars*, foi criado um exemplo de utilizador administrador ('admin\_ceo'@'localhost') com permissões completas para todos os objetos da base de dados. Os administradores são responsáveis pela administração do sistema de bases de dados e gestão geral das entidades que não devem ser acedidas pelos restantes utilizadores acima descritos.

```
-- Utilizador tipo ADMINISTRADOR
CREATE USER IF NOT EXISTS 'admin_ceo'@'localhost' IDENTIFIED BY 'admin123';
GRANT admin TO 'admin_ceo'@'localhost';
```

**Nota:** É importante referir que os clientes não são criados na base de dados como utilizadores, pois é insensato estes acederem diretamente à base de dados. Em vez disso, as informações dos clientes são geridas e acedidas através de aplicações intermediárias que interagem com o sistema de base de dados.

### 5.2.6. Atribuição de *Roles* por Defeito

Por fim, cada utilizador recebe automaticamente o seu respetivo grupo (*role*) como função por defeito, simplificando a gestão de permissões.

```
-- Atribuir ROLES por defeito
SET DEFAULT ROLE ALL TO
'utilizador_guest'@'localhost',
'cliente1'@'localhost', 'cliente2'@'localhost',
'func1'@'localhost', 'func2'@'localhost',
'admin_ceo'@'localhost';
```

## 5.3. Povoamento da base de dados

De forma a possibilitar o teste eficiente do sistema desenvolvido, e de forma a servir de exemplo de como ficaria a base de dados povoada, procedeu-se à criação de um *script SQL* que preenche-se as tabelas de forma coerente.

A ordem de povoamento das tabelas foi planeada de forma a garantir que as dependências entre as tabelas fossem respeitadas. Primeiro, as tabelas “Cliente”, “Stand” e “Funcionário” foram populadas, seguidas pelas tabelas “Veículo” e, por último, a tabela “Aluguer”. As tabelas que correspondem a relacionamentos

do tipo “1:1” (por exemplo: “Contactos\_Funcionário”, “Contactos\_Cliente”, “Localizacao”, ...) foram realizadas no fim para garantir que as chaves estrangeiras correspondentes já tinham sido inseridas.

Assim, cada tabela foi povoada de forma a criar um povoamento de dados consistente e coerente, preparando a base de dados para ser utilizada eficientemente nas operações do sistema.

### 5.3.1. Tabela Funcionário

O povoamento da tabela “Funcionário” foi realizado através de uma série de execuções da instrução INSERT com valores que representam funcionários únicos. Os restantes atributos associados aos Contactos de um funcionário estão associados à posterior inserção na tabela “Contactos\_Funcionario”.

```
-- Entidade: Funcionario
INSERT INTO Funcionario (Id, Nome, Departamento, Funcao, ID_Stand)
VALUES
(1, 'Carlos Mendes', 'Comercial', 'Vendedor', 1),
(2, 'Ana Costa', 'Administração', 'Gestora', 2),
(3, 'Miguel Rocha', 'Manutenção', 'Mecânico', 3),
(4, 'Sofia Ribeiro', 'Comercial', 'Vendedora', 1),
(5, 'Tiago Lopes', 'Administração', 'Contabilista', 2);
```

### 5.3.2. Tabela Cliente

O povoamento da tabela “Cliente” foi realizado inserindo diretamente os dados na base de dados, usando o INSERT. Cada registo representa um cliente específico, contendo informações como o nome, data de nascimento e NIF. As restantes informações de contacto serão associadas a cada cliente na tabela “Contactos\_Cliente” com recurso a uma chave estrangeira.

```
-- Entidade: Cliente
INSERT INTO Cliente (ID, Nome, DataNascimento, NIF)
VALUES
(1, 'João Silva', '1985-06-15', 123456780),
(2, 'Maria Oliveira', '1990-10-20', 234567890),
(3, 'Rui Gonçalves', '1975-03-05', 345678901),
(4, 'Inês Martins', '1995-12-30', 456789012),
(5, 'Pedro Ferreira', '1980-09-10', 567890123);
```

### 5.3.3. Tabela Stand

O povoamento da tabela “Stand” foi realizado através de uma série de execuções da instrução INSERT com valores que representam stands únicos. Cada stand tem informações como nome, NIF e localização associada (através da referência ao atributo *id* respetivo da tabela “Localizacao”).

```
-- Entidade: Stand
INSERT INTO Stand (Id, Nome, NIF, ID_Localizacao)
VALUES
(1, 'BeloCars Lisboa', 123456789, 1),
(2, 'BeloCars Braga', 987654321, 2),
(3, 'BeloCars Viseu', 456789123, 3);
```

### 5.3.4. Tabela Veículo

O povoamento da tabela “Veículo” foi realizado inserindo diretamente os dados na base de dados, usando o INSERT. Cada registo representa um veículo específico, contendo informações como a matrícula, marca, modelo, ano, consumo, transmissão, combustível, preço diário e stand associado.

```
-- Entidade: Veiculo
INSERT INTO Veiculo (Matricula, NPassageiros, Tara, CargaMaxima, Transmissao,
Combustivel, Consumo, Modelo, Ano, Marca, Disponivel, PrecoDiario, Tipo, ID_Stand)
VALUES
('00-AA-11', 5, NULL, NULL, 'M', 'G', 6.5, 'Civic', 2020, 'Honda', TRUE, 45.00, 'P', 1),
('11-BB-22', NULL, 1500, 2500, 'A', 'D', 7.8, 'Sprinter', 2021, 'Mercedes', TRUE, 60.00,
```

```
'M', 2),
('22-CC-33', 5, 1100, 2400, 'PM', 'G', 5.8, 'Polo', 2019, 'Volkswagen', TRUE, 40.00,
'P', 3),
('33-DD-44', 7, NULL, NULL, 'A', 'D', 6.9, 'Sharan', 2022, 'Volkswagen', TRUE, 70.00,
'P', 1),
('44-EE-55', NULL, 1300, 3500, 'M', 'E', 0.0, 'eSprinter', 2023, 'Mercedes', TRUE,
80.00, 'M', 2);
```

### 5.3.5. Tabela Aluguer

O povoamento da tabela “Aluguer” foi realizado inserindo diretamente os dados na base de dados, usando o INSERT. Cada registo representa um aluguer específico, contendo informações como a data de início, data de fim, valor, método de pagamento, veículo alugado e cliente associado.

```
-- Entidade: Aluguer
INSERT INTO Aluguer (Id, MetodoPagamento, Valor, DataInicio, DataFim, ID_Cliente,
Matricula, ID_Funcionario)
VALUES
(1, 'Cartão Crédito', 135.00, '2024-05-01', '2024-05-04', 1, '00-AA-11', 1),
(2, 'MB Way', 180.00, '2024-05-10', '2024-05-13', 2, '11-BB-22', 2),
(3, 'Cartão Débito', 120.00, '2024-06-01', '2024-06-04', 3, '22-CC-33', 3),
(4, 'Paypal', 210.00, '2024-06-10', '2024-06-14', 4, '33-DD-44', 4),
(5, 'Transferência', 190.00, '2024-06-20', '2024-06-23', 5, '44-EE-55', 5);
```

### 5.3.6. Tabela Localização

O povoamento da tabela “Localizacao” foi realizado através da inserção direta de registos com identificadores únicos, moradas e respetivos códigos postais. Esta entidade permite localizar geograficamente os stands da *Belo Cars* (que corresponde ao requisito referente às 3 localizações atuais da empresa, e possibilidade de expansão).

```
-- Entidade: Localizacao
INSERT INTO Localizacao (Id, Endereco, CodigoPostal)
VALUES
(1, 'Rua Central, 123', '1000-001'),
(2, 'Av. das Laranjeiras, 45', '2000-002'),
(3, 'Rua das Oliveiras, 10', '3000-003'),
(4, 'Praça da República, 50', '4000-004'),
(5, 'Estrada Nacional 1, Km 120', '5000-005');
```

### 5.3.7. Tabela Contactos\_Stand

A tabela “Contactos\_Stand” armazena os contactos principais dos stands da empresa, associando-os a cada unidade geográfica. Os dados foram inseridos com os respetivos emails e telefones para cada stand.

```
-- Entidade: Contactos_Stand
INSERT INTO Contactos_Stand (ID_Stand, Email, Telefone)
VALUES
(1, 'lisboa@belocars.pt', '912345678'),
(2, 'braga@belocars.pt', '934567890'),
(3, 'viseu@belocars.pt', '938112233');
```

### 5.3.8. Tabela Contactos\_Cliente

A tabela “Contactos\_Cliente” armazena dados multivalorados associados aos clientes, como email e número de telefone. Estes dados foram adicionados após o povoamento da tabela principal “Cliente”.

```
-- Entidade: Contactos_Cliente
INSERT INTO Contactos_Cliente (ID_Cliente, Email, Telefone)
VALUES
(1, 'joao.silva@email.com', '911223344'),
(2, 'maria.oliveira@email.com', '966778899'),
```



```
(3, 'rui.goncalves@email.com', '922334455'),
(4, 'ines.martins@email.com', '933445566'),
(5, 'pedro.ferreira@email.com', '944556677');
```

### 5.3.9. Tabela Contactos\_Funcionario

A tabela “Contactos\_Funcionario” armazena dados multivalorados associados aos contactos dos funcionários, como email e número de telefone. Estes dados foram adicionados após o povoamento da tabela principal “Funcionário”.

```
-- Entidade: Contactos_Funcionario
INSERT INTO Contactos_Funcionario (ID_Funcionario, Email, Telefone)
VALUES
(1, 'carlos.mendes@belocars.pt', '913456789'),
(2, 'ana.costa@belocars.pt', '923456789'),
(3, 'miguel.rocha@belocars.pt', '945667788'),
(4, 'sofia.ribeiro@belocars.pt', '956778899'),
(5, 'tiago.lopes@belocars.pt', '967889900');
```

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

### 5.4.1. Tamanho da base de dados inicial

```
SELECT
  table_name AS 'Tabela',
  ROUND((data_length + index_length) / 1024, 2) AS 'Tamanho_KB',
  ROUND((data_length + index_length) / 1024 / 1024, 2) AS 'Tamanho_MB'
FROM information_schema.tables
WHERE table_schema = 'BeloCars';
```

Com base na execução do comando SQL acima, foi possível apurar o tamanho total da base de dados sem quaisquer dados inseridos (isto é, apenas com a estrutura das tabelas e os respetivos índices criados - de acordo com o Plano de Indexação que iremos referir mais à frente). O resultado do bloco SQL referido é o seguinte:

Tabela	Tamanho_KB	Tamanho_MB
aluguer	64.00	0.06
cliente	32.00	0.03
contactos_cliente	48.00	0.05
contactos_funcionario	48.00	0.05
contactos_stand	48.00	0.05
funcionario	32.00	0.03
localizacao	16.00	0.02
stand	48.00	0.05
veiculo	32.00	0.03

Figura 45: Tamanho relativo a cada tabela (incluindo índices)

### Tamanho total (tabelas + índices):

$64 + 32 + 48 + 48 + 48 + 32 + 16 + 48 + 32 = 368 \text{ KB}$  (ou aproximadamente 0.36 MB)

Este valor representa o espaço reservado pelas estruturas físicas das tabelas, chaves primárias, chaves estrangeiras e índices adicionais definidos no esquema, mesmo sem conter registos de dados.

#### 5.4.2. Tamanho da base de dados no futuro (crescimento anual)

Para calcular a taxa de crescimento anual da base de dados da *Belo Cars*, com base na sua estrutura física, assumiremos um cenário realista de uso e aplicaremos estimativas fundamentadas sobre:

- Número de registos por tabela por ano
- Tamanho de cada registo, com base nos tipos de dados
- Fatores de crescimento (ex.: mais clientes e alugueres ao longo dos anos)

##### 1. Suposições do Cenário:

As seguintes estimativas baseiam-se no número esperado de registos por ano para cada tabela, considerando o uso realista do sistema.

Tabela	Crescimento Anual Esperado
Localizacao	+1 por ano
Stand	+1 por ano
Contactos_Stand	+1 por ano
Cliente	+150 por ano
Contactos_Cliente	+150 por ano
Funcionario	+10 por ano
Contactos_Funcionario	+10 por ano
Veiculo	+200 por ano
Aluguer	+2000 por ano

Tabela 1: Dados relativos ao número inicial de registos em cada tabela e à estimativa de crescimento anual esperado.

## 2. Estimativa de Espaço por Registo:

Os tamanhos são calculados com base nos tipos de dados utilizados em MySQL (por exemplo, **INT** = 4 B, **DATE** = 3 B, **BOOLEAN** = 1 B, **VARCHAR(N)** ≈ N).

Tabela	Tamanho Estimado por Registo
Localizacao	≈ 64 B
Stand	≈ 62 B
Contactos_Stand	≈ 70 B
Cliente	≈ 61 B
Contactos_Cliente	≈ 70 B
Funcionario	≈ 158 B
Contactos_Funcionario	≈ 70 B
Veiculo	≈ 99 B
Aluguer	≈ 71 B

Tabela 2: Estimativa do espaço ocupado por cada registo em cada tabela, tendo em conta os tipos de dados e *overheads* típicos de armazenamento no MySQL.

## 3. Crescimento por Ano (1º ano após inserção):

Multiplicação do número de novos registos por ano pelo respetivo tamanho de cada registo.

Tabela	Registos/Ano	Tamanho Registo	Total Anual (KB)
Localizacao	1	64 B	0.06
Stand	1	62 B	0.06
Contactos_Stand	1	70 B	0.07
Cliente	150	61 B	9.15
Contactos_Cliente	150	70 B	10.5
Funcionario	10	158 B	1.58
Contactos_Funcionario	10	70 B	0.36
Veiculo	200	99 B	19.8
Aluguer	2000	71 B	142
*	*	*	*
Total			≈ 183.58 KB

Tabela 3: Cálculo do crescimento estimado do tamanho da base de dados ao longo de um ano, considerando o número de novos registos por tabela e o respetivo espaço por registo.

#### 4. Crescimento Acumulado ao Longo dos Anos:

Assumindo crescimento linear e sem remoção de dados:

Ano	Crescimento Total Estimado (KB)	Tamanho Acumulado (KB)
2025 (inicial)	368	368
2026	+184	≈ 552
2027	+184	≈ 736
2028	+184	≈ 920
2029	+184	≈ 1104
2030	+184	≈ 1288

Tabela 4: Evolução acumulada do tamanho total da base de dados (em KB) ao longo dos anos, considerando um crescimento constante segundo os valores definidos anteriormente.

A base de dados **BeloCars** tem uma taxa de crescimento estimada de aproximadamente **184 KB por ano** mantendo-se bastante eficiente em termos de armazenamento. Não esquecendo o resultado inicial da implementação, calculado anteriormente, de **368KB**, no primeiro ano a base de dados já teria cerca de **0.55MB** de tamanho estimado ( $0.368 + 0.184 = 0.552\text{MB}$ ); e após cinco anos de operação com os volumes projetados, estima-se um tamanho acumulado de cerca de **1.3 MB**.

Este crescimento é sustentável, mas poderão ser implementadas estratégias como:

- Limpeza periódica de alugueres antigos
- Compressão de *backups* e cópias de segurança
- Otimização dos atributos e respetivos tipos, e índices

Estas medidas ajudam a manter a performance e escalabilidade a longo prazo.

## 5.5. Definição e caracterização de vistas de utilização em SQL

A implementação de vistas de utilização em *SQL* oferece uma maneira eficaz de simplificar o acesso e a gestão de dados complexos na base de dados. As vistas são consultas pré-definidas que associam e filtram dados de várias tabelas, proporcionando uma visão mais específica e organizada das informações relevantes. De seguida, podemos visualizar a definição e caracterização de várias vistas criadas para diferentes propósitos do sistema de base de dados da *Belo Cars* de aluguer de veículos.

- *View* que mostra todos os veículos da base de dados, útil para consultas rápidas sobre a frota disponível.

```
create view vwVeiculosGeral as
select * from Veiculo;
```

- *View* que apresenta um resumo de todos os alugueres efetuados, mostrando quem alugou, que carro foi alugado, por quanto tempo e por quanto dinheiro.

```
create view vwAlugueresGeral as
select Matricula as Carro, ID_Cliente as Cliente,
       ID_Funcionario as Funcionario, DataInicio,
       DataFim, Valor, MetodoPagamento
from Aluguer;
```

- *View* que identifica os funcionários com melhor desempenho, com base no número de alugueres efetuados. Pode ser usada para avaliações de produtividade.

```
create view vwFuncionariosDesempenho as
select F.Id as Funcionario, F.Nome,
       count(AL.Id) as Alugueres
from Funcionario as F
left join Aluguer as AL on F.Id = AL.ID_Funcionario
group by F.Id, F.Nome
order by Alugueres desc;
```

- View que associa alugueres a clientes, com informações complementares dos veículos alugados (marca e modelo). Ideal para histórico de alugueres por cliente.

```
create view vwAlugueresCliente as
select AL.ID_Cliente, AL.Matricula,
       AL.DataInicio, AL.DataFim,
       V.Marca, V.Modelo
from Aluguer as AL
join Veiculo as V on AL.Matricula = V.Matricula;
```

## 5.6. Indexação do Sistema de Dados

A implementação de um plano de Indexação adequado a um problema visa a otimização das operações associadas às tabelas mais “concorridas” de um SGBD.

No caso de estudo da *Belo Cars* a ideia foi selecionar as tabelas que contêm um maior grau de relacionamentos e que, por isso, podem ser as tabelas que contém mais incidência no que toca à execução de operações (SELECT, INSERT, UPDATE, ...) sobre as mesmas. Este processo culminou na seleção das tabelas **Aluguer**, **Veiculo** e **Funcionario**, pois é evidente pelo número de relacionamentos que possuem que serão as mais interrogadas (como vimos até agora nas *queries* implementadas) - principalmente a tabela **Alu** o ponto central de todo este problema.

A ideia é melhorar o desempenho do sistema, associando mais rapidamente a sua chave primária à posição do registo numa dada a a forma otimizar a procura (sem necessidade de procura exaustiva por um dado registo) associada a um determinado atributo muito procurado, por exemplo, *Id's* (chaves primárias e/ou estrangeiras), intervalos de datas, disponibilidade (*booleanos*), entre outros que iremos exemplificar de seguida.

Desta forma seguem-se exemplos de índices que foram implementados com vista a aumentar a performance das consultas (e outras operações) ao sistema implementado:

```
-- =====
-- Índices essenciais para a Tabela: Aluguer
-- =====

-- Acesso frequente por cliente
CREATE INDEX idx_Aluguer_ID_Cliente ON Aluguer(ID_Cliente);

-- Acesso por intervalo de datas (ex: históricos)
CREATE INDEX idx_Aluguer_DataInicio ON Aluguer(DataInicio);

-- =====
-- Índices essenciais para a Tabela: Veiculo
-- =====

-- Consulta por stand (veículos de um stand)
CREATE INDEX idx_Veiculo_ID_Stand ON Veiculo(ID_Stand);

-- Pesquisa por disponibilidade
CREATE INDEX idx_Veiculo_Disponivel ON Veiculo(Disponivel);

-- Consulta rápida por marca e modelo
```

```
CREATE INDEX idx_Veiculo_Marca_Modelo ON Veiculo(Marca, Modelo);

-- =====
-- Índices essenciais para a Tabela: Funcionario
-- =====

-- Acesso por stand (funcionários de um stand)
CREATE INDEX idx_Funcionario_ID_Stand ON Funcionario(ID_Stand);
```

## 5.7. Implementação de procedimentos, funções e gatilhos

- Calcular o valor total feito nos alugueres desse dia:

```
DELIMITER $$
CREATE FUNCTION fn_ValorDiario(
    f_data DATE
)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE valorDiario DECIMAL(10,2);

    SELECT IFNULL(SUM(Valor), 0) INTO valorDiario
    FROM Aluguer
    WHERE DATE(DataInicio) = f_data;
    RETURN valorDiario;
END $$
DELIMITER ;
```

Esta função calcula o valor total de alugueres iniciados numa determinada data. Para isso, procura na tabela Aluguer todas as entradas cuja data de início corresponde à data fornecida (*f\_data*) e soma os valores de cada aluguer. Se não houver alugueres para esse dia, retorna 0.

- Filtra funcionários por departamento e/ou função:

```
DROP PROCEDURE IF EXISTS filtraFuncionarios;
DELIMITER $$
CREATE PROCEDURE filtraFuncionarios(
    IN p_Departamento VARCHAR(50),
    IN p_Funcao VARCHAR(50)
)
BEGIN
    SELECT *
    FROM vwFuncionariosGeral
    WHERE (p_Departamento IS NULL OR Departamento = p_Departamento)
    AND (p_Funcao IS NULL OR Funcao = p_Funcao);
END $$
DELIMITER ;
```

Este procedimento permite listar os funcionários com base no departamento e/ou função que desempenham. Os parâmetros são opcionais: se forem passados como **NULL**, o filtro correspondente é ignorado. A informação é obtida da **vwFuncionariosGeral**, uma *view* com os dados relevantes dos funcionários.

- Devolve o top 3 de clientes que mais dinheiro gastaram num determinado stand:

```
DELIMITER $$
CREATE PROCEDURE sp3TopClientesStand
    (IN NrStand INT)
BEGIN
```

```

SELECT S.Id AS "Stand",
       S.Nome AS "Nome Stand",
       C.Nome AS "Cliente",
       SUM(A.Valor) AS "Valor Total Pago"
FROM Aluguer AS A
     INNER JOIN Cliente AS C
         ON A.ID_Cliente = C.ID
     INNER JOIN Funcionario AS F
         ON F.Id = A.ID_Funcionario
     INNER JOIN Stand AS S
         ON F.ID_Stand = S.Id
WHERE S.Id = NrStand
GROUP BY C.ID, C.Nome, S.Id, S.Nome
ORDER BY SUM(A.Valor) DESC
LIMIT 3;
END $$
DELIMITER ;

```

Este procedimento devolve os 3 clientes que mais dinheiro gastaram em alugueres num determinado stand. Para isso, junta as tabelas Aluguer, Cliente, Funcionario e Stand, usando o stand como filtro principal e ordenando os clientes pelo valor total gasto em alugueres.

- Lista todos os alugueres supervisionados por um funcionário, podendo ter um intervalo de datas:

```

DELIMITER $$
CREATE PROCEDURE listaAlugueresFuncionario(
    IN p_Funcionario INT,
    IN p_DataInicio DATE,
    IN p_DataFim DATE
)
BEGIN
    SELECT VWAF.Matricula, VWAF.DataInicio,
           VWAF.DataFim, VWAF.Valor FROM vwAlugueresFuncionario AS VWAF
    JOIN Funcionario AS F ON VWAF.ID_Funcionario = F.Id
    WHERE (VWAF.ID_Funcionario = p_Funcionario)
           AND (p_DataInicio IS NULL OR VWAF.DataInicio >= p_DataInicio)
           AND (p_DataFim IS NULL OR VWAF.DataFim <= p_DataFim);
END $$
DELIMITER ;

```

Este procedimento permite consultar os alugueres atribuídos a um funcionário específico. É possível filtrar os resultados por um intervalo de datas, através dos parâmetros opcionais **p\_DataInicio** e **p\_DataFim**. A informação é lida da *view* **vwAlugueresFuncionario**.

- Insere um novo veículo:

```

DELIMITER $$
CREATE PROCEDURE spAdicionarVeiculo(
    IN p_Matricula VARCHAR(16),
    IN p_NPassageiros INT,
    IN p_Tara INT,
    IN p_CargaMaxima INT,
    IN p_Transmissao VARCHAR(1),
    IN p_Combustivel VARCHAR(1),
    IN p_Consumo DECIMAL(10,2),
    IN p_Modelo VARCHAR(32),
    IN p_Ano INT,
    IN p_Marca VARCHAR(16),
    IN p_Disponivel BOOLEAN,
    IN p_PrecoDiario DECIMAL(10,2),
    IN p_Tipo VARCHAR(2),

```

```

        IN p_ID_Stand INT
    )
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Stand WHERE Id = p_ID_Stand) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Erro: 0 stand especificado não existe.';
    END IF;

    IF p_Tipo = 'P' THEN
        IF p_Tara IS NOT NULL OR p_CargaMaxima IS NOT NULL THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Erro: Veículo de passageiros não deve ter Tara ou Carga
Máxima.';
        END IF;

        IF p_NPassageiros IS NULL THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Erro: Veículo de passageiros deve ter número de
passageiros válido.';
        END IF;

    ELSEIF p_Tipo = 'M' THEN
        IF p_Tara IS NULL OR p_CargaMaxima IS NULL THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Erro: Veículo de mercadorias precisa de Tara e Carga
Máxima.';
        END IF;

        IF p_NPassageiros IS NOT NULL THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Erro: Veículo de mercadorias não deve ter número de
passageiros.';
        END IF;

    ELSEIF p_Tipo = 'PM' THEN
        IF p_Tara IS NULL OR p_CargaMaxima IS NULL THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Erro: Veículo misto precisa de Tara e Carga Máxima.';
        END IF;

        IF p_NPassageiros IS NULL THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Erro: Veículo misto deve ter número de passageiros
válido.';
        END IF;

    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Erro: Tipo de veículo inválido.';
    END IF;

    INSERT INTO Veiculo (
        Matricula, NPassageiros, Tara, CargaMaxima,
        Transmissao, Combustivel, Consumo, Modelo,
        Ano, Marca, Disponivel, PrecoDiario, Tipo, ID_Stand
    )
VALUES (
    p_Matricula, p_NPassageiros, p_Tara, p_CargaMaxima,
    p_Transmissao, p_Combustivel, p_Consumo, p_Modelo,
    p_Ano, p_Marca, p_Disponivel, p_PrecoDiario, p_Tipo, p_ID_Stand
);

```



```
END $$
DELIMITER ;
```

Este procedimento insere um novo veículo na base de dados, validando os dados fornecidos com base no tipo de veículo (**P** - Passageiros, **M** - Mercadorias, **PM** - Misto). Garante a existência do stand de destino e aplica validações específicas ao número de passageiros, tara e carga máxima. Em caso de erro, é gerado um sinal com uma mensagem descritiva.

- Remove um Funcionário:

```
DELIMITER $$
CREATE PROCEDURE spRemoverFuncionario(
    IN p_Id INT
)
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Funcionario WHERE Id = p_Id) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Erro: O funcionário especificado não existe.';
    ELSE
        DELETE FROM Funcionario WHERE Id = p_Id;
    END IF;
END $$
DELIMITER ;
```

Este procedimento remove um funcionário com base no seu ID, desde que ele exista. Caso contrário, é lançado um erro informando que o funcionário não existe. Útil para manter a integridade e evitar remoções acidentais.

- Trigger para indisponibilizar um veículo:

```
DELIMITER $$
CREATE TRIGGER trg_IndisponibilizarVeiculo
AFTER UPDATE ON Aluguer
FOR EACH ROW
BEGIN
    IF OLD.MetodoPagamento IS NULL AND NEW.MetodoPagamento IS NOT NULL THEN
        UPDATE Veiculo
        SET Disponivel = FALSE
        WHERE Matricula = NEW.Matricula;
    END IF;
END $$
DELIMITER ;
```

Este *trigger* é executado após uma atualização na tabela Aluguer. Caso o campo **MetodoPagamento** mude de **NULL** para um valor válido, o *trigger* considera que o aluguer foi confirmado/pago, ou seja, o veículo foi levantado pelo cliente, e define o veículo associado como indisponível.

## 6. Implementação do Sistema de Migração de Dados

A implementação de um sistema de migração de dados visa melhorar e facilitar o povoamento de dados nas diferentes tabelas pertencentes à base de dados. Foram elaborados 3 diferentes *scripts* na linguagem de programação *Python* de forma a fazer a transcrição dos dados guardados em diferentes formatos: CSV, JSON e Postgres. De forma a facilitar a introdução dos dados, recorreremos a bibliotecas como o **pandas** e **json**., tendo em conta o requisito que referia a possibilidade de as 3 localizações atuais da empresa possuírem um sistema de armazenamento em formatos diferentes que precisam de ser tratados e convertidos para o presente modeloeste

### 6.1. CSV

Um **ficheiro CSV** é um formato de ficheiro simples que armazena os dados em forma de tabela, onde cada linha representa um registo e cada coluna os diferentes campos dos registos presentes.

#### 6.1.1. Importação de Bibliotecas e Configuração Inicial

```
import pandas as pd
import mysql.connector
import os
from dotenv import load_dotenv
```

Primeiramente importam-se as tabelas necessárias: *pandas* de forma a possibilitar a inserção e a manipulação de dados tabelados, *mysql.connector* permite a conexão com a base de dados em MySQL, *os* permite a aplicação *python* interagir com o sistema operativo de forma a possuir acesso a caminhos de ficheiros e outras funcionalidades do sistema, *load\_dotenv* carrega as variáveis definidas no ficheiro *.env*, este permite separar as credenciais do código fonte.

#### 6.1.2. Conexão com a Base de Dados

```
db = mysql.connector.connect(
    host=os.getenv("HOST"),
    user=os.getenv("USER"),
    password=os.getenv("PASSWORD"),
    database=os.getenv("DATABASE"),
    ssl_disabled=True
)
```

O bloco de código apresentado permite estabelecer a conexão com a base de dados. As credenciais são obtidas nas variáveis de ambiente, esta função permite melhorar a segurança, de forma a evitar a introdução de dados sensíveis diretamente no código e a portabilidade permite a conexão com vários ambientes.

### 6.1.3. Extração dos dados guardados em ficheiro

```
def extrair_csvs():
    print("Extraindo ficheiros CSV...")
    dados = {
        "veiculo": pd.read_csv("dados/veiculo.csv"),
        "cliente": pd.read_csv("dados/cliente.csv"),
        "contactos_cliente": pd.read_csv("dados/contactoscliente.csv"),
        "localizacao": pd.read_csv("dados/localizacao.csv"),
        "stand": pd.read_csv("dados/stand.csv"),
        "contactos_stand": pd.read_csv("dados/contactosstand.csv"),
        "aluguer": pd.read_csv("dados/aluguer.csv"),
        "funcionario": pd.read_csv("dados/funcionario.csv"),
        "contactos_funcionario": pd.read_csv("dados/contactosfuncionario.csv")
    }
    return dados
```

A função `extrair_csvs`, utiliza a biblioteca `pandas` de forma a automatizar o processo de armazenamento dos dados guardados nos ficheiros CSV, no dicionário `dados`, de forma a que em cada *DataFrame* do dicionário contenha apenas registos referentes a uma tabela da base de dados.

### 6.1.4. Padronização de dados

```
def transformar_dados(dfs):
    print("Transformando dados...")

    def limpar(df):
        df.columns = df.columns.str.lower().str.strip()

        for colunas in ['datanascimento', 'datainicio', 'datafim']:
            if colunas in df.columns:
                df[colunas] = pd.to_datetime(df[colunas], errors='coerce')
        return df

    return {nome: limpar(df) for nome, df in dfs.items()}
```

A função `transformar_dados` primeiramente transforma o nome das colunas, convertendo-as em minúsculas e retirando os espaços em branco de forma a evitar erros sintáticos. De seguida converte as diferentes datas introduzidas para o tipo de dados usado no MySQL.

### 6.1.5. Ordem de introdução das tabelas na base de dados

```
ordem_tabelas = [  
    "localizacao",  
    "stand",  
    "contactos_stand",  
    "cliente",  
    "contactos_cliente",  
    "funcionario",  
    "contactos_funcionario",  
    "veiculo",  
    "aluguer"  
]
```

A lista apresentada representa a ordem da introdução das tabelas de forma a não surgirem conflitos entre tabelas no que consta às chaves estrangeiras.

### 6.1.6. Introdução dos registos na base de dados

```
def carregar_para_mysql_manual(df, tabela, db):  
    cursor = db.cursor()  
    colunas = df.columns.tolist()  
    placeholders = ", ".join(["%s"] * len(colunas))  
    cols_formatted = ", ".join(colunas)  
  
    for _, row in df.iterrows():  
        valores = tuple(row[col] for col in colunas)  
        sql = f"INSERT INTO {tabela} ({cols_formatted}) VALUES ({placeholders})"  
        try:  
            cursor.execute(sql, valores)  
        except Exception as e:  
            print(f"Erro ao inserir na tabela '{tabela}': {e}")  
    db.commit()  
    cursor.close()
```

No trecho de código apresentado é efetuada a introdução dos registos. Em primeiro lugar é criado um cursor de forma a executar os comandos do MySQL. Os *dataframes* são executados linha a linha extraíndo os valores correspondentes e executando a instrução INSERT com os valores extraídos imediatamente antes. Caso a execução das instruções não funcione, este lança uma exceção devolvendo uma mensagem de erro contendo a tabela no qual surgiu o equívoco.

### 6.1.7. Função geral de introdução dos dados

```
def carregar_para_sql_ordenado(dfs_transformados, db):  
    for tabela in ordem_tabelas:  
        df = dfs_transformados.get(tabela)  
        if df is not None:  
            print(f"A carregar a tabela '{tabela}'...")  
            try:  
                carregar_para_mysql_manual(df, tabela, db)  
                print(f"Tabela '{tabela}' carregada com sucesso.")  
            except Exception as e:  
                print(f"Erro ao carregar '{tabela}': {e}")  
        else:  
            print(f"Aviso: tabela '{tabela}' não encontrada nos dados.")
```

O código apresentado garante que todas as tabelas presentes na lista `ordem_tabelas` são adicionadas à base de dados, utiliza a função `carregar_para_mysql_manual` para a inserção de todos os registos em cada tabela. Caso a inserção não seja concluída com sucesso, lança uma exceção na tabela referente. Se alguma tabela pertencente na lista da ordem de inserção, não estiver presente nos dados extraídos e transformados, é retornado um aviso a referir a tabela em questão.

### 6.1.8. Função principal do programa

```
def processo_Execucao():
    dados = extrair_csvs()
    dados_transformados = transformar_dados(dados)
    carregar_para_sql_ordenado(dados_transformados, db)

if __name__ == "__main__":
    processo_Execucao()
```

A função `processo_Execucao` coordena os processos a executar: Extração, transformação e Inserção dos dados presentes nos ficheiros CSV na base de dados BeloCars

### 6.1.9. Características do sistema

Vantagens do sistema utilizado:

- **Segurança:** O uso do `.env` separa as credenciais do código fonte
- **Automatização:** A migração dos dados é feita de forma automática permite reduzir os erros humanos e torna o processo constante
- **Normalização de dados:** reduz os erros de inserção e garante a consistência dos dados
- **Flexibilidade:** Em projetos futuros caso seja necessário a introdução de novas tabelas, apenas é necessário alterar a ordem das tabelas a inserir e as funções de extração e transformação
- **Dependências Relacionais:** Garante que todas as chaves estrangeiras sejam respeitadas evitando falhas de inserção

## 6.2. JSON

Esta secção documenta um *script Python* desenvolvido para carregar dados de ficheiros JSON e inseri-los numa base de dados MySQL de forma automatizada.

### 6.2.1. Importação de Bibliotecas e Configuração Inicial

```
import mysql.connector
import json
import os
from dotenv import load_dotenv

load_dotenv()
```

O código começa por importar as bibliotecas necessárias: `mysql.connector` para a conexão com a base de dados MySQL, `json` para processar ficheiros JSON, `os` para operações do sistema operativo, e `load_dotenv` para carregar variáveis de ambiente. A função `load_dotenv()` carrega as credenciais da base de dados a partir de um ficheiro `.env`, garantindo segurança ao não expor dados sensíveis no código sendo que este tipo de ficheiros vai ser diferente localmente.

### 6.2.2. Estabelecimento da Conexão com a Base de Dados

```
db = mysql.connector.connect(  
    host=os.getenv("HOST"),  
    user=os.getenv("USER"),  
    password=os.getenv("PASSWD"),  
    database=os.getenv("DATABASE"),  
    ssl_disabled=True  
)
```

```
cursor = db.cursor()
```

Estabelece-se a conexão com a base de dados MySQL utilizando as credenciais carregadas das variáveis de ambiente. O parâmetro `ssl_disabled=True` desativa a encriptação SSL, útil em ambientes de desenvolvimento. O cursor é criado para executar comandos SQL.

### 6.2.3. Definição da Diretoria dos Ficheiros Fonte

```
MODEL_DIR = os.path.join(os.path.dirname(__file__), "model")
```

Define-se o caminho para a diretoria “model” onde estão localizados os ficheiros JSON. Utiliza-se `os.path.join()` para garantir compatibilidade entre diferentes sistemas operativos.

### 6.2.4. Mapeamento das Funções de Inserção

```
insert_map = {  
    "localizacao": lambda l: cursor.execute(  
        "INSERT INTO Localizacao (ID, Endereco,CodigoPostal) VALUES (%s, %s, %s)",  
        (l["Id"], l["Endereco"], l["CodigoPostal"])  
    ),  
    "stand": lambda s: cursor.execute(  
        "INSERT INTO Stand (ID, Nome, NIF, ID_Localizacao) VALUES (%s, %s, %s, %s)",  
        (s["Id"], s["Nome"], s["NIF"], s["ID_Localizacao"])  
    ),  
    # ... outras funções de inserção  
}
```

Cria-se um dicionário que mapeia nomes de ficheiros para funções lambda que executam comandos SQL INSERT. Cada função lambda recebe um item dos dados JSON e executa o comando apropriado para inserir na tabela correspondente. Utilizam-se parâmetros preparados (%s) para prevenir injeção SQL.

### 6.2.5. Função Principal de Carregamento e Inserção

```
def load_and_insert():  
    for key, insert_fn in insert_map.items():  
        file_path = os.path.join(MODEL_DIR, f"{key}.json")  
        if not os.path.exists(file_path):  
            continue  
        with open(file_path, encoding="utf-8") as f:  
            try:  
                data = json.load(f)  
                for item in data.get(key, []):  
                    try:  
                        insert_fn(item)  
                    except mysql.connector.Error as e:  
                        print(f"Erro ao inserir em {key}: {e}")  
            except Exception as e:  
                print(f"Erro ao ler {key}.json: {e}")
```

Esta função itera sobre o mapeamento de inserções, constrói o caminho para cada ficheiro JSON, e processa-os se existirem. Para cada ficheiro, carrega os dados JSON, obtém a lista de itens associada à chave,

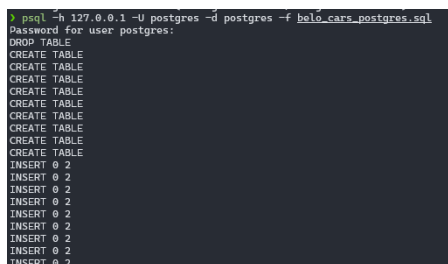
e tenta inserir cada item na base de dados usando a função de inserção correspondente. Implementa tratamento de erros robusto, capturando exceções específicas do MySQL e exceções gerais.

### 6.2.6. Função Principal do Programa

```
def main():
    try:
        load_and_insert()
        db.commit()
        print("Dados inseridos com sucesso!")
    except Exception as e:
        db.rollback()
        print("Erro geral:", e)
    finally:
        cursor.close()
        db.close()

if __name__ == "__main__":
    main()
```

A função `main()` orquestra todo o processo: chama `load_and_insert()`, confirma as transações com `commit()` se tudo correr bem, ou desfaz as alterações com `rollback()` em caso de erro. O bloco `finally` garante que a conexão com a base de dados é sempre fechada, independentemente do resultado. O `if __name__ == "__main__":` assegura que o código só executa quando o ficheiro é executado diretamente.



```
psql -h 127.0.0.1 -U postgres -d postgres -f holo_cars_postgres.sql
Password for user postgres:
DROP TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
INSERT 0 2
INSERT 0 2
INSERT 0 2
INSERT 0 2
INSERT 0 2
INSERT 0 2
INSERT 0 2
INSERT 0 2
```

Figura 46: Inserção de dados no postgres para Teste com leitura de ficheiros Json dados\_inseridos\_json4

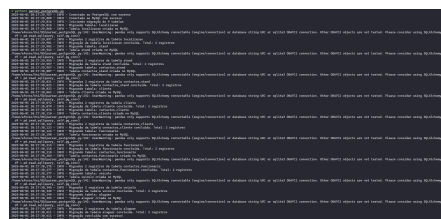


Figura 47: Migração de dados do postgres para o MySQL

### 6.2.7. Características do Sistema

Este sistema apresenta várias vantagens:

- **Segurança:** Utiliza variáveis de ambiente para credenciais e parâmetros preparados para prevenir injeção SQL;
- **Flexibilidade:** O mapeamento permite adicionar facilmente novas tabelas e tipos de dados;
- **Robustez:** Implementa tratamento abrangente de erros e gestão adequada de transações;
- **Escalabilidade e Modularização:** Código bem estruturado e documentado, facilitando futuras modificações.

## 6.3. PostgreSQL

Este documento explica um *script Python* desenvolvido para migrar dados de uma base de dados PostgreSQL para MySQL de forma automatizada, preservando a estrutura das tabelas e convertendo os tipos de dados apropriadamente.

### 6.3.1. Importação de Bibliotecas e Configuração de Logging

```
import psycpg2
import mysql.connector
import pandas as pd
from typing import List, Dict
import logging
import os
from dotenv import load_dotenv

logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)
logger = logging.getLogger(__name__)

load_dotenv()
```

Importam-se as bibliotecas essenciais para conectar às bases de dados e processar dados. O sistema de *logging* é configurado para fornecer informações detalhadas sobre o progresso da migração, incluindo *timestamps* e níveis de severidade das mensagens. A função `load_dotenv()` carrega as configurações a partir de um ficheiro `.env`, garantindo que credenciais sensíveis não são expostas no código fonte.

### 6.3.2. Classe *DatabaseMigrator* - Inicialização

```
class DatabaseMigrator:
    def __init__(self, postgres_config: Dict, mysql_config: Dict):
        self.postgres_config = postgres_config
        self.mysql_config = mysql_config
        self.pg_conn = None
        self.mysql_conn = None
```

A classe *DatabaseMigrator* encapsula toda a funcionalidade de migração. O construtor recebe as configurações de conexão para ambas as bases de dados e inicializa as variáveis de conexão como `None`, que serão estabelecidas posteriormente.

### 6.3.3. Estabelecimento de Conexões

```
def connect_databases(self):
    try:
        self.pg_conn = psycpg2.connect(**self.postgres_config)
        logger.info("Conectado ao PostgreSQL com sucesso")

        self.mysql_conn = mysql.connector.connect(**self.mysql_config)
        logger.info("Conectado ao MySQL com sucesso")
    except Exception as e:
        logger.error(f"Erro ao conectar às bases de dados: {e}")
        raise
```

Este método estabelece conexões simultâneas com ambas as bases de dados utilizando as configurações fornecidas. Implementa tratamento de erros robusto, registando sucessos e falhas através do sistema de *logging*.



### 6.3.4. Obtenção de Lista de Tabelas

```
def get_postgres_tables(self) -> List[str]:
    """Obtém lista de tabelas do PostgreSQL com nomes respeitando maiúsculas/
    minúsculas"""
    try:
        cursor = self.pg_conn.cursor()
        cursor.execute("""
            SELECT c.relname
            FROM pg_class c
            JOIN pg_namespace n ON n.oid = c.relnamespace
            WHERE c.relkind = 'r' AND n.nspname = 'public'
        """)
        tables = [row[0] for row in cursor.fetchall()]
        cursor.close()
        return tables
    except Exception as e:
        logger.error(f"Erro ao obter tabelas do PostgreSQL: {e}")
        raise
```

Esta função consulta o catálogo do sistema PostgreSQL para obter todas as tabelas do esquema público. Utiliza uma consulta específica que acede às tabelas de metadados `pg_class` e `pg_namespace` para garantir que apenas tabelas regulares são retornadas, preservando a sensibilidade a maiúsculas/minúsculas dos nomes.

### 6.3.5. Obtenção do Esquema das Tabelas

```
def get_table_schema(self, table_name: str) -> List[Dict]:
    try:
        cursor = self.pg_conn.cursor()
        cursor.execute(f"""
            SELECT column_name, data_type, is_nullable, column_default
            FROM information_schema.columns
            WHERE table_name = %s AND table_schema = 'public'
            ORDER BY ordinal_position
        """, (table_name,))

        columns = []
        for row in cursor.fetchall():
            columns.append({
                'name': row[0],
                'type': row[1],
                'nullable': row[2] == 'YES',
                'default': row[3]
            })
        cursor.close()
        return columns
    except Exception as e:
        logger.error(f"Erro ao obter esquema da tabela {table_name}: {e}")
        raise
```

Esta função extrai os metadados de uma tabela específica, incluindo nomes de colunas, tipos de dados, nulabilidade e valores padrão. Utiliza o `information_schema` padrão SQL para obter informações estruturais, ordenando as colunas pela sua posição original na tabela.

### 6.3.6. Conversão de Tipos de Dados

```
def convert_postgres_to_mysql_type(self, pg_type: str) -> str:
    type_mapping = {
        'integer': 'INT',
        'bigint': 'BIGINT',
        'smallint': 'SMALLINT',
```

```

        'numeric': 'DECIMAL',
        'real': 'FLOAT',
        'double precision': 'DOUBLE',
        'character varying': 'VARCHAR(255)',
        'varchar': 'VARCHAR(255)',
        'text': 'TEXT',
        'char': 'CHAR',
        'boolean': 'TINYINT(1)',
        'date': 'DATE',
        'timestamp without time zone': 'DATETIME',
        'timestamp with time zone': 'TIMESTAMP',
        'time': 'TIME',
        'bytea': 'BLOB'
    }

    if '(' in pg_type:
        base_type = pg_type.split('(')[0]
        if base_type in type_mapping:
            if base_type == 'character varying':
                return pg_type.replace('character varying', 'VARCHAR')
            return type_mapping[base_type]

    return type_mapping.get(pg_type, 'TEXT')

```

Esta função crítica mapeia tipos de dados PostgreSQL para os equivalentes MySQL. Inclui tratamento especial para tipos parametrizados (como VARCHAR com tamanho específico) e fornece um tipo padrão (TEXT) para casos não mapeados, garantindo compatibilidade máxima entre os sistemas.

### 6.3.7. Criação de Tabelas MySQL

```

def create_mysql_table(self, table_name: str, columns: List[Dict]):
    try:
        cursor = self.mysql_conn.cursor()
        column_definitions = []
        for col in columns:
            mysql_type = self.convert_postgres_to_mysql_type(col['type'])
            nullable = "NULL" if col['nullable'] else "NOT NULL"
            col_def = f" {col['name']} {mysql_type} {nullable}"
            column_definitions.append(col_def)

        create_sql = f"CREATE TABLE IF NOT EXISTS {table_name} (\n"
        create_sql += ",\n".join(column_definitions)
        create_sql += "\n) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4"

        cursor.execute(create_sql)
        self.mysql_conn.commit()
        cursor.close()
        logger.info(f"Tabela {table_name} criada no MySQL")
    except Exception as e:
        logger.error(f"Erro ao criar tabela {table_name} no MySQL: {e}")
        raise

```

### 6.3.8. Criação de Tabelas MySQL

```

def create_mysql_table(self, table_name: str, columns: List[Dict]):
    try:
        cursor = self.mysql_conn.cursor()
        column_definitions = []
        for col in columns:
            mysql_type = self.convert_postgres_to_mysql_type(col['type'])
            nullable = "NULL" if col['nullable'] else "NOT NULL"

```

```

col_def = f" {col['name']} {mysql_type} {nullable}"
column_definitions.append(col_def)

create_sql = f"CREATE TABLE IF NOT EXISTS {table_name} (\n"
create_sql += ",\n".join(column_definitions)
create_sql += "\n) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4"

cursor.execute(create_sql)
self.mysql_conn.commit()
cursor.close()
logger.info(f"Tabela {table_name} criada no MySQL")
except Exception as e:
    logger.error(f"Erro ao criar tabela {table_name} no MySQL: {e}")
    raise

```

Este método constrói e executa comandos DDL (*Data Definition Language*) para recriar a estrutura das tabelas no MySQL. Remove tabelas existentes, aplica conversões de tipos, preserva restrições de nulabilidade, e configura o motor InnoDB (motor convencional mais usado) com codificação UTF-8 para suporte completo a caracteres *Unicode*.

### 6.3.9. Migração de Dados por Lotes

```

def migrate_table_data(self, table_name: str, batch_size: int = 1000):
    try:
        offset = 0
        total_rows = 0

        while True:
            query = f'SELECT * FROM "{table_name}" LIMIT {batch_size} OFFSET {offset}'
            df = pd.read_sql(query, self.pg_conn)
            if df.empty:
                break

            cursor = self.mysql_conn.cursor()

            columns = list(df.columns)
            placeholders = ', '.join(['%s'] * len(columns))
            insert_sql = f"INSERT INTO `{table_name}` ({', '.join([f'`{col}`' for col in
columns])}) VALUES ({placeholders})"

            data = []
            for _, row in df.iterrows():
                row_data = []
                for value in row:
                    if pd.isna(value):
                        row_data.append(None)
                    elif isinstance(value, pd.Timestamp):
                        row_data.append(value.to_pydatetime())
                    else:
                        row_data.append(value)
                data.append(tuple(row_data))

            cursor.executemany(insert_sql, data)
            self.mysql_conn.commit()
            cursor.close()

            total_rows += len(df)
            offset += batch_size
            logger.info(f"Migrados {total_rows} registros da tabela {table_name}")

        logger.info(f"Migração da tabela {table_name} concluída. Total: {total_rows}")

```

```
registros")
    except Exception as e:
        logger.error(f"Erro ao migrar dados da tabela {table_name}: {e}")
        raise
```

Esta função implementa migração de dados por lotes para otimizar performance e gestão de memória. Processa dados em *chunks* de tamanho configurável, converte tipos especiais (como *timestamps*), trata valores nulos adequadamente, e fornece *feedback* contínuo sobre o progresso através de *logging*.

#### 6.3.10. Orquestração da Migração Completa

```
def migrate_database(self, tables: List[str] = None, batch_size: int = 1000):
    try:
        self.connect_databases()

        if tables is None:
            tables = self.get_postgres_tables()

        logger.info(f"Iniciando migração de {len(tables)} tabelas")

        for table_name in tables:
            logger.info(f"Migrando tabela: {table_name}")

            schema = self.get_table_schema(table_name)

            self.create_mysql_table(table_name, schema)

            self.migrate_table_data(table_name, batch_size)

        logger.info("Migração concluída com sucesso!")
    except Exception as e:
        logger.error(f"Erro durante a migração: {e}")
        raise
    finally:
        self.close_connections()
```

Este método coordena todo o processo de migração: estabelece conexões, determina quais tabelas migrar, e executa sequencialmente a criação de esquemas e migração de dados para cada tabela. O bloco *finally* garante que as conexões são sempre fechadas, mesmo em caso de erro.

#### 6.3.11. Gestão de Conexões

```
def close_connections(self):
    if self.pg_conn:
        self.pg_conn.close()
        logger.info("Conexão PostgreSQL fechada")

    if self.mysql_conn:
        self.mysql_conn.close()
        logger.info("Conexão MySQL fechada")
```

Método responsável por fechar adequadamente todas as conexões de base de dados, evitando vazamento de recursos e registrando as ações através do sistema de *logging*.

#### 6.3.12. Função Principal com Configurações Seguras

```
def main():
    postgres_config = {
        'host': os.getenv('PG_HOST', '127.0.0.1'),
        'port': int(os.getenv('PG_PORT', '5432')),
        'database': os.getenv('PG_DATABASE', 'postgres'),
        'user': os.getenv('PG_USER', 'postgres'),
```

```

        'password': os.getenv('PG_PASSWORD')
    }

mysql_config = {
    'host': os.getenv('MYSQL_HOST', '127.0.0.1'),
    'port': int(os.getenv('MYSQL_PORT', '3306')),
    'database': os.getenv('MYSQL_DATABASE', 'BeloCars'),
    'user': os.getenv('MYSQL_USER', 'root'),
    'password': os.getenv('MYSQL_PASSWORD'),
    'autocommit': False
}

migrator = DatabaseMigrator(postgres_config, mysql_config)

try:
    migrator.migrate_database()
except Exception as e:
    logger.error(f"Falhou a migração: {e}")

if __name__ == "__main__":
    main()

```

A função `main()` agora carrega as configurações das bases de dados a partir de variáveis de ambiente usando `os.getenv()`. Esta abordagem fornece valores padrão para parâmetros não críticos (como *host* e porta) mas requer que as *passwords* sejam definidas no ficheiro `.env`. O parâmetro `autocommit=False` no MySQL garante controlo transaccional adequado.

### 6.3.13. Ficheiro de Configuração (.env)

```

# Configurações PostgreSQL
PG_HOST=127.0.0.1
PG_PORT=5432
PG_DATABASE=postgres
PG_USER=postgres
PG_PASSWORD=sua_password_postgresql

# Configurações MySQL
MYSQL_HOST=127.0.0.1
MYSQL_PORT=3306
MYSQL_DATABASE=BeloCars
MYSQL_USER=root
MYSQL_PASSWORD=sua_password_mysql

```

O ficheiro `.env` deve ser criado na raiz do projeto contendo todas as configurações de conexão. Este ficheiro não deve ser partilhado em nenhum controlador de versões ou a ninguém que possa ter intenções maliciosas, pois pode comprometer a segurança e integridade dos dados.

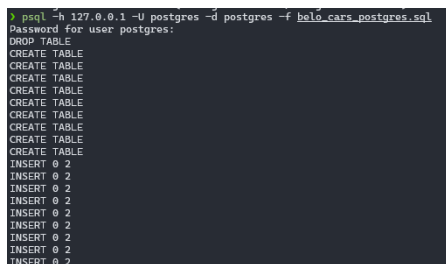


Figura 48: Inserção de dados no postgres para teste

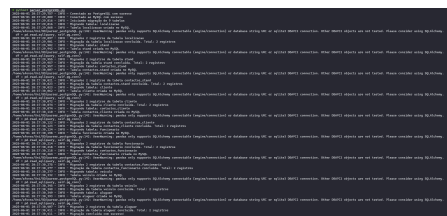


Figura 49: Migração de dados do postgres para o MySQL

### 6.3.14. Características do Sistema de Migração

Este sistema apresenta várias vantagens técnicas:

- **Escalabilidade:** Processamento por lotes evita problemas de memória com tabelas grandes;
- **Robustez:** Tratamento abrangente de erros e *logging* detalhado;
- **Flexibilidade:** Permite migração seletiva de tabelas específicas;
- **Compatibilidade:** Mapeamento completo de tipos entre PostgreSQL e MySQL;
- **Segurança:** Gestão adequada de transações e recursos.

## 7. Conclusões Críticas

### 7.1. Dificuldades Sentidas e Estratégias de Superação

No decorrer de todo o processo, tentamos validar a nossa resolução do problema com a equipa docente (tanto o professor regente como os professores auxiliares), para ter uma noção de que estávamos a ir em direção ao caminho mais acertado. Nessas situações percebemos que as nossas maiores dúvidas e inconsistências estavam na definição dos Requisitos certos. Percebemos rapidamente que sem os requisitos certos por onde nos guiar não tínhamos uma base sólida para continuar o processo de Modelação Conceitual, e muito menos a Modelação Lógica. Este foi o aspeto que nos deu mais trabalho a “limar” para que ficasse o mais adequado possível; como tudo neste trabalho poderia ser ainda refinado e otimizado, mas podemos dizer que superamos essa dificuldade e temos agora uma base sólida para o problema, e por isso, conseguimos ter uma implementação bem sucedida e coerente com o pedido.

Podemos ainda ressaltar, o facto deste trabalho envolver o manuseio de ferramentas novas para todos nós como o **BrModelo** e o **MySQL Workbench**, que se tornaram um desafio interessante devido à exploração e aprendizagem que tivemos que adquirir para entregar o melhor projeto possível.

### 7.2. Alterações em relação à 1ª Fase

A equipa entregou a 21 de abril, uma primeira fase do Projeto contendo apenas a realização do processo até à Modelação Lógica. A realização posterior da consequente Implementação Física não se revelou isenta de pequenas alterações relativas a essa primeira entrega. Dessas alterações podemos enumerar as seguintes:

- Como já referido na secção anterior, os Requisitos foram alvos de algumas adições e remoções, tais como: acrescentou-se um requisito de Manipulação referente ao Sistema de Migração de Dados através de 3 fontes diferente de dados, relativo às três localizações diferentes da empresa (aspeto que não tínhamos incluído na primeira versão do trabalho); e a remoção de requisitos que achamos que estavam fora do escopo do projeto, nomeadamente um requisito de controlo referente ao manuseio da base de dados por parte dos Clientes (através de utilizadores específicos que iríamos criar), mas que consideramos não ser apropriado para a aplicação deste sistema;
- Acrescentou-se também as Tabelas de Descrição relativas às entidades do nosso modelo, aspeto este que não tínhamos incluído na versão anterior mas que consideramos incluir na presente iteração (adicionado na secção de **Identificação e Caracterização das Entidades/Relacionamentos**).

### 7.3. Análise do Diagrama de Gantt Real vs Planeado

No início do projeto foi elaborado um diagrama de Gantt (Figura 44) de forma a planear a execução do trabalho. Este diagrama espelhava a nossa expectativa de tempos para realização de cada tarefa/tópico, tendo em conta os prazos que tínhamos e o pouco conhecimento que tínhamos à priori da prática deste processo de desenvolvimento de um Sistema de Base de Dados Relacional.

Ao longo do trabalho, tentámos o mais rigorosamente possível apontar as datas em que terminávamos cada tópico, o mais fiel possível à realidade.

Agora numa fase posterior possuímos Definição de Requisitos foi a que mais divergiu do planeamento e, como já dito anteriormente, foi uma das fases em que tivemos mais dúvidas e que quisemos ter a certeza que estaríamos no caminho mais acertado. Devido à natureza decisiva desta fase, passamos um tempo considerável a refinar os requisitos para que estivessem na sua melhor forma, pois compreendemos que os requisitos são as regras base de tudo o que iríamos desenvolver para a frente;

- As seguintes fases de trabalho foram mais curtas do que o planeado. Acreditamos que por termos investido tempo nos requisitos isso levou a que tivéssemos menos problemas nas fases posteriores, e isso levou a que conseguíssemos terminar a maioria das tarefas seguintes num menor tempo em comparação àquilo que esperávamos.

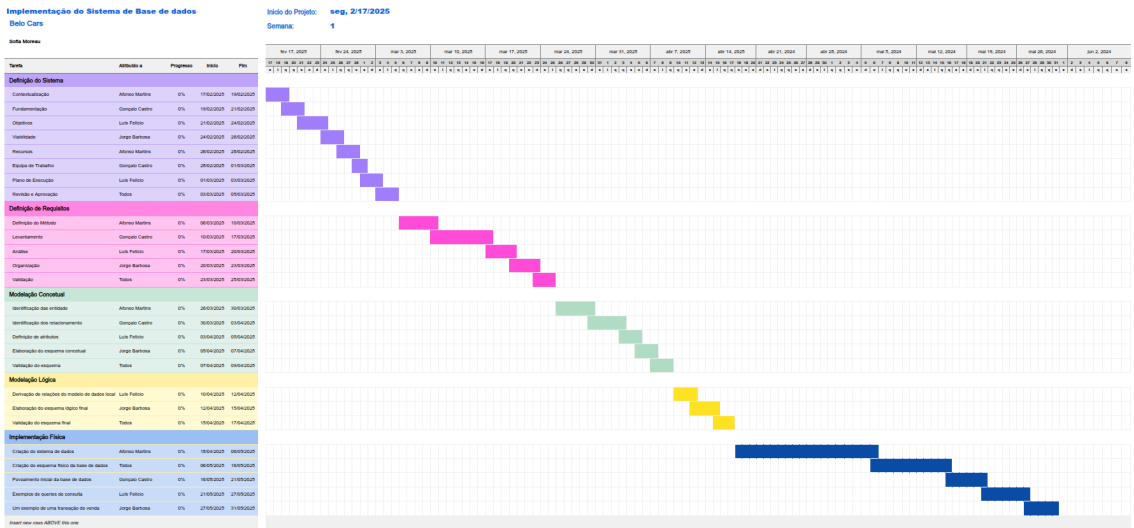


Figura 50: Diagrama de Gantt Planeado

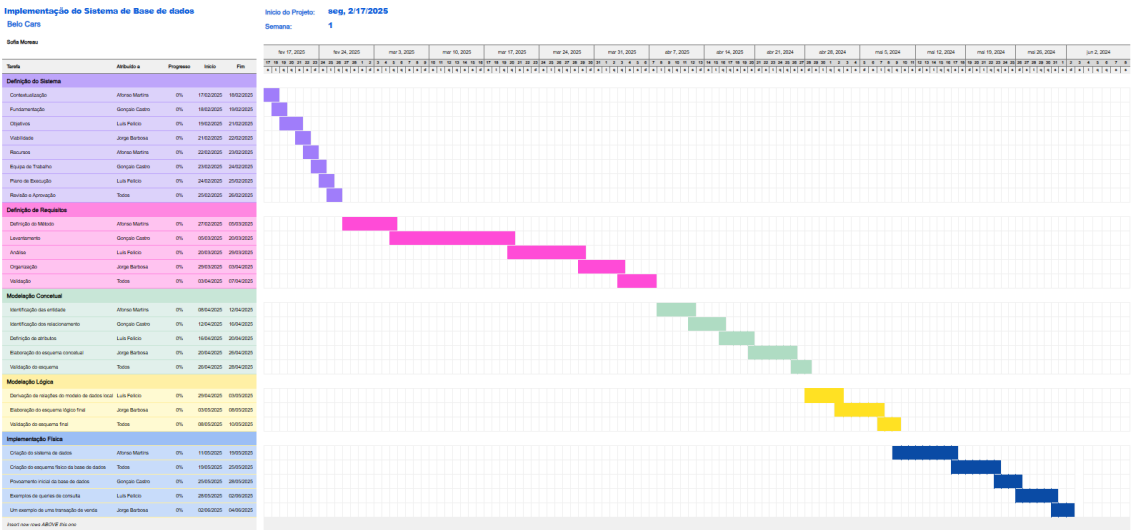


Figura 51: Diagrama de Gantt Final (preenchido ao longo do desenvolvimento do projeto)

## 7.4. Trabalho Futuro

Apesar de termos conseguido desenvolver uma solução sólida e coerente com os objetivos propostos, existe sempre espaço para melhorias e evoluções, tanto a nível técnico como conceptual.

- **Revisão e Normalização Avançada:** Poderíamos explorar uma normalização mais rigorosa das tabelas, especialmente em atributos como Marca, Modelo e Função, criando tabelas auxiliares para reduzir redundância e facilitar a manutenção a longo prazo;



- **Melhoria de Performance com Base em Dados Reais:** A análise do crescimento foi feita com base em estimativas. Com dados reais e históricos de utilização, seria possível afinar os índices criados, removendo alguns desnecessários e criando outros mais focados em *queries* frequentes detetadas por monitorização e uso num contexto “real”.
- **Integração de Mecanismos de Segurança:** Apesar de termos estruturado bem as tabelas e as suas ligações, a segurança ao nível de acessos e permissões de utilizadores poderia ser mais profundamente explorada. Poderíamos estudar o uso de *roles*, privilégios específicos e até separação por *schema* para ambientes de produção, teste e desenvolvimento.
- **Automatização de Backups e Recuperação:** Para garantir resiliência do sistema, será importante planear e implementar mecanismos automáticos de *backup* e restauro, com frequência e armazenamento adequados ao volume de dados esperado.
- **Documentação Técnica e Manual de Utilizador:** O projeto beneficiaria da criação de documentação mais detalhada, quer técnica (*scripts*, relacionamentos, lógica de negócio), quer para utilizadores finais (como interagir com o sistema, regras de negócio, entre outros...).
- **Simulação e Testes de Escalabilidade:** Poderiam ser realizados testes de stress para verificar como a base de dados se comporta com grandes volumes de dados, especialmente em cenários simulados de crescimento anual — validando a estrutura proposta e os índices definidos.
- **Possível Migração para Arquitetura Distribuída:** Caso o projeto cresça significativamente (por exemplo, expansão para novos países ou mais localizações), seria relevante pensar numa arquitetura distribuída ou em *sharding*, a fim de garantir escalabilidade horizontal.
- **Exploração de Ferramentas de BI:** A implementação de *dashboards* com ferramentas como *Power BI* ou *Grafana* sobre a base de dados permitiria extrair valor analítico dos dados armazenados, útil para decisões estratégicas da empresa.

## 8. Referência Bibliográfica

1. Chen, P. (1976, março). The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems, 1(1), 9–36. <https://doi.org/10.1145/320434.320440>
2. Sis4. (s.d.). Modelo BR. Obtido de <http://www.sis4.com/brModelo/>
3. Chen, P. (2002). Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned [PDF]. Em Software Pioneers (pp. 296–310). Springer-Verlag. ISBN 978-3-540-43081-0.
4. MySQL. (s.d.). MySQL Workbench. Obtido de <https://www.mysql.com/products/workbench/>
5. Kessler, J. (2022, setembro). Relax. Obtido de <https://dbis-uibk.github.io/relax/landing>
6. Fuchs, M. (2021, 3 de março). Entity-Relationship Diagram (ERD). Obtido de <https://michael-fuchs-sql.netlify.app/2021/03/03/entity-relationship-diagram-erd/>
7. MySQL (s.d.). Documentação SQL. Obtido de <https://dev.mysql.com/doc/>

## Lista de Siglas e Acrónimos

**SBD** Sistema de Base de Dados

**RD** Requisito de Descrição

**RM** Requisito de Manipulação

**RC** Requisito de Controlo

# Anexos

## Anexo 1: Requisitos Gerais Pt.1      Anexo 2: Requisitos Gerais Pt.2

Belo Cars

Documento de Recolha de Requisitos

Desenvolvimento de um Sistema de Base de Dados

Março de 2025

Levantamento Geral:

Levantamento de Requisitos					
Nº	Data	Descrição	Área	Fonte	Analista
1	4 de março	Cada veículo contém um identificador único(matrícula), marca, modelo, ano de fabrico, preço diário, o consumo, o nº de passageiros que transporta, o tipo de combustível, o tipo (Mercadoria ou Passageiros) e estado atual (se está disponível ou não)	Veículos	Sofia Moreau	Gonçalo Castro
2	4 de março	Se um Veículo for do tipo mercadoria deverá ainda possuir a tara (peso quando vazio) e a carga máxima	Veículos		Afonso Martins
3	4 de março	Cada cliente denota um identificador único, nome, data de nascimento, contacto (email e telemóvel) e nº de contribuinte (NIF)	Clientes	Sofia Moreau	Gonçalo Castro
4	4 de março	Cada Aluguer possui um identificador único (número sequencial), data de início, data de fim, valor da transação e o método de pagamento	Aluguer	Sofia Moreau	Luís Felício
5	4 de março	Cada funcionário possui um identificador único, nome, contactos (email e telefone), departamento e a sua função no departamento.	Funcionários	Sofia Moreau	Jorge Barbosa
6	4 de março	Os dados acerca dos carros podem ser consultados por qualquer pessoa	Veículos	Sofia Moreau	Gonçalo Castro
7	4 de março	Os dados acerca dos clientes podem ser acedidos pelo próprio ou por um Funcionário mediante a inserção do identificador	Clientes	Sofia Moreau	Afonso Martins
8	4 de março	Os dados dos funcionários podem ser acedidos pelo próprio ou por um superior (admin)	Funcionários	Sofia Moreau	Luís Felício
9	9 de março	Os veículos podem ser filtrados por qualquer uma das especificações: marca, modelo, ano de fabrico, preço diário, o consumo, o nº de passageiros, o tipo de combustível, o tipo do veículo e disponibilidade	Veículos	Sofia Moreau	Jorge Barbosa
10	9 de março	Os funcionários também possuem autorização para editar todos os alugueres mediante a introdução do ID	Funcionários / Alugueres	Sofia Moreau	Gonçalo Castro

11	9 de março	Os alugueres podem ser filtrados por: intervalo de tempo (dado uma data de início e data de fim), intervalo de valores de pagamentos e por método de pagamento.	Alugueres	Sofia Moreau	Luís Felício
12	9 de março	Cada Aluguer tem de estar associado a um e um só cliente, mas um cliente pode ter 0 ou vários alugueres	Alugueres / Clientes	Sofia Moreau	Jorge Barbosa
13	10 de março	Cada Aluguer possui um e um só veículo, mas um veículo pode estar em 0 ou mais alugueres.	Alugueres / Veículos	Sofia Moreau	Gonçalo Castro
14	10 de março	Cada aluguer tem de ter associado um funcionário, um funcionário pode ter vários alugueres ou não possuir qualquer aluguer associado	Alugueres / Funcionários	Sofia Moreau	Afonso Martins
15	10 de março	Cada Stand inclui um identificador único, uma localização (que possui endereço e código postal) e contactos (email e telefone)	Stands	Sofia Moreau	Luís Felício
16	10 de março	Um Stand tem de ter um ou mais veículos a ele associados, mas um veículo só pode pertencer a um stand	Stands / Veículos	Sofia Moreau	Jorge Barbosa
17	25 de março	Os carros podem ser transferidos para outro stand pelos funcionários	Veículos / Funcionários	Sofia Moreau	Gonçalo Castro
18	25 de março	Tem de ser possível a listagem de todos os alugueres efetuados por um cliente mediante a introdução do seu id (incluindo o id, a marca e o modelo do veículo associado) e, opcionalmente, um intervalo de datas limite	Alugueres / Clientes	Sofia Moreau	Afonso Martins
19	25 de março	Listar os alugueres supervisionados por um dado funcionário, dado o id do funcionário e, opcionalmente, um intervalo de datas	Alugueres / Funcionários	Sofia Moreau	Luís Felício
20	25 de março	Consulta de todos os veículos de um dado stand que estão disponíveis para alugar	Veículos / Stands	Sofia Moreau	Jorge Barbosa
21	25 de março	Calcular o valor da receita num dado dia, associado a todos as transações efetuadas nesse dia	Alugueres	Sofia Moreau	Afonso Martins
22	25 de março	Listar por ordem decrescente os funcionários que mais efetuaram alugueres para monitorizar o desempenho	Funcionários / Alugueres	Sofia Moreau	Gonçalo Castro
23	25 de março	Cada stand pode empregar um ou mais funcionários. Cada funcionário está associado apenas e unicamente a um stand	Stand / Funcionários	Sofia Moreau	Afonso Martins
24	3 de maio	Desenvolvimento de um sistema de migração de dados, sendo que as 3 localizações possuem formatos diferentes (JSON, CSV e Postgres)	Migração	Sofia Moreau	Luís Felício

## Anexo 3: Tabela de Requisitos de Descrição

### Organização e Categorização:

Requisitos de Descrição						
Nº		Data	Descrição	Área	Fonte	Revisor
RD01	1	4 de março	Cada veículo contém um identificador único(matrícula), marca, modelo, ano de fabrico, preço diário, o consumo, o nº de passageiros que transporta, o tipo de combustível, o tipo (Mercadoria ou Passageiros) e estado atual (se está disponível ou não)	Veículos	Sofia Moreau	Gonçalo Castro
RD02	2	4 de março	Se um Veículo for do tipo mercadoria deverá ainda possuir a tara (peso quando vazio) e a carga máxima	Veículos		Afonso Martins
RD03	3	4 de março	Cada cliente denota um identificador único, nome, data de nascimento, contacto (email e telemóvel) e nº de contribuinte (NIF)	Clientes	Sofia Moreau	Gonçalo Castro
RD04	4	4 de março	Cada Aluguer possui um identificador único (número sequencial), data de início, data de fim, valor da transação e o método de pagamento	Aluguer	Sofia Moreau	Luís Felício
RD05	5	4 de março	Cada funcionário possui um identificador único, nome, contactos (email e telefone), departamento e a sua função no departamento.	Funcionários	Sofia Moreau	Jorge Barbosa
RD06	15	10 março	Cada Stand inclui um identificador único, uma localização (que possui endereço e código postal) e contactos (email e telefone)	Stands	Sofia Moreau	Luís Felício

## Anexo 4: Tabela de Requisitos de Manipulação

Requisitos de Manipulação					
Nº		Data	Descrição	Área	Fonte Revisor
RM01	9	9 de março	Os veículos podem ser filtrados por qualquer uma das especificações: marca, modelo, ano de fabrico, preço diário, o consumo, o nº de passageiros, o tipo de combustível, o tipo do veículo e disponibilidade	Veículos	Sofia Moreau Jorge Barbosa
RM02	11	9 de março	Os alugueres podem ser filtrados por: intervalo de tempo (dado uma data de início e data de fim), intervalo de valores de pagamentos e por método de pagamento.	Alugueres	Sofia Moreau Luís Felício
RM03	18	25 março	Tem de ser possível a listagem de todos os alugueres efetuados por um cliente mediante a introdução do seu id (incluindo o id, a marca e o modelo do veículo associado) e, opcionalmente, um intervalo de datas limite	Alugueres / Clientes	Sofia Moreau Afonso Martins
RM04	19	25 março	Listar os alugueres supervisionados por um dado funcionário, dado o id do funcionário e, opcionalmente, um intervalo de datas	Alugueres / Funcionários	Sofia Moreau Luís Felício
RM05	20	25 março	Consulta de todos os veículos de um dado stand que estão disponíveis para alugar	Veículos / Stands	Sofia Moreau Jorge Barbosa
RM06	21	25 março	Calcular o valor da receita num dado dia, associado a todos as transações efetuadas nesse mesmo dia	Alugueres	Sofia Moreau Afonso Martins
RM07	22	25 março	Listar por ordem decrescente os funcionários que mais efetuaram alugueres para monitorizar o desempenho	Funcionários / Alugueres	Sofia Moreau Gonçalo Castro
RM08	24	3 de maio	Desenvolvimento de um sistema de migração de dados tendo em conta que as 3 localizações possuem formatos diferentes (JSON, .CSV e Postgres)	Migração	Sofia Moreau Luís Felício

## Anexo 5: Tabela de Requisitos de Controlo

Requisitos de Controlo						
Nº		Data	Descrição	Área	Fonte	Revisor
RC01	6	4 de março	Os dados acerca dos carros podem ser consultados por qualquer pessoa (Convidado)	Veículos	Sofia Moreau	Gonçalo Castro
RC02	7	4 de março	Os dados acerca dos clientes podem ser acedidos por um Funcionário mediante a inserção do identificador	Cientes	Sofia Moreau	Afonso Martins
RC03	8	4 de março	Os dados dos funcionários podem ser acedidos pelo próprio ou por um superior (admin)	Funcionários	Sofia Moreau	Luís Felício
RC04	10	9 de março	Os funcionários também possuem autorização para editar todos os alugueres mediante a introdução do ID	Funcionários / Alugueres	Sofia Moreau	Gonçalo Castro
RC05	12	9 de março	Cada Aluguer tem de estar associado a um e um só cliente, mas um cliente pode ter 0 ou vários alugueres	Alugueres / Cientes	Sofia Moreau	Jorge Barbosa
RC06	13	10 março	Cada Aluguer possui um e um só veículo, mas um veículo pode estar em 0 ou mais alugueres.	Alugueres / Veículos	Sofia Moreau	Gonçalo Castro
RC07	14	10 março	Cada aluguer tem de ter associado um funcionário, um funcionário pode ter vários alugueres ou não possuir qualquer aluguer associado	Alugueres / Funcionários	Sofia Moreau	Afonso Martins
RC08	16	10 março	Um Stand tem de ter um ou mais veículos a ele associados, mas um veículo só pode pertencer a um stand	Stands / Veículos	Sofia Moreau	Jorge Barbosa
RC09	17	25 março	Os carros podem ser transferidos para outro stand pelos funcionários	Veículos / Funcionários	Sofia Moreau	Gonçalo Castro
RC10	23	25 de março	Cada stand pode empregar um ou mais funcionários. Cada funcionário está associado apenas e unicamente a um stand	Stand / Funcionários	Sofia Moreau	Afonso Martins

## Anexo 6: Primeira Página do Questionário que serviu de inquérito aos funcionários da Belo Cars

Questionário Interno – Recolha de Opiniões e Sugestões (Funcionários *Belo Cars*)

Este formulário tem como objetivo recolher a opinião dos colaboradores sobre o funcionamento da empresa e obter sugestões para melhorar os processos internos, a comunicação e a experiência de trabalho.

[Seguinte](#) [Limpar formulário](#)

Nunca envie palavras-passe através dos Google Forms.

Este conteúdo não foi criado nem aprovado pela Google - [Termos de Utilização](#) - [Política de privacidade](#)

Este formulário parece suspeito? [Relatar](#)

Google Formulários

## Anexo 7: Segunda Página do Questionário que serviu de inquérito aos funcionários da Belo Cars

Questionário Interno – Recolha de Opiniões e Sugestões (Funcionários *Belo Cars*)

\* Indica uma pergunta obrigatória

**Secção 1: Dados Gerais (opcional)**

Nome

A sua resposta

Departamento em que trabalha \*

Selecionar

Função / Cargo atual \*

A sua resposta

Tempo na Empresa \*

☐ Menos de 1 ano

☐ 1 a 3 anos

☐ 4 a 6 anos

☐ Mais de 6 anos

[Anterior](#) [Seguinte](#) [Limpar formulário](#)

## Anexo 8: Terceira Página do Questionário que serviu de inquérito aos funcionários da Belo Cars

Questionário Interno – Recolha de Opiniões e Sugestões (Funcionários *Belo Cars*)

\* Indica uma pergunta obrigatória

**Avaliação dos Processos Atuais**

Como avalia os atuais processos de reserva e gestão de veículos? \*

1 2 3 4 5

Muito Ineficiente ☐ ☐ ☐ ☐ ☐ Muito Eficiente

Quais são os maiores desafios que enfrenta no dia a dia do seu trabalho? \*

A sua resposta

Sente que os recursos disponíveis são suficientes para executar bem o seu trabalho? \*

☐ Sim

☐ Em parte

☐ Não

☐ Não sei / Prefiro não responder

[Anterior](#) [Seguinte](#) [Limpar formulário](#)

## Anexo 9: Quarta Página do Questionário que serviu de inquérito aos funcionários da Belo Cars

Questionário Interno – Recolha de Opiniões e Sugestões (Funcionários *Belo Cars*)

\* Indica uma pergunta obrigatória

**Sugestões e Inovação**

Que melhorias gostaria de ver implementadas no novo sistema de gestão (base de dados, reservas, etc...)? \*

A sua resposta

Há algum processo específico que considera desnecessário ou que poderia ser simplificado? \*

A sua resposta

Tem sugestões para melhorar a comunicação entre departamentos?

A sua resposta

Outras observações ou sugestões gerais

A sua resposta

[Anterior](#) [Seguinte](#) [Limpar formulário](#)



# Anexo 10: Quinta Página do Questionário que serviu de inquérito aos funcionários da Belo Cars

Questionário Interno – Recolha de Opiniões e Sugestões (Funcionários *Belo Cars*)

\* Indica uma pergunta obrigatória

Secção Final

Gostaria de ser contactado para esclarecer melhor as suas respostas ou aprofundar as suas ideias? \*

☐ Sim

☐ Não

Se respondeu 'Sim' acima, indique o seu email de contacto

A sua resposta

Anterior

Enviar

Limpar formulário

# Anexo 11: Primeira Página da Ata de uma das Reuniões com a Sofia Moreau

ATA DE REUNIÃO

Projeto: Implementação do Sistema de Base de Dados da Belo Cars

Data: 17 de março de 2025

Hora: 10h00 – 11h30

Local: Sala de Reuniões Principal, Sede da Belo Cars (Lisboa)

Presenças:

- Sofia Moreau (CEO e Coordenadora do Projeto)

- Gonçalo Castro (Responsável pela Modelação de Relacionamentos e Levantamento de Requisitos)

- Afonso Martins (Responsável pela Contextualização e Entidades)

- Luis Felício (Responsável pelos Atributos e Esquema Lógico)

- Jorge Barbosa (Responsável pela Viabilidade Técnica e Esquema Conceptual)

1. Abertura da Reunião

A reunião foi iniciada por Sofia Moreau, que reforçou o compromisso da Belo Cars com a modernização tecnológica e agradeceu o esforço da equipa no progresso já realizado até à fase de definição de requisitos.

2. Ponto de Situação do Projeto

- Afonso Martins apresentou a Contextualização e confirmou que a descrição histórica da empresa foi finalizada.

- Gonçalo Castro explicou o progresso no Levantamento de Requisitos, destacando a necessidade de um sistema acessível remotamente, com foco em reservas online.

- Luis Felício destacou a importância de [normalização](#) de dados e propôs um modelo inicial para atributos dos clientes e veículos.

- Jorge Barbosa referiu que a viabilidade técnica foi validada com base em requisitos de escalabilidade e segurança.

3. Deliberações

Sofia Moreau reforçou que o novo sistema deve permitir:

- Consulta de reservas em tempo real.

- Gestão descentralizada das frotas de cada stand.

## Anexo 12: Segunda Página da Ata de uma das Reuniões com a Sofia Moreau

- Relatórios automáticos para análise de desempenho e previsões de utilização.

Foi decidido que o modelo conceptual deve estar completo até 7 de abril de 2025, incluindo todos os relacionamentos e regras de negócio.

A equipa de TI irá validar a infraestrutura atual e propor melhorias até ao final de março.

**4. Ações a Desenvolver**

Tarefa	Responsável	Prazo
Concluir levantamento funcional	Gonçalo Castro	25/03/2025
Apresentar rascunho do modelo ER	Jorge Barbosa	01/04/2025
Rever atributos e normalização	Luís Felício	02/04/2025
Validar compatibilidade técnica	Afonso Martins	28/03/2025

**5. Encerramento**

A reunião foi encerrada às 11h30 com palavras de motivação da CEO:

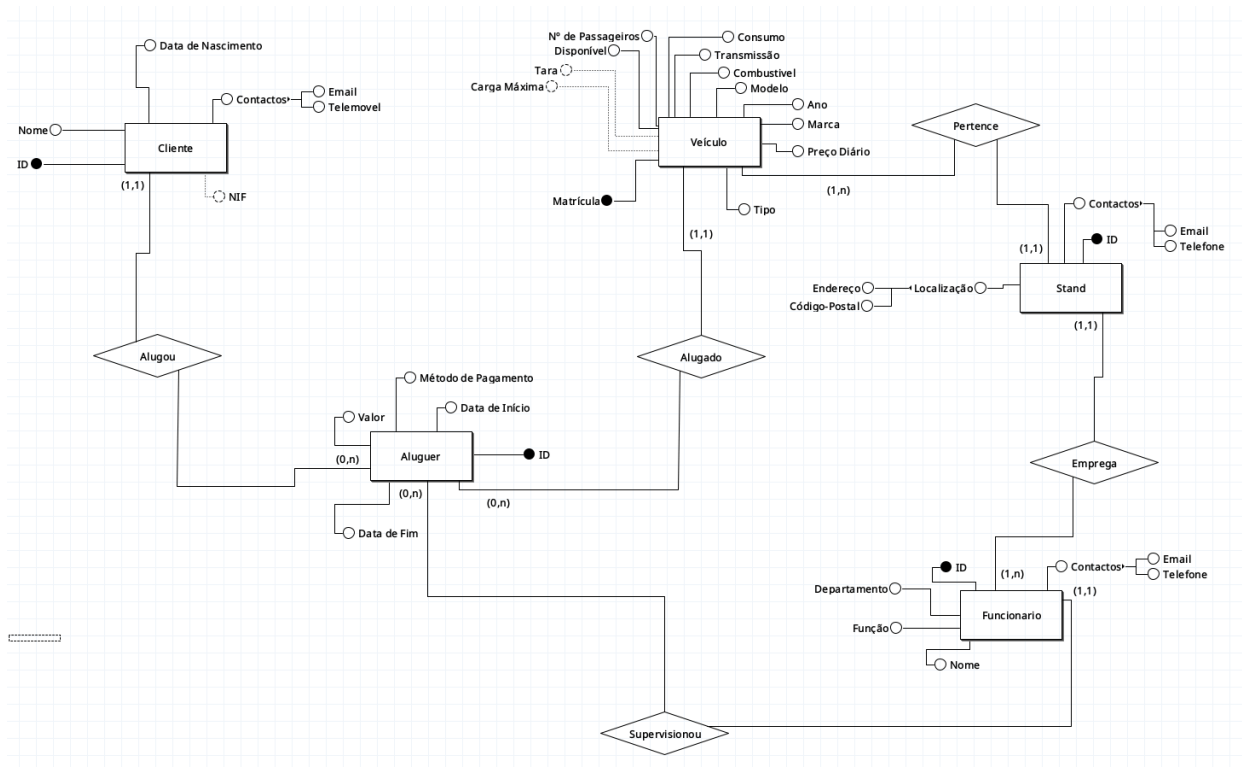
"A Beo Cars nasceu de uma ideia simples: mobilidade com confiança. Este projeto é mais do que uma atualização — é a base do nosso futuro digital." — Sofia Moreau

**Próxima Reunião:**

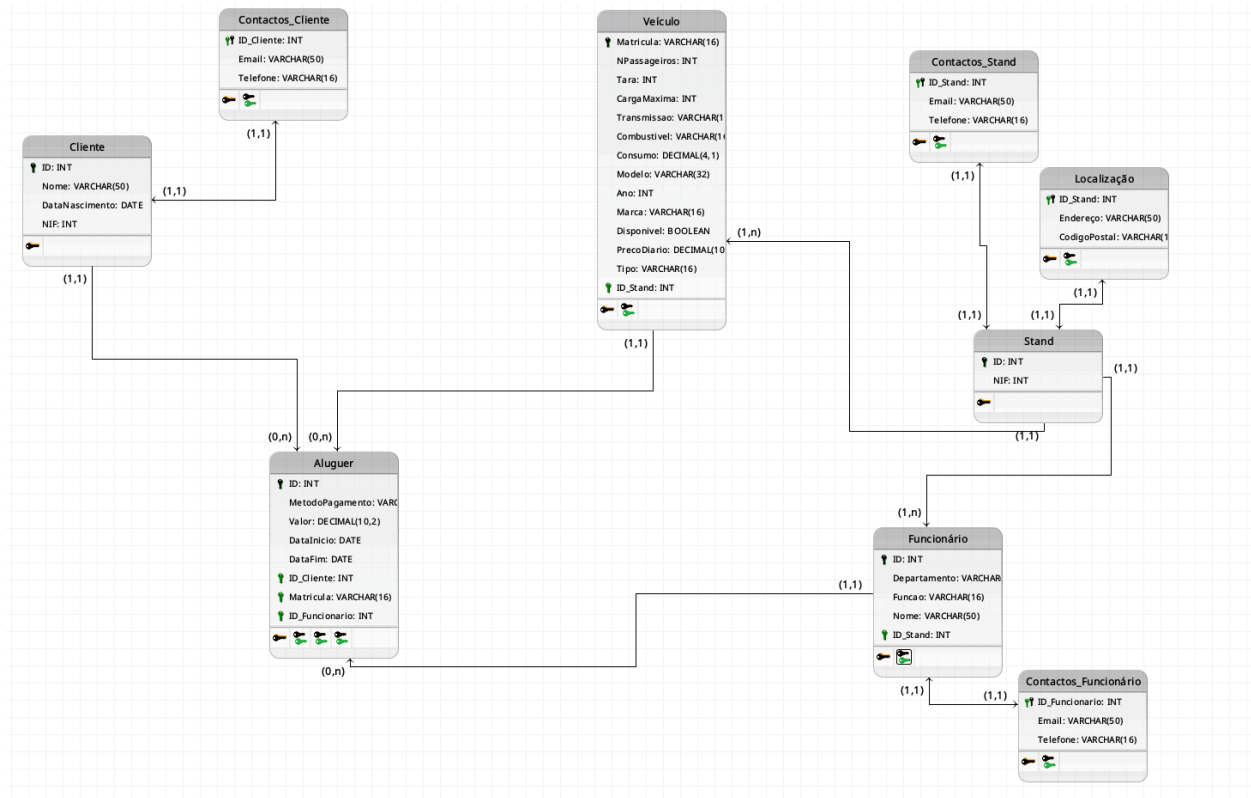
Data Provisória: 27 de março de 2025

Objetivo: Validação final do modelo conceptual e integração de requisitos técnicos.

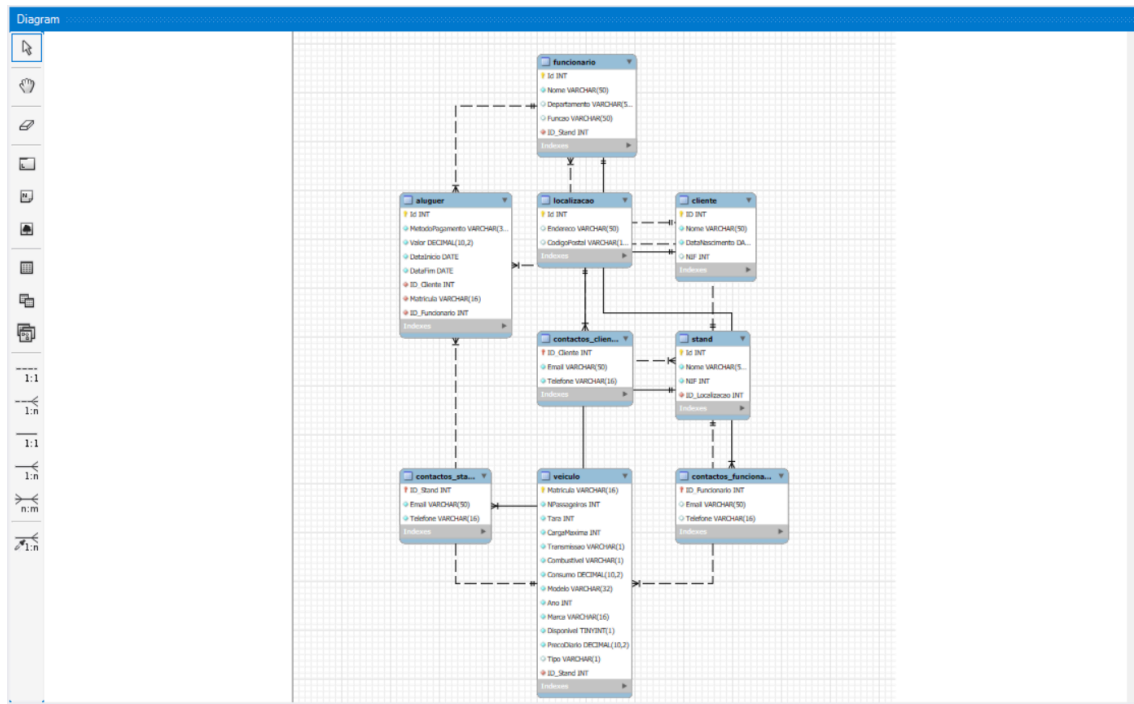
## Anexo 13: Modelo Conceptual Final



## Anexo 14: Modelo Lógico Final



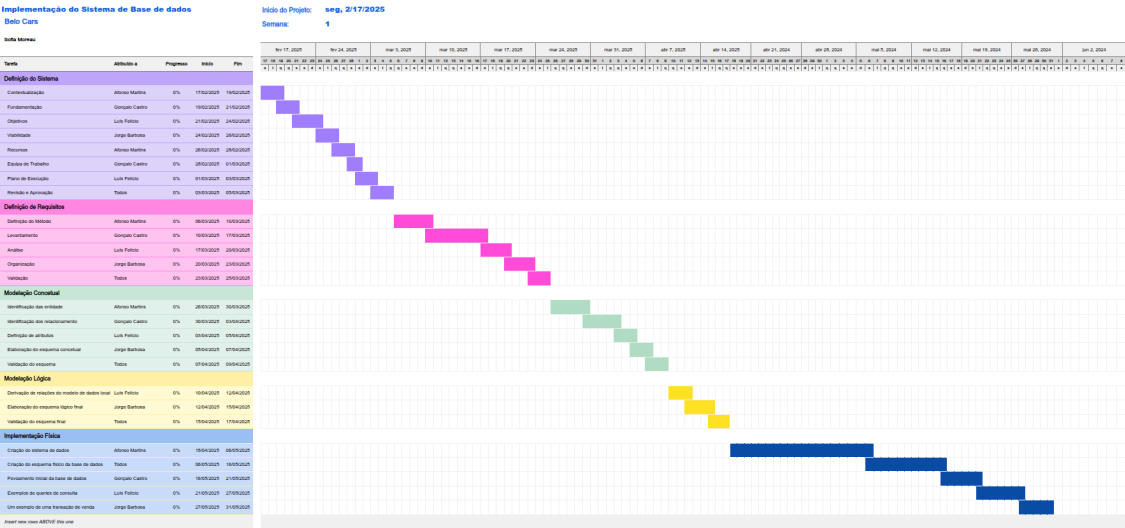
## Anexo 15: Esquema Físico gerado no MySQL Workbench



# Anexo 16: Tamanho relativo a cada tabela (incluindo índices)

Result Grid				Filter Rows:		Export:	Wrap Cell Content:
	Tabela	Tamanho_KB	Tamanho_MB				
▶	aluguer	64.00	0.06				
▶	cliente	32.00	0.03				
	contactos_cliente	48.00	0.05				
	contactos_funcionario	48.00	0.05				
	contactos_stand	48.00	0.05				
	funcionario	32.00	0.03				
	localizacao	16.00	0.02				
	stand	48.00	0.05				
	veiculo	32.00	0.03				

# Anexo 17: Diagrama de Gantt Planeado



# Anexo 18: Diagrama de Gantt Final (preenchido ao longo do desenvolvimento do projeto)

