Departamento de Informática – Escola de Engenharia – Universidade do Minho
Licenciatura em Engenharia Informática

Universidade do Minho

HOLE IN ONE

Base de dados - 2024/2025

Duarte Escairo – a106936

João Rodrigues – a104435

Luís Soares – a106932

Tiago Figueiredo - a106856

5 de junho de 2025

ÍNDICE

DEFINIÇÃO DE SISTEMA

Contexto

A Hole in One é uma empresa de aluguer de carros de golfe fundada em 2004 por Orlando Esbelto. Sediada em Vilamoura, agora conta com três stands e seis funcionários.

Motivação e Objetivos

Com o crescimento da Hole in One e o aumento da popularidade do golfe, o Sr. Esbelto decidiu investir numa base de dados.

Objetivos:

- Facilitar o registo e a gestão de clientes;
- Organizar e estruturar a informação;
- Apoiar a gestão da empresa;
- Melhorar o contacto com clientes.

DEFINIÇÃO DE SISTEMA

- Viabilidade
  - Investimento recuperado em 3 anos
  - +22% de Lucros /ano
- Recursos
  - Equipa da Empresa:
    - Orlando Esbelto
    - José Calado
    - Rogério Samora
    - Edgar Novo
  - Recursos Materiais:
    - MySQL Workbench
    - brModelo
    - Computadores pessoais dos membros do grupo

LEVANTAMENTO E ANÁLISE DE REQUISITOS

- **Métodos:**

Reuniões por videoconferência:

- Reunião geral com fundador e gerentes;
- Reunião individual com o Sr. Esbelto.

Questionário ao fundador:

- Avaliou expectativas e necessidades específicas.

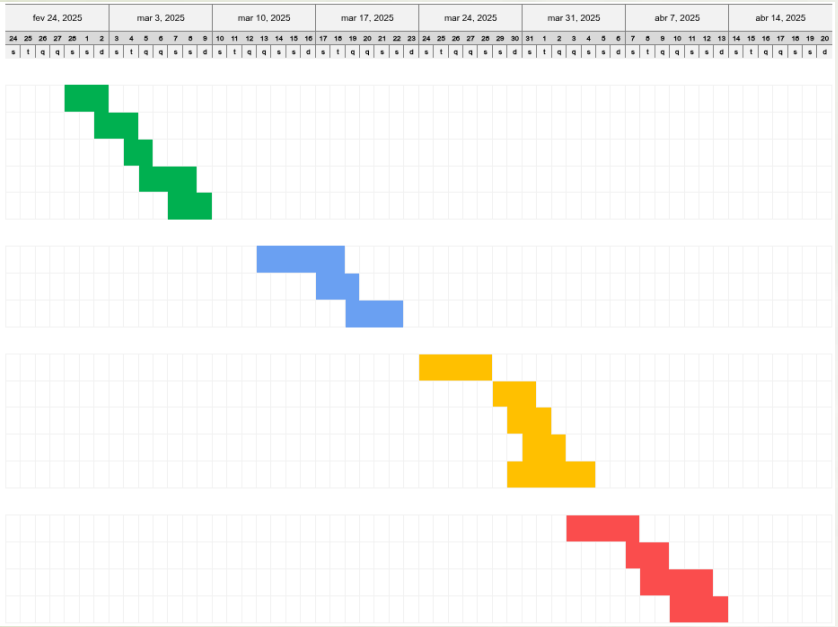
Análise documental:

- Permitiu compreender o processo de aluguer e a informação existente sobre clientes.

DEFINIÇÃO DE SISTEMA

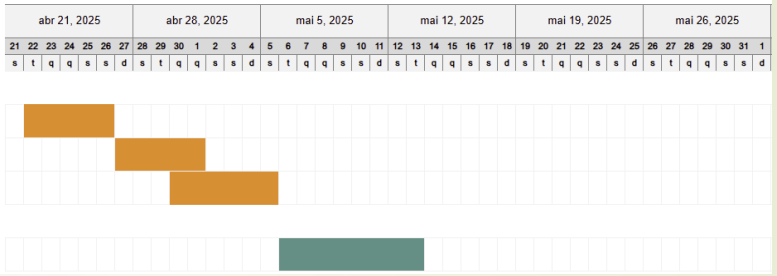
- Plano de execução

Tarefa	Responsável	Início	Fim
Definição do Sistema			
Contexto de aplicação	Duarte Escairo	2/28/25	3/2/25
Motivação e Objetivos do trabalho	Luís Soares	3/2/25	3/4/25
Análise da Viabilidade do processo	João Rodrigues	3/4/25	3/5/25
Recursos e Equipa de trabalho	João Rodrigues	3/5/25	3/8/25
Plano de execução do projeto	Tiago Figueiredo	3/7/25	3/9/25
Definição dos Requisitos			
Método de Levantamento de requisitos	Tiago Figueiredo	3/13/25	3/18/25
Organização dos Requisitos levantados	Duarte Escairo	3/17/25	3/19/25
Análise e Validação dos Requisitos	Luís Soares	3/19/25	3/22/25
Modelação Conceptual			
Abordagem da Modelação	Luís Soares	3/24/25	3/28/25
Identificação e Caracterização das Entidades	Duarte Escairo	3/29/25	3/31/25
Identificação e Caracterização dos Relacionamentos	João Rodrigues	3/30/25	4/1/25
Identificação e Caracterização dos Atributos	Duarte Escairo	3/31/25	4/2/25
Apresentação e Explicação do Modelo produzido	Tiago Figueiredo	3/30/25	4/4/25
Modelação Lógica			
Construção e Validação do Modelo Lógico	João Rodrigues	4/3/25	4/7/25
Apresentação e Explicação do Modelo Lógico	Tiago Figueiredo	4/7/25	4/9/25
Normalização de Dados	Luís Soares	4/8/25	4/12/25
Validação do Modelo com Interrogações do Utilizador	Tiago Figueiredo	4/10/25	4/13/25



1° Fase

Tarefa	Responsável	Início	Fim
Implementação Física			
Criação	Duarte Escairo	4/22/25	4/26/25
Povoamento	Duarte Escairo	4/27/25	5/1/25
Implementação de queries	João Pedro Rodrigues	4/30/25	5/5/25
Exploração e Monitorização			
Análise de Dados	Tiago Figueiredo	5/6/25	5/13/25



2° Fase

LEVANTAMENTO E ANÁLISE DE REQUISITOS

Organização

Os requisitos foram organizados em 3 categorias:

- **Descrição** – Informações sobre entidades, atributos e restrições;
- **Manipulação** – Operações possíveis na base de dados;
- **Controlo** – Permissões e acessos atribuídos aos utilizadores.

Nr	Descrição	Data	Fonte	Analista
RD1	O cliente é caracterizado por ter um identificador único, nome, data de nascimento, telefone, email, morada.	17/03/2025	Reunião Geral	Luís Soares

Requisito de Descrição

Nr	Descrição	Data	Fonte	Analista
RM1	Os dados de um cliente devem poder ser acedidos através do identificador do cliente.	18/03/2025	Reunião com o Sr. Esbelto	Luís Soares

Requisito de Manipulação

Nr	Descrição	Data	Fonte	Analista
RC1	O fundador da empresa tem permissão para acrescentar, modificar, apagar e ler todos os registos do sistema.	17/03/2025	Reunião Geral	Tiago Figueiredo

Requisito de Controlo

LEVANTAMENTO E ANÁLISE DE REQUISITOS

- **Análise e Viabilidade dos requisitos**

Após a recolha, foi realizada uma nova reunião para confirmar e validar todos os requisitos.

Durante a sessão, foram corrigidos erros.

Foram também acrescentados atributos importantes:
- Data de nascimento (Cliente);
- Número de lugares (Carro de golfe).

Após todos os presentes validarem o trabalho que fora realizado, o grupo de estudantes tinha agora uma base sólida onde se apoiar para o restante desenvolvimento do processo.

MODELAÇÃO CONCEPTUAL

- Após a reunião final com todos os intervenientes, o levantamento e organização de requisitos foi totalmente validado.
- De seguida, foram identificadas as entidades, os relacionamentos e os atributos, representadas pelas seguintes tabelas:

Entidade	Descrição	Sinónimos	Ocorrência	Requisito
Cliente	Pessoa que recorre aos serviços de aluguer prestados pela empresa.	...	Um cliente precisa de estar registado na empresa para poder fazer um aluguer de um carro.	RD1
Aluguer	Representa um registo feito, depois de um cliente requisitar os serviços da empresa.	...	Cada aluguer tem o número de identificação único associado no momento do registo.	RD3
Funcionário	Pessoa que presta diversos serviços dentro da empresa.	Empregado	Um funcionário exerce uma função mediante o seu estatuto dentro da empresa.	RD2
Carro	Produto usado nos serviços prestados pela empresa.	Veículo	Cada aluguer feito pela empresa apenas tem associado um carro.	RD4
Função	Representa um dos trabalhos realizados pelos funcionários.	Cargo	A cada funcionário é atribuída uma única função.	RD9

Tabela Entidades

Entidade	Relacionamento	Cardinalidade	Participação	Entidade	Requisito
Cliente	Faz	1:N	T:T	Aluguer	RD5
Aluguer	De Um	N:1	T:P	Carro	RD6
Aluguer	Tratado	N:1	T:P	Funcionário	RD7
Funcionário	Exerce	N:1	T:P	Função	RD8

Tabela Relacionamentos

Entidade: Cliente

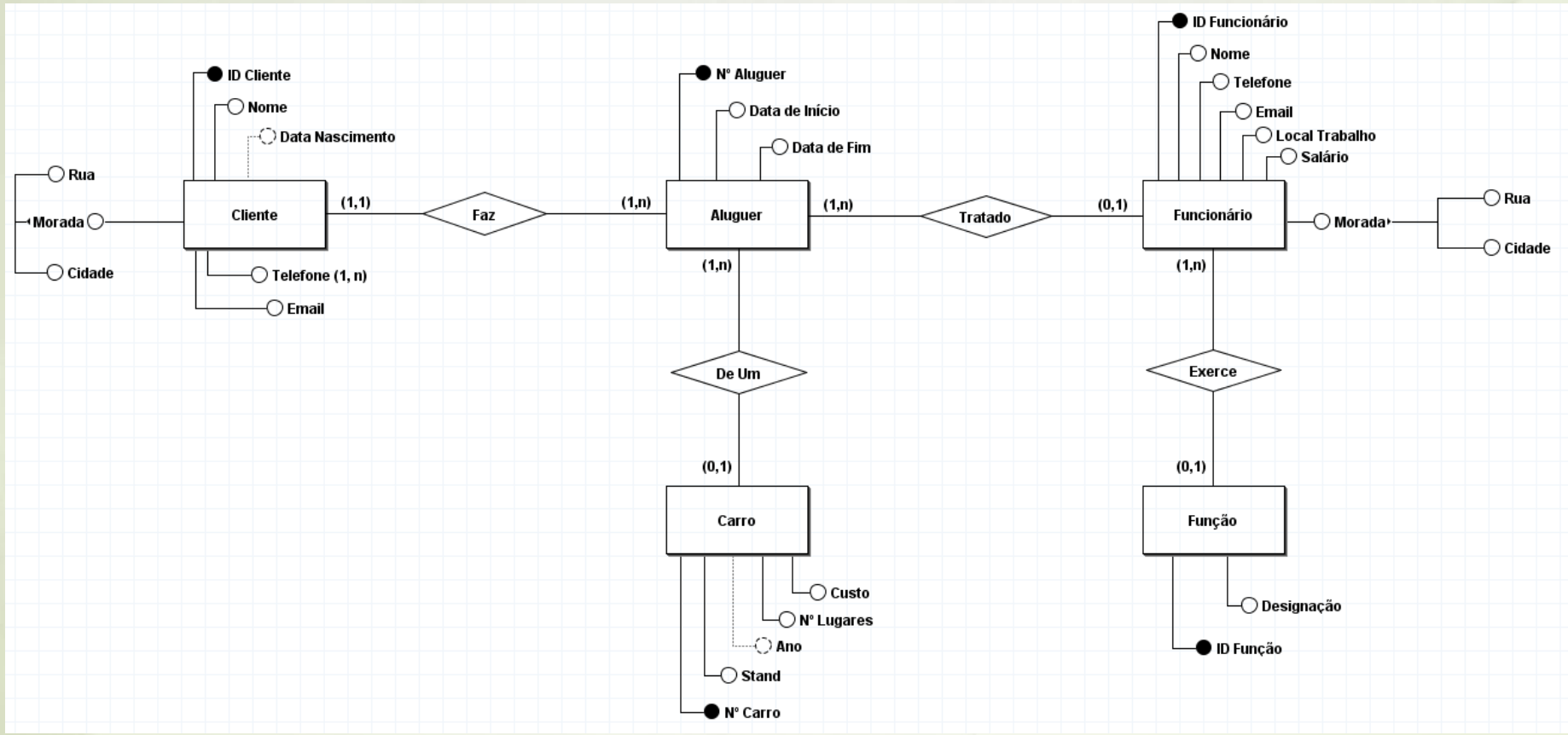
Atributos	Descrição	Tipo	NULL	Multivalorado	Chave Primária	Exemplo	Requisito
ID Cliente	Identificador único do cliente	INT	N	N	S	84	RD1
Nome	Nome do cliente	VARCHAR(100)	N	N	N	Tomás Pimenta	RD1
Data Nascimento	Data de nascimento do cliente	DATE	S	N	N	26/11/2005	RD1
Telefone	Lista de números de telefone	INT	N	S	N	[912345678, 926769123]	RD1
Email	Email do cliente	VARCHAR(50)	N	N	N	tomas.pimenta@yahoo.com	RD1
Morada (Rua, Cidade)	Morada do cliente	VARCHAR(100); VARCHAR(50)	N	N	N	Rua dos Pedreiros, Vila Verde	RD1

S/N - SIM/NÃO

Tabela Atributos do Cliente

MODELAÇÃO CONCEPTUAL

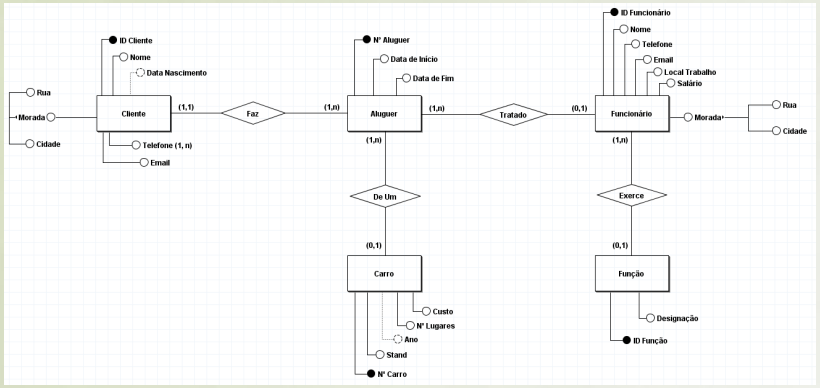
- Desta forma procedeu-se à construção do diagrama ER(Entidade – Relacionamento).

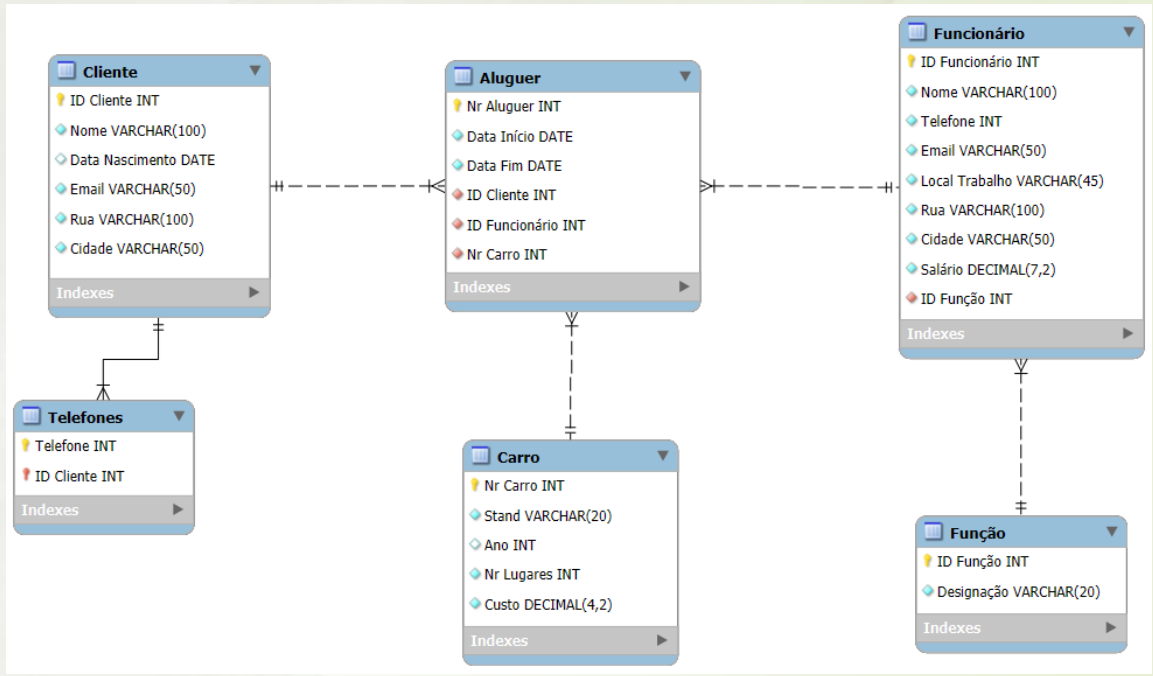


MODELAÇÃO LÓGICA

- Após a construção do modelo conceptual, foi derivado o modelo lógico, utilizando o MySQL Workbench.
- As tabelas criadas correspondem diretamente às entidades identificadas:
- Cliente
- Telefones
- Funcionario
- Funcao
- Carro
- Aluguer
- Aplicaram-se as três primeiras Formas Normais para garantir integridade, consistência e controlo de redundância.

MODELAÇÃO LÓGICA





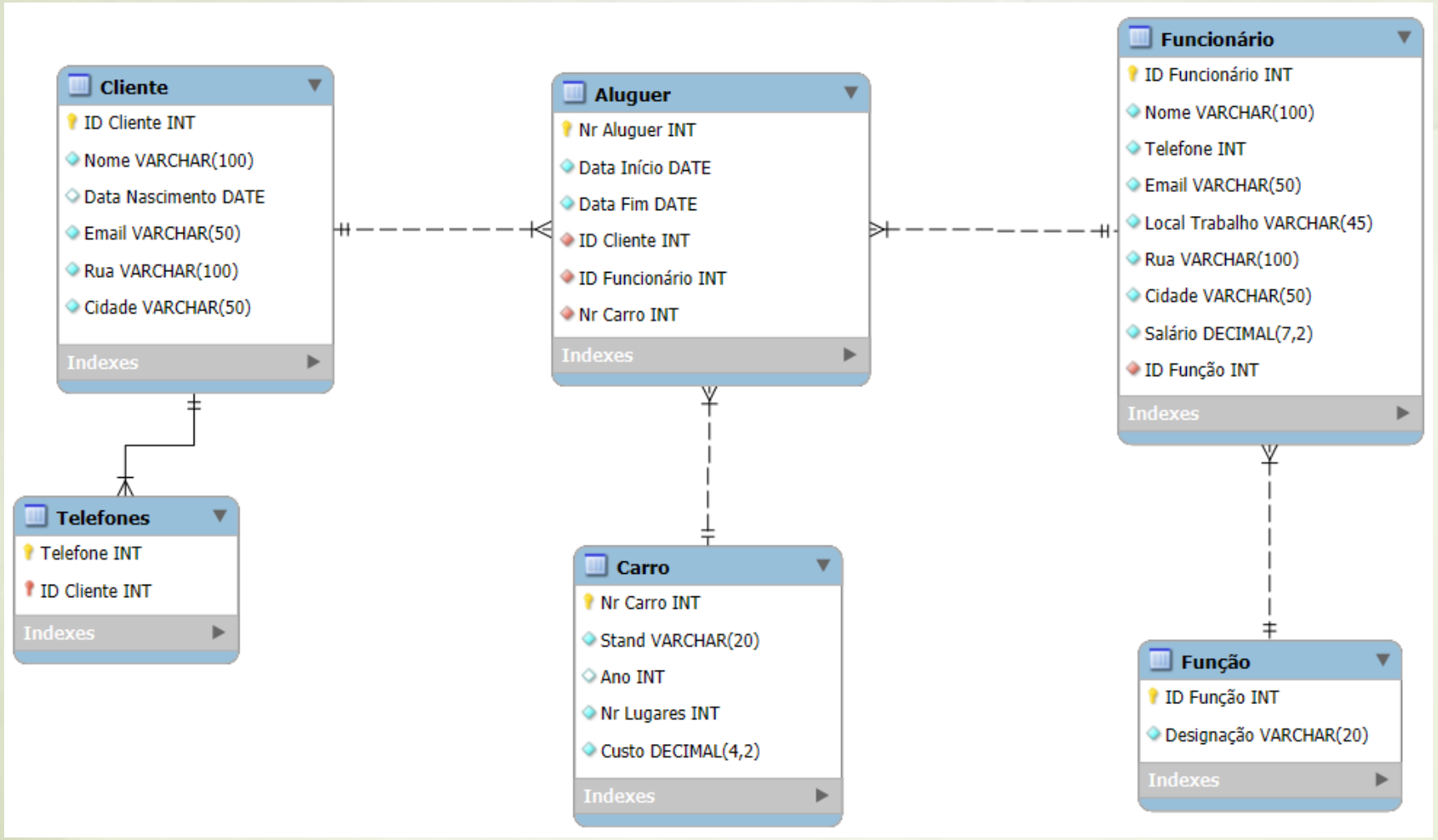
1 FN

2 FN

3 FN

MODELAÇÃO LÓGICA

- Deste forma, apresenta-se o Modelo Lógico.



MODELAÇÃO LÓGICA

- **Álgebra Relacional**

  Para validar o modelo lógico, o grupo implementou queries correspondentes aos requisitos de manipulação, traduzindo-as em expressões de Álgebra Relacional.

  Exemplos de validação:

  - RM4 : É possível listar todos identificadores de cliente, e o seu respetivo nome.

    $$\pi \text{ ID Cliente, Nome (Cliente)};$$

  - RM5 : É possível listar os números de alugueres que um determinado cliente fez, bem como o seu identificador e a data de inicio desses lugares.

    $$\pi \text{ Nr Aluguer, ID Cliente, Data Início } (\sigma \text{ ID Cliente} = X \text{ (Aluguer))}.$$

  - RM6 : Deve ser possível listar os nomes de todos os clientes que foram atendidos por um determinado funcionário.

    $$\pi \text{ Nome (Cliente} \bowtie \text{Cliente.ID Cliente=Aluguer.ID Cliente } (\sigma \text{ ID Funcionário} = X \text{ (Aluguer)))}$$

IMPLEMENTAÇÃO FÍSICA

- Após definida a estrutura lógica, foi desenvolvida a implementação física da BD HoleInOne, utilizando scripts SQL.

Criação da BD:

```
CREATE DATABASE IF NOT EXISTS HoleInOne;
-- DROP DATABASE HoleInOne;
USE HoleInOne;
```

Ordem de criação das tabelas (devido às chaves estrangeiras):

Cliente → Telefones → Carro → Função → Funcionário → Aluguer;

```
CREATE TABLE Carro(
    Nr_Carro INT NOT NULL,
    Stand VARCHAR(20) NOT NULL,
    Ano INT,
    Nr_Lugares INT NOT NULL,
    Custo DECIMAL(4,2) NOT NULL,
    PRIMARY KEY(Nr_Carro)
);
```

Criação da tabela Carro

IMPLEMENTAÇÃO FÍSICA

- Para garantir a segurança e o controlo do acesso à base de dados, foram criados diferentes ROLES com permissões especificas, conforme os requisitos de controlo.
- Existem 3 **ROLES** : **Dono**, **Gerente** e **Secretário**.

Exemplos de atribuição de permissões:

```
GRANT SELECT, INSERT, UPDATE, DELETE
    ON HoleInOne.Cliente TO Gerente;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE
    ON HoleInOne.Telefones TO Gerente;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE
    ON HoleInOne.Aluguer TO Gerente;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE
    ON HoleInOne.Carro TO Gerente;
```

```
GRANT INSERT, SELECT ON HoleInOne.Cliente TO Secretario;

GRANT INSERT, SELECT ON HoleInOne.Telefones TO Secretario;

GRANT INSERT, SELECT ON HoleInOne.Aluguer TO Secretario;
```

```
CREATE ROLE Dono;
CREATE ROLE Gerente;
CREATE ROLE Secretario;
```

IMPLEMENTAÇÃO FÍSICA

- Para inserir dados nas 6 tabelas da base de dados, foram utilizados 3 métodos diferentes, os quais são :
  - Ficheiros CSV;
  - Ficheiros JSON;
  - SQL (INSERT INTO).

```csv
Nr_Carro,Stand,Ano,Nr_Lugares,Custo
1000,Vilamoura,2017,4,25.0
1001,Vilamoura,2020,4,25.0
1002,Vilamoura,2016,2,15.0
1003,Braga,2013,2,15.0
1004,Braga,2015,2,15.0
1005,Braga,2021,4,25.0
1006,Porto,2017,4,25.0
1007,Porto,2013,2,15.0
1008,Porto,2016,4,25.0
1009,Porto,2009,2,15.0
```

Fonte CSV

Ficheiro Carro.csv

```json
[
  {"ID_Funcao": 1, "Designacao": "Dono"},
  {"ID_Funcao": 2, "Designacao": "Gerente"},
  {"ID_Funcao": 3, "Designacao": "Secretaria"}
]
```

Fonte JSON

Ficheiro Funcao.json

```sql
INSERT INTO Carro
        (Nr_Carro, Stand, Ano, Nr_Lugares, Custo)
        VALUES (1000, 'Vilamoura', 2017, 4, 25.0),
            (1001, 'Vilamoura', 2020, 4, 25.0),
            (1002, 'Vilamoura', 2016, 2, 15.0),
            (1003, 'Braga', 2013, 2, 15.0),
            (1004, 'Braga', 2015, 2, 15.0),
            (1005, 'Braga', 2021, 4, 25.0),
            (1006, 'Porto', 2017, 4, 25.0),
            (1007, 'Porto', 2013, 2, 15.0),
            (1008, 'Porto', 2016, 4, 25.0),
            (1009, 'Porto', 2009, 2, 15.0);
```

SQL

Inserção SQL

IMPLEMENTAÇÃO FÍSICA

Cálculo do Espaço da Base de Dados

A estimativa do espaço ocupado pela base de dados permite prever as necessidades de hardware e garantir a escalabilidade futura do sistema.

Tamanhos por tipo de dados:

- DATE → 3 bytes;
- INT → 4 bytes;
- VARCHAR → até N x 4 bytes + 1 ou 2 bytes de prefixo;
- DECIMAL → depende do nº de dígitos.

Base de Dados			
Relação	Número de Registos	Tamanho p/ Registo (bytes)	Tamanho (bytes)
Cliente	11	313	3443
Telefones	11	8	88
Carro	10	35	350
Funcao	3	25	75
Funcionario	7	368	2576
Aluguer	11	22	242
Tamanho total: 6774 bytes ≈ 6.8kB			

IMPLEMENTAÇÃO FÍSICA

View

- Apesar de não haver necessidade de uma vista nesta base de dados, a equipa ainda assim achou pertinente mostrar o exemplo de uma.
- A vista "NumerosCarrosAlugadosporFuncionariosTipo2" tem como objetivo apresentar os números dos carros, cujos alugueres foram realizados por funcionários do tipo Gerente, ou seja, com **ID_Funcao = 2.**

```
CREATE VIEW NumerosCarrosAlugadosporFuncionárioTipo2 AS
   SELECT Ca.Nr_Carro
      FROM Carro AS Ca
         INNER JOIN( Aluguer AS A
            INNER JOIN(
               SELECT * FROM Funcionario
                        WHERE ID_Funcao = 2) AS F
            ON A.ID_Funcionario = F.ID_Funcionario)
         ON Ca.Nr_Carro = A.Nr_Carro;
```

IMPLEMENTAÇÃO FÍSICA

- O objetivo do uso de interrogações é responder às principais questões operacionais da empresa através do SQL, como:
  - Consultar dados de clientes e funcionários;
  - Aceder a alugueres;
  - Relacionar dados entre tabelas.
- **Exemplos de interrogações:**
  - **RM1** – Dados de um cliente;

```
SELECT *
    FROM Cliente
    WHERE ID_Cliente = 100;
```

  - **RM2** – Dados de um funcionário;

```
SELECT *
    FROM Funcionario
    WHERE ID_Funcionario = 1;
```

  - **RM6** – Clientes atendidos por um funcionário;

```
SELECT C.Nome
    FROM Cliente AS C
    INNER JOIN (SELECT * FROM Aluguer WHERE ID_Funcionario = 1) AS A
        ON C.ID_Cliente = A.ID_Cliente;
```

  - **RM7** – Carros alugados por gerentes.

```
SELECT Ca.Nr_Carro
FROM Carro AS Ca
    INNER JOIN Aluguer AS A
        ON Ca.Nr_Carro = A.Nr_Carro
    INNER JOIN (
        SELECT ID_Funcionario
        FROM Funcionario
        WHERE ID_Funcao = 2
    ) AS F
        ON A.ID_Funcionario = F.ID_Funcionario;
```

IMPLEMENTAÇÃO FÍSICA

- **Procedimento**

  - **"Adiciona_User_e_Telefone"** – insere um novo registo nas tabelas Clientes e Telefones na base de dados através de uma transação (TRANSACTION), o que torna a operação contida no procedimento atómica.

```
CREATE PROCEDURE Adiciona_User_e_Telefone(IN Id INT,
                                          IN Nome VARCHAR(100),
                                          IN Nasc DATE,
                                          IN Email VARCHAR(50),
                                          IN Rua VARCHAR(100),
                                          IN Cidade VARCHAR(50),
                                          IN Telefone INT)
BEGIN

START TRANSACTION;

INSERT INTO Cliente
(ID_Cliente, Nome, Data_Nascimento, Email, Rua, Cidade)
    VALUES (Id,Nome,Nasc,Email,Rua,Cidade);

INSERT INTO Telefones
(Telefone, ID_Cliente)
    VALUES (Telefone, Id);

COMMIT;

END $$
```

IMPLEMENTAÇÃO FÍSICA

- Trigger
  - **"VerificarDisponibilidadeCarro"** – Este gatilho (TRIGGER) é responsável por, antes de uma inserção na tabela Aluguer, verificar se a data inserida na nova instância de Aluguer pertence a um intervalo de tempo ao qual o carro já esteja alugado.

```
DELIMITER $$
CREATE TRIGGER VerificarDisponibilidadeCarro
BEFORE INSERT ON Aluguer
FOR EACH ROW
BEGIN
    DECLARE alugado INT;

    SELECT COUNT(*) INTO alugado
    FROM Aluguer
    WHERE Nr_Carro = NEW.Nr_Carro
      AND (
        NEW.Data_Inicio BETWEEN Data_Inicio AND Data_Fim
        OR
        NEW.Data_Fim BETWEEN Data_Inicio AND Data_Fim
        OR
        Data_Inicio BETWEEN NEW.Data_Inicio AND NEW.Data_Fim
        OR
        Data_Fim BETWEEN NEW.Data_Inicio AND NEW.Data_Fim
      );
    IF alugado > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Erro: O carro já está alugado nesse período.';
    END IF;
END $$
```

IMPLEMENTAÇÃO FÍSICA

- **Function**

  • **"TotalAlugueresCliente"** – Esta função(Function) recebe como argumento um INT que corresponde a identificador de utilizador e retorna o número de alugueres realizados por esse cliente.

```
DELIMITER $$

CREATE FUNCTION TotalAlugueresCliente(cliente_id INT) RETURNS INT
DETERMINISTIC

BEGIN
    DECLARE total INT;
    SELECT COUNT(*) INTO total
        FROM Aluguer
        WHERE ID_Cliente = cliente_id;
    RETURN total;
END $$


DELIMITER $$
```

IMPLEMENTAÇÃO FÍSICA

Indexs

- Os índices são estruturas de dados que aceleram a busca e acesso a dados numa tabela.
- Na realização deste trabalho prático decidimos que os índices poderiam ser utilizados em algumas das estruturas que criamos no contexto da nossa BD.

• **"idx_Datas_Aluguer"** - Este índice otimiza consultas que filtram ou ordenam registros com base no intervalo de datas de aluguer (**Data_Inicio** e **Data_Fim**).

```
CREATE INDEX idx_Datas_Aluguer ON Aluguer(Data_Inicio, Data_Fim);
```

• **"idx_aluguer_Nr_Carro"** - Acelera consultas que procuram por alugueres de um carro específico num aluguer (**Nr_Carro**).

```
CREATE INDEX idx_aluguer_Nr_Carro ON Aluguer(Nr_Carro);
```

IMPLEMENTAÇÃO DO SISTEMA DE MIGRAÇÃO DE DADOS

- Objetivo: Integrar dados provenientes de fontes CSV, JSON e relacionais(SQL) na BD projetada(HoleInOne).
- Solução: Desenvolver um programa python que recolhe esses dados, constrói dinamicamente as instruções para inserir a informação na BD e envia-as para o servidor para serem executadas.

```python
def inserir_dados_csv(nome_tabela, caminho_csv, colunas, conn):
    try:
        #conn = mysql.connector.connect(**config)
        cursor = conn.cursor()

        with open(caminho_csv, newline='', encoding='utf-8') as csvfile:
            reader = csv.DictReader(csvfile)
            for row in reader:
                valores = [row[col] for col in colunas]
                placeholders = ', '.join(['%s'] * len(colunas))
                query = f"INSERT INTO {nome_tabela} ({', '.join(colunas)}) VALUES ({placeholders})"
                cursor.execute(query, valores)
```

- Lê um ficheiro **CSV** linha a linha e insere os dados na tabela da **BD**, referente ao ficheiro lido, através de instruções **SQL** do tipo **INSERT INTO** criadas dinamicamente.

IMPLEMENTAÇÃO DO SISTEMA DE MIGRAÇÃO DE DADOS

```python
def carregar_json(nome_ficheiro):
    with open(nome_ficheiro, 'r', encoding='utf-8') as jsonfile:
        return json.load(jsonfile)
```

```python
def inserir_dados_json(tabela, dados, conn):
    cursor = conn.cursor()

    if not dados:…

    colunas = dados[0].keys()
    placeholders = ", ".join(["%s"] * len(colunas))
    colunas_str = ", ".join(colunas)

    sql = f"INSERT INTO {tabela} ({colunas_str}) VALUES ({placeholders})"

    try:
        for linha in dados:
            valores = tuple(linha[col] for col in colunas)
            cursor.execute(sql, valores)
        conn.commit()
```

- Abre um ficheiro **JSON** dado o seu ***path*** e retorna uma estrutura de dicionários ***python*** a ser usada em **"inserir_dados_json"**.

- Recebe como argumento informação obtida de um ficheiro **JSON** aberto antes da chamada da função e usa essa informação para inserir dados em uma tabela da **BD**, referente ao ficheiro **JSON** em questão.

IMPLEMENTAÇÃO DO SISTEMA DE MIGRAÇÃO DE DADOS

```python
def inserir_dados_relacionais(conn):
    cursor = conn.cursor()
    try:
        # Inserir dados na tabela Carro
        cursor.execute("""
            INSERT INTO Carro (Nr_Carro, Stand, Ano,
            (1000, 'Vilamoura', 2017, 4, 25.0),
            (1001, 'Vilamoura', 2020, 4, 25.0),
            (1002, 'Vilamoura', 2016, 2, 15.0),
            (1003, 'Braga', 2013, 2, 15.0),
            (1004, 'Braga', 2015, 2, 15.0),
            (1005, 'Braga', 2021, 4, 25.0),
            (1006, 'Porto', 2017, 4, 25.0),
            (1007, 'Porto', 2013, 2, 15.0),
            (1008, 'Porto', 2016, 4, 25.0),
            (1009, 'Porto', 2009, 2, 15.0);
        """)
```

• Função executa comandos **SQL** previamente definidos para inserir registos diretamente nas tabelas **Carro** e **Aluguer**.

```python
def main():

    # Configurações da ligação MySQL
    config = {
            'host': '******',
            'user': '******',
            'password': '******',
            'database': 'HoleInOne',
    }
```

- Função que define as configurações da ligação **MySQL** e chama as funções mencionadas anteriormente.

CONCLUSÃO

Departamento de Informática – Escola de Engenharia – Universidade do Minho
Licenciatura em Engenharia Informática

Universidade do Minho

HOLE IN ONE

Base de dados - 2024/2025

Duarte Escairo – a106936

João Rodrigues – a104435

Luís Soares – a106932

Tiago Figueiredo - a106856

5 de junho de 2025