Universidade do Minho
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Letivo de 2021/2022

Alojamento Local em NoSQL

David Machado, Inês Presa, Ivo Lima, Tiago Carriço

Janeiro, 2022

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Alojamento Local em NoSQL

David Machado, Inês Presa, Ivo Lima, Tiago Carriço

A91665, A90355, A90214, A91695

Janeiro, 2022

1 Resumo

Este relatório começa por apresentar a correção dos erros referentes ao modelo relacional anteriormente desenvolvido, seguida da prova fundamentada da satisfação das formas de normalização. Subsequentemente, expõem-se e justifica-se a criação de índices e procedimentos na mesma base de dados.

A segunda parte, apresenta o processo da concessão e implementação do sistema de dados em *MongoDB,* estando aqui envolvidos a descrição dos passos da migração para *NoSQL* e a definição do esquema da coleção.

Finalmente, é exposta a tradução dos requisitos de exploração para a linguagem *MongoDB* e demostradas os respetivos resultados.

Área de Aplicação: Criação e Implementação de uma Base de Dados não relacional para Gestão de um Alojamento Local.

Palavras-Chave: Base de Dados não relacionais, MongoDB, *NoSQL, MySQL*

2 Índice

3 Índice de Ilustrações

4 Índice de Tabelas

1...Sistema Relacional Implementado

1.1. Apresentação do Sistema

Como referido na conclusão da primeira parte do trabalho, a base de dados apresentada anteriormente requeria alguns melhoramentos, por conseguinte, apresentaremos de seguida os novos esquemas concetual e lógico.

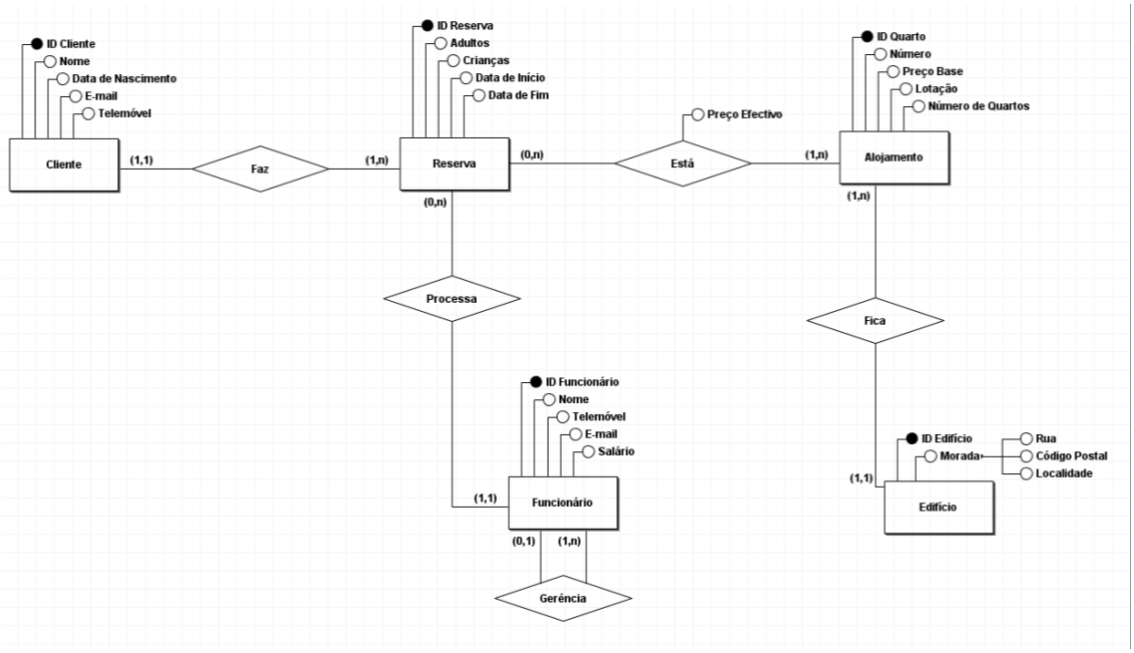

Ilustração 1 - Modelo Concetual

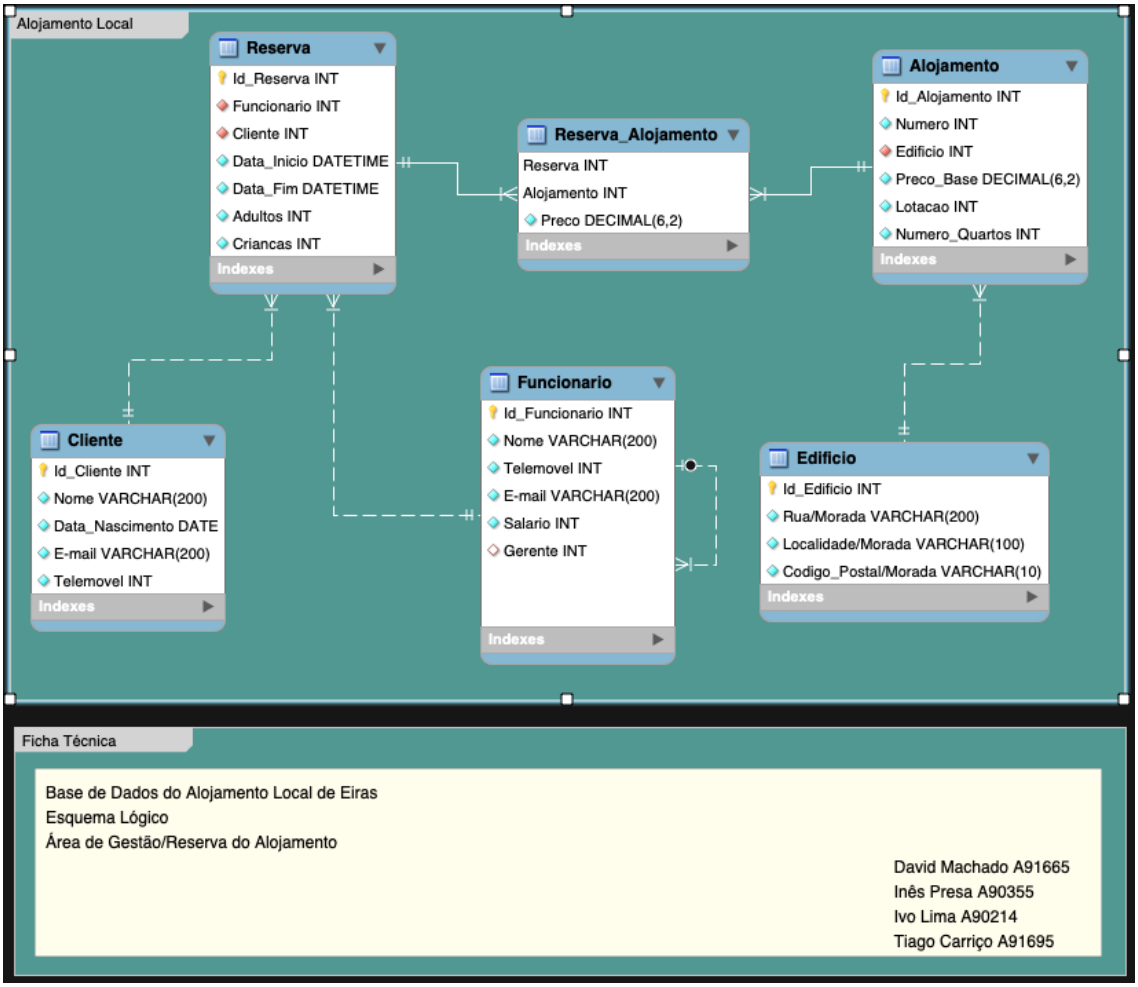

Ilustração 2 - Modelo Lógico

Com as alterações efetuadas ao esquema tornou-se necessário corrigir algumas *Queries* e possível responder a novos requisitos de exploração.

- **RE07 – Listar os quartos que estão disponíveis até um certo preço**

```sql
-- RE07 – Listar os quartos que estão disponíveis até um certo preço
SELECT Id_Alojamento, Preco_Base FROM Alojamento WHERE Preco_Base <= 100;
```

Ilustração 3 - Código *SQL* para RE07

Id_Alojamento	Preco_Base
4	70.00
5	70.00
NULL	NULL

Ilustração 4 - Resultado da *Query* relativa a RE07

- **RE08 – Listar os clientes por ordem decrescente de dinheiro gasto**

```sql
-- RE08 - Listar os clientes por ordem descrescente de dinheiro gasto
SELECT R.Cliente, C.Nome, SUM(RA.Preco) AS Total_Gasto FROM Reserva AS R
    INNER JOIN Cliente AS C
        ON C.Id_Cliente = R.Cliente
    INNER JOIN Reserva_Alojamento AS RA
        ON RA.Reserva = R.Id_Reserva
    GROUP BY R.Cliente
    ORDER BY Total_Gasto DESC;
```

Ilustração 5 - Código *SQL* para RE08

Cliente	Nome	Total_Gasto
74	Vítor Vaz-Antunes	570.00
73	Anita Gomes	510.00
60	Adriana Soares	510.00
52	Rita-Beatriz Esteves	510.00
5	Ivo Nascimento	510.00
80	Camila Henriques	510.00
50	Carlota Nunes	510.00
89	Matilde Castro	510.00
28	Mara Mota	510.00
27	Igor Simões	510.00
72	Cláudio Nunes	450.00
86	Rodrigo Anjos	450.00
36	Ismael do Brito	450.00
95	Afonso Oliveira	450.00
15	Bernardo Pinho	450.00

Ilustração 6 - Resultado da *Query* relativa a RE08

- **RE09 – Calcular o valor total dos alugueres de cada quarto**

```sql
# Calcular o valor total dos alugueres de cada quarto
select Alojamento, sum(Preco) from Reserva_Alojamento
    group by Alojamento;
```

Ilustração 7 - Código *SQL* para RE09

Alojamento	sum(Preco)
1	16910.00
2	3640.00
3	4160.00
4	490.00
5	350.00
6	4420.00
7	2990.00

Ilustração 8 - Resultado da *Query* relativa a RE09

1.2.Normalização de Dados

A normalização tem como principal objetivo evitar a redundância de dados, proporcionando um maior rendimento do modelo, uma vez que este processo evita as anomalias provocadas pela inserção, exclusão e alteração de registos na base de dados.

Para avaliar a normalização do sistema devemos verificar que as 3 formas normais são satisfeitas, tendo em conta a análise das tabelas das diferentes entidades. Que se encontram apresentadas de seguida.

Id_Alojamento	Numero	Edificio	Preco_Base	Lotacao	Numero_Quartos
1	1	1	190.00	6	3
2	2	1	130.00	4	2
3	3	1	130.00	4	2
4	1	2	70.00	2	1
5	2	2	70.00	2	1
6	3	2	130.00	4	2
7	4	2	130.00	4	2
NULL	NULL	NULL	NULL	NULL	NULL

Tabela 1 - Alojamento

Id_Cliente	Nome	Data_Nascimento	E-mail	Telemovel
3	Mauro Valente	1968-01-13	iaraamorim@sapo.pt	924696238
4	Dinis Ramos	1982-10-13	wilson65@clix.pt	966185225
5	Ivo Nascimento	1989-10-22	helenaamaral@gmail.com	274873305
6	Vasco Monteiro	1973-01-11	moreiraivan@gmail.com	968986430
7	William Pinto	1996-02-07	leonardopinto@clix.pt	257597846
8	Emma Jesus	1962-01-02	irinaaraujo@sapo.pt	969513367
9	Xavier Anjos	1997-10-05	sofianascimento@hotmail.com	245808031
10	Kévim Amaral	1991-02-18	rodrigoalves@hotmail.com	208118237
11	Jéssica Costa-...	1995-03-09	afonseca@sapo.pt	920741120
12	Gaspar Magalh...	1966-11-29	jose27@sapo.pt	913670849

Tabela 2 - Cliente

Id_Edificio	Rua/Morada	Localidade/Morada	Codigo_Postal/Morada
1	Rua Principal, 1	Chaves	5400-623
2	Rua Principal, 2	Chaves	5400-623
NULL	NULL	NULL	NULL

Tabela 3 - Edificio

Id_Funcionario	Nome	Telemovel	E-mail	Salario	Gerente
1	Luís Presa	239956636	lpresa@sapo.com	1515	NULL
2	Vasco Morais	927124008	xcarneiro@gmail.com	665	1
3	Rafaela Campos-Paiva	967127667	araujoalicia@hotmail.com	665	1
4	Cláudio Baptista	282992682	martim65@gmail.com	665	1
NULL	NULL	NULL	NULL	NULL	NULL

Tabela 4 - Funcionario

Id_Reserva	Funcionario	Cliente	Data_Inicio	Data_Fim	Adultos	Criancas
3	1	2	2020-01-14 00:00:00	2020-01-17 00:00:00	3	1
4	1	2	2020-01-17 00:00:00	2020-01-20 00:00:00	3	0
5	2	3	2020-01-23 00:00:00	2020-01-26 00:00:00	2	2
6	3	3	2020-01-23 00:00:00	2020-01-26 00:00:00	1	3
7	1	3	2020-01-30 00:00:00	2020-02-02 00:00:00	3	3
8	3	4	2020-02-01 00:00:00	2020-02-04 00:00:00	2	0
9	3	4	2020-02-04 00:00:00	2020-02-07 00:00:00	1	0
10	2	4	2020-02-06 00:00:00	2020-02-09 00:00:00	2	1
11	2	5	2020-02-13 00:00:00	2020-02-16 00:00:00	1	2
12	2	5	2020-02-16 00:00:00	2020-02-19 00:00:00	2	3

Tabela 5 - Reserva

Reserva	Alojamento	Preco
1	1	190.00
2	3	130.00
3	1	190.00
4	3	130.00
5	6	130.00
6	6	130.00
7	1	190.00
8	6	130.00
9	3	130.00
10	6	130.00

Tabela 6 - Reserva_Alojamento

A Primeira Forma Normal (1FN) enuncia que os atributos devem ser atómicos, ou seja, nas tabelas não podem existir valores repetidos, nem podem existir atributos multivalorados. Com uma simples análise às tabelas do nosso modelo, reparamos que tal não acontece, tornando válida a 1FN.

A Segunda Forma Normal (2FN), depende da satisfaçam da 1FN, citando que os atributos normais devem depender apenas da chave primária da tabela. Tomemos como exemplo a Data_Nascimento que está presente na entidade Cliente, este atributo depende somente da chave primária, Id_Cliente. Verificando-se a mesma correlação com os restantes atributos do modelo, validando a 2FN.

Por último, a Terceira Forma Normal (3FN) que também está sujeito à validade das 2 formas anteriores, impõe a verificação da existência de vínculos entre os diferentes atributos, isto é, se existe algum atributo que pode ser gerado a partir de outro. No caso do nosso modelo, nenhum atributo é proporcional ou possível de obter a partir de outro, cumprindo assim a 3FN.

Por conseguinte, podemos afirmar que o nosso modelo está normalizado.

1.3. Indexação do Sistema de Dados

A utilização de índices em *SQL* permite reduzir o tempo gasto na procura de informação numa base de dados, tendo como inconvenientes um maior consumo de espaço para armazenamento da informação e redução da eficiência das operações de inserção e atualização.

Quando é utilizada a funcionalidade *Forward Engineering* no *SQL Workbench*, o *script* de inicialização da Base de Dados devolvido já criará um conjunto de índices, associados às colunas com restrições a chaves primárias e chaves estrangeiras, chamados de índices implícitos.

Dados os factos apresentados percebemos que devemos tentar definir índices para as colunas que são consultadas inúmeras vezes, mas que não são atualizadas frequentemente.

Uma boa escolha para a definição de um índice será o atributo Id_Alojamento da tabela Alojamento, visto que, estamos sempre a consultar/utilizar esta informação e a mesma se mantém constante ao longo de toda a vida da BD. Assim, para criar este índice, usamos o seguinte comando:

```
CREATE INDEX nAlojamento ON Alojamento (Id_Alojamento);
```

Ilustração 9 - Código *SQL* para índice nAlojamento

Outro índice que poderá ter grande influência na diminuição da complexidade temporal da BD é o atributo Preco_Base visto que este também é constantemente consultado (cada vez que um alojamento é reservado). Deste modo, criamos esse índice usando o seguinte comando:

```
CREATE INDEX pAlojamento ON Alojamento (Preco_Base);
```

Ilustração 10 - Código SQL para índice pAlojamento

Antes da criação dos índices, a *Query* relativa a RE07 seria executada em 0.014 segundos.

Ilustração 11 - Tempo de execução antes de inserir índices

Após a criação do índice nAlojamento, ao executar a mesma *Query*, o tempo necessário diminui significativamente, demorando, neste caso, 0.00035 segundos.

Ilustração 12 - Tempo de execução após inserção de índice nAlojamento

No caso da criação do índice pAlojamento verifica-se novamente uma melhoria considerável, sendo que o novo tempo necessário para obter resposta à *Query* relativa a RE07 é 0.00041 segundos.

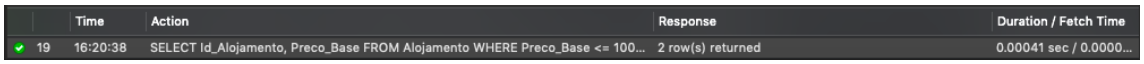

Ilustração 13 - Tempo de execução após inserção de índice pAlojamento

1.4. Procedimentos Implementados

De forma a ser possível alterar o valor de cada quarto decidimos implementar o seguinte procedimento, apresentando também o resultado desse mesmo procedimento aplicado ao Alojamento 1 com uma percentagem de 3%.

```
DELIMITER $$
CREATE PROCEDURE UpdatePrecoAlojamento (in Percentagem int, in Id int)
BEGIN
UPDATE Alojamento
  SET Preco_Base = Preco_Base * (1+Percentagem/100)
  WHERE Id_Alojamento = Id;
END $$

CALL UpdatePrecoAlojamento(3, 1);
```

Ilustração 14 - Código *SQL* relativo ao Procedimento UpdatePrecoAlojamento

Id_Alojamento	Numero	Edificio	Preco_Base	Lotacao	Numero_Quartos
1	1	1	195.70	6	3
NULL	NULL	NULL	NULL	NULL	NULL

Ilustração 15 - Resultado do Procedimento UpdatePrecoAlojamento

De seguida pensamos na possibilidade de alterar o salário de um funcionário, criando para procedimento apresentado a seguir, sucedido do resultado do mesmo aplicado ao funcionário 1 atribuindo-lhe um nova salário de 955€.

```
DELIMITER $$
CREATE PROCEDURE UpdateSalarioFuncionario (in novoSalario int, in Id int)
BEGIN
UPDATE Funcionario
  SET Salario = novoSalario
  WHERE Id_Funcionario = Id;
END $$

CALL UpdateSalarioFuncionario(955, 1);
```

Ilustração 16 - Código *SQL* relativo ao Procedimento UpdateSalarioFuncionario

Id_Funcionario	Nome	Telemovel	E-mail	Salario	Gerente
1	Luís Presa	239956636	lpresa@sapo.com	955	NULL
NULL	NULL	NULL	NULL	NULL	NULL

Ilustração 17 - Resultado do Procedimento UpdateSalarioFuncionario

2... Conceção e Implementação de um Sistema de Dados em *MongoDB*

2.1. Definição do Esquema da Base de Dados

Os dados no *MongoDB* têm um esquema flexível, altamente escalável com suporte de dados semiestruturados, não estruturados e complexos. Tendo em atenção estas vantagens decidimos migrar a nossa Base de Dados relacional para uma não relacional mantendo toda a informação que estava presente na inicial, tomando em atenção as decisões na sua modelação para não afetar o desempenho e a capacidade da mesma.

Os documentos incorporados (*embedded documents*) possuem um esquema próprio e estão incluídos noutros documentos. Já os subdocumentos possuem os mesmos recursos que os seus 'pais', mas não podem ser guardados individualmente. Se os dados dos documentos incorporados aumentarem de tamanho, então a melhor solução seria criar documentos por referência.

2.2. Criação da Base de Dados e das Coleções

Para implementar a Base de Dados, decidimos criar várias coleções: Alojamento, Cliente, Edificio, Funcionario e Reserva.

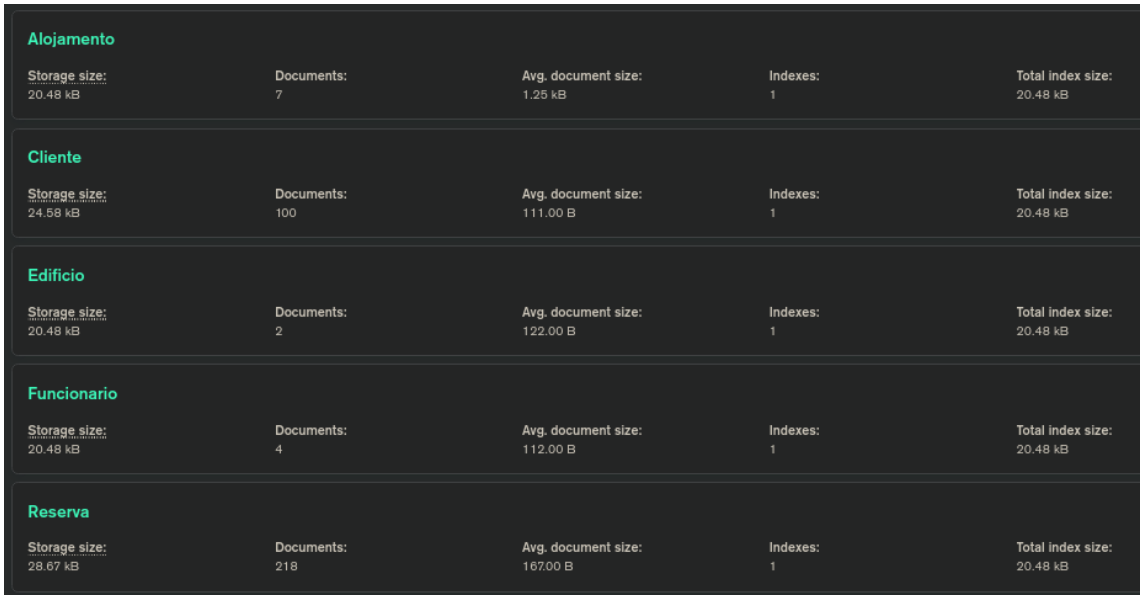

Ilustração 18 - Coleções em *MongoDB*

2.3. O Processo de Migração de Dados

O processo de migração de uma Base de Dados relacional para *NoSQL*, elimina as relações entre tabelas, mas impõe que o armazenamento seja feito de uma maneira específica para que consigamos responder às necessidades impostas pelo Sr. Luís. Este processo pode ser efetuado de várias maneiras, no nosso caso optámos por criar um *script* na linguagem *Python*, que recolhe todos os dados por meio de *Queries* em *SQL*, criando coleções para cada entidade e preenchendo os campos. O código fonte pode ser visto no Anexo 4.

2.4.Exploração de Dados em *MongoDB*

Nesta secção será apresentado o código *MongoDB* que permite obter resposta a cada requisito de exploração e exemplos da resposta com o povoamento atual.

- **RE01 – Listar os clientes que frequentaram o estabelecimento**

```
db.Cliente.aggregate([
    {
        $lookup: {
            from: "Reserva",
            localField: "_id",
            foreignField: "Cliente",
            as: "Reservas",
        },
    },
]);
```

Ilustração 19 - Código *MongoDB* relativo a RE01

```
[
  {
    _id: 1,
    Nome: 'Jéssica Abreu',
    Data_Nascimento: ISODate("2000-07-31T00:00:00.000Z"),
    'E-mail': 'sara37@hotmail.com',
    Telemovel: 928884516,
    Reservas: [
      {
        _id: 1,
        Funcionario: 1,
        Cliente: 1,
        Data_Inicio: ISODate("2020-01-07T00:00:00.000Z"),
        Data_Fim: ISODate("2020-01-10T00:00:00.000Z"),
        Adultos: 3,
        Criancas: 0,
        Alojamentos: [ { Alojamento: 1, Preco: 190 } ]
      }
    ]
  }
]
```

Ilustração 20 - Resultado da *Query* relativa a RE01

- **RE02 – Listar os edifícios e respetivos alojamentos**

```
db.Edificio.aggregate([
    {
        $lookup: {
            from: "Alojamento",
            localField: "_id",
            foreignField: "Edificio",
            as: "Alojamentos",
        },
    },
    {
        $project: {
            "Rua/Morada": 1,
            "Alojamentos.Numero": 1,
            "Alojamentos.Numero_Quartos": 1,
            "Alojamentos.Lotacao": 1,
        },
    },
]).sort({ _id: 1 });
```

Ilustração 21 - Código *MongoDB* relativo a RE02

```
[
  {
    _id: 1,
    'Rua/Morada': 'Rua Principal, 1',
    Alojamentos: [
      { Numero: 1, Lotacao: 6, Numero_Quartos: 3 },
      { Numero: 2, Lotacao: 4, Numero_Quartos: 2 },
      { Numero: 3, Lotacao: 4, Numero_Quartos: 2 }
    ]
  },
  {
    _id: 2,
    'Rua/Morada': 'Rua Principal, 2',
    Alojamentos: [
      { Numero: 1, Lotacao: 2, Numero_Quartos: 1 },
      { Numero: 2, Lotacao: 2, Numero_Quartos: 1 },
      { Numero: 3, Lotacao: 4, Numero_Quartos: 2 },
      { Numero: 4, Lotacao: 4, Numero_Quartos: 2 }
    ]
  }
]
```

Ilustração 22 - Resultado da *Query* relativa a RE02

- **RE03 – Listar todas as reservas realizadas num certo alojamento**

```
db.Alojamento.aggregate([
    { $match: {} },
    { $unwind: "$Reservas" },
    {
        $lookup: {
            from: "Reserva",
            localField: "Reservas.Reserva",
            foreignField: "_id",
            as: "Reserva",
        },
    },
    { $unwind: "$Reserva" },
    {
        $group: {
            _id: "$_id",
            Reservas: {
                $push: {
                    _id: "$Reserva._id",
                    Data_Inicio: "$Reserva.Data_Inicio",
                    Data_Fim: "$Reserva.Data_Fim",
                },
            },
        },
    },
]);
```

Ilustração 23 - Código *MongoDB* relativo a RE03

```
[
  {
    _id: 3,
    Reservas: [
      {
        _id: 2,
        Data_Inicio: ISODate("2020-01-08T00:00:00.000Z"),
        Data_Fim: ISODate("2020-01-11T00:00:00.000Z")
      },
      {
        _id: 4,
        Data_Inicio: ISODate("2020-01-17T00:00:00.000Z"),
        Data_Fim: ISODate("2020-01-20T00:00:00.000Z")
      },
      {
        _id: 9,
        Data_Inicio: ISODate("2020-02-04T00:00:00.000Z"),
        Data_Fim: ISODate("2020-02-07T00:00:00.000Z")
      },
      {
        _id: 15,
        Data_Inicio: ISODate("2020-02-26T00:00:00.000Z"),
        Data_Fim: ISODate("2020-02-29T00:00:00.000Z")
      },
      {
        _id: 32,
        Data_Inicio: ISODate("2020-05-06T00:00:00.000Z"),
        Data_Fim: ISODate("2020-05-09T00:00:00.000Z")
      },
```

Ilustração 24 - Resultado da *Query* relativa a RE0

- **RE04 – Listar todos os funcionários e respetivo responsável**

```
db.Funcionario.find({}, { Nome: 1, Gerente: 1 });
```

Ilustração 25 - Código *MongoDB* relativo a RE04

```
[
  { _id: 1, Nome: 'Luís Presa', Gerente: null },
  { _id: 2, Nome: 'Vasco Morais', Gerente: 1 },
  { _id: 3, Nome: 'Rafaela Campos-Paiva', Gerente: 1 },
  { _id: 4, Nome: 'Cláudio Baptista', Gerente: 1 }
]
```

Ilustração 26 - Resultado da *Query* relativa a RE04

- **RE05 – Calcular o número de vezes que cada alojamento foi reservado**

```
db.Alojamento.aggregate([
    { $match: {} },
    { $unwind: "$Reservas" },
    {
        $lookup: {
            from: "Reserva",
            localField: "Reservas.Reserva",
            foreignField: "_id",
            as: "Reserva",
        },
    },
    { $unwind: "$Reserva" },
    {
        $group: {
            _id: "$_id",
            Reservas: { $sum: 1 },
        },
    },
]);
```

Ilustração 27 - Código *MongoDB* relativo a RE05

```
[
  { _id: 3, Reservas: 32 },
  { _id: 5, Reservas: 5 },
  { _id: 4, Reservas: 7 },
  { _id: 6, Reservas: 34 },
  { _id: 1, Reservas: 89 },
  { _id: 7, Reservas: 23 },
  { _id: 2, Reservas: 28 }
]
```

Ilustração 28 - Resultado da *Query* relativa a RE05

- **RE06 – Calcular o número de pedidos que um funcionário processou**

```
db.Funcionario.aggregate([
    {
        $lookup: {
            from: "Reserva",
            localField: "_id",
            foreignField: "Funcionario",
            as: "Reservas",
        },
    },
    { $unwind: "$Reservas" },
    {
        $group: {
            _id: "$_id",
            Nome: { $first: "$Nome" },
            Reservas: { $sum: 1 },
        },
    },
]).sort({ _id: 1 });
```

Ilustração 29 - Código *MongoDB* relativo a RE06

```
[
  { _id: 1, Nome: 'Luis Presa', Reservas: 51 },
  { _id: 2, Nome: 'Vasco Morais', Reservas: 57 },
  { _id: 3, Nome: 'Rafaela Campos-Paiva', Reservas: 55 },
  { _id: 4, Nome: 'Cláudio Baptista', Reservas: 55 }
]
```

Ilustração 30 - Resultado da *Query* relativa a RE06

- **RE07 – Listar os quartos que estão disponíveis até um certo preço**

```
db.Alojamento.find({ Preco_Base: { $lte: 100 } });
```

Ilustração 31 - Código *MongoDB* relativo a RE07

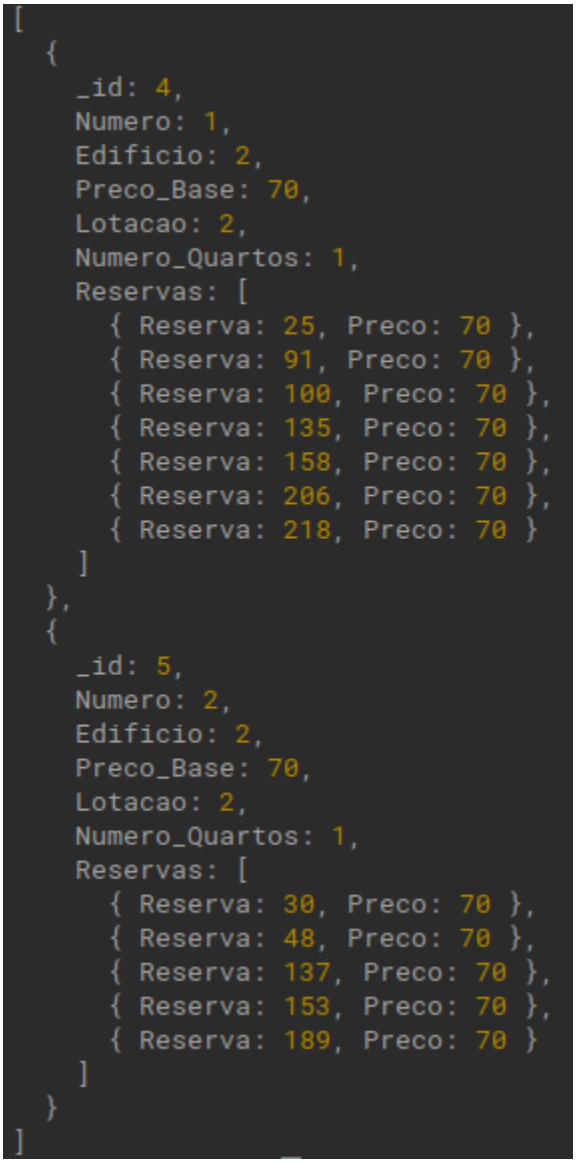

Ilustração 32 - Resultado da *Query* relativa a RE07

- **RE08 – Listar os clientes por ordem decrescente de dinheiro gasto**

```
db.Reserva.aggregate([
    {
        $lookup: {
            from: "Cliente",
            localField: "Cliente",
            foreignField: "_id",
            as: "ClienteCol",
        },
    },
    { $unwind: "$ClienteCol" },
    { $unwind: "$Alojamentos" },
    {
        $group: {
            _id: "$Cliente",
            Nome: { $first: "$ClienteCol.Nome" },
            Total_Gasto: { $sum: "$Alojamentos.Preco" },
        },
    },
]).sort({ Total_Gasto: -1, _id: 1 });
```

Ilustração 33 - Código *MogoDB* relativo a RE08

```
[
  { _id: 74, Nome: 'Vitor Vaz-Antunes', Total_Gasto: 570 },
  { _id: 5, Nome: 'Ivo Nascimento', Total_Gasto: 510 },
  { _id: 27, Nome: 'Igor Simões', Total_Gasto: 510 },
  { _id: 28, Nome: 'Mara Mota', Total_Gasto: 510 },
  { _id: 50, Nome: 'Carlota Nunes', Total_Gasto: 510 },
  { _id: 52, Nome: 'Rita-Beatriz Esteves', Total_Gasto: 510 },
  { _id: 60, Nome: 'Adriana Soares', Total_Gasto: 510 },
  { _id: 73, Nome: 'Anita Gomes', Total_Gasto: 510 },
  { _id: 80, Nome: 'Camila Henriques', Total_Gasto: 510 },
  { _id: 89, Nome: 'Matilde Castro', Total_Gasto: 510 },
  { _id: 2, Nome: 'Samuel Pinto', Total_Gasto: 450 },
  { _id: 3, Nome: 'Mauro Valente', Total_Gasto: 450 },
  { _id: 15, Nome: 'Bernardo Pinho', Total_Gasto: 450 },
  { _id: 17, Nome: 'Paulo Assunção', Total_Gasto: 450 },
  { _id: 25, Nome: 'Camila Cardoso', Total_Gasto: 450 },
  { _id: 29, Nome: 'Enzo de Nascimento', Total_Gasto: 450 },
  { _id: 31, Nome: 'Pilar Freitas', Total_Gasto: 450 },
  { _id: 33, Nome: 'Raquel Branco', Total_Gasto: 450 },
  { _id: 34, Nome: 'Simão Miranda', Total_Gasto: 450 },
  { _id: 35, Nome: 'Benedita Cruz', Total_Gasto: 450 }
]
```

Ilustração 34 - Resultado da *Query* relativa a RE08

3... Conclusões e Trabalho Futuro

Findado o nosso trabalho, continuamos a considerar que a nossa base de dados não é perfeita, sendo exemplo disso o posicionamento das datas da reserva que, para permitir uma verificação mais eficiente da disponibilidade de um quarto, deveriam estar na relação Reserva_Alojamento. Não obstante, consideramos que os objetivos foram cumpridos e foi criada uma Base de Dados consistente e robusta capaz de apoiar a gestão do Alojamento do Senhor Luís.

Quanto à passagem de *SQL* para *NoSQL,* sentimos uma maior liberdade na escrita de *Queries* e nos modelos de dados, o que nos deu a impressão de possivelmente ser mais fácil trabalhar com *MongoDB* caso estivéssemos a desenvolver um projeto sem o auxílio de um especialista de Base de Dados, devido à sua maior flexibilidade.

Inicialmente, tivemos alguma dificuldade em nos adaptarmos ao a*ggregation Framework,* mas após consultarmos o mapeamento de conceitos de *SQL* para *MongoDB* tudo se tornou mais fácil.

Concluindo, apesar de termos tido algumas dificuldades, conseguimos superá-las e atingir as nossas espectativas.

5 Referências Bibliográficas

MongoDB Compass - https://docs.mongodb.com/compass/current/query/filter/

Thomas M. Connolly, Carolyn E. Begg - Database Systems: A Practical Approach to Design, Implementation and Management - 4th Edition

Feliz Gouveia – Fundamentos de Bases de Dados

6 Anexos

I. Anexo 1 – Novo *Script* de Povoamento

```
INSERT INTO Cliente (`Nome`, `Data_Nascimento`, `E-mail`, `Telemovel`) VALUES
('Jéssica Abreu', '2000-07-31', 'sara37@hotmail.com', '928884516'),
('Samuel Pinto', '1993-01-29', 'vitoria81@sapo.pt', '937891801'),
('Mauro Valente', '1968-01-13', 'iaraamorim@sapo.pt', '924696238'),
('Dinis Ramos', '1982-10-13', 'wilson65@clix.pt', '966185225'),
('Ivo Nascimento', '1989-10-22', 'helenaamaral@gmail.com', '274873305'),
('Vasco Monteiro', '1973-01-11', 'moreiraivan@gmail.com', '968986430'),
('William Pinto', '1996-02-07', 'leonardopinto@clix.pt', '257597846'),
('Emma Jesus', '1962-01-02', 'irinaaraujo@sapo.pt', '969513367'),
('Xavier Anjos', '1997-10-05', 'sofianascimento@hotmail.com', '245808031'),
('Kévim Amaral', '1991-02-18', 'rodrigoalves@hotmail.com', '208118237'),
('Jéssica Costa-Assunção', '1995-03-09', 'afonseca@sapo.pt', '920741120'),
('Gaspar Magalhães', '1966-11-29', 'jose27@sapo.pt', '913670849'),
('Ana Batista', '1992-11-13', 'castrojessica@sapo.pt', '936434352'),
('Alice Vaz', '1965-05-01', 'salmeida@hotmail.com', '931494860'),
('Bernardo Pinho', '1965-01-28', 'ymota@gmail.com', '962060367'),
('Rodrigo Costa-Oliveira', '1981-10-12', 'jaimemiranda@hotmail.com', '270953752'),
('Paulo Assunção', '1993-07-11', 'nlourenco@clix.pt', '910458214'),
('Yasmin Melo', '1991-02-22', 'daniel66@sapo.pt', '963262891'),
('Matias do Macedo', '1991-05-17', 'paivahelena@clix.pt', '936229267'),
('Cristiano do Tavares', '1998-04-16', 'gustavodomingues@sapo.pt', '939856768'),
('Luísa Antunes', '1997-02-03', 'frederico34@gmail.com', '915568051'),
('Matilde Assunção', '1976-01-15', 'alice91@sapo.pt', '935499807'),
('Hugo Castro', '1996-05-12', 'rubenvalente@gmail.com', '920511471'),
('Iara-Jéssica Melo', '1973-03-22', 'erikaramos@sapo.pt', '922047243'),
('Camila Cardoso', '1963-09-08', 'matiasborges@gmail.com', '257910957'),
('Alice-Lia Araújo', '1993-07-06', 'gabrielmendes@sapo.pt', '967216506'),
('Igor Simões', '1964-12-13', 'santiago19@clix.pt', '911264826'),
('Mara Mota', '1963-03-02', 'goncalo64@sapo.pt', '927742828'),
('Enzo de Nascimento', '1966-02-23', 'mariana39@sapo.pt', '913101201'),
('Flor Gaspar', '1973-11-29', 'ovicente@clix.pt', '965935576'),
('Pilar Freitas', '1996-11-21', 'renatojesus@hotmail.com', '914728644'),
('Martim Borges', '1983-12-26', 'barbosahenrique@clix.pt', '963263525'),
```

('Raquel Branco', '1976-06-30', 'barrospetra@sapo.pt', '922660193'),
('Simão Miranda', '1984-06-20', 'soraiafonseca@hotmail.com', '939369268'),
('Benedita Cruz', '1968-07-09', 'flor98@sapo.pt', '969849071'),
('Ismael do Brito', '1970-03-18', 'francisconeves@hotmail.com', '928557385'),
('Fernando Pinheiro', '1974-08-16', 'noliveira@clix.pt', '250482074'),
('Débora Henriques', '1969-06-21', 'wlourenco@clix.pt', '930391267'),
('Tomás Macedo', '1968-05-23', 'goncalvesmia@clix.pt', '929834901'),
('Mauro da Marques', '1994-01-13', 'cesar61@sapo.pt', '939039007'),
('Larissa Neto', '1980-02-04', 'rodrigo90@gmail.com', '210533769'),
('Gabriel Morais', '1981-07-27', 'lmachado@clix.pt', '932679475'),
('Pilar Freitas', '1975-12-11', 'miriamsousa@clix.pt', '916670187'),
('Ismael Fernandes', '1991-01-04', 'martim69@sapo.pt', '216713127'),
('Ângelo Coelho', '1981-10-04', 'usantos@sapo.pt', '939858062'),
('Diego do Moura', '1996-11-03', 'sebastiao14@clix.pt', '923166387'),
('Joel Pinho', '1975-05-12', 'bernardogoncalves@sapo.pt', '923152845'),
('Enzo Loureiro', '1984-04-19', 'almeidamarco@clix.pt', '928856139'),
('Carlota Lima', '1993-09-12', 'frederico13@hotmail.com', '926464803'),
('Carlota Nunes', '1985-01-03', 'maiateresa@clix.pt', '933307420'),
('Edgar Costa', '1974-07-10', 'edgarnogueira@gmail.com', '930268523'),
('Rita-Beatriz Esteves', '1973-05-05', 'vvieira@gmail.com', '917944651'),
('Henrique da Marques', '1984-03-05', 'anitaanjos@gmail.com', '963297373'),
('Adriana Castro', '1967-01-20', 'nmoura@hotmail.com', '924527906'),
('Isabel Gonçalves', '1971-03-22', 'brancofernando@sapo.pt', '912416358'),
('Cláudio Carvalho', '1997-09-18', 'azevedoisabel@sapo.pt', '927441869'),
('Núria Cunha', '1994-07-19', 'marcio18@sapo.pt', '278931173'),
('Luana Ramos', '1968-06-03', 'rui82@hotmail.com', '969361513'),
('Cristiano Alves-Rodrigues', '1978-08-10', 'carlotacruz@gmail.com', '964820549'),
('Adriana Soares', '1995-05-05', 'diogomonteiro@clix.pt', '961125844'),
('Núria Moreira', '1992-06-14', 'rafael30@gmail.com', '283418663'),
('Irina Vaz', '1991-06-08', 'qreis@clix.pt', '926057596'),
('Martim Jesus', '1988-08-06', 'arturbarbosa@sapo.pt', '273483816'),
('Artur Pinto', '1979-08-10', 'micael42@hotmail.com', '961024930'),
('Diogo Vicente', '1995-07-26', 'jaime86@hotmail.com', '968428671'),
('Tiago de Leite', '1991-09-03', 'tassuncao@clix.pt', '939570809'),
('Kévim Baptista', '1971-03-02', 'tlopes@gmail.com', '223926647'),
('Ema Barbosa', '1998-01-18', 'rcarvalho@hotmail.com', '232532507'),
('Helena Marques', '1973-03-13', 'leandroanjos@sapo.pt', '929187867'),

('Ivan Brito', '1967-05-06', 'carlosbarros@clix.pt', '910816985'),
('Alexandre do Moreira', '1997-06-10', 'baptistaleticia@gmail.com', '967463498'),
('Cláudio Nunes', '1976-03-07', 'claudio56@sapo.pt', '910046600'),
('Anita Gomes', '1975-03-30', 'saraleite@hotmail.com', '961831625'),
('Vítor Vaz-Antunes', '1980-11-23', 'torresclaudio@gmail.com', '926551274'),
('Renata Amorim', '1980-07-03', 'coelhojoao@gmail.com', '918069378'),
('Brian Oliveira', '1997-06-28', 'zcruz@gmail.com', '961032985'),
('Renato Campos', '1992-06-02', 'icosta@hotmail.com', '966908109'),
('Miriam Coelho', '1990-09-26', 'patricia07@clix.pt', '273807386'),
('Diego Azevedo', '2000-03-25', 'cristianobarros@clix.pt', '960510451'),
('Camila Henriques', '1980-10-09', 'eduardamartins@clix.pt', '916843211'),
('William Pires', '1966-07-22', 'ferreiramicael@gmail.com', '969768331'),
('Clara Baptista', '1979-05-27', 'mateus49@gmail.com', '280434449'),
('Kelly Jesus', '1977-12-13', 'qmorais@hotmail.com', '964714916'),
('Andreia Lourenço', '1997-10-17', 'vmota@sapo.pt', '920796562'),
('Rafael Pinho', '1992-09-05', 'biancagomes@sapo.pt', '934292154'),
('Rodrigo Anjos', '1979-10-06', 'amaralemilia@sapo.pt', '960483152'),
('Débora Moreira', '1994-10-16', 'beatrizmaia@clix.pt', '961581160'),
('Letícia-Constança Ferreira', '1982-02-13', 'sousapatricia@sapo.pt', '939958353'),
('Matilde Castro', '1960-08-05', 'aliciabaptista@hotmail.com', '964322164'),
('Eva do Melo', '1999-06-11', 'wleal@hotmail.com', '922601612'),
('Daniel Nogueira-Oliveira', '1996-09-12', 'glima@sapo.pt', '926171360'),
('Anita Paiva', '1994-11-24', 'salmeida@sapo.pt', '921921406'),
('Lourenço Alves', '1976-07-07', 'miguelpinho@sapo.pt', '916482758'),
('Vasco Lourenço-Batista', '1969-09-16', 'matildeazevedo@hotmail.com', '937587871'),
('Afonso Oliveira', '1994-12-05', 'victoriaoliveira@hotmail.com', '253892400'),
('Alice Melo', '1994-06-02', 'psousa@sapo.pt', '933657169'),
('Xavier Tavares', '1972-05-26', 'yneves@hotmail.com', '272434166'),
('José Andrade', '1985-06-27', 'camila88@hotmail.com', '927823376'),
('Noa Vieira', '1968-11-08', 'vbarros@clix.pt', '969521914'),
('Patrícia-Inês Pacheco', '1974-07-28', 'alicecastro@sapo.pt', '925281463');

INSERT INTO Funcionario (`Nome`, `Telemovel`, `E-mail`, `Salario`, `Gerente`) VALUES
('Luís Presa', '239956636', 'lpresa@sapo.com', 1515, NULL),
('Vasco Morais', '927124008', 'xcarneiro@gmail.com', 665, 1),
('Rafaela Campos-Paiva', '967127667', 'araujoalicia@hotmail.com', 665, 1),
('Cláudio Baptista', '282992682', 'martim65@gmail.com', 665, 1);

```
INSERT INTO Edificio (`Rua/Morada`, `Localidade/Morada`, `Codigo_Postal/Morada`) VALUES
('Rua Principal, 1', 'Chaves', '5400-623'),
('Rua Principal, 2', 'Chaves', '5400-623');

INSERT INTO Alojamento (`Edificio`, `Numero`, `Preco_Base`, `Lotacao`, `Numero_Quartos`) VALUES
(1, 1, 190, 6, 3),
(1, 2, 130, 4, 2),
(1, 3, 130, 4, 2),
(2, 1, 70, 2, 1),
(2, 2, 70, 2, 1),
(2, 3, 130, 4, 2),
(2, 4, 130, 4, 2);

INSERT INTO Reserva (`Funcionario`, `Cliente`, `Data_Inicio`, `Data_Fim`, `Adultos`, `Criancas`) VALUES
(1, 1, '2020-01-07', '2020-01-10', 3, 0),
(2, 2, '2020-01-08', '2020-01-11', 1, 3),
(1, 2, '2020-01-14', '2020-01-17', 3, 1),
(1, 2, '2020-01-17', '2020-01-20', 3, 0),
(2, 3, '2020-01-23', '2020-01-26', 2, 2),
(3, 3, '2020-01-23', '2020-01-26', 1, 3),
(1, 3, '2020-01-30', '2020-02-02', 3, 3),
(3, 4, '2020-02-01', '2020-02-04', 2, 0),
(3, 4, '2020-02-04', '2020-02-07', 1, 0),
(2, 4, '2020-02-06', '2020-02-09', 2, 1),
(2, 5, '2020-02-13', '2020-02-16', 1, 2),
(2, 5, '2020-02-16', '2020-02-19', 2, 3),
(1, 5, '2020-02-22', '2020-02-25', 3, 1),
(1, 6, '2020-02-24', '2020-02-27', 1, 3),
(3, 6, '2020-02-26', '2020-02-29', 3, 0),
(3, 6, '2020-03-01', '2020-03-04', 2, 2),
(3, 7, '2020-03-06', '2020-03-09', 3, 3),
(3, 7, '2020-03-10', '2020-03-13', 1, 1),
(4, 8, '2020-03-13', '2020-03-16', 3, 3),
(4, 8, '2020-03-23', '2020-03-26', 2, 1),
(1, 9, '2020-03-27', '2020-03-30', 1, 2),
(3, 9, '2020-03-31', '2020-04-03', 2, 3),
```

(1, 10, '2020-04-02', '2020-04-05', 2, 3),

(3, 11, '2020-04-06', '2020-04-09', 1, 0),

(4, 11, '2020-04-12', '2020-04-15', 2, 0),

(3, 12, '2020-04-14', '2020-04-17', 1, 2),

(4, 12, '2020-04-17', '2020-04-20', 2, 2),

(1, 12, '2020-04-22', '2020-04-25', 3, 1),

(2, 13, '2020-04-22', '2020-04-25', 3, 1),

(4, 14, '2020-05-02', '2020-05-05', 1, 1),

(2, 15, '2020-05-06', '2020-05-09', 2, 1),

(2, 15, '2020-05-06', '2020-05-09', 3, 0),

(1, 15, '2020-05-12', '2020-05-15', 2, 1),

(2, 16, '2020-05-16', '2020-05-19', 3, 2),

(2, 16, '2020-05-17', '2020-05-20', 2, 0),

(2, 17, '2020-05-26', '2020-05-29', 3, 1),

(2, 17, '2020-05-29', '2020-06-01', 1, 3),

(3, 17, '2020-05-30', '2020-06-02', 1, 1),

(4, 18, '2020-06-05', '2020-06-08', 3, 2),

(2, 19, '2020-06-11', '2020-06-14', 1, 2),

(3, 20, '2020-06-09', '2020-06-12', 1, 2),

(2, 20, '2020-06-15', '2020-06-18', 3, 2),

(2, 21, '2020-06-23', '2020-06-26', 2, 0),

(4, 21, '2020-06-21', '2020-06-24', 2, 0),

(2, 21, '2020-06-26', '2020-06-29', 1, 2),

(1, 22, '2020-07-01', '2020-07-04', 2, 2),

(3, 23, '2020-07-09', '2020-07-12', 1, 3),

(1, 23, '2020-07-13', '2020-07-16', 1, 1),

(3, 23, '2020-07-14', '2020-07-17', 1, 2),

(1, 24, '2020-07-18', '2020-07-21', 2, 2),

(4, 24, '2020-07-25', '2020-07-28', 3, 0),

(4, 25, '2020-07-25', '2020-07-28', 3, 1),

(1, 25, '2020-08-02', '2020-08-05', 1, 3),

(4, 25, '2020-07-31', '2020-08-03', 2, 1),

(3, 26, '2020-08-06', '2020-08-09', 3, 1),

(4, 26, '2020-08-14', '2020-08-17', 3, 1),

(3, 27, '2020-08-13', '2020-08-16', 3, 3),

(4, 27, '2020-08-20', '2020-08-23', 2, 3),

(2, 27, '2020-08-25', '2020-08-28', 1, 1),

(3, 28, '2020-08-26', '2020-08-29', 2, 3),

(1, 28, '2020-09-03', '2020-09-06', 3, 2),

(3, 28, '2020-09-02', '2020-09-05', 2, 1),

(3, 29, '2020-09-10', '2020-09-13', 1, 2),

(4, 29, '2020-09-09', '2020-09-12', 3, 2),

(3, 29, '2020-09-19', '2020-09-22', 2, 1),

(3, 30, '2020-09-19', '2020-09-22', 1, 3),

(3, 30, '2020-09-23', '2020-09-26', 1, 1),

(1, 30, '2020-09-28', '2020-10-01', 2, 2),

(2, 31, '2020-10-05', '2020-10-08', 3, 3),

(4, 31, '2020-10-04', '2020-10-07', 2, 2),

(2, 31, '2020-10-11', '2020-10-14', 2, 1),

(3, 32, '2020-10-14', '2020-10-17', 3, 3),

(4, 33, '2020-10-21', '2020-10-24', 2, 0),

(3, 33, '2020-10-21', '2020-10-24', 1, 3),

(1, 33, '2020-10-25', '2020-10-28', 2, 3),

(2, 34, '2020-10-31', '2020-11-03', 1, 2),

(4, 34, '2020-11-02', '2020-11-05', 1, 2),

(2, 34, '2020-11-07', '2020-11-10', 2, 0),

(4, 35, '2020-11-10', '2020-11-13', 3, 2),

(2, 35, '2020-11-16', '2020-11-19', 1, 2),

(3, 35, '2020-11-19', '2020-11-22', 2, 2),

(4, 36, '2020-11-25', '2020-11-28', 2, 0),

(3, 36, '2020-11-29', '2020-12-02', 1, 0),

(2, 36, '2020-12-01', '2020-12-04', 3, 3),

(1, 37, '2020-12-03', '2020-12-06', 2, 2),

(4, 38, '2020-12-10', '2020-12-13', 2, 1),

(3, 38, '2020-12-10', '2020-12-13', 1, 3),

(3, 38, '2020-12-20', '2020-12-23', 1, 0),

(3, 39, '2020-12-20', '2020-12-23', 3, 0),

(1, 39, '2020-12-28', '2020-12-31', 3, 0),

(3, 39, '2020-12-26', '2020-12-29', 1, 0),

(3, 40, '2021-01-01', '2021-01-04', 1, 0),

(1, 41, '2021-01-09', '2021-01-12', 3, 3),

(2, 42, '2021-01-09', '2021-01-12', 1, 3),

(3, 43, '2021-01-13', '2021-01-16', 2, 1),

(2, 43, '2021-01-19', '2021-01-22', 3, 1),

(4, 43, '2021-01-24', '2021-01-27', 2, 3),

(2, 44, '2021-01-25', '2021-01-28', 3, 3),

(1, 45, '2021-01-27', '2021-01-30', 3, 2),

(4, 45, '2021-02-02', '2021-02-05', 2, 0),

(2, 45, '2021-02-10', '2021-02-13', 3, 2),

(4, 46, '2021-02-11', '2021-02-14', 3, 0),

(4, 46, '2021-02-12', '2021-02-15', 2, 2),

(3, 46, '2021-02-17', '2021-02-20', 2, 1),

(4, 47, '2021-02-22', '2021-02-25', 2, 3),

(4, 47, '2021-03-01', '2021-03-04', 1, 1),

(1, 48, '2021-03-03', '2021-03-06', 1, 3),

(4, 48, '2021-03-04', '2021-03-07', 2, 3),

(4, 49, '2021-03-08', '2021-03-11', 3, 2),

(1, 50, '2021-03-16', '2021-03-19', 3, 0),

(4, 50, '2021-03-19', '2021-03-22', 3, 2),

(1, 50, '2021-03-25', '2021-03-28', 2, 3),

(1, 51, '2021-03-24', '2021-03-27', 3, 2),

(4, 51, '2021-03-28', '2021-03-31', 3, 3),

(1, 52, '2021-04-03', '2021-04-06', 1, 1),

(3, 52, '2021-04-06', '2021-04-09', 1, 2),

(2, 52, '2021-04-12', '2021-04-15', 1, 2),

(2, 53, '2021-04-15', '2021-04-18', 3, 1),

(4, 53, '2021-04-23', '2021-04-26', 2, 2),

(2, 54, '2021-04-27', '2021-04-30', 1, 0),

(3, 55, '2021-04-25', '2021-04-28', 2, 2),

(4, 56, '2021-05-05', '2021-05-08', 3, 3),

(4, 56, '2021-05-07', '2021-05-10', 3, 2),

(1, 57, '2021-05-13', '2021-05-16', 2, 2),

(4, 57, '2021-05-11', '2021-05-14', 3, 2),

(4, 58, '2021-05-16', '2021-05-19', 3, 0),

(3, 58, '2021-05-19', '2021-05-22', 2, 3),

(4, 58, '2021-05-25', '2021-05-28', 2, 1),

(1, 59, '2021-06-02', '2021-06-05', 3, 1),

(4, 59, '2021-06-02', '2021-06-05', 1, 2),

(4, 59, '2021-06-09', '2021-06-12', 3, 1),

(4, 60, '2021-06-13', '2021-06-16', 3, 3),

(1, 60, '2021-06-12', '2021-06-15', 3, 3),

(3, 60, '2021-06-18', '2021-06-21', 1, 3),

(2, 61, '2021-06-22', '2021-06-25', 1, 0),

(4, 61, '2021-06-24', '2021-06-27', 3, 0),

(2, 61, '2021-07-04', '2021-07-07', 2, 0),

(4, 62, '2021-07-04', '2021-07-07', 1, 3),

(2, 62, '2021-07-06', '2021-07-09', 3, 3),

(3, 63, '2021-07-13', '2021-07-16', 1, 3),

(3, 63, '2021-07-19', '2021-07-22', 2, 0),

(2, 64, '2021-07-18', '2021-07-21', 2, 0),

(3, 64, '2021-07-27', '2021-07-30', 3, 1),

(2, 64, '2021-07-26', '2021-07-29', 1, 1),

(4, 65, '2021-08-05', '2021-08-08', 1, 3),

(1, 65, '2021-08-03', '2021-08-06', 3, 2),

(3, 66, '2021-08-09', '2021-08-12', 1, 2),

(3, 66, '2021-08-14', '2021-08-17', 3, 3),

(1, 67, '2021-08-21', '2021-08-24', 2, 1),

(2, 67, '2021-08-25', '2021-08-28', 3, 2),

(1, 67, '2021-08-25', '2021-08-28', 3, 1),

(2, 68, '2021-09-01', '2021-09-04', 2, 0),

(1, 69, '2021-08-31', '2021-09-03', 1, 0),

(3, 69, '2021-09-10', '2021-09-13', 2, 2),

(3, 69, '2021-09-08', '2021-09-11', 3, 1),

(2, 70, '2021-09-15', '2021-09-18', 3, 1),

(2, 71, '2021-09-16', '2021-09-19', 1, 3),

(2, 71, '2021-09-21', '2021-09-24', 1, 0),

(4, 72, '2021-09-27', '2021-09-30', 3, 2),

(1, 72, '2021-10-02', '2021-10-05', 1, 1),

(2, 72, '2021-10-03', '2021-10-06', 3, 1),

(2, 73, '2021-10-07', '2021-10-10', 3, 3),

(4, 73, '2021-10-11', '2021-10-14', 3, 3),

(1, 73, '2021-10-14', '2021-10-17', 3, 0),

(1, 74, '2021-10-19', '2021-10-22', 2, 3),

(2, 74, '2021-10-28', '2021-10-31', 2, 3),

(4, 74, '2021-10-30', '2021-11-02', 3, 3),

(2, 75, '2021-11-02', '2021-11-05', 2, 3),

(3, 76, '2021-11-08', '2021-11-11', 1, 0),

(3, 77, '2021-11-07', '2021-11-10', 3, 2),

(2, 77, '2021-11-16', '2021-11-19', 3, 3),

(4, 78, '2021-11-16', '2021-11-19', 2, 0),

(1, 78, '2021-11-19', '2021-11-22', 1, 3),

(1, 79, '2021-11-23', '2021-11-26', 3, 3),

(4, 80, '2021-12-01', '2021-12-04', 3, 0),

(1, 80, '2021-12-01', '2021-12-04', 2, 1),

(2, 80, '2021-12-07', '2021-12-10', 3, 2),

(3, 81, '2021-12-12', '2021-12-15', 2, 3),

(3, 81, '2021-12-13', '2021-12-16', 3, 3),

(2, 82, '2021-12-20', '2021-12-23', 2, 3),

(1, 83, '2021-12-22', '2021-12-25', 1, 3),

(2, 84, '2021-12-31', '2022-01-03', 2, 1),

(1, 85, '2021-12-31', '2022-01-03', 3, 1),

(1, 85, '2022-01-07', '2022-01-10', 2, 3),

(3, 86, '2022-01-07', '2022-01-10', 2, 1),

(1, 86, '2022-01-15', '2022-01-18', 1, 2),

(2, 86, '2022-01-17', '2022-01-20', 3, 3),

(4, 87, '2022-01-23', '2022-01-26', 2, 1),

(4, 87, '2022-01-22', '2022-01-25', 2, 0),

(3, 88, '2022-01-28', '2022-01-31', 3, 2),

(4, 88, '2022-02-03', '2022-02-06', 2, 1),

(4, 89, '2022-02-03', '2022-02-06', 2, 1),

(4, 89, '2022-02-13', '2022-02-16', 2, 3),

(2, 89, '2022-02-17', '2022-02-20', 1, 2),

(1, 90, '2022-02-19', '2022-02-22', 3, 2),

(1, 90, '2022-02-21', '2022-02-24', 1, 2),

(2, 91, '2022-02-23', '2022-02-26', 2, 1),

(3, 91, '2022-03-04', '2022-03-07', 3, 0),

(1, 92, '2022-03-06', '2022-03-09', 3, 2),

(1, 92, '2022-03-09', '2022-03-12', 2, 1),

(4, 93, '2022-03-14', '2022-03-17', 1, 3),

(1, 93, '2022-03-19', '2022-03-22', 2, 1),

(2, 93, '2022-03-23', '2022-03-26', 2, 3),

(2, 94, '2022-03-27', '2022-03-30', 3, 2),

(3, 94, '2022-03-31', '2022-04-03', 1, 2),

(4, 95, '2022-04-04', '2022-04-07', 1, 0),

(3, 95, '2022-04-10', '2022-04-13', 2, 3),

(2, 95, '2022-04-14', '2022-04-17', 1, 1),

(1, 96, '2022-04-17', '2022-04-20', 3, 0),

(3, 97, '2022-04-22', '2022-04-25', 3, 1),

(2, 97, '2022-04-23', '2022-04-26', 3, 1),

(1, 97, '2022-04-28', '2022-05-01', 2, 2),

(3, 98, '2022-04-29', '2022-05-02', 3, 1),

(4, 98, '2022-05-04', '2022-05-07', 3, 2),

(2, 99, '2022-05-10', '2022-05-13', 2, 1),

(2, 99, '2022-05-14', '2022-05-17', 3, 1),

(4, 100, '2022-05-20', '2022-05-23', 2, 2),

(1, 100, '2022-05-18', '2022-05-21', 2, 0);

INSERT INTO Reserva_Alojamento (`Reserva`, `Alojamento`, `Preco`) VALUES

(1, 1, 190),

(2, 3, 130),

(3, 1, 190),

(4, 3, 130),

(5, 6, 130),

(6, 6, 130),

(7, 1, 190),

(8, 6, 130),

(9, 3, 130),

(10, 6, 130),

(11, 2, 130),

(12, 1, 190),

(13, 1, 190),

(14, 6, 130),

(15, 3, 130),

(16, 2, 130),

(17, 1, 190),

(18, 6, 130),

(19, 1, 190),

(20, 2, 130),

(21, 2, 130),

(22, 1, 190),

(23, 1, 190),

(24, 2, 130),

(25, 4, 70),

(26, 6, 130),

(27, 7, 130),

(28, 6, 130),

(29, 1, 190),

(30, 5, 70),

(31, 1, 190),

(32, 3, 130),

(33, 6, 130),

(34, 1, 190),

(35, 2, 130),

(36, 1, 190),

(37, 3, 130),

(38, 3, 130),

(39, 1, 190),

(40, 1, 190),

(41, 3, 130),

(42, 1, 190),

(43, 3, 130),

(44, 3, 130),

(45, 2, 130),

(46, 7, 130),

(47, 1, 190),

(48, 5, 70),

(49, 3, 130),

(50, 3, 130),

(51, 2, 130),

(52, 7, 130),

(53, 1, 190),

(54, 6, 130),

(55, 6, 130),

(56, 3, 130),

(57, 1, 190),

(58, 1, 190),

(59, 6, 130),

(60, 1, 190),

(61, 1, 190),

(62, 2, 130),

(63, 7, 130),

(64, 1, 190),

(65, 7, 130),

(66, 7, 130),

(67, 7, 130),

(68, 3, 130),

(69, 1, 190),

(70, 6, 130),

(71, 6, 130),

(72, 1, 190),

(73, 7, 130),

(74, 6, 130),

(75, 1, 190),

(76, 3, 130),

(77, 1, 190),

(78, 2, 130),

(79, 1, 190),

(80, 2, 130),

(81, 6, 130),

(82, 3, 130),

(83, 3, 130),

(84, 1, 190),

(85, 6, 130),

(86, 6, 130),

(87, 3, 130),

(88, 7, 130),

(89, 6, 130),

(90, 3, 130),

(91, 4, 70),

(92, 3, 130),

(93, 1, 190),

(94, 1, 190),

(95, 7, 130),

(96, 6, 130),

(97, 1, 190),

(98, 1, 190),

(99, 1, 190),
(100, 4, 70),
(101, 1, 190),
(102, 2, 130),
(103, 2, 130),
(104, 6, 130),
(105, 1, 190),
(106, 1, 190),
(107, 7, 130),
(108, 1, 190),
(109, 1, 190),
(110, 7, 130),
(111, 1, 190),
(112, 1, 190),
(113, 1, 190),
(114, 1, 190),
(115, 1, 190),
(116, 1, 190),
(117, 3, 130),
(118, 6, 130),
(119, 1, 190),
(120, 6, 130),
(121, 3, 130),
(122, 1, 190),
(123, 1, 190),
(124, 7, 130),
(125, 1, 190),
(126, 7, 130),
(127, 1, 190),
(128, 6, 130),
(129, 2, 130),
(130, 1, 190),
(131, 2, 130),
(132, 1, 190),
(133, 1, 190),
(134, 2, 130),
(135, 4, 70),

(136, 2, 130),
(137, 5, 70),
(138, 6, 130),
(139, 1, 190),
(140, 7, 130),
(141, 1, 190),
(142, 7, 130),
(143, 6, 130),
(144, 2, 130),
(145, 1, 190),
(146, 1, 190),
(147, 3, 130),
(148, 1, 190),
(149, 2, 130),
(150, 1, 190),
(151, 3, 130),
(152, 1, 190),
(153, 5, 70),
(154, 7, 130),
(155, 2, 130),
(156, 2, 130),
(157, 6, 130),
(158, 4, 70),
(159, 1, 190),
(160, 3, 130),
(161, 6, 130),
(162, 1, 190),
(163, 1, 190),
(164, 2, 130),
(165, 1, 190),
(166, 1, 190),
(167, 1, 190),
(168, 1, 190),
(169, 6, 130),
(170, 1, 190),
(171, 1, 190),
(172, 7, 130),

(173, 7, 130),
(174, 1, 190),
(175, 6, 130),
(176, 1, 190),
(177, 1, 190),
(178, 1, 190),
(179, 1, 190),
(180, 1, 190),
(181, 3, 130),
(182, 6, 130),
(183, 2, 130),
(184, 1, 190),
(185, 6, 130),
(186, 3, 130),
(187, 1, 190),
(188, 3, 130),
(189, 5, 70),
(190, 1, 190),
(191, 6, 130),
(192, 1, 190),
(193, 1, 190),
(194, 2, 130),
(195, 1, 190),
(196, 1, 190),
(197, 7, 130),
(198, 2, 130),
(199, 1, 190),
(200, 1, 190),
(201, 2, 130),
(202, 7, 130),
(203, 1, 190),
(204, 1, 190),
(205, 2, 130),
(206, 4, 70),
(207, 1, 190),
(208, 1, 190),
(209, 3, 130),

(210, 3, 130),

(211, 6, 130),

(212, 3, 130),

(213, 7, 130),

(214, 1, 190),

(215, 3, 130),

(216, 2, 130),

(217, 7, 130),

(218, 4, 70);

II. Anexo 2 - *Script* de criação da Base de Dados em SQL

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS AlojamentoLocal DEFAULT CHARACTER SET utf8 ;
USE AlojamentoLocal ;

-- -----------------------------------------------------
-- Table AlojamentoLocal.`Edificio`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS AlojamentoLocal.`Edificio` (
  `Id_Edificio` INT NOT NULL AUTO_INCREMENT,
  `Rua/Morada` VARCHAR(200) NOT NULL,
  `Localidade/Morada` VARCHAR(100) NOT NULL,
  `Codigo_Postal/Morada` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`Id_Edificio`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table AlojamentoLocal.`Alojamento`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS AlojamentoLocal.`Alojamento` (
  `Id_Alojamento` INT NOT NULL AUTO_INCREMENT,
  `Numero` INT NOT NULL,
```

```
  `Edificio` INT NOT NULL,
  `Preco_Base` DECIMAL(6,2) NOT NULL,
  `Lotacao` INT NOT NULL,
  `Numero_Quartos` INT NOT NULL,
  INDEX `fk_Quartos_Edificio1_idx` (`Edificio` ASC),
  PRIMARY KEY (`Id_Alojamento`),
  CONSTRAINT `fk_Quartos_Edificio1`
    FOREIGN KEY (`Edificio`)
    REFERENCES AlojamentoLocal.`Edificio` (`Id_Edificio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table AlojamentoLocal.`Funcionario`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS AlojamentoLocal.`Funcionario` (
  `Id_Funcionario` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(200) NOT NULL,
  `Telemovel` INT NOT NULL,
  `E-mail` VARCHAR(200) NOT NULL,
  `Salario` INT NOT NULL,
  `Gerente` INT NULL,
  PRIMARY KEY (`Id_Funcionario`),
  INDEX `funcionario_gerente_idx` (`Gerente` ASC),
  CONSTRAINT `funcionario_gerente`
    FOREIGN KEY (`Gerente`)
    REFERENCES AlojamentoLocal.`Funcionario` (`Id_Funcionario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table AlojamentoLocal.`Cliente`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS AlojamentoLocal.`Cliente` (
  `Id_Cliente` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(200) NOT NULL,
  `Data_Nascimento` DATE NOT NULL,
  `E-mail` VARCHAR(200) NOT NULL,
  `Telemovel` INT NOT NULL,
  PRIMARY KEY (`Id_Cliente`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table AlojamentoLocal.`Reserva`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS AlojamentoLocal.`Reserva` (
  `Id_Reserva` INT NOT NULL AUTO_INCREMENT,
  `Funcionario` INT NOT NULL,
  `Cliente` INT NOT NULL,
  `Data_Inicio` DATETIME NOT NULL,
  `Data_Fim` DATETIME NOT NULL,
  `Adultos` INT NOT NULL,
  `Criancas` INT NOT NULL,
  PRIMARY KEY (`Id_Reserva`),
  INDEX `fk_Reservas_Funcionarios1_idx` (`Funcionario` ASC),
  INDEX `fk_Reserva_Cliente1_idx` (`Cliente` ASC),
  CONSTRAINT `fk_Reservas_Funcionarios1`
    FOREIGN KEY (`Funcionario`)
    REFERENCES AlojamentoLocal.`Funcionario` (`Id_Funcionario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Reserva_Cliente1`
    FOREIGN KEY (`Cliente`)
    REFERENCES AlojamentoLocal.`Cliente` (`Id_Cliente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table AlojamentoLocal.`Funcionario_Edificio`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS AlojamentoLocal.`Funcionario_Edificio` (
  `Funcionario` INT NOT NULL,
  `Edificio` INT NOT NULL,
  PRIMARY KEY (`Funcionario`, `Edificio`),
  INDEX `fk_funcionarios_edificio_Edificio1_idx` (`Edificio` ASC),
  CONSTRAINT `fk_funcionarios_edificio_Edificio1`
    FOREIGN KEY (`Edificio`)
    REFERENCES AlojamentoLocal.`Edificio` (`Id_Edificio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_funcionarios_edificio_Funcionarios1`
    FOREIGN KEY (`Funcionario`)
    REFERENCES AlojamentoLocal.`Funcionario` (`Id_Funcionario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table AlojamentoLocal.`Reserva_Alojamento`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS AlojamentoLocal.`Reserva_Alojamento` (
  `Reserva` INT NOT NULL,
  `Alojamento` INT NOT NULL,
  `Preco` DECIMAL(6,2) NOT NULL,
  PRIMARY KEY (`Reserva`, `Alojamento`),
  INDEX `fk_Reserva_Quarto_Quarto1_idx` (`Alojamento` ASC),
  CONSTRAINT `fk_Reserva_Quarto_Reserva1`
    FOREIGN KEY (`Reserva`)
    REFERENCES AlojamentoLocal.`Reserva` (`Id_Reserva`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Reserva_Quarto_Quarto1`
    FOREIGN KEY (`Alojamento`)
```

```
  REFERENCES AlojamentoLocal.`Alojamento` (`Id_Alojamento`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

III. Anexo 3 - *Script* de Implementação das *Queries* em *SQL*

```
-- RE01 - Listar os clientes que frequentaram o estabelecimento;
SELECT Id_Cliente, Nome, `E-mail`, Telemovel FROM Cliente
        WHERE Id_Cliente IN
        (SELECT Cliente FROM Reserva);

-- RE02 – Listar os edifícios e respetivos alojamentos
SELECT E.Id_Edificio, E.`Rua/Morada`, A.Numero AS Numero_Alojamento, A.Numero_Quartos, A.Lotacao
FROM Edificio AS E
        INNER JOIN Alojamento AS A
                ON A.Edificio = E.Id_Edificio
  ORDER BY Id_Edificio;

-- RE03 - Listar todas as reservas realizadas num certo alojamento
SELECT RA.Reserva, R.Data_Inicio, R.Data_Fim FROM Reserva_Alojamento AS RA
  INNER JOIN Reserva AS R
                ON R.Id_Reserva = RA.Reserva
        WHERE Alojamento = 1;

-- RE04 – Listar todos os funcionários e respetivo responsável
SELECT Id_Funcionario, Nome, Gerente FROM Funcionario;

-- RE05 – Calcular o número de vezes que cada alojamento foi reservado
SELECT P.Id_Alojamento, P.Numero, P.Edificio, COUNT(P.Reserva) AS Total_Reservas FROM
        (SELECT A.Id_Alojamento, A.Numero, A.Edificio, RA.Reserva FROM Reserva_Alojamento AS RA
                INNER JOIN Alojamento AS A
                        ON A.Id_Alojamento = RA.Alojamento) AS P
        GROUP BY Id_Alojamento;

-- RE06 – Calcular o número de pedidos que um funcionário processou
SELECT R.Funcionario, F.Nome, COUNT(R.Id_Reserva) AS Total_Reservas FROM Reserva AS R
        INNER JOIN Funcionario AS F
                ON F.Id_Funcionario = R.Funcionario
        GROUP BY Funcionario;
```

```
-- RE07 – Listar os quartos que estão disponíveis até um certo preço
SELECT Id_Alojamento, Preco_Base FROM Alojamento WHERE Preco_Base <= 100;


-- RE08 - Listar os clientes por ordem descrescente de dinheiro gasto
SELECT R.Cliente, C.Nome, SUM(RA.Preco) AS Total_Gasto FROM Reserva AS R
   INNER JOIN Cliente AS C
      ON C.Id_Cliente = R.Cliente
   INNER JOIN Reserva_Alojamento AS RA
      ON RA.Reserva = R.Id_Reserva
   GROUP BY R.Cliente
   ORDER BY Total_Gasto DESC;

-- RE09 -  Calcular o valor total dos alugueres de cada quarto
select Alojamento, sum(Preco) from Reserva_Alojamento
   group by Alojamento;
```

IV. Anexo 4 - *Script* da migração para *MongoDB*

```
import decimal
import datetime
import mysql.connector
import pymongo

mongoClient = pymongo.MongoClient("mongodb://localhost:27017/")
mongoDb = mongoClient["AlojamentoLocal"]

db = mysql.connector.connect(
   host = "localhost",
   user = "universidade",
   password  = "universidade",
   database = "AlojamentoLocal"
)

def migracao(tabela, atributos):
   cursor = db.cursor()
   cursor.execute(f"select * from {tabela}")
   res = []
   for row in cursor:
      print(row)
      obj = []
      for i in range(len(row)):
         if isinstance(row[i], decimal.Decimal):
            obj.append(float(row[i]))
         elif isinstance(row[i], datetime.date):
            obj.append(datetime.datetime(row[i].year, row[i].month, row[i].day))
         else:
            obj.append(row[i])
      res.append(tuple(obj))

   lista = []
   for el in res:
      lista.append({})
```

```
        for i in range(len(el)):
            lista[-1][atributos[i]] = el[i]
    return lista

###################################
# Edificio
###################################

atributos = ["_id", "Rua/Morada", "Localidade/Morada", "Codigo_Postal/Morada"]

edificio = migracao("Edificio", atributos)
mongoColection = mongoDb["Edificio"]
mongoColection.drop()
x = mongoColection.insert_many(edificio)
print("Coleção Edificio criada.")

###################################
# Alojamento
###################################

atributos = ["_id", "Numero", "Edificio", "Preco_Base", "Lotacao", "Numero_Quartos"]

alojamento = migracao("Alojamento", atributos)

###################################
# Funcionario
###################################

atributos = ["_id", "Nome", "Telemovel", "E-mail", "Salario", "Gerente"]

funcionario = migracao("Funcionario", atributos)
mongoColection = mongoDb["Funcionario"]
mongoColection.drop()
x = mongoColection.insert_many(funcionario)
print("Coleção Funcionario criada.")

###################################
```

```
# Cliente
###################################

atributos = ["_id", "Nome", "Data_Nascimento", "E-mail", "Telemovel"]

cliente = migracao("Cliente", atributos)
mongoColection = mongoDb["Cliente"]
mongoColection.drop()
x = mongoColection.insert_many(cliente)
print("Coleção Cliente criada.")

###################################
# Reserva
###################################

atributos = ["_id", "Funcionario", "Cliente", "Data_Inicio", "Data_Fim", "Adultos", "Criancas"]

reservas = migracao("Reserva", atributos)

###################################
# Reserva_Alojamento
###################################

atributos = ["Reserva", "Alojamento", "Preco"]

reserva_alojamento = migracao("Reserva_Alojamento", atributos)

###################################
# Adicionar relação Reserva/Alojamento
###################################

for x in reserva_alojamento:
  id_reserva = x["Reserva"]
  id_alojamento = x["Alojamento"]
  preco = x["Preco"]

  for reserva in filter(lambda e: e["_id"] == id_reserva, reservas):
```

```
        if "Alojamentos" not in reserva:
            reserva["Alojamentos"] = []
        reserva["Alojamentos"].append({"Alojamento": id_alojamento, "Preco": preco})

    for aloj in filter(lambda e: e["_id"] == id_alojamento, alojamento):
        if "Reservas" not in aloj:
            aloj["Reservas"] = []
        aloj["Reservas"].append({"Reserva": id_reserva, "Preco": preco})

mongoColection = mongoDb["Alojamento"]
mongoColection.drop()
x = mongoColection.insert_many(alojamento)
print("Coleção Alojamento criada.")

mongoColection = mongoDb["Reserva"]
mongoColection.drop()
x = mongoColection.insert_many(reservas)
print("Coleção Reserva criada.")
```

V. Anexo 4 - *Script* das *Queries* em *MongoDB*

```
// RE01 - Listar os clientes que frequentaram o estabelecimento

db.Cliente.aggregate([
  {
    $lookup: {
      from: "Reserva",
      localField: "_id",
      foreignField: "Cliente",
      as: "Reservas",
    },
  },
]);

// RE02 – Listar os edifícios e respetivos alojamentos

db.Edificio.aggregate([
  {
    $lookup: {
      from: "Alojamento",
      localField: "_id",
      foreignField: "Edificio",
      as: "Alojamentos",
    },
  },
  {
    $project: {
      "Rua/Morada": 1,
      "Alojamentos.Numero": 1,
      "Alojamentos.Numero_Quartos": 1,
      "Alojamentos.Lotacao": 1,
    },
  },
]).sort({ _id: 1 });
```

```
// RE03 - Listar todas as reservas realizadas num certo alojamento

db.Alojamento.aggregate([
  { $match: {} },
  { $unwind: "$Reservas" },
  {
    $lookup: {
      from: "Reserva",
      localField: "Reservas.Reserva",
      foreignField: "_id",
      as: "Reserva",
    },
  },
  { $unwind: "$Reserva" },
  {
    $group: {
      _id: "$_id",
      Reservas: {
        $push: {
          _id: "$Reserva._id",
          Data_Inicio: "$Reserva.Data_Inicio",
          Data_Fim: "$Reserva.Data_Fim",
        },
      },
    },
  },
]);

// RE04 – Listar todos os funcionários e respetivo responsável

db.Funcionario.find({}, { Nome: 1, Gerente: 1 });

// RE05 – Calcular o número de vezes que cada alojamento foi reservado

db.Alojamento.aggregate([
  { $match: {} },
  { $unwind: "$Reservas" },
```

```
  {
    $lookup: {
      from: "Reserva",
      localField: "Reservas.Reserva",
      foreignField: "_id",
      as: "Reserva",
    },
  },
  { $unwind: "$Reserva" },
  {
    $group: {
      _id: "$_id",
      Reservas: { $sum: 1 },
    },
  },
]);

// RE06 – Calcular o número de pedidos que um funcionário processou

db.Funcionario.aggregate([
  {
    $lookup: {
      from: "Reserva",
      localField: "_id",
      foreignField: "Funcionario",
      as: "Reservas",
    },
  },
  { $unwind: "$Reservas" },
  {
    $group: {
      _id: "$_id",
      Nome: { $first: "$Nome" },
      Reservas: { $sum: 1 },
    },
  },
]).sort({ _id: 1 });
```

```
// RE07 – Listar quantos quartos estão disponíveis até um certo preço

db.Alojamento.find({ Preco_Base: { $lte: 100 } });

// RE08 - Listar os clientes por ordem descrescente de dinheiro gasto

db.Reserva.aggregate([
  {
    $lookup: {
      from: "Cliente",
      localField: "Cliente",
      foreignField: "_id",
      as: "ClienteCol",
    },
  },
  { $unwind: "$ClienteCol" },
  { $unwind: "$Alojamentos" },
  {
    $group: {
      _id: "$Cliente",
      Nome: { $first: "$ClienteCol.Nome" },
      Total_Gasto: { $sum: "$Alojamentos.Preco" },
    },
  },
]).sort({ Total_Gasto: -1, _id: 1 });
```