

**Unidade Curricular de Base de Dados**

Ano Letivo de 2020/2021

junho, 2021

**Jaques Resende 8190214**

**Rui Soares 8150289**

**Luís Oliveira 8190370**

**SGBD CliPet - Clínicas veterinárias**

|  |  |
| --- | --- |
| Data de Receção |  |
| Responsável |  |
| Avaliação |  |
| Observações |  |

**Jaques Resende 8190214**

**Rui Soares 8150289**

**Luís Oliveira 8190370**

junho, 2021

# Agradecimentos

Ao professor Bruno, pela disponibilidade mostrada para o esclarecimento das dúvidas dos autores. A Patrícia Santos, médica veterinária do centro veterinário de Centro Veterinário Vila Meã – Aparecida, por esclarecer os autores sobre o domínio deste projeto.

# Resumo

No seguimento do início do novo projeto da gerência da “CliPet” para o desenvolvimento de um sistema informático que suprima as necessidades da empresa, surge a necessidade de implementação de uma base de dados para suportar o funcionamento da aplicação que registe informação sobre a gestão de consultas e exames.

Este relatório descreve as várias fases da criação e implementação dessa base de dados, tais como o planeamento, a descrição dos objetivos e limites da aplicação, identificação de requisitos, desenho conceptual, desenho lógico para o modelo relacional e a tradução desse modelo lógico global para o SGBD.

Este projeto é relativo à componente prática da Unidade Curricular de Base de Dados.

# Índice

[Agradecimentos iii](#_Toc75878889)

[Resumo iv](#_Toc75878890)

[Índice v](#_Toc75878891)

[Índice de Figuras vii](#_Toc75878892)

[Índice de Tabelas viii](#_Toc75878893)

[Índice Excertos de SQL ix](#_Toc75878894)

[1. Introdução 1](#_Toc75878895)

[1.1 Contextualização 1](#_Toc75878896)

[1.2 Apresentação do Caso de Estudo 1](#_Toc75878897)

[1.3 Motivação e Objetivos 1](#_Toc75878898)

[1.4 Estrutura do Relatório 1](#_Toc75878899)

[2 Requisitos Gerais e Use Cases 3](#_Toc75878900)

[2.1 Requisitos Gerais 3](#_Toc75878901)

[3 Desenho Conceptual 5](#_Toc75878902)

[3.1 Identificação das entidades 6](#_Toc75878903)

[3.2 Relações entre Entidades 7](#_Toc75878904)

[3.3 Multiplicidade 8](#_Toc75878905)

[3.2 Atributos para as Entidades 9](#_Toc75878906)

[3.5 Documentação de atributos 10](#_Toc75878907)

[3.6. Atribuição de chaves primarias 14](#_Toc75878908)

[4 Desenho Lógico 16](#_Toc75878909)

[4.1 Derivação de relações para o modelo de dados lógico 17](#_Toc75878910)

[4.1.1 Entidades fortes 17](#_Toc75878911)

[4.1.2 Entidades fracas 17](#_Toc75878912)

[4.1.3 Relação de um para muitos (1 : \*) 18](#_Toc75878913)

[4.1.4 Relação de um para um (1 : 1) 20](#_Toc75878914)

[4.1.5 Relação de muitos para muitos (\* : \*) 21](#_Toc75878915)

[4.1.6 Multi-valued attributes 22](#_Toc75878916)

[4.2 Normalização 23](#_Toc75878917)

[4.2.1 Primeira Forma Normal (1FN) 23](#_Toc75878918)

[4.2.2 Segunda Forma Normal (2FN) 23](#_Toc75878919)

[4.2.3 Terceira Forma Normal (3FN) 23](#_Toc75878920)

[4.2.4 Mockups 24](#_Toc75878921)

[5 Restrições de integridade – Regras de negócio 33](#_Toc75878922)

[6 Desenho Físico 36](#_Toc75878923)

[6.1 Criação da Tabelas, relacionamentos e restrições aos atributos 38](#_Toc75878924)

[6.1.1 Criação da Tabela Animal 38](#_Toc75878925)

[6.1.2 Criação da Tabela AnimalCliente 40](#_Toc75878926)

[6.1.3 Criação da Tabela Cliente 41](#_Toc75878927)

[6.1.4 Criação da Tabela Clínica 42](#_Toc75878928)

[6.1.5 Criação da Tabela DetalhesMarcacao 43](#_Toc75878929)

[6.1.6 Criação da Tabela Especie 44](#_Toc75878930)

[6.1.7 Criação da Tabela Fatura 45](#_Toc75878931)

[6.1.8 Criação da Tabela Funcionario 46](#_Toc75878932)

[6.1.9 Criação da Tabela Marcação 47](#_Toc75878933)

[6.1.10. Criação da Tabela ModoPagamento 49](#_Toc75878934)

[6.1.11. Criação da Tabela Raca 49](#_Toc75878935)

[6.1.12. Criação da Tabela Servico 50](#_Toc75878936)

[6.1.13. Criação da Tabela Telefone 50](#_Toc75878937)

[6.1.14. Criação da Tabela TipoFuncionario 51](#_Toc75878938)

[6.2 T-SQL 52](#_Toc75878939)

[6.2.1 Triggers 52](#_Toc75878940)

[6.2.2 Stored Procedures 54](#_Toc75878941)

[6.2.3 Scalar-Valued Functions 58](#_Toc75878942)

[6.3 Views 60](#_Toc75878943)

[7. Conclusões e Trabalho Futuro 64](#_Toc75878944)

[Bibliografia 65](#_Toc75878945)

[Referências WWW 66](#_Toc75878946)

[Lista de Siglas e Acrónimos 67](#_Toc75878947)

[Anexos 68](#_Toc75878948)

[ANEXO I 69](#_Toc75878949)

# Índice de Figuras

[Figura 1 - Diagrama E-R sem atributos 7](https://ipppt.sharepoint.com/sites/2021BDLEI/Documentos%20Partilhados/Relatorio_BD_Grupo10.docx#_Toc75894508)

[Figura 2 - Diagrama concetual só com chaves primárias 14](#_Toc75894509)

[Figura 3 - Diagrama Conceptual ER 15](https://ipppt.sharepoint.com/sites/2021BDLEI/Documentos%20Partilhados/Relatorio_BD_Grupo10.docx#_Toc75894510)

[Figura 4 - Mockup Marcação 24](https://ipppt.sharepoint.com/sites/2021BDLEI/Documentos%20Partilhados/Relatorio_BD_Grupo10.docx#_Toc75894511)

[Figura 5 - Mockup Cliente 26](https://ipppt.sharepoint.com/sites/2021BDLEI/Documentos%20Partilhados/Relatorio_BD_Grupo10.docx#_Toc75894512)

[Figura 6 - Mockup Histórico Animal 27](https://ipppt.sharepoint.com/sites/2021BDLEI/Documentos%20Partilhados/Relatorio_BD_Grupo10.docx#_Toc75894513)

[Figura 7 - Mockup Detalhes de Marcação 29](https://ipppt.sharepoint.com/sites/2021BDLEI/Documentos%20Partilhados/Relatorio_BD_Grupo10.docx#_Toc75894514)

[Figura 8 - Mockup Registo Funcionário 31](https://ipppt.sharepoint.com/sites/2021BDLEI/Documentos%20Partilhados/Relatorio_BD_Grupo10.docx#_Toc75894515)

[Figura 9 Diagrama ER do SQL Management Studio 37](https://ipppt.sharepoint.com/sites/2021BDLEI/Documentos%20Partilhados/Relatorio_BD_Grupo10.docx#_Toc75894516)

# Índice de Tabelas

[Tabela 1 - Identificação das entidades 6](#_Toc75895452)

[Tabela 2 - Multiplicidade 8](#_Toc75895453)

[Tabela 3 - Atributos para as Entidades 9](#_Toc75895454)

[Tabela 4 - **Relação** - Funcionario 10](#_Toc75895455)

[Tabela 5 - **Relação** - Clinica 10](#_Toc75895456)

[Tabela 6 - **Relação** - Servico 11](#_Toc75895457)

[Tabela 7 - **Relação** - Animal 11](#_Toc75895458)

[Tabela 8 - **Relação** - Marcacao 12](#_Toc75895459)

[Tabela 9 - **Relação** - Cliente 12](#_Toc75895460)

[Tabela 10 - **Relação** - Fatura 13](#_Toc75895461)

[Tabela 11 - **Relação** - DetalhesMarcacao 13](#_Toc75895462)

[Tabela 12 - Integridade Referencial 35](#_Toc75895463)

[Tabela 13 - Tipos de dados 36](#_Toc75895464)

# Índice Excertos de SQL

[Excerto SQL 1 - Criaçao Tabela Animal 39](#_Toc75895466)

[Excerto SQL 2 - Criação da Tabela AnimalCliente 40](#_Toc75895467)

[Excerto SQL 3 - Criação da Tabela Cliente 41](#_Toc75895468)

[Excerto SQL 4 - Criação da Tabela Clínica 42](#_Toc75895469)

[Excerto SQL 5 - Criação da Tabela DetalhesMarcacao 43](#_Toc75895470)

[Excerto SQL 6 - Criação da Tabela Especie 44](#_Toc75895471)

[Excerto SQL 7 - Criação da Tabela Fatura 45](#_Toc75895472)

[Excerto SQL 8 - Criação da Tabela Funcionario 46](#_Toc75895473)

[Excerto SQL 9 - Criação da Tabela Marcação 48](#_Toc75895474)

[Excerto SQL 10 - Criação da Tabela ModoPagamento 49](#_Toc75895475)

[Excerto SQL 11 - Criação da Tabela Raca 49](#_Toc75895476)

[Excerto SQL 12 - Criação da Tabela Servico 50](#_Toc75895477)

[Excerto SQL 13 - Criação da Tabela Telefone 50](#_Toc75895478)

[Excerto SQL 14 - Criação da Tabela TipoFuncionario 51](#_Toc75895479)

[Excerto SQL 15 - Criação do Trigger DonoAnimal 53](#_Toc75895480)

[Excerto SQL 16 - Criação do Trigger updateAnimal 53](#_Toc75895481)

[Excerto SQL 17 - Criação do procedimento HistoricoAnimal 55](#_Toc75895482)

[Excerto SQL 18 - Criação do procedimento InsereMarcacao 57](#_Toc75895483)

[Excerto SQL 19 - Criação do Procedimento InserirFatura 57](#_Toc75895484)

[Excerto SQL 20 - Criação da função TotalFaturadoMesAno 58](#_Toc75895485)

[Excerto SQL 21 - Criação da função CalcHoraFim 59](#_Toc75895486)

[Excerto SQL 22 - Consultar clientes com mais de 20% de consultas remarcadas/canceladas 60](#_Toc75895487)

[Excerto SQL 23 - Consultar dono atual do animal 61](#_Toc75895488)

[Excerto SQL 24 - Veterinário com mais consultas 61](#_Toc75895489)

[Excerto SQL 25 - Consulta da percentagem de exames realizados que têm como origem uma consulta realizada 62](#_Toc75895490)

[Excerto SQL 26 - Consultar o tipo de animal e respetiva raça, que gera maior volume de faturação 62](#_Toc75895491)

[Excerto SQL 27 - Consultar total de marcações por cliente 63](#_Toc75895492)

[Excerto SQL 28 - Consultar valor faturado em um mês de um determinado ano 63](#_Toc75895493)

# 1. Introdução

### 1.1 Contextualização

Este projeto é relativo à componente prática da Unidade Curricular de Base de Dados e tem como objetivo principal o desenvolvimento de uma base de dados com base nos conhecimentos adquiridos ao longo do semestre nesta UC.

### 1.2 Apresentação do Caso de Estudo

O grupo de clínicas veterinárias “CliPet” procuram o desenvolvimento de um sistema informático que permita aos seus clientes a marcação de consultas em qualquer uma das duas suas clínicas, podendo a qualquer altura cancelar/ remarcar o serviço. Que também permita a consulta do histórico clínico do animal, resumos de consulta, registo do motivo da consulta, a descrição da receita e outras informações por parte do médico veterinário. Neste grupo, apenas algumas clínicas realizam exames.

### 1.3 Motivação e Objetivos

O objetivo principal deste projeto é a implementação de uma base de dados para suportar o funcionamento da aplicação referida anteriormente, aplicando os conhecimentos adquiridos nesta UC. Esta base de dados deverá permitir as seguintes consultas/queries:

* Ver o valor faturado num determinado ano.
* Ver a percentagem de exames que tem como origem uma consulta.
* Ver quais são os veterinários com mais consultas
* Ver que animal gera mais faturação, por tipo de animal e raça.
* Ver o valor faturado em um mês do ano.
* Ver quais os clientes com mais de 20% de consultas remarcadas/canceladas.
* Consultar histórico clínico (detalhes de marcações e serviços feitos nas marcações).

### 1.4 Estrutura do Relatório

De forma a facilitar a sua consulta, a estrutura do relatório é dividida em capítulos e subcapítulos, onde se descreve os passos para a realização deste projeto.

Os autores apresentam o tema, a contextualização e o caso de estudo deste projeto na introdução. De seguida, descrevem o processo de desenvolvimento deste projeto e por fim, uma reflexão sobre o projeto.

## 2 Requisitos Gerais e Use Cases

### 2.1 Requisitos Gerais

Para a identificação dos requisitos, além da informação disponibilizada no enunciado, os autores entrevistaram médicos veterinários, para uma melhor compreensão e contextualização deste trabalho.

Foram identificados os seguintes requisitos:

1. O veterinário pode visualizar o histórico do animal;
2. Os animais ficam registados em todas as clínicas do mesmo grupo
3. Na marcação o cliente seleciona o serviço que pretende e depois do estado da marcação da
4. Alguns serviços podem ter outros dependentes, isto resultará num relacionamento recursivo;
5. Todas as clínicas do mesmo grupo têm o mesmo horário de funcionamento;
6. Uma consulta de urgência é uma consulta fora do horário de funcionamento da clínica;
7. Uma consulta de urgência tem um custo mais elevado;
8. Todas os serviços de clínicas do mesmo grupo têm os mesmos preços;
9. A clínica pode ter mais do que um número de telefone;
10. O valor total da fatura inclui o preço do serviço prestado numa marcação;
11. Qualquer consulta ou exame tem de ter obrigatoriamente uma marcação prévia, se for uma emergência a marcação será feita na hora;
12. As marcações poderão ter quatro estados Cancelada, Remarcada, Confirmada e Terminada;
13. O grupo de clínicas não suporta serviços com duração maior que 24h, como internamentos noturnos. No caso de internamentos noturnos o paciente será transferido para outra clínica/hospital fora do grupo;
14. Apos o horário de fecho da clínica, o atendimento presencial será por marcação prévia por telefone com atendimento 24h;
15. Certas marcações podem precisar de um funcionário externo ao grupo de clínicas para realizar um certo serviço;
16. Uma clínica recentemente inaugurada não tem faturas nem marcações associadas;
17. Um animal só pode ter um dono ao mesmo tempo e vários ao longo do tempo
18. A primeira consulta de um animal é um serviço e é sempre mais cara que as restantes
19. Uma clínica que não faça exames não permite o cliente marcar exames
20. Uma fatura só pode ser emitida depois do estado da consulta estiver terminada
21. Um funcionário só pode atender um animal por marcação
22. Uma fatura é emitida por um funcionário da clínica, após o serviço prestado
23. As clínicas da CliPet são um grupo a nível nacional
24. O animal no registo é pesado e é medida a temperatura, frequência cardíaca e altura.

## 3 Desenho Conceptual

O primeiro passo para o desenho de uma base de dados é definir o modelo conceptual de dados. Este trabalho realizar-se-á ao longo deste capítulo nas fases seguintes:

1. Identificação de tipo de entidades

2. Relações entre entidades:

3. Atributos para as entidades

4. Documentação de atributos

5. Atribuição de chaves primarias

6. Documentação de chaves primarias (PK)

7. Utilização de conceitos de modelação avançada

8. Validação do modelo conceptual de dados de forma a satisfazer as transações dos utilizadores

9. Revisão do modelo conceptual de dados com o utilizador

### 3.1 Identificação das entidades

Tabela 1 - Identificação das entidades

|  |  |  |
| --- | --- | --- |
| **Entidade** | **Descrição** | **Ocorrência** |
| Clínica | Informação geral das clínicas disponíveis | Cada clínica tem consultas, clientes, funcionários, animais e agenda |
| Servico | Informação sobre os serviços | Um serviço vai ser associado a uma marcação |
| Cliente | Informação geral sobre o cliente | Cada cliente tem um animal de estimação |
| Animal | Informação geral sobre o animal | Cada animal tem um dono, que por sua vez se torna um cliente da clínica |
| Funcionario | Informação geral sobre o Funcionário | O Funcionário está associado a consultas e exames |
| Marcacao | Informação de um serviço marcado previamente | O cliente marca um serviço |
| Fatura | Informação sobre um serviço realizado, a data de emissão da fatura e o modo de pagamento | Quando o serviço acontece pode ser feito uma fatura do mesmo |
| DetalhesMarcacao | Informação sobre o serviço que aconteceu | Quando acontece um serviço o veterinário pode ou não guardar informações do mesmo nos detalhes do serviço |

### 3.2 Relações entre Entidades

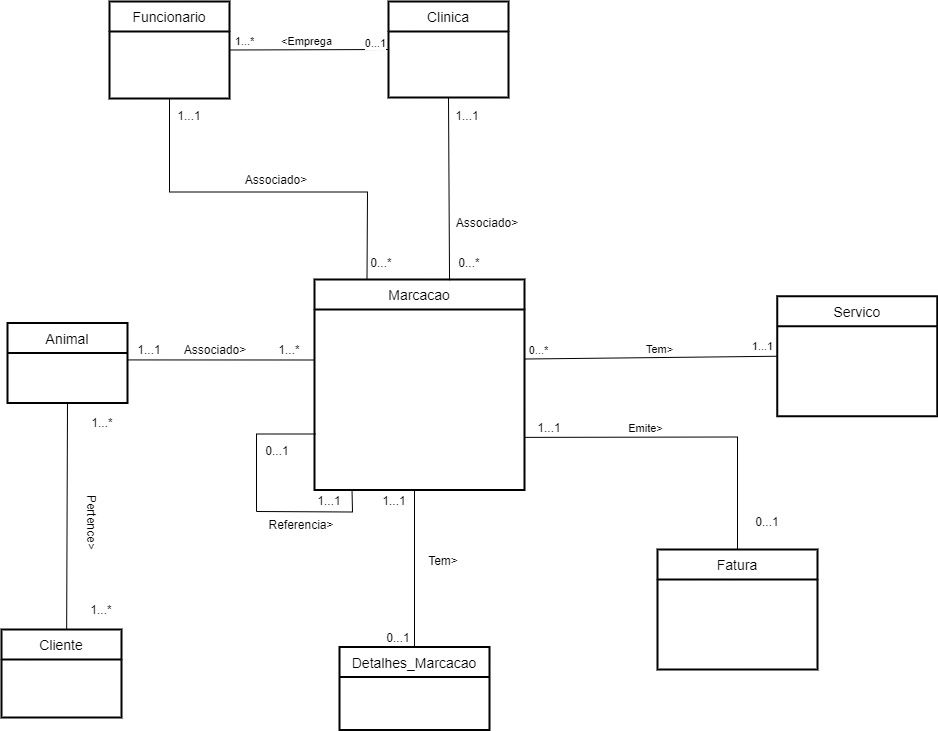


Figura 1 - Diagrama E-R sem atributos

### 3.3 Multiplicidade

Tabela 2 - Multiplicidade

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entidade | Multiplicidade | Relação | Multiplicidade | Entidade |
| Clinica | 0...1 | Emprega | 1…\* | **Funcionario** |
| Funcionario | 1...1 | Associado | 0...\* | **Marcacao** |
| Animal | 1...1 | Associado | 1...\* | **Marcacao** |
| Animal | 1...\* | Pertence | 1...\* | **Cliente** |
| Marcacao | 1...0 | Pertence | 1...1 | **Servico** |
| Marcacao | 1...1 | Tem | 0...1 | **DetalhesMarcacao** |
| Fatura | 0...\* | Refere | 1...1 | **Marcacao** |
| Marcacao | 0…\* | Refere | 1...1 | **Marcacao** |
| Clinica | 1...1 | Associado | 0...\* | **Marcacao** |

### 3.2 Atributos para as Entidades

Tabela 3 - Atributos para as Entidades

|  |  |
| --- | --- |
| Funcionario | funcionarioID, primeiroNome, ultimoNome, sexo, dataDeNascimento, dataDeModificacao, tipoFuncionario |
| Clinica | clinicaID, tel, email, rua, cidade, codPostal, pais, realizaExame |
| Animal | animalID, nome, dataDeNascimento, especie, comportamento, raca, microchip, peso, altura, sexo, estado |
| Marcacao | marcacaoID, data, horaInicio, horaFim, estado, descricao, marcacaoRef, urgencia |
| Cliente | clienteID, primeiroNome, ultimoNome, cc, nif, email, rua, cidade, codPostal, pais |
| Fatura | faturaID, modoPagamento, dataEmissao, horaEmissao |
| DetalhesMarcacao | detalhesID, motivo, tratamento, obs, peso, temperatura, freqCardiaca |
| Servico | ServicoID, descricao, tipoServico, precoBase, duração |

## 3.5 Documentação de atributos

Tabela 4 - **Relação** - Funcionario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Atributo | Descrição | Tipo de informação | Nulls | Multi-Valued |
| funcionarioID | Identificador único de cada funcionário | Até 3 números inteiros | Não | Não |
| primeiroNome | Sobrenome do funcionário | 15 carateres variáveis | Não | Não |
| ultimoNome | Apelido do funcionário | 15 carateres variáveis | Não | Não |
| sexo | Género do funcionário | 1 caracter | Não | Não |
| dataDeNascimento | Data de nascimento do funcionário | Tipo data | Não | Não |
| tipoFuncionario | Identificação da função do funcionário | 25 carateres variáveis | Não | Não |
| dataDeModificacao | Data e Hora de modificação da informação de funcionário | Data e hora | Não | Não |

Tabela 5 - **Relação** - Clinica

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Descrição** | **Tipo de informação** | **Nulls** | **Multi-Valued** |
| **clinicaID** | Identificador único da clínica | Até 4 números inteiros | Não | Não |
| **telNo** | Número de telemóvel da clínica | 9 dígitos numéricos | Não | Sim |
| **email** | Email da clínica | 256 carateres variáveis | Não | Não |
| **rua** | Rua da Clínica | 50 carateres variáveis | Não | Não |
| **cidade** | Cidade onde a clínica está localizada | 50 carateres variáveis | Não | Não |
| **codPostal** | Código de postal da clínica | XXXX-XXX | Não | Não |
| **pais** | País onde se encontra a clínica | 2 carateres | Não | Não |
| **realizaExame** | Confirma se a clínica faz ou não exames | Valor booleano | Não | Não |

Tabela 6 - **Relação** - Servico

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Descrição** | **Tipo de informação** | **Nulls** | **Multi-Valued** |
| **servicoID** | Identificação única de cada serviço | Até 3 números inteiros | Não | Não |
| **tipoServico** | Identificação do tipo do serviço | 30 carateres variáveis | Não | Não |
| **precoBase** | Preço do serviço | Smallmoney | Não | Não |
| **descricao** | Descrição do serviço | 100 carateres variáveis | Não | Não |
| **duracao** | Duração prevista para o serviço | 3 números inteiros | Não | Não |

Tabela 7 - **Relação** - Animal

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Descrição** | **Tipo de informação** | **Nulls** | **Multi-Valued** |
| **animalID** | Identificação única de cada serviço | Até 6 números inteiros | Não | Não |
| **nome** | Nome de animal | 15 carateres variáveis | Não | Não |
| **dataNascimento** | Data de nascimento do animal | Tipo data | Não | Não |
| **especie** | Identificação do tipo do serviço | 15 carateres variáveis | Não | Não |
| **comportamento** | Registo do tipo de comportamento do animal | 15 carateres variáveis | Não | Não |
| **raca** | Designação da raça da raça do animal | 50 carateres variáveis | Não | Não |
| **microchip** | Número de Identificação do chip do animal | 15 dígitos numéricos | Sim | Não |
| **peso** | Peso do animal | decimal (5,2) | Não | Não |
| **altura** | Altura do animal | decimal (5,2) | Não | Não |
| **sexo** | Género do animal | 1 caracter | Não | Não |
| **estado** | Identifica se registo está ativo (true) ou não (false) | Valor booleano | Não | Não |

Tabela 8 - **Relação** - Marcacao

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Descrição** | **Tipo de informação** | **Nulls** | **Multi-Valued** |
| **marcacaoID** | Identificação única do serviço | Até 9 números inteiros | Não | Não |
| **data** | Data da marcação | Data | Não | Não |
| **horaInicio** | Hora do início da marcação | Time | Não | Não |
| **horaFim** | Hora do fim da marcação | Time | Sim | Não |
| **estado** | Estado da marcação (cancelada, confirmada etc) | 15 Carateres variáveis | Não | Não |
| **descricao** | Descrição da marcação | 100 carateres variáveis | Não | Não |
| **marcacaoRef** | Referencia de marcação anterior | Unsigned int | Sim | Não |
| **urgencia** | A marcação é urgência ou não | Valor booleano | Não | Não |

Tabela 9 - **Relação** - Cliente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Descrição** | **Tipo de informação** | **Nulls** | **Multi-Valued** |
| **clienteID** | Número de identificação do cliente | Até 9 números inteiros | Não | Não |
| **primeiroNome** | Primeiro nome de cliente | 15 carateres variáveis | Não | Não |
| **ultimoNome** | Último nome de cliente | 15 carateres variáveis | Não | Não |
| **cc** | número de cartão de cidadão | 8 dígitos numéricos |  |  |
| **nif** | Número de identificação fiscal | 9 dígitos numéricos | Não | Não |
| **email** | Email do cliente | 256 carateres variáveis | Não | Não |
| **rua** | Rua do cliente | 50 carateres variáveis | Não | Não |
| **Cidade** | Cidade onde cliente habita | 50 carateres variáveis | Não | Não |
| **codPostal** | Código de postal do cliente | XXXX-XXX | Não | Não |
| **Pais** | Pais onde o cliente reside | 2 carateres | Não | Não |

Tabela 10 - **Relação** - Fatura

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Descrição** | **Tipo de informação** | **Nulls** | **Multi-Valued** |
| **faturaID** | Nº do documento | 5 carateres | Não | Não |
| **modoPagamento** | Descrição do modo de pagamento | 25 carateres variáveis | Não | Não |
| **dataEmissao** | Data de emissão do documento | Date | Não | Não |
| **horaEmissao** | Tempo de emissão do documento | Time | Não | Não |
| **valor** | Valor total do documento | Smallmoney | Não | Não |

Tabela 11 - **Relação** - DetalhesMarcacao

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Descrição** | **Tipo de informação** | **Nulls** | **Multi-Valued** |
| **detalhesID** | Identificação dos detalhes | Unsigned int | não | Não |
| **motivo** | Motivo da consulta | 50 carateres variáveis | não | Não |
| **tratamento** | Receita ou prescrição da consulta | 100 carateres variáveis | não | Não |
| **obs** | Observações da marcação registada pelo veterinário | 100 carateres variáveis | Sim | Não |
| **peso** | Resultado da pesagem do animal | Número de 5 dígitos com 2 casas decimais | sim | Não |
| **temperatura** | Temperatura do animal na marcação | 3 dígitos numéricos | sim | Não |
| **freqCardiaca** | Frequência cardíacos do animal | 3 dígitos numéricos | sim | Não |

### 3.6. Atribuição de chaves primarias

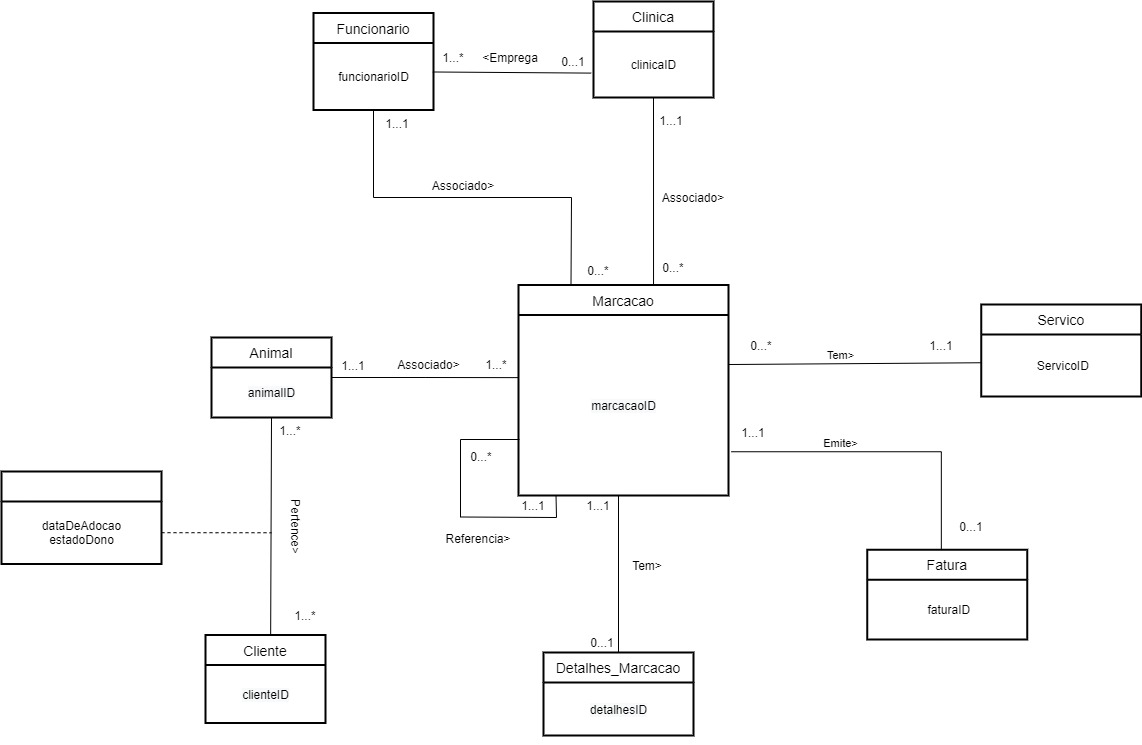


Figura 2 - Diagrama concetual só com chaves primárias

**Documentação de chaves primarias (PK)**

|  |  |  |
| --- | --- | --- |
| Entidade | Chave Primaria | Chaves Candidatas |
| Funcionario | funcionarioID | \_ |
| Clinica | clinicaID | Email |
| Servico | servicoID | \_ |
| Animal | animalID | \_ |
| Marcacao | marcacaoID | \_ |
| Cliente | clienteID | Nif | cc |
| Fatura | faturaID | \_ |
| DetalhesMarcacao | detalhesID | \_ |

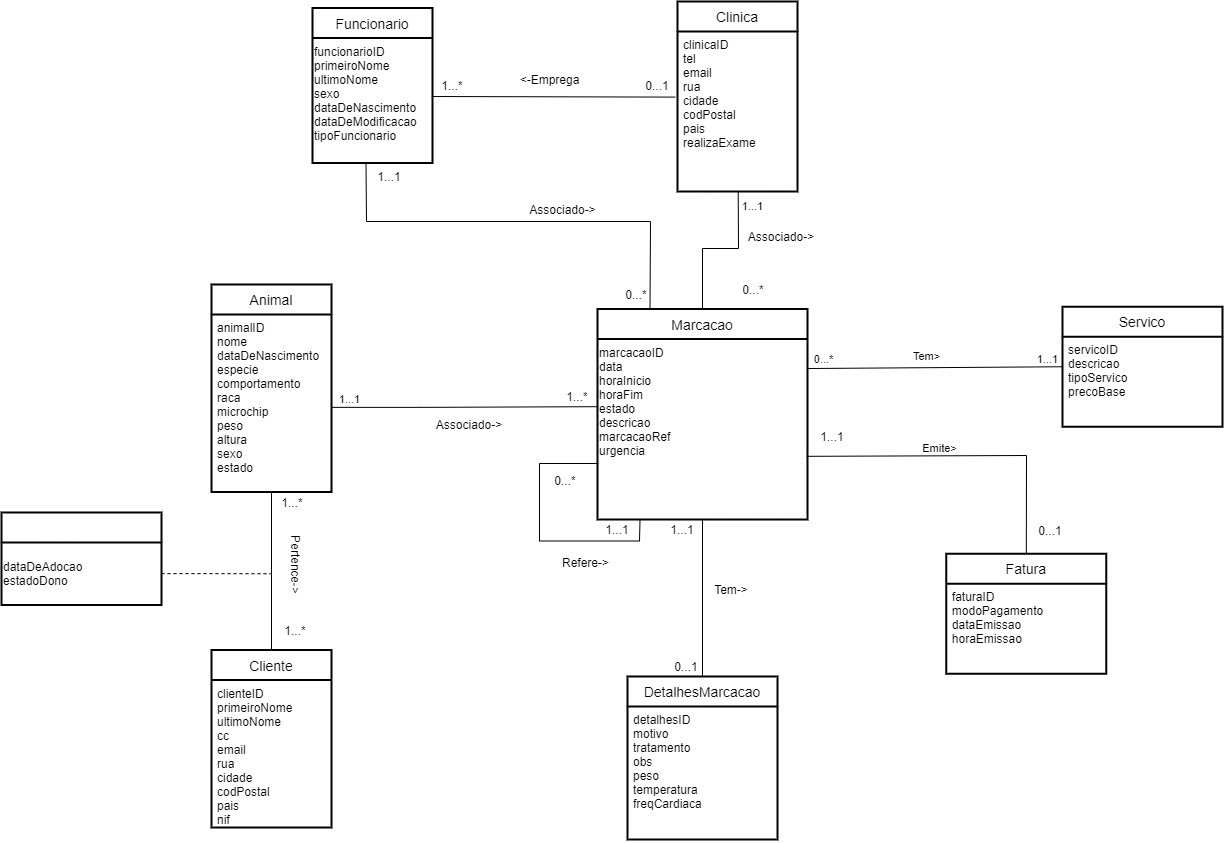


Figura 3 - Diagrama Conceptual ER

# 4 Desenho Lógico

Nesta fase o modelo conceptual vai ser traduzido para o modelo lógico, que será visto pelos utilizadores de forma a verificar se este se encontra estruturalmente correto, as fases desta etapa são:

1. Derivação de relações para o modelo de dados lógico;
2. Documentação de relações entre entidades;
3. Normalização 4. Restrições de integridade;
4. Rever o modelo lógico de dados com o utilizador.

## 4.1 Derivação de relações para o modelo de dados lógico

### 4.1.1 Entidades fortes

**Funcionario** { funcionarioID, primeiroNome, ultimoNome, sexo, dataDeNascimento, dataDeModificacao, tipoFuncionario }

**Chave primária:** funcionarioID

**Clinica**{ clinicaID, tel, email, rua, cidade, codPostal, pais, realizaExame }

**Chave primária:** clinicaID

**Animal** { animalID, nome, dataDeNascimento, especie, comportamento, raca, peso, altura, sexo, estado }

**Chave primária:** animalID

**Marcacao** { marcacaoID, data, horaInicio, horaFim, estado, descricao, marcacaoRef, urgencia }

**Chave primária:** marcacaoID

**Cliente** { clienteID, primeiroNome, ultimoNome, cc, nif, email, rua, cidade, codPostal, pais }

**Chave primária:** clienteID

### 4.1.2 Entidades fracas

**Fatura** { faturaID, modoPagamento, dataEmissao, horaEmissao }

**Chave primária:** faturaID

**DetalhesMarcacao** {detalhesID, motivo, tratamento, obs, peso, temperatura, freqCardiaca }

**Chave primária:** detalhesID

**Servico** { servicoID, descricao, tipoServico, precoBase, duracao }

**Chave primária:** servicoID

### 4.1.3 Relação de um para muitos (1 : \*)

**Clinica 1 : \* Funcionario**

***Entidade pai***

**Clinica**{ clinicaID, tel, email, rua, cidade, codPostal, pais, realizaExame }

**Chave primária:** funcionarioID

***Entidade filho***

**Funcionario**  { funcionarioID, primeiroNome, UltimoNome, sexo, dataDeNascimento, dataDeModificacao, tipoFuncionario, clinicaID }

**Chave primária:** funcionarioID

**Chave estrangeira** clinicaID **refere** Clinica (clinicaID)

Põe clinicaID em Funcionario para modelar uma relação 1 : \*

**Clinica 1 : \* Marcacao**

***Entidade pai***

**Clinica**{ clinicaID, tel, email, rua, cidade, codPostal, pais, realizaExame }

**Chave primaria** funcionarioID

***Entidade filho***

**Marcacao** {marcacaoID, data, horaInicio, horaFim, estado, descricao, marcacaoRef, urgencia, clinicaID}

**Chave primária:** marcacaoID

**Chave estrageira** clinicaID **refere** Clinica (clinicaID)

Põe clinicaID em Marcacao para modelar uma relação 1 : \*

**Animal 1 : \* Marcacao**

**Entidade pai**

**Animal** { animalID, nome, dataDeNascimento, especie, comportamento, raca, peso, altura, sexo, estado, clinicaID }

**Chave primaria** animalID

**Chave estrangeira** clinicaID **refere** Clinica (clinicaID)

**Entidade filho**

**Marcacao** { marcacaoID, data, horaInicio, horaFim, estado, descricao, marcacaoRef, urgencia, clinicaID, animalID }

**Chave primária:** marcacaoID

**Chave estrageira** clinicaID **refere** Clinica (clinicaID)

animalID **refere** Animal ( animalID)

Põe animalID em Marcacao para modelar uma relação 1 : \*

**Marcacao 1 : \* Servico**

***Entidade pai***

**Servico{**servicoID, tipoServico, descricao, precoBase, duracao**}**

**Chave primaria** servicoID

***Entidade filho***

**Marcacao** { marcacaoID, data, horaInicio, horaFim, estado, descricao, marcacaoRef, urgencia, clinicaID, animalID, servicoID }

**Chave primaria** marcacaoID

**Chave estrageira** clinicaID **refere** Clinica (clinicaID)

animalID **refere** Animal ( animalID)

servicoID refere Servico(servicoID)

Põe marcacaoIDem ServicoMarcado para modelar uma relação 1 : \*

**Marcacao 1 : \* Marcacao**

***Entidade pai***

**Marcacao** { marcacaoID, data, horaInicio, horaFim, estado, descricao, marcacaoRef, urgencia, animalID, clinicaID }

**Chave primaria** marcacaoID

**Chave estrageira** clinicaID **refere** Clinica (clinicaID)

animalID **refere** Animal ( animalID)

servicoID refere Servico(servicoID)

***Entidade filho***

**Marcacao** { marcacaoID, data, horaInicio, horaFim, estado, descricao, marcacaoRef, urgencia, animalID, clinicaID, servicoID }

**Chave primaria** marcacaoID

**Chave estrageira** clinicaID **refere** Clinica (clinicaID)

animalID **refere** Animal ( animalID)

servicoID refere Servico(servicoID)

marcacaoRef **refere** Marcacao (marcacaoID)

Põe marcacaoIDem Marcacao para modelar uma relação 1 : \*

**Funcionario 1 : \* Marcacao**

***Entidade pai***

**Funcionario**  { funcionarioID, primeiroNome, UltimoNome, sexo, dataDeNascimento, dataDeModificacao, tipoFuncionario, clinicaID }

**Chave primaria** funcionarioID

**Chave estrangeira** clinicaID **refere** Clinica (clinicaID)

***Entidade filho***

**Marcacao** { marcacaoID, data, horaInicio, horaFim, estado, descricao, marcacaoRef, urgencia, animalID, clinicaID, servicoID, funcionarioID }

**Chave primaria** marcacaoID

**Chave estrageira** clinicaID **refere** Clinica (clinicaID)

animalID **refere** Animal ( animalID)

servicoID refere Servico(servicoID)

marcacaoRef **refere** Marcacao (marcacaoID)

funcionarioID **refere** Funcionario(funcionarioID)

### 4.1.4 Relação de um para um (1 : 1)

**Marcacao 1 : 1 DetalhesMarcacao**

***Entidade pai***

**Marcacao** { marcacaoID, data, horaInicio, horaFim, estado, descricao, marcacaoRef, urgencia, animalID, clinicaID, servicoID, funcionarioID }

**Chave primaria:** marcacaoID

**Chave estrageira** clinicaID **refere** Clinica (clinicaID)

animalID **refere** Animal ( animalID)

servicoID refere Servico(servicoID)

marcacaoRef **refere** Marcacao (marcacaoID)

funcionarioID **refere** Funcionario(funcionarioID)

***Entidade filho***

**DetalhesMarcacao** { detalhesID, motivo, prescricao, obs, peso, temperatura, freqCardiaco, marcacaoID }

**Chave primaria** detalhesID

**Chave estrangeira** marcacaoID **refere** Marcacao (MarcacaoID)

Relação 1 : 1 com participação obrigatória do lado da marcacao.

**Marcacao 1 : 1 Fatura**

***Entidade pai***

**Marcacao** {marcacaoID, data, horaInicio, horaFim, estado, descricao, marcacaoRef, urgencia, animalID, clinicaID, servicoID, funcionarioID}

**Chave primaria** marcacaoID

**Chave estrageira** clinicaID **refere** Clinica (clinicaID)

animalID **refere** Animal ( animalID)

servicoID **refere** Servico(servicoID)

marcacaoRef **refere** Marcacao (marcacaoID)

funcionarioID **refere** Funcionario(funcionarioID)

***Entidade filho***

Fatura{faturaID, modoPagamento, dataEmissao, horaEmissao}

Chave primaria faturaID

Chave estrangeira marcacaoID **refere** Marcacao (MarcacaoID)

### 4.1.5 Relação de muitos para muitos (\* : \*)

**Animal \* : \* Cliente**

**Animal** {animalID, nome, dataDeNascimento, especie, comportamento, raca, peso, altura, sexo, estado, clinicaID }

**Chave primaria** animalID

**Chave estrangeira** clinicaID **refere** Clinica (clinicaID)

**Cliente** { clienteID, primeiroNome, ultimoNome, cc, nif, email, rua, cidade, codPostal, pais }

**Chave primaria** clienteID

**AnimalCliente {** clienteID , animalID, dataDeAdocao, estadoDono }

**Chave primaria** clienteID , animalID

**Chave estrangeira** clienteID **refere** Cliente(clienteID)

animalID **refere** Animal(animalID)

### 4.1.6 Multi-valued attributes

**Clinica**{clinicaID, email, rua, cidade, codPostal, pais, realizaExame}

**Chave primaria** clinicaID

**Chave estrangeira** tel **refere** Telefone (telNumber)

**Telefone** { telNo**,** clinicaID}

**Chave primaria** telNo

**Chave estrangeira** clinicaID

## 4.2 Normalização

O processo de normalização surge como uma atividade associada ao Desenho Lógico que visa validar se as relações obtidas anteriormente são válidas.

Este processo de validação assenta no estudo das dependências funcionais existentes entre atributos de uma relação

### 4.2.1 Primeira Forma Normal (1FN)

A normalização de uma tabela na 1.ª Forma Normal (1FN) exige que a tabela tenha uma estrutura bidimensional correta, ou seja, cada linha deve corresponder a um só registo e cada coluna a um só campo.

O objetivo é eliminar redundância e introduzir dados apropriados nas colunas vazias das linhas que contêm grupos repetidos

Os autores fizeram uso das dependências funcionais que descrevem o relacionamento entre atributos de uma relação, para ajudar nesta forma de normalização e nas próximas.

### 4.2.2 Segunda Forma Normal (2FN)

A 2.ª forma normal diz que a tabela tem de estar na 1FN e que cada atributo não chave tem de ser funcionalmente dependente da totalidade da chave primária e não apenas de uma parte dessa chave.

Assim depois de identificada a chave primária de uma tabela, pode dar-se um dos dois casos:

A chave primária é constituída por um só atributo (chave elementar). Neste caso, a tabela está seguramente na 2FN (nenhum atributo depende de uma parte da chave, visto que a chave não é composta por partes);

A chave primária é constituída por mais que um atributo (chave primária composta). Neste caso, se existe algum ou alguns atributos que dependem de uma parte da chave (ou seja, de algum atributo que constitui a chave), então a tabela não está na 2FN.

### 4.2.3 Terceira Forma Normal (3FN)

A 3.ª Forma Normal (3FN) diz que a tabela tem de estar na 2FN e que nenhum atributo não chave pode depender funcionalmente de algum outro atributo que não seja a chave primária.

É baseada no conceito de dependência transitiva.

Portanto, para normalizar uma tabela de acordo com a 3FN, deve-se analisar todos os atributos não chave e verificar se existem se existem dependências transitivas sobre a chave primária, removê-las e colocá-las numa nova relação

### 4.2.4 Mockups

#### 4.2.4.1 Marcação de serviços

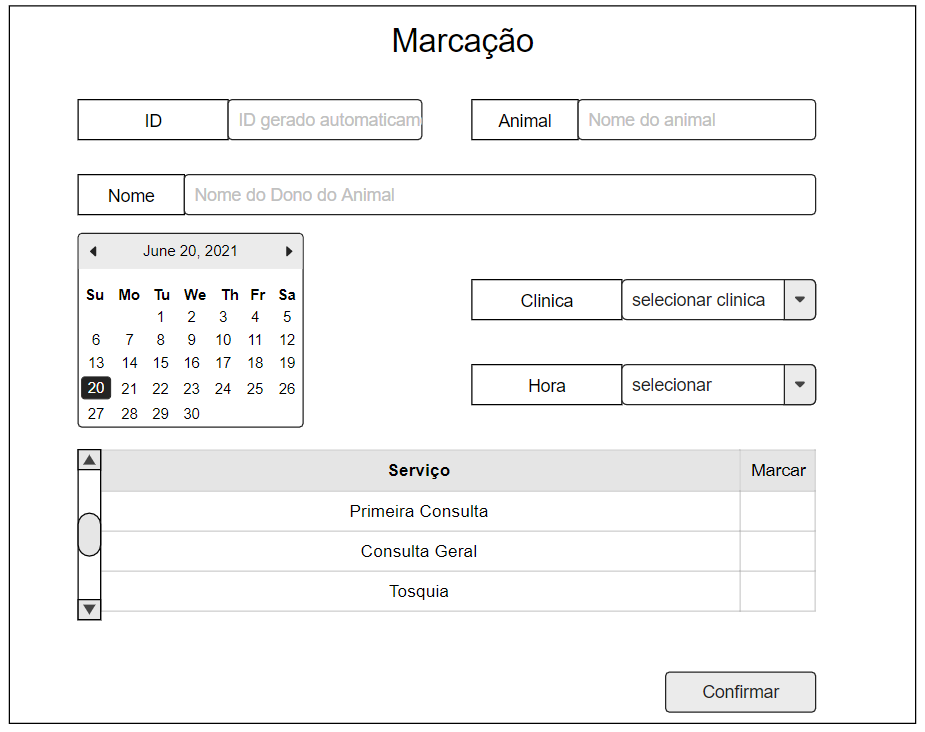


Figura 4 - Mockup Marcação

**Forma não normalizada**

**UNF** { clinicaID, telID, telNo, email, rua, cidade, codPostal, pais, realizaExame, animalID, nome, dataDeNascimento, especie, comportamento, raca, microchip, peso, altura, sexo, estado, marcacaoID, data, horaInicio, horaFim, estado, descricao, marcacaoRef, urgencia, servicoID, tipoServico, precoBase, descricao, duracao }

**1º Forma**

**Dependências funcionais**

clinicaID -> email, rua, cidade, codPostal, pais, realizaExame.

telID -> telNo

marcacaoID -> data, horaInicio, horaFim, estado, descricao, marcacaoRef, urgencia

animalID -> nome, dataDeNascimento, especie, comportamento, raca, microchip, peso, altura, sexo, estado,

servicoID -> descricao, tipoServico, precoBase, duracao

Foi feito um ‘flattening’ da tabela e obteve-se as entidades como estão nas dependências funcionais.

**2ª Forma**

**Dependências funcionais totais**

**Tabelas**

**Clinica**

ClinicaID -> email, rua, cidade, codPostal, pais, realizaExame.

**Marcacao**

marcacaoID -> data, horaInicio, horaFim, estado, descrica, marcacaoRef, urgencia

**Animal**

animalID -> nome, dataDeNascimento, especie, comportamento, raca, microchip, peso, altura, sexo, estado

**Servico**

ServicoID -> descricao, tipoServico, precoBase, duracao

Como têm todas só uma chave primaria então não existem dependências parciais o que significa que já estão todas na segunda forma de normalização

**3º Forma**

animalID -> nome, dataDeNascimento, especie, comportamento, raca, microchip, peso, altura, sexo, estado,

A raça depende da espécie.

Serão criadas 2 novas entidades

**Raca** { **racaID**, descricao, especieID}

**Especie** { **especieID**, raca}

**Animal** {animalID, nome, dataDeNascimento, comportamento, racaID, microchip, peso, altura, sexo, estado}

clinicaID -> email, rua, cidade, codPostal, pais, realizaExame

Endereço {codPostal, rua, cidade, pais}

Clinica{clinicaID, email, realizaExame}

Como não há nenhuma vantagem em criar uma entidade para endereço mantém-se a entidade cliente como está.

#### 4.2.4.2 Criação de um cliente

Figura 5 - Mockup Cliente

**Forma não normalizada**

**UNF** { animalID, nome, dataDeNascimento, especie, comportamento, raca, microchip, peso, altura, sexo, estado, clienteID, primeiroNome, ultimoNome, cc, email, rua, cidade, codPostal, pais, nif }

**1º Forma**

**Dependencias funcionais**

clienteID -> primeiroNome, ultimoNome, cc, email, rua, cidade, codPostal, pais, nif

animalID -> nome, dataDeNascimento, especie, comportamento, raca, microchip, peso, altura, sexo, estado

**2ª Forma**

**Dependências funcionais totais**

**Tabelas**

**Cliente**

clienteID, primeiroNome, ultimoNome, cc, email, rua, cidade, codPostal, pais, nif

**Animal**

animalID, nome, dataDeNascimento, especie, comportamento, raca, microchip, peso, altura, sexo, estado

Como têm todas só uma chave primaria então não existem dependências parciais o que significa que já estão todas na segunda forma de normalização

**3º Forma**

clienteID, primeiroNome, ultimoNome, cc, email, rua, cidade, codPostal, pais, nif

**Cliente** {clienteID, primeiroNome, ultimoNome, cc, nif}

**Endereço** { CodPostal, rua, cidade, pais}

Como não há endereços iguais no codpostal, não há nenhuma vantagem em criar uma entidade para endereço vamos manter a entidade cliente como está.

A entidade Animal ficará como no estudo anterior

**Raca** { racaID, descricao, especieID}

**Especie** { especieID, raca}

**Animal** {animalID, nome, dataDeNascimento, comportamento, racaID, microchip, peso, altura, sexo, estado}

#### 4.2.4.3 Visualização do histórico do animal

Figura 6 - Mockup Histórico Animal

**Forma não normalizada**

**UNF** { animalID, nome, dataDeNascimento, especie, comportamento, raca, microchip, peso, altura, sexo, estado, detalhesID, motivo, tratamento, obs, peso, temperatura, freqCardiaca, clinicaID, telID, telNo, email, rua, cidade, codPostal, pais, realizaExame }

**1º Forma**

**Dependências funcionais**

animalID -> nome, dataDeNascimento, especie, comportamento, raca, microchip, peso, altura, sexo, estado

detalhesID -> motivo, tratamento, obs, peso, temperatura, freqCardiaca

clinicaID -> email, rua, cidade, codPostal, pais, realizaExame

telD -> telNo

Após um preenchimento da UNF verificou-se que havia uma repetição da raça do animal, portanto criou-se duas novas relações.

especieID -> descricao

racaID -> descricao

**2ª Forma**

**Dependências funcionais totais**

**Tabelas**

**DetalhesMarcacao**

detalhesID, motivo, tratamento, obs, peso, temperatura, freqCardiaca,

**Animal**

animalID, nome, dataDeNascimento, especie, comportamento, raca, microchip, peso, altura, sexo, estado

**Especie**

especieID, descricao

**Raca**

racaID, descricao

**Clinica**

clinicaID, email, rua, cidade, codPostal, pais, realizaExame

**Telefone**

telD, telNo

Como têm todas só uma chave primaria então não existem dependências parciais o que significa que já estão todas na segunda forma de normalização

**3º Forma**

clinicaID -> email, rua, cidade, codPostal, pais, realizaExame

Endereço {codPostal, rua, cidade, pais}

Clinica{clinicaID, email, realizaExame}

Como não há nenhuma vantagem em criar uma entidade para endereço mantém-se a entidade cliente como está.

A entidade Animal ficará como no estudo anterior

**Raca** { racaID, descricao, especieID}

**Especie** { especieID, raca}

**Animal** {animalID, nome, dataDeNascimento, comportamento, racaID, microchip, peso, altura, sexo, estado}

#### 4.2.4.4 Preencher detalhes de Marcação



Figura 7 - Mockup Detalhes de Marcação

**Forma não normalizada**

**UNF** { animalID, nome, dataDeNascimento, especie, comportamento, raca, microchip, peso, altura, sexo, estado, detalhesID, motivo, tratamento, obs, peso, temperatura, freqCardiaca, marcacaoID, data, horaInicio, horaFim, duracao, estado, descricao, marcacaoRef, urgencia, funcionarioID, primeiroNome, ultimoNome, sexo, dataDeNascimento, dataDeModificacao, tipoFuncionario, clienteID, primeiroNome, ultimoNome, cc, email, rua, cidade, codPostal, pais, nif }

**1º Forma**

**Dependencias funcionais**

animalID -> nome, dataDeNascimento, especie, comportamento, raca, microchip, peso, altura, sexo, estado

detalhesID -> motivo, tratamento, obs, peso, temperatura, freqCardiaca

clienteID -> primeiroNome, ultimoNome, cc, email, rua, cidade, codPostal, pais, nif

telD -> telNo

funcionarioID -> primeiroNome, ultimoNome, sexo, dataDeNascimento, dataDeModificacao, tipoFuncionario

marcacaoID -> data, horaInicio, horaFim, duracao, estado, descricao, marcacaoRef, urgencia

Após um preenchimento da UNF verificou-se que havia uma repetição do tipo de funcionário, portanto criou-se uma nova relação.

**2ª Forma**

**Dependências funcionais totais**

**Tabelas**

**Funcionario**

funcionarioID, primeiroNome, ultimoNome, sexo, dataDeNascimento, dataDeModificacao

**TipoFuncionario**

tipoID, tipoFuncionario

**Marcacao**

marcacaoID, data, horaInicio, horaFim, duracao, estado, descricao, marcacaoRef, urgencia

**DetalhesMarcacao**

detalhesID, motivo, tratamento, obs, peso, temperatura, freqCardiaca

**Animal**

animalID, nome, dataDeNascimento, comportamento, microchip, peso, altura, sexo, estado

**Raca**

racaID, descricao

**Especie**

especieID, descricao

**Cliente**

clienteID, primeiroNome, ultimoNome, cc, email, rua, cidade, codPostal, pais, nif

**Telefone**

telD, telNo

Como têm todas só uma chave primaria então não existem dependências parciais o que significa que já estão todas na segunda forma de normalização

**3ª Forma**

A entidade Animal ficará como no estudo anterior

**Raca** { racaID, descricao, especieID }

**Especie** { especieID, raca }

**Animal** { animalID, nome, dataDeNascimento, comportamento, racaID, microchip, peso, altura, sexo, estado }

**Cliente** { clienteID, primeiroNome, ultimoNome, cc, nif }

**Endereço** { CodPostal, rua, cidade, pais }

Como não há endereços iguais no codpostal, não há nenhuma vantagem em criar uma entidade para endereço, mantém-se a entidade cliente como está.

#### 4.2.4.6 Registar Funcionário

Figura 8 - Mockup Registo Funcionário

**Forma não normalizada**

**UNF** {funcionarioID, primeiroNome, ultimoNome, sexo, dataDeDascimento, dataDeModificacao, tipoFuncionario }

**1º Forma**

**Dependências funcionais**

funcionarioID -> primeiroNome, ultimoNome, sexo, dataDeNascimento, dataDeModificacao, tipoFuncionario

Após o preenchimento da UNF ao fazer o ‘flatening’ da tabela verificou-se que havia uma repetição do tipo de funcionário portanto criou-se uma nova relação.

**TipoFuncionario**

tipoID, TipoFuncionario

**2ª Forma**

**Dependências funcionais totais**

**Tabelas**

**Funcionario**

funcionarioID, primeiroNome, ultimoNome, sexo, dataDeNascimento, dataDeModificacao

**TipoFuncionario**

tipoID, tipoFuncionario

Como têm todas só uma chave primaria então não existem dependências parciais o que significa que já estão todas na segunda forma de normalização

**3º Forma**

A relação está na 2ª Forma Normal, mas não tem dependências transitivas (não existirem atributos descritores a dependerem funcionalmente de outros atributos descritores (não chaves)), ou seja, assume-se que se encontra na 3ª Forma Normal.

# 5 Restrições de integridade – Regras de negócio

**Cliente**

- Primeiro e Último nomes, CC, NIF e tel são campos de preenchimento obrigatório.

- O CC deve ter exatamente 8 dígitos e ser valido.

- O sexo é definido por M – Masculino, F – Feminino ou O – outro.

- O NIF deve ter exatamente 9 dígitos e serem válidos.

- O codPostal teve ter o formato XXXX-XXX e ser validado.

- O clienteID deve ter até 6 dígitos.

**Funcionário**

- O FuncionarioID, Primeiro e Último nomes e tipoFuncionario são de preenchimento obrigatório.

- O funcionarioID deve até 5 digitos.

- O sexo é definido por M – Masculino, F – Feminino ou O – outro.

**Animal**

- Campos animalID, nome, dataDeNascimento, especie, raca são de preenchimento obrigatório. Nos casos de desconhecimento do(s) campo(s) dataDeNascimento e/ou raca, é o veterinário que os define.

- O sexo é definido por M – Masculino ou F – Feminino.

- O animalID deve ter até 6 dígitos.

- Comportamento tem que ser Agressivo, calmo ou Normal.

- O estado tem que ser 0 ou 1 para funcionar como um boolean.

**Marcação**

- Todos os campos são obrigatórios á exceção de marcacaoRef e horaFim;

- A hora de fim não pode ser maior que a hora de início.

- Não podem ser marcadas fora do horário de trabalho marcações que não têm true na urgencia

- Marcações aos fim-de-semanas são sempre urgência.

- marcaçãoID deve ter exatamente 9 dígitos

**Servico**

- Todos os campos são obrigatórios.

- O tipo é definido por serviço ou marcação.

**AnimalCliente**

-Todos os campos são obrigatórios.

**Raca**

- Todos os campos são obrigatórios.

**Especie**

- Todos os campos são obrigatórios.

**Telefone**

- Todos os campos são obrigatórios.

- O número de telefone deve ter exatamente 9 dígitos.

**ModoPagamentos**

- Todos os campos são obrigatórios.

**Fatura**

- Todos os campos são obrigatórios.

**TipoFuncionario**

- Todos os campos são obrigatórios.

**Clinica**

- Todos os campos são obrigatórios.

- O codPostal teve ter o formato XXXX-XXX e ser validado.

- O número de telefone deve ter exatamente 9 dígitos.

**DetalhesMarcacao**

-DetalhesID, motivo e tratamento são obrigatórios.

**Integridade referencial**

Tabela 12 - Integridade Referencial

|  |  |  |  |
| --- | --- | --- | --- |
| Tabela base da primary key | Tabela base da foreign key | Clausula  DELETE | Clausula  UPDATE |
| Funcionario | Marcacao | No Action | No Action |
| Clinica | Marcacao | No Action | No Action |
| Animal | Marcacao | No Action | No Action |
| Servico | Marcacao | No Action | No Action |
| Marcacao | DetalhesMarcacao | No Action | CASCADE |
| Marcacao | Fatura | No Action | CASCADE |
| Clinica | Funcionario | No Action | CASCADE |
| TipoFuncionario | Funcionario | No Action | CASCADE |
| Animal | AnimalCliente | No Action | CASCADE |
| Cliente | AnimalCliente | No Action | CASCADE |
| ModoPagamento | Fatura | No Action | CASCADE |
| Telefone | Clinica | No Action | CASCADE |
| Especie | Raca | No Action | CASCADE |
| Raca | Animal | No Action | CASCADE |
| Marcacao | Marcacao | No Action | No Action |

# 6 Desenho Físico

Nesta Etapa, o modelo logico será traduzido para um modelo físico através de um SGBD, que neste caso foi usado o Microsoft SQL Server, e relatado o processo nas fases:

1. Desenho das relações e restrições

2. Representação de Dados Derivados

3. Desenho das Restrições Gerais

4. Desenho das Vistas de Utilizador

5. Desnormalização

6. Monitorização e Manutenção do Sistema

**Criação de tipos de dados**

Tabela 13 - Tipos de dados

|  |  |  |
| --- | --- | --- |
| Nome | Tipo de dado | Tamanho | precisão | escala |
| tipoFuncionarioID | Numeric | (1) |
| funcionarioID | Numeric | (3) |
| NIF | Numeric | (9) |
| Tel | Numeric | (9) |
| animalID | Numeric | (6) |
| clinicaID | Numeric | (4) |
| marcacaoID | Numeric | (9) |
| servicoID | Numeric | (3) |
| faturaID | Numeric | (9) |
| modoPagamentoID | Numeric | (1) |
| detalhesID | Numeric | (3) |
| clienteID | Numeric | (6) |
| racaID | Numeric | (4) |
| especieID | Numeric | (2) |

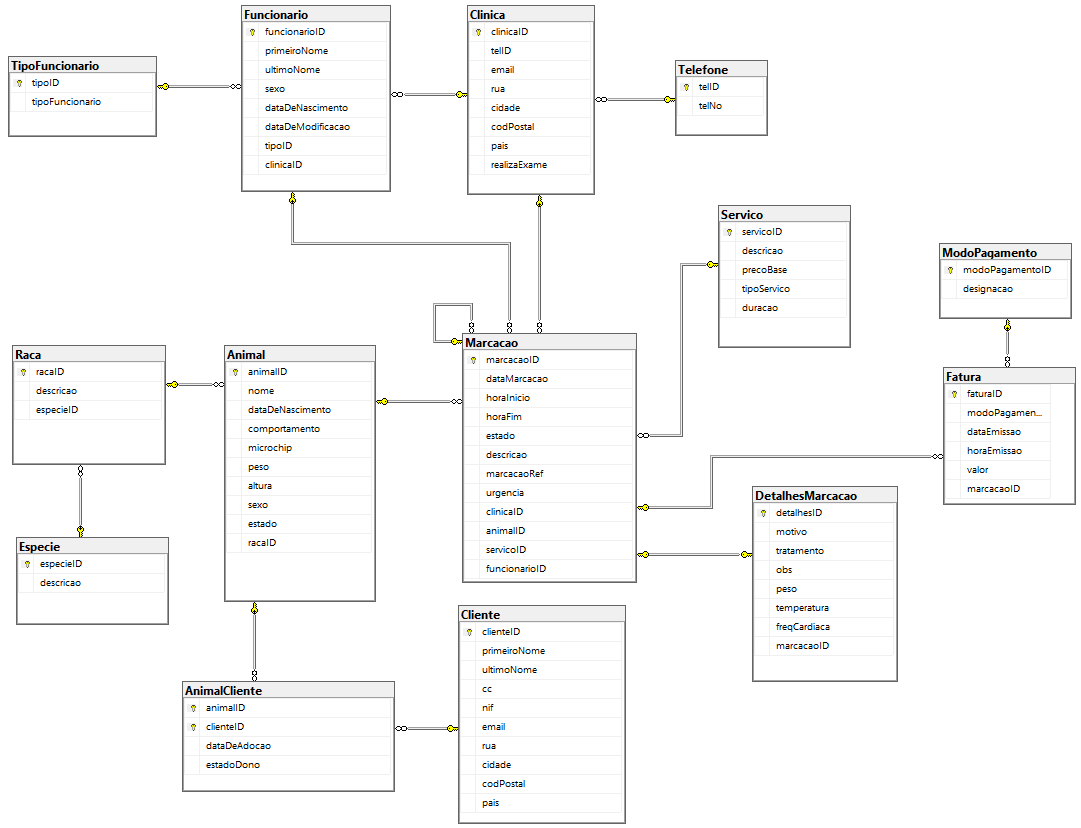


Figura 9 Diagrama ER do SQL Management Studio

## 6.1 Criação da Tabelas, relacionamentos e restrições aos atributos

### 6.1.1 Criação da Tabela Animal

USE [cliPet]

GO

-- Object: Table [dbo].[Animal]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Animal](

[animalID] [dbo].[animalIDType] IDENTITY(1,1) NOT NULL,

[nome] [varchar](15) NOT NULL,

[dataDeNascimento] [date] NOT NULL,

[comportamento] [varchar](15) NOT NULL,

[microchip] [dbo].[microchipType] NULL,

[peso] [dbo].[measurementType] NOT NULL,

[altura] [dbo].[measurementType] NOT NULL,

[sexo] [varchar](1) NOT NULL,

[estado] [bit] NOT NULL,

[racaID] [dbo].[racaID] NOT NULL,

CONSTRAINT [PK\_Animal] PRIMARY KEY CLUSTERED

(

[animalID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Animal] ADD CONSTRAINT [DF\_Animal\_sexo] DEFAULT (N'M') FOR [sexo]

GO

ALTER TABLE [dbo].[Animal] WITH CHECK ADD CONSTRAINT [FK\_Animal\_Raca] FOREIGN KEY([racaID])

REFERENCES [dbo].[Raca] ([racaID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[Animal] CHECK CONSTRAINT [FK\_Animal\_Raca]

GO

ALTER TABLE [dbo].[Animal] WITH CHECK ADD CONSTRAINT [CK\_comportamento] CHECK (([comportamento]='Normal' OR [comportamento]='Calmo' OR [comportamento]='Agressivo'))

GO

ALTER TABLE [dbo].[Animal] CHECK CONSTRAINT [CK\_comportamento]

GO

ALTER TABLE [dbo].[Animal] WITH CHECK ADD CONSTRAINT [CK\_sexo] CHECK (([sexo]='F' OR [sexo]='M'))

GO

ALTER TABLE [dbo].[Animal] CHECK CONSTRAINT [CK\_sexo]

GO

Excerto SQL 1 - Criaçao Tabela Animal

### 6.1.2 Criação da Tabela AnimalCliente

USE [cliPet]

GO

-- Object: Table [dbo].[AnimalCliente]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AnimalCliente](

[animalID] [dbo].[animalIDType] NOT NULL,

[clienteID] [dbo].[clienteIDType] NOT NULL,

[dataDeAdocao] [date] NOT NULL,

[estadoDono] [varchar](10) NOT NULL,

CONSTRAINT [PK\_AnimalCliente] PRIMARY KEY CLUSTERED

(

[animalID] ASC,

[clienteID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[AnimalCliente] WITH CHECK ADD CONSTRAINT [FK\_AnimalCliente\_Animal] FOREIGN KEY([animalID])

REFERENCES [dbo].[Animal] ([animalID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[AnimalCliente] CHECK CONSTRAINT [FK\_AnimalCliente\_Animal]

GO

ALTER TABLE [dbo].[AnimalCliente] WITH CHECK ADD CONSTRAINT [FK\_AnimalCliente\_Cliente] FOREIGN KEY([clienteID])

REFERENCES [dbo].[Cliente] ([clienteID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[AnimalCliente] CHECK CONSTRAINT [FK\_AnimalCliente\_Cliente]

GO

ALTER TABLE [dbo].[AnimalCliente] WITH CHECK ADD CONSTRAINT [CK\_AnimalCliente] CHECK (([estadoDono]='NaoAtivo' OR [estadoDono]='Ativo'))

GO

ALTER TABLE [dbo].[AnimalCliente] CHECK CONSTRAINT [CK\_AnimalCliente]

GO

Excerto SQL 2 - Criação da Tabela AnimalCliente

### 6.1.3 Criação da Tabela Cliente

USE [cliPet]

GO

-- Object: Table [dbo].[Cliente]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Cliente](

[clienteID] [dbo].[clienteIDType] IDENTITY(1,1) NOT NULL,

[primeiroNome] [varchar](15) NOT NULL,

[ultimoNome] [varchar](15) NOT NULL,

[cc] [dbo].[ccType] NOT NULL,

[nif] [dbo].[telNifType] NOT NULL,

[email] [varchar](256) NOT NULL,

[rua] [varchar](50) NOT NULL,

[cidade] [varchar](50) NOT NULL,

[codPostal] [varchar](8) NOT NULL,

[pais] [varchar](2) NOT NULL,

CONSTRAINT [PK\_Cliente] PRIMARY KEY CLUSTERED

(

[clienteID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Cliente] ADD CONSTRAINT [DF\_Cliente\_pais] DEFAULT (N'PT') FOR [pais]

GO

ALTER TABLE [dbo].[Cliente] WITH CHECK ADD CONSTRAINT [CK\_CC] CHECK ((len([cc])=(8)))

GO

ALTER TABLE [dbo].[Cliente] CHECK CONSTRAINT [CK\_CC]

GO

ALTER TABLE [dbo].[Cliente] WITH CHECK ADD CONSTRAINT [CK\_CodPostal] CHECK (([codPostal] like '[1-9][0-9][0-9][0-9][-][0-9][0-9][0-9]'))

GO

ALTER TABLE [dbo].[Cliente] CHECK CONSTRAINT [CK\_CodPostal]

GO

ALTER TABLE [dbo].[Cliente] WITH CHECK ADD CONSTRAINT [CK\_Email] CHECK (([email] like '%[A-Z0-9][@][A-Z0-9]%[.][A-Z0-9]%'))

GO

ALTER TABLE [dbo].[Cliente] CHECK CONSTRAINT [CK\_Email]

GO

ALTER TABLE [dbo].[Cliente] WITH CHECK ADD CONSTRAINT [CK\_NIF] CHECK ((len([nif])=(9)))

GO

ALTER TABLE [dbo].[Cliente] CHECK CONSTRAINT [CK\_NIF]

GO

Excerto SQL 3 - Criação da Tabela Cliente

### 6.1.4 Criação da Tabela Clínica

USE [cliPet]

GO

-- Object: Table [dbo].[Clinica]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Clinica](

[clinicaID] [dbo].[clinicaIDType] IDENTITY(1,1) NOT NULL,

[telID] [dbo].[teIIDType] NOT NULL,

[email] [varchar](256) NOT NULL,

[rua] [varchar](50) NOT NULL,

[cidade] [varchar](50) NOT NULL,

[codPostal] [varchar](8) NOT NULL,

[pais] [varchar](2) NOT NULL,

[realizaExame] [bit] NOT NULL,

CONSTRAINT [PK\_Clinica] PRIMARY KEY CLUSTERED

(

[clinicaID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Clinica] WITH CHECK ADD CONSTRAINT [FK\_Clinica\_Telefone] FOREIGN KEY([telID])

REFERENCES [dbo].[Telefone] ([telID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[Clinica] CHECK CONSTRAINT [FK\_Clinica\_Telefone]

GO

ALTER TABLE [dbo].[Clinica] WITH CHECK ADD CONSTRAINT [CK\_ClinicaEmail] CHECK (([email] like '%[A-Z0-9][@][A-Z0-9]%[.][A-Z0-9]%'))

GO

ALTER TABLE [dbo].[Clinica] CHECK CONSTRAINT [CK\_ClinicaEmail]

GO

ALTER TABLE [dbo].[Clinica] WITH CHECK ADD CONSTRAINT [CK\_ClinicaPostal] CHECK (([codPostal] like '[1-9][0-9][0-9][0-9][-][0-9][0-9][0-9]'))

GO

ALTER TABLE [dbo].[Clinica] CHECK CONSTRAINT [CK\_ClinicaPostal]

GO

Excerto SQL 4 - Criação da Tabela Clínica

### 6.1.5 Criação da Tabela DetalhesMarcacao

USE [cliPet]

GO

-- Object: Table [dbo].[DetalhesMarcacao]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[DetalhesMarcacao](

[detalhesID] [dbo].[detalhesIDType] IDENTITY(1,1) NOT NULL,

[motivo] [varchar](50) NOT NULL,

[tratamento] [varchar](100) NOT NULL,

[obs] [varchar](100) NULL,

[peso] [dbo].[measurementType] NULL,

[temperatura] [decimal](3, 1) NULL,

[freqCardiaca] [numeric](3, 0) NULL,

[marcacaoID] [dbo].[marcacaoID] NOT NULL,

CONSTRAINT [PK\_DetalhesMarcacao] PRIMARY KEY CLUSTERED

(

[detalhesID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[DetalhesMarcacao] WITH CHECK ADD CONSTRAINT [FK\_DetalhesMarcacao\_Marcacao] FOREIGN KEY([marcacaoID])

REFERENCES [dbo].[Marcacao] ([marcacaoID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[DetalhesMarcacao] CHECK CONSTRAINT [FK\_DetalhesMarcacao\_Marcacao]

GO

Excerto SQL 5 - Criação da Tabela DetalhesMarcacao

### 6.1.6 Criação da Tabela Especie

USE [cliPet]

GO

-- Object: Table [dbo].[Especie]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Especie](

[especieID] [dbo].[especieID] IDENTITY(1,1) NOT NULL,

[descricao] [varchar](15) NOT NULL,

CONSTRAINT [PK\_Especie] PRIMARY KEY CLUSTERED

(

[especieID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

Excerto SQL 6 - Criação da Tabela Especie

### 6.1.7 Criação da Tabela Fatura

USE [cliPet]

GO

-- Object: Table [dbo].[Fatura]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Fatura](

[faturaID] [dbo].[faturaIDType] IDENTITY(1,1) NOT NULL,

[modoPagamentoID] [dbo].[modoPagamentoIDType] NOT NULL,

[dataEmissao] [date] NOT NULL,

[horaEmissao] [time](7) NOT NULL,

[valor] [smallmoney] NOT NULL,

[marcacaoID] [dbo].[marcacaoID] NOT NULL,

CONSTRAINT [PK\_Fatura] PRIMARY KEY CLUSTERED

(

[faturaID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Fatura] WITH CHECK ADD CONSTRAINT [FK\_Fatura\_Marcacao] FOREIGN KEY([marcacaoID])

REFERENCES [dbo].[Marcacao] ([marcacaoID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[Fatura] CHECK CONSTRAINT [FK\_Fatura\_Marcacao]

GO

ALTER TABLE [dbo].[Fatura] WITH CHECK ADD CONSTRAINT [FK\_Fatura\_ModoPagamento] FOREIGN KEY([modoPagamentoID])

REFERENCES [dbo].[ModoPagamento] ([modoPagamentoID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[Fatura] CHECK CONSTRAINT [FK\_Fatura\_ModoPagamento]

GO

Excerto SQL 7 - Criação da Tabela Fatura

### 6.1.8 Criação da Tabela Funcionario

USE [cliPet]

GO

-- Object: Table [dbo].[Funcionario]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Funcionario](

[funcionarioID] [dbo].[funcionarioIDType] IDENTITY(1,1) NOT NULL,

[primeiroNome] [varchar](15) NOT NULL,

[ultimoNome] [varchar](15) NOT NULL,

[sexo] [varchar](1) NOT NULL,

[dataDeNascimento] [date] NOT NULL,

[dataDeModificacao] [datetime] NOT NULL,

[tipoID] [dbo].[tipoFuncionarioIDType] NOT NULL,

[clinicaID] [dbo].[clinicaIDType] NOT NULL,

CONSTRAINT [PK\_Funcionario] PRIMARY KEY CLUSTERED

(

[funcionarioID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Funcionario] WITH CHECK ADD CONSTRAINT [FK\_Funcionario\_Clinica] FOREIGN KEY([clinicaID])

REFERENCES [dbo].[Clinica] ([clinicaID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[Funcionario] CHECK CONSTRAINT [FK\_Funcionario\_Clinica]

GO

ALTER TABLE [dbo].[Funcionario] WITH CHECK ADD CONSTRAINT [FK\_Funcionario\_TipoFuncionario] FOREIGN KEY([tipoID])

REFERENCES [dbo].[TipoFuncionario] ([tipoID])

ON UPDATE CASCADE

GO

ALTER TABLE [dbo].[Funcionario] CHECK CONSTRAINT [FK\_Funcionario\_TipoFuncionario]

GO

ALTER TABLE [dbo].[Funcionario] WITH CHECK ADD CONSTRAINT [CK\_Funcionario] CHECK (((datepart(year,getdate())-datepart(year,[dataDeNascimento]))>=(18)))

GO

ALTER TABLE [dbo].[Funcionario] CHECK CONSTRAINT [CK\_Funcionario]

GO

ALTER TABLE [dbo].[Funcionario] WITH CHECK ADD CONSTRAINT [CK\_FuncionarioSexo] CHECK (([sexo]='O' OR [sexo]='F' OR [sexo]='M'))

GO

ALTER TABLE [dbo].[Funcionario] CHECK CONSTRAINT [CK\_FuncionarioSexo]

GO

Excerto SQL 8 - Criação da Tabela Funcionario

### 6.1.9 Criação da Tabela Marcação

USE [cliPet]

GO

-- Object: Table [dbo].[Marcacao]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Marcacao](

[marcacaoID] [dbo].[marcacaoID] IDENTITY(1,1) NOT NULL,

[dataMarcacao] [date] NOT NULL,

[horaInicio] [time](7) NOT NULL,

[horaFim] [time](7) NULL,

[estado] [varchar](15) NOT NULL,

[descricao] [varchar](100) NOT NULL,

[marcacaoRef] [dbo].[marcacaoID] NULL,

[urgencia] [bit] NOT NULL,

[clinicaID] [dbo].[clinicaIDType] NOT NULL,

[animalID] [dbo].[animalIDType] NOT NULL,

[servicoID] [dbo].[servicoIDType] NOT NULL,

[funcionarioID] [dbo].[funcionarioIDType] NOT NULL,

CONSTRAINT [PK\_Marcacao] PRIMARY KEY CLUSTERED

(

[marcacaoID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Marcacao] WITH CHECK ADD CONSTRAINT [FK\_Marcacao\_Animal] FOREIGN KEY([animalID])

REFERENCES [dbo].[Animal] ([animalID])

GO

ALTER TABLE [dbo].[Marcacao] CHECK CONSTRAINT [FK\_Marcacao\_Animal]

GO

ALTER TABLE [dbo].[Marcacao] WITH CHECK ADD CONSTRAINT [FK\_Marcacao\_Clinica] FOREIGN KEY([clinicaID])

REFERENCES [dbo].[Clinica] ([clinicaID])

GO

ALTER TABLE [dbo].[Marcacao] CHECK CONSTRAINT [FK\_Marcacao\_Clinica]

GO

ALTER TABLE [dbo].[Marcacao] WITH CHECK ADD CONSTRAINT [FK\_Marcacao\_Funcionario] FOREIGN KEY([funcionarioID])

REFERENCES [dbo].[Funcionario] ([funcionarioID])

GO

ALTER TABLE [dbo].[Marcacao] CHECK CONSTRAINT [FK\_Marcacao\_Funcionario]

GO

ALTER TABLE [dbo].[Marcacao] WITH CHECK ADD CONSTRAINT [FK\_Marcacao\_Marcacao] FOREIGN KEY([marcacaoRef])

REFERENCES [dbo].[Marcacao] ([marcacaoID])

GO

ALTER TABLE [dbo].[Marcacao] CHECK CONSTRAINT [FK\_Marcacao\_Marcacao]

GO

ALTER TABLE [dbo].[Marcacao] WITH CHECK ADD CONSTRAINT [FK\_Marcacao\_Servico] FOREIGN KEY([servicoID])

REFERENCES [dbo].[Servico] ([servicoID])

GO

ALTER TABLE [dbo].[Marcacao] CHECK CONSTRAINT [FK\_Marcacao\_Servico]

GO

ALTER TABLE [dbo].[Marcacao] WITH CHECK ADD CONSTRAINT [CK\_Hora] CHECK (([horaFim]>[horaInicio]))

GO

ALTER TABLE [dbo].[Marcacao] CHECK CONSTRAINT [CK\_Hora]

GO

ALTER TABLE [dbo].[Marcacao] WITH CHECK ADD CONSTRAINT [CK\_MarcacaoEstado] CHECK (([estado]='Terminada' OR [estado]='Remarcada' OR [estado]='Cancelada' OR [estado]='Confirmada'))

GO

ALTER TABLE [dbo].[Marcacao] CHECK CONSTRAINT [CK\_MarcacaoEstado]

GO

Excerto SQL 9 - Criação da Tabela Marcação

### 6.1.10. Criação da Tabela ModoPagamento

USE [cliPet]

GO

-- Object: Table [dbo].[ModoPagamento]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[ModoPagamento](

[modoPagamentoID] [dbo].[modoPagamentoIDType] IDENTITY(1,1) NOT NULL,

[designacao] [varchar](25) NOT NULL,

CONSTRAINT [PK\_ModoPagamento] PRIMARY KEY CLUSTERED

(

[modoPagamentoID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

Excerto SQL 10 - Criação da Tabela ModoPagamento

### 6.1.11. Criação da Tabela Raca

USE [cliPet]

GO

-- Object: Table [dbo].[Raca]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Raca](

[racaID] [dbo].[racaID] IDENTITY(1,1) NOT NULL,

[descricao] [varchar](50) NOT NULL,

[especieID] [dbo].[especieID] NOT NULL,

CONSTRAINT [PK\_Raca] PRIMARY KEY CLUSTERED

(

[racaID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Raca] WITH CHECK ADD CONSTRAINT [FK\_Raca\_Especie] FOREIGN KEY([especieID])

REFERENCES [dbo].[Especie] ([especieID])

GO

ALTER TABLE [dbo].[Raca] CHECK CONSTRAINT [FK\_Raca\_Especie]

GO

Excerto SQL 11 - Criação da Tabela Raca

### 6.1.12. Criação da Tabela Servico

USE [cliPet]

GO

-- Object: Table [dbo].[Servico]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Servico](

[servicoID] [dbo].[servicoIDType] IDENTITY(1,1) NOT NULL,

[descricao] [varchar](100) NOT NULL,

[precoBase] [smallmoney] NOT NULL,

[tipoServico] [varchar](50) NOT NULL,

[duracao] [numeric](3, 0) NOT NULL,

CONSTRAINT [PK\_Servico] PRIMARY KEY CLUSTERED

(

[servicoID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Servico] WITH CHECK ADD CONSTRAINT [CK\_Servico] CHECK (([tipoServico]='Consulta' OR [tipoServico]='Exame'))

GO

ALTER TABLE [dbo].[Servico] CHECK CONSTRAINT [CK\_Servico]

GO

Excerto SQL 12 - Criação da Tabela Servico

### 6.1.13. Criação da Tabela Telefone

USE [cliPet]

GO

-- Object: Table [dbo].[Telefone]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Telefone](

[telID] [dbo].[teIIDType] IDENTITY(1,1) NOT NULL,

[telNo] [dbo].[telNifType] NOT NULL,

CONSTRAINT [PK\_Telefone] PRIMARY KEY CLUSTERED

(

[telID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

Excerto SQL 13 - Criação da Tabela Telefone

### 6.1.14. Criação da Tabela TipoFuncionario

USE [cliPet]

GO

-- Object: Table [dbo].[TipoFuncionario]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[TipoFuncionario](

[tipoID] [dbo].[tipoFuncionarioIDType] IDENTITY(1,1) NOT NULL,

[tipoFuncionario] [varchar](25) NOT NULL,

CONSTRAINT [PK\_TipoFuncionario] PRIMARY KEY CLUSTERED

(

[tipoID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

Excerto SQL 14 - Criação da Tabela TipoFuncionario

## 6.2 T-SQL

### 6.2.1 Triggers

Nesta BD foram criados dois triggers de forma a validar as restrições de negócio, referidas anteriormente no relatório, e os constraints de cada relação.

Os triggers criados para a base de dados foram:

- **DonoAnimal**

Este trigger refere-se à inserção de dados na Tabela AnimaCliente. Valida se o animal tem um só dono de cada vez, através da contagem dos animais com donos ativos na base de dados

- **UpdateAnimal**

Este trigger refere-se à inserção de dados na Tabela DetalhesMarcacao. Se o veterinário preencher o peso do animal nos detalhes de uma marcação, também atualiza o valor do peso na relação Animal, isto é, se o peso na relação DetalhesMarcacao não for nulo

USE [cliPet]

GO

-- Object: Trigger [dbo].[DonoAnimal]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TRIGGER [dbo].[DonoAnimal]

ON [dbo].[AnimalCliente]

INSTEAD OF INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE @animalID NUMERIC(6,0),

@estado NCHAR(10),

@nDonos INT

SELECT @animalID = i.animalID, @estado = i.estadoDono

FROM inserted i

SET @nDonos = (

SELECT COUNT(@animalID) AS animalDono

FROM AnimalCliente

WHERE animalID = @animalID AND estadoDono = 'Ativo'

)

IF(@estado = 'Ativo' AND @nDonos < 1)

BEGIN

INSERT INTO AnimalCliente

SELECT \* FROM inserted

END

ELSE IF (@estado = 'NaoAtivo')

BEGIN

INSERT INTO AnimalCliente

SELECT \* FROM inserted

END

ELSE

BEGIN

RAISERROR('O animal não pode ter mais do que um dono

ao mesmo tempo',1,1);

END

GO

ALTER TABLE [dbo].[AnimalCliente] ENABLE TRIGGER [DonoAnimal]

GO

Excerto SQL 15 - Criação do Trigger DonoAnimal

USE [cliPet]

GO

-- Object: Trigger [dbo].[updateAnimal]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TRIGGER [dbo].[updateAnimal]

ON [dbo].[DetalhesMarcacao]

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE @peso DECIMAL,

@animalID NUMERIC,

@marcID NUMERIC

SELECT @peso = i.peso, @marcID = marcacaoID

FROM inserted i

SELECT @animalID = animalID

FROM Marcacao

WHERE marcacaoID = @marcID

BEGIN TRY

IF( @peso IS NOT NULL)

BEGIN

UPDATE Animal

SET Animal.peso = @peso

WHERE animalID = @animalID

END

END TRY

BEGIN CATCH

RAISERROR('O peso inserido nos detalhes da marcação não foi atualizado no animal',1,1);

END CATCH

END

GO

ALTER TABLE [dbo].[DetalhesMarcacao] ENABLE TRIGGER [updateAnimal]

GO

Excerto SQL 16 - Criação do Trigger updateAnimal

### 6.2.2 Stored Procedures

Para cumprir as restrições de negócio foram criados dois procedimentos:

**HistoricoAnimal -** é responsável por apresentar o histórico das marcações de um animal, seja qualquer o estado da marcação (Confirmada, Cancelada, Terminada ou Remarcada). Isto é, possível através da junção de três tabelas: Marcacao, Servico e DetalhesMarcacao

**InserirFatura -** é responsável por validar se a data e hora de emissão da fatura são superiores ou iguais à hora de fim e data de uma marcação. Para além disso também valida se o estado da marcação está Terminada

**InsereMarcacao -** é responsável pela validação da integridade dos dados inseridos numa marcação. Logicamente, este procedimento pode ser dividido em várias sub tarefas como:

* Verificação se a hora de início e fim da marcação está dentro do horário de funcionamento do grupo das clínicas CliPet, isto se o tipo de serviço prestado na consulta não for urgente. Optou-se por colocar o mesmo horário de funcionamento em todas as clínicas do mesmo grupo para facilitar a sua validação no procedimento.
* Verificação se a clínica selecionada realiza exames, para além de consultas, através de uma flag na relação Clinica que sinaliza o mesmo e pela designação do tipo de serviço que se pode tomar dois valores: consulta ou exame.
* Cálculo automático da hora de fim de uma marcação com base na duração comum ou por defeito do serviço prestado pelo funcionário
* Evitar colisões de agendamento das marcações associadas ao mesmo funcionário, com base na hora de inicio e fim de cada uma. Isto foi feito com recurso a cursores, de modo a percorrer todos os tuplos relativos a uma marcação

O grupo teve alguma dificuldade em escolher a abordagem para as tarefas deste procedimento. A abordagem inicial era dividir em vários triggers, contudo concluiu-se que não ideal devido ao uso do comando ROLLBACK que resultava uma mensagem de erro.

Após algum feedback do professor da UC os autores optaram por converter os triggers num só procedimento já que validam assuntos referentes à mesma relação.

USE [cliPet]

GO

-- Object: StoredProcedure [dbo].[HistoricoAnimal]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[HistoricoAnimal]

@Animal NUMERIC

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SELECT tb1.marcacaoID,

tb1.descricao AS Servico,dt.motivo,

dt.tratamento,

dt.obs,

dt.peso,

dt.temperatura,

dt.freqCardiaca

FROM (SELECT m.marcacaoID,s.descricao

FROM Marcacao m , Servico s

WHERE m.servicoID = s.servicoID and m.animalID = @Animal) AS tb1

LEFT JOIN DetalhesMarcacao dt

ON tb1.marcacaoID = dt.marcacaoID

END

GO

Excerto SQL 17 - Criação do procedimento HistoricoAnimal

USE [cliPet]

GO

-- Object: StoredProcedure [dbo].[InsereMarcacao]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[InsereMarcacao]

-- Add the parameters for the stored procedure here

@data DATE,

@horaInicio TIME,

@estado VARCHAR(15),

@descricao VARCHAR(100),

@marcacaoRef NUMERIC,

@urgencia BIT,

@clinicaID NUMERIC,

@animalID NUMERIC,

@servicoID NUMERIC,

@funcionarioID NUMERIC

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Verificação do horario válido de marcação

IF (@urgencia = 0 AND

(DATEPART(HOUR, @horaInicio) NOT BETWEEN 9 AND 12) AND

(DATEPART(HOUR, @horaInicio) NOT BETWEEN 14 AND 19))

BEGIN

RAISERROR(‘Marcação está fora das restrições do horario de funcionamento', 1, 1);

RETURN;

END

-- Verificar se a clínica faz exames

DECLARE @tipoServico VARCHAR(30),

@tipoServicoID NUMERIC,

@realiza BIT,

@duracao NUMERIC

SELECT @realiza = realizaExame

FROM Clinica

WHERE clinicaID = @clinicaID

SELECT @tipoServico = tipoServico, @duracao = duracao

FROM Servico

WHERE servicoID = @servicoID

IF(@realiza = 0 AND @tipoServico = 'Exame')

BEGIN;

RAISERROR( 'Esta Clinica não faz exames', 1, 1);

return;

END;

-- Atribuir hora do fim

DECLARE @horaFim TIME

SELECT @horaFim = dbo.CalcHoraFim(@duracao,@horaInicio)

/\*Verificar se há colisão de marcações\*/

DECLARE cursorFunc CURSOR FOR

SELECT horaInicio, horaFim

FROM Marcacao

WHERE dataMarcacao = @data AND funcionarioID = @funcionarioID

FOR READ ONLY

OPEN cursorFunc

DECLARE @horaInicioOutro TIME,@horaFimOutro TIME

FETCH cursorFunc INTO @horaInicioOutro, @horaFimOutro

WHILE (@@FETCH\_STATUS = 0)

BEGIN

IF((@horaInicio BETWEEN @horaInicioOutro AND @horaFimOutro) OR (@horaFim BETWEEN @horaInicioOutro AND @horaFimOutro))

BEGIN

RAISERROR( 'Marcação está a colidir com outra marcacao', 1, 1);

CLOSE cursorFunc

DEALLOCATE cursorFunc

return;

END

FETCH NEXT FROM cursorFunc INTO @horaInicioOutro , @horaFimOutro

END

CLOSE cursorFunc

DEALLOCATE cursorFunc

BEGIN TRY

INSERT INTO Marcacao

VALUES( @data, @horaInicio, @horaFim, @estado, @descricao, @marcacaoRef,

@urgencia, @clinicaID, @animalID, @servicoID, @funcionarioID )

END TRY

BEGIN CATCH

RAISERROR( 'Ocorreu um erro ao inserir marcação', 1, 1);

return;

END CATCH

END

GO

Excerto SQL 18 - Criação do procedimento InsereMarcacao

USE [cliPet]

GO

-- Object: StoredProcedure [dbo].[InserirFatura]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[InserirFatura]

-- Add the parameters for the stored procedure here

@modoPagamento NUMERIC,

@dataEmissao DATE,

@horaEmissao TIME,

@marcacaoID NUMERIC

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @horaFim TIME,

@data DATE,

@valor SMALLMONEY,

@estado VARCHAR(15)

-- Insert statements for procedure here

SET @horaFim = (SELECT horaFim FROM Marcacao WHERE marcacaoID = @marcacaoID)

SET @estado = (SELECT estado FROM Marcacao WHERE marcacaoID = @marcacaoID)

SET @data = (SELECT dataMarcacao FROM Marcacao WHERE marcacaoID = @marcacaoID)

SET @valor = (SELECT SUM(s.precoBase) AS Total FROM Marcacao m, Servico s WHERE m.servicoID = s.servicoID AND m.estado = 'Terminada' AND m.marcacaoID = @marcacaoID)

IF(@estado <> 'Terminada')

BEGIN

RAISERROR('O estado da marcação tem de estar terminado', 1, 1);

RETURN;

END

IF (DATEPART(HOUR, @horaEmissao) < DATEPART(HOUR, @horaFim) OR DATEPART(DAYOFYEAR, @dataEmissao) < DATEPART(DAYOFYEAR, @data))

BEGIN

RAISERROR('A data e hora de emissão da fatura tem ser maior ou igual à data e hora de fim da marcação', 1, 1);

RETURN;

END

BEGIN TRY

INSERT INTO Fatura

VALUES( @modoPagamento, @dataEmissao, @horaEmissao, @valor, @marcacaoID)

END TRY

BEGIN CATCH

RAISERROR( 'Ocorreu um erro a emitir a fatura', 1, 1);

return;

END CATCH

END

GO

Excerto SQL 19 - Criação do Procedimento InserirFatura

### 6.2.3 Scalar-Valued Functions

As funções forma utilizadas para implementar cálculos simples. Com base nas restrições da BD só foi necessário implementar duas funções:

**TotalFaturadoMesAno -** retorna uma representação em tabela do total faturado num mês de um determinado ano

**CalcHoraFim -** retorna o cálculo da hora de fim de uma marcação, com base na duração pré-definida do serviço associado. Esta função é auxiliar ao procedimento InsereMarcaao

USE [cliPet]

GO

-- Object: UserDefinedFunction [dbo].[TotalFaturadoMesAno]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE FUNCTION [dbo].[TotalFaturadoMesAno]

(

-- Add the parameters for the function here

@Ano INT,

@Mes INT

)

RETURNS decimal

AS

BEGIN

-- Declare the return variable here

DECLARE @Result decimal

SELECT @Result = SUM(s.precoBase)

FROM Marcacao m,Servico s

WHERE m.servicoID = s.servicoID AND

DATEPART(MONTH,m.dataMarcacao) = @Mes AND

DATEPART(YEAR,m.dataMarcacao) = @Ano

-- Return the result of the function

RETURN @Result

END

GO

Excerto SQL 20 - Criação da função TotalFaturadoMesAno

USE [cliPet]

GO

-- Object: UserDefinedFunction [dbo].[CalcHoraFim]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE FUNCTION [dbo].[CalcHoraFim]

(

-- Add the parameters for the function here

@duracao NUMERIC,

@horaInicio TIME

)

RETURNS TIME

AS

BEGIN

DECLARE @horaFim TIME

SET @horaFim = DATEADD(MINUTE, @duracao, @horaInicio)

RETURN @horaFim

END

GO

Excerto SQL 21 - Criação da função CalcHoraFim

## 6.3 Views/Vistas

As vistas SQL foram utilizadas como auxílio para as consultas da BD. No total foram criadas 7 vistas:

**cliPlus20PercentCancel -** permite obter os clientes com mais de 20% de consultas remarcadas/canceladas, através da junção entre as Tabelas: Marcacao, Cliente, AnimalCliente e Animal e da vista auxiliar TotalMarcacoesCliente

**DonosAtuais -** permite saber o dono atual de um animal, através do estado do seu dono da relação AnimalCliente. Isto é importante porque o animal pode ter múltiplos donos ao longo do tempo, mas só um dono ao mesmo tempo

**maxConsultas -** imprime uma lista dos top 10 veterinários com maior número de consultas, que representa uma marcação onde o serviço prestado é o do tipo consulta

**PercExamRefConsulta -** permite obter o valor da percentagem de exames realizados que têm como origem uma consulta realizada. Isto é feito através da referência recursiva presente numa marcação, que permite saber as marcações associadas a outras

**RacaCMaisFaturacao -** permite descobrir a raça do animal com mais marcações faturadas. Isto foi determinado através do estado da marcação que era obrigatório ser Terminada para poder ser incluído numa fatura

**TotalMarcacoesCliente -** é uma vista auxiliar que permite obter todas as marcações associadas a cada cliente. O propósito da criação desta vista foi facilitar a leitura da consulta efetuda pela vista ***cliPlus20PercentCancel***

**ValFaturadoMes -** permite calcular o valor total faturado num mês de um determinado ano, através da junção das Tabelas: Marcacao e Servico. O preço da marcação é equivalente ao preço base definido em cada serviço, contudo a fatura pode ter várias marcações, logo o valor na fatura será o somatório das marcações associadas

USE [cliPet]

GO

-- Object: View [dbo].[cliPlus20PercentCancel]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[cliPlus20PercentCancel]

AS SELECT tb1.clienteID, tb1.primeiroNome, tb1.ultimoNome, tb1.nCanceladas, tb2.NMarcacoes

FROM (SELECT clienteID, primeiroNome, ultimoNome,CAST(COUNT(clienteID)AS decimal) AS nCanceladas

FROM PercentagemConsultas

WHERE estado = 'Cancelada' OR estado = 'Remarcada'

GROUP BY clienteID, primeiroNome, ultimoNome) as tb1,

(SELECT clienteID, COUNT(clienteID) AS Nmarcacoes FROM PercentagemConsultas

GROUP BY clienteID) as tb2

WHERE tb1.clienteID = tb2.clienteID AND ((nCanceladas / NMarcacoes)\*100) > 20

GO

Excerto SQL 22 - Consultar clientes com mais de 20% de consultas remarcadas/canceladas

USE [cliPet]

GO

-- Object: View [dbo].[DonosAtuais]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[DonosAtuais]

AS

SELECT animalID, clienteID, dataDeAdocao, estadoDono

FROM AnimalCliente

WHERE estadoDono = 'Ativo'

GO

Excerto SQL 23 - Consultar dono atual do animal

USE [cliPet]

GO

-- Object: View [dbo].[max\_Consultas]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[max\_Consultas]

AS SELECT TOP(10) f.funcionarioID, f.primeiroNome, f.ultimoNome, tf.tipoFuncionario, COUNT(f.funcionarioID) AS NConsultas

FROM Funcionario f, TipoFuncionario tf, Marcacao m, Servico s

WHERE tf.tipoID = f.tipoID AND

f.funcionarioID = m.funcionarioID AND

m.servicoID = s.servicoID AND

tf.tipoFuncionario = 'Veterinário' AND

s.tipoServico = 'Consulta'

GROUP BY f.funcionarioID, f.primeiroNome, f.ultimoNome, tf.tipoFuncionario

ORDER BY NConsultas DESC

GO

Excerto SQL 24 - Veterinário com mais consultas

USE [cliPet]

GO

-- Object: View [dbo].[PercExamRefConsulta]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[PercExamRefConsulta]

AS

SELECT tb1.numExamRefConsulta/numExams \*100 AS PercExamRefConsulta

FROM (

SELECT COUNT( m.marcacaoID) As numExamRefConsulta

FROM Marcacao m, Servico s

WHERE m.servicoID = s.servicoID AND

s.tipoServico = 'Exame' AND

marcacaoRef IN (SELECT m.marcacaoID

FROM Marcacao m, Servico s

WHERE m.servicoID = s.servicoID AND

s.tipoServico = 'Consulta')

) AS tb1,

(

SELECT CAST(COUNT( m.marcacaoID) AS DECIMAL) As numExams

FROM Marcacao m, Servico s

WHERE m.servicoID = s.servicoID AND

s.tipoServico = 'Exame'

) AS tb2

GO

Excerto SQL 25 - Consulta da percentagem de exames realizados que têm como origem uma consulta realizada

USE [cliPet]

GO

-- Object: View [dbo].[RacaCMaisFaturacao]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[RacaCMaisFaturacao]

AS

SELECT TOP (1) r.descricao AS Raca, e.descricao AS Especie, SUM(s.precoBase) AS SomaPrecoBase

FROM Raca r ,Especie e, Marcacao m, Servico s, Animal a

WHERE e.especieID = r.especieID AND

a.racaID = r.racaID AND

m.animalID = a.animalID AND

m.servicoID = s.servicoID AND

m.estado ='Terminada'

GROUP BY r.descricao, e.descricao

ORDER BY SomaPrecoBase DESC

GO

Excerto SQL 26 - Consultar o tipo de animal e respetiva raça, que gera maior volume de faturação

USE [cliPet]

GO

-- Object: View [dbo].[TotalMarcacoesCliente]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[TotalMarcacoesCliente]

AS SELECT c.clienteID, c.primeiroNome, c.ultimoNome, m.marcacaoID, m.estado

FROM Cliente c, Marcacao m, AnimalCliente ac, Animal a

WHERE c.clienteID = ac.clienteID AND

a.animalID = ac.animalID AND

m.animalID = a.animalID AND

ac.estadoDono = 'Ativo'

GO

Excerto SQL 27 - Consultar total de marcações por cliente

USE [cliPet]

GO

-- Object: View [dbo].[ValFaturadoMes]

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[ValFaturadoMes]

AS

SELECT SUM(s.precoBase) AS Total, MONTH(m.dataMarcacao) AS Mes, YEAR(m.dataMarcacao) AS Ano

FROM Marcacao m,Servico s

WHERE m.servicoID = s.servicoID AND m.estado = 'Terminada'

GROUP BY MONTH(m.dataMarcacao), YEAR(m.dataMarcacao)

GO

Excerto SQL 28 - Consultar valor faturado em um mês de um determinado ano

# 7. Conclusões e Trabalho Futuro

A criação e manutenção de uma base de dados é algo demorado e complexo. Requer concentração e cometimento da parte de toda a equipa de desenvolvimento de forma serem alcançados os melhores resultados.

Durante a realização deste projeto surgiram várias dúvidas que incentivaram a procura por respostas, vários tentativas/erros que permitiram aprender e vários momentos de debate que permitiram aos autores colocar em prática e consolidar toda a matéria lecionada na UC de BD.

O resultado esperado foi obtido, a base de dados para o grupo “Clivet” foi efetuada com sucesso com a ressalva de que há espaço para o melhoramento da mesma. Todas as decisões e abordagens tomadas pelos autores tiveram a finalidade de obter

No futuro, os autores esperam que, com mais experiências e conhecimento, melhorar esta base de dados e/ou outras de forma a apresentar sempre o melhor resultado possível.

# Bibliografia

Connolly, T., & Begg, C. (s.d.). *Database Systems A Pratical Approach to Design, Implementation and Management.* Sixth Edition.

# Referências WWW

[01] [**https://app.diagrams.net**](https://app.diagrams.net)

Página utilizada para a elaboração de diagramas para a metodologia de base de dados

[02] [**https://docs.microsoft.com/en-us/sql**](https://docs.microsoft.com/en-us/sql)

Página com documentação oficial da ultima versão do SQL Server

[03] [**https://pt.stackoverflow.com/**](https://pt.stackoverflow.com/)

Site de perguntas e respostas sobre programação

[04] [**https://app.moqups.com/**](https://app.moqups.com/)

Ferramenta para a elaboração de mockups para a base de dados

# Lista de Siglas e Acrónimos

**BD** Base de Dados

**DW** Data Warehouse

**OLTP** *On-Line Analytical Processing*

**SGBD** Sistema de Gestão de Base de Dados

**ER** Entidade-Relação

**SQL** Structured Query Language

**PK** Primary Key

**FK** Foreign Key

**NIF** Número de Identificação Fiscal

**CC** Nº de cartão de cidadão

# Anexos

ANEXO I

# ANEXO I

1. Diagramas\_ER.draw.io
2. cliPet.bak
3. Mockup01\_marcacao.png
4. Mockup02\_cliente.png
5. Mockup03\_animal.png
6. Mockup04\_Funcionario.png
7. Mockup05\_DetalhesMarcacao.png
8. Mockup06\_registo\_animal.png