



ESCOLA  
SUPERIOR  
DE TECNOLOGIA  
E GESTÃO

## **Unidade Curricular de Base de Dados**

Ano Letivo de 2020/2021

## **SGBD CliPet - Clínicas**

**Jaques Resende 8190214**

**Rui Soares 8150289**

**Luís Oliveira 8190370**

junho, 2021

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 registre 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
Resumo .....	iv
Índice.....	v
Índice de Figuras .....	vii
Índice de Tabelas.....	viii
Índice Excertos de SQL.....	ix
1. Introdução .....	1
1.1 Contextualização .....	1
1.2 Apresentação do Caso de Estudo .....	1
1.3 Motivação e Objetivos .....	1
1.4 Estrutura do Relatório .....	1
2 Requisitos Gerais e Use Cases .....	3
2.1 Requisitos Gerais .....	3
3 Desenho Conceptual.....	5
3.1 Identificação das entidades.....	6
3.2 Relações entre Entidades .....	7
3.3 Multiplicidade.....	8
3.2 Atributos para as Entidades .....	9
3.5 Documentação de atributos .....	10
3.6. Atribuição de chaves primarias .....	14
4 Desenho Lógico .....	16
4.1 Derivação de relações para o modelo de dados lógico.....	17
4.1.1 Entidades fortes.....	17
4.1.2 Entidades fracas .....	17
4.1.3 Relação de um para muitos (1 : *) .....	18
4.1.4 Relação de um para um (1 : 1).....	20
4.1.5 Relação de muitos para muitos (* : *) .....	21
4.1.6 Multi-valued attributes.....	22
4.2 Normalização .....	23
4.2.1 Primeira Forma Normal (1FN).....	23
4.2.2 Segunda Forma Normal (2FN) .....	23
4.2.3 Terceira Forma Normal (3FN) .....	23

4.2.4 Mockups.....	24
5 Restrições de integridade – Regras de negócio .....	33
6 Desenho Físico .....	36
6.1 Criação da Tabelas, relacionamentos e restrições aos atributos .....	38
6.1.1 Criação da Tabela Animal.....	38
6.1.2 Criação da Tabela AnimalCliente .....	40
6.1.3 Criação da Tabela Cliente.....	41
6.1.4 Criação da Tabela Clínica.....	42
6.1.5 Criação da Tabela DetalhesMarcacao.....	43
6.1.6 Criação da Tabela Especie .....	44
6.1.7 Criação da Tabela Fatura .....	45
6.1.8 Criação da Tabela Funcionario .....	46
6.1.9 Criação da Tabela Marcação .....	47
6.1.10. Criação da Tabela ModoPagamento .....	49
6.1.11. Criação da Tabela Raca .....	49
6.1.12. Criação da Tabela Servico .....	50
6.1.13. Criação da Tabela Telefone .....	50
6.1.14. Criação da Tabela TipoFuncionario.....	51
6.2 T-SQL .....	52
6.2.1 Triggers.....	52
6.2.2 Stored Procedures.....	54
6.2.3 Scalar-Valued Functions .....	58
6.3 Views .....	60
7. Conclusões e Trabalho Futuro .....	64
Bibliografia.....	65
Referências WWW.....	66
Lista de Siglas e Acrónimos .....	67
Anexos .....	68
ANEXO I.....	69

# Índice de Figuras

Figura 1 - Diagrama E-R sem atributos .....	7
Figura 2 - Diagrama concetual só com chaves primárias .....	14
Figura 3 - Diagrama Conceptual ER .....	15
Figura 4 - Mockup Marcação .....	24
Figura 5 - Mockup Cliente .....	26
Figura 6 - Mockup Histórico Animal .....	27
Figura 7 - Mockup Detalhes de Marcação .....	29
Figura 8 - Mockup Registo Funcionário .....	31
Figura 9 Diagrama ER do SQL Management Studio .....	37

# Índice de Tabelas

Tabela 1 - Identificação das entidades .....	6
Tabela 2 - Multiplicidade.....	8
Tabela 3 - Atributos para as Entidades .....	9
Tabela 4 - <b>Relação</b> - Funcionario.....	10
Tabela 5 - <b>Relação</b> - Clinica.....	10
Tabela 6 - <b>Relação</b> - Servico .....	11
Tabela 7 - <b>Relação</b> - Animal .....	11
Tabela 8 - <b>Relação</b> - Marcacao.....	12
Tabela 9 - <b>Relação</b> - Cliente .....	12
Tabela 10 - <b>Relação</b> - Fatura .....	13
Tabela 11 - <b>Relação</b> - DetalhesMarcacao.....	13
Tabela 12 - Integridade Referencial .....	35
Tabela 13 - Tipos de dados.....	36



# Índice Excertos de SQL

Excerto SQL 1 - Criação Tabela Animal.....	39
Excerto SQL 2 - Criação da Tabela AnimalCliente.....	40
Excerto SQL 3 - Criação da Tabela Cliente.....	41
Excerto SQL 4 - Criação da Tabela Clínica.....	42
Excerto SQL 5 - Criação da Tabela DetalhesMarcacao .....	43
Excerto SQL 6 - Criação da Tabela Especie .....	44
Excerto SQL 7 - Criação da Tabela Fatura .....	45
Excerto SQL 8 - Criação da Tabela Funcionario .....	46
Excerto SQL 9 - Criação da Tabela Marcação .....	48
Excerto SQL 10 - Criação da Tabela ModoPagamento.....	49
Excerto SQL 11 - Criação da Tabela Raca .....	49
Excerto SQL 12 - Criação da Tabela Servico.....	50
Excerto SQL 13 - Criação da Tabela Telefone.....	50
Excerto SQL 14 - Criação da Tabela TipoFuncionario .....	51
Excerto SQL 15 - Criação do Trigger donoAnimal.....	53
Excerto SQL 16 - Criação do Trigger updateAnimal.....	53
Excerto SQL 17 - Criação do procedimento HistoricoAnimal .....	55
Excerto SQL 18 - Criação do procedimento InsereMarcacao .....	57
Excerto SQL 19 - Criação do Procedimento InserirFatura .....	57
Excerto SQL 20 - Criação da função TotalFaturadoMesAno.....	58
Excerto SQL 21 - Criação da função CalcHoraFim .....	59
Excerto SQL 22 - Consultar clientes com mais de 20% de consultas remarcadas/canceladas .....	60
Excerto SQL 23 - Consultar dono atual do animal .....	61
Excerto SQL 24 - Veterinário com mais consultas .....	61
Excerto SQL 25 - Consulta da percentagem de exames realizados que têm como origem uma consulta realizada .....	62
Excerto SQL 26 - Consultar o tipo de animal e respetiva raça, que gera maior volume de faturação .....	62
Excerto SQL 27 - Consultar total de marcações por cliente .....	63
Excerto SQL 28 - Consultar valor faturado em um mês de um determinado ano .....	63

# **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. Após 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

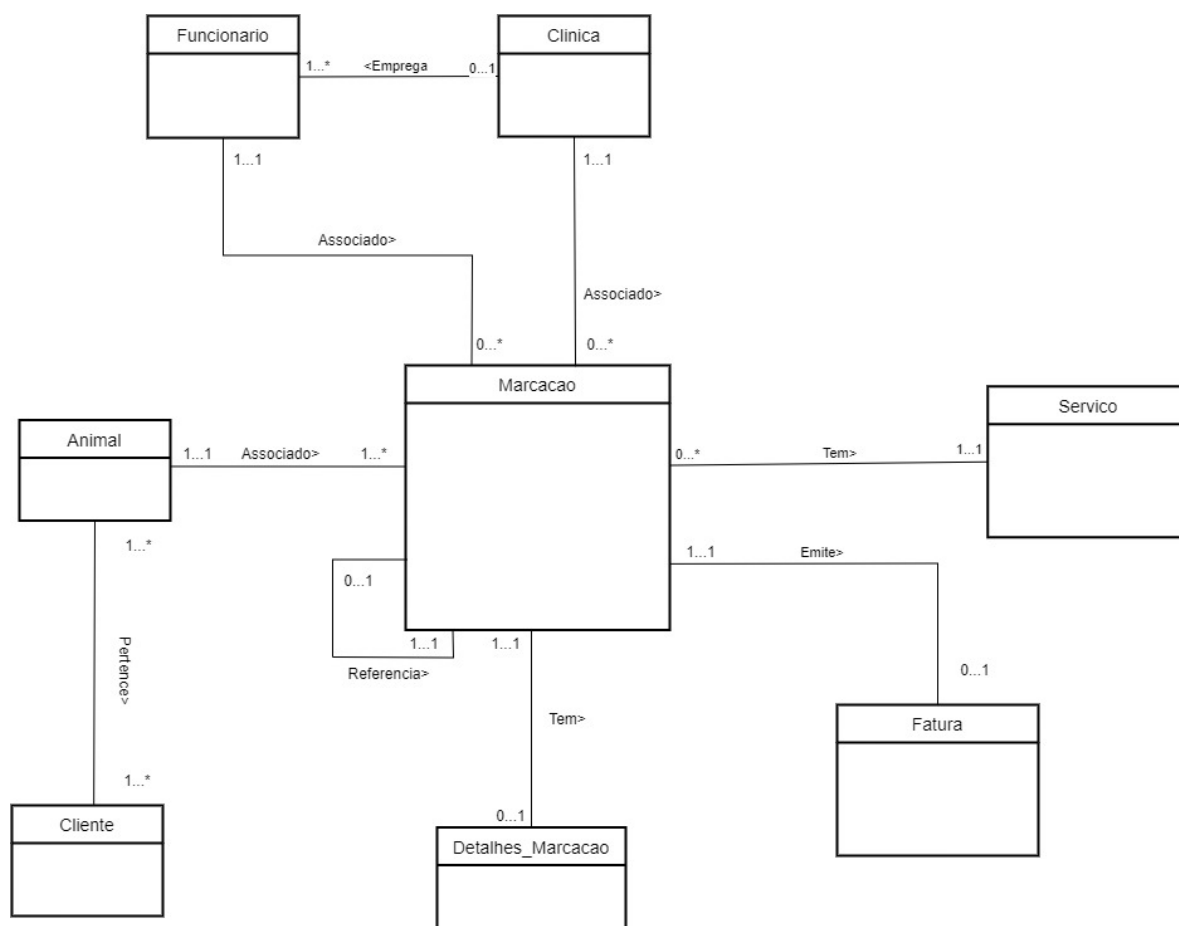


Figura 1 - Diagrama E-R sem atributos



### 3.3 Multiplicidade

*Tabela 2 - Multiplicidade*

Entidade	Multiplicidade	Relação	Multiplicidade	Entidade
<b>Clinica</b>	0...1	Emprega	1...*	<b>Funcionario</b>
<b>Funcionario</b>	1...1	Associado	0...*	<b>Marcacao</b>
<b>Animal</b>	1...1	Associado	1...*	<b>Marcacao</b>
<b>Animal</b>	1...*	Pertence	1...*	<b>Cliente</b>
<b>Marcacao</b>	1...0	Pertence	1...1	<b>Servico</b>
<b>Marcacao</b>	1...1	Tem	0...1	<b>DetalhesMarcacao</b>
<b>Fatura</b>	0...*	Refere	1...1	<b>Marcacao</b>
<b>Marcacao</b>	0...*	Refere	1...1	<b>Marcacao</b>
<b>Clinica</b>	1...1	Associado	0...*	<b>Marcacao</b>

## 3.2 Atributos para as Entidades

Tabela 3 - Atributos para as Entidades

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

### 3.5 Documentação de atributos

Tabela 4 - **Relação** - Funcionario

Atributo	Descrição	Tipo de informação	Nulls	Multi-Valued
<b>funcionarioID</b>	Identificador único de cada funcionário	Até 3 números inteiros	Não	Não
<b>primeiroNome</b>	Sobrenome do funcionário	15 carateres variáveis	Não	Não
<b>ultimoNome</b>	Apelido do funcionário	15 carateres variáveis	Não	Não
<b>sexo</b>	Género do funcionário	1 caracter	Não	Não
<b>dataDeNascimento</b>	Data de nascimento do funcionário	Tipo data	Não	Não
<b>tipoFuncionario</b>	Identificação da função do funcionário	25 carateres variáveis	Não	Não
<b>dataDeModificacao</b>	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
<b>clinicaID</b>	Identificador único da clínica	Até 4 números inteiros	Não	Não
<b>telNo</b>	Número de telemóvel da clínica	9 dígitos numéricos	Não	Sim
<b>email</b>	Email da clínica	256 carateres variáveis	Não	Não
<b>rua</b>	Rua da Clínica	50 carateres variáveis	Não	Não
<b>cidade</b>	Cidade onde a clínica está localizada	50 carateres variáveis	Não	Não
<b>codPostal</b>	Código de postal da clínica	XXXX-XXX	Não	Não
<b>pais</b>	País onde se encontra a clínica	2 carateres	Não	Não
<b>realizaExame</b>	Confirma se a clínica faz ou não exames	Valor booleano	Não	Não

Tabela 6 - **Relação** - Serviço

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

Tabela 7 - **Relação** - Animal

Atributo	Descrição	Tipo de informação	Nulls	Multi-Valued
<b>animalID</b>	Identificação única de cada serviço	Até 6 números inteiros	Não	Não
<b>nome</b>	Nome de animal	15 carateres variáveis	Não	Não
<b>dataNascimento</b>	Data de nascimento do animal	Tipo data	Não	Não
<b>especie</b>	Identificação do tipo do serviço	15 carateres variáveis	Não	Não
<b>comportamento</b>	Registo do tipo de comportamento do animal	15 carateres variáveis	Não	Não
<b>raca</b>	Designação da raça da raça do animal	50 carateres variáveis	Não	Não
<b>microchip</b>	Número de Identificação do chip do animal	15 dígitos numéricos	Sim	Não
<b>peso</b>	Peso do animal	decimal (5,2)	Não	Não
<b>altura</b>	Altura do animal	decimal (5,2)	Não	Não
<b>sexo</b>	Género do animal	1 caracter	Não	Não
<b>estado</b>	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
<b>marcacaoID</b>	Identificação única do serviço	Até 9 números inteiros	Não	Não
<b>data</b>	Data da marcação	Data	Não	Não
<b>horaInicio</b>	Hora do início da marcação	Time	Não	Não
<b>horaFim</b>	Hora do fim da marcação	Time	Sim	Não
<b>estado</b>	Estado da marcação (cancelada, confirmada etc)	15 Carateres variáveis	Não	Não
<b>descricao</b>	Descrição da marcação	100 carateres variáveis	Não	Não
<b>marcacaoRef</b>	Referencia de marcação anterior	Unsigned int	Sim	Não
<b>urgencia</b>	A marcação é urgência ou não	Valor booleano	Não	Não

Tabela 9 - **Relação** - Cliente

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

Tabela 10 - **Relação** - Fatura

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

Tabela 11 - **Relação** - DetalhesMarcacao

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

### 3.6. Atribuição de chaves primarias

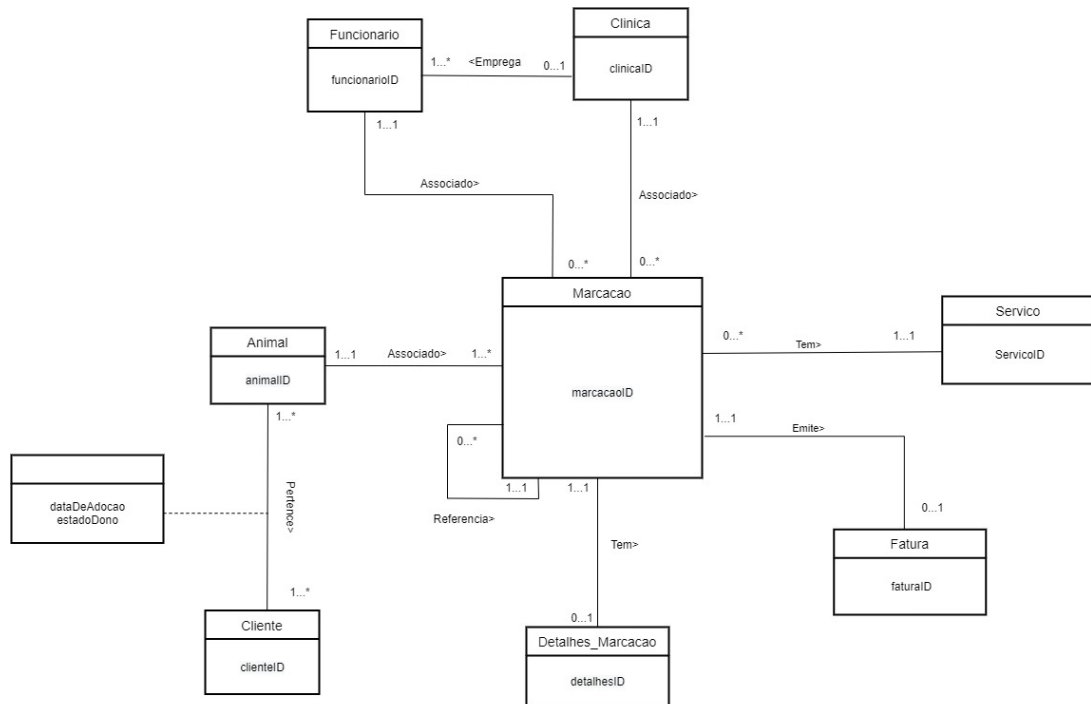


Figura 2 - Diagrama conceitual só com chaves primárias

### Documentação de chaves primarias (PK)

Entidade	Chave Primaria	Chaves Candidatas
<b>Funcionario</b>	funcionariID	—
<b>Clinica</b>	clinicaID	Email
<b>Servico</b>	servicoID	—
<b>Animal</b>	animalID	—
<b>Marcacao</b>	marcacaoID	—
<b>Cliente</b>	clienteID	Nif   cc
<b>Fatura</b>	faturaID	—
<b>DetalhesMarcacao</b>	detalhesID	—

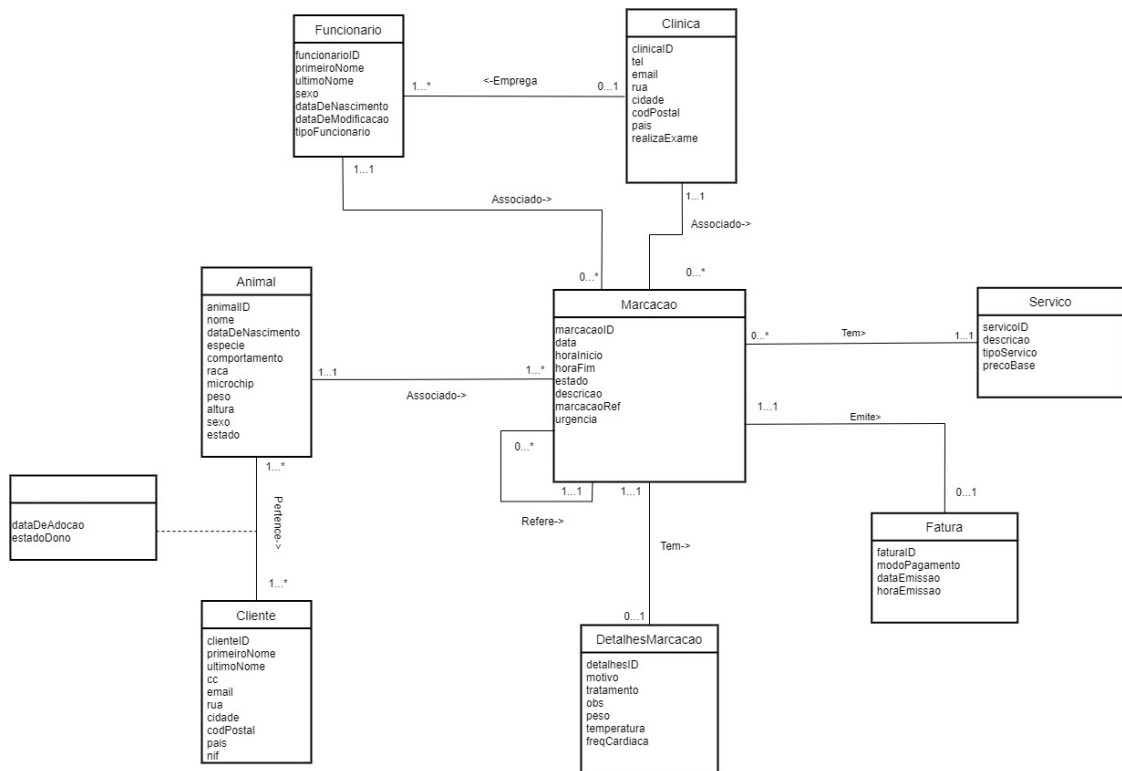


Figura 3 - Diagrama Conceptual ER



## 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:** funcionariolD

##### *Entidade filho*

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

**Chave primária:** funcionariolD

**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** funcionariolD

##### *Entidade filho*

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

**Chave primária:** marcacaoID

**Chave estrangeira** 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 estrangeira** 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 estrangeira** clinicaID **refere** Clinica (clinicaID)  
animalID **refere** Animal ( animalID)  
servicoID **refere** Servico(servicoID)

Põe marcacaoID em 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 estrangeira** clinicaID **refere** Clinica (clinicaID)  
animalID **refere** Animal ( animalID)  
servicoID **refere** Servico(servicoID)

#### ***Entidade filho***

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

**Chave primaria** marcacaoID

**Chave estrangeira** clinicalID **refere** Clinica (clinicalID)

animalID **refere** Animal ( animalID)

servicoID **refere** Servico(servicoID)

marcacaoRef **refere** Marcacao (marcacaoID)

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

---

#### **Funcionario 1 : \* Marcacao**

#### ***Entidade pai***

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

**Chave primaria** funcionarioID

**Chave estrangeira** clinicalID **refere** Clinica (clinicalID)

#### ***Entidade filho***

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

**Chave primaria** marcacaoID

**Chave estrangeira** clinicalID **refere** Clinica (clinicalID)

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, clinicalID, servicoID, funcionarioID }

**Chave primaria:** marcacaoID

**Chave estrangeira** 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 estrangeira** 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** clientID

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

**Chave primaria** clientID , animalID

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

#### 4.1.6 Multi-valued attributes

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

Chave primaria clinicalD

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

**Telephone** { telNo, clinicalID}

**Chave primaria** telNo

Chave estrangeira clinicalD

## 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.<sup>a</sup> 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.<sup>a</sup> 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.<sup>a</sup> 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 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

### Marcação

ID

ID gerado automaticam

Animal

Nome do animal

Nome

Nome do Dono do Animal

June 20, 2021

Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Clinica

selecionar clinica

▼

Hora

selecionar

▼

▲

▼

Serviço	Marcar
Primeira Consulta	
Consulta Geral	
Tosquia	

Confirmar

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, horalnicio, 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, horalnicio, 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, descricao, 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

## 3ª Forma

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

A raça depende da espécie.

Serão criadas 2 novas entidades

**Raca** { racalID, descricao, especieID }

**Especie** { especieID, raca }

**Animal** { animalID, nome, dataDeNascimento, comportamento, racalID, 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



Ficha Animal

Histórico Médico

Tutor

☒ Ativo

Ultima Visita

01-01-2001

ID

nrº de ID do cliente

Primeiro Nome

Nome

Sexo

selecionar sexo

Ultimo Nome

Nome

CC

9xxxxxxxx

Data Nascimento

06/20/2021

NIF

9xxxxxxxx

email

xxxxxx@xxx.xxx

Tel

9xxxxxxxx

Rua

cidade

Guardar

Apagar

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

**Animal**

 **Ficha Animal** **Histórico Médico** **Tutor**

☒ Ativo

**Ultima Visita**  
01-01-2001

**Consulta**  
**Clinica:** Clinica 2 **Veterinário:** Dr. One  
Motivo: atropelamento  
Observações: Diagnostico x  
Tratamento: 1 comprimido de Y  
Peso: 10 kg Temperatura.: 36°C Freq. Cardiaca: Z bpm

**Exame: Radiografia**  
**Clinica:** Clinica 1 **Veterinário:** Dr. Two  
Motivo: atropelamento  
Observações: Fratura 2º falange

**Guardar** **Apagar**

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

telID -> 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

telID, 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{clínicaID, 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

### Detalhes de Marcação

06/21/2021

ID Marcação  
Placeholder

Tutor  
Nome do tutor (dono)

ID Animal  
Placeholder

Nome Animal  
Nome do animal

Espécie  
selecionar

Microchip  
Placeholder

Sexo  
selecionar sexo

Comportamento  
selecionar

Data Nascimento  
06/20/2021

ID Veterinário  
ID do veterinario

Peso  
Placeholder

Temp  
Placeholder

Freq. Card.  
Placeholder

Motivo:  
Type here

Motivo:  
Type here

Tratamento  
Type here

Finalizar

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

### Dependências 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

telID -> telNo

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

marcacaoID -> data, horaNicio, 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, horaNicio, 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

telID, 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

### Funcionário



ID	<input type="text" value="nrº de ID do funcionario"/>	Primeiro Nome	<input type="text" value="Nome"/>
Sexo	<input type="text" value="selecionar sexo"/>	Ultimo Nome	<input type="text" value="Nome"/>

Ultima alteração

01-01-2001	Data Nasc.	<input type="text" value="06/20/2021"/>	Função	<input type="text" value="selecionar função"/>
------------	------------	---	--------	--

GuardarApagar

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 deve 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 dígitos.
- 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
<b>Funcionario</b>	Marcacao	No Action	No Action
<b>Clinica</b>	Marcacao	No Action	No Action
<b>Animal</b>	Marcacao	No Action	No Action
<b>Servico</b>	Marcacao	No Action	No Action
<b>Marcacao</b>	DetalhesMarcacao	No Action	CASCADE
<b>Marcacao</b>	Fatura	No Action	CASCADE
<b>Clinica</b>	Funcionario	No Action	CASCADE
<b>TipoFuncionario</b>	Funcionario	No Action	CASCADE
<b>Animal</b>	AnimalCliente	No Action	CASCADE
<b>Cliente</b>	AnimalCliente	No Action	CASCADE
<b>ModoPagamento</b>	Fatura	No Action	CASCADE
<b>Telefone</b>	Clinica	No Action	CASCADE
<b>Especie</b>	Raca	No Action	CASCADE
<b>Raca</b>	Animal	No Action	CASCADE
<b>Marcacao</b>	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)

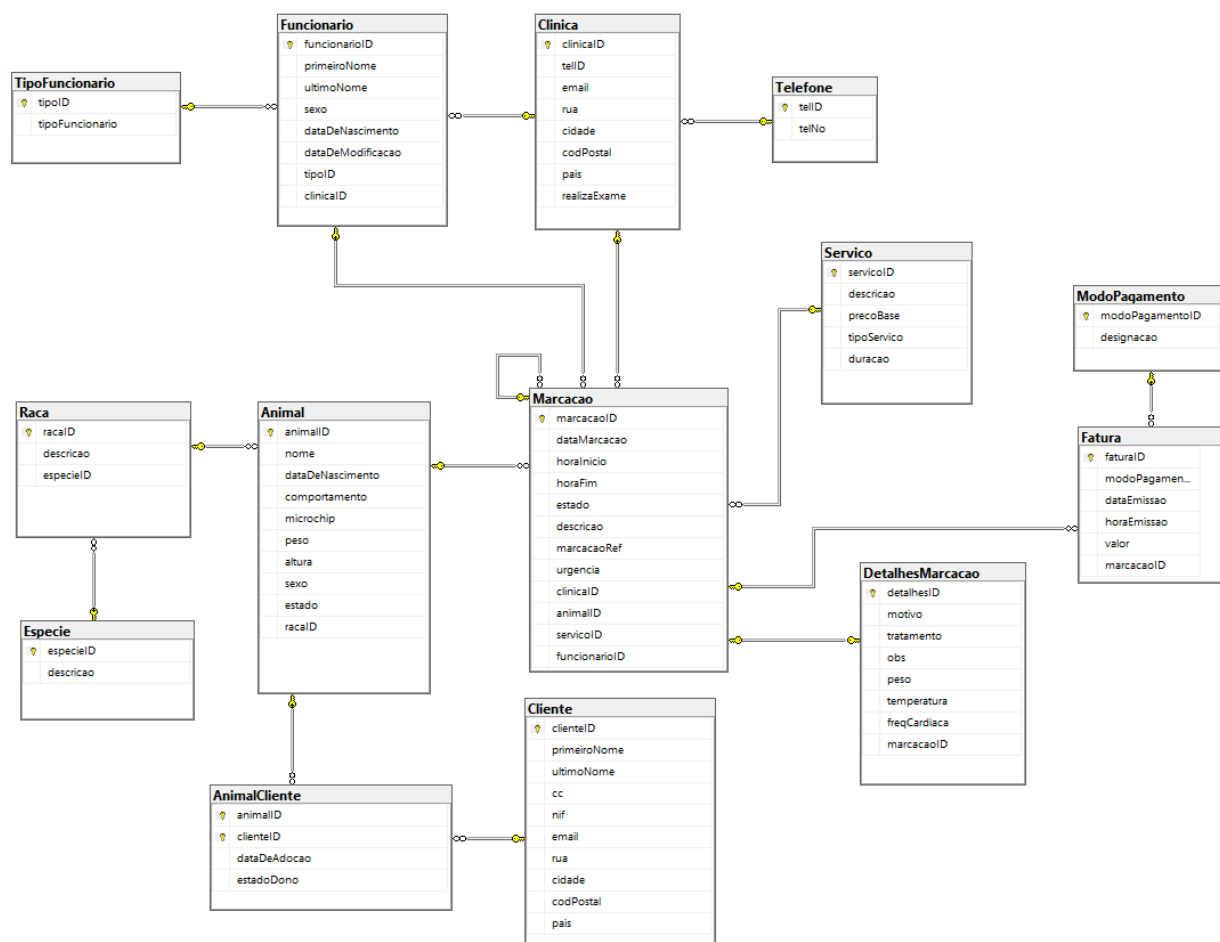


Figura 9 Diagrama ER do SQL Management Studio

## 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 - Criação 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

**InsererMarcacao** - 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].[InserereMarcacao]
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[InserereMarcacao]
    -- 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

```

```

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(
@valor, @marcacaoID)
    END TRY
    BEGIN CATCH
        RAISERROR( 'Ocorreu um erro a emitir a fatura', 1, 1);
        return;
    END CATCH
END
GO

```

## 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 efetuada 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 Espécie, 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, várias 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>  
Página utilizada para a elaboração de diagramas para a metodologia de base de dados
- [02]     <https://docs.microsoft.com/en-us/sql>  
Página com documentação oficial da ultima versão do SQL Server
- [03]     <https://pt.stackoverflow.com/>  
Site de perguntas e respostas sobre programação
- [04]     <https://app.mogups.com/>  
Ferramenta para a elaboração de mockups para a base de dados

## Lista de Siglas e Acrónimos

<b>BD</b>	Base de Dados
<b>DW</b>	Data Warehouse
<b>OLTP</b>	<i>On-Line Analytical Processing</i>
<b>SGBD</b>	Sistema de Gestão de Base de Dados
<b>ER</b>	Entidade-Relação
<b>SQL</b>	Structured Query Language
<b>PK</b>	Primary Key
<b>FK</b>	Foreign Key
<b>NIF</b>	Número de Identificação Fiscal
<b>CC</b>	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