INSTITUTO DE ENSINO SUPERIOR – ICEV

CURSO: Engenharia de Software

SISTEMA DE GESTÃO DE PETSHOP E CLÍNICA VETERINÁRIA

COMPONENTES:

Kaielly Vitória Sousa De Morais

Mateus Farias Dos Santos

Vinícius Azevedo Da Fonseca

Ygor Jivago Leal Félix

Teresina – PI

2025

SISTEMA DE GESTÃO DE PETSHOP E CLÍNICA VETERINÁRIA

Trabalho apresentado à disciplina de Banco de Dados, do curso de Engenharia de Software do Instituto de Ensino Superior - ICEV, como requisito parcial para obtenção da nota da segunda avaliação.

Professor: Hilson Barbosa Da Silva

Teresina - PI

2025

SUMÁRIO

1. INTRODUÇÃO

O presente trabalho consiste no desenvolvimento de um projeto completo de banco de dados voltado para o gerenciamento de informações em um sistema de Petshop e Clínica Veterinária. O projeto foi elaborado em equipe, seguindo as etapas fundamentais do ciclo de modelagem de dados, que incluem as fases conceitual, lógica e física.

Durante o processo, foram definidos os principais elementos do sistema, como entidades, atributos, relacionamentos e restrições, a partir de um cenário de minimundo que representa as necessidades de uma clínica veterinária integrada a um petshop. O trabalho busca demonstrar, de forma prática, a aplicação dos conceitos teóricos de modelagem de dados.

A proposta envolve desde a concepção inicial do modelo conceitual (MER) até a implementação física do banco de dados, com inserção de registros e execução de consultas SQL. Todo o desenvolvimento foi realizado com base nas boas práticas de projeto e nas orientações de autores reconhecidos na área, como Elmasri e Navathe (2017) e Date (2004), que destacam a importância de um modelo de dados bem estruturado para garantir consistência, desempenho e confiabilidade das informações.

1.1. OBJETIVO

O objetivo deste trabalho é desenvolver um projeto de banco de dados voltado para a gestão de um petshop e clínica veterinária, abordando todas as etapas de modelagem conceitual, lógica e física. A proposta é criar uma estrutura que permita organizar as informações de maneira clara e funcional, facilitando o controle de clientes, animais, atendimentos, serviços e pagamentos.

O projeto também tem como finalidade aplicar os conhecimentos adquiridos em sala de aula, simulando situações reais do cotidiano de uma clínica. Dessa forma, busca-se mostrar como um banco de dados bem planejado pode contribuir para a melhoria dos processos, reduzir falhas e tornar o gerenciamento mais rápido e eficiente.

2. FERRAMENTAS

Para o desenvolvimento do projeto foram utilizadas ferramentas que auxiliam em cada etapa da criação do banco de dados. O BrModelo 3.32 foi usado na fase conceitual, onde foi elaborado o Modelo Entidade-Relacionamento (MER), representando as tabelas, atributos e os relacionamentos do sistema.

Na sequência, o MySQL, Workbench foi utilizado para a parte lógica e física do projeto, permitindo a criação das tabelas, definição das chaves e execução dos comandos SQL. A ferramenta também possibilitou a geração do Diagrama de Entidade Relacional (DER), facilitando a visualização da estrutura final do banco de dados.

Por fim, o Google Docs foi utilizado para a documentação do trabalho, reunindo as informações e descrições de cada etapa desenvolvida.

3. MODELO CONCEITUAL

O modelo conceitual é a primeira etapa da criação de um banco de dados e tem como função representar, de forma clara, as informações mais importantes do sistema. Nessa fase, o objetivo é entender o contexto da empresa, o que precisa ser controlado e os dados se relacionam entre si. A partir das informações levantadas no minimundo, são identificados as entidades, seus atributos e os principais relacionamentos que formam a base da estrutura do banco.

No caso da Scooby Petshop, o modelo foi construído considerando as necessidades reais da empresa, levando em conta o funcionamento interno, os processos de atendimento, o controle de produtos, pagamentos e demais atividades que fazem parte da rotina do negócio. Essa etapa foi essencial para definir como o sistema armazenará e organizará as informações de forma eficiente e coerente com a realidade observada.

3.1. DEFINIÇÃO DO MINIMUNDO

Segundo Elmasri e Navathe (2017), o minimundo representa uma parte do mundo real que será modelada em um banco de dados, ou seja, o conjunto de informações e processos relevantes que o sistema precisa armazenar e gerenciar.

Essa etapa é fundamental para compreender o contexto do problema e traduzir as necessidades do usuário em estruturas lógicas e consistentes.

Com base nesse conceito, o presente projeto foi desenvolvido a partir do cenário da Scooby Petshop, uma clínica veterinária e loja de produtos para animais localizada em Teresina-PI. A empresa contratou nossa equipe para desenvolver um sistema de gerenciamento que otimize o controle de suas atividades. A Scooby Petshop foi fundada por um casal apaixonado por animais, Marcos e Daniela, que se mudou do interior do Maranhão para a capital piauiense em busca de novas oportunidades. Movidos pelo amor aos animais e pela vontade de oferecer um atendimento mais humanizado e especializado, decidiram abrir o petshop após anos de experiência com resgates e cuidados voluntários. A filha do casal, Mali, cresceu nesse ambiente e hoje é responsável pelas redes sociais da empresa, ajudando na divulgação de campanhas de adoção e conscientização sobre os cuidados com os pets.

Com o aumento da clientela e a expansão dos serviços, os proprietários perceberam a necessidade de adotar uma solução tecnológica que centralizasse os processos e reduzisse falhas no controle manual de dados. Assim, foi solicitado o desenvolvimento de um sistema de gerenciamento personalizado, voltado à realidade da empresa, com foco no controle de atendimentos, produtos, pagamentos e estoque.

O sistema tem como objetivo integrar todas as áreas da Scooby Petshop, oferecendo uma ferramenta que facilite o acompanhamento das operações, garanta a organização das informações e melhore a comunicação entre os setores. Além disso, busca automatizar tarefas repetitivas e gerar relatórios que auxiliem na tomada de decisões.

Para representar corretamente as informações do minimundo, foram definidas as seguintes regras de funcionamento, que orientaram a modelagem conceitual do banco de dados:

- Um cliente pode possuir um ou vários pets, mas cada pet pertence a apenas um cliente.
- Um cargo pode estar associado a vários funcionários, mas cada funcionário ocupa somente um cargo.
- Um funcionário pode realizar um ou vários atendimentos, mas cada atendimento é feito por apenas um funcionário.

- Um serviço pode ser executado em vários atendimentos, mas cada atendimento está vinculado a apenas um serviço.
- Um pet pode ter um ou vários atendimentos, mas cada atendimento é destinado a um único pet.
- Um atendimento pode gerar diferentes versões no histórico, mas cada registro histórico pertence a apenas um atendimento.
- Um pagamento pode possuir várias atualizações registradas em seu histórico, mas cada histórico está vinculado a um único pagamento.
- Um estoque pode conter diversos produtos, mas cada produto pertence a apenas um estoque.
- Um produto pode ter várias movimentações registradas em seu histórico, mas cada movimentação está ligada a um único produto.
- Um pagamento pode incluir um ou vários produtos, e um mesmo produto pode aparecer em diferentes pagamentos.
- Um pagamento pode quitar um ou vários atendimentos, e um atendimento pode estar relacionado a diferentes pagamentos.

3.2. DICIONÁRIO DE DADOS

O dicionário de dados é uma parte essencial na construção de um banco de dados, pois descreve de forma organizada as informações que o sistema irá armazenar. Ele detalha as tabelas, seus campos, tipos de dados, tamanhos e chaves, servindo como guia para a criação e manutenção da estrutura física do banco.

Segundo Elmasri e Navathe (2017), o dicionário de dados permite documentar todos os elementos que compõem o banco, garantindo padronização e clareza no processo de modelagem. Essa documentação contribui para a integridade das informações e facilita futuras atualizações no sistema.

No caso do sistema da Scooby Petshop, o dicionário de dados foi elaborado com base nas entidades definidas no modelo conceitual, representando as informações necessárias para o gerenciamento de clientes, pets, atendimentos, produtos, estoque, pagamentos e funcionários. Cada tabela foi construída com seus respectivos campos, tipos de dados e chaves primárias e estrangeiras, garantindo a integridade e o relacionamento entre os registros.

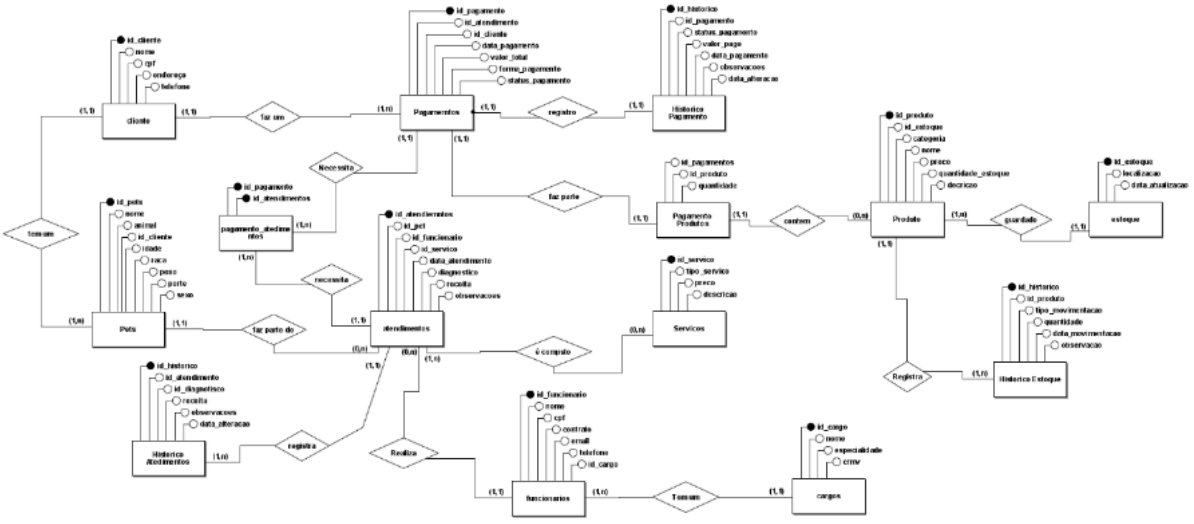
Abaixo está um exemplo da estrutura adotada:

Entidade Clientes			
Atributo	**Descrição**	**Domínio**	**Restrição do atributo**
id_cliente	Identificador único do cliente	INT	Chave Primária, Não nulo, Auto Incremento
nome	Nome completo do cliente	VARCHAR(100)	Não nulo
cpf	CPF do cliente	VARCHAR(14)	Não nulo, Valor único
endereco	Endereço completo do cliente	VARCHAR(150)	-
telefone	Número de telefone para contato	VARCHAR(25)	-

Tabela 002 – Dicionário de Dados - DD - Clientes

Figura 1: A Figura 1 apresenta a estrutura do dicionário de dados referente à entidade Cliente, que armazena informações pessoais e de contato dos clientes da Scooby Petshop. **FONTE**: elaborado pela equipe.

3.3. MODELO DE ENTIDADE-RELACIONAMENTO (MER)



4. MODELO LÓGICO

O modelo lógico do sistema Scooby Petshop foi criado com base no modelo conceitual, representando a estrutura do banco de dados de forma mais próxima da implementação. Nessa etapa, as informações do minimundo foram organizadas em tabelas, definindo os campos, tipos de dados e relacionamentos entre elas. O desenvolvimento foi feito no MySQL Workbench, aplicando as formas normais para evitar duplicação de dados e garantir a integridade das informações.

As tabelas principais são Clientes, Pets, Funcionários, Cargos, Serviços, Atendimentos, Pagamentos, Produtos e Estoque, além das tabelas de histórico e relacionamento. Todas estão ligadas por chaves primárias e estrangeiras, o que permite o controle e a conexão entre os dados. O diagrama criado no Workbench mostra essas ligações de forma clara e serviu como base para a criação do banco físico e para os testes de funcionamento do sistema.

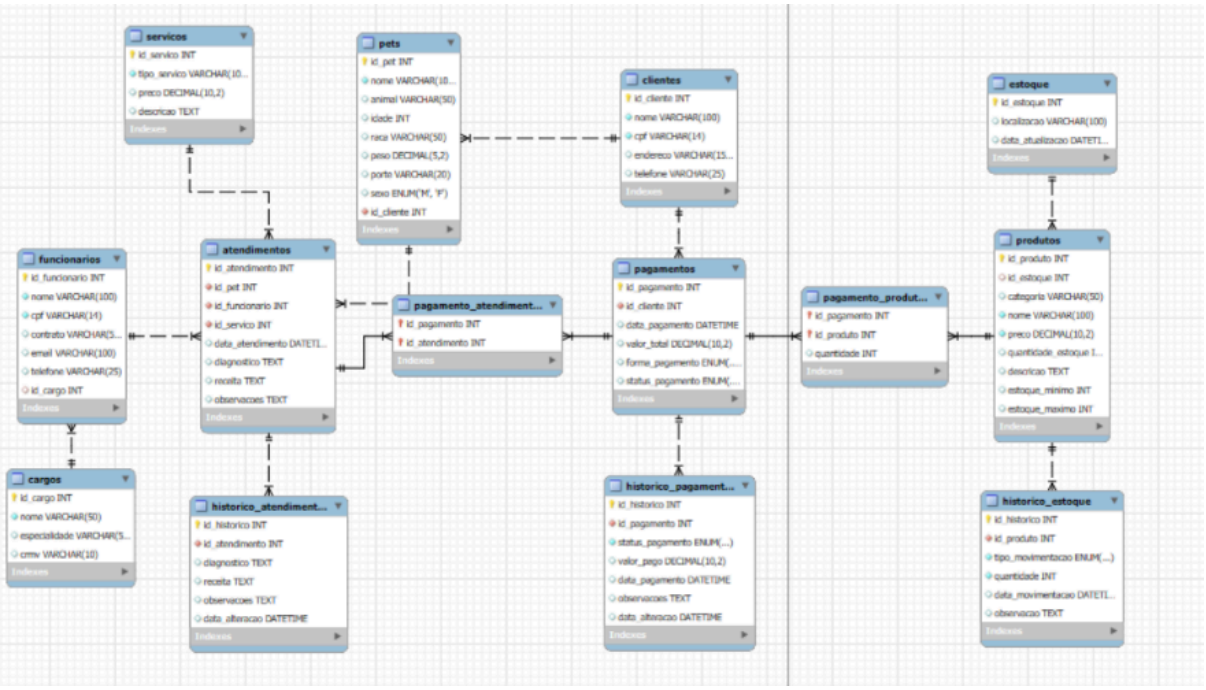


FIGURA 3: Apresenta o diagrama do banco de dados da Scooby Petshop, criado no MySQL Workbench. O modelo mostra as tabelas e os relacionamentos entre elas, servindo como base para a criação e validação do banco de dados. **FONTE**: elaborado pela equipe.

5. MODELO FÍSICO

O modelo físico é a parte final da construção do banco de dados, onde tudo o que foi planejado nas etapas anteriores é realmente criado e testado. Nessa fase, o banco da Scooby Petshop foi implementado no MySQL Workbench, com suas tabelas, relacionamentos e chaves definidos de acordo com o modelo lógico.

5.1. Criação das Tabelas

As tabelas foram criadas com base no modelo lógico, seguindo as ligações e regras definidas entre as entidades. O banco recebeu o nome clinica_petshop, e cada tabela foi configurada com seus campos, tipos de dados e restrições de integridade.

As figuras de 4 a 10 a seguir apresentam a criação das tabelas do banco de dados Scooby Petshop.

```sql
-- ------------------------------------------------------
-- BANCO DE DADOS
-- ------------------------------------------------------
CREATE DATABASE IF NOT EXISTS clinica_petshop
CHARACTER SET utf8mb4
COLLATE utf8mb4_general_ci;
USE clinica_petshop;



-- ------------------------------------------------------
-- TABELA: Clientes
-- ------------------------------------------------------
CREATE TABLE clientes (
    id_cliente INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    cpf VARCHAR(14) UNIQUE NOT NULL,
    endereco VARCHAR(150),
    telefone VARCHAR(25)
);



-- ------------------------------------------------------
-- TABELA: Pets
-- ------------------------------------------------------
CREATE TABLE pets (
    id_pet INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    animal VARCHAR(50),
    idade INT,
    raca VARCHAR(50),
    peso DECIMAL(5,2),
    porte VARCHAR(20),
    sexo ENUM('M', 'F'),
    id_cliente INT NOT NULL,
    FOREIGN KEY (id_cliente) REFERENCES clientes(id_cliente)
        ON DELETE CASCADE
);
```

FIGURA 4 - FONTE: elaborado pela equipe.

```sql
-- ------------------------------------------------------
-- TABELA: Cargos
-- ------------------------------------------------------
CREATE TABLE cargos (
    id_cargo INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(50) NOT NULL,
    especialidade VARCHAR(50),
    crmv VARCHAR(10)
);



-- ------------------------------------------------------
-- TABELA: Funcionarios
-- ------------------------------------------------------
CREATE TABLE funcionarios (
    id_funcionario INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    cpf VARCHAR(14) UNIQUE NOT NULL,
    contrato VARCHAR(50),
    email VARCHAR(100),
    telefone VARCHAR(25),
    id_cargo INT,
    FOREIGN KEY (id_cargo) REFERENCES cargos(id_cargo)
);



-- ------------------------------------------------------
-- TABELA: Servicos
-- ------------------------------------------------------
CREATE TABLE servicos (
    id_servico INT AUTO_INCREMENT PRIMARY KEY,
    tipo_servico VARCHAR(100) NOT NULL,
    preco DECIMAL(10,2),
    descricao TEXT
);
```

FIGURA 5 - FONTE: elaborado pela equipe.

```sql
-- ---------------------------------------------------------
-- TABELA: Atendimentos
-- ---------------------------------------------------------
CREATE TABLE atendimentos (
    id_atendimento INT AUTO_INCREMENT PRIMARY KEY,
    id_pet INT NOT NULL,
    id_funcionario INT NOT NULL,
    id_servico INT NOT NULL,
    data_atendimento DATETIME DEFAULT CURRENT_TIMESTAMP,
    diagnostico TEXT,
    receita TEXT,
    observacoes TEXT,
    FOREIGN KEY (id_pet) REFERENCES Pets(id_pet)
        ON DELETE CASCADE,
    FOREIGN KEY (id_funcionario) REFERENCES funcionarios(id_funcionario),
    FOREIGN KEY (id_servico) REFERENCES servicos(id_servico)
);



-- ---------------------------------------------------------
-- TABELA: Historico de Atendimentos
-- ---------------------------------------------------------
CREATE TABLE historico_atendimentos (
    id_historico INT AUTO_INCREMENT PRIMARY KEY,
    id_atendimento INT NOT NULL,
    diagnostico TEXT,
    receita TEXT,
    observacoes TEXT,
    data_alteracao DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_atendimento) REFERENCES atendimentos(id_atendimento)
        ON DELETE CASCADE
);
```

FIGURA 6 - FONTE: elaborado pela equipe.

```sql
-- ---------------------------------------------------------
-- TABELA: Pagamentos
-- ---------------------------------------------------------
CREATE TABLE pagamentos (
    id_pagamento INT AUTO_INCREMENT PRIMARY KEY,
    id_cliente INT NOT NULL,
    data_pagamento DATETIME DEFAULT NOW(),
    valor_total DECIMAL(10,2),
    forma_pagamento ENUM('Dinheiro', 'Cartão', 'Pix', 'Outro') NULL,
    status_pagamento ENUM('Pendente', 'Pago', 'Cancelado') DEFAULT 'Pendente',
    FOREIGN KEY (id_cliente) REFERENCES clientes(id_cliente)
        ON DELETE CASCADE
);



-- ---------------------------------------------------------
-- TABELA: Historico de Pagamentos
-- ---------------------------------------------------------
CREATE TABLE historico_pagamentos (
    id_historico INT AUTO_INCREMENT PRIMARY KEY,
    id_pagamento INT NOT NULL,
    status_pagamento ENUM('Pendente', 'Pago', 'Cancelado') NOT NULL,
    valor_pago DECIMAL(10, 2),
    data_pagamento DATETIME,
    observacoes TEXT,
    data_alteracao DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_pagamento) REFERENCES pagamentos(id_pagamento)
        ON DELETE CASCADE
);



-- ---------------------------------------------------------
-- TABELA: Estoque
-- ---------------------------------------------------------
CREATE TABLE estoque (
    id_estoque INT AUTO_INCREMENT PRIMARY KEY,
    localizacao VARCHAR(100),
    data_atualizacao DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

FIGURA 7 - FONTE: elaborado pela equipe.

FIGURA 8 - FONTE: elaborado pela equipe.

```sql
-- ---------------------------------------------------------
-- TABELA INTERMEDIÁRIA: Produtos em Pagamentos (N:N)
-- ---------------------------------------------------------
CREATE TABLE pagamento_produtos (
    id_pagamento INT,
    id_produto INT,
    quantidade INT DEFAULT 1,
    PRIMARY KEY (id_pagamento, id_produto),
    FOREIGN KEY (id_pagamento) REFERENCES pagamentos(id_pagamento)
        ON DELETE CASCADE,
    FOREIGN KEY (id_produto) REFERENCES produtos(id_produto)
);



-- ---------------------------------------------------------
-- TABELA INTERMEDIÁRIA: Atendimentos em Pagamentos (N:N)
-- ---------------------------------------------------------
CREATE TABLE pagamento_atendimentos (
    id_pagamento INT,
    id_atendimento INT,
    PRIMARY KEY (id_pagamento, id_atendimento),
    FOREIGN KEY (id_pagamento) REFERENCES pagamentos(id_pagamento)
        ON DELETE CASCADE,
    FOREIGN KEY (id_atendimento) REFERENCES atendimentos(id_atendimento)
        ON DELETE CASCADE
);



-- ---------------------------------------------------------
-- ÍNDICES DE BUSCA TEXTUAL
-- ---------------------------------------------------------
CREATE INDEX idx_nome_cliente ON clientes(nome);
CREATE INDEX idx_nome_pet ON pets(nome);
CREATE INDEX idx_nome_funcionario ON funcionarios(nome);
CREATE INDEX idx_tipo_servico ON servicos(tipo_servico);
```

FIGURA 9 - FONTE: elaborado pela equipe.

```sql
-- ------------------------------------------------------
-- ÍNDICES DE RELATÓRIOS E DESEMPENHO
-- ------------------------------------------------------

CREATE INDEX idx_pagamentos_cliente ON pagamentos(id_cliente);
CREATE INDEX idx_pagamentos_status ON pagamentos(status_pagamento);
CREATE INDEX idx_pagamentos_data ON pagamentos(data_pagamento);

CREATE INDEX idx_pagamento_produto ON pagamento_Produtos(id_produto);
CREATE INDEX idx_pagamento_pagamento ON pagamento_Produtos(id_pagamento);

CREATE INDEX idx_atendimentos_data ON atendimentos(data_atendimento);
CREATE INDEX idx_atendimentos_funcionario ON atendimentos(id_funcionario);
CREATE INDEX idx_atendimentos_pet ON atendimentos(id_pet);
CREATE INDEX idx_atendimentos_servico ON atendimentos(id_servico);

CREATE INDEX idx_pets_cliente ON pets(id_cliente);
```

FIGURA 10 - FONTE: elaborado pela equipe.

5.2. Inserção de Registros

Após a criação das tabelas, o banco de dados foi preenchido com alguns registros de teste para verificar o funcionamento e a integridade das tabelas. As inserções foram realizadas no MySQL Workbench, utilizando comandos SQL simples e as procedures criadas para inserções, das quais temos alguns exemplos nas Figuras 11 a 13.

```sql
-- ======================================================
-- CARGOS
-- ======================================================
INSERT INTO cargos (nome, especialidade, crmv) VALUES
('Veterinário Clínico', 'Clínica Geral', 'PI-12345'),
('Veterinário Cirurgião', 'Cirurgias e Emergências', 'PI-54321'),
('Tosador / Banho e Tosa', 'Higiene e Estética Animal', NULL),
('Atendente de Loja / Caixa', 'Vendas e Atendimento', NULL),
('Atendente de Recepção', 'Recepção e Agendamentos', NULL);

-- ======================================================
-- FUNCIONÁRIOS
-- ======================================================
INSERT INTO funcionarios (nome, cpf, email, telefone, id_cargo, contrato) VALUES
('Rafael Duarte', '931.139.273-15', 'rafael@clinicavet.com', '(86) 99999-0000', 1, 'CLT'),
('Milena Menezes', '782.577.513-91', 'milena@clinicavet.com', '(86) 98888-1112', 2, 'CLT'),
('Carlos Souza', '043.862.323-11', 'carlos@clinicavet.com', '(86) 98888-1234', 3, 'PJ'),
('Luciana Torres', '264.318.723-71', 'luciana@clinicavet.com', '(86) 97777-4321', 3, 'PJ'),
('Amanda Ribeiro', '760.174.013-78', 'amanda@clinicavet.com', '(86) 96666-4321', 4, 'CLT'),
('Ana Lima', '838.892.493-10', 'ana@clinicavet.com', '(86) 97777-4321', 5, 'CLT');

-- ======================================================
-- CLIENTES E PETS
-- ======================================================
INSERT INTO clientes (nome, cpf, endereco, telefone) VALUES
('João Pereira', '111.222.333-44', 'Rua das Flores, 100', '(86) 98800-1234'),
('Maria Silva', '555.666.777-88', 'Av. Frei Serafim, 3000', '(86) 98700-4321'),
('Carla Oliveira', '999.888.777-66', 'Rua Coelho de Resende, 250', '(86) 98600-9999'),
('Pedro Santos', '444.333.222-11', 'Av. Presidente Kennedy, 900', '(86) 98555-6666');

INSERT INTO pets (nome, animal, idade, raca, peso, porte, sexo, id_cliente) VALUES
('Thor', 'Cachorro', 3, 'Golden Retriever', 25.5, 'Grande', 'M', 1),
('Mia', 'Gato', 2, 'Persa', 3.5, 'Pequeno', 'F', 1),
('Luna', 'Cachorro', 5, 'Poodle', 6.0, 'Pequeno', 'F', 2),
('Zeca', 'Cachorro', 1, 'SRD', 9.0, 'Médio', 'M', 2),
('Bento', 'Cachorro', 1, 'Shih-Tzu', 6.0, 'Pequeno', 'M', 3),
('Toby', 'Gato', 4, 'Siamês', 5.0, 'Pequeno', 'M', 4);
```

FIGURA 11 - FONTE: elaborado pela equipe.

```sql
-- ================================================
-- ESTOQUE E PRODUTOS
-- ================================================
INSERT INTO estoque (localizacao, data_atualizacao) VALUES
('Depósito Central', NOW()),
('Balcão Principal', NOW());

INSERT INTO produtos (id_estoque, categoria, nome, preco, quantidade_estoque, descricao) VALUES
(1, 'Medicamento', 'Vermífugo Canino', 35.00, 50, 'Elimina vermes'),
(1, 'Vacina', 'Vacina V10', 70.00, 30, 'Protege contra múltiplas doenças'),
(2, 'Higiene', 'Shampoo Neutro', 25.00, 60, 'Para pele sensível'),
(2, 'Acessório', 'Coleira LED', 40.00, 15, 'Alta visibilidade noturna');

-- ================================================
-- SERVIÇOS
-- ================================================
INSERT INTO servicos (tipo_servico, preco, descricao) VALUES
('Consulta Veterinária', 80.00, 'Check-up completo'),
('Vacinação', 70.00, 'Aplicação de vacina V10'),
('Banho e Tosa', 55.00, 'Banho e tosa higiênica'),
('Cirurgia de Castração', 300.00, 'Cirurgia padrão'),
('Emergência Clínica', 150.00, 'Atendimento de urgência');

-- ================================================
-- ATENDIMENTOS (via procedure)
-- ================================================
CALL registrar_atendimento(1, 1, 1, 'Check-up geral', 'Recomendado vermífugo', 'Pet saudável');
CALL registrar_atendimento(2, 1, 2, 'Vacinação anual', 'Retornar em 12 meses', 'Sem reações');
CALL registrar_atendimento(3, 3, 3, 'Higienização completa', NULL, 'Pet tranquilo');
CALL registrar_atendimento(4, 2, 4, 'Castração com sucesso', 'Reposição de antibiótico', 'Recuperação boa');
CALL registrar_atendimento(5, 1, 5, 'Emergência por vômito', 'Soro e dieta leve', 'Estável');
```

FIGURA 12 - FONTE: elaborado pela equipe.

```sql
-- ================================================
-- PAGAMENTOS + ASSOCIAÇÕES (via procedures)
-- ================================================
CALL criar_pagamento(1); -- João
CALL associar_atendimento_pagamento(1, 1);
CALL associar_pagamento_produto(1, 1, 2); -- Vermífugo

CALL criar_pagamento(2); -- João
CALL associar_atendimento_pagamento(2, 2);
CALL associar_pagamento_produto(2, 2, 1); -- Vacina

CALL criar_pagamento(3); -- Carla
CALL associar_atendimento_pagamento(3, 3);
CALL associar_pagamento_produto(3, 3, 1); -- Shampoo

CALL criar_pagamento(4); -- Maria
CALL associar_atendimento_pagamento(4, 4);

CALL criar_pagamento(5); -- Pedro
CALL associar_atendimento_pagamento(5, 5);
CALL associar_pagamento_produto(5, 4, 1); -- Coleira LED
```

FIGURA 13 - FONTE: elaborado pela equipe.

5.3. CONSULTAS

As consultas SQL foram desenvolvidas para permitir a visualização prática e o teste das funcionalidades implementadas no banco de dados. Usando *views* e *procedures* que ajudaram a reunir informações importantes em consultas pré-definidas, facilitando a análise de dados como atendimentos e pagamentos

```sql
-- ------------------------------------------------------------------------
-- Consulta histórico de um cliente
-- ------------------------------------------------------------------------
DELIMITER $$

CREATE PROCEDURE consultar_historico_cliente (
  IN p_id_cliente INT
)
BEGIN
  DECLARE v_cliente_existente INT;

  -- Verifica se o cliente existe
  SELECT COUNT(*) INTO v_cliente_existente
  FROM Clientes
  WHERE id_cliente = p_id_cliente;

  IF v_cliente_existente = 0 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Cliente não encontrado.';
  END IF;

  -- Consulta o histórico do cliente
  SELECT
    c.nome AS cliente,
    p.nome AS pet,
    s.tipo_servico,
    a.data_atendimento,
    pg.status_pagamento,
    pg.valor_total
  FROM Clientes c
  LEFT JOIN Pets p ON c.id_cliente = p.id_cliente
  LEFT JOIN Atendimentos a ON p.id_pet = a.id_pet
  LEFT JOIN Servicos s ON a.id_servico = s.id_servico
  LEFT JOIN Pagamento_Atendimentos pa ON pa.id_atendimento = a.id_atendimento
  LEFT JOIN Pagamentos pg ON pa.id_pagamento = pg.id_pagamento
  WHERE c.id_cliente = p_id_cliente
  ORDER BY a.data_atendimento DESC;
END $$

DELIMITER ;
```

FIGURA 14 - FONTE: elaborado pela equipe.

```sql
-- ------------------------------------------------------------------------
-- Lista serviços de um pet
-- ------------------------------------------------------------------------
DELIMITER $$

CREATE PROCEDURE listar_servicos_por_pet (
  IN p_id_pet INT
)
BEGIN
  DECLARE v_pet_existente INT;

  SELECT COUNT(*) INTO v_pet_existente
  FROM Pets
  WHERE id_pet = p_id_pet;

  IF v_pet_existente = 0 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Pet não encontrado.';
  END IF;

  SELECT
    p.nome AS pet,
    s.tipo_servico,
    a.data_atendimento,
    a.diagnostico,
    a.receita
  FROM Atendimentos a
  JOIN Servicos s ON a.id_servico = s.id_servico
  JOIN Pets p ON p.id_pet = a.id_pet
  WHERE a.id_pet = p_id_pet
  ORDER BY a.data_atendimento DESC;
END $$

DELIMITER ;
```

FIGURA 15- FONTE: elaborado pela equipe.

```sql
-- ------------------------------------------------------------------------
-- Lista produtos de um pagamento
-- ------------------------------------------------------------------------
DELIMITER $$

CREATE PROCEDURE listar_produtos_por_pagamento (
  IN p_id_pagamento INT
)
BEGIN
  DECLARE v_pagamento_existente INT;

  SELECT COUNT(*) INTO v_pagamento_existente
  FROM Pagamentos
  WHERE id_pagamento = p_id_pagamento;

  IF v_pagamento_existente = 0 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Pagamento não encontrado.';
  END IF;

  SELECT
    pr.nome AS produto,
    pr.preco,
    pp.quantidade,
    (pr.preco * pp.quantidade) AS subtotal
  FROM Pagamento_Produtos pp
  JOIN Produtos pr ON pr.id_produto = pp.id_produto
  WHERE pp.id_pagamento = p_id_pagamento;
END $$
```

FIGURA 16 - FONTE: elaborado pela equipe.

```sql
-- ------------------------------------------------------------------------
-- Lista atendimentos de um funcionário
-- ------------------------------------------------------------------------
DELIMITER $$

CREATE PROCEDURE listar_atendimentos_por_funcionario (
  IN p_id_funcionario INT
)
BEGIN
  DECLARE v_func_existente INT;

  SELECT COUNT(*) INTO v_func_existente
  FROM Funcionarios
  WHERE id_funcionario = p_id_funcionario;

  IF v_func_existente = 0 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Funcionário não encontrado.';
  END IF;

  SELECT
    f.nome AS funcionario,
    c.nome AS cliente,
    p.nome AS pet,
    s.tipo_servico,
    a.data_atendimento
  FROM Atendimentos a
  JOIN Funcionarios f ON f.id_funcionario = a.id_funcionario
  JOIN Pets p ON p.id_pet = a.id_pet
  JOIN Clientes c ON c.id_cliente = p.id_cliente
  JOIN Servicos s ON s.id_servico = a.id_servico
  WHERE a.id_funcionario = p_id_funcionario
  ORDER BY a.data_atendimento DESC;
END $$

DELIMITER ;
```

FIGURA 17 - FONTE: elaborado pela equipe.

```sql
-- ------------------------------------------------------------------------
-- Lista pagamentos que possuem status específico
-- ------------------------------------------------------------------------
DELIMITER $$

CREATE PROCEDURE relatorio_pagamentos_por_status()
BEGIN
  SELECT
    status_pagamento,
    COUNT(*) AS quantidade,
    SUM(COALESCE(valor_total,0)) AS total
  FROM Pagamentos
  GROUP BY status_pagamento;
END $$

DELIMITER ;
```

FIGURA 18- FONTE: elaborado pela equipe.

```sql
-- ------------------------------------------------------------------------
-- Atualizar data no estoque após mudança na quantidade
-- ------------------------------------------------------------------------
DELIMITER $$

CREATE TRIGGER trg_update_data_estoque
BEFORE UPDATE ON produtos
FOR EACH ROW
BEGIN
  IF OLD.quantidade_estoque <> NEW.quantidade_estoque THEN
    UPDATE estoque
    SET data_atualizacao = NOW()
    WHERE id_estoque = NEW.id_estoque
    AND (data_atualizacao IS NULL OR data_atualizacao < NOW());
  END IF;
END $$

DELIMITER ;



-- ------------------------------------------------------------------------
-- Histórico de atendimento automático
-- ------------------------------------------------------------------------
DELIMITER $$

CREATE TRIGGER trg_atendimento_historico_update
AFTER UPDATE ON atendimentos
FOR EACH ROW
BEGIN
  IF (TRIM(OLD.diagnostico) <> TRIM(NEW.diagnostico) OR TRIM(OLD.receita) <> TRIM(NEW.receita) OR TRIM(OLD.observacoes) <> TRIM(NEW.observacoes)) THEN
    INSERT INTO historico_atendimentos (id_atendimento, diagnostico, receita, observacoes, data_alteracao)
    VALUES (NEW.id_atendimento, NEW.diagnostico, NEW.receita, NEW.observacoes, NOW());
  END IF;
END $$

DELIMITER ;
```

FIGURA 19- FONTE: elaborado pela equipe.

```sql
-- ------------------------------------------------------------------------
-- Histórico de atendimento inicial (ao criar)
-- ------------------------------------------------------------------------
DELIMITER $$

CREATE TRIGGER trg_atendimento_historico_insert
AFTER INSERT ON atendimentos
FOR EACH ROW
BEGIN
  INSERT INTO historico_atendimentos (
    id_atendimento,
    diagnostico,
    receita,
    observacoes,
    data_alteracao
  )
  VALUES (
    NEW.id_atendimento,
    NEW.diagnostico,
    NEW.receita,
    NEW.observacoes,
    NOW()
  );
END $$

DELIMITER ;
```

FIGURA 20- FONTE: elaborado pela equipe.

```sql
-- ----------------------------------------------------
-- VIEWS DE PERFORMANCE
-- ----------------------------------------------------

-- pagamentos consolidados
CREATE OR REPLACE VIEW v_pagamentos_detalhados AS
SELECT
  pg.id_pagamento,
  c.nome AS cliente,
  pg.data_pagamento,
  pg.forma_pagamento,
  pg.status_pagamento,
  pg.valor_total,

  -- Total de produtos no pagamento
  (
    SELECT SUM(pp.quantidade * pr.preco)
    FROM pagamento_produtos pp
    JOIN produtos pr ON pr.id_produto = pp.id_produto
    WHERE pp.id_pagamento = pg.id_pagamento
  ) AS valor_produtos,

  -- Total de atendimentos no pagamento
  (
    SELECT SUM(s.preco)
    FROM pagamento_atendimentos pa
    JOIN atendimentos a ON a.id_atendimento = pa.id_atendimento
    JOIN servicos s ON s.id_servico = a.id_servico
    WHERE pa.id_pagamento = pg.id_pagamento
  ) AS valor_atendimentos

FROM pagamentos pg
JOIN clientes c ON pg.id_cliente = c.id_cliente
ORDER BY pg.id_pagamento;



-- produtividade de funcionários
CREATE OR REPLACE VIEW v_funcionarios_performance AS
SELECT
```

```sql
-- produtividade de funcionários
CREATE OR REPLACE VIEW v_funcionarios_performance AS
SELECT
  f.id_funcionario,
  f.nome AS funcionario,
  COUNT(DISTINCT a.id_atendimento) AS total_atendimentos,
  COALESCE(SUM(pg.valor_total), 0) AS total_faturado
FROM funcionarios f
LEFT JOIN atendimentos a ON f.id_funcionario = a.id_funcionario
LEFT JOIN pagamento_atendimentos pa ON pa.id_atendimento = a.id_atendimento
LEFT JOIN pagamentos pg ON pg.id_pagamento = pa.id_pagamento
GROUP BY f.id_funcionario, f.nome
ORDER BY total_faturado DESC;



-- Ranking de funcionários por número de atendimentos
CREATE OR REPLACE VIEW v_funcionarios_ranking AS
SELECT
  f.id_funcionario,
  f.nome AS funcionario,
  c.nome AS cargo,
  COUNT(a.id_atendimento) AS total_atendimentos
FROM funcionarios f
LEFT JOIN cargos c ON f.id_cargo = c.id_cargo
LEFT JOIN atendimentos a ON f.id_funcionario = a.id_funcionario
GROUP BY f.id_funcionario, f.nome, c.nome
ORDER BY total_atendimentos DESC;
```

FIGURA 21- FONTE: elaborado pela equipe.

FIGURA 22- FONTE: elaborado pela equipe.

6. CONSIDERAÇÕES FINAIS

O desenvolvimento do sistema de gestão Scooby Petshop foi uma experiência importante para colocar em prática o conteúdo estudado sobre modelagem e implementação de banco de dados. Durante o trabalho foi possível compreender como cada etapa, desde o levantamento do minimundo até a criação das consultas SQL, se conecta para formar um sistema funcional e coerente.

A estrutura criada atende às principais necessidades de uma clínica e petshop, permitindo o controle de clientes, pets, atendimentos, serviços, produtos e pagamentos de forma integrada. As consultas e views ajudaram a validar as tabelas e demonstrar o funcionamento do banco, mostrando como os dados se relacionam no dia a dia do sistema.

De forma geral o projeto mostrou a importância de planejar bem a modelagem antes da implementação, além de reforçar o trabalho em equipe e a atenção aos detalhes que fazem diferença no resultado.

7. REFERÊNCIAS BIBLIOGRÁFICAS

- ELMASRI, R.; NAVATHE, S. *Sistemas de Banco de Dados.* 7. ed. São Paulo: Pearson, 2017

- DATE, C. J. *Introdução a Sistemas de Bancos de Dados.* 8. ed. Rio de Janeiro: Elsevier, 2004.