# Sistema de Gerenciamento de Consultas Veterinárias

Ana Luiza Campos Souza
Célio Júnio de Freitas Eduardo
Giulia Moura Ferreira
Letícia Xavier de Oliveira

15 de fevereiro de 2025

Universidade de Brasília - Instituto de Ciências Exatas
Departamento de Ciência da Computação - CIC0134 - Banco de Dados
Professora Maristela Holanda
Prédio CIC/EST - Campus Universitário Darcy Ribeiro
Asa Norte 70919-970 Brasília, DF
211055441@aluno.unb.br
211010350@aluno.unb.br
200018795@aluno.unb.br
202065912@aluno.unb.br

**Resumo**

O VetMed é um sistema desenvolvido para otimizar a gestão de clínicas veterinárias, facilitando a organização de consultas, exames, receitas médicas, tratamentos e pagamentos. Para estruturar o banco de dados, foi elaborado um Modelo de Entidade-Relacionamento (MER), posteriormente transformado em Modelo Relacional. Além disso, foram criadas consultas em Álgebra Relacional envolvendo múltiplas tabelas e analisadas formas normais em cinco tabelas do banco para evitar redundâncias. Também foi elaborado o diagrama da camada de mapeamento, que auxilia na compreensão do armazenamento e acesso aos dados.
**Palavras-chave:** *Gerenciamento; Banco de Dados; Clínica Veterinária; Tecnologia.*

# 1 Introdução

O VetMed foi criado para auxiliar clínicas veterinárias na organização dos atendimentos, permitindo o gerenciamento eficiente de consultas, exames, receitas médicas e pagamentos. A necessidade surgiu da importância de digitalizar processos, reduzindo o uso de sistemas não informatizados e melhorando o acompanhamento dos pacientes. Com acesso rápido às informações, o sistema proporciona um atendimento mais ágil e preciso.

# 2 Modelo Entidade-Relacionamento (MER)

O Modelo de Entidade-Relacionamento (MER) ilustra como os dados estão organizados e interligados no sistema.

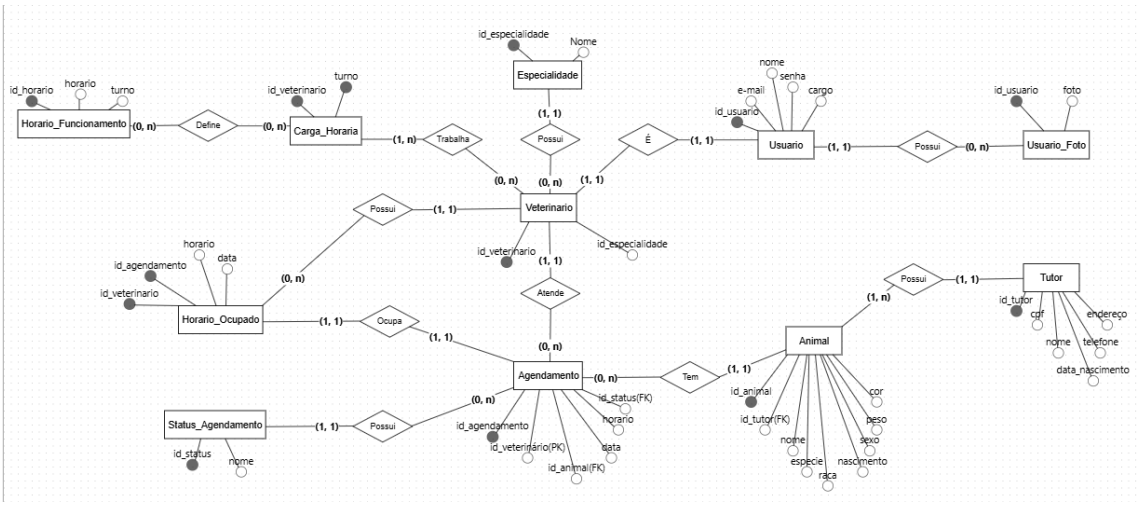


Figura 1: MER - VetMed

Para melhor visualização da imagem veja em MER - VetMed.

# 3 Modelo Relacional

O Modelo Relacional traduz o MER em tabelas estruturadas, normalizadas para evitar redundância e inconsistências.

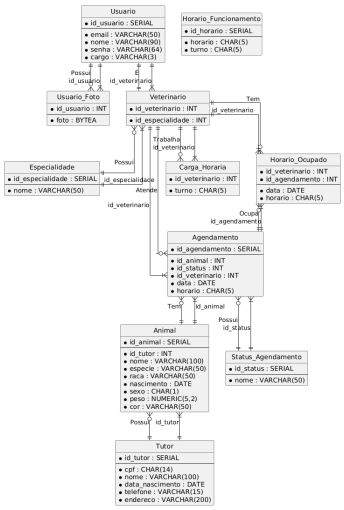


Figura 2: Modelo Relacional

Para melhor visualização da imagem, recomenda-se visualizar usando o seguinte link Modelo de Dados Relacional.

# 4 Álgebra Relacional

Com base no Modelo Relacional apresentado anteriormente, realizamos cinco consultas em Álgebra Relacional, utilizando alias para facilitar a leitura e respeitar o espaço disponível no documento. Os conceitos e a notação utilizados nas consultas foram baseados no livro (ELMASRI; NAVATHE, 2018).

## 4.1 Listar os agendamentos com informações do animal, tutor e veterinário

Passo 1: Relacionamos os agendamentos aos respectivos animais e tutores:

$$R_1 \leftarrow a \bowtie an \bowtie t$$

Passo 2: Relacionamos os agendamentos aos veterinários e usuários:

$$R_2 \leftarrow R_1 \bowtie v \bowtie u$$

Passo 3: Selecionamos apenas os atributos desejados:

$$\pi_{a.id\_agendamento,an.nome,t.nome,u.nome,a.data,a.horario}(R_2)$$

onde: $a = \text{Agendamento}, an = \text{Animal}, t = \text{Tutor}, v = \text{Veterinario}, u = \text{Usuario}$

## 4.2 Listar os veterinários disponíveis em determinado turno e especialidade

Passo 1: Relacionamos os veterinários aos usuários e suas especialidades:

$$R_1 \leftarrow v \bowtie u \bowtie e$$

Passo 2: Relacionamos com a carga horária e filtramos pelo turno:

$$R_2 \leftarrow \sigma_{c.turno='manha'}(R_1 \bowtie c)$$

Passo 3: Selecionamos os atributos desejados:

$$\pi_{v.id\_veterinario,u.nome,e.nome,c.turno}(R_2)$$

onde: $v = \text{Veterinario}, u = \text{Usuario}, e = \text{Especialidade}, c = \text{Carga\_Horaria}$

## 4.3 Listar os horários ocupados dos veterinários, incluindo nome do veterinário, nome do animal e status do agendamento

Passo 1: Relacionamos os horários ocupados aos agendamentos:

$$R_1 \leftarrow h \bowtie a$$

Passo 2: Relacionamos com os veterinários e seus respectivos usuários:

$$R_2 \leftarrow R_1 \bowtie v \bowtie u$$

Passo 3: Relacionamos com os animais e status do agendamento:

$$R_3 \leftarrow R_2 \bowtie an \bowtie s$$

Passo 4: Selecionamos os atributos desejados:

$$\pi_{v.id\_veterinario,u.nome,an.nome,a.data,a.horario,s.nome}(R_3)$$

onde: $h$ = Horario_Ocupado, $a$ = Agendamento, $v$ = Veterinario, $u$ = Usuario, $an$ = Animal, $s$ = Status_Agendamento

## 4.4 Listar todos os animais de um tutor com seus respectivos agendamentos e veterinários

Passo 1: Relacionamos os tutores aos animais:

$$R_1 \leftarrow t \bowtie an$$

Passo 2: Relacionamos com os agendamentos:

$$R_2 \leftarrow R_1 \bowtie a$$

Passo 3: Relacionamos com os veterinários e usuários:

$$R_3 \leftarrow R_2 \bowtie v \bowtie u$$

Passo 4: Selecionamos apenas os atributos desejados e filtramos pelo tutor desejado:

$$\pi_{t.id\_tutor,t.nome,an.nome,a.id\_agendamento,u.nome,a.data,a.horario}(\sigma_{t.id\_tutor=10}(R_3))$$

onde: $t$ = Tutor, $an$ = Animal, $a$ = Agendamento, $v$ = Veterinario, $u$ = Usuario

## 4.5 Listar os veterinários que não têm nenhum agendamento em um determinado dia

Passo 1: Obtemos todos os veterinários com seus nomes e especialidades:

$$R_1 \leftarrow \pi_{v.id\_veterinario,u.nome,e.nome}(v \bowtie u \bowtie e)$$

Passo 2: Obtemos os veterinários que possuem agendamentos em uma data específica:

$$R_2 \leftarrow \pi_{v.id\_veterinario,u.nome,e.nome}(\sigma_{a.data='2025-02-14'}(v \bowtie u \bowtie e \bowtie a))$$

Passo 3: Subtraímos os que têm agendamentos dos que existem:

$$R_1 - R_2$$

onde: $v$ = Veterinario, $u$ = Usuario, $e$ = Especialidade, $a$ = Agendamento

# 5 Avaliação das Formas Normais

A seguir, analisamos cinco tabelas do modelo relacional e verificamos se atendem às formas normais (1FN, 2FN e 3FN).

## 5.1 Tabela Agendamento

**Atributos:** id_agendamento, id_animal, id_veterinario, data, horario, id_status

**1FN:** Atende, pois todos os atributos possuem valores atômicos e não há grupos repetitivos.

**2FN:** Atende, pois a chave primária (id_agendamento) é única e não há dependências parciais.

**3FN:** Atende, pois todos os atributos dependem unicamente da chave primária, sem dependências transitivas.

## 5.2 Tabela Animal

**Atributos:** id_animal, nome, espécie, raça, id_tutor

**1FN:** Atende, pois todos os atributos são atômicos.

**2FN:** Atende, pois a chave primária (id_animal) determina todos os atributos sem dependências parciais.

**3FN:** Atende, pois não há dependências transitivas (raça e espécie dependem apenas do id_animal).

## 5.3 Tabela Tutor

**Atributos:** id_tutor, nome, telefone, e-mail

**1FN:** Atende, pois os atributos são atômicos.

**2FN:** Atende, pois id_tutor determina unicamente todos os outros atributos.

**3FN:** Atende, pois não há dependências transitivas entre os atributos.

## 5.4 Tabela Veterinario

**Atributos:** id_veterinario, id_usuario, id_especialidade

**1FN:** Atende, pois todos os atributos são atômicos.

**2FN:** Atende, pois a chave primária (id_veterinario) determina todos os outros atributos.

**3FN:** Atende, pois id_usuario e id_especialidade dependem diretamente da chave primária, sem dependências transitivas.

## 5.5 Tabela Carga_Horaria

**Atributos:** id_carga, id_veterinario, turno, horas

**1FN:** Atende, pois não há grupos repetitivos.

**2FN:** Atende, pois a chave primária (id_carga) determina todos os atributos sem dependências parciais.

**3FN:** Atende, pois turno e horas dependem apenas da chave primária.

# 6 Script SQL para Geração do Banco de Dados

O script SQL utilizado para gerar o banco de dados é extenso e pode ser acessado diretamente através do link abaixo:

Script SQL para Geração do Banco de Dados.

Este script inclui todas as instruções necessárias para criar as tabelas, relacionamentos e demais estruturas do banco de dados utilizado neste projeto.

# 7 Camada de Mapeamento

O sistema de agendamento foi projetado para gerenciar consultas veterinárias de forma eficiente, garantindo a validação dos dados antes da persistência no banco de dados. A arquitetura é baseada em uma API Flask que interage com um banco de dados PostgreSQL e módulos especializados para tratamento das regras de negócio e utilitários auxiliares. O desenvolvimento do mapeamento e das interações com o banco de dados foi feito com o auxílio do livro de (TEOREY et al., 2014) e, a construção de *Views*, *Procedures* e a criação das tabelas, na documentação do PostgreSQL versão 17 (GROUP, 2024) e finalmente, todos os diagramas foram baseados no (BRMODELOWEB, 2025) e no (PLANTTEXT, 2025) que utiliza o PlantUML.

## 7.1 Diagrama de Componentes

A estrutura geral do sistema pode ser melhor compreendida através da Figura 3 ou no link:

Neste diagrama, a API Flask funciona como ponto central da aplicação, sendo responsável por receber requisições e interagir com os diferentes módulos. O banco de dados PostgreSQL contém as tabelas que armazenam informações sobre agendamentos, usuários, veterinários, animais e seus tutores. Já o módulo de utilitários auxilia na formatação de dados e na obtenção de informações necessárias para a criação dos agendamentos. Por fim, o módulo de agendamentos gerencia a lógica de criação, validação e persistência das marcações.

## 7.2 Fluxo de Agendamento

O processo de criação de um agendamento segue uma sequência específica de interações entre os componentes do sistema, como mostrado na Figura 4 abaixo ou no link:

Quando o usuário envia uma requisição para criar um novo agendamento, a *API Flask* repassa a solicitação para o controlador, que valida os dados e os encaminha para o serviço de agendamentos. Esse serviço, por sua vez, consulta o módulo de utilidades para verificar informações como a disponibilidade de horários e a validade dos dados fornecidos. Caso todas as verificações sejam bem-sucedidas, os dados são enviados para o repositório, que executa a inserção no banco de dados através de uma *stored procedure*. Finalmente, a resposta é retornada ao usuário confirmando o sucesso da operação ou informando eventuais erros.

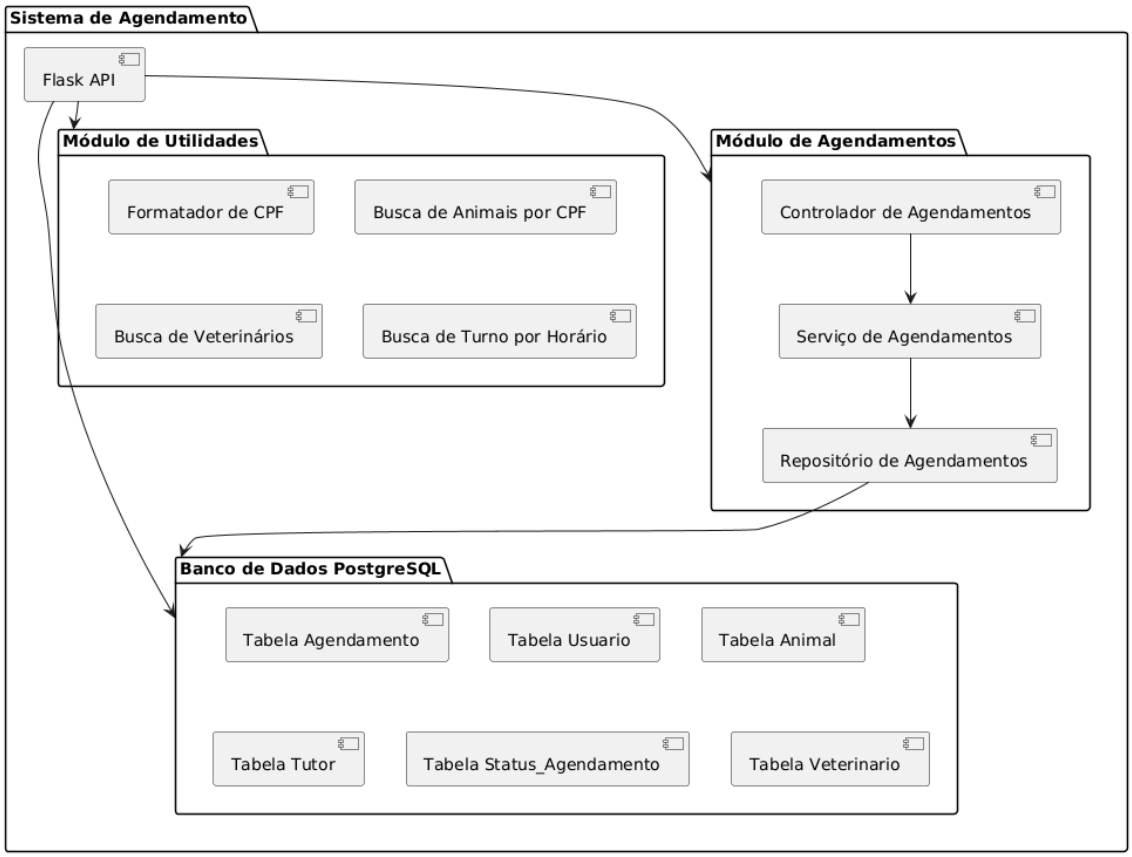


Figura 3: Diagrama de Componentes para a Implementação do Agendamento

## 7.3 Resumo

A modelagem do sistema facilita a organização e manutenção do código ao separar responsabilidades em diferentes módulos. O diagrama de componentes ajuda a visualizar a interação entre as partes do sistema, enquanto o diagrama de sequência detalha o fluxo de uma requisição específica. Juntos, esses diagramas auxiliam na compreensão da arquitetura do projeto e garantem uma documentação clara para futuras melhorias ou expansões.

# 8 Conclusão

O desenvolvimento do projeto final da disciplina de Banco de Dados proporcionou uma compreensão aprofundada dos conceitos e técnicas abordados em sala de aula, incluindo modelagem, implementação de sistemas gerenciadores de banco de dados (SGBDs), desenvolvimento de operações *CRUD* (Create, Read, Update, Delete), e manipulação de dados. Durante o processo, o grupo pôde perceber a importância de uma estruturação eficiente e bem planejada para o sucesso do projeto. Dessa forma, o trabalho em equipe foi essencial, garantindo integridade, desempenho e organização das informações, com o objetivo de cumprir os requisitos estabelecidos na especificação e entregar o trabalho no prazo.

Além disso, a aplicação prática de temas como modelagem relacional, normaliza-

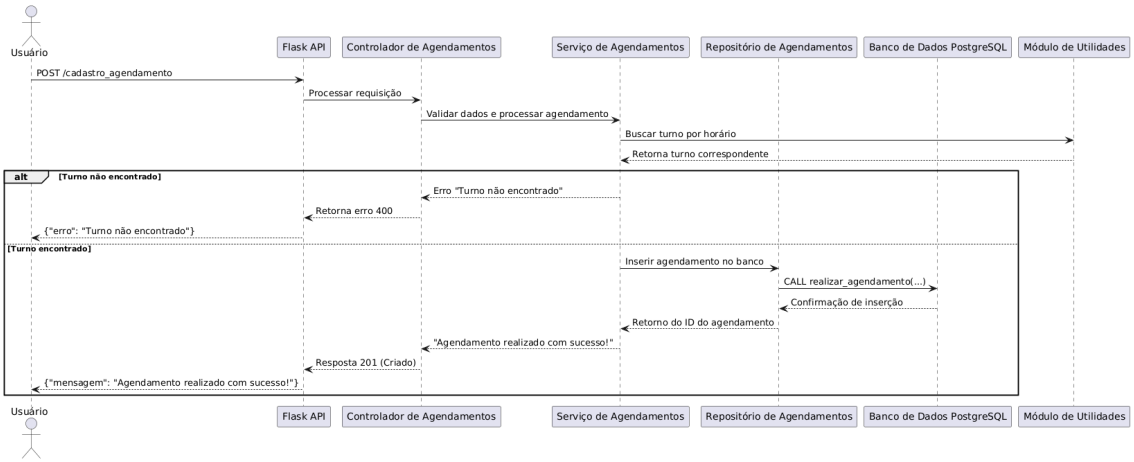


Figura 4: Diagrama de Sequência para a Implementação do Agendamento

ção e linguagem SQL permitiu a consolidação do conhecimento adquirido em aula, tornando o aprendizado mais dinâmico ao aplicá-lo a cenários reais. Implementar um sistema completo, que incluiu desde a modelagem até a implementação de um CRUD funcional, foi fundamental para adquirir o conhecimento prático necessário no desenvolvimento de soluções tecnológicas baseadas em banco de dados de maneira eficaz.

A experiência obtida ao longo do projeto foi crucial para o desenvolvimento de sistemas mais robustos e eficientes no futuro. Trabalhar em grupo também proporcionou uma valiosa oportunidade de desenvolver habilidades de colaboração e gerenciamento de tempo, assegurando que todas as tarefas fossem concluídas de maneira coordenada e dentro do prazo estipulado. Com isso, o projeto cumpriu seu papel de reforçar os conceitos teóricos aprendidos, ao mesmo tempo em que preparou os alunos para enfrentar desafios práticos no campo da tecnologia da informação.

# 9 GitHub

Link `https://github.com/celio-eduardo/BDProjeto2024-2`

# Referências


BRMODELOWEB. *BRModeloWeb*. Acesso em: 21 jan. 2025. 2025. Disponível em: <`https://www.brmodeloweb.com/lang/pt-br/index.html`>.

ELMASRI, Ramez; NAVATHE, Shamkant B. *Sistemas de banco de dados*. Edição: nzo Seraphim e Thatyana de Faria Piola Seraphim. Tradução: Daniel Vieira. 7. ed. São Paulo: Pearson Education do Brasil, 2018.

GROUP, PostgreSQL Global Development. *PostgreSQL 17 Documentation*. 2024. Acesso em: 07 jan. 2025. Disponível em: <`https://www.postgresql.org/docs/17/index.html`>.

PLANTTEXT. *PlantText*. Acesso em: 21 jan. 2025. 2025. Disponível em: <https://www.planttext.com/>.

TEOREY, Tobey J. et al. *Projeto e modelagem de banco de dados*. Tradução: Daniel Vieira. 2. ed. Rio de Janeiro: Elsevier, 2014.