PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

CIÊNCIA DA COMPUTAÇÃO

ERICK MAESTRI DE SOUZA

Desenvolvimento de um Sistema de Cadastro e Gerenciamento de Informações em Clínicas Veterinárias com Interface Gráfica em PyQt5, utilizando SQLAlchemy e SQLite

CURITIBA

2024

ERICK MAESTRI DE SOUZA

Desenvolvimento de um Sistema de Cadastro e Gerenciamento de Informações em Clínicas Veterinárias com Interface Gráfica em PyQt5, utilizando SQLAlchemy e SQLite

Trabalho Discente Efetivo realizado ao curso de Ciência da Computação da Pontifícia Universidade Católica do Paraná como requisito parcial da disciplina de Arquitetura de Banco de Dados.

Orientador: Prof. Ms. Rodrigo da Silva do Nascimento

CURITIBA

2024

SUMÁRIO

RESUMO

O trabalho apresenta o desenvolvimento de um sistema de gerenciamento para uma clínica veterinária, abrangendo a administração de dados de donos, animais, consultas, vacinas veterinários e espécies. Esse sistema é resultado de um estudo aplicado que visa criar uma solução prática para a realização de operações CRUD (criar, ler, atualizar e deletar) sobre todas as entidades envolvidas. O objetivo principal do projeto é fornecer uma ferramenta que facilite o controle de informações importantes, como nome, endereço e números de contato dos proprietários de animais, bem como os dados referentes aos próprios animais e aos veterinários responsáveis.

A metodologia utilizada inclui a implementação de um banco de dados SQLite, integrado ao sistema por meio de um mapeamento objeto-relacional (ORM) utilizando a biblioteca SQLAlchemy. Além disso, uma interface gráfica foi desenvolvida com PyQt5, permitindo uma interação mais intuitiva e acessível ao usuário.

Com a interface, foram implementadas funcionalidades que permitem selecionar facilmente os registros dos donos e seus respectivos animais, tornando a consulta, atualização e exclusão de dados mais ágeis. O sistema, que oferece um gerenciamento eficiente e integrado das diferentes entidades, representa uma base robusta e prática, com potencial para aprimorar a gestão de dados na clínica veterinária.

Palavras-chave: Gerenciamento Veterinário, CRUD, SQLite, SQLAlchemy, PyQt5.

ABSTRACT

The project presents the development of a management system for a veterinary clinic, encompassing the administration of data related to owners, animals, appointments, vaccinations, veterinarians, and species. This system is the result of an applied study aimed at creating a practical solution for performing CRUD (create, read, update, and delete) operations on all the involved entities. The main objective of the project is to provide a tool that facilitates the control of important information, such as the names, addresses, and contact numbers of pet owners, as well as the data related to the animals themselves and the responsible veterinarians.

The methodology used includes the implementation of a SQLite database, integrated into the system through an object-relational mapping (ORM) using the SQLAlchemy library. Additionally, a graphical interface was developed with PyQt5, allowing for a more intuitive and accessible user interaction.

With the interface, functionalities were implemented that allow users to easily select the records of owners and their respective animals, making the querying, updating, and deleting of data more efficient. The system, which offers an integrated and efficient management of the different entities, represents a robust and practical foundation, with the potential to enhance data management in the veterinary clinic.

Keywords: Veterinary Management, CRUD, SQLite, SQLAlchemy, PyQt5.

INTRODUÇÃO

Este trabalho tem como objetivo desenvolver um sistema de gerenciamento para uma clínica veterinária, focando na administração dos dados dos proprietários dos animais atendidos. A clínica veterinária, como entidade que lida com um grande volume de dados de clientes e seus animais, requer um sistema que permita o gerenciamento eficaz e organizado desses dados.

O objetivo geral do projeto é implementar um sistema que permita a administração eficiente dos dados dos principais componentes de uma clínica veterinária, com a possibilidade de realizar operações CRUD (criar, ler, atualizar e deletar). Especificamente, o sistema se concentrará na implementação de um banco de dados SQLite integrado com SQLAlchemy para gerenciar as informações dos donos e garantir a integridade e acessibilidade dos dados. Além disso, foi utilizada a biblioteca PyQt5 para desenvolver uma interface gráfica intuitiva, facilitando a interação do usuário com o sistema.

O documento está estruturado da seguinte forma: na seção de Desenvolvimento do Sistema, estão descritos os detalhes técnicos e a implementação do banco de dados. A Análise dos Resultados discutirá o desempenho e a eficácia do sistema desenvolvido. Finalmente, a Conclusão apresentará as principais conclusões e possíveis direções para futuras melhorias.

DESENVOLVIMENTO DO SISTEMA

8.1 Modelo Textual

História B

Mariana é a administradora de uma clínica veterinária chamada "focinhos fofinhos." A clínica oferece serviços de atendimento para vários tipos de animais, incluindo cães, gatos e aves. Cada animal atendido na clínica tem um histórico de consultas, vacinas e tratamentos realizados. Na clínica, cada animal é associado a um dono, que pode ser uma pessoa física ou uma instituição de adoção de animais. Os donos têm informações pessoais como nome, endereço e telefone. Além disso, a clínica possui uma equipe de veterinários que atende aos animais. Cada veterinário tem um número de registro profissional e uma especialização. Mariana quer criar um banco de dados para gerenciar os atendimentos dos animais, informações dos donos e os dados dos veterinários.

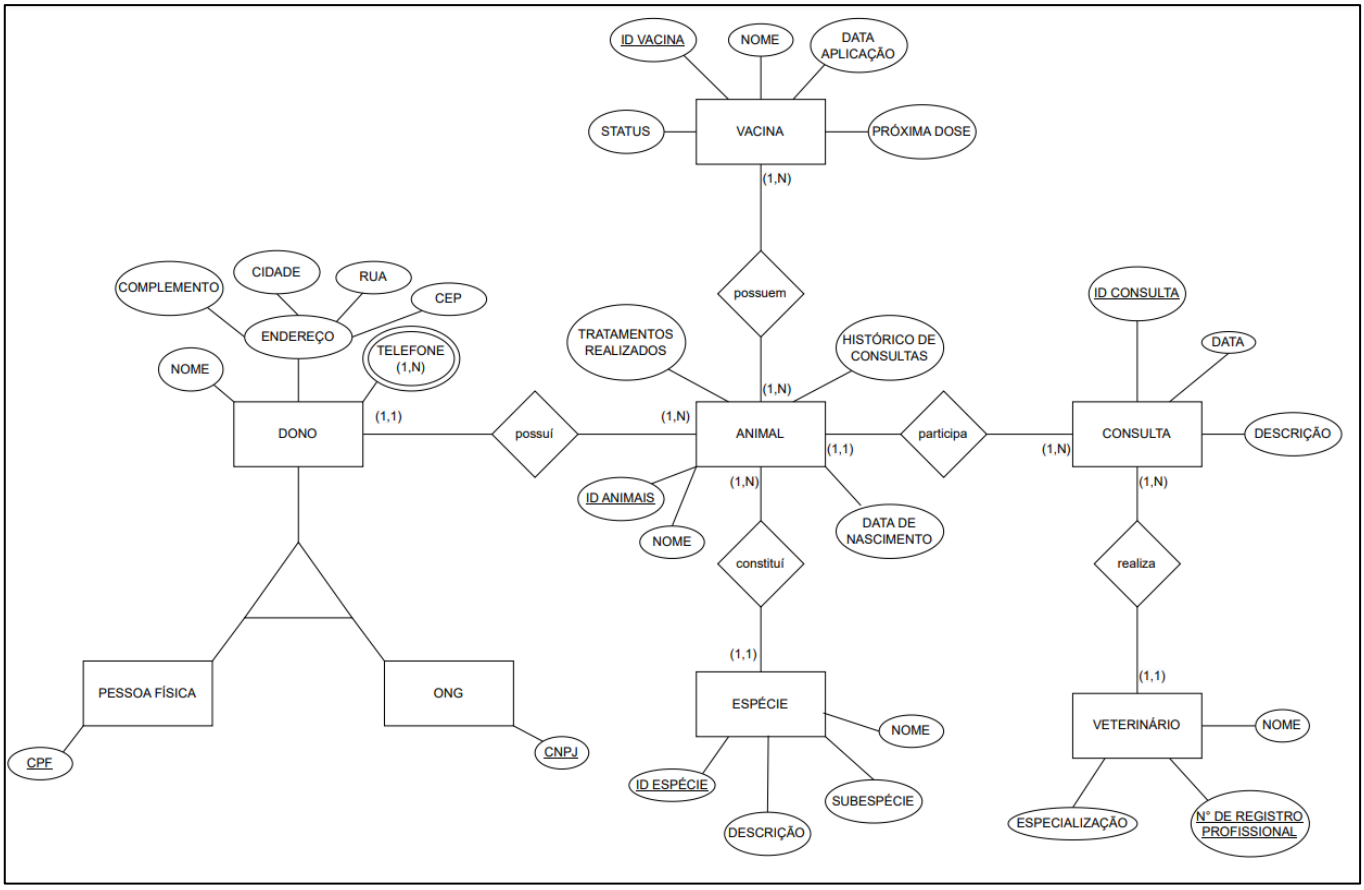
O modelo textual utilizado para a construção do sistema de gestão da clínica veterinária "Focinhos Fofinhos" foi baseado em um cenário fictício, que descreve as necessidades e interações entre os elementos fundamentais da clínica. A clínica, administrada por Mariana, oferece diversos serviços para animais de diferentes espécies, como cães, gatos e aves.

Cada animal que recebe atendimento na clínica possui um histórico detalhado de consultas, vacinas e tratamentos realizados. Esses animais estão vinculados a seus respectivos donos, que podem ser tanto pessoas físicas quanto instituições de adoção de animais. As informações pessoais dos donos, como nome, endereço e telefone, são registradas no sistema.

A clínica conta com uma equipe de veterinários, cada um com seu número de registro profissional e especialização, responsáveis por realizar os atendimentos. O objetivo de Mariana, como administradora, é criar um banco de dados que permita o gerenciamento eficiente dessas informações, contemplando o controle de atendimentos, dados dos donos e veterinários.

Esse modelo textual orienta a estrutura do sistema a ser desenvolvido, garantindo que as principais relações e atributos necessários estejam claramente definidos para a criação de um banco de dados robusto e funcional.

8.2 Modelo Conceitual



O modelo conceitual desenvolvido para a clínica veterinária "Focinhos Fofinhos" estrutura as principais entidades e suas interações, otimizando a gestão dos dados no sistema. As entidades centrais incluem Donos, Animais, Espécie, Veterinários, Consultas e Vacinas.

A entidade Dono possui especializações em ONG e Pessoa Física, diferenciando entre instituições e indivíduos que possuem animais. Cada dono pode ter múltiplos Telefones associados, que são armazenados em uma tabela separada, além de estar vinculado a um ou mais Animais sob sua responsabilidade. As informações pessoais dos donos incluem nome, endereço e, para Pessoa Física, o CPF, enquanto as ONGs têm o CNPJ registrado.

A entidade Animal é associada a um Dono específico e pertence a uma Espécie, categorizando animais como cães, gatos, aves, etc. As informações de cada animal incluem vacinas, com dados detalhados sobre a aplicação e data da próxima dose, e um histórico de consultas e tratamentos realizados. Além disso, um animal pode ser atendido por vários Veterinários através de consultas.

A entidade Veterinário armazena informações dos profissionais, incluindo especialização e número de registro profissional. Cada veterinário pode realizar múltiplas consultas e atender diferentes animais. A associação entre Veterinários e Animais é registrada em uma tabela de relacionamento, permitindo múltiplos atendimentos e consultas.

As Consultas mantêm um histórico dos atendimentos realizados, com a data, descrição e o veterinário responsável por cada consulta. Isso facilita o acompanhamento de cada caso e dos tratamentos aplicados.

Por fim, o conceito de Vacinas permite o registro detalhado das imunizações de cada animal, incluindo o status, a data de aplicação e a próxima dose. Esse controle contribui para o acompanhamento preventivo da saúde dos animais atendidos.

O modelo conceitual foi desenvolvido em diagramas de entidade-relacionamento (ER), evidenciando as interações e os relacionamentos entre as entidades. Essa organização simplifica a transição para o banco de dados e assegura a precisão na implementação do sistema de gestão da clínica.

8.3 Descrição do Sistema

O sistema foi desenvolvido com o objetivo de gerenciar os dados de uma clínica veterinária, facilitando o cadastro, consulta e atualização de informações relacionadas a Animais, seus Donos e os Veterinários responsáveis pelo atendimento. A interface gráfica implementada permite realizar operações de CRUD (Criar, Ler, Atualizar e Deletar) em todas as entidades do sistema.

A entidade central do modelo é Animais, que armazena informações detalhadas sobre cada animal, incluindo histórico de consultas, tratamentos realizados e vacinas aplicadas. O sistema também gerencia os dados dos Donos, incluindo nome, endereço, CPF e múltiplos números de telefone, que são representados em uma tabela associada como um atributo multivalorado.

Além disso, o sistema registra as informações dos Veterinários, que incluem especialização e número de registro profissional. Essa estrutura permite uma integração completa entre os animais tratados, seus donos e os profissionais responsáveis pelo atendimento, promovendo um gerenciamento eficiente e organizado das informações da clínica.

Embora o foco do sistema não seja exclusivamente o acompanhamento de históricos de consultas e tratamentos, ele foi projetado para registrar e manter esses dados de maneira eficiente. A interface gráfica proporciona um acesso fácil e intuitivo para a inclusão e gerenciamento de vacinas e outros dados relevantes, garantindo suporte abrangente às operações da clínica veterinária.

8.4 Implementação do Banco de Dados

O banco de dados foi implementado utilizando a biblioteca SQLAlchemy para realizar o mapeamento objeto-relacional (ORM), facilitando a interação entre o código Python e o banco de dados relacional, permitindo a manipulação de dados como se fossem objetos Python. Além disso, a biblioteca os, foi utilizada para garantir que o banco de dados fosse limpo sempre que o sistema fosse iniciado.

8.5 Funcionalidades e CRUD

O sistema implementa operações básicas de CRUD (Create, Read, Update, Delete) para todas as entidades presentes nele. As principais funcionalidades desenvolvidas são:

- **Create**: Permite o cadastro de uma nova pessoa física, inserindo dados como nome, CPF, endereço e números de telefone. Cada telefone é armazenado como um registro separado na tabela "Telefone", garantindo que uma pessoa possa ter múltiplos números associados.
- **Read**: O sistema oferece a funcionalidade de consulta de pessoas físicas cadastradas com base no CPF. Durante a consulta, o sistema exibe todas as informações associadas à pessoa, incluindo seus números de telefone, se houver.
- **Update**: O sistema permite a atualização das informações de uma pessoa física, incluindo nome, endereço e números de telefone. Caso um número de telefone precise ser alterado, o usuário pode selecionar qual telefone modificar.
- **Delete**: Também é possível excluir registros de pessoas físicas, removendo todos os dados associados, incluindo os números de telefone registrados.

8.6 Implementação da Interface Gráfica

A implementação da interface gráfica foi realizada com a biblioteca PyQt5, que possibilitou a criação de uma experiência de usuário intuitiva e interativa. Utilizando widgets e layouts oferecidos pelo PyQt5, desenvolvemos janelas modais para exibição de informações, formulários de entrada de dados e diálogos de confirmação, garantindo uma navegação fluida entre as diferentes funcionalidades do sistema. O design da interface foi pensado para ser claro e acessível, permitindo que os usuários realizassem operações de CRUD (criação, leitura, atualização e exclusão) de maneira eficiente. Além disso, a integração com o banco de dados foi facilitada através do uso de eventos e sinais do PyQt5, permitindo que as ações do usuário se refletissem instantaneamente nas operações realizadas no banco de dados.

ANÁLISE DE RESULTADOS

Durante o desenvolvimento do sistema de gerenciamento de dados para uma clínica veterinária, observou-se como a implementação do modelo relacional contribuiu para a organização e manipulação eficiente das informações. O uso do SQLAlchemy como ORM simplificou o mapeamento das entidades e suas relações, proporcionando uma interface mais natural e intuitiva para a interação com o banco de dados. A modelagem das entidades principais — como Dono**,** Pessoa Física**,** ONG**,** Animal**,** Veterinário**,** Vacina e Consulta — estruturou adequadamente o sistema para suportar operações de CRUD completas em todas as entidades. Esse design não apenas facilitou o gerenciamento dos dados de **Pessoa Física**, mas também permitiu o armazenamento de múltiplos telefones por dono e múltiplas vacinas por animal, por meio de relacionamentos com tabelas associativas, como Telefone e Vacinas.

CRUD em Python

Analisando o CRUD implementado na interface gráfica, o "*MenuPrincipal"* centraliza o gerenciamento de diferentes entidades como "Pessoa Física," "ONG," "Animal," "Espécie," "Consulta," "Veterinário" e "Vacinas." Este menu principal fornece um acesso intuitivo a cada entidade através de botões dedicados, permitindo realizar operações CRUD com facilidade.

A função "open_menu" é um exemplo de modularidade e reutilização de código, pois cria uma janela de menu CRUD genérica que é personalizada para cada entidade ao receber funções específicas de criação, leitura, atualização e exclusão. Esse uso de parâmetros e de lambda para conectar os botões de CRUD às funções respectivas demonstra um design coeso, onde o fluxo para cada operação é consistente e previsível, facilitando a interação do usuário e mantendo o código organizado.

As funções de "handle_create", "handle_read", "handle_update" e "handle_delete" incluem mensagens de feedback que indicam sucesso ou falha nas operações, melhorando a usabilidade ao informar o status das ações realizadas. Além disso, o uso de "QMessageBox" para mensagens de erro garante que o usuário seja alertado em casos de falhas, mantendo a transparência do sistema e auxiliando na resolução de problemas.

A seguir, como exemplo, as funcionalidades do CRUD de Pessoa Física com mais detalhes:

CRUD de Pessoa Física

O código implementa as operações de CRUD (Create, Read, Update, Delete) para a entidade Pessoa Física. Ele utiliza a biblioteca Qt para interface gráfica, exibindo janelas e formulários para inserção, edição e visualização de dados. Abaixo, uma descrição das funcionalidades implementadas:

1. Create (Criar Pessoa Física)

- **Descrição**: Abre uma janela de diálogo (QDialog) onde o usuário pode inserir os dados de uma nova Pessoa Física.
- **Campos**: Nome, CPF, CEP, Rua, Cidade, Complemento e Telefones (adicionados como lista, separados por vírgulas).
- **Processo**: Após preencher os campos e clicar em "Salvar", a função salvar_pessoa_fisica é chamada. Esta função:
    - Cria um objeto PessoaFisica e o adiciona ao banco de dados.
    - Adiciona cada número de telefone como uma nova entrada associada a PessoaFisica.
- **Feedback**: Exibe uma mensagem de confirmação ao usuário.

2. Read (Ler Pessoa Física)

- **Descrição**: Permite ao usuário buscar uma Pessoa Física pelo CPF e exibir os dados correspondentes em uma janela de diálogo.
- **Funcionamento**: O CPF é solicitado ao usuário e, caso encontrado, os dados da pessoa são exibidos (Nome, CPF, Endereço, Telefones). Se o CPF não existir, uma mensagem de erro é exibida.
- **Resultado**: Apresenta as informações na tela ou informa que a pessoa não foi encontrada.

3. Update (Atualizar Pessoa Física)

- **Descrição**: Permite que o usuário atualize informações específicas de uma Pessoa Física já cadastrada, identificada pelo CPF.
- **Processo**:
    - Após inserir o CPF, uma lista de atributos disponíveis para atualização é exibida. O usuário escolhe o atributo e insere o novo valor.
    - Telefones podem ser atualizados ao fornecer uma nova lista de números (os telefones antigos são apagados).
- **Resultado**: Exibe uma mensagem de confirmação quando a atualização é bem-sucedida.

4. Delete (Deletar Pessoa Física)

- **Descrição**: Permite que o usuário remova uma Pessoa Física do sistema, identificando-a pelo CPF.
- **Processo**: O CPF é solicitado, e, se encontrado, o registro é removido do banco de dados.
- **Resultado**: Exibe uma mensagem confirmando a remoção ou informa que a pessoa não foi encontrada.

CONCLUSÃO

O sistema desenvolvido para a gestão de uma clínica veterinária demonstrou ser uma solução eficiente para o gerenciamento de dados relacionados a pessoas físicas, ong, animais, espécies, vacinas, veterinários e consultas. A aplicação do SQLAlchemy como ferramenta ORM facilitou a implementação das operações de CRUD, o mapeamento objeto-relacional e o estabelecimento de relações complexas entre as entidades.

A interface gráfica, construída com PyQt5, contribuiu para a usabilidade do sistema, permitindo que os usuários realizassem operações de forma intuitiva e acessível. Embora o objetivo inicial não fosse gerenciar o histórico completo de consultas e tratamentos, o sistema foi projetado de forma que permita futuras expansões, incorporando novas funcionalidades sem a necessidade de grandes alterações na base existente.

Portanto, o sistema ainda pode evoluir a unificação das entidades Pessoa Física e ONG em uma única entidade chamada Dono. A análise dos resultados mostrou que o sistema desenvolvido não apenas facilita o acesso e a atualização de dados, mas também garante a integridade das informações armazenadas. Como direções futuras, sugere-se a implementação de funcionalidades adicionais, como o agendamento de consultas e a integração com sistemas de notificação, que poderiam aprimorar ainda mais a experiência do usuário e a eficiência operacional da clínica.

REFERÊNCIAS

- SQLAlchemy. SQLAlchemy: Database Toolkit for Python. Disponível em: https://www.sqlalchemy.org/.
- VAN ROSSEM, H. Python: Programming Language. Disponível em: https://www.python.org/.
- SQLite. SQLite: Database Engine. Disponível em: https://www.sqlite.org/. Acesso em: 16 set. 2024.
- PYTHON SOFTWARE FOUNDATION. os — Miscellaneous operating system interfaces. Disponível em: https://docs.python.org/3/library/os.html.
- SQLAlchemy. SQLAlchemy ORM Documentation. Disponível em: https://docs.sqlalchemy.org/en/20/orm/.
- SQLAlchemy. SQLAlchemy Core: SQL Expression Language. Disponível em: https://docs.sqlalchemy.org/en/20/core/metadata.html.
- SQLAlchemy. SQLAlchemy ORM Documentation: Relationships and Loading Strategies. Disponível em: https://docs.sqlalchemy.org/en/20/orm/relationships.html.
- DUNN, J. J. CRUD Operations: The Basics of Database Interaction. Disponível em: https://www.davidkinnard.com/blog/crud-operations-basics/.
- POPA, L. Python e Banco de Dados Relacionais. Disponível em: https://realpython.com/python-sqlalchemy/.
- ESSLAM, A. et al. SQLAlchemy ORM documentation. SQLAlchemy Project. Disponível em: https://docs.sqlalchemy.org/en/14/orm/.
- UNESP – Universidade Estadual Paulista "Júlio de Mesquita Filho". Listas Opcionais de Documentos. Disponível em: https://www.ict.unesp.br/Home/sobreoict/biblioteca/dn-listas-opcionais.pdf.
- PYQT. PyQt Documentation. Disponível em: https://www.riverbankcomputing.com/static/Docs/PyQt5/.
- SUMMERFIELD, M. Rapid GUI Programming with Python and Qt. Prentice Hall, 2007.
- KLEPMANN, M. Designing Data-Intensive Applications. O'Reilly Media, 2017.
- RAMALHO, L. Fluent Python. O'Reilly Media, 2015.