PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
CIÊNCIA DA COMPUTAÇÃO

TRABALHO DISCENTE EFETIVO 3
ARQUITETURA DE BANCO DE DADOS

RAFAEL CALIXTO MALUF

CURITIBA 2024

RAFAEL CALIXTO MALUF

TRABALHO DISCENTE EFETIVO 3
ARQUITETURA DE BANCO DE DADOS

Relatório técnico apresentado como documentação para entrega de trabalho discente efetivo 3 para a matéria Arquitetura de Banco de Dados

CURITIBA 2024

RESUMO

O objetivo deste trabalho é desenvolver um sistema de gerenciamento de dados utilizando a abordagem de mapeamento objeto-relacional (ORM) com SQLAlchemy em Python. O sistema visa a implementação e integração de operações CRUD (Create, Read, Update, Delete) para um banco de dados MySQL.
Inicialmente, o sistema foi projetado com uma interface de linha de comando, mas foi estendido para incluir uma interface gráfica utilizando o PyQt5, facilitando ainda mais a interação do usuário com as funcionalidades CRUD de forma intuitiva e visual.
A aplicação utiliza SQLAlchemy para criar e gerenciar a estrutura do banco de dados, permitindo ao usuário realizar operações básicas e avançadas de forma interativa. A implementação deste sistema demonstra a aplicação prática dos conceitos de ORM e CRUD em um ambiente de banco de dados relacional, destacando a capacidade do Python e SQLAlchemy na construção de soluções robustas para o gerenciamento de dados, agora complementada por uma interface gráfica.
Este trabalho inclui três melhorias principais: implementação de consultas avançadas usando álgebra relacional, controle de acesso baseado em funções de usuário, e gerenciamento de transações com checkpoints persistentes e rollback.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A crescente complexidade dos sistemas de informação e a necessidade de uma gestão eficiente dos dados têm impulsionado o desenvolvimento de soluções que integrem práticas avançadas de gerenciamento de banco de dados. No contexto atual, a manipulação de dados e a realização de operações básicas, como criação, leitura, atualização e exclusão (CRUD), são fundamentais para o funcionamento eficaz de muitas aplicações.

Este trabalho se propõe a explorar a implementação de um sistema de gerenciamento de dados utilizando a abordagem de mapeamento objeto-relacional (ORM) com o SQLAlchemy, uma biblioteca Python amplamente reconhecida por sua capacidade de interagir com bancos de dados relacionais de forma intuitiva e eficiente. O SQLAlchemy fornece uma abstração que permite aos desenvolvedores trabalharem com dados em termos de objetos Python, facilitando a integração com o banco de dados e a manipulação dos dados de maneira mais natural.

O sistema desenvolvido neste trabalho é projetado para gerenciar um banco de dados que modela quatro entidades principais: Clientes, Apólices de Seguro, Apartamentos e Acidentes. O objetivo inicial era fornecer uma interface interativa de linha de comando que permitisse ao usuário realizar operações CRUD para cada uma dessas entidades. No entanto, o sistema evoluiu para incluir uma interface gráfica construída com PyQt5, oferecendo uma interação mais amigável e acessível.

Além de demonstrar a aplicação prática dos conceitos de ORM e CRUD, este projeto destaca a utilidade do SQLAlchemy e PyQt5 na construção de sistemas robustos e flexíveis para o gerenciamento de dados, contribuindo para a eficiência e a eficácia dos processos de manipulação de informações em ambientes de banco de dados relacionais.

Demonstra também funcionalidades que põem em pratica conceitos de álgebra relacional, usando operadores para consultas avançadas, controle de acesso, usando roles para controlar permissões e gerenciamento de transações, usando rollbacks para garantir uma integridade dos dados.

2 DESCRIÇÃO DE PROJETO

Para iniciar o projeto, devemos antes fazer a descrição textual e conceitual para entender a fundo o banco de dados.

Descrição Textual:

O projeto desenvolvido é um sistema de gerenciamento de seguros, com foco na administração de clientes, apólices de seguro, apartamentos segurados e registros de acidentes. O objetivo principal é fornecer uma solução que permita o gerenciamento eficiente e organizado das informações relacionadas a seguros.

O banco de dados é estruturado para suportar as seguintes funcionalidades:

Cliente: Armazena dados básicos dos clientes, incluindo CPF, nome, contato, data de nascimento e sexo

.

Apólice: Registra as apólices de seguro, com informações sobre o número do seguro, data de início, valor mensal, cobertura e a associação com um cliente específico.

Apartamento: Contém informações sobre os apartamentos segurados, como logradouro, cidade, metragem, número do seguro associado, valor de mercado e número de moradores.

Acidente: Guarda os detalhes dos acidentes ocorridos, incluindo a data, quantidade de acidentes, logradouro do apartamento onde ocorreu o acidente, descrição e número de envolvidos.

Modelo Conceitual:

O modelo conceitual representa visualmente a estrutura do banco de dados e os relacionamentos entre as entidades descritas. Ele é fundamental para garantir que todas as partes do sistema estejam corretamente interligadas e para facilitar o entendimento e a implementação do banco de dados.

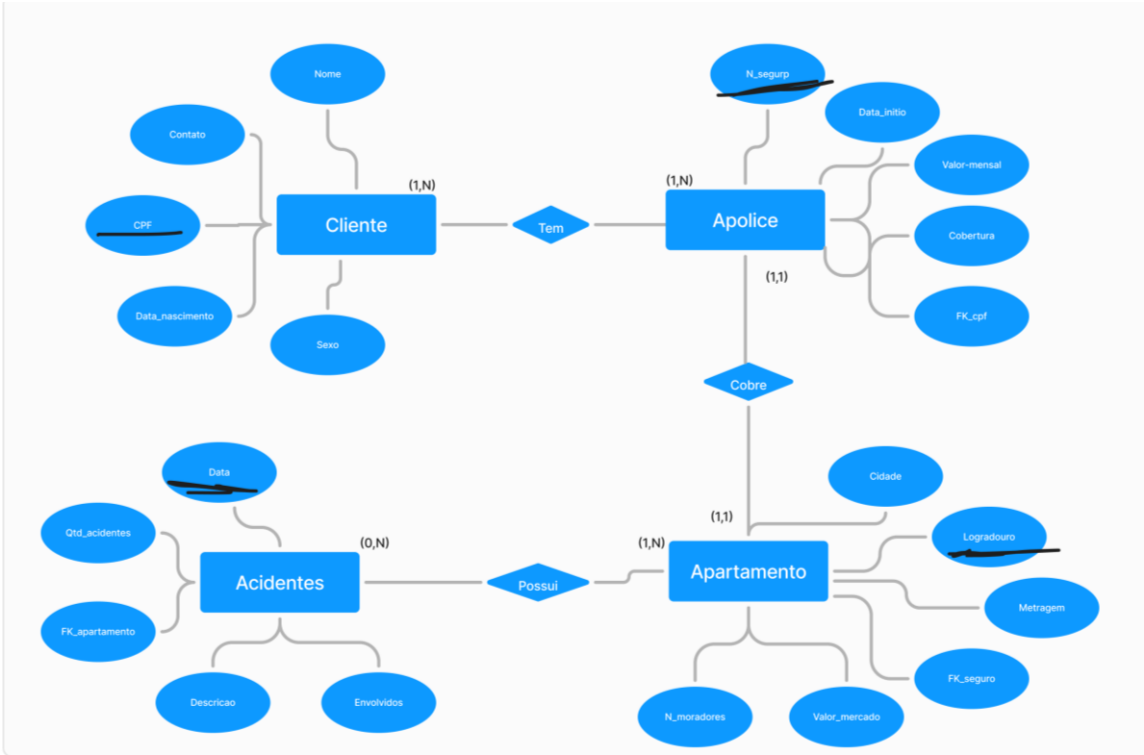

Figura 1 – Diagrama Entidade-Relacionamento
Fonte: Elaboração do autor, 2024.

Conforme mostra a Figura 1, no modelo conceitual do banco de dados, cada elemento possui uma representação visual específica para facilitar a compreensão das estruturas e dos relacionamentos. As entidades são representadas por retângulos com seu nome escrito dentro, e são ligadas nos atributos, representados por círculos, também com seu nome dentro, com uma exceção, sendo essa a Primary Key, que tem seu nome sublinhado para diferenciá-la dos demais. As relações entre entidades são representadas por losangos ligados em ambas as entidades, com o modo de relacionamento escrito entre elas. A cardinalidade da relação é indicada por símbolos nas extremidades das entidades, no formato de (x, n).

3 RESULTADOS

O desenvolvimento do sistema de gerenciamento de dados para o projeto de mapeamento objeto-relacional utilizando SQLAlchemy foi concluído com sucesso, atingindo todos os objetivos propostos. A seguir, são apresentados os principais resultados obtidos:

Criação e Configuração do Banco de Dados:

O banco de dados foi configurado de acordo com o modelo conceitual definido. As tabelas para as entidades Cliente, Apólice, Apartamento, e Acidente foram criadas e estão funcionando corretamente. A configuração das chaves primárias e estrangeiras foi realizada com precisão, garantindo a integridade referencial entre as tabelas. A imagem (Tabela 1) a seguir mostra as tabelas logo após serem criadas:

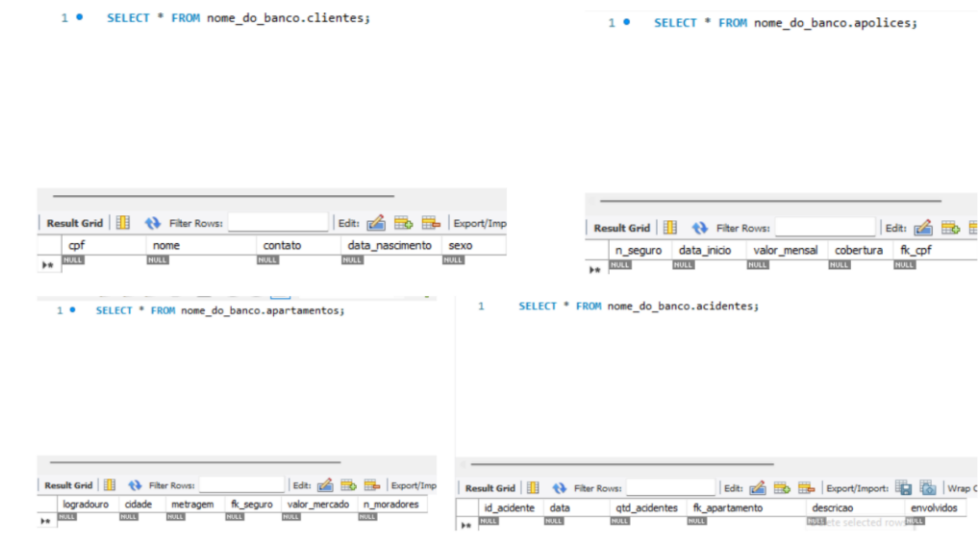

Tabela 1 – Tabela criadas pelo programa em python
Fonte: Elaboração do autor, 2024.

Implementação do Modelo Objeto-Relacional:

Utilizando o SQLAlchemy, foram definidos os modelos de dados que correspondem às tabelas do banco de dados. As classes Cliente, Apolice, Apartamento, e Acidente foram implementadas com suas respectivas colunas e relacionamentos, refletindo fielmente o diagrama ER criado. A Figura 2 contempla segmentos do código para demonstrar o funcionamento do mesmo:

Figura 2 – Código utilizado para criação das tabelas e funções CRUDs
Fonte: Elaboração do autor, 2024.

Interface de Usuário:

Uma interface de linha de comando foi criada para permitir a interação com o sistema. O menu principal e os menus de operações específicas para cada entidade foram implementados, oferecendo uma experiência de usuário intuitiva para realizar as operações CRUD. O sistema orienta o usuário por meio de instruções claras e valida as entradas fornecidas. O menu mostra perguntas claras que levam o usuário a escolherem o trecho que querem seguir para tomar a decisão que desejam, como mostrado a seguir pela Figura 3:

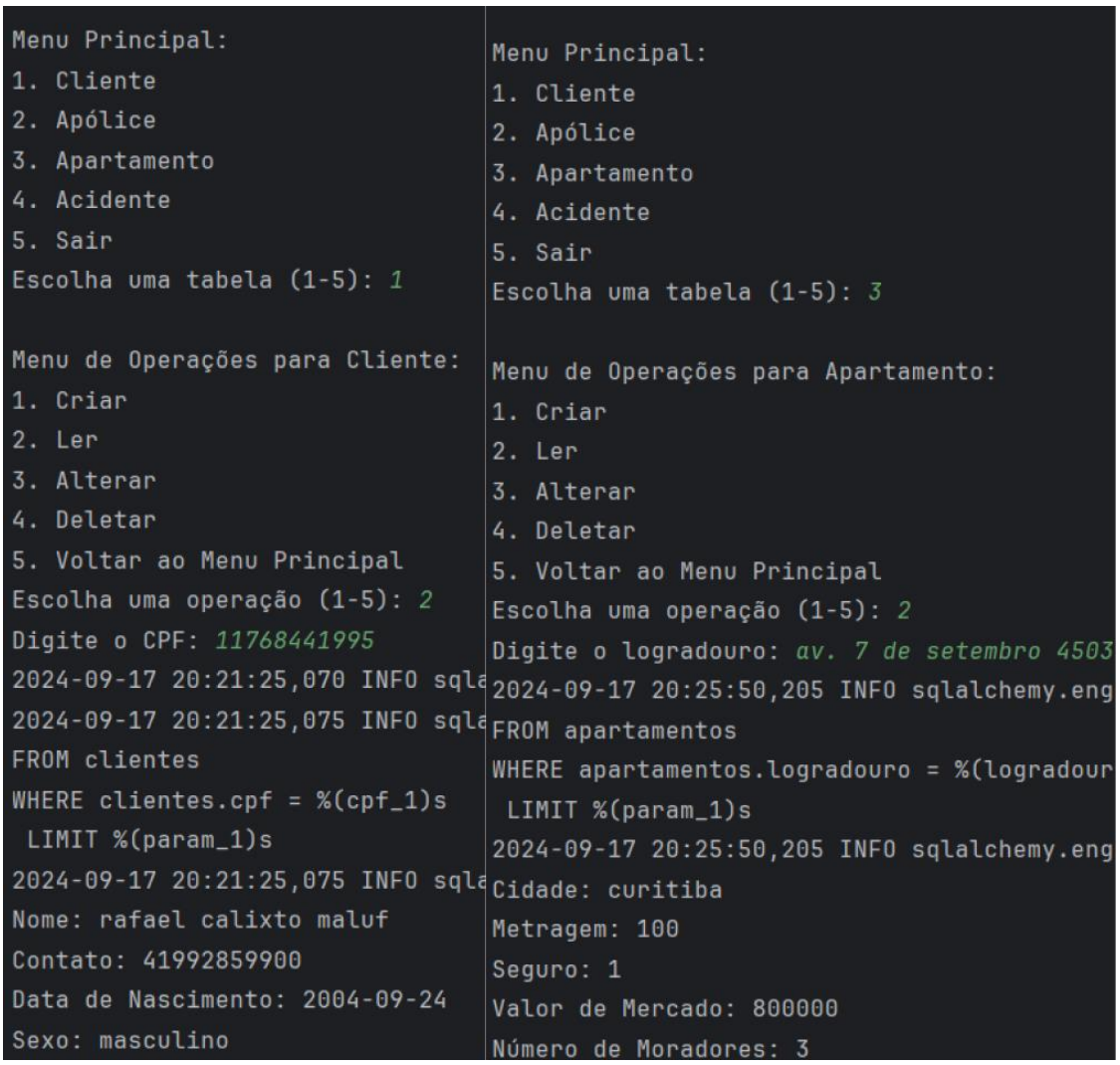

Figura 3 – Menu interativo utilizado para interagir com o banco de dados
Fonte: Elaboração do autor, 2024

Funcionalidades CRUD (Create, Read, Update, Delete):

As operações básicas de CRUD foram implementadas para todas as entidades. As funções para criar, ler, atualizar e deletar registros em cada tabela foram desenvolvidas e testadas com sucesso. O sistema permite interagir com o banco de dados de maneira eficiente, realizando operações conforme necessário.

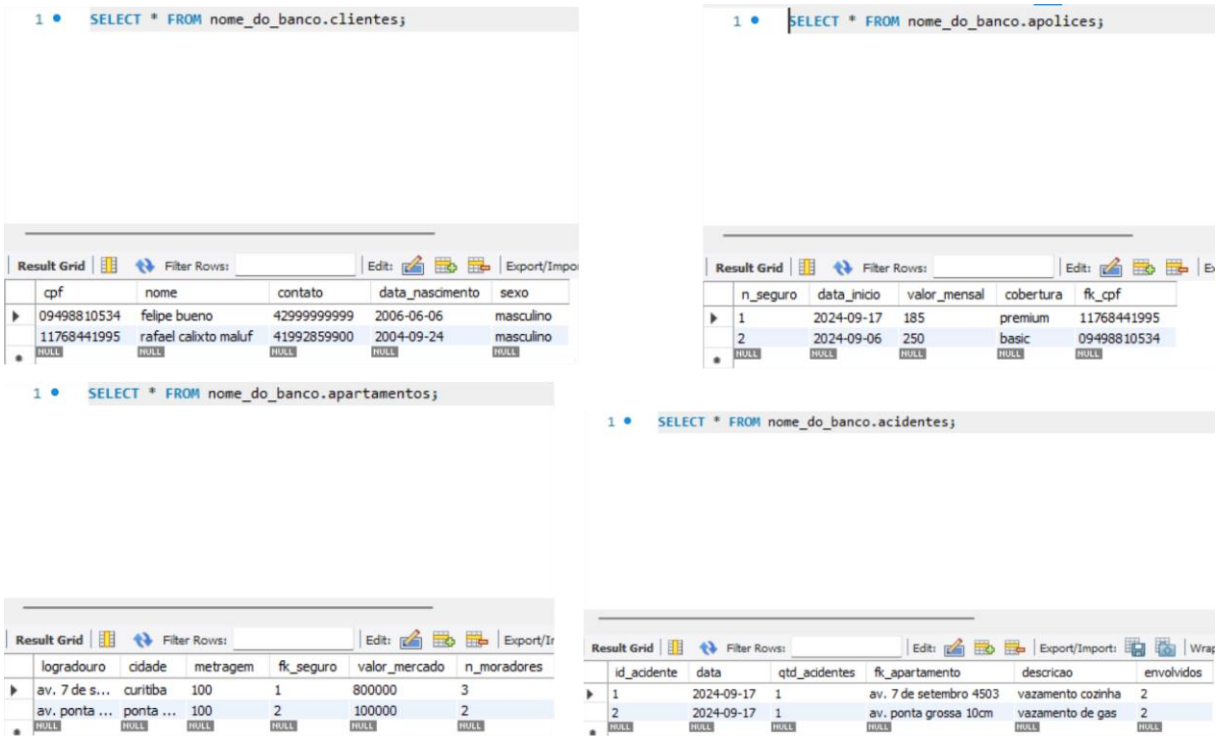

Tabela 2 – Tabelas modificadas pelo menu interativo
Fonte: Elaboração do autor, 2024.

A tabela 2, ilustra as tabelas após as funções CRUDs e mudanças feitas pelo usuário, mostrando o banco de dados com algumas linhas e colunas já preenchidas.

Após o desenvolvimento da interface em linha de comando, o sistema foi expandido para uma interface gráfica utilizando a biblioteca PyQt5. A nova interface facilita a interação do usuário, apresentando menus suspensos para a seleção de entidades (Cliente, Apólice, Apartamento, Acidente) e operações CRUD. Além disso, os campos de inserção são exibidos de acordo com a operação escolhida, sendo validados antes de enviar as informações ao banco de dados.

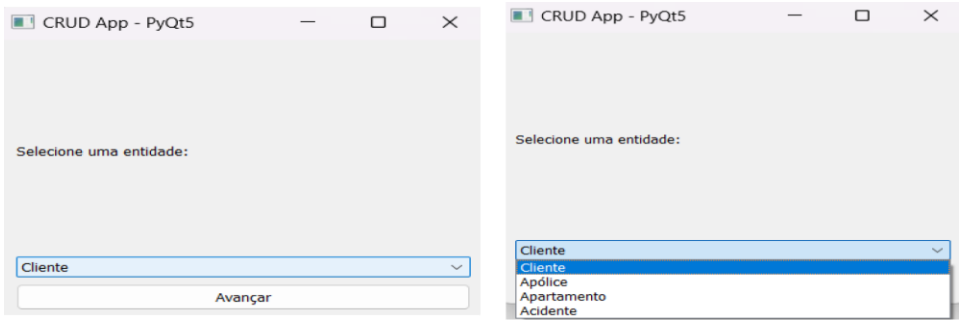

Figura 4 – Main Window Interface Gráfica
Fonte: Elaboração do autor, 2024.

A interface gráfica foi projetada para ser intuitiva, com botões claramente estilizados para ações de "Executar" e "Voltar", além de margens e espaçamentos adequados para melhorar a usabilidade. O uso do PyQt5 permite uma interação mais natural, reduzindo a curva de aprendizado e tornando o sistema mais acessível a usuários que não estejam familiarizados com interfaces de linha de comando.

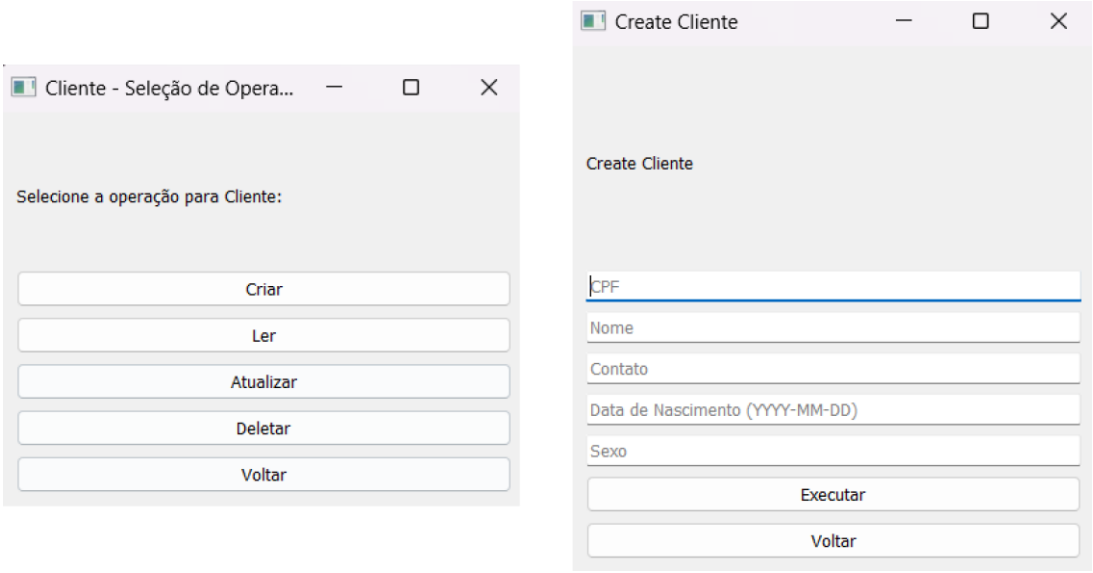

Figura 5 – Funções CRUD Interface Gráfica
Fonte: Elaboração do autor, 2024.

As operações CRUD foram totalmente integradas ao PyQt5, preservando a funcionalidade robusta que já havia sido desenvolvida na versão em linha de comando. A interface gráfica foi testada com sucesso para todas as entidades, mostrando que é uma evolução eficaz da interação com o sistema.

O projeto foi otimizado com a implementação de um sistema robusto, funcional e integrado, que atende aos requisitos definidos e oferece uma solução eficiente para o gerenciamento de dados.

Após a implementação da interface gráfica, foram adicionadas novas funcionalidades, como consultas avançadas, controle de acesso e gerenciamento de transação além de melhorias no estilo da interface.

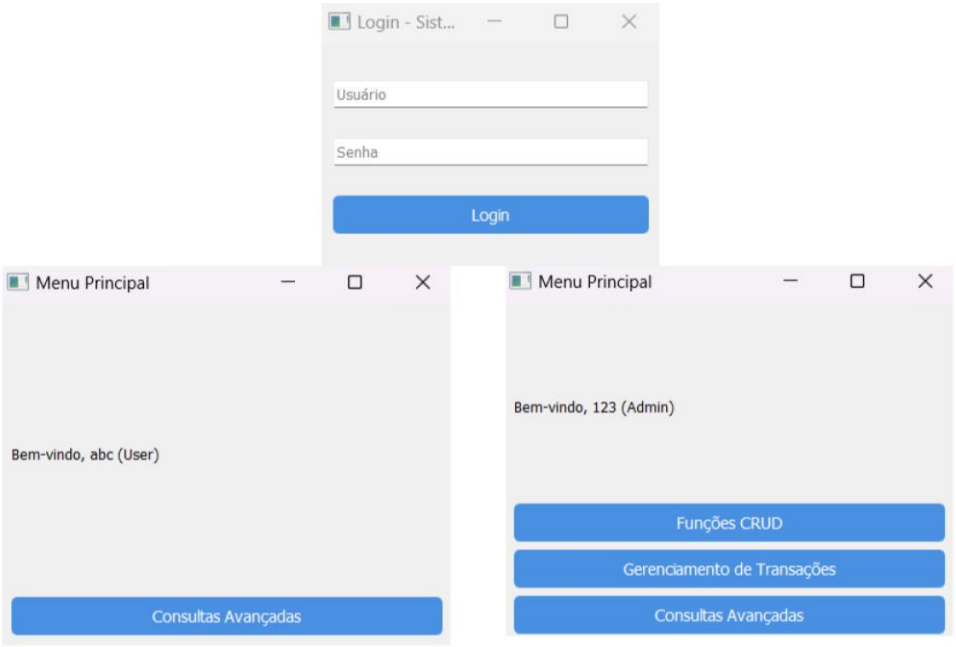

Figura 6 – Funções de Controle de Acesso
Fonte: Elaboração do autor, 2024.

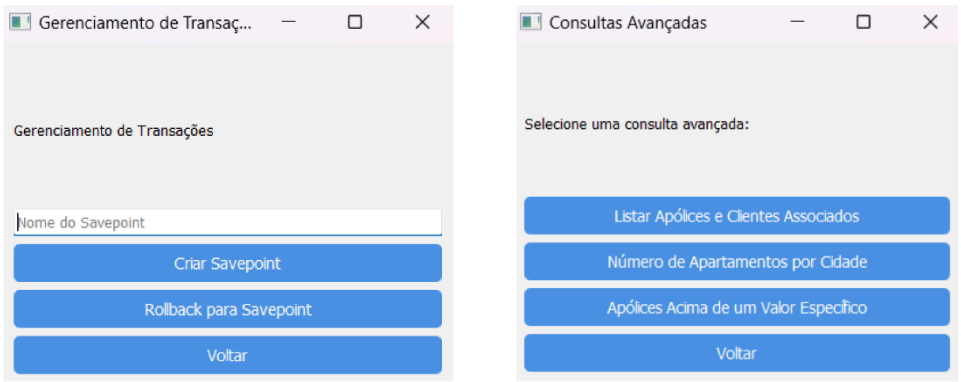

Figura 7 – Funções de operadores algébricos e controle de transações
Fonte: Elaboração do autor, 2024.

As melhorias implementadas aumentaram a segurança, a flexibilidade e a robustez do sistema, demonstrando o uso prático de conceitos como álgebra relacional, controle de acesso e gerenciamento de transações no desenvolvimento de sistemas baseados em bancos de dados.

REFERÊNCIAS

SQLAlchemy Documentation. (2024). SQLAlchemy ORM Documentation. Disponível em: https://docs.sqlalchemy.org/. Acesso em: 13 set. 2024.

Elmasri, R., & Navathe, S. B. (2016). Fundamentals of Database Systems. Pearson.

Connolly, T., & Begg, C. (2014). Database Systems: A Practical Approach to Design, Implementation, and Management. Pearson.

Beaulieu, A. (2009). Learning SQL: Master SQL Fundamentals. O'Reilly Media.

Summerfield, M. (2018). *Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming*. Addison-Wesley Professional.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. Database System Concepts. 7ª ed. New York: McGraw-Hill Education, 2020. Capítulo: Relational Algebra and Calculus. Disponível em: https://www.db-book.com. Acesso em: 12 nov. 2024

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Role-Based Access Control. Gaithersburg: NIST, 2022. Disponível em: https://csrc.nist.gov/publications/detail/sp/800-162/final. Acesso em: 12 nov. 2024.

GRAY, Jim; REUTER, Andreas. Transaction Processing: Concepts and Techniques. 1ª ed. San Francisco: Morgan Kaufmann, 1992. Capítulo: Checkpointing and Recovery.
.