

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE
SÃO PAULO**

Raul Dias Pereira

Sistema de Gerenciamento de Usuários

**CAMPOS DO JORDÃO
2025**

RESUMO

Este trabalho apresenta o projeto e a implementação de um Sistema de Gerenciamento de Usuários, desenvolvido como uma aplicação desktop multiplataforma. Utilizando a linguagem de programação C++ em conjunto com o framework Qt 6 e a IDE Qt Creator, o sistema foi estruturado sob o paradigma da Programação Orientada a Objetos (POO). A metodologia focou no desenvolvimento incremental, partindo de uma tela de autenticação segura para um painel administrativo com funcionalidades completas de CRUD (Create, Read, Update, Delete). O resultado é uma aplicação funcional e robusta, que permite a um administrador gerenciar um conjunto de usuários de forma intuitiva, validando na prática os conceitos teóricos de engenharia de software e desenvolvimento de interfaces gráficas.

Palavras-Chave: C++; Qt Framework; Programação Orientada a Objetos; Sistema de Gerenciamento; Aplicação Desktop.

LISTA DE ILUSTRAÇÕES

FIGURA 1 – TELA DE AUTENTICAÇÃO DO SISTEMA.	11
FIGURA 2 – MENSAGEM DE ERRO PARA CREDENCIAIS INVÁLIDAS.	12
FIGURA 3 – PAINEL DO ADMINISTRADOR (DASHBOARD) COM A LISTA DE USUÁRIOS.	13
FIGURA 4 – JANELA PARA CADASTRO DE UM NOVO CLIENTE.	14
FIGURA 5 – DIÁLOGO DE CONFIRMAÇÃO PARA A EXCLUSÃO DE UM USUÁRIO.	15

SUMÁRIO

1 INTRODUÇÃO.....	5
1.1 Objetivos.....	5
1.2 Justificativa.....	5
1.3 Aspectos Metodológicos.....	6
1.4 Aporte Teórico.....	6
2 METODOLOGIA.....	7
2.1 Considerações iniciais.....	7
2.2 Ferramentas Utilizadas.....	7
2.3 Descrição do Projeto Desenvolvido.....	8
3 Resultados Obtidos.....	10
3.1 Autenticação e Acesso ao Sistema.....	10
4 CONCLUSÃO.....	16
REFERÊNCIAS BIBLIOGRÁFICAS.....	18

1 INTRODUÇÃO

Neste capítulo serão introduzidos todos os assuntos abordados por este documento. Pretende-se apresentar a motivação, os objetivos e a organização do texto.

1.1 Objetivos

Este trabalho tem por objetivo desenvolver um sistema desktop funcional para o gerenciamento de usuários, com controle de acesso para um perfil administrativo.

- Para a consecução deste objetivo foram estabelecidos os objetivos específicos:
- Implementar um sistema de autenticação seguro para o acesso do administrador;
- Desenvolver um painel administrativo para a visualização e manipulação de dados de usuários;
- Propor e implementar as funcionalidades de Criar, Ler, Atualizar e Excluir (CRUD) registros de usuários;
- Aplicar os conceitos da Programação Orientada a Objetos para modelar a arquitetura do sistema;
- Construir uma interface gráfica de usuário intuitiva utilizando o framework Qt.

1.2 Justificativa

A relevância deste trabalho se manifesta na aplicação prática de conceitos teóricos de engenharia de software e desenvolvimento de sistemas em um projeto concreto. A justificativa para a sua elaboração reside na necessidade de construir uma solução para o problema comum de gerenciamento de usuários, utilizando ferramentas de nível industrial como a linguagem C++ e o framework Qt, que são amplamente empregadas no desenvolvimento de aplicações de alto desempenho. O projeto serve, portanto, como uma ponte entre o conhecimento acadêmico e as

práticas do mercado de desenvolvimento de software.

1.3 Aspectos Metodológicos

O presente estudo fez uso de pesquisa de natureza bibliográfica, para o que remete ao levantamento do aporte teórico sobre C++, Programação Orientada a Objetos e o framework Qt. Para a parte prática, adotou-se uma metodologia de desenvolvimento de protótipo funcional, caracterizada pela construção incremental de funcionalidades, testes contínuos e refatoração do código para atender a novos requisitos.

1.4 Aporte Teórico

A pesquisa se fundamentou em três pilares teóricos principais: a Programação Orientada a Objetos (POO), com seus conceitos de encapsulamento, herança e polimorfismo; a arquitetura de software em camadas, que permitiu a separação entre a interface de usuário, a lógica de negócios e o gerenciamento de dados; e o paradigma de programação orientada a eventos, exemplificado pelo mecanismo de Sinais e Slots do Qt.

A estrutura deste documento está organizada da seguinte forma: a seção 2 apresenta a metodologia de desenvolvimento, detalhando as ferramentas utilizadas e a arquitetura do sistema proposto. A seção 3 expõe os resultados obtidos, ilustrados por capturas de tela da aplicação em funcionamento. A seção 4 apresenta a conclusão deste trabalho, com as considerações finais e sugestões para melhorias futuras. Por fim, a seção 5 lista as referências bibliográficas que serviram de base para esta pesquisa.

2 METODOLOGIA

Esta seção detalha os procedimentos metodológicos, as ferramentas e a arquitetura de software empregada no desenvolvimento do projeto. A abordagem adotada foi a de desenvolvimento de software com prototipagem de interface, permitindo a construção e validação de funcionalidades de forma incremental.

2.1 Considerações iniciais

O escopo inicial do projeto foi definido para a criação de um sistema desktop autocontido, focado na funcionalidade de gerenciamento de usuários por um perfil de administrador. A premissa era desenvolver um sistema que não apenas funcionasse, mas que também fosse construído sobre uma base de código bem estruturada, aplicando os princípios da Programação Orientada a Objetos para garantir a manutenibilidade e escalabilidade do software. O desenvolvimento partiu de um requisito fundamental de controle de acesso, evoluindo para um painel administrativo com as operações essenciais de manipulação de dados.

2.2 Ferramentas Utilizadas

A seleção de ferramentas foi orientada pela busca de robustez, desempenho e um ecossistema de desenvolvimento maduro. As seguintes tecnologias foram empregadas:

- Linguagem de Programação: C++ (padrão C++17), escolhida por seu alto desempenho e controle sobre os recursos do sistema.
- Framework: Qt (versão 6.x), utilizado por seu completo conjunto de bibliotecas para desenvolvimento de aplicações multiplataforma. O módulo Qt Widgets foi a base para a construção da interface gráfica.
- Ambiente de Desenvolvimento Integrado (IDE): Qt Creator, por sua integração nativa com o framework Qt e suas ferramentas, como o depurador e o designer de interfaces.

- Compilador: MinGW 64-bit (no ambiente Windows).
- Sistema de Build: qmake, o sistema de build padrão do Qt, utilizado para gerenciar as dependências e o processo de compilação do projeto.
- Ferramenta de Design de UI: Qt Designer, integrado ao Qt Creator, foi utilizado para a prototipagem e o design visual das janelas da aplicação (.ui files), permitindo uma separação clara entre o layout e a lógica de programação.

2.3 Descrição do Projeto Desenvolvido

A arquitetura do sistema foi projetada de forma modular e em camadas, distinguindo claramente as responsabilidades de cada componente, conforme os princípios da Programação Orientada a Objetos.

A estrutura é composta pelas seguintes classes principais:

- LoginDialog: Representa a camada de apresentação inicial e controle de acesso. Esta classe, herdada de QDialog, é a porta de entrada do sistema, responsável por capturar as credenciais do administrador e validá-las. Sua natureza modal garante que nenhuma outra parte do sistema seja acessível sem uma autenticação bem-sucedida.
- AdminDashboard: O painel de controle principal do administrador, herdado de QMainWindow. Ele exibe todos os usuários cadastrados em um componente QWidget, oferecendo uma visualização tabular dos dados. Esta tela centraliza as ações de gerenciamento, contendo os botões para iniciar os fluxos de "Adicionar Cliente", "Editar Cliente" e "Excluir Cliente".
- MainWindow: Reutilizada como uma janela de diálogo para cadastro e edição. Herdada de QDialog, esta classe possui uma lógica interna que a permite operar em dois modos: "Cadastro" (com os campos vazios) ou "Edição" (com os campos pré-carregados com os dados de um cliente existente).
- Cliente: É a classe de Modelo (Model) da aplicação. Ela encapsula os

atributos e o estado de um usuário (nome, e-mail, hash da senha), abstraindo a representação dos dados e garantindo a integridade através de seus métodos.

- GerenciadorDeDados: Atua como a camada de serviço e acesso aos dados. Implementada como um Singleton, esta classe centraliza a lógica de negócios, como a manipulação da lista de clientes (atuando como um banco de dados em memória) e o fornecimento de métodos para as operações CRUD que são chamados pelas janelas da interface.

A comunicação entre esses componentes é realizada primariamente pelo mecanismo de Sinais e Slots (Signals and Slots) do Qt. Este paradigma de programação orientada a eventos permite que objetos emitam sinais quando seu estado muda (ex: um botão é clicado), e outros objetos que possuem slots compatíveis podem se conectar a esses sinais para reagir ao evento. Isso resulta em um código mais limpo, desacoplado e de fácil manutenção.

3 Resultados Obtidos

O desenvolvimento do projeto, seguindo a metodologia descrita, culminou na criação de uma aplicação desktop plenamente funcional. O sistema atende a todos os objetivos específicos delineados, oferecendo uma solução robusta para o gerenciamento de usuários por um perfil administrativo. O software se mostrou estável e responsivo durante os testes de funcionalidade.

As seções a seguir apresentam as principais interfaces do sistema, demonstrando o fluxo de interação do usuário e as funcionalidades implementadas.

3.1 Autenticação e Acesso ao Sistema

A porta de entrada do sistema é a tela de autenticação (Figura 1), projetada para ser simples e direta, solicitando as credenciais do administrador para garantir o acesso seguro ao painel de controle.

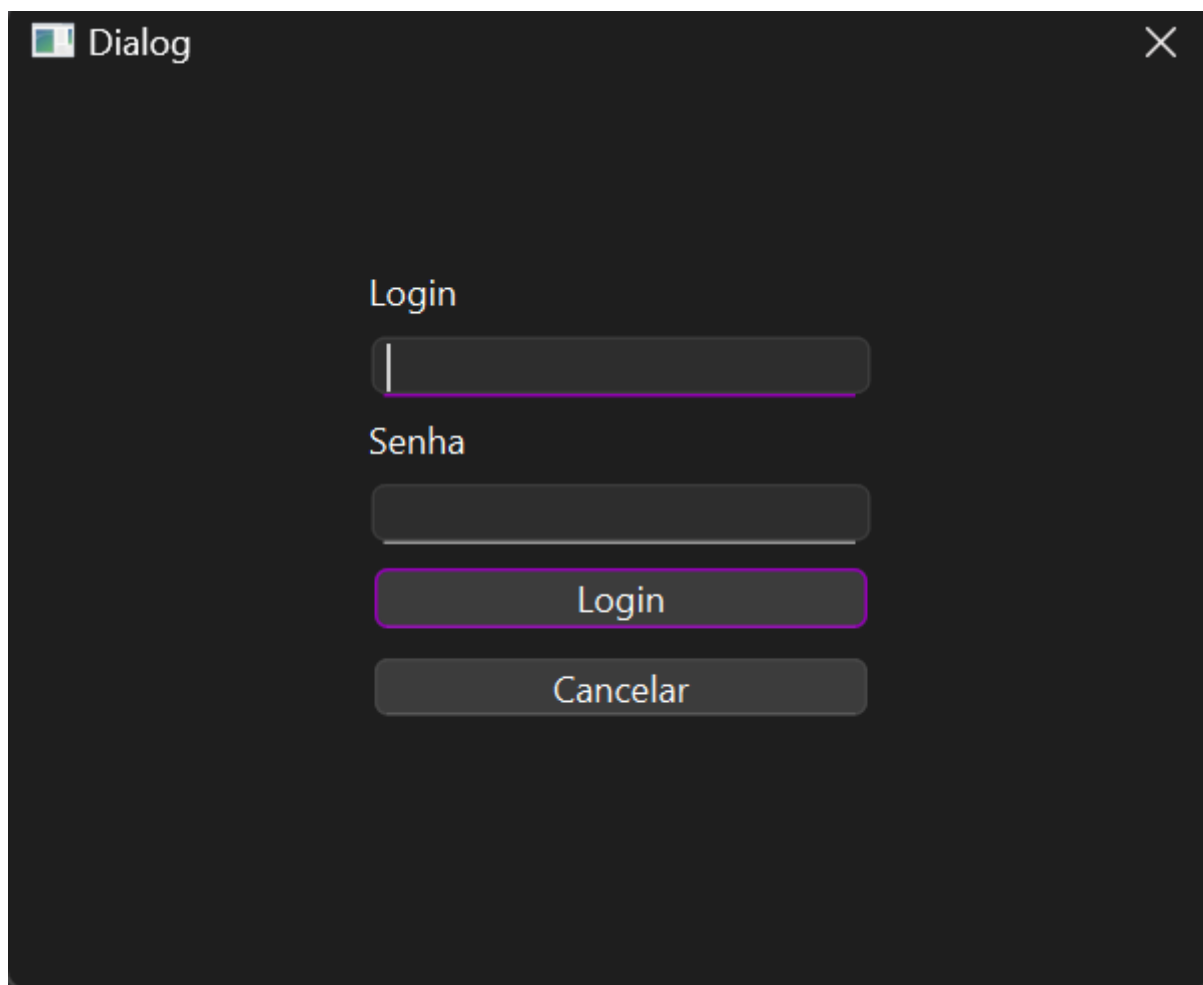


Figura 1 – Tela de autenticação do sistema.

O sistema valida as credenciais inseridas em tempo real. Em caso de falha na autenticação, seja por usuário ou senha incorretos, uma mensagem de erro informativa é exibida ao usuário em uma caixa de diálogo modal, impedindo o acesso não autorizado ao sistema, conforme ilustrado na Figura 2.

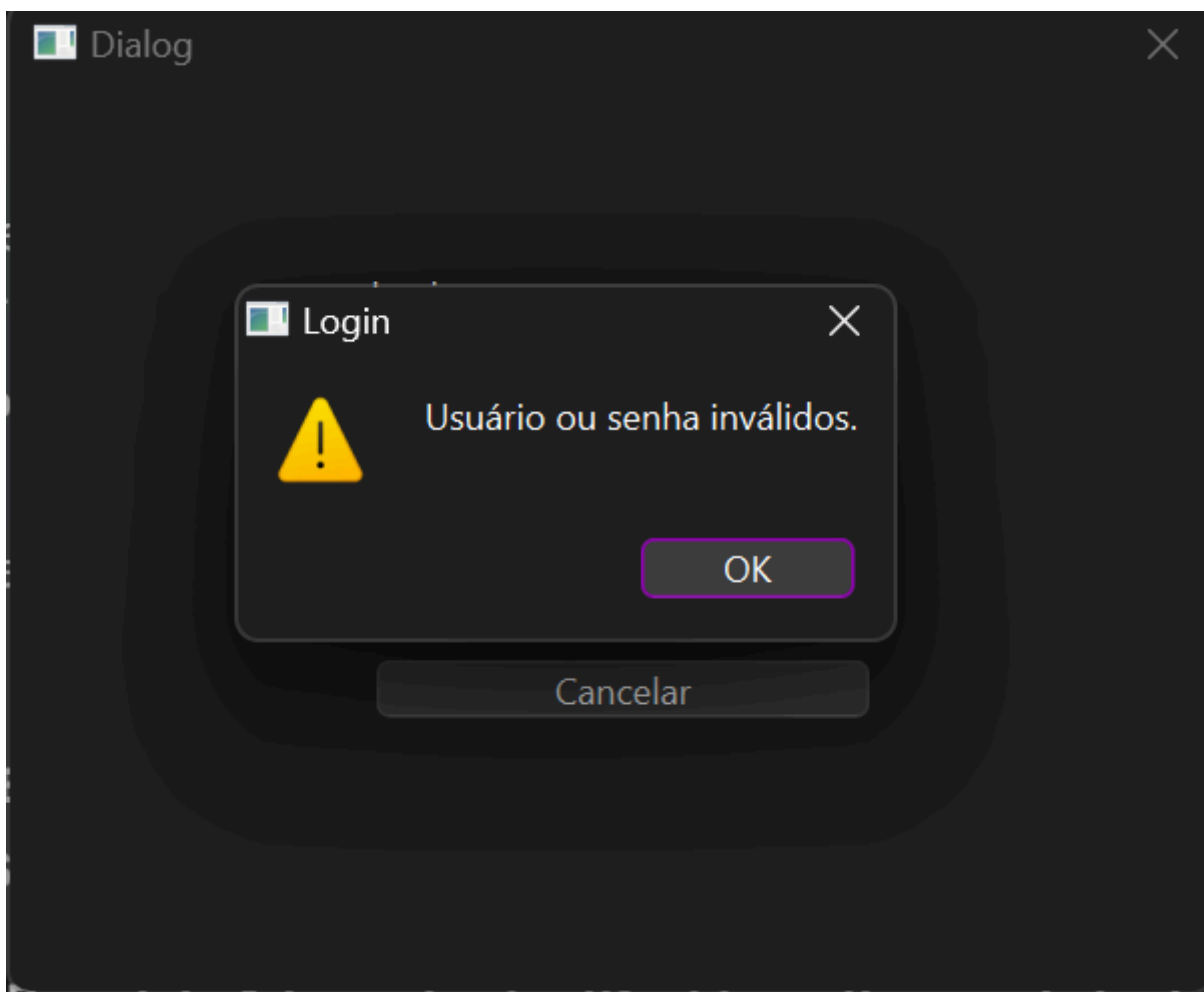


Figura 2 – Mensagem de erro para credenciais inválidas.

Após a autenticação bem-sucedida, o administrador é direcionado ao painel de controle principal (Figura 3). Esta tela centraliza as informações, exibindo a lista de usuários previamente cadastrados em um componente `QTableWidget`. A tabela oferece uma visualização clara e organizada dos dados. Além disso, a tela provê acesso direto às funcionalidades de gerenciamento através dos botões de ação: "Adicionar Cliente", "Editar Cliente" e "Excluir Cliente".

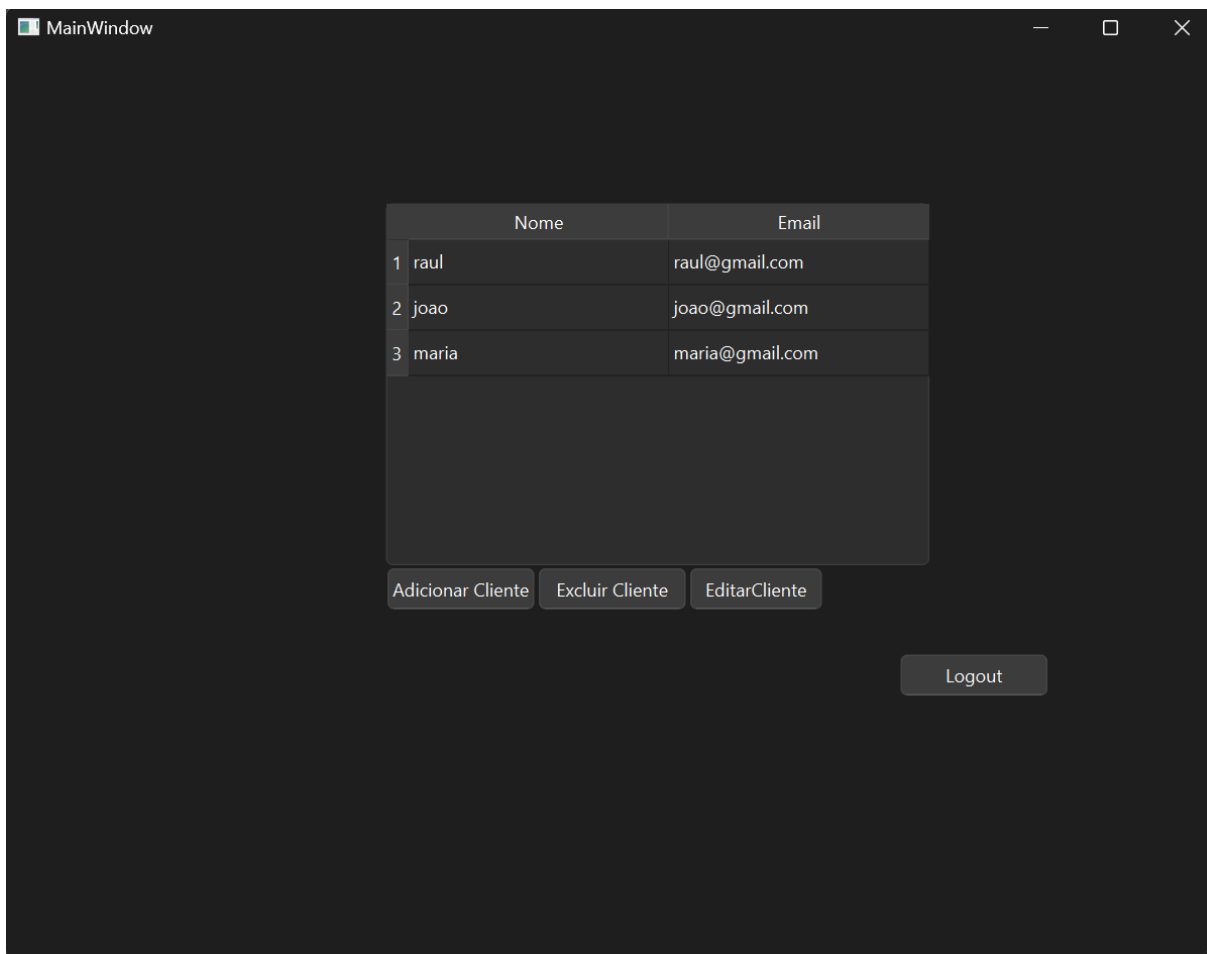
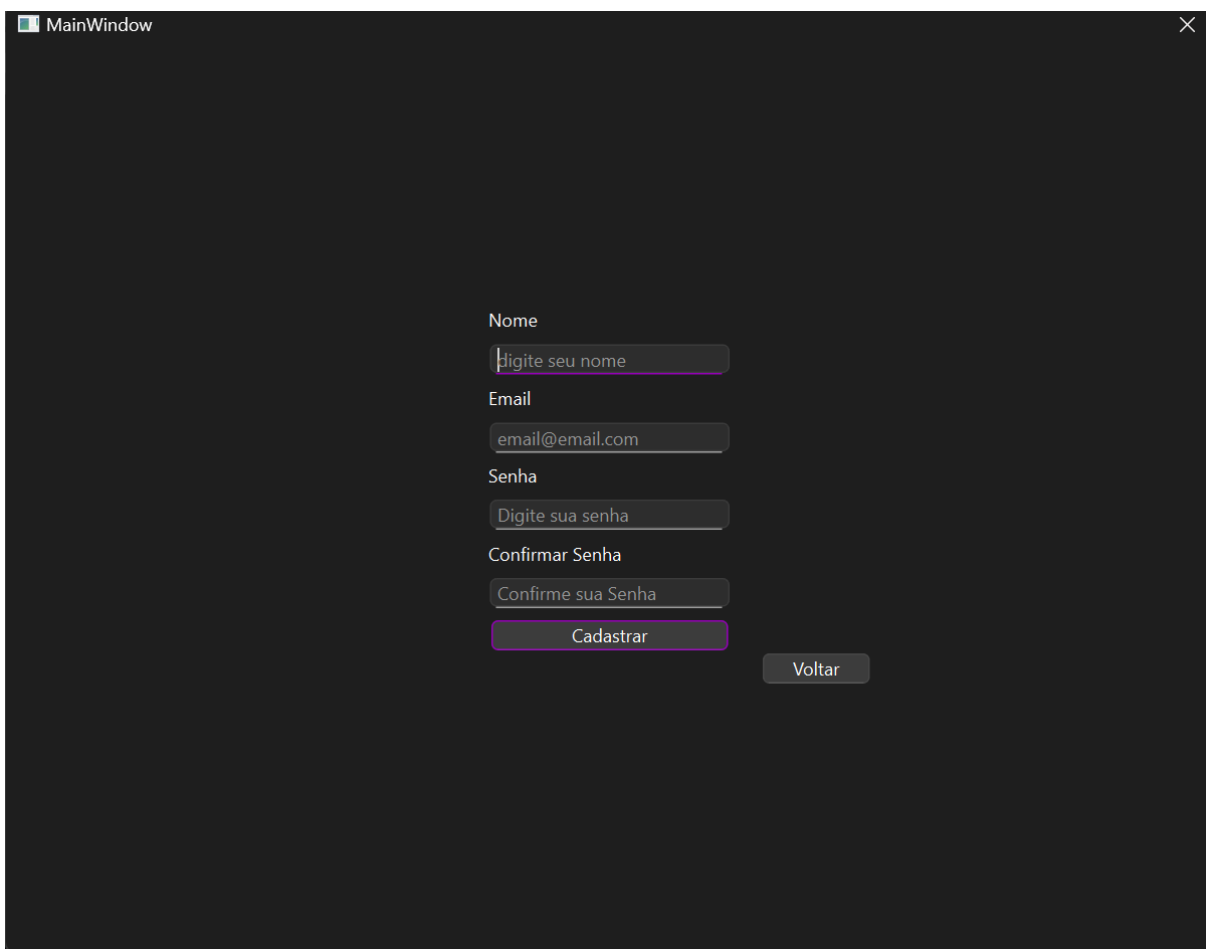


Figura 3 – Painel do Administrador (Dashboard) com a lista de usuários.

O sistema implementa o ciclo completo de gerenciamento de dados (CRUD). Ao acionar a função “Adicionar Cliente”, uma nova janela modal é apresentada (Figura 4), contendo o formulário para a inserção dos dados de um novo usuário.



The image shows a dark-themed user interface window titled "MainWindow". It contains a registration form with the following elements:

- Nome**: A text input field with the placeholder text "Digite seu nome".
- Email**: A text input field with the placeholder text "email@email.com".
- Senha**: A text input field with the placeholder text "Digite sua senha".
- Confirmar Senha**: A text input field with the placeholder text "Confirme sua Senha".
- Cadastrar**: A button with a purple border, positioned below the "Confirmar Senha" field.
- Voltar**: A button positioned to the right of the "Cadastrar" button.

Figura 4 – Janela para cadastro de um novo cliente.

Para evitar a remoção acidental de registros, a operação de exclusão implementa uma etapa de confirmação (Figura 5). Uma caixa de diálogo solicita que o administrador confirme sua intenção antes que o registro seja permanentemente removido do sistema, aumentando a segurança e a usabilidade da aplicação.

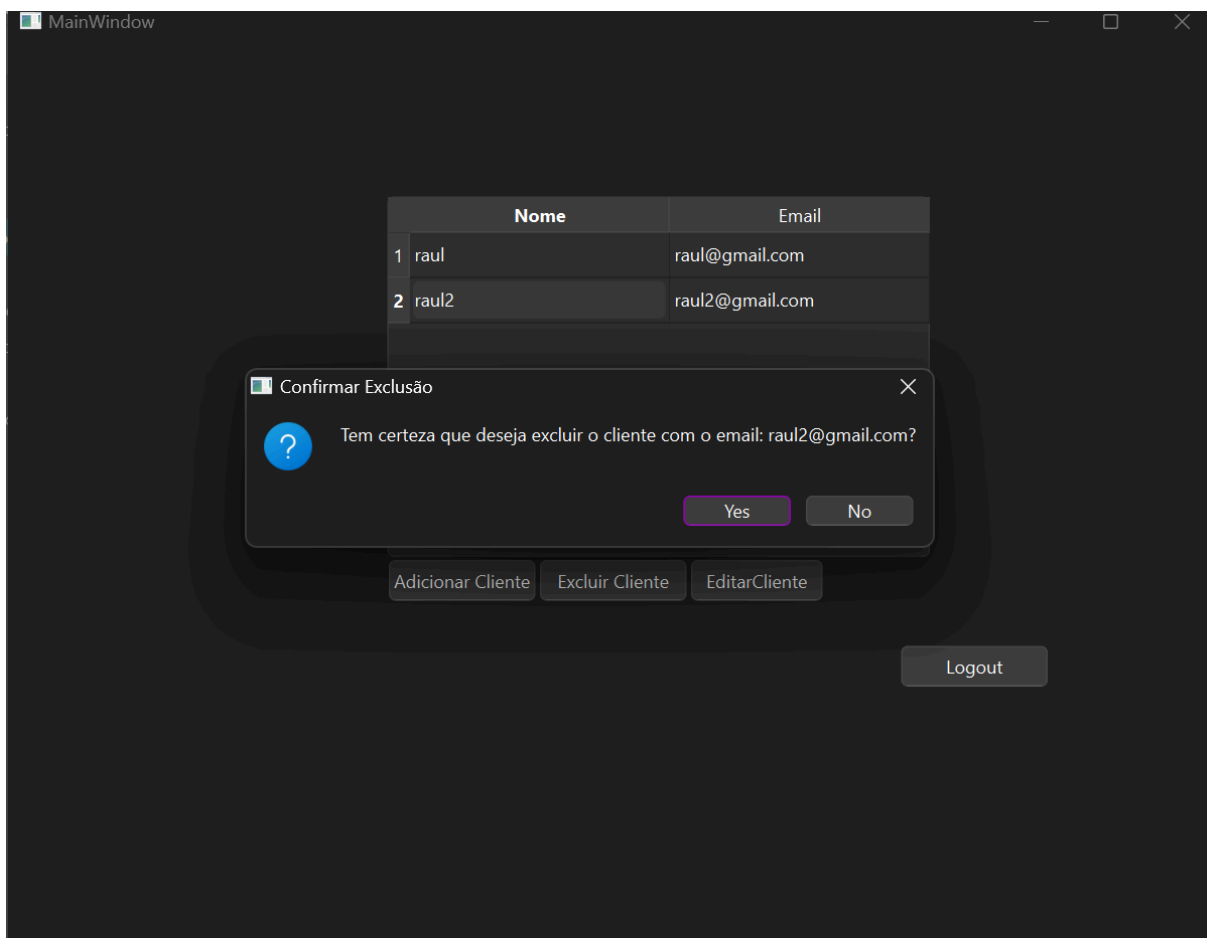


Figura 5 – Diálogo de confirmação para a exclusão de um usuário.

4 CONCLUSÃO

O presente trabalho atingiu com êxito seu objetivo principal: o desenvolvimento de uma aplicação desktop funcional e robusta para o gerenciamento de usuários. Todos os objetivos específicos, desde a implementação de um sistema de autenticação seguro até a disponibilização de funcionalidades CRUD (Criar, Ler, Atualizar e Excluir) completas em um painel administrativo, foram plenamente implementados e validados por meio de testes funcionais.

A aplicação prática dos conceitos de Programação Orientada a Objetos e da arquitetura em camadas resultou em um código-fonte organizado, modular e de fácil manutenção, demonstrando a eficácia dessas metodologias no desenvolvimento de software de qualidade. A utilização da linguagem C++ e do framework Qt se mostrou uma escolha acertada, provendo o desempenho e os recursos necessários para a construção de uma aplicação estável e com uma interface de usuário responsiva.

A execução do projeto proporcionou um aprendizado significativo que transcende o resultado final do software. Além do aspecto técnico, que solidificou o conhecimento em C++ e Qt, o processo de desenvolvimento iterativo — incluindo as fases de design, implementação, depuração de erros e refatoração de código — ofereceu uma valiosa experiência sobre o ciclo de vida real do desenvolvimento de software, ressaltando a importância do planejamento e da capacidade de adaptação a novos requisitos.

Embora o sistema atual seja completo dentro de seu escopo definido, há um vasto campo para evoluções e melhorias futuras. As seguintes sugestões são indicadas para dar continuidade e aprimorar o projeto:

- **Persistência de Dados com Banco de Dados:** A melhoria mais impactante seria substituir o gerenciador de dados em memória por um sistema de banco de dados real. A utilização do SQLite é recomendada por sua simplicidade de integração com o Qt e por não exigir um servidor dedicado, garantindo que os dados sejam salvos permanentemente entre as sessões da aplicação.

- **Segurança Aprimorada:** Implementar a técnica de salting no processo de hash das senhas. Isso adicionaria uma camada extra e fundamental de segurança, protegendo os dados dos usuários contra ataques de dicionário e rainbow tables.
- **Diferentes Níveis de Acesso:** Expandir o sistema para incluir múltiplos perfis de usuário (ex: "Gerente", "Usuário Comum"), cada um com diferentes permissões. Isso tornaria o sistema mais flexível e aplicável a cenários de negócios mais complexos, onde nem todos os usuários podem ter acesso às funcionalidades administrativas.
- **Melhorias na Usabilidade (UI/UX):** Aprimorar o painel administrativo com funcionalidades de usabilidade, como um campo de busca para filtrar usuários por nome ou e-mail, paginação para lidar com grandes volumes de dados de forma eficiente e a capacidade de ordenar a tabela ao clicar nos cabeçalhos das colunas.
- **Relatórios e Exportação:** Desenvolver um módulo capaz de gerar relatórios simples ou de exportar a lista de usuários para formatos de arquivo padrão, como CSV ou PDF, uma funcionalidade comum e útil em sistemas de gerenciamento.

Em suma, o projeto cumpriu seu papel como uma ferramenta de aprendizado e validação de conhecimento, resultando em um produto de software sólido que serve como uma excelente base para futuras expansões e aprimoramentos.

REFERÊNCIAS BIBLIOGRÁFICAS

THE QT COMPANY. **Qt Development Tools**. Qt, 2025. Disponível em: <https://www.qt.io/product/development-tools>. Acesso em: 1 jul. 2025.