INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

TIFANY LUIZA DE JESUS MOREIRA

DESENVOLVIMENTO DE APLICAÇÃO “TO-DO-LIST” UTILIZANDO C++ E QT CREATOR

CAMPOS DO JORDÃO

2025

RESUMO

Este relatório descreve o desenvolvimento de uma aplicação de "To-Do List" simples, implementada em C++ utilizando o framework Qt. O objetivo principal foi criar uma ferramenta de gerenciamento de tarefas que, **embora inicialmente focada em operações em memória, foi aprimorada com a integração de persistência de dados em arquivo**, demonstrando as operações CRUD (Create, Read, Update, Delete) em uma interface gráfica de usuário. A aplicação permite aos usuários adicionar novas tarefas, visualizar uma lista de tarefas existentes, editar o texto de tarefas selecionadas e remover tarefas concluídas ou indesejadas. A escolha do Qt permitiu a criação de uma interface intuitiva e responsiva, utilizando componentes como QListWidget e QLineEdit, consolidando conceitos de programação orientada a objetos de forma prática e acessível.

Palavras-Chave: C++, Qt, To-Do List, CRUD, Aplicação Desktop, Gerenciamento de Tarefas.

SUMÁRIO

1 INTRODUÇÃO

No cenário contemporâneo do desenvolvimento de software, a capacidade de organizar e gerenciar informações é um pilar fundamental para a produtividade pessoal e a eficiência operacional. Aplicações de gerenciamento de tarefas, conhecidas como "To-Do Lists", exemplificam essa necessidade ao proverem uma estrutura para o acompanhamento de atividades. Este documento detalha a concepção e a implementação de uma aplicação de "To-Do List" desenvolvida integralmente em C++, fazendo uso do robusto e versátil framework Qt. A escolha por uma solução **inicialmente baseada em memória, sem a complexidade adicional de um banco de dados relacional, foi deliberada para focar na essência das operações de manipulação de dados em uma interface gráfica de usuário, sendo posteriormente estendida para incluir persistência em arquivo.**

O projeto "To-Do List Simples" foi arquitetado com o propósito de oferecer uma plataforma intuitiva para as operações elementares de um sistema CRUD: a criação de novas tarefas, a visualização de uma lista consolidada de atividades, a modificação de entradas existentes e a remoção de tarefas concluídas ou obsoletas. A interface foi cuidadosamente pensada para ser clara e acessível, facilitando a interação do usuário e a compreensão das funcionalidades básicas. A robustez do framework Qt, com suas capacidades abrangentes para o desenvolvimento de interfaces gráficas, foi fundamental para materializar essa visão, permitindo a utilização de componentes como QListWidget para a apresentação das tarefas e QLineEdit para a entrada e edição de texto de forma fluida e responsiva.

A finalidade primordial deste empreendimento transcende a mera construção de uma ferramenta utilitária; ele se configura como um exercício prático aprofundado na aplicação dos princípios da Programação Orientada a Objetos em um contexto de desenvolvimento de aplicações de desktop. O projeto visa demonstrar, de maneira tangível e direta, a utilização de conceitos fundamentais como encapsulamento e a interação entre objetos para construir um sistema interativo e coeso. A escolha do C++ e do Qt, além de cumprir os requisitos de uma plataforma de desenvolvimento de alta performance, permitiu um controle preciso sobre a arquitetura do software e forneceu as ferramentas essenciais para a prototipagem e a implementação gráfica.

A abordagem adotada na criação deste sistema, distanciando-se de templates pré-configurados, reforça o compromisso com a compreensão profunda de cada camada da aplicação e a construção orgânica de suas funcionalidades.

O processo de desenvolvimento seguiu uma metodologia iterativa, enfatizando a modelagem de cada elemento funcional como uma entidade discreta dentro do paradigma de objetos. Essa abordagem possibilitou a criação de uma estrutura modular, onde a lógica de adição, atualização e remoção de tarefas foi encapsulada de forma clara e eficiente. Os fundamentos teóricos que balizaram este projeto incluem a arquitetura de aplicações GUI, a gestão de eventos por meio do sistema de sinais e slots do Qt, e a implementação de mecânicas de interação para o usuário. Além disso, a gestão de dados **inicialmente em memória, e posteriormente com a inclusão da persistência em arquivo,** impulsionou uma implementação rigorosa da lógica de CRUD, solidificando o aprendizado teórico em uma aplicação prática e performática. Este relatório detalhará minuciosamente cada etapa do processo de desenvolvimento, desde a configuração inicial no ambiente Qt Creator até a apresentação dos resultados alcançados.

2. METODOLOGIA

A presente seção detalha a abordagem metodológica empregada no desenvolvimento da aplicação "To-Do List", abrangendo desde a concepção inicial até a implementação das funcionalidades e a escolha das ferramentas. O processo foi guiado pelos princípios da Programação Orientada a Objetos (POO) e focado na construção de uma Interface Gráfica de Usuário (GUI) robusta e intuitiva com o framework Qt.

A concepção do projeto "To-Do List Simples" foi impulsionada pela necessidade de desenvolver uma aplicação funcional que não apenas exemplificasse os princípios da Programação Orientada a Objetos, mas também servisse como um ambiente prático para a construção de interfaces gráficas de usuário (GUI) utilizando a linguagem C++ e o framework Qt. A premissa central

consistia em criar um sistema de gerenciamento de tarefas que, embora inicialmente concebido para operar com dados exclusivamente em memória, evoluiu para incorporar a funcionalidade de persistência de dados em arquivo, intencionalmente abstraindo a complexidade de um banco de dados relacional. Tal abordagem visou concentrar os esforços no domínio das operações CRUD (Create, Read, Update, Delete) em um ambiente gráfico interativo, proporcionando uma plataforma didática para a compreensão dos mecanismos fundamentais de manipulação de dados em UI. O objetivo primordial não se limitou à mera funcionalidade do produto final, mas estendeu-se à estruturação do código de modo a evidenciar os benefícios de um design orientado a objetos em um ambiente de software leve, responsivo e com capacidade de manter as informações do usuário entre as sessões.

Para o desenvolvismento deste projeto, foram empregadas ferramentas e um ambiente de desenvolvimento específicos, escolhidos por sua adequação aos requisitos acadêmicos e técnicos. A Linguagem de Programação C++, um requisito fundamental, permitiu explorar o alto desempenho e o controle granular sobre a gestão de recursos de hardware, essenciais para a responsividade de aplicações desktop. O Framework de Desenvolvimento Qt, predefinido como a base para a construção da interface gráfica, ofereceu uma abstração eficaz para a renderização de elementos de UI e o gerenciamento de eventos em um contexto multiplataforma, com o módulo QtWidgets sendo o pilar para os componentes visuais. Complementarmente, o Ambiente de Desenvolvimento Integrado (IDE) Qt Creator providenciou o ecossistema completo e otimizado para o projeto, integrando edição de código, design visual da interface (.ui), compilação, depuração e execução, o que otimizou significativamente o fluxo de trabalho.

A arquitetura do sistema foi delineada como uma aplicação Qt Widgets padrão, centralizada na classe MainWindow, que deriva de QMainWindow. Esta classe assume a responsabilidade principal pela coordenação de todos os elementos da interface gráfica e pela orquestração das operações de manipulação das tarefas. O ciclo de vida da aplicação é implicitamente gerido pela instância de QApplication do Qt, que supervisiona o laço de eventos. Este laço é fundamental para a reatividade da GUI, processando continuamente as interações do usuário e as atualizações de estado da interface de forma assíncrona. A gestão dos estados

de interação é controlada de forma modular, prevenindo interferências indesejadas e promovendo uma execução fluida do software.

A interface do usuário foi meticulosamente projetada no Qt Designer, uma ferramenta visual integrada ao Qt Creator, visando assegurar uma experiência coesa e intuitiva para o usuário. Os principais componentes visuais e suas interações foram definidos para otimizar a usabilidade. O QLineEdit (txtTask) serve como o ponto de entrada principal para o usuário inserir novas tarefas ou para visualizar e editar tarefas existentes, com validações para impedir textos vazios. O QListWidget (listWidget) constitui o componente central para a exibição das tarefas, permitindo visualização, seleção, atualização e remoção. Os QPushButtons (btnAdd, btnUpdate, btnDelete) atuam como acionadores para as operações fundamentais do CRUD, conectados a *slots* específicos na classe MainWindow e customizados visualmente, com ativação/desativação dinâmica para btnAdd.

A detecção e o tratamento de eventos são pilares da interatividade, implementados através do poderoso mecanismo de Sinais e Slots do Qt, que permite uma comunicação desacoplada e eficiente entre os objetos. Por exemplo, o sinal clicked() de um botão é conectado a um *slot* correspondente (e.g., on_btnAdd_clicked()), executando a lógica da operação. Similarmente, itemClicked(QListWidgetItem *item) do QListWidget conecta-se a on_listWidget_itemClicked(QListWidgetItem *item) para carregar o texto da tarefa no txtTask. O sinal textChanged(const QString &arg1) do QLineEdit controla a habilitação dinâmica do botão "Adicionar".

As operações CRUD (Create, Read, Update, Delete) representam o cerne funcional da aplicação, implementadas com base em diretrizes específicas e funcionalidades interligadas, e foram complementadas por uma camada crucial de persistência de dados. A função Adicionar (on_btnAdd_clicked()) captura e valida o texto do txtTask, adicionando-o ao QListWidget e exibindo uma mensagem de sucesso. A função Atualizar (on_btnUpdate_clicked()) modifica o texto de um item selecionado na lista com base no txtTask e fornece feedback visual de sucesso. A função Excluir (on_btnDelete_clicked()) gerencia a remoção de tarefas, exibindo uma caixa de diálogo de confirmação (QMessageBox::question) antes de remover o item do QListWidget com takeItem(). A funcionalidade Ler é inerente ao QListWidget

e é complementada pela interação de clique que carrega o texto da tarefa no txtTask para visualização e edição. Para a Persistência de Dados em Arquivo, as funções loadTasks() e saveTasks() foram desenvolvidas: loadTasks() (chamada no construtor da MainWindow) lê tarefas de tasks.txt para o QListWidget, e saveTasks() (no destrutor) escreve o conteúdo do QListWidget de volta para tasks.txt, assegurando que os dados do usuário não sejam perdidos entre as sessões.

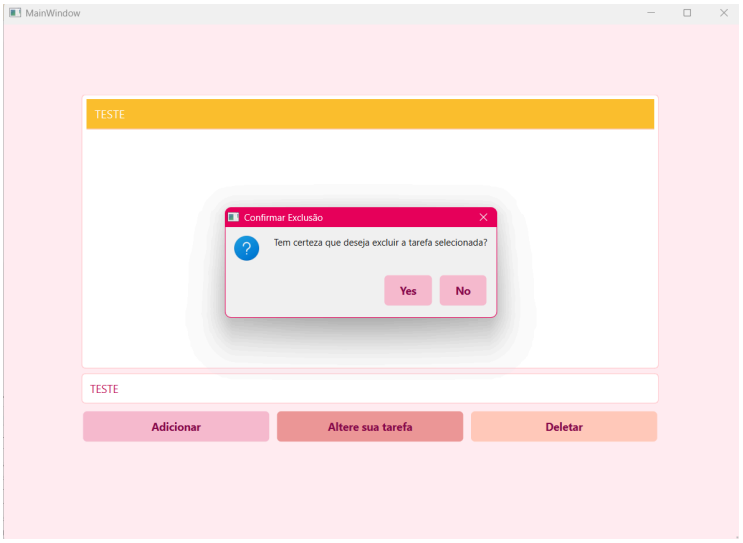
A gestão de memória para os QListWidgetItems é intrinsecamente controlada pelo próprio QListWidget, que gerencia a alocação e liberação ao adicionar ou remover itens, garantindo a ausência de vazamentos de memória. Mensagens informativas e de aviso, apresentadas via QMessageBox (como information, warning e question), são empregadas para guiar o usuário e fornecer feedback claro sobre o status de cada operação. Esta metodologia garantiu que todos os requisitos do projeto fossem atendidos de forma clara, funcional e com aderência aos princípios de desenvolvimento de software, resultando em uma aplicação robusta e utilizável.

3. RESULTADOS OBTIDOS

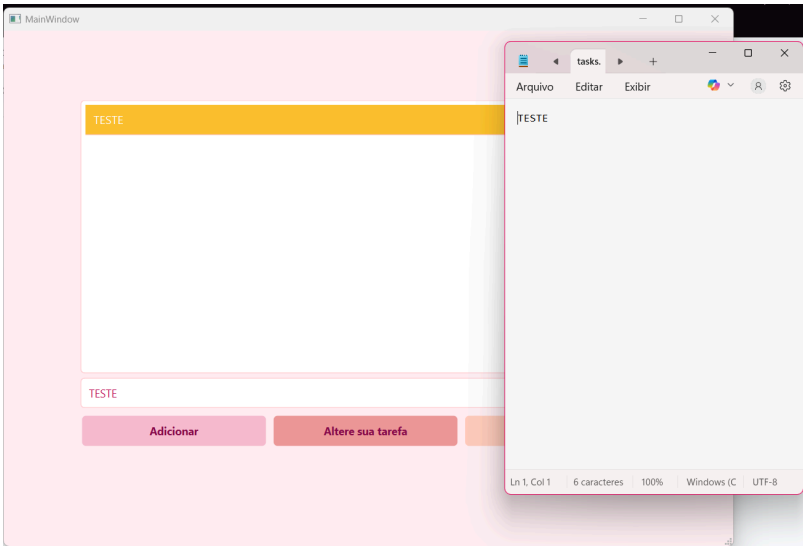
A concretização do projeto "ToDoList" em Qt resultou em uma aplicação funcional e intuitiva, projetada para auxiliar o usuário na organização de suas tarefas diárias. A aplicação opera em um ambiente gráfico moderno, construído com o framework Qt, que permitiu a criação de uma interface de usuário clara, direta e visualmente atraente, utilizando uma paleta de cores rosa pastel para um design feminino e acolhedor.

A interface principal do aplicativo exibe uma lista dinâmica de tarefas, onde o usuário pode facilmente visualizar, adicionar, atualizar e excluir itens. A entrada de novas tarefas é facilitada por um campo de texto interativo, e a usabilidade foi aprimorada com a desativação automática do botão "Adicionar" quando o campo de entrada está vazio, prevenindo a submissão de tarefas em branco e guiando o usuário.

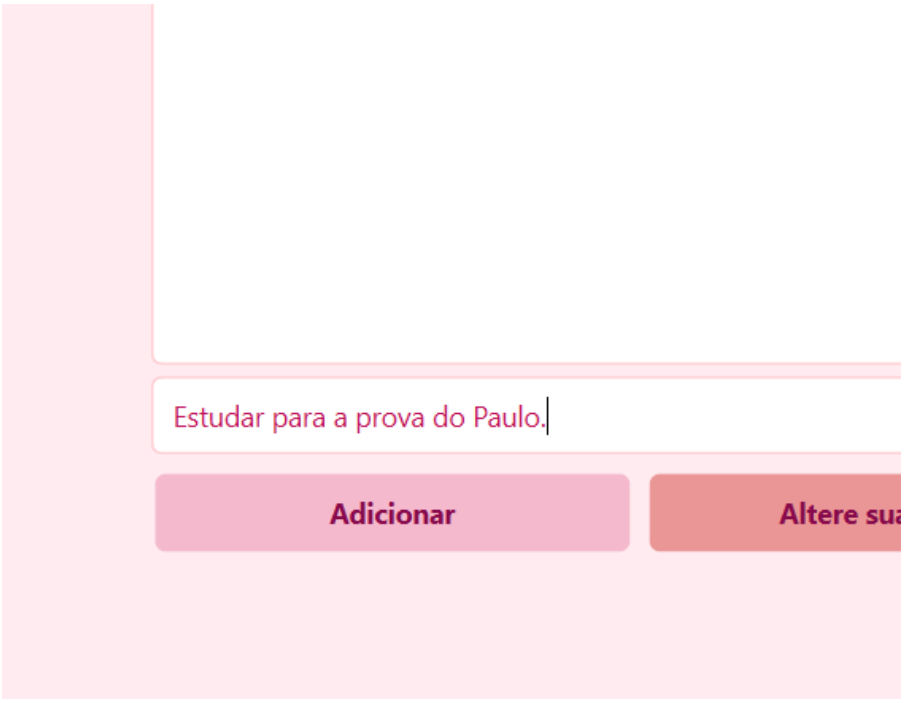
Um dos resultados mais significativos do projeto é a implementação da **persistência de dados**. As tarefas são automaticamente salvas em um arquivo (tasks.txt) ao fechar o aplicativo e carregadas novamente ao iniciá-lo. Isso garante que as informações do usuário não sejam perdidas entre as sessões, tornando a ferramenta prática e confiável para o gerenciamento contínuo de tarefas. A comunicação com o usuário é complementada por mensagens de alerta e sucesso claras, informando sobre o status das operações (como adição, atualização, exclusão e avisos sobre seleção de itens).



Captura de Tela demonstrando uma tentativa de exclusão de uma tarefa cadastrada



Captura de Tela demonstrando a persistência de dados de uma tarefa cadastrada.



Captura de Tela demonstrando a escrita de uma tarefa.

4. CONCLUSÃO

O desenvolvimento do projeto "ToDoList" em Qt representou a concretização bem-sucedida de um aplicativo funcional e intuitivo, que atende aos objetivos propostos de um gerenciador de tarefas pessoais. Este projeto demonstrou a aplicação prática dos princípios da Programação Orientada a Objetos (POO) e a eficácia do framework Qt na construção de interfaces gráficas de usuário robustas e atraentes em C++.

A aplicação oferece um conjunto de funcionalidades essenciais para a gestão de tarefas, permitindo aos usuários adicionar, atualizar e excluir itens de forma eficiente. O design da interface, caracterizado por uma paleta de cores rosa pastel, contribui para uma experiência visual acolhedora e moderna. A usabilidade foi significativamente aprimorada com a implementação de validações em tempo real, como a desativação do botão de adição quando o campo de entrada de texto está vazio, guiando o usuário e prevenindo erros.

Um dos resultados mais críticos e valiosos alcançados foi a implementação da **persistência de dados**. A capacidade de salvar automaticamente as tarefas em um arquivo local ao fechar o aplicativo e carregá-las novamente ao iniciar garante que o progresso do usuário seja mantido, transformando o "ToDoList" em uma ferramenta confiável e prática para o uso diário. Além disso, o sistema de feedback através de mensagens de sucesso e aviso claras otimiza a interação e a compreensão do usuário sobre as ações realizadas.

Em suma, o projeto "ToDoList" não apenas entregou uma aplicação funcional e com design aprimorado, mas também serviu como uma poderosa plataforma para o aprofundamento em técnicas de programação com Qt e C++, consolidando os conhecimentos adquiridos e estabelecendo uma base sólida para futuros desenvolvimentos e extensões de suas funcionalidades.

REFERÊNCIAS

QT COMPANY. **Qt Documentation**. Disponível em: https://doc.qt.io/.

QT COMPANY. **Qt Widgets (Qt 6.9.1)**. Disponível em: https://doc.qt.io/qt-6/qtwidgets-index.html.