INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

MATHEUS LOPES DE OLIVEIRA

Este documento se trata da entrega projeto final da disciplina de Programação Orientada a Objetos, do curso de Tecnologia em Análise e Desenvolvimento de Sistemas, do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - Campus Campos do Jordão (IFSP-CJO).

PROFESSOR: Paulo Giovani de Faria Zeferino.

Sistema de Movimentação de Estoque com Qt Creator

CAMPOS DO JORDÃO

2025

RESUMO

O presente trabalho descreve o desenvolvimento de um sistema de controle de estoque utilizando a linguagem de programação C++ e a biblioteca Qt Creator. O principal objetivo do sistema é fornecer uma ferramenta eficiente e prática para a gestão de produtos e movimentação de estoque, voltada para o uso de uma consultora de uma empresa de cosméticos. O projeto explora conceitos de Programação Orientada a Objetos (POO), como a modularização do código e a criação de classes que representam as funcionalidades do sistema, como o cadastro de produtos, movimentação de estoque, consulta de produtos e gerenciamento de usuários. O sistema permite a atualização e o armazenamento de dados por meio de arquivos JSON, facilitando o controle de entradas e saídas de produtos, além da consulta de informações em tempo real. A interface gráfica foi criada utilizando Qt Creator, proporcionando uma experiência interativa para o usuário, com menus, botões e campos de entrada eficientes. O trabalho também sugere diversas melhorias, como a otimização na troca de telas, adicionar logs de alterações de estoque, aprimorar o design visual e a usabilidade, além da expansão das funcionalidades para futuras versões, como a inclusão de relatórios de movimentação de estoque.

Palavras-chave: C++, Qt Creator, POO, Controle de Estoque, JSON, Gestão de Produtos, Sistema de Movimentação de Estoque.

ABSTRACT

This work describes the development of an inventory control system using the C++ programming language and the Qt Creator library. The main objective of the system is to provide an efficient and practical tool for managing products and inventory movement, specifically designed for use by a consultant of a cosmetics company. The project explores Object-Oriented Programming (OOP) concepts, such as code modularization and the creation of classes that represent the system's functionalities, including product registration, inventory movement, product queries, and user management. The system allows for the updating and storage of data via JSON files, facilitating the tracking of product entries and exits, as well as real-time information retrieval. The graphical interface was built using Qt Creator, providing an interactive user experience with efficient menus, buttons, and input fields. The work also suggests several improvements, such as optimizing screen transitions, adding inventory change logs, enhancing visual design and usability, and expanding the system's capabilities for future versions, such as the inclusion of inventory movement reports.

Keywords: C++, Qt Creator, OOP, Inventory Control, JSON, Product Management, Inventory Movement System.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Figura 1 - Qt

O desenvolvimento de sistemas informatizados é uma área central no aprendizado de programação e na aplicação de conceitos de Programação Orientada a Objetos (POO). A construção de sistemas práticos, como um sistema de controle de estoque, permite a integração de teoria e prática, proporcionando aos estudantes a oportunidade de consolidar seus conhecimentos e habilidades. Este trabalho tem como objetivo o desenvolvimento de um sistema de controle de estoque de produtos cosméticos, utilizando a linguagem C++ e a biblioteca Qt.

A proposta deste projeto é aplicar conceitos essenciais da programação orientada a objetos, como encapsulamento, herança e polimorfismo, para implementar funcionalidades típicas de um sistema de gestão de estoque. O sistema permite o cadastro de produtos, movimentação de estoque (entradas e saídas), consulta de itens armazenados e manipulação de dados com o uso de arquivos JSON. A escolha do Qt como framework de desenvolvimento se baseia em sua robustez, flexibilidade e facilidade para criação de interfaces gráficas, além de seu suporte a sistemas multiplataforma.

Este sistema foi desenvolvido com o objetivo de ajudar no gerenciamento de estoque de uma consultora de uma empresa de cosméticos, oferecendo uma ferramenta prática e eficiente para o controle das quantidades, entradas e saídas de produtos. Assim, além de aplicar o aprendizado dos conceitos de POO, o projeto busca trazer uma solução real e útil para o cotidiano de profissionais que necessitam de uma gestão mais eficiente do estoque.

O sistema desenvolvido oferece uma interface gráfica para o usuário interagir de forma intuitiva, com funcionalidades como filtros de busca por nome, marca ou grupo de produtos, além da capacidade de realizar operações de movimentação de estoque e visualizar os dados de forma organizada em tabelas. Este projeto visa não apenas aplicar os conceitos de POO, mas também proporcionar uma experiência de desenvolvimento de software completo, com foco em usabilidade e boas práticas de programação.

Durante a implementação do sistema, foram explorados aspectos como a organização do código em classes e métodos, o uso de estruturas de dados eficientes, a persistência de dados através de arquivos JSON e a criação de uma interface gráfica interativa e amigável. Além disso, o projeto foi uma oportunidade para reforçar os conceitos de lógica de programação e técnicas de design de interação.

Este relatório descreve as etapas de desenvolvimento do sistema, desde arquitetura até a implementação e os resutados, oferecendo uma visão crítica sobre os desafios enfrentados e os aprendizados adquiridos ao longo do processo.

1.1 Objetivo

O objetivo deste projeto é consolidar os conhecimentos adquiridos na disciplina de **Programação Orientada a Objetos (POO)**, utilizando a linguagem **C++** e o framework **Qt** para o desenvolvimento de um **sistema de controle de estoque de produtos cosméticos**. Além disso, busca-se aprimorar habilidades na criação de aplicações gráficas interativas, com ênfase na usabilidade e na manipulação de dados.
Os objetivos específicos incluem:

- **Desenvolver um sistema de controle de estoque**: Criar uma aplicação onde seja possível cadastrar produtos, registrar movimentações de entrada e saída, realizar consultas rápidas e acompanhar a quantidade disponível de cada item em estoque. O sistema permite também a manipulação de dados em formato **JSON** para armazenamento persistente.
- **Aprofundar o conhecimento na linguagem C++**: Aplicar conceitos da linguagem, como manipulação de classes, uso de estruturas condicionais, laços, e armazenamento de dados em arquivos **JSON**. O sistema será capaz de ler e escrever dados, proporcionando uma solução prática para o gerenciamento de estoque.

- **Praticar conceitos de orientação a objetos**: Estruturar o projeto de forma modular, implementando classes para representar os diferentes componentes do sistema, como **produtos**, **movimentações de estoque**, e **consultas de dados**. O uso de métodos e atributos encapsulados visa promover uma melhor organização do código e facilitar a manutenção.
- **Desenvolver uma aplicação com foco em usabilidade e funcionalidade**: Criar uma interface gráfica intuitiva e eficiente, com funcionalidades como filtros de busca, exibição de tabelas e atualizações em tempo real dos dados de estoque. O objetivo é proporcionar uma experiência de uso fluida e sem complicações para o usuário.

Este projeto também serve como base para futuros aprimoramentos, seja na adição de novas funcionalidades, como relatórios avançados e alertas de baixo estoque, ou no refino das mecânicas de gestão de dados e interação com o usuário. É uma oportunidade de aplicar conceitos teóricos em um contexto prático e contribuir para o aprendizado contínuo no desenvolvimento de sistemas orientados a objetos.

1.2 Justificativa

A escolha deste tema, que envolve o desenvolvimento de um sistema de controle de estoque de produtos cosméticos, é motivada por diversos aspectos que abrangem tanto interesses pessoais quanto objetivos educacionais relacionados ao aprendizado da programação e ao domínio de ferramentas específicas para o desenvolvimento de sistemas. O gerenciamento de estoque é uma tarefa crucial para empresas de diferentes setores, e a construção de uma ferramenta prática e eficiente de controle de produtos oferece um ambiente desafiador e valioso para a aplicação de conceitos fundamentais de desenvolvimento de software, como manipulação de dados, organização de classes e criação de interfaces gráficas.

A adoção desse tema se justifica, em primeiro lugar, pela oportunidade de aplicar os conhecimentos adquiridos na disciplina de Programação Orientada a Objetos (POO) em um contexto real, utilizando a linguagem C++ e a biblioteca Qt. A escolha do Qt para o desenvolvimento da interface gráfica se baseia em sua robustez e na facilidade de criação de aplicações multiplataforma, o que permite ao sistema ser facilmente adaptado para diferentes ambientes de trabalho.

Além disso, o projeto busca aprimorar habilidades no uso de estruturas de dados como vetores, listas e mapas, fundamentais para a organização e manipulação das informações sobre os produtos em estoque. O uso de arquivos JSON para persistência de dados também foi escolhido, pois oferece uma solução simples e eficaz para o armazenamento de informações, permitindo que o sistema possa ser facilmente modificado e expandido no futuro.

A escolha de desenvolver este sistema também é motivada pela necessidade de fornecer uma solução prática para empresas de pequeno porte ou consultores que necessitam de uma ferramenta de controle de estoque acessível e eficiente. Ao combinar o aprendizado técnico com a aplicação prática em um projeto real, este trabalho contribui tanto para o crescimento acadêmico quanto para a criação de um sistema útil que pode ser utilizado por profissionais em seu cotidiano.

1.3 Aporte Teórico

O desenvolvimento de sistemas informatizados envolve a aplicação de diversos conhecimentos, como programação, design de interface, gerenciamento de dados e interação do usuário. Este projeto, focado no desenvolvimento de um sistema de controle de estoque, visa explorar aspectos importantes da criação de sistemas eficientes e funcionais, como a programação orientada a objetos (POO), o uso de ferramentas adequadas para o gerenciamento de dados e a construção de uma interface gráfica interativa para o usuário.

- **Linguagem C++ no Desenvolvimento de Sistemas:** A linguagem C++ é amplamente utilizada no desenvolvimento de sistemas devido à sua alta performance e controle sobre os recursos do sistema. O C++ permite que o programador tenha controle direto sobre a memória, o que é essencial para o gerenciamento eficiente de grandes volumes de dados, como os relacionados ao estoque de produtos. A linguagem também oferece suporte à programação orientada a objetos (POO), o que permite uma organização modular do código, fundamental para a construção de sistemas mais complexos e de fácil manutenção.

  Em sistemas como o proposto, a programação orientada a objetos facilita a criação de classes que representam as entidades do sistema, como produtos, movimentações de estoque e consultas. Cada classe pode ser responsável por

comportamentos específicos, como o cadastro de produtos, a movimentação de itens e a visualização de dados, tornando o código mais organizado e eficiente.

- **Programação Orientada a Objetos (POO):** A Programação Orientada a Objetos (POO) é um paradigma que organiza o código em torno de "objetos", que são instâncias de "classes" que possuem atributos e métodos. No contexto deste sistema, a POO permite modelar as entidades envolvidas, como os produtos, as movimentações de estoque e as consultas, de maneira estruturada, facilitando sua manipulação e interação.
  Por exemplo, no sistema de controle de estoque, a classe Produto pode representar os produtos armazenados, enquanto a classe Movimentacao modela as operações de entrada e saída de itens do estoque, e a classe Consulta pode ser responsável por exibir os produtos filtrados de acordo com os critérios de busca. A utilização de métodos e atributos encapsulados promove uma melhor organização do código e facilita a manutenção do sistema.

- **Qt para Desenvolvimento de Interfaces Gráficas:** O Qt é um framework poderoso e amplamente utilizado para o desenvolvimento de interfaces gráficas, tanto em sistemas multiplataforma quanto em aplicativos desktop. Ele facilita a criação de interfaces interativas, como formulários, tabelas e caixas de diálogo, proporcionando uma experiência de uso intuitiva para o usuário. O Qt também oferece recursos para a manipulação de dados, como leitura e escrita de arquivos JSON, o que é fundamental para o armazenamento persistente dos dados do sistema.
  No projeto em questão, o Qt é utilizado para criar uma interface gráfica que permite ao usuário realizar o cadastro de produtos, visualizar o estoque e registrar movimentações. A manipulação dos dados é feita através de arquivos JSON, permitindo que o sistema armazene as informações de maneira simples e eficiente.

- **Manipulação de Dados com JSON:** O uso de JSON (JavaScript Object Notation) como formato de armazenamento de dados tem se tornado uma prática comum em sistemas modernos devido à sua simplicidade e flexibilidade. O Qt

oferece uma excelente integração com JSON, facilitando a leitura, a escrita e a atualização dos dados armazenados. No sistema de controle de estoque, os dados dos produtos e das movimentações são armazenados em arquivos JSON, permitindo que o sistema seja facilmente modificado e expandido no futuro.

- **Design de Sistemas e Usabilidade:** O design de sistemas envolve a criação de uma estrutura funcional e eficiente, com foco na experiência do usuário. Em um sistema como o proposto, o objetivo é criar uma interface gráfica que seja clara, objetiva e fácil de usar. A definição de regras claras para o cadastro de produtos, movimentação de itens e realização de consultas é essencial para garantir uma boa usabilidade.

  Além disso, o design de sistemas envolve a criação de uma interação eficiente entre o usuário e a aplicação. A interface gráfica deve ser intuitiva, permitindo que o usuário realize operações como a entrada e saída de produtos de maneira rápida e sem erros. A criação de filtros de busca e a exibição organizada dos produtos são alguns dos aspectos que contribuem para a melhoria da experiência do usuário.

- **Persistência de Dados e Escalabilidade:** A persistência de dados é um aspecto fundamental para garantir que as informações do sistema sejam armazenadas de forma confiável e possam ser acessadas posteriormente. A escolha do JSON para a persistência dos dados é uma solução simples e eficaz, que permite ao sistema ser facilmente modificado e expandido.

Além disso, o sistema foi projetado para ser escalável, permitindo que novas funcionalidades sejam adicionadas no futuro. A utilização de classes e métodos bem estruturados facilita a manutenção e a evolução do sistema, tornando-o adequado para atender às necessidades de empresas em crescimento.

2 METODOLOGIA

A metodologia empregada neste trabalho foi fundamentada em uma combinação de pesquisa bibliográfica e aplicação prática dos conhecimentos adquiridos, com o objetivo de desenvolver um sistema de controle de estoque de produtos cosméticos utilizando C++ e a biblioteca Qt. Através da utilização da Programação Orientada a Objetos (POO) e das ferramentas fornecidas pelo Qt, foi possível criar um sistema modular e eficiente para o gerenciamento das funcionalidades do controle de estoque, como cadastro de produtos, movimentação de entradas e saídas, consultas e persistência de dados.

A primeira etapa do desenvolvimento envolveu uma pesquisa bibliográfica sobre o uso do Qt para o desenvolvimento de sistemas gráficos e manipulação de dados. Essa pesquisa, realizada inicialmente, foi essencial para a fundamentação do uso do Qt, permitindo o entendimento de suas funcionalidades, como criação de interfaces gráficas, manipulação de eventos de usuário e comunicação com arquivos JSON. A pesquisa também incluiu o estudo da documentação oficial do Qt, que foi fundamental para a integração da biblioteca ao projeto.

Além disso, foram analisadas outras fontes relacionadas à Programação Orientada a Objetos (POO), com o objetivo de estruturar o código de forma eficiente, modular e reutilizável. A utilização de conceitos de POO como encapsulamento, herança e polimorfismo permitiu a criação de classes específicas para gerenciar as entidades do sistema, como Produto, Movimentação e Consulta. A documentação do Qt ofereceu insights sobre a utilização de recursos como QTableWidget para exibição de dados em tabelas, QJsonDocument e QJsonArray para manipulação de dados em formato JSON, além de widgets como QPushButton, QLineEdit e QComboBox para interações do usuário com o sistema.

Após a fundamentação teórica, foi realizada a instalação e configuração das ferramentas de desenvolvimento, incluindo o Qt Creator, para a criação das interfaces gráficas e o desenvolvimento das funcionalidades do sistema. O processo de implementação foi dividido em várias etapas: primeiro, a criação da interface gráfica com Qt Designer; depois, a implementação das funcionalidades de cadastro de produtos, movimentação de estoque e consultas de dados; e, por fim, a implementação da persistência de dados utilizando arquivos JSON.

Durante o desenvolvimento, o sistema foi constantemente testado para garantir que as funcionalidades estivessem funcionando conforme o esperado. Testes de validação foram realizados para garantir que as entradas de dados estivessem sendo corretamente registradas e armazenadas no arquivo JSON, além de garantir que as movimentações de estoque e consultas estivessem sendo corretamente realizadas.

2.1 Ferramentas Utilizadas

O desenvolvimento deste projeto foi realizado utilizando um conjunto de ferramentas poderosas e amplamente utilizadas para a criação de sistemas interativos. A seguir, será apresentada uma descrição detalhada de cada uma dessas ferramentas: **Qt**, **C++**, **GCC Compiler**, **PlantUML**, e **Google Gemini**.

2.1.1 Qt

Qt é um framework de desenvolvimento multiplataforma utilizado para a criação de interfaces gráficas de usuário (GUIs) e sistemas complexos. O Qt foi escolhido para este projeto devido à sua robustez, flexibilidade e facilidade de uso, sendo especialmente útil para o desenvolvimento de aplicações desktop. Ele oferece um conjunto completo de widgets para a criação de interfaces ricas, como QTableWidget para exibição de tabelas e QLineEdit para entradas de texto, essenciais para a funcionalidade do sistema de controle de estoque. Além disso, o Qt facilita a manipulação de arquivos, como a leitura e escrita de arquivos JSON, que são usados para a persistência dos dados dos produtos e movimentações de estoque.

2.1.2 C++

C++ é uma das linguagens de programação mais populares e poderosas, amplamente utilizada no desenvolvimento de sistemas de alto desempenho e aplicações que exigem manipulação direta de hardware. A escolha do C++ para este projeto foi motivada pela sua performance, controle sobre os recursos do sistema e pela facilidade com que se pode trabalhar com a Programação Orientada a Objetos (POO). A

utilização de C++ permitiu uma abordagem modular e eficiente, com a criação de classes específicas para representar as entidades do sistema, como Produto, Movimentacao e Consulta, tornando o código mais organizado e fácil de manter.

2.1.3 GCC Compiler

O **GCC (GNU Compiler Collection)** é um compilador de código aberto amplamente utilizado para a compilação de código nas linguagens C e C++. Ele foi escolhido neste projeto devido à sua robustez, capacidade de otimização e compatibilidade com múltiplos sistemas operacionais, como Linux, Windows e macOS. O GCC foi utilizado para compilar o código C++ e gerar o arquivo executável, garantindo que o sistema fosse portável e eficiente em diferentes ambientes. A utilização do GCC também possibilitou a criação de código otimizado para garantir que o sistema funcione de forma rápida e eficiente, mesmo em máquinas com recursos limitados.

2.1.4 PlantUML

PlantUML foi utilizado para gerar os diagramas de classe do sistema, representando visualmente a estrutura do código e as interações entre as classes. A ferramenta permite criar diagramas de classes e outros diagramas UML de forma rápida e eficiente, utilizando uma linguagem de marcação simples. Os diagramas gerados com PlantUML ajudaram na visualização da organização do sistema e facilitaram o entendimento da arquitetura do código, sendo fundamentais para a documentação e o planejamento do projeto.

2.1.5 Google Gemini

Google Gemini foi utilizado para criar a logo do sistema. Embora o sistema de controle de estoque não envolva gráficos complexos ou animações, a criação de uma identidade visual para o projeto foi importante para fornecer uma aparência profissional à interface. O Gemini foi utilizado para criar a imagem da logo, alinhando-a com o estilo visual do sistema e garantindo que o design fosse coeso e atrativo.

Essas ferramentas, combinadas com o uso de conceitos de Programação Orientada a Objetos (POO) e uma abordagem modular, foram fundamentais para o desenvolvimento e sucesso do projeto, proporcionando tanto a base técnica quanto a parte visual e interativa do sistema.

2.2 Desenvolvimento

A segunda etapa foi a aplicação prática do conhecimento adquirido. Durante essa fase, foi necessário planejar a estrutura do sistema de controle de estoque, com foco na criação de um sistema funcional para cadastro, movimentação e consulta de produtos, utilizando C++ e o framework Qt. A utilização de Programação Orientada a Objetos (POO) foi crucial para modularizar o código, tornando-o mais organizado, legível e flexível. Cada componente do sistema foi implementado como uma classe independente, permitindo a criação de objetos com responsabilidades bem definidas e facilitando a manutenção do código.

A seguir, será apresentada uma descrição detalhada das principais classes, seus métodos e funcionalidades dentro do sistema:

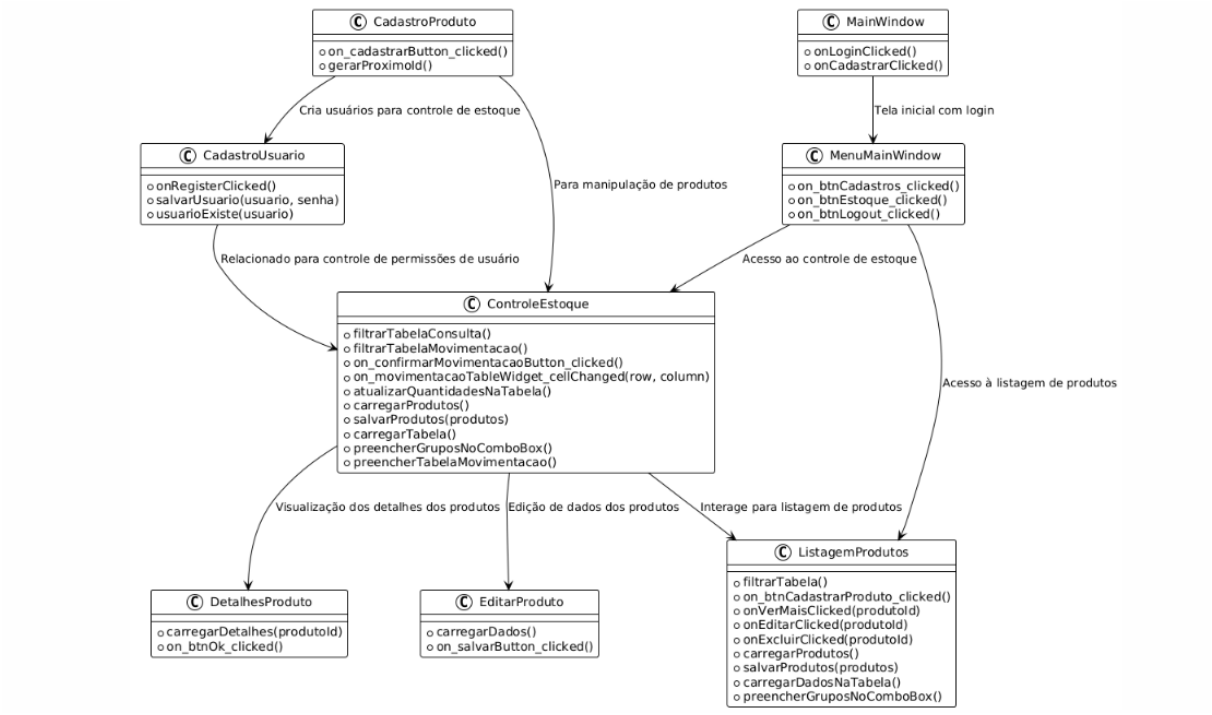

Figura 2 – Diagrama de Classes

- Classe CadastroProduto **(CadastroProduto.hpp);**

A classe **CadastroProduto** é responsável pela criação de novos produtos no sistema de controle de estoque. Ela permite a entrada de dados como nome, grupo, marca, quantidade, preço e código de barras dos produtos. A principal funcionalidade dessa classe é o método **on_cadastrarButton_clicked()**, que é acionado quando o botão de cadastro é clicado. Esse método valida os dados inseridos e, se tudo estiver correto, adiciona o produto ao arquivo **produtos.json**. O método **gerarProximoId()** é utilizado para gerar o próximo ID disponível para o produto a ser cadastrado.

- Classe CadastroUsuario **(CadastroUsuario.hpp);**

A classe **CadastroUsuario** gerencia o processo de cadastro de usuários no sistema. Ela permite a criação de novos usuários, com a inserção de **nome de usuário** e **senha**. O método **onRegisterClicked()** é responsável por verificar se os campos de nome de usuário e senha foram preenchidos e se o nome de usuário já existe. Caso o nome seja único, o usuário é salvo no arquivo **usuarios.json**. A classe também possui métodos para verificar se o nome de usuário já foi cadastrado, através do método **usuarioExiste()**.

- Classe ControleEstoque **(ControleEstoque.hpp);**

A classe **ControleEstoque** é o coração do sistema, sendo responsável pela lógica do gerenciamento de estoque. Ela contém diversas funcionalidades, como o cadastro e a movimentação de produtos. A classe é composta por métodos para atualizar a tabela de produtos (**carregarTabela()**), registrar movimentações de entrada e saída (**preencherTabelaMovimentacao()**), e realizar filtros de busca para consulta de produtos no estoque (**filtrarTabelaConsulta()** e **filtrarTabelaMovimentacao()**). Essa classe também lida com o carregamento e salvamento de dados no arquivo **produtos.json**.

A interação com a interface gráfica é feita por meio de eventos que acionam esses métodos, permitindo que o usuário cadastre produtos, visualize o estoque e registre movimentações.

- Classe DetalhesProduto **(DetalhesProduto.hpp);**

A classe **DetalhesProduto** permite ao usuário visualizar os detalhes completos de um produto cadastrado no sistema. Ao clicar em um produto na lista de consulta, o usuário pode acessar uma tela com informações detalhadas sobre o nome, grupo, marca, código de barras, quantidade, preço e descrição do produto. O método **carregarDetalhes()** é responsável por carregar os dados do produto a partir do arquivo **produtos.json** e exibi-los na interface gráfica.

- Classe EditarProduto **(EditarProduto.hpp);**

A classe **EditarProduto** oferece a funcionalidade de editar os dados de um produto já cadastrado no sistema. O método **carregarDados()** carrega os dados do produto selecionado e os exibe nos campos de edição, enquanto o método **on_salvarButton_clicked()** valida e salva as alterações feitas pelo usuário. Essa classe também possui validações para garantir que os dados sejam inseridos corretamente antes de serem atualizados no arquivo **produtos.json**.

- Classe ListagemProdutos **(ListagemProdutos.hpp);**

A classe **ListagemProdutos** é responsável pela interface de consulta e listagem de produtos no sistema. Ela permite ao usuário buscar produtos de acordo com critérios como nome, marca, grupo e código de barras. O método **carregarDadosNaTabela()** preenche a tabela de produtos com os dados armazenados no arquivo **produtos.json**, enquanto o método **adicionarBotoesAcao()** adiciona botões de ação, como Ver Mais, Editar e Excluir, a cada linha da tabela, permitindo ao usuário interagir com os produtos.
Além disso, a classe também possui métodos de filtragem, permitindo que o usuário refine a busca para localizar produtos mais facilmente.

A interação entre as classes é essencial para o bom funcionamento do sistema. A classe **ControleEstoque** gerencia a criação, atualização e movimentação de produtos e interage com as demais classes, como **CadastroProduto**, **DetalhesProduto**, **EditarProduto** e **ListagemProdutos**. A classe **CadastroProduto** permite adicionar

novos produtos ao sistema, enquanto **DetalhesProduto** e **EditarProduto** possibilitam a visualização e edição dos produtos. A classe **ListagemProdutos** facilita a busca e a consulta dos produtos cadastrados, apresentando-os de forma clara para o usuário.

A modularização através de POO facilitou o processo de desenvolvimento, permitindo que as funcionalidades fossem implementadas de maneira independente, mas integradas de forma eficiente. Isso também facilitou a manutenção e a expansão do sistema, possibilitando a adição de novas funcionalidades no futuro, como relatórios ou gráficos de movimentação de estoque, sem comprometer a estrutura já existente.

2.3 Estudos Realizados

A pesquisa realizada para o desenvolvimento deste sistema envolveu o estudo de técnicas de manipulação de arquivos, como a leitura e escrita de dados em arquivos JSON, que são utilizados para armazenar informações sobre os produtos e as movimentações no estoque. A função carregarProdutos() foi implementada com base nos conceitos aprendidos sobre manipulação de arquivos em C++, permitindo que o sistema carregue os produtos e os dados de movimentação diretamente de um arquivo JSON, garantindo persistência entre as execuções do programa.

Além disso, foi realizada uma pesquisa sobre design de interfaces gráficas utilizando o Qt. A construção das telas de cadastro de produtos, movimentação de estoque, consulta e listagem de produtos foi baseada em práticas recomendadas para a criação de interfaces interativas e intuitivas. Durante o desenvolvimento, ajustei os parâmetros da interface, como a disposição dos widgets, tamanhos de tabelas e botões, com base em testes contínuos de usabilidade e feedback, visando uma experiência de usuário agradável e funcional.

Foi também importante estudar sobre controle de entradas de usuário e validação de dados. No caso da movimentação de estoque, a função ValidarMovimentacao() foi implementada para garantir que não ocorra um erro de movimentação, como a tentativa de saída de produtos quando não há quantidade suficiente em estoque. A validação de dados foi um passo crucial para garantir a integridade das informações armazenadas no sistema.

Além disso, a integração de todas as classes e funcionalidades foi realizada com testes contínuos e ajustes. As principais interações entre as classes Produto,

Movimentacao, Estoque e Consulta foram verificadas para garantir o correto funcionamento do sistema. O processo de depuração e otimização foi conduzido utilizando as ferramentas de depuração da IDE Qt Creator, além de testes manuais de entradas e saídas de dados, o que ajudou a identificar e corrigir problemas de desempenho, como o tempo de resposta nas operações de busca e movimentação de estoque.

Em resumo, a metodologia aplicada neste projeto envolveu uma combinação de pesquisa bibliográfica sobre manipulação de arquivos JSON, design de interfaces gráficas com Qt, e programação orientada a objetos. Além disso, a aplicação prática de conceitos de validação de dados e testes contínuos foi essencial para a criação de um sistema funcional e eficiente. As fontes de pesquisa forneceram a base necessária para o desenvolvimento de um sistema de controle de estoque confiável e fácil de usar, que atende às necessidades de gerenciamento de produtos e movimentações de forma eficiente e sem erros.

3 RESULTADOS OBTIDOS

Os resultados obtidos neste projeto são demonstrados por meio de capturas de tela (prints) do sistema em funcionamento. Essas imagens ilustram os diferentes aspectos e etapas do sistema de controle de estoque, incluindo a interface gráfica, as interações do usuário e as funcionalidades principais, como o cadastro de produtos, movimentação de estoque, consulta de produtos, login de usuário e menu principal. O fluxo do sistema, desde a inicialização até a execução das operações, é visível nas capturas, proporcionando uma visão detalhada de como cada parte do sistema funciona.

Figura 3 – Tela de Login

A Tela de Login permite que o usuário se autentique antes de acessar o sistema. O print exibe a tela de login com campos para nome de usuário e senha, além de botões para entrar e registrar-se. O usuário precisa inserir as credenciais corretamente para acessar o sistema. Se os dados estiverem corretos, o sistema permite o acesso ao Menu Principal. Caso contrário, uma mensagem de erro é exibida, indicando que as credenciais estão incorretas. A tela também apresenta a opção de Cadastro de Usuário para novos usuários.

Figura 4 – Cadastro de Usuário

A Tela de Cadastro de Usuário permite que novos usuários se registrem no sistema. O print mostra a tela de cadastro com campos para nome de usuário e senha, além de um botão para registrar o novo usuário. Ao preencher os campos e clicar em Registrar, o sistema verifica se o nome de usuário já está cadastrado. Se não, o novo usuário é salvo no arquivo de usuarios.json. Caso o nome de usuário já exista, uma mensagem de erro é exibida, pedindo que o usuário escolha outro nome.

Figura 5 – Menu Principal

Após o login, o Menu Principal é exibido. O print mostra o menu com as opções de Cadastrar Produto, Movimentação de Estoque e Sair. O menu é simples e acessível, permitindo ao usuário navegar para as diferentes funcionalidades do sistema. O layout é claro, e as opções são bem definidas, proporcionando uma navegação fácil e intuitiva.

Figura 6 – Tela de Cadastro de Produto

Na Tela de Cadastro de Produto, o usuário pode inserir dados de novos produtos, como nome, grupo, marca, quantidade, preço e código de barras. O print mostra a interface do formulário de cadastro, com campos de entrada para esses dados e um botão de Cadastrar. Após preencher todos os campos, o usuário pode cadastrar o produto no sistema, que será salvo no arquivo produtos.json. Se algum campo obrigatório estiver vazio, o sistema exibe uma mensagem de erro, pedindo que o usuário preencha todos os campos.

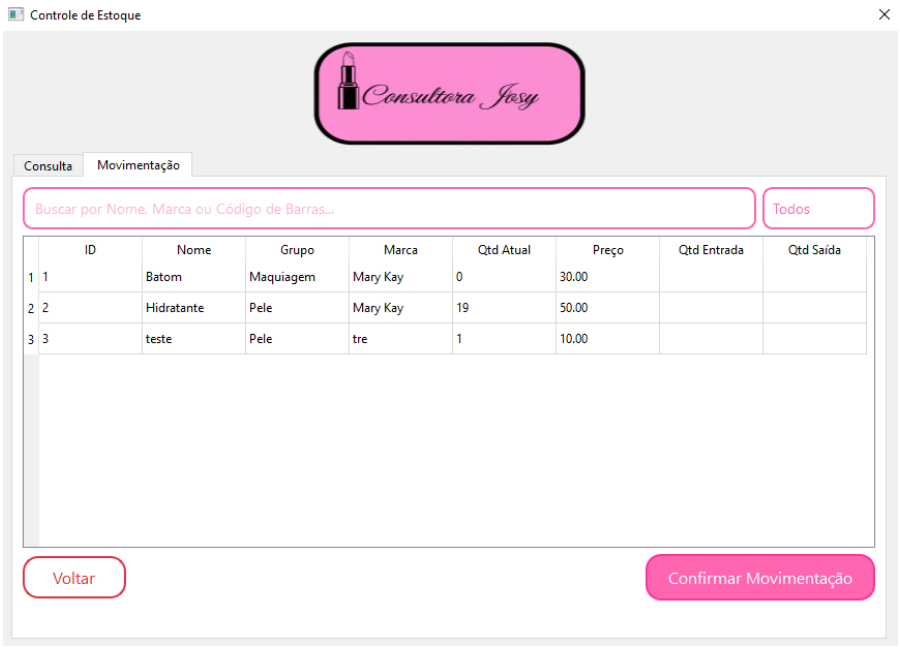

Figura 7 – Tela de Movimentação de Estoque

Na Tela de Movimentação de Estoque, o usuário pode registrar a entrada ou saída de produtos no estoque. A captura de tela mostra a tabela com todos os produtos cadastrados, e campos para informar a quantidade movimentada. O usuário escolhe o tipo de movimentação (entrada ou saída), insere a quantidade e clica no botão Confirmar Movimentação. O sistema valida se a movimentação é possível (por exemplo, não permitir uma saída maior que a quantidade disponível no estoque) e, se estiver correta, atualiza o estoque no arquivo produtos.json.

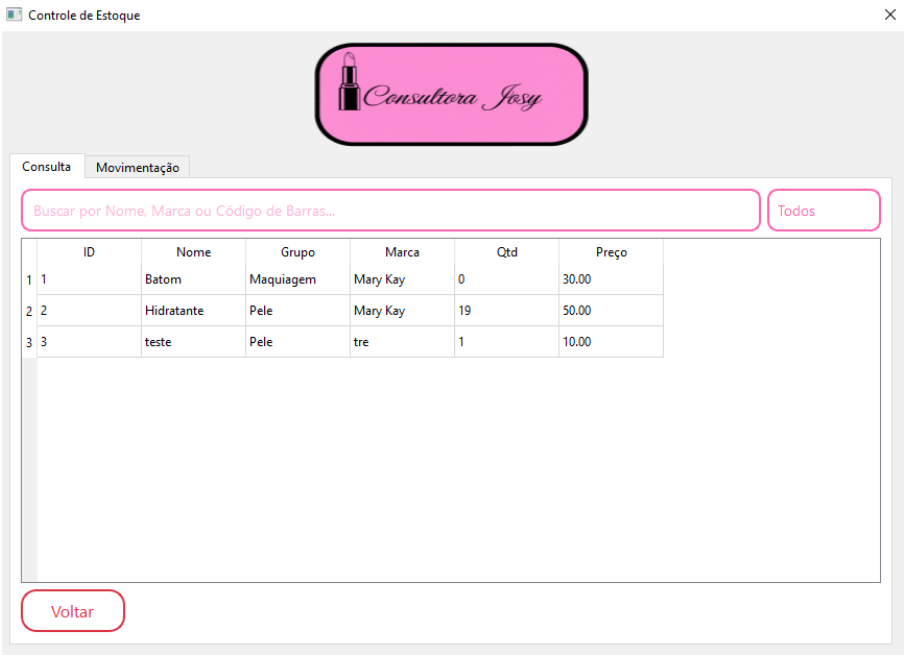

Figura 8 – Tela de Consulta de Produtos

A Tela de Consulta de Produtos exibe todos os produtos cadastrados no sistema. O print mostra uma tabela com as informações dos produtos, como ID, Nome, Grupo, Marca, Quantidade e Preço. O usuário pode filtrar os produtos utilizando a barra de busca, filtrando por nome, marca, grupo ou código de barras. Quando o usuário seleciona um produto, ele pode visualizar os Detalhes do Produto ou Editar ou Excluir o produto, dependendo da ação desejada.

Figura 9 – Tela de Detalhes do Produto

A Tela de Detalhes de Produto exibe informações detalhadas de um produto específico. O print mostra a tela com os dados do produto selecionado, como nome, grupo, marca, código de barras, quantidade, preço e descrição. Essa tela oferece uma visão completa das informações do produto e é acessada a partir da Tela de Consulta de Produtos. Caso o usuário precise editar o produto, ele pode ser redirecionado para a Tela de Edição de Produto.

Figura 10 – Tela de Edição do Produto

A Tela de Edição de Produto permite que o usuário edite as informações de um produto já cadastrado. O print mostra os campos preenchidos com as informações do produto selecionado e os campos editáveis. O usuário pode alterar o nome, preço, quantidade, entre outras informações do produto. Após fazer as alterações, o usuário clica em Salvar Alterações, e o sistema atualiza os dados do produto no arquivo produtos.json.

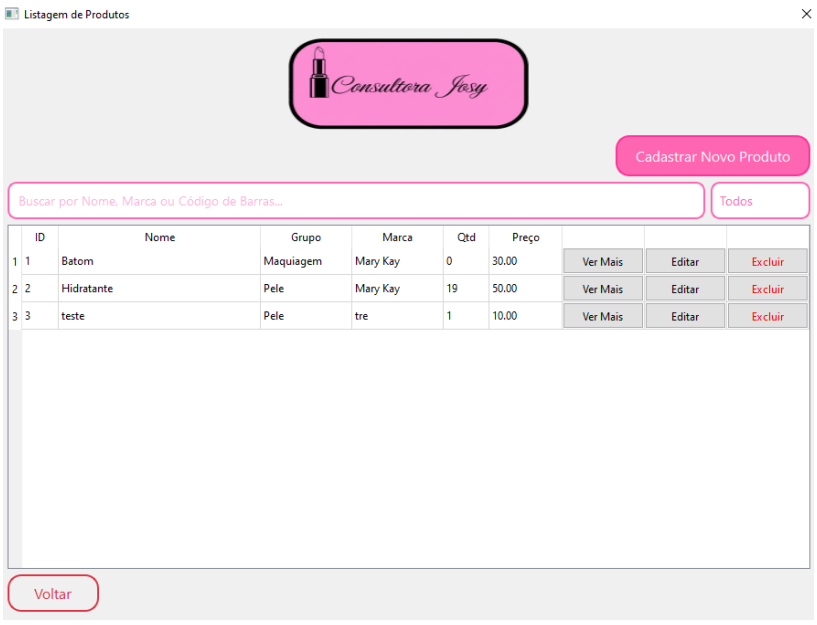

Figura 10 – Tela de Cadastros

A Tela de Cadastros visualize os produtos cadastrados, clique para ver mais detalhes do produto, clique para editar o produto, exclua o produto, e cadastre novos produtos no sistema..

As capturas de tela fornecem uma visão clara e detalhada das diversas funcionalidades do sistema, mostrando como o usuário interage com o sistema de controle de estoque. Cada tela foi projetada para ser intuitiva e eficiente, garantindo uma navegação fluída e fácil para o usuário. Desde o login até o cadastro, movimentação, consulta e edição de produtos, o sistema oferece todas as ferramentas necessárias para um gerenciamento completo de estoque. O projeto conseguiu atingir seus objetivos, proporcionando uma experiência funcional e intuitiva de controle de estoque, com foco na usabilidade e na facilidade de navegação.

4 CONCLUSÃO

O projeto desenvolvido tem como objetivo fornecer um sistema eficiente e prático para o controle de estoque de produtos. Durante o desenvolvimento do sistema, foi possível consolidar os conhecimentos adquiridos em Programação Orientada a Objetos (POO) e C++, utilizando as técnicas e boas práticas de programação para criar uma aplicação modular e escalável. O uso de C++ e POO proporcionou uma experiência prática e profunda no design e implementação de sistemas orientados a objetos, o que é essencial para o aprimoramento acadêmico, especialmente para um domínio tão complexo quanto o desenvolvimento de software.

A aplicação de conceitos de POO permitiu a construção de classes com responsabilidades bem definidas, como o cadastro de produtos, a movimentação de estoque, a consulta de produtos, e o gerenciamento de usuários. Isso resultou em um código mais organizado, fácil de manter e expandir, além de facilitar a detecção e correção de erros.

A biblioteca Qt desempenhou um papel fundamental na criação da interface gráfica do sistema, oferecendo ferramentas poderosas e flexíveis para a construção

de telas interativas, validação de dados e comunicação com arquivos JSON para persistência de dados. Além disso, o C++ garantiu uma implementação eficiente, com controle preciso de recursos e melhor desempenho na manipulação de grandes volumes de dados, essenciais para o gerenciamento de estoque.

No entanto, como em qualquer projeto de software, sempre há espaço para melhorias. Algumas áreas foram identificadas para aprimoramento, que, quando implementadas, podem proporcionar uma experiência ainda mais fluida e eficiente para o usuário.

4.1 Melhorias na Sincronização da Troca de Telas

Atualmente, ao voltar de uma tela para a anterior, o sistema exige que o usuário clique duas vezes no botão "Voltar", o que pode ser um pouco inconveniente. Uma melhoria importante seria otimizar a navegação entre telas para garantir uma transição mais suave e intuitiva, com a necessidade de um único clique para retornar à tela anterior. Além disso, a implementação de **modo de tela cheia** pode melhorar a experiência do usuário, tornando o sistema mais imersivo e proporcionando um layout mais atrativo, sem distratores.

4.2 Adição de Logs de Alterações de Estoque

Uma funcionalidade que poderia ser útil seria a implementação de logs de alterações de estoque, onde o sistema registraria todas as movimentações realizadas, como entradas e saídas de produtos, com a data, a quantidade alterada e o responsável pela ação. Esses registros seriam úteis para rastrear o histórico de movimentações e garantir maior transparência e controle sobre o estoque. A captura dessas informações também ajudaria no caso de auditorias e na análise do comportamento do estoque ao longo do tempo.

4.3 Melhoria no Design Visual e Layout

Embora o design atual do sistema seja funcional, há espaço para melhorias estéticas. A interface gráfica poderia ser aprimorada com a inclusão de texturas refinadas, animações suaves e transições dinâmicas para tornar o sistema visualmente mais agradável e moderno. A melhoria no design dos botões e a introdução de ícones ou elementos gráficos também poderiam tornar a navegação mais intuitiva e agradável. Essas melhorias seriam especialmente importantes para tornar o sistema mais atrativo para os usuários finais, como no caso da consultora de uma empresa de cosméticos, garantindo uma boa experiência de uso.

4.4 Considerações Finais

Em resumo, o sistema de controle de estoque desenvolvido é uma implementação funcional e eficiente, que conseguiu atender aos objetivos iniciais, oferecendo uma ferramenta útil para o gerenciamento de produtos em estoque. A aplicação de POO e C++ foi essencial para o sucesso do projeto, permitindo que o código fosse estruturado de maneira clara e organizada.

Apesar das conquistas, várias melhorias podem ser feitas, tanto na interface de usuário quanto nas funcionalidades do sistema, como a otimização na troca de telas e a inclusão de logs de alterações de estoque. Essas melhorias não apenas melhorariam a experiência do usuário, mas também tornariam o sistema mais robusto e escalável, com foco em expandir suas capacidades no futuro.

Este projeto proporcionou uma oportunidade valiosa de aplicar conceitos de Programação Orientada a Objetos e C++ em um contexto prático, resultando em uma ferramenta útil que, além de consolidar o aprendizado teórico, pode ser adaptada para diferentes tipos de usuários, como a consultora de uma empresa de cosméticos que pode usá-lo para gerenciar o estoque de produtos de maneira eficiente e organizada.

REFERÊNCIAS

Qt Documentation. *Official Qt Documentation.* Disponível em: https://doc.qt.io . Acesso em: 3 maio 2025.

Qt Creator Documentation. *Qt Creator IDE Documentation.* Disponível em: https://doc.qt.io/qtcreator/index.html . Acesso em: 7 maio 2025.

Video Tutorial. "Qt Creator Installation and Setup on Windows". *YouTube Video.* Disponível em: https://www.youtube.com/watch?v=ZoFykbMDoFg. Acesso em: 12 maio 2025.

Qt for Beginners. *Getting Started with Qt and Qt Creator.* Disponível em: https://www.qt.io/qt-for-beginners . Acesso em: 15 maio 2025.

Qt Creator IDE. *Qt Creator IDE Guide.* Disponível em: https://www.qt.io/qt-creator . Acesso em: 18 maio 2025.

Qt Installation Guide. *How to install Qt Creator on Windows, Linux, and macOS.* Disponível em: https://doc.qt.io/qt-5.15/qtdoc-installation-guide.html . Acesso em: 22 maio 2025.

Qt 5 Documentation. *Qt 5 API Reference.* Disponível em: https://doc.qt.io/qt-5.15/index.html . Acesso em: 24 maio 2025.

Stack Overflow. "Qt Creator Tag." *Qt Questions on Stack Overflow.* Disponível em: https://stackoverflow.com/questions/tagged/qt . Acesso em: 26 maio 2025.

Qt Development Tutorials. "Qt C++ Programming Tutorials." *YouTube Playlist.* Disponível em: https://www.youtube.com/playlist?list=PL0eT16YfDAzyXBZtIaj5e5Vdf17gmh0DA . Acesso em: 28 maio 2025.

Qt Community Forum. "Qt Developer Forum." *Qt Discussions and Questions.* Disponível em: https://forum.qt.io . Acesso em: 30 maio 2025.

Qt Documentation on GitHub. *Qt's Official GitHub Repositories.* Disponível em: https://github.com/qt . Acesso em: 31 maio 2025.

Qt Documentation. *JSON in Qt: Working with JSON in Qt.* Disponível em: https://doc.qt.io/qt-5/qjsondocument.html . Acesso em: 10 maio 2025.