INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

ANDRE LUIZ LIMA RODRIGUES

SISTEMA ATM

CAMPOS DO JORDÃO

ANO 2024

RESUMO

Este trabalho tem como objetivo o desenvolvimento de um sistema bancário baseado no modelo de caixa eletrônico (ATM) proposto no livro C++ Como Programar de H.M. Deitel. O sistema foi projetado para exemplificar a interação de um usuário com uma instituição bancária, oferecendo funcionalidades essenciais, como consulta de saldo, depósitos e saques. A implementação foi realizada utilizando os conceitos de Programação Orientada a Objetos (POO), com ênfase na modularização do sistema, que foi dividido em várias classes e funções-membro. O trabalho abordou a aplicação de técnicas de encapsulamento, herança, polimorfismo e abstração, fundamentais para o desenvolvimento de sistemas escaláveis e de fácil manutenção. A modularização permitiu o desenvolvimento independente das partes do sistema, ao mesmo tempo em que garantiu sua integração eficiente. Além disso, a documentação foi elaborada conforme as normas da ABNT e os requisitos do Instituto Federal, assegurando a conformidade acadêmica. O projeto não se limitou à criação de um sistema funcional, mas também serviu como estudo de caso para a aplicação de conceitos de POO e boas práticas de engenharia de software, promovendo a consolidação dos conhecimentos adquiridos na disciplina de Programação Orientada a Objetos.

Palavras-Chave: Programação Orientada a Objetos; C++; Sistema Bancário; Caixa Eletrônico; Modularização; Engenharia de Software; POO.

ABSTRACT

This work aims to develop a banking system based on the ATM model proposed in H.M. Deitel's book C++ How to Program. The system was designed to exemplify user interaction with a financial institution, offering essential functionalities such as balance inquiries, deposits, and withdrawals. The implementation was carried out using Object-Oriented Programming (OOP) concepts, with an emphasis on modularity, dividing the system into multiple classes and member functions. The work focused on applying techniques such as encapsulation, inheritance, polymorphism, and abstraction, which are essential for the development of scalable and maintainable systems. The modularization allowed for independent development of the system components while ensuring their efficient integration. Additionally, the documentation was prepared in accordance with ABNT standards and the requirements of the Federal Institute, ensuring academic compliance. The project not only aimed to create a functional system but also served as a case study for applying OOP concepts and software engineering best practices, reinforcing the knowledge acquired in the Object-Oriented Programming course.

Keywords: Object-Oriented Programming, C++; Banking System; ATM; Modularity; Software Engineering; OOP.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE SIGLAS

IFSP Instituto Federal de Educação, Ciência e Tecnologia de São Paulo

POO *Programação Orientada a Objetos*

ABNT *Associação Brasileira de Normas Técnicas*

SUMÁRIO

1 INTRODUÇÃO

O uso de ferramentas e técnicas de programação tem se tornado cada vez mais relevante em diversas áreas do conhecimento, especialmente no campo da computação e da engenharia de software. A Programação Orientada a Objetos (POO) é um dos paradigmas mais amplamente adotados na construção de sistemas modernos, permitindo que desenvolvedores criem software de forma modular, reutilizável e de fácil manutenção. Este paradigma, que organiza o código em torno de "objetos" que interagem entre si, promove não apenas a eficiência no desenvolvimento, mas também uma melhor compreensão dos processos envolvidos na implementação de sistemas complexos. Dentro desse contexto, a disciplina de Programação Orientada a Objetos busca capacitar os alunos para aplicar esses conceitos de forma prática em projetos reais, sendo fundamental para a formação de um programador habilidoso.

Para a realização e conclusão dessa disciplina, foi desenvolvido um projeto que se baseia no trabalho final do curso superior de Análise e Desenvolvimento de Sistemas, seguindo as normas e diretrizes estabelecidas pela ABNT. O projeto teve como objetivo a implementação do sistema de caixa eletrônico (ATM), descrito no livro C++ Como Programar, de Deitel, que serve como exemplo prático para ilustrar os conceitos abordados ao longo do curso. Neste livro, o autor utiliza o exemplo de um sistema ATM para demonstrar, de maneira prática, como os conceitos de POO, como classes, objetos, herança e polimorfismo, podem ser aplicados em um projeto real, facilitando a compreensão desses tópicos e mostrando sua relevância no desenvolvimento de softwares complexos.

O sistema ATM descrito por Deitel permite realizar funcionalidades comuns em caixas eletrônicos, como consulta de saldo, depósitos e saques. Ao implementar este sistema, o objetivo foi não apenas reproduzir o código apresentado no livro, mas também aprofundar o entendimento sobre os conceitos de POO, ao mesmo tempo em que se propôs a personalização e adaptação do projeto às necessidades do desenvolvimento acadêmico. Dessa forma, o trabalho foi dividido em duas partes: a primeira consiste na entrega dos arquivos de código, com a implementação completa do sistema ATM; a segunda parte é um relatório acadêmico, no qual são discutidos os principais pontos do projeto, as etapas de implementação, os desafios enfrentados e as soluções encontradas. Essa estrutura permite uma reflexão crítica sobre o processo de desenvolvimento, destacando a aplicação prática dos conceitos de Programação

Orientada a Objetos e evidenciando o aprendizado obtido ao longo da realização do projeto.

1.1 Objetivos

Este trabalho tem como objetivo principal o desenvolvimento do sistema proposto por Deitel em seu livro C++ Como Programar, que descreve a construção de um sistema de caixa eletrônico (ATM) modularizado. O sistema é estruturado a partir de diversas classes e funções-membro, que interagem entre si para realizar as funcionalidades típicas de um caixa eletrônico, como consulta de saldo, depósitos, saques e transferências. O desenvolvimento do sistema visa garantir a implementação de todas as funcionalidades descritas no livro de forma eficiente, assegurando que cada operação seja realizada com precisão e que o sistema atenda aos requisitos propostos para um caixa eletrônico real.

Além disso, o trabalho visa à elaboração de um documento acadêmico formal, em conformidade com as normas e diretrizes estabelecidas pelo Instituto Federal e pela ABNT. Este documento será estruturado de acordo com os padrões exigidos para trabalhos acadêmicos, garantindo uma apresentação clara e organizada, com todos os elementos necessários, como introdução, desenvolvimento, conclusões e referências, para proporcionar uma compreensão completa e detalhada do processo realizado.

Por fim, o trabalho propõe uma análise crítica sobre os desafios enfrentados durante o desenvolvimento do sistema. O terceiro objetivo específico visa refletir sobre as dificuldades encontradas ao longo do processo de implementação, como questões relacionadas à integração das diferentes classes e funcionalidades do sistema, e as soluções adotadas para superá-las. Além disso, será feita uma avaliação do aprendizado adquirido com o projeto, destacando como o uso da programação orientada a objetos (POO) e a aplicação dos conceitos abordados no livro de Deitel contribuíram para o desenvolvimento do sistema. Essa reflexão crítica tem como propósito não apenas relatar as dificuldades, mas também demonstrar a evolução do conhecimento técnico e acadêmico ao longo da execução do projeto, evidenciando a capacidade de resolução de problemas e a aplicação prática dos conceitos de POO no contexto real do desenvolvimento de software.

Para a consecução deste objetivo foram estabelecidos os objetivos específicos:

- Desenvolver o sistema ATM conforme o modelo proposto por Deitel, implementando as funcionalidades descritas no livro de forma modular e eficiente;
- Elaborar um documento acadêmico que esteja em conformidade com as normas estabelecidas pelo Instituto Federal e a ABNT, respeitando as diretrizes de formatação, estruturação e apresentação;
- Realizar a análise crítica e documentar os principais desafios enfrentados durante o desenvolvimento do sistema, refletindo sobre as soluções adotadas e o aprendizado adquirido ao longo do processo;

1.2 Justificativa

A relevância deste trabalho está intrinsecamente ligada à demonstração da evolução dos conhecimentos adquiridos em Programação Orientada a Objetos (POO), essencial para a formação de um programador capacitado na construção de sistemas complexos e eficientes. O projeto, que se concentra na implementação de um sistema de caixa eletrônico (ATM), visa aplicar os fundamentos teóricos adquiridos ao longo do curso, proporcionando uma integração prática entre teoria e prática. Através da construção deste sistema, é possível consolidar a compreensão de conceitos avançados de desenvolvimento de software, como modularização, encapsulamento, herança e polimorfismo, permitindo a implementação de soluções robustas e escaláveis. Além disso, o trabalho exige a elaboração de toda a documentação acadêmica necessária, cumprindo rigorosamente as normas e os padrões estabelecidos, com o intuito de garantir a entrega de um material completo, bem estruturado e de alta qualidade. Este processo não só valida os conhecimentos teóricos adquiridos, mas também proporciona a oportunidade de aprimorar as habilidades práticas na criação de sistemas eficientes. Ao focar na modularização e na reutilização de código, o projeto destaca-se como um exercício de boas práticas de engenharia de software, que são fundamentais para o desenvolvimento de programas sustentáveis e de fácil manutenção. Dessa forma, o trabalho não apenas atende aos objetivos acadêmicos do curso, mas também prepara o aluno para os desafios práticos encontrados no mercado de desenvolvimento de software, fornecendo uma base sólida para futuras iniciativas profissionais.

1.3 Aspectos Metodológicos

O presente trabalho adotou uma abordagem metodológica de caráter bibliográfico para a construção da fundamentação teórica, utilizando livros e artigos especializados que forneceram a base conceitual necessária para a compreensão dos tópicos abordados. A pesquisa bibliográfica desempenhou um papel essencial ao fornecer uma visão ampla e detalhada dos conceitos fundamentais da programação orientada a objetos, bem como das melhores práticas de desenvolvimento de sistemas. Para a seção prática, a metodologia aplicada foi centrada no desenvolvimento e implementação do sistema proposto, em que o foco principal foi a aplicação direta dos conhecimentos teóricos adquiridos. O desenvolvimento do sistema não se limitou a uma simples construção de código, mas visou integrar os conceitos de POO com as necessidades práticas do sistema bancário proposto, proporcionando uma experiência completa e funcional. A parte prática envolveu a criação de uma solução funcional, que permitiu testar e validar os conceitos previamente estudados, evidenciando a relação entre teoria e prática. Além disso, o processo de implementação exigiu a adaptação de várias ferramentas e a resolução de problemas técnicos que enriqueceram a experiência de aprendizagem. Esse enfoque permitiu não apenas a aplicação dos conceitos, mas também a identificação de desafios que proporcionaram uma reflexão crítica sobre a execução do projeto. Por fim, ao alinhar a teoria com a prática, o trabalho demonstrou a viabilidade do sistema e consolidou a compreensão dos conceitos fundamentais de programação, permitindo a construção de um software funcional e eficiente, que reflete as diretrizes acadêmicas e os objetivos do curso.

1.4 Aporte Teórico

O embasamento teórico deste trabalho acadêmico fora predominantemente fundamentado em livros especializados e documentações de plataformas amplamente disponíveis online. Para a realização do sistema proposto, utilizou-se como principal referência o livro C++ Como Programar, de Deitel (2006). Esta obra forneceu tanto o código-fonte quanto explicações objetivas sobre as funções implementadas, além de proporcionar um entendimento aprofundado sobre o paradigma da Programação Orientada a Objetos (POO), um componente essencial para o desenvolvimento de aplicações modernas. Outro ponto relevante é que o livro de Deitel também se destaca

como uma importante fonte de consulta sobre a linguagem de programação C++, reconhecida por sua robustez, confiabilidade, rapidez e flexibilidade, características que a tornaram a escolha ideal para o desenvolvimento deste sistema.

Outro autor de grande importância para o entendimento da POO é Maitino Neto (2018), que, em seu livro Programação Orientada a Objetos, adota uma linguagem acessível e de fácil compreensão, tornando temas complexos e conceituais mais claros e de fácil absorção. Sua abordagem simplifica a compreensão de conceitos como polimorfismo e herança, facilitando a aplicação prática desses conceitos no desenvolvimento de sistemas orientados a objetos.

Complementando o referencial teórico, foi consultado o livro Engenharia de Software, de Ian Sommerville (2011), que aborda de forma clara e aprofundada conceitos fundamentais da engenharia de software, como diagramas de casos de uso e, neste trabalho, o diagrama de classes. A obra contribuiu significativamente para a compreensão e desenvolvimento desse diagrama, tornando seu processo mais tangível e acessível. Embora o uso do diagrama de classes tenha sido relativamente limitado neste trabalho, a obra de Sommerville se revelou uma valiosa fonte de conhecimento na área de engenharia de software, oferecendo um embasamento técnico importante.

Além das obras mencionadas, foram utilizadas as documentações digitais do Visual Studio Code (VSCode) e do Draw.io, disponíveis nos respectivos sites da Microsoft e do Draw.io. Essas documentações complementaram o entendimento sobre as funcionalidades e recursos dessas plataformas, que desempenharam papéis cruciais no desenvolvimento e na organização visual do sistema. O VSCode, como editor de código-fonte, proporcionou um ambiente eficiente e flexível para a escrita do código, enquanto o Draw.io foi utilizado para a criação dos diagramas necessários, facilitando a representação visual do sistema de forma clara e estruturada.

2 METODOLOGIA

Nesta seção, serão apresentadas de forma detalhada as considerações iniciais do projeto, juntamente com sua descrição, bem como as ferramentas utilizadas e os desafios enfrentados durante a sua execução.

2.1 Considerações Iniciais

Desde o surgimento da internet, soluções tecnológicas têm desempenhado um papel fundamental na resolução de problemas, variando desde questões simples até as mais complexas que os seres humanos enfrentam no cotidiano. Um exemplo notável é a possibilidade de realizar diversas atividades por meio de ferramentas online, que possibilitam a resolução de problemas no conforto do lar. Esse avanço torna mais acessível a execução de tarefas que, anteriormente, exigiam deslocamentos ou esforço significativo, como, por exemplo, ir a um banco.

Com esse cenário em mente, o autor do livro C++ Como Programar (H.M. Deitel) propõe a criação de um sistema bancário simples, visando exemplificar a relação entre um cliente e uma instituição financeira, além de consolidar os conceitos abordados ao longo do texto. O sistema, embora envolva uma certa complexidade devido à sua divisão em várias partes que compõem o sistema total, tem grande importância no processo de materialização dos conhecimentos de programação orientada a objetos (POO) e na implementação de sistemas modularizados. A divisão do sistema em módulos independentes permite que as diferentes funcionalidades sejam desenvolvidas, testadas e mantidas de maneira mais eficiente, contribuindo para o aprendizado de boas práticas de engenharia de software.

O desenvolvimento desse sistema, baseado nos conceitos de POO, oferece uma excelente oportunidade de aplicação dos princípios de encapsulamento, herança, polimorfismo e abstração, aspectos essenciais para o domínio de C++ e para a construção de programas escaláveis e de fácil manutenção. Além disso, o projeto facilita a compreensão da modularização, permitindo que diferentes partes do sistema sejam desenvolvidas separadamente, mas interajam de maneira eficaz para atender aos requisitos do usuário. Assim, a implementação do exemplo do livro não apenas serve como um estudo de caso para a aplicação de conceitos de programação, mas também como um exercício para aprimorar a capacidade de criar soluções tecnológicas que

atendem às necessidades do mundo real.

2.2 Descrição do Projeto

O sistema desenvolvido foi baseado no modelo de um caixa eletrônico (ATM) apresentado no livro de Deitel, no qual o foco principal é a interação de um usuário com um sistema bancário, oferecendo funcionalidades essenciais, como consultar o saldo da conta, realizar retiradas e depósitos de dinheiro, entre outras operações. De acordo com Deitel, a programação orientada a objetos (POO) é uma abordagem poderosa, pois permite a utilização de técnicas que otimizam o tempo de desenvolvimento e a quantidade de código, resultando em uma programação mais eficiente, refinada e elegante. A POO introduz conceitos como herança e polimorfismo, que, aliados ao uso de bibliotecas especializadas, facilitam a criação de software reutilizável e flexível, promovendo um desenvolvimento ágil e organizado.

O sistema ATM é composto por diversas classes e construtores, que representam as diferentes funcionalidades e componentes do sistema bancário. Cada classe tem um papel específico, com atributos e métodos que interagem entre si para proporcionar uma experiência fluida e eficiente ao usuário. A estrutura do sistema é modularizada, o que permite que diferentes partes do programa sejam desenvolvidas, testadas e mantidas de forma independente, mas que, ao mesmo tempo, funcionem de maneira integrada. Além disso, o uso da herança entre classes facilita a reutilização de código, o que torna o desenvolvimento mais rápido e menos propenso a erros.

Ao adotar uma abordagem modular e orientada a objetos, o projeto busca não apenas implementar um sistema funcional, mas também demonstrar os benefícios da POO na criação de software de fácil manutenção, escalabilidade e reutilização. O desenvolvimento do sistema ATM, portanto, não se limita a uma simples implementação de funcionalidades bancárias, mas serve também como uma ferramenta para aprimorar o entendimento dos conceitos fundamentais da programação orientada a objetos, bem como a aplicação de boas práticas de engenharia de software, ambas apresentadas na disciplina de Programação Orientada a Objetos.

2.3 Ferramentas Utilizadas

Para o desenvolvimento do sistema ATM, foram utilizadas diversas ferramentas que não apenas facilitam a criação de aplicações eletrônicas, mas também otimizam o tempo e os recursos durante o progresso do projeto. Entre as ferramentas empregadas, destaca-se o próprio livro de Deitel, que, além de fornecer os códigos necessários, oferece uma explicação clara e objetiva sobre cada um dos componentes do sistema, complementada por diagramas que facilitam a compreensão do funcionamento do projeto.

Outra ferramenta essencial foi o Visual Studio Code (VSCode), que, juntamente com a linguagem C++, desempenhou um papel fundamental na construção e finalização do código. O VSCode proporcionou um ambiente de desenvolvimento integrado eficiente, com recursos que facilitam a escrita e depuração do código, enquanto a linguagem C++ foi utilizada pela sua robustez e versatilidade, sendo ideal para a implementação de sistemas orientados a objetos como o ATM.

Além disso, uma parte significativa do desenvolvimento foi a elaboração do diagrama de classes, que foi desenvolvido na plataforma online draw.io. Este diagrama é crucial para representar visualmente a estrutura do sistema, suas classes e as relações entre elas, facilitando tanto a compreensão do design do sistema quanto a comunicação entre os desenvolvedores.

A seguir, será feito um aprofundamento sobre o uso e a importância de cada uma dessas ferramentas no processo de desenvolvimento do sistema ATM.

Linguagem C++: A linguagem C++ é uma extensão da linguagem C, desenvolvida por Bjarne Stroustrup na década de 1980, enquanto ele trabalhava nos Bell Laboratories. Stroustrup introduziu aprimoramentos significativos na linguagem, especialmente no que diz respeito ao paradigma de Programação Orientada a Objetos (POO). Embora o C++ tenha evoluído e adquirido características distintas em relação ao C, ele mantém uma forte herança da linguagem C, que foi originalmente criada por Dennis Ritchie, também nos Bell Laboratories, a partir de conceitos das linguagens B e BCPL.

Embora o C++ tenha perdido certa relevância no cenário atual, devido à popularização de outras linguagens de programação, ele continua sendo uma das mais poderosas e flexíveis. Sua velocidade e a capacidade de manipulação de recursos de baixo nível fazem do C++ uma escolha preferencial em áreas que exigem alta perfor-

mance, como o desenvolvimento de sistemas embarcados, jogos e aplicações financeiras. Mesmo em um contexto onde novas linguagens emergem constantemente, o C++ se mantém relevante devido à sua robustez e ao controle preciso sobre os recursos computacionais, o que o torna ideal para projetos que demandam eficiência e flexibilidade.

No desenvolvimento do sistema ATM, a linguagem C++ foi essencial para a implementação das funcionalidades e para a estruturação do código de maneira modular e orientada a objetos, o que facilita a manutenção e a expansão do sistema. A capacidade da linguagem de trabalhar com manipulação direta de memória e seu suporte ao paradigma de POO foram determinantes para a construção de um sistema eficiente e organizado, capaz de realizar operações bancárias de maneira segura e rápida (DEITEL; 2006).

Visual Studio Code: O Visual Studio Code (VSCode) é um editor de código-fonte desenvolvido pela Microsoft, projetado para ser leve, rápido e compatível com os principais sistemas operacionais, como Linux, Windows e macOS. Essas características foram fundamentais para sua popularidade, tornando-o um dos editores de código mais amplamente utilizados atualmente. Além disso, o VSCode oferece uma ampla variedade de atalhos, extensões e a possibilidade de personalização durante a instalação, o que proporciona grande flexibilidade e facilita sua adaptação às necessidades de diferentes profissionais de programação.

No contexto do desenvolvimento do sistema ATM, o VSCode foi uma ferramenta essencial. Sua interface simples e intuitiva, aliada a recursos avançados como depuração integrada, destaque de sintaxe e sugestões automáticas de código, permitiu um ambiente de desenvolvimento ágil e organizado. O editor também se mostrou altamente eficiente ao fornecer suporte completo para C++, além de permitir a integração com outras ferramentas, como controle de versão através do Git, o que facilitou a gestão do código-fonte ao longo do projeto. Dessa forma, o VSCode contribuiu significativamente para a produtividade no desenvolvimento do sistema, tornando o processo mais rápido e eficiente (MICROSOFT; 2024).

Draw.io: O Draw.io, atualmente conhecido como diagrams.net, foi desenvolvido pela JGraph, uma empresa fundada por David Benson, e teve sua primeira versão lançada em 2010. Trata-se de uma plataforma online amplamente utilizada para a criação e

manipulação de diagramas, destacando-se pela sua variedade de funcionalidades. Entre os recursos oferecidos, é possível criar fluxogramas, mapas mentais, diagramas de rede, plantas baixas, entre outros. A popularidade do Draw.io pode ser atribuída à sua versatilidade, especialmente nas áreas de desenvolvimento de software e infraestrutura de TI. A ferramenta também oferece diversos modelos predefinidos que facilitam a diagramação e a organização visual de processos e sistemas, tornando-a uma escolha preferencial para profissionais que necessitam de soluções rápidas e eficientes para representação gráfica.

No desenvolvimento do sistema ATM, o Draw.io foi utilizado para a elaboração do diagrama de classes, uma etapa fundamental para a visualização e estruturação do sistema. A ferramenta permitiu criar representações claras e detalhadas das classes e suas interações, facilitando a compreensão da arquitetura do software. Sua interface intuitiva e a capacidade de exportar os diagramas em diversos formatos contribuíram para a documentação eficaz do projeto. Assim, o uso do Draw.io foi essencial para a organização visual do funcionamento do sistema, garantindo que a estrutura do código fosse bem definida e facilmente comunicável por meios visuais (DRAW.IO; 2024).

Livro: O livro C++ Como Programar, de H. M. Deitel, é uma referência amplamente reconhecida no ensino de programação, especialmente na linguagem C++. A obra, que abrange desde os conceitos fundamentais até técnicas avançadas de programação, oferece uma abordagem clara e estruturada para o aprendizado da linguagem. Deitel apresenta não apenas os códigos necessários para a implementação de diversos exemplos, mas também fornece explicações detalhadas e acessíveis sobre cada um dos conceitos abordados, facilitando a compreensão de tópicos complexos como programação orientada a objetos, manipulação de arquivos, e estruturas de dados. A obra também é complementada por diagramas e exemplos práticos que ajudam a consolidar os conhecimentos adquiridos ao longo do estudo.

No contexto do desenvolvimento do sistema ATM, o livro de Deitel foi uma ferramenta essencial, fornecendo os fundamentos necessários para a implementação de um sistema bancário simples, mas eficaz. Através de exemplos práticos e explicações detalhadas, o autor possibilitou a aplicação de conceitos como classes, objetos, herança e polimorfismo, fundamentais para o desenvolvimento de software modular e

eficiente. Além disso, a abordagem didática do livro permitiu que os conceitos de programação orientada a objetos fossem aplicados diretamente no projeto, facilitando a construção do código e a organização das diferentes funcionalidades do sistema. O C++ Como Programar serviu, portanto, como um guia importante para a execução bem-sucedida do projeto ATM, ajudando na implementação das boas práticas de programação e no entendimento profundo da linguagem C++.

Para encerrar esta seção, é importante destacar que, apesar das dificuldades enfrentadas durante o desenvolvimento do sistema ATM, o projeto proporcionou uma valiosa oportunidade de aprendizagem e desenvolvimento técnico. A aplicação dos conceitos de Programação Orientada a Objetos (POO) foi fundamental para garantir a estruturação eficiente e escalável do sistema, e as ferramentas utilizadas, como o Visual Studio Code, o Draw.io e o livro de Deitel, desempenharam papéis essenciais ao facilitar o desenvolvimento e a organização do código. Embora os desafios encontrados, como a complexidade do material do livro e a segmentação do código em múltiplas partes, tenham exigido dedicação extra, eles também serviram como pontos de crescimento, ampliando a compreensão sobre a importância da modularização, reutilização de código e boas práticas de engenharia de software. Além disso, a experiência prática de superar obstáculos, como a execução do código no Prompt de Comando, fortaleceu a habilidade de resolver problemas técnicos de forma mais ágil e eficiente. A conclusão bem-sucedida deste projeto não só atendeu aos objetivos propostos, mas também contribuiu para a consolidação do conhecimento adquirido ao longo do curso, proporcionando uma experiência prática e enriquecedora para o desenvolvimento de soluções de software mais complexas no futuro.

3 PROJETO DESENVOLVIDO

Nesta seção, será apresentado o desenvolvimento do projeto, que consistiu na implementação do sistema de caixa eletrônico (ATM) descrito no livro de Deitel, incorporando todas as funcionalidades apresentadas e detalhadas na obra. A proposta envolveu a aplicação de conceitos de programação orientada a objetos, com o objetivo de demonstrar um domínio aprofundado sobre esses tópicos e evidenciar a evolução no conhecimento adquirido ao longo do processo de desenvolvimento. A construção do sistema incluiu a elaboração de todos os componentes descritos nos requisitos do projeto, garantindo que as funcionalidades fossem implementadas conforme as especificações e proporcionando uma experiência de uso fiel ao sistema proposto no livro. Além disso, foi desenvolvido um diagrama de classes que ilustra a estrutura do sistema, incluindo seus atributos e métodos, para fornecer uma visão detalhada da organização e interações entre os componentes do projeto.

3.1 Diagrama de Classes

Os diagramas de classes são ferramentas essenciais no desenvolvimento de sistemas modulares, pois permitem representar de forma simplificada objetos do mundo real, como clientes, contas ou transações, em uma abstração mais geral denominada classes. Segundo Sommerville (2011), o primeiro passo na elaboração de um diagrama de classes é observar o mundo real e identificar os objetos que podem existir no sistema, como um usuário ou um produto, entre outros. Esses objetos são então agrupados em classes, que representam não apenas um objeto específico, mas uma coleção de objetos com características semelhantes.

Para a criação do diagrama de classes do sistema ATM, foram aplicados os conceitos de abstração, representando o funcionamento do sistema e suas interações iniciais entre os diversos objetos. De acordo com Maitino Neto (2018), o desenvolvimento orientado a objetos exige a abstração da realidade de forma que seja manipulável e compreensível, ao mesmo tempo em que preserve a flexibilidade característica do paradigma orientado a objetos.

Assim, nesta seção, será apresentado o diagrama de classes do sistema ATM,

com uma explicação detalhada sobre o que ele representa, a razão de sua utilização e uma visão geral de seu conteúdo. O objetivo é esclarecer as principais dúvidas relacionadas a essa representação. O diagrama de classes é uma das representações mais importantes na modelagem de sistemas orientados a objetos, pois descreve a estrutura estática do sistema, incluindo suas classes, atributos, métodos e os relacionamentos entre eles. Ao detalhar o diagrama, será possível compreender melhor a organização interna do sistema ATM e entender como suas partes interagem para atender aos requisitos propostos no livro de Deitel. A seguir, será apresentado o diagrama de classes, extraído do livro de Deitel, que fornece uma representação esquemática das classes, contendo apenas seus nomes e as interações com outras classes.

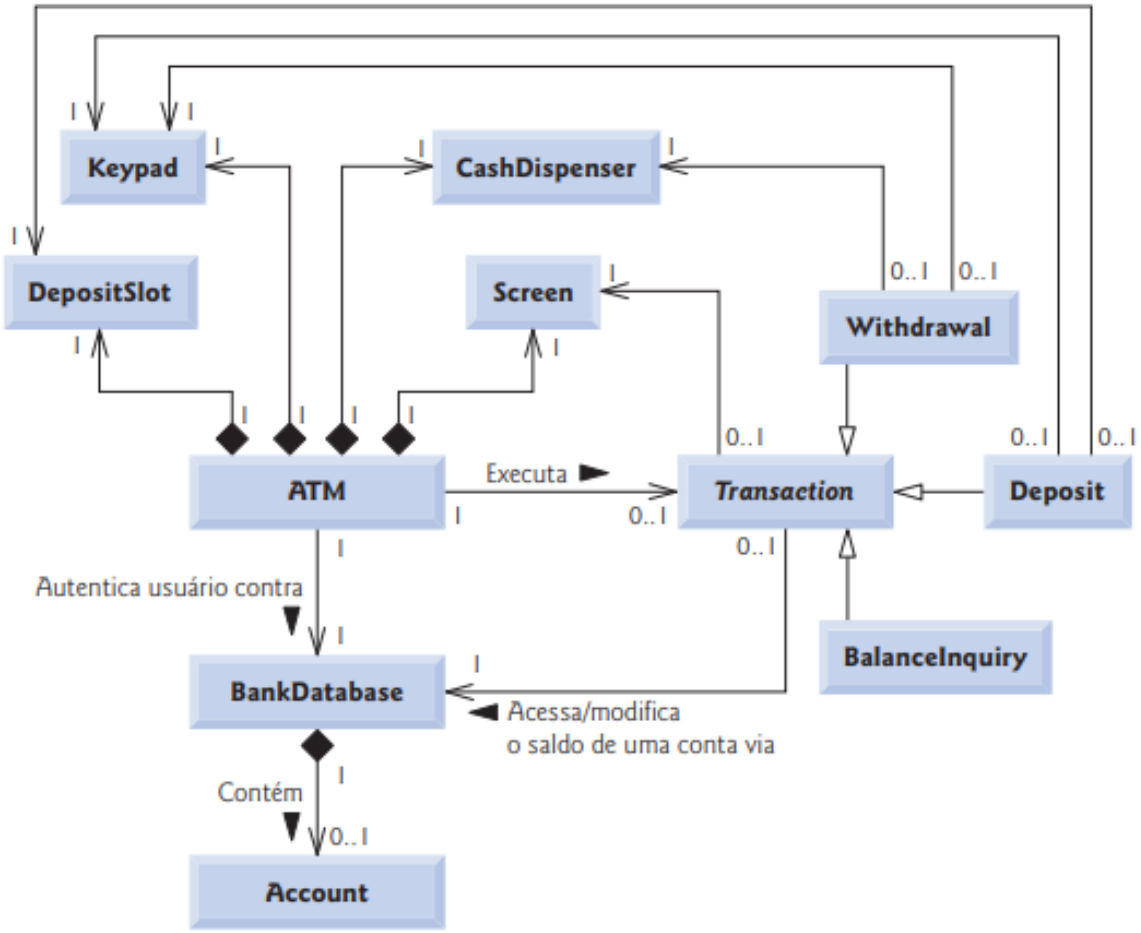

Figura 01 – Diagrama de Classes (DEITEL; 2006, p588).

Conforme mencionado, a imagem a seguir é meramente ilustrativa e mostra as classes em suas interações, variando de simples conexões a complexos relacionamentos envolvendo heranças, como as derivadas da classe Transaction. Embora seja apenas uma representação visual, ela oferece uma visão inicial do sistema ATM e de

como seus módulos se integram para formar o sistema completo e funcional. A seguir, apresenta-se um diagrama de classes mais detalhado, que inclui os atributos e métodos de cada classe.

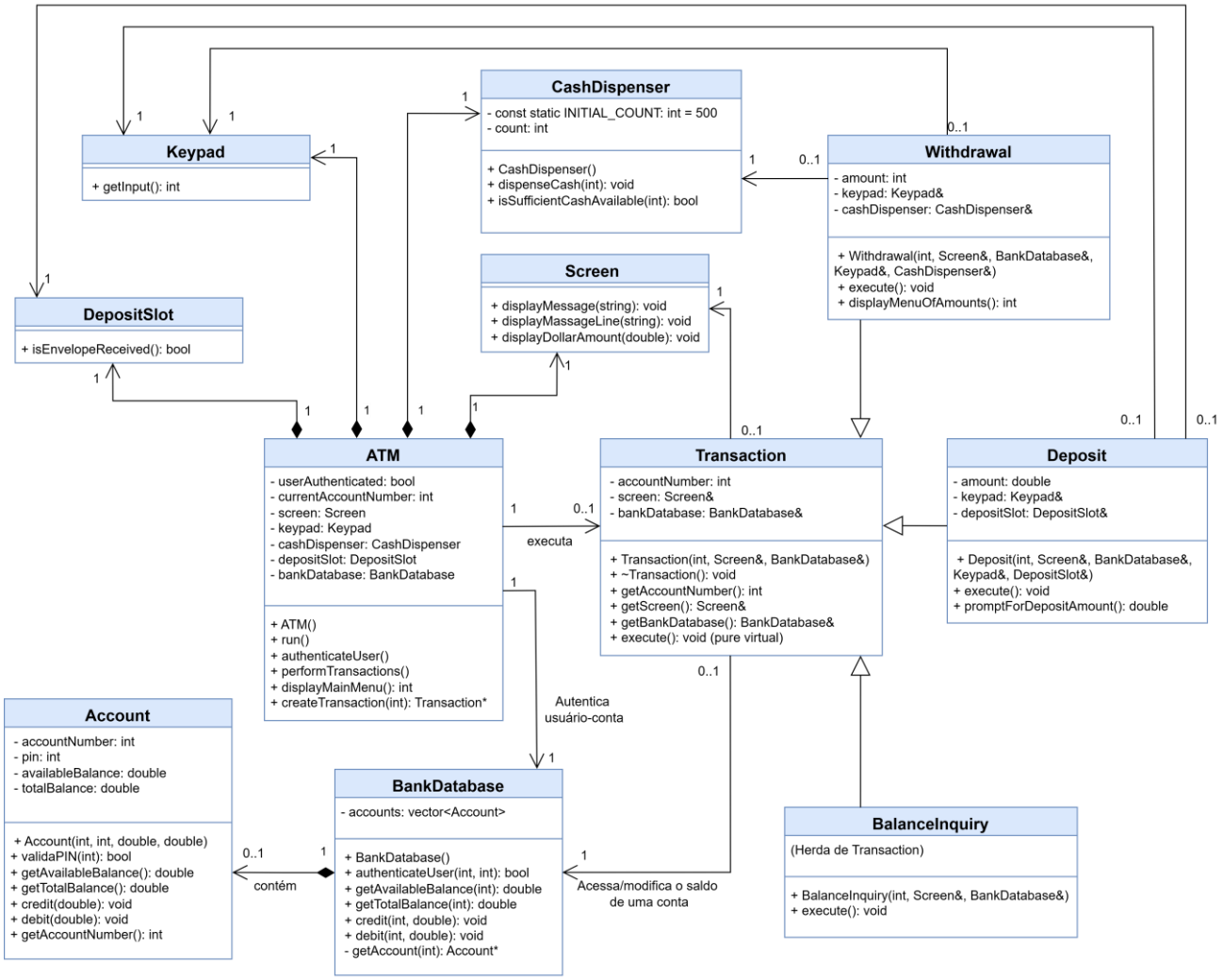

Figura 02 – Diagrama de Classes do Sistema (ATM).

O sistema do caixa eletrônico (ATM) é composto por diversas classes que trabalham juntas para realizar transações bancárias, como consultas de saldo, saques e depósitos. Cada classe tem um papel específico e se comunica com as outras para garantir o funcionamento eficiente do sistema. A classe principal do sistema é a ATM, que é responsável por iniciar e coordenar as operações do caixa eletrônico. Ela gerencia a autenticação do usuário e orquestra as transações. Para isso, a classe ATM interage

com outras classes como Screen, Keypad, CashDispenser, DepositSlot e BankDatabase.

A classe Screen tem a função de exibir as mensagens e informações na tela do caixa eletrônico, como instruções e resultados das transações. Já a classe Keypad permite que o usuário insira dados, como o PIN ou o valor a ser retirado ou depositado. O CashDispenser é responsável por entregar as cédulas durante a transação de saque, enquanto o DepositSlot representa o local onde o usuário coloca o envelope com o depósito. A classe BankDatabase armazena todas as informações sobre as contas bancárias, permitindo verificar o saldo, realizar saques e depósitos e validar os dados dos usuários.

Para lidar com as transações, existe a classe Transaction, que serve como uma classe base para todas as transações realizadas no caixa eletrônico. Ela define a estrutura básica que será compartilhada entre todas as transações, como o número da conta do usuário e o acesso ao banco de dados. A classe Transaction inclui um método chamado execute(), que será implementado nas subclasses específicas de transação, como BalanceInquiry, Withdrawal e Deposit.

A classe BalanceInquiry representa a transação de consulta de saldo. Quando o usuário deseja verificar o saldo da sua conta, essa classe executa a transação, acessando o banco de dados e exibindo o saldo na tela. A classe Withdrawal, por sua vez, é responsável pela transação de saque. Quando o usuário solicita um saque, a classe Withdrawal utiliza a classe Keypad para capturar o valor desejado e a classe CashDispenser para dispensar o dinheiro. Além disso, verifica se há saldo suficiente para realizar o saque. Já a classe Deposit é responsável pela transação de depósito. Quando o usuário deseja depositar um valor, a classe Deposit usa o Keypad para capturar o valor digitado e o DepositSlot para receber o envelope com o dinheiro. Em seguida, o valor é processado e adicionado ao saldo da conta.

A classe Account representa uma conta bancária e contém informações como o número da conta, o PIN de segurança, o saldo disponível e o saldo total. Ela também possui métodos que permitem validar o PIN do usuário, consultar os saldos e realizar operações como créditos e débitos. A classe BankDatabase, por sua vez, gerencia todas as contas bancárias armazenadas no sistema. Ela armazena os objetos da classe Account e oferece métodos para autenticar usuários, consultar saldos e realizar transações como depósitos e saques.

As classes ATM, Transaction, BalanceInquiry, Withdrawal, Deposit, Account, BankDatabase, Screen, Keypad, CashDispenser e DepositSlot formam um sistema integrado e modular, onde cada classe tem um papel específico. A classe ATM coordena a interação entre as outras classes, garantindo que o processo de transação bancária seja realizado de forma segura e eficiente. O BankDatabase armazena as informações das contas, enquanto a classe Account representa cada conta bancária individual. As classes de transação, como BalanceInquiry, Withdrawal e Deposit, executam as operações específicas, acessando e modificando os dados das contas conforme necessário. As classes de entrada e saída, como Screen, Keypad, CashDispenser e DepositSlot, são responsáveis pela interação do usuário com o sistema físico do caixa eletrônico.

Essas classes trabalham de forma integrada, com cada uma contribuindo para uma parte do processo, garantindo que o usuário possa realizar suas transações bancárias de forma simples e segura. No apêndice deste trabalho, será apresentado um detalhamento completo de cada classe, com seus atributos e métodos, para fornecer uma compreensão mais profunda de como cada componente do sistema funciona.

3.2 Desenvolvimento do Sistema (ATM)

O desenvolvimento e a construção do código do sistema ATM, baseado totalmente no livro de Deitel, iniciaram com a instalação e configuração do Visual Studio Code (VSCode) e da linguagem de programação C++. Após essa etapa, o processo de desenvolvimento foi iniciado com a criação das classes e a definição de seus respectivos construtores. O código foi organizado em dois tipos principais de arquivos: os arquivos de cabeçalho (.hpp) e os arquivos de implementação (.cpp). Nos arquivos .hpp, são declarados os cabeçalhos das classes, incluindo variáveis e métodos. Já nos arquivos .cpp, são implementados os construtores e as funções-membro das classes, ou seja, os métodos são desenvolvidos, possibilitando a execução do programa. Cada construtor corresponde à sua respectiva classe, e, quando necessário, outras classes são incluídas para garantir o funcionamento adequado. Isso é feito por meio da diretiva #include, que permite a inclusão de outros arquivos de código.

Após essa explicação inicial sobre como o C++ permite trabalhar com classes e objetos no paradigma de orientação a objetos, seguimos para o desenvolvimento

do sistema. O trabalho começou com a implementação da classe ATM e seu construtor, definidos nos arquivos ATM.hpp e ATM.cpp, que formam a base do sistema. A classe ATM é responsável por orquestrar a execução do sistema e interagir com outros componentes, como a classe BankDatabase. Em seguida, foram implementadas as classes Screen e Keypad, que, apesar de simples, desempenham funções essenciais: a Screen exibe as informações e interações do usuário na tela do caixa eletrônico, enquanto a Keypad permite que o usuário insira dados e faça escolhas durante a interação com o sistema.

Com o avanço do desenvolvimento, novas classes e funções-membro foram implementadas, como a BankDatabase, Account e Transaction. Essas classes, embora com funções distintas, trabalham de forma interligada para garantir que o sistema funcione de maneira ordenada e eficiente, proporcionando ao usuário uma experiência agradável. A classe BankDatabase gerencia o acesso aos dados bancários, enquanto a Account lida com as informações da conta do usuário, e a Transaction realiza operações como saques e depósitos. Essas interações entre as classes são essenciais para o funcionamento coeso do sistema como um todo.

À medida que o desenvolvimento avançava, outras classes e funções membro foram adicionadas ao sistema. Modificações no código eram necessárias para atender às especificações do livro de Deitel, que introduzia novos conceitos como herança, os quais foram integrados ao sistema conforme o necessário. Por fim, foi criado o programa ATMCaseStudy.cpp, que integra todo o código desenvolvido até aquele momento e permite a execução do sistema. Esse arquivo inicializa a classe ATM e executa a função run(), que é a responsável por executar as operações principais do sistema, como saques e depósitos.

Vale ressaltar que, para executar o programa, é necessário compilar o código utilizando o terminal de comando (CMD ou Prompt de Comando). Para isso, é preciso navegar até o diretório onde os arquivos estão localizados e executar o comando: “g++ ATMCaseStudy.cpp ATM.cpp Screen.cpp Keypad.cpp CashDispenser.cpp DepositSlot.cpp Account.cpp BankDatabase.cpp Transaction.cpp BalanceInquiry.cpp Withdrawal.cpp Deposit.cpp -o ATM”. Este comando compila os arquivos fontes e gera um arquivo executável chamado ATM. Para rodar o programa, basta digitar ATM.exe no terminal, e o sistema estará pronto para ser utilizado.

3.3 Desafios no Desenvolvimento do Sistema (ATM)

Os desafios enfrentados durante a elaboração e conclusão do sistema ATM foram, em sua maioria, de natureza pontual. Um dos principais obstáculos foi o estilo de escrita adotado no livro, que, embora técnico, às vezes gerava certa dificuldade na compreensão dos conceitos apresentados. Esse aspecto, ao invés de facilitar a assimilação, muitas vezes aumentava a distância entre os temas abordados. Além disso, o sistema estava segmentado em várias classes e funções, distribuídas ao longo de diferentes capítulos, o que exigia um esforço considerável para localizar e integrar as partes do código necessárias. Esse formato exigiu um tempo significativo de busca, o que não foi o meu caso, uma vez que a versão do material que consultei disponibilizava o código completo em apêndices, facilitando o processo. Outro ponto que apresentou desafios foi a minha limitada experiência em executar arquivos de código por meio do Prompt de Comando (CMD), o que causou um pequeno atraso na execução final do sistema. Para superar essa dificuldade, foram necessárias algumas consultas e orientações adicionais, o que facilitou o progresso na conclusão do projeto.

Para finalizar esta subseção e a seção como um todo, é importante destacar que, apesar dos desafios enfrentados, o processo de desenvolvimento do sistema ATM foi enriquecedor e instrutivo. Cada obstáculo apresentou uma oportunidade de aprendizado, permitindo aprimorar habilidades técnicas e de resolução de problemas. O entendimento profundo dos conceitos de Programação Orientada a Objetos (POO), juntamente com as dificuldades encontradas na execução e integração do código, contribuiu para o crescimento acadêmico e profissional. O sucesso na conclusão do projeto reflete o esforço contínuo e a capacidade de superar as adversidades, consolidando os conhecimentos adquiridos ao longo do processo de desenvolvimento.

4 RESULTADOS OBTIDOS

Os resultados do desenvolvimento do sistema ATM são evidenciados pelos códigos elaborados, acompanhados de imagens que demonstram o sistema em funcionamento. Esses resultados deixam claro a plena funcionalidade do sistema, destacando o alcance bem-sucedido do objetivo principal deste trabalho: a implementação do sistema de caixa eletrônico descrito no livro C++ Como Programar, de Deitel.

4.1 Apresentação de Figuras

Nesta subseção, serão apresentadas as imagens do sistema em funcionamento, com o objetivo de ilustrar suas operações e comprovar a eficácia das funcionalidades implementadas. Essas imagens visam fornecer uma visão clara do sistema em ação, demonstrando como ele responde às interações do usuário e executa as tarefas propostas. A seguir, encontram-se as imagens acompanhadas de suas respectivas descrições detalhadas:

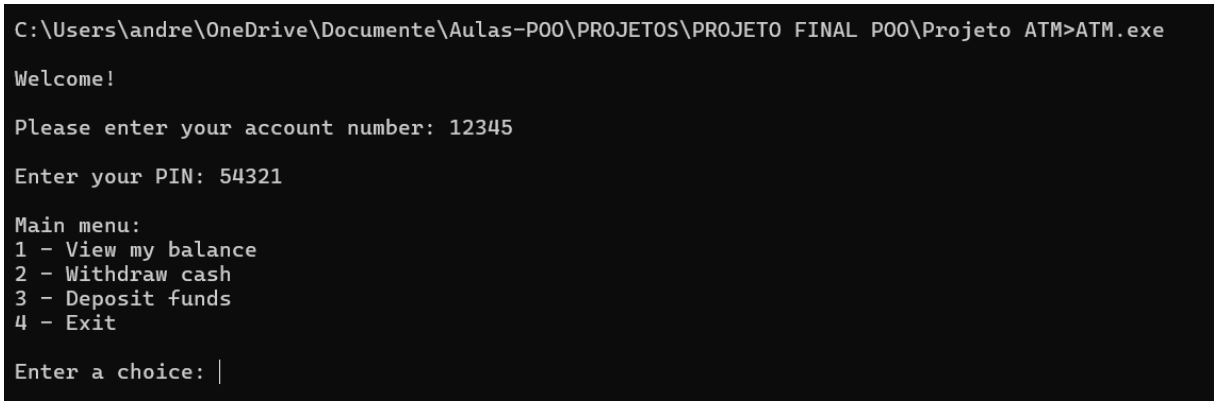

Figura 03 - Menu do Sistema.

A Figura 03 apresenta o sistema após o usuário inserir seu número de conta e senha. Com a autenticação bem-sucedida, o sistema exibe o menu inicial, o qual oferece as opções de consultar saldo, realizar saque, efetuar depósito ou encerrar a sessão. Esse menu permite ao usuário escolher a ação desejada, proporcionando uma navegação intuitiva e facilitando a interação com as funcionalidades do caixa eletrônico, conforme suas necessidades. Além disso, o design do menu é pensado para ser

claro e acessível, com botões bem definidos e opções de fácil compreensão. O sistema visa otimizar a experiência do usuário, garantindo que cada ação seja realizada de forma simples e rápida. Isso torna a utilização do caixa eletrônico mais eficiente, reduzindo o tempo necessário para a execução das operações.

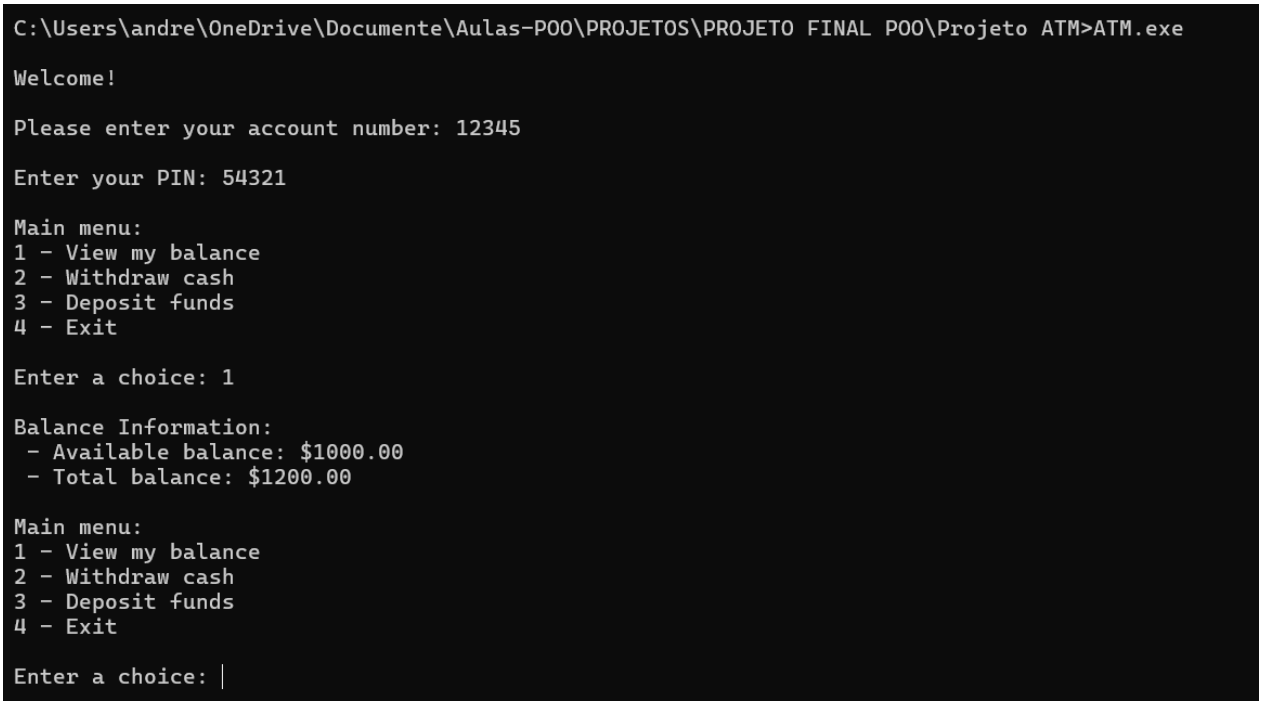

Figura 04 - Exemplo de Visualização do Saldo.

A Figura 04 ilustra o sistema após o usuário realizar uma interação, especificamente a consulta de saldo. Neste momento, o sistema exibe as informações solicitadas, mostrando tanto o saldo disponível quanto o saldo total da conta, proporcionando ao usuário uma visão clara e detalhada de sua situação financeira atual. Essas informações são fundamentais para que o usuário tome decisões informadas sobre suas próximas ações no sistema. O saldo disponível indica o valor que o usuário pode utilizar, enquanto o saldo total reflete o montante total na conta, incluindo eventuais valores não disponíveis para transações imediatas. Após a exibição dessas informações, o sistema retorna ao menu inicial, aguardando uma nova ação por parte do usuário. O menu oferece opções como realizar saques, depósitos ou consultar outros dados, permitindo uma navegação intuitiva e fluida. O design do sistema visa otimizar a experiência do usuário, tornando a interação rápida e eficiente. O sistema permanece disponível para novas interações até que o usuário decida encerrar a

sessão, dessa forma formando um loop, que fica rodando à espera de um novo usuário que decida interagir com o sistema. O ciclo se repete continuamente, garantindo que o sistema esteja sempre pronto para fornecer as informações e funcionalidades desejadas.

```
Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice: 3

Please enter a deposit amount in CENTS (or 0 to cancel): 100

Please insert a deposit envelope containing $1.00 in the deposit slot.

Your envelope has been received.
NOTE: The money just will not be available until we
verify the amount of any enclosed cash, and any enclosed checks clear.

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice: 1

Balance Information:
 - Available balance: $1000.00
 - Total balance: $1201.00

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice:
```

Figura 05 - Exemplo de um Depósito.

A imagem ilustra uma situação em que o usuário realizou um depósito no valor de 1 dólar na conta, demonstrando o funcionamento da funcionalidade de depósito implementada no sistema. Esta funcionalidade foi projetada para permitir que os usuários efetuem depósitos de forma simples e segura, adicionando valor à sua conta diretamente através do caixa eletrônico. O sistema, como exemplificado nesta situação, oferece essa opção como parte de seu conjunto de funcionalidades, permitindo aos usuários realizar depósitos a qualquer momento durante sua interação com o caixa

eletrônico. A implementação desta funcionalidade visa proporcionar maior conveniência e flexibilidade para os usuários ao gerenciar suas finanças de maneira prática. Como penúltimo exemplo de demonstração do funcionamento do sistema ATM, se encontra o saque de dinheiro oriunda das funcionalidades implantadas.

```
Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice: 2

Withdrawal options:
1 - $20
2 - $40
3 - $60
4 - $100
5 - $200
6 - Cancel transaction

Choose a withdrawal option (1-6): 1

Please take your cash from the cash dispenser.

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice: 1

Balance Information:
 - Available balance: $980.00
 - Total balance: $1180.00

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice:
```

Figura 06 – Exemplo de Saque de Dinheiro.

Esta imagem, assim como as anteriores, ilustra uma das funcionalidades mencionadas nas seções anteriores, especificamente a de saque, que pode ser considerada

uma das mais características e marcantes de um caixa eletrônico. Embora os exemplos apresentados sejam simples, eles cumprem eficazmente a função de demonstração, evidenciando o correto funcionamento das funcionalidades implementadas.

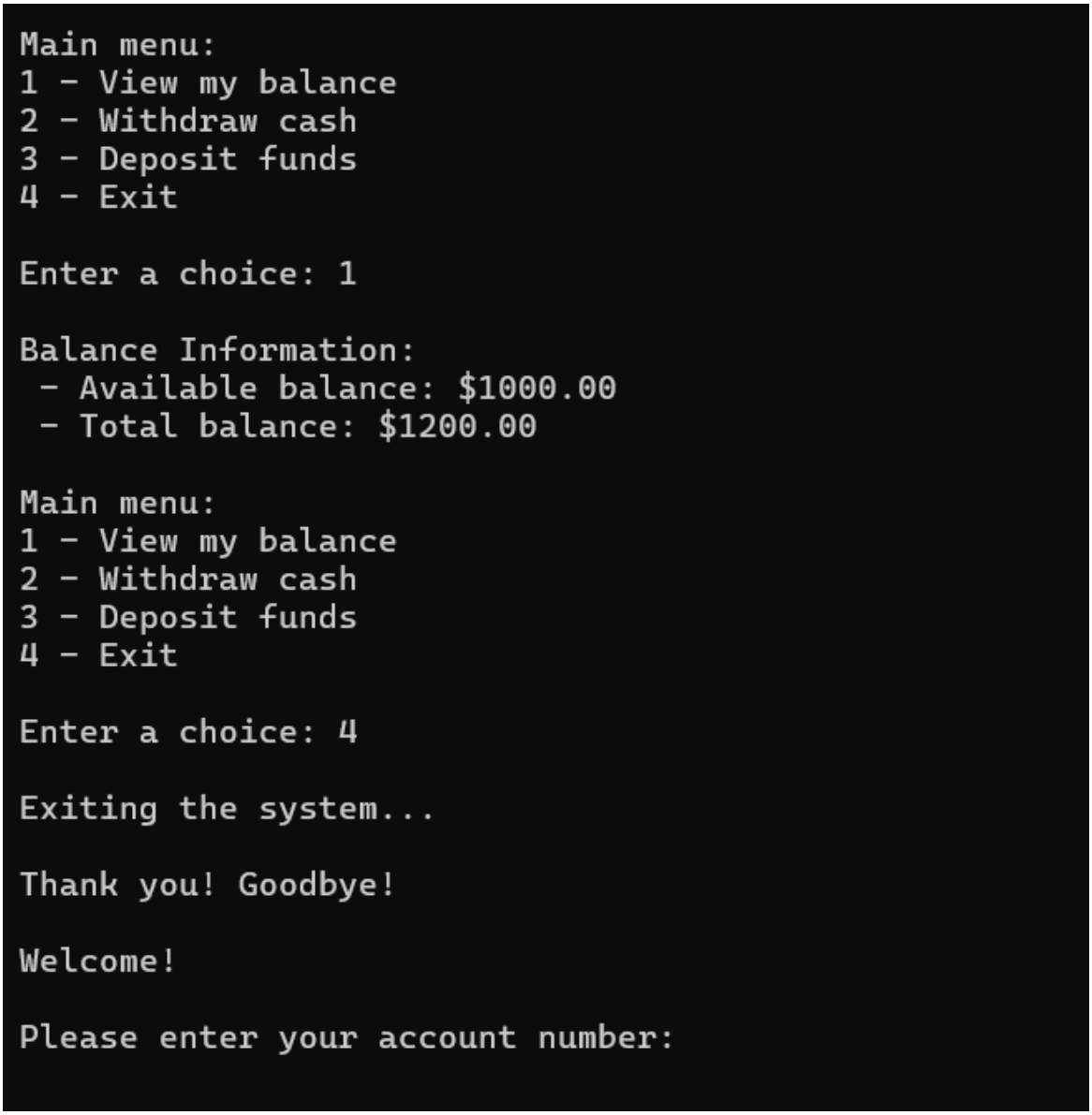

Figura 07 – Exemplo de Saída do Sistema.

A última imagem ilustra claramente o processo de encerramento da sessão por parte de um usuário, após a consulta de saldo. Embora o exemplo seja simples, ele exemplifica de maneira eficaz o ciclo contínuo e a dinâmica do funcionamento do sistema de caixa eletrônico. Cada interação do usuário, como consultar saldo ou realizar um saque, é seguida de uma volta ao estado inicial do sistema, permitindo que um novo

usuário tenha acesso ao sistema e realize suas operações de maneira independente. Esse comportamento de "reset", ou "loop", é crucial para garantir que o sistema esteja sempre pronto para a próxima utilização, proporcionando uma experiência contínua, eficiente e sem interrupções para diversos usuários.

As imagens apresentadas servem como uma confirmação visual da eficácia do sistema implantado, validando as funcionalidades propostas e a interação entre o usuário e o sistema. Mesmo que os exemplos mostrados sejam simples, eles cumprem a função de demonstrar que o sistema está implementado com sucesso, realizando todas as operações previstas e funcionando conforme esperado. Assim, as imagens não apenas ilustram o processo, mas também confirmam que o sistema de caixa eletrônico foi implantado com êxito, proporcionando uma experiência fluida e eficiente aos usuários.

Além disso, o sistema implementado demonstra de maneira clara a importância da modularização e da programação orientada a objetos (POO) no desenvolvimento de software. Cada funcionalidade do caixa eletrônico foi estruturada de forma independente, em diferentes módulos ou classes, facilitando tanto a manutenção quanto a evolução do sistema. A utilização de POO permitiu a criação de um código mais organizado, reutilizável e escalável, possibilitando a integração de novas funcionalidades no futuro sem comprometer a estrutura já existente. Por exemplo, a adição de novas opções, como transferência entre contas ou consulta de extrato, pode ser feita de forma modular, garantindo que as mudanças não afetem outras partes do sistema. Além disso, a clareza e a coesão do código foram mantidas ao longo de todo o processo de desenvolvimento, tornando o sistema mais fácil de entender e modificar. Isso não apenas contribui para a eficiência do sistema atual, mas também cria uma base sólida para melhorias futuras, exemplificando os benefícios da programação orientada a objetos na criação de sistemas complexos e flexíveis.

5 CONCLUSÃO

Nas considerações finais, destaca-se o cumprimento dos objetivos propostos neste trabalho, que envolvem o desenvolvimento do sistema ATM, a elaboração do documento acadêmico e a análise crítica do processo. O sistema foi construído com base no livro C++ Como Programar, utilizando Programação Orientada a Objetos (POO), o que permitiu a criação de um software modular e eficiente. A elaboração do trabalho escrito seguiu as normas acadêmicas, refletindo a evolução na redação necessária para a conclusão do curso, com ênfase na clareza e coesão. Além disso, a análise crítica do desenvolvimento foi realizada no Capítulo 3, detalhando todo o processo e as decisões tomadas.

Adicionalmente, o trabalho contribuiu para o aprimoramento das habilidades de organização e planejamento no desenvolvimento de sistemas, uma vez que o processo de modularização e a aplicação de boas práticas de engenharia de software, como a reutilização de código e a criação de soluções escaláveis, exigiram um planejamento detalhado e uma execução cuidadosa de cada etapa do projeto.

Por fim, o desenvolvimento do sistema proporcionou uma vivência prática significativa das metodologias e conceitos aprendidos ao longo da disciplina, possibilitando uma compreensão mais profunda de como estruturar e manter sistemas de software eficientes e de alta qualidade. O trabalho foi uma valiosa oportunidade de consolidar os conhecimentos técnicos e acadêmicos, preparando para a atuação profissional e desafios futuros no campo da programação e engenharia de software.

REFERÊNCIAS

DEITEL, Paul; DEITEL, Harvey. **C++: How to Program**. 5. ed. Upper Saddle River: Pearson Prentice Hall, 2006.

DRAW.IO. **Draw.io Documentation**. Disponível em: https://www.drawio.com/doc/. Acesso em: 28 nov. 2024.

MAITINO NETO, Roque. **Programação Orientada a Objetos**. 1. ed. São Paulo: Editora, 2018.

MICROSOFT. **Visual Studio Code: Setup Overview**. Disponível em: https://code.visualstudio.com/docs/setup/setup-overview. Acesso em: 17 nov. 2024

SOMMERVILLE, Ian. **Engenharia de software**. Tradução Ivan Bosnic e Kalinka G. de O. Gonçalves; revisão técnica Kechi Hirama. 9. ed. São Paulo: Pearson Prentice Hall, 2011. ISBN 978-85-7936-108-1.

GLOSSÁRIO

Loop: Um padrão de design (Design Pattern) descreve uma solução geral reutilizável para um problema recorrente no desenvolvimento de sistemas de software orientados a objetos. Não é um código final, mas sim uma descrição ou modelo de como resolver o problema em questão, sendo aplicável em diversas situações diferentes.

ATM: “Sigla para "Automated Teller Machine" (Caixa Eletrônico). Refere-se a um sistema bancário automatizado que permite aos usuários realizar operações como consultas, saques e depósitos, sem a necessidade de interação direta com um caixa humano.

APÊNDICE A: DETALHAMENTO DAS CLASSES

A.01 - Classe ATM

Classe ATM	
Atributos:	- userAuthenticated: bool: Armazena se o usuário foi autenticado. - currentAccountNumber: int: Armazena o número da conta do usuário autenticado. - screen: Screen: Instância da classe Screen, representando a tela do caixa eletrônico. - keypad: Keypad: Instância da classe Keypad, representando o teclado do caixa eletrônico. - cashDispenser: CashDispenser: Instância da classe CashDispenser, representando o dispensador de cédulas. - depositSlot: DepositSlot: Instância da classe DepositSlot, representando o local de depósito. - bankDatabase: BankDatabase: Instância da classe BankDatabase, responsável por armazenar as informações das contas bancárias.
Métodos:	+ ATM(): Construtor da classe, responsável por inicializar os membros de dados. + run(): void: Método que inicia a operação do ATM. + authenticateUser(): void: Tenta autenticar o usuário. + performTransactions(): void: Realiza transações após a autenticação do usuário. + displayMainMenu(): int: Exibe o menu principal para o usuário. + createTransaction(int): Transaction*: Cria e retorna um objeto da classe Transaction, dependendo do tipo especificado.

Quadro A.01 - Classe ATM

A.02 - Classe Screen

Classe Screen	
Atributos:	Não possui.
Métodos:	+ displayMessage(string): void: Exibe uma mensagem na tela. + displayMassageLine(string): void: Exibe uma mensagem com nova linha no final. + displayDollarAmount(double): void: Exibe um valor monetário em dólares na tela.

Quadro A.02 - Classe Screen

A.03 - Classe Keypad

Classe Keypad	
Atributos:	Não possui.
Métodos:	+ getInput(): int: Método que retorna um valor inteiro inserido pelo usuário no teclado do caixa eletrônico.

Quadro A.03 - Classe Kaypad

A.04 - Classe CashDispenser

Classe CashDispenser	
Atributos:	- const static INITIAL_COUNT: int = 500: Um valor constante e estático, que define a quantidade inicial de cédulas no dispensador como 500. - count: int: Armazena o número de cédulas de $20 que estão disponíveis no dispensador.
Métodos:	+ CashDispenser(): Construtor que inicializa o contador de cédulas com o valor de 500. + dispenseCash(int): void: Método que simula a retirada da quantia especificada de cédulas. + isSufficientCashAvailable(int): bool: Método que verifica se o dispensador possui cédulas suficientes para entregar a quantia desejada.

Quadro A.04 - Classe CashDispenser

A.05 - Classe DepositSlot

Classe DepositSlot	
Atributos:	Não possui
Métodos:	+ isEnvelopeReceived(): bool: Método que retorna um valor boole-ano indicando se o envelope foi recebido pelo slot de depósito.

Quadro A.05 - Classe DepositSlot

A.06 - Classe Account

Classe Account	
Atributos:	- accountNumber: int: Armazena o número da conta bancária. - pin: int: Armazena o PIN de autenticação da conta. - availableBalance: double: Armazena o saldo disponível para retirada. - totalBalance: double: Armazena o saldo total (fundos disponíveis + fundos esperando compensação).
Métodos:	+ Account(int, int, double, double): Construtor que configura os atributos accountNumber, pin, availableBalance e totalBalance. + validaPIN(int): bool: Verifica se o PIN fornecido pelo usuário é válido. + getAvailableBalance(): double: Retorna o saldo disponível para retirada. + getTotalBalance(): double: Retorna o saldo total, incluindo fundos esperando compensação. + credit(double): void: Adiciona um valor ao saldo total da conta. + debit(double): void: Subtrai um valor do saldo disponível da conta. + getAccountNumber(): int: Retorna o número da conta.

Quadro A.06 - Classe Account

A.07 - Classe BankDatabase

Classe BankDatabase	
Atributos:	- accounts: vector<Account>: Vetor que armazena os objetos Account, representando todas as contas bancárias do banco.
Métodos:	+ BankDatabase(): Construtor que inicializa as contas no banco. + authenticateUser(int, int): bool: Método que autentica um usuário verificando se o número da conta e o PIN correspondem a uma conta existente no banco. + getAvailableBalance(int): double: Obtém o saldo disponível de uma conta. + getTotalBalance(int): double: Obtém o saldo total de uma conta (incluindo fundos esperando compensação). + credit(int, double): void: Adiciona um valor ao saldo da conta especificada. + debit(int, double): void: Subtrai um valor do saldo da conta especificada. - getAccount(int): Account*: Método privado que retorna um ponteiro para o objeto Account correspondente ao número da conta fornecido.

Quadro A.07 - Classe BankDatabase

A.08 - Classe Transaction

Classe Transaction	
Atributos:	- accountNumber: int: O número da conta envolvida na transação. - screen: Screen&: Uma referência à instância da classe Screen, que representa a tela do caixa eletrônico. - bankDatabase: BankDatabase&: Uma referência à instância da classe BankDatabase, que representa o banco de dados de contas.
Métodos:	+ Transaction(int, Screen&, BankDatabase&): Construtor que inicializa a transação com o número da conta, a referência à tela (Screen) e a referência ao banco de dados (BankDatabase). + ~Transaction(): void: Destruidor virtual com corpo vazio, usado para garantir que as classes derivadas sejam destruídas corretamente. + getAccountNumber(): int: Retorna o número da conta envolvida na transação. + getScreen(): Screen&: Retorna uma referência à tela (Screen). + getBankDatabase(): BankDatabase&: Retorna uma referência ao banco de dados de informações das contas. + execute(): void (pure virtual): Este é um método virtual puro, que precisa ser implementado em classes derivadas. Ele define o comportamento específico da transação, como saque ou depósito.

Quadro A.08 - Classe Transaction

A.09 - Classe BalanceInquiry

Classe BalanceIquiry	
Atributos:	A classe BalanceInquiry não possui atributos privados adicionais. Ela herda os atributos da classe base Transaction, que são: accountNumber, screen, e bankDatabase.
Métodos:	+ BalanceInquiry(int, Screen&, BankDatabase&): Construtor que inicializa a classe BalanceInquiry com o número da conta, uma referência à instância da classe Screen e uma referência ao banco de dados BankDatabase. Esse construtor é chamado através da classe base Transaction. + execute(): void: Método que implementa a transação de consulta de saldo. O comportamento específico da consulta de saldo será definido dentro deste método.

Quadro A.09 - Classe BalanceInquiry

A.010 - Classe Withdrawal

Classe Withdrawal	
Atributos:	- amount: int: A quantia a ser sacada pelo usuário. - keypad: Keypad&: Referência ao objeto Keypad (teclado do caixa eletrônico) usado para a entrada do valor. - cashDispenser: CashDispenser&: Referência ao objeto CashDispenser (dispensador de cédulas) usado para entregar as cédulas ao usuário.
Métodos:	+ Withdrawal(int, Screen&, BankDatabase&, Keypad&, CashDispenser&): Construtor que inicializa a transação de saque com o número da conta, referências para a tela (Screen), o banco de dados (BankDatabase), o teclado (Keypad) e o dispensador de cédulas (CashDispenser). + execute(): void: Método que executa a transação de saque, usando os objetos envolvidos (como o teclado e o dispensador de cédulas). + displayMenuOfAmounts(): int: Método que exibe um menu de opções de quantias para saque, permitindo ao usuário selecionar o valor desejado.

Quadro A.010 - Classe Withdrawal

A.011 - Classe Deposit

Classe Deposit	
Atributos:	- amount: double: A quantia a ser depositada pelo usuário. - keypad: Keypad&: Referência ao objeto Keypad (teclado do caixa eletrônico) usado para a entrada do valor do depósito. - depositSlot: DepositSlot&: Referência ao objeto DepositSlot (abertura de depósito do caixa eletrônico) onde o usuário insere o envelope de depósito.
Métodos:	+ Deposit(int, Screen&, BankDatabase&, Keypad&, DepositSlot&): Construtor que inicializa a transação de depósito com o número da conta, referências para a tela (Screen), o banco de dados (BankDatabase), o teclado (Keypad) e a abertura de depósito (DepositSlot). + execute(): void: Método que executa a transação de depósito, realizando as operações necessárias no banco de dados e interagindo com o teclado e o depósito. + promptForDepositAmount(): double: Método que solicita ao usuário a quantia a ser depositada. Ele utiliza o teclado para capturar o valor digitado.

Quadro A.011 - Classe Deposit