TAMBABANK: Desenvolvimento de um Sistema Bancário em Java

Leonardo de Albuquerque Saraiva [1]
Messias Rafael Batista [2]

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema bancário denominado Tambabank, utilizando a linguagem de programação Java e as metodologias de modelagem orientada a objetos. O objetivo do sistema é fornecer funcionalidades básicas de gerenciamento de contas correntes, contas poupança e contas de investimento. Inicialmente, foram levantados os requisitos do sistema por meio da análise de casos de uso, identificando as principais funcionalidades e interações entre os atores e o sistema. Em seguida, foram utilizados diagramas de caso de uso e diagramas de classe para representar as características e estrutura do sistema. A implementação foi realizada de forma incremental, adicionando funcionalidades gradualmente de acordo com as necessidades do projeto. O Eclipse IDE foi utilizado como ambiente de desenvolvimento integrado, proporcionando suporte para a escrita, compilação e execução do código Java. Durante o processo de desenvolvimento, foram aplicadas as melhores práticas de programação orientada a objetos, visando a modularidade, reutilização de código e clareza da estrutura do sistema. Além disso, foram realizadas revisões periódicas do código para garantir a qualidade e corrigir possíveis problemas. O resultado foi um sistema Tambabank funcional, capaz de realizar operações como ver saldo, depositar dinheiro, sacar dinheiro, transferir valores entre contas e calcular rendimentos para contas poupança e contas de investimento. O trabalho contribui para o entendimento das etapas de desenvolvimento de um sistema bancário utilizando a linguagem Java e a modelagem orientada a objetos.

Palavras-chaves: Java; Banco; projeto prático.

ABSTRACT

This paper presents the development of a banking system called Tambabank, using the Java programming language and object-oriented modeling methodologies. The system aims to provide basic functionalities for managing current accounts, savings accounts, and investment accounts. Initially, system requirements were identified through the analysis of use cases, identifying the main features and interactions between actors and the system. Subsequently, use case diagrams and class diagrams were used to represent the system's characteristics and structure. The implementation was done incrementally, gradually adding functionalities according to the project's needs. The Eclipse IDE was used as the integrated development environment, providing support for Java code writing, compilation, and execution. Throughout the development process, best practices of object-oriented programming were applied, focusing on modularity, code reuse, and clarity of the system's structure. Regular code reviews were conducted to ensure quality and address possible issues. The result was a functional Tambabank system capable of performing operations such as checking balances, depositing funds, withdrawing funds, transferring values between accounts, and calculating returns for savings

[1] Graduando do Curso de Sistemas para Internet. E-mail: leosaraiv4@gmail.com
[2] Professor Orientador. E-mail: messias.batista@iesp.edu.br

and investment accounts. This work contributes to understanding the development stages of a banking system using the Java language and object-oriented modeling.

Keywords: Java; Banking; practical project.

1 INTRODUÇÃO

No mundo atual, a praticidade e conveniência dos serviços financeiros online têm ganhado cada vez mais destaque. Com a evolução da tecnologia, os bancos virtuais se tornaram uma alternativa popular para realizar transações bancárias de forma ágil e segura. Neste contexto, apresentamos o projeto Tambabank, um sistema bancário desenvolvido em Java, que visa simular as funcionalidades de um banco, oferecendo aos usuários a possibilidade de gerenciar suas contas correntes, contas poupanças e contas de investimento de forma simples e intuitiva.

1.1 Objetivos e Justificativa

O objetivo deste trabalho é desenvolver um sistema bancário virtual que demonstre a aplicação dos conceitos de encapsulamento, composição, herança, polimorfismo, classes e métodos abstratos, bem como a utilização de métodos construtores. Além disso, busca-se criar uma experiência interativa para o usuário, onde ele possa navegar por diferentes menus e realizar operações com suas contas.

A escolha de um projeto voltado para a área financeira se dá pela importância desse setor na sociedade, bem como a relevância do desenvolvimento de soluções tecnológicas que facilitem a gestão financeira pessoal. O Tambabank oferece uma plataforma amigável e de fácil utilização, que permite aos usuários realizar operações bancárias de forma segura e eficiente.

1.2 Elicitação dos Requisitos

A etapa inicial do projeto consistiu na elicitação dos requisitos, ou seja, na identificação das funcionalidades e características que o sistema deveria possuir. Para isso, foram consideradas as principais operações realizadas em um banco, como consulta de saldo, depósito, saque, transferência entre contas e encerramento de contas.

A partir dessa análise, foram definidas três classes principais: ContaCorrente, ContaPoupanca e ContaInvestimento, que representam os diferentes tipos de contas disponíveis no Tambabank. Cada uma dessas classes possui métodos específicos para realizar as operações relacionadas às contas, como verificar o saldo, realizar depósitos, saques e aplicar rendimentos.

A estrutura de menus e a lógica de navegação também foram definidas nessa etapa. Optou-se por apresentar um menu principal ao usuário, onde ele pode selecionar a opção desejada. Com base na opção escolhida, o programa direciona o usuário para o menu correspondente, oferecendo as funcionalidades relacionadas àquela opção específica. Dessa forma, o Tambabank proporciona uma experiência intuitiva e facilita a interação do usuário com o sistema.

2 FUNDAMENTAÇÃO TEÓRICA

Neste tópico, serão apresentadas as ferramentas, tecnologias e recursos utilizados no desenvolvimento do projeto Tambabank, bem como a linguagem de programação, os diagramas e o software utilizado para modelagem e desenvolvimento.

2.1 Linguagem de Programação e Recursos da Linguagem

O projeto Tambabank foi desenvolvido utilizando a linguagem de programação Java. Java é uma linguagem de programação de alto nível, orientada a objetos e amplamente utilizada no desenvolvimento de aplicativos, sistemas e projetos de larga escala. Sua sintaxe simples e legível, juntamente com recursos avançados, como encapsulamento, herança e polimorfismo, tornam Java uma escolha popular para o desenvolvimento de software (SCHILDT, 2007).

No projeto Tambabank, foram explorados recursos importantes da linguagem Java, como:

Encapsulamento: O encapsulamento permite a proteção dos dados e funcionalidades internas de uma classe, permitindo o acesso apenas através de métodos definidos. Isso garante a integridade e segurança dos dados do usuário (SIERRA; BATES, 2020);

Composição: A composição é uma relação entre objetos onde um objeto contém outros objetos como parte de sua estrutura. No Tambabank, as classes ContaCorrente, ContaPoupanca e ContaInvestimento utilizam a composição para relacionar os dados e comportamentos específicos de cada tipo de conta (SIERRA; BATES, 2020);

Herança: A herança é um mecanismo que permite que uma classe herde atributos e métodos de outra classe. No projeto Tambabank, as classes ContaCorrente, ContaPoupanca e ContaInvestimento herdam atributos e métodos da classe abstrata Conta (SIERRA; BATES, 2020);

Polimorfismo: O polimorfismo permite que objetos de classes diferentes sejam tratados de maneira uniforme através de uma interface comum. No Tambabank, o polimorfismo é utilizado ao tratar diferentes tipos de contas como objetos do tipo Conta, simplificando a implementação de operações comuns (SIERRA; BATES, 2020).

2.2 Diagramas

Durante a fase de análise e modelagem do projeto Tambabank, foram utilizados dois tipos de diagramas: o diagrama de classes e o diagrama de caso de uso. Esses diagramas desempenham papéis fundamentais no processo de modelagem e representação das características e interações do sistema.

O diagrama de caso de uso descreve as interações entre os atores e o sistema, identificando os principais casos de uso do sistema. Ele mostra os diferentes cenários de uso do sistema, as ações realizadas pelos atores e as respostas do sistema. Esse diagrama ajuda a entender os requisitos funcionais do sistema e serve como base para o desenvolvimento das funcionalidades do sistema (BOOCH, 2005).

Por outro lado, o diagrama de classes é uma representação visual das classes do sistema, seus atributos, métodos e relacionamentos. Ele fornece uma visão estrutural do sistema, mostrando como as classes estão relacionadas entre si e como os objetos interagem. Esse diagrama permite identificar as entidades principais do sistema, suas propriedades e comportamentos, e serve como base para a implementação do código (LARMAN, 2012). Nesse projeto, o diagrama de classes foi utilizado para visualizar a hierarquia das classes ContaCorrente, ContaPoupanca e ContaInvestimento.

2.3 Software Utilizado

No desenvolvimento do projeto Tambabank, foi utilizado o software Eclipse. O Eclipse é um ambiente de desenvolvimento integrado (IDE) amplamente utilizado para desenvolvimento em Java. Ele fornece uma série de recursos e ferramentas que facilitam a escrita, depuração e execução de código Java.

O Eclipse oferece suporte para criação de projetos, edição de código, autocompletar, depuração passo a passo, gerenciamento de bibliotecas e plugins, facilitando o desenvolvimento e aumentando a produtividade dos programadores.

Para a modelagem dos diagramas, foi utilizada uma ferramenta de modelagem UML, chamada *Lucida.App*. Essa ferramenta permite a criação de diagramas de classes, diagramas de caso de uso e outros diagramas UML de maneira intuitiva e visual, auxiliando na compreensão e comunicação do projeto.

3 METODOLOGIA

Neste tópico, será apresentada a metodologia utilizada no desenvolvimento do projeto Tambabank, descrevendo como cada ferramenta, tecnologia e conceito foram aplicados no decorrer do trabalho.

A linguagem de programação escolhida para o desenvolvimento do projeto foi o Java. O Java é amplamente utilizado na indústria de desenvolvimento de software devido à sua portabilidade, orientação a objetos e vasta biblioteca de classes. Através do Java, foi possível implementar as classes, métodos e atributos necessários para o sistema Tambabank.

O Eclipse IDE foi utilizado como ambiente de desenvolvimento integrado para a codificação do projeto Tambabank. Ele oferece uma interface amigável e recursos avançados que facilitam a escrita, compilação e execução de código Java. O Eclipse proporcionou uma experiência de desenvolvimento suave e eficiente, permitindo a criação e organização das classes do sistema.

O diagrama de caso de uso foi utilizado para modelar as interações entre os atores e o sistema Tambabank. Ele descreve as funcionalidades do sistema do ponto de vista do usuário, identificando as ações que podem ser realizadas e as relações entre os atores e os casos de uso. Através desse diagrama, foi possível definir as principais funcionalidades oferecidas pelo sistema.

O diagrama de classe foi utilizado para representar a estrutura do sistema Tambabank. Ele mostra as classes do sistema, seus atributos e métodos, bem como as relações de associação, herança e composição entre as classes. O diagrama de classe proporcionou uma visão geral da organização e das interações entre as entidades do sistema.

3.1 Metodologia de Desenvolvimento

O desenvolvimento do projeto Tambabank seguiu uma abordagem ágil, baseada em iterações curtas e incrementais. Inicialmente, foram levantados os requisitos do sistema por meio da análise de casos de uso. Com base nesses requisitos, foram identificadas as classes necessárias e definidos os atributos e métodos de cada classe.

A implementação do sistema foi feita em etapas, onde as funcionalidades foram adicionadas progressivamente, de acordo com as necessidades do projeto. Utilizando a linguagem Java e o Eclipse IDE, as classes foram desenvolvidas e os relacionamentos entre elas foram estabelecidos com base no diagrama de classe.

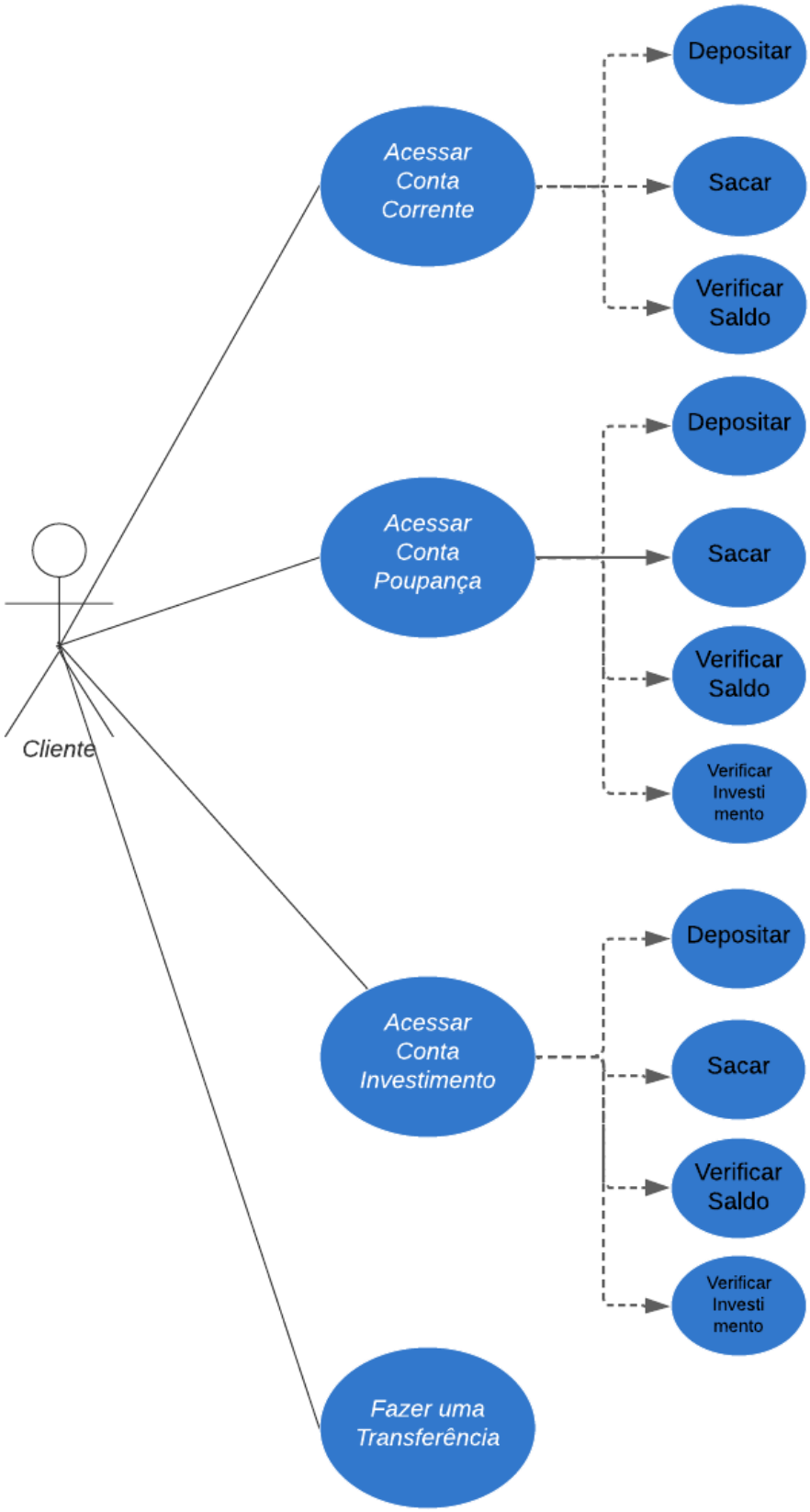
Durante o desenvolvimento, foram realizadas revisões periódicas do código para garantir a qualidade e a correção das implementações. A abordagem ágil permitiu uma maior flexibilidade no desenvolvimento do projeto, possibilitando ajustes e melhorias conforme necessário.

4 RESULTADO E DISCUSSÃO

Neste tópico, será abordada a implementação e a estrutura do projeto Tambabank e os seus diagramas de caso de uso e de classe. Serão detalhando as classes, os métodos e as funcionalidades desenvolvidas para simular um sistema bancário virtual.

4.1 Diagrama de Caso de Uso

Figura 1. Diagrama de Caso de Uso



O diagrama de caso de uso do projeto Tambabank (Figura 1) representa as interações entre o ator "Cliente" e as funcionalidades disponíveis no sistema. O ator "Cliente" desempenha

um papel fundamental ao realizar ações no sistema, como acessar diferentes tipos de contas e realizar operações bancárias, além das seguintes ações:

- Acessar Conta Corrente: O ator "Cliente" pode acessar sua conta corrente para realizar operações como depósito, saque e verificar saldo. Essas ações estão relacionadas ao uso da conta corrente e são representadas como casos de uso específicos;

- Acessar Conta Poupança: O ator "Cliente" também pode acessar sua conta poupança, que oferece funcionalidades adicionais em relação à conta corrente. Além das operações de depósito, saque e verificação de saldo, o cliente pode verificar o rendimento da conta poupança;

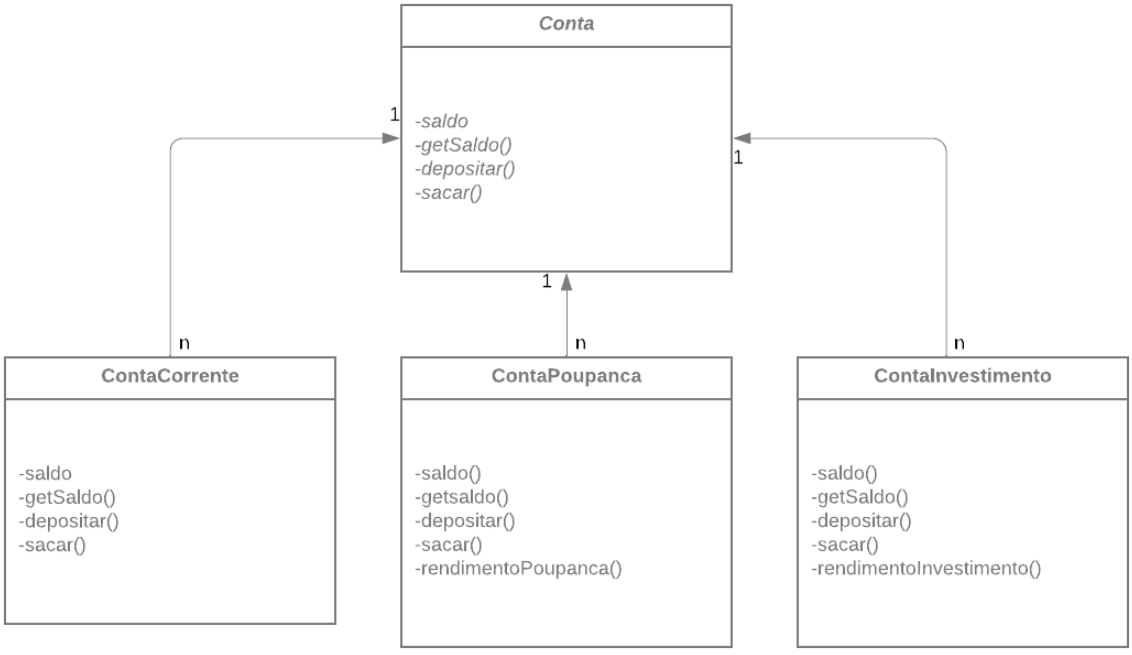
- Acessar Conta Investimento: Da mesma forma, o ator "Cliente" tem a opção de acessar sua conta de investimento. Além das operações básicas, como depósito, saque e verificação de saldo, o cliente pode verificar o status do investimento, que pode incluir informações sobre prazo, taxa de retorno e outras características relevantes.

- Transferir Valores: O ator "Cliente" pode realizar uma transferência de valores entre contas, seja entre a conta corrente e a conta poupança, entre a conta corrente e a conta de investimento, ou entre a conta poupança e a conta de investimento. Essa funcionalidade permite a movimentação de recursos entre as contas do cliente.

O diagrama de caso de uso ilustra a interação entre o ator "Cliente" e as funcionalidades do sistema, permitindo uma compreensão clara das principais ações que o cliente pode realizar. Essas ações estão diretamente relacionadas aos casos de uso específicos, que representam as operações disponíveis para cada tipo de conta.

4.2 Diagrama de Classe

Figura 2. Diagrama de Classes



No diagrama de classes do projeto Tambabank (Figura 2), é possível identificar a presença da classe abstrata "Conta", que serve como uma entidade genérica para representar contas bancárias. A classe abstrata "Conta" contém atributos e métodos comuns a todas as contas, como número da conta, saldo e operações básicas de depósito e saque.

Além da classe abstrata "Conta", são apresentadas outras classes que herdam seus métodos e atributos. Por exemplo, as classes "ContaCorrente", "ContaPoupanca" e

"ContaInvestimento" são subclasses da classe abstrata "Conta". Essas classes específicas representam diferentes tipos de contas disponíveis no sistema Tambabank.

Essa estrutura de herança permite a reutilização de código e a definição de comportamentos específicos para cada tipo de conta, ao mesmo tempo em que mantém a consistência com os métodos e atributos compartilhados pela classe abstrata "Conta". Dessa forma, o diagrama de classes do projeto Tambabank ilustra a hierarquia de classes e a relação de herança entre a classe abstrata "Conta" e suas subclasses, representando os diferentes tipos de contas bancárias disponíveis no sistema.

4.3 Estrutura e implementação do Projeto

O projeto Tambabank é organizado em diferentes classes que representam as entidades e funcionalidades do sistema bancário. A estrutura do projeto é composta pelos seguintes arquivos:

- TambaBank.java: Classe principal que contém o método *main* responsável por iniciar o programa e gerenciar a interação com o usuário;

Figura 3. Trecho da Classe Tambabank.java

```
 3 import java.util.Scanner;
 4
 5 public class Tambabank {
 6     private ContaCorrente contaCorrente;
 7     private ContaPoupanca contaPoupanca;
 8     private ContaInvestimento contaInvestimento;
 9
10     public Tambabank() {
11         contaCorrente = new ContaCorrente();
12         contaPoupanca = new ContaPoupanca();
13         contaInvestimento = new ContaInvestimento();
14     }
15
16     public void iniciar() {
17         System.out.println("Obrigado por utilizar o Tambabank. Selecione uma opção do
18         exibirMenuPrincipal();
19
20         int opcao = lerOpcao();
21
22         while (opcao != 6) {
23             switch (opcao) {
24                 case 1:
25                     menuContaCorrente();
26                     break;
27                 case 2:
28                     menuContaPoupanca();
29                     break;
30                 case 3:
31                     menuContaInvestimento();
32                     break;
33                 case 4:
```

- Conta.java: Classe abstrata que representa uma conta bancária genérica. Define os atributos comuns a todas as contas, como número da conta e saldo, e os métodos básicos para consulta de saldo e depósito;

Figura 4. Classe Conta.java

```java
package tambabank;

public abstract class Conta {
    protected double saldo;

    public Conta() {
        this.saldo = 0.0;
    }

    public double getSaldo() {
        return saldo;
    }

    public abstract void depositar(double valor);

    public abstract void sacar(double valor);
}

```

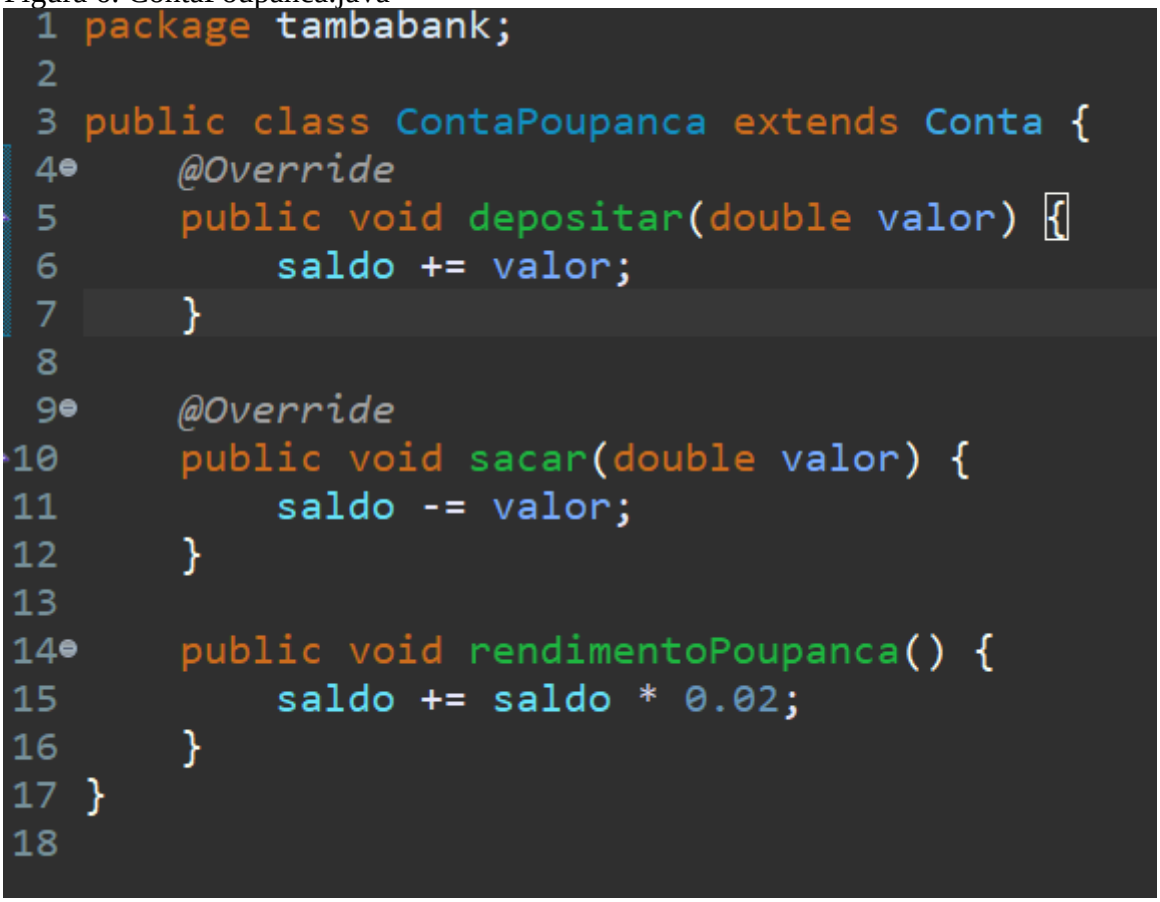
- ContaCorrente.java: Classe que herda da classe Conta e representa uma conta corrente. Implementa métodos específicos para saque e rendimento da conta corrente;

Figura 5. Classe ContaCorrente.java

```java
package tambabank;

public class ContaCorrente extends Conta {
    @Override
    public void depositar(double valor) {
        saldo += valor;
    }

    @Override
    public void sacar(double valor) {
        saldo -= valor;
    }
}

```

- ContaPoupanca.java: Classe que herda da classe Conta e representa uma conta poupança. Implementa métodos específicos para saque, rendimento e aplicação de juros da conta poupança;

Figura 6. ContaPoupanca.java



```java
package tambabank;

public class ContaPoupanca extends Conta {
    @Override
    public void depositar(double valor) {
        saldo += valor;
    }

    @Override
    public void sacar(double valor) {
        saldo -= valor;
    }

    public void rendimentoPoupanca() {
        saldo += saldo * 0.02;
    }
}

```

- ContaInvestimento.java: Classe que herda da classe Conta e representa uma conta de investimento. Implementa métodos específicos para saque, rendimento e aplicação de rendimentos da conta de investimento.

Figura 7. ContaInvestimento.java

```java
package tambabank;

public class ContaInvestimento extends Conta {
    @Override
    public void depositar(double valor) {
        saldo += valor;
    }

    @Override
    public void sacar(double valor) {
        saldo -= valor;
    }

    public void rendimentoInvestimento() {
        saldo += saldo * 0.10;
    }
}
```

4.2 Funcionalidades Implementadas

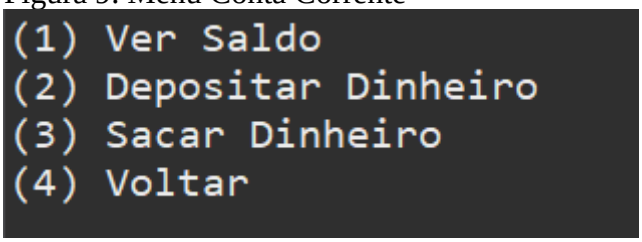
O projeto Tambabank oferece diversas funcionalidades para gerenciamento das contas bancárias. Ao iniciar o programa, o usuário é apresentado a um menu principal onde pode escolher as seguintes opções:

Figura 8. Menu principal

```
Obrigado por utilizar o Tambabank. Selecione uma opção do menu:

(1) Conta Corrente
(2) Conta Poupança
(3) Conta Investimento
(4) Transferência
(5) Encerrar Conta
(6) Sair

Digite a opção desejada: 1
```

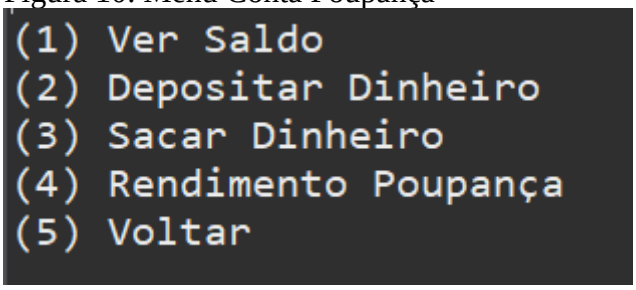
- Conta Corrente: Permite visualizar o saldo, realizar depósito, sacar dinheiro e voltar ao menu principal;

Figura 9. Menu Conta Corrente



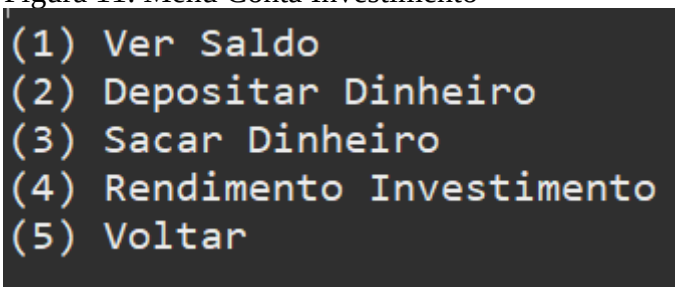
- Conta Poupança: Permite visualizar o saldo, realizar depósito, sacar dinheiro, aplicar rendimento e voltar ao menu principal;

Figura 10. Menu Conta Poupança



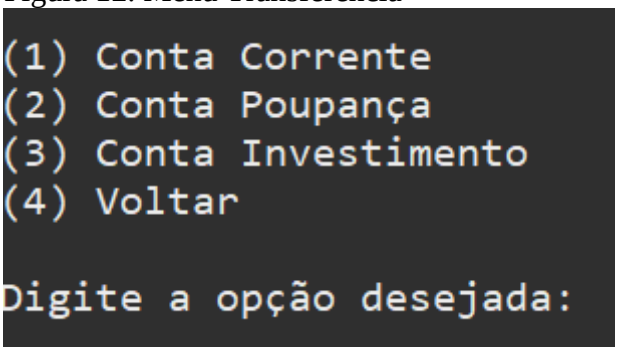
- Conta Investimento: Permite visualizar o saldo, realizar depósito, sacar dinheiro, aplicar rendimento e voltar ao menu principal;

Figura 11. Menu Conta Investimento



- Transferência: Permite transferir dinheiro entre diferentes contas, selecionando a conta de origem e a conta de destino;

Figura 12. Menu Transferência



- Encerrar Conta: Encerra todas as contas, somando o valor total e exibindo a mensagem de encerramento;
- Sair: Finaliza o programa e exibe uma mensagem de encerramento.

Cada uma dessas funcionalidades foi implementada considerando a estrutura de classes e métodos definidos no projeto. A interação com o usuário é feita através de entradas pelo teclado e exibição de mensagens na tela, permitindo uma experiência interativa e intuitiva.

5 CONSIDERAÇÕES FINAIS

O projeto Tambabank foi desenvolvido com o objetivo de simular um sistema bancário virtual, proporcionando ao usuário uma experiência interativa de gerenciamento de contas bancárias. Ao longo do documento, foram apresentados diversos aspectos relacionados ao projeto, desde a elicitação dos requisitos até a implementação e estruturação do sistema.

Durante a fase de elicitação dos requisitos, foram selecionadas as funcionalidades principais a serem implementadas, levando em consideração as necessidades dos usuários e os objetivos do projeto. A partir dessas funcionalidades, foram criadas as classes, métodos e atributos necessários para a construção do sistema.

A linguagem de programação Java foi utilizada como base para o desenvolvimento do projeto, aproveitando seus recursos e conceitos de orientação a objetos. O uso de recursos como encapsulamento, herança, polimorfismo e composição permitiu uma estrutura sólida e flexível do sistema, possibilitando a extensibilidade e a manutenção do código.

Para a modelagem do sistema, foram utilizados diagramas, como o diagrama de classes e o diagrama de caso de uso. Esses diagramas auxiliaram na visualização da estrutura das classes e na compreensão das interações entre os objetos.

O software Eclipse foi escolhido como ambiente de desenvolvimento, fornecendo uma plataforma robusta e amigável para a escrita, compilação e execução do código Java. Através dele, foi possível agilizar o processo de desenvolvimento, aproveitar recursos como autocompletar e depuração, e garantir a organização e eficiência do projeto.

Durante o desenvolvimento do projeto Tambabank, alguns desafios foram encontrados, como a implementação correta das funcionalidades, a garantia da segurança dos dados e a validação das entradas do usuário. No entanto, esses desafios foram superados através do estudo e aplicação dos conceitos aprendidos, bem como da pesquisa e consulta de recursos adicionais.

Em termos de resultados alcançados, o Tambabank se mostrou eficiente na simulação das operações bancárias, fornecendo ao usuário um ambiente controlado e seguro para realizar transações financeiras virtuais. As funcionalidades implementadas, como consulta de saldo, depósito, saque e transferência, possibilitam ao usuário uma experiência completa de gerenciamento de contas.

Como melhorias futuras, é possível considerar a implementação de novas funcionalidades, como pagamento de contas, agendamento de transferências, histórico de transações e criação de contas adicionais. Além disso, aperfeiçoamentos na interface do usuário e na validação de entradas podem ser explorados para proporcionar uma experiência mais intuitiva e segura.

Em suma, o projeto Tambabank foi uma oportunidade de aplicar os conhecimentos teóricos em um cenário prático, desenvolvendo um sistema bancário virtual em Java. Através da utilização adequada das ferramentas, da compreensão dos conceitos da linguagem e da elaboração de diagramas, foi possível criar um sistema funcional e eficiente. O processo de desenvolvimento permitiu adquirir experiência, enfrentar desafios e aprimorar habilidades de programação, modelagem e solução de problemas.

REFERÊNCIAS

BOOCH, Grady. **The unified modeling language user guide**. Pearson Education India, 2005.
SCHILDT, Herbert. **Java: the complete reference**. 2007.
LARMAN, Craig. **Applying UML and patterns: an introduction to object oriented analysis and design and interative developmen**t. Pearson Education India, 2012.
ORACLE. The Java Tutorials – Disponível em: https://docs.oracle.com/javase/tutorial/
SIERRA, Kathy; BATES, Bert. **Use a cabeça!: Java**. Alta Books, 2020.