RELATO DE EXPERIÊNCIA SOBRE O DESENVOLVIMENTO DE UM SISTEMA PARA SIMULAÇÃO DE UMA AGÊNCIA BANCÁRIA NO PROJETO INTEGRADOR

Gabriel de Oliveira Pontes 1

Matheus L S Maul de Andrade 2

Pabllo Cristian Ferreira de Lima 3

RESUMO

Esta é uma aplicação feita em Java que funciona como um banco, é um sistema criado para gerenciar transações financeiras, contas de cliente e demais operações relacionadas a uma instituição bancária. Através deste aplicativo, os usuários podem criar uma conta bancária ao fornecerem os dados necessários, e realizar transações com segurança. O sistema utiliza recursos avançados de segurança para proteger as informações pessoais e financeiras de seus clientes, como por exemplo, a criptografia de dados e autenticação do usuário. Além destas funções básicas, o aplicativo também pode fornecer outras funções, como geração de extratos, o histórico de transações e a possibilidade de definir lembretes para pagamentos de contas. Em suma, se trata de uma aplicação em Java que funciona como um banco e que é uma solução poderosa e segura para a gestão de transações financeiras e contas de clientes de uma instituição bancária, fornecendo as funcionalidades devidas para o banco.

Palavras-chave: Banco; Java; Transação.

ABSTRACT

This is an application made in Java that works like a bank, it is a system created to manage financial transactions, customer accounts and other operations related to a banking institution. Through this application, users can create a bank account by providing the necessary data, and carry out transactions safely. The system uses advanced security features to protect the personal and financial information of its customers, such as data encryption and user authentication. In addition to these basic functions, the application can also provide other functions such as generation of statements, transaction history and the possibility to set reminders for bill payments. In short, it is a Java application that works like a bank and is a powerful and secure solution for the management of financial transactions and customer accounts of a banking institution, providing the necessary functionalities for the bank.

Keywords: Bank; Java; Transaction.

1. INTRODUÇÃO

FUNDAMENTAÇÃO TEÓRICA O projeto (PROJETO INTEGRADOR-FUNDAMENTOS DA PROGRAMAÇÃO) se trata de uma agência bancária desenvolvida em linguagem Java. Utilizando conceitos de programação orientada a objetos e padrões de projeto para a construção da aplicação, todo código foi hospedado no repositório GITHUB do participante "GabrielOliveira04".
É uma aplicação que tem como objetivo (finalidade) principal simular o funcionamento de um banco e servir de modelo para outras agências bancárias, mapear as funcionalidades do APP, desenvolver as classes em linguagem Java como apresentado em sala. Nesta, permite que o usuário realize operações bancárias como depósito, saques, transferências e consultas de saldo.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Diagrama de caso de uso: são ferramentas de modelagem que descrevem as interações entre participantes e sistemas. Definir casos de uso e recursos propostos. Com este diagrama, visualizam-se como os atores interagem com o sistema, facilitando a identificação dos pré-requisitos escopo e comunicação entre as partes interessadas do projeto.

2.2 Diagrama de classes: é uma representação visual da estrutura e relacionamentos de classes em um sistema de software. Exibe classes, atributos, métodos e associações. Ajuda a entender a arquitetura do sistema e facilitar o design e a comunicação entre as equipes de desenvolvimento.

2.3 Java: é uma linguagem de programação muito usada. Que é conhecida por sua portabilidade e ênfase na segurança. É baseada em objetos e possui uma grande biblioteca padrão. É amplamente utilizado no desenvolvimento de aplicativos Android e sua comunidade ativa oferece suporte aos desenvolvedores com recursos e tutoriais. Em resumo, Java é uma linguagem versátil, segura e amplamente aceita em muitas áreas de desenvolvimento de software.

3. METODOLOGIA

No desenvolvimento do projeto, seguimos metodologias de desenvolvimento de software, como o Software Development Life Cycle (SDLC). Essa abordagem envolve as etapas de análise de requisitos, projeto, implementação, teste e manutenção do sistema. Durante a análise de requisitos, entendemos os requisitos funcionais e não funcionais do sistema, como criar e gerenciar contas bancárias, processar transações e calcular saldos. Utilizamos para isso ferramentas, como o IntelliJ IDEA. Além disso, contamos com a documentação disponibilizada pelo professor da disciplina, a própria linguagem Java, bem como pesquisas externas e discussões com colegas sobre o tema e o código.

4. RESULTADOS E DISCUSSÃO

Neste projeto, desenvolvemos um aplicativo bancário simples em Java utilizando conceitos básicos de orientação a objetos e a IDE IntelliJ. O aplicativo permite que o usuário crie contas bancárias, realize depósitos e saques, e verifique o histórico de transações. Ao analisarmos os resultados obtidos, podemos afirmar que o aplicativo atingiu seu principal objetivo.

A funcionalidade de criação de contas bancárias foi implementada com sucesso, permitindo que o usuário informe o número da conta, o titular e realize um depósito inicial. Isso proporciona ao usuário uma experiência intuitiva e fácil ao iniciar suas transações bancárias.

A capacidade de realizar transações de depósito e saque também foi implementada de forma efetiva. As transações são devidamente registradas no histórico de transações de cada conta bancária, permitindo que o usuário acompanhe detalhadamente as operações realizadas. O saldo atual da conta é atualizado corretamente após cada transação, garantindo a integridade dos dados.

O projeto utiliza adequadamente os conceitos de herança e encapsulamento, com a classe Conta servindo como base para as subclasses ContaCorrente, ContaInvestimento e ContaPoupanca. Essa abordagem possibilita a adição de comportamentos específicos para cada tipo de conta, facilitando a implementação de recursos adicionais no futuro, se necessário.

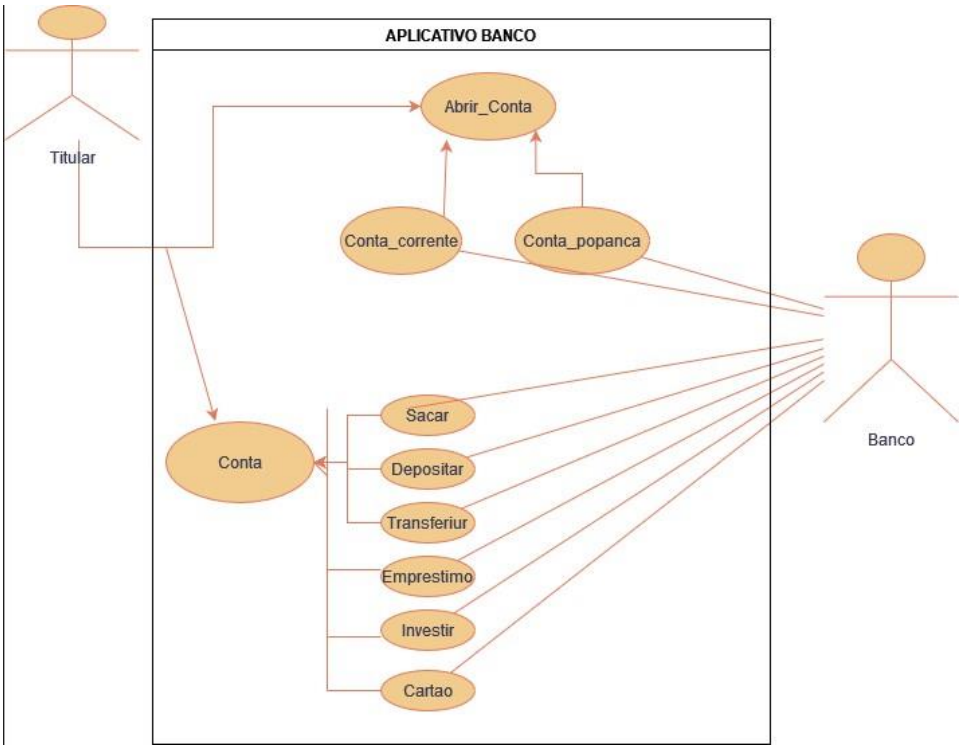
A classe AgenciaBancaria desempenha um papel fundamental na criação e gerenciamento das contas bancárias. Ela oferece métodos para interagir com as contas, como criar uma nova conta, recuperar informações de uma conta existente, realizar transações e imprimir o histórico de transações. Essa estrutura organizada e bem definida promove a modularidade do código e torna o aplicativo mais fácil de ser mantido e estendido.

A classe Pessoa é responsável por representar o titular da conta bancária, armazenando informações como nome, CPF e data de nascimento. Essa classe é utilizada como atributo nas classes Conta e suas subclasses, estabelecendo a relação entre uma pessoa e uma conta bancária.

No entanto, é importante observar que este projeto é um aplicativo bancário simples e pode ser aprimorado de várias maneiras. Por exemplo, poderíamos implementar funcionalidades adicionais, como transferências entre contas e cálculo de juros para contas de investimento. Também poderíamos melhorar a interface do usuário, tornando-a mais amigável e visualmente atraente, utilizando bibliotecas gráficas como JavaFX.

No geral, o projeto do aplicativo bancário em Java alcançou os objetivos pretendidos. Implementamos com sucesso as funcionalidades básicas de criação de contas, transações de depósito e saque, e registro do histórico de transações. As classes AgenciaBancaria, Conta, ContaCorrente, ContaInvestimento, ContaPoupanca e Pessoa foram utilizadas adequadamente, seguindo os princípios de orientação a objetos. A classe AgenciaBancaria desempenhou um papel importante na gestão das contas. Existem oportunidades de melhoria, como adicionar mais recursos e aprimorar a interface do usuário, que podem ser exploradas para enriquecer o aplicativo bancário em futuras iterações do projeto.

4.1 Diagrama de Caso de Uso



A imagem apresenta o diagrama de Caso de Uso, apresentando a interligação desde o usuário (Titular) até o próprio ”Banco”.

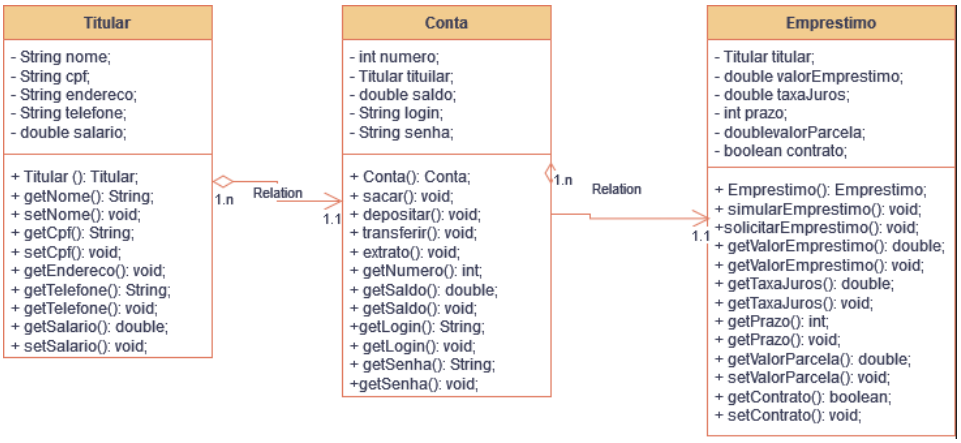
- Sistemas: Resumidamente é o retângulo de título “APLICATIVO BANCO”, ele fica entre os “bonecos”, quero dizer, entre os atores (titular e banco), o que está fora do retângulo não acontece no aplicativo bancário.

- Atores: Representado pelo boneco palito, no nosso APP temos dois atores, o “Titular” e o “Banco”, quem ou o que vai utilizar nosso aplicativo bancário. O Titular é nosso ator principal (Inicia a utilização do sistema) e o Banco é nosso ator secundário (Reage), os atores principais são posicionados a esquerda do sistema e os secundários a direita, como mostra no esquema.

- Casos de Uso: Representa uma ação que realiza uma tarefa dentro do sistema, são as bolinhas ovais, como exemplo: Sacar, Depositar, Transferir, Empréstimo, Investir e Cartão.

- Relacionamentos: O ator usa o sistema para atingir uma meta, logo, cada ator tem que interagir com pelo menos um caso de uso dentro do nosso sistema, como exemplo o ator “Titular” interage (associa) com “Abrir_Conta” e com “Conta” através de linhas sólidas.

4.2 Diagrama de Classe



A imagem apresenta o diagrama de Classes, descrevendo-as e apresentando a cardinalidade entre elas, através dos sinais “-“ e “+” é representado atributos privados e públicos respectivamente (“-“ para atributos privados e “+” para os públicos).

Classes: No diagrama apresentamos três classes, Titular, Conta e Empréstimo.

Atributos: contém valores que descrevem cada instância dessa classe, dentro das classes estão localizados no quadrado superior.

Métodos: permitem especificar as características comportamentais de uma classe, dentro das classes estão localizados no quadrado inferior.

5. CONSIDERAÇÕES FINAIS

O projeto de desenvolvimento de aplicativos bancários em Java atingiu seu objetivo. Com a ajuda dos conceitos básicos de orientação a objetos e do IntelliJ IDEA, implementamos funções importantes, como a criação de contas, transações de depósito e saque, e a manutenção do histórico de transações.

A estrutura organizada de classes, o uso de herança e encapsulamento, juntamente com a classe AgenciaBancaria para gerenciar as contas, proporcionaram uma experiência suave e intuitiva para os usuários do aplicativo. A classe Pessoa foi utilizada para representar os titulares das contas bancárias, armazenando informações relevantes, como nome, CPF e data de nascimento.

Apesar de ainda existirem oportunidades de desenvolvimento, o projeto permitiu consolidar competências práticas e compreender a importância da modularidade no software bancário. Cada classe desempenhou um papel específico: a classe Conta serviu como base para as subclasses ContaCorrente, ContaInvestimento e ContaPoupanca, permitindo a adição de comportamentos específicos para cada tipo de conta.

Em conclusão, o projeto de aplicativo bancário em Java conseguiu programar funções-chave, como a criação de contas, transações e o registro do histórico de transações. O uso correto dos conceitos de orientação a objetos, como herança e encapsulamento, juntamente com as classes AgenciaBancaria, Conta, ContaCorrente, ContaInvestimento, ContaPoupanca e Pessoa, proporcionou uma experiência de usuário satisfatória. Ainda há espaço para melhorias e desenvolvimento adicional, como a implementação de recursos adicionais e aprimoramentos da interface do usuário.

REFERÊNCIAS

Website wrschools, links que foram usados:

Java Abstraction (w3schools.com)

Java ArrayList (w3schools.com)

Java Class Attributes (w3schools.com)

Java Classes and Objects (w3schools.com)

Java Class Methods (w3schools.com)

Java Constructors (w3schools.com)

Java Date and Time (w3schools.com)

Java Encapsulation and Getters and Setters (w3schools.com)

Java Exceptions (Try...Catch) (w3schools.com)

Java Inheritance (Subclass and Superclass) (w3schools.com)

Java Inner Class (Nested Class) (w3schools.com)

Java Iterator (w3schools.com)

Java LinkedList (w3schools.com)

Java Modifiers (w3schools.com)

Java OOP (Object-Oriented Programming) (w3schools.com)

Java Packages (w3schools.com)

Java Polymorphism (w3schools.com)

Java Threads (w3schools.com)

Java User Input (Scanner class) (w3schools.com)

Java Wrapper Classes (w3schools.com)