# BANKAPP: RELATO DE EXPERIÊNCIA SOBRE O DESENVOLVIMENTO DE UM SISTEMA BANCÁRIO NO PROJETO INTEGRADOR

Diego Moura Araújo[1]
Géssica Gomes Melo [2]
Leiliane Silva de Morais[3]
Leonardo do Nascimento Peixoto da Silva[4]

## RESUMO

O seguinte trabalho tem como objetivo apresentar um sistema bancário desenvolvido e implementado aplicando os conhecimentos adquiridos sobre Programação Orientada a Objetos (POO). Para isso, foi realizado primeiramente uma análise de como o sistema comportar-se-ia mediante dos diagramas de casos de uso e de classe. Por meio deles, construiu-se uma aplicação em que se pode, usando o agente Administrador: criar contas em diferentes agências; executar saques, depósitos e transferências entre elas; consultar extratos; e realizar rendimento. Ao final, esse projeto conseguiu utilizar e explorar amplamente os 4 pilares da Programação Orientada a Objetos, possibilitando, assim, um entendimento maior de como esse poderoso conceito pode ser aproveitado no dia a dia dos profissionais de desenvolvimento.

**Palavras-chaves**: POO, sistema bancário, desenvolvimento de softwares

## ABSTRACT

The following paper aims to present a banking system developed and implemented applying the knowledge acquired on Object-Oriented Programming (OOP). For this, an analysis of how the system would behave was first performed through the diagrams of use cases and class. Through them, an application was built in which one can, using the Manager agent: create accounts in different agencies; execute withdrawals, deposits and transfers between them; consult extracts; and realize income. In the end, this project was able to widely use and explore the 4 pillars of Object-Oriented Programming, thus enabling a greater understanding

[1] Graduando do Curso de Sistemas para Internet E-mail: 2022215510093@iesp.edu.br
[2] Graduando do Curso de Sistemas para Internet E-mail: 2022211510030@iesp.edu.br
[3] Graduando do Curso de Sistemas para Internet E-mail: 2022211510005@iesp.edu.br
[4] Graduando do Curso de Sistemas para Internet E-mail: 2022211510033@iesp.edu.br

of how this powerful concept can be leveraged in the daily lives of development professionals.

**Keywords: OOP, banking system, software development**

## 1 INTRODUÇÃO

Ao passar dos anos, e com o desenvolvimento das tecnologias, foram implementados ajustes em várias áreas e uma delas foi a do setor bancário. No Brasil, a digitalização dos bancos se iniciou há pouco mais de cinquenta anos, com o lançamento do primeiro cartão de crédito. Nesse contexto, foi vista uma necessidade de trazer otimização e praticidade para seus usuários.

Visando atender tal demanda, os bancos digitais surgem e trazem consigo uma série de benefícios. Entre eles:

- A criação de uma conta de banco segura em minutos, apenas com o uso do seu celular e um aplicativo (agilidade na abertura das contas);

- Ter a opção de obter um cartão de crédito sem anuidade e/ou uma conta corrente sem tarifa;

- Facilidade em aplicar para uma variedade de diferentes tipos de investimentos;

- A possibilidade de se obter informações que anteriormente só era possível indo a uma agência bancária;

- E o cliente, agora, pode optar por outras formas de gerir seus bens e, acima de tudo, seu tempo (ganho desse recurso em atividades que, antes, eram só realizadas em uma unidade física do banco) .

Diante dessa situação, o presente trabalho teve como objetivo relatar a experiência adquirida durante o desenvolvimento de um sistema baseado em bancos digitais, o BankApp. Esse dispõe de um usuário principal, o administrador, que pode: criar contas em agências pré-existentes; executar depósito/saque de uma determinada conta em um agência; transferir dinheiro entre contas de mesmas agências ou distintas; consultar extrato de uma conta; e realizar rendimentos em conta poupança e de investimento.

## 2 FUNDAMENTAÇÃO TEÓRICA

### 2.1 PROGRAMAÇÃO ORIENTADA A OBJETOS

Quando se trata de desenvolvimento de softwares para solução de problemas, esse pode ser realizado de diferentes maneiras. Em tal cenário gigantesco, surge diversos paradigmas de programação que têm como objetivo principal ajudar desenvolvedores a implementar seus programas em uma determinada visão e lógica.

Um dos modelos mais conhecidos em relação a paradigmas de programação é Programação Orientada a Objetos (POO). Esse consiste em abstrair conceitos e coisas do mundo real para um contexto em que todos são tratados como objetos (os quais possuem atributos, características,  e ações - métodos) de determinadas classes (DEVMEDIA, 2023).

Como a maioria dos conceitos concebidos no mundo da computação, a POO possui uma base bem estabelecida. Essa dividida em 4 pilares fundamentais associados: Abstração, Encapsulamento, Herança e Polimorfismo. Os primeiros deles, Abstração e Encapsulamento, podem ser um pouco complicados de compreender em um primeiro momento. A Abstração consiste na ideia de representar o que é realmente seu problema para o paradigma POO, ocultando partes que são consideradas, por ora ou não, não importantes para o sistema. Já o Encapsulamento, como o nome sugere, surge na necessidade de definir quem terá acesso - e de qual modo - aos seus dados. Por meio dessa concepção, consegue-se trabalhar questões relevantes como a segurança e confiabilidade de seus dados nos objetos. (TOLEDO, 2023).

Os dois últimos estão bastante associados com área de Biologia. A herança acontece quando se tem uma classe X e se quer aproveitar - no caso, herdar - suas características e ações para uma outra. Por exemplo, pode-se ter uma classe denominada Pessoa e uma outra chamada Herói. A Herói herda as características (altura, peso, data de nascimento, nome e etc…) e ações (comer, andar, falar, pensar e etc…) de Pessoa ao mesmo tempo que pode ter outras propriedades particulamente dele como ação de Voar ou possuir um nome específico de Herói. Enquanto isso, o Polimorfismo aparece com um ideia de que uma mesma ação pode ter comportamentos divergentes. Um exemplo clássico é da relação de gatos e cachorros. Ambos animais (de classes diferentes que possuem uma classe Mãe em comum, classe Mamífero) são capaz de emitir som, mas, como dito anteriormente, com de maneiras diferentes (HENRIQUE, 2023).

## 3 METODOLOGIA

Para o desenvolvimento da aplicação, foram-se elaborados os diagramas de casos de uso e de classes primeiramente. Após esse esboço inicial que se teve para o projeto, iniciou a

implementação pelo código. As ferramentas utilizadas para isso foram a linguagem de programação Java e o ambiente de desenvolvimento Intellij, além de conhecimentos fundamentais em POO.

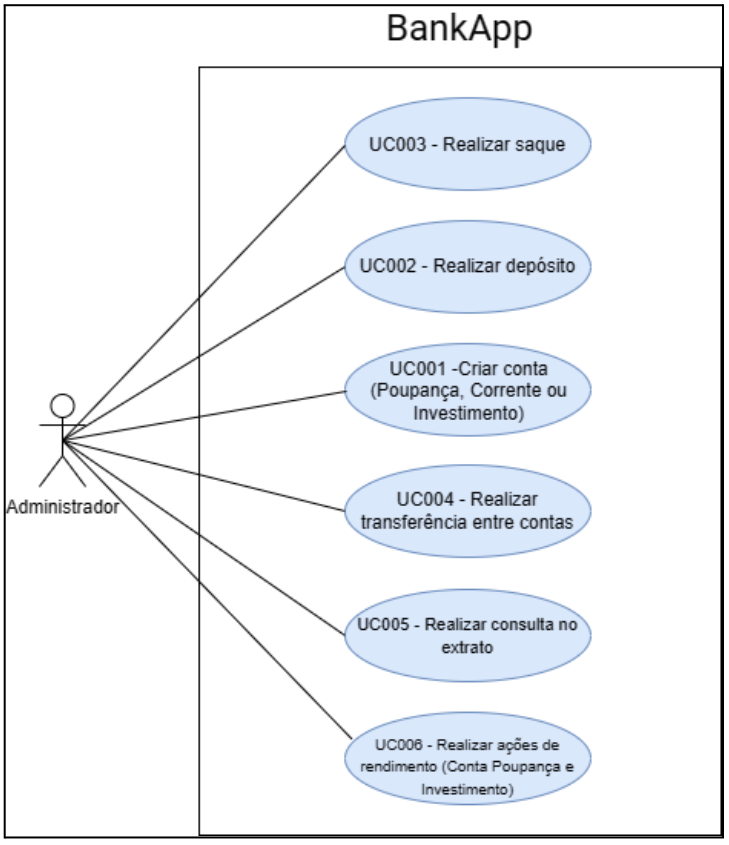
## 4 RESULTADO E DISCUSSÃO

### 4.1 Diagrama de Casos de Uso

Sobre o diagrama de Casos de Uso (Figura 1), esse foi projetado de uma maneira bastante simples, demonstrando as operações que o sistema, nesse momento, é capaz de realizar. Nele, encontra-se apenas um agente (Administrador) que pode fazer as seguintes operações no programa:

1. Criar conta (tipos Poupança, Corrente ou Investimento);
2. Sacar;
3. Depositar;
4. Transferir entre contas;
5. Consultar extrato;
6. Render saldo nas contas Poupança e Investimento

Figura 1 - Diagrama de Casos de Uso do sistema BankApp.



Fonte: própria, 2023.

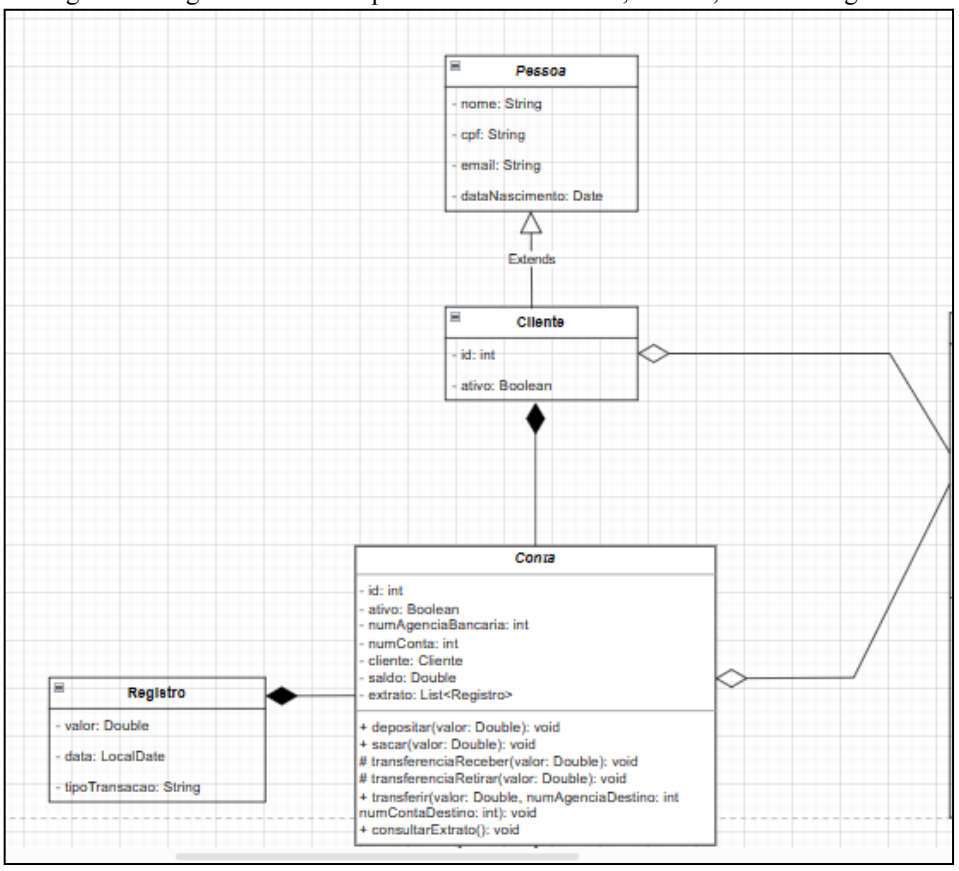
No futuro, espera-se que esse agente Administrador possa ser ramificado em mais um, o Cliente. Assim, esse novo agente ficaria responsável por essas operações autenticado em sua conta e o Administador se ocuparia de funções mais generalistas como: abrir nova agência; ativar/desativar conta; e procurar conta ou agência - com suas informações - pelo seu número.

## 4.2 Diagrama de Classe

Em relação ao diagrama de classe, conforme as figuras 2, 3, 4, 5 e 6, esse está disposto da seguinte forma:

A classe abstrata Pessoa possui uma classe filha, Cliente, que se relaciona com a classe abstrata Conta. Tal classe conta ainda é composta por um atributo que tem o tipo da classe Registro (valor, data e tipo de transação) (Figura 2).

Figura 2: Diagrama de classes parte I - Classes Pessoa, Cliente, Conta e Registro.

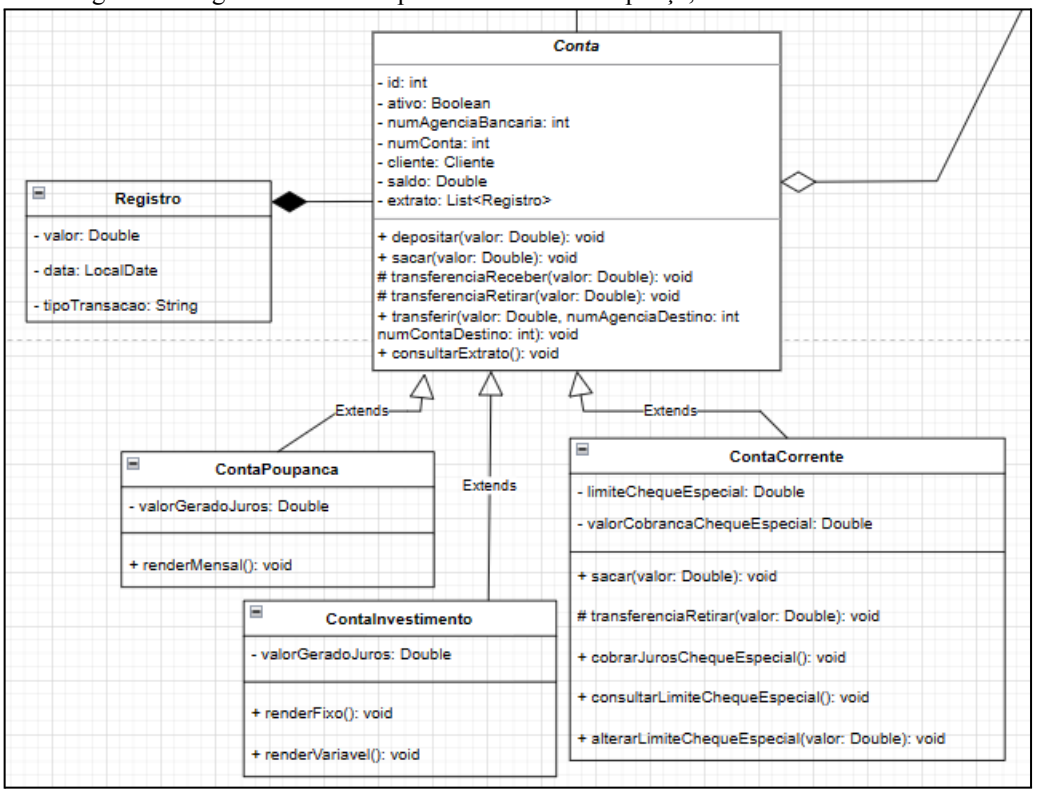



Fonte: própria, 2023.

Essa classe abstrata Conta, como se pode notar na figura 3, ainda possui três classes filhas que herdam seus atributos e métodos: classe Conta Poupança (que beneficia o cliente

por meio de juros de 2% em cima de seu saldo no dia de seu aniversário em cada mês); Conta Corrente (que possui cheque especial, ou seja, caso o cliente não tenha saldo, ele pode recorrer ao banco até um limite determinado e com juros); e Conta Investimento, essa dispõe de dois métodos diferentes: renderFixo (gera um rendimento de 10% sobre o saldo) e renderVariavel (a taxa de juros do rendimento pode variar entre 10% a 25%).

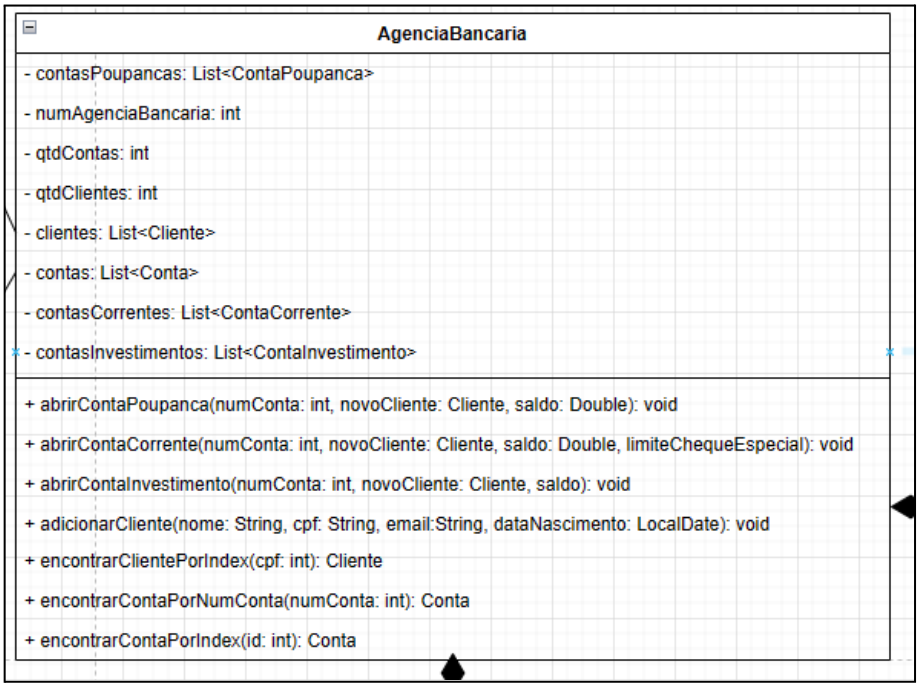
Figura 3: Diagrama de classes parte II - Classes Poupança, Corrente e Investimento.




Fonte: própria, 2023.

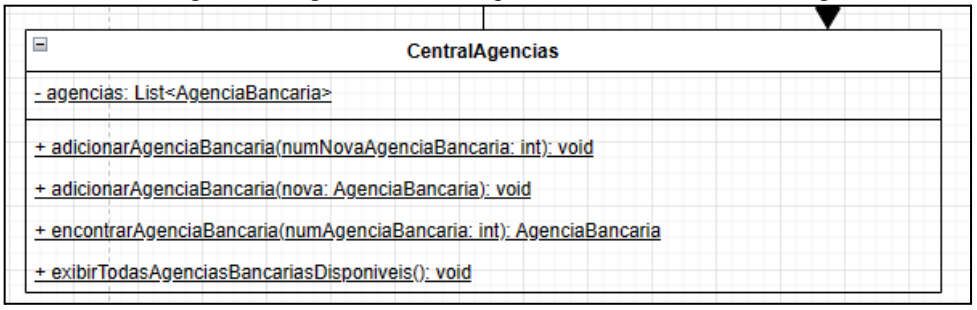
Por fim, a última parte, figuras 5, 6 e 7, é onde está localizado o núcleo do sistema (além da parte da contas), com as classes: Agência bancária, Central de agências e Interface BankApp. A primeira delas é composta por uma Conta (Poupança ou Corrente) e Cliente, além de outros dados, como quantidade de clientes e contas, e principais operações feitas pela agência . Já a segunda é o setor responsável por cuidar das agências registradas. Nele, terão as seguintes ações estáticas: adicionar uma nova agência (ou uma agência já existente) em uma lista; encontrar uma agência por seu número; e exibir todas as agências adicionadas na lista pré-estabelecida. Já última classe, Interface BankApp, é encarregada em preparar os outros e mecanismos para se apresentarem para os usuário final: administrador.

Figura 5: Diagrama de classes parte III - Classe Agência bancária.



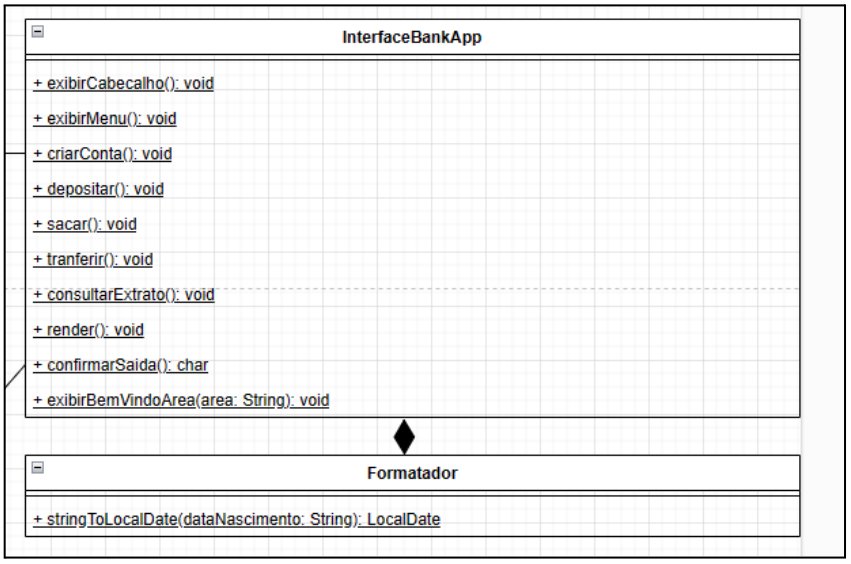

Fonte: própria, 2023.

Figura 6: Diagrama de classes parte IV - Classe Central de Agências.




Fonte: própria, 2023.

Figura 7: Diagrama de classes parte V - InterfaceBankApp e Formatador (que auxilia na conversão de String para LocalDate)..




Fonte: própria, 2023.

## 5 CONSIDERAÇÕES FINAIS

Diante do que foi apresentado durante esse trabalho, pode-se notar que diversos conceitos de: classes; classes abstratas; agregações e composições; encapsulamento de dados; e herança e polimorfismo de sobreposição entre classes foram postos, empregados, para que esse foi executado.

Com isso, tal projeto conseguiu utilizar e explorar amplamente os 4 pilares da Programação Orientada a Objetos (Abstração, Encapsulamento, Herança e Polimorfismo), possibilitando, assim, um entendimento maior de como esse poderoso conceito pode ser aproveitado no dia a dia dos profissionais da área de desenvolvimento de softwares.

## REFERÊNCIAS

HENRIQUE, J. POO: O que é programação orientada a objetos?. **ALURA**, 2023. Disponível em: <https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos>. Acesso em: 05 de junho de 2023.

OS 4 pilares da Programação Orientada a Objetos. **DEVMEDIA**, 2023. Disponível em: <https://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>. Acesso em: 05 de junho de 2023.

TOLEDO, S. Os 4 pilares da Programação Orientada a Objetos. **SÉRGIO TOLEDO**, 2023. Disponível em: <https://www.sergiotoledo.com.br/artigos/os-4-pilares-da-programacao-orientada-a-objetos>. Acesso em: 05 de junho de 2023.