Giullia Maria de Camargo Ferreira Meneses - GU305554X
Maria Eduarda Rodrigues - GU3055108

Relatório de Arquitetura e Design: Protótipo VcRiquinho

1. Introdução

O presente documento descreve as decisões de design de software aplicadas no desenvolvimento do protótipo "VcRiquinho". A implementação foca na aplicação prática dos pilares da Programação Orientada a Objetos (POO) com a linguagem Java, visando a escalabilidade e a manutenibilidade do sistema financeiro.

2. Fundamentação Técnica e Decisões de Projeto

2.1. Abstração e Generalização

Foram empregadas **classes abstratas** para as entidades `Cliente` e `Conta`.

- **Justificativa:** No domínio bancário, não faz sentido instanciar um "Cliente" genérico, mas sim especializações concretas (Pessoa Física ou Jurídica). A abstração permitiu definir o comportamento comum (atributos de identificação) enquanto delega às subclasses a implementação de regras fiscais específicas.

2.2. Programação por Interface

A hierarquia de investimentos foi modelada através da **Interface** `ProdutoInvestimento`.

- **Justificativa:** Ao utilizar uma interface em vez de uma classe concreta, estabelecemos um "contrato de rendimento". Isso permite que o sistema seja aberto para extensão e fechado para modificação (Princípio Open/Closed do SOLID), possibilitando a adição de novos produtos financeiros sem impactar a lógica de simulação existente.

2.3. Encapsulamento e Integridade

O sistema aplica um **encapsulamento rigoroso**.

- **Justificativa:** O uso de modificadores `private` e `protected`, associado a métodos acessores e modificadores (Getters/Setters), garante que o estado interno do objeto (como o saldo bancário) só seja alterado através de métodos de negócio que validam as operações, prevenindo estados inconsistentes.

2.4. Polimorfismo de Inclusão e Dinâmico

O polimorfismo é a base da funcionalidade de simulação.

- **Justificativa:** O método `simular(int dias)` é resolvido em tempo de execução (*late binding*). Isso permite tratar uma lista de contas diversas de forma uniforme, onde cada objeto responde com sua própria lógica de cálculo (CDI, Corrente ou Investimento Automático) de acordo com sua especialização.

3. Conclusão

A arquitetura adotada separa as Responsabilidades de Modelo (dados), Serviços (lógica de gerenciamento CRUD) e Aplicação (interface com o usuário). Essa estrutura não apenas atende aos requisitos funcionais do projeto, mas também demonstra a aplicação de padrões de design que facilitam a evolução do software para ambientes de produção.