

## **FASE 1 - CONCEITUAÇÃO**

### **Ritual Comum de Processamento de Pagamentos**

O processamento de pagamentos segue um **ritual fixo** comum a todos os meios de pagamento:

#### **Validar → Autorizar/Capturar → Confirmar**

Cada etapa possui variações específicas conforme o meio de pagamento:

- **Cartão:**
  - Validação: dados do cartão (número, CVV, validade)
  - Autorização/Captura: comunicação com operadora do cartão
  - Confirmação: emissão de comprovante com código de autorização
- **PIX:**
  - Validação: chave PIX (CPF, e-mail, telefone)
  - Autorização/Captura: geração de QR Code e espera do pagamento
  - Confirmação: notificação via webhook e comprovante com código transação
- **Boleto:**
  - Validação: dados do emitente e beneficiário
  - Autorização/Captura: geração de linha digitável e instruções
  - Confirmação: aguardar compensação bancária e comprovante com data

### **Políticas Independentes**

A plataforma aplica políticas combináveis por pedido:

- **Antifraude:** análise de risco baseada em valor e histórico
- **Câmbio:** conversão de moeda para transações internacionais

### **Justificativa das Decisões de Design**

#### **Herança para meios de pagamento:**

Cada meio (Cartão, PIX, Boleto) é uma especialização legítima do ritual comum, com variações controladas nos mesmos passos.

#### **Composição para políticas:**

Antifraude e câmbio são características independentes que podem ser

combinadas livremente com qualquer meio de pagamento, evitando explosão combinatorial de subclasses.

## **FASE 2 - DESIGN ORIENTADO A OBJETOS**

### **Contrato da Classe Base Pagamento**

#### **Método Público Principal:**

- **Processar():** orquestra o ritual fixo na sequência: Validar → Aplicar Antifraude → Aplicar Câmbio → AutorizarOuCapturar → Confirmar

#### **Ganchos Protegidos para Especialização:**

- **Validar():** verifica dados específicos do meio de pagamento
- **AutorizarOuCapturar(valor):** executa a autorização ou geração do método de pagamento
- **Confirmar():** finaliza o processo e emite comprovante
- **ObterValor():** retorna o valor base para cálculos

### **Regras LSP (Princípio da Substituição de Liskov)**

1. **Substituibilidade:** Qualquer código cliente que utiliza a classe base Pagamento deve continuar funcionando corretamente quando recebe instâncias de PagamentoCartao, PagamentoPix ou PagamentoBoleto, sem necessidade de verificações de tipo específico
2. **Invariantes Preservadas:** As validações essenciais definidas na classe base devem ser mantidas ou fortalecidas pelas classes derivadas, nunca enfraquecidas ou ignoradas
3. **Contratos de Saída Equivalentes:** O método Processar() sempre produz um resultado coerente - seja confirmação bem-sucedida ou exceção apropriada - independente do meio de pagamento específico

### **Eixos Plugáveis por Composição (Delegates)**

#### **Antifraude:**

- **Assinatura:** função que recebe decimal (valor) e retorna bool (aprovado/reprovado)
- **Papel:** aplica regras de análise de risco e segurança na transação
- **Exemplo:** pode reprovar transações acima de determinado limite

#### **Cambio:**

- **Assinatura:** função que recebe decimal (valor original) e retorna decimal (valor convertido)
- **Papel:** realiza conversão monetária quando necessário
- **Exemplo:** aplica taxa de câmbio para converter de real para dólar

#### **Estrutura das Classes Derivadas**

- **PagamentoCartao:** especializa validacao para dados de cartão, autorização via operadora, confirmação com código de autorização
- **PagamentoPix:** especializa validacao para chave PIX, geração de QR Code, confirmação via webhook
- **PagamentoBoleto:** especializa validacao para dados bancários, geração de linha digitável, confirmação por compensação

**Restrição Crítica:** As classes derivadas são seladas e não introduzem novos métodos públicos, mantendo pura especialização do comportamento existente.