

## FASE 1 - CONCEITUAÇÃO

### Ritual Comum de Processamento de Pedidos

O processamento de pedidos segue um ritual fixo e consistente composto por três etapas sequenciais:

1. **Validar** - verifica a integridade dos dados do pedido e itens incluídos
2. **Calcular Total** - determina o valor final considerando subtotal e políticas aplicáveis
3. **Emitir Recibo** - gera o documento fiscal apropriado para o cliente

### Variações entre Pedidos Nacionais e Internacionais

Para **Pedidos Nacionais**, as variações concentram-se em:

- Aplicação de impostos internos (ICMS, ISS, PIS, COFINS)
- Formatação de recibo conforme modelo fiscal brasileiro (NF-e)
- Validações específicas do regime tributário nacional

Para **Pedidos Internacionais**, as diferenças principais são:

- Cálculo de taxas de importação e custos aduaneiros
- Conversão monetária considerando câmbio flutuante
- Emissão de Commercial Invoice como documento de saída
- Validações alfandegárias e restrições de exportação

### Justificativa para Uso de Herança

A herança é aplicada para **especialização de ritual fixo**, onde o fluxo principal permanece idêntico, mas pontos específicos (cálculo de subtotal e formato de recibo) variam conforme o tipo de pedido. Isso permite reutilizar a orquestração central enquanto especializa apenas os componentes que diferem.

### Justificativa para Uso de Composição

Políticas como frete, embalagem, seguro e promoção são **características independentes e combináveis** que podem ser aplicadas a qualquer tipo de pedido. A composição via delegates permite trocar essas políticas livremente sem criar uma explosão combinatorária de subclasses, mantendo o acoplamento baixo e a flexibilidade alta.

## FASE 2 - DESIGN OO

## **Contrato da Classe Base Pedido**

A classe Pedido define o seguinte contrato público e protegido:

- **Método Público:** Processar(): string - orquestra o ritual fixo
- **Ganchos Protegidos Virtual:**
  - protected virtual void Validar() - validações específicas (opcional)
  - protected virtual decimal CalcularSubtotal(): decimal - cálculo base antes das políticas
  - protected virtual string EmitirRecibo(decimal total): string - formatação do recibo

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

1. **Substituibilidade Total:** Qualquer código cliente que opere com a classe base Pedido deve continuar funcionando corretamente quando receber instâncias de PedidoNacional ou PedidoInternacional, sem necessidade de verificações de tipo ou downcasting.
2. **Preservação de Invariantes:** As classes derivadas devem manter ou fortalecer as pré-condições e pós-condições estabelecidas pela classe base, especialmente nas validações básicas do pedido.
3. **Consistência de Contratos:** O método Processar() deve sempre retornar um recibo formatado corretamente e coerente com o total calculado, sem introduzir exceções não esperadas pelo contrato da base.

## **Eixos Plugáveis via Delegates**

- **Frete:** Func<decimal, decimal> - Aplica custos de transporte sobre o valor corrente do pedido. Exemplo: frete fixo de R\$ 15,00 ou percentual de 5%.
- **Promocao:** Func<decimal, decimal> - Aplica descontos ou cupons promocionais. Exemplo: desconto fixo de R\$ 10,00 ou desconto percentual de 15%.