Documento de Arquitetura - Sistema de Lista de Compras

(Tema: Lista de Compras)

1. Contexto e Decisões Tecnológicas

O objetivo é desenvolver uma aplicação para gerenciar uma **Lista de Compras**. O sistema permite adicionar itens que precisamos comprar, definir a quantidade e o preço.

- **Linguagem:** Java (JDK 8 ou superior).
- **Interface Gráfica (GUI):** Swing (javax.swing).
- **Estrutura de Dados:** ArrayList (Lista em memória).
- **Padrão de Arquitetura:** MVC (Model-View-Controller).

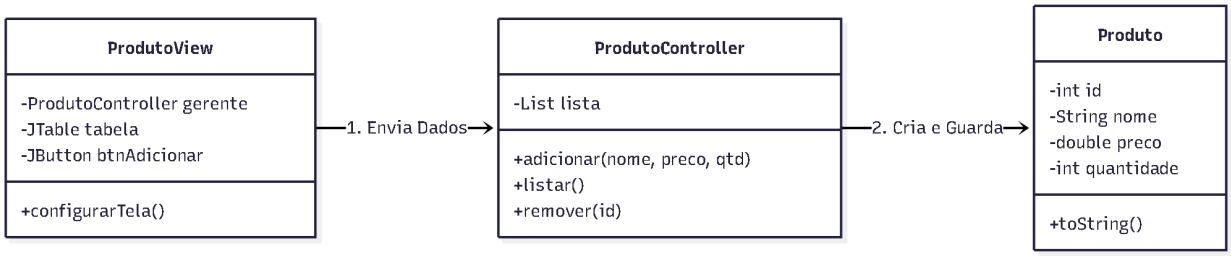
2. Estrutura do Projeto (Pacotes)

A organização segue a hierarquia padrão. Cada integrante deve trabalhar no seu respectivo pacote.

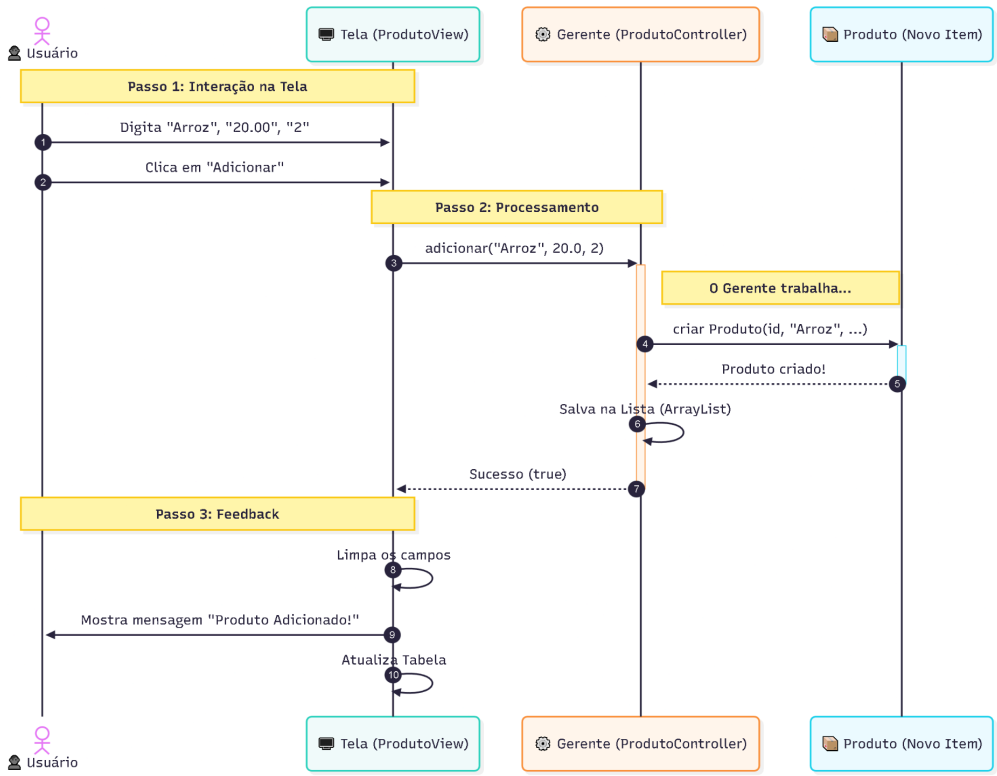
```
ProjetoListaCompras/
│
└── src/
    └── com/projeto/
        │
        ├── model/
        │   └── Produto.java          # Representa um ITEM da lista de compras
        │
        ├── controller/
        │   └── ProdutoController.java  # Gerencia a lista (adicionar item, remover, etc)
        │
        └── view/
            └── ProdutoView.java       # Inicialização e tela da Lista de Compras
```

3. Diagrama de Classes (UML)

Obs :: Produto representa o "Item da Lista de Compras".



4.FLUXOGRAMA:



5. Especificação Técnica (Contrato das Classes)

5.1. Classe Modelo (model.Produto)

Representa um item que vai para o carrinho.

- **Atributos:**
  - nome: O que comprar (ex: "Arroz", "Leite").
  - preço: Preço unitário estimado ou máximo a pagar.
  - quantidade: Quantas unidades comprar.
- **Encapsulamento:** Usar Getters e Setters.
- **Método toString():** Formatar para algo legível na lista, ex: "Leite (2x) - R$ 4.50".

5.2. Classe Controller (controller.ProdutoController)

Controlador da lista de compras.

- **Lista:** List<Produto> produtos = new ArrayList<>();.
- **Adicionar:** Ao adicionar um item, o ID é gerado automaticamente (contadorId++).
- **Regra de Negócio:** Se tentar remover um item que não existe, retornar false.
- **Totalização (Opcional/Extra):** Se sobrar tempo, criar um método calcularTotalEstimado() que soma preco * quantidade de todos os itens.

5.3. Classe View (view.ProdutoView)

A interface visual onde o usuário monta a lista.

- **Componentes:**
  - Campos de texto para: Item, Valor Unitário, Quantidade.
  - Botões: "Adicionar à Lista", "Remover Item", "Editar Item".
  - Tabela: Deve mostrar as colunas [ID | Item | Preço | Qtd].
- **Layout:** Pode usar GridLayout para os campos e BorderLayout para a tabela.

6. Fluxo de Trabalho (Divisão de Tarefas)

1. **Dev 1 (Modelo):** Cria a classe Produto (o item).
2. **Dev 2 (Controller):** Cria a lógica da lista (adicionar no Array, remover do Array).
3. **Dev 3 e 4 (View):** Criam a janela. Um foca no design (posicionar botões) e o outro foca nos eventos (fazer o botão chamar o Controller).