Relatório: Sistema de Gerenciamento de Produtos

Ian Monteiro / Renata Bueno

2 de Dezembro de 2025

ESTRUTURA DO PROJETO

O projeto foi desenvolvido em **Python** utilizando a biblioteca gráfica **Tkinter**. A arquitetura do software segue estritamente o padrão de projeto **MVC (Model-View-Controller)**, o que garante a separação de responsabilidades e facilita a manutenção do código. O sistema é composto por três módulos principais orquestrados pelo arquivo `main.py`:

- **Model (`produto.py`):** Responsável pela representação dos dados. A classe `Produto` utiliza encapsulamento (atributos privados como `__id`, `__nome`) e fornece métodos acessores (*Getters* e *Setters*) para garantir a integridade das informações.
- **View (`produtoView.py`):** Gerencia a interação com o usuário. Implementa a interface gráfica utilizando widgets do Tkinter (`Entry`, `Label`, `Button`) e exibe os dados em uma tabela (`ttk.Treeview`). Esta camada captura eventos e delega as ações para o Controller, exibindo feedback visual através de `messagebox`.
- **Controller (`produtoController.py`):** Contém a regra de negócios e o gerenciamento de estado. Mantém uma lista em memória (`self.produtos`) e fornece métodos para manipular esses dados, servindo como ponte entre a interface e o modelo de dados.

DESCRIÇÃO DOS MÉTODOS DO CRUD

As operações de CRUD (*Create, Read, Update, Delete*) foram implementadas através da manipulação direta de estruturas de dados em memória (listas de objetos), conforme detalhado abaixo:

1. **Create (Adicionar):** A View captura os dados dos campos de texto, instancia um novo objeto `Produto` e o envia ao Controller. O método `adicionar` do Controller insere este objeto na lista interna utilizando o método `append()`.
2. **Read (Listar):** O método `listar` retorna a lista completa de objetos armazenados. A View itera sobre essa lista, extraindo os atributos via *Getters* e preenchendo as linhas do componente `Treeview`. A listagem é atualizada automaticamente após inserções, edições ou remoções.
3. **Update (Atualizar):** Para atualizar, o sistema busca o produto pelo ID. O Controller percorre a lista de produtos; ao encontrar o ID correspondente, utiliza os métodos *Setters* (`set_nome`, `set_preco`, etc.) para modificar os atributos do objeto existente com os novos valores fornecidos pela View.
4. **Delete (Remover):** A remoção também ocorre via ID. O Controller itera sobre a lista de produtos, localiza o objeto com o ID informado e o remove da lista utilizando o método `list.remove()`. A interface exibe uma mensagem de sucesso ou erro caso o ID não seja encontrado.

DECISÕES DE DESIGN

Para atender aos requisitos de simplicidade e organização acadêmica, foram tomadas as seguintes decisões:

- **Adoção do Padrão MVC:** Escolhido para desacoplar a lógica visual da lógica de dados. Isso permite que a interface gráfica seja alterada no futuro sem necessidade de reescrever as regras de negócio ou a estrutura dos dados.
- **Armazenamento em Memória:** Optou-se por utilizar listas voláteis em vez de um banco de dados persistente (SQL). Essa decisão simplifica a execução do projeto, eliminando a necessidade de configurações de ambiente ou arquivos de banco de dados externos.
- **Encapsulamento:** O uso de atributos privados na classe `Produto` protege os dados contra modificações acidentais diretas, forçando o uso de métodos seguros para alteração de estado.