Parte 1 - Relatório de Testes

# 1. Identificação do Projeto

Nome do Sistema: Cadastro de Produtos 1

Responsáveis pelo teste: Alessandra Leite

Data de Início: 10/06/2025

Data de Conclusão: \_\_\_\_/\_\_\_\_/\_\_\_\_\_\_\_\_

# 2. Objetivo do Sistema

Objetivo do sistema é cadastrar produto, para o sucesso no cadastro cada produto deve conter: id, nome, preço e estoque.

# 3. Plano de Teste

Observações de entradas e testes automáticos com auxílio do pytest.

# 4. Resultados dos Testes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Tipo de Teste | Descrição | Resultado Esperado | Resultado Obtido | | Teste Unitário | Entradas: código, nome, preço e estoque | Erro no cadastro por falta de dados | Cadastro é feito com falta de dados | | Teste Sistema | Funcionalidade | Cadastro de múltiplos produtos | Apenas um produto é cadastrado | | Teste Sistema | Funcionalidade | Após um erro, deve permitir a correção deste e continuar o cadastro | Após o erro o sistema é encerrado | | Automático | Usando o pytest | Dois testes passarem | Um teste passou , e o outro falhou | |

# 5. Conclusão

Descreva aqui as falhas encontradas, correções realizadas e sugestões para melhoria do sistema.

Tratamento de dados, ele não possui restrições o que pode permitir o usuário a cometer erros de cadastro. Correções a serem feitas nessas falhas seria a verificação dos dados antes de atribuir o valor a alguma variável.

O sistema apenas é executado uma vez e apenas cadastra um produto. A implementação de um comando de repetição e a criação de uma lista seria uma boa prática, pois vai permitir que mais produtos sejam cadastrados.

Não permite a correção de algum erro durante o cadastro. O sistema não tem muitas restrições seria uma boa prática ou a confirmação das informações antes de salvar ou não permitir a entrada de dados nula.

Sugestões finais:

* Adicionar comando de repetição, para que o usuário consiga cadastrar mais de um produto;
* Criação de uma lista ou um arquivo .txt, para fazer o armazenamento dos dados dos produtos;
* Verificação ou a correção de dados, permitir a correção de dados ou sempre fazer a verificação antes de salvar os dados;
* Implementação de um CRUD, sistema básico para o usuário onde ele vai conseguir cadastrar o produto, corrigir produto, excluir produto e listar os produtos.

Parte 2 – AtividadeControle de Versão

**Atividade Prática: "Controle de Versão com Testes Automatizados"**

**Descrição:**  
Cada aluno deverá:

1. **Criar um repositório no GitHub** com o nome sistema-produto-teste.
2. **Adicionar os arquivos** produto.py e test\_produto.py.
3. **Executar os testes com Pytest** e **registrar os resultados** (pode ser em um arquivo .md ou .txt no repositório).
4. Fazer **commits separados** com mensagens claras para cada etapa:
   * Primeiro commit: feat: adicionar módulo de produto
   * Segundo commit: test: criar testes automatizados para o sistema de produto
   * Terceiro commit: doc: adicionar resultado dos testes
5. Criar uma **branch** chamada ajustes e modificar uma função (ex: adicionar tratamento de erro ou nova categoria).
6. Fazer **merge da branch** com a principal (main) usando **Pull Request** (PR).
7. Compartilhar o link do repositório com o professor e turma no **Teams** atividade Teste de Sistema.

**Checklist para resolver possíveis problemas**

1. **O arquivo de teste deve começar com test\_**  
   Exemplo correto:

test\_produto.py

Exemplo errado:

* + produto\_test.py

1. **As funções de teste também devem começar com test\_**  
   Exemplo:

def test\_valor\_total():

assert calcular\_total(10, 2) == 20

1. **Não pode haver erros de indentação ou sintaxe.**  
   Um erro na estrutura do código pode fazer o pytest ignorar tudo.
2. **O arquivo está salvo corretamente na pasta onde o pytest foi executado?**  
   Ele precisa estar no diretório certo quando você roda o comando pytest.
3. **Não coloque o código de teste dentro de if \_\_name\_\_ == "\_\_main\_\_"**  
   O pytest ignora isso, pois procura funções globalmente acessíveis que comecem com test\_.

Referencias

<https://www.youtube.com/watch?v=vvn2ZKnT_e8>