Entrega da Atividade Prática: Ciclo de Vida de Testes de Software

1 Análise de Requisitos

1.1 Requisitos Funcionais

- O sistema deve permitir adicionar tarefas.
- O sistema deve permitir remover tarefas existentes.
- O sistema deve listar todas as tarefas cadastradas.
- Se não houver tarefas cadastradas, deve exibir uma mensagem apropriada.

1.2 Critérios de Aceitação

ID	Requisito	Critério de Aceitação
RQ01	Adicionar uma tarefa	O sistema deve armazenar a tarefa e exibir uma confirmação.
RQ02	Remover uma tarefa existente	O sistema deve excluir a tarefa e exibir uma confirmação.
RQ03	Listar todas as tarefas	O sistema deve retornar a lista de tarefas cadastradas.
RQ04	Nenhuma tarefa cadastrada	O sistema deve exibir a mensagem "Nenhuma tarefa cadastrada".

2 Planejamento de Teste

2.1 Escopo dos Testes

- Testes funcionais
- Testes de unidade
- Testes de regressão
- Testes negativos

2.2 Ferramentas Utilizadas

- Python unittest
- Jupyter Notebook

3 Design de Teste

ID	Caso de Teste	Entrada	Saída Esperada
CT01	Adicionar uma tarefa válida	"Estudar para a prova"	"Tarefa adicionada"
CT02	Adicionar uma tarefa vazia		"Tarefa inválida"
CT03	Remover uma tarefa existente	"Estudar para a prova"	"Tarefa removida"
CT04	Remover uma tarefa inexistente	"Ir ao cinema"	"Tarefa não encontrada"
CT05	Listar tarefas com itens cadastrados	"Estudar para a prova"	["Estudar para a prova"]
CT06	Listar tarefas sem itens cadastrados	—	"Nenhuma tarefa cadastrada"

4 Execução dos Testes

4.1 Código de Teste Automatizado

```
import unittest
from task_manager import TaskManager

class TestTaskManager(unittest.TestCase):
    def setUp(self):
        self.tm = TaskManager()

    def test_add_task(self):
        self.assertEqual(self.tm.add_task("Estudar para prova"), "Tarefa adicionada")
        self.assertIn("Estudar para prova", self.tm.list_tasks())

    def test_add_empty_task(self):
        self.assertEqual(self.tm.add_task(""), "Tarefa inválida")

    def test_remove_existing_task(self):
        self.tm.add_task("Estudar")
        self.assertEqual(self.tm.remove_task("Estudar"), "Tarefa removida")

    def test_remove_non_existing_task(self):
        self.assertEqual(self.tm.remove_task("Academia"), "Tarefa não encontrada")

    def test_list_tasks(self):
        self.tm.add_task("Trabalho")
        self.assertEqual(self.tm.list_tasks(), ["Trabalho"])

    def test_list_tasks_empty(self):
        self.assertEqual(self.tm.list_tasks(), "Nenhuma tarefa cadastrada")
```

5 Relatório Final

- **Total de Casos de Teste:** 6
- **Casos de Teste Aprovados:** 6
- **Casos de Teste com Falha:** 0
- **Principais Defeitos Encontrados:** Nenhum
- **Sugestões de Melhorias:**
  - Permitir edição de tarefas.
  - Implementar persistência de dados.