



**UNIVERSIDADE DE FORTALEZA
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA: PROGRAMAÇÃO FUNCIONAL**

SISTEMA DE GERENCIAMENTO DE TAREFAS

**EQUIPE: 22
BRUNA MICHELLE DE QUEIROZ
MATRÍCULA: 2319452**

**FORTALEZA-CE
MARÇO/2025**



1. INTRODUÇÃO:

Este documento descreve os requisitos do Sistema de Gerenciamento de Tarefas, abordando funcionalidades principais, requisitos técnicos e mapeamento com a implementação em código. O sistema foi desenvolvido utilizando conceitos de Programação Funcional em Python.

2. REQUISITOS:

2.1 Requisitos Funcionais:

1. **Criar tarefas:** O sistema deve permitir a criação de tarefas com descrição, categoria e status.
 - **Implementação:** Função *adicionar_tarefa*
2. **Listar tarefas:** O sistema deve exibir a lista de tarefas cadastradas.
 - **Implementação:** Função *listar_tarefas*
3. **Filtrar tarefas por status ou categoria:** O sistema deve permitir filtragem das tarefas conforme o status ou a categoria.
 - **Implementação:** Função *filtrar_tarefas*
4. **Remover tarefas concluídas:** O sistema deve permitir remover tarefas que foram marcadas como concluídas.
 - **Implementação:** Função *remover_concluidas*

2.2 Requisitos Não Funcionais:

1. **Uso de Programação Funcional:** O código deve aplicar conceitos de programação funcional.
 - **Implementação:** Uso de *lambda*, *list comprehension*, *closure* e *função de alta ordem*.
2. **Persistência Temporária:** O sistema deve armazenar as tarefas em memória enquanto estiver em execução.
 - **Implementação:** Estrutura de lista em Python.
3. **Código Modular e Testável:** O código deve ser modularizado e testável com *unittest*.
 - **Implementação:** Testes unitários em *test_tarefas.py*.

3. Mapeamento de Código e Conceitos de Programação Funcional:

Conceito:	Implementação:
Função Lambda:	Utilizada na função <i>filtrar_tarefas</i> para aplicar a condição de filtragem.
List Comprehension:	Usada na função <i>listar_tarefas</i> para formatar a saída das tarefas.
Closure:	Implementada na função <i>gerador_id</i> , que retorna uma função interna responsável por gerar identificadores únicos.
Função de Alta Ordem:	A função <i>filtrar_tarefas</i> recebe como argumento uma função de filtragem (lambda), permitindo personalizar os critérios de seleção de tarefas.

4. Testes Implementados:

O sistema conta com testes unitários cobrindo as seguintes funcionalidades:

- Criar tarefas (***test_criar_tarefa***)
- Listar tarefas (***test_listar_tarefas***)
- Filtrar tarefas (***test_filtrar_tarefas***)
- Remover tarefas concluídas (***test_remover_tarefas_concluidas***)

Os testes foram implementados no arquivo ***test_tarefas.py*** utilizando ***unittest***.

5. Repositório no GitHub:

O código-fonte e este documento estão disponíveis no repositório:

<https://github.com/Bruna-Queiroz/Sistema-de-Gerenciamento-de-Tarefas>

6. Uso de Chatbot (ChatGPT):

Durante a realização desta atividade, utilizei o ChatGPT para obter suporte nos seguintes aspectos:

1. Correção e depuração de código:

- **Pergunta:** Meu teste unitário está falhando ao remover uma tarefa concluída. O que pode estar errado?
- **Resposta do ChatGPT:** O erro ocorre porque a lista de tarefas ainda contém a tarefa concluída. Verifique se a função de remoção está modificando a lista original ou apenas retornando uma nova lista sem a tarefa.

2. Escrita do documento de requisitos:

- **Pergunta:** Como posso estruturar os requisitos funcionais e não funcionais de forma clara?
- **Resposta do ChatGPT:** Você pode separar os requisitos em duas seções. Os requisitos funcionais descrevem o que o sistema deve fazer (ex: Criar, listar e filtrar tarefas). Já os requisitos não funcionais abordam aspectos como desempenho, segurança e usabilidade.

3. Explicação de conceitos de Programação Funcional:

- **Pergunta:** Como garantir que minha implementação contenha uma closure?
- **Resposta do ChatGPT:** Uma closure ocorre quando uma função interna acessa variáveis da função externa. Você pode definir uma função que cria e retorna outra função, como um contador que mantém um estado interno.

7. Conclusão:

O presente documento apresenta o desenvolvimento do Sistema de Gerenciamento de Tarefas, abordando sua implementação com base em Programação Funcional. O código foi testado e validado com **unittest**, garantindo que todas as funcionalidades operam conforme esperado. O repositório GitHub contém todo o material necessário para avaliação.