INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE PERNAMBUCO

CAMPUS JABOATÃO DOS GUARARAPES

Curso: Análise e Desenvolvimento de Sistemas

Disciplina: Estrutura de Dados

PROJETO FINAL – GERENCIADOR DE TAREFAS EM PYTHON: UMA ABORDAGEM COM MIN-HEAP E LISTA ENCADEADA

Aluno: Hérik Thiury Silva de Lima

Professora: Viviane Cristina Oliveira Aureliano

Jaboatão dos Guararapes – PE

16 de Julho de 2025

# Sumário

1. Introdução

2. Objetivo Geral

3. Requisitos do Projeto

4. Fundamentação Teórica: Estruturas de Dados

4.1. Nó (Node)

4.2. Lista Encadeada Não Ordenada (UnorderedList)

4.3. Min-Heap (Heap Mínimo)

5. Classe de Entidade: Tarefa

6. Arquitetura do Sistema

7. Funcionalidades Implementadas

8. Demonstração da Aplicação (Modo Console)

9. Desafios Enfrentados e Soluções

10. Próximos Passos e Aprimoramentos

11. Conclusão

12. Referências

## 1. Introdução

Este documento apresenta o "Gerenciador de Tarefas em Python", um projeto desenvolvido como parte da disciplina de Estrutura de Dados. O objetivo principal é aplicar e demonstrar o funcionamento de estruturas de dados essenciais – o Min-Heap e a Lista Encadeada – na resolução de um problema prático e cotidiano: a organização e priorização de tarefas.  
O sistema, atualmente em modo console, permite aos usuários adicionar tarefas com diferentes níveis de prioridade, visualizar a próxima tarefa a ser realizada, marcar tarefas como concluídas e acessar um histórico das tarefas finalizadas.

## 2. Objetivo Geral

Desenvolver uma aplicação funcional que utilize, de forma significativa, as estruturas de dados estudadas para resolver um problema real de gerenciamento de tarefas. O sistema deve priorizar tarefas com base em sua importância e manter um histórico eficiente das atividades concluídas.

## 3. Requisitos do Projeto

3.1. Aspectos Técnicos  
- Estruturas utilizadas: Min-Heap e Lista Encadeada Não Ordenada.  
- Interface em modo console.  
- Código modularizado e documentado.

3.2. Documentação Obrigatória  
Este relatório cumpre os requisitos de documentação exigidos, contendo:  
- Justificativa do problema  
- Explicação das estruturas utilizadas  
- Registro dos desafios enfrentados e soluções aplicadas

## 4. Fundamentação Teórica: Estruturas de Dados

### 4.1. Nó (Node)  
Um Node é o bloco fundamental de muitas estruturas de dados lineares, como listas encadeadas. Ele contém os dados e uma referência para o próximo nó na sequência.

```python  
class Node:  
 def \_\_init\_\_(self, initdata):  
 self.data = initdata  
 self.next = None  
```

### 4.2. Lista Encadeada Não Ordenada (UnorderedList)  
Lista composta por nós ligados entre si, sem uma ordem específica.

```python  
class UnorderedList:  
 def \_\_init\_\_(self):  
 self.head = None  
 # Métodos: isEmpty, add, size, search, remove  
```

### 4.3. Min-Heap (Heap Mínimo)  
Estrutura de dados onde o menor valor está sempre no topo.

```python  
class MinHeap:  
 def \_\_init\_\_(self):  
 self.heap\_list = [0]  
 self.current\_size = 0  
```

## 5. Classe de Entidade: Tarefa

Classe responsável por encapsular os dados de cada tarefa, como descrição e prioridade.

```python  
class Tarefa:  
 def \_\_init\_\_(self, descricao, prioridade):  
 self.descricao = descricao  
 self.prioridade = prioridade

def \_\_lt\_\_(self, other):  
 return self.prioridade < other.prioridade

def \_\_eq\_\_(self, other):  
 return self.descricao == other.descricao and self.prioridade == other.prioridade

def \_\_str\_\_(self):  
 return f"Tarefa: '{self.descricao}' (Prioridade: {self.prioridade})"  
```

## 6. Arquitetura do Sistema

- estruturas\_dados.py: Contém as classes Node, UnorderedList, MinHeap e Tarefa.  
- app.py: Implementa a lógica principal do programa, com menus e manipulação de tarefas.

## 7. Funcionalidades Implementadas

- Adicionar nova tarefa  
- Visualizar tarefa prioritária  
- Concluir tarefa atual  
- Ver histórico de tarefas finalizadas

## 8. Demonstração da Aplicação (Modo Console)

(Inserir capturas de tela no Word após colar o texto)

- 8.1 Menu Principal   
- 8.2 Inserção de Tarefas   
- 8.3 Conclusão de Tarefa   
- 8.4 Histórico

## 9. Desafios Enfrentados e Soluções

- Adaptação de estruturas para objetos personalizados (`Tarefa`)  
- Validação de entrada usando try/except  
- Modularização do código em múltiplos arquivos

## 10. Próximos Passos e Aprimoramentos

- Persistência de dados em JSON ou CSV  
- Edição e busca de tarefas  
- Suporte a ordenações alternativas

## 11. Conclusão

O projeto evidenciou a importância de estruturas de dados como Min-Heap e Lista Encadeada para problemas do cotidiano, como o gerenciamento de tarefas. Além de fortalecer o domínio técnico, reforçou práticas de validação, organização de código e abstração.

## 12. Referências

- Material da disciplina de Estrutura de Dados – IFPE   
- Python Official Docs: https://docs.python.org/3/   
- Estruturas de Dados com Python – Curso IFPE