



Universidade do Estado da Bahia  
Departamento de Ciências Exatas e da Terra  
Curso de Bacharelado em Engenharia de Produção Civil

Discentes: Guilherme Souza e Sara Costa

Gerenciador de tarefas para engenharia civil

Salvador  
2024

# Sumário

Lista de Figuras e Tabelas	1
1 Introdução	2
2 Referencial Teórico	2
3 Proposta do Programa	2
4 Script preliminar em python	6
5 Conclusões parciais	9
Referências	10

## Lista de Figuras

1	Exemplificação do paradigma de orientado a objetos . . . . .	2
2	Menu com as opções que podem ser executadas no sistema . . . . .	4
3	Área para o usuário adicionar nova tarefa . . . . .	5
4	Área para o usuário remover uma tarefa cadastrada . . . . .	5
5	Área para o usuário visualizar as tarefas cadastradas . . . . .	5

## Lista de Tabelas

1	Exemplo de tabela contendo tarefas . . . . .	3
---	--	---

# 1 Introdução

O gerenciamento de projetos é uma área muito importante na engenharia civil, pois permite que seja possível o controle total sobre o andamento da obra, sendo possível saber todos gastos com material e mão de obra, e conseguir determinar com precisão se o prazo está sendo cumprido. Sendo assim, este projeto teve como objetivo criar um aplicativo que possibilitasse o gerenciamento básico de uma obra.

## 2 Referencial Teórico

Os principais conceitos utilizados foram o uso de programação orientada a objetos e a biblioteca pandas. Programação orientada a objetos é um paradigma que busca criar sistemas como se fossem objetos do mundo real. Um objeto possui atributos que são as características desse objeto, e também métodos que são ações que esse objeto pode executar, podendo modificar através dessas ações seus atributos e interagir com outros objetos. O que define cada objeto são as classes, que moldam quais serão os atributos e métodos, funcionando desta forma como uma "forma" para o objeto [1].



Figura 1: Exemplificação do paradigma de orientado a objetos

O pandas é uma biblioteca do python muito utilizada na ciência de dados para tratamento e análise de dados. No contexto deste projeto, essa biblioteca foi utilizada para fazer o gerenciamento de tabelas, pois a classe DataFrame, presente no pandas, permite a criação e manipulação de tabelas [2].

## 3 Proposta do Programa

Este projeto teve como objetivo criar um programa em python que fosse capaz gerenciar tarefas básicas de uma obra da construção civil, buscando proporcionar uma maior organização nas tarefas que deverão ser executadas. Para isso, o programa cadastra os dados de uma tarefa, como o nome da tarefa, membros da equipe que irão executar a tarefa, os equipamentos necessários, e os materiais que serão utilizados, e esses dados são salvos em uma tabela excel. O programa também permite visualizar e remover as tarefas, sendo assim tendo ações simples como adicionar, remover e visualizar tarefas.

	nomeTarefa	equipe	equipamentos	materiais
0	Contruir parede	Joao, Maria, Jose	Martelo, Furadeira	areia, cimento
1	Construir ponte	Joao, Maria	Martelo	Madeira, Aco, Prego

Tabela 1: Exemplo de tabela contendo tarefas

O programa utilizou de 4 bibliotecas ao todo sendo elas pandas, os, platform e sys. O pandas, como já foi explicado anteriormente, foi utilizado para a criação e manipulação dos dados. A biblioteca os foi utilizada por conta do seu método "system" que permite executar comandos no terminal. A biblioteca platform foi utilizada para verificar o sistema operacional e garantir compatibilidade. A biblioteca sys foi utilizada por conta do seu método "exit", utilizado para finalizar o programa.

```

1 import pandas as pd # utilizado para gerenciar os dados e tabelas
2 from os import system # utilizado para limpar o terminal com os
  metodos system("cls") e system("clear")
3 import platform # utilizado para verificar o sistema operacional e
  deste modo utilizar o comando respectivo do sistema para limpar o
  terminal utilizando a funcao system do modulo os
4 import sys # Utilizado para fechar o programa
5

```

Listing 1: Bibliotecas utilizadas

Para o gerenciamento das tarefas foi criado a classe TaskManager. O método construtor da classe verifica se já existe um arquivo "tasks\_table.xlsx", pois é esse arquivo onde as tarefas são salvas, e carrega os dados para a variável privada "\_\_dfTask" que é uma instância da classe DataFrame do pandas. Se não houver o arquivo "tasks\_table.xlsx", então é criada a variável privada "\_\_dfTask", contendo uma instância da classe DataFrame, sendo passado como parâmetro no método construtor da classe um dicionário contendo as colunas necessárias "nomeTarefa", "equipe", "equipamentos" e "materiais". A classe possui um método "updateTable" que utiliza o método "to\_excel" do pandas para atualizar os dados da tabela excel com os dados presentes em "\_\_dfTask". A classe também tem o método "addTask" que adiciona novas tarefas a "\_\_dfTask" e chama a função "updateTable" para atualizar a tabela. Possui também o método "removeTask" que remove uma tarefa do DataFrame "\_\_dfTask" e atualiza a tabela com o método "updateTable". E por fim, possui o método "getTableTask" que retorna o DataFrame "\_\_dfTask".

```

1 # Classe utilizada para gerenciar as tarefas
2 class TaskManager:
3     # funcao construtora da classe que verifica se ja existe uma tabela
    com dados, se nao cria uma tabela para salvar os dados
4     def __init__(self):
5         try:
6             self.__dfTask = pd.read_excel("tasks_table.xlsx")
7             print("Lendo dados...")
8         except FileNotFoundError:
9             print("Criando tabela...")
10            self.__dfTask = pd.DataFrame({
11                "nomeTarefa": [],
12                "equipe": [],
13                "equipamentos": [],

```

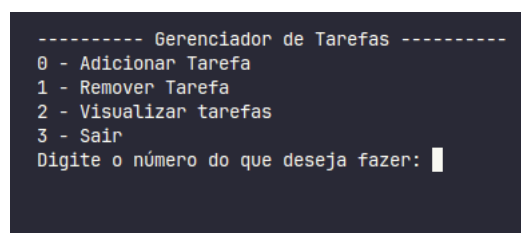
```

14         "materiais": [],
15     })
16     self.updateTable()
17
18     # Atualiza a tabela(excel) com os dados presentes na tabela __dfTask(
19     # DataFrame do pandas)
20     def updateTable(self):
21         self.__dfTask.to_excel("tasks_table.xlsx", index=False)
22         print("Atualizando tabela...")
23
24     # Adiciona uma nova tarefa a tabela
25     def addTask(self, nomeTarefa: str, membrosEquipe: list, equipamentos:
26     list, materiais: list):
27         newLine = {
28             "nomeTarefa": nomeTarefa,
29             "equipe": ', '.join(membrosEquipe),
30             "equipamentos": ', '.join(equipamentos),
31             "materiais": ', '.join(materiais)
32         }
33         self.__dfTask = pd.concat([self.__dfTask, pd.DataFrame([newLine])
34         ], ignore_index=True)
35         self.updateTable()
36
37     # Remove uma tarefa da tabela
38     def removeTask(self, index):
39         try:
40             self.__dfTask.drop(index, inplace=True)
41             self.__dfTask.reset_index(drop=True, inplace=True)
42             self.updateTable()
43             print('Removido com sucesso!')
44         except KeyError:
45             print('Index nao encontrado!')
46
47     # Retorna a tabela __dfTask
48     def getTableTasks(self):
49         return self.__dfTask

```

Listing 2: Classe TaskManager

O programa fornece uma interface simples (2), possibilitando o usuário realizar as seguintes ações: adicionar tarefa, remover tarefa, visualizar tarefas e sair.



```

----- Gerenciador de Tarefas -----
0 - Adicionar Tarefa
1 - Remover Tarefa
2 - Visualizar tarefas
3 - Sair
Digite o número do que deseja fazer: █

```

Figura 2: Menu com as opções que podem ser executadas no sistema

O menu é feito através da função "main", que limpa o terminal através da função "cleanTerminal" e imprime as opções que podem ser escolhidas pelo usuário, permitindo que ele escolha entre as opções de 0 a 3, e a escolha do usuário é armazenada na variável "numMenu", sendo esta utilizada numa logica para verificar qual opção foi escolhida pelo o usuário e o levando para área escolhida. Se o usuário digitar 0, a opção escolhida será

de "adicionar tarefa", então a função "optionAdd" será chamada, e permitirá que o usuário possa adicionar uma nova tarefa (3).

```
----- Adicionar Tarefa -----
Digite o nome da tarefa: Construir Parede
Digite o nome do membro(Pressione Enter duas vezes quando terminar): Joao
Digite o nome do membro(Pressione Enter duas vezes quando terminar): Maria
Digite o nome do membro(Pressione Enter duas vezes quando terminar): Jose
Digite o nome do membro(Pressione Enter duas vezes quando terminar):
Digite o nome do equipamento(Pressione Enter duas vezes quando terminar): Furadeira
Digite o nome do equipamento(Pressione Enter duas vezes quando terminar): Martelo
Digite o nome do equipamento(Pressione Enter duas vezes quando terminar):
Digite o nome do material(Pressione Enter duas vezes quando terminar): Areia
Digite o nome do material(Pressione Enter duas vezes quando terminar): Cimento
Digite o nome do material(Pressione Enter duas vezes quando terminar):
Atualizando tabela...
Tarefa adicionada com sucesso!
(Pressione Enter para voltar ao menu)
```

Figura 3: Área para o usuário adicionar nova tarefa

Se o usuário digitar 1, a opção escolhida será de "remover tarefa", então a função "optionRemove" será chamada, e permitirá que o usuário possa remover uma das tarefas cadastradas (4).

```
----- Remover Tarefa -----
Indice: 0   Tarefa Construir parede
Indice: 1   Tarefa Construir ponte
Indice: 2   Tarefa Construir Parede
Digite o índice da tarefa que sera removida(Deixe vazio para cancelar): 0
Atualizando tabela...
Removido com sucesso!
(Pressione Enter para voltar ao menu)
```

Figura 4: Área para o usuário remover uma tarefa cadastrada

Se o usuário digitar 2, a opção escolhida será de "visualizar tarefas", então a função "optionView" será chamada, e mostrara ao usuário as tarefas cadastradas no sistema (5).

```
----- Visualizar Tarefas -----
1 - Nome Tarefa: Construir ponte
   Membros: Joao, Maria
   Equipamentos: Martelo
   Material: Madeira, Aco, Pregos
2 - Nome Tarefa: Construir Parede
   Membros: Joao, Maria, Jose
   Equipamentos: Furadeira, Martelo
   Material: Areia, Cimento
(Pressione Enter para voltar ao menu)
```

Figura 5: Área para o usuário visualizar as tarefas cadastradas

Se o usuário digitar 3, o programa executara o método "exit" da biblioteca sys, encerrando o programa. Esta função é necessária pois a função "main" está dentro de uma estrutura de repetição while que possui a codição "True", portanto sendo executada infinitamente e fazendo necessário o uso da função "sys.exit()" para finalização do programa.

## 4 Script preliminar em python

```
1 #Alunos
2 #Guilherme Souza Lopes - 072320015
3 #Sara Stephanie Costa - 072320039
4
5 # !!! Necessario instalar a biblioteca pandas e openpyxl(utilizada pelo
6 # pandas para criar tabelas excel) !!!
7
8 import pandas as pd # utilizado para gerenciar os dados e tabelas
9 from os import system # utilizado para limpar o terminal com os metodos
10 system("cls") e system("clear")
11
12 import platform # utilizado para verificar o sistema operacional e deste
13 # modo utilizar o comando respectivo do sistema para limpar o terminal
14 # utilizando a funcao system do modulo os
15 import sys # Utilizado para fechar o programa
16
17 # O seguinte programa e um gerenciador de tarefas que pode adicionar,
18 # remover e visualizar as tarefas, e salva os dados em uma tabela excel
19
20 # Classe utilizada para gerenciar as tarefas
21 class TaskManager:
22     # funcao construtora da classe que verifica se ja existe uma tabela
23     # com dados, se nao cria uma tabela para salvar os dados
24     def __init__(self):
25         try:
26             self.__dfTask = pd.read_excel("tasks_table.xlsx")
27             print("Lendo dados...")
28         except FileNotFoundError:
29             print("Criando tabela...")
30             self.__dfTask = pd.DataFrame({
31                 "nomeTarefa": [],
32                 "equipe": [],
33                 "equipamentos": [],
34                 "materiais": [],
35             })
36             self.updateTable()
37
38     # Atualiza a tabela(excel) com os dados presentes na tabela __dfTask(
39     # DataFrame do pandas)
40     def updateTable(self):
41         self.__dfTask.to_excel("tasks_table.xlsx", index=False)
42         print("Atualizando tabela...")
43
44     # Adiciona uma nova tarefa a tabela
45     def addTask(self, nomeTarefa: str, membrosEquipe: list, equipamentos:
46     list, materiais: list):
47         newLine = {
48             "nomeTarefa": nomeTarefa,
49             "equipe": ', '.join(membrosEquipe),
50             "equipamentos": ', '.join(equipamentos),
51             "materiais": ', '.join(materiais)
52         }
53         self.__dfTask = pd.concat([self.__dfTask, pd.DataFrame([newLine])
54 ], ignore_index=True)
55         self.updateTable()
56
57     # Remove uma tarefa da tabela
58     def removeTask(self, index):
59         try:
```



```

50         self.__dfTask.drop(index, inplace=True)
51         self.__dfTask.reset_index(drop=True, inplace=True)
52         self.updateTable()
53         print('Removido com sucesso!')
54     except KeyError:
55         print('Index nao encontrado!')
56
57     # Retorna a tabela __dfTask
58     def getTableTasks(self):
59         return self.__dfTask
60
61 # Instacia de um objeto TaskManager
62 taskManager = TaskManager()
63
64 # Metodo que imprimi o menu principal da aplicacao e permiti escolher
65 # outras acoes no sistema
66 def main():
67     cleanTerminal()
68
69     print('-'*10, 'Gerenciador de Tarefas', '-'*10)
70     print('0 - Adicionar Tarefa')
71     print("1 - Remover Tarefa")
72     print('2 - Visualizar tarefas')
73     print('3 - Sair')
74
75     numMenu = int(input('Digite o numero do que deseja fazer: '))
76
77     if numMenu == 0:
78         optionAdd()
79     elif numMenu == 1:
80         optionRemove()
81     elif numMenu == 2:
82         print('visulizar tarefas...')
83         optionView()
84     elif numMenu == 3:
85         sys.exit()
86
87 # Imprimi no terminal a area de adicionar novas tarefas ao objeto
88 # taskManager
89 def optionAdd():
90     cleanTerminal()
91     print('-'*10, 'Adicionar Tarefa', '-'*10)
92
93     nomeTarefa = input('Digite o nome da tarefa: ')
94     membrosEquipe = []
95     equipamentos = []
96     materiais = []
97
98     while True:
99         nomeMembro = input("Digite o nome do membro(Pressione Enter duas
100 vezes quando terminar): ").strip()
101
102         if nomeMembro != "":
103             membrosEquipe.append(nomeMembro)
104         else:
105             break
106
107     while True:
108         equipamento = input("Digite o nome do equipamento(Pressione Enter
109 duas vezes quando terminar): ").strip()

```

```

106         if equipamento != "":
107             equipamentos.append(equipamento)
108         else:
109             break
110
111
112     while True:
113         material = input("Digite o nome do material(Pressione Enter duas
vezes quando terminar): ").strip()
114
115         if material != "":
116             materiais.append(material)
117         else:
118             break
119
120     taskManager.addTask(nomeTarefa, membrosEquipe, equipamentos, materiais
)
121
122     print('Tarefa adicionada com sucesso!')
123     input('(Pressione Enter para voltar ao menu)')
124
125 # Imprime a area para remover tarefas cadastradas no sistema
126 def optionRemove():
127     cleanTerminal()
128     print('-'*10, 'Remover Tarefa', '-'*10)
129     if len(taskManager.getTableTasks()) != 0:
130         i = 0
131         while i < len(taskManager.getTableTasks()):
132             print('Indice:', i, ' ', 'Tarefa', taskManager.getTableTasks().
loc[i, 'nomeTarefa'])
133             i += 1
134             indexLine = input("Digite o indice da tarefa que sera removida(
Deixe vazio para cancelar): ").strip()
135
136             if indexLine != "":
137                 taskManager.removeTask(int(indexLine))
138             else:
139                 print('Remocao cancelada!')
140         else:
141             print("Sem tarefas cadastradas!")
142         input('(Pressione Enter para voltar ao menu)')
143
144 #Imprimi as tarefas cadastradas no sistema
145 def optionView():
146     cleanTerminal()
147
148     print('-'*10, 'Visualizar Tarefas', '-'*10)
149
150     tb = taskManager.getTableTasks()
151     i = 0
152
153     if len(tb) > 0:
154         while i < len(tb):
155             print((i + 1), '- Nome Tarefa:', tb.loc[i, "nomeTarefa"])
156             print('      Membros:', tb.loc[i, 'equipe'])
157             print('      Equipamentos:', tb.loc[i, 'equipamentos'])
158             print('      Material:', tb.loc[i, 'materiais'])
159             i += 1
160         else:
161             print("Nenhuma tarefa cadastrada!")

```

```

162     input('(Pressione Enter para voltar ao menu)')
163
164 # Limpa o terminal para manter as informacoes organizadas
165 def cleanTerminal():
166     operatingSystem = platform.system()
167     if operatingSystem == 'Windows':
168         system('cls')
169     else:
170         system('clear')
171
172 # Loop que repetir de forma infinita chamando o metodo main(). Este loop
    so acabara se o metodo sys.exit() for chamado dentro da logica da
    funcao main()
173 while True:
174     main()

```

Listing 3:Codigo Python

## 5 Conclusões parciais

No inicio do projeto, estabelecemos que o nosso objetivo seria fazer uma aplicação que fosse capaz de organizar e armazenar tarefas de uma obra de engenharia civil. Nesse sentido, conseguimos criar um programa com uma interface simples e que consegue adicionar, remover e visualizar tarefas.

## Referências

- [1] 9. classes — documentação python 3.12.3. <https://docs.python.org/pt-br/3/tutorial/classes.html#classes>, 2024. Accessed: 2024-04-13.
- [2] User guide — pandas 2.2.2 documentation. [https://pandas.pydata.org/docs/user\\_guide/index.html](https://pandas.pydata.org/docs/user_guide/index.html), 2024. Accessed: 2024-04-13.