

Controle de Almoxarifado na Construção Civil

Jailmária Marques e Andreza Pereira.

Computação aplicada à engenharia.

1.Introdução

- ▶ O controle de almoxarifado desempenha um papel fundamental na eficiência e sucesso dos projetos na construção civil. No contexto da construção civil, o controle de almoxarifado refere-se ao processo de rastrear, organizar e controlar os materiais utilizados em uma obra

2.Objetivos

- ▶ Objetivo Geral: analisar práticas de controle de almoxarifado e desenvolver um modelo referência com base nisso.
- ▶ Objetivos específicos: controle de entrada e saída de materiais, gestão de estoque e reposição facilitar a rastreabilidade e controle de validade, otimizar o tempo de obra.

3. Benefícios do Controle de Almoxarifado

- ▶ Redução de desperdícios
- ▶ Melhor aproveitamento dos recursos
- ▶ Evita paralisações de obra por falta de materiais
- ▶ Planejamento mais eficiente
- ▶ Redução de custos e aumento da lucratividade

4. Metodologia de pesquisa

- ▶ A pesquisa foi realizada por meio de uma revisão bibliográfica que abordou tópicos como:
- ▶ Conceitos chave em relação à administração de almoxarifados;
- ▶ Práticas mais adequadas de organização e planejamento;
- ▶ Métodos recomendados de controle e estoque.

5. Funcionamento do código Python

🔗 CONTROLE DE ALMOXARIFADO - 2ª ETAPA.py > ...

```
1  ∨ import csv
2  import datetime
3
4  ∨ class Catalogo:
5  ∨     def __init__(self):
6  |         self.arquivo_catalogo = "catalogo.csv"
7  |         self.arquivo_historico = "historico.csv"
8  |
9  ∨     def adicionar_item(self):
10 |         codigo = input("Digite o código do item: ")
11 |         descricao = input("Digite a descrição do item: ")
12 |         quantidade = input("Digite a quantidade disponível: ")
13 |
14 |         with open(self.arquivo_catalogo, "a", newline="") as arquivo:
15 |             writer = csv.writer(arquivo)
16 |             writer.writerow([codigo, descricao, quantidade])
17 |
18 |         print("Item adicionado com sucesso!")
19 |
20 ∨     def pesquisar_item(self):
21 |         codigo = input("Digite o código do item: ")
22 |
23 |         try:
24 |             with open(self.arquivo_catalogo, "r") as arquivo:
25 |                 reader = csv.reader(arquivo)
26 |                 for linha in reader:
```



```
49
50         with open(self.arquivo_catalogo, "w", newline="") as arquivo:
51             writer = csv.writer(arquivo)
52             writer.writerows(linhas_atualizadas)
53
54         print("Quantidade atualizada com sucesso!")
55     except FileNotFoundError:
56         print("Arquivo do catálogo não encontrado.")
57     except Exception as e:
58         print("Ocorreu um erro:", e)
59
60     def exibir_catalogo(self):
61         try:
62             with open(self.arquivo_catalogo, "r") as arquivo:
63                 reader = csv.reader(arquivo)
64                 for linha in reader:
65                     print("Código:", linha[0])
66                     print("Descrição:", linha[1])
67                     print("Quantidade disponível:", linha[2])
68                     print()
69         except FileNotFoundError:
70             print("Arquivo do catálogo não encontrado.")
71         except Exception as e:
72             print("Ocorreu um erro:". e)
```



```
72         print("Ocorreu um erro:", e)
73
74     def exibir_historico(self):
75         try:
76             with open(self.arquivo_historico, "r") as arquivo:
77                 reader = csv.reader(arquivo)
78                 for linha in reader:
79                     data = datetime.datetime.strptime(linha[0], "%Y-%m-%d %H:%M:%S")
80                     descricao = linha[1]
81                     quantidade = linha[2]
82                     print("Data e hora:", data)
83                     print("Descrição:", descricao)
84                     print("Quantidade:", quantidade)
85                     print()
86             except FileNotFoundError:
87                 print("Arquivo do histórico não encontrado.")
88             except Exception as e:
89                 print("Ocorreu um erro:", e)
90
91     def remover_item(self):
92         try:
93             codigo =
94             input("Digite o código do item a ser removido: ")
95             item encontrado =
```

```
95         item_encontrado =
96         linhas_atualizadas = []
97         linhas_a
98         with open(self.arquivo_catalogo, "r") as arquivo:
99             reader = csv.reader(arquivo)
100             read
101             for linha in reader:
102                 if linha[0] == codigo:
103                     item_encontrado =
104
105             else:
106                 linhas_atualizadas.append(linha)
107                 linhas_atualizad
108         if item_encontrado:
109             with open(self.arquivo_catalogo, "w", newline="") as arquivo:
110                 writer = csv.writer(arquivo)
111                 writer.writerows(linhas_atualizadas)
112                 writer = csv.writer(arquivo)
113                 writer.writerows(linhas_atualizadas)
114                 writer = csv.writer(arquivo)
115                 print("Item removido com sucesso!")
116             else:
117                 print("Item não encontrado.")
118             except FileNotFoundError:
```

```
119
120     def menu(self):
121         while True:
122             print("1. Adicionar item")
123             print("2. Pesquisar item")
124             print("3. Atualizar quantidade")
125             print("4. Exibir catálogo")
126             print("5. Exibir histórico")
127             print("6. Sair")
128
129             opcao = input("Digite a opção desejada: ")
130
131             if opcao == "1":
132                 self.adicionar_item()
133             elif opcao == "2":
134                 self.pesquisar_item()
135             elif opcao == "3":
136                 self.atualizar_quantidade()
137             elif opcao == "4":
138                 self.exibir_catalogo()
139             elif opcao == "5":
140                 self.exibir_historico()
141             elif opcao == "6":
142                 break
```

```
141         elif opcao == "6":
142             break
143         else:
144             print("Opção inválida. Tente novamente.")
145
146     print()
147
148     # Cria uma instância da classe Catalogo
149     catalogo = Catalogo()
150
151     # Executa o programa
152     catalogo.menu()
```

6. REDE DE PETRI

