# Substitua a função carregar\_saidas por esta versão:

def carregar\_saidas():

conexao = sqlite3.connect("dados.db")

terminal\_sql = conexao.cursor()

# Limpa a tabela

for item in tree\_saida.get\_children():

tree\_saida.delete(item)

# Busca as saídas no banco de dados

terminal\_sql.execute("""

SELECT nome\_produto, quantidade, data\_hora

FROM saidas

ORDER BY data\_hora DESC

""")

saidas = terminal\_sql.fetchall()

# Preenche a tabela com os dados reais

for saida in saidas:

tree\_saida.insert('', tk.END, values=saida)

conexao.close()

# Substitua a função adicionar\_item\_saida por esta versão simplificada:

def adicionar\_item\_saida():

global saida\_produtos

nome\_produto = entrada\_nome\_qtde\_saida.get().strip()

qtde\_retirada = entrada\_qtde\_retirada.get().strip()

if not nome\_produto or not qtde\_retirada:

showinfo("Aviso", "Preencha todos os campos!")

return

try:

qtde = int(qtde\_retirada)

if qtde <= 0:

showinfo("Aviso", "A quantidade deve ser maior que zero!")

return

except ValueError:

showinfo("Aviso", "Quantidade inválida!")

return

# Verifica se o produto já está na lista

encontrado = False

for i, item in enumerate(saida\_produtos):

if item['nome'] == nome\_produto:

saida\_produtos[i]['quantidade'] += qtde

encontrado = True

break

if not encontrado:

# Adiciona novo item ao vetor

saida\_produtos.append({

'nome': nome\_produto,

'quantidade': qtde

})

# Atualiza a exibição na interface

atualizar\_lista\_saida()

# Limpa o campo de quantidade

entrada\_qtde\_retirada.delete(0, tk.END)

# Na seção do frame\_relatorio\_saida, substitua a geração de dados aleatórios por:

tree\_saida.grid(row=2, column=0, padx=0, pady=10, columnspan=4)

carregar\_saidas() # Carrega os dados reais do banco