CENTRO UNIVERSITÁRIO INTERNACIONAL UNINTER
ESCOLA SUPERIOR POLITÉCNICA
NOME DO CURSO

ATIVIDADE PRÁTICA ENGENHARIA DE SOFTWARE

WEYDISON DOS SANTOS ANDRADE – 4330561

MANAUS - AMAZONAS
2025

HISTÓRIA DE USUÁRIO: ENTRADA DE PRODUTO NO ESTOQUE
Um funcionário do setor de almoxarifado precisa registrar a entrada de produtos no estoque, para que o sistema reflita corretamente os itens disponíveis após uma nova entrega ou compra. Critérios de Aceitação: • O sistema deve permitir selecionar um produto já cadastrado. • Deve ser possível informar a quantidade recebida e a data da entrada. • O estoque do produto deve ser atualizado automaticamente. Além disso, precisará adicionar neste sistema algumas funcionalidades para ajudar na distribuição de materiais para registrar a saída de produtos do estoque mantendo o controle preciso dos itens utilizados ou entregues. Critérios de Aceitação: • O sistema deve permitir selecionar um produto e informar a quantidade retirada. • Deve validar se há quantidade suficiente no estoque antes de confirmar a saída. • O sistema deve registrar a movimentação com data e responsável.

A partir da **HISTÓRIA DE USUÁRIO** responda as seguintes perguntas:

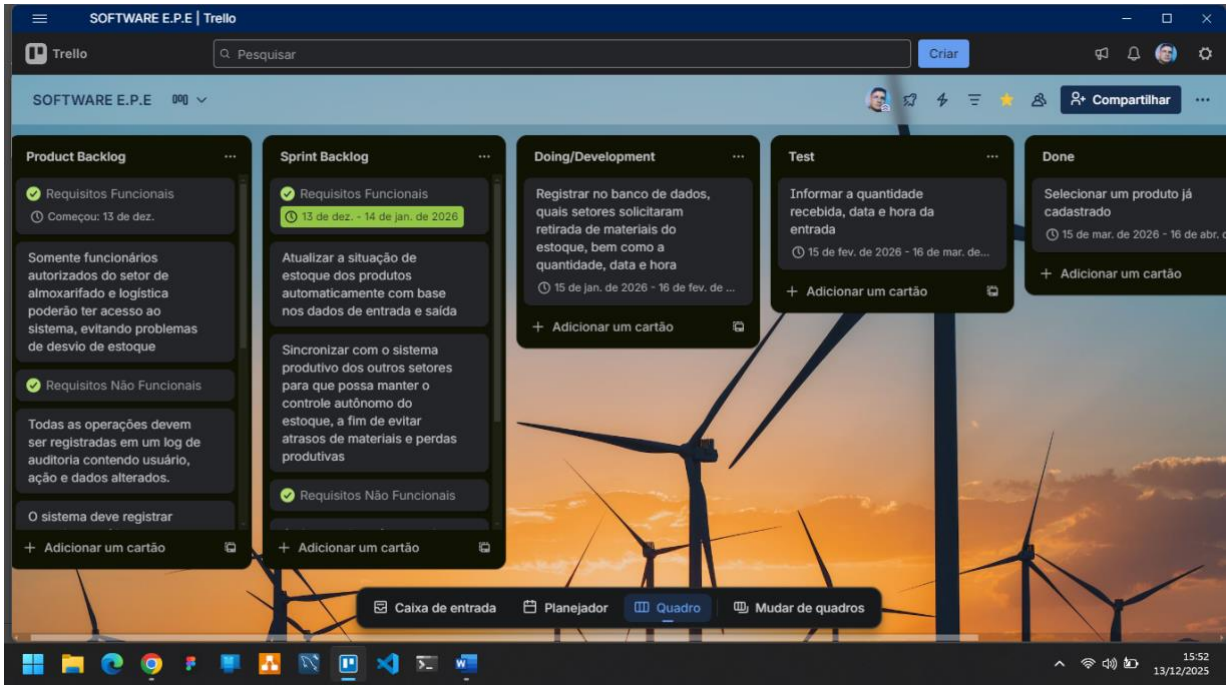
1. Após a leitura da História de Usuário, você deverá preencher as duas tabelas a seguir descrevendo no mínimo 6 requisitos funcionais e 6 requisitos não funcionais do sistema.

	REQUISITO FUNCIONAL
RF01	O sistema deve permitir selecionar um produto já cadastrado
RF02	O sistema deve informar a quantidade recebida, data e hora da entrada.
RF03	O sistema deve atualizar a situação de estoque dos produtos automaticamente com base nos dados de entrada e saída
RF04	O sistema deverá registrar no banco de dados, quais setores solicitaram retirada de materiais do estoque, bem como a quantidade, data e hora

RF05	O sistema deverá estar sincronizado com o sistema produtivo dos outros setores para que possa manter o controle autônomo do estoque, a fim de evitar atrasos de materiais e perdas produtivas
RF06	Somente funcionários autorizados do setor de almoxarifado e logística poderão ter acesso ao sistema, evitando problemas de desvio de estoque

	REQUISITO NÃO FUNCIONAL
RNF01	O sistema deverá estar ativo pelo tempo diário em que a empresa se mantiver em funcionamento, dependendo da sua carga horária diária/semanal, ficando fora do ar apenas nos dias que não forem úteis
RNF02	Todas as operações devem ser registradas em um log de auditoria contendo usuário, ação e dados alterados.
RNF03	O sistema deve registrar entradas e saídas mesmo em caso de falha temporária do banco, armazenando-as localmente até a reconexão.
RNF04	Todas as movimentações devem estar disponíveis para consulta por pelo menos 3 anos
RNF05	Deve suportar o aumento de até 5x na quantidade de itens cadastrados sem perda de desempenho significativo
RNF06	O sistema deve permitir a adição de novas categorias de itens, sem necessidades de alterações estruturais no banco de dados.

2. Desenvolver um quadro para acompanhar o andamento das tarefas do projeto. **Ver a Aula Prática 2 a partir do instante 34 minutos.** O quadro deverá seguir a metodologia ágil Scrum. Ao utilizar ferramentas online para a criação do quadro, deverá inserir o nome e o RU de todos os integrantes do grupo. Certifique-se de que a imagem está legível para correção e não será permitido enviar links!



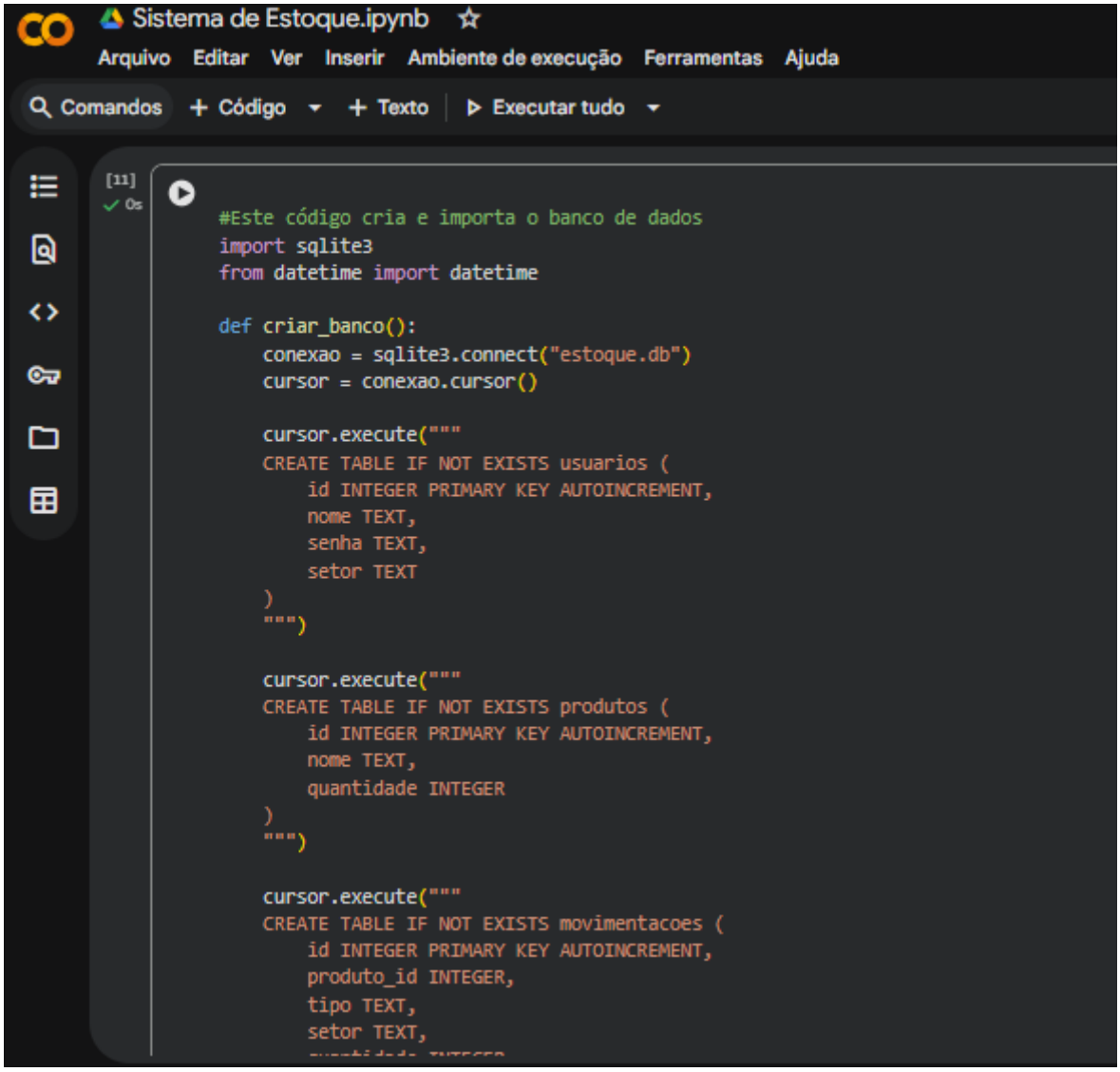
Fonte: Trello - Weydison dos Santos Andrade: 4330561

3. Desenvolver um sistema simples de controle de estoque em **Python**, com funcionalidades de **entrada** e **saída** de produtos, conforme a História de Usuário fornecida.

COLOQUE AQUI A RESPOSTA!!!

```
#Este código cria e importa o banco de dados
import sqlite3
from datetime import datetime


def criar_banco():
    conexao = sqlite3.connect("estoque.db")
    cursor = conexao.cursor()


    cursor.execute("""
    CREATE TABLE IF NOT EXISTS usuarios (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
    nome TEXT,
    senha TEXT,
    setor TEXT
)
""")

cursor.execute("""
CREATE TABLE IF NOT EXISTS produtos (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT,
    quantidade INTEGER
)
""")

cursor.execute("""
CREATE TABLE IF NOT EXISTS movimentacoes (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    produto_id INTEGER,
    tipo TEXT,
    setor TEXT,
    quantidade INTEGER,
    data_hora TEXT
)
""")

conexao.commit()
conexao.close()
```



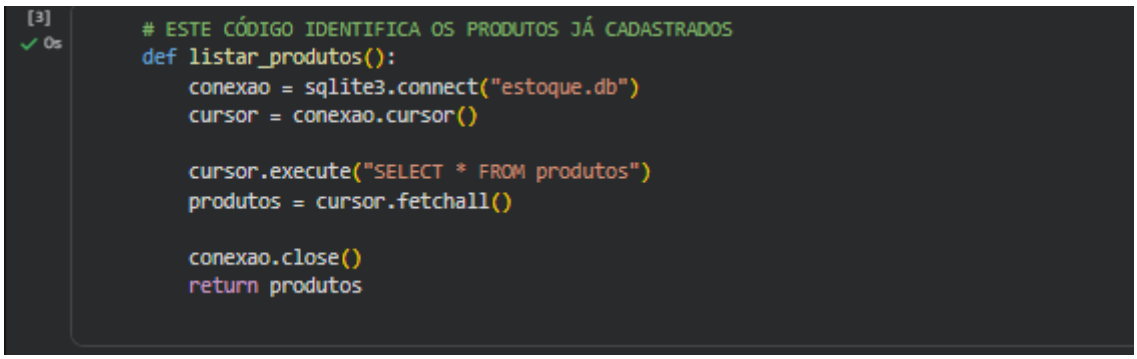
Fonte: Weydison Andrade

```
# ESTE CÓDIGO CRIA A FUNCIONALIDADE DE LOGIN DE FUNCIONÁRIOS AUTORIZADOS

def login():
    nome = input("Usuário: ")
    senha = input("Senha: ")


    conexao = sqlite3.connect("estoque.db")
```

```
cursor = conexao.cursor()

cursor.execute("""
SELECT * FROM usuarios
WHERE nome=? AND senha=?
""", (nome, senha))

usuario = cursor.fetchone()
conexao.close()

if usuario:
    print("✅ Acesso autorizado!")
    return usuario
else:
    print("❌ Acesso negado!")
    return None
```

Fonte: Weydison Andrade

```
# ESTE CÓDIGO IDENTIFICA OS PRODUTOS JÁ CADASTRADOS
def listar_produtos():
    conexao = sqlite3.connect("estoque.db")
    cursor = conexao.cursor()

    cursor.execute("SELECT * FROM produtos")
    produtos = cursor.fetchall()

    conexao.close()
    return produtos
```



Fonte: Weydison Andrade

```
# ESTE CÓDIGO REGISTRA A ENTRADA DE PRODUTOS NO SISTEMA DE ESTOQUE
def entrada_produto():
    produtos = listar_produtos()

    for p in produtos:
        print(f"{p[0]} - {p[1]} (Estoque: {p[2]})")
```

```
produto_id = int(input("Selecione o ID do produto: "))
quantidade = int(input("Quantidade recebida: "))
data_hora = datetime.now().strftime("%d/%m/%Y %H:%M:%S")

conexao = sqlite3.connect("estoque.db")
cursor = conexao.cursor()

cursor.execute("""
UPDATE produtos
SET quantidade = quantidade + ?
WHERE id = ?
""", (quantidade, produto_id))

cursor.execute("""
INSERT INTO movimentacoes
(produto_id, tipo, setor, quantidade, data_hora)
VALUES (?, 'ENTRADA', 'Fornecedor', ?, ?)
""", (produto_id, quantidade, data_hora))

conexao.commit()
conexao.close()

print("📦 Entrada registrada com sucesso!")
```

```
[4]
✓ 0s
# ESTE CÓDIGO REGISTRA A ENTRADA DE PRODUTOS NO SISTEMA DE ESTOQUE
def entrada_produto():
    produtos = listar_produtos()

    for p in produtos:
        print(f"{p[0]} - {p[1]} (Estoque: {p[2]})")

    produto_id = int(input("Selecione o ID do produto: "))
    quantidade = int(input("Quantidade recebida: "))
    data_hora = datetime.now().strftime("%d/%m/%Y %H:%M:%S")

    conexao = sqlite3.connect("estoque.db")
    cursor = conexao.cursor()

    cursor.execute("""
    UPDATE produtos
    SET quantidade = quantidade + ?
    WHERE id = ?
    """, (quantidade, produto_id))

    cursor.execute("""
    INSERT INTO movimentacoes
    (produto_id, tipo, setor, quantidade, data_hora)
    VALUES (?, 'ENTRADA', 'Fornecedor', ?, ?)
    """, (produto_id, quantidade, data_hora))

    conexao.commit()
    conexao.close()

    print("📦 Entrada registrada com sucesso!")
```

Fonte: Weydison Andrade

#REGISTRA A SAÍDA DE PRODUTOS DO SISTEMA POR SETOR

def saida_produto():

produtos = listar_produtos()

for p in produtos:

print(f"{p[0]} - {p[1]} (Estoque: {p[2]})")

produto_id = int(input("Selecione o ID do produto: "))

setor = input("Setor solicitante: ")

quantidade = int(input("Quantidade retirada: "))

data_hora = datetime.now().strftime("%d/%m/%Y %H:%M:%S")

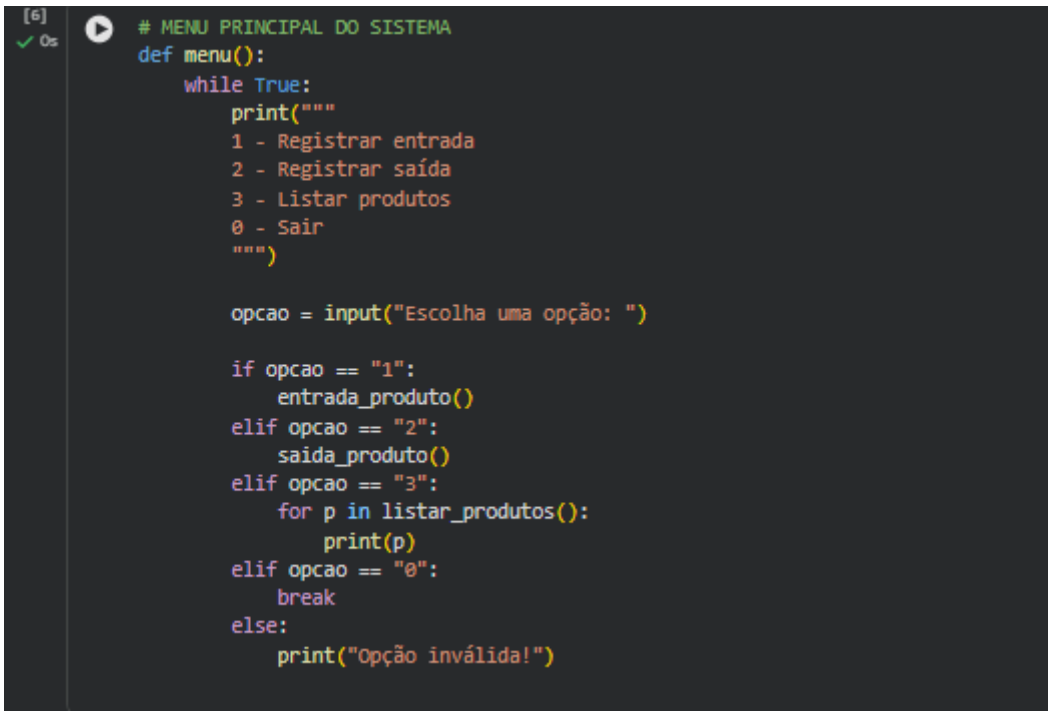
```
conexao = sqlite3.connect("estoque.db")
cursor = conexao.cursor()

cursor.execute("""
UPDATE produtos
SET quantidade = quantidade - ?
WHERE id = ? AND quantidade >= ?
""", (quantidade, produto_id, quantidade))

if cursor.rowcount == 0:
    print("❌ Estoque insuficiente!")
else:
    cursor.execute("""
    INSERT INTO movimentacoes
    (produto_id, tipo, setor, quantidade, data_hora)
    VALUES (?, 'SAÍDA', ?, ?, ?)
    """, (produto_id, setor, quantidade, data_hora))

    conexao.commit()
    print("📤 Saída registrada com sucesso!")

conexao.close()
```

```python
#REGISTRA A SAÍDA DE PRODUTOS DO SISTEMA POR SETOR
def saida_produto():
    produtos = listar_produtos()

    for p in produtos:
        print(f"{p[0]} - {p[1]} (Estoque: {p[2]})")

    produto_id = int(input("Selecione o ID do produto: "))
    setor = input("Setor solicitante: ")
    quantidade = int(input("Quantidade retirada: "))
    data_hora = datetime.now().strftime("%d/%m/%Y %H:%M:%S")

    conexao = sqlite3.connect("estoque.db")
    cursor = conexao.cursor()

    cursor.execute("""
    UPDATE produtos
    SET quantidade = quantidade - ?
    WHERE id = ? AND quantidade >= ?
    """, (quantidade, produto_id, quantidade))

    if cursor.rowcount == 0:
        print("❌ Estoque insuficiente!")
    else:
        cursor.execute("""
        INSERT INTO movimentacoes
        (produto_id, tipo, setor, quantidade, data_hora)
        VALUES (?, 'SAÍDA', ?, ?, ?)
        """, (produto_id, setor, quantidade, data_hora))

        conexao.commit()
        print("📤 Saída registrada com sucesso!")
```

Fonte: Weydison Andrade

```
# MENU PRINCIPAL DO SISTEMA
def menu():
    while True:
        print("""
        1 - Registrar entrada
        2 - Registrar saída
        3 - Listar produtos
        0 - Sair
        """)

        opcao = input("Escolha uma opção: ")
```

```
if opcao == "1":
    entrada_produto()
elif opcao == "2":
    saida_produto()
elif opcao == "3":
    for p in listar_produtos():
        print(p)
elif opcao == "0":
    break
else:
    print("Opção inválida!")
```

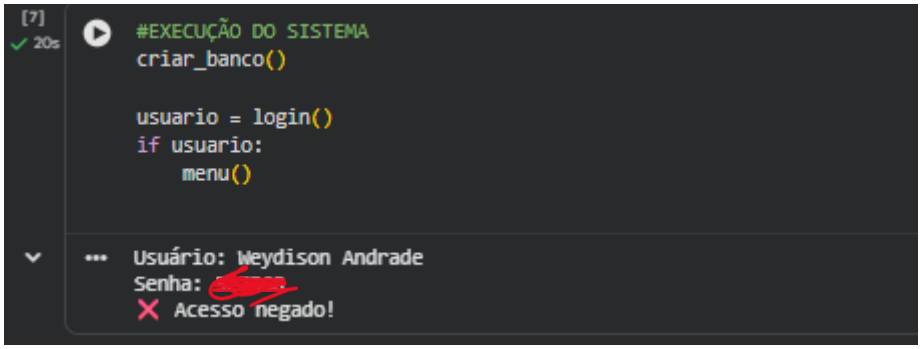


Fonte: Weydison Andrade

#EXECUÇÃO DO SISTEMA

criar_banco()

```
usuario = login()
if usuario:
    menu()
```



Fonte: Weydison Andrade

```
#FUNÇÃO DE CADASTRAR USUÁRIO NO SISTEMA
import sqlite3

def cadastrar_usuario():
    nome = input("Nome do usuário: ")
    senha = input("Senha: ")
    setor = input("Setor (Almoxarifado/Logística): ")

    if setor.lower() not in ["almoxarifado", "logística", "logistica"]:
        print("❌ Setor não autorizado!")
        return

    conexao = sqlite3.connect("estoque.db")
    cursor = conexao.cursor()

    cursor.execute("""
```

```
    INSERT INTO usuarios (nome, senha, setor)
    VALUES (?, ?, ?)
""", (nome, senha, setor))

conexao.commit()
conexao.close()

print("✅ Usuário cadastrado com sucesso!")
```

```
[8]
✓ 0s
#FUNÇÃO DE CADASTRAR USUÁRIO NO SISTEMA
import sqlite3

def cadastrar_usuario():
    nome = input("Nome do usuário: ")
    senha = input("Senha: ")
    setor = input("Setor (Almoxarifado/Logística): ")

    if setor.lower() not in ["almoxarifado", "logística", "logistica"]:
        print("❌ Setor não autorizado!")
        return

    conexao = sqlite3.connect("estoque.db")
    cursor = conexao.cursor()

    cursor.execute("""
        INSERT INTO usuarios (nome, senha, setor)
        VALUES (?, ?, ?)
    """, (nome, senha, setor))

    conexao.commit()
    conexao.close()

    print("✅ Usuário cadastrado com sucesso!")
```

Fonte: Weydison Andrade

```
#FUNÇÃO DE EXCLUIR USUÁRIO DO SISTEMA
def excluir_usuario():
    conexao = sqlite3.connect("estoque.db")
    cursor = conexao.cursor()
```

```
cursor.execute("SELECT id, nome, setor FROM usuarios")
usuarios = cursor.fetchall()

if not usuarios:
    print("⚠️ Nenhum usuário cadastrado.")
    conexao.close()
    return

print("\nUsuários cadastrados:")
for u in usuarios:
    print(f"ID: {u[0]} | Nome: {u[1]} | Setor: {u[2]}")

usuario_id = int(input("\nDigite o ID do usuário a ser excluído: "))

cursor.execute("DELETE FROM usuarios WHERE id = ?", (usuario_id,))

if cursor.rowcount == 0:
    print("❌ Usuário não encontrado.")
else:
    conexao.commit()
    print("🗑️ Usuário excluído com sucesso!")

conexao.close()
```

[9] ✓ 0s

```python
#FUNÇÃO DE EXCLUIR USUÁRIO DO SISTEMA
def excluir_usuario():
    conexao = sqlite3.connect("estoque.db")
    cursor = conexao.cursor()

    cursor.execute("SELECT id, nome, setor FROM usuarios")
    usuarios = cursor.fetchall()

    if not usuarios:
        print("⚠️ Nenhum usuário cadastrado.")
        conexao.close()
        return

    print("\nUsuários cadastrados:")
    for u in usuarios:
        print(f"ID: {u[0]} | Nome: {u[1]} | Setor: {u[2]}")

    usuario_id = int(input("\nDigite o ID do usuário a ser excluído: "))

    cursor.execute("DELETE FROM usuarios WHERE id = ?", (usuario_id,))

    if cursor.rowcount == 0:
        print("❌ Usuário não encontrado.")
    else:
        conexao.commit()
        print("🗑️ Usuário excluído com sucesso!")

    conexao.close()
```