Se quiser, posso:

* Gerar arquivos separados prontos para rodar;
* Ajudar a estruturar os endpoints adicionais como GET /status, POST /pessoas, etc.;
* Incluir autenticação no servidor.

Me avise como deseja seguir!

**Você disse:**

faça isso por favor

**O ChatGPT disse:**

Memória salva atualizada

Ótimo! Vamos transformar seu sistema em um modelo com **servidor Flask (API)** e **cliente Tkinter**.

**✅ Visão geral da separação:**

1. **Servidor Flask (API REST)**: centraliza o banco de dados e fornece rotas para registrar presença, consultar dados, exportar CSV etc.
2. **Cliente Tkinter**: envia requisições para o servidor via HTTP usando requests.

**🔧 Parte 1: Código do servidor Flask**

Salve como servidor.py:

python

CopiarEditar

from flask import Flask, request, jsonify, send\_file

import sqlite3

from datetime import datetime

import csv

app = Flask(\_\_name\_\_)

def conectar\_banco():

return sqlite3.connect('registro\_presenca.db')

@app.before\_first\_request

def criar\_tabelas():

conn = conectar\_banco()

cur = conn.cursor()

cur.execute('''

CREATE TABLE IF NOT EXISTS pessoas (

matricula TEXT PRIMARY KEY,

nome TEXT,

sessao TEXT

)

''')

cur.execute('''

CREATE TABLE IF NOT EXISTS registros (

id INTEGER PRIMARY KEY AUTOINCREMENT,

nome TEXT,

sessao TEXT,

data TEXT,

hora TEXT,

status TEXT

)

''')

conn.commit()

conn.close()

@app.route("/registrar", methods=["POST"])

def registrar():

dados = request.json

matricula = dados.get("matricula")

status = dados.get("status")

conn = conectar\_banco()

cur = conn.cursor()

cur.execute("SELECT nome, sessao FROM pessoas WHERE matricula = ?", (matricula,))

resultado = cur.fetchone()

if resultado:

nome, sessao = resultado

agora = datetime.now()

data = agora.strftime('%d/%m/%Y')

hora = agora.strftime('%H:%M:%S')

cur.execute("INSERT INTO registros (nome, sessao, data, hora, status) VALUES (?, ?, ?, ?, ?)",

(nome, sessao, data, hora, status))

conn.commit()

conn.close()

return jsonify({"mensagem": f"{status} registrada com sucesso!"}), 200

else:

return jsonify({"erro": "Matrícula não encontrada"}), 404

@app.route("/status", methods=["GET"])

def status\_presenca():

conn = conectar\_banco()

cur = conn.cursor()

cur.execute('''

SELECT nome, sessao, status, data, hora FROM registros

WHERE (nome, data, id) IN (

SELECT nome, data, MAX(id) FROM registros GROUP BY nome, data

)

''')

registros = cur.fetchall()

conn.close()

return jsonify(registros)

@app.route("/exportar", methods=["GET"])

def exportar\_csv():

conn = conectar\_banco()

cur = conn.cursor()

cur.execute("SELECT nome, sessao, data, hora, status FROM registros")

dados = cur.fetchall()

caminho = "registros\_exportados.csv"

with open(caminho, "w", newline='', encoding='utf-8') as f:

escritor = csv.writer(f)

escritor.writerow(["Nome", "Sessão", "Data", "Hora", "Status"])

escritor.writerows(dados)

return send\_file(caminho, as\_attachment=True)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host="0.0.0.0", port=5000)

**🖥️ Parte 2: Cliente Tkinter básico**

Salve como cliente.py:

python

CopiarEditar

import requests

from tkinter import \*

from tkinter import messagebox

API\_URL = "http://127.0.0.1:5000"

def registrar\_presenca(status):

matricula = entry\_matricula.get()

if not matricula:

messagebox.showerror("Erro", "Digite a matrícula.")

return

resposta = requests.post(f"{API\_URL}/registrar", json={"matricula": matricula, "status": status})

if resposta.status\_code == 200:

msg = resposta.json().get("mensagem")

messagebox.showinfo("Sucesso", msg)

else:

erro = resposta.json().get("erro", "Erro desconhecido.")

messagebox.showerror("Erro", erro)

entry\_matricula.delete(0, END)

janela = Tk()

janela.title("Cliente - Registro de Presença")

Label(janela, text="Digite sua matrícula:").pack(pady=10)

entry\_matricula = Entry(janela)

entry\_matricula.pack(pady=5)

Button(janela, text="Registrar Entrada", command=lambda: registrar\_presenca("Entrada")).pack(pady=5)

Button(janela, text="Registrar Saída", command=lambda: registrar\_presenca("Saída")).pack(pady=5)

janela.mainloop()

**✅ Como testar**

1. Execute servidor.py (isso cria o banco e sobe o servidor).
2. Cadastre manualmente uma pessoa no banco ou crie uma rota POST /cadastrar.
3. Execute cliente.py em outro terminal ou outro PC (use IP da máquina com o servidor).
4. Teste entradas e saídas.

Se quiser que eu converta também a **parte administrativa (cadastrar, editar, excluir pessoas)** ou **mostrar status e exportar via cliente**, posso fazer na próxima etapa.

Deseja isso?