import sqlite3

import time

import re

# Conectar ao banco de dados SQLite

conn = sqlite3.connect(':memory:') # Usando um banco de dados em memória para simplificação

c = conn.cursor()

# Criar tabela para visitantes

c.execute('''

CREATE TABLE IF NOT EXISTS visitantes (

id INTEGER PRIMARY KEY,

nome TEXT,

cpf TEXT,

placa TEXT,

entrada TIMESTAMP,

saida TIMESTAMP

)

''')

conn.commit()

# Função para validar a placa do automóvel

def validar\_placa(placa):

# Regex para validar o formato da placa (ABC1D23)

padrao = r'^[A-Z]{3}\d[A-Z]\d{2}$'

return re.match(padrao, placa) is not None

# Função para validar CPF

def validar\_cpf(cpf):

# Remove caracteres não numéricos

cpf = re.sub(r'\D', '', cpf)

# Verifica se o CPF tem 11 dígitos

if len(cpf) != 11 or not cpf.isdigit():

return False

# Verifica se o CPF é uma sequência de números repetidos

if cpf == cpf[0] \* 11:

return False

# Validação dos dígitos verificadores

def calcular\_digito(cpf, peso):

soma = sum(int(d) \* peso for d, peso in zip(cpf[:peso-1], range(peso, 1, -1)))

digito = 11 - (soma % 11)

return digito if digito < 10 else 0

primeiro\_digito = calcular\_digito(cpf, 10)

segundo\_digito = calcular\_digito(cpf, 11)

return cpf[-2:] == f"{primeiro\_digito}{segundo\_digito}"

# Função para registrar entrada de visitante

def registrar\_entrada(nome, cpf, placa):

entrada = time.time()

c.execute("INSERT INTO visitantes (nome, cpf, placa, entrada) VALUES (?, ?, ?, ?)", (nome, cpf, placa, entrada))

conn.commit()

print(f"Visitante {nome} com CPF {cpf} e placa {placa} registrado na entrada.")

salvar\_em\_txt(nome, cpf, placa, "Entrada", entrada) # Salva no arquivo de texto

# Função para registrar saída de visitante

def registrar\_saida(nome):

saida = time.time()

c.execute("UPDATE visitantes SET saida = ? WHERE nome = ? AND saida IS NULL", (saida, nome))

conn.commit()

print(f"Visitante {nome} registrado na saída.")

salvar\_em\_txt(nome, None, None, "Saída", saida) # Salva no arquivo de texto

# Função para salvar os registros no arquivo .txt

def salvar\_em\_txt(nome, cpf, placa, tipo, timestamp):

with open('registros\_visitantes.txt', 'a') as f:

entrada\_saida = "Entrada" if tipo == "Entrada" else "Saída"

f.write(f"{entrada\_saida}: Nome: {nome}, CPF: {cpf if cpf else 'N/A'}, Placa: {placa if placa else 'N/A'}, Hora: {time.ctime(timestamp)}\n")

# Função para listar visitantes

def listar\_visitantes():

c.execute("SELECT \* FROM visitantes")

visitantes = c.fetchall()

for visitante in visitantes:

print(f"ID: {visitante[0]}, Nome: {visitante[1]}, CPF: {visitante[2]}, Placa: {visitante[3]}, Entrada: {time.ctime(visitante[4])}, Saída: {time.ctime(visitante[5]) if visitante[5] else 'Ainda presente'}")

# Exemplo de uso

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

while True:

# Solicitar entrada do usuário

nome = input("Digite o nome do visitante (ou 'sair' para encerrar): ")

if nome.lower() == 'sair':

break

# Loop para garantir que o CPF seja válido

while True:

cpf = input("Digite o CPF do visitante (apenas números, 11 dígitos): ")

if validar\_cpf(cpf):

break # CPF válido, sai do loop

else:

print("CPF inválido! Por favor, insira um CPF com 11 dígitos numéricos e que não seja uma sequência repetida.")

# Loop para garantir que a placa seja válida

while True:

placa = input("Digite a placa do visitante (formato: ABC1D23): ")

if validar\_placa(placa):

break # Placa válida, sai do loop

else:

print("Placa inválida! Por favor, insira uma placa no formato correto (ABC1D23).")

# Registrar entrada do visitante

registrar\_entrada(nome, cpf, placa)

# Listar visitantes

print("\nVisitantes registrados:")

listar\_visitantes()

# Perguntar se deseja registrar a saída de algum visitante

registrar\_saida\_input = input("Deseja registrar a saída de algum visitante? (sim/não): ")

if registrar\_saida\_input.lower() == 'sim':

nome\_saida = input("Digite o nome do visitante para registrar a saída: ")

registrar\_saida(nome\_saida)

print("Encerrando o sistema...")