import os  
import sqlite3  
import tkinter as tk  
from datetime import datetime  
from tkinter import ttk, messagebox, simpledialog  
  
import win32api  
import win32print  
from PIL import Image, ImageTk  
from reportlab.lib.pagesizes import letter  
from reportlab.lib.utils import ImageReader  
from reportlab.pdfgen import canvas  
  
  
def criar\_entry(frame, label\_text, row):  
 label = ttk.Label(frame, text=label\_text)  
 label.grid(row=row, column=0, padx=5, pady=5, sticky="e")  
 entry = ttk.Entry(frame)  
 entry.grid(row=row, column=1, padx=5, pady=5, sticky="ew")  
 return entry  
  
  
def selecionar\_impressora():  
 impressoras = [printer[2] for printer in  
 win32print.EnumPrinters(win32print.PRINTER\_ENUM\_LOCAL | win32print.PRINTER\_ENUM\_CONNECTIONS)]  
 impressora\_selecionada = simpledialog.askstring("Selecionar Impressora", "Escolha uma impressora:",  
 initialvalue=impressoras[0])  
 return impressora\_selecionada  
  
  
def obter\_impressoras\_instaladas():  
 try:  
 import win32print  
 return win32print.EnumPrinters(win32print.PRINTER\_ENUM\_LOCAL, None, 1)  
 except ImportError:  
 messagebox.showerror("Erro", "Módulo win32print não encontrado.")  
 return []  
  
  
class GerenciadorOrdensServico:  
 def \_\_init\_\_(self):  
 self.index\_selecionado = None  
 self.label\_info = None  
 self.label\_background = None  
 self.listbox\_pdfs = None  
 self.listbox\_ordens = None  
 self.combobox\_status = None  
 self.entry\_defeito = None  
 self.entry\_numero\_serie = None  
 self.entry\_modelo = None  
 self.entry\_marca = None  
 self.entry\_email = None  
 self.entry\_endereco = None  
 self.entry\_telefone = None  
 self.entry\_cpf\_cnpj = None  
 self.entry\_nome = None  
 self.entry\_data\_final = None  
 self.entry\_data\_inicio = None  
 self.background\_label = None  
 self.background\_photo = None  
 self.logo\_photo = None  
 self.background\_image = None  
 self.logo\_label = None  
 self.logo\_image = None  
 self.entry\_relatorio = None  
 self.img\_logo = None  
 self.conn = sqlite3.connect('ordens\_servico.db')  
 self.cursor = self.conn.cursor()  
 self.criar\_tabela()  
  
 self.window = tk.Tk()  
 self.window.title("Gerenciador de Ordens de Serviço")  
  
 cor\_azul\_claro = "#ADD8E6"  
 "#D3D3D3"  
 "#90EE90"  
  
 # Configurar cor de fundo principal  
 self.window.configure(bg=cor\_azul\_claro)  
  
 # Carregar imagens  
 self.logo\_path = "logo.png"  
 self.background\_path = "background.png"  
  
 # Carregar o logo  
 self.carregar\_logotipo()  
  
 # Configurar imagem de fundo  
 self.configurar\_imagem\_de\_fundo()  
  
 # Informações sobre o programa  
 self.versao = "1.1"  
 self.data\_criacao = "18/06/2024"  
 self.nome\_criador = "John Patrick"  
 self.telefone\_criador = "(62) 9-9814-5605"  
 self.email\_criador = "tecboxon@gmail.com"  
  
 self.logo\_path = "logo.png"  
 self.background\_path = "background.jpg"  
  
 self.setup\_gui()  
  
 self.carregar\_logotipo()  
 self.configurar\_imagem\_de\_fundo()  
 self.atualizar\_informacoes\_programa()  
 self.atualizar\_listagem\_ordens()  
 self.atualizar\_listagem\_pdfs()  
  
 self.window.mainloop()  
  
 def criar\_tabela(self):  
 self.cursor.execute('''  
 CREATE TABLE IF NOT EXISTS ordens\_servico (  
 id INTEGER PRIMARY KEY,  
 nome\_cliente TEXT NOT NULL,  
 cpf\_cnpj TEXT NOT NULL,  
 telefone TEXT,  
 endereco TEXT,  
 email TEXT,  
 marca TEXT,  
 modelo TEXT,  
 numero\_serie TEXT,  
 defeito TEXT,  
 relatorio TEXT,  
 data\_inicio TEXT,  
 data\_final TEXT,  
 status TEXT,  
 data\_criacao TEXT  
 )  
 ''')  
 self.conn.commit()  
  
 def setup\_gui(self):  
 frame\_cliente = ttk.LabelFrame(self.window, text="Dados do Cliente")  
 frame\_cliente.grid(row=0, column=0, padx=10, pady=5, sticky="ew")  
  
 self.entry\_nome = criar\_entry(frame\_cliente, "Nome:", 0)  
 self.entry\_cpf\_cnpj = criar\_entry(frame\_cliente, "CPF/CNPJ:", 1)  
 self.entry\_telefone = criar\_entry(frame\_cliente, "Telefone:", 2)  
 self.entry\_endereco = criar\_entry(frame\_cliente, "Endereço:", 3)  
 self.entry\_email = criar\_entry(frame\_cliente, "E-mail:", 4)  
  
 frame\_equipamento = ttk.LabelFrame(self.window, text="Dados do Equipamento")  
 frame\_equipamento.grid(row=1, column=0, padx=10, pady=5, sticky="ew")  
  
 self.entry\_marca = criar\_entry(frame\_equipamento, "Marca:", 0)  
 self.entry\_modelo = criar\_entry(frame\_equipamento, "Modelo:", 1)  
 self.entry\_numero\_serie = criar\_entry(frame\_equipamento, "Número de Série:", 2)  
 self.entry\_defeito = criar\_entry(frame\_equipamento, "Defeito:", 3)  
 self.entry\_defeito = criar\_entry(frame\_equipamento, "relatorio:", 4)  
 self.entry\_defeito = criar\_entry(frame\_equipamento, "data inicio:", 5)  
 self.entry\_defeito = criar\_entry(frame\_equipamento, "data final:", 6)  
  
 frame\_botoes = ttk.Frame(self.window)  
 frame\_botoes.grid(row=2, column=0, padx=10, pady=5, sticky="ew")  
  
 self.combobox\_status = ttk.Combobox(frame\_botoes, values=["Aberta", "Em Andamento", "Concluída", "Cancelada"],  
 state="readonly")  
 self.combobox\_status.set("Aberta")  
 self.combobox\_status.pack(side=tk.LEFT, padx=5)  
  
 btn\_cadastrar = ttk.Button(frame\_botoes, text="Cadastrar", command=self.cadastrar\_ordem)  
 btn\_cadastrar.pack(side=tk.LEFT, padx=5)  
  
 btn\_editar = ttk.Button(frame\_botoes, text="Editar", command=self.editar\_ordem)  
 btn\_editar.pack(side=tk.LEFT, padx=5)  
  
 btn\_excluir = ttk.Button(frame\_botoes, text="Excluir", command=self.excluir\_ordem)  
 btn\_excluir.pack(side=tk.LEFT, padx=5)  
  
 btn\_gerar\_pdf = ttk.Button(frame\_botoes, text="Gerar PDF", command=self.gerar\_pdf)  
 btn\_gerar\_pdf.pack(side=tk.LEFT, padx=5)  
  
 frame\_listagem\_ordens = ttk.LabelFrame(self.window, text="Listagem de Ordens de Serviço")  
 frame\_listagem\_ordens.grid(row=0, column=1, padx=10, pady=5, sticky="ew")  
  
 self.listbox\_ordens = tk.Listbox(frame\_listagem\_ordens, width=80, height=11)  
 self.listbox\_ordens.pack(padx=5, pady=5, fill=tk.BOTH, expand=True)  
 self.listbox\_ordens.bind("<<ListboxSelect>>", self.selecionar\_ordem)  
  
 frame\_botoes\_extra = ttk.Frame(self.window)  
 frame\_botoes\_extra.grid(row=3, column=0, padx=10, pady=5, sticky="ew")  
  
 btn\_atualizar\_ordens = ttk.Button(frame\_botoes\_extra, text="Atualizar Lista",  
 command=self.atualizar\_listagem\_ordens)  
 btn\_atualizar\_ordens.pack(side=tk.LEFT, padx=5)  
  
 btn\_pesquisar = ttk.Button(frame\_botoes\_extra, text="Pesquisar Ordem", command=self.pesquisar\_ordem)  
 btn\_pesquisar.pack(side=tk.LEFT, padx=5)  
  
 frame\_listagem\_pdfs = ttk.LabelFrame(self.window, text="Listagem de PDFs Gerados")  
 frame\_listagem\_pdfs.grid(row=1, column=1, padx=10, pady=5, sticky="ew")  
  
 self.listbox\_pdfs = tk.Listbox(frame\_listagem\_pdfs, width=100, height=11)  
 self.listbox\_pdfs.pack(padx=5, pady=5, fill=tk.BOTH, expand=True)  
  
 btn\_atualizar\_pdfs = ttk.Button(frame\_listagem\_pdfs, text="Atualizar Lista de PDFs",  
 command=self.atualizar\_listagem\_pdfs)  
 btn\_atualizar\_pdfs.pack(side=tk.LEFT, padx=5)  
  
 btn\_imprimir\_pdf = ttk.Button(frame\_listagem\_pdfs, text="Imprimir PDF", command=self.imprimir\_pdf)  
 btn\_imprimir\_pdf.pack(side=tk.LEFT, padx=5)  
  
 def validar\_entradas(self):  
 if not self.entry\_nome.get() or not self.entry\_cpf\_cnpj.get():  
 messagebox.showwarning("Aviso", "Os campos Nome e CPF/CNPJ são obrigatórios.")  
 return False  
 return True  
  
 def cadastrar\_ordem(self):  
 if not self.validar\_entradas():  
 return  
  
 self.cursor.execute('''  
 INSERT INTO ordens\_servico (nome\_cliente, cpf\_cnpj, telefone, endereco, email, marca, modelo, numero\_serie, defeito, relatorio, data\_inicio, data\_final, status, data\_criacao)  
 VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)  
 ''', (  
 self.entry\_nome.get(),  
 self.entry\_cpf\_cnpj.get(),  
 self.entry\_telefone.get(),  
 self.entry\_endereco.get(),  
 self.entry\_email.get(),  
 self.entry\_marca.get(),  
 self.entry\_modelo.get(),  
 self.entry\_numero\_serie.get(),  
 self.entry\_defeito.get(),  
 self.entry\_relatorio.get(),  
 self.entry\_data\_inicio.get(),  
 self.entry\_data\_final.get(),  
 self.combobox\_status.get(),  
 datetime.now().strftime("%Y-%m-%d %H:%M:%S")  
 ))  
  
 self.conn.commit()  
 self.limpar\_campos()  
 self.atualizar\_listagem\_ordens()  
 messagebox.showinfo("Sucesso", "Ordem de serviço cadastrada com sucesso!")  
  
 def cadastrar\_ordem(self):  
 if self.validar\_entradas():  
 nome = self.entry\_nome.get()  
 cpf\_cnpj = self.entry\_cpf\_cnpj.get()  
 telefone = self.entry\_telefone.get()  
 endereco = self.entry\_endereco.get()  
 email = self.entry\_email.get()  
 marca = self.entry\_marca.get()  
 modelo = self.entry\_modelo.get()  
 numero\_serie = self.entry\_numero\_serie.get()  
 defeito = self.entry\_defeito.get()  
 status = self.combobox\_status.get()  
 data\_criacao = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  
  
 self.cursor.execute('''  
 INSERT INTO ordens\_servico (nome\_cliente, cpf\_cnpj, telefone, endereco, email, marca, modelo, numero\_serie, defeito, status, data\_criacao)  
 VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)  
 ''', (  
 nome, cpf\_cnpj, telefone, endereco, email, marca, modelo, numero\_serie, defeito, status, data\_criacao))  
 self.conn.commit()  
 self.atualizar\_listagem\_ordens()  
 self.limpar\_campos()  
  
 def editar\_ordem(self):  
 if self.validar\_entradas():  
 try:  
 # Obter o índice selecionado previamente  
 index = self.index\_selecionado  
 if index is None:  
 raise IndexError("Nenhuma ordem de serviço selecionada.")  
  
 ordem\_id = self.listbox\_ordens.get(index).split(' ')[0]  
  
 nome = self.entry\_nome.get()  
 cpf\_cnpj = self.entry\_cpf\_cnpj.get()  
 telefone = self.entry\_telefone.get()  
 endereco = self.entry\_endereco.get()  
 email = self.entry\_email.get()  
 marca = self.entry\_marca.get()  
 modelo = self.entry\_modelo.get()  
 numero\_serie = self.entry\_numero\_serie.get()  
 defeito = self.entry\_defeito.get()  
 relatorio = self.entry\_relatorio.get()  
 status = self.combobox\_status.get()  
  
 self.cursor.execute('''  
 UPDATE ordens\_servico  
 SET nome\_cliente=?, cpf\_cnpj=?, telefone=?, endereco=?, email=?, marca=?, modelo=?, numero\_serie=?, defeito=?, relatorio=?, status=?  
 WHERE id=?  
 ''', (  
 nome, cpf\_cnpj, telefone, endereco, email, marca, modelo, numero\_serie, defeito, relatorio, status, ordem\_id))  
 self.conn.commit()  
 self.atualizar\_listagem\_ordens()  
 self.limpar\_campos()  
 self.index\_selecionado () # Limpar a seleção após a edição  
 except IndexError:  
 messagebox.showerror("Erro", "Selecione uma ordem de serviço para editar.")  
 def excluir\_ordem(self):  
 try:  
 index = self.listbox\_ordens.curselection()[0]  
 ordem\_id = self.listbox\_ordens.get(index).split(' ')[0]  
  
 self.cursor.execute('DELETE FROM ordens\_servico WHERE id=?', (ordem\_id,))  
 self.conn.commit()  
 self.atualizar\_listagem\_ordens()  
 self.limpar\_campos()  
 except IndexError:  
 messagebox.showerror("Erro", "Selecione uma ordem de serviço para excluir.")  
  
 def selecionar\_ordem(self, event):  
 try:  
 ordem\_selecionada = self.listbox\_ordens.curselection()[0]  
 ordem\_id = self.listbox\_ordens.get(ordem\_selecionada).split()[0]  
 except IndexError:  
 return  
  
 self.cursor.execute('SELECT \* FROM ordens\_servico WHERE id=?', (ordem\_id,))  
 ordem = self.cursor.fetchone()  
  
 if ordem:  
 self.entry\_nome.delete(0, tk.END)  
 self.entry\_nome.insert(0, ordem[1])  
  
 self.entry\_cpf\_cnpj.delete(0, tk.END)  
 self.entry\_cpf\_cnpj.insert(0, ordem[2])  
  
 self.entry\_telefone.delete(0, tk.END)  
 self.entry\_telefone.insert(0, ordem[3])  
  
 self.entry\_endereco.delete(0, tk.END)  
 self.entry\_endereco.insert(0, ordem[4])  
  
 self.entry\_email.delete(0, tk.END)  
 self.entry\_email.insert(0, ordem[5])  
  
 self.entry\_marca.delete(0, tk.END)  
 self.entry\_marca.insert(0, ordem[6])  
  
 self.entry\_modelo.delete(0, tk.END)  
 self.entry\_modelo.insert(0, ordem[7])  
  
 self.entry\_numero\_serie.delete(0, tk.END)  
 self.entry\_numero\_serie.insert(0, ordem[8])  
  
 self.entry\_defeito.delete(0, tk.END)  
 self.entry\_defeito.insert(0, ordem[9])  
  
 self.combobox\_status.set(ordem[10])  
  
 def preencher\_campos(self, ordem):  
 self.entry\_nome.delete(0, tk.END)  
 self.entry\_nome.insert(0, ordem[1])  
 self.entry\_cpf\_cnpj.delete(0, tk.END)  
 self.entry\_cpf\_cnpj.insert(0, ordem[2])  
 self.entry\_telefone.delete(0, tk.END)  
 self.entry\_telefone.insert(0, ordem[3])  
 self.entry\_endereco.delete(0, tk.END)  
 self.entry\_endereco.insert(0, ordem[4])  
 self.entry\_email.delete(0, tk.END)  
 self.entry\_email.insert(0, ordem[5])  
 self.entry\_marca.delete(0, tk.END)  
 self.entry\_marca.insert(0, ordem[6])  
 self.entry\_modelo.delete(0, tk.END)  
 self.entry\_modelo.insert(0, ordem[7])  
 self.entry\_numero\_serie.delete(0, tk.END)  
 self.entry\_numero\_serie.insert(0, ordem[8])  
 self.entry\_defeito.delete(0, tk.END)  
 self.entry\_defeito.insert(0, ordem[9])  
 self.entry\_relatorio.delete(0, tk.END)  
 self.entry\_relatorio.insert(0, ordem[10])  
 self.entry\_data\_inicio.delete(0, tk.END)  
 self.entry\_data\_inicio.insert(0, ordem[11])  
 self.entry\_data\_final.delete(0, tk.END)  
 self.entry\_data\_final.insert(0, ordem[12])  
 self.combobox\_status.set(ordem[13])  
  
 def limpar\_campos(self):  
 self.entry\_nome.delete(0, tk.END)  
 self.entry\_cpf\_cnpj.delete(0, tk.END)  
 self.entry\_telefone.delete(0, tk.END)  
 self.entry\_endereco.delete(0, tk.END)  
 self.entry\_email.delete(0, tk.END)  
 self.entry\_marca.delete(0, tk.END)  
 self.entry\_modelo.delete(0, tk.END)  
 self.entry\_numero\_serie.delete(0, tk.END)  
 self.entry\_defeito.delete(0, tk.END)  
 self.entry\_relatorio.delete(0, tk.END)  
 self.entry\_data\_inicio.delete(0, tk.END)  
 self.entry\_data\_final.delete(0, tk.END)  
 self.combobox\_status.set("Aberta")  
 self.combobox\_status.set("Fechada")  
  
 def atualizar\_listagem\_ordens(self):  
 self.listbox\_ordens.delete(0, tk.END)  
 self.cursor.execute('SELECT id, nome\_cliente, cpf\_cnpj, telefone, endereco, email, marca, modelo, '  
 'numero\_serie, defeito, status, data\_criacao FROM ordens\_servico')  
 for row in self.cursor.fetchall():  
 self.listbox\_ordens.insert(tk.END, f"{row[0]} - {row[1]} - {row[2]} - {row[3]}")  
  
 def gerar\_pdf(self):  
 try:  
 ordem\_selecionada = self.listbox\_ordens.curselection()[0]  
 ordem\_id = self.listbox\_ordens.get(ordem\_selecionada).split()[0]  
 self.cursor.execute('SELECT \* FROM ordens\_servico WHERE id=?', (ordem\_id,))  
 ordem = self.cursor.fetchone()  
 except IndexError:  
 messagebox.showwarning("Aviso", "Nenhuma ordem de serviço selecionada.")  
 return  
  
 if not os.path.exists("pdfs\_gerados"):  
 os.makedirs("pdfs\_gerados")  
  
 caminho\_arquivo = os.path.join("pdfs\_gerados", f"ordem\_servico\_{ordem[0]}.pdf")  
 c = canvas.Canvas(caminho\_arquivo, pagesize=letter)  
 altura = letter[1]  
  
 if os.path.exists(self.logo\_path):  
 c.drawImage(ImageReader(self.logo\_path), 40, altura - 70, width=120, height=80)  
  
 c.setFont("Helvetica-Bold", 18)  
 c.drawString(200, altura - 120, "Ordem de Serviço")  
  
 c.setFont("Helvetica", 19)  
 c.drawString(40, altura - 120, f"ID: {ordem[0]}")  
 c.drawString(40, altura - 140, f"Cliente: {ordem[1]}")  
 c.drawString(40, altura - 160, f"CPF/CNPJ: {ordem[2]}")  
 c.drawString(40, altura - 180, f"Telefone: {ordem[3]}")  
 c.drawString(40, altura - 200, f"Endereço: {ordem[4]}")  
 c.drawString(40, altura - 220, f"Marca: {ordem[6]}")  
 c.drawString(40, altura - 240, f"Modelo: {ordem[7]}")  
 c.drawString(40, altura - 260, f"Número de Série: {ordem[8]}")  
 c.drawString(40, altura - 280, f"Defeito: {ordem[9]}")  
 c.drawString(40, altura - 300, f"Status: {ordem[10]}")  
 c.drawString(40, altura - 320, f"Data de Criação: {ordem[11]}")  
  
  
 c.showPage()  
 c.save()  
  
 self.atualizar\_listagem\_pdfs()  
  
 messagebox.showinfo("Sucesso", f"PDF gerado com sucesso: {caminho\_arquivo}")  
  
 def atualizar\_listagem\_pdfs(self):  
 self.listbox\_pdfs.delete(0, tk.END)  
 if not os.path.exists("pdfs\_gerados"):  
 os.makedirs("pdfs\_gerados")  
 for arquivo in os.listdir("pdfs\_gerados"):  
 if arquivo.endswith(".pdf"):  
 self.listbox\_pdfs.insert(tk.END, arquivo)  
  
 def imprimir\_pdf(self):  
 try:  
 pdf\_selecionado = self.listbox\_pdfs.curselection()[0]  
 pdf\_path = os.path.join("pdfs\_gerados", self.listbox\_pdfs.get(pdf\_selecionado))  
 except IndexError:  
 messagebox.showwarning("Aviso", "Nenhum PDF selecionado.")  
 return  
  
 impressora\_selecionada = selecionar\_impressora()  
 if impressora\_selecionada:  
 win32print.SetDefaultPrinter(impressora\_selecionada)  
 win32api.ShellExecute("print", pdf\_path, None, ".", 0)  
 messagebox.showinfo("Sucesso", f"PDF enviado para a impressora: {impressora\_selecionada}")  
  
 def pesquisar\_ordem(self):  
 termo\_pesquisa = simpledialog.askstring("Pesquisar Ordem", "Digite o nome ou CPF/CNPJ do cliente:")  
 if termo\_pesquisa:  
 self.listbox\_ordens.delete(0, tk.END)  
 self.cursor.execute('''  
 SELECT id, nome\_cliente, cpf\_cnpj, telefone, status FROM ordens\_servico  
 WHERE nome\_cliente LIKE ? OR cpf\_cnpj LIKE ?  
 ''', (f"%{termo\_pesquisa}%", f"%{termo\_pesquisa}%"))  
 for row in self.cursor.fetchall():  
 self.listbox\_ordens.insert(tk.END, f"{row[0]} - {row[1]} - {row[2]} - {row[3]} - {row[4]}")  
  
 def carregar\_logotipo(self):  
  
 self.logo\_image = Image.open(self.logo\_path)  
 self.logo\_photo = ImageTk.PhotoImage(self.logo\_image)  
 self.logo\_label = tk.Label(self.window, image=self.logo\_photo)  
 self.logo\_label.grid(row=4, column=0, pady=0, padx=0, sticky=tk.N)  
  
 def configurar\_imagem\_de\_fundo(self):  
 if os.path.exists(self.background\_path):  
 img\_background = Image.open(self.background\_path)  
 img\_background = img\_background.resize((1024, 768), Image.ANTIALIAS)  
 self.background\_image = ImageTk.PhotoImage(img\_background)  
 self.label\_background = tk.Label(self.window, image=self.background\_image)  
 self.label\_background.place(relwidth=1, relheight=1)  
  
 def atualizar\_informacoes\_programa(self):  
 info\_texto = f"Versão: {self.versao}\n" \  
 f"Data de Criação: {self.data\_criacao}\n" \  
 f"Criado por: {self.nome\_criador}\n" \  
 f"Telefone: {self.telefone\_criador}\n" \  
 f"E-mail: {self.email\_criador}"  
 self.label\_info = tk.Label(self.window, text=info\_texto, bg="#ADD8E6", justify=tk.LEFT)  
 self.label\_info.grid(row=4, column=1, pady=10, padx=10, sticky=tk.N)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = GerenciadorOrdensServico()