from urllib.parse import quote\_plus

import customtkinter as ctk

import tkinter as tk

from tkinter import filedialog, messagebox

import pandas as pd

from sqlalchemy import create\_engine, text

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.application import MIMEApplication

import smtplib

import os

import datetime

from io import BytesIO

from reportlab.pdfgen import canvas

from reportlab.lib.pagesizes import A4

from tkinter import ttk

import tkinter.filedialog as fd

# ========== Configurações iniciais ===========

ctk.set\_appearance\_mode("System")

ctk.set\_default\_color\_theme("blue")

class App(*ctk*.*CTk*):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.title("Gestão de Clientes e Envio de E-mails")

        self.geometry("1000x700")

        # Conexão ao banco

        self.engine = self.get\_engine(

            host="192.168.52.180",

            port="1433",

            database="factura\_email",

            username="sa",

            password="loucoste9850053"

        )

        # Login do remetente

        self.remetente = "p.costa@edec.cv"

        self.senha\_app = "vmqxbylwajqaliwq"

        # Layout

        self.menu\_lateral = ctk.CTkFrame(self, width=200)

        self.menu\_lateral.pack(side="left", fill="y")

        self.container = ctk.CTkFrame(self)

        self.container.pack(side="right", expand=True, fill="both")

        # Botões do menu

        botoes = [

            ("📋 Cadastro", self.abrir\_cadastro),

            ("📧 CC Conhecimento", self.abrir\_cc),

            ("🔍 Consulta", self.abrir\_consulta),

            ("📤 Enviar E-mails", self.abrir\_envio),

            ("📊 Relatório", self.abrir\_relatorio)

        ]

        for texto, comando in botoes:

            btn = ctk.CTkButton(self.menu\_lateral, text=texto, command=comando)

            btn.pack(pady=10, padx=10, fill="x")

        # Inicialização com tela de cadastro

        self.abrir\_cadastro()

    # ========== Conexão DB ==========

    def get\_engine(self, host, database, username, password, port="1433"):

        from urllib.parse import quote\_plus

        try:

            password\_enconde = quote\_plus(password)

            connection\_url = (

                f"mssql+pyodbc://{username}:{password\_enconde}@{host},{port}/{database}"

                "?driver=ODBC+Driver+17+for+SQL+Server"

            )

            engine = create\_engine(connection\_url)

            return engine

        except Exception as e:

            messagebox.showerror("Erro de conexão", str(e))

            return None

    # ========== Navegação ==========

    def limpar\_container(self):

        for widget in self.container.winfo\_children():

            widget.destroy()

    def abrir\_cadastro(self):

        self.limpar\_container()

        CadastroFrame(self.container, self.engine).pack(expand=True, fill="both")

    def abrir\_cc(self):

        self.limpar\_container()

        CCFrame(self.container, self.engine).pack(expand=True, fill="both")

    def abrir\_consulta(self):

        self.limpar\_container()

        ConsultaFrame(self.container, self.engine).pack(expand=True, fill="both")

    def abrir\_envio(self):

        self.limpar\_container()

        EnvioFrame(self.container, self.engine, self.remetente, self.senha\_app).pack(expand=True, fill="both")

    def abrir\_relatorio(self):

        self.limpar\_container()

        RelatorioFrame(self.container, self.engine).pack(expand=True, fill="both")

#=============== cadastro ====================

class CadastroFrame(*ctk*.*CTkFrame*):

    def \_\_init\_\_(self, master, engine):

        super().\_\_init\_\_(master)

        self.engine = engine

        ctk.CTkLabel(self, text="📋 Cadastro de Cliente", font=ctk.CTkFont(size=20, weight="bold")).pack(pady=20)

        self.form\_frame = ctk.CTkFrame(self)

        self.form\_frame.pack(pady=10)

        # Campos do formulário

        self.cil\_var = tk.StringVar()

        self.nome\_var = tk.StringVar()

        self.email\_var = tk.StringVar()

        ctk.CTkLabel(self.form\_frame, text="CIL").grid(row=0, column=0, padx=10, pady=5, sticky="e")

        self.cil\_entry = ctk.CTkEntry(self.form\_frame, textvariable=self.cil\_var, width=250)

        self.cil\_entry.grid(row=0, column=1, padx=10, pady=5)

        ctk.CTkLabel(self.form\_frame, text="Nome").grid(row=1, column=0, padx=10, pady=5, sticky="e")

        self.nome\_entry = ctk.CTkEntry(self.form\_frame, textvariable=self.nome\_var, width=250)

        self.nome\_entry.grid(row=1, column=1, padx=10, pady=5)

        ctk.CTkLabel(self.form\_frame, text="Email").grid(row=2, column=0, padx=10, pady=5, sticky="e")

        self.email\_entry = ctk.CTkEntry(self.form\_frame, textvariable=self.email\_var, width=250)

        self.email\_entry.grid(row=2, column=1, padx=10, pady=5)

        ctk.CTkLabel(self.form\_frame, text="Nome do PDF").grid(row=3, column=0, padx=10, pady=5, sticky="e")

        self.nome\_pdf\_label = ctk.CTkLabel(self.form\_frame, text="", text\_color="gray")

        self.nome\_pdf\_label.grid(row=3, column=1, padx=10, pady=5)

        # Botões

        self.btn\_frame = ctk.CTkFrame(self)

        self.btn\_frame.pack(pady=15)

        ctk.CTkButton(self.btn\_frame, text="Cadastrar Cliente", command=self.cadastrar\_cliente).grid(row=0, column=0, padx=10)

        ctk.CTkButton(self.btn\_frame, text="Limpar", command=self.limpar\_campos).grid(row=0, column=1, padx=10)

        # Atualiza nome do PDF quando CIL muda

        self.cil\_var.trace\_add("write", self.atualizar\_nome\_pdf)

    def atualizar\_nome\_pdf(self, \*args):

        cil = self.cil\_var.get()

        nome\_pdf = f"{cil}.pdf" if cil else ""

        self.nome\_pdf\_label.configure(text=nome\_pdf)

    def limpar\_campos(self):

        self.cil\_var.set("")

        self.nome\_var.set("")

        self.email\_var.set("")

        self.nome\_pdf\_label.configure(text="")

    def cadastrar\_cliente(self):

        cil = self.cil\_var.get().strip()

        nome = self.nome\_var.get().strip()

        email = self.email\_var.get().strip()

        nome\_pdf = f"{cil}.pdf"

        if not cil or not nome or not email:

            messagebox.showwarning("Campos obrigatórios", "Por favor, preencha todos os campos.")

            return

        try:

            with self.engine.begin() as conn:

                result = conn.execute(

                    text("SELECT COUNT(\*) FROM clientes WHERE cil = :cil"),

                    {"cil": cil}

                )

                count = result.scalar()

                if count > 0:

                    messagebox.showwarning("Duplicado", "Cliente com este CIL já está cadastrado.")

                else:

                    conn.execute(

                        text("INSERT INTO clientes (cil, nome, email, arquivo\_anexo) VALUES (:cil, :nome, :email, :anexo)"),

                        {"cil": cil, "nome": nome, "email": email, "anexo": nome\_pdf}

                    )

                    messagebox.showinfo("Sucesso", "Cliente cadastrado com sucesso!")

                    self.limpar\_campos()

        except Exception as e:

            messagebox.showerror("Erro", f"Erro ao cadastrar: {e}")

#========= para conhecimento==================

class CCFrame(*ctk*.*CTkFrame*):

    def \_\_init\_\_(self, master, engine):

        super().\_\_init\_\_(master)

        self.engine = engine

        self.df = pd.DataFrame()

        self.edits = {}

        self.check\_vars = {}

        ctk.CTkLabel(self, text="📧 E-mails para Conhecimento (CC)", font=ctk.CTkFont(size=20, weight="bold")).pack(pady=20)

        # Tabela de CC

        self.tabela\_frame = ctk.CTkScrollableFrame(self, height=300)

        self.tabela\_frame.pack(padx=20, pady=10, fill="both")

        self.carregar\_dados()

        # Formulário de novo e-mail

        ctk.CTkLabel(self, text="➕ Adicionar novo e-mail CC").pack(pady=(20, 5))

        self.novo\_email\_var = tk.StringVar()

        ctk.CTkEntry(self, textvariable=self.novo\_email\_var, width=300).pack()

        ctk.CTkButton(self, text="Adicionar", command=self.adicionar\_email).pack(pady=10)

        # Botões de ações

        self.btn\_frame = ctk.CTkFrame(self)

        self.btn\_frame.pack(pady=15)

        ctk.CTkButton(self.btn\_frame, text="🗑️ Excluir Selecionados", command=self.excluir\_selecionados).grid(row=0, column=0, padx=10)

        ctk.CTkButton(self.btn\_frame, text="💾 Salvar Alterações", command=self.salvar\_edicoes).grid(row=0, column=1, padx=10)

    def carregar\_dados(self):

        for widget in self.tabela\_frame.winfo\_children():

            widget.destroy()

        try:

            self.df = pd.read\_sql("SELECT \* FROM cc\_email", self.engine)

        except Exception as e:

            messagebox.showerror("Erro", f"Erro ao carregar dados: {e}")

            return

        self.check\_vars = {}

        self.edits = {}

        for i, row in self.df.iterrows():

            var = tk.BooleanVar()

            self.check\_vars[row["id"]] = var

            email\_var = tk.StringVar(value=row["email\_cc"])

            self.edits[row["id"]] = email\_var

            ctk.CTkCheckBox(self.tabela\_frame, text="", variable=var).grid(row=i, column=0, padx=5)

            ctk.CTkEntry(self.tabela\_frame, textvariable=email\_var, width=300).grid(row=i, column=1, padx=5)

    def adicionar\_email(self):

        novo = self.novo\_email\_var.get().strip()

        if not novo:

            messagebox.showwarning("Atenção", "Informe o e-mail antes de adicionar.")

            return

        try:

            df\_novo = pd.DataFrame({"email\_cc": [novo]})

            df\_novo.to\_sql("cc\_email", con=self.engine, if\_exists="append", index=False)

            self.novo\_email\_var.set("")

            self.carregar\_dados()

            messagebox.showinfo("Sucesso", "E-mail adicionado com sucesso.")

        except Exception as e:

            messagebox.showerror("Erro", f"Erro ao adicionar e-mail: {e}")

    def excluir\_selecionados(self):

        selecionados = [k for k, v in self.check\_vars.items() if v.get()]

        if not selecionados:

            messagebox.showwarning("Atenção", "Nenhum e-mail selecionado.")

            return

        try:

            with self.engine.begin() as conn:

                for id\_ in selecionados:

                    conn.execute(text("DELETE FROM cc\_email WHERE id = :id"), {"id": id\_})

            self.carregar\_dados()

            messagebox.showinfo("Sucesso", "E-mails excluídos com sucesso.")

        except Exception as e:

            messagebox.showerror("Erro", f"Erro ao excluir: {e}")

    def salvar\_edicoes(self):

        try:

            with self.engine.begin() as conn:

                for id\_, var in self.edits.items():

                    conn.execute(

                        text("UPDATE cc\_email SET email\_cc = :email WHERE id = :id"),

                        {"email": var.get().strip(), "id": id\_}

                    )

            messagebox.showinfo("Sucesso", "Alterações salvas com sucesso.")

        except Exception as e:

            messagebox.showerror("Erro", f"Erro ao salvar alterações: {e}")

#========= consultar cadastro =================

class ConsultaFrame(*ctk*.*CTkFrame*):

    def \_\_init\_\_(self, master, engine):

        super().\_\_init\_\_(master)

        self.engine = engine

        self.df\_original = pd.DataFrame()

        self.edits = {}

        self.check\_vars = {}

        ctk.CTkLabel(self, text="🔍 Consultar Cadastro de Clientes", font=ctk.CTkFont(size=20, weight="bold")).pack(pady=20)

        self.filtro\_frame = ctk.CTkFrame(self)

        self.filtro\_frame.pack(pady=10)

        ctk.CTkLabel(self.filtro\_frame, text="Filtrar por CIL:").grid(row=0, column=0, padx=10, pady=5)

        self.cil\_filtro = ctk.CTkComboBox(self.filtro\_frame, values=["Todos"], command=self.filtrar)

        self.cil\_filtro.grid(row=0, column=1, padx=10, pady=5)

        self.tabela\_frame = ctk.CTkScrollableFrame(self, height=350)

        self.tabela\_frame.pack(pady=10, fill="both", expand=True)

        # Botões

        btns = ctk.CTkFrame(self)

        btns.pack(pady=10)

        ctk.CTkButton(btns, text="🗑️ Excluir Selecionados", command=self.excluir\_selecionados).grid(row=0, column=0, padx=10)

        ctk.CTkButton(btns, text="💾 Salvar Alterações", command=self.salvar\_alteracoes).grid(row=0, column=1, padx=10)

        self.carregar\_dados()

    def carregar\_dados(self):

        for widget in self.tabela\_frame.winfo\_children():

            widget.destroy()

        try:

            self.df\_original = pd.read\_sql("SELECT cil, nome, email FROM clientes", self.engine)

        except Exception as e:

            messagebox.showerror("Erro", f"Erro ao carregar clientes: {e}")

            return

        self.lista\_cil = self.df\_original["cil"].dropna().astype(str).tolist()

        self.cil\_filtro.configure(values=["Todos"] + self.lista\_cil)

        self.cil\_filtro.set("Todos")

        self.exibir\_tabela(self.df\_original)

    def filtrar(self, valor):

        if valor == "Todos":

            self.exibir\_tabela(self.df\_original)

        else:

            filtrado = self.df\_original[self.df\_original["cil"] == valor]

            self.exibir\_tabela(filtrado)

    def exibir\_tabela(self, df):

        for widget in self.tabela\_frame.winfo\_children():

            widget.destroy()

        self.check\_vars = {}

        self.edits = {}

        for i, row in df.iterrows():

            var = tk.BooleanVar()

            self.check\_vars[row["cil"]] = var

            email\_var = tk.StringVar(value=row["email"])

            self.edits[row["cil"]] = email\_var

            ctk.CTkCheckBox(self.tabela\_frame, text="", variable=var).grid(row=i, column=0, padx=5)

            ctk.CTkLabel(self.tabela\_frame, text=row["cil"], width=100).grid(row=i, column=1, padx=5)

            ctk.CTkLabel(self.tabela\_frame, text=row["nome"], width=200).grid(row=i, column=2, padx=5)

            ctk.CTkEntry(self.tabela\_frame, textvariable=email\_var, width=250).grid(row=i, column=3, padx=5)

    def excluir\_selecionados(self):

        selecionados = [cil for cil, var in self.check\_vars.items() if var.get()]

        if not selecionados:

            messagebox.showwarning("Atenção", "Nenhum cliente selecionado.")

            return

        try:

            with self.engine.begin() as conn:

                for cil in selecionados:

                    conn.execute(text("DELETE FROM clientes WHERE cil = :cil"), {"cil": cil})

            self.carregar\_dados()

            messagebox.showinfo("Sucesso", "Clientes excluídos com sucesso.")

        except Exception as e:

            messagebox.showerror("Erro", f"Erro ao excluir: {e}")

    def salvar\_alteracoes(self):

        try:

            with self.engine.begin() as conn:

                for cil, email\_var in self.edits.items():

                    conn.execute(

                        text("UPDATE clientes SET email = :email WHERE cil = :cil"),

                        {"email": email\_var.get().strip(), "cil": cil}

                    )

            messagebox.showinfo("Sucesso", "Alterações salvas com sucesso.")

        except Exception as e:

            messagebox.showerror("Erro", f"Erro ao salvar alterações: {e}")

#============ envio email =====================

class EnvioFrame(*ctk*.*CTkFrame*):

    def \_\_init\_\_(self, master, engine, remetente, senha\_app):

        super().\_\_init\_\_(master)

        self.engine = engine

        self.remetente = remetente

        self.senha\_app = senha\_app

        self.pdf\_dict = {}

        ctk.CTkLabel(self, text="📤 Envio de E-mails com Anexo PDF", font=ctk.CTkFont(size=20, weight="bold")).pack(pady=20)

        # Botão para selecionar múltiplos arquivos PDF

        ctk.CTkButton(self, text="Selecionar Directotio", command=self.selecionar\_pdfs).pack(pady=10)

        self.arquivos\_label = ctk.CTkLabel(self, text="")

        self.arquivos\_label.pack(pady=5)

        # Botão para iniciar envio

        ctk.CTkButton(self, text="📨 Enviar E-mails", command=self.enviar\_em\_lote).pack(pady=15)

        self.status\_box = tk.Text(self, height=15, width=100)

        self.status\_box.pack(pady=10)

    def selecionar\_pdfs(self):

        """Selecionar diretório contendo arquivos PDF"""

        self.pdfs = filedialog.askdirectory(title="Selecionar diretório")

        if self.pdfs:

            print(f"Diretório selecionado: {self.pdfs}")

            # Lista os PDFs do diretório e atualiza a label e dicionário

            self.pdf\_dict.clear()

            nomes = []

            for nome\_arquivo in os.listdir(self.pdfs):

                if nome\_arquivo.lower().endswith('.pdf'):

                    caminho\_completo = os.path.join(self.pdfs, nome\_arquivo)

                    self.pdf\_dict[nome\_arquivo] = caminho\_completo

                    nomes.append(nome\_arquivo)

            self.arquivos\_label.configure(text=f"{len(nomes)} arquivo(s) PDF encontrado(s)")

    def enviar\_em\_lote(self):

        if not self.pdf\_dict:

            messagebox.showwarning("Atenção", "Nenhum arquivo PDF selecionado.")

            return

        try:

            df\_clientes = pd.read\_sql("SELECT cil, nome, email, arquivo\_anexo FROM clientes", self.engine)

            df\_cc = pd.read\_sql("SELECT email\_cc FROM cc\_email", self.engine)

            emails\_cc = df\_cc["email\_cc"].dropna().tolist()

        except Exception as e:

            messagebox.showerror("Erro", f"Erro ao carregar dados: {e}")

            return

        relatorio\_envio = []

        self.status\_box.delete(1.0, tk.END)

        for idx, row in df\_clientes.iterrows():

            cil, nome, email, anexo = row["cil"], row["nome"], row["email"], row["arquivo\_anexo"]

            caminho\_pdf = self.pdf\_dict.get(anexo)

            if not caminho\_pdf or not os.path.isfile(caminho\_pdf):

                self.status\_box.insert(tk.END, f"[IGNORADO] {cil} - Arquivo '{anexo}' não encontrado.\n")

                continue

            status, mensagem = self.enviar\_email(email, nome, cil, caminho\_pdf, emails\_cc)

            relatorio\_envio.append((nome, email, cil, status, mensagem, datetime.datetime.now()))

            self.status\_box.insert(tk.END, f"[{status}] {nome} - {mensagem}\n")

            self.status\_box.see(tk.END)

            self.update()

        # Gravar no banco de dados

        try:

            with self.engine.begin() as conn:

                for r in relatorio\_envio:

                    conn.execute(text("""

                        INSERT INTO relatorio (nome, email, cil, status, mensagem, data\_envio)

                        VALUES (:nome, :email, :cil, :status, :mensagem, :data\_envio)

                    """), {

                        "nome": r[0], "email": r[1], "cil": r[2],

                        "status": r[3], "mensagem": r[4], "data\_envio": r[5]

                    })

            messagebox.showinfo("Concluído", "Todos os e-mails foram processados.")

        except Exception as e:

            messagebox.showerror("Erro", f"Erro ao salvar relatório: {e}")

    def obter\_corpo\_padrao(self):

        with self.engine.connect() as conn:

            result = conn.execute(text("SELECT conteudo FROM corpo\_email WHERE id = 1"))

            row = result.fetchone()

            return row[0] if row else ""

    def enviar\_email(self, destinatario, nome, cil, caminho\_anexo, cc\_list=None):

        cc\_list = cc\_list or []

        #buscar coprp de texto

        corpo\_padrao = self.obter\_corpo\_padrao()

        msg = MIMEMultipart()

        msg['From'] = self.remetente

        msg['To'] = destinatario

        msg['Subject'] = "Factura de Energia da Empresa EDEC"

        if cc\_list:

            msg['Cc'] = ", ".join(cc\_list)

        corpo = f"""

        Olá {nome},

        📄 CIL: {cil}

        {corpo\_padrao}

        """

        msg.attach(MIMEText(corpo, 'plain'))

        with open(caminho\_anexo, 'rb') as f:

            part = MIMEApplication(f.read(), Name=os.path.basename(caminho\_anexo))

        part['Content-Disposition'] = f'attachment; filename="{os.path.basename(caminho\_anexo)}"'

        msg.attach(part)

        try:

            with smtplib.SMTP('smtp.office365.com', 587) as smtp:

                smtp.starttls()

                smtp.login(self.remetente, self.senha\_app)

                smtp.send\_message(msg)

            return "Sucesso", "Email enviado com sucesso."

        except Exception as e:

            return "Erro", str(e)

#========= relatorio ==========================

class RelatorioFrame(*ctk*.*CTkFrame*):

    def \_\_init\_\_(self, master, engine):

        super().\_\_init\_\_(master)

        self.engine = engine

        self.df = pd.DataFrame()

        ctk.CTkLabel(self, text="📊 Relatório de Envios", font=ctk.CTkFont(size=20, weight="bold")).pack(pady=20)

        self.filtro\_frame = ctk.CTkFrame(self)

        self.filtro\_frame.pack(pady=10)

        self.inicio\_var = tk.StringVar()

        self.fim\_var = tk.StringVar()

        ctk.CTkLabel(self.filtro\_frame, text="Data Início (AAAA-MM-DD):").grid(row=0, column=0, padx=5)

        ctk.CTkEntry(self.filtro\_frame, textvariable=self.inicio\_var, width=120).grid(row=0, column=1, padx=5)

        ctk.CTkLabel(self.filtro\_frame, text="Data Fim (AAAA-MM-DD):").grid(row=0, column=2, padx=5)

        ctk.CTkEntry(self.filtro\_frame, textvariable=self.fim\_var, width=120).grid(row=0, column=3, padx=5)

        ctk.CTkButton(self.filtro\_frame, text="🔍 Filtrar", command=self.filtrar).grid(row=0, column=4, padx=10)

        self.tabela = tk.Text(self, height=15, width=120)

        self.tabela.pack(pady=10)

        self.btn\_frame = ctk.CTkFrame(self)

        self.btn\_frame.pack(pady=10)

        ctk.CTkButton(self.btn\_frame, text="📥 Exportar Excel", command=self.exportar\_excel).grid(row=0, column=0, padx=10)

        ctk.CTkButton(self.btn\_frame, text="📄 Exportar PDF", command=self.exportar\_pdf).grid(row=0, column=1, padx=10)

        self.carregar\_dados()

    def carregar\_dados(self):

        try:

            self.df = pd.read\_sql("SELECT nome, email, cil, status, mensagem, data\_envio FROM relatorio ORDER BY data\_envio DESC", self.engine)

            self.df["data\_envio"] = pd.to\_datetime(self.df["data\_envio"], errors="coerce")

            self.exibir()

        except Exception as e:

            messagebox.showerror("Erro", f"Erro ao carregar relatório: {e}")

    def filtrar(self):

        inicio = self.inicio\_var.get().strip()

        fim = self.fim\_var.get().strip()

        try:

            df\_filtrado = self.df.copy()

            if inicio:

                inicio\_dt = pd.to\_datetime(inicio).normalize()

                df\_filtrado = df\_filtrado[df\_filtrado["data\_envio"] >= inicio\_dt]

            if fim:

                fim\_dt = pd.to\_datetime(fim).normalize() + pd.Timedelta(days=1) - pd.Timedelta(seconds=1)

                df\_filtrado = df\_filtrado[df\_filtrado["data\_envio"] <= fim\_dt]

            self.exibir(df\_filtrado)

        except Exception as e:

            messagebox.showerror("Erro", f"Erro ao aplicar filtro: {e}")

    def exibir(self, df=None):

        df = df if df is not None else self.df

        self.tabela.delete(1.0, tk.END)

        for \_, row in df.iterrows():

            linha = f"{row['data\_envio'].strftime('%Y-%m-%d %H:%M')} | {row['nome']} | {row['email']} | CIL: {row['cil']} | {row['status']}\n"

            self.tabela.insert(tk.END, linha)

    def exportar\_excel(self):

        if self.df.empty:

            messagebox.showwarning("Aviso", "Nenhum dado para exportar.")

            return

        try:

            file\_path = filedialog.asksaveasfilename(defaultextension=".xlsx", filetypes=[("Excel", "\*.xlsx")])

            if not file\_path:

                return

            with pd.ExcelWriter(file\_path, engine='openpyxl') as writer:

                self.df.to\_excel(writer, index=False, sheet\_name="Relatório")

            messagebox.showinfo("Sucesso", f"Arquivo salvo: {file\_path}")

        except Exception as e:

            messagebox.showerror("Erro", f"Erro ao exportar: {e}")

    def exportar\_pdf(self):

        if self.df.empty:

            messagebox.showwarning("Aviso", "Nenhum dado para exportar.")

            return

        try:

            file\_path = filedialog.asksaveasfilename(defaultextension=".pdf", filetypes=[("PDF", "\*.pdf")])

            if not file\_path:

                return

            buffer = BytesIO()

            c = canvas.Canvas(buffer, pagesize=A4)

            largura, altura = A4

            y = altura - 40

            c.setFont("Helvetica", 10)

            for \_, row in self.df.iterrows():

                linha = f"{row['data\_envio'].strftime('%d/%m/%Y %H:%M')} | {row['nome']} | {row['email']} | CIL: {row['cil']} | {row['status']}"

                c.drawString(40, y, linha)

                y -= 15

                if y < 50:

                    c.showPage()

                    c.setFont("Helvetica", 10)

                    y = altura - 40

            c.save()

            with open(file\_path, "wb") as f:

                f.write(buffer.getvalue())

            messagebox.showinfo("Sucesso", f"PDF salvo: {file\_path}")

        except Exception as e:

            messagebox.showerror("Erro", f"Erro ao gerar PDF: {e}")

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

    app = App()

    app.mainloop()