import streamlit as st

import pandas as pd

import io

import os

import time

from datetime import datetime

from googleapiclient.discovery import build

from googleapiclient.http import MediaIoBaseDownload, MediaIoBaseUpload, MediaFileUpload

from google.oauth2 import service\_account

from fpdf import FPDF

# CONFIGURACIÓN DE CONEXIÓN

SERVICE\_ACCOUNT\_FILE = 'C:/Users/user/Desktop/APP DIRECCION/CREDENCIALES/semaforo-direccion-c20318ce5a60.json'

SCOPES = ['https://www.googleapis.com/auth/drive']

credentials = service\_account.Credentials.from\_service\_account\_file(SERVICE\_ACCOUNT\_FILE, scopes=SCOPES)

service = build('drive', 'v3', credentials=credentials)

# ID carpeta principal COMPARTIDO

ID\_CARPETA\_PADRE = '1Sh2Pt\_ZsKNrRz6GM6NbON0ICapYovCyS'

ID\_BASE\_ASIGNACIONES = '1XhxVi0YRCfZmeqEgKaJJo6SSbBq7UVgm'

# CONFIGURACIÓN APP

st.set\_page\_config(page\_title="App Semáforos Web + Closers", page\_icon="🚦", layout="wide")

st.sidebar.title("🚦 Menú Principal")

st.markdown("""

<style>

body {background-color: #f5f7fa;}

div.stButton > button {

color: white;

background-color: #0073e6;

border-radius: 8px;

height: 3em;

width: 100%;

font-size: 1.1em;

}

</style>

""", unsafe\_allow\_html=True)

# FUNCIONES AUXILIARES

def buscar\_id\_carpeta(nombre\_carpeta):

query = f"mimeType = 'application/vnd.google-apps.folder' and name = '{nombre\_carpeta}' and '{ID\_CARPETA\_PADRE}' in parents"

resultados = service.files().list(q=query, fields="files(id, name)").execute()

archivos = resultados.get('files', [])

return archivos[0]['id'] if archivos else None

def crear\_carpeta\_closer(nombre\_closer):

nombre\_carpeta = f"COMPARTIDO & {nombre\_closer}"

carpeta\_id = buscar\_id\_carpeta(nombre\_carpeta)

if not carpeta\_id:

file\_metadata = {

'name': nombre\_carpeta,

'mimeType': 'application/vnd.google-apps.folder',

'parents': [ID\_CARPETA\_PADRE]

}

folder = service.files().create(body=file\_metadata, fields='id').execute()

st.success(f"📂 Carpeta creada: {nombre\_carpeta}")

return folder.get('id')

return carpeta\_id

def mover\_archivo\_a\_carpeta(nombre\_closer, df\_clientes):

carpeta\_id = crear\_carpeta\_closer(nombre\_closer)

excel\_buffer = io.BytesIO()

with pd.ExcelWriter(excel\_buffer, engine='openpyxl') as writer:

df\_clientes.to\_excel(writer, index=False)

excel\_buffer.seek(0)

media = MediaIoBaseUpload(excel\_buffer, mimetype='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet', resumable=True)

file\_metadata = {

'name': f"Asignaciones\_{nombre\_closer}.xlsx",

'parents': [carpeta\_id]

}

service.files().create(body=file\_metadata, media\_body=media, fields='id').execute()

def listar\_archivos\_en\_carpeta(carpeta\_id):

query = f"'{carpeta\_id}' in parents and trashed = false"

resultados = service.files().list(q=query, fields="files(id, name)").execute()

return resultados.get('files', [])

def buscar\_semaforo\_en\_carpeta(id\_carpeta):

query = f"mimeType != 'application/vnd.google-apps.folder' and name contains 'SEMAFORO' and name contains '.xlsm' and '{id\_carpeta}' in parents"

resultados = service.files().list(q=query, fields="files(id, name)").execute()

archivos = resultados.get('files', [])

if archivos:

return archivos[0]['id'], archivos[0]['name']

return None, None

def descargar\_archivo(file\_id):

try:

request = service.files().get\_media(fileId=file\_id)

fh = io.BytesIO()

downloader = MediaIoBaseDownload(fh, request)

done = False

while not done:

status, done = downloader.next\_chunk()

fh.seek(0)

return pd.read\_excel(fh)

except Exception:

return pd.DataFrame()

def descargar\_base\_asignaciones():

try:

request = service.files().get\_media(fileId=ID\_BASE\_ASIGNACIONES)

fh = io.BytesIO()

downloader = MediaIoBaseDownload(fh, request)

done = False

while not done:

status, done = downloader.next\_chunk()

fh.seek(0)

return pd.read\_excel(fh)

except Exception:

return pd.DataFrame()

def guardar\_base\_asignaciones(df):

excel\_buffer = io.BytesIO()

with pd.ExcelWriter(excel\_buffer, engine='openpyxl') as writer:

df.to\_excel(writer, index=False)

excel\_buffer.seek(0)

media = MediaIoBaseUpload(excel\_buffer, mimetype='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet', resumable=True)

service.files().update(fileId=ID\_BASE\_ASIGNACIONES, media\_body=media).execute()

def limpiar\_texto(texto):

if str(texto).strip() == '✅':

return 'SI'

elif str(texto).strip() == '❌':

return 'NO'

elif pd.isna(texto) or str(texto).lower() == 'nan':

return ''

else:

return ''.join(c for c in str(texto) if ord(c) < 256)

def generar\_pdf(df, nombre\_archivo):

pdf = FPDF(orientation='L', unit='mm', format='A4')

pdf.add\_page()

pdf.set\_font("Arial", size=8)

page\_width = pdf.w - 2 \* pdf.l\_margin

col\_width = page\_width / (len(df.columns) + 0.5)

row\_height = pdf.font\_size \* 2

for col in df.columns:

pdf.cell(col\_width, row\_height, limpiar\_texto(col), border=1)

pdf.ln(row\_height)

for index, row in df.iterrows():

for item in row:

pdf.cell(col\_width, row\_height, limpiar\_texto(item), border=1)

pdf.ln(row\_height)

pdf.output(nombre\_archivo)

# (continúa igual como antes desde la parte de carga inicial...)

# --- CARGA INICIAL ---

if 'datos\_cargados' not in st.session\_state:

with st.spinner('⏳ Cargando datos iniciales...'):

st.session\_state['semaforos\_carpetas'] = {

"SEMAFORO ELCHE 2.0": "COMPARTIDO ELCHE 2.0",

"SEMAFORO ELCHE 3.0": "COMPARTIDO ELCHE 3.0",

"SEMAFORO ELCHE 4.0": "COMPARTIDO ELCHE 4.0",

"SEMAFORO VIGO 1.0": "COMPARTIDO VIGO 1.0",

"SEMAFORO VIGO 2.0": "COMPARTIDO VIGO 2.0",

"SEMAFORO VIGO 3.0": "COMPARTIDO VIGO 3.0",

"SEMAFORO LEON 1.0": "COMPARTIDO LEON 1.0"

}

st.session\_state['clientes\_rojos'] = []

for nombre\_semaforo, nombre\_carpeta in st.session\_state['semaforos\_carpetas'].items():

carpeta\_id = buscar\_id\_carpeta(nombre\_carpeta)

if carpeta\_id:

archivos = listar\_archivos\_en\_carpeta(carpeta\_id)

archivo\_semaforo = next((a for a in archivos if a['name'].startswith("SEMAFORO") and a['name'].endswith(".xlsm")), None)

if archivo\_semaforo:

df = descargar\_archivo(archivo\_semaforo['id'])

if not df.empty:

if 'DÍA' in df.columns:

df['DÍA'] = pd.to\_datetime(df['DÍA'], errors='coerce').dt.strftime('%d/%m/%Y')

df\_rojos = df[df["SEMAFORO"] == "ROJO"]

st.session\_state['clientes\_rojos'].append(df\_rojos)

if st.session\_state['clientes\_rojos']:

st.session\_state['clientes\_rojos'] = pd.concat(st.session\_state['clientes\_rojos'], ignore\_index=True)

st.session\_state['clientes\_rojos']["CLOSER ASIGNADO"] = ""

df\_asignaciones = descargar\_base\_asignaciones()

if not df\_asignaciones.empty:

if 'DÍA' in df\_asignaciones.columns:

df\_asignaciones['DÍA'] = pd.to\_datetime(df\_asignaciones['DÍA'], errors='coerce').dt.strftime('%d/%m/%Y')

if 'ESTADO' in df\_asignaciones.columns:

df\_asignaciones['ESTADO'] = df\_asignaciones['ESTADO'].astype(str)

df\_asignaciones['selected'] = False

st.session\_state['seguimiento'] = df\_asignaciones

else:

st.session\_state['seguimiento'] = pd.DataFrame()

st.session\_state['datos\_cargados'] = True

# --- VARIABLES EN MEMORIA ---

semaforos\_carpetas = st.session\_state['semaforos\_carpetas']

clientes\_rojos = st.session\_state['clientes\_rojos']

seguimiento = st.session\_state['seguimiento']

# --- MENÚ PRINCIPAL ---

opcion = st.sidebar.radio("Ir a:", ("🏠 Inicio", "📂 Semáforos Comerciales", "🎯 Asignar Closers", "📋 Seguimiento de Asignaciones"))

# --- SECCIÓN INICIO ---

if opcion == "🏠 Inicio":

st.title("🚦 Bienvenido a la App de Semáforos Web")

st.success("¡Del contacto, al cierre! 🚀")

# --- SECCIÓN SEMÁFOROS COMERCIALES ---

elif opcion == "📂 Semáforos Comerciales":

st.title("📂 Semáforos Comerciales")

semaforo\_elegido = st.selectbox("Selecciona un Semáforo:", list(semaforos\_carpetas.keys()))

carpeta\_nombre = semaforos\_carpetas[semaforo\_elegido]

carpeta\_id = buscar\_id\_carpeta(carpeta\_nombre)

if carpeta\_id:

with st.spinner('⏳ Descargando semáforo...'):

file\_id, file\_name = buscar\_semaforo\_en\_carpeta(carpeta\_id)

if file\_id:

df\_semaforo = descargar\_archivo(file\_id)

st.dataframe(df\_semaforo, use\_container\_width=True)

else:

st.warning(f"⚠️ No se encontró archivo SEMÁFORO en {carpeta\_nombre}.")

else:

st.warning(f"⚠️ No se encontró carpeta {carpeta\_nombre}.")

# --- SECCIÓN ASIGNAR CLOSERS ---

elif opcion == "🎯 Asignar Closers":

st.title("🎯 Asignar Closers")

if clientes\_rojos.empty:

st.info("ℹ️ No hay clientes ROJOS pendientes.")

else:

edited\_df = st.data\_editor(

clientes\_rojos,

use\_container\_width=True,

num\_rows="dynamic",

)

if st.button("✅ Asignar Clientes"):

with st.spinner('⏳ Asignando clientes...'):

nuevas\_asignaciones = []

closers\_creados = {}

for \_, row in edited\_df.iterrows():

closer = str(row["CLOSER ASIGNADO"]).strip()

if closer:

nueva = row.to\_dict()

nueva["CLOSER"] = str(row["CLOSER ASIGNADO"]).strip() # 🔥 Copiamos el nombre del Closer

nueva["ESTADO"] = ""

nuevas\_asignaciones.append(nueva)

if closer not in closers\_creados:

closers\_creados[closer] = []

closers\_creados[closer].append(nueva)

if nuevas\_asignaciones:

# Guardar asignaciones nuevas

df\_nuevas = pd.DataFrame(nuevas\_asignaciones)

df\_existente = descargar\_base\_asignaciones()

df\_final = pd.concat([df\_existente, df\_nuevas], ignore\_index=True)

guardar\_base\_asignaciones(df\_final)

# Actualizar seguimiento en pantalla

df\_final['selected'] = False

st.session\_state['seguimiento'] = df\_final

# Crear carpetas y mover archivos de Closers

for closer, clientes in closers\_creados.items():

mover\_archivo\_a\_carpeta(closer, pd.DataFrame(clientes))

# Limpiar clientes ROJOS que ya fueron asignados

clientes\_rojos = edited\_df[edited\_df["CLOSER ASIGNADO"].str.strip() == ""]

st.session\_state['clientes\_rojos'] = clientes\_rojos

st.success("✅ Clientes asignados correctamente.")

st.rerun()

else:

st.warning("⚠️ No has asignado ningún Closer.")

elif opcion == "📋 Seguimiento de Asignaciones":

st.title("📋 Seguimiento de Asignaciones")

if seguimiento.empty:

st.info("ℹ️ No hay asignaciones registradas todavía.")

else:

# 📍 Nuevo: Filtro para mostrar solo activos

mostrar\_pendientes = st.checkbox("👀 Mostrar solo asignaciones activas", value=True)

edited\_df = st.session\_state['seguimiento'].copy()

# Si queremos ver solo los activos

if mostrar\_pendientes:

if "ESTADO" in edited\_df.columns:

edited\_df = edited\_df[edited\_df["ESTADO"].str.upper() != "TERMINADO"]

# Editor de datos

edited\_df = st.data\_editor(

edited\_df,

column\_order=[

"selected", "CALL", "COMERCIAL", "CLIENTE", "DÍA",

"F2025", "F2026", "HL", "VIGILANCIA",

"IMPLANT", "DENUNCIAS", "SEMAFORO",

"NOTAS", "CLOSER", "ESTADO"

],

use\_container\_width=True,

num\_rows="dynamic",

)

col1, col2 = st.columns([1, 1])

with col1:

if st.button("💾 Guardar cambios"):

with st.spinner('⏳ Guardando asignaciones...'):

df\_guardar = edited\_df.copy()

# Asegurar que todas las columnas necesarias existen

columnas\_necesarias = [

"CALL", "COMERCIAL", "CLIENTE", "DÍA", "F2025", "F2026",

"HL", "VIGILANCIA", "IMPLANT", "DENUNCIAS", "SEMAFORO",

"NOTAS", "CLOSER", "ESTADO"

]

for col in columnas\_necesarias:

if col not in df\_guardar.columns:

df\_guardar[col] = ""

df\_guardar["selected"] = False # reset columna seleccionados

st.session\_state['seguimiento'] = df\_guardar

guardar\_base\_asignaciones(df\_guardar)

st.success("✅ Asignaciones guardadas correctamente.")

st.rerun()

with col2:

if st.button("⬇️ ¿Quieres descragar la información en PDF?"):

with st.spinner('⏳ Generando PDF...'):

nombre\_pdf = "seguimiento\_asignaciones.pdf"

df\_para\_pdf = edited\_df.drop(columns=["selected"])

generar\_pdf(df\_para\_pdf, nombre\_pdf)

with open(nombre\_pdf, "rb") as file:

st.download\_button("Descarga aquí tu PDF", file, file\_name=nombre\_pdf)

# --- FIN DEL CÓDIGO ---