Prompt de Correção Definitiva para `app.py` (Manipulação de ZIP)

Este prompt visa refatorar a função `upload_fotos` em `app.py` para lidar com o upload de arquivos ZIP, descompactá-los, processar a estrutura de pastas e títulos contida no ZIP, e então gerar o relatório Word. Ele também inclui a adaptação da lógica de `auto.py` para o ambiente web.

Instruções:

1. **Abra o arquivo `app.py` no seu ambiente Replit.**
2. **Substitua o conteúdo da função `upload_fotos` pelo código fornecido abaixo.**
3. **Certifique-se de que as dependências necessárias (`zipfile`, `os`, `tempfile`, `shutil`) estão importadas no início do `app.py`.**
4. **Verifique se `word_utils.py` e `interface.py` (embora `interface.py` não seja mais usado para seleção de arquivos na web) estão presentes no seu projeto.**

```python
 import os
import tempfile
import zipfile
import shutil
from flask import Flask, render_template, request, redirect, url_for, flash,
send_file
from werkzeug.utils import secure_filename
from docx import Document

# Importe inserir_conteudo de word_utils.py
from word_utils import inserir_conteudo

app = Flask(__name__)
app.secret_key = 'sua_chave_secreta_aqui' # Mude para uma chave secreta forte
em produção

ORDEM_PASTAS = ["- Área externa", "- Área interna", "- Segundo piso"]
UPLOAD_FOLDER = 'uploads'
MODELOS_FOLDER = '01 - MODELOS - auto'

if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

@app.route('/')
def index():
    modelos = []
    if os.path.exists(MODELOS_FOLDER):
        modelos = [f for f in os.listdir(MODELOS_FOLDER) if
f.endswith('.docx')]
    return render_template('index.html', modelos=modelos)

@app.route('/upload_fotos', methods=['POST'])
def upload_fotos():
    if 'pasta_fotos' not in request.files:
        flash('Nenhum arquivo enviado.')
        return redirect(url_for('index'))

    uploaded_file = request.files['pasta_fotos']
    modelo_selecionado = request.form.get('modelo')

    if uploaded_file.filename == '':
        flash('Nenhum arquivo selecionado.')
        return redirect(url_for('index'))

    if not modelo_selecionado:
        flash('Nenhum modelo de relatório selecionado.')
        return redirect(url_for('index'))

    modelo_path = os.path.join(MODELOS_FOLDER,
secure_filename(modelo_selecionado))
    if not os.path.exists(modelo_path):
        flash(f'Modelo "{modelo_selecionado}" não encontrado.')
        return redirect(url_for('index'))

    # Capturar todos os campos do formulário
    campos = {
        'nome': request.form.get('nome', ''),
        'ctr': request.form.get('ctr', ''),
        'os': request.form.get('os', ''),
        'elb': 'Ygor Augusto Fernandes',  # Valor fixo
        'data_elb': request.form.get('data_elb', ''),
```

```
        'ag': request.form.get('ag', ''),
        'nome_dependencia': request.form.get('nome_dependencia', ''),
        'uf': request.form.get('uf', ''),
        'tipo': request.form.get('tipo', ''),
        'data_att': request.form.get('data_att', ''),
    }

    temp_dir = None
    try:
        # Criar um diretório temporário para descompactar o ZIP
        temp_dir = tempfile.mkdtemp()
        zip_path = os.path.join(temp_dir,
secure_filename(uploaded_file.filename))
        uploaded_file.save(zip_path)

        # Descompactar o arquivo ZIP
        with zipfile.ZipFile(zip_path, 'r') as zip_ref:
            zip_ref.extractall(temp_dir)

        # Encontrar a pasta raiz dentro do ZIP (assumindo que há uma única
pasta raiz)
        # Ou, se o ZIP contiver arquivos diretamente, temp_dir será a pasta
raiz
        extracted_contents = os.listdir(temp_dir)
        pasta_raiz_processamento = temp_dir
        if len(extracted_contents) == 1 and
os.path.isdir(os.path.join(temp_dir, extracted_contents[0])):
            pasta_raiz_processamento = os.path.join(temp_dir,
extracted_contents[0])

        # --- Lógica de Leitura de Pastas e Títulos (adaptada de auto.py) ---
        conteudo = []
        nome_pasta_raiz_original =
os.path.basename(pasta_raiz_processamento.strip(os.sep))

        for root, dirs, files in os.walk(pasta_raiz_processamento):
            # Ordenar subdiretórios apenas no nível da pasta raiz de
processamento
            if root == pasta_raiz_processamento:
                dirs.sort(key=lambda x: (ORDEM_PASTAS.index(x) if x in
ORDEM_PASTAS else len(ORDEM_PASTAS), x))

            # Calcular o nível da pasta em relação à pasta raiz de
processamento
            # Ajuste para garantir que o nível 0 seja a pasta raiz de
processamento
            rel_path = os.path.relpath(root, pasta_raiz_processamento)
            path_parts = []
            if rel_path != '.': # Evita que o diretório raiz seja dividido em
partes vazias
                path_parts = rel_path.split(os.sep)

            nivel = len(path_parts)

            # Adicionar títulos de pastas
            if nivel == 0 and rel_path == '.': # A própria pasta raiz de
processamento
                # Não adicionamos a pasta raiz como título, pois o relatório é
sobre ela
                pass
            elif nivel == 1:
                conteudo.append(path_parts[0])
```

```
            elif nivel == 2:
                conteudo.append(f"»{path_parts[1]}")
            elif nivel == 3:
                conteudo.append(f"»»{path_parts[2]}")
            else:
                # Para níveis mais profundos, ou se houver um erro na estrutura
                if path_parts: # Garante que path_parts não está vazio
                    conteudo.append(f"»»»- {path_parts[-1]}")

            # Adicionar caminhos das imagens
            arquivos_imagens = [
                os.path.join(root, file)
                for file in files
                if file.lower().endswith(('.png', '.jpg', '.jpeg'))
            ]
            arquivos_imagens.sort() # Garante uma ordem consistente das imagens
            conteudo.extend(arquivos_imagens)

        # --- Fim da Lógica de Leitura de Pastas e Títulos ---

        # Gerar o nome do arquivo de saída
        output_filename = f"RELATÓRIO FOTOGRÁFICO - {nome_pasta_raiz_original}
- LEVANTAMENTO PREVENTIVO.docx"
        output_path = os.path.join(UPLOAD_FOLDER, output_filename)

        # Inserir conteúdo no modelo
        # A função inserir_conteudo agora também aceitará os campos do
formulário
        inserir_conteudo(modelo_path, conteudo, output_path, campos)

        flash('Relatório gerado com sucesso!')
        return send_file(output_path, as_attachment=True,
download_name=output_filename)

    except zipfile.BadZipFile:
        flash('O arquivo enviado não é um arquivo ZIP válido.')
        return redirect(url_for('index'))
    except Exception as e:
        flash(f'Ocorreu um erro ao processar o arquivo: {e}')
        return redirect(url_for('index'))
    finally:
        # Limpar o diretório temporário
        if temp_dir and os.path.exists(temp_dir):
            shutil.rmtree(temp_dir)
```

Adaptação em `word_utils.py`

Você precisará adaptar a função `inserir_conteudo` em `word_utils.py` para aceitar os `campos` do formulário e usá-los para preencher o documento. Isso envolve adicionar um novo parâmetro à função e usar a funcionalidade de substituição de texto do `python-docx`.

Instruções:

1. **Abra o arquivo `word_utils.py` no seu ambiente Replit.**
2. **Modifique a assinatura da função `inserir_conteudo` para incluir `campos` como um argumento.**
3. **Adicione a lógica para substituir os placeholders no documento Word pelos valores de `campos`.**

```python
# word_utils.py

import os
from docx import Document
from docx.shared import Cm, Pt
from docx.enum.text import WD_PARAGRAPH_ALIGNMENT, WD_BREAK
from PIL import Image, UnidentifiedImageError

PASTAS_TEXTO_NORMAL = ["- Detalhes", "- Vista ampla"]

def aplicar_estilo(run, tamanho, negrito=False):
    run.font.name = "Arial"
    run.font.size = Pt(tamanho)
    run.bold = negrito

def inserir_conteudo(modelo_path, conteudo, output_path, campos=None): #
Adicione campos=None
    doc = Document(modelo_path)
    contador_imagens = 0
    conteudo_processado = False
    paragrafo_insercao_index = None

    # --- Nova Lógica: Substituir placeholders com dados do formulário ---
    if campos:
        for p in doc.paragraphs:
            for key, value in campos.items():
                # Use um formato de placeholder que você espera no seu modelo
Word, por exemplo, {{NOME}}
                # Certifique-se de que os nomes das chaves em 'campos'
correspondem aos seus placeholders
                if f'{{{{{key.upper()}}}}}' in p.text: # Ex: {{NOME}}, {{CTR}}
                    p.text = p.text.replace(f'{{{{{key.upper()}}}}}',
str(value))
        for table in doc.tables:
            for row in table.rows:
                for cell in row.cells:
                    for p in cell.paragraphs:
                        for key, value in campos.items():
                            if f'{{{{{key.upper()}}}}}' in p.text:
                                p.text =
p.text.replace(f'{{{{{key.upper()}}}}}', str(value))
    # --- Fim da Nova Lógica ---

    for i, paragrafo in enumerate(doc.paragraphs):
        if "{{start_here}}" in paragrafo.text:
            paragrafo_insercao_index = i
            break

    if paragrafo_insercao_index is None:
        print("Marca '{{start_here}}' não encontrada no modelo.")
        return contador_imagens

    conteudo_invertido = list(reversed(conteudo))

    for item in conteudo_invertido:
        if isinstance(item, str):
            titulo = item.replace("»", "").strip() + ":"
            nivel = item.count("»")

            p =
doc.paragraphs[paragrafo_insercao_index].insert_paragraph_before('')
```

```
            run = p.add_run(titulo)

            if any(pasta in titulo for pasta in PASTAS_TEXTO_NORMAL):
                aplicar_estilo(run, 11, negrito=True)
                p.alignment = WD_PARAGRAPH_ALIGNMENT.JUSTIFY
            elif nivel == 0:
                p.style = 'Heading 1'
            elif nivel == 1:
                p.style = 'Heading 2'
            elif nivel == 2:
                p.style = 'Heading 3'
            else:
                aplicar_estilo(run, 12, negrito=True)

        elif isinstance(item, dict) and 'image_path' in item: # Assume que
imagens são passadas como dicionários
            image_path = item['image_path']
            caption = item.get('caption', '')
            try:
                img = Image.open(image_path)
                width, height = img.size
                aspect_ratio = width / height

                # Definir largura máxima para a imagem no Word (ex: 15 cm)
                max_width_cm = 15
                max_width_emu = int(max_width_cm * 360000)

                # Calcular altura proporcional
                new_width_emu = max_width_emu
                new_height_emu = int(new_width_emu / aspect_ratio)

                p =
doc.paragraphs[paragrafo_insercao_index].insert_paragraph_before('')
                r = p.add_run()
                r.add_picture(image_path, width=new_width_emu,
height=new_height_emu)
                p.alignment = WD_PARAGRAPH_ALIGNMENT.CENTER

                if caption:
                    caption_p =
doc.paragraphs[paragrafo_insercao_index].insert_paragraph_before('')
                    caption_run = caption_p.add_run(caption)
                    aplicar_estilo(caption_run, 9) # Estilo para legenda
                    caption_p.alignment = WD_PARAGRAPH_ALIGNMENT.CENTER

                # Adicionar quebra de página após cada imagem para melhor
formatação
                p_break =
doc.paragraphs[paragrafo_insercao_index].insert_paragraph_before('')
                run_break = p_break.add_run()
                run_break.add_break(WD_BREAK.PAGE)

                contador_imagens += 1
            except UnidentifiedImageError:
                print(f"Erro: Não foi possível identificar a imagem em
{image_path}")
            except FileNotFoundError:
                print(f"Erro: Imagem não encontrada em {image_path}")
            except Exception as e:
                print(f"Erro ao inserir imagem {image_path}: {e}")
```

```
    doc.save(output_path)
    return contador_imagens
```

Adaptação em `templates/index.html`

Você precisará garantir que seu formulário HTML em `templates/index.html` esteja configurado para enviar um único arquivo (o ZIP) e que os campos de texto para `nome`, `ctr`, `os`, etc., estejam presentes.

Instruções:

1. **Abra o arquivo `templates/index.html` no seu ambiente Replit.**
2. **Certifique-se de que o formulário tenha `enctype="multipart/form-data"` e um input de arquivo para o ZIP.**
3. **Adicione os campos de texto para os dados do formulário.**

```
<!-- templates/index.html (Exemplo simplificado) -->

<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Gerador de Relatórios</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        form { max-width: 600px; margin: auto; padding: 20px; border: 1px solid
#ccc; border-radius: 8px; }
        div { margin-bottom: 10px; }
        label { display: block; margin-bottom: 5px; font-weight: bold; }
        input[type="text"], input[type="file"], select { width: calc(100% -
22px); padding: 10px; border: 1px solid #ddd; border-radius: 4px; }
        button { padding: 10px 20px; background-color: #007bff; color: white;
border: none; border-radius: 4px; cursor: pointer; }
        button:hover { background-color: #0056b3; }
        .flash { background-color: #ffe0b2; padding: 10px; border-radius: 4px;
margin-bottom: 10px; border: 1px solid #ffcc80; }
    </style>
</head>
<body>
    <h1>Gerador de Relatórios Fotográficos</h1>

    {% with messages = get_flashed_messages() %}
        {% if messages %}
            <ul class="flash">
                {% for message in messages %}
                    <li>{{ message }}</li>
                {% endfor %}
            </ul>
        {% endif %}
    {% endwith %}

    <form action="{{ url_for('upload_fotos') }}" method="post"
enctype="multipart/form-data">
        <div>
            <label for="pasta_fotos">Selecione o arquivo ZIP com as fotos:
</label>
            <input type="file" id="pasta_fotos" name="pasta_fotos"
accept=".zip" required>
        </div>

        <div>
            <label for="modelo">Selecione o Modelo do Relatório:</label>
            <select id="modelo" name="modelo" required>
                <option value="">-- Selecione um modelo --</option>
                {% for modelo in modelos %}
                    <option value="{{ modelo }}">{{ modelo }}</option>
                {% endfor %}
            </select>
        </div>

        <!-- Campos do formulário -->
        <div>
            <label for="nome">Nome:</label>
            <input type="text" id="nome" name="nome" placeholder="Nome do
Projeto/Cliente">
        </div>
```

```
        <div>
            <label for="ctr">CTR:</label>
            <input type="text" id="ctr" name="ctr" placeholder="Número do CTR">
        </div>
        <div>
            <label for="os">OS:</label>
            <input type="text" id="os" name="os" placeholder="Número da OS">
        </div>
        <!-- ELB é fixo em app.py -->
        <div>
            <label for="data_elb">Data ELB:</label>
            <input type="date" id="data_elb" name="data_elb">
        </div>
        <div>
            <label for="ag">AG:</label>
            <input type="text" id="ag" name="ag" placeholder="Número da AG">
        </div>
        <div>
            <label for="nome_dependencia">Nome da Dependência:</label>
            <input type="text" id="nome_dependencia" name="nome_dependencia"
placeholder="Nome da Dependência">
        </div>
        <div>
            <label for="uf">UF:</label>
            <input type="text" id="uf" name="uf" placeholder="Estado (UF)">
        </div>
        <div>
            <label for="tipo">Tipo:</label>
            <input type="text" id="tipo" name="tipo" placeholder="Tipo de
Levantamento">
        </div>
        <div>
            <label for="data_att">Data Atual:</label>
            <input type="date" id="data_att" name="data_att">
        </div>

        <button type="submit">Gerar Relatório</button>
    </form>
</body>
</html>
```

Observações Finais

- **Estrutura do ZIP**: O ZIP deve conter a estrutura de pastas que você deseja que seja refletida no relatório. Por exemplo, se você tem `MinhasFotos/Area Externa/Detalhes/foto1.jpg`, o ZIP deve ser criado a partir da pasta `MinhasFotos`.
- **Placeholders no Modelo Word**: Certifique-se de que seu arquivo de modelo `.docx` (`01 - MODELOS - auto/seu_modelo.docx`) contém os placeholders exatos que você deseja substituir, usando o formato `{{NOME}}`, `{{CTR}}`, `{{OS}}`, etc., conforme definido na lógica de substituição em `word_utils.py`.

- **`interface.py` e `auto.py`**: Com essas mudanças, `interface.py` (que usa `tkinter`) não será mais usado no fluxo web. A lógica essencial de `auto.py` para percorrer pastas foi integrada diretamente em `app.py`. Você pode manter `auto.py` e `interface.py` no repositório, mas eles não serão executados pela aplicação Flask.
- **Instalação de Dependências**: No Replit, execute `pip install -r requirements.txt` para garantir que todas as bibliotecas necessárias (Flask, python-docx, Pillow, etc.) estejam instaladas.

Prompt de Correção Definitiva para `interface.py`

Para a versão web da aplicação, o arquivo `interface.py` (que utiliza `tkinter` para seleção de pastas e arquivos) **não requer nenhuma alteração**. Suas funcionalidades foram substituídas pela interface web (`templates/index.html`) e pela lógica de upload de arquivos em `app.py`.

Você pode manter o arquivo `interface.py` no seu repositório, mas ele não será utilizado pela aplicação Flask.