**PROMPT**

Gera uma API em Python que recebe como input um ficheiro de código e a API faz o code review desse código. Para além do code review deve também auditar issues de segurança a vulnerabilidades. A API deve retornar um ficheiro ASCII com a lista de erros, e vulnerabilidades detetadas. O outuput deve ter o seguinte formato, sendo um erro/vulnerabilidade por linha:

<#linha de código>|<Criticidade>|<Descrição do Erro>|<Ação necessária para resolução>

o nível de criticidade deve pertencer à seguinte lista:

CRI: Crítico - Se vulnerabilidade de segurança Crítica ou erro de código que implique que a aplicação termine ou bloqueie

MAI : Major - Se provocar redução de consumo de recursos ou alguma vulnerabilidade MAJOR

MEN - Minor - Se vulnerabilidade de Segurança Minor ou código que afete ligeiramente a *perfomance* do código

WRN: Warning -Se o for somente um aviso sobre utilização de funções deprecated ou em fase de descontinuação ou propostas de melhoria do código.

**RESPOSTA**

from flask import Flask, request, render\_template\_string, send\_file, send\_from\_directory

import os

app = Flask(\_\_name\_\_)

def analisar\_codigo(codigo):

    resultado = []

    linhas = codigo.split('\n')

    for numero\_linha, linha in enumerate(linhas, start=1):

        if 'eval(' in linha:

            resultado.append(f"{numero\_linha}|CRI|Uso da função eval pode ser inseguro|Substituir eval por alternativas mais seguras")

        elif 'exec(' in linha:

            resultado.append(f"{numero\_linha}|CRI|Uso da função exec pode ser inseguro|Evitar o uso de exec para prevenir vulnerabilidades")

        elif 'import \*' in linha:

            resultado.append(f"{numero\_linha}|MAI|Importação global pode levar a conflitos|Importar apenas os módulos necessários")

        elif 'print(' in linha:

            resultado.append(f"{numero\_linha}|WRN|Uso da função print encontrado|Utilizar logging em vez de print para produção")

        # Adicionar mais regras de análise conforme necessário

    return resultado

@app.route('/', methods=['GET'])

def index():

    return render\_template\_string('''

    <!DOCTYPE html>

    <html lang="pt">

    <head>

        <meta charset="UTF-8">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <title>Analisador de Código</title>

        <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

    </head>

    <body>

        <div class="container mt-5">

            <h1 class="text-center">Analisador de Código</h1>

            <form action="/analisar" method="post" enctype="multipart/form-data" class="mt-4">

                <div class="form-group">

                    <label for="ficheiro">Escolher Ficheiro</label>

                    <input type="file" name="ficheiro" id="ficheiro" class="form-control-file">

                </div>

                <button type="submit" class="btn btn-primary">Analisar Código</button>

            </form>

        </div>

        <script>

            document.getElementById('ficheiro').onchange = function () {

                var fileName = this.value.split('\\\\').pop();

                if (fileName) {

                    this.nextElementSibling.innerText = fileName;

                } else {

                    this.nextElementSibling.innerText = 'Nenhum ficheiro escolhido';

                }

            };

            // Alterar o texto do botão de upload de ficheiro para Português

            document.addEventListener('DOMContentLoaded', function() {

                var fileInput = document.getElementById('ficheiro');

                var fileLabel = fileInput.nextElementSibling;

                fileLabel.innerText = 'Escolher Ficheiro';

            });

        </script>

    </body>

    </html>

    ''')

@app.route('/analisar', methods=['POST'])

def analisar():

    ficheiro = request.files['ficheiro']

    nome\_ficheiro = ficheiro.filename

    codigo = ficheiro.read().decode('utf-8')

    resultado = analisar\_codigo(codigo)

    primeiras\_linhas = '\n'.join(resultado[:10]) if resultado else "Tudo OK. Sem erros e vulnerabilidades detetadas!"

    # Salvar o arquivo completo para visualização

    caminho\_arquivo = os.path.join('uploads', 'resultado.txt')

    with open(caminho\_arquivo, 'w', encoding='utf-8') as f:

        f.write(f"Nome do ficheiro analisado: {nome\_ficheiro}\n")

        if resultado:

            for linha in resultado:

                f.write(linha + '\n')

        else:

            f.write("Tudo OK. Sem erros e vulnerabilidades detetadas!\n")

    return render\_template\_string('''

    <!DOCTYPE html>

    <html lang="pt">

    <head>

        <meta charset="UTF-8">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <title>Resultado da Análise</title>

        <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

    </head>

    <body>

        <div class="container mt-5">

            <h1 class="text-center">Resultado da Análise</h1>

            <pre class="bg-light p-3">{{ primeiras\_linhas }}</pre>

            <div class="text-center">

                <a href="{{ url\_for('visualizar', filename='resultado.txt') }}" target="\_blank" class="btn btn-primary">Ver Análise Completa</a>

                <a href="{{ url\_for('index') }}" class="btn btn-secondary">Voltar à Página Inicial</a>

            </div>

        </div>

    </body>

    </html>

    ''', primeiras\_linhas=primeiras\_linhas)

@app.route('/visualizar/<filename>', methods=['GET'])

def visualizar(filename):

    caminho\_arquivo = os.path.join('uploads', filename)

    return send\_from\_directory(directory='uploads', path=filename)

if \_\_name\_\_ == '\_\_main\_\_':

    if not os.path.exists('uploads'):

        os.makedirs('uploads')

    app.run(debug=True)