ATIVIDADE PRÁTICA

CRIANDO E CONSUMINDO API

O código a seguir cria uma API para “Produtos”, capaz de cadastrar e listar os produtos de um banco em SQLite.

1) Utilizando a ferramenta DBBrowser (ou outra qualquer) crie um banco SQLite “db-produtos.db” com uma tabela “produtos” com os seguintes campos “idproduto: autoincremento, descrição: string, precocompra, precovenda e datacriacao” para indicar quando o registro foi criado.

2) De acordo com os conceitos de API Rest, implemente os demais métodos/rotas para excluir um registro e alterar.

3) Teste a API utilizando alguma página HTML (criada por você) ou algum código em Python (também criado por você) ou até mesmo utilizando softwares de terceiros como o Postman (pesquise sobre).

```
from flask import Flask
from flask import request
from flask import render_template

from datetime import date

import sqlite3
from sqlite3 import Error


#########################################################
# Instancia da Aplicacao Flask

app = Flask(__name__)


#########################################################
# 1. Cadastrar produtos

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

def cadastrar():

    if request.method == 'POST':

        descricao = request.form['descricao']
        precocompra = request.form['precocompra']
        precovenda = request.form['precovenda']
        datacriacao = date.today()

        mensagem = 'Erro - nao cadastrado'

        if descricao and precocompra and precovenda and datacriacao:
            registro = (descricao, precocompra, precovenda, datacriacao)

            try:
                conn = sqlite3.connect('database/db-produtos.db')

                sql = ''' INSERT INTO produtos(descricao, precocompra,
precovenda, datacriacao)
                              VALUES(?,?,?,?) '''
```

```
                cur = conn.cursor()

                cur.execute(sql, registro)

                conn.commit()

                mensagem = 'Sucesso - cadastrado'

            except Error as e:
                print(e)
            finally:
                conn.close()

    return render_template('cadastrar.html')


##########################################################
# 2. Listar produtos

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

def listar():
    try:
        conn = sqlite3.connect('database/db-produtos.db')

        sql = '''SELECT * FROM produtos'''

        cur = conn.cursor()

        cur.execute(sql)

        registros = cur.fetchall()

        return render_template('listar.html', regs=registros)

    except Error as e:
        print(e)
    finally:
        conn.close()



##########################################################
# Rota de Erro

@app.errorhandler(404)
def pagina_nao_encontrada(e):
    return render_template('404.html'), 404



##########################################################
# Execucao da Aplicacao

if __name__ == '__main__':
    app.run()
```