at_S9_A1_SL06_backend

Roteiro de Atividade Prática

Nome: _____ Turma: _____

Título da atividade: APIs RESTful

Objetivo:
Compreender e implementar APIs RESTful utilizando Python, aplicando conceitos de rotas, métodos HTTP e tratamento de dados, explorando a arquitetura REST para desenvolver serviços web escaláveis e eficientes.

Conceito técnico: APIs RESTful
APIs RESTful são interfaces de comunicação que seguem os princípios arquiteturais do REST (Representational State Transfer). Elas utilizam métodos HTTP como **GET**, **POST**, **PUT** e **DELETE** para manipular recursos em um servidor. Cada recurso é identificado por uma **URI** (Uniform Resource Identifier) e é representado em formatos como JSON ou XML.
Os princípios fundamentais de uma API RESTful incluem:

- **Statelessness** – Cada requisição de cliente para o servidor deve conter todas as informações necessárias para ser processada;
- **Representações de recursos** – Os recursos são representados em formatos simples e legíveis, como JSON;
- **Uniform interface** – A comunicação entre cliente e servidor segue padrões bem definidos de uso de métodos HTTP.

Objetivo da atividade:
Implementar uma **API RESTful** para gerenciar produtos cadastrados em um sistema de e-commerce. A API deve listar, adicionar, atualizar e remover produtos utilizando métodos HTTP.

Enunciado:
Você foi encarregado de criar uma API RESTful para gerenciar produtos de um e-commerce. A API deve apresentar as seguintes funcionalidades:

- **GET/produtos** – Listar todos os produtos;

- POST/produtos – Adicionar um novo produto;
- PUT/produtos/{id} – atualizar um produto existente;
- DELETE/produtos/{id} – Remover um produto.

Implemente essa API utilizando **Python** com **Flask**, respeitando os princípios de REST e garantindo que a comunicação entre cliente e servidor siga as boas práticas do padrão.

Código de programação:
Abaixo está um exemplo de como implementar essa API RESTful utilizando **Flask** em Python.

Exemplo em Python com Flask:

```
from flask import Flask, jsonify, request

app = Flask(__name__)

# Dados iniciais (simulação de banco de dados)
produtos = [
    {"id": 1, "nome": "Camiseta", "preco": 50.00},
    {"id": 2, "nome": "Tênis", "preco": 120.00}
]

# Rota para listar todos os produtos (GET)
@app.route('/produtos', methods=['GET'])
def listar_produtos():
    return jsonify(produtos)

# Rota para adicionar um novo produto (POST)
@app.route('/produtos', methods=['POST'])
def adicionar_produto():
    novo_produto = request.get_json()
    produtos.append(novo_produto)
    return jsonify({"mensagem": "Produto adicionado com sucesso!"}), 201

# Rota para atualizar um produto existente (PUT)
```

```
@app.route('/produtos/<int:id>', methods=['PUT'])
def atualizar_produto(id):
    produto = next((p for p in produtos if p['id'] == id), None)
    if produto is None:
        return jsonify({"erro": "Produto não encontrado!"}), 404

    dados_atualizados = request.get_json()
    produto.update(dados_atualizados)
    return jsonify({"mensagem": "Produto atualizado com sucesso!"})

# Rota para remover um produto (DELETE)
@app.route('/produtos/<int:id>', methods=['DELETE'])
def remover_produto(id):
    produto = next((p for p in produtos if p['id'] == id), None)
    if produto is None:
        return jsonify({"erro": "Produto não encontrado!"}), 404

    produtos.remove(produto)
    return jsonify({"mensagem": "Produto removido com sucesso!"})

# Executa a aplicação
if __name__ == '__main__':
    app.run(debug=True)
```

Explicação técnica:

- GET/produtos – Retorna uma lista de todos os produtos, representados em formato JSON. É o método utilizado para leitura de dados;
- POST/produtos – Adiciona um novo produto ao sistema. O corpo da requisição deve conter os dados do produto em formato JSON;
- PUT/produtos/{id} – Atualiza um produto cadastrado, localizando-o pelo ID passado pela URI e recebendo os novos dados via corpo da requisição;
- DELETE/produtos/{id} – Remove um produto do sistema, identificando-o pelo ID fornecido na URI.

Perguntas para conclusão da atividade:

- Explique por que o método GET é utilizado para listar produtos e quais são os benefícios de utilizar esse método em uma API RESTful;
- Por que o método POST é o mais adequado para adicionar novos produtos em uma API RESTful? Descreva o que ocorre no servidor quando o método POST é chamado;
- Descreva o funcionamento do método DELETE em uma API RESTful. O que acontece quando tentamos remover um recurso que não existe e como a API deve responder nesse caso?