# Lab API REST com Flask

Neste laboratório, você receberá a estrutura básica (scaffold) de uma aplicação backend em Python usando **Flask**. O banco de dados **SQLite** já está configurado.

Você, desenvolvedor, deve implementar a lógica das rotas para completar as operações **CRUD** (Create, Read, Update, Delete) de um sistema de gerenciamento de produtos.

*Crie um pasta para o front-end e outra para o back-end

**para o front-end:**
nano index.html
para rodar um servidor web: python3 -m http.server 8000
http://localhost:8000

---

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Produtos</title>
  <style>
    body { font-family: 'Segoe UI', sans-serif; background-color: #f4f4f9; display: flex;
justify-content: center; padding: 20px; }
    .container { background: white; padding: 30px; border-radius: 8px; box-shadow: 0 0
10px rgba(0,0,0,0.1); width: 100%; max-width: 600px; }
    h2 { text-align: center; color: #333; }

    .form-group { display: flex; gap: 10px; margin-bottom: 20px; }
    input { padding: 10px; border: 1px solid #ddd; border-radius: 4px; flex: 1; }
    button { padding: 10px 20px; border: none; border-radius: 4px; cursor: pointer; color:
white; font-weight: bold; }
    #btnSalvar { background-color: #28a745; }
    #btnSalvar:hover { background-color: #218838; }

    table { width: 100%; border-collapse: collapse; margin-top: 20px; }
    th, td { padding: 12px; text-align: left; border-bottom: 1px solid #ddd; }
    th { background-color: #f8f9fa; }

    .btn-edit { background-color: #ffc107; color: #333; padding: 5px 10px; font-size: 0.9em;
margin-right: 5px; }
    .btn-delete { background-color: #dc3545; padding: 5px 10px; font-size: 0.9em; }
  </style>
</head>
<body>
```

```
<div class="container">
  <h2>Gerenciador de Produtos</h2>

  <form id="produtoForm">
    <input type="hidden" id="produtoId">
    <div class="form-group">
      <input type="text" id="nome" placeholder="Nome do Produto" required>
      <input type="number" id="preco" placeholder="Preço (R$)" step="0.01" required>
      <button type="submit" id="btnSalvar">Salvar</button>
    </div>
  </form>

  <table>
    <thead>
      <tr>
        <th>ID</th>
        <th>Nome</th>
        <th>Preço</th>
        <th>Ações</th>
      </tr>
    </thead>
    <tbody id="tabelaCorpo">
      </tbody>
  </table>
</div>

<script>
  const API_URL = 'http://127.0.0.1:5000/api/produtos';

  async function carregarProdutos() {
    const response = await fetch(API_URL);
    const produtos = await response.json();

    const tbody = document.getElementById('tabelaCorpo');
    tbody.innerHTML = '';

    produtos.forEach(produto => {
      const tr = document.createElement('tr');
      tr.innerHTML = `
        <td>${produto.id}</td>
        <td>${produto.nome}</td>
        <td>R$ ${produto.preco.toFixed(2)}</td>
        <td>
          <button class="btn-edit" onclick="prepararEdicao(${produto.id},
'${produto.nome}', ${produto.preco})">Editar</button>
          <button class="btn-delete"
onclick="deletarProduto(${produto.id})">Excluir</button>
        </td>
```

```
            `;
            tbody.appendChild(tr);
        });
    }

    document.getElementById('produtoForm').addEventListener('submit', async (e) => {
        e.preventDefault();

        const id = document.getElementById('produtoId').value;
        const nome = document.getElementById('nome').value;
        const preco = parseFloat(document.getElementById('preco').value);

        const dados = { nome, preco };

        if (id) {
            await fetch(`${API_URL}/${id}`, {
                method: 'PUT',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(dados)
            });
        } else {
            await fetch(API_URL, {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(dados)
            });
        }

        document.getElementById('produtoForm').reset();
        document.getElementById('produtoId').value = '';
        carregarProdutos();
    });

    async function deletarProduto(id) {
        if (confirm("Tem certeza que deseja excluir?")) {
            await fetch(`${API_URL}/${id}`, { method: 'DELETE' });
            carregarProdutos();
        }
    }

    window.prepararEdicao = (id, nome, preco) => {
        document.getElementById('produtoId').value = id;
        document.getElementById('nome').value = nome;
        document.getElementById('preco').value = preco;
    }

    carregarProdutos();
</script>
```

```
</body>
</html>
```

---

**Para o back-end:**
Você deve editar o arquivo **app.py** e os comentários pelo código funcional em cada uma das 4 rotas (métodos: GET, POST, PUT, DELETE)

```
python3 -m venv .venv
source .venv/bin/activate
pip install flask flask-cors
nano app.py
```

---

```
from flask import Flask, jsonify, request
from flask_cors import CORS
import sqlite3

app = Flask(__name__)
CORS(app)

DB_NAME = "produtos.db"

def init_db():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS produtos (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            nome TEXT NOT NULL,
            preco REAL NOT NULL
        )
    ''')
    conn.commit()
    conn.close()

def get_db_connection():
    conn = sqlite3.connect(DB_NAME)
    conn.row_factory = sqlite3.Row
    return conn
```

```
@app.route('/api/produtos', methods=['GET'])
def get_produtos():
  #1: Conectar no banco
  #2: Selecionar todos os produtos
  #3: Converter para lista de dicionários
  #4: Retornar jsonify

@app.route('/api/produtos', methods=['POST'])
def add_produto():
  #1: Pegar dados do request.get_json()
  #2: Inserir no banco (INSERT)
  #3: Commit e fechar conexão
  #4: Retornar mensagem de sucesso e status 201

@app.route('/api/produtos/<int:id>', methods=['PUT'])
def update_produto(id):
  #1: Pegar novos dados do JSON
  #2: Atualizar o produto com o ID específico (UPDATE)

@app.route('/api/produtos/<int:id>', methods=['DELETE'])
def delete_produto(id):
  #1: Deletar o produto com o ID específico (DELETE)

if __name__ == '__main__':
  init_db() # Inicializa o banco ao rodar
  app.run(debug=True, port=5000)
```

---