PROGRAMAÇÃO WEB II

Curso Técnico Integrado em Informática

Lucas Sampaio Leite

Verbos HTTP

INSTITUTO FEDERAL
Baiano

- Os verbos HTTP (ou métodos HTTP) são ações que indicam o que deve ser feito com um determinado recurso em uma requisição feita a um servidor web.
- Eles fazem parte do protocolo HTTP, usado na comunicação entre clientes (como navegadores) e servidores.
- Principais verbos HTTP:
  - GET
  - POST
  - PUT
  - PATH
  - DELETE

Verbos HTTP

INSTITUTO FEDERAL
Baiano

- GET: Solicita dados de um recurso.
  - Não altera nada no servidor (somente leitura).
  - Os dados podem ser enviados na URL (parâmetros).
  - Exemplo: GET /produtos/123
- POST: Envia dados para o servidor para criar um novo recurso.
  - Usado para criar algo.
  - Os dados são enviados no corpo da requisição.
  - Exemplo: POST /usuarios
    - Corpo: { "nome": "Ana", "idade": 22 }

Verbos HTTP

INSTITUTO FEDERAL
Baiano

- PUT: Atualiza completamente um recurso existente.
  - Substitui todos os dados do recurso com os dados enviados.
  - Exemplo: PUT /produtos/123
    - Corpo: { "nome": "Caderno", "preco": 10.00 }
- PATCH: Atualiza parcialmente um recurso.
  - Apenas os campos especificados são modificados.
  - Exemplo: PATCH /produtos/123
    - Corpo: { "preco": 12.00 }

Verbos HTTP

- DELETE: Remover um recurso.
  - Exemplo: DELETE /usuarios/45

INSTITUTO FEDERAL
Baiano

Tratando Diferentes Métodos HTTP em Flask

INSTITUTO FEDERAL
Baiano

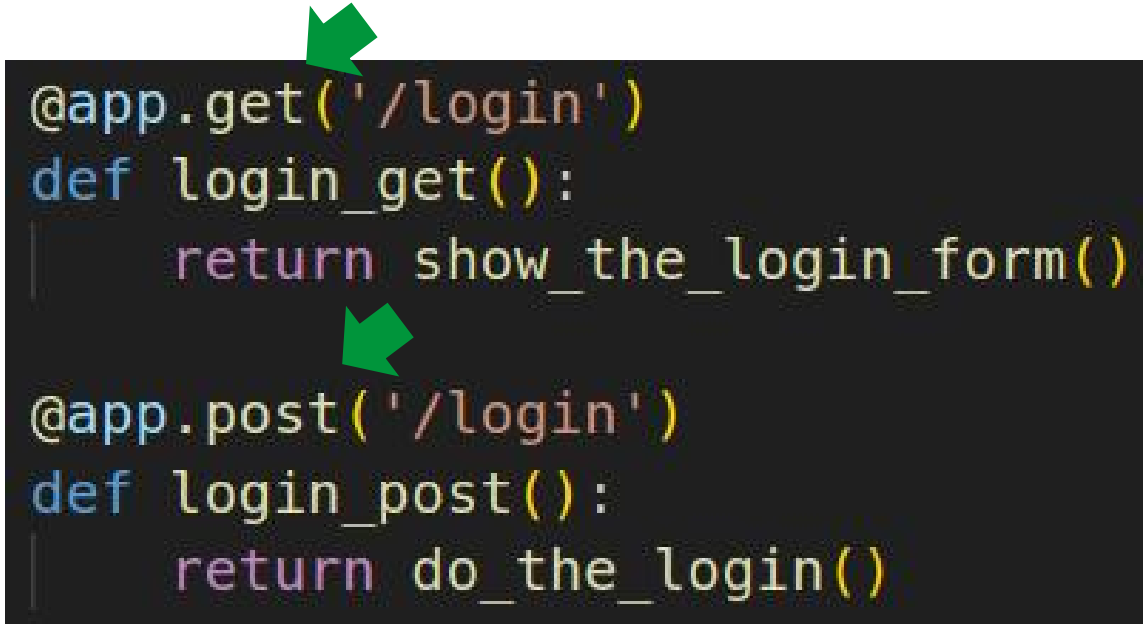
- Por padrão, uma rota responde apenas a requisições do tipo GET. Para permitir outros métodos HTTP, é necessário utilizar o argumento methods no decorador route().

```python
from flask import Flask, request

app = Flask(__name__)

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return do_the_login()
    else:
        return show_the_login_form()
```

Tratando Diferentes Métodos HTTP em Flask

INSTITUTO FEDERAL
Baiano

- O Flask oferece uma forma mais direta e legível de associar funções às rotas que respondem a métodos HTTP específicos:
  - Em vez de usar o decorador @app.route() com o argumento methods, pode-se utilizar os decoradores especializados, como @app.get(), @app.post(), @app.put(), entre outros.



Tratando Diferentes Métodos HTTP em Flask

INSTITUTO FEDERAL
Baiano

- Voltando ao nosso exercício:

```python
from flask import Flask

app = Flask(__name__)

agenda = {
    "ana": "99999-1234",
    "joao": "98888-5678",
    "maria": "97777-9012"
}

@app.route('/contato/<nome>')
def buscar_contato(nome):
    nome = nome.lower()
    if nome in agenda:
        return f"Telefone de {nome.capitalize()}: {agenda[nome]}"
    else:
        return "Contato não encontrado."
```

Formato JSON

INSTITUTO FEDERAL
Baiano

- JSON (JavaScript Object Notation) é um formato leve de troca de dados, baseado em texto, fácil de ler e escrever para humanos, e fácil de interpretar e gerar por máquinas.
- Por que usar JSON em APIs?
  - Padrão amplamente suportado
  - Formato leve e eficiente
  - Estrutura clara e compatível com objetos
  - Facilidade de uso com JavaScript (frontend)
  - Integração com ferramentas modernas
  - Padronização em APIs REST

Formato JSON

INSTITUTO FEDERAL
Baiano

- Exemplo de resposta em Json:

- Essa estrutura pode ser facilmente interpretada em qualquer linguagem:
  - Em Python, por um dicionário
  - Em JavaScript, por um objeto
  - Em Java, pode ser mapeado para uma classe com Jackson ou Gson

Formato JSON

INSTITUTO FEDERAL
Baiano

- Usar JSON em APIs melhora a interoperabilidade, desempenho e facilidade de integração entre sistemas.
- Ele se tornou o padrão de fato porque oferece um excelente equilíbrio entre legibilidade humana e eficiência computacional.

Implementando uma busca de um contato usando @app.get

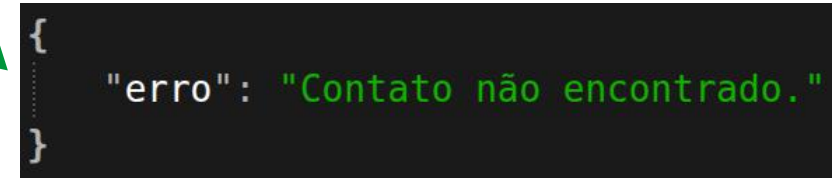
INSTITUTO FEDERAL
Baiano

```python
@app.get('/contato/<nome>')
def buscar_contato(nome):
    nome = nome.lower()
    if nome in agenda:
        return jsonify({
            'nome': nome.capitalize(),
            'telefone': agenda[nome]
        })
    else:
        return jsonify({
            'erro': 'Contato não encontrado.'
        })
```

Implementando uma busca de um contato usando @app.get

INSTITUTO FEDERAL
Baiano

```python
@app.get('/contato/<nome>')
def buscar_contato(nome):
    nome = nome.lower()
    if nome in agenda:
        return jsonify({
            'nome': nome.capitalize(),
            'telefone': agenda[nome]
        })
    else:
        return jsonify({
            'erro': 'Contato não encontrado.'
        })
```



Implementando uma busca de um contato usando @app.get

INSTITUTO FEDERAL
Baiano

- O jsonify é uma função tradicional do Flask usada para converter objetos Python (como dicionários, listas) em uma resposta JSON válida, configurando automaticamente o cabeçalho Content-Type: application/json e serializando os dados para JSON.
- A partir do Flask 1.1 (e versões recentes), pode-se simplesmente retornar um dicionário Python que o Flask:
  - converte automaticamente para JSON
  - adiciona o cabeçalho correto (Content-Type: application/json)

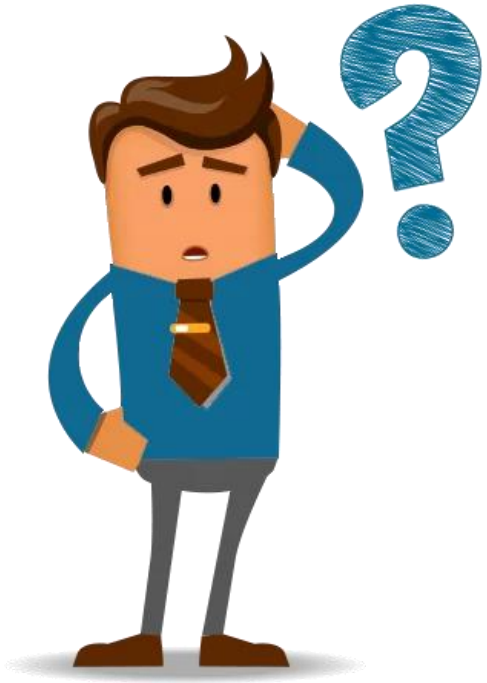
Implementando uma busca de um contato usando @app.get

INSTITUTO FEDERAL
Baiano

```python
@app.get('/contato/<nome>')
def buscar_contato(nome):
    nome = nome.lower()
    if nome in agenda:
        return {
            'nome': nome.capitalize(),
            'telefone': agenda[nome]
        }
    else:
        return {
            'erro': f"Contato '{nome}' não encontrado."
        }
```

Dúvidas

INSTITUTO FEDERAL
Baiano



PROGRAMAÇÃO WEB II

Curso Técnico Integrado em Informática

Lucas Sampaio Leite