

Trabalho API Rest – GET

Joinville, 23 de maio de 2023.

UC: Sistemas Distribuídos e Mobile.

Professora: Marisangila Alves.

Alunos: Éverton Gambeta.

Jean Carlos Severino.

1 – INTRODUÇÃO

O significado de REST é Representational State Transfer, que significa Transferência Representacional de Estado é um modelo de arquitetura e não uma linguagem ou tecnologia de programação, que fornece diretrizes para que os sistemas distribuídos se comuniquem diretamente usando os princípios e protocolos existentes da Web sem a necessidade de SOAP ou outro protocolo sofisticado. A arquitetura REST é simples e fornece acesso aos recursos para que o cliente REST acesse e renderize os recursos no lado do cliente. No estilo REST, URI ou IDs globais ajudam a identificar cada recurso. Esta arquitetura usa várias representações de recursos para representar seu tipo, como XML, JSON, Texto, Imagens e assim por diante.

2 – FUNCIONALIDADE

Existe no REST um princípio chamado STATELESSNESS (sem estado), onde o servidor não precisa saber em qual estado o cliente está e vice-versa.

O **cliente** é o componente solicitante de um serviço e envia solicitações para vários tipos de serviços ao servidor já o **servidor** é o componente que é o provedor de serviços e fornece continuamente serviços ao cliente conforme as solicitações.

Nesta arquitetura ou modelo, cliente-servidor ajuda na separação de responsabilidades entre a interface do usuário e o armazenamento de dados. Ou seja, quando uma solicitação REST é realizada, o servidor envia uma representação dos estados que foram requeridos.

A comunicação entre cliente e servidor ocorre através da troca de mensagens usando um padrão de solicitação-resposta. O cliente basicamente envia uma solicitação de serviço e o servidor retorna uma resposta.

3 – REQUISIÇÕES

O REST precisa que um cliente faça uma requisição para o servidor para enviar ou modificar dados. Uma requisição consiste em:

Um verbo ou método HTTP, que define que tipo de operação o servidor vai realizar;

Um header, com o cabeçalho da requisição que passa informações sobre a requisição;

Um path (caminho ou rota) para o servidor, como por exemplo <https://www.alura.com.br/artigos/golang-trabalhando-com-datas>;

Informação no corpo da requisição, sendo esta informação opcional.

4 – MÉTODOS HTTP

Em aplicação REST, os métodos mais utilizados são:

O método GET é o método mais comum, geralmente é usado para solicitar que um servidor envie um recurso;

O método POST foi projetado para enviar dados de entrada para o servidor. Na prática, é frequentemente usado para suportar formulários HTML;

O método PUT edita e atualiza documentos em um servidor;

O método DELETE que como o próprio nome já diz, deleta certo dado ou coleção do servidor.

5 – CÓDIGO DE RESPOSTAS

Cada resposta que a aplicação REST retorna, é enviado um código definindo o status da requisição. Por exemplo:

200 (OK), requisição atendida com sucesso;

201 (CREATED), objeto ou recurso criado com sucesso;

204 (NO CONTENT), objeto ou recurso deletado com sucesso;

400 (BAD REQUEST), ocorreu algum erro na requisição (podem existir inúmeras causas);

404 (NOT FOUND), rota ou coleção não encontrada;

500 (INTERNAL SERVER ERROR), ocorreu algum erro no servidor.

6 – TRABALHO COM A API REST

Utilizando 3 (três) solicitações GET do site Brasil API:

<https://brasilapi.com.br/docs>

São elas:

<https://brasilapi.com.br/docs#tag/Feriados-Nacionais>

<https://brasilapi.com.br/docs#tag/CEP>

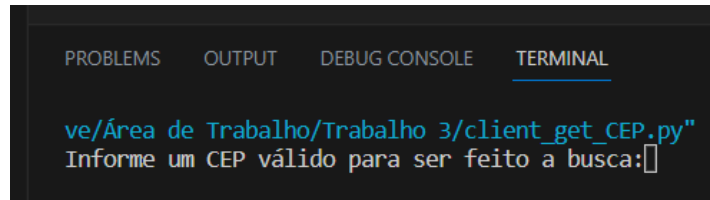
<https://brasilapi.com.br/docs#tag/TAXAS>

6.1 – CEP

Assim temos o código:

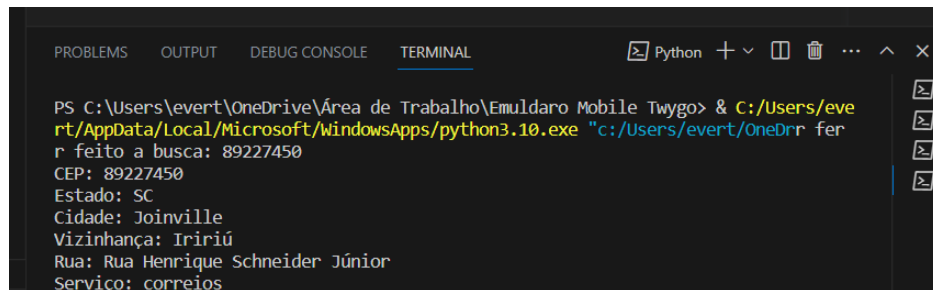
```
client_get_CEP.py X
C: > Users > evert > OneDrive > Área de Trabalho > Trabalho 3 > client_get_CEP.py > ...
1  import requests
2  import os
3
4  while True:
5      # Solicita a inclusão de um CEP.
6      cep = input("Informe um CEP válido para ser feito a busca:")
7
8      # Concatena a URL e o parâmetro necessário (CEP informado pelo usuário)
9      url = "https://brasilapi.com.br/api/cep/v1/"+cep
10
11     # Realiza a solicitação HTTP GET
12     response = requests.get(url)
13
14     # Verifica se a resposta foi bem sucedida (código 200)
15     if response.status_code == 200:
16
17         data = response.json()
18
19         if 'erro' not in data: # Verifica se a API retornou um erro:
20
21             # Imprime a resposta:
22
23             print("CEP:", data.get('cep'))
24             print("Estado:", data.get('state'))
25             print("Cidade:", data.get('city'))
26             print("Vizinhança:", data.get('neighborhood'))
27             print("Rua:", data.get('street'))
28             print("Serviço:", data.get('service'))
29         else:
30             print("CEP não encontrado.")
31     else:
32         # Imprime o código de erro HTTP
33         print("Erro HTTP %d - %s" % (response.status_code, response.reason))
34
35         break
36
37     os.system("PAUSE")
```

Ao rodar o código o terminal pergunta para informar um CEP:



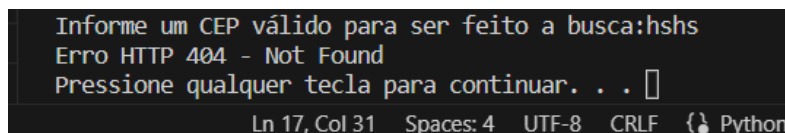
```
ve/Área de Trabalho/Trabalho 3/client_get_CEP.py"
Informe um CEP válido para ser feito a busca:
```

Ao informar o CEP e confirmar (ENTER), o resultado é mostrado:



```
PS C:\Users\event\OneDrive\Área de Trabalho\Emulador Mobile Twygo> & C:/Users/event/AppData/Local/Microsoft/WindowsApps/python3.10.exe "c:/Users/event/OneDrive/Trabalho 3/client_get_CEP.py"
Informe um CEP válido para ser feito a busca: 89227450
CEP: 89227450
Estado: SC
Cidade: Joinville
Vizinhança: Iririú
Rua: Rua Henrique Schneider Júnior
Serviço: correios
```

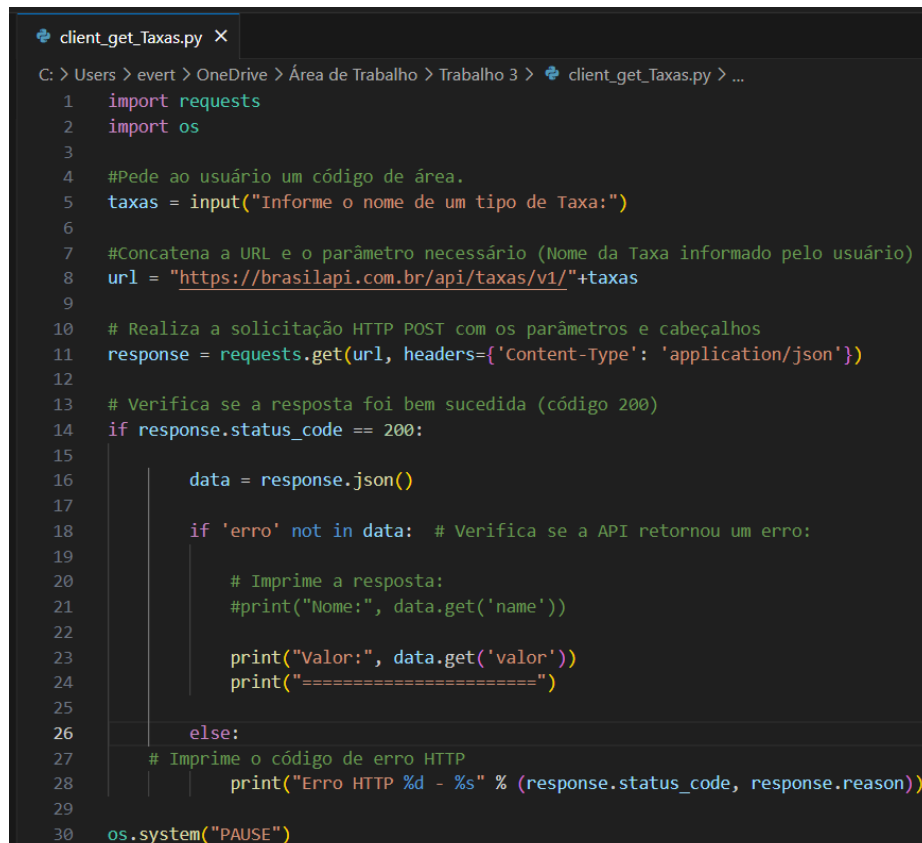
Caso o valor informado seja inválido será mostrado um erro:



```
Informe um CEP válido para ser feito a busca:hshs
Erro HTTP 404 - Not Found
Pressione qualquer tecla para continuar. . .
```

6.2 – Taxas

Assim temos o código:



```
client_get_Taxas.py X
C: > Users > event > OneDrive > Área de Trabalho > Trabalho 3 > client_get_Taxas.py > ...
1 import requests
2 import os
3
4 #Pede ao usuário um código de área.
5 taxas = input("Informe o nome de um tipo de Taxa:")
6
7 #Concatena a URL e o parâmetro necessário (Nome da Taxa informado pelo usuário)
8 url = "https://brasilapi.com.br/api/taxas/v1/"+taxas
9
10 # Realiza a solicitação HTTP POST com os parâmetros e cabeçalhos
11 response = requests.get(url, headers={'Content-Type': 'application/json'})
12
13 # Verifica se a resposta foi bem sucedida (código 200)
14 if response.status_code == 200:
15     data = response.json()
16
17     if 'erro' not in data: # Verifica se a API retornou um erro:
18
19         # Imprime a resposta:
20         #print("Nome:", data.get('name'))
21
22         print("Valor:", data.get('valor'))
23         print("=====")
24
25     else:
26
27         # Imprime o código de erro HTTP
28         print("Erro HTTP %d - %s" % (response.status_code, response.reason))
29
30 os.system("PAUSE")
```

Ao rodar o código o terminal pergunta para informar um nome de uma “Taxa”:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\event\OneDrive\Área de Trabalho\Emuldar o M
ive\Área de Trabalho\Trabalho 3/client_get_Taxas.py"
Informe o nome de um tipo de Taxa:
```

Ao informar o nome de uma “Taxa” e confirmar (ENTER), o resultado é mostrado.
Exemplo taxa “Selic” e taxa “CDI”.

```
PROBLEMS TERMINAL ... Python + - [ ] [ ] ... ^ X

ive\Área de Trabalho\Trabalho 3/client_get_Taxas.py"
Informe o nome de um tipo de Taxa:CDI
Valor: 13.65
=====
Pressione qualquer tecla para continuar. . .
PS C:\Users\event\OneDrive\Área de Trabalho\Emuldar o Mob
ile Twygo> C:/Users/event/AppData/Local/Microsoft/Windo
wsApps/python3.10.exe "c:/Users/event/OneDrive/Área de T
rabalho/Trabalho 3/client_get_Taxas.py"
Informe o nome de um tipo de Taxa:SELIC
Valor: 13.75
=====
```

6.3 – Feriados Nacionais

Assim temos o código:

```
Welcome client_get_Feriados_Nacionais.py X
C: > Users > event > OneDrive > Área de Trabalho > Trabalho 3 > client_get_Feriados_Nacionais.py > ...

1 import requests
2 import os
3
4 #Pede ao usuário um código de área.
5 ano = input("Informe o Ano desejado:")
6
7 #Concatena a URL e o parâmetro necessário (DDD informado pelo usuário)
8 url = "https://brasilapi.com.br/api/feriados/v1/"+ano
9
10 # Realiza a solicitação HTTP POST com os parâmetros e cabeçalhos
11 response = requests.get(url, headers={'Content-Type': 'application/json'})
12
13 # Verifica se a resposta foi bem sucedida (código 200)
14 if response.status_code == 200:
15     data = response.json()
16     for feriado in data:
17         #print(feriado)
18         print("Data", feriado.get('date'))
19         print("Nome", feriado.get('name'))
20         print("=====")
21     else:
22         # Imprime o código de erro HTTP
23         print("Erro HTTP %d - %s" % (response.status_code, response.reason))
24
25 os.system("PAUSE")
```

Ao rodar o código o terminal pergunta para informar um “Ano”:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\event\OneDrive\Área de Trabalho\Emuldar Mobile Twygo> &
ive/Área de Trabalho/Trabalho 3/client_get_Feriados_Nacionais.py"
Informe o Ano desejado:[]
```

Ao informar o “Ano” e confirmar (ENTER), o resultado é mostrado:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

=====
PS C:\Users\event\OneDrive\Área de Trabalho\Emuldar Mobile Twygo> &
ive/Área de Trabalho/Trabalho 3/client_get_Feriados_Nacionais.py"
Informe o Ano desejado:2022
Data 2022-01-01
Nome Confraternização mundial
=====
Data 2022-03-01
Nome Carnaval
=====
Data 2022-04-15
Nome Sexta-feira Santa
=====
Data 2022-04-17
Nome Páscoa
=====
Data 2022-04-21
Nome Tiradentes
=====
Data 2022-05-01
Nome Dia do trabalho
=====
Data 2022-06-16
Data 2022-11-02
Nome Finados
=====
Data 2022-11-15
Nome Proclamação da República
=====
Data 2022-12-25
Nome Natal
=====
```

Fonte: https://www.alura.com.br/artigos/rest-conceito-e-fundamentos?gclid=EAlaIQobChMI_ayBmLKH_wIVwCvUAR0QgAMLEAAYASAAEqID7fD_BwE