Roteiro de Atividades – *Webservices*

Procedimentos preliminares - Instalação das ferramentas

1 Instalar PIP (gerenciador de pacotes para Python)
- 1.1 Linux/Python 2.x: em um terminal, usar o comando `sudo apt install python-pip`
- 1.2 Linux/Python 3.x: em um terminal, usar o comando `sudo apt install python3-pip`
- 1.3 Windows: ver instruções em pip.pypa.io/en/stable/installing/

2 Instalar o *virtualenv*
- 2.1 Linux: `sudo apt install virtualenv` ou `sudo apt install python-virtualenv`
- 2.2 Windows: ver instruções em https://packaging.python.org/guides/installing-using-pip-and-virtual-environments/

3 Instalar o *Flask*
- 3.1 Linux/Python 2.x: em um terminal, usar o comando `sudo pipinstall flask && sudo apt install python-flask`
- 3.2 Linux/Python 3.x: em um terminal, usar o comando `sudo pip3 install flask && sudo apt install python-flask`
- 3.3 Windows: ver instruções em flask.palletsprojects.com/en/1.1.x/installation/

4 Iniciar um ambiente virtual para o desenvolvimento da aplicação:
- 4.1 Em um terminal, criar e acessar um diretório (ou *pasta*) para armazenar a aplicação
- 4.2 Usar o seguinte comando: `virtualenv flask`
- 4.3 Verificar se, no terminal será exibida uma saída semelhante umas das exibidas abaixo.

```
Running virtualenv with interpreter /usr/bin/python2
New python executable in /python_rest/flask/bin/python2
Also creating executable in /python_rest/flask/bin/python
Installing setuptools, pkg_resources, pip, wheel...done.
```

ou

```
created virtual environment CPython3.8.10.final.0-64 in 248ms
  creator CPython3Posix(dest=.../flask, clear=False, global=False)
  seeder FromAppData(download=False, pip=latest, setuptools=latest, wheel=latest, pkg_resources=latest,
via=copy, app_data_dir=.../.local/share/virtualenv/seed-app-data/v1.0.1.debian.1)
  activators
BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
```

Parte 1 – Obtendo dados do servidor

1 Criar um arquivo texto (ASCII) com o seguinte conteúdo:

```
#!flask/bin/python
from flask import Flask, jsonify

app = Flask(__name__)
app.config['JSON_AS_ASCII'] = False

@app.route('/hello')
def sayHello():
    return "Hello world!"

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

Observações:

1. A porta padrão do Flask é a 5000. Para utilizar outra porta, alterar a última linha para `app.run(debug=True, port=<numero da porta>)`
2. Ao executar a aplicação em um servidor externo, alterar a última linha para `app.run(debug=True, port=<numero da porta>, host='0.0.0.0')`

2 Salvar o arquivo com extensão "*.py*" *na* pasta criada no passo 4.1 dos procedimentos preliminares

3 Executar a aplicação utilizando o comando `python3 <nome do arquivo>`

4 Em um navegador, acessar o endereço *http://localhost:5000/hello* e analisar o resultado

3. Atividades – Parte 2 – Enviando parâmetros ao servidor

1. Adicionar a seguinte função à classe criada anteriormente :

```
@app.route('/hello/<string:name>')
def sayHelloName(name):
     return "Hello, " + name
```

Observação: a função deve ser inserida antes da linha "`if __name__ == '__main__':`"

2. Acessar o endereço *http://localhost:5000/hello/nome* e analisar o resultado.

4. Atividades – Parte 3 – Obtendo dados do servidor em formato JSON

1. Adicionar a seguinte função à classe criada anteriormente :

```
@app.route('/hello/hellojson')
def sayHelloJson():
   return jsonify({"english": "hello world","português": "olá, mundo",
                   "français": "bonjour, tout le monde"})
```

Observação: a função deve ser inserida antes da linha "`if __name__ == '__main__':`"

2. Acessar o endereço *http://localhost:5000/hello/hellojson*