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
- 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 `pip install flask && sudo apt install python-flask`
- 3.2 Linux/Python 2.x: em um terminal, usar o comando `pip 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
```

Atividades

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

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

app = Flask(__name__)

alunos = [
          {'id':1, 'nome':'alice', 'curso':'Engenharia de Controle e Automação', 'idade':20},
          {'id':2, 'nome':'bob', 'curso':'Engenharia de Materiais', 'idade':18}
         ]


@app.route('/alunos/getalunos', methods=['GET'])
def get_alunos():
   return jsonify({'alunos':alunos})


#continua na próxima página...
```

```
# ...continuação

@app.route('/alunos/getaluno/<int:aluno_id>', methods=['GET'])
def get_aluno(aluno_id):
    a = None
    for aluno in alunos:
        if aluno['id'] == aluno_id:
            a = aluno
    if a == None:
        abort(404)
    return jsonify(a)


@app.route('/alunos', methods=['POST'])
def create_aluno():
    if not request.json or not 'nome' in request.json or not 'curso' in request.json:
        abort(400)
    aluno = { 'id': alunos[-1]['id']+1,
              'nome': request.json['nome'],
              'curso': request.json['curso'],
              'idade': request.json['idade']
            }
    alunos.append(aluno)
    return jsonify({'aluno':aluno}),201



if __name__ == '__main__':
    app.run(host='localhost', port=8081, debug=True)
```

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 as URLs a seguir, observando os resultados:
- 4.1 http://localhost:8081/alunos/getalunos
- 4.2 http://localhost:8081/alunos/getaluno/1
- 4.3 http://localhost:8081/alunos/getaluno/2

5 Em um terminal, digitar o seguinte comando (sem quebra de linhas):

```
curl --header "Content-Type: application/json" --request POST --data
'{"nome":"alice","curso":"textil","idade":25}' http://localhost:8081/alunos
```

6 Em um navegador, acessar http://localhost:8081/alunos/getalunos e analisar o resultado.