Programação na Prática & API's

Bem-vindos!

Objetivos

- ✔ Flask

LET'S CODE

Instalando e conhecendo o Postman

https://www.getpostman.com/downloads/

Tipos de requisições HTTP

- GET;
- POST;
- PUT;
- DELETE;

Para mais tipo, consultar: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods

Requisição GET

```
from flask import Flask, request, current_app, jsonify

app = Flask(__name__)

@app.route("/show_config", methods=['GET'])
def show_config():
    querystring_args = request.args.to_dict()
    if int(querystring_args['id']) == 10:
        return jsonify(
            id = 10,
            nome = "Renato",
            cargo = "Professor"
        )
    return "Não há usuário com esse id.", 200

app.run()
```

GET: localhost:5000/show_config?id=10

Requisição POST

```
from flask import Flask, request, current_app, jsonify

app = Flask(__name__)

@app.route("/show_config", methods=['POST'])
def show_config():
    post_args = request.json
    if post_args['id'] == 10:
        return "Usuário salvo com sucesso!", 200
    return "Usuário não pôde ser salvo.", 200

app.run()
```

POST: localhost:5000/show_config

```
{
        "id": 10,
        "nome": "Renato",
        "cargo": "Professor"
}
```

Requisição PUT

```
from flask import Flask, request, current_app, jsonify

app = Flask(__name__)

@app.route("/show_config", methods=['PUT'])
def show_config():
    querystring_args = request.args.to_dict()
    post_args = request.json
    if post_args['id'] == 10:
        post_args['nome'] = "Renato"
        return 'Usuário atualizado com sucesso!', 200
    return 'Usuário não foi atualizado', 200

app.run()
```

PUT: localhost:5000/show_config

```
{
        "id": 10,
        "nome": "Fulano"
}
```

Requisição DELETE

```
from flask import Flask, request, current_app, jsonify

app = Flask(__name__)

@app.route("/show_config", methods=['DETELE'])
def show_config():
    querystring_args = request.args.to_dict()
    if int(querystring_args['id']) == 10:
        return 'Usuário deletado com sucesso!', 200
    return 'Usuário não foi deletado', 200

app.run()
```

DELETE: localhost:5000/show_config?id=10

Envio de Arquivos

```
from flask import Flask, request, current_app, jsonify
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

app = Flask(__name__)

@app.route("/show_config", methods=['POST'])
def is_hot_dog():
    if not 'file' in request.files:
        return "Erro! Não há arquivo na requisição.", 400
    image = Image.open(request.files['file'])
    arr = np.array(image)
    plt.imshow(arr)
    plt.show()
    return "Imagem visualizada corretamente!", 200

app.run()
```

Download de Arquivos

```
from flask import Flask, send_from_directory

app = Flask(__name__)

@app.route('/show_config', methods=['GET'])
def download():
    return send_from_directory(directory="", filename="iris.csv")

app.run()
```

GET: localhost:5000/show_config

Fazer uma requisição pelo Flask

```
import pandas as pd
import requests

def proxy_example():
    csv =
requests.get("https://gist.githubusercontent.com/netj/8836201/raw/6f9306ad21398ea43cba4f7d537619d0
e07d5ae3/iris.csv").content
    f = open('data.csv', 'wb')
    f.write(csv)
    f.close()
    df = pd.read_csv('data.csv')
    print(df)

proxy_example()
```

Headers

```
from flask import Flask, request

app = Flask(__name__)

@app.route("/show_config", methods=['POST'])
def show_config():
    post_args = request.json
    headers = request.headers['teste']
    print(headers)
    if post_args['id'] == 10:
        return "Usuário salvo com sucesso!", 200
    return "Usuário não pôde ser salvo.", 200

app.run()
```

POST: localhost:5000/show_config

Desafio

Squad 1:
- Testar apis dos outros squads usando postman;

Squad 2:
- Criar endpoint de get que retorna o csv da base de dados iris;
- Criar endpoint de post que adiciona um dado ao csv, sobrescrevendo o csv. O json recebido deve ser: {“sepal.length”: x.x, “sepal.width”: x.x, “petal.length”: x.x, “petal.width”: x.x, “variety”: “Tipo”};
- Criar endpoint de put que atualiza um dado ao csv, sobrescrevendo o csv. O json recebido deve ser: {“id”: x, “sepal.length”: x.x, “sepal.width”: x.x, “petal.length”: x.x, “petal.width”: x.x, “variety”: “Tipo”}, em que o id é a linha do arquivo;
- Criar endpoint de delete que atualiza um dado ao csv, sobrescrevendo o csv. Deve ser enviado o id e deletado a linha do csv correspondente a esse id;

Squad 3:
- Criar método para leitura do csv vindo do Squad 2;
- Criar endpoint de post que retorna o gráfico de dispersão e tem como input um json com as coordenadas a serem plotadas. Por exemplo: {“coords”: [“sepal.length”, “petal.length”]
- Criar endpoint de get que retorna o histograma;

Squad 4:
- Criar método para leitura do csv vindo do Squad 2;
- Criar método post que recebe o json: {“sepal.length”: x.x, “sepal.width”: x.x, “petal.length”: x.x, “petal.width”: x.x}, classificar o dado e retornar classe a que o dado pertence;