Universidade Federal de Viçosa - Campus Florestal

Trabalho Prático V
CarnaTroca

Professor: Thais Regina de Moura Braga Silva

Disciplina: Sistemas Distribuídos e Paralelos (CCF 355)

Arthur Marciano - 3019
Vinicius Julio - 3495

Florestal, MG
2022

ÍNDICE

1. Introdução

O objetivo deste trabalho é a implementação do sistema de escambo CarnaTroca, utilizado nos trabalhos anteriores, utilizando o middleware Web Service Rest e seu framework Flask.

É importante ressaltar que os requisitos implementados anteriormente não foram alterados, o que muda é apenas a forma como o sistema é implementado, além disso a interface do sistema continua sendo por linha de comando.

2. Desenvolvimento

Utilização do Rest e Flask

Apesar do Rest prover várias requisições HTTP para serem utilizadas, neste trabalho prático foram implementadas apenas requisições **GET e POST.**

Algumas modificações que são importantes de ser citadas foram as da classe Servidor e Cliente.

Na classe Servidor, agora utilizamos uma *rota* ou *endpoint* que servirá como uma associação entre o método HTTP (POST no exemplo abaixo), o cliente que neste exemplo irá acessar a rota chamando o *cadastraUser* e a função do servidor que também é *cadastraUser*. Além disso, outro detalhe que deve ser comentado é o retorno da função, neste caso utilizamos o método *jsonify* importado do *flask*, que irá transformar o conteúdo do banco de dados em uma mensagem no formato JSON, de forma que a mesma possa enviada pela rede.

Seguimos o modelo mostrado abaixo para todas as funções da classe servidor.

```
You, 31 minutes ago | 1 author (You)
from flask import Flask, request, jsonify
from flask_restful import Api
import banco_dados.db as bd


app = Flask(__name__)
app.config['DEBUG']  = True
api = Api(app)


You, 31 minutes ago | 1 author (You)
class Servidor():

    @app.route('/cadastraUser', methods=['POST'])
    def cadastraUser():
        nome = request.form.get('nome')
        senha = request.form.get('senha')
        result = bd.cadastroUser_db(nome, senha)
        return jsonify(result)
```

Imagem 1: Implementação da classe Servidor

Já na classe Cliente, agora utiliza-se o *requests* que é uma biblioteca HTTP para a linguagem de programação Python e tem o objetivo de tornar as solicitações HTTP mais simples e mais fáceis de usar.

No exemplo mostrado na imagem abaixo, o que este código faz é enviar uma solicitação POST para a URL relacionada à função *cadastraUser* (função na classe Servidor*)*. Neste caso estamos usando a função .post(), mas também utilizamos a função .get() em outras solicitações, como por exemplo quando queremos ver a lista de fantasias disponíveis para troca.

Basicamente seguimos o exemplo abaixo para todas as funções da classe cliente, apenas alterando a função entre post() e get() e os parâmetros passados.

```
import requests
      You, 44 minutes ago • Adicionado o TP4 e TP5 à main …
You, 44 minutes ago | 1 author (You)
class Cliente(object):

    def __init__(self, *args, **kwargs):
        pass

    def run ():
        tela_inicial.tela_inicio()
        num = input('Escolha uma das opcoes acima para continuar:')
        tela_inicial.switch_tela_inicio(num)


    def cadastraUser(nome, senha):
        request = requests.post('http://localhost:5000/cadastraUser',data = {'nome': nome, 'senha':senha})
        resultado = request.json()
        return resultado
```

Imagem 2: Implementação da classe Cliente

Método de Execução

A forma de execução continua sendo a mesma dos trabalhos anteriores, bastando executar primeiro o comando:

python3 servidor.py

E em seguida executar:

python3 cliente.py

No entanto, para que possa utilizar as bibliotecas do flask e requests é necessário que as mesmas sejam instaladas. Caso não possua estas bibliotecas instaladas basta utilizar os seguintes comandos:

pip install flask
pip install flask-restful
pip install requests

3. Conclusão

A realização desta última etapa do trabalho serviu para que os alunos pudessem trabalhar um pouco com o middleware REST que é cada vez mais utilizado no “mundo da computação”. Além de mostrar na prática como é feita uma comunicação entre Web Services utilizando o HTTP, este trabalho também foi de grande importância pois fez com o grupo aprendesse a utilizar um pouco do REST e seu framework FLASK, servindo como um aprendizado não só do conteúdo presente na disciplina mas também destas grandes ferramenta que vêm ganhando muito espaço atualmente.

Fazendo uma comparação entre as partes anteriores do trabalho, foi de consenso do grupo que a parte 5 se mostrou a mais fácil, tanto no aprendizado quanto no desenvolvimento da mesma. A parte 3 (implementação utilizando API de Sockets) com certeza foi a que mais demandou esforço durante sua implementação, sendo a parte em que o código ficou menos modularizado e organizado. No entanto, na parte 4 (GRPC) tivemos um pouco mais de dificuldade para aprender e realmente começar a desenvolver o sistema se comparado com as outras etapas.

Após a realização de todas as etapas, concluímos que todo o esforço valeu a pena, de modo que o grupo conseguiu entender a grande maioria dos conceitos apresentados previamente em sala, e conseguiu desenvolver um Sistema Distribuído bastante funcional, apesar de não possuir uma interface gráfica.

-