UNIVERSIDADE FEDERAL DE ALAGOAS

INSTITUTO DE COMPUTAÇÃO

RELATÓRIO: PROJETO DE REDES DE COMPUTADORES

MACEIÓ – AL

2023

ARTURO JIMÉNEZ LOAIZA

ARTHUR VINÍCIUS ROZENDO SANTOS

MANOEL ROCHA DOS SANTOS NETO

RELATÓRIO: PROJETO DE REDES DE COMPUTADORES

Relatório apresentado junto ao Instituto de Computação da Universidade Federal de Alagoas, como requisito para composição da nota AB2 referente a disciplina Rede de computadores.

Orientador: Prof. Almir Pereira Guimarães.

MACEIÓ – AL

2023

Objetivo do Trabalho

O presente projeto tem como objetivo principal a construção de um mural de posts, esses construídos por diversos clientes e tendo seu processamento em um servidor.

Para isso, foi utilizado a biblioteca socket, em Python. Outrossim, também foi utilizado a biblioteca Thread. Em auxílio a aplicação foi construído um banco de dados com o uso de arquivo JSON. Tal construção permite a comunicação entre cliente e servidor por meio de protocolos.

Link do Github: https://github.com/arthurvrs/projeto-redes-2022.2

Principais funcionalidades da aplicação

A primeira funcionalidade utiliza-se do método HTTP POST para a produção de um post por parte do cliente. No conteúdo, terá um título, descrição, nome do autor e os dados de tempo do post. Enviando ao servidor para fins de tratamento e retorno da mensagem de sucesso.

A segunda funcionalidade utiliza-se do método HTTP GET para a requisição de um arquivo JSON com uma lista de posts. O servidor acaba por retornar o arquivo por meio de uma String que tem seu tratamento no cliente e torna-se visível no mesmo.

Por fim, a terceira funcionalidade utiliza-se do método HTTP DELETE para reinicialização do Mural. No qual o cliente aplica a senha de Administrador e envia a mensagem com o método. Sendo assim, o servidor processa a mensagem, realiza o delete do mural e retorna uma mensagem de sucesso ao cliente.

Protocolos implementados

IP (Internet Protocol) é um protocolo de comunicação utilizado para a conexão de dispositivos em uma rede de computadores. Tal comunicação ocorre por meio do uso de pacotes. Na aplicação foi utilizado o tipo de endereço IPv4 como, por exemplo, no servidor foi utilizado ‘localhost’.

TCP (Transmission Control Protocol) é um protocolo de comunicação que tem sua aplicação no estabelecimento de conexões seguras entre os dispositivos de uma rede. Também é responsável pelo controle de fluxo dos pacotes, com o uso de portas do remetente e destinatário como, por exemplo, no servidor foi utilizado a porta 8000.

HTTP (Hypertext Transfer Protocol) é um protocolo de comunicação utilizado sobre o TCP/IP para desenvolvimento de um servidor que utiliza o World Wide Web. Desse modo, foram aplicados os métodos POST, GET e DELETE.

Funcionalidades a serem implementadas

A funcionalidade seria a utilização do método PUT, que seria a possibilidade de atualização de um post já existente no servidor, por meio de uma requisição do cliente.

Dificuldades apresentadas

A primeira dificuldade foi a construção do servidor a partir da utilização de sockets e Threads. Visto que trata-se de um conjunto de detalhes para que o servidor funcione corretamente. Outrossim, foi a implementação de novos métodos HTTP, a partir de suas funções que se ordenam com operações específicas.

Código Fonte

Código server.py

```
import socket
from threading import Thread
from dataBase.dataBase import setPost, returnPosts, resetBase

#Definição dos Dados Fixos da Aplicação
NameServer = 'Servidor Mural'
ADDR = 'localhost'
PORT = 8000

#Função que inicia o Servidor
def startServer():
    print('Servidor ', NameServer,' está Online!')
    #Criação do socket Servidor e atribuição de um endereço IP e Número de Porta e
configuração para aceitar conexões
    socketServer = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    socketServer.bind((ADDR, PORT))
    socketServer.listen(10)

    #Processo que aplica Thread e processamento de solicitações dos clientes
    while True:
        socketClient, clienteAddr = socketServer.accept()
        Thread(target=process(socketClient, clienteAddr), args=(socketClient,
clienteAddr)).start()

#Função que realiza o processamento do Servidor
def process(socketClient, clienteAddr):
    #Processamento da Mensagem e tratamento dos Métodos POST, GET, DELETE solicitados
pelo cliente
    dataReceived = socketClient.recv(1024)
```

```
    dataReceived = dataReceived.decode()
    message = dataReceived.split('\r\n')[-1]
    if dataReceived.startswith("POST"):
        print('\nProcessamento Método POST')
        data = message.split('\r\n')[-1]
        setPost(data)
        print('Processamento Método POST Concluido com Sucesso!\n')
        socketClient.sendall(f'Post adicionado com Sucesso!'.encode('utf-8'))
    if dataReceived.startswith("GET"):
        print('\nProcessamento Método GET')
        respondeHead = f'HTTP/1.1 200 OK\r\n\r\n'
        file = returnPosts()
        response = respondeHead + file.read()
        print('Processamento Método GET Concluido com Sucesso!\n')
        socketClient.sendall(response.encode('utf-8'))
    if dataReceived.startswith("DELETE"):
        print('\nProcessamento Método DELETE')
        resetBase()
        print('Processamento Método DELETE Concluido com Sucesso!\n')
        socketClient.sendall(f'Mural deletado com Sucesso!'.encode('utf-8'))
    socketClient.close()


#Inicialização do Servidor
startServer()
```

Código client.py

```
import socket
from datetime import datetime as date
import json

#Definição dos Dados Fixos da Aplicação
ADDR = '127.0.0.1'
PORT = 8000

#Função que inicia o Cliente
def startClient():
    control = True
    while control:
        option = int(input('Digite a opção desejada:\n1 - Postar no Mural\n2 - Visualizar Mural\n3 - 
Deletar Mural\n4 - Fechar aplicação\n'))
        if option == 1:
            #Objetivo utilizar Mensagem HTTP - POST
            newPost()
```

```
        elif option == 2:
            #Objetivo utilizar Mensagem HTTP - GET
            getPost()
        elif option == 3:
            #Objetivo utilizar Mensagem HTTP - DELETE
            deleteWall()
        elif option == 4:
            control = False
        else:
            print('Tente novamente!')

#Conjunto de Funções Method POST
##Função geral que realiza a inicialização do método
def newPost():
    title = input('Digite o título do Post:\n')
    description = input('Digite a descrição do Post:\n')
    author = input('Digite o nome do Autor do Posto:\n')
    time =  date.now().strftime('%d/%m/%Y %Hh %Mmin')
    obj = constructor(title, description, author, time)
    methodPOST(obj)

##Função que cria o Objeto a ser enviado pelo método
def constructor(title, description,author, time):
    Obj = {
        'title':title,
        'description':description,
        'author':author,
        'time':time
    }
    return Obj

##Função que aplica os conceitos do método POST
def methodPOST(data):
    #Criação do socket Cliente e atribuição da conexão com endereço IP e Número de Porta
    socketClient = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    socketClient.connect((ADDR,PORT))

    #Modificação do dado a ser enviado
    postDataEncode = json.dumps(data).encode('utf-8')

    #Construção do cabeçalho da mensagem POST
    headers = [
        f"POST / HTTP/1.1",
        f"Host: {ADDR}",
        f"Content-Type: application/json",
        f"Content-Length: {len(postDataEncode)}",
```

```
        f"Connection: close",
        "",
    ]
    headerEnconde = "\r\n".join(headers).encode('utf-8')

    #Processamento do envio da mensagem POST e resposta do Servidor
    socketClient.sendall(headerEnconde + postDataEncode)
    resposta = socketClient.recv(1024)
    print('\n',resposta.decode(),'\n')
    socketClient.close()
    return


#Conjunto  de Funções Method GET
##Função geral que realiza a inicialização do método
def getPost():
    print('\n\nMURAL DE POSTS:\n\n')
    message = methodGET()
    message = message.split('\r\n\r\n')[-1]
    message = message[1:len(message)-1]
    displayPosts(message)
    if len(message) == 0:
        print('Sem posts no momento!\nTente novamente mais tarde ou faça um você mesmo
:)\n')
    print()
    return

##Função auxiliar que trata a mensagem recebida do método
def displayPosts(messages):
    #Tratamento dos dados
    messages = messages.split('}')
    listMessages = []
    for message in messages:
        types = message.split(',')
        listAux = []
        for type in types:
            type = type.split(':')[-1]
            type = type[2:len(type)-1]
            listAux.append(type)
        if len(listAux) == 4:
            listMessages.append(listAux)
        elif len(listAux) == 5:
            listAux.remove(listAux[0])
            listMessages.append(listAux)
    #plotagem do Mural
    for m in listMessages:
```

```
        print('   Título do Post: ', m[0])
        print('Descrição do Post: ', m[1])
        print('    Autor do Post: ', m[2])
        print('  Período do Post: ', m[3])
        print()
    return

##Função que aplica os conceitos do método GET
def methodGET():
    #Criação do socket Cliente e atribuição da conexão com endereço IP e Número de Porta
    socketClient = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    socketClient.connect((ADDR,PORT))

    #Construção do cabeçalho da mensagem GET
    headers = [
        f"GET / HTTP/1.1",
        f"Host: {ADDR}",
        f"Connection: close",
        "",
    ]
    headerEnconde = "\r\n".join(headers).encode('utf-8')

    #Processamento do envio da mensagem GET e resposta do Servidor
    socketClient.sendall(headerEnconde)
    resposta = socketClient.recv(1024)
    r = resposta.decode()
    socketClient.close()
    return r


#Conjunto  de Funções Method DELETE
##Função geral que realiza a inicialização do método
def deleteWall():
    password = input('Digite a senha ADM:\n')
    if(password == '1234'):
        methodDelete()
    else:
        print('Senha Incorreta!')
        return

##Função que aplica os conceitos do método DELETE
def methodDelete():
    #Criação do socket Cliente e atribuição da conexão com endereço IP e Número de Porta
    socketClient = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    socketClient.connect((ADDR,PORT))
```

```
    #Construção do cabeçalho da mensagem DELETE
    headers = [
        f"DELETE / HTTP/1.1",
        f"Host: {ADDR}",
        f"Connection: close",
        "",
    ]
    headerEnconde = "\r\n".join(headers).encode('utf-8')

    #Processamento do envio da mensagem DELETE e resposta do Servidor
    socketClient.sendall(headerEnconde)
    resposta = socketClient.recv(1024)
    print('\n',resposta.decode(),'\n')
    socketClient.close()
    return


#Inicialização do Cliente
startClient()
```

Código dataBase.py

```
import json

#Definição dos Dados Fixos da Aplicação
PATH = "dataBase\\dataPosts.json"

#Função que adiciona o Post ao Banco de Dados
def setPost(data):
    with open(PATH) as file:
        listFile = json.load(file)
    listFile.append(editJson(data))
    with open(PATH,"w") as file:
        json.dump(listFile, file)

#Função que modifica o dado para o formato JSON correto
def editJson(data):
    Lista = []
    size = len(data)
    data = data[1:size-1]
    data = data.split(',')
    for dt in data:
        dt = dt.split(':')[-1]
        s = len(dt)
        dt = dt[2:s-1]
        Lista.append(dt)
```

```
    Obj = {
        'title' : Lista[0],
        'description' : Lista[1],
        'author' : Lista[2],
        'time' : Lista[3]
    }
    return Obj

#Função que retorna o Banco de Dados
def returnPosts():
    return open('dataBase\\dataPosts.json','r', encoding='utf-8')

#Função que reseta o Banco de Dados
def resetBase():
    listFile = []
    with open(PATH,"w") as file:
        json.dump(listFile, file)
```

Conclusão

Desenvolver um servidor e um cliente em Python para interações utilizando o protocolo HTTP, permitindo a comunicação entre ambos e a manipulação de um banco de dados em formato JSON.

O servidor é responsável por receber solicitações dos clientes, processar as requisições (métodos POST, GET e DELETE) e responder adequadamente.

O cliente permite ao usuário interagir com o servidor, realizando ações como postar mensagens, visualizar o mural de mensagens e excluir o mural.

O banco de dados é implementado em formato JSON e armazena as informações dos posts, sendo possível adicionar novos posts, retornar o conteúdo do banco de dados e redefinir o banco de dados para um estado vazio.

Através dessas funcionalidades, foi possível criar uma aplicação simples de mural de mensagens, onde os usuários podem compartilhar informações e visualizar as mensagens de outros usuários.

O uso do protocolo HTTP e do formato JSON proporciona uma comunicação eficiente e uma estrutura flexível para o armazenamento e troca de dados.

O trabalho demonstra a aplicação prática de conceitos de programação em Python, manipulação de arquivos JSON, comunicação cliente-servidor e uso de protocolos de rede.