ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет ИКТ

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине: «Web - программирование»

на тему:

«Работа с сокетами»

Выполнила:

студентка группы К33402

Мищенко Виолетта

Преподаватель:

Говоров Антон Игоревич

Санкт-Петербург, 2021 год

Цель: овладеть практическими навыками и умениями реализации web-серверов и использования сокетов.

Оборудование: компьютерный класс.

Ход работы:

1.Реализовать клиентскую и серверную часть приложения. Клиент отсылает серверу сообщение «Hello, server». Сообщение должно отразиться на стороне сервера. Сервер в ответ отсылает клиенту сообщение «Hello, client». Сообщение должно отобразиться у клиента.

Серверная часть:

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #сокет
sock.bind(("localhost",9090))
sock.listen(5) #прослушиваем порт
print("Server is working")
while True:
    client , address = sock.accept()#новый сокет кот подключился,адрес машины,с которой подключаются
    data = client.recv(1024) #buff size, возвр данные
    print(data.decode()) #отражается на стороне сервера
    client.send("Hello, client".encode())
sock.close()
```

Клиентская часть:

```
import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(("localhost", 9090))#подключаемся
sock.send("Hello, server".encode())
data = sock.recv(1024)
print(data.decode())
sock.close()
```

2. Реализовать клиентскую и серверную часть приложения. Клиент запрашивает у сервера выполнение математической операции, параметры, которые вводятся с клавиатуры. Сервер обрабатывает полученные данные и возвращает результат клиенту. Вариант 19 (Поиск площади трапеции.)

Серверная часть:

```python
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #сокет
sock.bind(("localhost",9090))
sock.listen(5) #прослушиваем порт
print("Server is working")
while True:
    client , address = sock.accept()#новый сокет кот подключился,адрес машины,с которой подключаются
    data = client.recv(1024) #buff size, возвр данные
    print(data.decode(), "from ", address[0]) #отражается на стороне сервера
    client.send("Hello, client. Ok, send the footings and the height by the space".encode())
    param = client.recv(1024)
    lst = param.decode().split()
    a = float(lst[0])
    b = float(lst[1])
    h = float(lst[2])
    s = ((a + b) * h)/2
    client.send(str(s).encode())
sock.close()
```

Клиентская часть:

```python
import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(("localhost", 9090))#подключаемся
sock.send("Hello, server. Please, calculate the area of trapezoid".encode())
data = sock.recv(1024)
print(data.decode())
param = input()
sock.send(param.encode())
result = sock.recv(1024)
print(result.decode())
sock.close()
```

3. Реализовать серверную часть приложения. Клиент подключается к серверу. В ответ клиент получает http-сообщение, содержащее html-страницу, которую сервер подгружает из файла index.html.

Файл index.html:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<p>Hello, Violetta!</p>
</body>
</html>
```

Серверная часть:

```
import socket

RESPONSE = """HTTP/1.1 200 OK\r
Content-Type: text/html; charset=utf-8\r
Connection: Keep-Alive\r
Content-Length: """

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #сокет
sock.bind(("localhost",9090))
sock.listen(5) #прослушиваем порт
file = open("index.html", "rb")
html_page = file.read()
file.close()
while True:
    client , address = sock.accept()#новый сокет кот подключился,адрес машины,с которой подключаются
    request = client.recv(1024) #buff size, возвр данные
    answer = RESPONSE + str(len(html_page)) + "\r\n\r\n"
    client.send(answer.encode() + html_page)
sock.close()
```

4. Реализовать двухпользовательский или многопользовательский чат. Реализация многопользовательского часа позволяет получить максимальное количество баллов.

Многопользовательский чат

Серверная часть:

```
import socket
import threading


def send_message_for_all(msg):
    for client in clients:
        client.send(msg)


def get_message_from_client(client_sock):
    while True:
        cur_msg = client_sock.recv(1024)
        send_message_for_all(cur_msg)


sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # сокет
sock.bind(("localhost",9090))
sock.listen(5)  # прослушиваем порт
print("Server is working")
```

```
clients = []
while True:
    client, address = sock.accept()
    clients.append(client)
    data = client.recv(1024)  # Приветственное сообщение
    send_message_for_all(data)
    x = threading.Thread(target=get_message_from_client, args=(client,))
    x.start()  # запустить поток
sock.close()
```

Клиентская часть:

```
import socket
import threading
import time


def recv_messages(client):
    while True:
        msg = client.recv(1024)
        print(msg.decode())

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print("Please, enter your name")
name = input()
sock.connect(("localhost", 9090))#подключаемся
sock.send((name + " joined the chat").encode())
x = threading.Thread(target=recv_messages, args=(sock,))
x.start()
while True:
    message = name + " [" + time.ctime(time.time()) + "]: " + input()
    sock.send(message.encode())
sock.close()
```