Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»
Факультет инфокоммуникационных технологий

Лабораторная работа №1

«Работа с сокетами»

по дисциплине

«Web-программирование»

Выполнил:

студент III курса ФИКТ

группы К33402

Ф.И.О. Кондрашов Егор Юрьевич

Проверил:

Говоров А. И.

Санкт-Петербург

2021

Цель работы:

Реализовать клиентскую и серверную часть четырёх программ на Python, использующих сокеты.

Выполнение работы:

Задание 1:

клиент:

```
import socket


HOST, PORT = "localhost", 9999

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
    sock.connect((HOST, PORT))
    sock.sendall(bytes("Hello, server" + "\n", "utf-8"))

    received = str(sock.recv(1024), "utf-8")
    print(f"Received data: {received}")
```

сервер:

```
import socketserver


class MyTCPHandler(socketserver.BaseRequestHandler):

    def handle(self):
        self.data = self.request.recv(1024).strip()
        print(f"Received data: {self.data.decode()}")
        if self.data.decode() == "Hello, server":
            self.request.sendall(b"Hello, client")
        else:
            self.request.sendall(b"Try again")


if __name__ == "__main__":
    HOST, PORT = "localhost", 9999

      with socketserver.TCPServer((HOST, PORT), MyTCPHandler) as
server:
        server.serve_forever()
```

Задание 2:

сервер:

```
import socketserver


class MyTCPHandler(socketserver.BaseRequestHandler):

    def calculate_area(self, base: float, altitude: float) ->
float:
        """Метод для расчёта площадь параллелограмма
        по заданной стороне и высоте, проведённой к ней"""
        area: float = base * altitude
        return area

    def handle(self):
        self.data = self.request.recv(1024).strip()
        print(f"Received data: {self.data.decode()}")
        try:
            base, altitude = self.data.decode().split()
            base = float(base)
            altitude = float(altitude)
            resp = bytes(str(self.calculate_area(
                base, altitude)) + "\n", "utf-8")
        except ValueError:
            resp = bytes(
                "Необходимо ввести сторону и высоту параллелограмма
через пробел",
                "utf-8")
        self.request.sendall(resp)


if __name__ == "__main__":
    HOST, PORT = "localhost", 9999

    with socketserver.TCPServer((HOST, PORT), MyTCPHandler) as
server:
        server.serve_forever()
```

Клиент:

```
import socket
```

```
HOST, PORT = "localhost", 9999

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
    sock.connect((HOST, PORT))
    params = input("Введите сторону и высоту параллелограмма: ")
    sock.sendall(bytes(params, "utf-8"))

    received = str(sock.recv(1024), "utf-8")
    print(f"Received data: {received}")
```

Задание 3.

Сервер:

```
import socketserver


class MyTCPHandler(socketserver.BaseRequestHandler):

    def handle(self):
        self.data = self.request.recv(1024).strip().decode()
        split_data = self.data.split()
        method = split_data[0]
        path = split_data[1]
        if method == "GET" and path == "/index.html":
            with open("index.html", "rb") as f:
                resp = f.read()
        else:
            resp = b"Unsupported request method or path\n"
        self.request.sendall(resp)


if __name__ == "__main__":
    HOST, PORT = "localhost", 9999

    with socketserver.TCPServer((HOST, PORT), MyTCPHandler) as
server:
        server.serve_forever()
```

Клиент:

```
import socket
```

```
HOST, PORT = "localhost", 9999

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
    sock.connect((HOST, PORT))
    sock.sendall((
        b"GET /index.html HTTP/1.1\r\n" +
          bytes(f"Host: {HOST}\r\nAccept: text/html\r\nConnection:
close\r\n\r\n",
                "utf-8")))
    received = str(sock.recv(1024), "utf-8")
    print(received)
```

Файл index.html:

```
<!DOCTYPE html>
<html>

<head>
    <title>Index page</title>
</head>

<body>

    <h1>Hello World</h1>

</body>

</html>
```

Задание 4.

Сервер:

```
import socketserver


class ThreadedTCPServer(socketserver.ThreadingTCPServer):

    def __init__(self, server_address, request_handler_class):
            super().__init__(server_address, request_handler_class,
True)
        print("Server started")
```

```
            self.receivers = set()

        def add_receiver(self, receiver):
            print("Client connected")
            self.receivers.add(receiver)

        def send_message(self, source, data):
            for receiver in self.receivers:
                if receiver.token != source.token:
                    receiver.request.sendall(data)

        def remove_receiver(self, receiver):
            self.receivers.remove(receiver)


    class ThreadedTCPRequestHandler(socketserver.BaseRequestHandler):

        RECEIVER = 0
        SENDER = 1
        kind = None
        token: str = ""

        def handle(self):
            while True:
                self.data = self.request.recv(1024).strip()
                if self.data:
                    print(f"Received data: {self.data.decode()}")

                    if b"Kind" in self.data:

                        if b"Kind: receiver" in self.data:
                            self.server.add_receiver(self)
                            self.kind = self.RECEIVER
                        elif b"Kind: sender" in self.data:
                            self.kind = self.SENDER

                        token = self.data.decode(
                        )[self.data.decode().find("Token")+6:]
                        self.token = token

                    else:
                        self.server.send_message(self, self.data)
```

```
        def finish(self):
            if self.kind == self.RECEIVER:
                self.server.remove_receiver(self)
            super().finish()


    if __name__ == "__main__":
        HOST, PORT = "localhost", 9999

                    server     =     ThreadedTCPServer((HOST,     PORT),
ThreadedTCPRequestHandler)
        server.serve_forever()
```

Клиент:

```
    import socket
    import threading
    from string import ascii_letters, digits

    from secrets import choice


    HOST, PORT = "localhost", 9999


    def receive_messages(token: str) -> None:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
            sock.connect((HOST, PORT))
            connect_msg = "Kind: receiver\nToken: " + token + "\n"
            sock.sendall(bytes(connect_msg, "utf-8"))
            while True:
                received = sock.recv(1024)
                if received:
                            print("Received message: " + str(received,
"utf-8"))


    def send_messages(token: str) -> None:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
            sock.connect((HOST, PORT))
            connect_msg = "Kind: sender\nToken: " + token + "\n"
            sock.sendall(bytes(connect_msg, "utf-8"))
            while True:
```

```
                inp = input()
                sock.sendall(bytes(inp, "utf-8"))


    if __name__ == "__main__":
        print("To send a message, type it into the terminal and press
enter")
        token = ''.join(choice(
                ascii_letters + digits
        ) for _ in range(16))
        recv = threading.Thread(target=receive_messages, args=(token,))
        recv.start()
        send = threading.Thread(target=send_messages, args=(token,))
        send.start()
```

Вывод:

В ходе работы были написаны 4 программы на Python, использующие сокеты для коммуникации между клиентом и сервером.