

**"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО"**

Факультет «Инфокоммуникационных технологий»  
Направление подготовки «09.03.03 Прикладная информатика»  
Бакалаврская программа «Мобильные и сетевые технологии»

**ОТЧЁТ**  
**по лабораторной работе №1**  
**по дисциплине «Web-программирование»**  
**Тема: «Работа с сокетами»**

**Выполнил:**

\_\_\_\_\_ / Шугинин Ю. А. (К33402)

**Проверил:**

\_\_\_\_\_ / Говоров А. И.

**Дата: 29.10.2021**

**Санкт-Петербург**  
**2021**

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

### Ход работы.

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

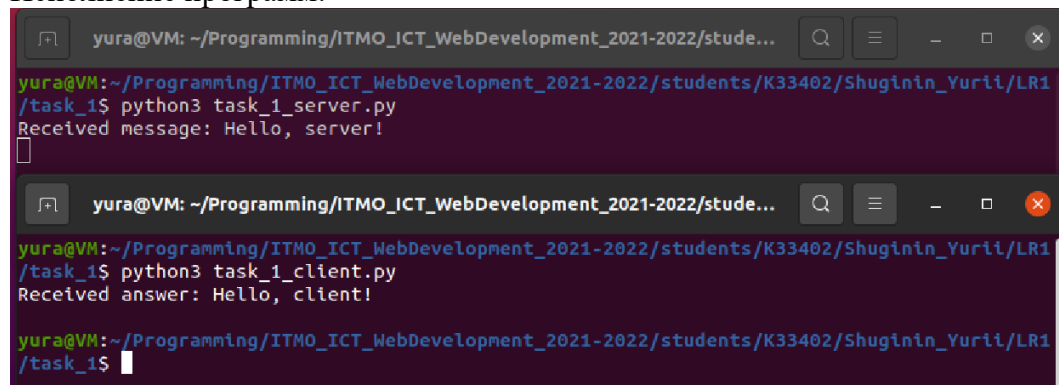
Файл task\_1\_server.py:

```
1  import socket
2
3  conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4  conn.bind(("127.0.0.1", 14900))
5  conn.listen(10)
6
7  while True:
8      try:
9          clientsocket, address = conn.accept()
10         data = clientsocket.recv(16384)
11         udata = data.decode("utf-8")
12         print("Received message: " + udata)
13
14         clientsocket.send(b"Hello, client!")
15
16     except KeyboardInterrupt:
17         print("\n")
18         conn.close()
19         break
20
```

Файл task\_1\_client.py:

```
1  import socket
2
3  conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4  conn.connect(("127.0.0.1", 14900))
5  conn.send(b"Hello, server!")
6
7  data = conn.recv(16384)
8  udata = data.decode("utf-8")
9  print("Received answer: " + udata + "\n")
10
11  conn.close()
12
```

Исполнение программ:



The screenshot shows two terminal windows. The top window shows the execution of the server program: `yura@VM: ~/Programming/ITMO_ICT_WebDevelopment_2021-2022/stude... /task_1$ python3 task_1_server.py`. It receives the message "Hello, server!". The bottom window shows the execution of the client program: `yura@VM: ~/Programming/ITMO_ICT_WebDevelopment_2021-2022/stude... /task_1$ python3 task_1_client.py`. It receives the answer "Hello, client!".

**Задание 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 С – Поиск площади трапеции.

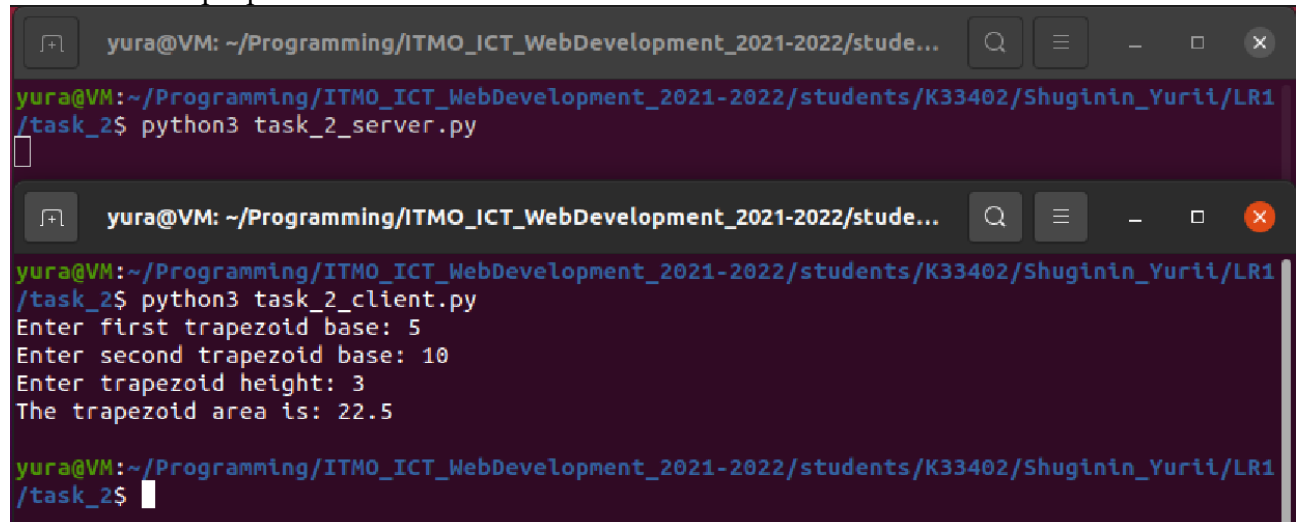
Файл task\_2\_server.py:

```
1  import socket
2
3  conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4  conn.bind(("127.0.0.1", 14900))
5  conn.listen(10)
6
7  while True:
8      try:
9          clientsocket, address = conn.accept()
10
11         data_bin = clientsocket.recv(16384)
12         data = data_bin.decode("utf-8")
13         data = data.split(" ")
14
15         area = (float(data[0]) + float(data[1])) * float(data[2]) / 2
16         clientsocket.send(str(area).encode())
17
18     except ValueError:
19         print("Received incorrect data. Waiting for another...")
20         clientsocket.close()
21
22     except KeyboardInterrupt:
23         print("\n")
24         conn.close()
25         break
26
```

Файл task\_2\_client.py:

```
1  import socket
2
3  conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4  conn.connect(("127.0.0.1", 14900))
5
6  data = input("Enter first trapezoid base: ").replace(" ", "")
7  data += " " + input("Enter second trapezoid base: ").replace(" ", "")
8  data += " " + input("Enter trapezoid height: ").replace(" ", "")
9  conn.send(data.encode())
10
11  area_bin = conn.recv(16384)
12  area = area_bin.decode("utf-8")
13
14  if area:
15      print("The trapezoid area is: " + area + "\n")
16  else:
17      print("Server couldn't calculate the area. \n")
18
19  conn.close()
20
```

Исполнение программ:



```
yura@VM: ~/Programming/ITMO_ICT_WebDevelopment_2021-2022/stude...
yura@VM:~/Programming/ITMO_ICT_WebDevelopment_2021-2022/students/K33402/Shuginin_Yurii/LR1/task_2$ python3 task_2_server.py

yura@VM:~/Programming/ITMO_ICT_WebDevelopment_2021-2022/students/K33402/Shuginin_Yurii/LR1/task_2$ python3 task_2_client.py
Enter first trapezoid base: 5
Enter second trapezoid base: 10
Enter trapezoid height: 3
The trapezoid area is: 22.5

yura@VM:~/Programming/ITMO_ICT_WebDevelopment_2021-2022/students/K33402/Shuginin_Yurii/LR1/task_2$
```

**Задание 3.** Необходимо написать простой веб-сервер для обработки GET и POST http-запросов средствами Python и библиотеки socket. Сделать сервер, который может:

- Принять и записать информацию о дисциплине и оценке по дисциплине.
- Отдать информацию обо всех оценках по дисциплине в виде html-страницы.

Файл task\_3\_server.py



```
1  import socket
2  import sys
3  from email.parser import Parser
4  from functools import lru_cache
5  from urllib.parse import parse_qs, urlparse
6
7  MAX_LINE = 64*1024
8  MAX_HEADERS = 100
9
10 class MyHTTPServer:
11     def __init__(self, host, port, server_name):
12         self._host = host
13         self._port = port
14         self._server_name = server_name
15         self._journal = {}
16
17     def serve_forever(self):
18         serv_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM, proto=0)
19
20         try:
21             serv_sock.bind((self._host, self._port))
22             serv_sock.listen()
23
24             while True:
25                 conn, _ = serv_sock.accept()
26                 try:
27                     self.serve_client(conn)
28                 except Exception as e:
29                     print('Client "forever" serving failed:', e)
30
```

```

31         finally:
32             serv_sock.close()
33
34     def serve_client(self, conn):
35         try:
36             req = self.parse_request(conn)
37             resp = self.handle_request(req)
38             self.send_response(conn, resp)
39         except ConnectionResetError:
40             conn = None
41         except Exception as e:
42             print('Client serving failed:', e)
43
44         if conn:
45             conn.close()
46
47     def parse_request(self, conn):
48         rfile = conn.makefile('rb')
49         raw = rfile.readline(MAX_LINE + 1)
50         if len(raw) > MAX_LINE:
51             raise Exception('Request line is too long')
52
53         req_line = str(raw, 'iso-8859-1')
54         req_line = req_line.rstrip('\r\n')
55         words = req_line.split()
56         if len(words) != 3:
57             raise Exception('Malformed request line')
58
59         method, target, ver = words
60         if ver != 'HTTP/1.1':
61             raise Exception('Unexpected HTTP version')
62
63         headers = self.parse_headers(rfile)
64         host = headers.get('Host')
65         if not host:
66             raise Exception('Bad request')
67         if host not in (self._server_name, f'{self._server_name}:{self._port}'):
68             raise Exception('Host not found')
69
70         return Request(method, target, ver, headers, rfile)
71
72     def parse_headers(self, rfile):
73         headers = []
74         while True:
75             line = rfile.readline(MAX_LINE + 1)
76             if len(line) > MAX_LINE:
77                 raise Exception('Header line is too long')
78
79             if line in (b'\r\n', b'\n', b''):
80                 break
81
82             headers.append(line)
83             if len(headers) > MAX_HEADERS:
84                 raise Exception('Too many headers')
85
86         sheaders = b''.join(headers).decode('iso-8859-1')
87         return Parser().parsestr(sheaders)
88

```

```

89     def handle_request(self, req):
90         if req.path == '/journal' and req.method == 'POST':
91             return self.handle_post_mark(req)
92
93         if req.path.startswith('/journal/') and req.method == 'GET':
94             subject = req.path[len('/journal/'): ]
95             return self.handle_get_subject(req, subject)
96
97         raise Exception('Request not found')
98
99     def handle_post_mark(self, req):
100         subject = req.query['subject'][0]
101         if subject not in self._journal.keys():
102             self._journal[subject] = []
103         self._journal[subject].append(req.query['mark'][0])
104         return Response(204, 'Added')
105
106     def handle_get_subject(self, req, subject):
107         if subject not in self._journal.keys():
108             raise Exception('Request not found')
109
110         contentType = 'text/html; charset=utf-8'
111         body = '<html><head></head><body>'
112
113         answer_line = f'{subject}: '
114         for mark in self._journal[subject]:
115             answer_line += mark + ' '
116
117         body += f'<div>{answer_line}</div>'
118         body += '</body></html>'
119
120         body = body.encode('utf-8')
121         headers = [('Content-Type', contentType), ('Content-Length', len(body))]
122         return Response(200, 'OK', headers, body)
123
124     def send_response(self, conn, resp):
125         wfile = conn.makefile('wb')
126         status_line = f'HTTP/1.1 {resp.status} {resp.reason}\r\n'
127         wfile.write(status_line.encode('iso-8859-1'))
128
129         if resp.headers:
130             for (key, value) in resp.headers:
131                 header_line = f'{key}: {value}\r\n'
132                 wfile.write(header_line.encode('iso-8859-1'))
133
134         wfile.write(b'\r\n')
135
136         if resp.body:
137             wfile.write(resp.body)
138
139         wfile.flush()
140         wfile.close()
141
142
143     class Request:
144         def __init__(self, method, target, version, headers, rfile):
145             self.method = method
146             self.target = target

```

```

147         self.version = version
148         self.headers = headers
149         self.rfile = rfile
150
151     @property
152     def path(self):
153         return self.url.path
154
155     @property
156     @lru_cache(maxsize=None)
157     def query(self):
158         return parse_qs(self.url.query)
159
160     @property
161     @lru_cache(maxsize=None)
162     def url(self):
163         return urlparse(self.target)
164
165
166 class Response:
167     def __init__(self, status, reason, headers=None, body=None):
168         self.status = status
169         self.reason = reason
170         self.headers = headers
171         self.body = body
172
173
174 if __name__ == '__main__':
175     host = sys.argv[1]
176     port = int(sys.argv[2])
177     name = sys.argv[3]
178
179     serv = MyHTTPServer(host, port, name)
180     try:
181         serv.serve_forever()
182     except KeyboardInterrupt:
183         pass
184

```

Исполнение и проверка работы:

```

yura@VM:~/Programming/Python/Web_programming$ python3 3_sr.py 127.0.0.1 53210 example.local

```

```

yura@VM:~/Programming/Python/Web_programming$ nc localhost 53210
POST /journal?subject=Info&mark=5 HTTP/1.1
Host: example.local

HTTP/1.1 204 Added

^C
yura@VM:~/Programming/Python/Web_programming$ nc localhost 53210
POST /journal?subject=Info&mark=4 HTTP/1.1
Host: example.local

HTTP/1.1 204 Added

^C
yura@VM:~/Programming/Python/Web_programming$ nc localhost 53210
GET /journal/Info HTTP/1.1
Host: example.local

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 60

<html><head></head><body><div>Info: 5 4 </div></body></html>
^C
yura@VM:~/Programming/Python/Web_programming$

```

**Задание 4.** Реализовать двухпользовательский или многопользовательский чат. Реализация многопользовательского чата позволяет получить максимальное количество баллов.

Файл task\_4\_server.py:

```
1  import socket
2  import threading
3
4  def display_msg(author, msg):
5      for client in clients:
6          if author != client:
7              client.send(msg.encode())
8
9
10 def client_handler(sock, address):
11     print(f"{address[0]}:{address[1]} connected")
12
13     while True:
14         message = sock.recv(16384).decode("utf-8")
15         if len(message) == 0:
16             break
17         message = f"{address[0]}:{address[1]}: " + message
18         display_msg(sock, message)
19
20     print(f"{address[0]}:{address[1]} disconnected")
21     clients.remove(sock)
22     sock.close()
23
24
25 serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
26 server_address = ("127.0.0.1", 9090)
27 serversocket.bind(server_address)
28 serversocket.listen(100)
29 clients = []
30
31 print(f"Starting Chat Server at {server_address[0]}:{server_address[1]}")
32 try:
33     while True:
34         clientsocket, client_address = serversocket.accept()
35         if clientsocket not in clients:
36             clients.append(clientsocket)
37         client_thread = threading.Thread(target=client_handler, args=(clientsocket, client_address))
38         client_thread.start()
39
40 except KeyboardInterrupt:
41     print("\n" + "Shutting down" + "\n")
42     serversocket.close()
43
```



Файл task\_4\_client.py:

```
1  import socket
2  import threading
3  import time
4
5  def get_msgs():
6      while not exit_event.is_set():
7          try:
8              msg = conn.recv(16384).decode("utf-8")
9              print(msg)
10         except socket.error:
11             time.sleep(0.25)
12             continue
13
14
15  conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
16  conn.connect(("127.0.0.1", 9090))
17  conn.setblocking(False)
18  print("Connecting to the Chat")
19
20  exit_event = threading.Event()
21  get_thread = threading.Thread(target=get_msgs)
22  get_thread.start()
23
24  try:
25      while True:
26          message = input()
27          if message:
28              conn.send(message.encode())
29
30  except KeyboardInterrupt:
31      print("\n" + "Disconnecting" + "\n")
32
33      exit_event.set()
34      get_thread.join()
35
36      conn.close()
37
```

Исполнение программ:

```
yura@VM: ~/Programming/ITMO_ICT_WebDevelopment_2021-2022/stude...
yura@VM:~/Programming/ITMO_ICT_WebDevelopment_2021-2022/students/K33402/Shuginin_Yurii/LR1
/task_4$ python3 task_4_server.py
Starting Chat Server at 127.0.0.1:9090
127.0.0.1:46486 connected
127.0.0.1:46488 connected
127.0.0.1:46486 disconnected
127.0.0.1:46488 disconnected

/task_4$ python3 task_4_client.py
Connecting to the Chat
Hello from user_1!
127.0.0.1:46488: Hello from user_2!
^C
Disconnecting

yura@VM:~/Programming/ITMO_ICT_WebDevelopment_2021-2022/students/K33402/Shuginin_Yurii/LR1
/task_4$

yura@VM:~/Programming/ITMO_ICT_WebDevelopment_2021-2022/students/K33402/Shuginin_Yurii/LR1
/task_4$ python3 task_4_client.py
Connecting to the Chat
127.0.0.1:46486: Hello from user_1!
Hello from user_2!
^C
Disconnecting

yura@VM:~/Programming/ITMO_ICT_WebDevelopment_2021-2022/students/K33402/Shuginin_Yurii/LR1
/task_4$
```

**Вывод:** в процессе выполнения лабораторной работы были получены практические навыки реализации web-серверов на языке Python. Были изучены методы работы с сокетами, потоками, а также протоколом передачи данных HTTP.