Федеральное государственное автономное образовательное учреждение высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий  
Кафедра «Информатика и информационные технологии»

Направление подготовки/ специальность: \_\_\_информационные системы и технологии\_\_\_\_\_

ОТЧЕТ

по Индивидуальному вариативному заданию

Студент: \_Кононова Елизавета Андреевна\_\_ Группа: \_241-332\_\_\_\_\_\_\_\_\_\_\_

Место прохождения практики: Московский Политех, кафедра «Информатика и информационные технологии»

Отчет принят с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики: Худайбердиева Гулшат

Москва 2025

**Создание HTTP-сервера на Python с нуля**

**Цель:** реализовать простой сервер на Python с использованием сокетов.

**Задачи:**

1. Изучить HTTP, ключевые компоненты,
2. Изучить работу сервера.
3. Скомпилировать код для работы сервера.

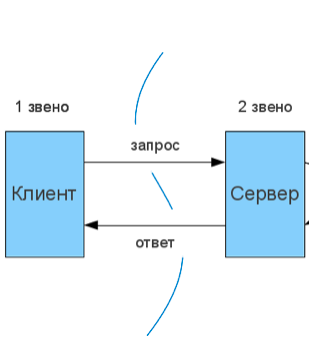
**Ход работы:**

1. **Исследование предметной области**HTTP (HyperText Transfer Protocol) — протокол прикладного уровня для передачи гипертекстовых документов.

Ключевые компоненты:

* Запросы (Request)
* Ответы (Response)
* Методы (GET, POST)
* Коды состояния (200, 404, 500)

Как работает сервер:



1. **Практическая реализация**
   1. Для начала создадим папку для работы с сервером, например my\_server в папке пользователя на диске C:\

2. В ней создадим файл server.py с кодом для нашего сервера и папку static для хранения статических файлов.

3. Теперь создадим базовый код:  
import socket  
import os  
import threading  
from datetime import datetime

class HTTPServer:  
def **init**(self, host='127.0.0.1', port=8080):

1. Инициализируем сервер, добавим хост - ip адрес для привязки, порт прослушивания, папку со статическими файлами  
   self.host = host  
   self.port = port  
   self.server\_socket = None #Основной сокет сервера  
   self.static\_dir = 'static'  
   self.running = False #Флаг работы сервера  
   if not os.path.exists(self.static\_dir): #Создаем папку static если ее нет  
   os.makedirs(self.static\_dir)  
   print(f"Создана папка {self.static\_dir}")
2. Теперь запуск сервера и настройка сокета  
   def start(self):  
   try:  
   # 1. Создаем TCP-сокет (IPv4, потоковый)  
   self.server\_socket = socket.socket(  
   socket.AF\_INET, # IPv4  
   socket.SOCK\_STREAM # TCP  
   )

Copy

Download

# 2. Разрешаем повторное использование адреса

self.server\_socket.setsockopt(

socket.SOL\_SOCKET,

socket.SO\_REUSEADDR,

1

)

# 3. Привязываем сокет к адресу

self.server\_socket.bind((self.host, self.port))

# 4. Начинаем слушать соединения (макс. 5 в очереди)

self.server\_socket.listen(5)

self.running = True

print(f"Сервер запущен на http://{self.host}:{self.port}")

# 5. Запускаем основной цикл обработки

self.serve\_forever()

except Exception as e:  
print(f"Ошибка запуска: {e}")  
self.stop()

1. Теперь создадим основной цикл обработки соединений  
   def serve\_forever(self):  
   try:  
   while self.running:  
   try:  
   # Устанавливаем таймаут 1 сек для проверки флага running  
   self.server\_socket.settimeout(1)

Copy

Download

# Принимаем новое соединение

client\_conn, client\_addr = self.server\_socket.accept()

# Создаем поток для обработки запроса

thread = threading.Thread(

target=self.handle\_request,

args=(client\_conn, client\_addr),

daemon=True # Поток завершится с основным

)

thread.start()

except socket.timeout:

continue # Таймаут для проверки running

except KeyboardInterrupt:  
print("\nСервер останавливается...")  
finally:  
self.stop()

1. Создадим обработку одного клиентского запроса  
   def handle\_request(self, client\_conn, client\_addr):  
   try:  
   # 1. Получаем данные запроса (макс. 4KB)  
   request\_data = client\_conn.recv(4096).decode('utf-8')  
   if not request\_data:  
   return

Copy

Download

# 2. Парсим первую строку запроса (например: GET / HTTP/1.1)

first\_line = request\_data.split('\r\n')[0]

parts = first\_line.split()

if len(parts) < 2:

return

method, path = parts[0], parts[1] # Метод и путь

# 3. Генерируем упрощенный ответ

response = self.generate\_simple\_response(method, path)

# 4. Отправляем ответ

client\_conn.sendall(response.encode('utf-8'))

except Exception as e:  
print(f"Ошибка обработки: {e}")  
error\_msg = "HTTP/1.1 500 Error\r\n\r\nServer Error"  
client\_conn.sendall(error\_msg.encode('utf-8'))

finally:  
client\_conn.close() # Закрываем соединение

1. Наконец создадим генерацию ответов сервера  
   def generate\_simple\_response(self, method, path):  
   if method != 'GET':  
   return "HTTP/1.1 405 Method Not Allowed\r\n\r\n"

Нормализуем путь

if path == '/':  
path = '/index.html'

Пытаемся найти файл

file\_path = os.path.join(self.static\_dir, path.lstrip('/'))

if os.path.isfile(file\_path):  
try:  
with open(file\_path, 'r') as f:  
content = f.read()

Copy

Download

# Минимальные необходимые заголовки

headers = [

"HTTP/1.1 200 OK",

f"Content-Length: {len(content)}",

"Connection: close",

"\r\n"

]

return "\r\n".join(headers) + content

except Exception:

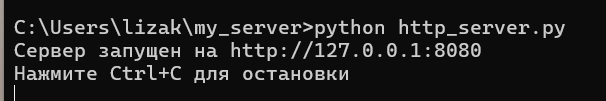
return "HTTP/1.1 500 Error\r\n\r\n"

else:  
return "HTTP/1.1 404 Not Found\r\n\r\n"

if **name** == "**main**":  
server = HTTPServer()  
server.start()

**Готово!**

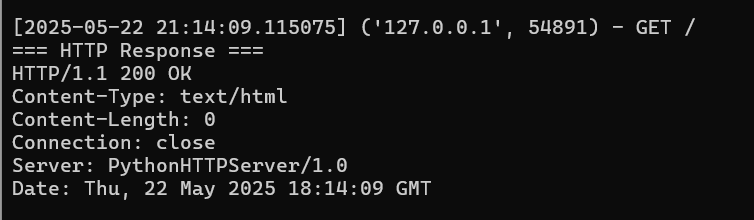
**Примеры работы**  
В терминале вводим python



Мы видим пустую белую страницу, это означает, что сервер запущен и работает



Пример ответа сервера



**Модификация**  
Добавим счетчик посещений

def **init**(self, host='127.0.0.1', port=8080):  
self.host = host  
self.port = port  
self.server\_socket = None  
self.static\_dir = 'static'  
self.running = False  
self.visits = 0

В метод handle\_request добавляем увеличение счетчика:  
def handle\_request(self, client\_conn, client\_addr):  
try:  
self.visits += 1 # <- Добавить эту строку (увеличиваем счетчик)

Copy

Download

request\_data = client\_conn.recv(4096).decode('utf-8')

if not request\_data:

return

В метод generate\_response добавляем вывод счетчика в страницу:  
if os.path.isfile(file\_path):  
try:  
with open(file\_path, 'r') as f:  
content = f.read()

Copy

Download

# Добавить эти 2 строки (вставка счетчика в HTML):

if path.endswith('.html'):

content = content.replace('</body>', f'<footer>Visits: {self.visits}</footer></body>')

**Вывод:** изучен HTTP, принцип его работы и способ реализации простого HTTP сервера на Python, добавлена модификация.

**Список использованной литературы:**

1. Python: Building a basic HTTP Server from scratch in Python <https://joaoventura.net/blog/2017/python-webserver/>
2. http.server — HTTP servers  
   <https://docs.python.org/3/library/http.server.html>
3. Python HTTP сервер: обработка GET и POST запросов  
   <https://sky.pro/wiki/python/python-http-server-obrabotka-get-i-post-zaprosov/>