



**Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника

ОТЧЕТ

по лабораторной работе № 1

Название: Взаимодействие с web сервисом через протокол HTTP

Дисциплина: Языки интернет программирования

Студент

ИУ6-64Б
(Группа)

(Подпись, дата)

Д. О. Романов
(И.О. Фамилия)

Преподаватель

(Подпись, дата)

В. И. Кузнецов
(И.О. Фамилия)

Москва, 2026

1. ЦЕЛЬ РАБОТЫ

Цель работы - знакомство с протоколом HTTP, изучение структуры HTTP-запросов и HTTP-ответов, практическое освоение методов HTTP, кодов состояния HTTP, а также получение навыков работы с HTTP с помощью инструментов разработчика браузера Safari и специализированного HTTP-клиента Insomnia.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. Назначение протокола HTTP

HTTP (Hypertext Transfer Protocol) - протокол передачи гипертекста, один из основных протоколов прикладного уровня стека TCP/IP. HTTP обеспечивает обмен данными между клиентом (например, браузером) и сервером в режиме «запрос-ответ». Основное назначение HTTP - загрузка веб-страниц и других ресурсов (изображений, видео, данных API) с веб-серверов. Протокол работает поверх TCP (по умолчанию на 80-м порту для HTTP и 443-м для HTTPS - защищенной версии с шифрованием). Актуальные версии протокола: HTTP/1.1 (RFC 9112), HTTP/2 (RFC 9113) и HTTP/3 (RFC 9114). В данной лабораторной работе рассматривается логика работы HTTP/1.1 и общая семантика HTTP (RFC 9110).

2.2. Структура URL

URL (Uniform Resource Locator) - унифицированный указатель местонахождения ресурса. Состав URL:

1. схема (протокол): http, https, ftp и др;
2. адрес сервера: доменное имя (например, www.example.com) или IP-адрес;
3. порт (необязательно): если не указан, используется порт по умолчанию;
4. путь к ресурсу: например, /courses/networks;
5. параметры запроса (необязательно): строка вида ?key1=value1&key2=value2;

6. якорь / фрагмент (необязательно): #section.

Пример полного URL:

http://www.example.com:8080/path/to/resource?id=123&lang=ru#top

2.3. Структура HTTP-запроса

HTTP-запрос состоит из трех частей:

1. стартовая строка (request line): Метод URL Версия_протокола

Пример: GET /courses/networks HTTP/1.1;

2. заголовки запроса (request headers):

Каждый заголовок имеет формат: Название_заголовка: значение

Пример: Host: www.w3.org. Заголовок Host обязателен в HTTP/1.1 и указывает доменное имя сервера;

3. тело запроса (request body) - необязательная часть, используется в методах POST, PUT, PATCH для передачи данных на сервер.

2.4. Методы HTTP

Метод HTTP указывает, какое действие клиент хочет выполнить с ресурсом.

Основные методы:

- GET: запрос на получение ресурса;
- POST: передача данных на сервер для обработки;
- PUT: размещение или замена ресурса на сервере;
- DELETE: удаление ресурса на сервере;
- HEAD: запрос заголовков ресурса без передачи его тела;
- OPTIONS: запрос поддерживаемых методов для данного ресурса;
- PATCH: частичное изменение ресурса.

Наиболее часто используются GET (для запроса страниц и данных) и POST (для отправки форм и данных).

2.5. Структура HTTP-ответа

HTTP-ответ также состоит из трех частей:

1. строка статуса (status line): Версия_протокола Код_статуса Пояснение
Пример: HTTP/1.1 200 OK;
2. заголовки ответа (response headers):
Пример: Content-Type: text/html; charset=UTF-8 Content-Length: 5161;
3. тело ответа (response body) - содержимое ресурса (HTML-код страницы, JSON, изображение и т.п.).

2.6. Классы кодов состояния HTTP

Код состояния HTTP - это трехзначное число, первая цифра которого определяет класс ответа:

1. 1xx (информационные): запрос принят, обработка продолжается.
Пример: 101 Switching Protocols;
2. 2xx (успешное выполнение): запрос успешно обработан.
Пример: 200 OK (успех), 201 Created (ресурс создан);
3. 3xx (перенаправление): необходимо выполнить дополнительные действия.
Пример: 301 Moved Permanently (постоянное перемещение), 302 Found (временное перемещение), 304 Not Modified (ресурс не изменился, можно взять из кэша);
4. 4xx (ошибка клиента): запрос содержит ошибку или не может быть выполнен.
Пример: 400 Bad Request (неверный синтаксис), 403 Forbidden (доступ запрещен), 404 Not Found (ресурс не найден);
5. 5xx (ошибка сервера): сервер не смог выполнить корректный запрос из-за внутренней ошибки.
Пример: 500 Internal Server Error (внутренняя ошибка сервера), 501 Not Implemented (функциональность не реализована).

3. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

3.1. Анализ запросов к www.w3.org в браузере Safari

Для анализа HTTP-запросов использовался браузер Safari со встроенными инструментами разработчика (Web Inspector). Последовательность действий:

1. В Safari включено меню «Разработка» (Safari - Настройки - Дополнения - «Показывать меню „Разработка“ в строке меню»);
2. Открыта вкладка Network в Web Inspector (Разработка - Показать веб-инспектор, затем вкладка «Сеть»);
3. В адресной строке введен адрес <https://www.w3.org> и выполнена загрузка страницы;
4. В списке запросов выбран первичный запрос типа document к ресурсу <https://www.w3.org/>.

Основные параметры запроса и ответа:

Метод: GET

URL: <https://www.w3.org/>

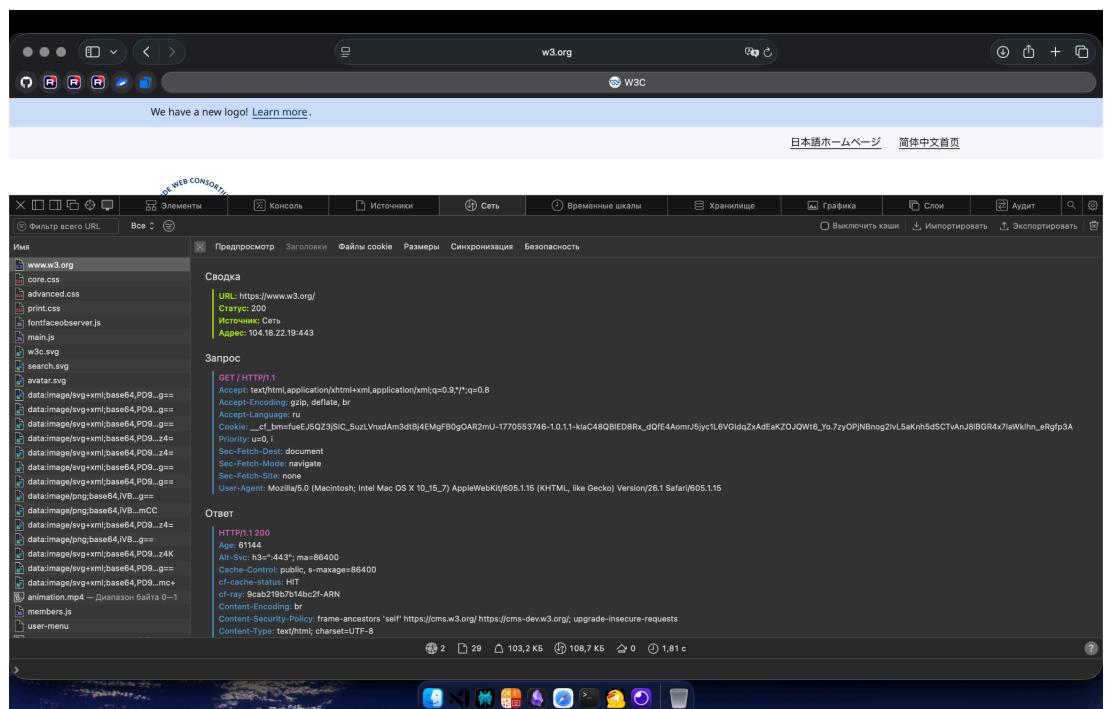


Рисунок 1 - Вкладка Headers в Safari Web Inspector с заголовками запроса и ответа.

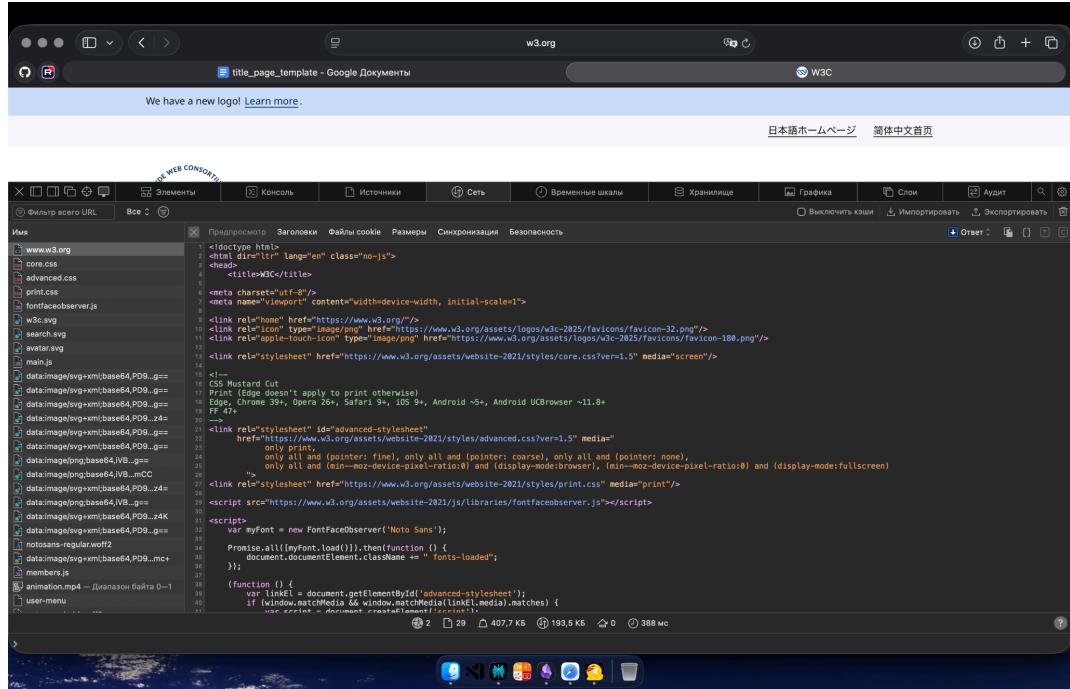


Рисунок 2 - Вкладка Response в Safari Web Inspector с HTML-кодом

3.2. Запрос к www.w3.org с помощью Insomnia

Для повторного выполнения запроса к www.w3.org вне браузера использовался специализированный HTTP-клиент Insomnia. Последовательность действий:

- 1) Установлен Insomnia для macOS.
- 3) Создан новый HTTP-запрос.
- 4) Указан метод GET и URL <https://www.w3.org>.
- 5) Нажата кнопка Send.

Результат запроса:

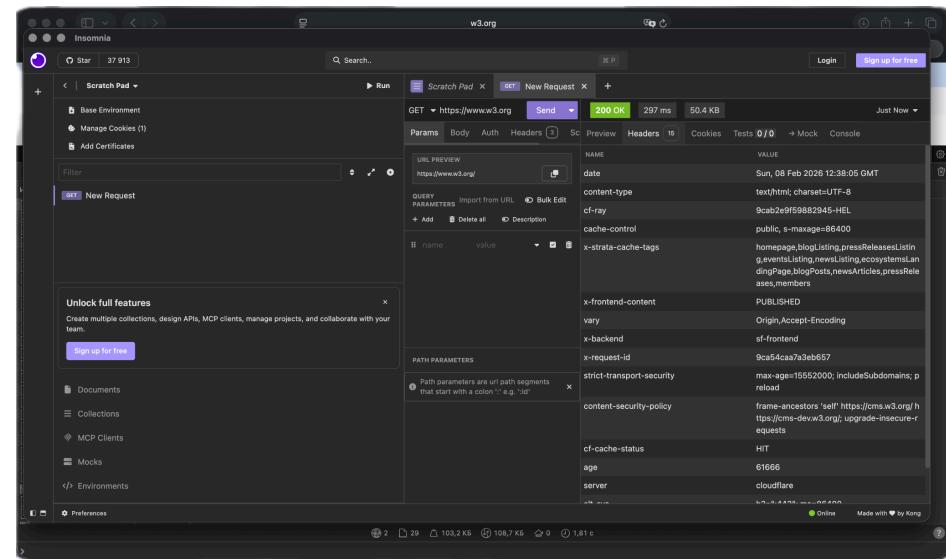
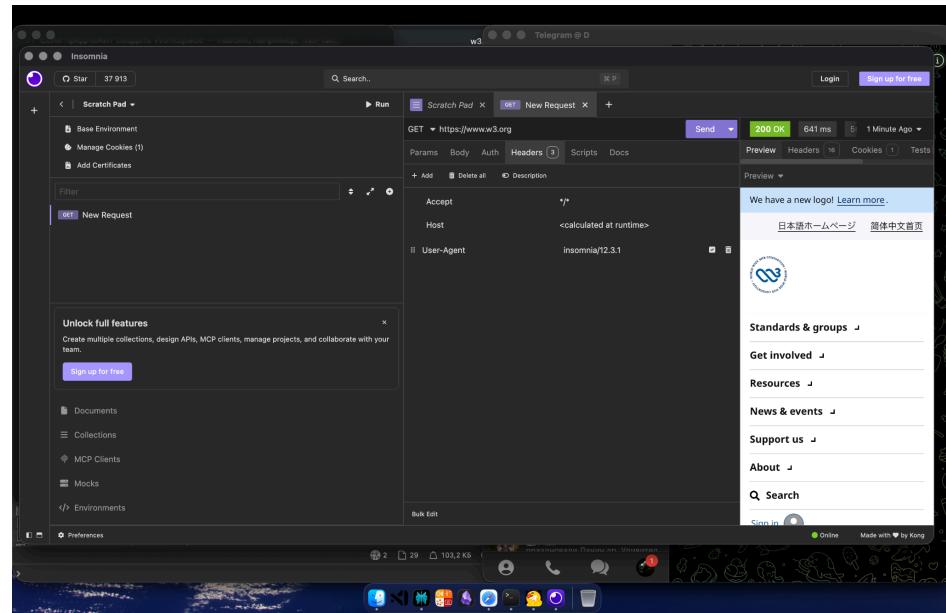


Рисунок 3,4 - Окно Insomnia с результатом GET <https://www.w3.org>, видны метод, URL, статус, фрагмент Body и заголовки

3.3. Локальные запросы к тестовому серверу

3.3.1. Запуск локального сервера

Для выполнения локальных запросов использовался тестовый веб-сервер на базе Flask (Python).

Последовательность действий:

1) Установлен Flask командой:

```
python3 -m pip install flask
```

2) Скачен файл base.py из репозитория:

<https://github.com/Emetless/YaIP/blob/main/lb1/base.py>

3) Сервер запущен командой:

python3 base.py

4) Сервер поднялся по адресу http://127.0.0.1:5000 (или http://localhost:5000).

Все последующие запросы направлялись на этот локальный адрес.

3.3.2. GET без указания пути (/)

Запрос:

Метод: GET

URL: http://127.0.0.1:5000/

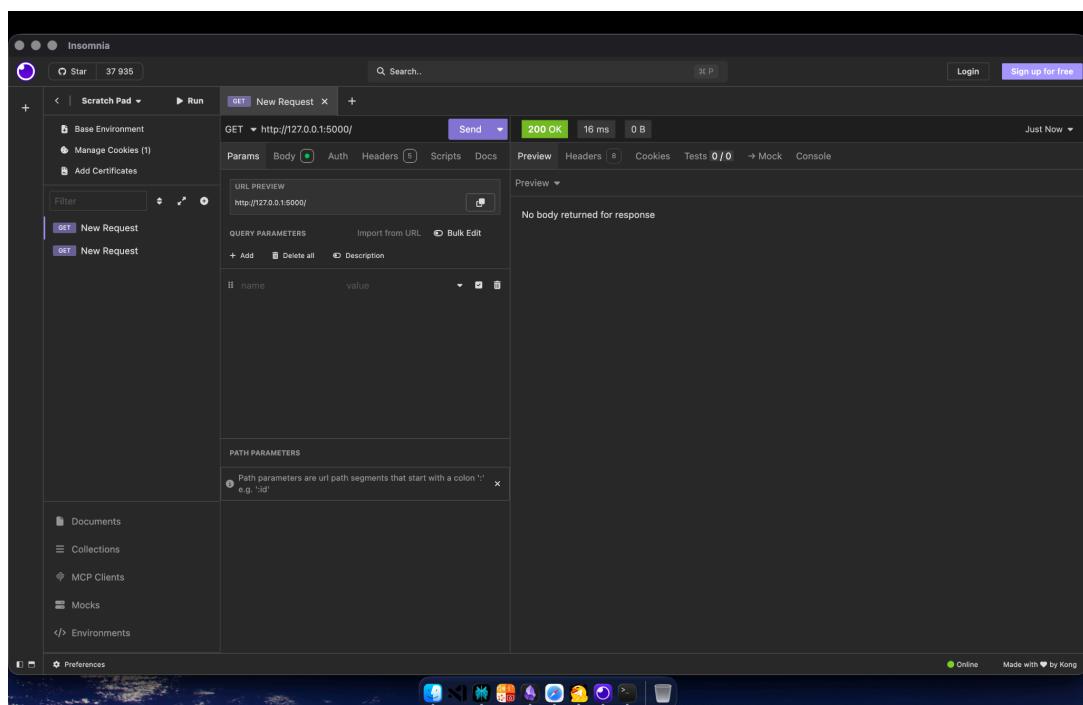


Рисунок 5 - Insomnia, запрос GET / с кодом ответа и телом

3.3.3. GET /success

Запрос:

Метод: GET

URL: http://127.0.0.1:5000/success

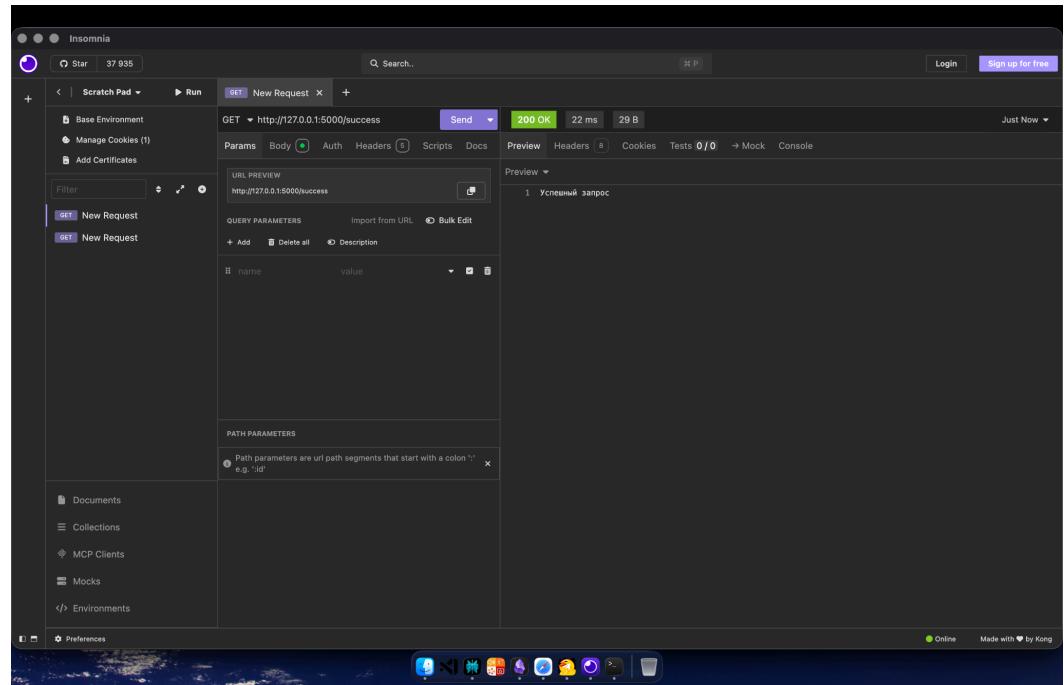


Рисунок 6 - Insomnia, запрос GET /success

3.3.4. GET /unsuccess

Запрос:

Метод: GET

URL: http://127.0.0.1:5000/unsuccess

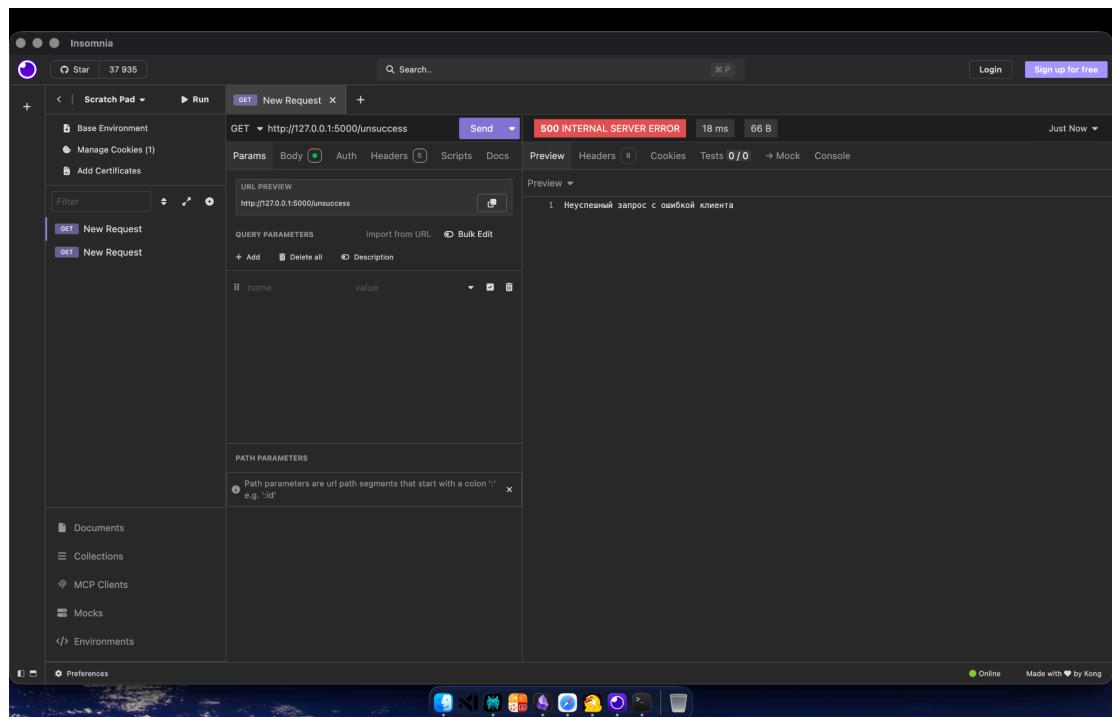


Рисунок 7 - Insomnia, запрос GET /unsuccess с кодом ошибки

Маршрут /unsuccess специально возвращает код ошибки (4xx или 5xx) для демонстрации обработки ошибочных ситуаций.

3.3.5. GET /check с заголовком MyHeader

Вариант 1: MyHeader = check

Запрос:

Метод: GET

URL: http://127.0.0.1:5000/check

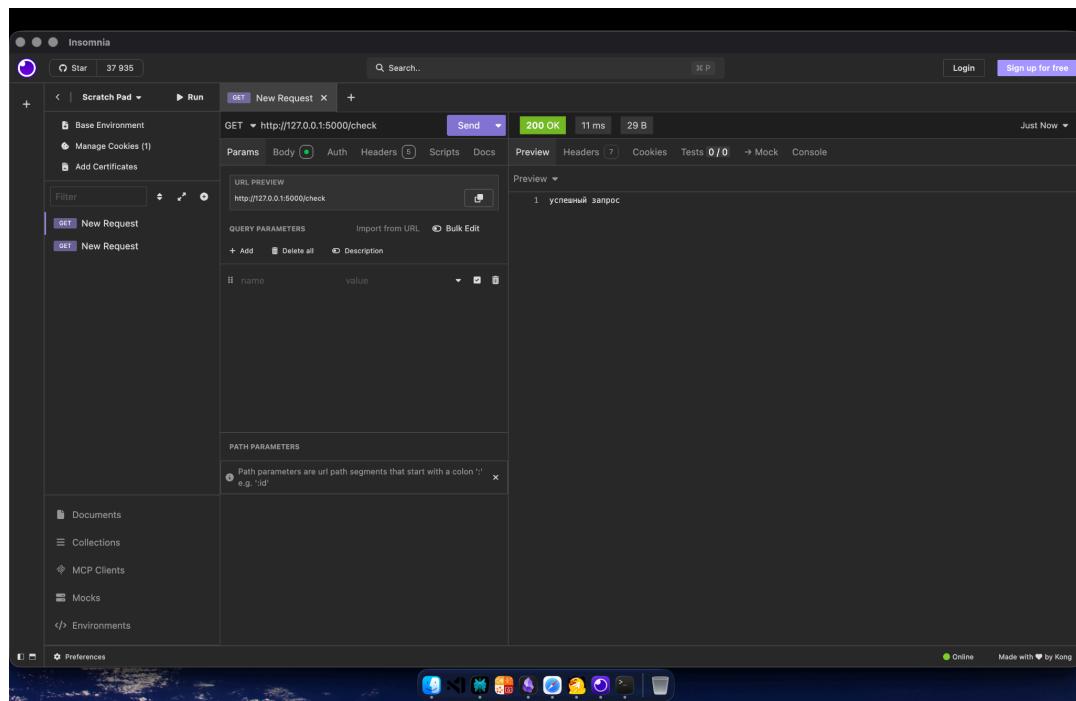


Рисунок 8 - Insomnia, запрос GET /check с MyHeader=check

Вариант 2: MyHeader = other

Запрос:

Метод: GET

URL: http://127.0.0.1:5000/check

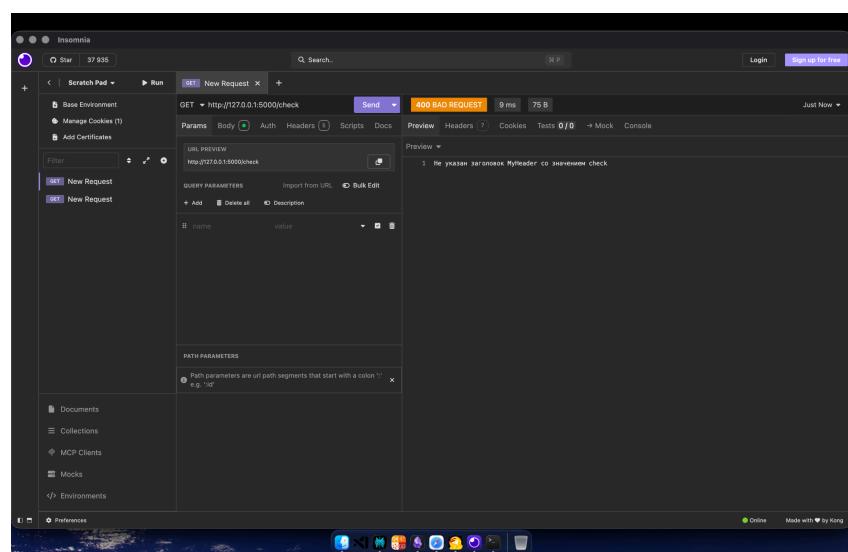


Рисунок 9 - Insomnia, запрос GET /check с MyHeader=other

При передаче заголовка MyHeader с любым другим значением сервер возвращает ошибку, показывая, что проверка значения заголовка выполняется на стороне сервера.

3.3.6. GET /getparam с параметром запроса

Запрос:

Метод: GET

URL: <http://127.0.0.1:5000/getparam?name=Bauman>

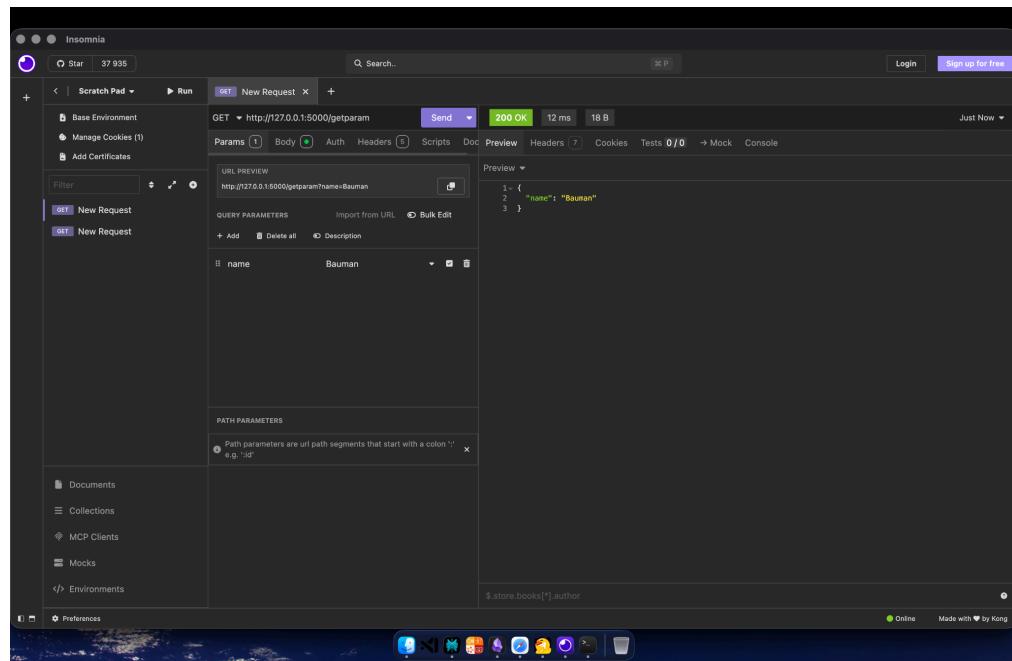


Рисунок 10 - Insomnia, запрос GET /getparam?name=Bauman, видна вкладка
Query с параметром и тело ответа

Сервер принимает параметр запроса name и возвращает его значение в теле ответа, демонстрируя корректную обработку query-параметров.

3.3.7. POST /post/check с телом запроса

Вариант 1: тело = "check"

Запрос:

Метод: POST

URL: <http://127.0.0.1:5000/post/check>

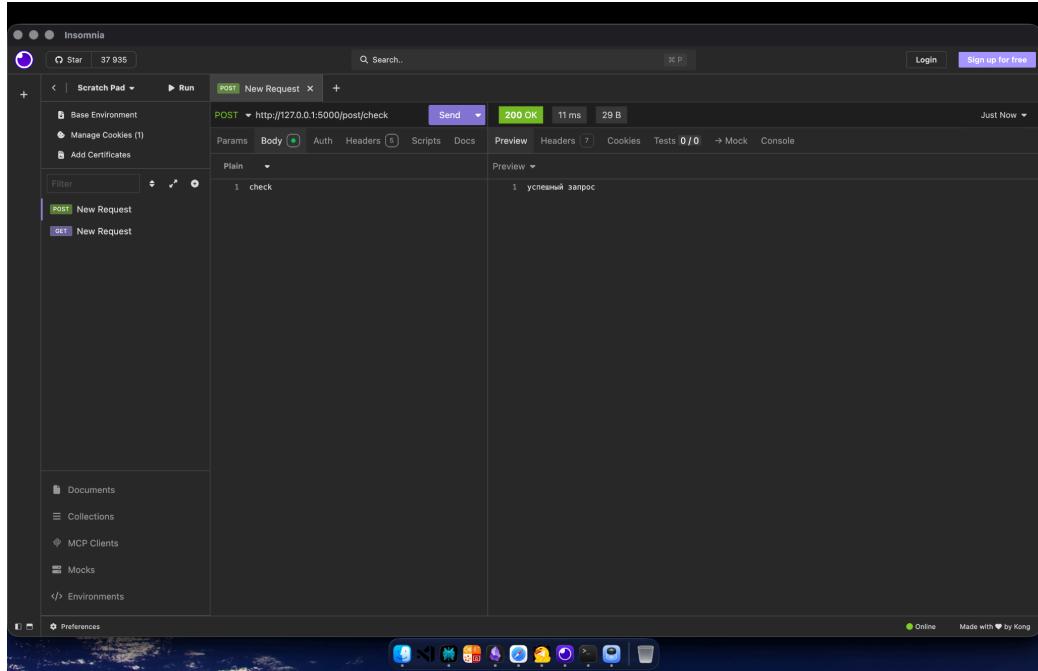


Рисунок 11 - Insomnia, POST /post/check с телом "check"

При передаче в теле запроса строки "check" сервер возвращает успешный ответ, подтверждающий корректность тела.

Вариант 2: тело = "hello"

Запрос:

Метод: POST

URL: http://127.0.0.1:5000/post/check

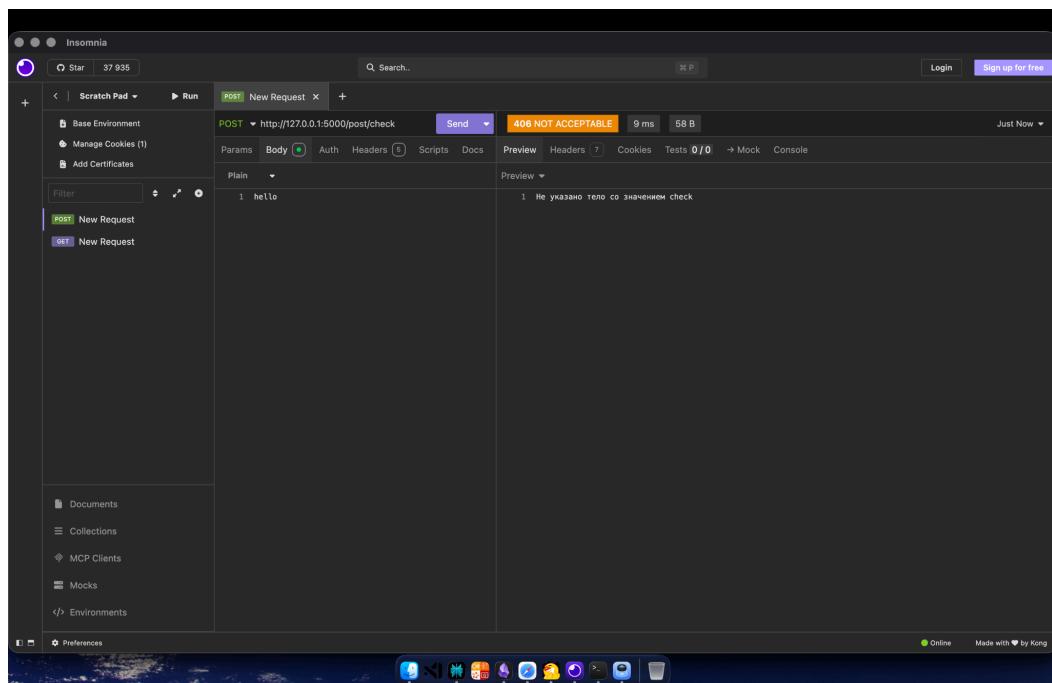


Рисунок 12 - Insomnia, POST /post/check с телом "hello"

При передаче любого другого значения в теле запроса сервер возвращает ошибку, демонстрируя валидацию тела запроса на стороне сервера.

4. ВЫВОДЫ

1. В ходе лабораторной работы изучена структура протокола HTTP и его основные компоненты: методы (GET, POST), заголовки запросов и ответов, коды состояния.
2. Освоены инструменты для анализа HTTP-трафика: встроенные средства разработчика браузера Safari (вкладка Network в Web Inspector) и специализированный HTTP-клиент Insomnia.
3. На практике продемонстрирована работа различных методов HTTP: метод GET используется для запроса ресурсов и передачи параметров через URL (query-параметры), метод POST – для передачи данных в теле запроса.
4. Подтверждено, что сервер может валидировать как заголовки (MyHeader), так и тело запроса, возвращая соответствующие коды состояния (200 OK при успехе, 4xx при ошибках клиента).
5. Получены навыки работы с локальным веб-сервером на базе Flask (Python), включая его запуск, отправку запросов и анализ ответов.

5. ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего предназначен HTTP протокол?

HTTP (Hypertext Transfer Protocol) предназначен для передачи гипертекстовых документов и других ресурсов в сети Интернет. Протокол работает в режиме «запрос-ответ»: клиент (браузер, мобильное приложение, API-клиент) отправляет запрос на сервер, сервер обрабатывает запрос и возвращает ответ (HTML-страницу, JSON-данные, изображение и т.п.). HTTP – основа функционирования World Wide Web и используется также в RESTful API, веб-сервисах и приложениях.

2. Состав URL

URL (Uniform Resource Locator) состоит из следующих компонентов:

1. Схема (протокол): http, https, ftp и др.
2. Доменное имя или IP-адрес сервера: например, www.example.com или 192.168.1.1.
3. Порт (необязательно): например, :8080. Если не указан, используется порт по умолчанию (80 для HTTP, 443 для HTTPS).
4. Путь к ресурсу на сервере: например, /path/to/resource.
5. Параметры запроса (query string, необязательно): строка после символа ?, например ?id=123&lang=ru.
6. Якорь или фрагмент (необязательно): строка после символа #, например #section1, используется для навигации внутри страницы.

Пример полного URL:

<https://www.example.com:8080/path/to/resource?id=123&lang=ru#section1>

3. Состав HTTP-запроса

HTTP-запрос состоит из трех основных частей:

1. Стартовая строка (request line):

Метод URL Версия_протокола

Пример: GET /index.html HTTP/1.1

2) Заголовки запроса (request headers):

Набор пар «название: значение», передающих метаданные запроса.

Пример:

Host: www.example.com

User-Agent: Mozilla/5.0

Accept: text/html

Заголовок Host обязателен в HTTP/1.1.

3) Тело запроса (request body, необязательная часть):

Используется в методах POST, PUT, PATCH для передачи данных на сервер (например, содержимое формы, JSON-объект, файл).

4. Методы HTTP

Основные методы HTTP:

- GET: запрос на получение ресурса. Параметры передаются через URL (query-параметры). Тело запроса отсутствует;
- POST: передача данных на сервер для обработки (например, отправка формы, загрузка файла). Данные передаются в теле запроса;
- PUT: размещение или полная замена ресурса на сервере;
- DELETE: удаление ресурса на сервере;
- HEAD: запрос заголовков ресурса без передачи тела (полезно для проверки существования ресурса и получения метаданных);
- OPTIONS: запрос поддерживаемых методов и параметров для данного ресурса;
- PATCH: частичное изменение ресурса.

Наиболее часто используемые методы – GET (для чтения данных) и POST (для отправки данных).

5. Классы кодов ответа HTTP

Коды состояния HTTP делятся на пять классов по первой цифре:

1xx (информационные): запрос получен, обработка продолжается.

Пример: 101 Switching Protocols – сервер предлагает переключиться на другой протокол (например, WebSocket).

2xx (успешное выполнение): запрос успешно обработан. Пример: 200 OK – запрос выполнен успешно, ресурс передан. 201 Created – ресурс успешно создан на сервере (например, после POST-запроса).

3xx (перенаправление): необходимо предпринять дополнительные действия для завершения запроса. Пример: 301 Moved Permanently – ресурс перемещен на новый URL (указан в заголовке Location). 302 Found – временное перемещение. 304 Not Modified – ресурс не изменился, можно использовать кэш.

4xx (ошибка клиента): запрос содержит синтаксическую ошибку или не может быть выполнен. Пример: 400 Bad Request – неверный синтаксис запроса. 403 Forbidden – доступ к ресурсу запрещен. 404 Not Found – ресурс не найден на сервере.

5xx (ошибка сервера): сервер не смог выполнить корректный запрос из-за внутренней ошибки. Пример: 500 Internal Server Error – внутренняя ошибка сервера. 501 Not Implemented – сервер не поддерживает запрошенную функциональность. 503 Service Unavailable – сервер временно недоступен (например, из-за перегрузки или обслуживания).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. RFC 9110 HTTP Semantics. IETF, 2022. URL:
<https://www.rfc-editor.org/rfc/rfc9110.html>
2. RFC 9112 HTTP/1.1. IETF, 2022. URL:
<https://www.rfc-editor.org/rfc/rfc9112.html>
3. RFC 1738 Uniform Resource Locators (URL). IETF, 1994. URL:
<https://www.rfc-editor.org/rfc/rfc1738.html>
4. Кузнецов В.И. Взаимодействие с web сервисом через протокол HTTP: методическое пособие к лабораторной работе 1 по дисциплине «Языки интернет-программирования». М.: МГТУ им. Н.Э. Баумана, 2025.
5. Mozilla Developer Network (MDN). HTTP. URL:
<https://developer.mozilla.org/en-US/docs/Web/HTTP>
6. Insomnia Documentation. URL: <https://docs.insomnia.rest/>