Сучасні технології Інтернет-програмування

Пашкова Ганна Олегівна
група ІПЗ-22-2

**Лабораторна робота № 2**

**з теми: Дослідження роботи протоколу HTTP**

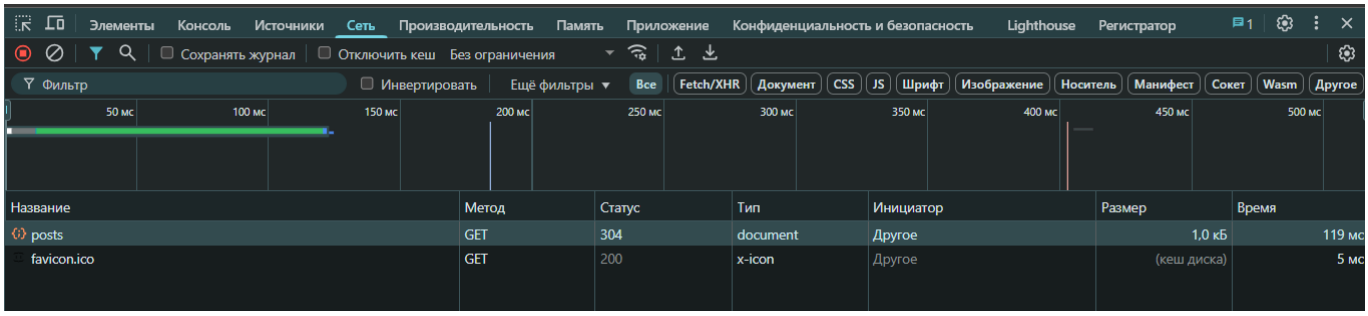
**Варіант 10**

**Завдання 1: Аналіз HTTP-запиту через браузер (варіанти 1-10)**

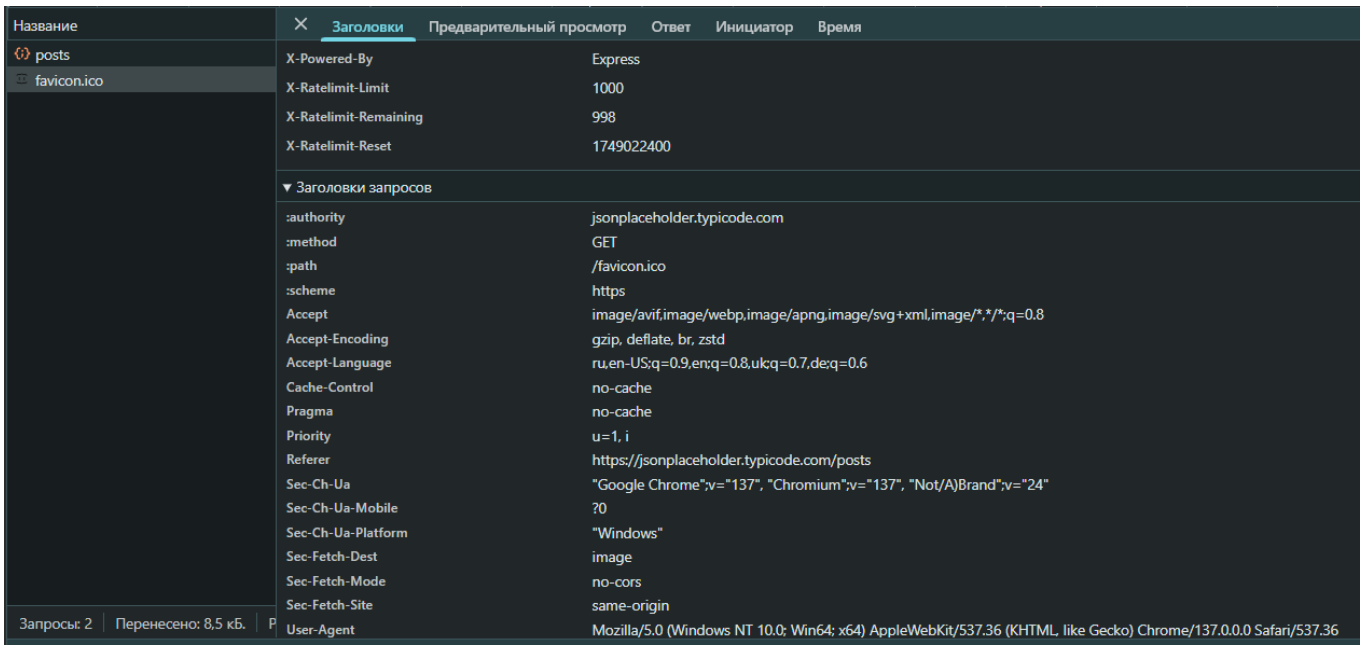
1. Відкрийте одну з веб-сторінок, згідно з вашим варіантом:
    - Варіант 1-9: https://example.com
    - Варіант 10-19: https://jsonplaceholder.typicode.com/posts
    - Варіант 20-29: https://openweathermap.org
2. Відкрийте інструменти розробника браузера (F12 або Ctrl + Shift + I).
3. Перейдіть на вкладку Network.
4. Оновіть сторінку.

Запишіть:

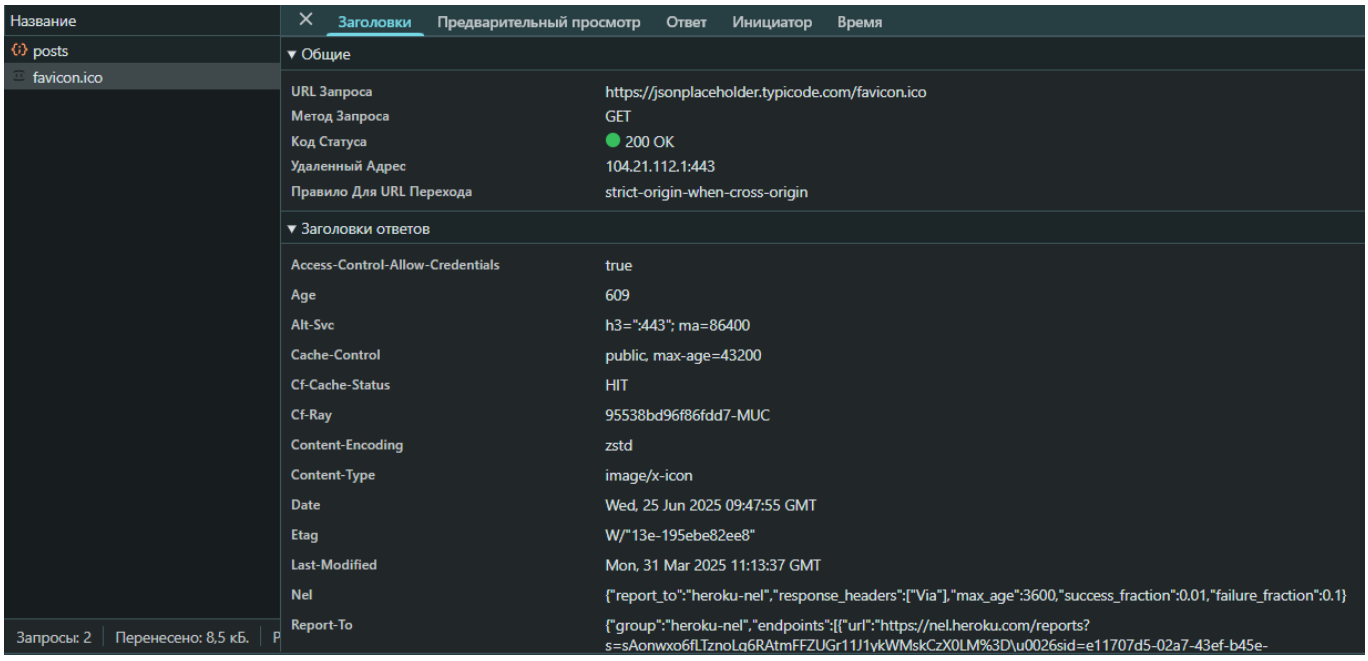
- Типи HTTP-запитів, які виконуються (GET, POST тощо).
- Заголовки запитів (User-Agent, Host, Content-Type).
- Відповідь сервера (статусний код, заголовки).




*Виконуються 2 GET запити. Заголовки запитів:*




*Відповідь сервера:*



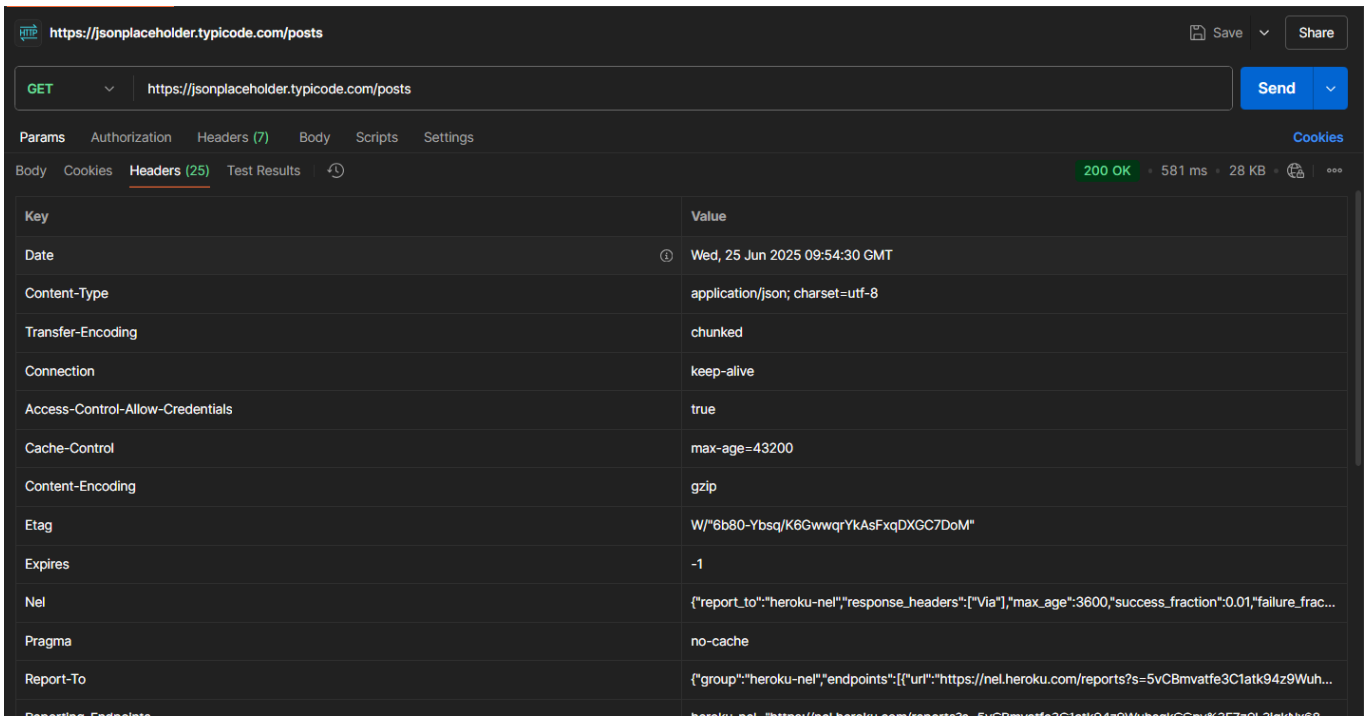

**Завдання 2: Надсилання запиту через Postman**

Відкрийте Postman (https://www.postman.com/explore) та створіть новий запит типу GET до наступного API відповідно до варіанту:

- Варіант 1-15: https://jsonplaceholder.typicode.com/posts
- Варіант 16-30: https://api.spacexdata.com/v4/launches

2. Проаналізуйте отриману відповідь (статусний код, формат даних).

3. Додайте заголовок Accept: application/json і знову надішліть запит.




Статусний код – 200 OK.

Формат даних – json.

*Додаємо Accept: application/json:*

| | |
|---|---|
| Reporting-Endpoints | heroku-nel="https://nel.heroku.com/reports?s=5vCBmvatfe3C1atk94z9WuhsgkCGpv%2F7z0L2jgkNx68... |
| Server | cloudflare |
| Vary | Origin, Accept-Encoding |
| Via | 2.0 heroku-router |
| X-Content-Type-Options | nosniff |
| X-Powered-By | Express |
| X-Ratelimit-Limit | 1000 |
| X-Ratelimit-Remaining | 999 |
| X-Ratelimit-Reset | 1749840030 |
| Age | 26030 |
| Cf-Cache-Status | HIT |
| CF-RAY | 9553957dffb6d65d-IAD |
| alt-svc | h3=":443"; ma=86400 |

Після додавання до запиту заголовка Accept: application/json відповідь сервера не зазнала суттєвих змін, оскільки сервер jsonplaceholder.typicode.com і без того повертає дані у форматі JSON за замовчуванням.

Однак тепер у заголовках запиту чітко вказано, що клієнт очікує отримати відповідь саме у форматі JSON.

**Завдання 3: Відправка POST-запиту**

1. Змініть тип запиту на POST.

2. Введіть URL відповідно до вашого варіанту:

    o Варіант 1-15: https://jsonplaceholder.typicode.com/posts

    o Варіант 16-30: https://reqres.in/api/users

3. Перейдіть на вкладку Body, оберіть raw і встановіть формат JSON.

4. Введіть наступні дані для вашого варіанту:

{

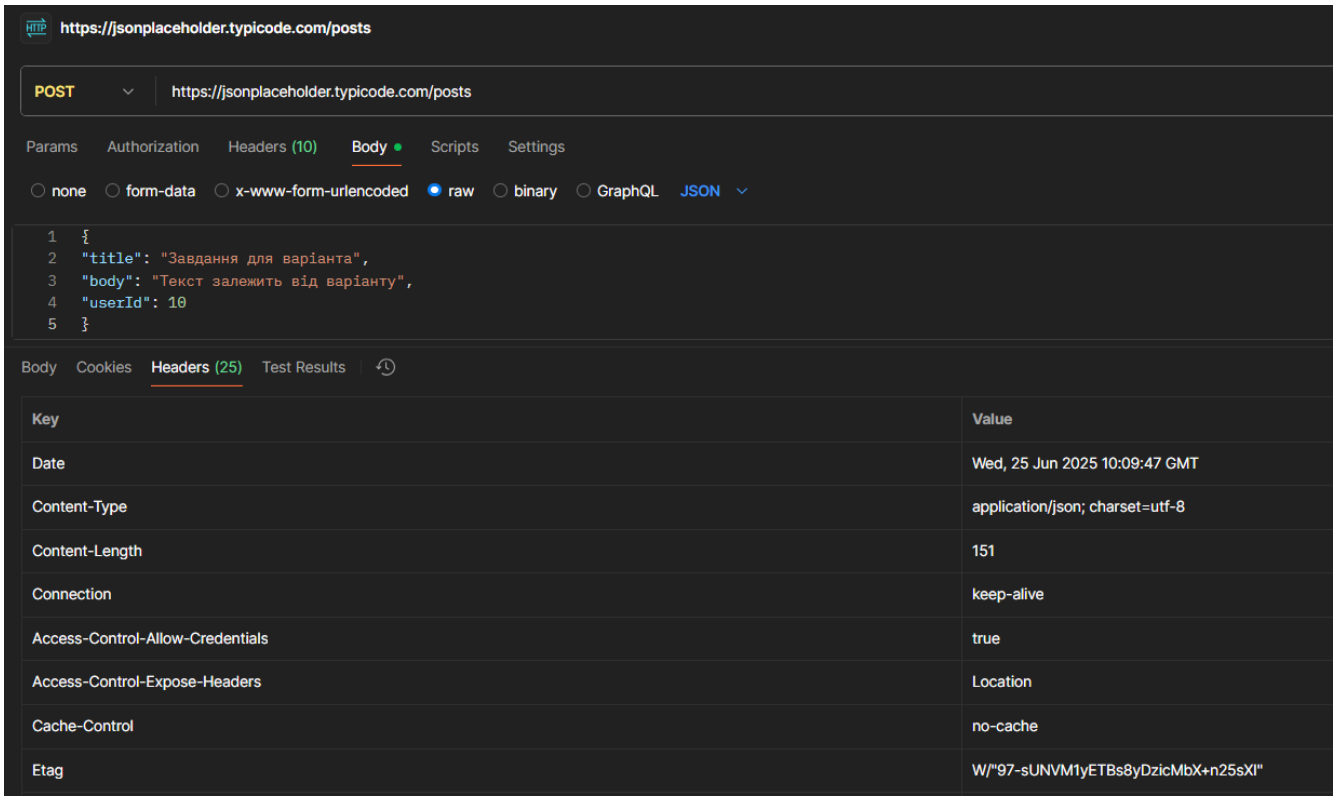
“title”: “Завдання для варіанта”,

“body”: “Текст залежить від варіанту”,

“userId”: <номер варіанту>

}

5. Надішліть запит і перегляньте відповідь сервера.



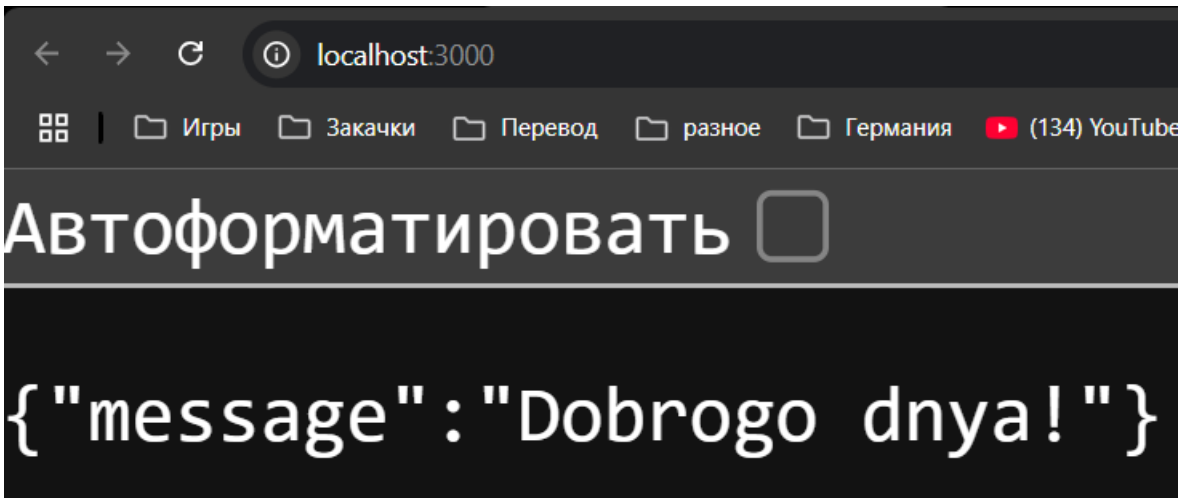

**Завдання 4: Створення простого HTTP-сервера**

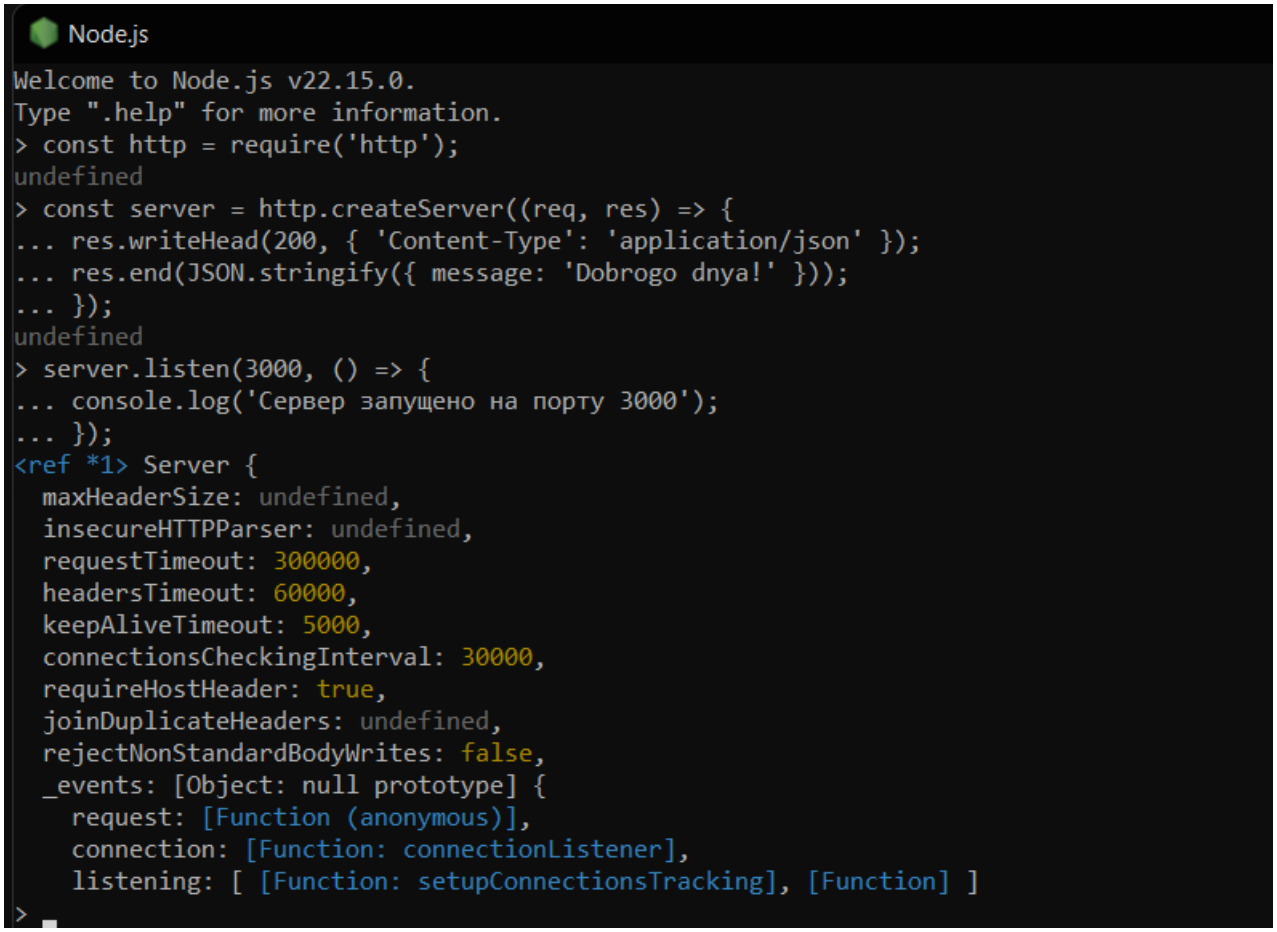
1. Створіть файл server.js із наступними варіантами реалізації для вашого завдання:

    o Варіант 1-9: Повертає текст “Привіт, світ!” на запит GET.

    o Варіант 10-19: Повертає JSON із повідомленням “Доброго дня!”.

    o Варіант >20: Повертає HTML із вбудованим стилем.




**Контрольні запитання**

1. Що таке HTTP і які його основні методи?

HTTP — це протокол обміну даними між клієнтом і сервером. Основні методи: GET, POST, PUT, DELETE, PATCH.

2. Для чого використовуються заголовки HTTP?

Заголовки передають додаткову інформацію про запит або відповідь, наприклад тип вмісту, авторизацію, кешування тощо.

3. Які групи статусних кодів ви знаєте?

Існують такі групи: 1xx — інформаційні, 2xx — успішні, 3xx — перенаправлення, 4xx — помилки клієнта, 5xx — помилки сервера.

4. Що таке REST API та його основні принципи?

REST API — це архітектурний стиль для створення вебсервісів. Принципи: клієнт-сервер, безстанність, кешування, єдиний інтерфейс, використання ресурсів.

5. Як HTTPS захищає передачу даних у мережі?

HTTPS шифрує дані за допомогою SSL/TLS, завдяки чому інформація передається захищено й не може бути перехоплена сторонніми.

**Короткі висновки**

У ході роботи було ознайомлено з основами HTTP-запитів, їхніми типами, заголовками та статусними кодами. За допомогою Postman і браузера вдалося на практиці дослідити, як саме відбувається взаємодія клієнта з сервером. Також було створено простий сервер на Node.js, який повертає JSON-відповідь.

Отримані знання є базовими для розуміння роботи вебсервісів та побудови REST API. Практична частина допомогла краще зрозуміти, як відбувається обробка HTTP-запитів та як важливо правильно формувати запити й обробляти відповіді.