

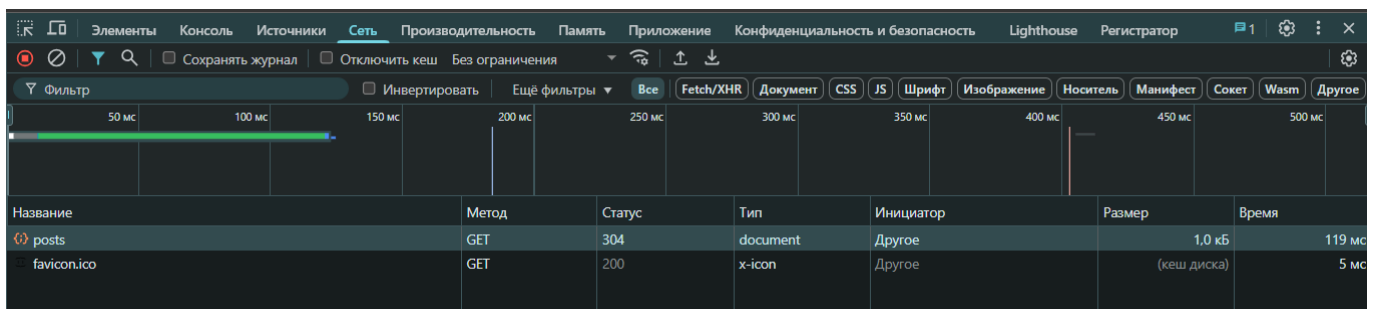
**Лабораторна робота № 2**  
**з теми: Дослідження роботи протоколу HTTP**  
**Варіант 10**

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

1. Відкрийте одну з веб-сторінок, згідно з вашим варіантом:
  - o Варіант 1-9: <https://example.com>
  - o Варіант 10-19: <https://jsonplaceholder.typicode.com/posts>
  - o 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).



The screenshot shows the Chrome DevTools Network tab. The 'posts' request is selected, showing a GET method, status 304, type document, and size 1.0 kB. The 'favicon.ico' request is also shown, with a GET method, status 200, type x-icon, and size (кеш диска). The table below summarizes the data from the screenshot.

Название	Метод	Статус	Тип	Инициатор	Размер	Время
posts	GET	304	document	Другое	1.0 kB	119 мс
favicon.ico	GET	200	x-icon	Другое	(кеш диска)	5 мс

*Виконуються 2 GET запити. Заголовки запитів:*

Название	Заголовки	Предварительный просмотр	Ответ	Инициатор	Время
posts	X-Powered-By	Express			
favicon.ico	X-Ratelimit-Limit	1000			
	X-Ratelimit-Remaining	998			
	X-Ratelimit-Reset	1749022400			
	▼ Заголовки запросов				
	:authority	jsonplaceholder.typicode.com			
	:method	GET			
	:path	/favicon.ico			
	:scheme	https			
	Accept	image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8			
	Accept-Encoding	gzip, deflate, br, zstd			
	Accept-Language	ru,en-US;q=0.9,en;q=0.8,uk;q=0.7,de;q=0.6			
	Cache-Control	no-cache			
	Pragma	no-cache			
	Priority	u=1, i			
	Referer	https://jsonplaceholder.typicode.com/posts			
	Sec-Ch-Ua	"Google Chrome";v="137", "Chromium";v="137", "Not(A)Brand";v="24"			
	Sec-Ch-Ua-Mobile	?0			
	Sec-Ch-Ua-Platform	"Windows"			
	Sec-Fetch-Dest	image			
	Sec-Fetch-Mode	no-cors			
	Sec-Fetch-Site	same-origin			
Запросы: 2   Перенесено: 8,5 кБ.   P	User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36			

*Відповідь сервера:*

Название	Заголовки	Предварительный просмотр	Ответ	Инициатор	Время
posts	▼ Общие				
favicon.ico	URL Запроса	https://jsonplaceholder.typicode.com/favicon.ico			
	Метод Запроса	GET			
	Код Статуса	200 OK			
	Удаленный Адрес	104.21.112.1:443			
	Правило Для URL Перехода	strict-origin-when-cross-origin			
	▼ Заголовки ответов				
	Access-Control-Allow-Credentials	true			
	Age	609			
	Alt-Svc	h3=":443"; ma=86400			
	Cache-Control	public, max-age=43200			
	Cf-Cache-Status	HIT			
	Cf-Ray	95538bd96f86fdd7-MUC			
	Content-Encoding	zstd			
	Content-Type	image/x-icon			
	Date	Wed, 25 Jun 2025 09:47:55 GMT			
	Etag	W/"13e-195ebe82ee8"			
	Last-Modified	Mon, 31 Mar 2025 11:13:37 GMT			
	Nel	{"report_to":"heroku-nel","response_headers":["Via"],"max_age":3600,"success_fraction":0.01,"failure_fraction":0.1}			
Запросы: 2   Перенесено: 8,5 кБ.   P	Report-To	{"group":"heroku-nel","endpoints":[{"url":"https://nel.heroku.com/reports?s=sAonwxo6fLTznoLq6RAtmFFZUGr1U1vkWMskCzX0LM%3D\u0026sid=e11707d5-02a7-43ef-b45e-"}]}			

## 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:

о Варіант 1-15: <https://jsonplaceholder.typicode.com/posts>

о Варіант 16-30: <https://api.spacexdata.com/v4/launches>

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

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

https://jsonplaceholder.typicode.com/posts		Save	Share
GET	https://jsonplaceholder.typicode.com/posts		
Params	Authorization	Headers (7)	Body Scripts Settings
Body	Cookies	Headers (25)	Test Results
		200 OK • 581 ms • 28 KB • 🌐	
Key	Value		
Date	Wed, 25 Jun 2025 09:54:30 GMT		
Content-Type	application/json; charset=utf-8		
Transfer-Encoding	chunked		
Connection	keep-alive		
Access-Control-Allow-Credentials	true		
Cache-Control	max-age=43200		
Content-Encoding	gzip		
Etag	W/"6b80-Ybsq/K6GwwqrYkAsFxqDXGC7DoM"		
Expires	-1		
Nel	{ "report_to": "heroku-nel", "response_headers": [ "Via" ], "max_age": 3600, "success_fraction": 0.01, "failure_fract...		
Pragma	no-cache		
Report-To	{ "group": "heroku-nel", "endpoints": [ { "url": "https://nel.heroku.com/reports?s=5vCBmvatfe3C1atk94z9WuhsgkCGpv%2F7z0L2jgkNx68..." } ] }		

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

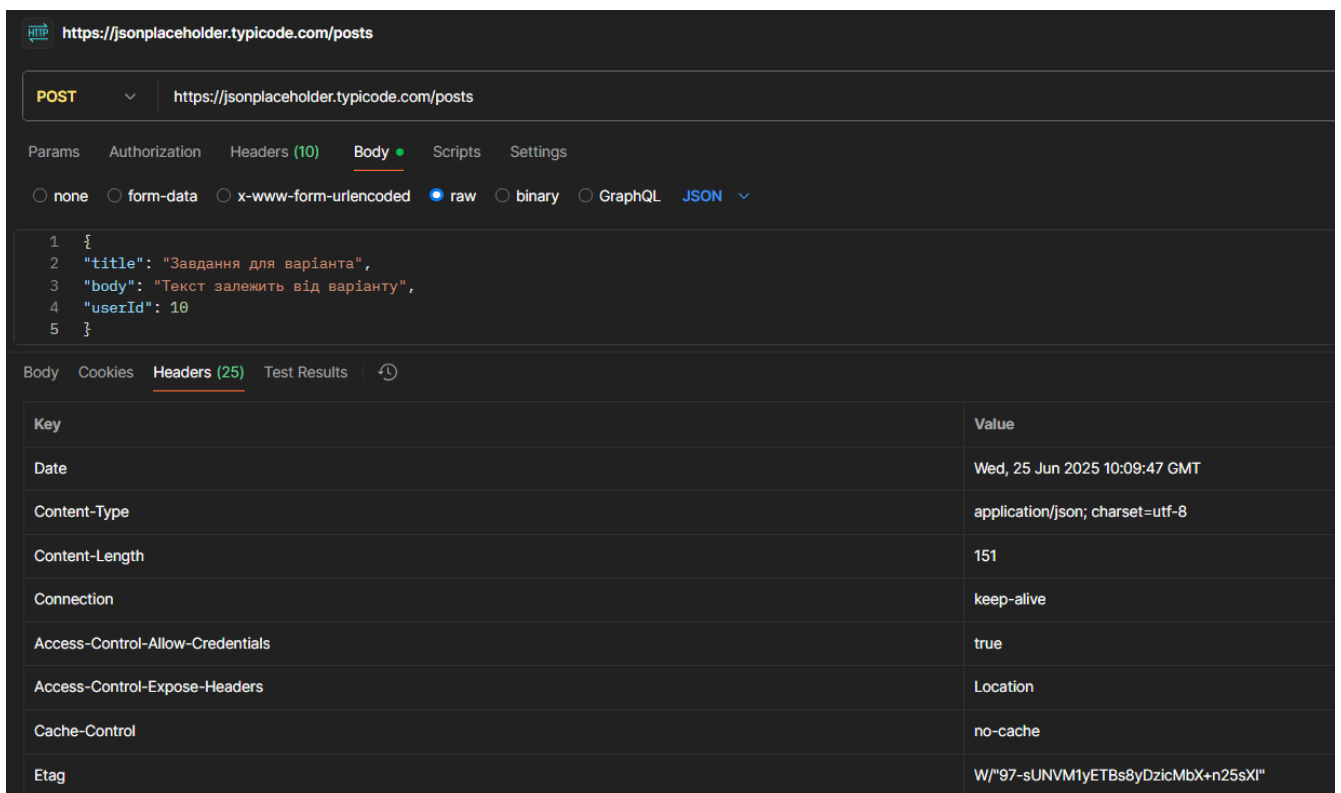
Після додавання до запиту заголовка Accept: application/json відповідь сервера не зазнала суттєвих змін, оскільки сервер jsonplaceholder.typicode.com і без того повертає дані у форматі JSON за замовчуванням.

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

### Завдання 3: Відправка POST-запиту

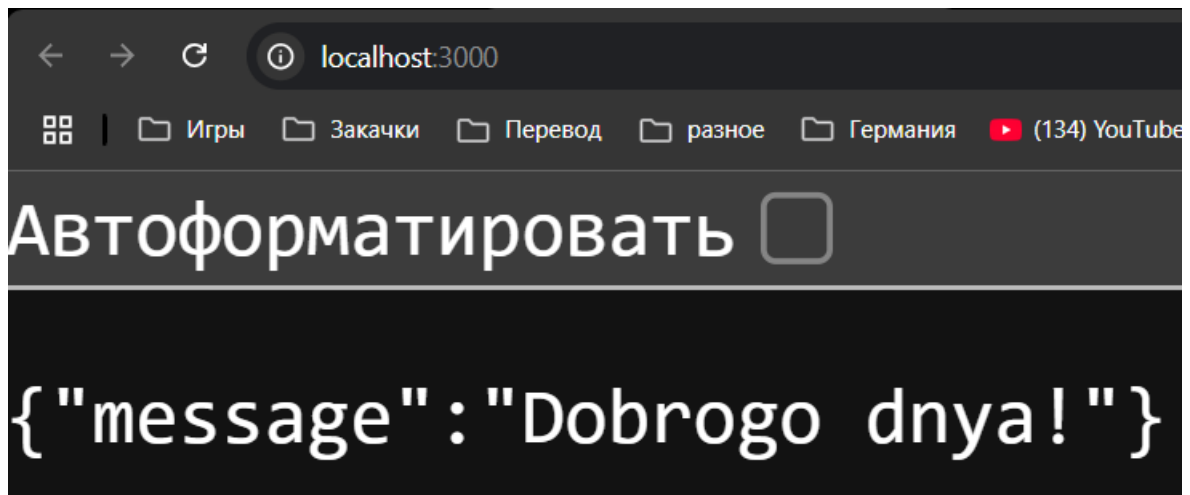
1. Змініть тип запиту на POST.
2. Введіть URL відповідно до вашого варіанту:
  - о Варіант 1-15: <https://jsonplaceholder.typicode.com/posts>
  - о Варіант 16-30: <https://reqres.in/api/users>
3. Перейдіть на вкладку Body, оберіть raw і встановіть формат JSON.
4. Введіть наступні дані для вашого варіанту:

```
{  
  "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:
  - о Варіант 1-9: Повертає текст “Привіт, світ!” на запит GET.
  - о Варіант 10-19: Повертає JSON із повідомленням “Доброго дня!”.
  - о Варіант >20: Повертає HTML із вбудованим стилем.



```
Node.js
Welcome to Node.js v22.15.0.
Type ".help" for more information.
> const http = require('http');
undefined
> const server = http.createServer((req, res) => {
... res.writeHead(200, { 'Content-Type': 'application/json' });
... res.end(JSON.stringify({ message: 'Dobrogo dnya!' }));
... });
undefined
> server.listen(3000, () => {
... console.log('Сервер запущено на порту 3000');
... });
<ref *1> Server {
  maxHeaderSize: undefined,
  insecureHTTPParser: undefined,
  requestTimeout: 300000,
  headersTimeout: 60000,
  keepAliveTimeout: 5000,
  connectionsCheckingInterval: 30000,
  requireHostHeader: true,
  joinDuplicateHeaders: undefined,
  rejectNonStandardBodyWrites: false,
  _events: [Object: null prototype] {
    request: [Function (anonymous)],
    connection: [Function: connectionListener],
    listening: [ [Function: setupConnectionsTracking], [Function] ]
  }
}
```

## 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.