

## Лабораторна робота 2

### Дослідження роботи протоколу HTTP

#### Мета:

Навчитися працювати з HTTP-запитами та відповідями, аналізувати їхні заголовки та коди стану, а також здійснювати роботу з REST API.

#### Необхідне програмне забезпечення:

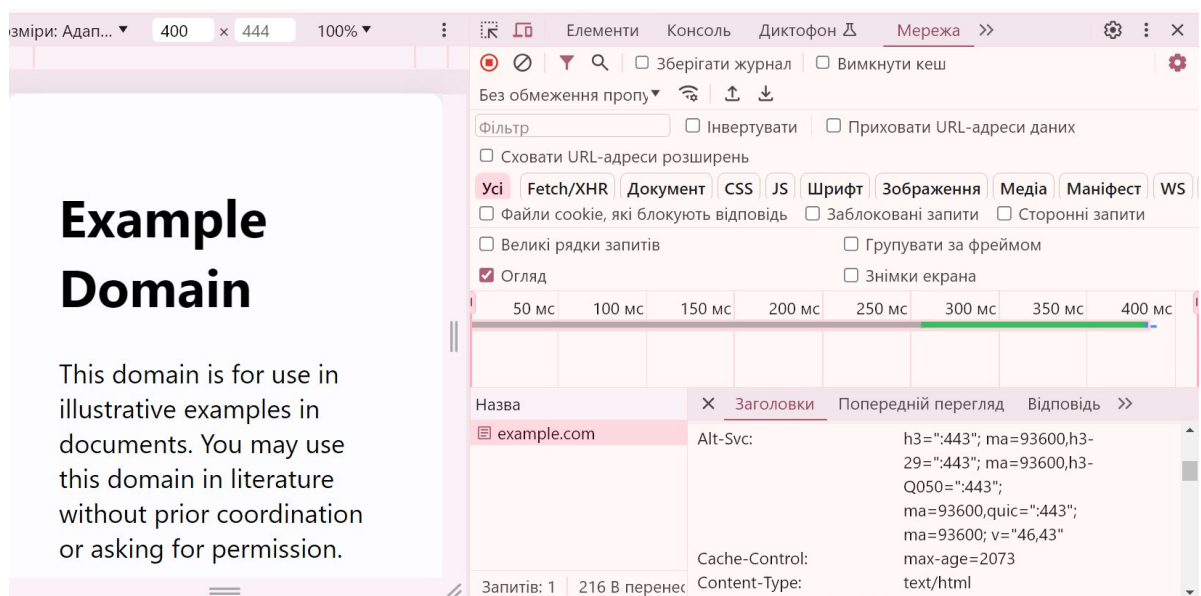
- Встановлений браузер (Google Chrome, Firefox або інший)
- Інструмент для тестування API: Postman
- Мова програмування для тестів (Node.js)
- Текстовий редактор (VSCode)

#### Хід роботи

Для кожного варіанту завдання додається окремий підпункт із конкретними параметрами для виконання.

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

1. Відкрийте одну з веб-сторінок, згідно з вашим варіантом:
  - Варіант 20-29: <https://openweathermap.org>
2. Відкрийте інструменти розробника браузера (F12 або Ctrl + Shift + I).
3. Перейдіть на вкладку **Network**.
4. Оновіть сторінку.



#### Зapiшіть:

- Типи HTTP-запитів, які виконуються (GET, POST тощо).
- Заголовки запитів (User-Agent, Host, Content-Type).

- Відповідь сервера (статусний код, заголовки).

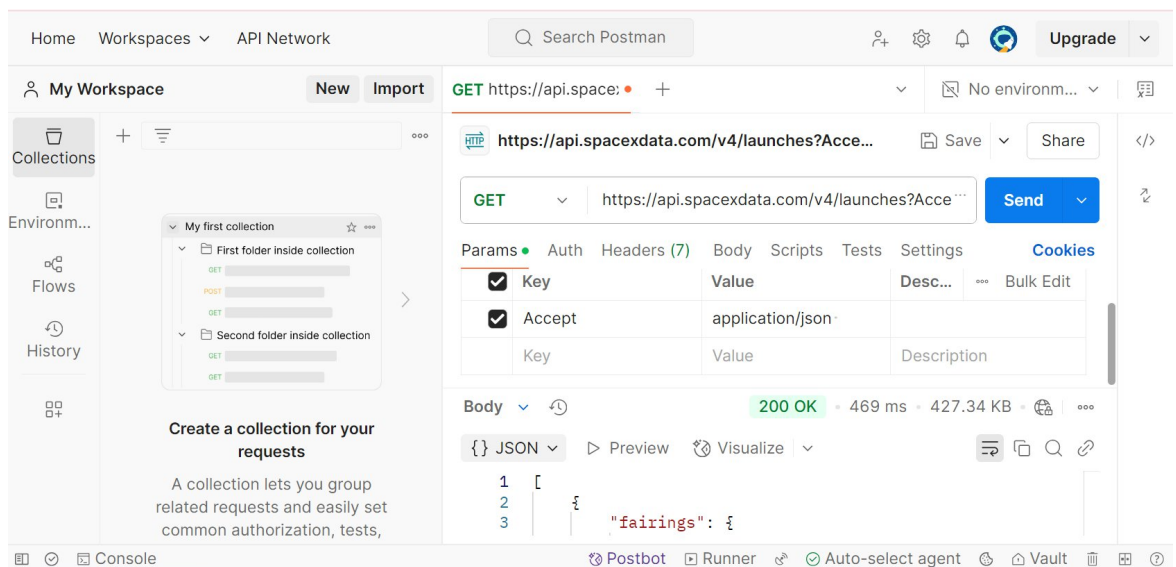
### Питання для обговорення:

- Який статусний код отримано для основного ресурсу сторінки?
- Які заголовки використовуються для кешування?

## 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>
1. Проаналізуйте отриману відповідь (статусний код, формат даних).
  3. Додайте заголовок `Accept: application/json` і знову надішліть запит.



### Запитання:

- Чи змінилася відповідь сервера після додавання заголовка?

## Завдання 3: Відправка POST-запиту


1. Змініть тип запиту на **POST**.
2. Введіть URL відповідно до вашого варіанту:
  - Варіант 1-15: <https://jsonplaceholder.typicode.com/posts>
  - 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:
5. {
6. "title": "Завдання для варіанта",
7. "body": "Текст залежить від варіанту",
8. "userId": <номер варіанту>
9. }
10. Надішліть запит і перегляньте відповідь сервера.

### Запитання:

- Який статусний код відповіді було отримано?
- Які заголовки були використані у відповіді сервера?

## Завдання 4: Створення простого HTTP-сервера

Для виконання завдання завантажуюмо Node.js (<https://nodejs.org/uk>).



```
Node.js
Welcome to Node.js v22.14.0.
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: 'Доброго дня!' }));
... });
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] ]
  },
  eventsCount: 3,
```

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

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

2. Код для варіантів 28-29 (JSON відповіді):

```
const http = require('http');

const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'application/json' });
  res.end(JSON.stringify({ message: 'Доброго дня!' }));
});

server.listen(3000, () => {
  console.log('Сервер запущено на порту 3000');
});
```

Запустіть сервер командою:

```
node server.js
```

Перейдіть у браузері за адресою <http://localhost:3000>.

Цей код створює простий HTTP-сервер на Node.js. Розберемо його по рядках:

### 1. Імпорт модуля HTTP

```
const http = require('http');
```

Модуль `http` вбудований у Node.js і дозволяє створювати веб-сервери.

```
require('http')
```

Підключає стандартний HTTP-модуль, який входить до Node.js і дозволяє створювати веб-сервери та відправляти HTTP-запити.

Створює об'єкт з методами для роботи з HTTP, наприклад:

- o `http.createServer()` — створення сервера.
- o `http.request()` — відповідь HTTP-запиту.
- o `http.get()` — отримання даних по HTTP.

## 2. Створення сервера

```
const server = http.createServer((req, res) => {
```

Функція `http.createServer()` створює сервер і приймає колбек-функцію `(req, res) => { ... }`, яка буде виконуватися під час кожного HTTP-запиту.

- `req (request)` — об'єкт запиту від клієнта.
- `res (response)` — об'єкт відповіді сервера.

## 3. Формування HTTP-відповіді

```
res.writeHead(200, { 'Content-Type': 'application/json' });
```

- `res.writeHead(200, {...})` — надсилає заголовки відповіді.
- `200` — код статусу HTTP (успішний запит).
- `{ 'Content-Type': 'application/json' }` — заголовок, що вказує, що сервер відправляє JSON-дані.

```
res.end(JSON.stringify({ message: 'Доброго дня!' }));
```

- `JSON.stringify({ message: 'Доброго дня!' })` — перетворює об'єкт `{ message: 'Доброго дня!' }` у JSON-рядок.
- `res.end(...)` — надсилає відповідь і завершує з'єднання.

## 4. Запуск сервера

```
server.listen(3000, () => {  
  console.log('Сервер запущено на порту 3000');  
});
```

- `server.listen(3000, ...)` — сервер починає прослуховувати порт 3000.
- Колбек-функція `() => { console.log(...) }` виводить повідомлення в консоль, коли сервер запущений.

## ◆ Як це працює?

1. Запускаємо сервер (`node ім'я_файлу.js`).
2. Відкриваємо в браузері `http://localhost:3000`.
3. Отримуємо відповідь:

```
json  
КопироватьРедактировать  
{ "message": "Доброго дня!" }
```

Цей код — мінімальний сервер на Node.js без фреймворків.

**Запитання:**

- Який статусний код відповіді повертає сервер?
- Який заголовок встановлено у відповіді сервера?

**Контрольні запитання:**

1. Що таке HTTP і які його основні методи?
2. Для чого використовуються заголовки HTTP?
3. Які групи статусних кодів ви знаєте?
4. Що таке REST API та його основні принципи?
5. Як HTTPS захищає передачу даних у мережі?

**Вимоги до звіту:**

1. Відповіді на всі контрольні запитання.
2. Скриншоти запитів у Postman та відповіді сервера.
3. Код HTTP-сервера.
4. Висновки про виконану роботу.