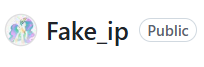
**Лабораторна робота №3**

**Мета:** Ознайомитися з принципами клієнт-серверної архітектури та зрозуміти її роль у сучасних програмних системах. Навчитися створювати RESTful API для взаємодії між клієнтом і сервером. Закріпити практичні навички роботи з HTTP-запитами та відповідями. Розвинути вміння проєктувати та реалізовувати ендпойнти для типових CRUD-операцій.

**Хід роботи**

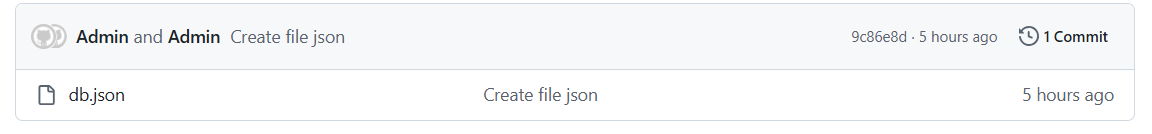
**Завдання №1**

1. Створимо репозиторій та назвемо Fake\_ip:



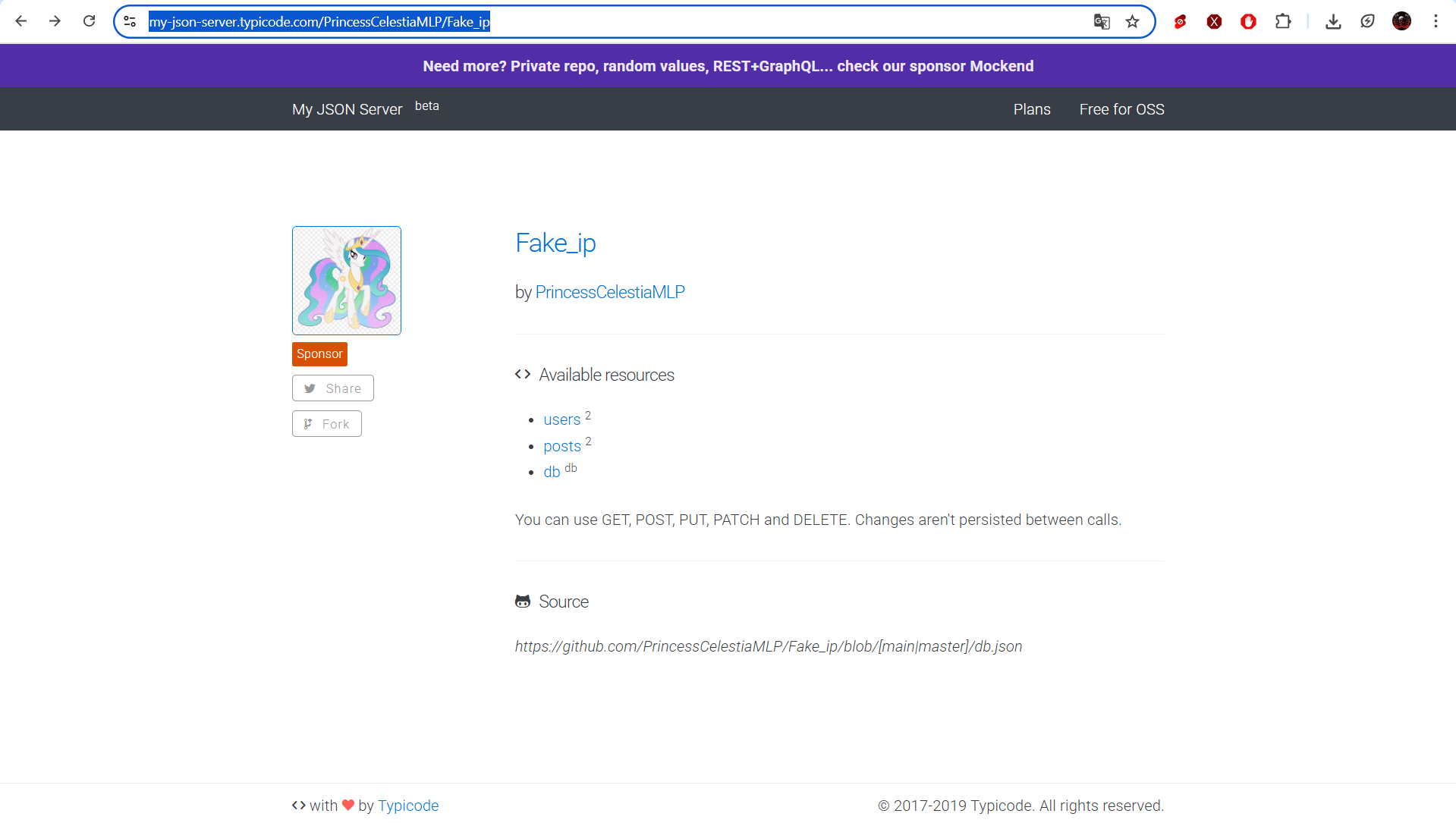
Силка: https://github.com/PrincessCelestiaMLP/Fake\_ip

1. Додамо туди файл **db.json** у корінь репозиторію. Додаємо через **Git Bash**



1. Запускаємо **Fake Online REST**

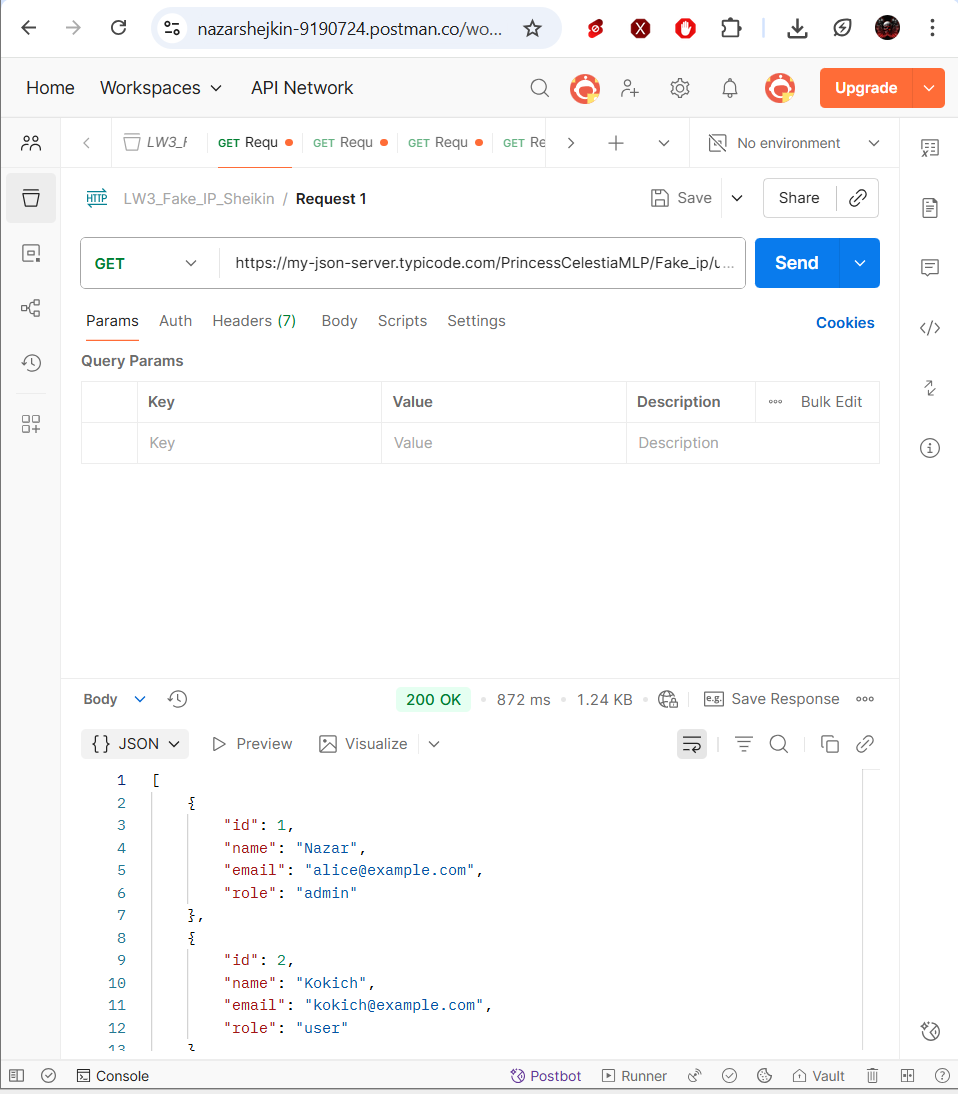
Базова URL-адреса після публікації: <https://my-json-server.typicode.com/PrincessCelestiaMLP/Fake_ip>

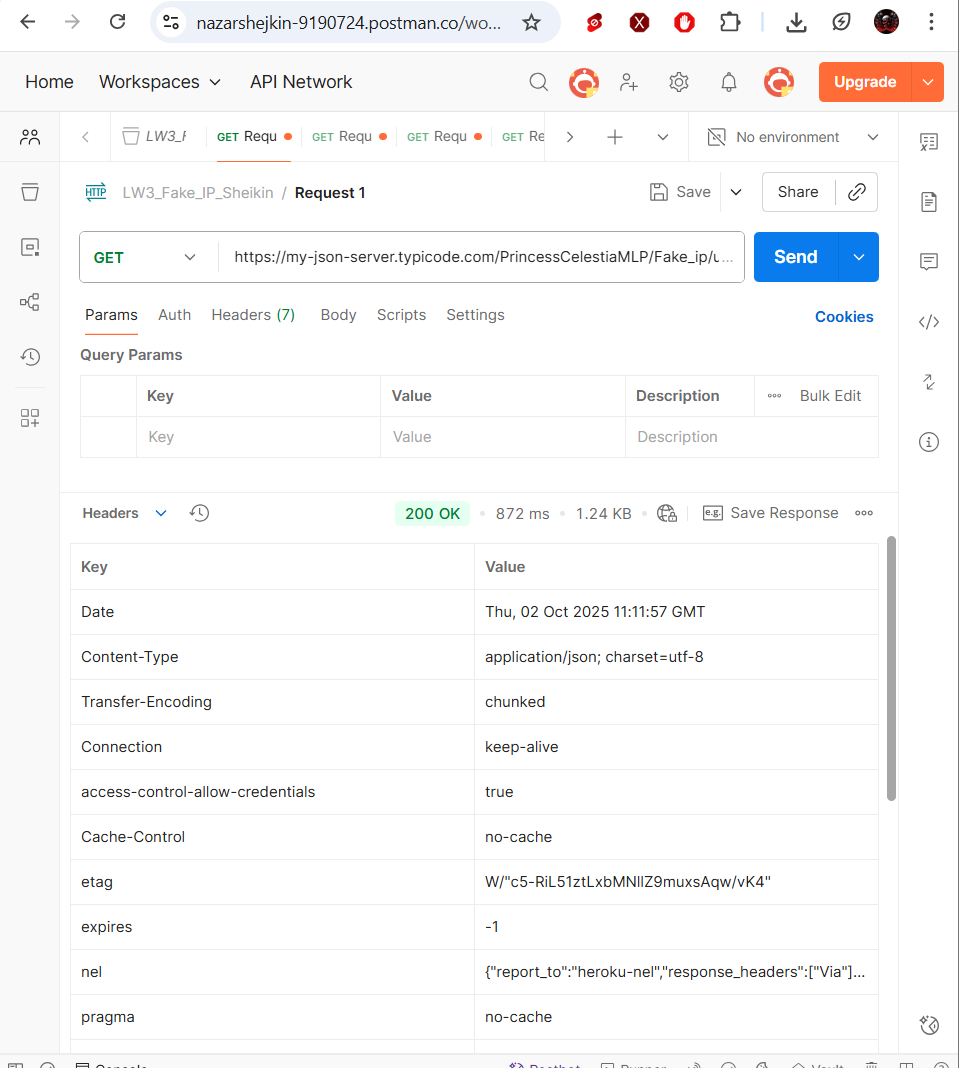


1. Перевірка запитів через PostMan

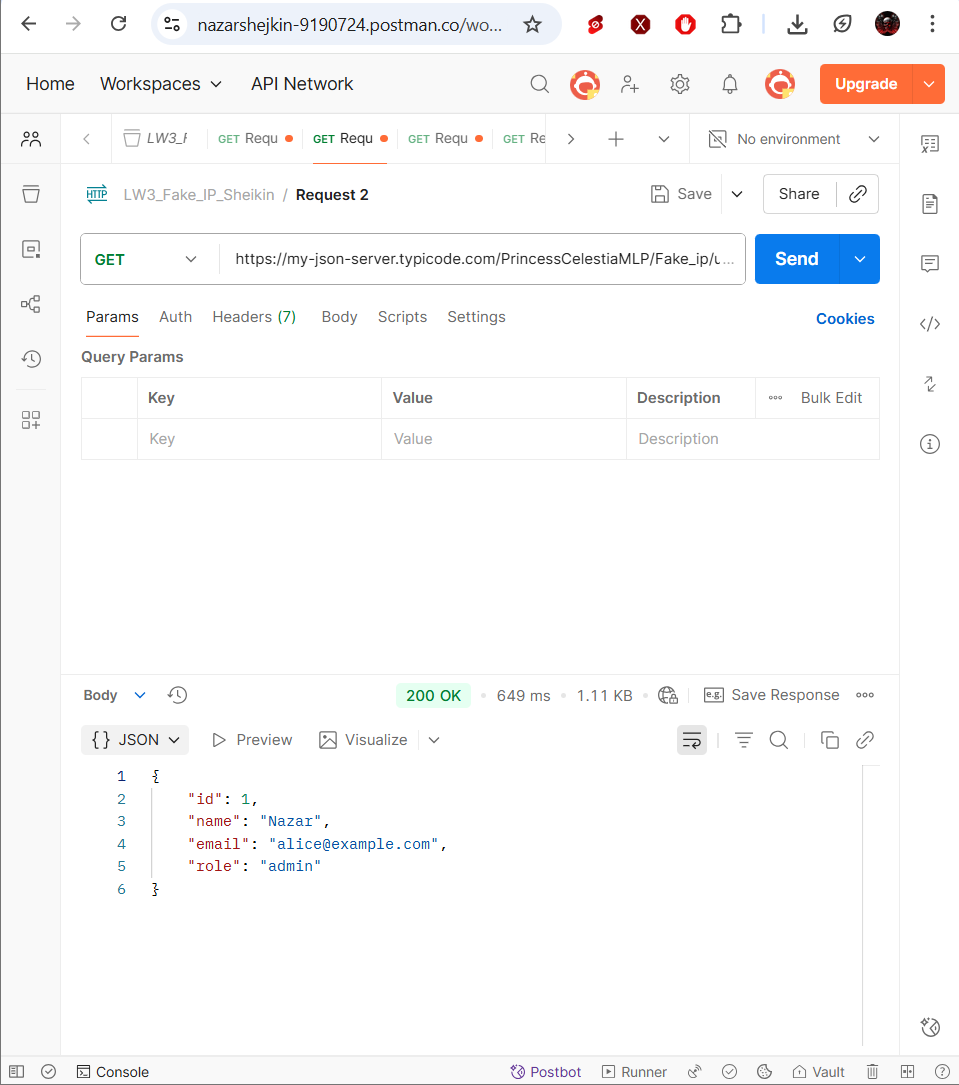
Створюємо колекцію і називаємо її **“LW3\_Fake\_IP\_Sheikin”**

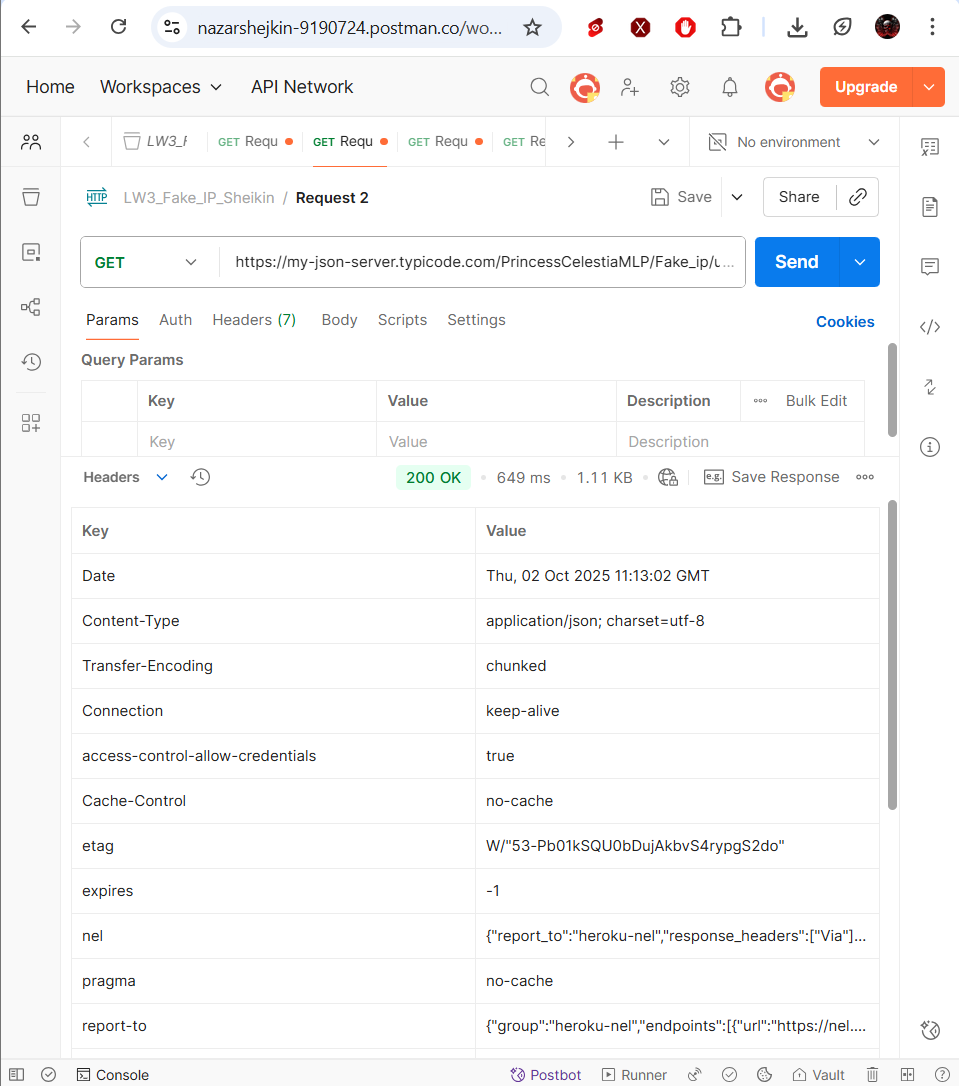
* 1. Виконаємо запит на усіх користувачів GET /users



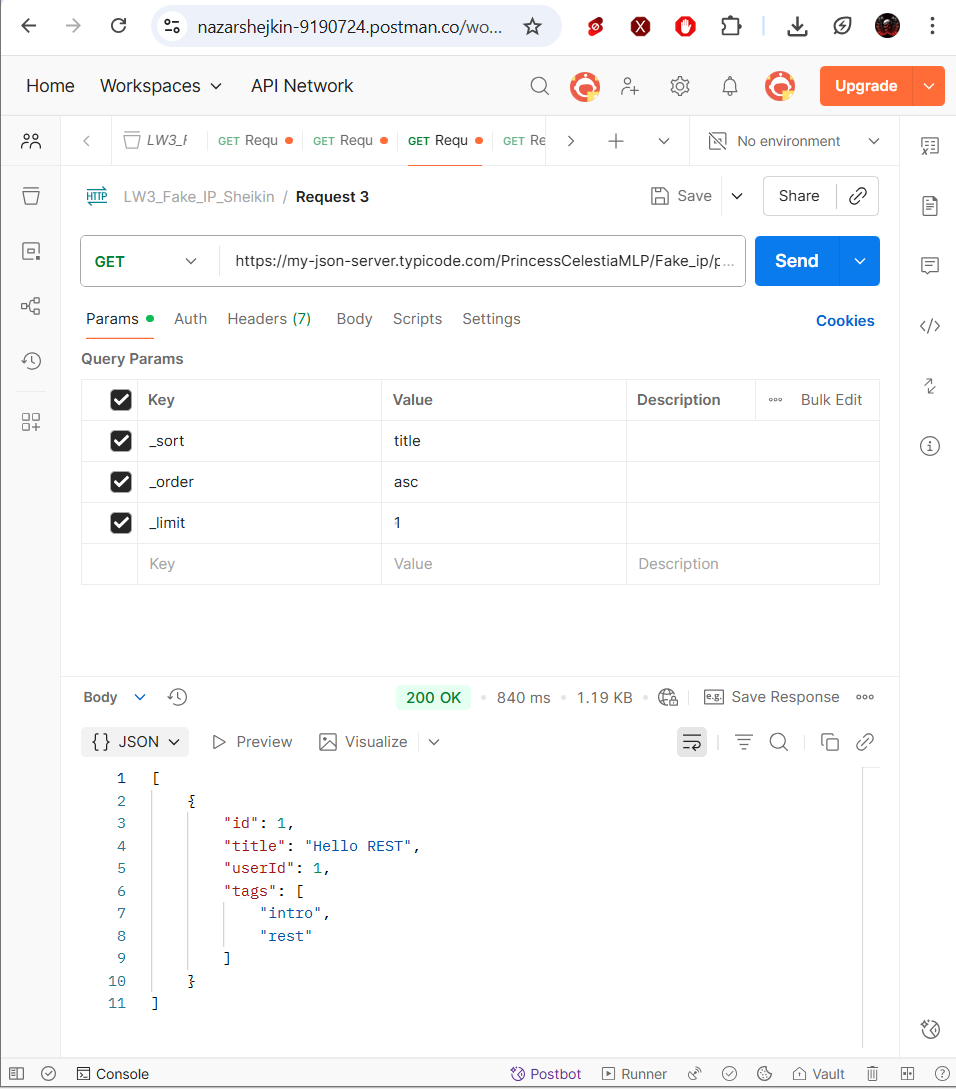


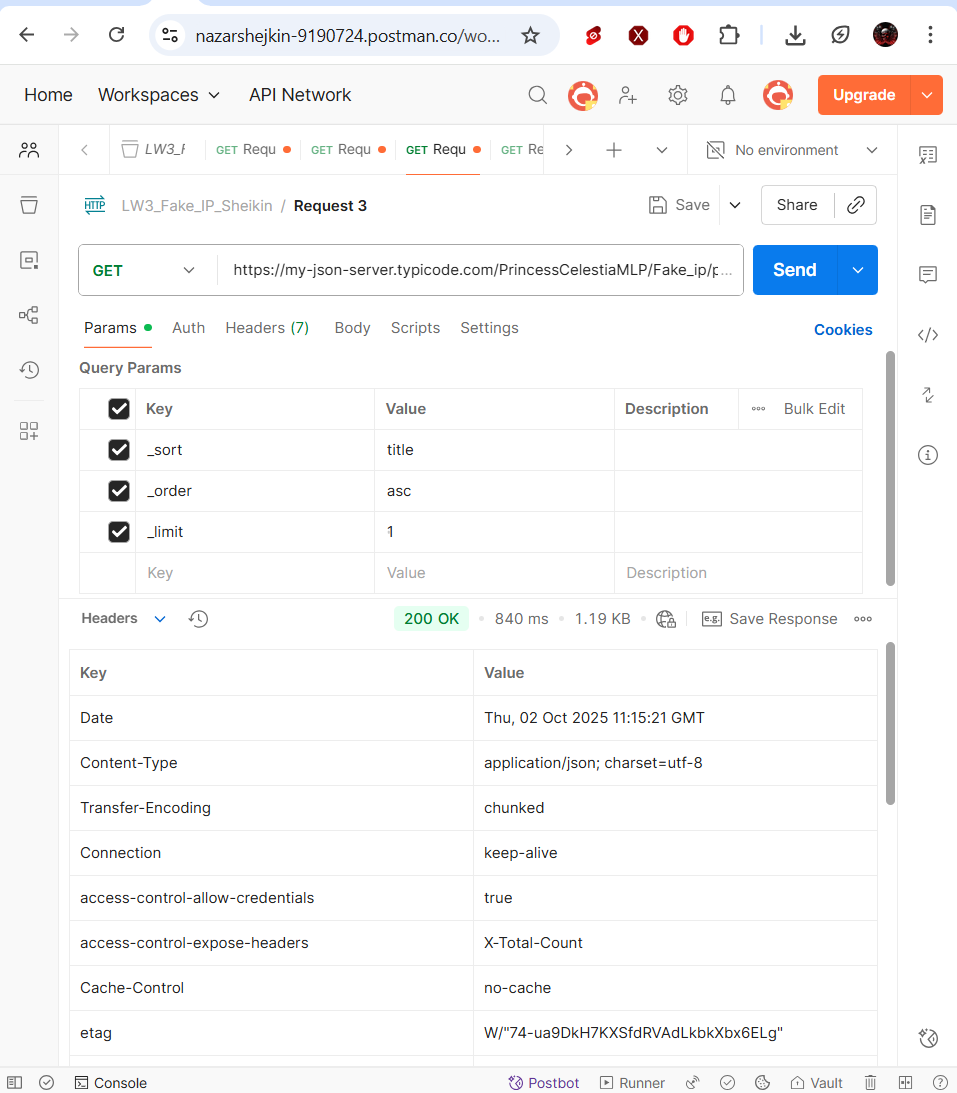
* 1. Виконаємо запит на одного користувача GET /users/1



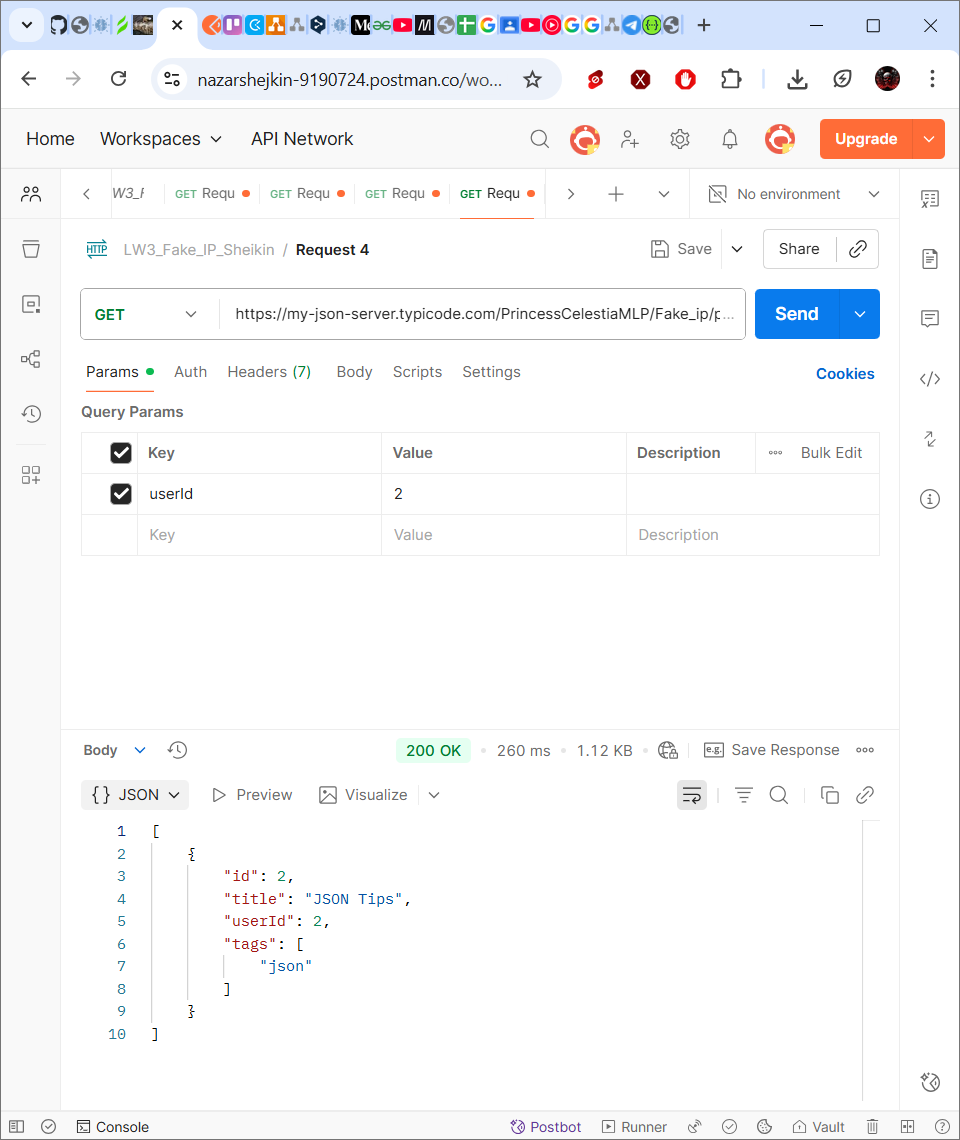


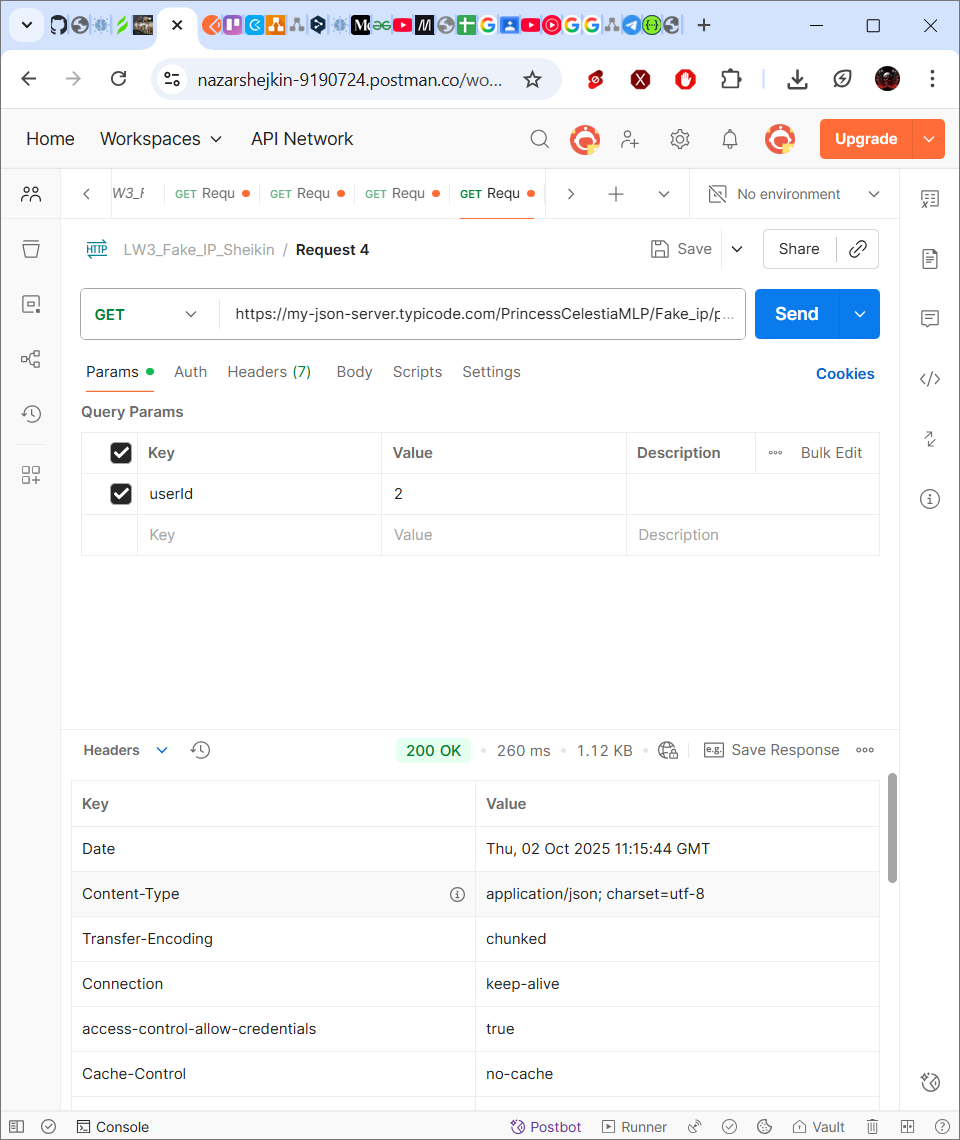
* 1. Виконаємо запит на сортування/ліміт GET /posts?\_sort=title&\_order=asc&\_limit=1



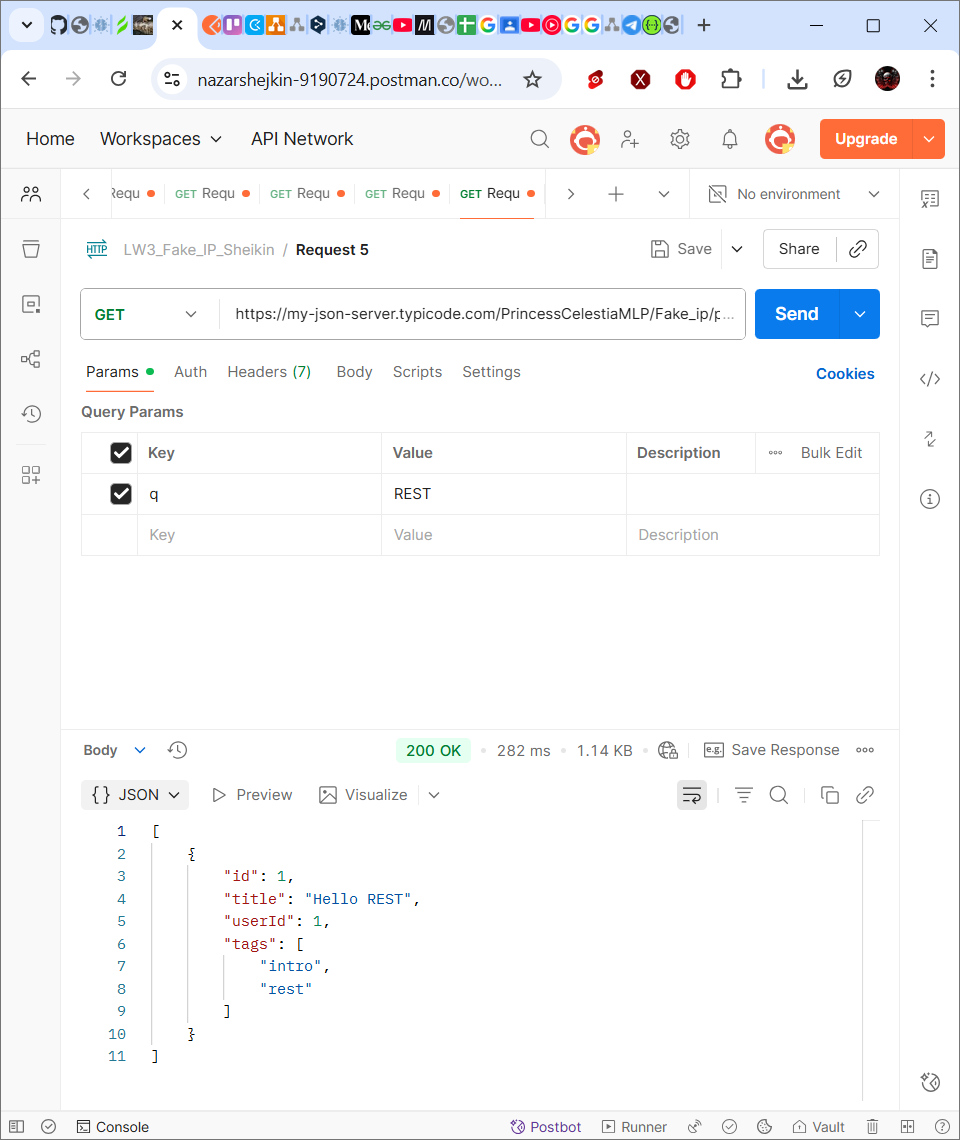


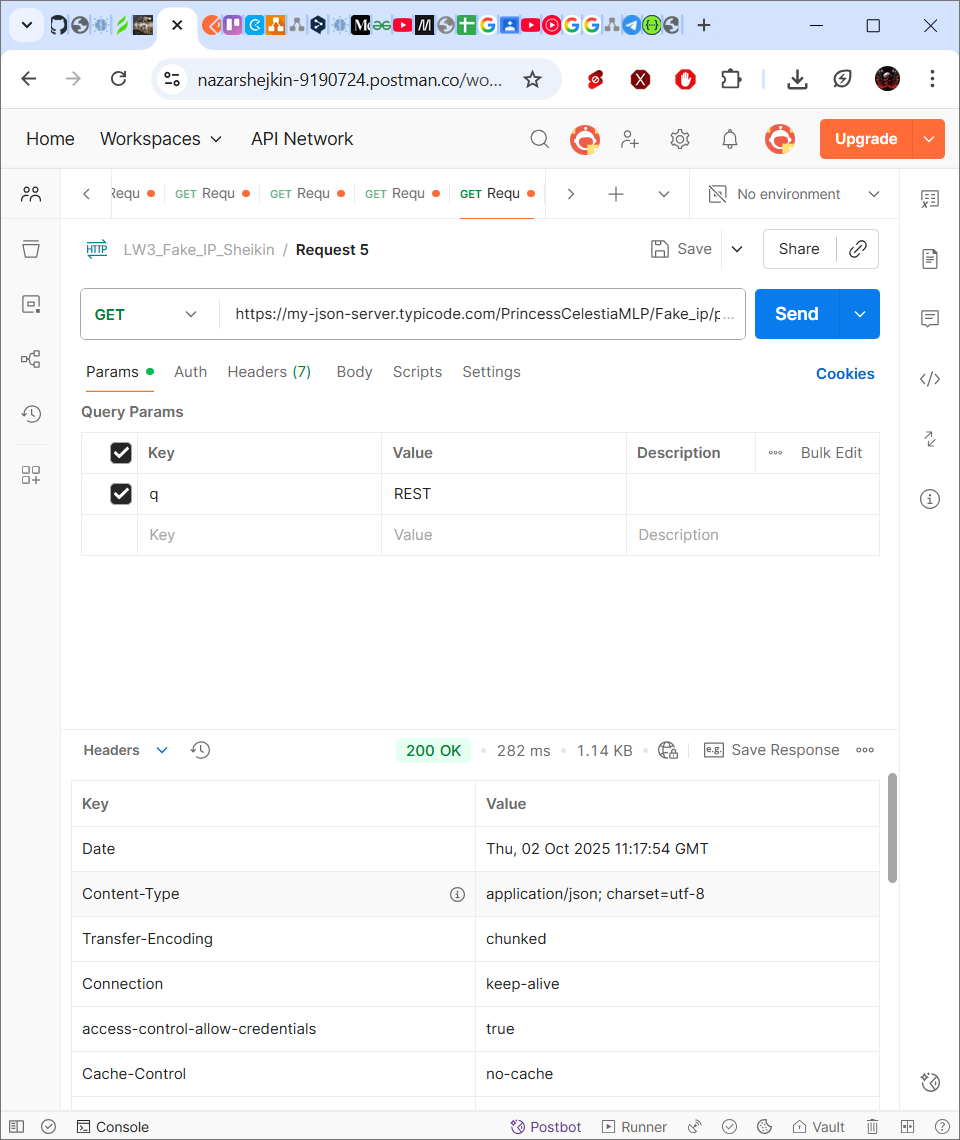
* 1. Виконаємо запит на пошук по фільтру GET /posts?userId=2



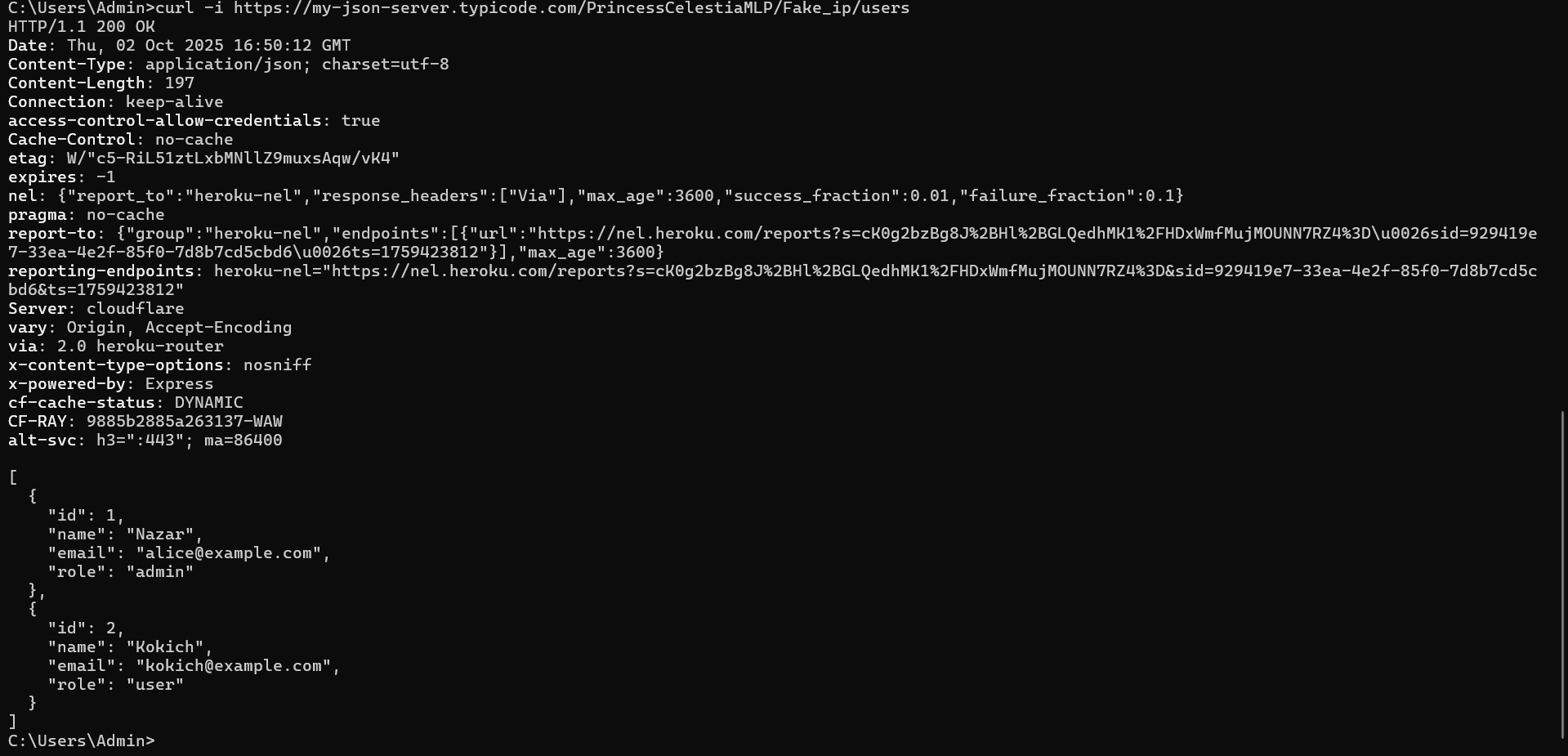


* 1. Виконаємо запит на пошук GET /posts?q=REST

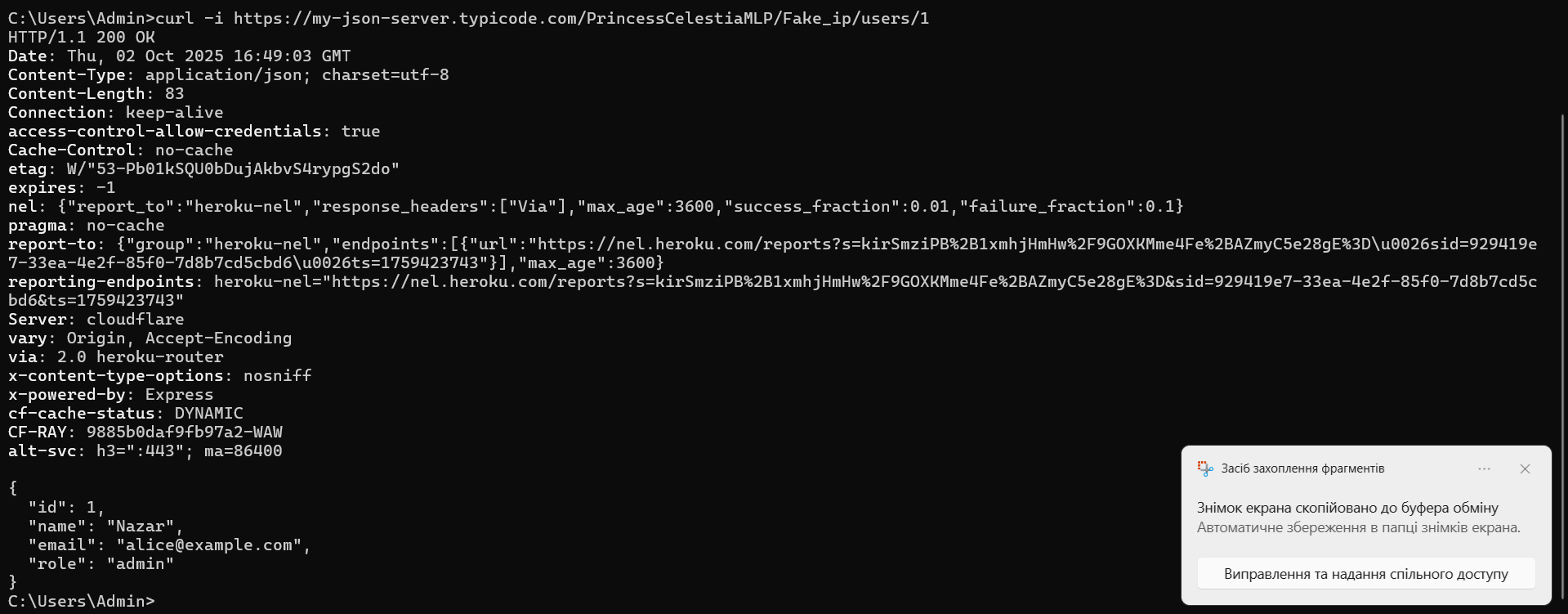




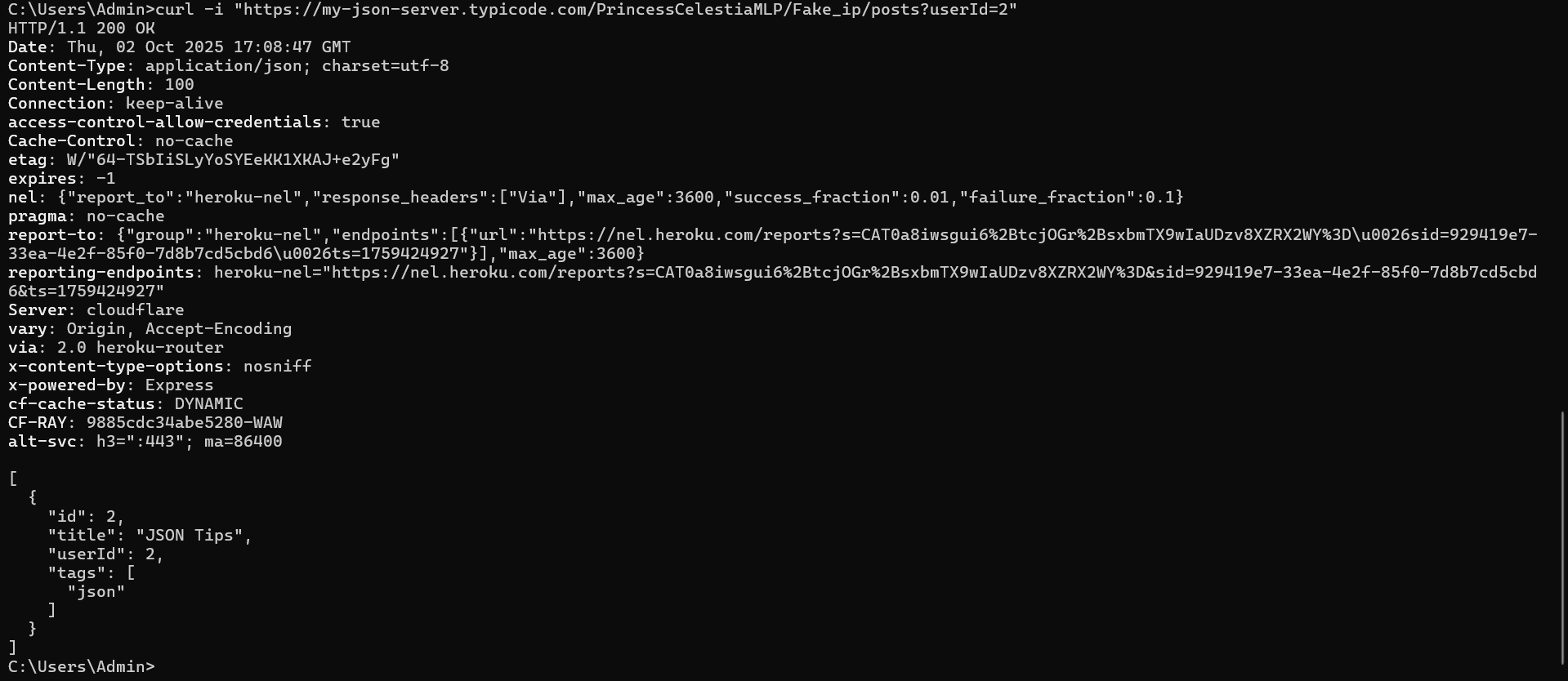
1. Приклади через curl -i:
   1. Усі користувачі



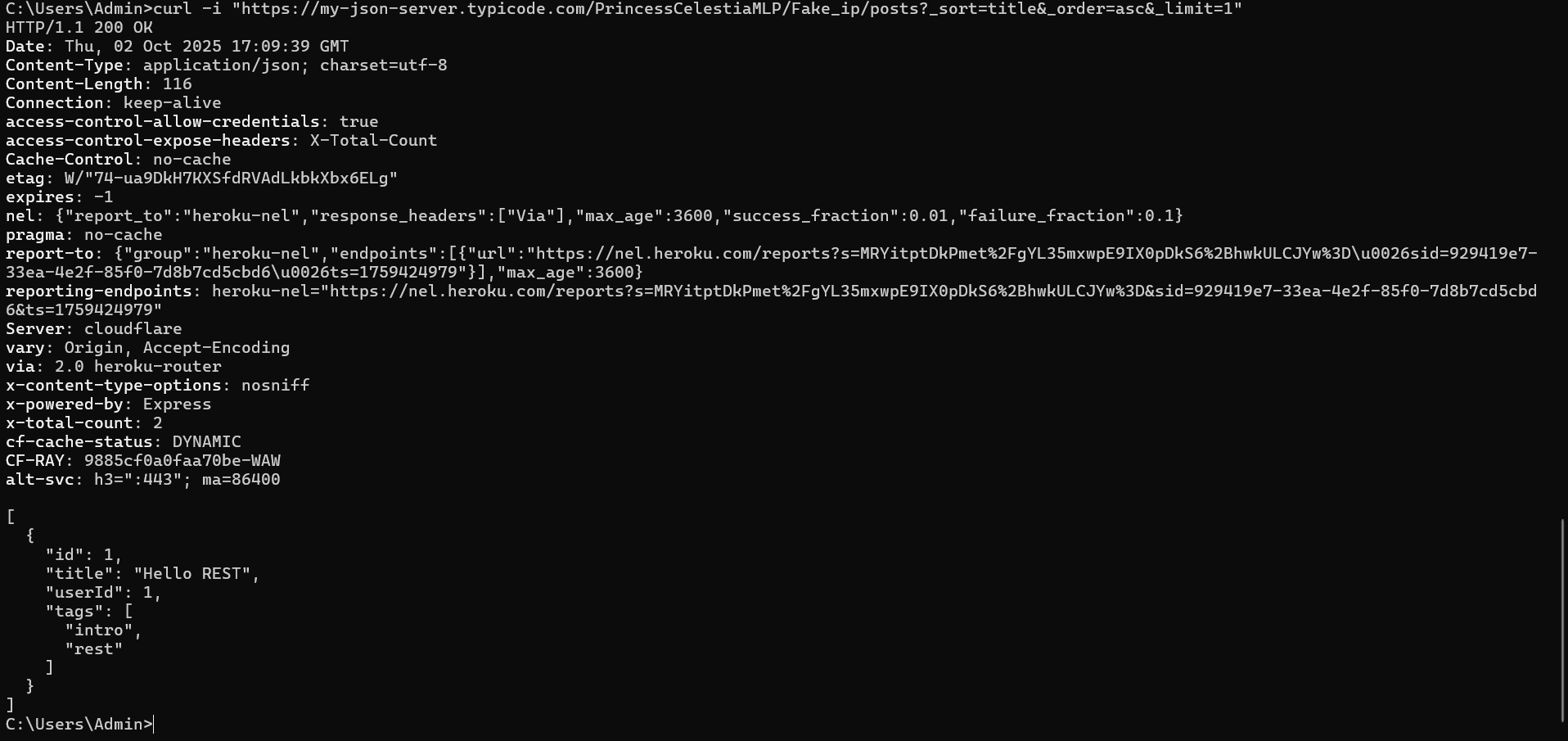
* 1. Один користувач



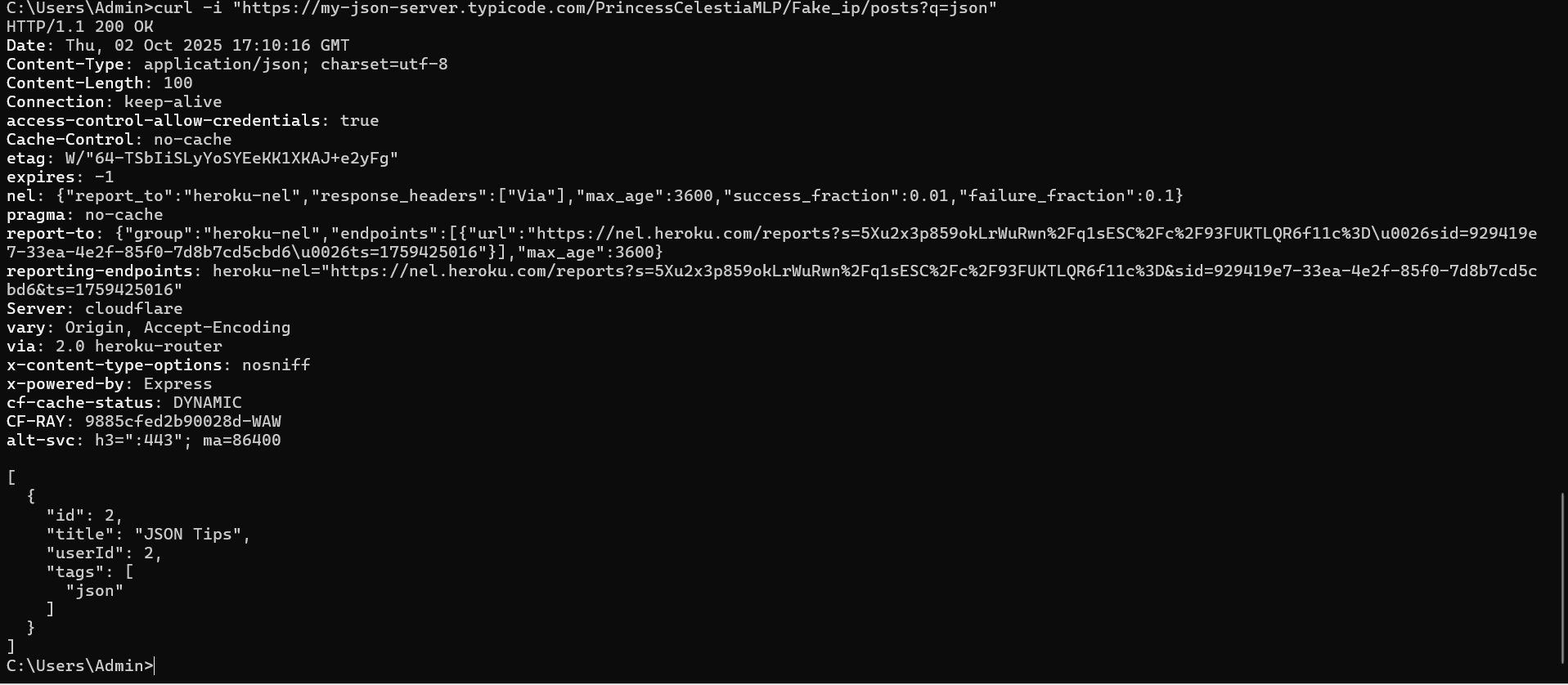
* 1. Фільтрація



* 1. Сортування + ліміт



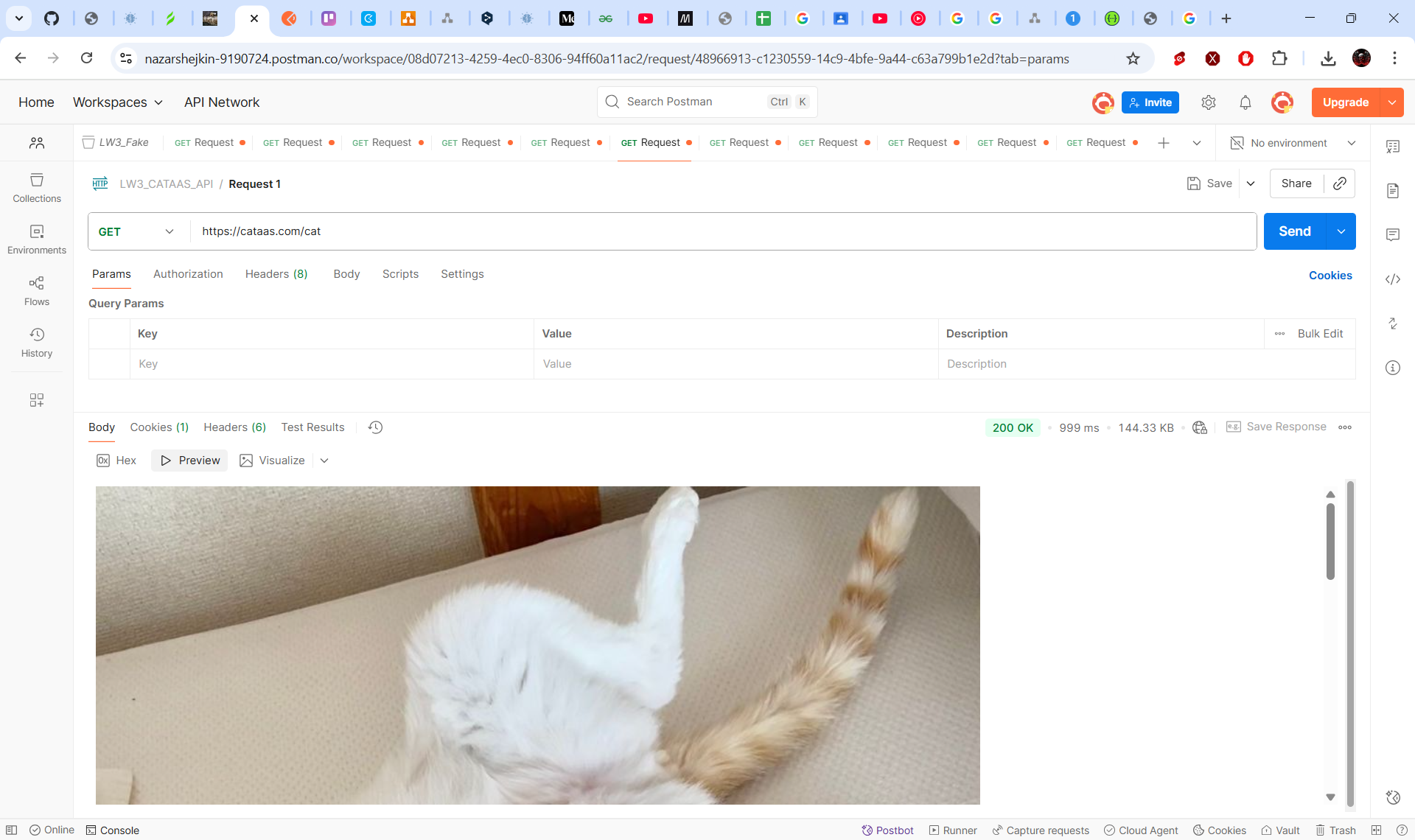
* 1. Пошук

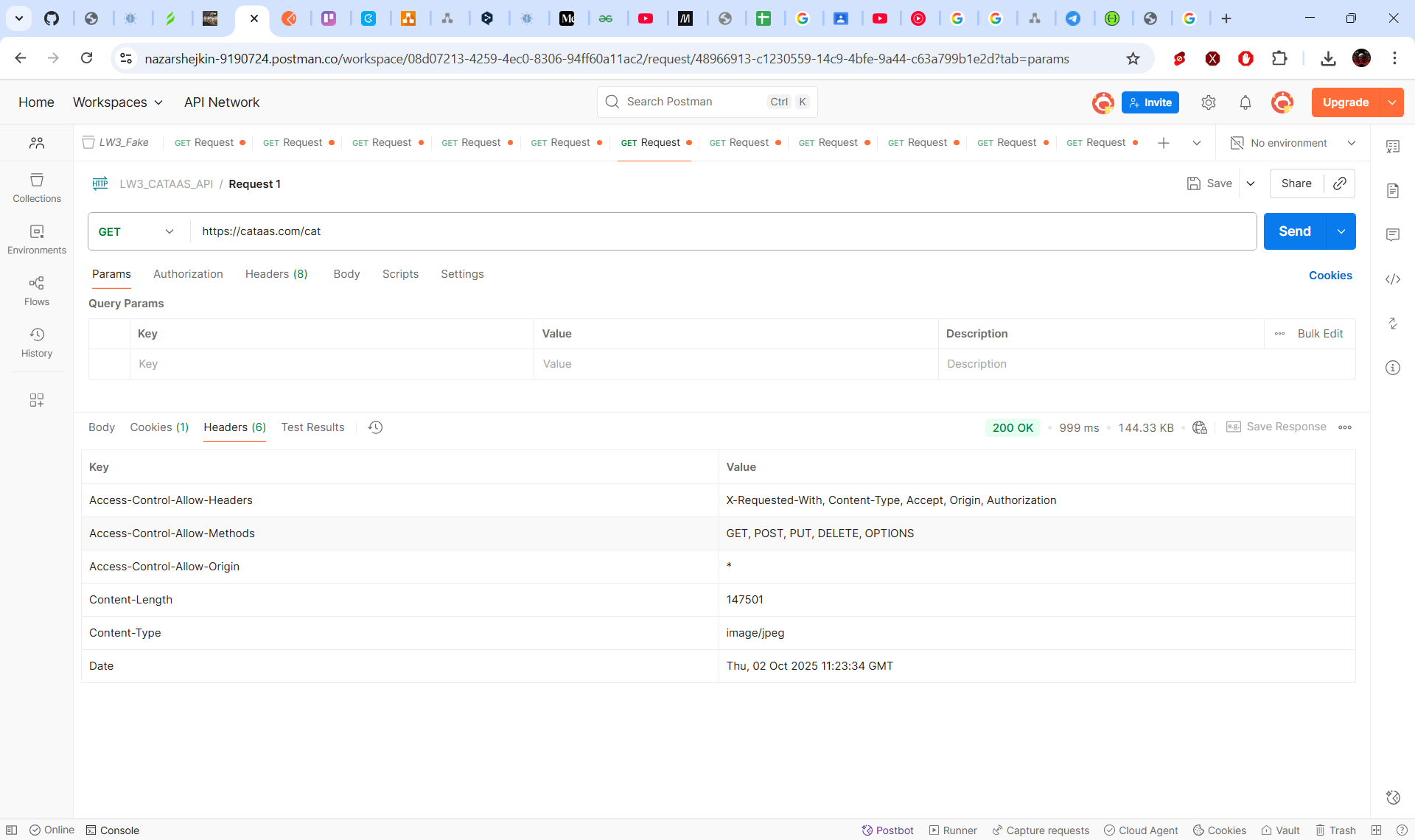


**Завдання №2**

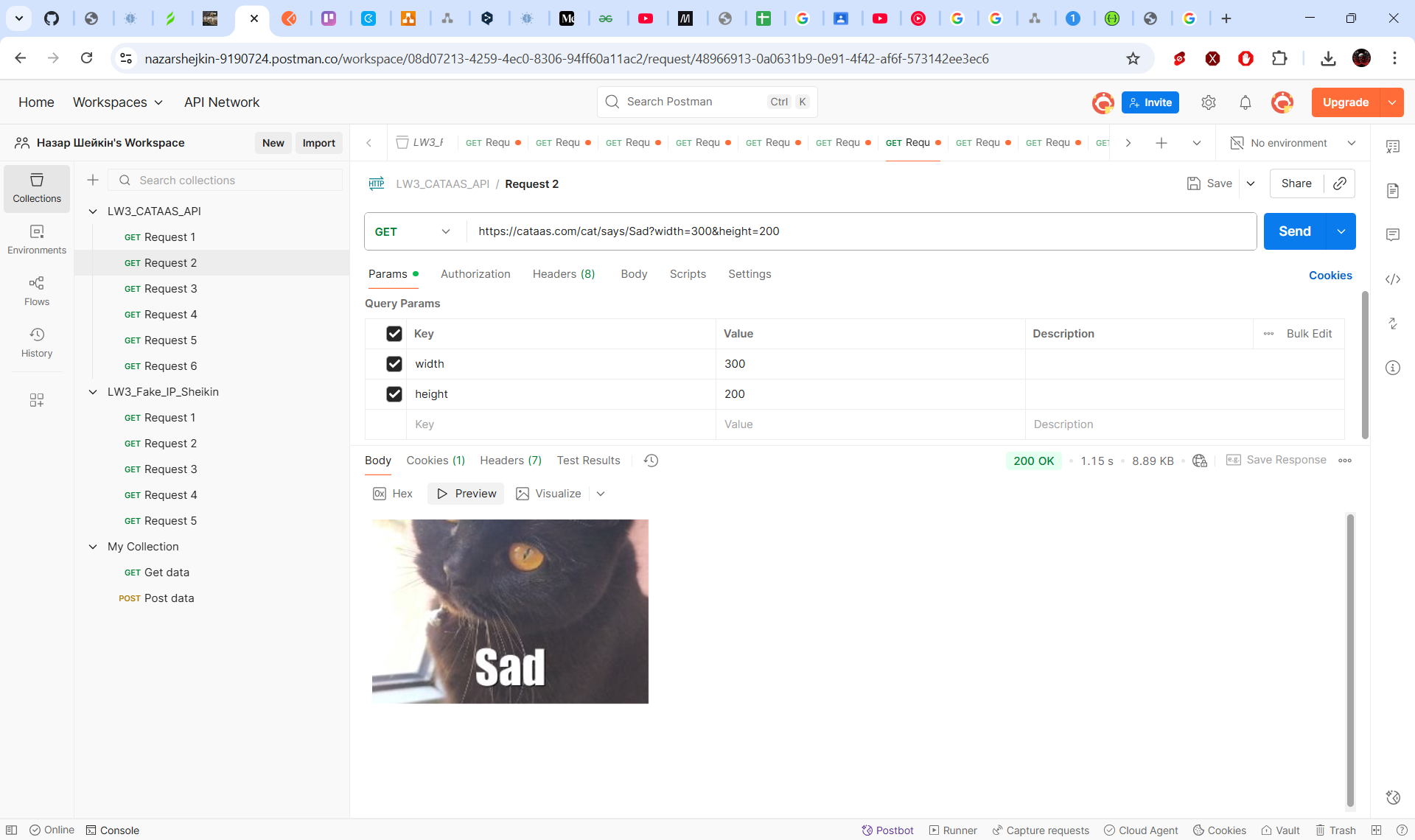
**Робота з API сервісу CATAAS (Cat as a Service) та SwaggerAPI**

1. Створюємо колекцію **LW3\_CATAAS\_API**
2. Виконуємо запити
   * Отримати випадкове зображення кота



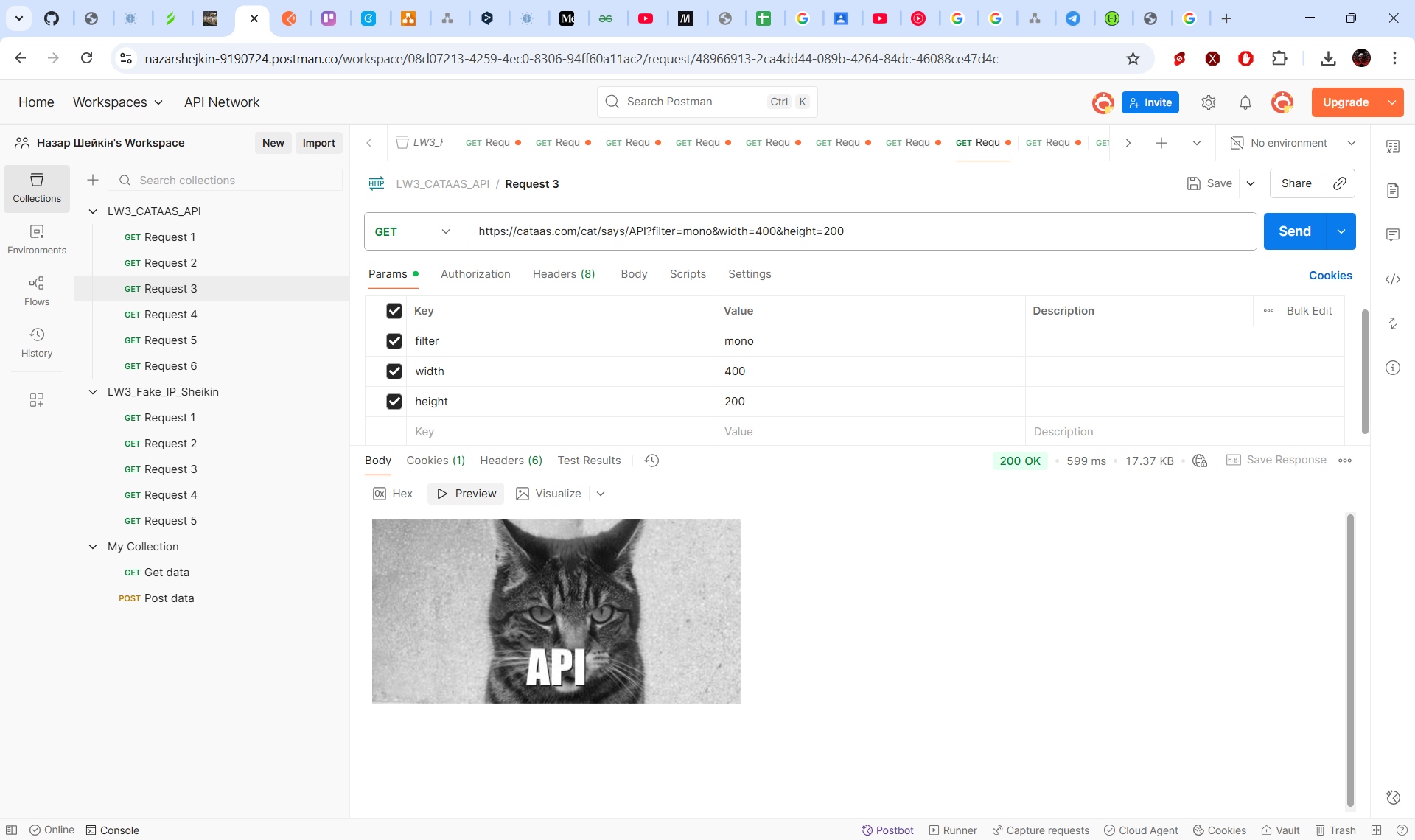


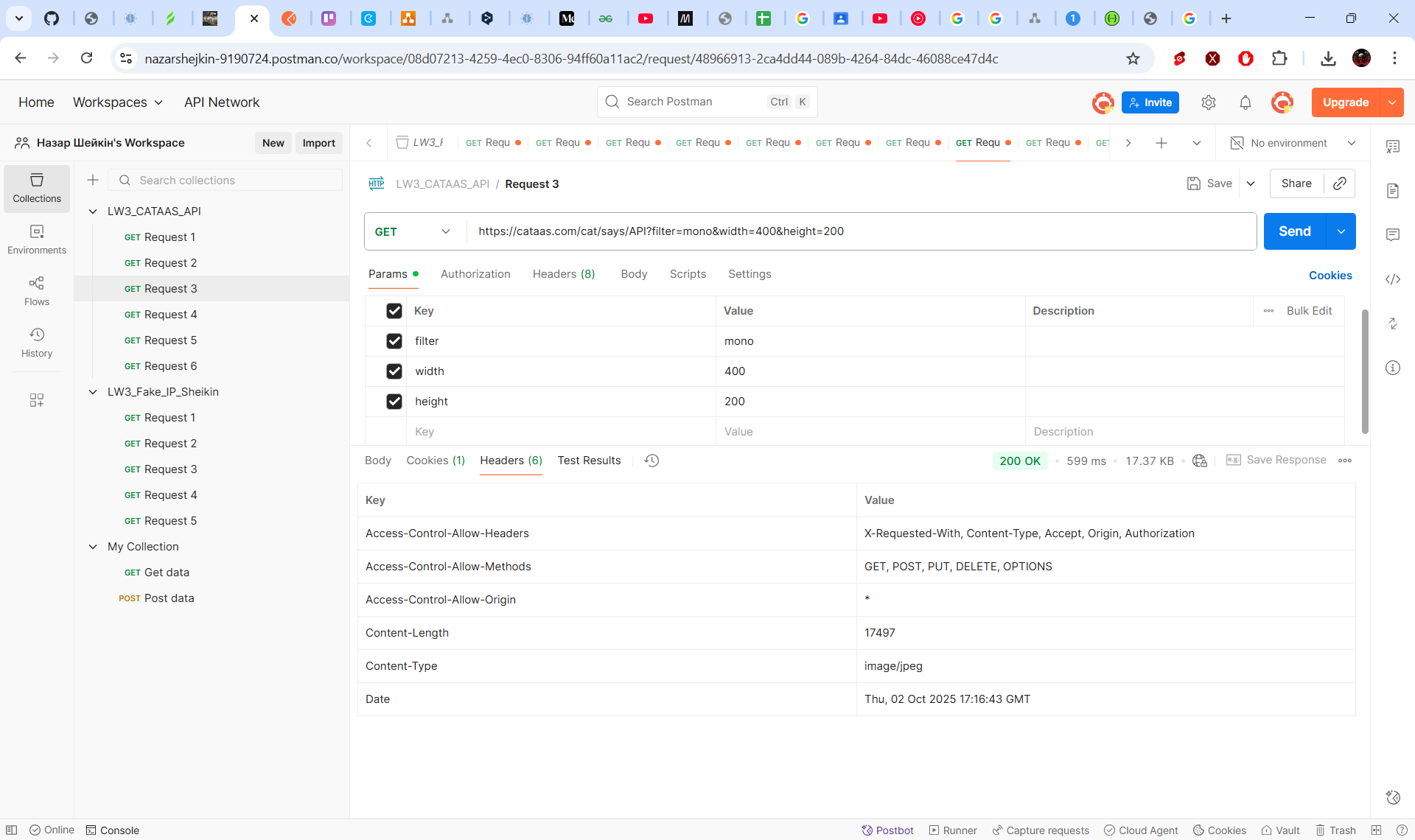
* Кіт із підписом вашого mood (картинка з текстом яка описує тебе).



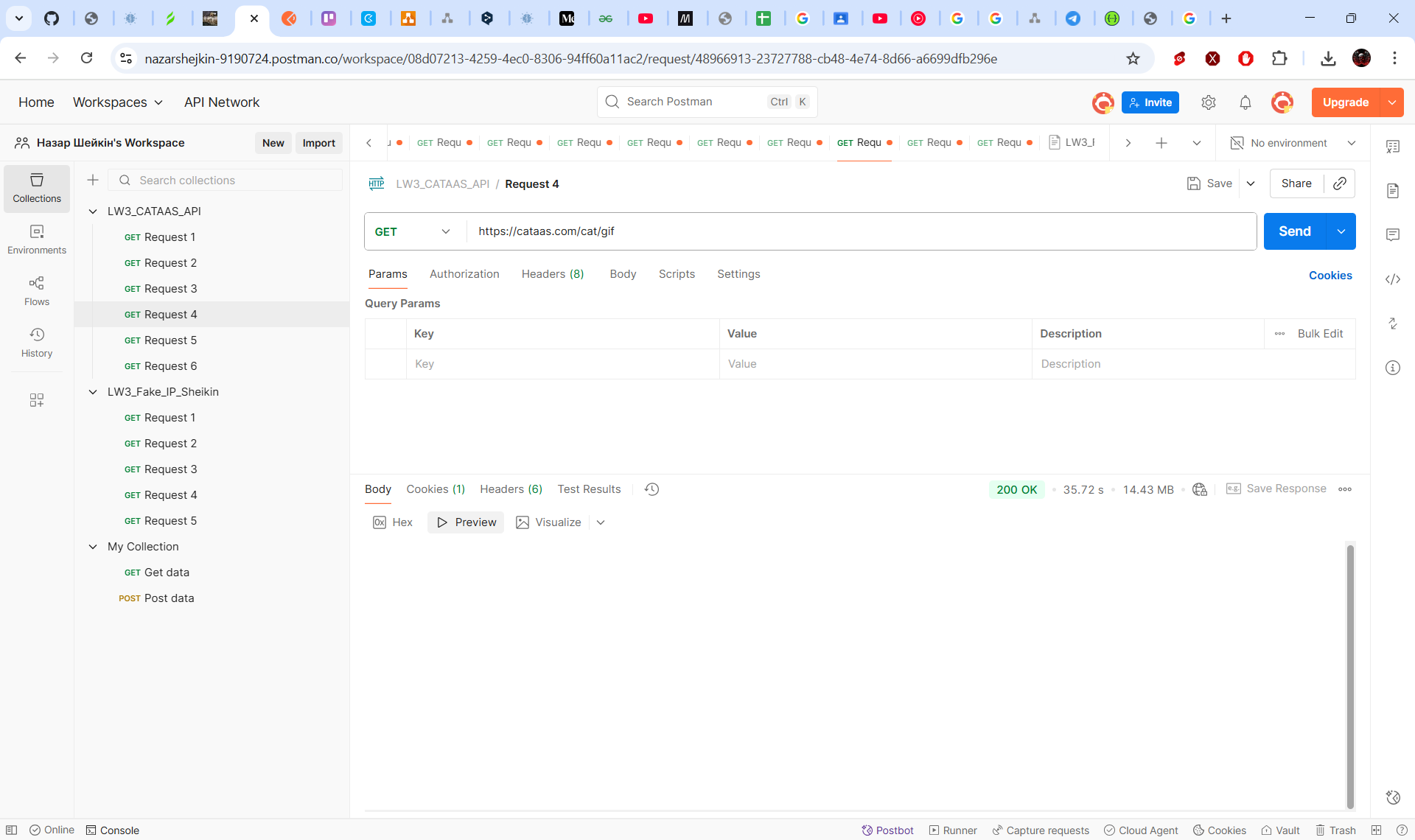


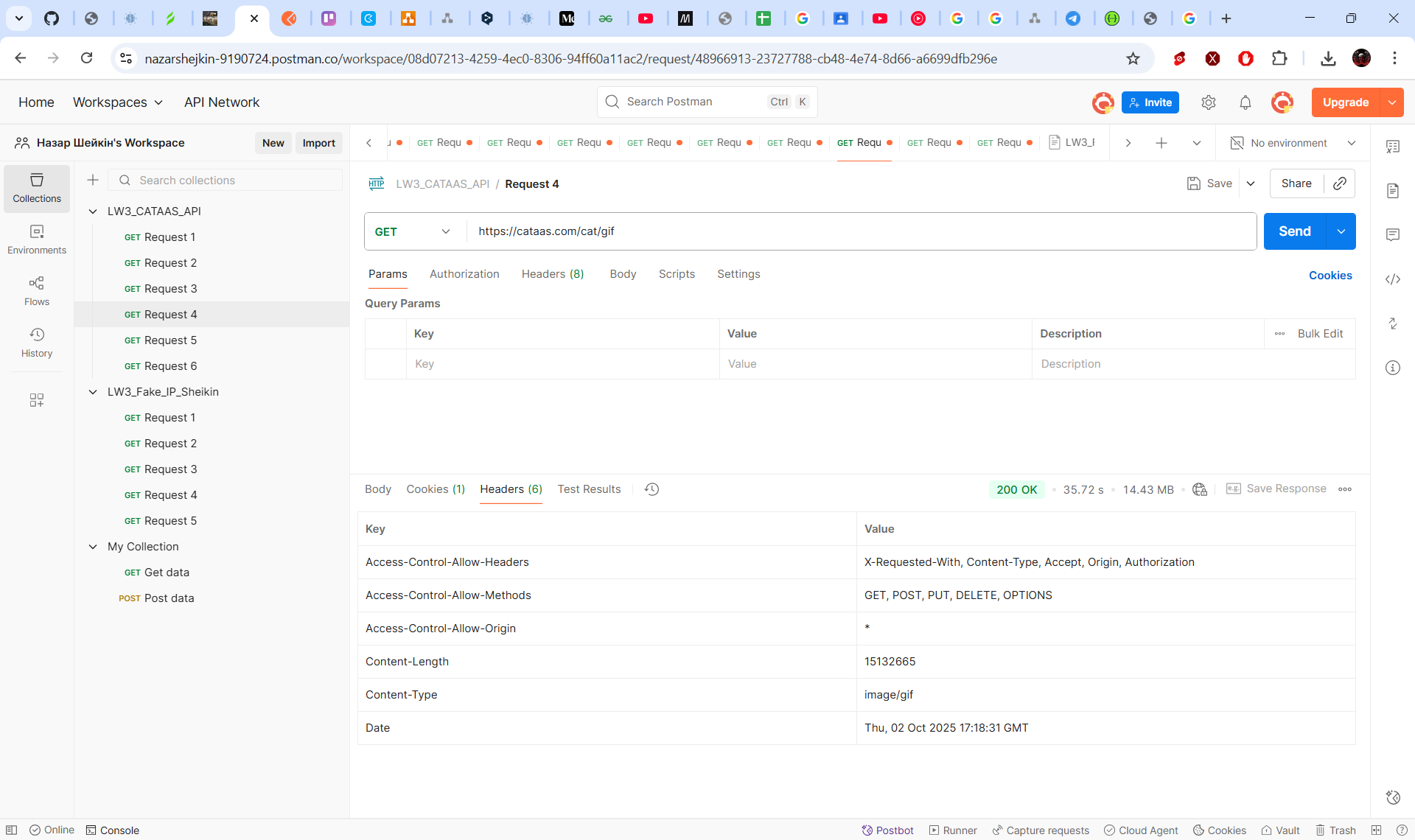
* Кіт із чорно-білим підписом. Спробуйте використати різні параметри запиту.



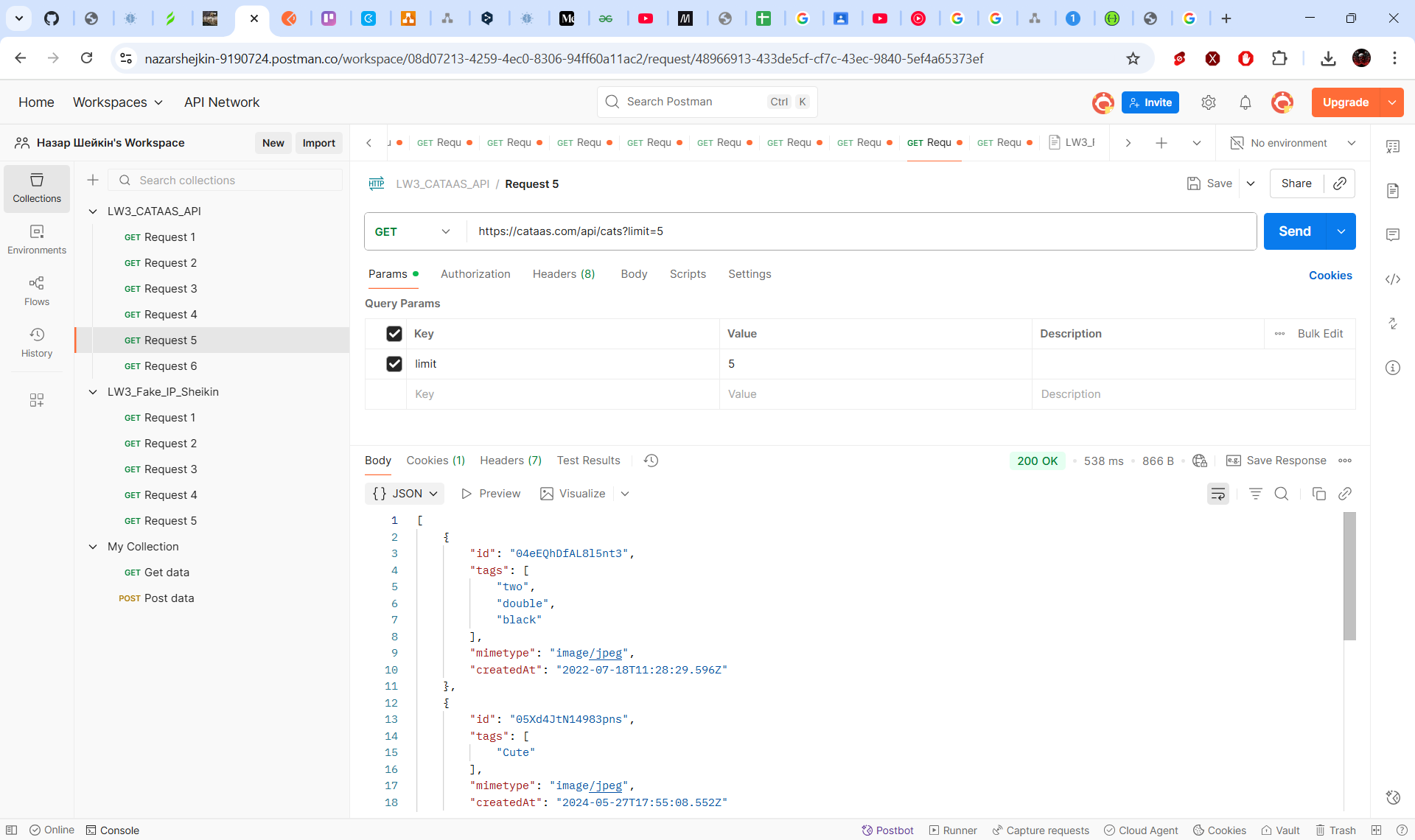


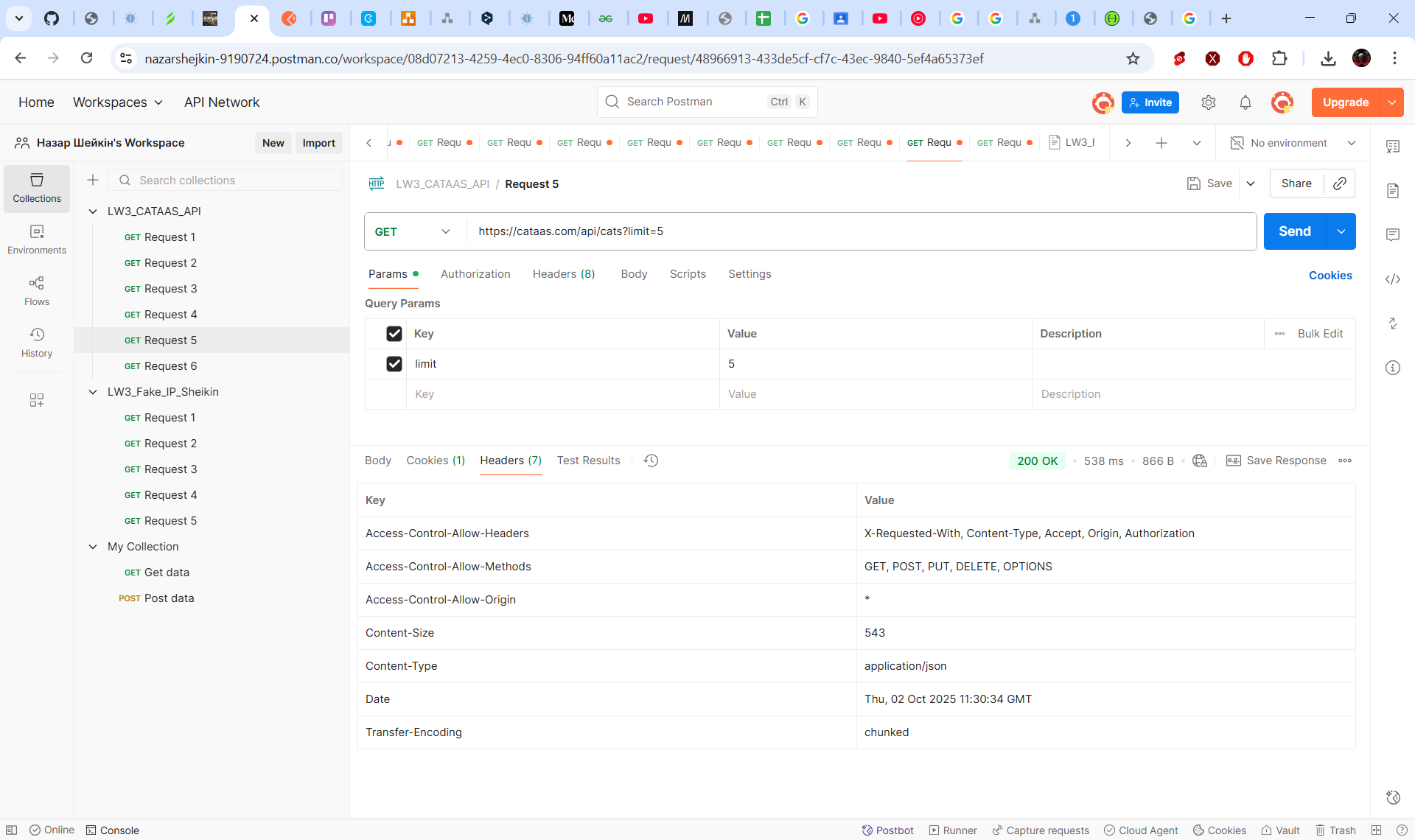
* Отримати випадкове gif-кота.





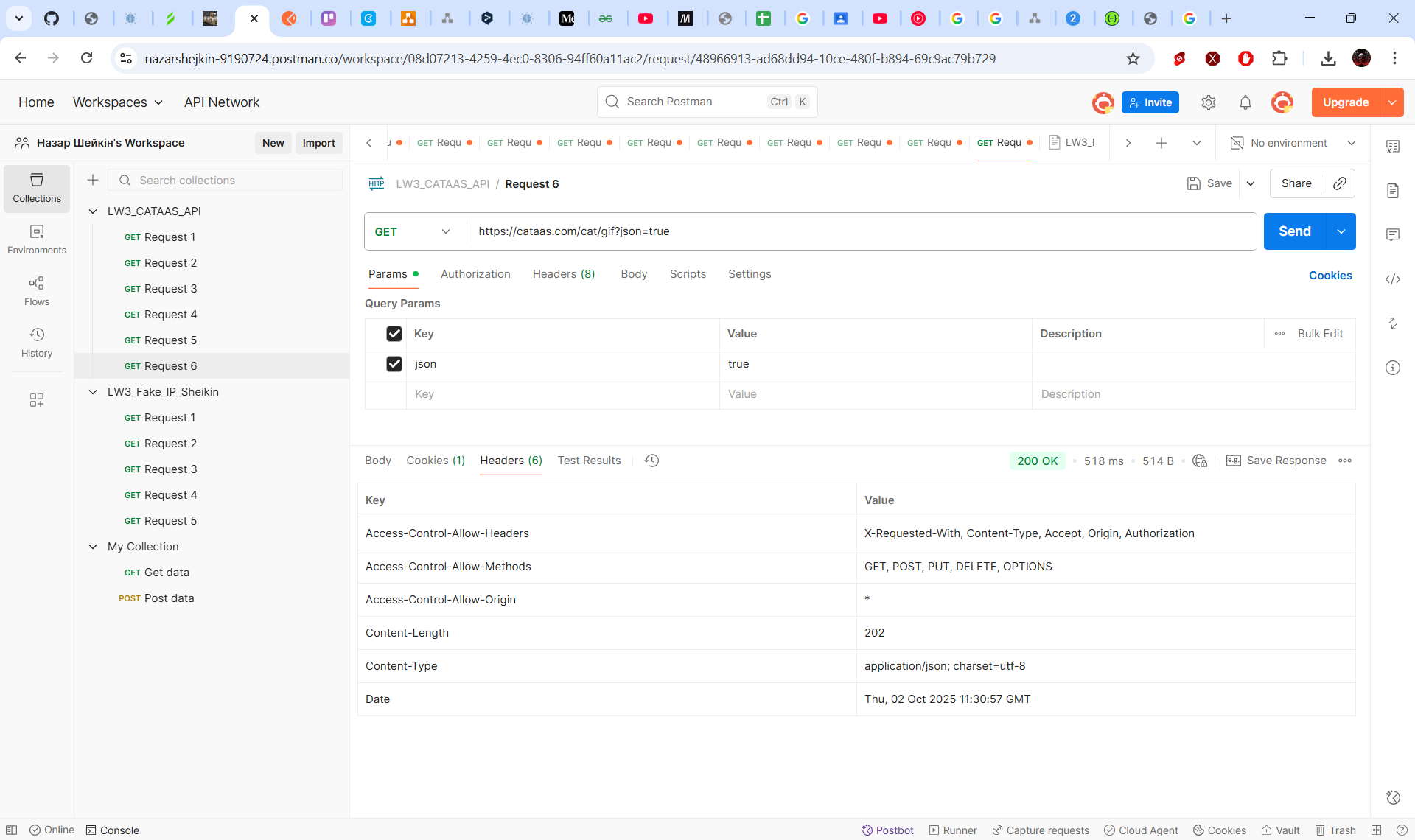
* Отримати JSON-список перших 5 котів.





* Отримати метадані кото-GIF у форматі JSON.





1. **Ознайомлення з документацією. Виконання запитів через Swagger UI**

Задамо код Swagger:

openapi: 3.0.0

info:

title: CATAAS demo via Swagger

version: 1.0.0

description: Демонстраційні ендпоїнти сервісу Cat as a Service.

servers:

# Added by API Auto Mocking Plugin

- description: SwaggerHub API Auto Mocking

url: https://virtserver.swaggerhub.com/hpkk/LW3MiA/1.0.0

- url: https://cataas.com

paths:

/cat:

get:

summary: Отримати випадкове зображення кота

responses:

'200':

description: Зображення кота

content:

image/jpeg:

schema:

type: string

format: binary

image/png:

schema:

type: string

format: binary

/cat/says/{text}:

get:

summary: Кіт із накладеним текстом

parameters:

- name: text

in: path

required: true

description: Текст, який буде відображено на зображенні

schema:

type: string

example: REST

responses:

'200':

description: Зображення кота з текстом

content:

image/jpeg:

schema:

type: string

format: binary

image/png:

schema:

type: string

format: binary

/api/cats:

get:

summary: Список метаданих зображень котів

responses:

'200':

description: JSON-масив із метаданими

content:

application/json:

schema:

type: array

items:

type: object

properties:

\_id:

type: string

description: Ідентифікатор зображення

created\_at:

type: string

description: Дата створення

tags:

type: array

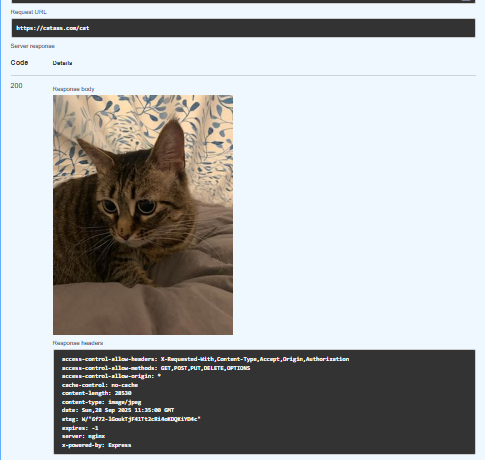
items:

type: string

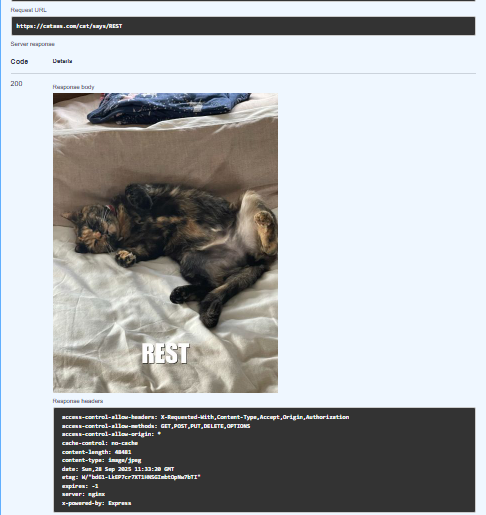
description: Теги зображення

**Виконаємо запити**

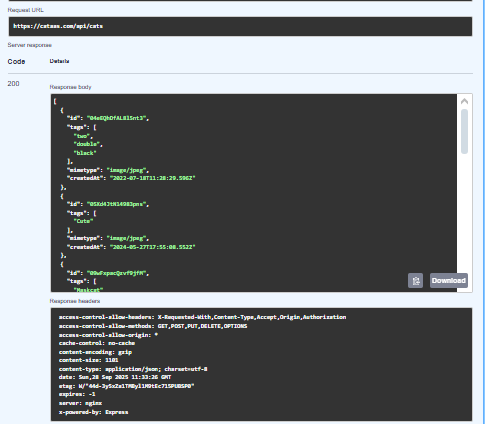
* 1. Виконаємо запит кота



* 1. Виконаємо запит на кота з текстом



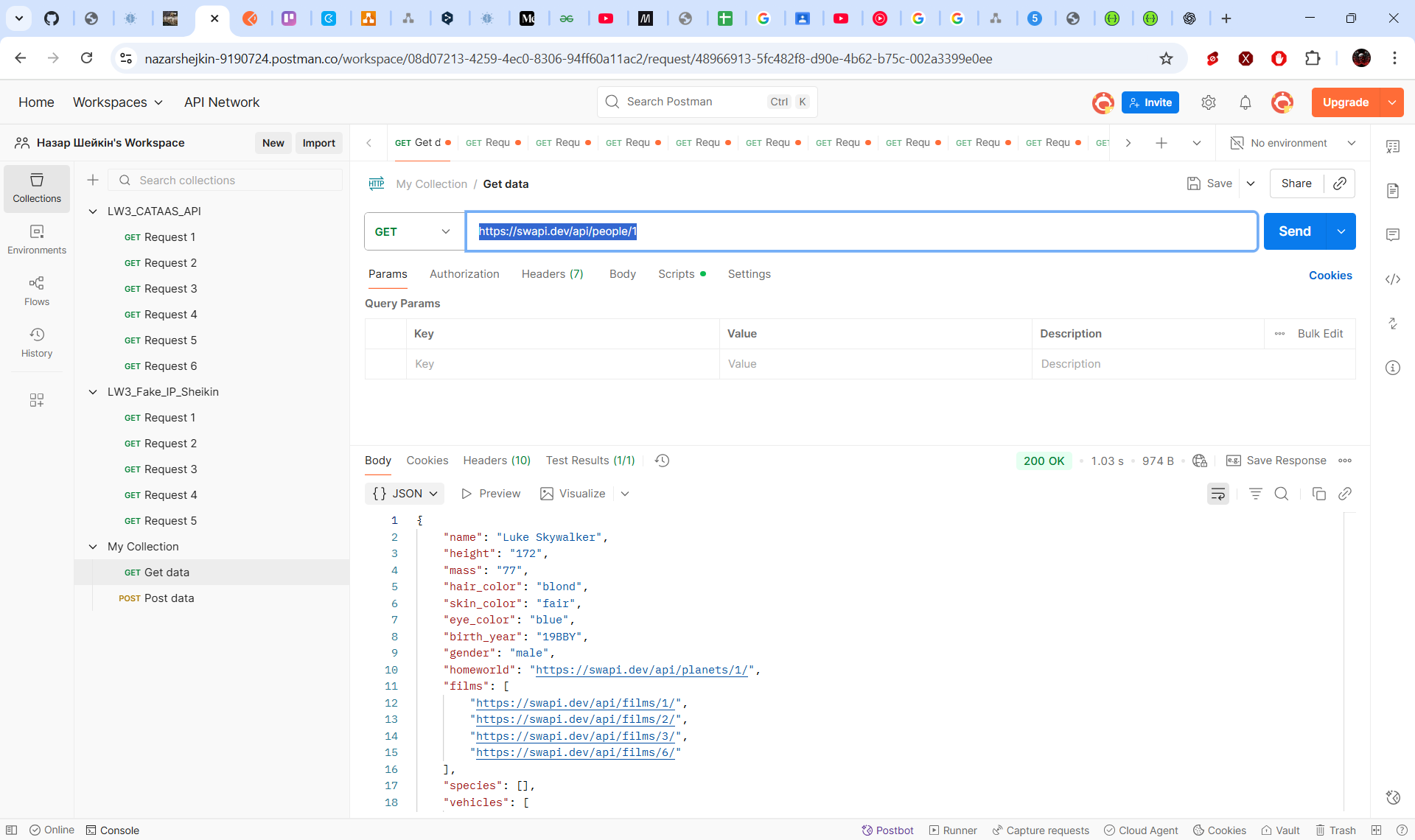
* 1. Виконаємо запит на усіх котів



**Завдання №3**

**Console-клієнт до публічного REST API**

1. Створюємо консольний проєкт С# з назвою API.
2. Виконаємо тестовий запит через PostMan



1. Створимо класи:

public class Person

{

public string name { get; set; } = string.Empty;

public string height { get; set; } = string.Empty;

public string mass { get; set; } = string.Empty;

public string hair\_color { get; set; } = string.Empty;

public string skin\_color { get; set; } = string.Empty;

public string eye\_color { get; set; } = string.Empty;

public string birth\_year { get; set; } = string.Empty;

public string gender { get; set; } = string.Empty;

public string homeworld { get; set; } = string.Empty;

public List<string> films { get; set; } = new();

public List<string> species { get; set; } = new();

public List<string> vehicles { get; set; } = new();

public List<string> starships { get; set; } = new();

public string created { get; set; } = string.Empty;

public string edited { get; set; } = string.Empty;

public string url { get; set; } = string.Empty;

}

1. Налаштуємо HttpClient.

private static readonly HttpClient \_http = new HttpClient

{

BaseAddress = new Uri("https://swapi.dev/api/"),

Timeout = TimeSpan.FromSeconds(15)

};

1. Виконаємо GET запит і запитуємо про 3 людину

var response = await var response = await \_http.GetAsync("people/3/");

response.EnsureSuccessStatusCode();

1. Отримаємо JSON як рядок

var json = await response.Content.ReadAsStringAsync();

Console.WriteLine("Raw JSON:");

Console.WriteLine(json);

1. Розпарсимо JSON

var person = JsonSerializer.Deserialize<Person>(json, new JsonSerializerOptions

{

PropertyNameCaseInsensitive = true

});

1. Використаємо дані з об'єкта

if (person != null)

{

Console.WriteLine("\n=== Результат парсингу ===");

Console.WriteLine($"Name: {person.name}");

Console.WriteLine($"Height: {person.height}");

Console.WriteLine($"Mass: {person.mass}");

Console.WriteLine($"Hair color: {person.hair\_color}");

Console.WriteLine($"Eye color: {person.eye\_color}");

Console.WriteLine($"Birth year: {person.birth\_year}");

Console.WriteLine($"Gender: {person.gender}");

}

1. Повний код програми:

using System;

using System.Net.Http;

using System.Text.Json;

using System.Threading.Tasks;

using System.Collections.Generic;

public class Person

{

public string name { get; set; } = string.Empty;

public string height { get; set; } = string.Empty;

public string mass { get; set; } = string.Empty;

public string hair\_color { get; set; } = string.Empty;

public string skin\_color { get; set; } = string.Empty;

public string eye\_color { get; set; } = string.Empty;

public string birth\_year { get; set; } = string.Empty;

public string gender { get; set; } = string.Empty;

public string homeworld { get; set; } = string.Empty;

public List<string> films { get; set; } = new();

public List<string> species { get; set; } = new();

public List<string> vehicles { get; set; } = new();

public List<string> starships { get; set; } = new();

public string created { get; set; } = string.Empty;

public string edited { get; set; } = string.Empty;

public string url { get; set; } = string.Empty;

}

class Program

{

private static readonly HttpClient \_http = new HttpClient

{

BaseAddress = new Uri("https://swapi.dev/api/"),

Timeout = TimeSpan.FromSeconds(15)

};

static async Task Main()

{

try

{

Console.WriteLine("Запит до SWAPI...");

var response = await \_http.GetAsync("people/1/"); // Люк Скайвокер

response.EnsureSuccessStatusCode();

var json = await response.Content.ReadAsStringAsync();

Console.WriteLine("Raw JSON:");

Console.WriteLine(json);

var person = JsonSerializer.Deserialize<Person>(json, new JsonSerializerOptions

{

PropertyNameCaseInsensitive = true

});

if (person != null)

{

Console.WriteLine("\n=== Результат парсингу ===");

Console.WriteLine($"Name: {person.name}");

Console.WriteLine($"Height: {person.height}");

Console.WriteLine($"Mass: {person.mass}");

Console.WriteLine($"Hair color: {person.hair\_color}");

Console.WriteLine($"Eye color: {person.eye\_color}");

Console.WriteLine($"Birth year: {person.birth\_year}");

Console.WriteLine($"Gender: {person.gender}");

}

}

catch (Exception ex)

{

Console.WriteLine("Помилка: " + ex.Message);

}

}

}

**Результат програми:**

