# Лаборатотрна робота №3

# Клієнт-серверна архітектура ПЗ. Створення RESTful API

#### 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**

Створити свій Fake Online REST Server і виконати базові HTTP-запити

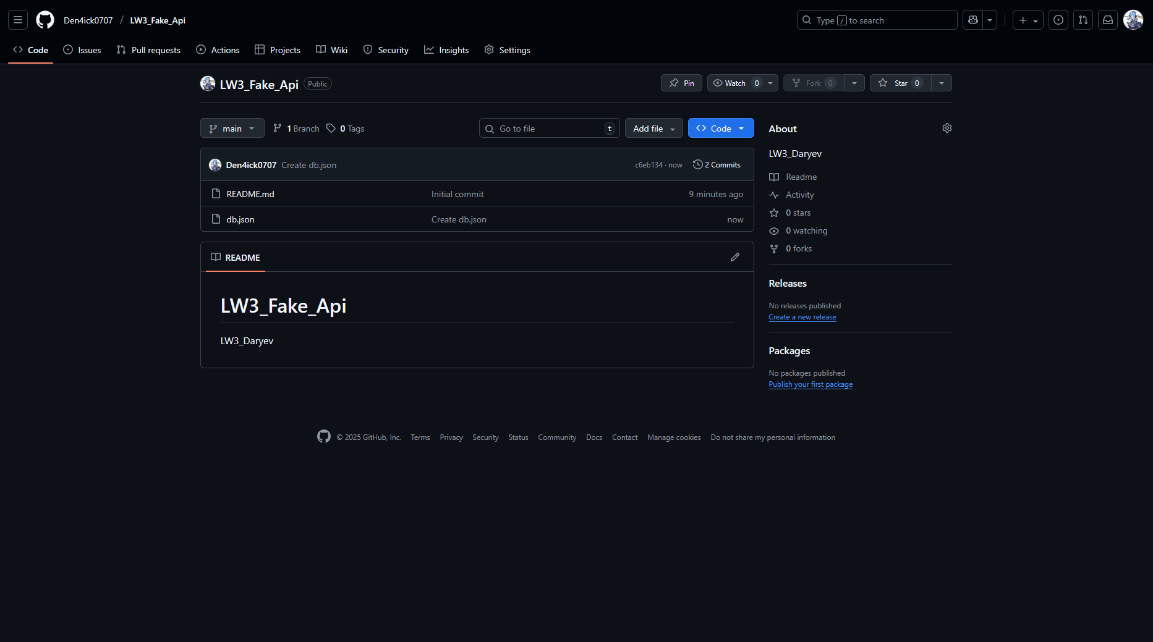


Рисунок 1 – Створення Fake\_Api репозиторію та db.json

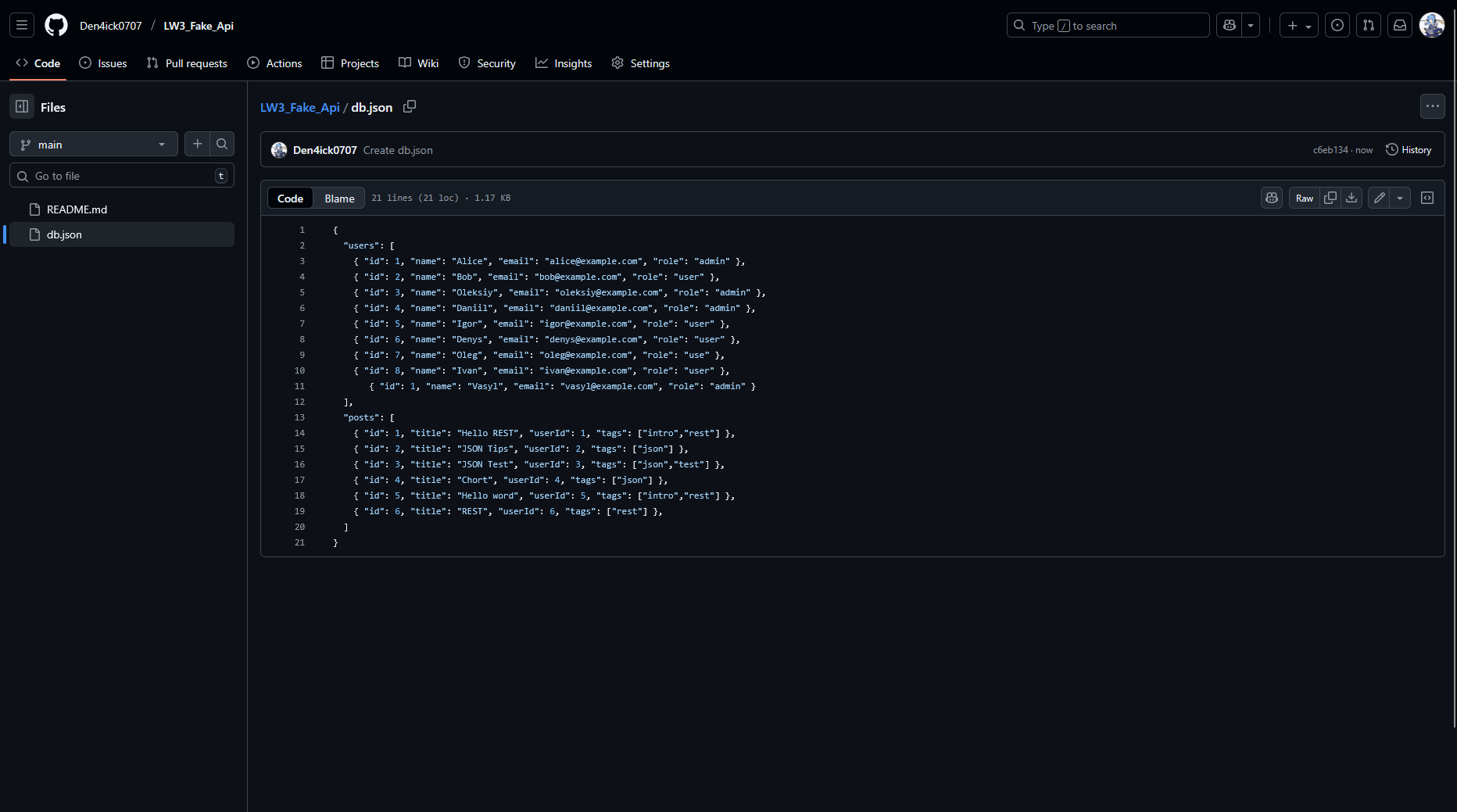


Рисунок 2 – Вміст db.json

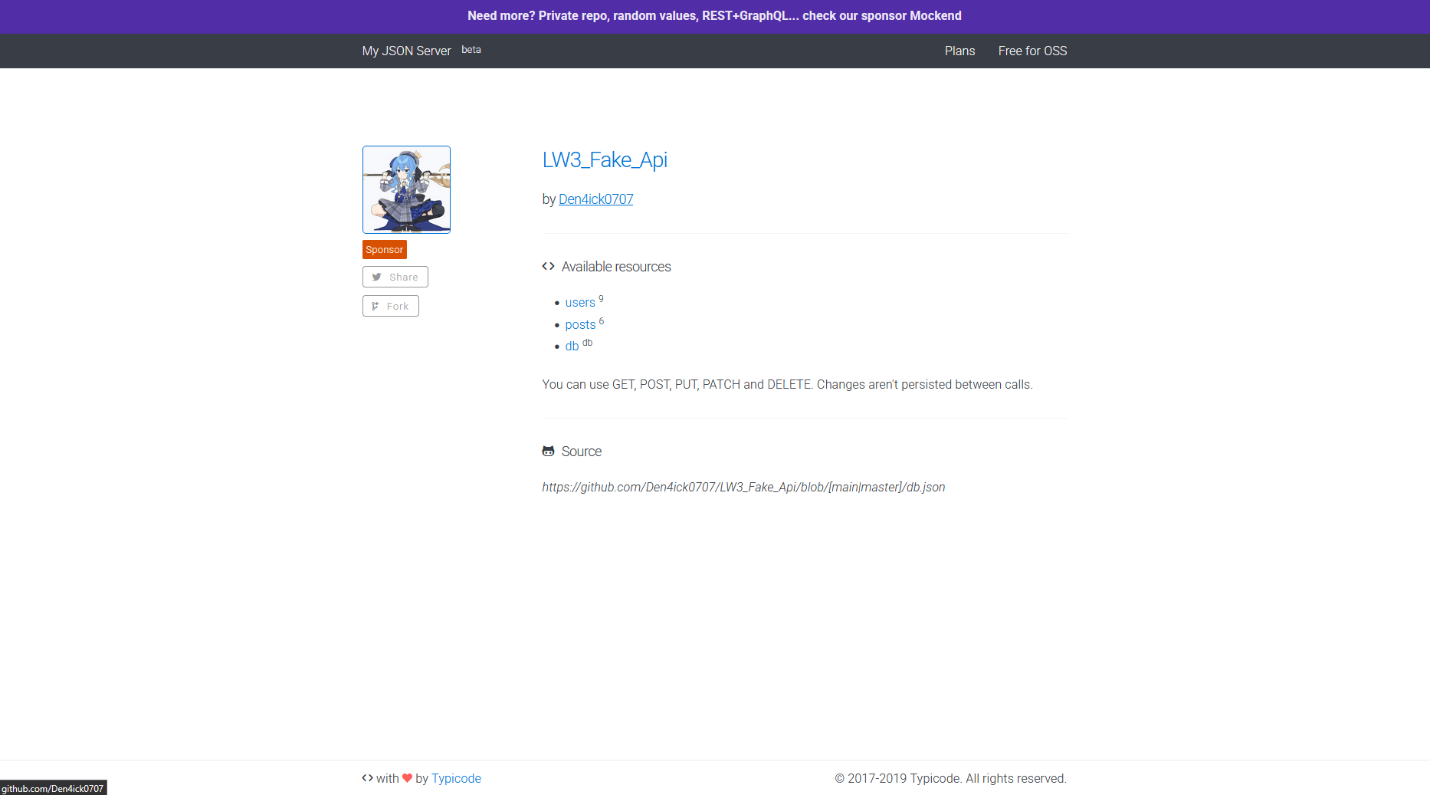


Рисунок 3 – Сайт <https://my-json-server.typicode.com/Den4ick0707/LW3_Fake_Api>

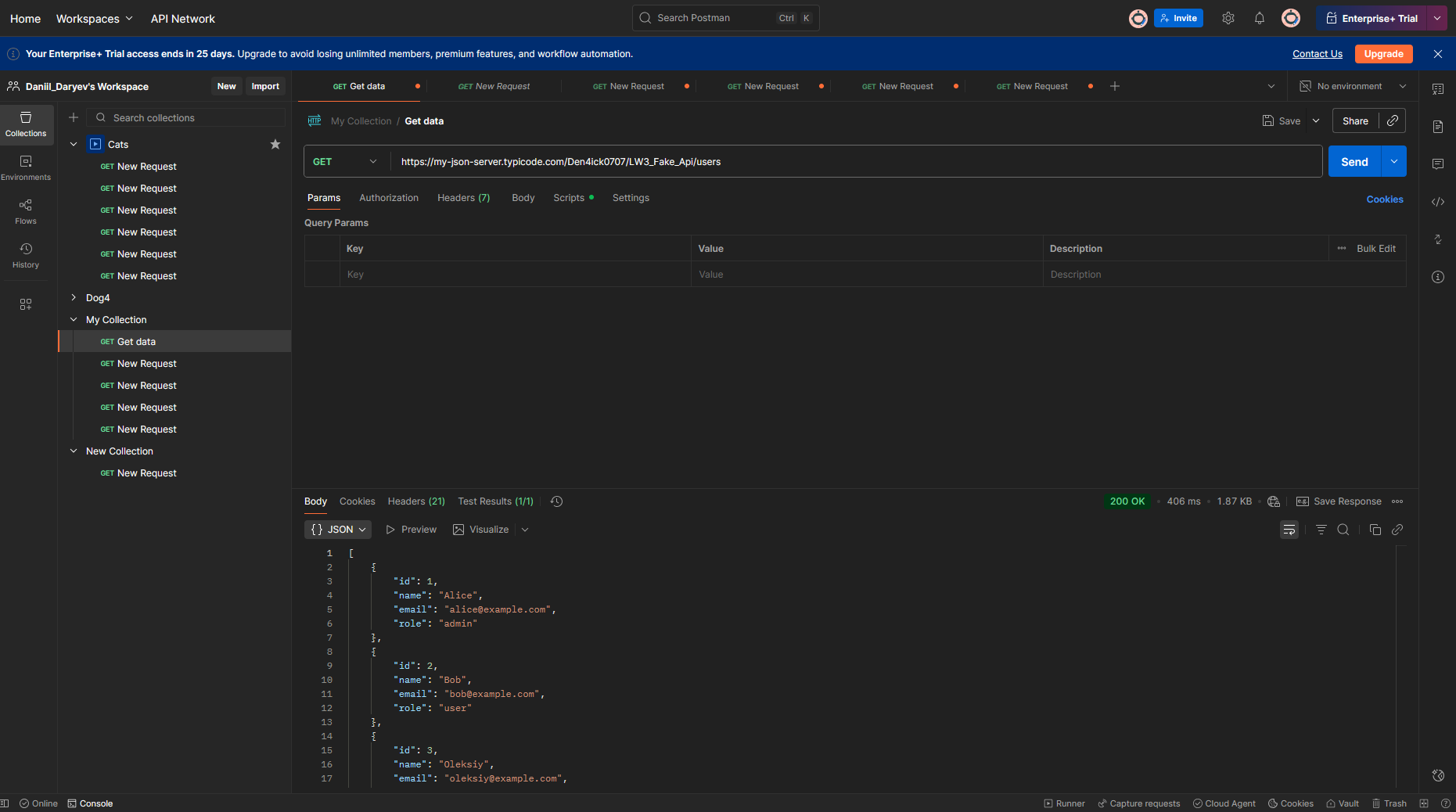


Рисунок 4 – Отримати список користувачів(Postman)

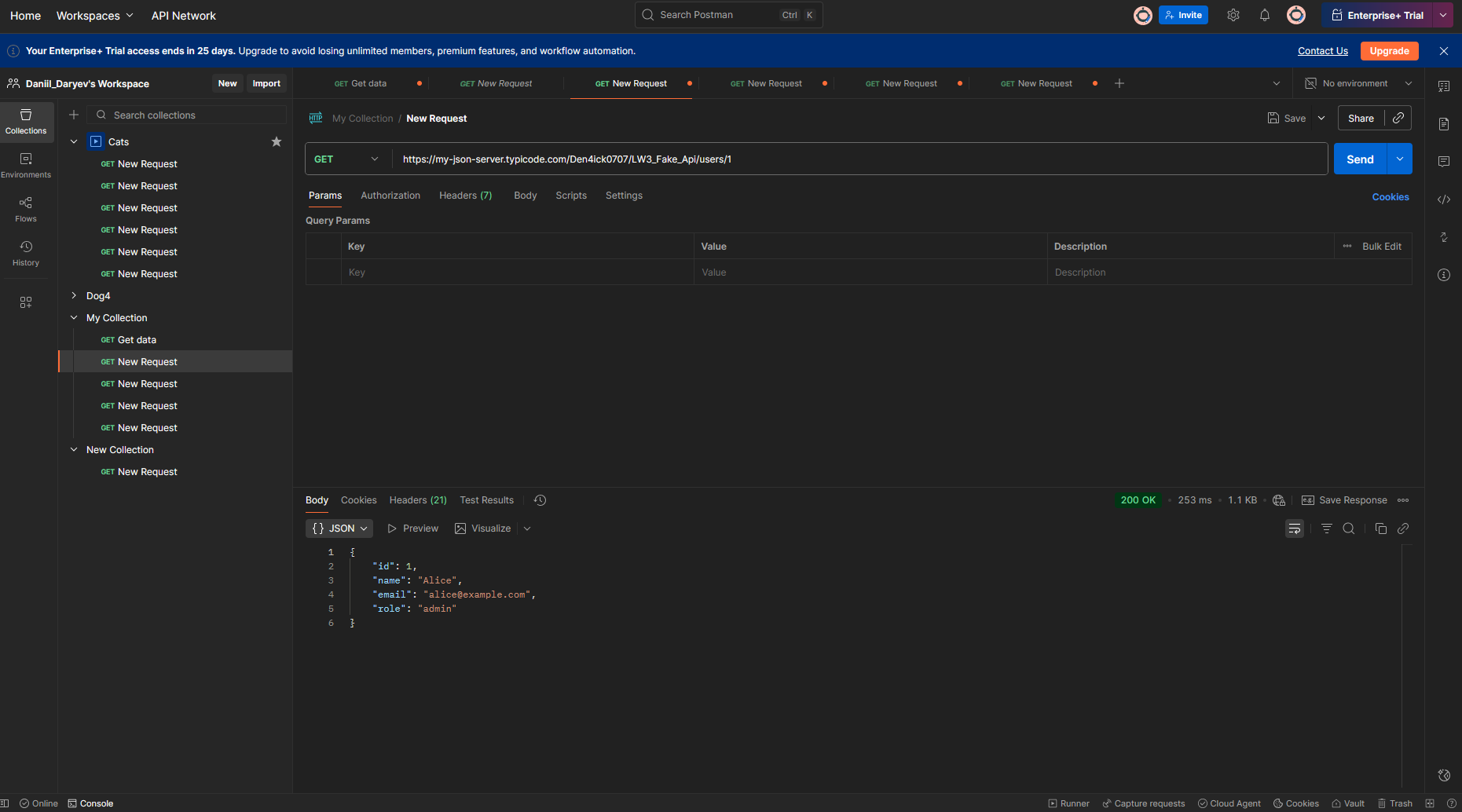


Рисунок 5 – Отримати одного користувача(Postman)

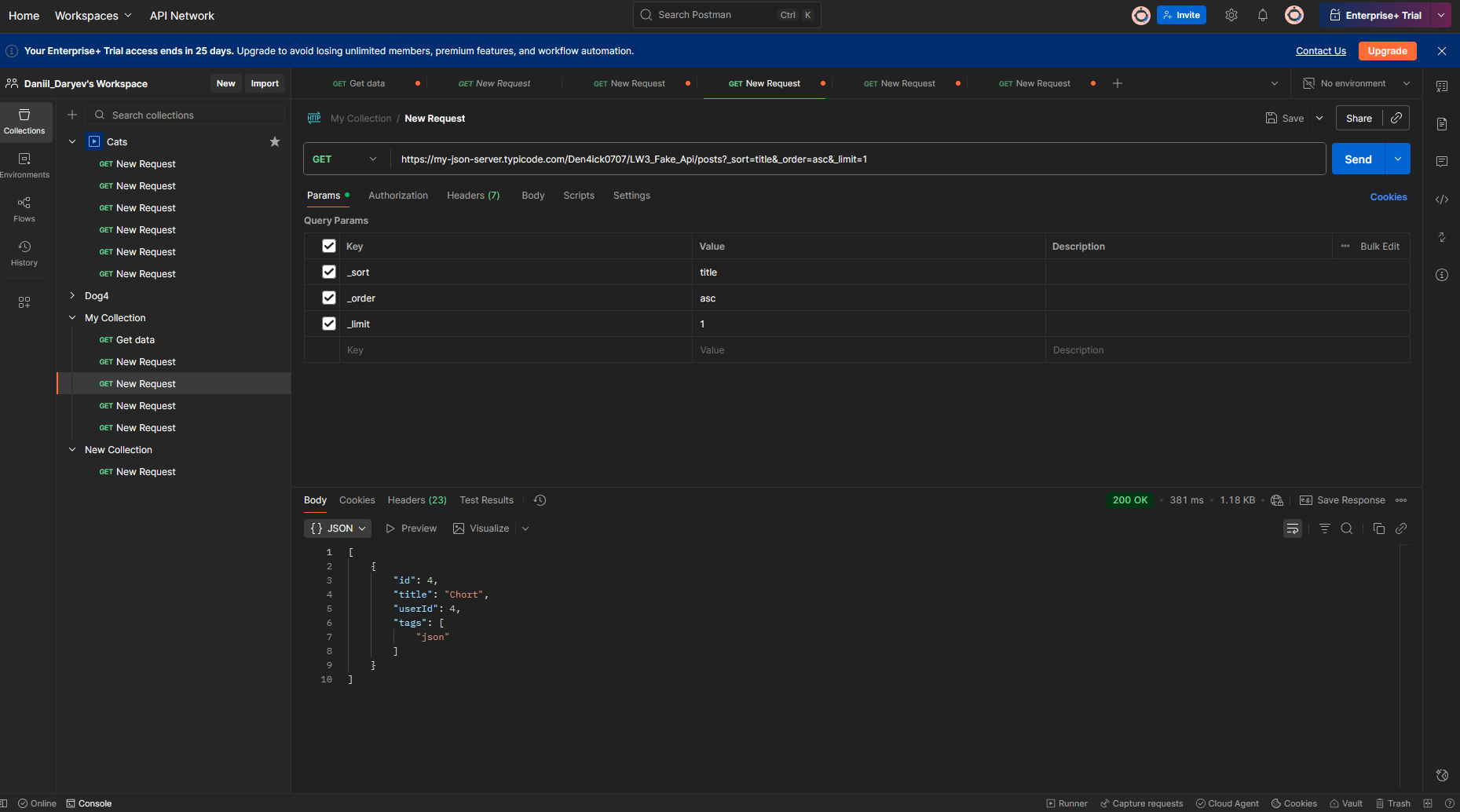


Рисунок 6 – Сортування за полем title та обмеження в один пост(Postman)

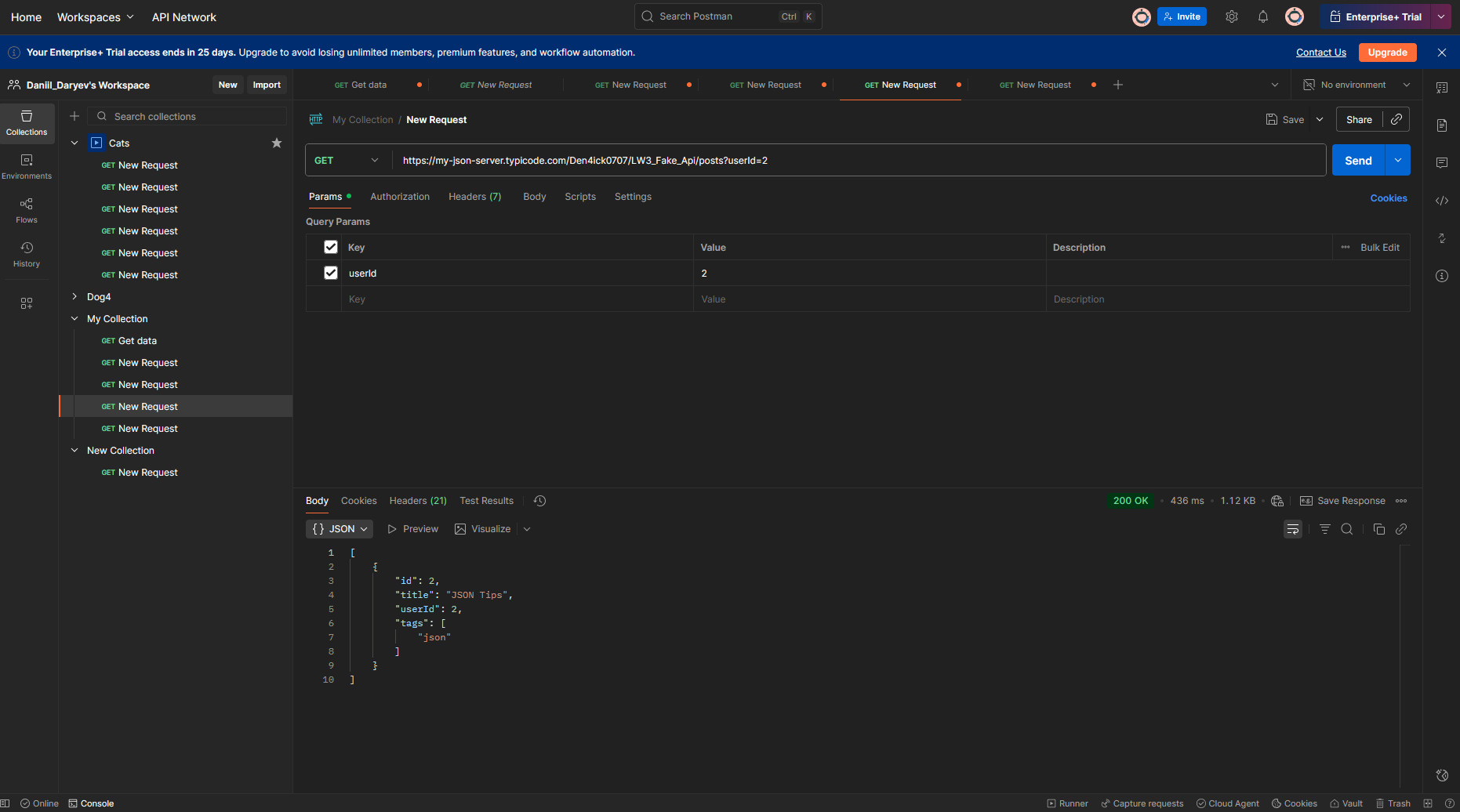


Рисунок 7 – Фільтрація за userId(Postman)

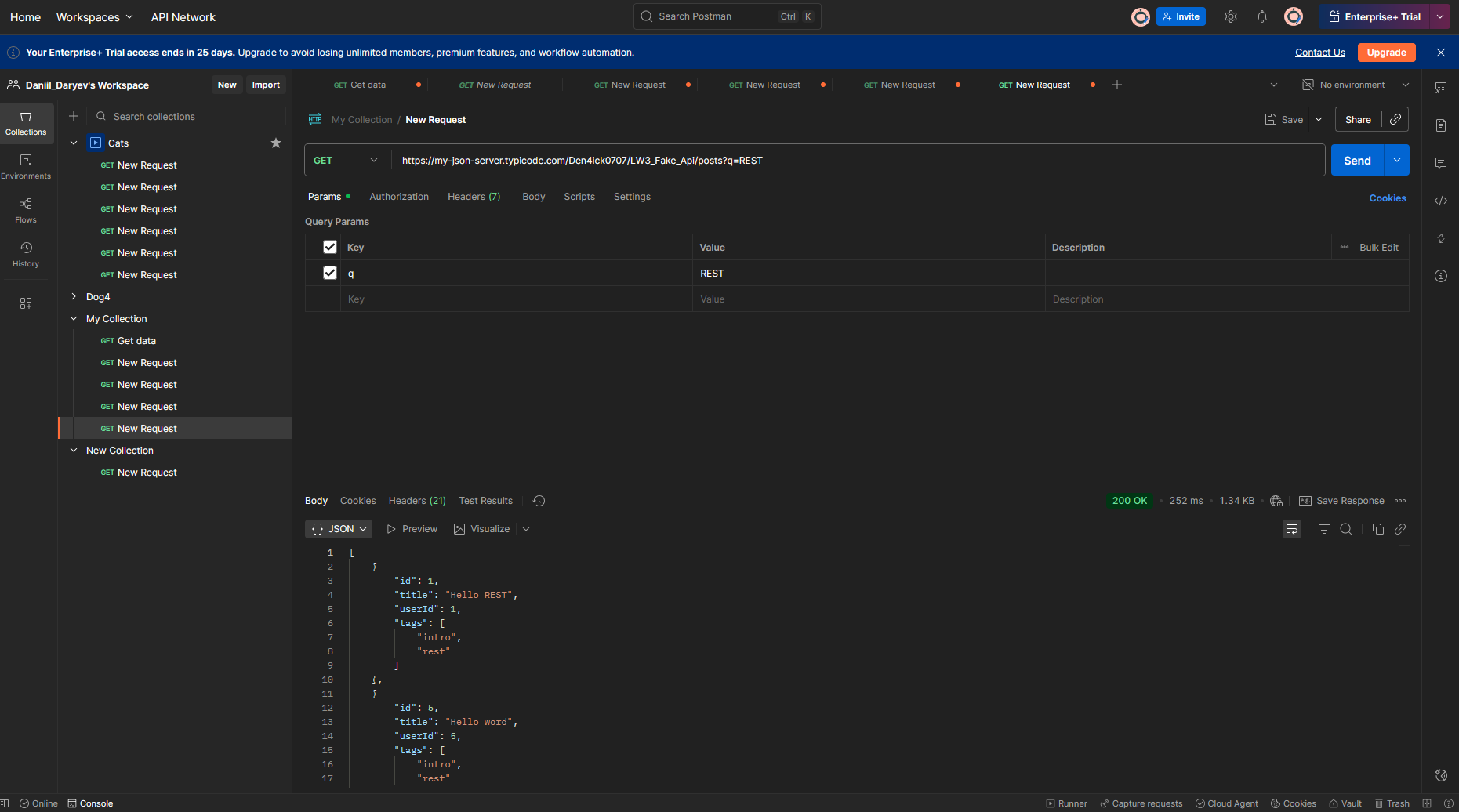


Рисунок 8 – Пошук(Postman)

# Завдання 2

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

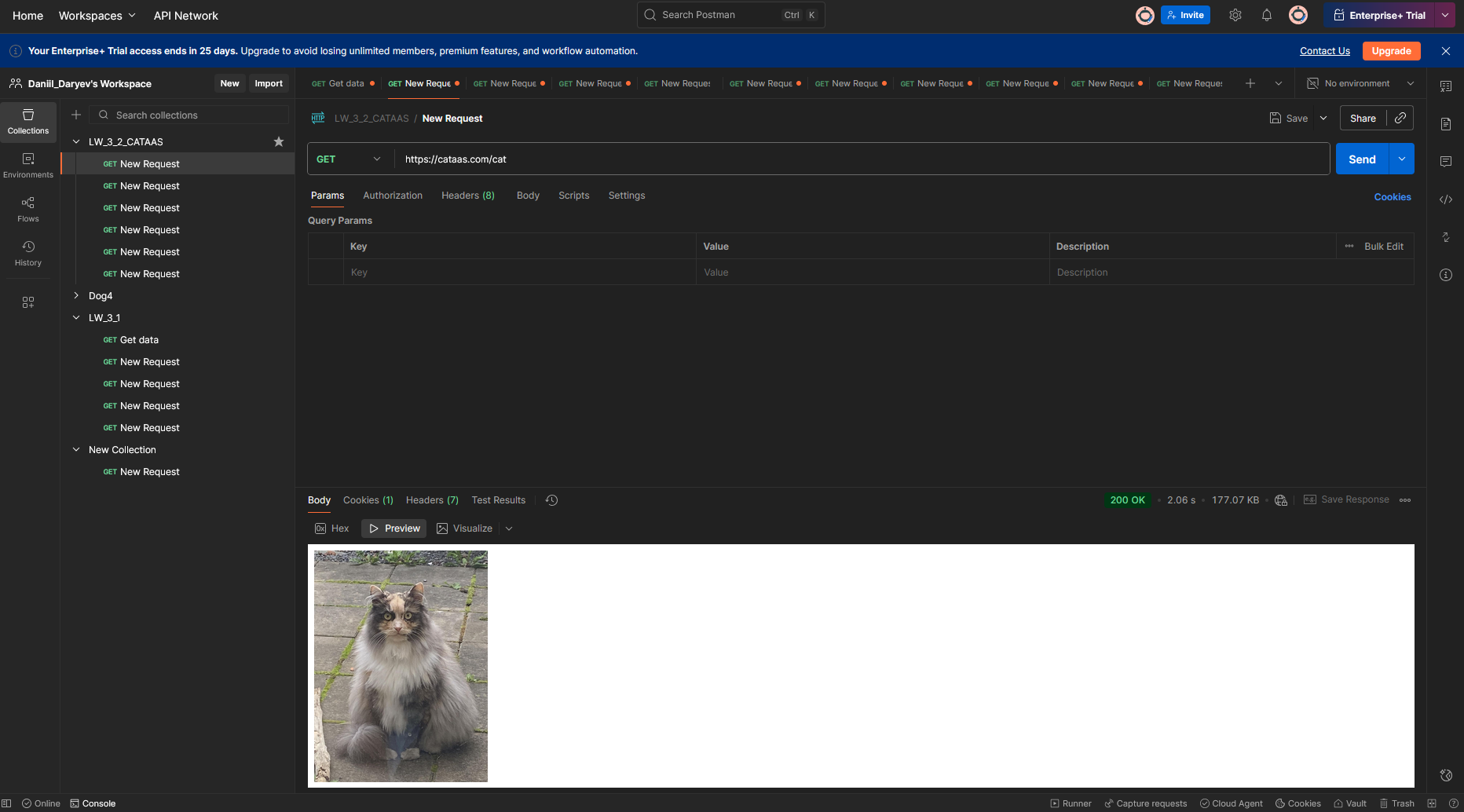


Рисунок 9 – Отримати випадкове зображення кота

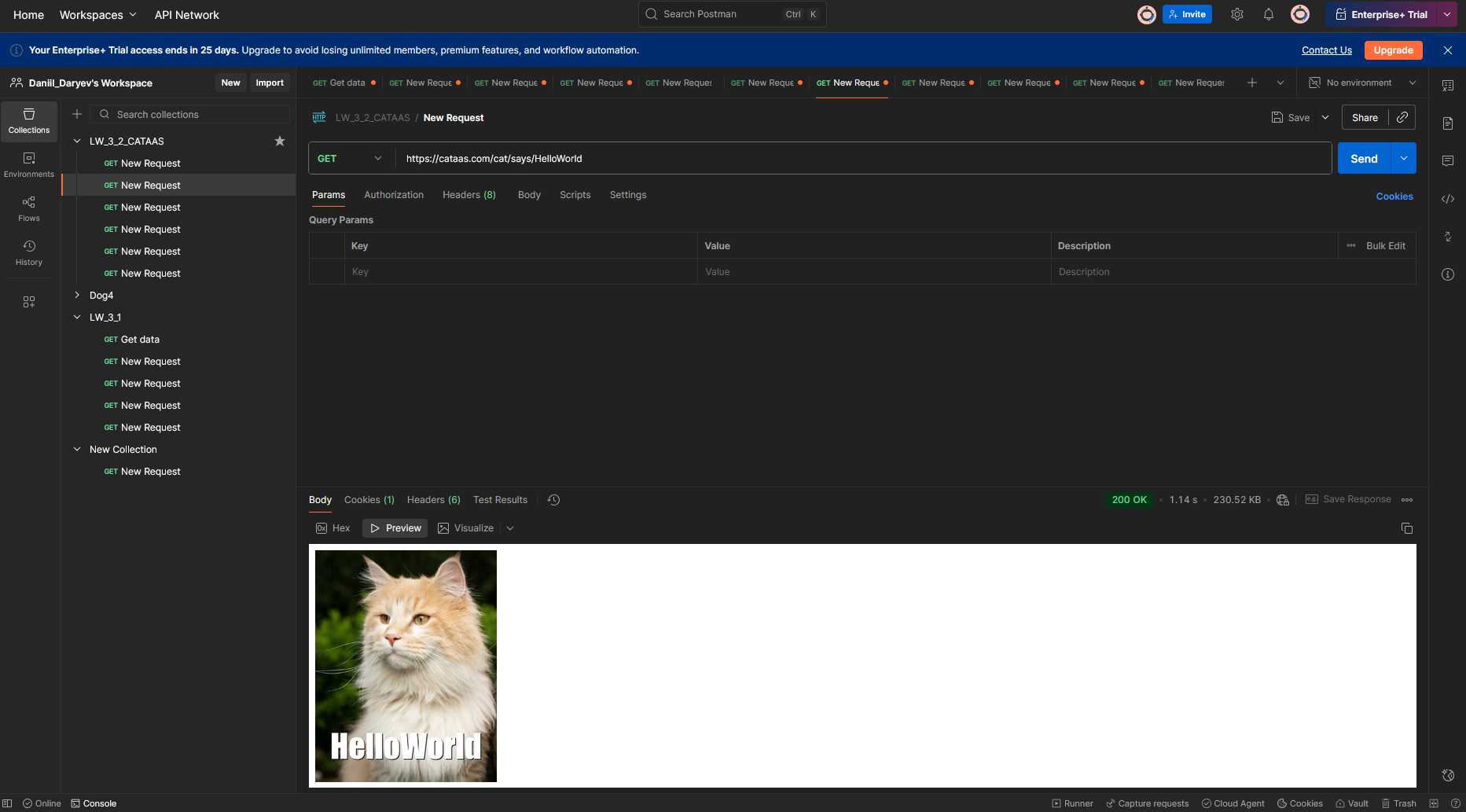


Рисунок 10 – Кіт з власним підписом

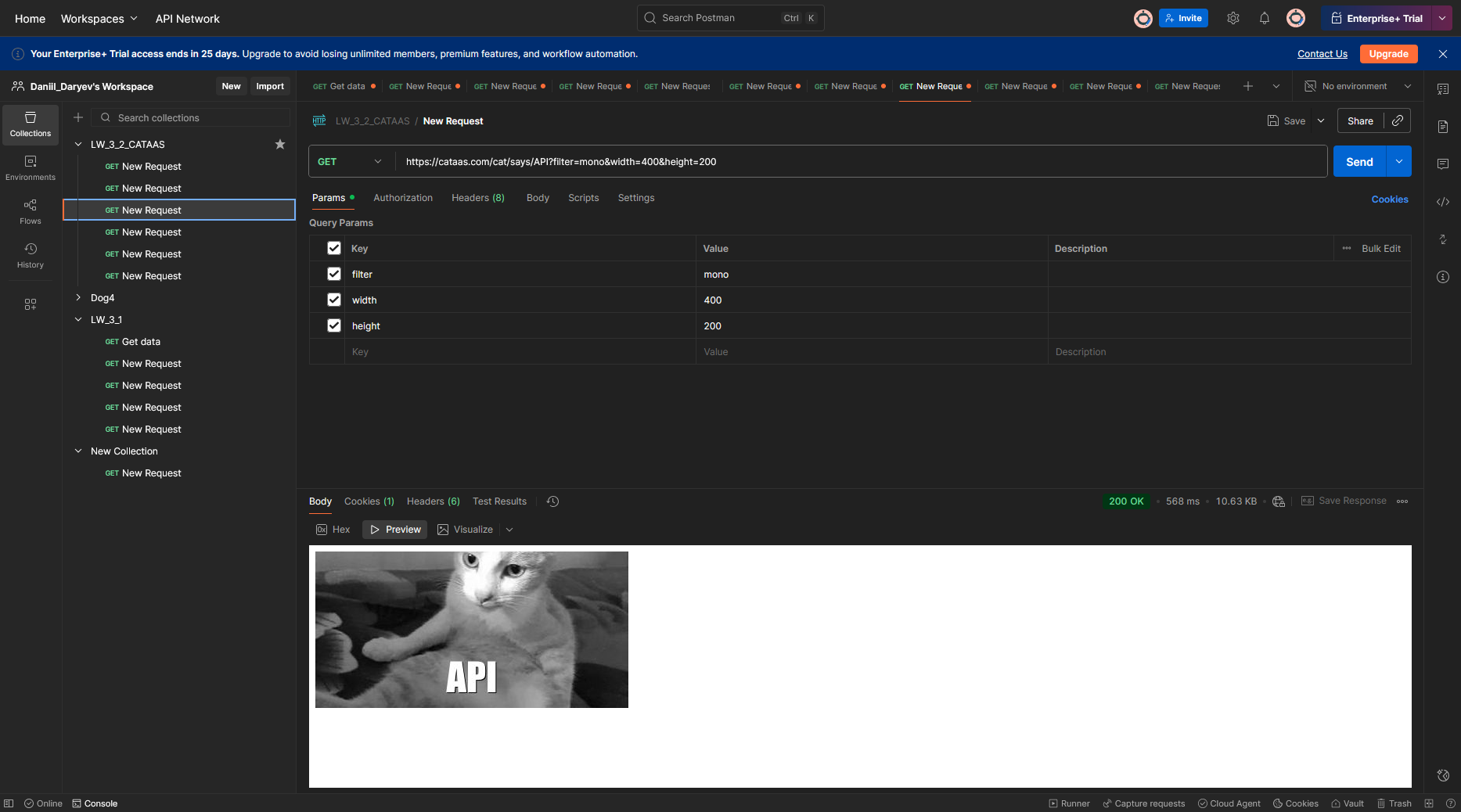


Рисунок 11 – Фільтри

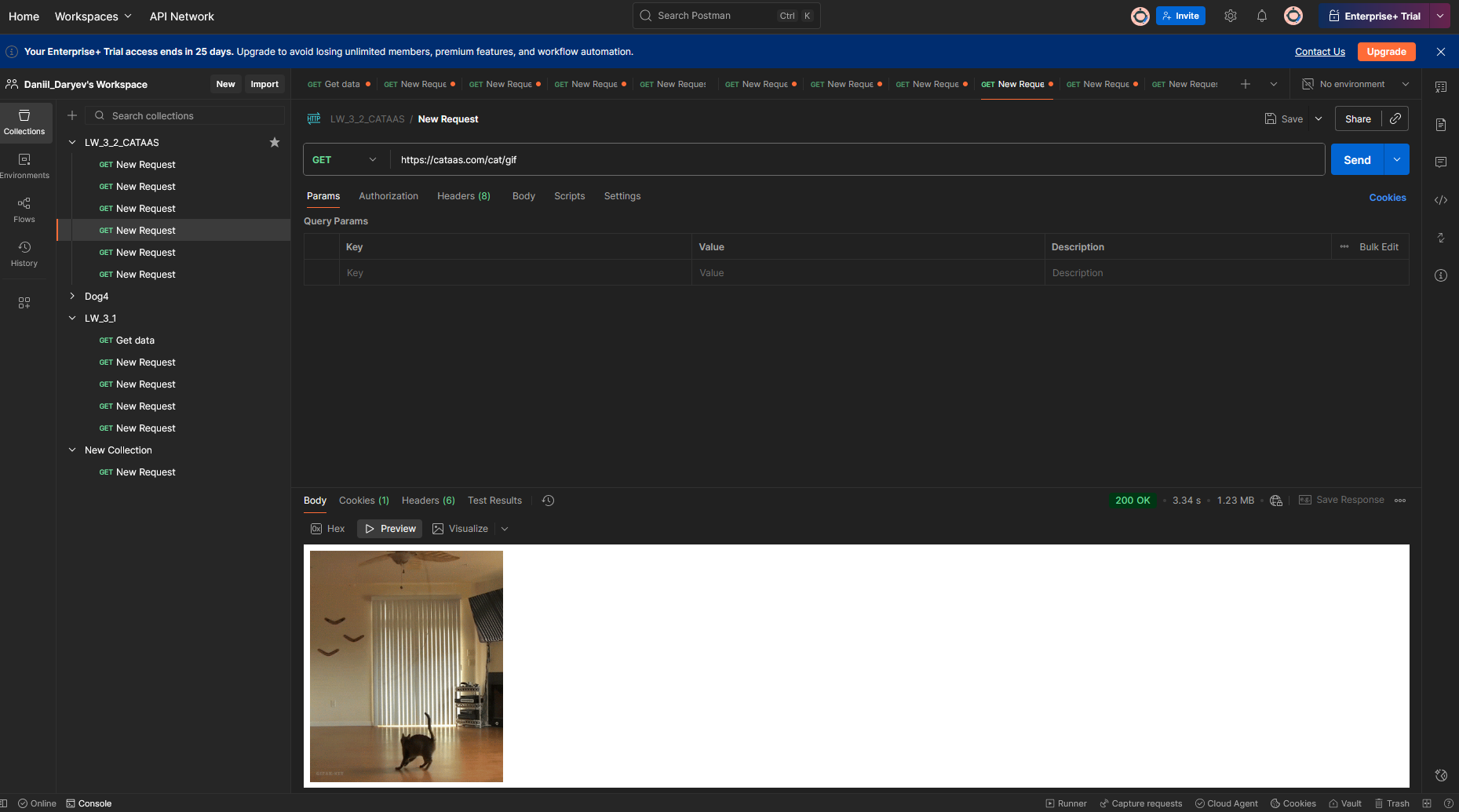


Рисунок 12 – Випадкова gif з котом

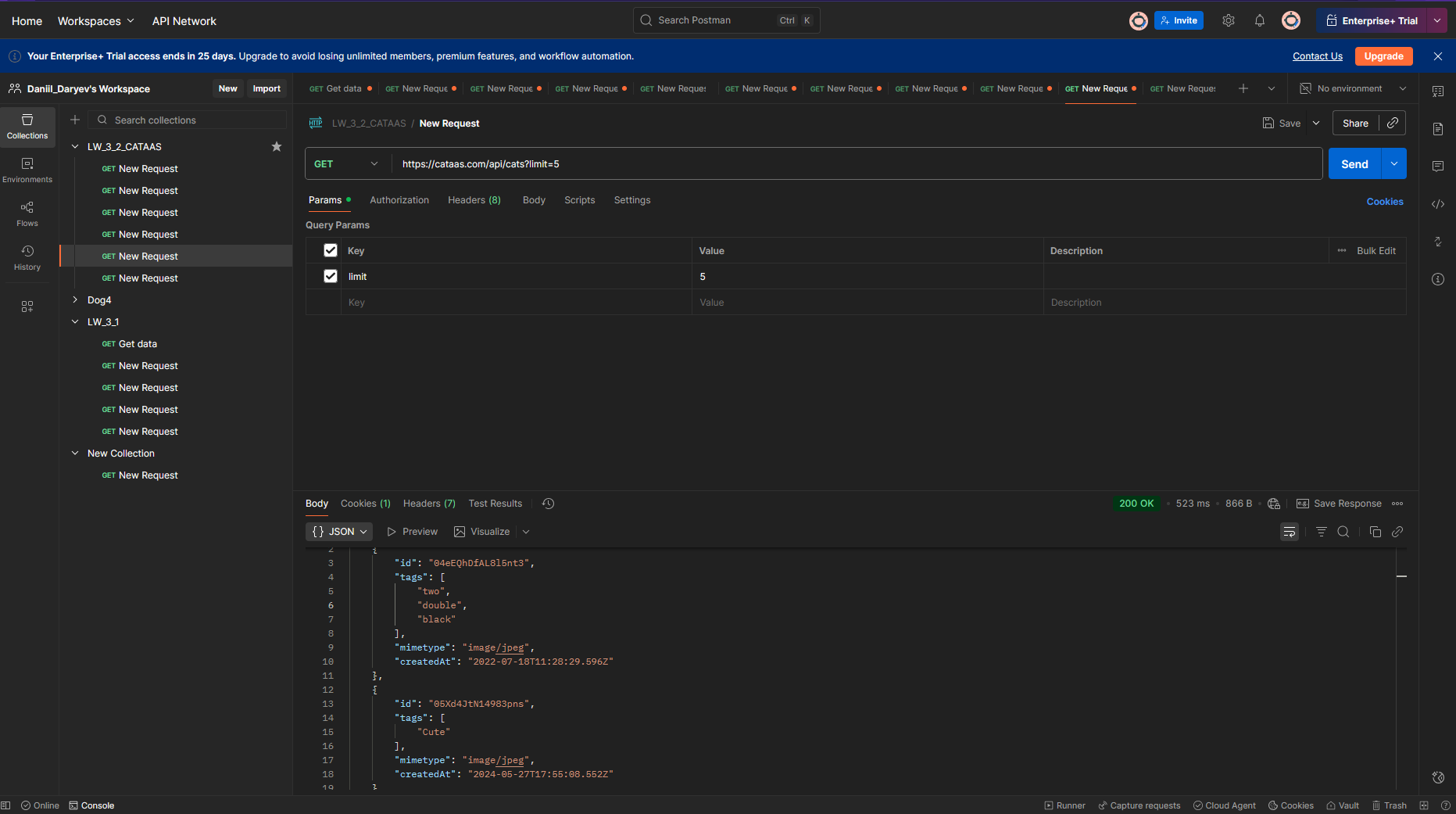


Рисунок 13 – Отримати Json перших 5 котів

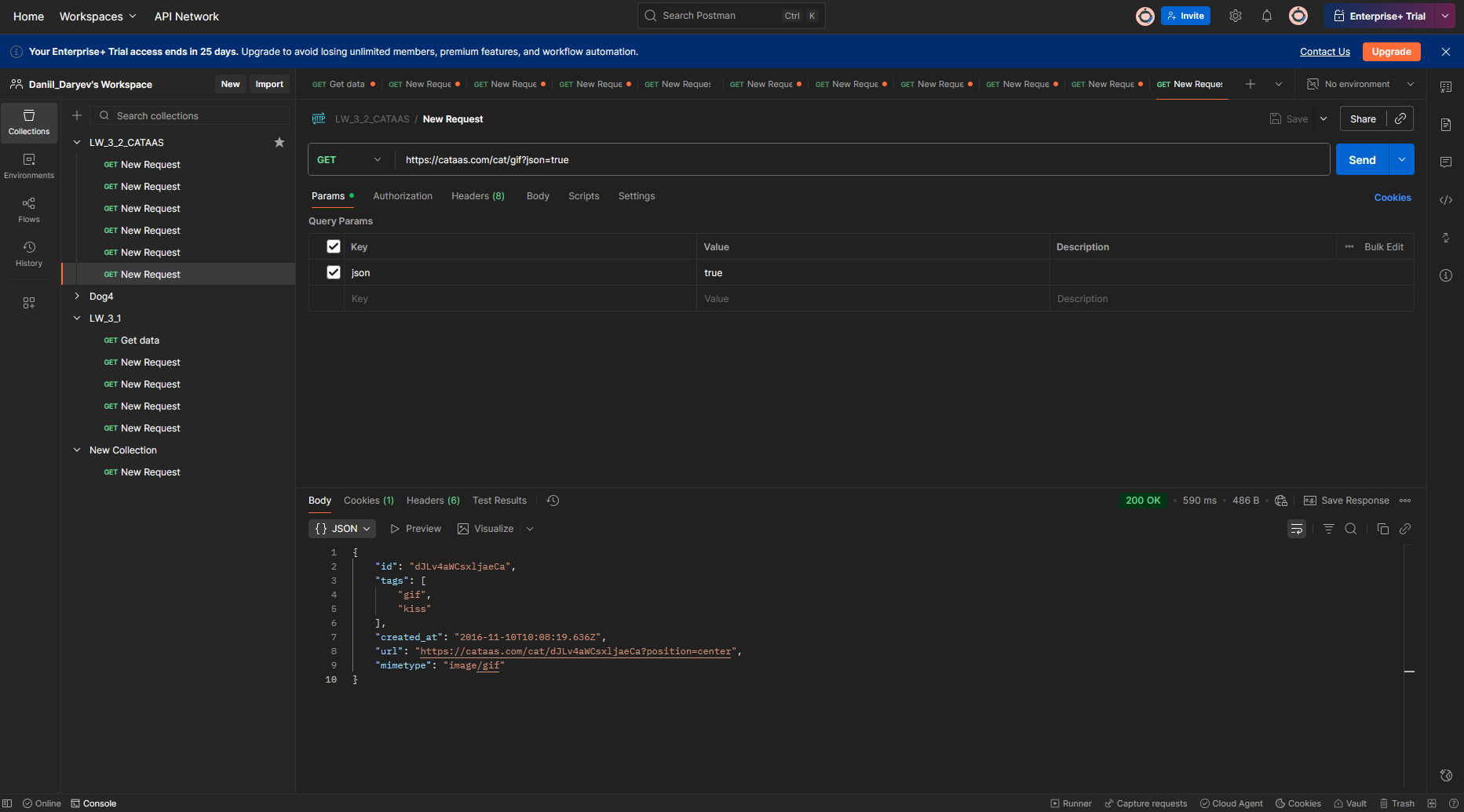


Рисунок 14 – Отримати метадані Кото-Gif у форматі Json

### **Завдання 3**

Console-клієнт до публічного REST API

**Код програми**

// See https://aka.ms/new-console-template for more information

// https://v2.jokeapi.dev/

using System;

using System.Threading.Tasks;

using System.Net.Http;

using System.Net.Http;

using System.Text.Json;

namespace LW3Daryev

{

struct Flags

{

public bool nsfw { get; set; }

public bool religious { get; set; }

public bool political { get; set; }

public bool racist { get; set; }

public bool sexist { get; set; }

public bool explicit\_ { get; set; }

}

internal class Joker

{

public bool error { get; set; }

public string category { get; set; }

public string type { get; set; }

public string setup { get; set; }

public string delivery { get; set; }

public Flags flags { get; set; }

public int id { get; set; }

public bool safe { get; set; }

public string lang { get; set; }

}

class MainProgram

{

private static readonly HttpClient \_http = new HttpClient

{

BaseAddress = new Uri("https://v2.jokeapi.dev/"),

Timeout = TimeSpan.FromSeconds(15)

};

static async Task Main(string[] args)

{

Console.WriteLine("Request to https://v2.jokeapi.dev/");

try

{

var response = await \_http.GetAsync("/joke/Programming");

response.EnsureSuccessStatusCode();

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

Console.WriteLine("Raw JSON:");

Console.WriteLine(json);

var post = JsonSerializer.Deserialize<Joker>(json, new JsonSerializerOptions

{

PropertyNameCaseInsensitive = true

});

if (post != null)

{

Console.WriteLine("\n=== Parsing result ===");

Console.WriteLine($"ID: {post.id}");

Console.WriteLine($"Category: {post.category}");

Console.WriteLine($"Type: {post.type}");

Console.WriteLine($"Setup: {post.setup}");

Console.WriteLine($"Delivery: {post.delivery}");

Console.WriteLine($"Lang: {post.lang}");

Console.WriteLine($"Flags: nsfw={post.flags.nsfw}, religious={post.flags.religious}, political={post.flags.political}, racist={post.flags.racist}, sexist={post.flags.sexist}, explicit\_={post.flags.explicit\_}");

Console.WriteLine($"Error: {post.error}");

Console.WriteLine($"Safe: {post.safe}");

}

}

catch (HttpRequestException e)

{

Console.WriteLine($"Request error: {e.Message}");

}

catch (TaskCanceledException e)

{

Console.WriteLine("Request timed out.");

}

catch (JsonException e)

{

Console.WriteLine($"JSON parsing error: {e.Message}");

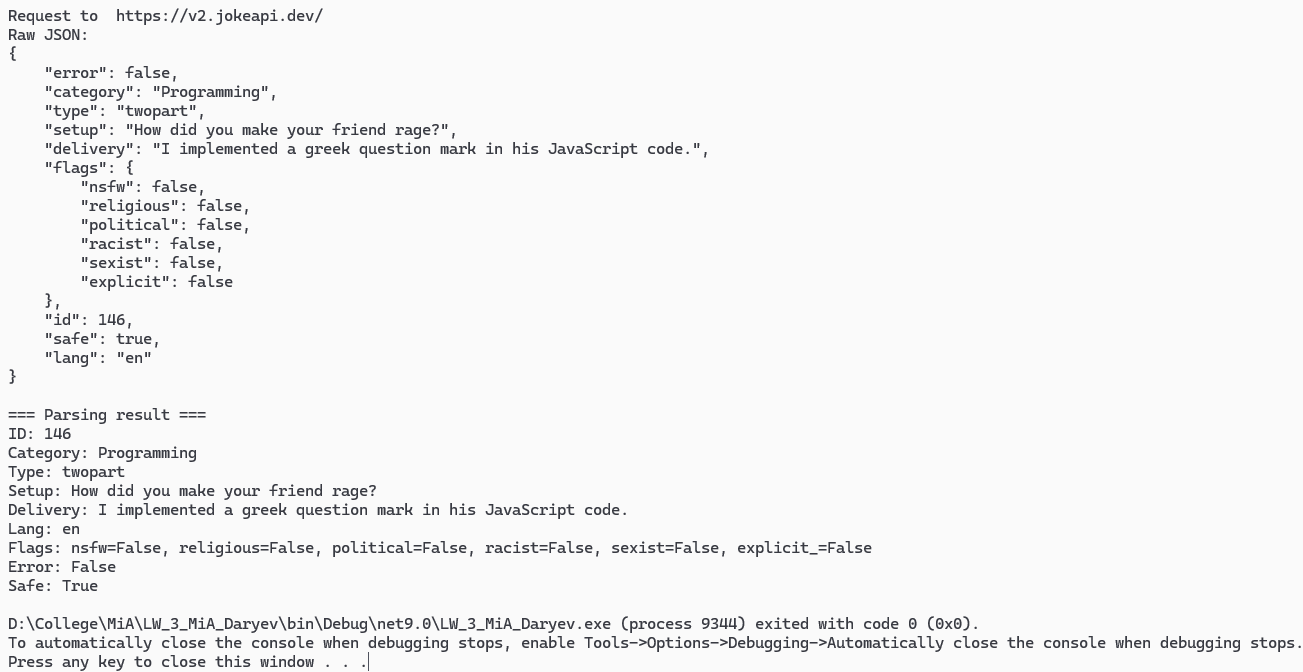
}

}

}

}

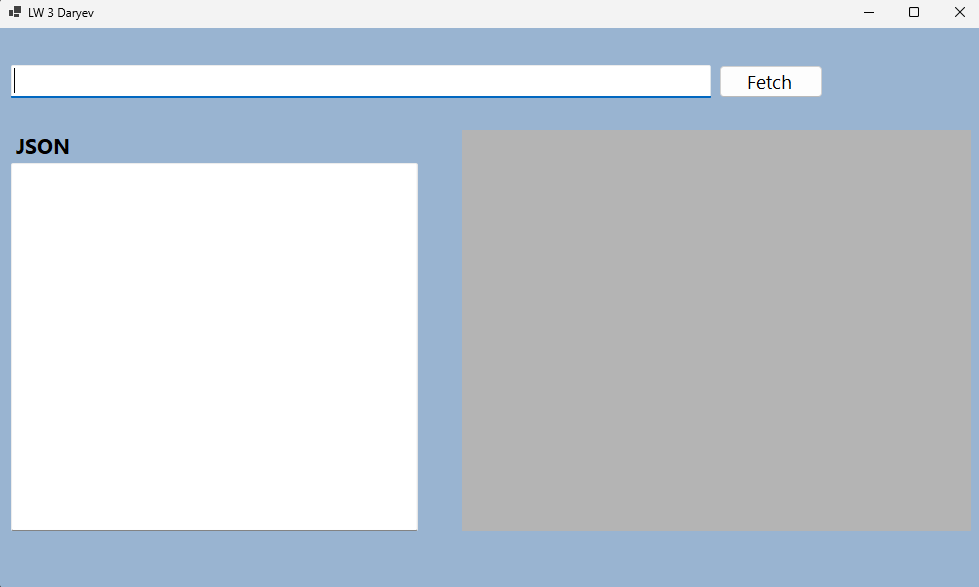
**Результат**

****

**Завдання 4.1**

Інтеграція з Dog.CEO API у WinForms

**Дизайн** **програрми**



**Код програми**

using System.Text.Json;

using System.Windows.Forms;

namespace LW\_3\_4\_Daryev1\_MiA

{

class Post

{

public string Message { get; set; }

public string Status { get; set; }

}

public partial class MainForm : Form

{

private static readonly HttpClient \_http = new HttpClient

{

BaseAddress = new Uri("https://dog.ceo/api/breeds/"),

Timeout = TimeSpan.FromSeconds(15)

};

public MainForm()

{

InitializeComponent();

}

public async Task LoadImageFromUrl(string url)

{

try

{

using var stream = await \_http.GetStreamAsync(url);

dogImage.Image = Image.FromStream(stream);

}

catch (Exception ex)

{

MessageBox.Show($"Error loading image: {ex.Message}");

}

}

private async void button1\_Click(object sender, EventArgs e)

{

try

{

var response = await \_http.GetAsync("image/random");

response.EnsureSuccessStatusCode();

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

jsonTextDeserialiser.Text = json.ToString();

var post = JsonSerializer.Deserialize<Post>(json, new JsonSerializerOptions

{

PropertyNameCaseInsensitive = true

});

if (post != null && post.Status == "success")

{

urlTB.Text = post.Message;

await LoadImageFromUrl(post.Message);

}

}

catch (HttpRequestException ex)

{

MessageBox.Show($"HTTP error: {ex.Message}");

}

catch (TaskCanceledException)

{

MessageBox.Show("Request timed out.");

}

catch (JsonException ex)

{

MessageBox.Show($"JSON parse error: {ex.Message}");

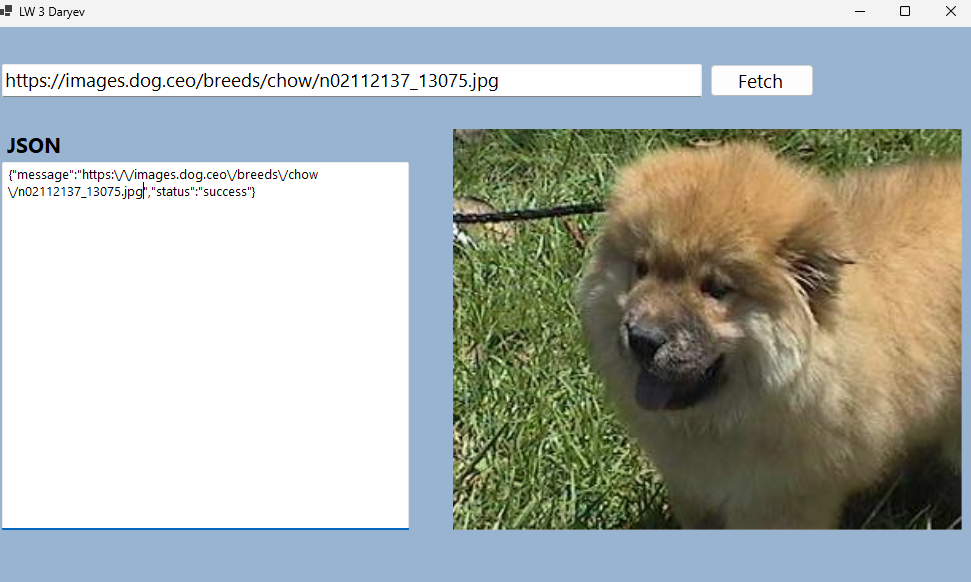
}

}

}

}

**Результат**

****

**Висновок:**

На лабораторній роботі ознайомлено з принципами клієнт-серверної архітектури та зрозуміти її роль у сучасних програмних системах та засвоєно знання створювати RESTful API для взаємодії між клієнтом і сервером а також закріплено практичні навички роботи з HTTP-запитами та відповідями ще розвинуто вміння проектувати та реалізовувати endpoints для типових CRUD-операцій.