

LW3 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.

Завдання

Завдання 1 — Створити свій Fake Online REST Server і виконати базові HTTP-запити

1. Підготовка репозиторію

- Створіть на GitHub публічний репозиторій, напр.: your-username/fake-api.
- Додайте файл db.json у корінь репозиторію з власними сутностями. Приклад:

```
{
  "users": [
    { "id": 1, "name": "Alice", "email": "alice@example.com", "role": "admin" },
    { "id": 2, "name": "Bob", "email": "bob@example.com", "role": "user" }
  ],
  "posts": [
    { "id": 1, "title": "Hello REST", "userId": 1, "tags": ["intro", "rest"] },
    { "id": 2, "title": "JSON Tips", "userId": 2, "tags": ["json"] }
  ]
}
```

- Переконайтесь, що id унікальні та цілі числа.

2. Запуск Fake Online REST

- Базова URL-адреса після публікації:

<https://my-json-server.typicode.com/<your-username>/<repo-name>>

your-username - GitHub nickname

repo-name - repository name

- Приклади ресурсів:

- GET /users
- GET /users/1
- GET /posts?userId=1
- GET /posts?q=aaaa (повнотекстовий пошук)

3. Перевірка запитів у Postman (або аналогах)

- Створіть колекцію LW3_Fake_REST.
- Додайте та виконайте мінімальний набір запитів:

GET /users — отримати список користувачів.

GET /users/1 — отримати одного користувача.

					ЛР.ОК24.ПІ231.13.03		
Змн.	Арк.	№ докум.	Підпис	Дата	Створення RESTful API		
Розроб.		Євчук М.В.					
Перевір.		Жереб Д.В.					
Н. Контр.							
Затверд.							
					Лім.	Арк.	Акрушів
						1	32
					ХПК		

GET /posts?_sort=title&_order=asc&_limit=1 — сортування/ліміт.

GET /posts?userId=2 — фільтрація за полем.

GET /posts?q=REST — пошук (працює по кількох полях).

У my-json-server записи не зберігаються після POST/PUT/PATCH/DELETE (це read-only), але ви можете відпрацювати формування тіла запитів у Postman (Body → raw → JSON) і перевірити, що сервер відповідає коректними статусами або повідомленнями (може повертати 404/405). Для реальної CRUD-практики — локально підняти json-server або використати інший мок-сервіс. Тут важливо навчитися формувати правильні запити.

- Зафіксуйте у звіті Status Code, Headers, Body. Зверніть увагу на Content-Type: application/json; charset=utf-8.

4. Приклади через curl (додатково до Postman)

1) Усі користувачі

```
curl -i https://my-json-server.typicode.com/<user>/<repo>/users
```

2) Один користувач

```
curl -i https://my-json-server.typicode.com/<user>/<repo>/users/1
```

3) Фільтр

```
curl -i "https://my-json-server.typicode.com/<user>/<repo>/posts?userId=2"
```

4) Сортування + ліміт

```
curl -i "https://my-json-server.typicode.com/<user>/<repo>/posts?_sort=title&_order=asc&_limit=1"
```

5) Пошук

```
curl -i "https://my-json-server.typicode.com/<user>/<repo>/posts?q=json"
```

Завдання 2 — Робота з API сервісу CATAAS (Cat as a Service) та SwaggerAPI

1. Перейдіть за посиланням <https://cataas.com/> та спробуйте виконати різні запити (натисніть відповідні кнопки)

Basic

Url	Description	Example
/cat	Will return a random cat	Random cat
/cat/:tag	Will return a random cat with a :tag. You can combine multiple tags with :tag separator	Random orange cute cat
/cat/gif	Will return a random gif cat \o/	Random gif cat
/cat/says/:text	Will return a random cat saying :text	Random cat saying hello
/cat/:tag/says/:text	Will return a random cat with a :tag and saying :text	Random cute cat saying hello
/cat/says/:text? fontSize=:size&fontColor=:color	Will return a random cat saying :text with text's :fontSize and text's :fontColor	Custom random cat saying hello

Advanced

Url	Description	Example
/cat?type=:type	Will return a random cat with image :type (xsmall, small, medium or square)	Random cat with type
/cat?filter=:filter	Will return a random cat with image filtered by :filter (blur, mono, negate or custom)	Random cat filtered
/cat? filter=custom&brightness=:brightness&lightness=:lightness&saturation=:saturation&hue=:hue	Will return a random cat with image filtered by :brightness, :lightness, :saturation and :hue	Random cat filtered

1. Створіть колекцію в Postman

Назвіть її: LW3_CATAAS_API.

2. Додайте та виконайте запити:

Базова адреса API: <https://cataas.com>

Основні ресурси:

- GET /cat — отримати випадкове фото кота.
- GET /cat/says/{text} — кіт із підписом.
- GET /api/cats — список котів (JSON).
- GET /cat/gif — GIF-анімація кота.

Зверніть увагу на

параметри ?filter=mono, ?type=gif, ?json=true, ?width=300&height=200 тощо.

Виконайте:

- v. GET /cat — отримати випадкове зображення кота.
- vi. GET /cat/says/HelloWorld — кіт із підписом вашого mood (картинка з текстом яка описує тебе).
- vii. GET /cat/says/API?filter=mono&width=400&height=200 — кіт із чорно-білим підписом. Спробуйте використати різні параметри запити
- viii. GET /cat/gif — отримати випадкове gif-кота.
- ix. GET /api/cats?limit=5 — отримати JSON-список перших 5 котів.
- x. GET /cat/gif?json=true — отримати метадані кото-GIF у форматі JSON.

Для кожного запити зафіксуйте у звіті:

- Status Code (200, 404, інші).
- Response Headers (зверніть увагу на Content-Type).
- Response Body (зображення або JSON).

					ЛР.ОК24.ПІ231.13.03	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		3

3. Ознайомлення з документацією. Виконання запитів через Swagger UI
Ознайомитися з матеріалом по SwaggerAPI:

Тестування API з Swagger: як перевірити функціональність API

У Swagger знайдіть розділ Endpoints. Виконайте мінімум 3 запити напряду через вебінтерфейс Swagger (наприклад: GET /cat, GET /cat/says/{text}, GET /api/cats).

Для кожного запиту зафіксуйте у звіті:

- Request URL
- Status Code
- Response Headers
- Response Body (JSON чи зображення).

Завдання 3 — Console-клієнт до публічного REST API

1. Створіть Console App (C# або будь-яку іншу мову програмування) і за допомогою бібліотеки HttpClient, надішліть GET-запит до сервісу відкритого REST API ([Додаток А](#))
2. Отримайте відповідь від сервера у форматі JSON
3. Створіть у проєкті DTO-клас із відповідними властивостями
4. Розпарсьте JSON-відповідь у C#-об'єкт за допомогою System.Text.Json.
5. Виведіть у консоль отримані дані у читабельному форматі.

Приклад виконання у [Додатк В](#).

Завдання 4 — Інтеграція з Dog.CEO API у WinForms

Документація <https://dog.ceo/dog-api/documentation/random>

4.1. Витягнути рандомне зображення собаки і відобразити у WinForms

1. Створіть WinForms-проєкт Template: *Windows Forms App* (.NET 6/7/8).
Наближений макет форми:

DISPLAY SINGLE RANDOM IMAGE FROM ALL DOGS COLLECTION

`https://dog.ceo/api/breeds/image/random` [Fetch!](#)

JSON

```
{
  "message": "https://images.dog.ceo/breeds/eskimo/n02109961_1678.jpg",
  "status": "success"
}
```

IMAGE



2. Використайте ендпойнт GET `https://dog.ceo/api/breeds/image/random` і отримайте рандомне зображення собаки і відобразіть його в PictureBox, при натисканні кнопки отримати нове зображення

4.2. Створити галерею світлин собак

1. За допомогою GET `https://dog.ceo/api/breeds/image/random/{n}` ендпойнта завантажити N зображень собак
2. Додайте перемикач кнопками < та > щоб змінювати світлину

Завдання 5 — WinForms-клієнт до публічного REST API

Створити простий клієнт на C# WinForms, який зчитує дані з відкритого REST API, відображає список ресурсів і деталі одного ресурсу, підтримує мінімальні фільтри/пошук, та коректно обробляє помилки (якщо API дозволяє). API можна обрати зі списку публічних API або власне. Заборонено копіювати чужі проекти. Якщо буде виявлено плагіат — “гайки” (нуль за роботу й інший варіант).

Що необхідно зробити:

1. Отримати колекцію (масив) JSON-об'єктів із відкритого API ([Додаток А](#)).
2. Відобразити в DataGridView (або ListView) з можливістю прокрутки.
3. Якщо можливості API дозволяють (більшість так) виконати мінімальний пошук/фільтр (по одному полю або query-параметру API).
4. При виборі рядка зі списку — виконати другий запит (наприклад, GET /resource/{id} або аналог) і показати деталі у правій панелі (Labels/TextBoxes/PictureBox).
5. Кнопка Refresh для повторного завантаження списку.
6. Відображати користувачу читабельні повідомлення при: відсутності інтернету, 4xx/5xx, таймауті, невалідному JSON.
7. Блокувати кнопки під час запиту, показувати індикатор стану (StatusStrip/Label).

Нефункціональні вимоги

- .NET 6/7/8, WinForms, C#, за бажанням інша технологія, інші технології до написання UI. It's up to you!
- Структура проекту:

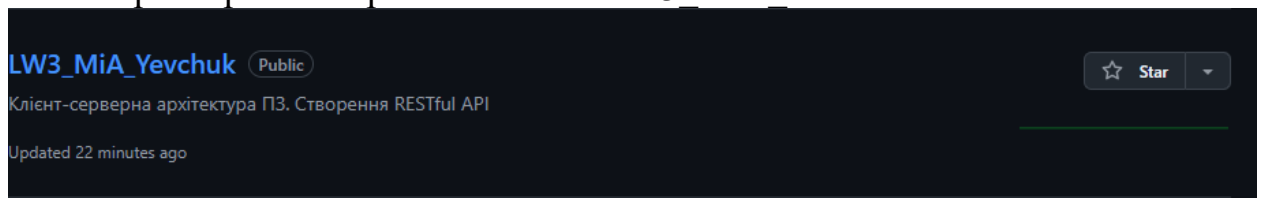
Forms/, Services/ApiClient.cs, Models/Dto.cs, Utils/.

- Код-стайл: читабельні назви, мінімальні коментарі, винесення констант (базова адреса API тощо).

Хід роботи

Завдання №1

1. Створимо репозиторій та назвемо LW3_MiA_Yevchuk



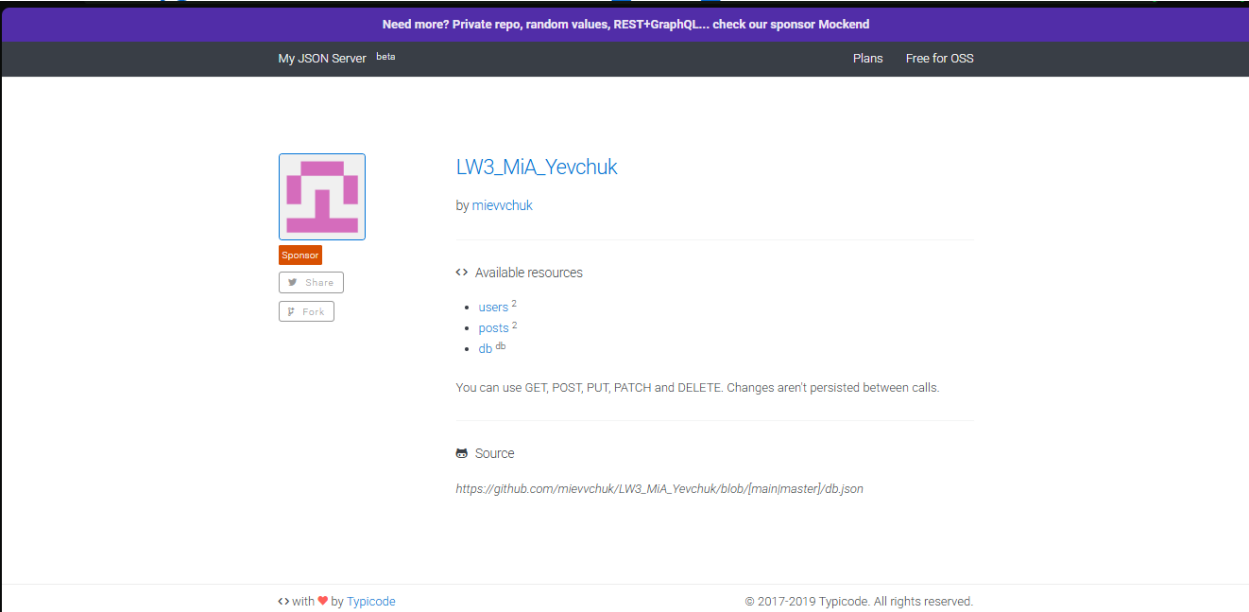
2. Додамо туди файл **db.json** у корінь репозиторію. Додаємо через **Git Bash**

					ЛР.ОК24.ПІ231.13.03	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

```
Mykha@DESKTOP-9MC4ROH MINGW64 /e/LW3 (main)
$ cat>db.json<<'EOF'
> {
>   "users":[
>   { "id":1,"name": "Alice","email":"alice@example.com","role":"admin"},
>   { "id":2,"name": "Bob","email":"bob@example.com","role":"user"}
> ],
>   "posts":[
>   { "id":1,"title":"Hello REST","userId":1,"tags":["intro","rest"]},
>   { "id":2,"title":"JSON Tips","userId":2,"tags":["json"]}
> ]
> }
> EOF
```

3. Запускаємо **Fake Online REST**

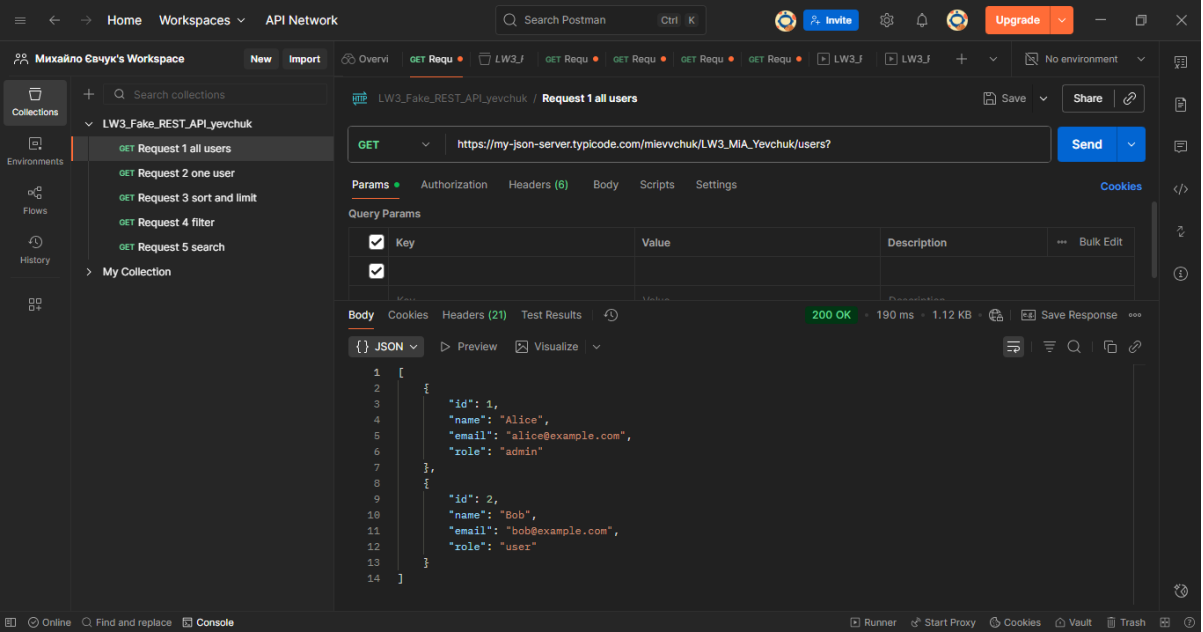
Базова URL-адреса після публікації: https://my-json-server.typicode.com/mievvchuk/LW3_MiA_Yevchuk

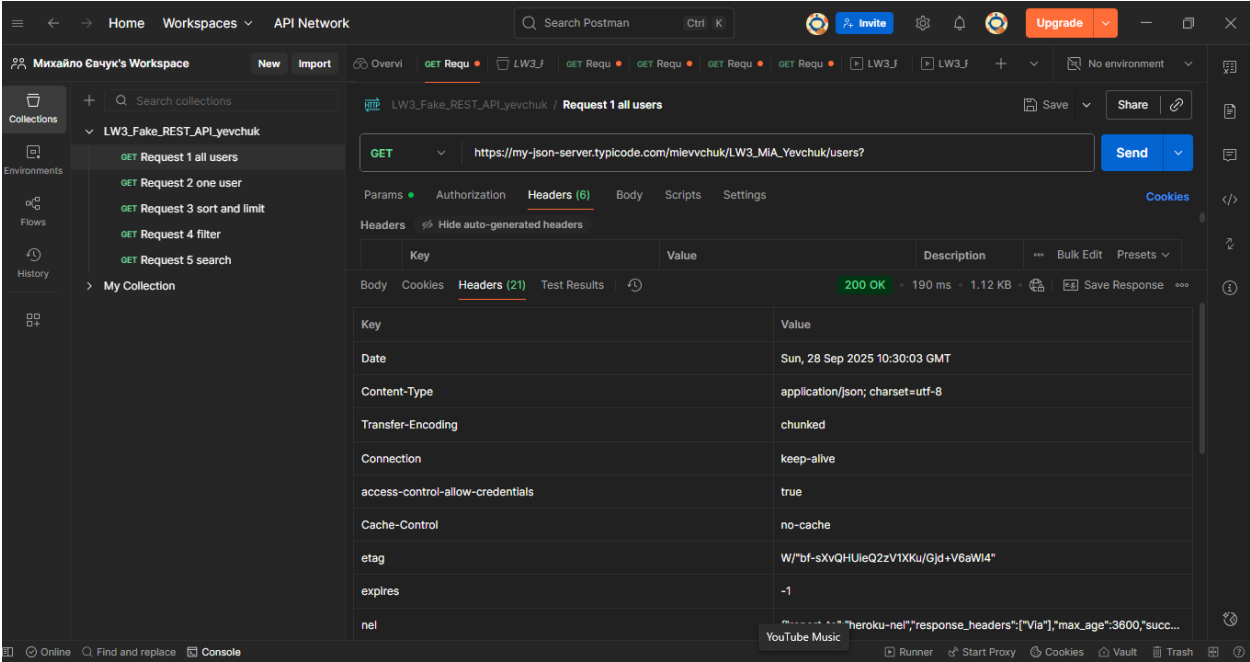


4. Перевірка запитів через PostMan

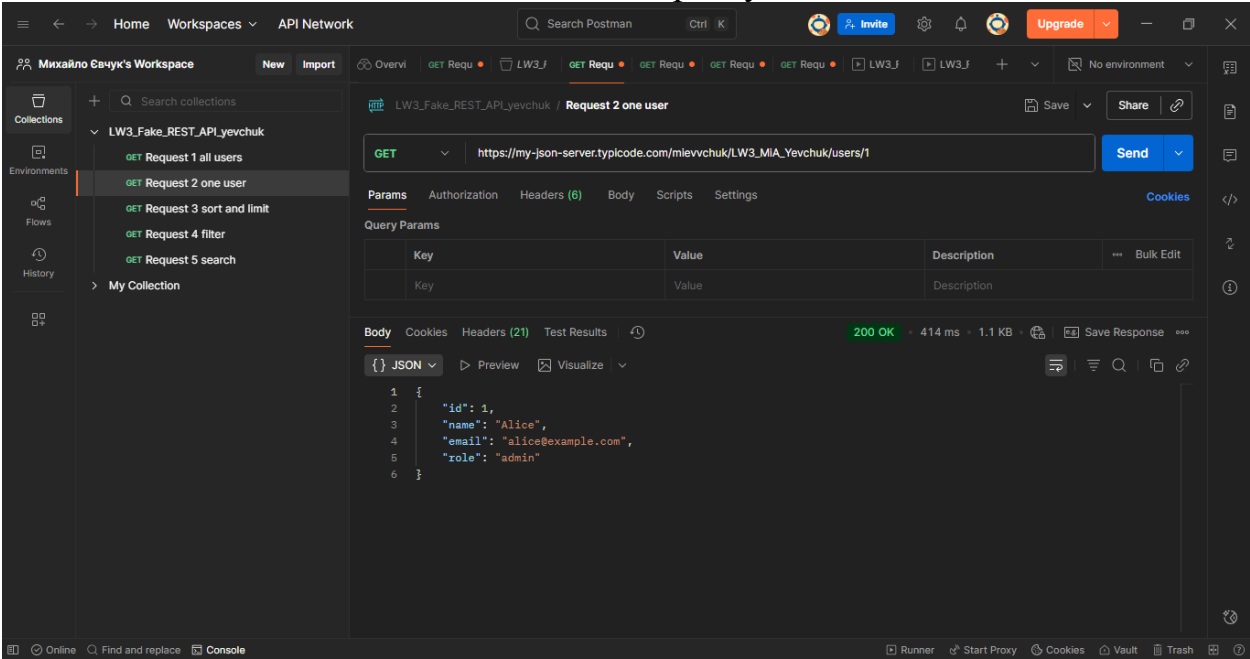
Створюємо колекцію і називаємо її “**LW3_Fake_REST_API_yevchuk**”

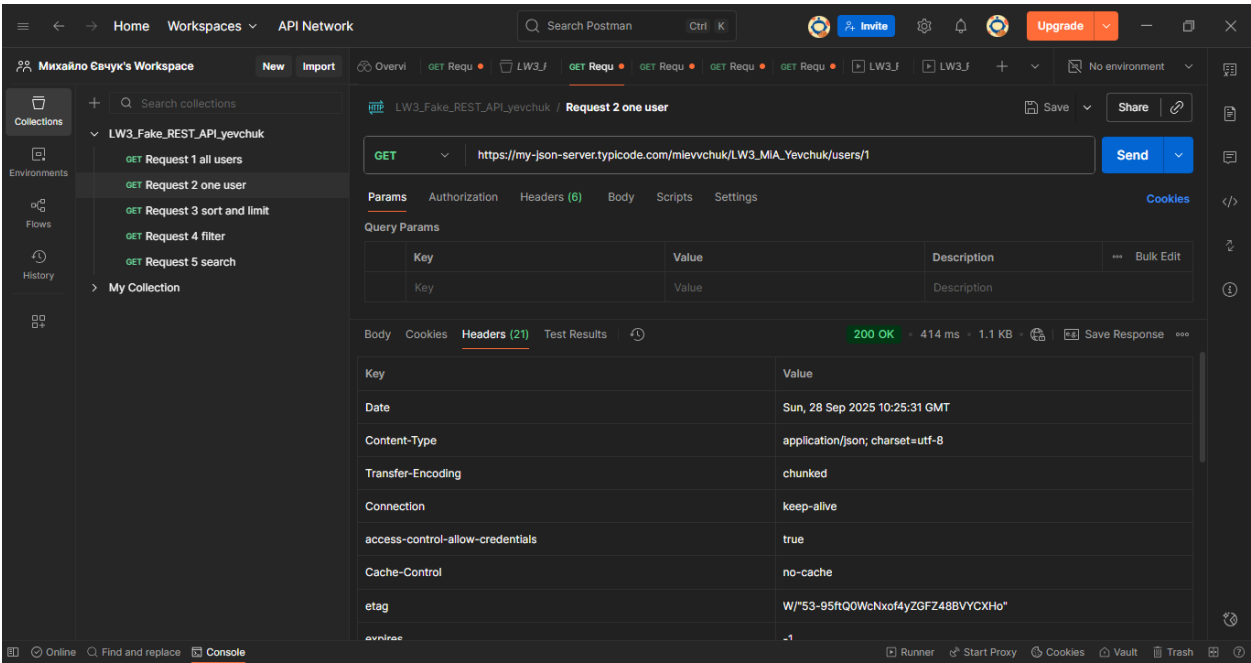
1. Виконаємо запит на усіх користувачів



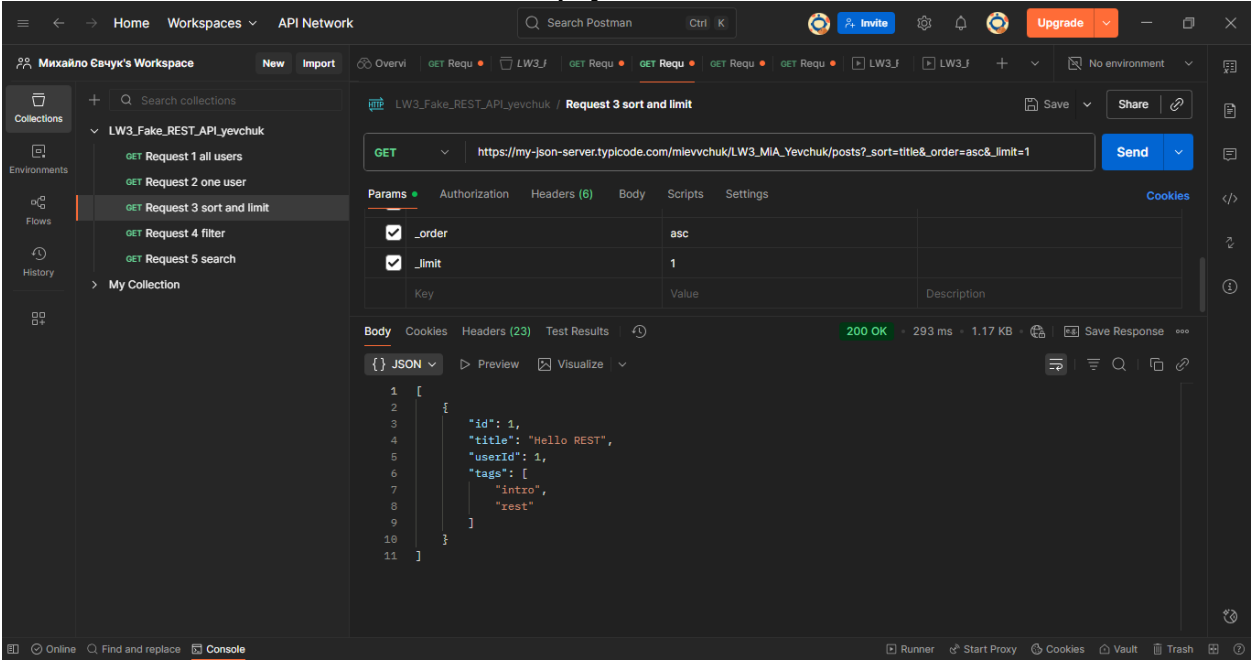


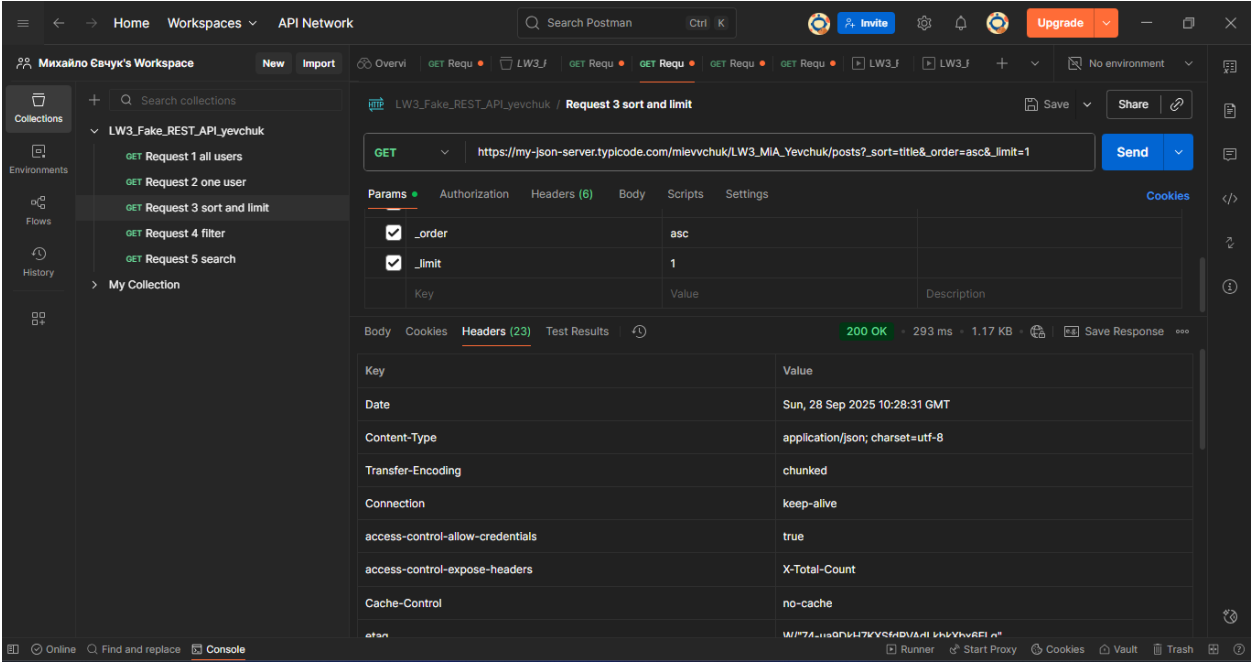
2. Виконаємо запит на одного користувача



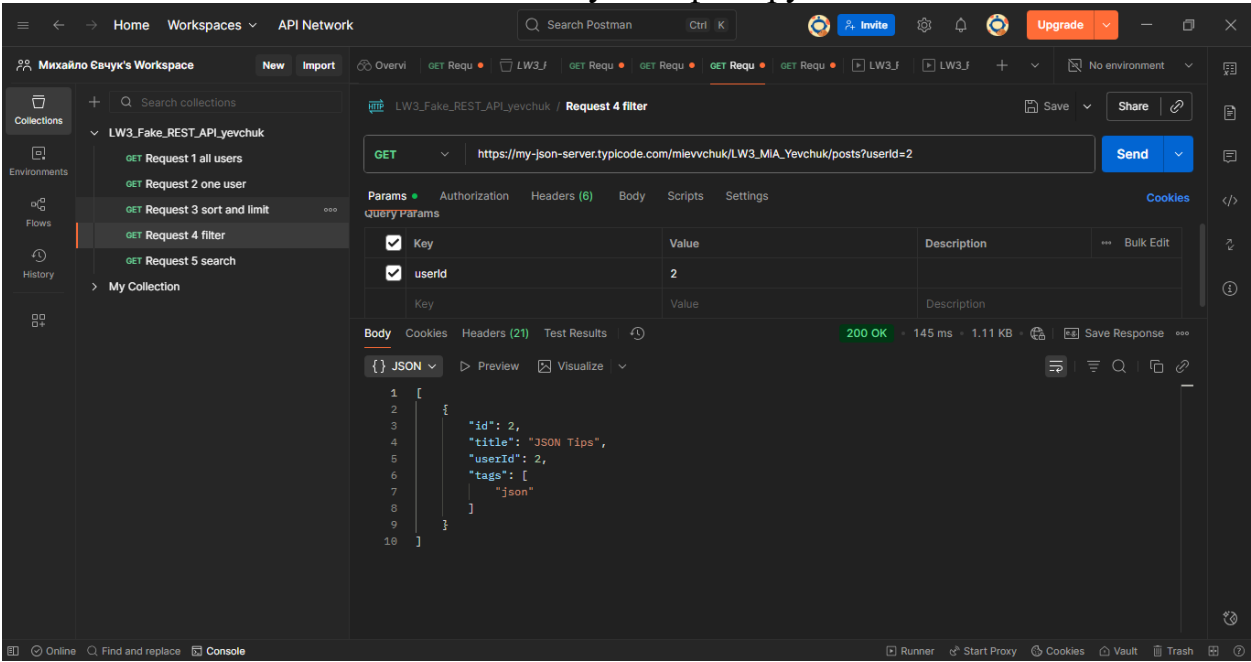


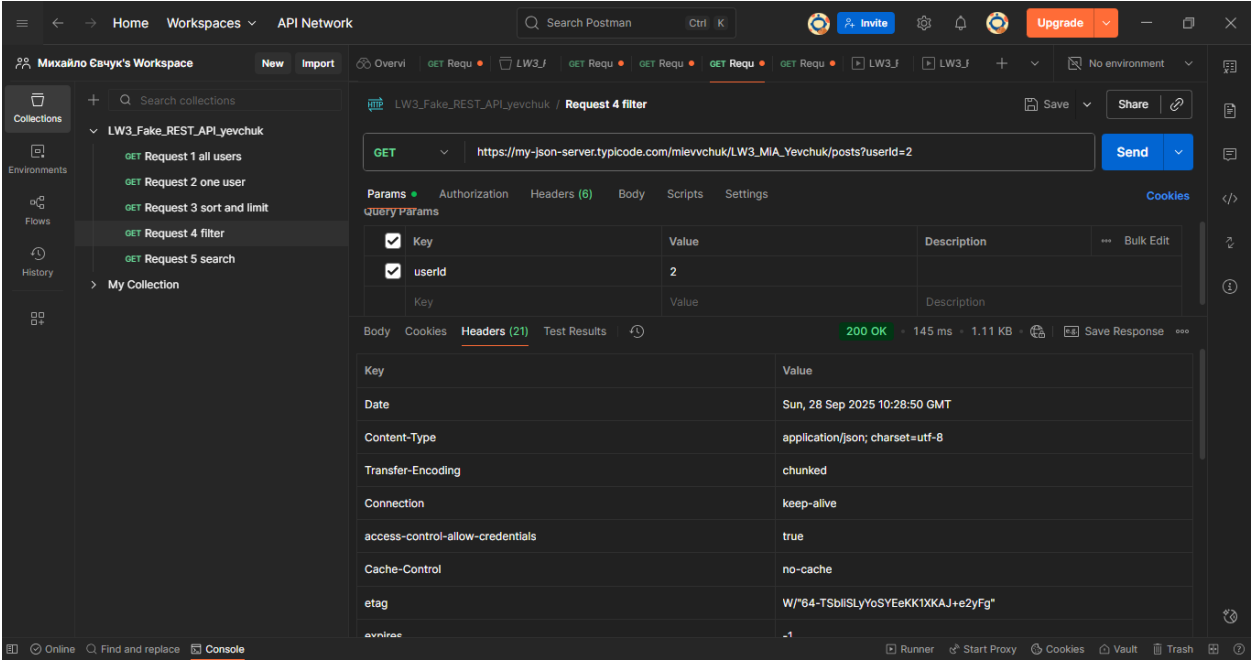
3. Виконаємо запит на сортування/ліміт



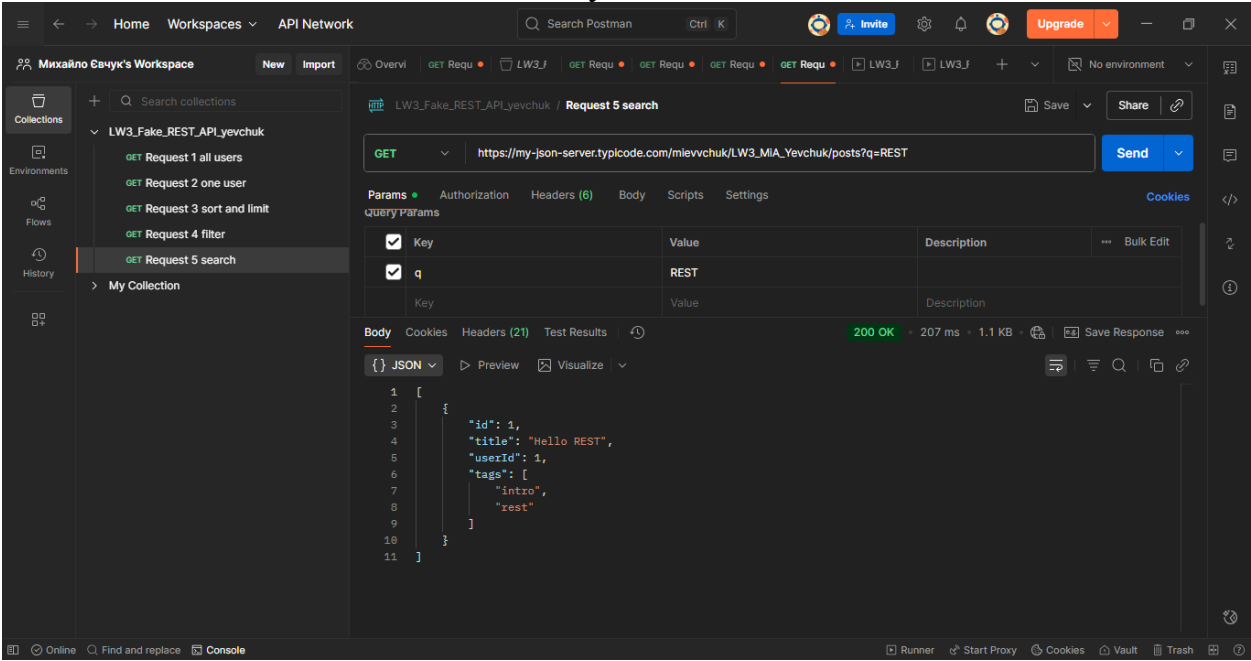


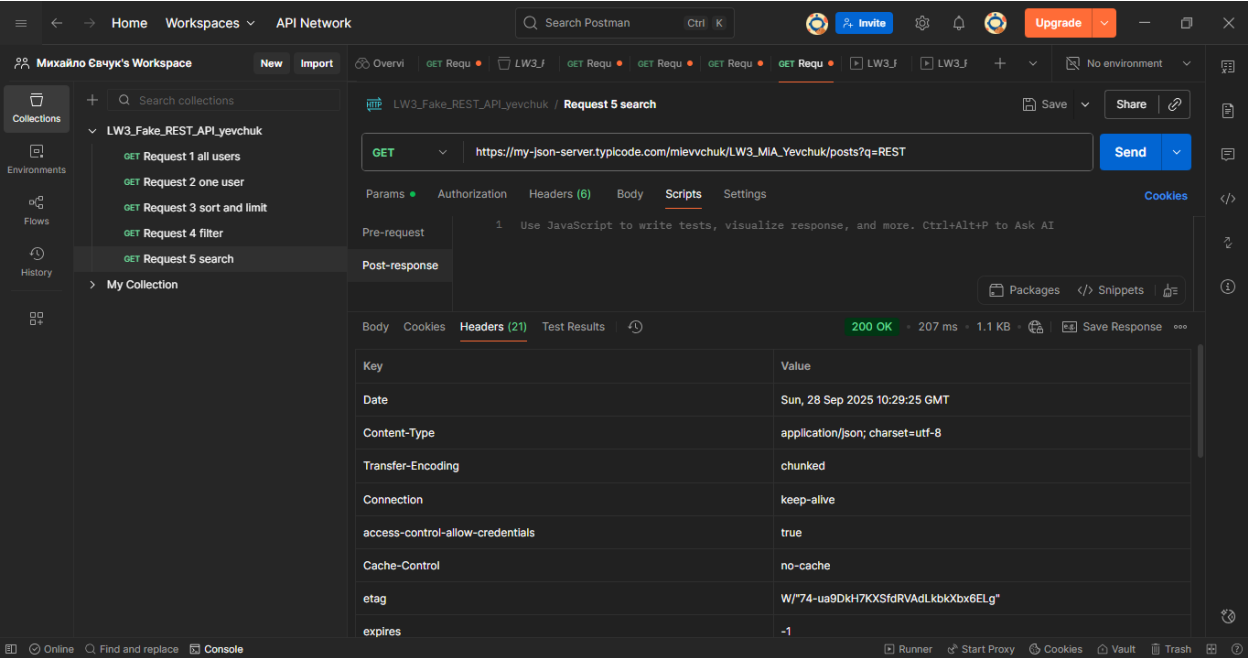
4. Виконаємо запит на пошук по фільтру





5. Виконаємо запит на пошук





5. Приклади через curl -i:

а. Усі користувачі

```
C:\Users\Mykha>curl -i https://my-json-server.typicode.com/mievchuk/LW3_MIA_Yevchuk/users
HTTP/1.1 200 OK
Date: Sun, 28 Sep 2025 10:31:06 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 191
Connection: keep-alive
access-control-allow-credentials: true
Cache-Control: no-cache
etag: W/"b4f-c5vqHJiaQ22v1X8u/GjdV6ahU4"
expires: -1
nel: {"report-to":"heroku-nel","response_headers":["Via"],"max_age":3600,"success_fraction":0.01,"failure_fraction":0.1}
pragma: no-cache
report-to: {"group":"heroku-nel","endpoints":[{"url":"https://nel.heroku.com/reports?s=48x9RqecPXCv4d9%2B1GzrjvP8ivsGf%2B9rTdrLnJg2pc%3D%u0026id=929419e7-33ea-4e2f-85f8-7d8b7cd5cbd6%u0026ts=1759055466"}],"max_age":3600}
Server: cloudflare
reporting-endpoints: heroku-nel="https://nel.heroku.com/reports?s=48x9RqecPXCv4d9%2B1GzrjvP8ivsGf%2B9rTdrLnJg2pc%3D%u0026id=929419e7-33ea-4e2f-85f8-7d8b7cd5cbd6%u0026ts=1759055466"
vary: Origin, Accept-Encoding
via: 2.0 heroku-router
x-content-type-options: nosniff
x-powered-by: Express
cf-cache-status: DYNAMIC
CF-RAY: 986291b62b553db6-WAW
alt-svc: h3=":443", ma=86400

[
  {
    "id": 1,
    "name": "Alice",
    "email": "alice@example.com",
    "role": "admin"
  },
  {
    "id": 2,
    "name": "Bob",
    "email": "bob@example.com",
    "role": "user"
  }
]
```

б. Один користувач

```
C:\Users\Mykha>curl -i https://my-json-server.typicode.com/mievchuk/LW3_MIA_Yevchuk/users/1
HTTP/1.1 200 OK
Date: Sun, 28 Sep 2025 10:32:10 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 83
Connection: keep-alive
access-control-allow-credentials: true
Cache-Control: no-cache
etag: W/"53-95r1Q0WdnXoHyZGF248BVCOho"
expires: -1
nel: {"report-to":"heroku-nel","response_headers":["Via"],"max_age":3600,"success_fraction":0.01,"failure_fraction":0.1}
pragma: no-cache
report-to: {"group":"heroku-nel","endpoints":[{"url":"https://nel.heroku.com/reports?s=xL3YBxBRVUpZuMIFslz%2FA%2Fs2x%2FwdtrmhPEeErrm8U3D%u0026id=929419e7-33ea-4e2f-85f8-7d8b7cd5cbd6%u0026ts=1759055530"}],"max_age":3600}
reporting-endpoints: heroku-nel="https://nel.heroku.com/reports?s=xL3YBxBRVUpZuMIFslz%2FA%2Fs2x%2FwdtrmhPEeErrm8U3D%u0026id=929419e7-33ea-4e2f-85f8-7d8b7cd5cbd6%u0026ts=1759055530"
Server: cloudflare
vary: Origin, Accept-Encoding
via: 2.0 heroku-router
x-content-type-options: nosniff
x-powered-by: Express
cf-cache-status: DYNAMIC
CF-RAY: 98629347399fbde-WAW
alt-svc: h3=":443", ma=86400

{
  "id": 1,
  "name": "Alice",
  "email": "alice@example.com",
  "role": "admin"
}
```

с. Фільтрація

```

C:\Users\Mykhailo>curl -i "https://my-json-server.typicode.com/mievvchuk/LW3_MIA_Yevchuk/posts?userId=2"
HTTP/1.1 200 OK
Date: Sun, 28 Sep 2025 10:32:39 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 189
Connection: keep-alive
access-control-allow-credentials: true
Cache-Control: no-cache
etag: W/"64-T5b1S1y0SvE0K1XAJ+e2yFg"
expires: -1
nel: {"report-to":"heroku-nel","response_headers":["Via"],"max_age":3600,"success_fraction":0.01,"failure_fraction":0.1}
pragma: no-cache
report-to: {"group":"heroku-nel","endpoints":[{"url":"https://nel.heroku.com/reports?s=6a80vZFLHF2ECy8Iz9ACZCD8W3F2LS21zL6Xp8dME30\u0026id=929419e7-33ea-4e2f-85f0-7d8b7cd5cbdb6\u0026ts=1759055559"}],"max_age":3600}
reporting-endpoints: heroku-nel="https://nel.heroku.com/reports?s=6a80vZFLHF2ECy8Iz9ACZCD8W3F2LS21zL6Xp8dME30\u0026id=929419e7-33ea-4e2f-85f0-7d8b7cd5cbdb6\u0026ts=1759055559"
Server: cloudflare
vary: Origin, Accept-Encoding
via: 2.0 heroku-router
x-content-type-options: nosniff
x-powered-by: Express
cf-cache-status: DYNAMIC
CF-RAY: 98629f56da5b6258-MAM
alt-svc: h3=":443"; ma=86400

[
  {
    "id": 2,
    "title": "JSON Tips",
    "userId": 2,
    "tags": [
      "json"
    ]
  }
]

```

d. Сортунання + ліміт

```

C:\Users\Mykhailo>curl -i "https://my-json-server.typicode.com/mievvchuk/LW3_MIA_Yevchuk/posts?sort=title&order=asc&limit=1"
HTTP/1.1 200 OK
Date: Sun, 28 Sep 2025 10:32:53 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 116
Connection: keep-alive
access-control-allow-credentials: true
access-control-expose-headers: X-Total-Count
Cache-Control: no-cache
etag: W/"74-uaS0dF7KX5fRAdL4k0xb6Elg"
expires: -1
nel: {"report-to":"heroku-nel","response_headers":["Via"],"max_age":3600,"success_fraction":0.01,"failure_fraction":0.1}
pragma: no-cache
report-to: {"group":"heroku-nel","endpoints":[{"url":"https://nel.heroku.com/reports?s=k2B90DhI6k2F2FdfJUBpnuRk2F4t883ziIje9TstgIam154s3D\u0026id=929419e7-33ea-4e2f-85f0-7d8b7cd5cbdb6\u0026ts=1759055573"}],"max_age":3600}
reporting-endpoints: heroku-nel="https://nel.heroku.com/reports?s=k2B90DhI6k2F2FdfJUBpnuRk2F4t883ziIje9TstgIam154s3D\u0026id=929419e7-33ea-4e2f-85f0-7d8b7cd5cbdb6\u0026ts=1759055573"
Server: cloudflare
vary: Origin, Accept-Encoding
via: 2.0 heroku-router
x-content-type-options: nosniff
x-powered-by: Express
x-total-count: 2
cf-cache-status: DYNAMIC
CF-RAY: 98629456db434a3-MAM
alt-svc: h3=":443"; ma=86400

[
  {
    "id": 1,
    "title": "Hello REST",
    "userId": 1,
    "tags": [
      "intro",
      "rest"
    ]
  }
]

```

е. Пошук

```

C:\Users\Mykhailo>curl -i "https://my-json-server.typicode.com/mievvchuk/LW3_MIA_Yevchuk/posts?q=json"
HTTP/1.1 200 OK
Date: Sun, 28 Sep 2025 10:33:47 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 100
Connection: keep-alive
access-control-allow-credentials: true
Cache-Control: no-cache
etag: W/"64-T5b1S1y0SvE0K1XAJ+e2yFg"
expires: -1
nel: {"report-to":"heroku-nel","response_headers":["Via"],"max_age":3600,"success_fraction":0.01,"failure_fraction":0.1}
pragma: no-cache
report-to: {"group":"heroku-nel","endpoints":[{"url":"https://nel.heroku.com/reports?s=197YUN60p0S1xXG5svVdITLCj6uK2BoTpv08Rbz1LjplkA3D\u0026id=929419e7-33ea-4e2f-85f0-7d8b7cd5cbdb6\u0026ts=1759055627"}],"max_age":3600}
reporting-endpoints: heroku-nel="https://nel.heroku.com/reports?s=197YUN60p0S1xXG5svVdITLCj6uK2BoTpv08Rbz1LjplkA3D\u0026id=929419e7-33ea-4e2f-85f0-7d8b7cd5cbdb6\u0026ts=1759055627"
Server: cloudflare
vary: Origin, Accept-Encoding
via: 2.0 heroku-router
x-content-type-options: nosniff
x-powered-by: Express
cf-cache-status: DYNAMIC
CF-RAY: 986295a5c04438c-MAM
alt-svc: h3=":443"; ma=86400

[
  {
    "id": 2,
    "title": "JSON Tips",
    "userId": 2,
    "tags": [
      "json"
    ]
  }
]

```

Завдання №2

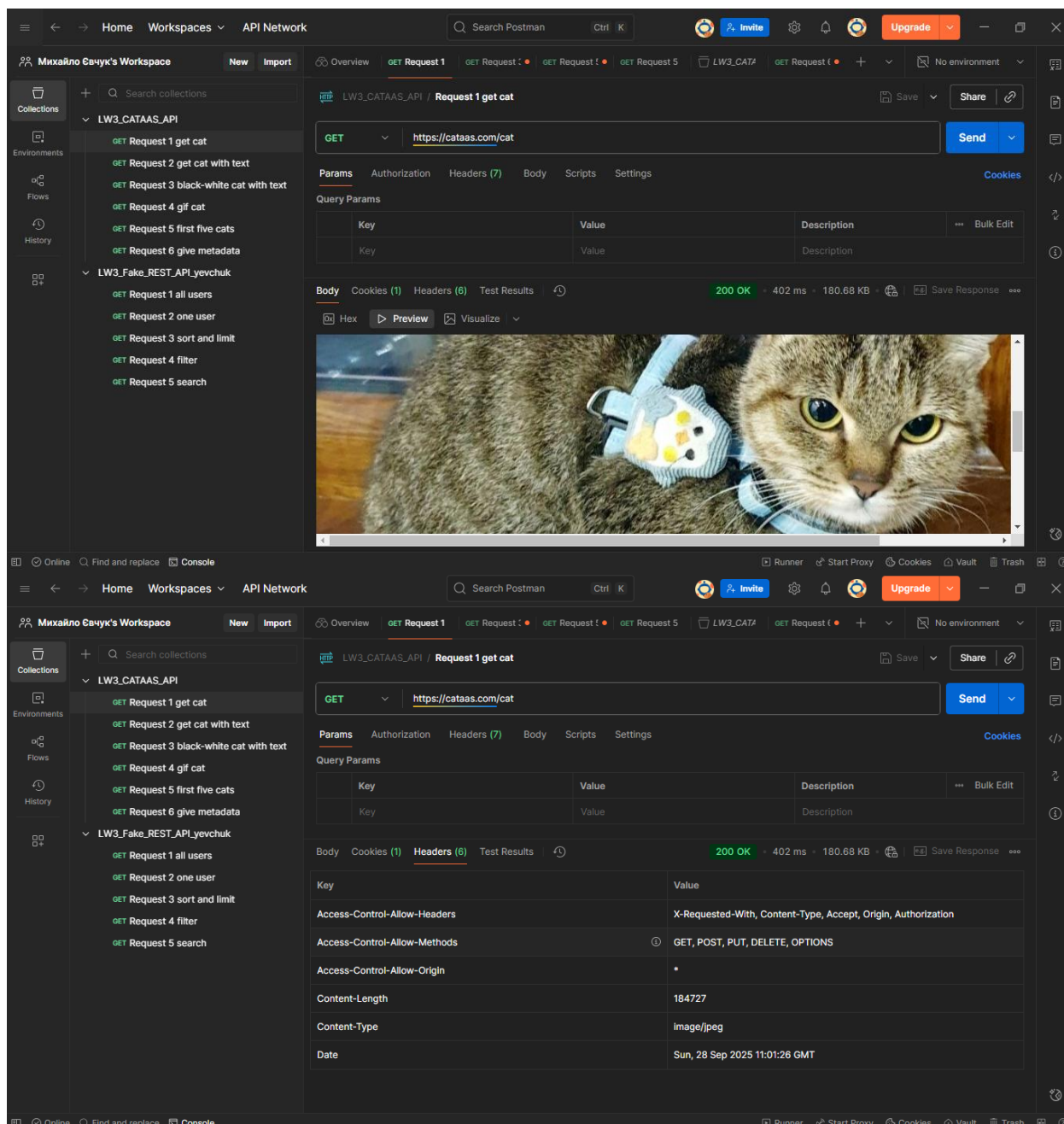
Робота з API сервісу CATAAS (Cat as a Service) та SwaggerAPI

1. Створюємо колекцію LW3_CATAAS_API

2. Виконуємо запити

— Отримати випадкове зображення кота

					ЛР.ОК24.ПІ231.13.03	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

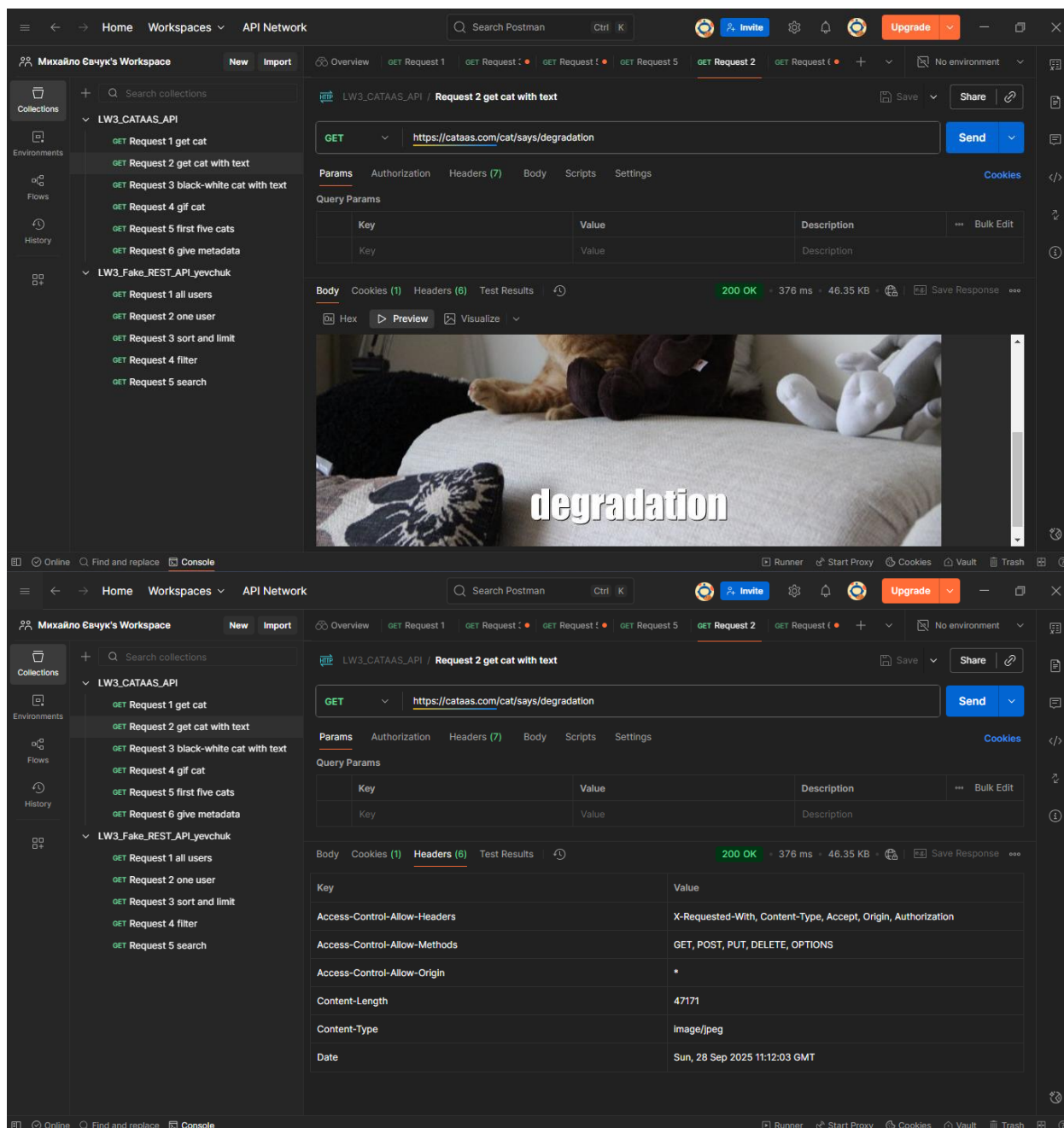


– Кіт із підписом вашого mood (картинка з текстом яка описує тебе).

Змн.	Арк.	№ докум.	Підпис	Дата

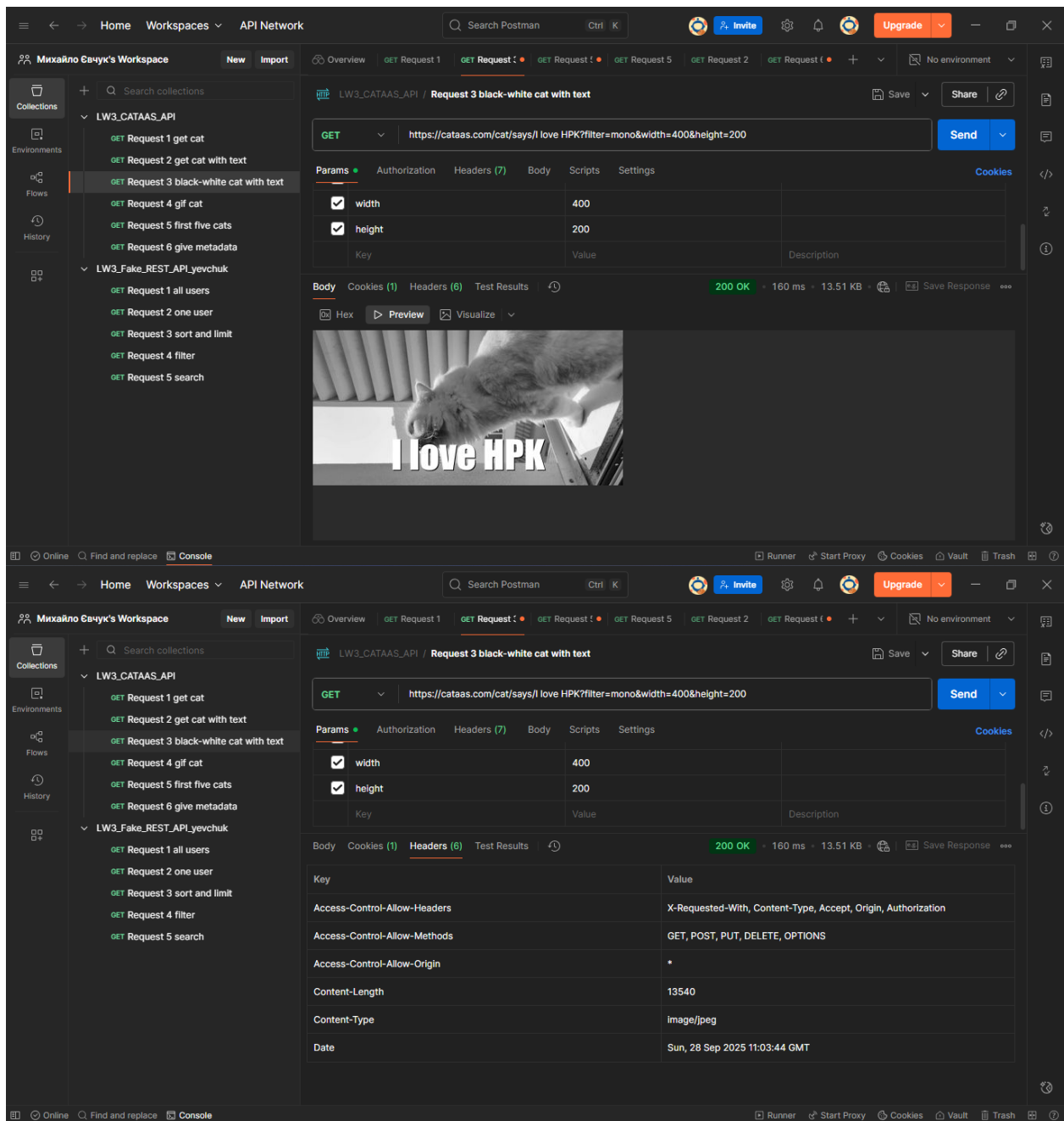
ЛР.ОК24.ПІ231.13.03

Арк.
13



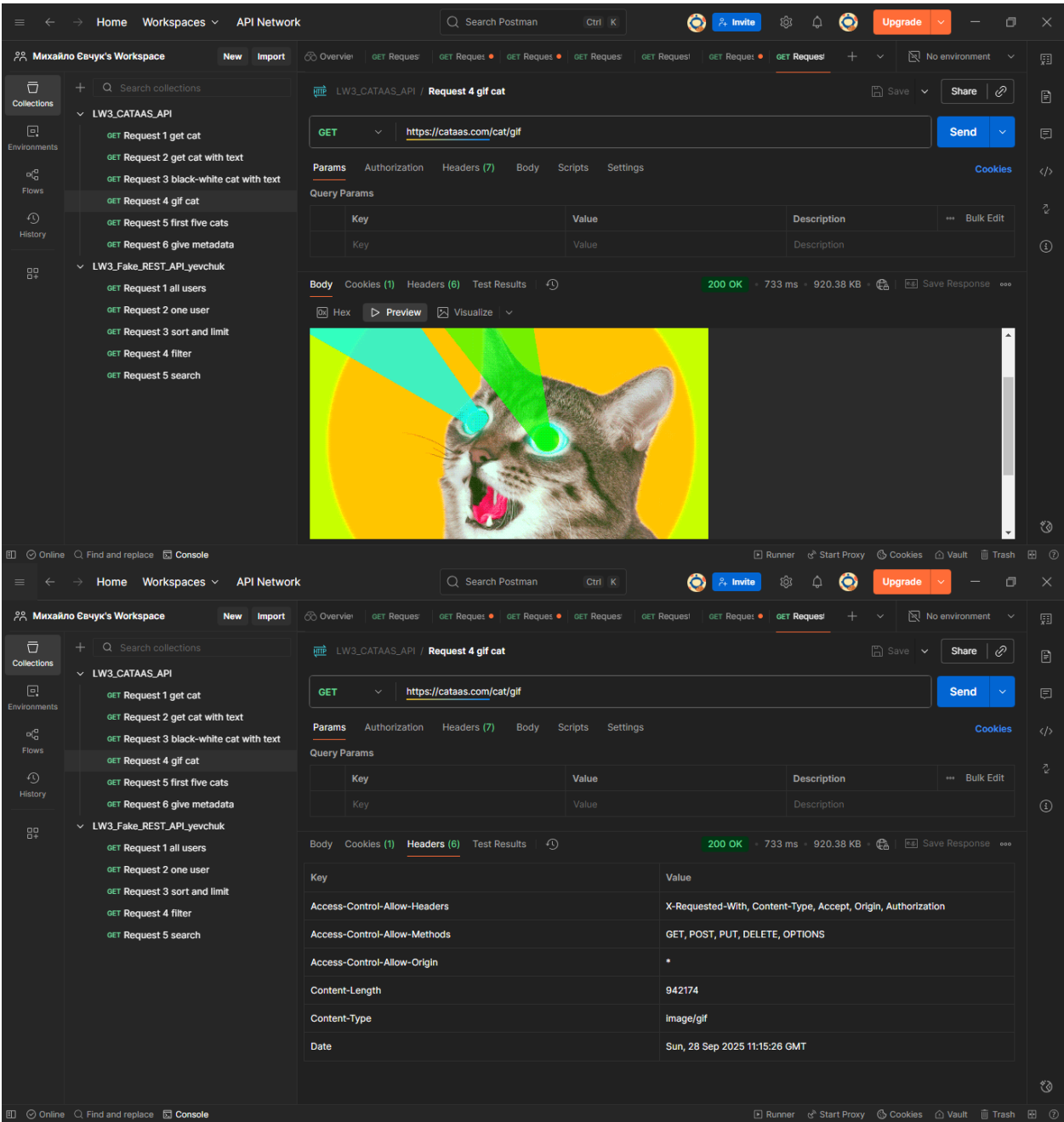
- Кіт із чорно-білим підписом. Спробуйте використати різні параметри запиту.

					ЛР.ОК24.ПІ231.13.03	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14



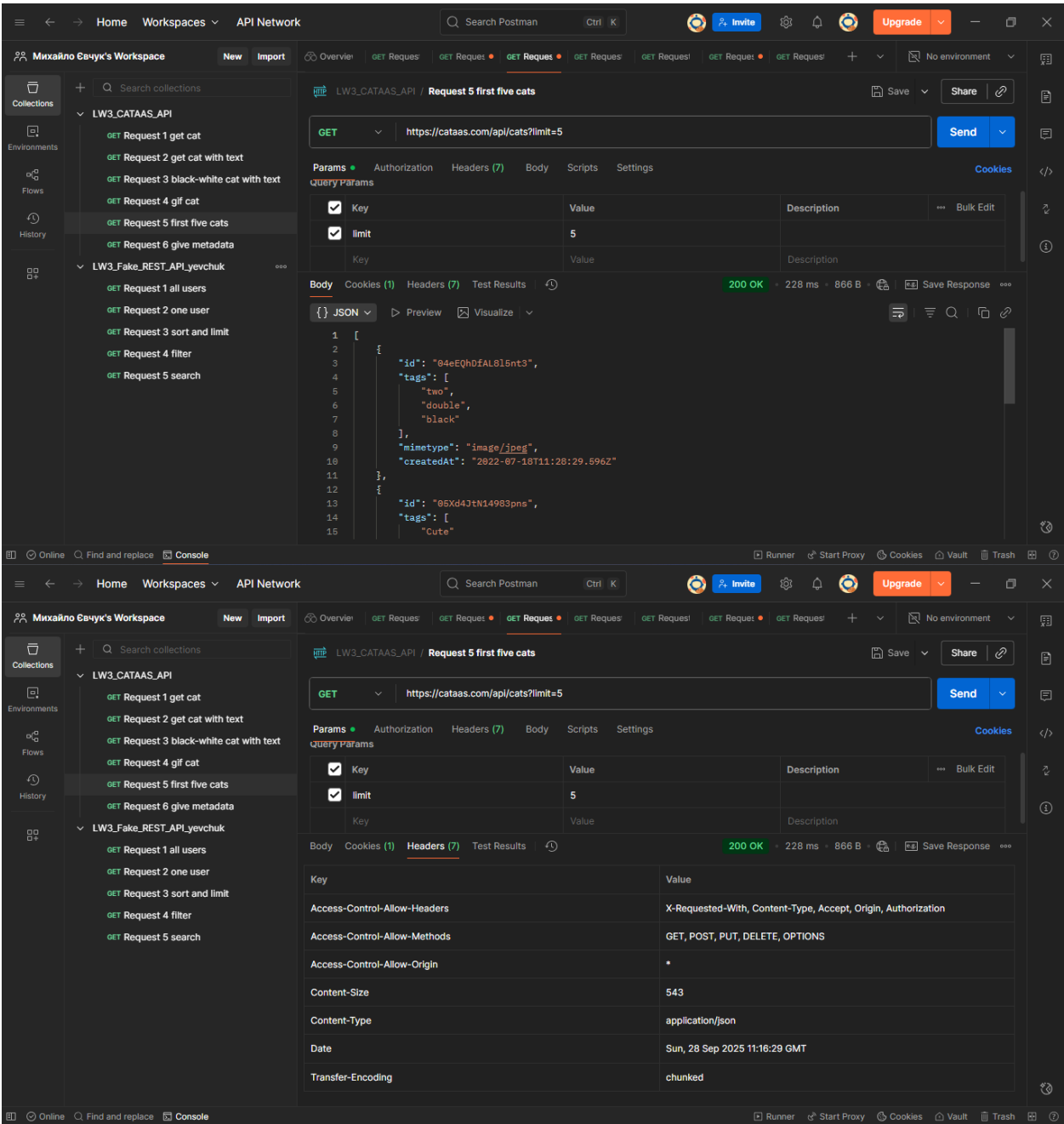
– Отримати випадкове gif-кота.

					ЛР.ОК24.ПІ231.13.03	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

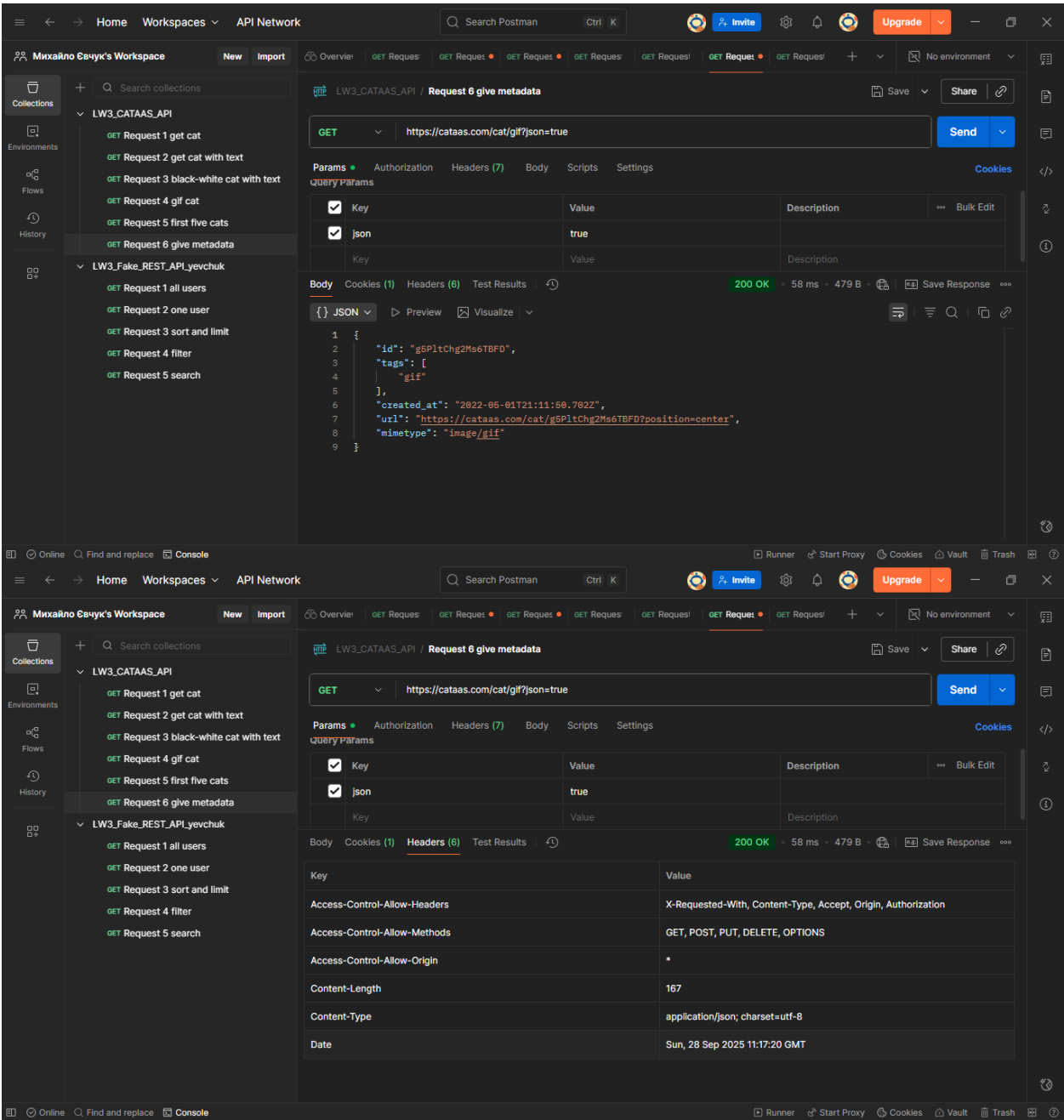


– Отримати JSON-список перших 5 котів.

Змн.	Арк.	№ докум.	Підпис	Дата



– Отримати метадані кото-GIF у форматі JSON.



3. 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:

					ЛР.ОК24.ПІ231.13.03	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

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:

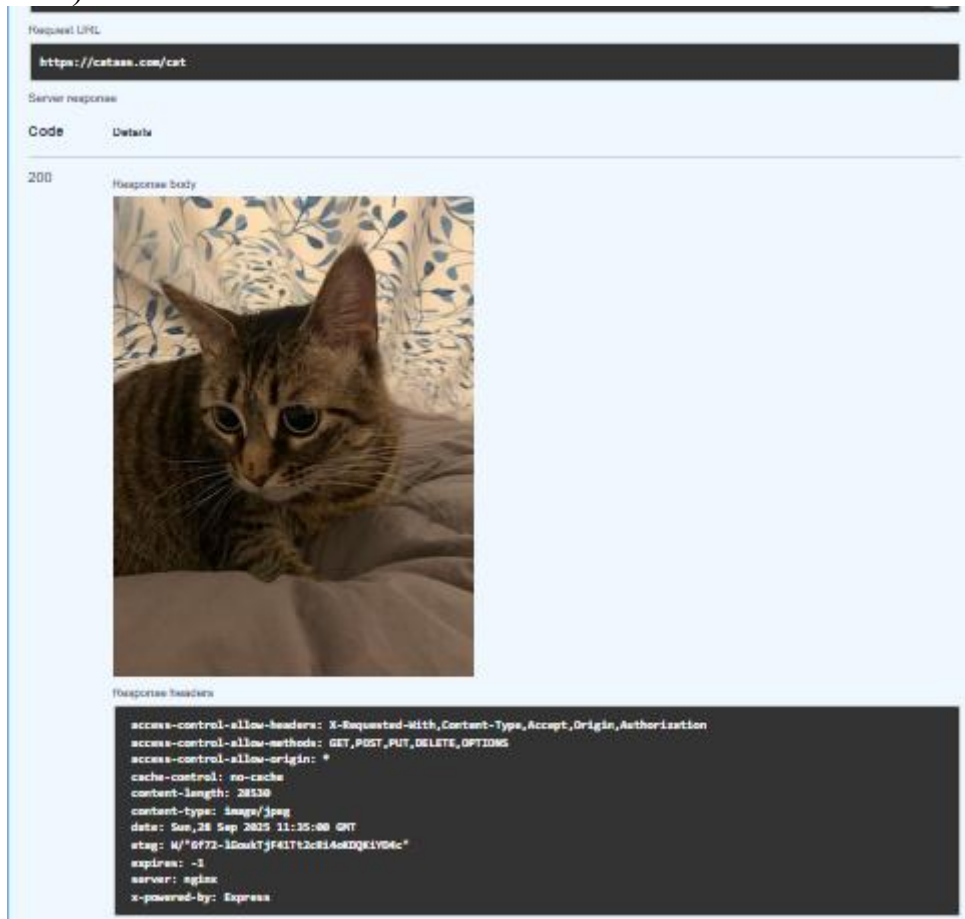
_id:

					ЛР.ОК24.ПІ231.13.03	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

type: string
description: Ідентифікатор зображення
created_at:
type: string
description: Дата створення
tags:
type: array
items:
type: string
description: Теги зображення

Виконаємо запити

а) Виконаємо запит кота



б) Виконаємо запит на кота з текстом

Request URL


<https://cataas.com/cat/says/REST>

Server response

Code Details

200

Response body



Response headers

```
access-control-allow-headers: X-Requested-With, Content-Type, Accept, Origin, Authorization
access-control-allow-methods: GET, POST, PUT, DELETE, OPTIONS
access-control-allow-origin: *
cache-control: no-cache
content-length: 48481
content-type: image/jpeg
date: Sun, 28 Sep 2025 11:33:26 GMT
etag: W/"b6f3-11e97c79713f8562eb09b7b71"
expires: -1
server: nginx
x-powered-by: Express
```

с) Виконаємо запит на усіх котів

Request URL

<https://cataas.com/api/cats>

Server response

Code Details

200

Response body

```
{
  {
    "id": "04e0b0fAL815nt3",
    "tags": [
      "two",
      "double",
      "black"
    ],
    "mimetype": "image/jpeg",
    "createdAt": "2022-07-18T11:28:29.596Z"
  },
  {
    "id": "05Ed47N34983pns",
    "tags": [
      "Cute"
    ],
    "mimetype": "image/jpeg",
    "createdAt": "2024-05-27T17:55:08.552Z"
  },
  {
    "id": "00uFspacQvF9jfn",
    "tags": [
      "Darkcat"
    ]
  }
}
```

Response headers

```
access-control-allow-headers: X-Requested-With, Content-Type, Accept, Origin, Authorization
access-control-allow-methods: GET, POST, PUT, DELETE, OPTIONS
access-control-allow-origin: *
cache-control: no-cache
content-encoding: gzip
content-size: 1103
content-type: application/json; charset=utf-8
date: Sun, 28 Sep 2025 11:33:26 GMT
etag: W/"44d-3ySoZa17Mdy11M0tfc715PUBSP0"
expires: -1
server: nginx
x-powered-by: Express
```

Змн.	Арк.	№ докум.	Підпис	Дата

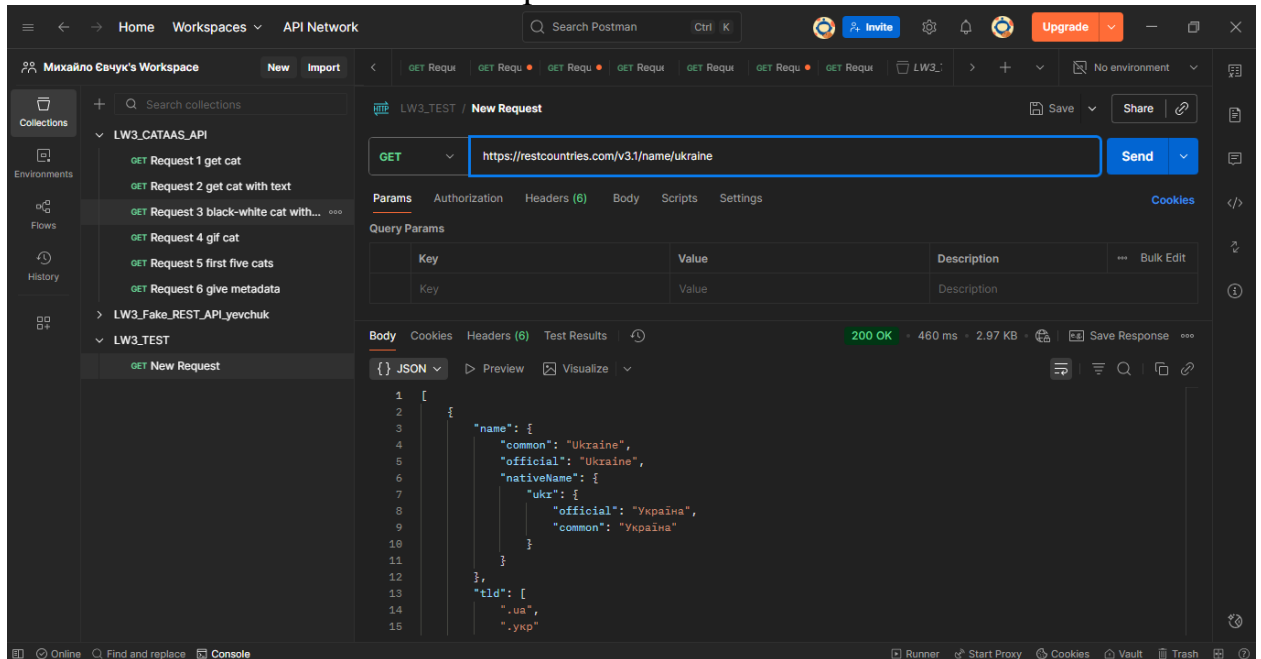
ЛР.ОК24.ПІ231.13.03

Завдання №3

Console-клієнт до публічного REST API

1. Створюємо консольний проєкт C# з назвою LW3_MiA.
2. Крок 1

Виконаємо тестовий запит через PostMan



3. Крок 2

Створимо класи:

```
public class CountryName
{
    public string common { get; set; } = string.Empty;
    public string official { get; set; } = string.Empty;
}

public class Flags
{
    public string png { get; set; } = string.Empty;
    public string svg { get; set; } = string.Empty;
    public string alt { get; set; } = string.Empty;
}

public class Country
{
    public CountryName name { get; set; } = new CountryName();
    public List<string> capital { get; set; } = new List<string>();
    public string region { get; set; } = string.Empty;
    public string subregion { get; set; } = string.Empty;
    public long population { get; set; }
    public Dictionary<string, string> languages { get; set; } = new
Dictionary<string, string>();
    public string cca2 { get; set; } = string.Empty;
    public Flags flags { get; set; } = new Flags();
}
```

4. Крок 3

Налаштуємо HttpClient.

```
private static readonly HttpClient _http = new HttpClient
{
    BaseAddress = new Uri("https://restcountries.com"),
    Timeout = TimeSpan.FromSeconds(15)
};
```

					ЛР.ОК24.ПІ231.13.03	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

5. Крок 4

Виконаємо GET запит і запитуємо назву, столицю, регіон, підрегіон, населення, мова, прапор

```
var response = await
_http.GetAsync("/v3.1/name/ukraine?fields=name,capital,region,subregion,population,languages,cca2,flags");

response.EnsureSuccessStatusCode();
```

6. Крок 5

Отримаємо JSON як рядок

```
var json = await response.Content.ReadAsStringAsync();
Console.WriteLine("Raw JSON:");
Console.WriteLine(json);
```

7. Крок 6

Розпарсимо JSON

```
var countries = JsonSerializer.Deserialize<List<Country>>(json, new
JsonSerializerOptions
{
    PropertyNameCaseInsensitive = true
});
```

8. Крок 7

Використаємо дані з об'єкта

```
if (countries != null && countries.Count > 0)
{
    var c = countries[0];

    Console.WriteLine("\n=== Результат парсингу ===");
    Console.WriteLine($"Назва:          {c.name.common} ({c.name.official})
[{c.cca2}]");
    Console.WriteLine($"Столиця:          {(c.capital.Count > 0 ? string.Join(", ",
c.capital) : "N/A")}");
    Console.WriteLine($"Регіон:          {c.region}");
    Console.WriteLine($"Підрегіон:       {c.subregion}");
    Console.WriteLine($"Населення:       {c.population:N0}");
    Console.WriteLine($"Мови:           {(c.languages.Count > 0 ? string.Join(",
", c.languages.Values) : "N/A")}");
    Console.WriteLine($"Прапор (PNG):    {c.flags.png}");
    Console.WriteLine($"Прапор (SVG):    {c.flags.svg}");
}
```

9. Повний код програми:

```
using System;
using System.Net.Http;
using System.Text.Json;
using System.Threading.Tasks;
using System.Collections.Generic;
public class CountryName
{
    public string common { get; set; } = string.Empty;
    public string official { get; set; } = string.Empty;
}
public class Flags
{
    public string png { get; set; } = string.Empty;
    public string svg { get; set; } = string.Empty;
    public string alt { get; set; } = string.Empty;
}
public class Country
{
}
```

					ЛР.ОК24.ПІ231.13.03	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    public CountryName name { get; set; } = new CountryName();
    public List<string> capital { get; set; } = new List<string>();
    public string region { get; set; } = string.Empty;
    public string subregion { get; set; } = string.Empty;
    public long population { get; set; }
    public Dictionary<string, string> languages { get; set; } = new
Dictionary<string, string>();
    public string cca2 { get; set; } = string.Empty;
    public Flags flags { get; set; } = new Flags();
}
class Program
{
    private static readonly HttpClient _http = new HttpClient
    {
        BaseAddress = new Uri("https://restcountries.com"),
        Timeout = TimeSpan.FromSeconds(15)
    };
    static async Task Main()
    {
        try
        {
            Console.WriteLine("Запит до REST Countries...");
            var url =
"/v3.1/name/ukraine?fields=name,capital,region,subregion,population,languages,c
ca2,flags";
            var response = await _http.GetAsync(url);
            Console.WriteLine($"Request URL: {_http.BaseAddress + url}");
            Console.WriteLine($"Status: {(int)response.StatusCode}
{response.ReasonPhrase}");
            Console.WriteLine($"Content-Type:
{response.Content.Headers.ContentType}");
            response.EnsureSuccessStatusCode();
            var json = await response.Content.ReadAsStringAsync();
            Console.WriteLine("Raw JSON:");
            Console.WriteLine(json);
            var countries = JsonSerializer.Deserialize<List<Country>>(json, new
JsonSerializerOptions
            {
                PropertyNameCaseInsensitive = true
            });

            if (countries != null && countries.Count > 0)
            {
                var c = countries[0];
                Console.WriteLine("\n=== Результат парсингу ===");
                Console.WriteLine($"Назва: {c.name.common}
({c.name.official}) [{c.cca2}]");
                Console.WriteLine($"Столиця: {(c.capital.Count > 0 ?
string.Join(", ", c.capital) : "N/A")}");
                Console.WriteLine($"Регіон: {c.region}");
                Console.WriteLine($"Підрегіон: {c.subregion}");
                Console.WriteLine($"Населення: {c.population:N0}");
                Console.WriteLine($"Мови: {(c.languages.Count > 0 ?
string.Join(", ", c.languages.Values) : "N/A")}");
                Console.WriteLine($"Прапор (PNG): {c.flags.png}");
                Console.WriteLine($"Прапор (SVG): {c.flags.svg}");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка: " + ex.Message);
        }
    }
}

```

Змн.	Арк.	№ докум.	Підпис	Дата

ЛР.ОК24.ПІ231.13.03

Арк.

24


```
}  
}
```

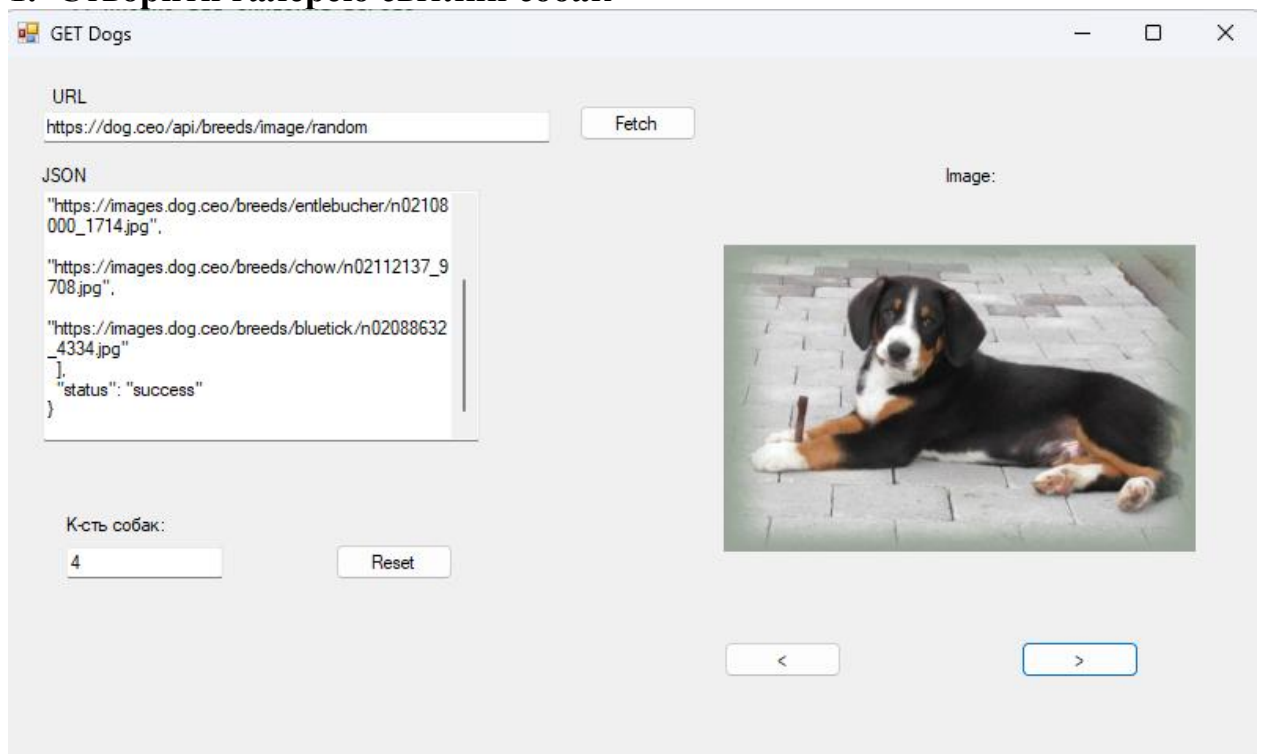
Результат програми:

```
Консоль отладки Microsoft Vi x + v  
Запит до REST Countries...  
Request URL: https://restcountries.com/v3.1/name/ukraine?fields=name,capital,region,subregion,population,languages,cca2,flags  
Status: 200 OK  
Content-Type: application/json  
Raw JSON:  
[{"flags":{"png":"https://flagcdn.com/w320/ua.png","svg":"https://flagcdn.com/ua.svg","alt":"The flag of Ukraine is composed of two equal horizontal bands of blue and yellow."},"name":{"common":"Ukraine","official":"Ukraine","nativeName":{"ukr":{"official":"Україна","common":"Україна"}}},"cca2":"UA","capital":["Kyiv"],"region":"Europe","subregion":"Eastern Europe","languages":{"ukr":"Ukrainian"},"population":44134693}]  
  
=== Результат парсингу ===  
Назва: Ukraine (Ukraine) [UA]  
Столиця: Kyiv  
Регіон: Europe  
Під регіон: Eastern Europe  
Населення: 44 134 693  
Мови: Ukrainian  
Прапор (PNG): https://flagcdn.com/w320/ua.png  
Прапор (SVG): https://flagcdn.com/ua.svg  
  
C:\Users\Mykha\source\repos\LW3_MiA\LW3_MiA\bin\Debug\net8.0\LW3_MiA.exe (процесс 18932) завершил работу с кодом 0 (0x0)  
.  
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".  
Нажмите любую клавишу, чтобы закрыть это окно:
```

Завдання №4

Інтеграція з Dog.CEO API у WinForms

1. Створити галерею світлин собак



Код програми:

```
using Newtonsoft.Json;  
using System;  
using System.Collections.Generic;  
using System.Drawing;  
using System.IO;  
using System.Net.Http;
```

					ЛР.ОК24.ПІ231.13.03	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

```

using System.Threading.Tasks;
using System.Windows.Forms;

namespace DogGalleryApp
{
    public partial class Form1 : Form
    {
        private static readonly HttpClient _http = new HttpClient();
        private List<string> dogImages = new List<string>();
        private int currentIndex = 0;

        public Form1()
        {
            InitializeComponent();
            pictureBox1.SizeMode = PictureBoxSizeMode.Zoom; // Переконайся, що
            pictureBox1 – це правильне ім'я елемента

            // Задаємо початковий URL
            textBox1.Text = "https://dog.ceo/api/breeds/image/random";
        }

        // Метод для завантаження зображень
        private async Task LoadImages(int count)
        {
            try
            {
                string url =
                $"https://dog.ceo/api/breeds/image/random/{count}";
                var json = await _http.GetStringAsync(url);
                var dogResponse =
                JsonConvert.DeserializeObject<DogResponse>(json);

                // Вивести отриманий JSON у textBox2
                textBox2.Text = JsonConvert.SerializeObject(dogResponse,
                Formatting.Indented); // Форматуємо JSON для читабельності

                if (dogResponse != null && dogResponse.message.Count > 0)
                {
                    dogImages = dogResponse.message;
                    currentIndex = 0; // Починаємо з першого зображення
                    DisplayImage();
                }
                else
                {
                    MessageBox.Show("Не вдалося отримати зображення.",
                    "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка: " + ex.Message, "Помилка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        // Метод для відображення зображення
        private void DisplayImage()
        {
            if (dogImages.Count > 0 && currentIndex >= 0 && currentIndex <
            dogImages.Count)
            {
                var imageUrl = dogImages[currentIndex];
            }
        }
    }
}

```

					ЛР.ОК24.ПІ231.13.03	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		26

```

        var imageBytes = _http.GetByteArrayAsync(imageUrl).Result;
        using (var ms = new MemoryStream(imageBytes))
        {
            pictureBox1.Image = Image.FromStream(ms);
        }
    }

    // Кнопка для завантаження N зображень
    private async void buttonLoad_Click(object sender, EventArgs e)
    {
        int count;
        // Перевірка, чи введено значення є числом
        if (int.TryParse(textBox3.Text, out count) && count >= 1)
        {
            await LoadImages(count); // Завантажуємо зображення
        }
        else
        {
            MessageBox.Show("Будь ласка, введіть коректне число більше 0.",
"Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    // Кнопка для перегляду попереднього зображення
    private void buttonPrev_Click(object sender, EventArgs e)
    {
        if (dogImages.Count > 0 && currentIndex > 0)
        {
            currentIndex--;
            DisplayImage();
        }
    }

    // Кнопка для перегляду наступного зображення
    private void buttonNext_Click(object sender, EventArgs e)
    {
        if (dogImages.Count > 0 && currentIndex < dogImages.Count - 1)
        {
            currentIndex++;
            DisplayImage();
        }
    }

    // Кнопка для скидання всього
    private void buttonReset_Click(object sender, EventArgs e)
    {
        textBox2.Clear();
        pictureBox1.Image = null;
        dogImages.Clear();
        currentIndex = 0; // Скидаємо індекс
        textBox3.Clear(); // Очищаємо TextBox
    }
}

// Клас для парсингу відповіді з API
public class DogResponse
{
    public List<string> message { get; set; } = new List<string>();
    public string status { get; set; } = string.Empty;
}
}

```

					ЛР.ОК24.ПІ231.13.03	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		27

Завдання №5

WinForms-клієнт до публічного REST API

Створити простий клієнт на C# WinForms, який зчитує дані з відкритого REST API, відображає список ресурсів і деталі одного ресурсу, підтримує мінімальні фільтри/пошук, та коректно обробляє помилки (якщо API дозволяє). API можна обрати зі списку публічних API або власне.

Що необхідно зробити:

1. Отримати колекцію (масив) JSON-об'єктів із відкритого API (Додаток А).
2. Відобразити в **DataGridView** (або **ListView**) з можливістю прокрутки.
3. Якщо можливості API дозволяють (більшіть так) виконати **мінімальний пошук/фільтр** (по одному полю або query-параметру API).
4. При виборі рядка зі списку — виконати **другий запит** (наприклад, GET /resource/{id} або аналог) і показати деталі у правій панелі (Labels/TextBoxes/PictureBox).
5. Кнопка **Refresh** для повторного завантаження списку.
6. Відображати користувачу читабельні повідомлення при: відсутності інтернету, 4xx/5xx, таймауті, невалідному JSON.
7. Блокувати кнопки під час запиту, показувати індикатор стану (**StatusStrip/Label**).

Нефункціональні вимоги

- .NET 6/7/8, WinForms, C#, за бажанням інша технологія, інші технології до написання UI. It's up to you!
- Структура проекту:

Forms/, Services/ApiClient.cs, Models/Dto.cs, Utils/.

- Код-стайл: читабельні назви, мінімальні коментарі, винесення констант (базова адреса API тощо).

Оберемо 13 варіант

13	TVMaze	https://api.tvmaze.com
----	--------	---

Макет програми:

					ЛР.ОК24.ПІ231.13.03	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		28

Show.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TVMazeClient.Models
{
    public class Show
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Language { get; set; }
        public string Premiered { get; set; }
        public string Summary { get; set; }
    }
}
```

ApiClient.cs

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Threading.Tasks;
using TVMazeClient.Models;

namespace TVMazeClient.Services
{
    public class ApiClient
    {
        private static readonly HttpClient client = new HttpClient();
        private const string BaseUrl = "https://api.tvmaze.com";

        // Отримання списку шоу
        public async Task<List<Show>> GetShowsAsync(string search = "")
```

					ЛР.ОК24.ПІ231.13.03	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    {
        try
        {
            var url = $"{BaseUrl}/search/shows?q={search}";
            var response = await client.GetStringAsync(url);
            var shows =
                JsonConvert.DeserializeObject<List<ApiShowResponse>>(response);
            return shows?.ConvertAll(s => new Show
            {
                Id = s.Show.Id,
                Name = s.Show.Name,
                Language = s.Show.Language,
                Premiered = s.Show.Premiered,
                Summary = s.Show.Summary
            });
        }
        catch (Exception ex)
        {
            throw new Exception("Error fetching shows: " + ex.Message);
        }
    }

    // Отримання деталей шоу
    public async Task<Show> GetShowDetailsAsync(int id)
    {
        try
        {
            var url = $"{BaseUrl}/shows/{id}";
            var response = await client.GetStringAsync(url);
            var show = JsonConvert.DeserializeObject<Show>(response);
            return show;
        }
        catch (Exception ex)
        {
            throw new Exception("Error fetching show details: " +
ex.Message);
        }
    }

    // Внутрішній клас для парсингу відповіді API
    private class ApiShowResponse
    {
        public Show Show { get; set; }
    }
}

```

Form1.cs

```

using System;
using System.Collections.Generic;
using System.Windows.Forms;
using TVMazeClient.Models;
using TVMazeClient.Services;

namespace LW3_Task5_MiA
{
    public partial class Form1 : Form
    {
        private readonly ApiClient _apiClient;
        private List<Show> _shows;

        public Form1()
        {

```

					ЛР.ОК24.ПІ231.13.03	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        InitializeComponent();
        _apiClient = new ApiClient();
    }

    // Завантаження шоу по запиту
    private async void btnLoad_Click(object sender, EventArgs e)
    {
        try
        {
            button1.Enabled = false;
            label1.Text = "Loading...";
            // Отримуємо шоу за допомогою API
            _shows = await _apiClient.GetShowsAsync(textBox1.Text);
            // Відображення шоу в DataGridView
            dataGridView1.DataSource = _shows;
            // Налаштування розтягування колонок

            dataGridView1.AutoSizeColumnsMode(DataGridViewAutoSizeColumnsMode.AllCells);
            // Якщо хочете вручну налаштувати ширину колонок, можна зробити
            так:

            dataGridView1.Columns["Id"].Width = 50;
            dataGridView1.Columns["Name"].Width = 200;
            dataGridView1.Columns["Language"].Width = 100;
            dataGridView1.Columns["Premiered"].Width = 120;
            dataGridView1.Columns["Summary"].Width = 270;
            // Для забезпечення гарного вигляду колонок можна додати
            мінімальну ширину
            dataGridView1.Columns["Id"].MinimumWidth = 50;
            dataGridView1.Columns["Name"].MinimumWidth = 100;
            dataGridView1.Columns["Language"].MinimumWidth = 80;
            dataGridView1.Columns["Premiered"].MinimumWidth = 100;
            dataGridView1.Columns["Summary"].MinimumWidth = 150;
            label1.Text = $"{_shows.Count} shows loaded.";
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error: {ex.Message}", "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            button1.Enabled = true;
        }
    }

    // Обробка кліку по шоу для отримання детальної інформації
    private async void dataGridView_CellClick(object sender,
    DataGridViewCellEventArgs e)
    {
        if (e.RowIndex >= 0)
        {
            int id = (int)dataGridView1.Rows[e.RowIndex].Cells["Id"].Value;
            var showDetails = await _apiClient.GetShowDetailsAsync(id);
            // Відображення деталей шоу
            label2.Text = showDetails.Name;
            label3.Text = showDetails.Summary;
            label4.Text = showDetails.Language;
            label5.Text = showDetails.Premiered;
        }
    }

    // Оновлення списку шоу
    private void btnRefresh_Click(object sender, EventArgs e)
    {

```

					ЛР.ОК24.ПІ231.13.03	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		31

```

    btnLoad_Click(sender, e);
}
}
}

```

Результат програми:

Form1

22

Id	Name	Language	Premiered	Summary
6646	22 Minutes	English	1993-10-11	<p>This skit-filled Canadian comedy show pokes fun ...
4733	NYC 22	English	2012-04-15	<p>Upper Manhattan, where students, longtime resid...
43063	22 комика	Russian	2019-07-22	<p>There is a chamber atmosphere here without dec...
45683	22. juli	Norwegian	2020-01-05	<p>They help the injured. They comfort the bereaved...
34363	Catch-22	English	2019-05-17	<p>Based on the acclaimed Joseph Heller novel, ...
62184	Space 22	English	2022-05-17	<p>Space 22 follows seven strangers, each ...
36080	M.K. 22	Hebrew	2004-03-21	<p>Somewhere is south Israel, the IDF guards the we...
45672	22/7	Japanese	2020-01-11	<p>In this animation series, the audience will witness t...
5529	11.22.63	English	2016-02-15	<p>Imagine having the power to change history. Wou...
53594	22 Kids and Counting	English	2021-02-22	<p>Series following Noel and Sue Radford as they co...

10 shows loaded.

22 Minutes 1993-10-11
 <p>This skit-filled Canadian comedy show pokes fun at the Canadian government and other national happenings.</p>
 English