

Національний технічний університет України «КПІ»  
Факультет інформатики та обчислювальної техніки  
Кафедра Інформаційних систем та технологій

Лабораторна робота №4  
з дисципліни « Сучасні технології розробки WEB-застосунків на  
платформі Microsoft.NET»  
на тему: «Імплементация REST API»

Виконав:  
студент гр. ІС-11  
Іваніцький М.  
Викладач:  
Бардін В.

2023 рік

## Завдання:

### Практична частина:

1. Використовуючи архітектуру розроблену в попередній роботі  
Імплементувати REST веб-API, використавши N Layered архітектуру.
2. Покрити модульними та інтеграційними тестами основні компоненти рішення.
3. Експортувати розроблене API до Postman.

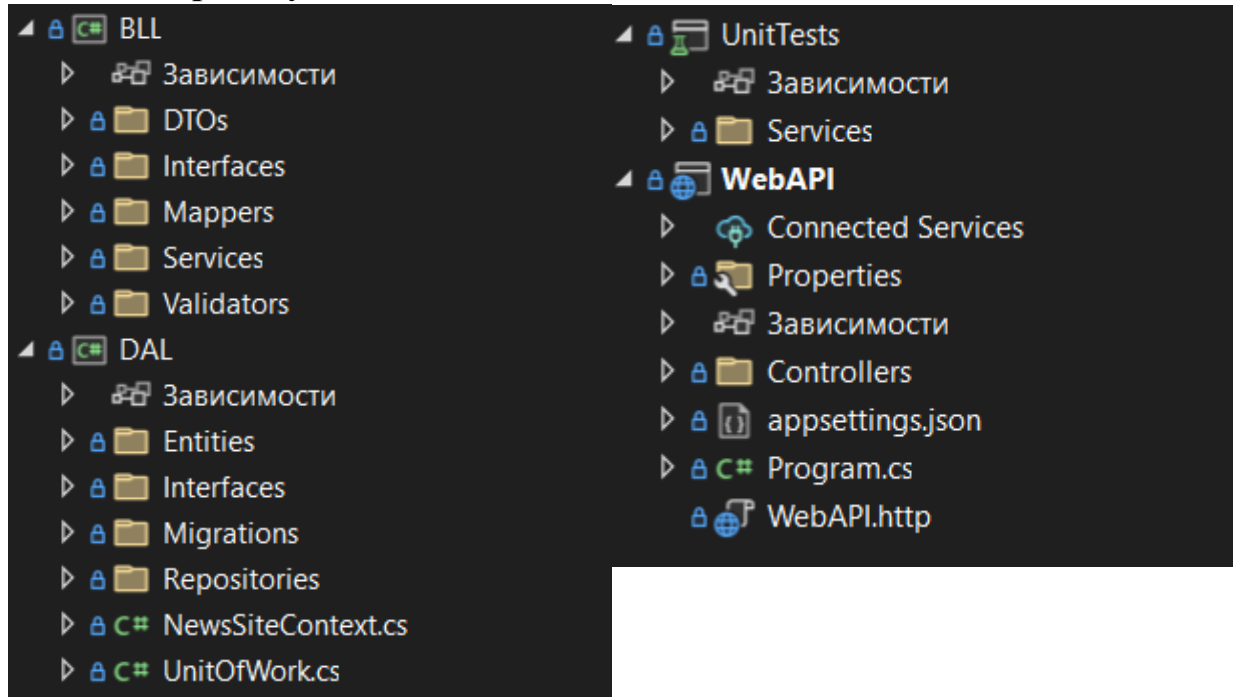
## Варіант:

10	Онлайн-новини. Сайт розташування новин зареєстрованим и авторами	<p>1. На сайті новин всі новини розташовуються за певними рубриками.</p> <p>2. Для новин формується набір тегів, за якими зручно шукати новини зазначеної тематики. Новини можуть вносити на сайт лише зареєстровані автори.</p> <p>3. Можливо передивлятися новини за рубрикою, тематикою, певного автора та за конкретний період.</p> <p><b>Функціональні вимоги:</b></p> <p>1. Супровід сайту новин;</p> <p>2. Забезпечення користування ним.</p>
----	---	--

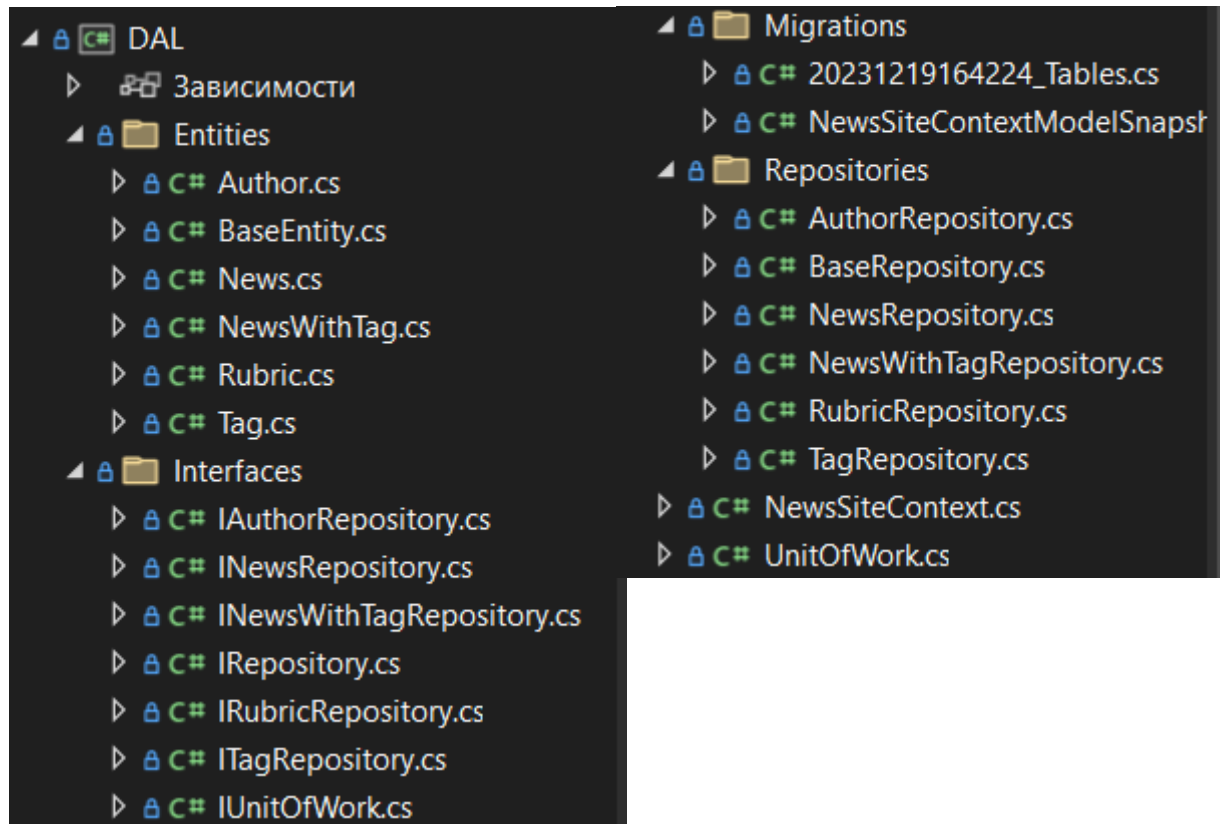
## Посилання на репозиторій:

[https://github.com/Maksvell/WebApi\\_NewsSite](https://github.com/Maksvell/WebApi_NewsSite)

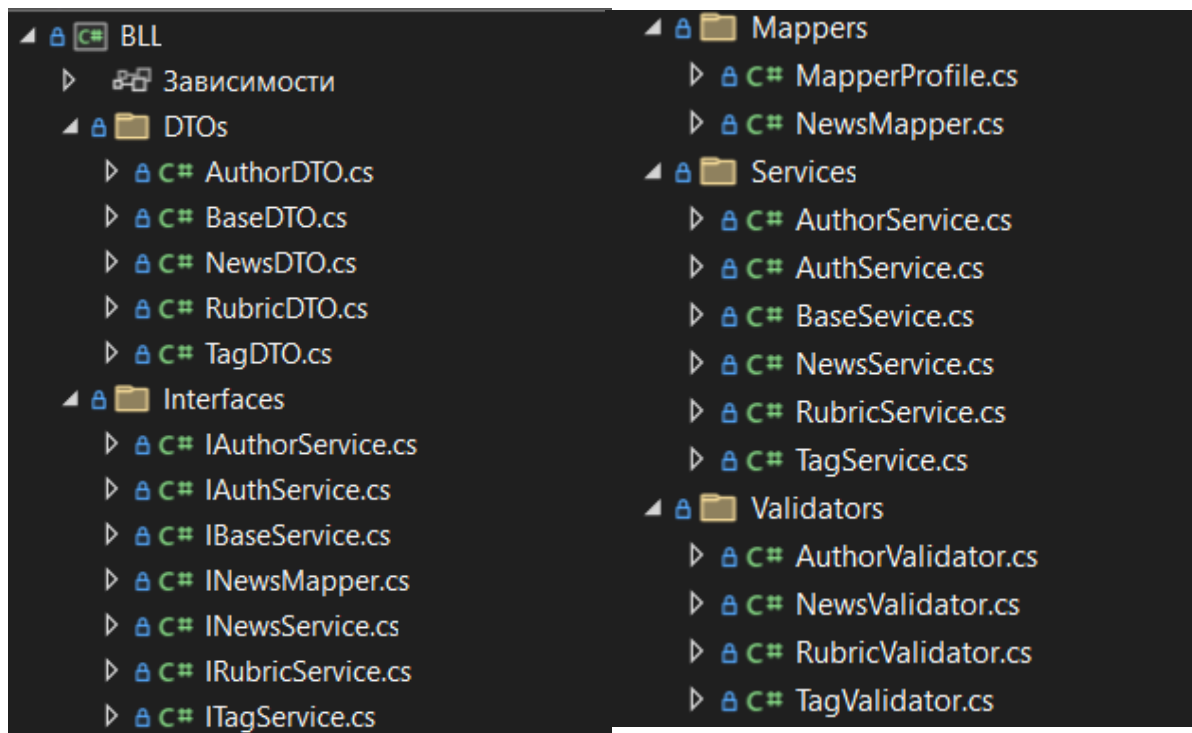
## Огляд проекту:



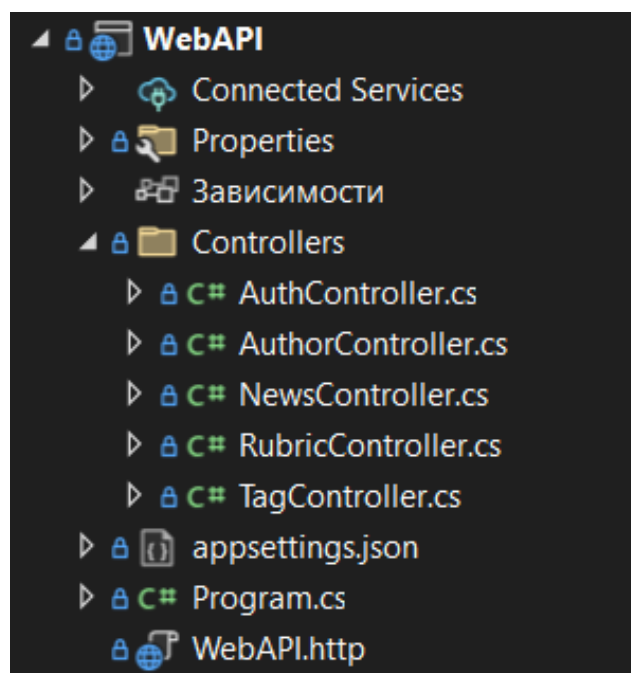
## Data Access Layer:



## Business Logic Layer:



## Presentation Layer:



## Тестування:

Обозреватель тестов

38 38 0

Поиск (Ctrl+I)

Запуск тестов завершен: тестов запущено 38 Предупреждений 0 Ошибок: 0

Тестирование	Длительность	Признаки
UnitTests (38)	262 мс	
UnitTests.Services (38)	262 мс	
AuthorServiceTests (11)	155 мс	
Add_DuplicateEmail_ShouldThrowException	106 мс	
Add_ValidAuthorDTO_ShouldReturnAuthor	13 мс	
CheckIfRegistered_ExistingUser_ReturnsTrue	1 мс	
CheckIfRegistered_NonExistingUser_ReturnsFalse	< 1 мс	
Delete_ExistingId_ShouldReturnTrue	2 мс	
Delete_NonExistingId_ShouldReturnFalse	1 мс	
GetAll_ShouldReturnListOfAuthors	9 мс	
GetById_ExistingId_ShouldReturnAuthor	2 мс	
GetById_NonExistingId_ShouldReturnNull	< 1 мс	
Update_ExistingIdAndValidDTO_ShouldReturnAuthor	2 мс	
Update_NonExistingId_ShouldReturnNull	19 мс	
AuthServiceTests (3)	54 мс	
NewsServiceTests (8)	18 мс	
RubricServiceTests (8)	20 мс	
TagServiceTests (8)	15 мс	

## **Контролери:**

### **AuthController:**

- (POST) api/register – Реєстрація автора;
- (POST) api/login – Вхід зареєстрованого автора в систему (генерація токена за допомогою якого можна в подальшому використовувати дії створення, редагування і тп);

### **AuthorController:**

- (GET) api/authors – Отримати усіх авторів;
- (GET) api/authors/{id} – Отримати інформацію про конкретного автора;
- (DELETE) api/authors/{id} – Видалити конкретного автора;
- (PUT) api/authors/{id} – Оновити інформацію про конкретного автора;

### **NewsController:**

- (GET) api/news – Отримати усі новини;
- (GET) api/news/{id} – Отримати конкретну новину;
- (GET) api/news/by-author/{id} – Отримати новини за конкретним автором;
- (GET) api/news/by-rubric/{id} – Отримати новини за конкретною рубрикою;
- (GET) api/news/by-tag/{id} – Отримати новини за конкретним тегом;
- (POST) api/news – Створити новину;
- (PUT) api/news/{id} – Оновити інформацію про конкретну новину;
- (DELETE) api/news/{id} – Видалити конкретну новину;

### **RubricController:**

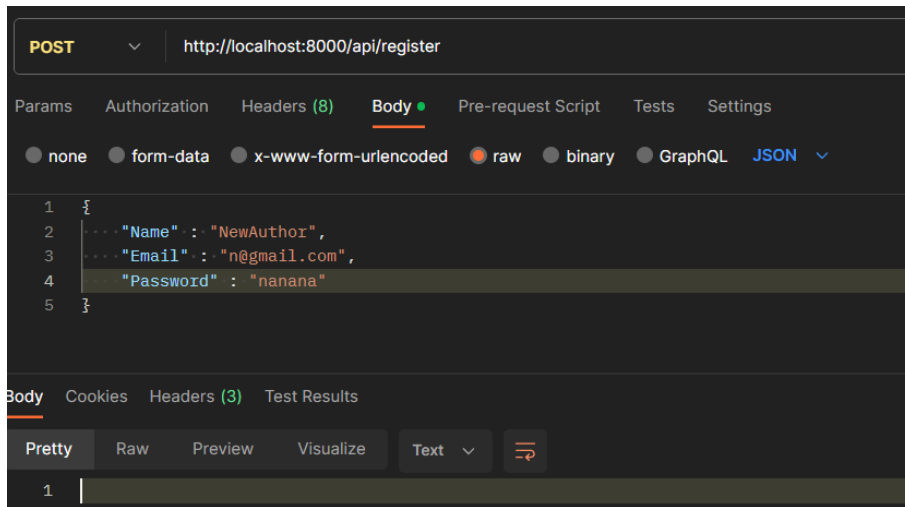
- (GET) api/rubrics – Отримати усіх авторів;
- (GET) api/rubrics/{id} – Отримати інформацію про конкретну рубрику;
- (POST) api/rubrics – Створити нову рубрику;
- (DELETE) api/rubrics/{id} – Видалити конкретну рубрику;
- (PUT) api/rubrics/{id} – Оновити інформацію про конкретну рубрику;

### **TagController:**

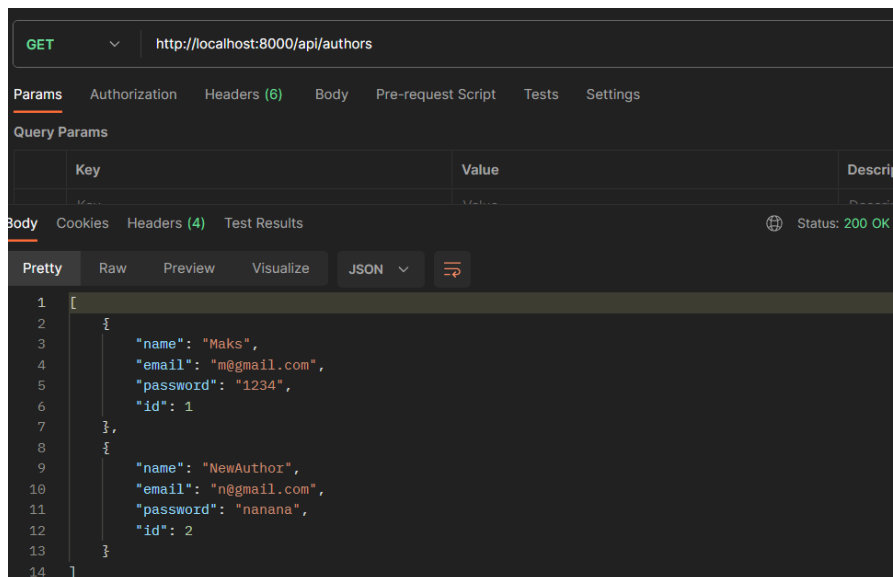
- (GET) api/tags – Отримати усіх авторів;
- (GET) api/tags/{id} – Отримати інформацію про конкретний тег;
- (POST) api/tags – Створити новий тег;
- (DELETE) api/tags/{id} – Видалити конкретний тег;
- (PUT) api/tags/{id} – Оновити інформацію про конкретний тег;

## Інтеграційне тестування:

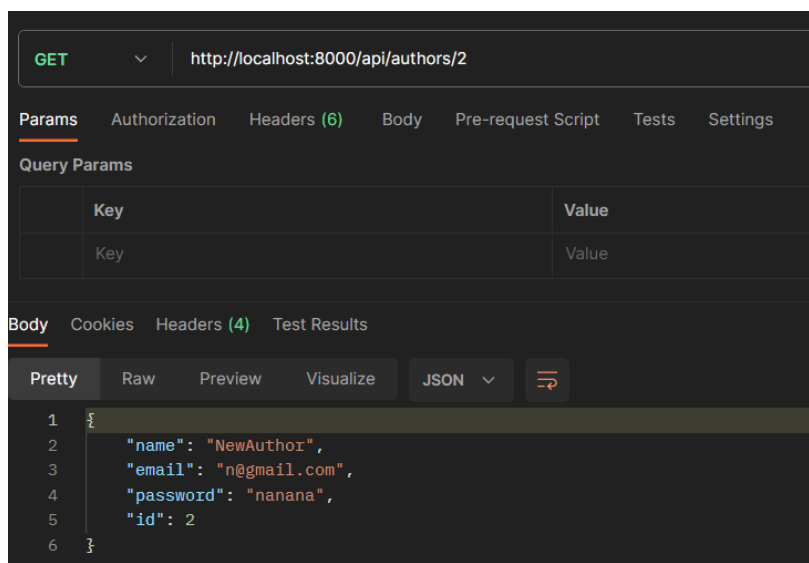
Створимо нового автора:



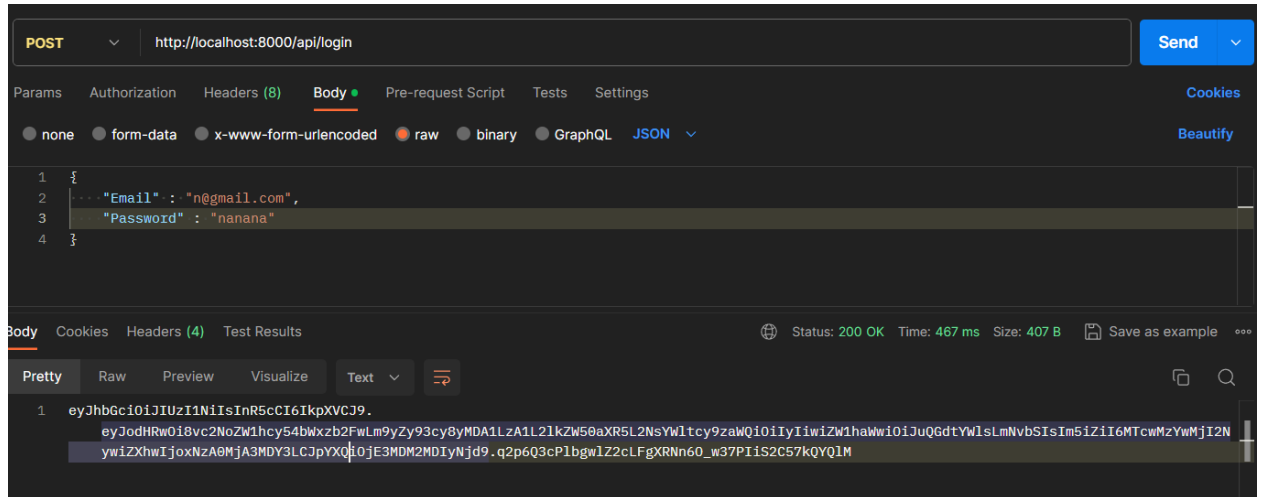
Перевіримо чи з'явився він серед авторів:



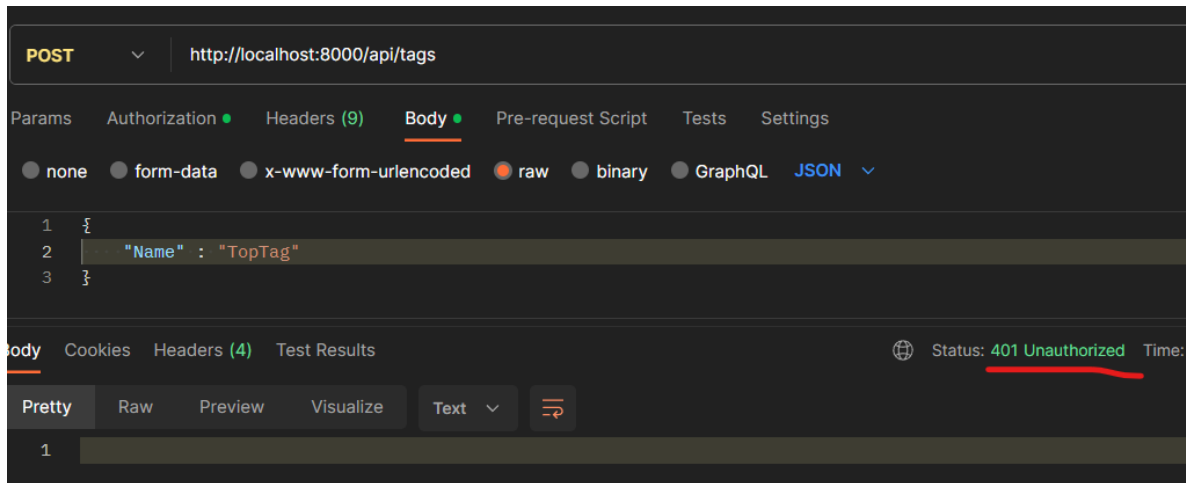
Перевіримо чи зможемо отримати його за Id:



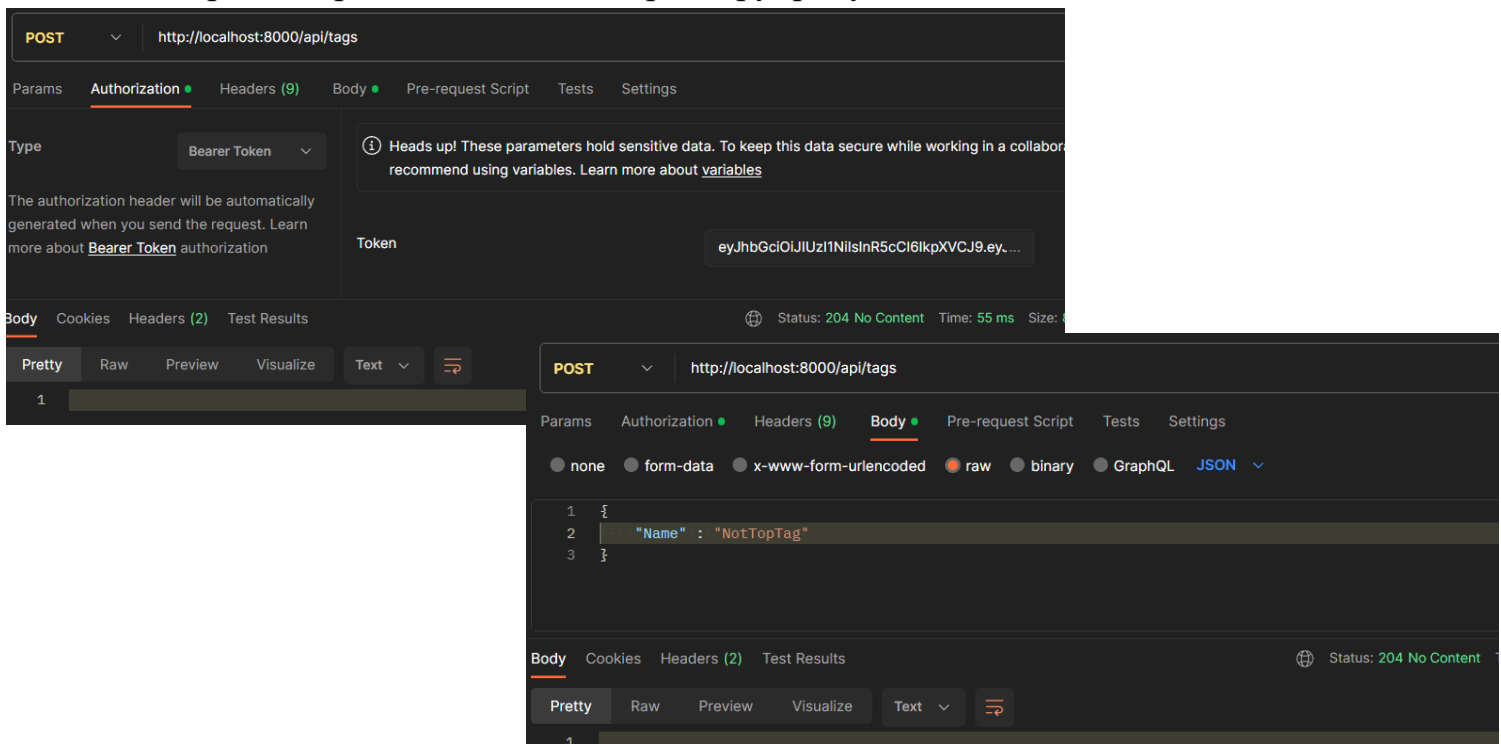
Спробуємо залогінитись та отримати токен:



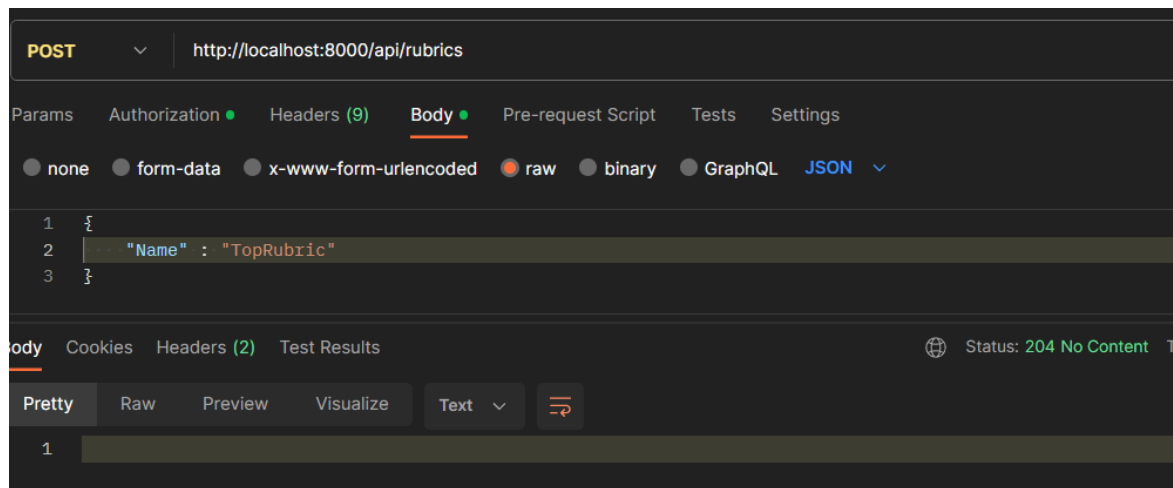
Додамо рубрику та декілька тегів:



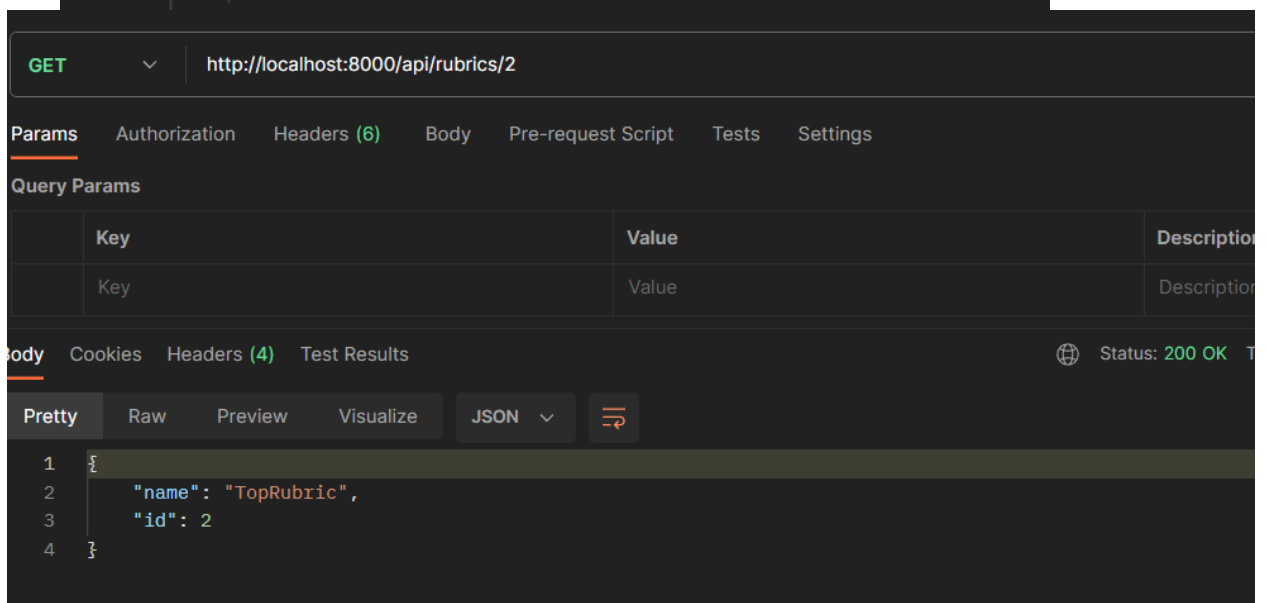
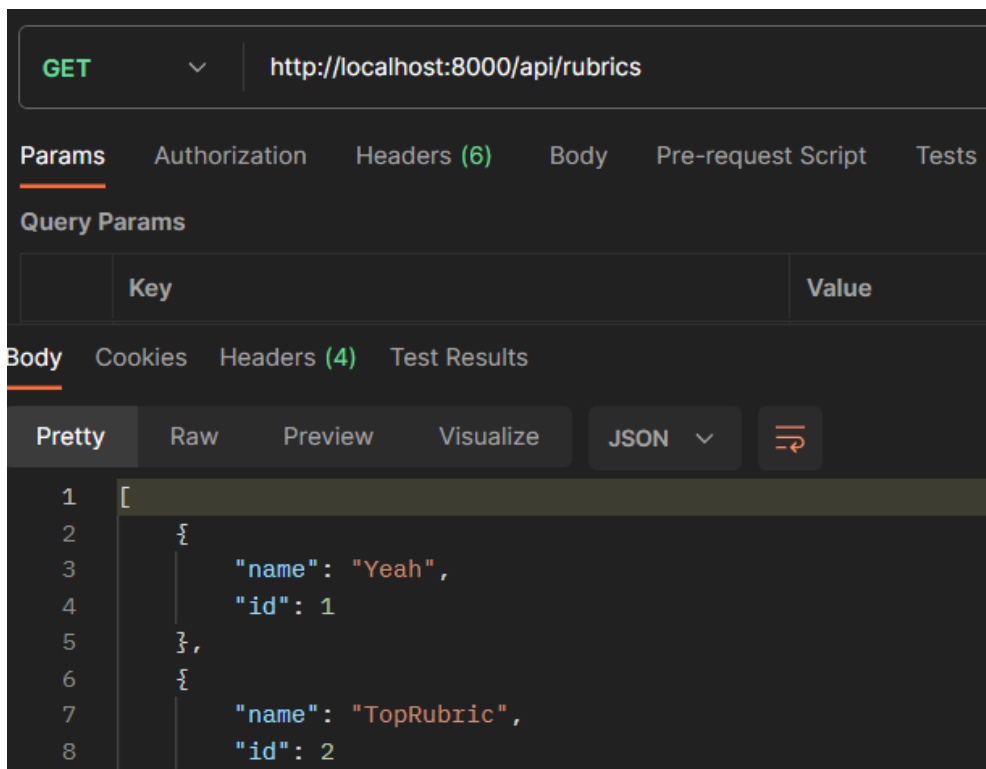
Не дозволяє, оскільки ми не авторизовані, використаємо токен, який отримали раніше та таки створимо рубрику та декілька тегів:







Перевіримо чи наявні вони у загальних списках та чи можемо ми їх отримати по Id:



GET

http://localhost:8000/api/tags

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body

Cookies

Headers (4)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize

JSON

```
1 [
2   {
3     "name": "TopTag",
4     "id": 1
5   },
6   {
7     "name": "NotTopTag",
8     "id": 2
9   }
10 ]
```

GET

http://localhost:8000/api/tags/1

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body

Cookies

Headers (4)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "name": "TopTag",
3   "id": 1
4 }
```

GET

http://localhost:8000/api/tags/2

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body

Cookies

Headers (4)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "name": "NotTopTag",
3   "id": 2
4 }
```

Відредагуємо рубрику з тегом та перевіримо чи змінились вони:

PUT http://localhost:8000/api/rubrics/2

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded **raw** binary GraphQL JSON

```
1 {
2   "Name" : "Top"
3 }
```

Body Cookies Headers (3) Test Results Status: 200 OK

GET http://localhost:8000/api/rubrics/2

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings

Query Params

	Key	Value	Description
--	-----	-------	-------------

Body Cookies Headers (4) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "name": "Top",
3   "id": 2
4 }
```

PUT http://localhost:8000/api/tags/2

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded **raw** binary GraphQL JSON

```
1 {
2   "Name" : "SecondTag"
3 }
```

Body Cookies Headers (3) Test Results Status: 200 OK

GET http://localhost:8000/api/tags/2

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings

Query Params

	Key	Value	Description
--	-----	-------	-------------

Body Cookies Headers (4) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "name": "SecondTag",
3   "id": 2
4 }
```

Також відредагуємо нашого автора:

The first screenshot shows a PUT request to `http://localhost:8000/api/authors/2` with a JSON body: `{ "Name": "TheBestAuthor", "Email": "n@gmail.com", "Password": "nanana" }`. The status is 200 OK.

The second screenshot shows a GET request to `http://localhost:8000/api/authors/2` with a JSON response: `{ "name": "TheBestAuthor", "email": "n@gmail.com", "password": "nanana", "id": 2 }`. The status is 200 OK.

І видалимо іншого автора:

The first screenshot shows a DELETE request to `http://localhost:8000/api/authors/1`. The status is 200 OK.

The second screenshot shows a GET request to `http://localhost:8000/api/authors` with a JSON response: `[ { "name": "TheBestAuthor", "email": "n@gmail.com", "password": "nanana", "id": 2 } ]`. The status is 200 OK.

Також видалимо од ну рубрику, т а створимо, а потім видалимо тег:

DELETE

http://localhost:8000/api/rubrics/1

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettings

Query Params

	Key	Value	Description
--	-----	-------	-------------

BodyCookiesHeaders (3)Test Results

Status: 200 OK

GET

http://localhost:8000/api/rubrics

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

Query Params

BodyCookiesHeaders (4)Test Results

Status: 200 OK

PrettyRawPreviewVisualizeJSON

```
1  [
2    {
3      "name": "Top",
4      "id": 2
5    }
6  ]
```

POST

http://localhost:8000/api/tags

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettings

noneform-datax-www-form-urlencodedrawbinaryGraphQLJSON

```
1  {
2    "Name" : "TrashTag"
3  }
```

BodyCookiesHeaders (2)

Status: 200 OK

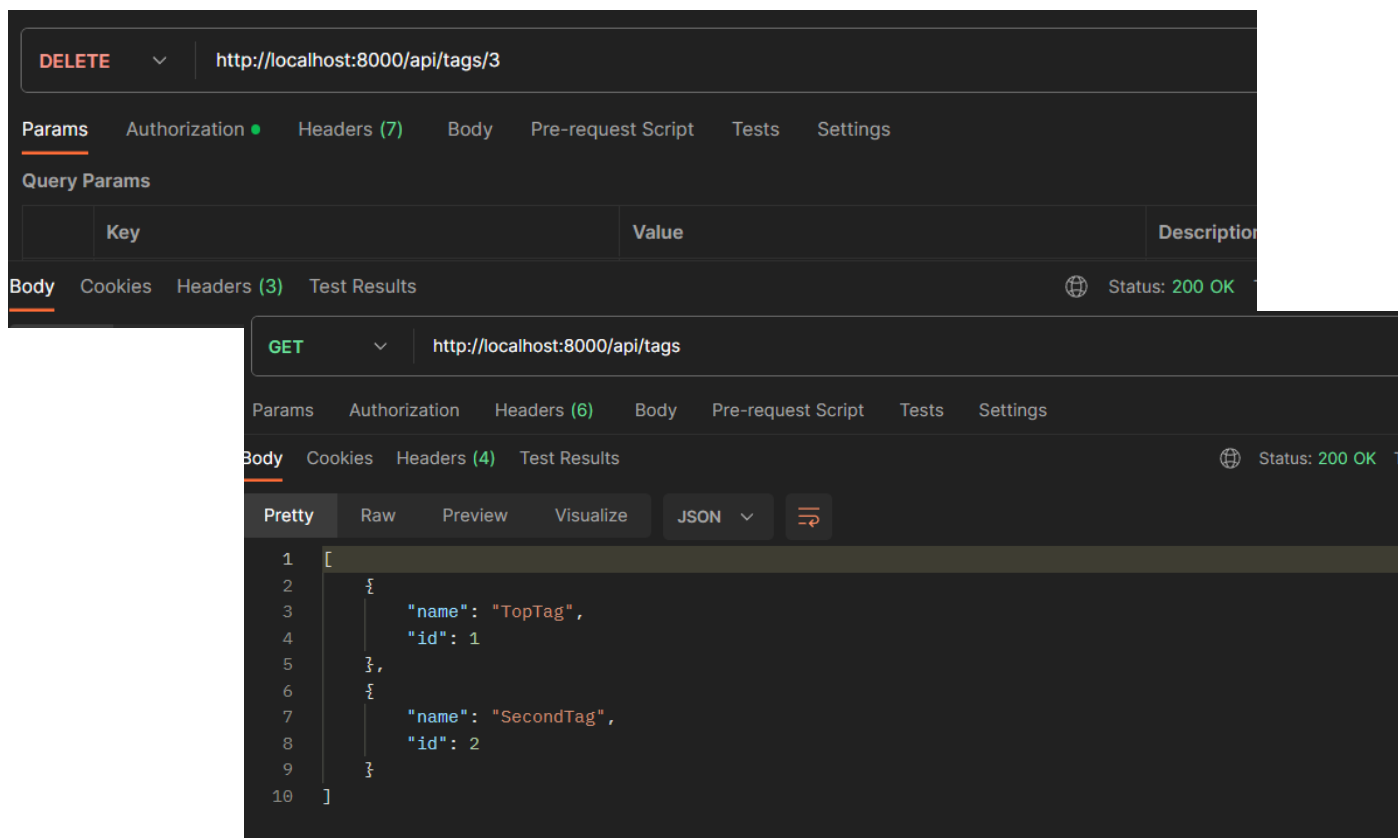
ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

BodyCookiesHeaders (4)Test Results

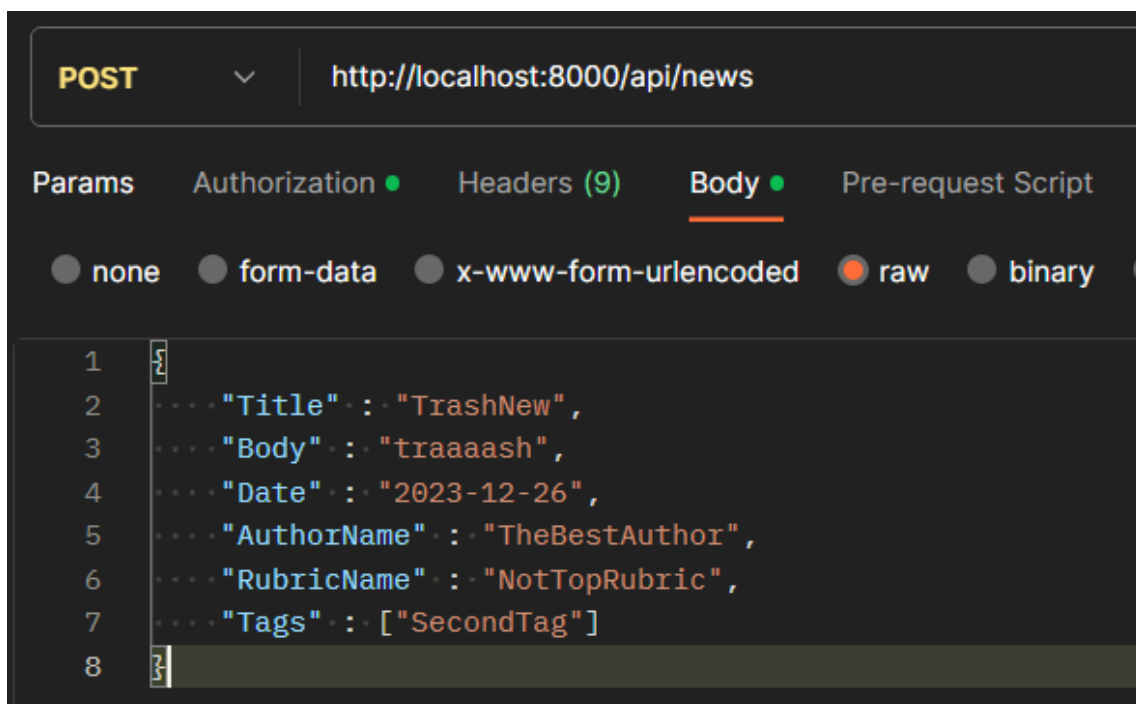
Status: 200 OK

PrettyRawPreviewVisualizeJSON

```
1  [
2    {
3      "name": "TopTag",
4      "id": 1
5    },
6    {
7      "name": "SecondTag",
8      "id": 2
9    },
10   {
11     "name": "TrashTag",
12     "id": 3
13   }
```



Тепер створимо декілька новин:



GET http://localhost:8000/api/news

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (4) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1  [
2    {
3      "title": "BestNew1",
4      "body": "lorem ipsum lalala",
5      "date": "2023-12-25T00:00:00",
6      "authorName": "TheBestAuthor",
7      "rubricName": "Top",
8      "tags": [
9        "TopTag",
10       "SecondTag"
11     ],
12     "id": 2
13   },
```

GET http://localhost:8000/api/news/2

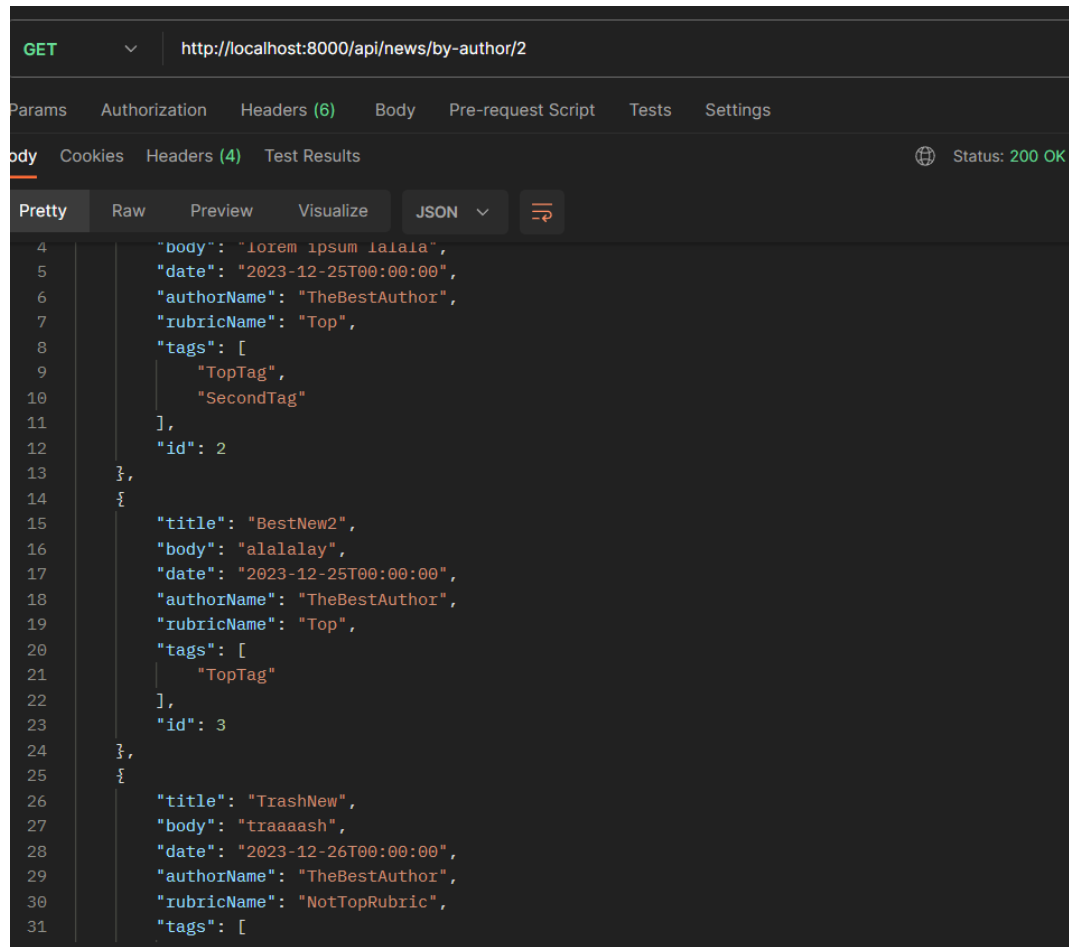
Params Authorization Headers (6) Body Pre-request Script Tests

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
1  {
2    "title": "BestNew1",
3    "body": "lorem ipsum lalala",
4    "date": "2023-12-25T00:00:00",
5    "authorName": "TheBestAuthor",
6    "rubricName": "Top",
7    "tags": [
8      "TopTag",
9      "SecondTag"
10   ],
11   "id": 2
12 }
```

Спробуємо отримати всі новини за автором, рубрикою та тегом:

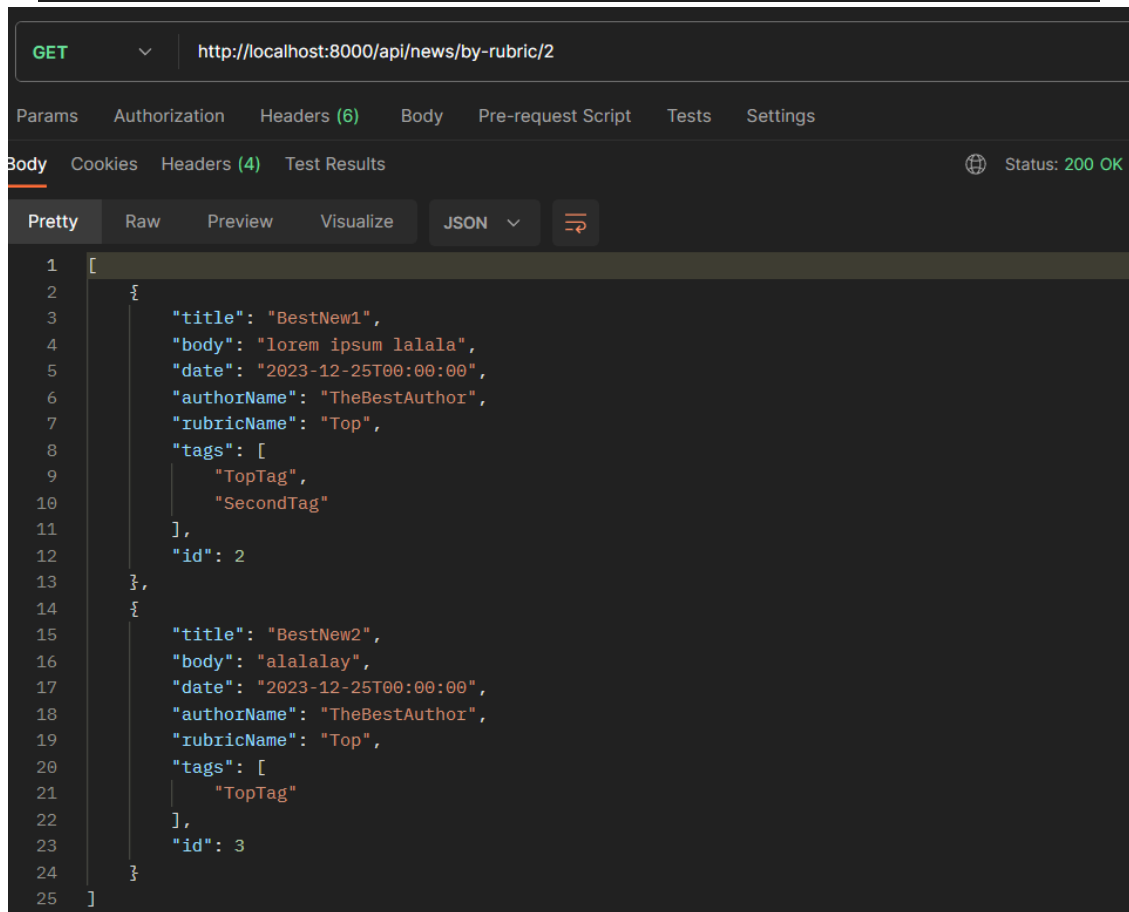


```
GET http://localhost:8000/api/news/by-author/2

Params Authorization Headers (6) Body Pre-request Script Tests Settings
Body Cookies Headers (4) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

[
  {
    "body": "lorem ipsum lalala",
    "date": "2023-12-25T00:00:00",
    "authorName": "TheBestAuthor",
    "rubricName": "Top",
    "tags": [
      "TopTag",
      "SecondTag"
    ],
    "id": 2
  },
  {
    "title": "BestNew2",
    "body": "alalalay",
    "date": "2023-12-25T00:00:00",
    "authorName": "TheBestAuthor",
    "rubricName": "Top",
    "tags": [
      "TopTag"
    ],
    "id": 3
  },
  {
    "title": "TrashNew",
    "body": "traaaash",
    "date": "2023-12-26T00:00:00",
    "authorName": "TheBestAuthor",
    "rubricName": "NotTopRubric",
    "tags": [
```



```
GET http://localhost:8000/api/news/by-rubric/2

Params Authorization Headers (6) Body Pre-request Script Tests Settings
Body Cookies Headers (4) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

[
  {
    "title": "BestNew1",
    "body": "lorem ipsum lalala",
    "date": "2023-12-25T00:00:00",
    "authorName": "TheBestAuthor",
    "rubricName": "Top",
    "tags": [
      "TopTag",
      "SecondTag"
    ],
    "id": 2
  },
  {
    "title": "BestNew2",
    "body": "alalalay",
    "date": "2023-12-25T00:00:00",
    "authorName": "TheBestAuthor",
    "rubricName": "Top",
    "tags": [
      "TopTag"
    ],
    "id": 3
  }
]
```



```
GET http://localhost:8000/api/news/by-tag/2

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (4) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

1 [
2   {
3     "title": "BestNew1",
4     "body": "lorem ipsum lalala",
5     "date": "2023-12-25T00:00:00",
6     "authorName": "TheBestAuthor",
7     "rubricName": "Top",
8     "tags": [
9       "TopTag",
10      "SecondTag"
11    ],
12    "id": 2
13  },
14  {
15    "title": "BestNew2",
16    "body": "alalalay",
17    "date": "2023-12-25T00:00:00",
18    "authorName": "TheBestAuthor",
19    "rubricName": "Top",
20    "tags": [
21      "TopTag"
22    ],
23    "id": 3
24  },
25 ]
```

Відредагуємо якусь новину:

```
PUT http://localhost:8000/api/news/4

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {
2   "Title": "RealyTrashNews",
3   "Body": "full trash it must be deleted",
4   "Date": "2023-12-26",
5   "AuthorName": "TheBestAuthor",
6   "RubricName": "NotTopRubric",
7   "Tags": ["SecondTag"]
8 }

Body Cookies Headers (5)

GET http://localhost:8000/api/news/4


Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (4) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

1 {
2   "title": "RealyTrashNews",
3   "body": "full trash it must be deleted",
4   "date": "2023-12-26T00:00:00",
5   "authorName": "TheBestAuthor",
6   "rubricName": "NotTopRubric",
7   "tags": [
8     "SecondTag"
9   ],
10  "id": 4
11 }
```


І видалимо її:


**DELETE**  `http://localhost:8000/api/news/4`


Params Authorization ● Headers (9) Body ● Pre-request Script Tests Settings

Query Params


	Key	Value	Description
--	-----	-------	-------------



body Cookies Headers (3) Test Results  Status: 200 OK

Pretty Raw Preview Visualize JSON 

**GET**  `http://localhost:8000/api/news`

Params Authorization Headers (6) Body Pre-request Script Tests Settings

body Cookies Headers (4) Test Results  Status: 200 OK

Pretty Raw Preview Visualize JSON  

```
1  [
2    {
3      "title": "BestNew1",
4      "body": "lorem ipsum lalala",
5      "date": "2023-12-25T00:00:00",
6      "authorName": "TheBestAuthor",
7      "rubricName": "Top",
8      "tags": [
9        "TopTag",
10       "SecondTag"
11     ],
12     "id": 2
13   },
14   {
15     "title": "BestNew2",
16     "body": "alalalay",
17     "date": "2023-12-25T00:00:00",
18     "authorName": "TheBestAuthor",
19     "rubricName": "Top",
20     "tags": [
21       "TopTag"
22     ],
23     "id": 3
24   }
25 ]
```

Усі запити в Postman:

<b>POST</b> Register	<b>PUT</b> UpdateTag
<b>POST</b> Login	<b>GET</b> GetTags
<b>GET</b> GetAuthors	<b>GET</b> GetTagById
<b>GET</b> GetAuthorById	<b>DEL</b> DeleteTag
<b>PUT</b> UpdateAuthor	<b>POST</b> AddNews
<b>DEL</b> DeleteAuthor	<b>GET</b> GetNews
<b>POST</b> AddRubric	<b>GET</b> GetNewsById
<b>PUT</b> UpdateRubric	<b>GET</b> GetNewsByAuthorId
<b>GET</b> GetRubrics	<b>GET</b> GetNewsByRubricId
<b>GET</b> GetRubricById	<b>GET</b> GetNewsByTagId
<b>DEL</b> DeleteRubric	<b>PUT</b> UpdateNews
<b>POST</b> AddTag	<b>PUT</b> DeleteNews

## **Відповіді на питання:**

### **1) N Layered та K Tiered архітектура:**

N Layered або K Tiered архітектури - це концепції організації коду в програмах. Вони визначають, як програмна функціональність повинна бути розподілена між різними рівнями або тирами. Наприклад, в N Layered архітектурі можуть бути рівні, такі як Presentation Layer, Business Logic Layer та Data Access Layer. В K Tiered архітектурі можуть бути тир, такі як Presentation Tier, Business Tier та Data Tier. Кожен рівень або тир відповідає за конкретний аспект функціональності програми, що дозволяє підтримувати чистоту коду та легше виявляти помилки.

### **2) Data Access Layer (DAL) в ASP.NET Core з EF Core:**

Data Access Layer в ASP.NET Core зазвичай використовує Entity Framework Core (EF Core) для забезпечення спрощеного доступу до бази даних. Основні компоненти DAL включають DbContext, який визначає сесію з базою даних, а також сутності (Entity Classes), які відображають таблиці бази даних. Використання LINQ дозволяє складати запити до бази даних, а атрибути в EF Core дозволяють визначати відносини між об'єктами та таблицями бази даних.

### **3) Business Logic Layer (BLL) в архітектурі N Layered:**

Business Logic Layer (BLL) в N Layered архітектурі включає в себе бізнес-логіку, що визначає, як система повинна поводитися у визначених сценаріях. Основні принципи роботи включають в себе валідацію даних перед їх передачею до Data Access Layer (DAL), управління бізнес-правилами та інші аспекти логіки, що не пов'язані з безпосереднім доступом до даних чи інтерфейсом користувача.

### **4) REST API в ASP.NET Core:**

REST API в ASP.NET Core реалізується за допомогою контролерів, які містять дії, доступні для виклику з клієнтських додатків. Контролери визначають різні HTTP-методи (GET, POST, PUT, DELETE) за допомогою атрибутів, таких як '[HttpGet]', '[HttpPost]'. Маршрутизація визначає, які URL відповідають конкретним діям контролера.

### **5) Specification Pattern:**

Specification Pattern є шаблоном проектування, який дозволяє визначити критерії для вибору об'єктів в системі. У контексті баз даних він може бути використаний для формулювання складних запитів. Цей паттерн інтегрується в архітектуру рішення, дозволяючи визначати умови запиту як об'єкти, що можуть бути повторно використані та легко комбіновані для складних вибірок.

### **6) NSubstitute для модульного тестування в .NET:**

NSubstitute - це бібліотека для створення фейкових об'єктів у тестах. Вона

дозволяє створювати замітники реальних об'єктів, спрощуючи тестування та дозволяючи контролювати поведінку об'єктів.

7) AutoBogus для генерації тестових даних:

AutoBogus - це бібліотека, яка автоматично генерує фейкові дані для тестування. Вона дозволяє швидко створювати тестові об'єкти з мінімальними зусиллями, що сприяє ефективному тестуванню.

8) Microsoft.DependencyInjection в ASP.NET Core:

Microsoft.DependencyInjection - це вбудований контейнер впровадження залежностей в ASP.NET Core. Він спрощує роботу з залежностями, дозволяючи зареєструвати служби та їх залежності в сервісному контейнері. Це полегшує ін'єкцію залежностей в класи та додає гнучкість до конфігурації служб.

9) Використання Specification Pattern для гнучких запитів до бази даних:

Specification Pattern у контексті баз даних дозволяє створювати гнучкі та повторно використовувані умови запитів. Об'єкти-специфікації описують критерії вибору записів з бази даних, і ці об'єкти можуть бути комбіновані для складних запитів.

10) Dependency Injection в ASP.NET Core:

Dependency Injection (DI) - це паттерн, який дозволяє впровадити залежності в класи. В ASP.NET Core використовується Microsoft.Extensions.DependencyInjection для реалізації DI. Це спрощує управління залежностями та робить код більш тестовим та модульним.

11) Фейкові об'єкти з NSubstitute для модульного тестування в ASP.NET Core:

При модульному тестуванні в ASP.NET Core, NSubstitute може бути використаний для створення фейкових об'єктів, щоб замінити реальні залежності. Це дозволяє тестувати окремі компоненти системи незалежно від решти.

12) Експорт API до Postman:

Експорт API до Postman дозволяє зберігати та документувати API. Postman генерує колекції запитів, які можна легко імпортувати та використовувати для тестування різних ендпоінтів. Це полегшує співпрацю між розробниками та тестувальниками та дозволяє ефективно тестувати та документувати API.

13) Міграції в EF Core:

Міграції в Entity Framework Core (EF Core) дозволяють автоматизовано оновлювати схему бази даних при зміні моделі даних. Це спрощує процес управління базою даних та зберігання структури сумісною з оновленнями програми.

14) Аутентифікація та авторизація в ASP.NET Core REST API:  
Аутентифікація та авторизація в ASP.NET Core REST API використовують системи ідентифікації, такі як JWT. Аутентифікація підтверджує ідентичність користувача, а авторизація визначає його доступ до ресурсів на основі визначених ролей чи політик.

15) Оптимізація масштабованості REST API в ASP.NET Core:  
Для оптимізації продуктивності та масштабованості REST API в ASP.NET Core можна використовувати різні техніки. Це включає в себе використання кешування для зменшення завантаження сервера, використання CDN для швидкої доставки ресурсів, горизонтальне масштабування за допомогою балансувальників навантаження та оптимізацію запитів та відповідей за допомогою асинхронності та інших технік оптимізації коду.