Лабораторна робота №1

**Тема:** Вступ до ASP.NET Core

**Мета:** ознайомитися з основними принципами роботи .NET, навчитися налаштовувати середовище розробки та встановлювати необхідні компоненти, набути навичок створення рішень та проектів різних типів, набути навичок обробки запитів з використанням middleware.

Хід роботи:

# Завдання 1. Встановлення інтегрованого середовища розробки (IDE) та необхідних компонентів

(вже раніше був створений тому пропуск)

Завдання 2. Створення проектів

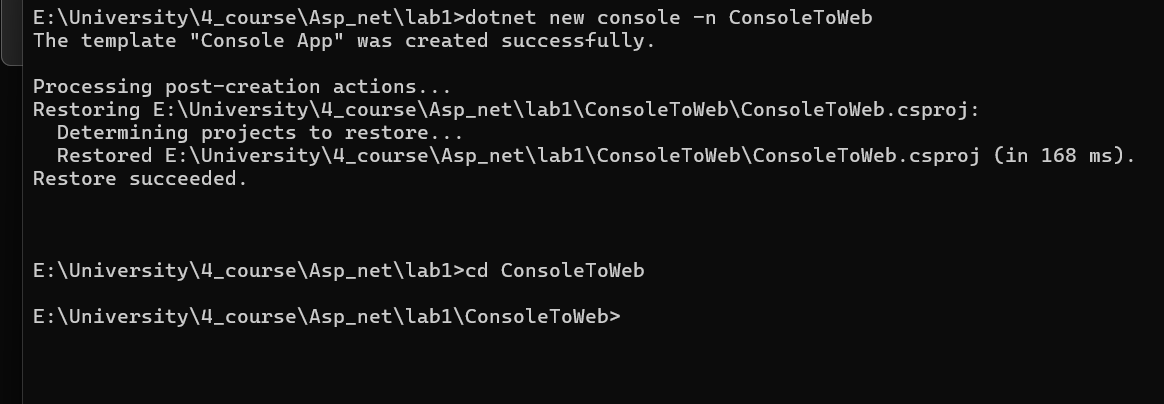
Частина 1.

Створіть проект для консольного додатку з назвою ConsoleToWeb з використанням [dotnet CLI](https://learn.microsoft.com/en-us/dotnet/core/tools/dotnet-new)

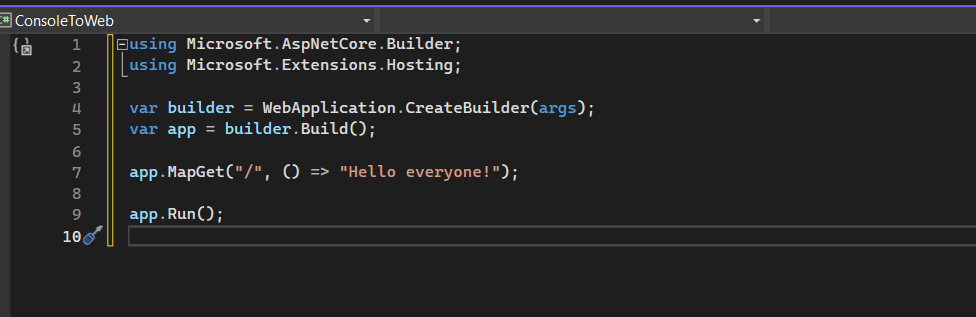
Перетворіть створений консольний додаток на веб-додаток

Опишіть виконані кроки у звіті

1) Створення ConsoleToWeb



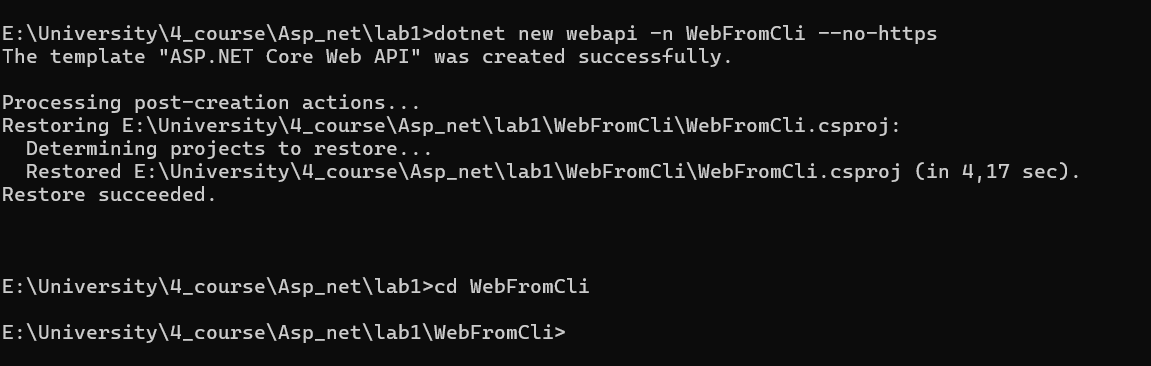
2) Перетворення на веб-додаток



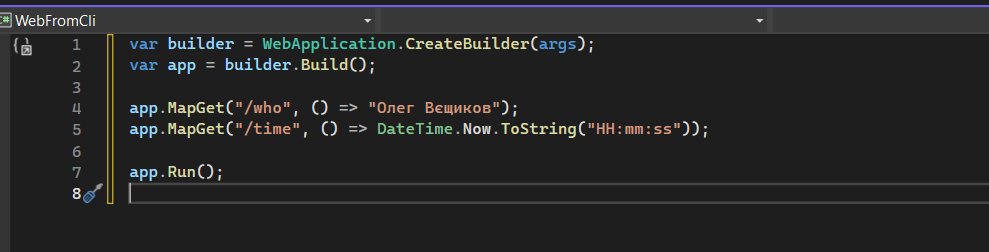
**Частина 2.**

1. Створіть ASP.NET WebAPI проект без авторизації з назвою WebFromCli з використанням [dotnet CLI](https://learn.microsoft.com/en-us/dotnet/core/tools/dotnet-new)
2. Реалізуйте GET-ендпоінт “/who”, який повертатиме у відповідь ваше ім’я та прізвище
3. Реалізуйте GET-ендпоінт “/time”, який повертатиме у відповідь поточний час на сервері
4. Наведіть лістинг реалізованих обробників у звіті

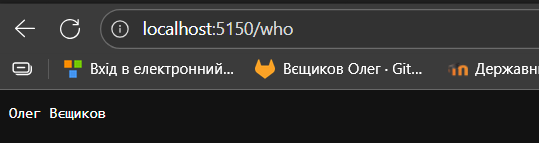
1) Створення WebAPI проекту

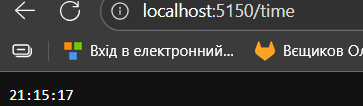


2) Реалізація ендпоінтів



3)

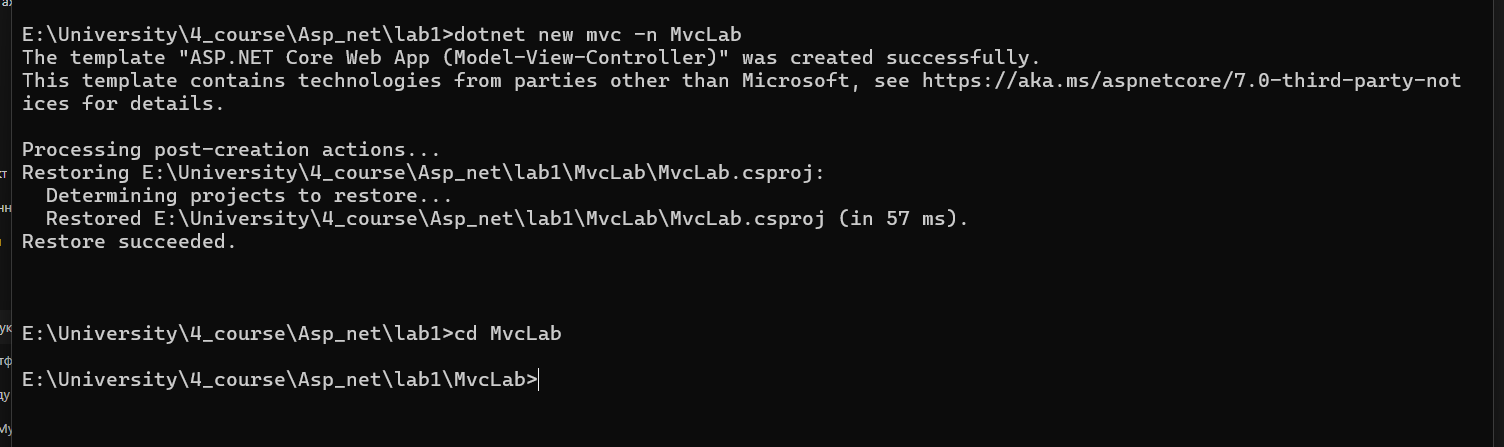




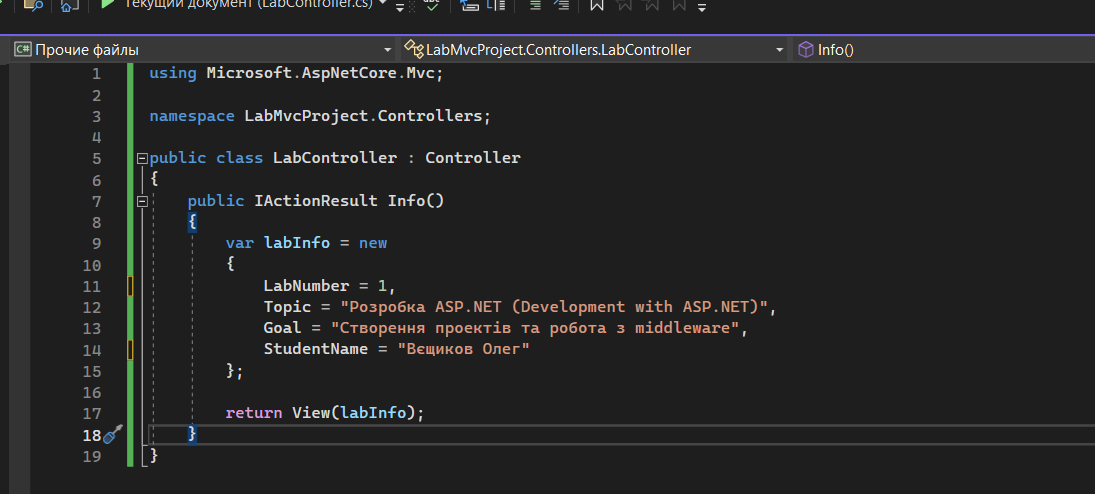
**Частина 3.**

1. Створіть ASP.NET MVC проект будь-яким зручним способом.
2. Реалізуйте контролер з назвою LabController
3. В створеному контролері реалізуйте обробник /info, який повертатиме View з даними про номер лабораторної роботи, тему, мету та ім’я та прізвище виконавця в табличному вигляді
4. Дані для відображення передати з контролера

1) Створення MVC проекту

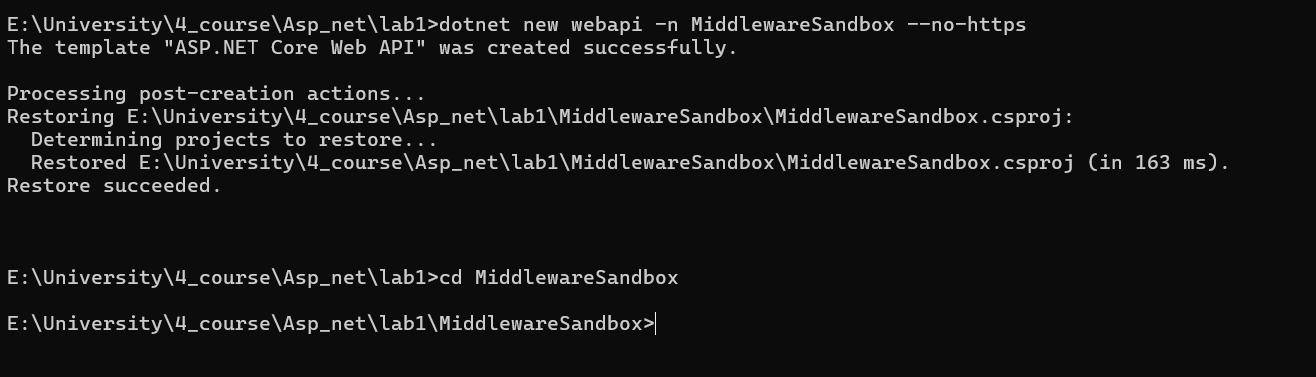


2) Додаємо контролер



Завдання 3. Робота з middleware

1. Створюємо WebAPI проект

  
2) Лістинг програми:

---------------------

3)Перевірка роботи:

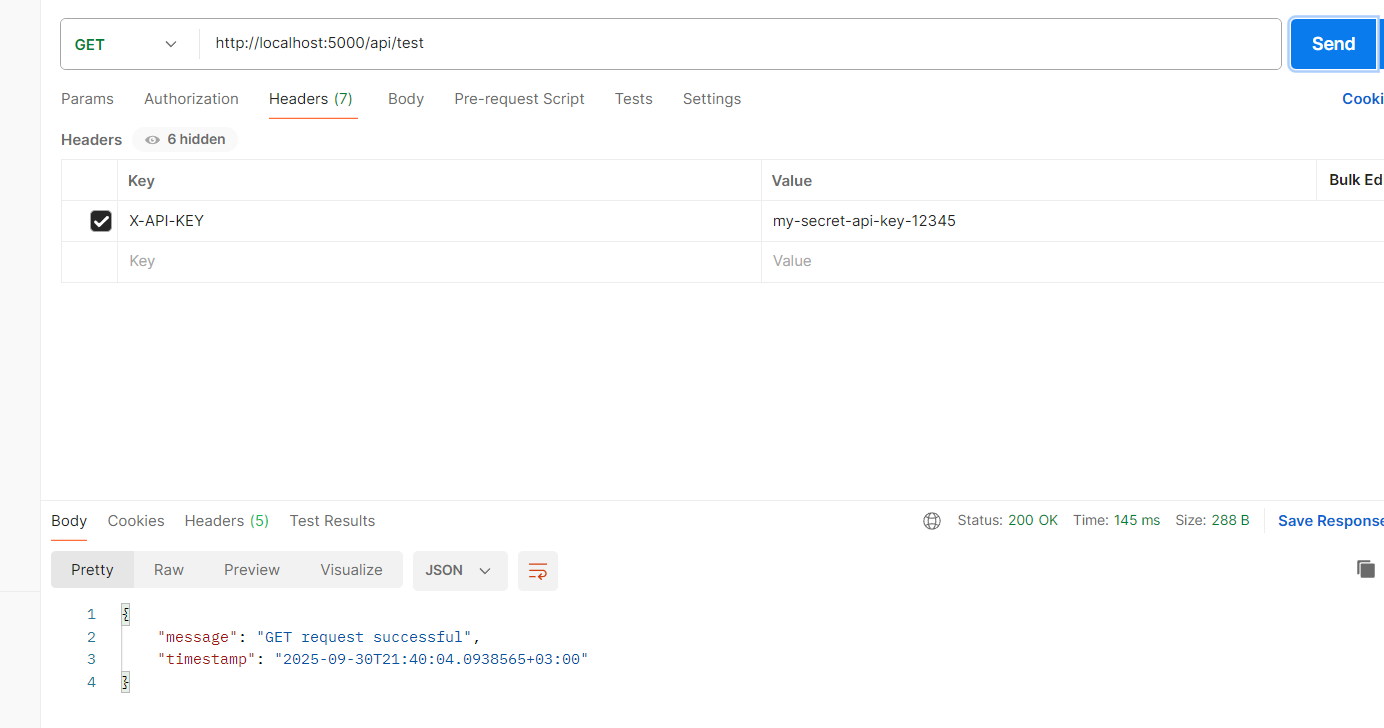
1) Тест CustomParameterMiddleware (без параметра custom)

Запит:

Method: GET

URL: http://localhost:5000/api/test

Headers: X-API-KEY: my-secret-api-key-12345



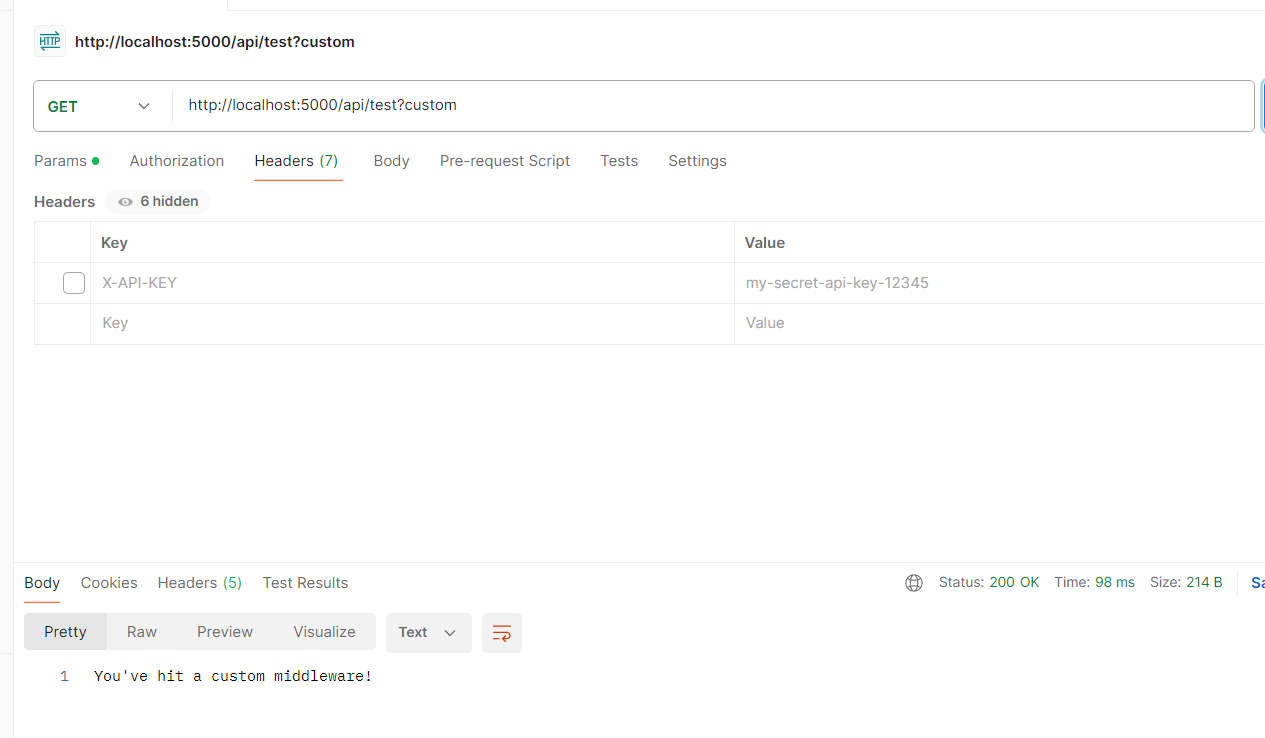
2) Тест CustomParameterMiddleware (параметр ?custom)

Запит:

Method: GET

URL: http://localhost:5000/api/test?custom

Headers: (без заголовків)



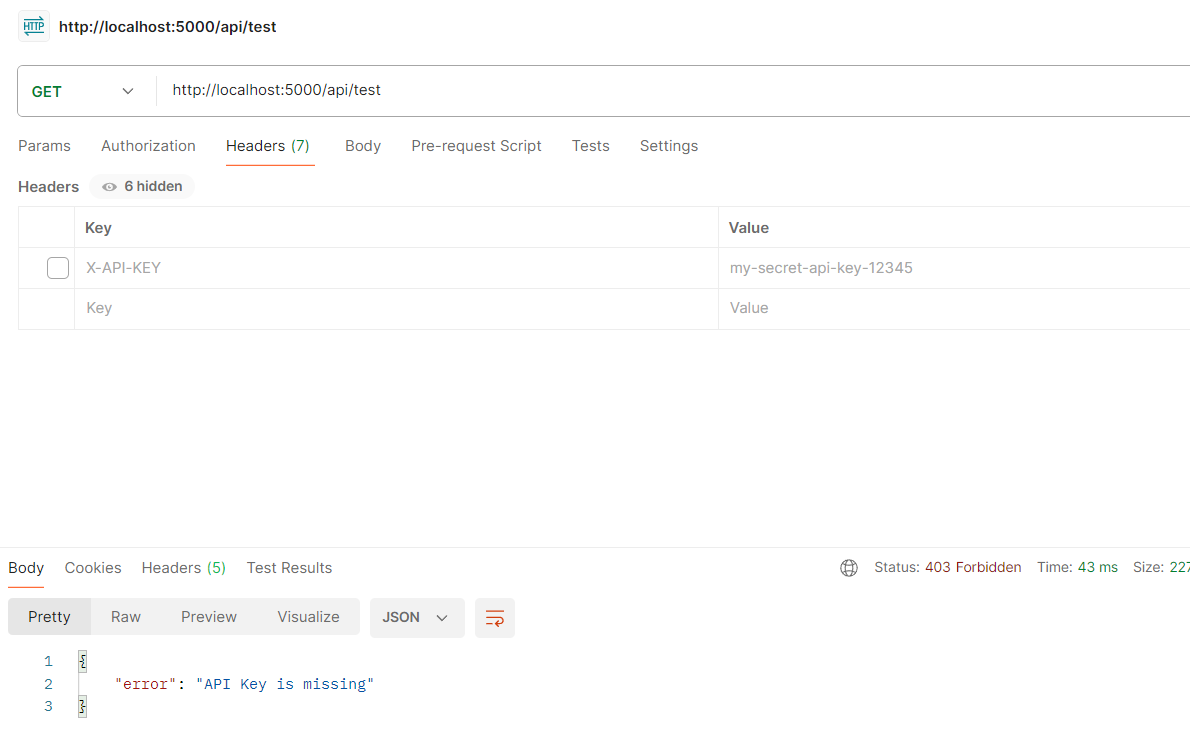
3) Тест ApiKeyValidationMiddleware (без API ключа)

Запит:

Method: GET

URL: http://localhost:5000/api/test

Headers: (без заголовків)



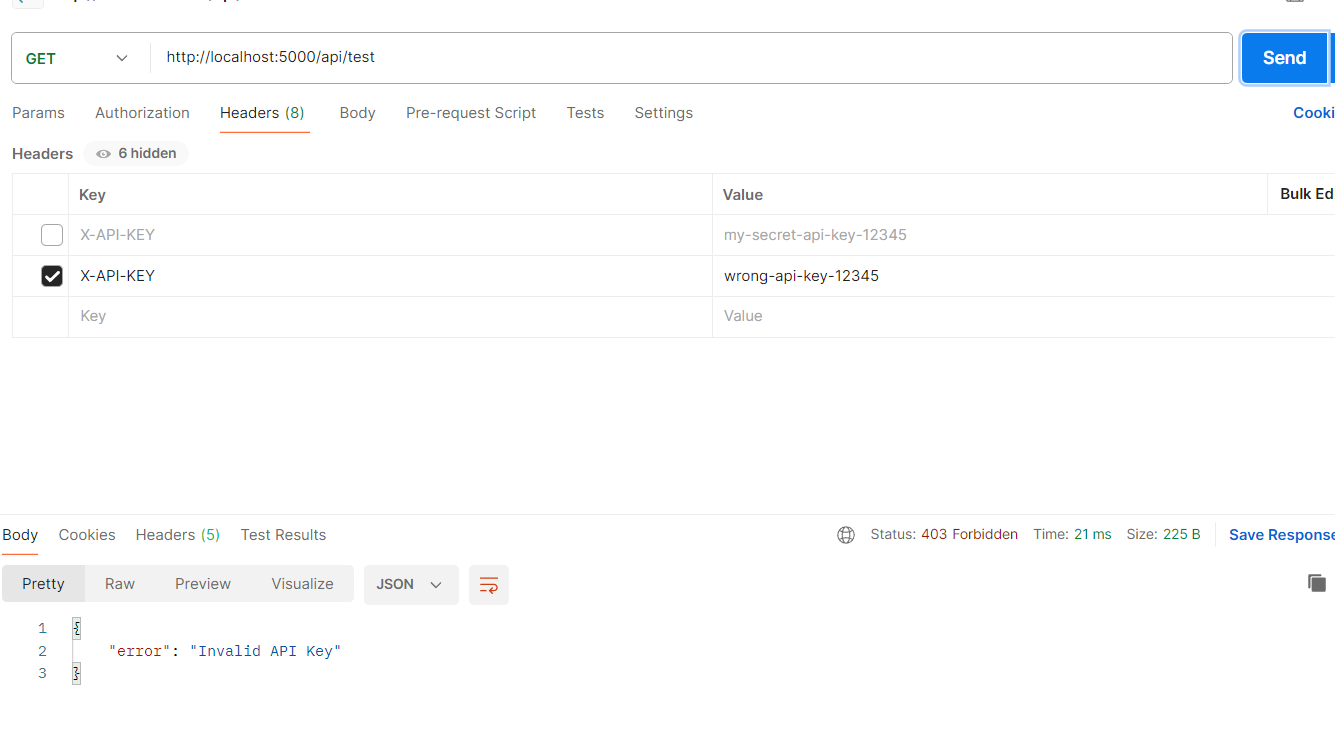
4) Тест ApiKeyValidationMiddleware (невірний API ключ)

Запит:

Method: GET

URL: http://localhost:5000/api/test

Headers:X-API-KEY: wrong-api-key-12345



5) Тест RequestLoggingMiddleware (POST запит)

Запит:

Method: POST

URL: http://localhost:5000/api/test

Headers:X-API-KEY: my-secret-api-key-12345

Content-Type: application/json

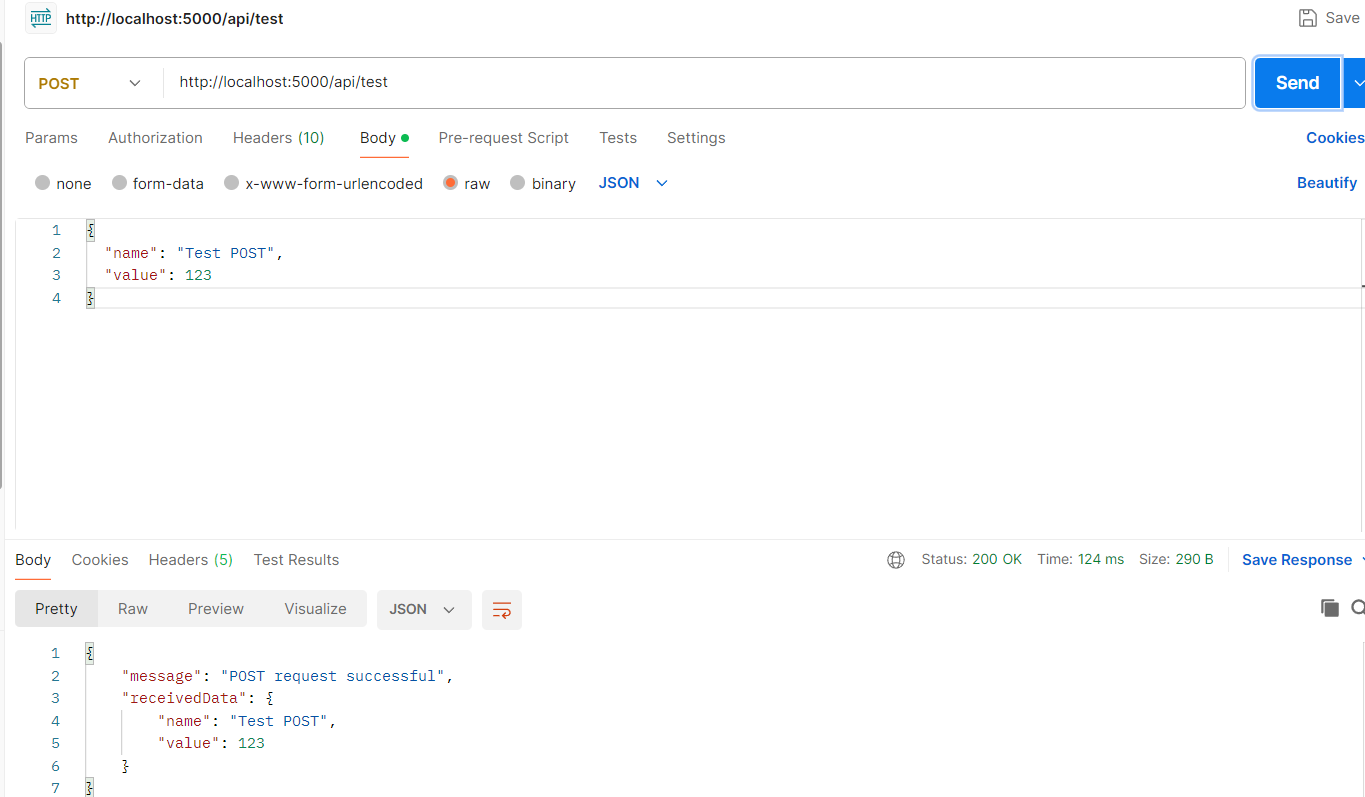
Body (raw JSON):

{

"name": "Test POST",

"value": 123

}



6) Тест RequestLoggingMiddleware (PUT запит)

Запит:

Method: PUT

URL: http://localhost:5000/api/test

Headers: X-API-KEY: my-secret-api-key-12345

Content-Type: application/json

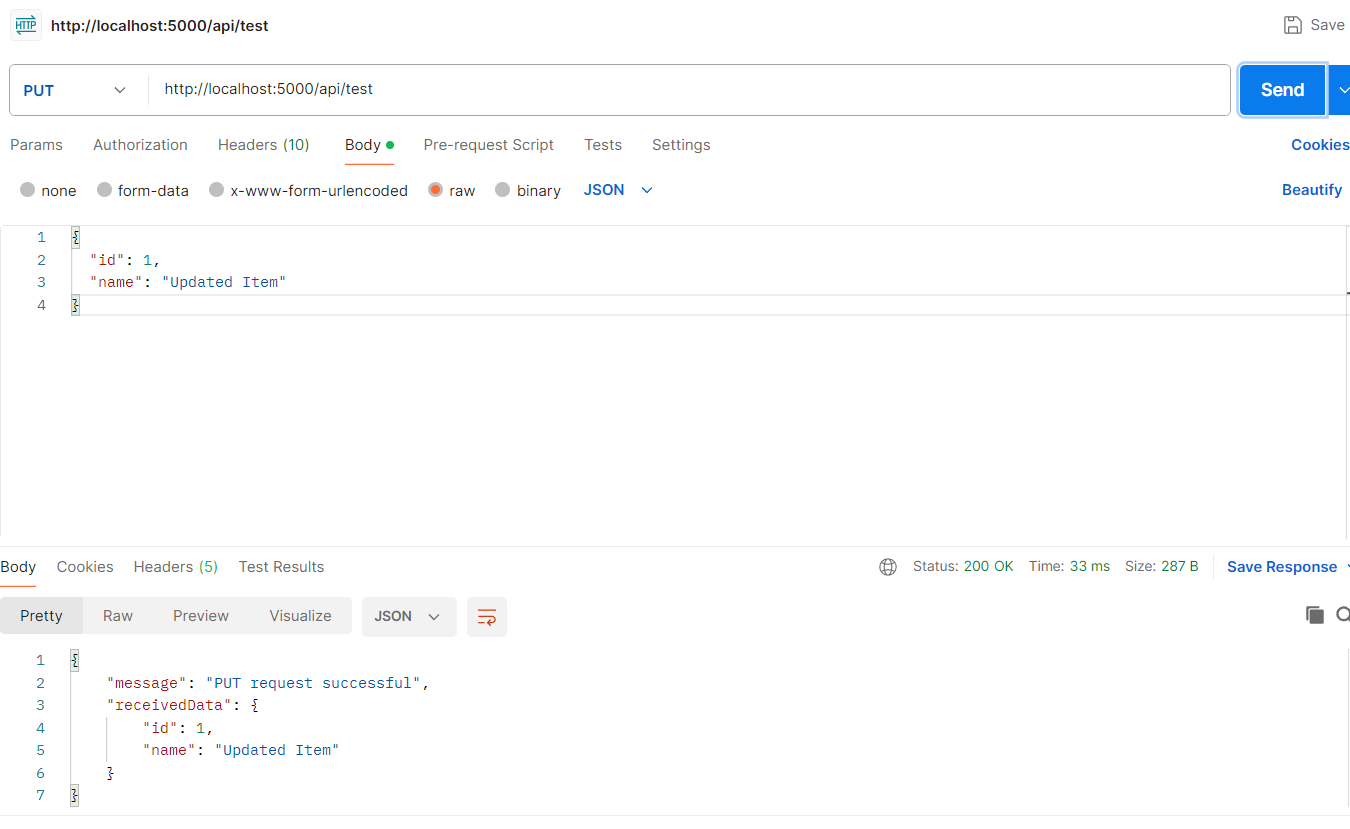
Body (raw JSON):

{

"id": 1,

"name": "Updated Item"

}



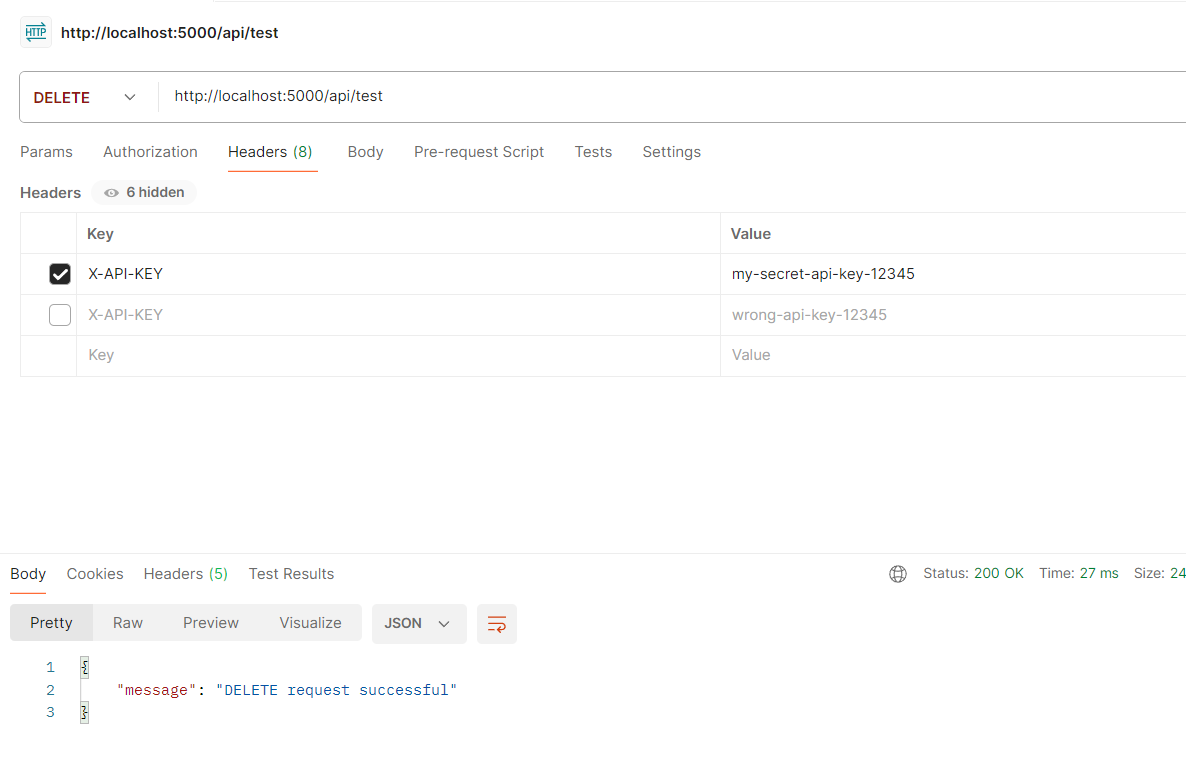
7) Тест RequestLoggingMiddleware (DELETE запит)

Запит:

Method: DELETE

URL: http://localhost:5000/api/test

Headers: X-API-KEY: my-secret-api-key-12345



Висновки:

RequestCounterMiddleware успішно підраховує всі запити до сервера і додає заголовок X-Request-Count до кожної відповіді.

CustomParameterMiddleware правильно аналізує query string параметри і повертає спеціальну відповідь при наявності параметра custom.

RequestLoggingMiddleware коректно логує всі запити (GET, POST, PUT, DELETE) у консоль з відображенням методу, шляху та статусу відповіді.

ApiKeyValidationMiddleware надійно перевіряє наявність та валідність API ключа в заголовку X-API-KEY, блокуючи несанкціоновані запити статусом 403 Forbidden.

Всі middleware працюють коректно як окремо, так і в комбінації, демонструючи правильну реалізацію pipeline обробки запитів у ASP.NET Core.