

Лабораторна робота №1

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

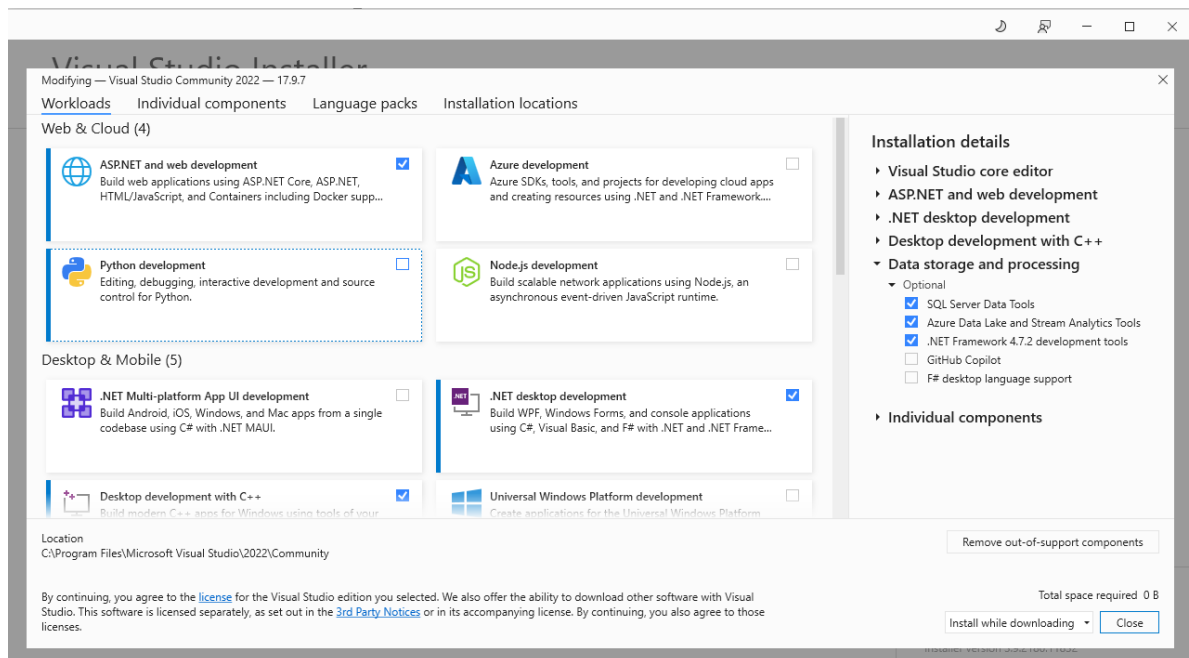


Рис. 1.1. Встановлення інтегрованого середовища розробки

Завдання 2. Створення проектів

Частина 1.

- Створіть проект для консольного додатку з назвою ConsoleToWeb з використанням dotnet CLI
- Перетворіть створений консольний додаток на веб-додаток

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.25.121.29.000 – Лр.1		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Трофімчук М.О.			Звіт з лабораторної роботи №1	Лім.	Арк.
Перевір.		Українець М.О.					1
Реценз.						ФІКТ, ар. ІПЗ-22-2	
Н. Контр.							
Зав.каф.							

3. Опишіть виконані кроки у звіті

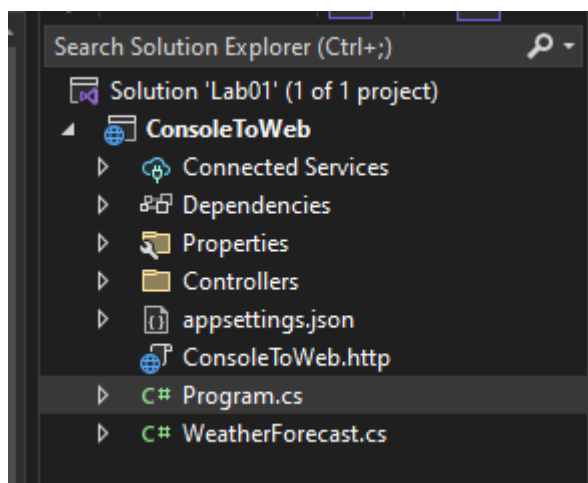


Рис. 2.1.1. Створений веб-додаток

Частина 2.

1. Створіть ASP.NET WebAPI проект без авторизації з назвою WebFromCli з використанням dotnet CLI

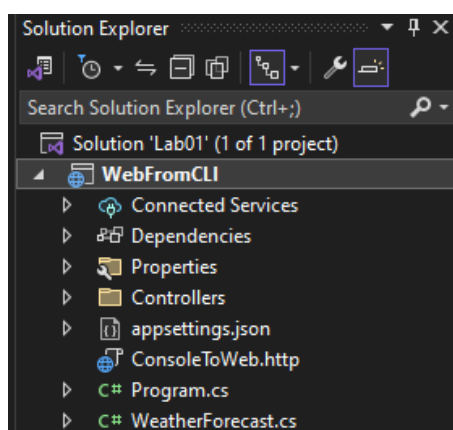


Рис. 2.2.1. Створений ASP.NET WebAPI додаток

2. Реалізуйте GET-ендпоінт “/who”, який повертатиме у відповідь ваше ім’я та прізвище

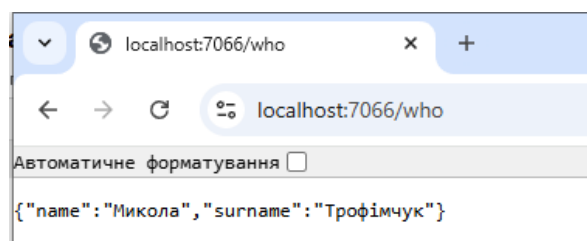


Рис. 2.2.2. GET-ендпоінт “/who”, який повертає у відповідь ім’я та прізвище

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.25.121.29.000 – Лр.1	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		2

3. Реалізуйте GET-ендпоінт “/time”, який повертатиме у відповідь поточний час на сервері

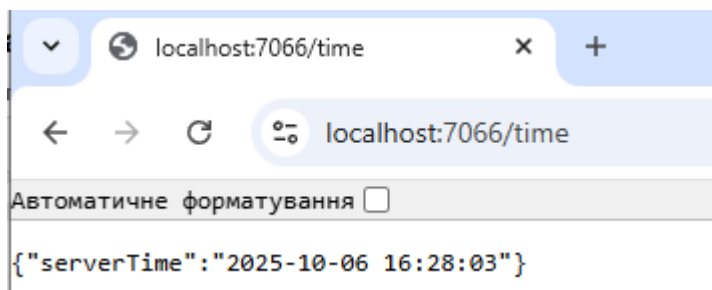


Рис. 2.2.3. GET-ендпоінт “/time”, який повертає у відповідь поточний час на сервері

4. Наведіть лістинг реалізованих обробників у звіті

Лістинг програми:

```
using Microsoft.AspNetCore.Builder;
using Microsoft.Extensions.Hosting;

var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

// --- GET /who ---
app.MapGet("/who", () => new
{
    Name = "Микола",
    Surname = "Трофімчук"
});

// --- GET /time ---
app.MapGet("/time", () => new
{
    ServerTime = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss")
});

app.Run();
```

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

Лістинг програми:

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.25.121.29.000 – Лр.1	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

LabController:

```
using Microsoft.AspNetCore.Mvc;

namespace MvcLabProject.Controllers
{
    public class LabController : Controller
    {
        public IActionResult Info()
        {
            var labData = new
            {
                Number = 1,
                Topic = "Вступ до ASP.NET Core",
                Purpose = "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",
                Author = "Микола Трофімчук ІПЗ-22-2"
            };

            // Передаємо дані через ViewData
            ViewData["Number"] = labData.Number;
            ViewData["Topic"] = labData.Topic;
            ViewData["Purpose"] = labData.Purpose;
            ViewData["Author"] = labData.Author;

            return View();
        }
    }
}
```

View/Lab/Info:

```
@{
    ViewData["Title"] = "Lab Info";
}

<h2 style="text-align:center;">Інформація про лабораторну роботу</h2>

<table border="1" cellpadding="10" cellspacing="0" style="margin:auto; border-collapse: collapse;">
    <tr>
        <th>Номер лабораторної</th>
        <td>@ViewData["Number"]</td>
    </tr>
    <tr>
        <th>Тема</th>
        <td>@ViewData["Topic"]</td>
    </tr>
    <tr>
        <th>Мета</th>
        <td>@ViewData["Purpose"]</td>
    </tr>
    <tr>
        <th>Виконавець</th>
        <td>@ViewData["Author"]</td>
    </tr>
</table>
```

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.25.121.29.000 – Лр.1	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

Результат роботи програми:

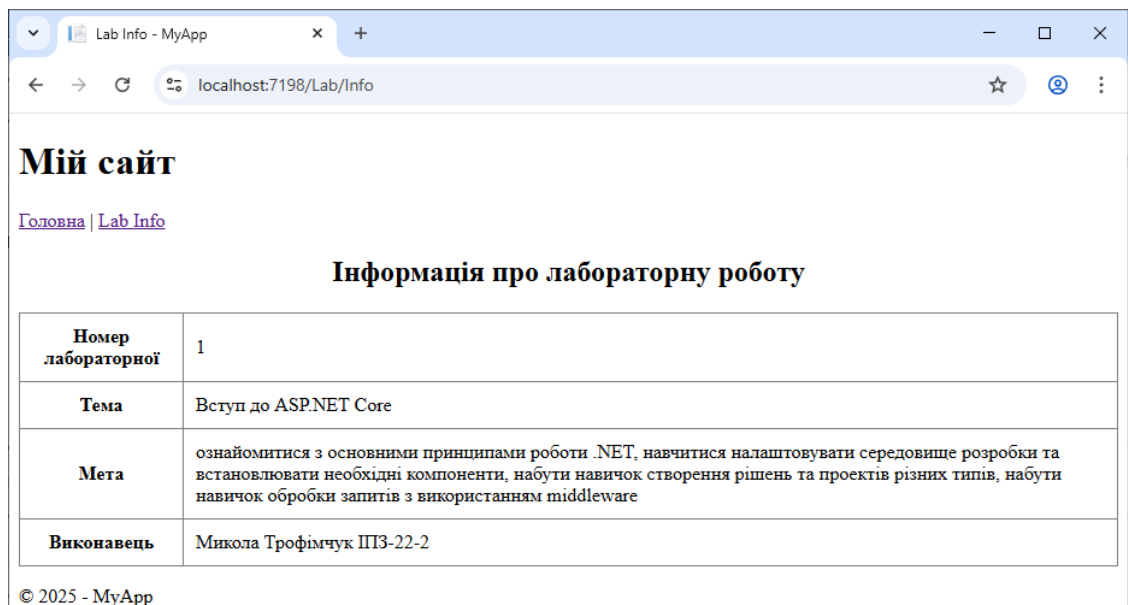


Рис. 2.3.1. Результат роботи програми

Завдання 3. Робота з middleware

1. Ознайомтеся з поняттям middleware в ASP.NET
2. Створіть ASP.NET WebAPI проект з назвою MiddlewareSandbox
3. Реалізуйте наступні middleware
 - a. Реалізуйте middleware, яке рахує кількість запитів до сервера і повертає це число у відповіді, наприклад: The amount of processed requests is X.
 - b. Створіть middleware, яке аналізує параметри рядка запиту. Якщо в рядку запиту (query string) присутній параметр “custom”, то повертати у відповідь “You’ve hit a custom middleware!”, інакше пропускати запит далі. Приклад запиту зображено на рисунку 3.1.

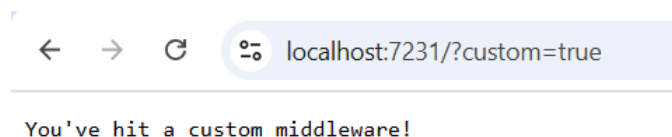


Рис. 3.1. Приклад виконання запиту

- c. Створіть middleware, яке логуватиме у консоль інформацію про всі запити. В логах відображати метод запиту (GET, POST тощо) та

його шлях (/user, /product тощо). Для перевірки POST та інших методів запити можна скористатися Postman або іншою подібною утилітою.

с. Реалізуйте middleware, яке перевіряє наявність API-ключа у заголовках запити. Для цього слід Перевіряти, чи є в запиті заголовок X-API-KEY. Якщо ключ неправильний або відсутній, повертати 403 Forbidden, не передавати запит для виконання далі. Якщо ключ в заголовку запити співпадає з тим, який заданий на сервері, передавати запит далі. Для додавання відповідного запити можна скористатися Postman або подібними утилітами.

Лістинг програми:

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Hosting;
using System.Threading.Tasks;

var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

// --- Middleware c: Логуювання всіх запитів ---
app.Use(async (context, next) =>
{
    Console.WriteLine($"[LOG] Method: {context.Request.Method}, Path:
    {context.Request.Path}");
    await next();
});

// --- Middleware b: Перевірка параметра query string "custom" ---
app.Use(async (context, next) =>
{
    if (context.Request.Query.ContainsKey("custom"))
    {
        await context.Response.WriteAsync("You've hit a custom
        middleware!");
    }
    else
    {
        await next();
    }
});

// --- Middleware d: Перевірка API ключа ---
const string apiKey = "12345"; // Ключ на сервері
app.Use(async (context, next) =>
{
    if (!context.Request.Headers.TryGetValue("X-API-KEY", out var
    extractedApiKey) || extractedApiKey != apiKey)
    {
        context.Response.StatusCode = 403; // Forbidden
        await context.Response.WriteAsync("Forbidden: invalid or missing
        API key.");
        return;
    }
    await next();
});
```

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.25.121.29.000 – Лр.1	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6

```
// --- Middleware a: Лічильник запитів ---
int requestCount = 0;
app.Use(async (context, next) =>
{
    requestCount++;
    await next();
});

// --- Простий GET ендпоінт для тесту ---
app.MapGet("/test", async context =>
{
    await context.Response.WriteAsync($"The amount of processed requests is {requestCount}");
});

app.Run();
```

Результат роботи програми:

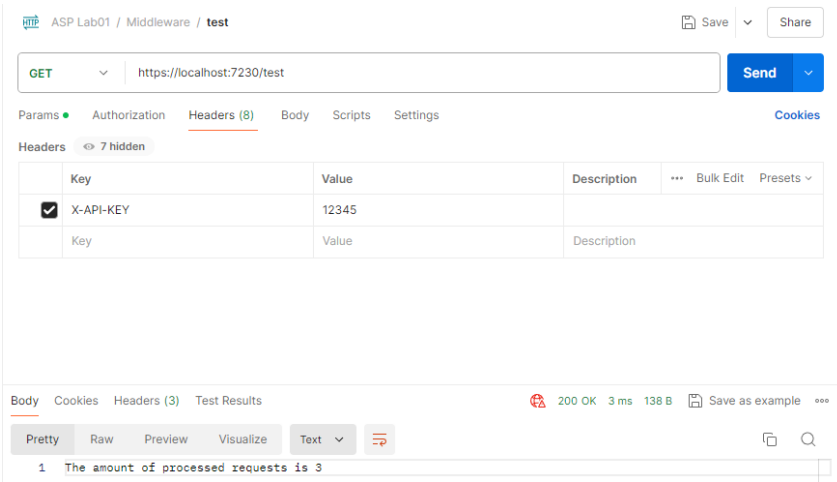


Рис. 3.1.1. middleware, яке рахує кількість запитів до сервера і повертає це число у відповіді

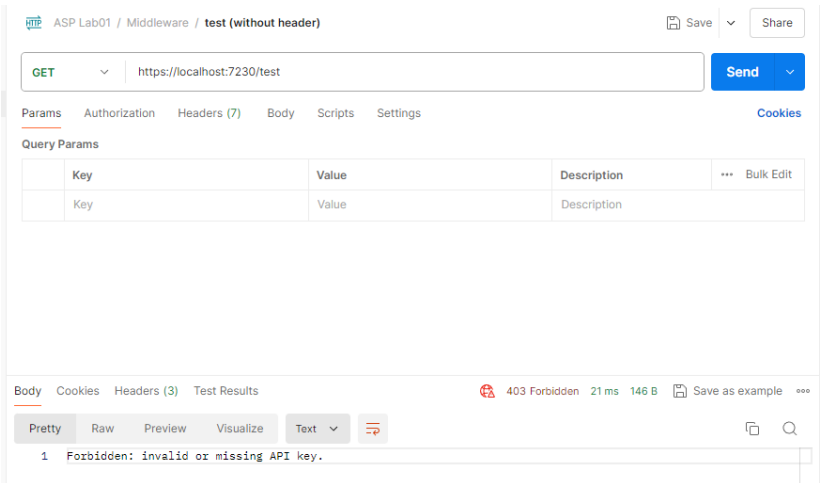


Рис. 3.1.2. middleware, яке перевіряє наявність API-ключа у заголовках запиту

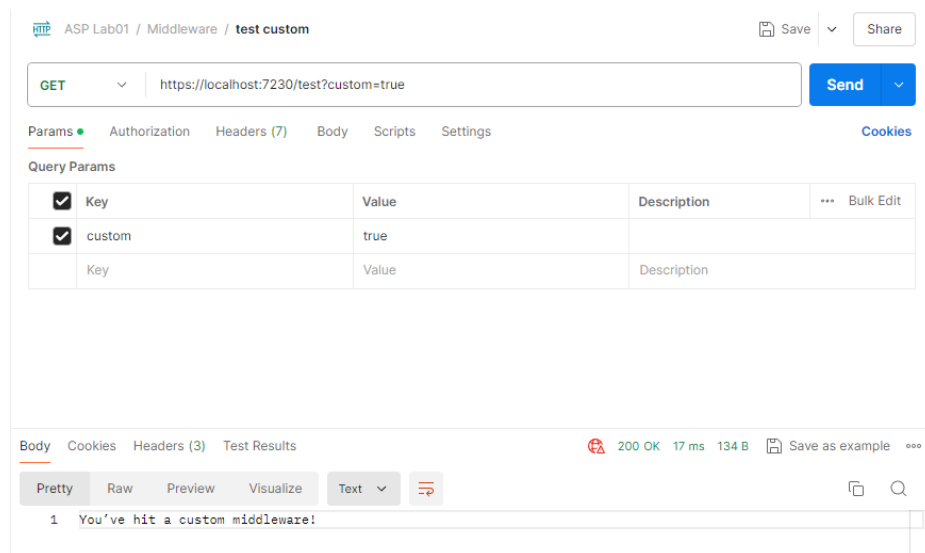


Рис. 3.1.3. middleware, яке аналізує параметри рядка запиту

```

D:\4 курс\ASP Net\Lab01\MiddlewareSandbox\bin\Debug\net8.0\MiddlewareSandbox.exe
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7230
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5293
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: D:\4 курс\ASP Net\Lab01\MiddlewareSandbox
[LOG] Method: GET, Path: /swagger
[LOG] Method: GET, Path: /favicon.ico
[LOG] Method: GET, Path: /test
[LOG] Method: GET, Path: /test
[LOG] Method: GET, Path: /test
[LOG] Method: GET, Path: /test
[LOG] Method: GET, Path: /test
[LOG] Method: GET, Path: /test

```

Рис. 3.1.4. middleware, яке логує у консоль інформацію про всі запити

Висновок: в ході виконання лабораторної роботи було ознайомлено з основними принципами роботи .NET, навчено налаштовувати середовище розробки та встановлювати необхідні компоненти, набуто навичок створення рішень та проектів різних типів, набуто навичок обробки запитів з використанням middleware. Роботу виконано в повному обсязі.