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

Перетворіть створений консольний додаток на веб-додаток

Опишіть виконані кроки у звіті

1) Створення ConsoleToWeb

```
E:\University\4_course\Asp_net\lab1>dotnet new console -n ConsoleToWeb
The template "Console App" was created successfully.

Processing post-creation actions...
Restoring E:\University\4_course\Asp_net\lab1\ConsoleToWeb\ConsoleToWeb.csproj:
  Determining projects to restore...
  Restored E:\University\4_course\Asp_net\lab1\ConsoleToWeb\ConsoleToWeb.csproj (in 168 ms).
Restore succeeded.



E:\University\4_course\Asp_net\lab1>cd ConsoleToWeb

E:\University\4_course\Asp_net\lab1\ConsoleToWeb>
```

2) Перетворення на веб-додаток

```
ConsoleToWeb
1  using Microsoft.AspNetCore.Builder;
2  using Microsoft.Extensions.Hosting;
3
4  var builder = WebApplication.CreateBuilder(args);
5  var app = builder.Build();
6
7  app.MapGet("/", () => "Hello everyone!");
8
9  app.Run();
10
```

Частина 2.

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

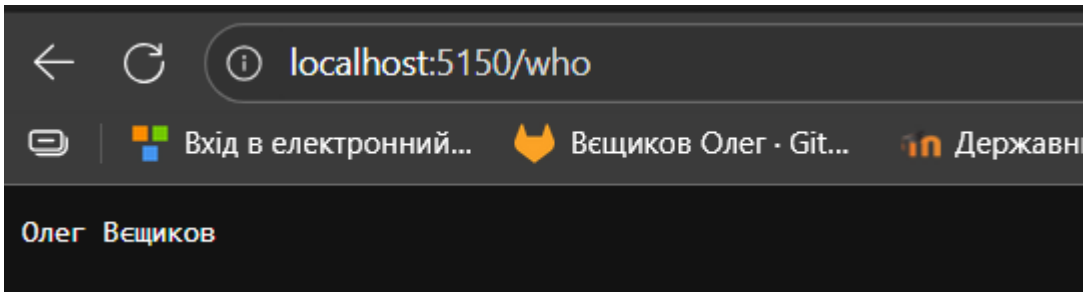
1) Створення WebAPI проекту

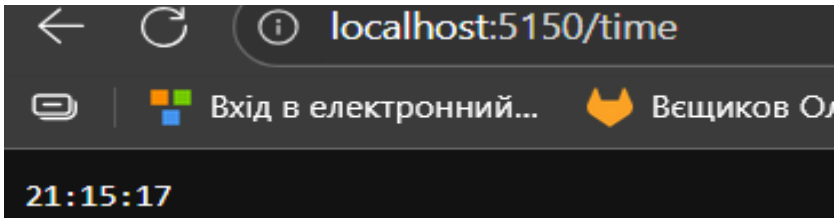
```
E:\University\4_course\Asp_net\lab1>dotnet new webapi -n WebFromCli --no-https
The template "ASP.NET Core Web API" was created successfully.

Processing post-creation actions...
Restoring E:\University\4_course\Asp_net\lab1\WebFromCli\WebFromCli.csproj:
  Determining projects to restore...
  Restored E:\University\4_course\Asp_net\lab1\WebFromCli\WebFromCli.csproj (in 4,17 sec).
Restore succeeded.



E:\University\4_course\Asp_net\lab1>cd WebFromCli

E:\University\4_course\Asp_net\lab1\WebFromCli>
```

2) Реалізація ендпоінтів

```
WebFromCli
1  var builder = WebApplication.CreateBuilder(args);
2  var app = builder.Build();
3
4  app.MapGet("/who", () => "Олег Вєщиков");
5  app.MapGet("/time", () => DateTime.Now.ToString("HH:mm:ss"));
6
7  app.Run();
8
```

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

```
E:\University\4_course\Asp_net\lab1>dotnet new mvc -n MvcLab
The template "ASP.NET Core Web App (Model-View-Controller)" was created successfully.
This template contains technologies from parties other than Microsoft, see https://aka.ms/aspnetcore/7.0-third-party-not
ices for details.

Processing post-creation actions...
Restoring E:\University\4_course\Asp_net\lab1\MvcLab\MvcLab.csproj:
  Determining projects to restore...
  Restored E:\University\4_course\Asp_net\lab1\MvcLab\MvcLab.csproj (in 57 ms).
Restore succeeded.


E:\University\4_course\Asp_net\lab1>cd MvcLab

E:\University\4_course\Asp_net\lab1\MvcLab>
```

2) Додаємо контролер

```
using Microsoft.AspNetCore.Mvc;

namespace LabMvcProject.Controllers;

public class LabController : Controller
{
    public IActionResult Info()
    {
        var labInfo = new
        {
            LabNumber = 1,
            Topic = "Розробка ASP.NET (Development with ASP.NET)",
            Goal = "Створення проектів та робота з middleware",
            StudentName = "Вещиков Олег"
        };

        return View(labInfo);
    }
}
```

Завдання 3. Робота з middleware

1) Створюємо WebAPI проект

```
E:\University\4_course\Asp_net\lab1>dotnet new webapi -n MiddlewareSandbox --no-https
The template "ASP.NET Core Web API" was created successfully.

Processing post-creation actions...
Restoring E:\University\4_course\Asp_net\lab1\MiddlewareSandbox\MiddlewareSandbox.csproj:
  Determining projects to restore...
  Restored E:\University\4_course\Asp_net\lab1\MiddlewareSandbox\MiddlewareSandbox.csproj (in 163 ms).
Restore succeeded.



E:\University\4_course\Asp_net\lab1>cd MiddlewareSandbox

E:\University\4_course\Asp_net\lab1\MiddlewareSandbox>
```

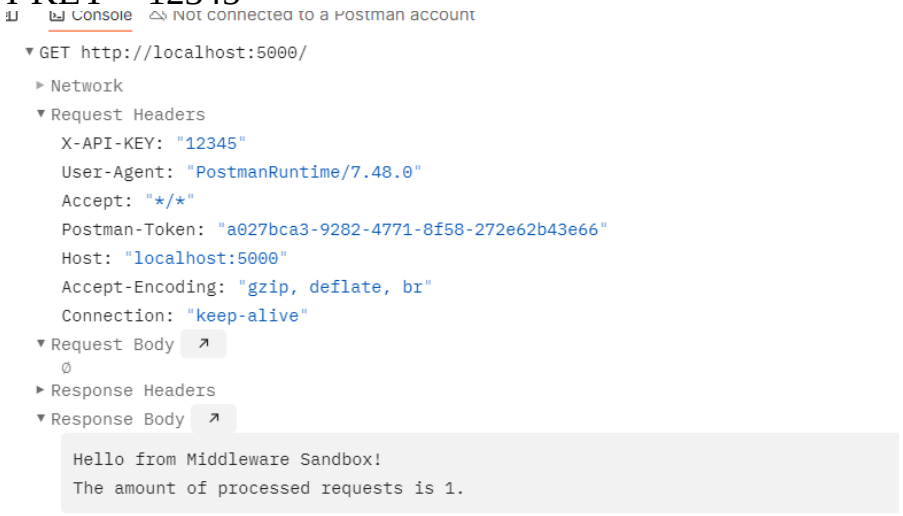
2) Лістинг програми:

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

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

int requestCount = 0;
string validApiKey = "12345";

// Middleware логування
app.Use(async (context, next) =>
{
    Console.WriteLine($"Request: {context.Request.Method} {context.Request.Path}");
    await next.Invoke();
});

// Middleware перевірки API-ключа
app.Use(async (context, next) =>
{
    if (!context.Request.Headers.TryGetValue("X-API-KEY", out var apiKey) || apiKey !=
validApiKey)
    {
        context.Response.StatusCode = 403;
        await context.Response.WriteAsync("Forbidden: Invalid or missing API key.");
        return;
    }
    await next.Invoke();
});

// Middleware аналізу query string
app.Use(async (context, next) =>
{
    if (context.Request.Query.ContainsKey("custom"))
    {
        await context.Response.WriteAsync("You've hit a custom middleware!");
        return;
    }
    await next.Invoke();
});

// Middleware підрахунку запитів
app.Use(async (context, next) =>
```

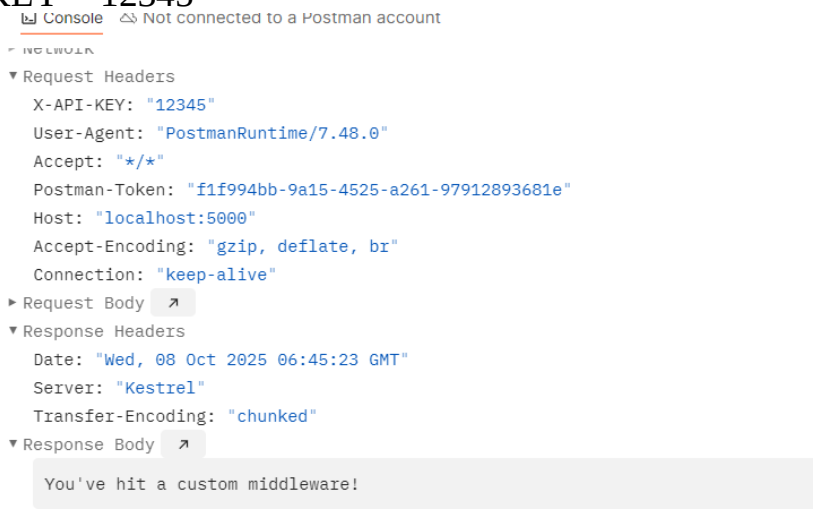
```
{
    requestCount++;
    await next.Invoke();
    await context.Response.WriteAsync($"\nThe amount of processed requests is
{requestCount}.");
});

app.MapGet("/", () => "Hello from Middleware Sandbox!");

app.Run();
```

Результат:

Звичайний GET-запит
GET http://localhost:5000/
Header: X-API-KEY = 12345

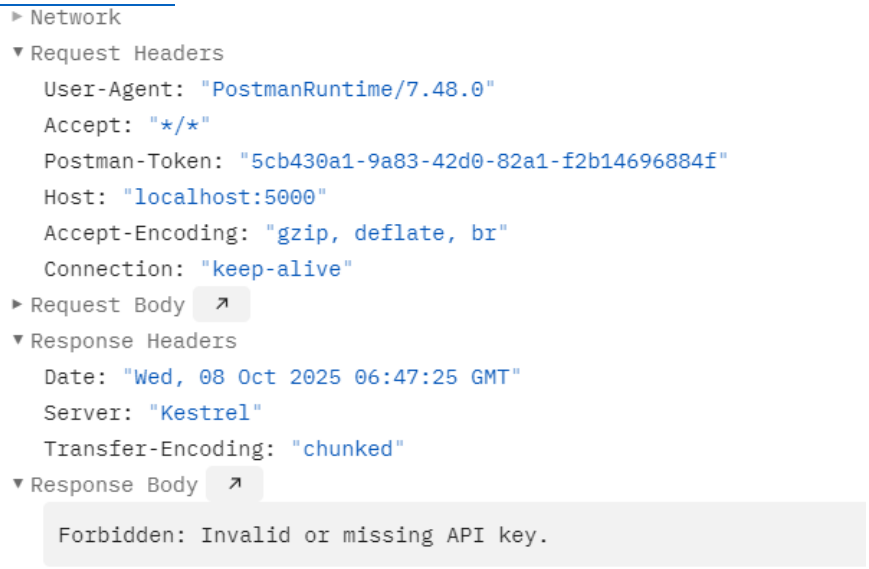


Запит із параметром custom у рядку запиту
GET http://localhost:5000/?custom=1
Header: X-API-KEY = 12345



Middleware аналізує query string і бачить параметр custom. У такому випадку інші middleware не виконуються, а відповідь

Запит без API-ключа
GET http://localhost:5000/



Middleware перевіряє наявність заголовка X-API-KEY.

Оскільки ключ відсутній, сервер повертає: Forbidden: Invalid or missing API key.

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.