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.

Завдання 2. Створення проектів

Частина 1.

1. Створіть проект для консольного додатку з назвою ConsoleToWeb з використанням dotnet CLI
2. Перетворіть створений консольний додаток на веб-додаток
3. Опишіть виконані кроки у звіті

Щоб створити консольний додаток за допомогою dotnet CLI, можна скористатись командою dotnet new console -n <назва проєкту>. Переходимо до новоствореної директорії з проєктом та додаємо необхідні для веб-додатку залежності командою dotnet add package Microsoft.AspNetCore.App. Відкриваємо файл Program.cs та реалізуємо базовий веб-додаток: створюємо builder, «білдимо» додаток, та додаємо потрібні ендпоінти.

Лістинг Program.cs:

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

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

app.MapGet("/", () => "Hello from Web App!");

app.Run();
```

Результат:

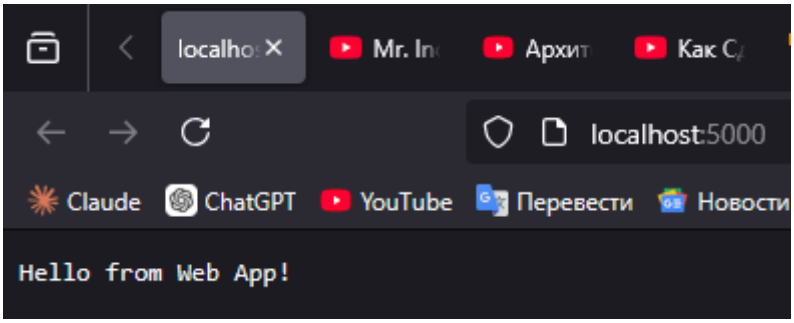

Рис. 1 Результат

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

Лістинг NameEndpoint.cs:

```
namespace WebFromCli.WebApi
{
    public class NameEndpoint
    {
        public static async Task Endpoint(HttpContext context)
        {
            await context.Response.WriteAsync("Rizhenko Jan");
        }
    }
}
```

Лістинг TimeEndpoint.cs

```
namespace WebFromCli.WebApi
{
    public class TimeEndpoint
    {
        public static async Task Endpoint(HttpContext context)
        {
```

```
                await context.Response.WriteAsync(DateTime.Now.ToString());
            }
        }
    }
```

Лістинг Program.cs:

```
using WebFromCli.WebApi;

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

app.MapGet("/who", NameEndpoint.Endpoint);
app.MapGet("/time", TimeEndpoint.Endpoint);

app.Run();
```

Результат:

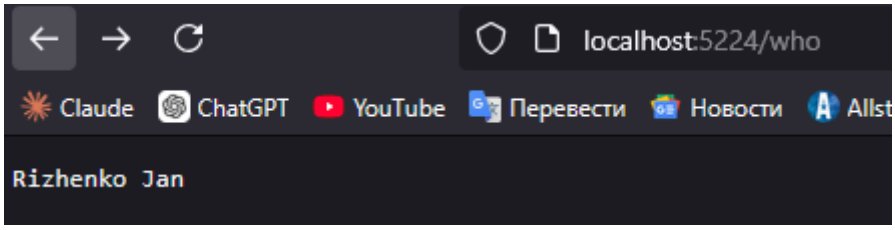

Рис. 2 Get-ендпоінт /who

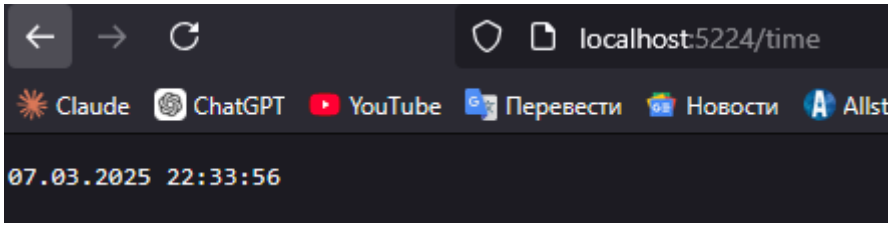

Рис. 3 Get-ендпоінт /time

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 LabController.cs:

```
using Microsoft.AspNetCore.Mvc;
using MvcProject.Models;

public class LabController : Controller
{
    public IActionResult Info()
    {
        var model = new LabInfoModel
        {
```

```
            LabNumber = "Лабораторна №1",
            Topic = "Вступ до ASP.NET Core",
            Goal = "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.",
            StudentName = "Ян Риженко"
        };
        return View(model);
    }
}
```

Лістинг LabInfoModel.cs:

```
namespace MvcProject.Models
{
    public class LabInfoModel
    {
            public string LabNumber { get; set; }
            public string Topic { get; set; }
            public string Goal { get; set; }
            public string StudentName { get; set; }
    }
}
```

Лістинг Info.cshtml:

```
@model LabInfoModel

<table border="1">
    <tr><th>Номер лабораторної</th><td>@Model.LabNumber</td></tr>
    <tr><th>Тема</th><td>@Model.Topic</td></tr>
    <tr><th>Мета</th><td>@Model.Goal</td></tr>
    <tr><th>Виконавець</th><td>@Model.StudentName</td></tr>
      </table>
```

Лістинг Program.cs:

```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();

var app = builder.Build();

if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseRouting();

app.UseAuthorization();

app.MapStaticAssets();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Lab}/{action=Info}/{id?}")
    .WithStaticAssets();


app.Run();
```

Результат:

Номер лабораторної	Лабораторна №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.
Виконавець	Ян Риженко

Рис. 4 Результат

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

1. Реалізуйте middleware, яке рахує кількість запитів до сервера і повертає це число у відповіді, наприклад: The amount of processed requests is X.

2. 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.

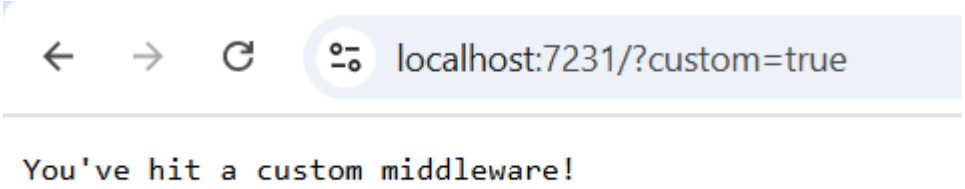

Рис. 5. Приклад виконання запиту

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 запиту можна скористатися Postman або іншою подібною утилітою.

c. Реалізуйте middleware, яке перевіряє наявність API-ключа у заголовках запиту. 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 запиту співпадає з тим, який заданий на сервері, передавати запит далі. Для додавання відповідного запиту можна скористатися Postman або подібними утилітами.

Лістинг CounterMiddleware.cs:

```
using static System.Net.Mime.MediaTypeNames;

namespace MiddlewareSandbox.Middlewares
{
    public class CounterMiddleware
    {
        private RequestDelegate _next;
        private int _counter = 0;

        public CounterMiddleware(RequestDelegate next)
        {
            _next = next;
        }

        public async Task Invoke(HttpContext ctx)
        {
            _counter++;
            await _next(ctx);
            await ctx.Response.WriteAsync($"\nThe amount of processed requests is:
{_counter}\n");
        }
    }
}
```

Лістинг CustomMiddleware.cs:

```
using System.Xml.Linq;
using static System.Net.Mime.MediaTypeNames;

namespace MiddlewareSandbox.Middlewares
{
    public class CustomMiddleware
    {
        private RequestDelegate _next;
        public CustomMiddleware(RequestDelegate next)
        {
            _next = next;
        }

       public async Task Invoke(HttpContext ctx)
           {
           if (ctx.Request.Query.ContainsKey("custom"))
           {
                await ctx.Response.WriteAsync("You've hit a custom middleware!");
           }
           else
           {
                await _next(ctx);
           }
       }
    }
}
```

Лістинг LoggingMiddleware.cs:

```
using static System.Net.Mime.MediaTypeNames;

namespace MiddlewareSandbox.Middlewares
{
    public class LoggingMiddleware
    {
        private RequestDelegate _next;
        private ILogger<LoggingMiddleware> _logger;


        public LoggingMiddleware(RequestDelegate next, ILogger<LoggingMiddleware> logger)
        {
            _next = next;
            _logger = logger;
        }

        public async Task Invoke(HttpContext ctx)
        {
            _logger.LogInformation($"Request: {ctx.Request.Method} {ctx.Request.Path}");
            await _next(ctx);
        }
    }
}
```

Лістинг CheckApiKeyMiddleware.cs:

```
namespace MiddlewareSandbox.Middlewares
{
    public class CheckApiKeyMiddleware
    {
        private readonly RequestDelegate _next;
        private readonly string _validApiKey = "my-secret-key";

        public CheckApiKeyMiddleware(RequestDelegate next)
        {
            _next = next;
        }

        public async Task Invoke(HttpContext ctx)
        {
            if (!ctx.Request.Headers.TryGetValue("X-API-KEY", out var apiKey) || apiKey
!= _validApiKey)
            {
                ctx.Response.StatusCode = StatusCodes.Status403Forbidden;
                await ctx.Response.WriteAsync("Forbidden: Invalid API Key");
                return;
            }

            await _next(ctx);
        }
    }
}
```

Лістинг Program.cs:

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using MiddlewareSandbox.Middlewares;
using System.Threading.Tasks;

var builder = WebApplication.CreateBuilder(args);
```

```
var app = builder.Build();


app.UseMiddleware<LoggingMiddleware>();
app.UseMiddleware<CheckApiKeyMiddleware>();
app.UseMiddleware<CounterMiddleware>();
app.UseMiddleware<CustomMiddleware>();

app.Run();
```

Результат:

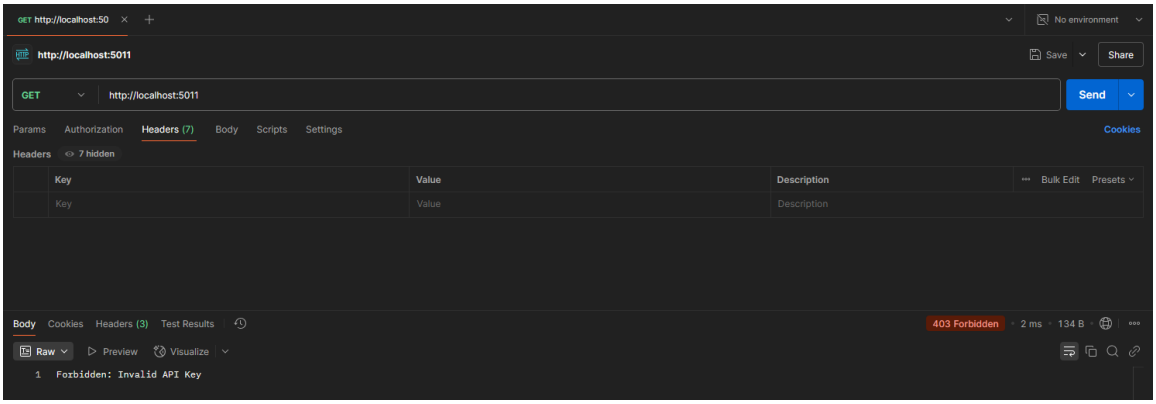

Рис. 5 Запит без наявного api-ключа

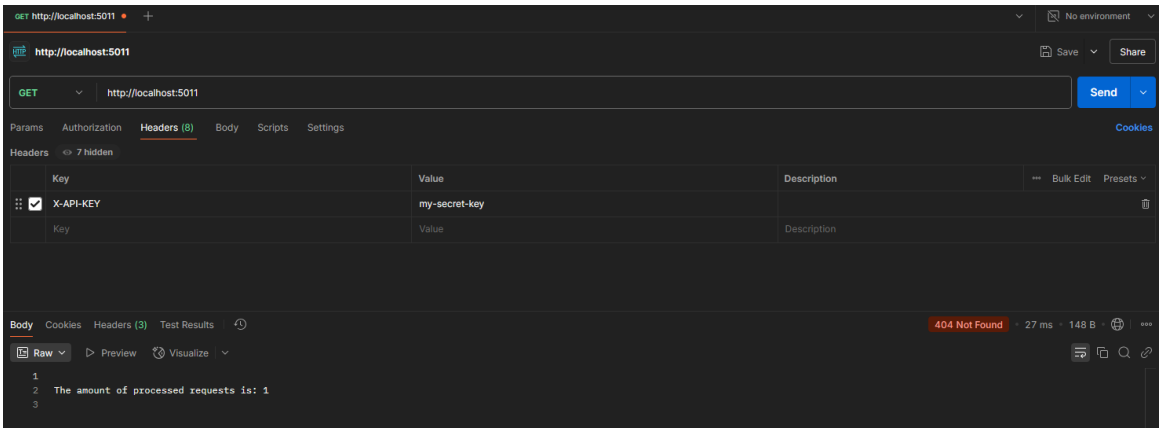

Рис. 6 Запит з наявним api-ключем

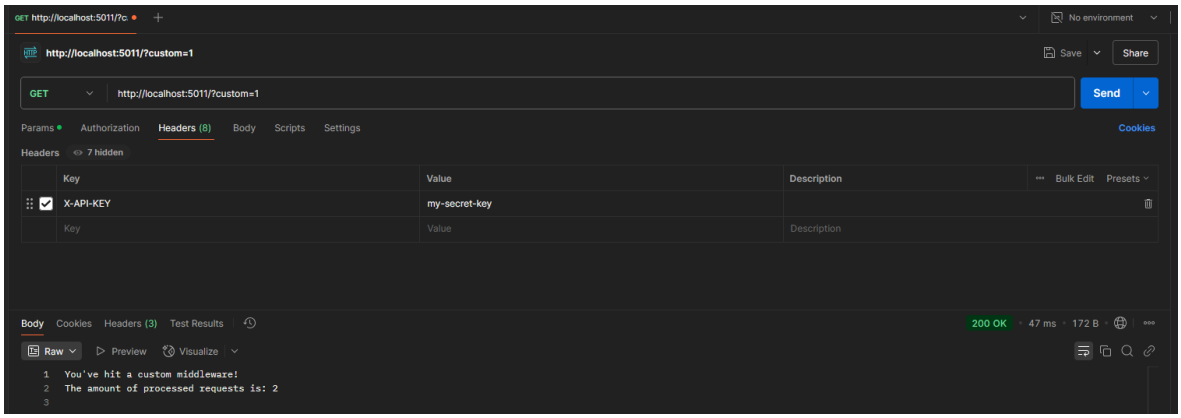

Рис. 7 Запит, що містить параметр “custom”

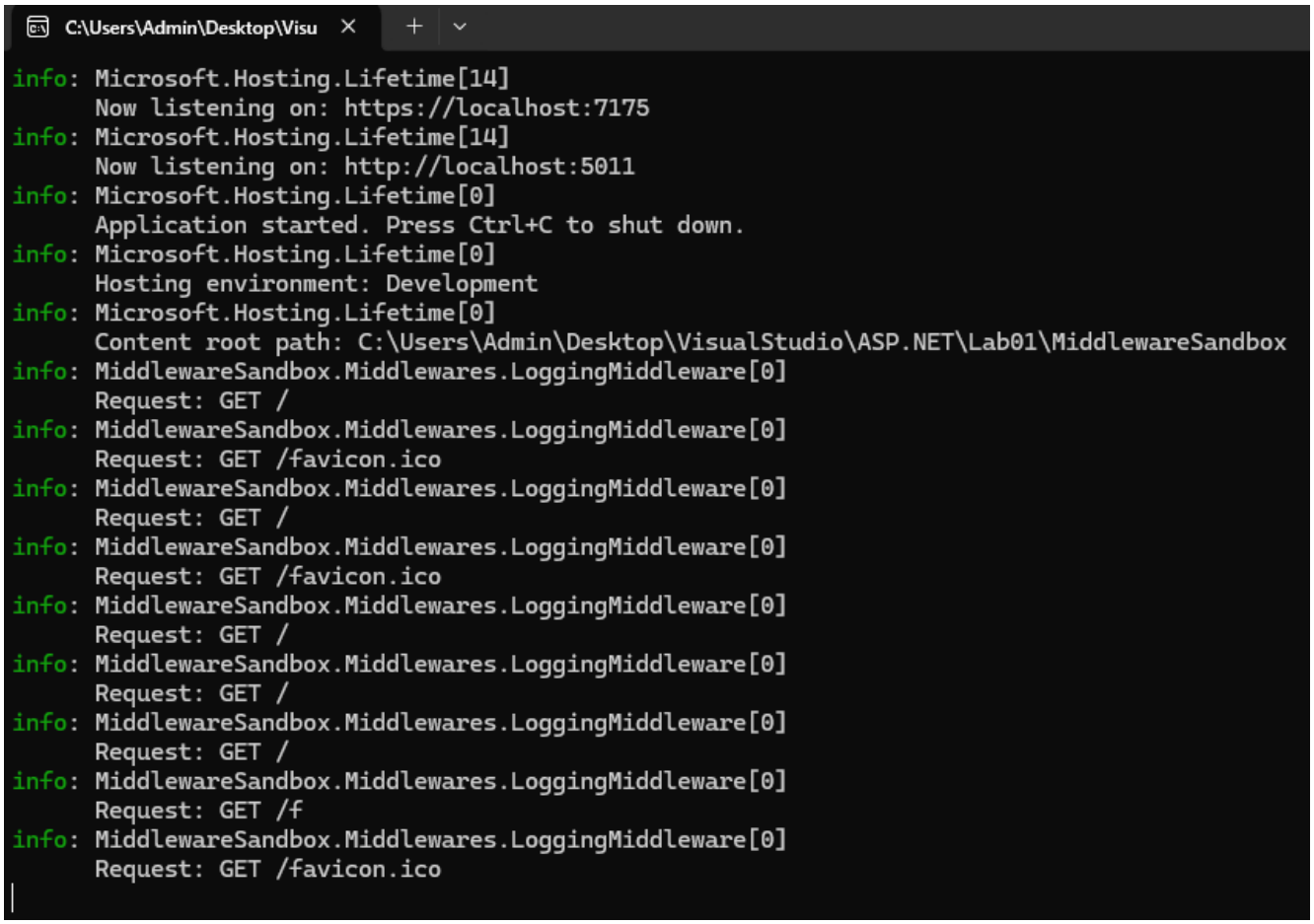

Рис. 8 Виведення запитів в консоль

Посилання на репозиторій: https://github.com/JanRizhenko/ASP.NET

Висновок: у результаті виконання цієї лабораторної роботи я ознайомитися з основними принципами роботи ASP.NET. Набув навичок обробки запитів з використанням middleware. Опанував роботу з .NET CLI.