

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.

Інтегроване середовище розробки (надалі IDE) було попередньо встановлене на персональному комп'ютері, тому етап встановлення та початкового налаштування було пропущено. Для виконання лабораторної роботи використовувалося середовище Visual Studio 2026 Community, яке підтримує розробку застосунків на платформі .NET.

Для створення та запуску проектів застосовувалася актуальна LTS-версія платформи .NET 10. Усі необхідні компоненти для веб-розробки та роботи з ASP.NET Core були встановлені.

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.

Для виконання першої частини завдання було створено консольний застосунок з назвою ConsoleToWeb з використанням інструменту dotnet CLI. Створення проекту

					ДУ «Житомирська політехніка».25.121.1.000 – Лр1							
Змн.	Арк.	№ докум.	Підпис	Дата								
Розроб.		Андріюк Д. Р.			Звіт з лабораторної роботи				Літ.	Арк.	Аркушів	
Перевір.		Українець М.О.									1	16
Керівник									ФІКТ Гр. ІПЗ-22-2[1]			
Н. контр.												
Зав. каф.												

виконувалося через командний рядок, що дозволяє продемонструвати роботу з кросплатформним інструментарієм .NET. (рис. 1.1).

```

Command Prompt
D:\Навчання\ASP.NET\Labs\lab1>dir
Volume in drive D is 
Volume Serial Number is 

Directory of D:\Навчання\ASP.NET\Labs\lab1

28.12.2025  21:45    <DIR>          .
28.12.2025  21:45    <DIR>          ..
               0 File(s)                0 bytes
               2 Dir(s)                bytes free

D:\Навчання\ASP.NET\Labs\lab1>dotnet new console -n ConsoleToWeb
The template "Console App" was created successfully.

Processing post-creation actions...
Restoring D:\Навчання\ASP.NET\Labs\lab1\ConsoleToWeb\ConsoleToWeb.csproj:
Restore succeeded.

D:\Навчання\ASP.NET\Labs\lab1>dir
Volume in drive D is 
Volume Serial Number is 

Directory of D:\Навчання\ASP.NET\Labs\lab1

28.12.2025  21:45    <DIR>          .
28.12.2025  21:45    <DIR>          ..
28.12.2025  21:45    <DIR>          ConsoleToWeb
               0 File(s)                0 bytes
               3 Dir(s)                bytes free

```

Рис. 2.1.1. Створення консольного проекту використовуючи dotnet CLI.

Після створення проекту було досліджено структуру консольного застосунку та файл конфігурації ConsoleToWeb.csproj. Оскільки за замовчуванням консольний проект не є веб-застосунком, у ньому не підключаються бібліотеки ASP.NET Core. Для перетворення консольного застосунку у веб-застосунок було змінено SDK проекту з Microsoft.NET.Sdk на Microsoft.NET.Sdk.Web.

		Андріюк Д. Р.			ДУ «Житомирська політехніка». 25.121.1.000 – Лр1	Арк.
		Українець М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		2

Лістинг конфігурації проекту до змін:

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net10.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>

</Project>
```

Лістинг конфігурації проекту після змін:

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net10.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>

</Project>
```

Після цього у файлі Program.cs було реалізовано мінімальний веб-сервер із використанням підходу Minimal API. Для обробки HTTP GET-запиту за кореневим маршрутом застосовано метод MapGet, який повертає текстове повідомлення у відповідь браузеру.

Лістинг коду програми:

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

app.MapGet("/", () => "Root page for ConsoleToWeb.");

app.Run();
```

Результат роботи застосунку було перевірено шляхом запуску проєкту та відкриття кореневої сторінки у веб-браузері.

		Андріюк Д. Р.			ДУ «Житомирська політехніка».25.121.1.000 – Лр1	Арк.
		Українець М.О.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Select D:\Навчання\ASP.NET\Labs\lab1\ConsoleToWeb\bin\Debug\net10.0\ConsoleToWeb.exe
dbug: Microsoft.AspNetCore.Watch.BrowserRefresh.BlazorWasmHotReloadMiddleware[0]
Middleware loaded
dbug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserScriptMiddleware[0]
Middleware loaded. Script /_framework/aspnetcore-browser-refresh.js (16509 B).
dbug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserScriptMiddleware[0]
Middleware loaded. Script /_framework/blazor-hotreload.js (799 B).
dbug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserRefreshMiddleware[0]
Middleware loaded: DOTNET_MODIFIABLE_ASSEMBLIES=debug, __ASPNETCORE_BROWSER_TOOLS=true
info: Microsoft.Hosting.Lifetime[14]
Now listening on: https://localhost:58164
info: Microsoft.Hosting.Lifetime[14]
Now listening on: http://localhost:58165
info: Microsoft.AspNetCore.Mvc.Infrastructure.DefaultActionDescriptorCollectionProvider[1]
No action descriptors found. This may indicate an incorrectly configured application or missing application parts.
To learn more, visit https://aka.ms/aspnet/mvc/app-parts
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:\Навчання\ASP.NET\Labs\lab1\ConsoleToWeb
info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
Request starting HTTP/2 GET https://localhost:58164/ - - -
info: Microsoft.AspNetCore.Routing.EndpointMiddleware[0]
Executing endpoint 'HTTP: GET /'
info: Microsoft.AspNetCore.Routing.EndpointMiddleware[1]
Executed endpoint 'HTTP: GET /'
info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
Request finished HTTP/2 GET https://localhost:58164/ - 200 - text/plain; charset=utf-8 293.9817ms
info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
Request starting HTTP/2 GET https://localhost:58164/favicon.ico - - -
info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
Request finished HTTP/2 GET https://localhost:58164/favicon.ico - 404 0 - 166.2853ms
info: Microsoft.AspNetCore.Hosting.Diagnostics[16]
Request reached the end of the middleware pipeline without being handled by application code. Request path: GET ht
tps://localhost:58164/favicon.ico, Response status code: 404

```

Рис. 2.1.2. Запущений проект з локальним хостингом сервісу.

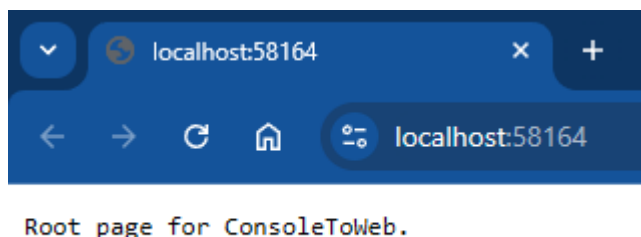


Рис. 2.1.3. Відображення головної сторінки веб додатку.

Частина 2. Створіть ASP.NET WebAPI проект без авторизації з назвою WebFromCli з використанням dotnet CLI. Реалізуйте GET-ендпоінт “/who”, який повертатиме у відповідь ваше ім’я та прізвище. Реалізуйте GET-ендпоінт “/time”, який повертатиме у відповідь поточний час на сервері.

На другому етапі лабораторної роботи було створено ASP.NET Web API проєкт з назвою WebFromCli без використання механізмів авторизації. Проєкт створювався за допомогою шаблону webapi через інструмент dotnet CLI.

```

C:\> Command Prompt
D:\Навчання\ASP.NET\Labs\lab1>dotnet new webapi -n WebFromCli --no-https
The template "ASP.NET Core Web API" was created successfully.

Processing post-creation actions...
Restoring D:\Навчання\ASP.NET\Labs\lab1\WebFromCli\WebFromCli.csproj:
Restore succeeded.

D:\Навчання\ASP.NET\Labs\lab1>

```

Рис. 2.2.1. Створення веб проєкту використовуючи dotnet CLI.

Після створення проєкту у головному файлі було реалізовано декілька GET-енд-поінтів з використанням підходу Minimal API. Ендпоінт /who повертає мої ім'я та прізвище англійською мовою, а ендпоінт /time повертає поточний серверний час.

Лістинг коду:

```

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

app.MapGet("/", () => "Root page for WebFromCli.");
app.MapGet("/who", () => "Denis Andriiuk");
app.MapGet("/time", () => DateTime.Now.ToString("HH:mm:ss"));

app.Run();

```

Для перевірки коректності роботи ендпоінтів застосунок було запущено, після чого виконано HTTP-запити через браузер.

```

Select D:\Навчання\ASP.NET\Labs\lab1\WebFromCli\bin\Debug\net10.0\WebFromCli.exe
dbug: Microsoft.AspNetCore.Watch.BrowserRefresh.BlazorWasmHotReloadMiddleware[0]
      Middleware loaded
dbug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserScriptMiddleware[0]
      Middleware loaded. Script /_framework/aspnetcore-browser-refresh.js (16505 B).
dbug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserScriptMiddleware[0]
      Middleware loaded. Script /_framework/blazor-hotreload.js (799 B).
dbug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserRefreshMiddleware[0]
      Middleware loaded: DOTNET_MODIFIABLE_ASSEMBLIES=debug, __ASPNETCORE_BROWSER_TOOLS=true
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5033
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:\Навчання\ASP.NET\Labs\lab1\WebFromCli

```

Рис. 2.2.2. Запуск додатку та відображення даних в консолі

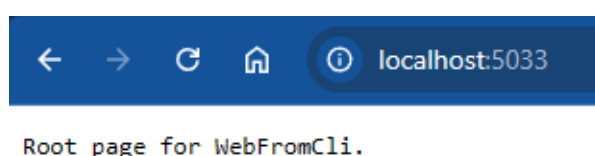


Рис. 2.2.3. Вигляд корневої сторінки

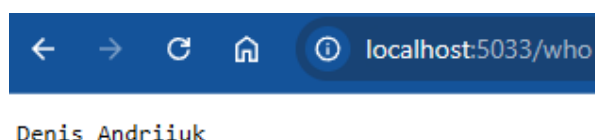


Рис. 2.2.4. Вигляд сторінки за шляхом «\who»

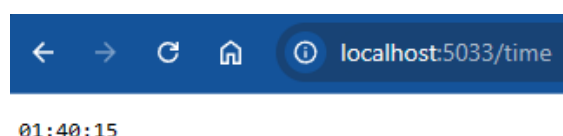


Рис. 2.2.5. Вигляд сторінки за шляхом «\time»

Частина 3. Створіть ASP.NET MVC проект будь-яким зручним способом. Реалізуйте контролер з назвою LabController. 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. Дані для відображення передати з контролера.

		Андріюк Д. Р.			ДУ «Житомирська політехніка».25.121.1.000 – Лр1	Арк.
		Українець М.О.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

У третій частині лабораторної роботи було створено ASP.NET MVC проєкт з використанням шаблону mvc. Даний тип проєкту дозволяє застосовувати архітектурний підхід Model-View-Controller, який забезпечує чітке розділення відповідальностей між логікою обробки даних та їх представленням.

```

D:\Навчання\ASP.NET\Labs\lab1>dotnet new mvc -n MVC
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/10.0-third-party-notices for details.

Processing post-creation actions...
Restoring D:\Навчання\ASP.NET\Labs\lab1\MVC\MVC.csproj:
Restore succeeded.

D:\Навчання\ASP.NET\Labs\lab1>

```

Рис. 2.3.1. Створення MVC проєкту використовуючи dotnet CLI.

У створеному проєкті було реалізовано контролер з назвою LabController. В даному контролері створено обробник /info, який формує модель з інформацією про номер лабораторної роботи, тему, мету та ім'я і прізвище виконавця. Дані передаються у представлення за допомогою ViewModel.

Лістинг контролера :

```

using Microsoft.AspNetCore.Mvc;
using MVC.Models;

namespace MVC.Controllers {
    public class LabController : Controller {
        public IActionResult Info() {
            var model = new LabInfoViewModel {
                LabNumber = 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 проєктів різних типів, набути навичок обробки запитів з використанням middleware.",
                StudentName = "Denis Andriiuk"
            };

            return View(model);
        }
    }
}

```

		Андріюк Д. Р.			ДУ «Житомирська політехніка».25.121.1.000 – Лр1	Арк.
		Українець М.О.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

Для передачі структурованих даних у View було створено модель LabInfoViewModel, яка містить необхідні властивості.

Лістинг моделі :

```
namespace MVC.Models {  
    public class LabInfoViewModel {  
        public int LabNumber { get; set; }  
        public string Topic { get; set; }  
        public string Purpose { get; set; }  
        public string StudentName { get; set; }  
    }  
}
```

У файлі представлення Info.cshtml було реалізовано HTML-таблицю, в якій відображаються отримані з контролера дані.

Лістинг вью моделі :

```
@model MVC.Models.LabInfoViewModel  
  
<h2>Інформація про лабораторну роботу</h2>  
  
<table border="1" cellpadding="8">  
    <tr>  
        <th>Параметр</th>  
        <th>Значення</th>  
    </tr>  
    <tr>  
        <td>Номер лабораторної роботи</td>  
        <td>@Model.LabNumber</td>  
    </tr>  
    <tr>  
        <td>Тема</td>  
        <td>@Model.Topic</td>  
    </tr>  
    <tr>  
        <td>Мета</td>  
        <td>@Model.Purpose</td>  
    </tr>  
    <tr>  
        <td>Студент</td>  
        <td>@Model.StudentName</td>  
    </tr>  
</table>
```

Результат роботи MVC-застосунку було перевірено шляхом переходу за маршрутом /Lab/Info у браузері.

		Андріюк Д. Р.			ДУ «Житомирська політехніка».25.121.1.000 – Лр1	Арк.
		Українець М.О.				8
Змн.	Арк.	№ докум.	Підпис	Дата		


```
D:\Навчання\ASP.NET\Labs\lab1\MVC\bin\Debug\net10.0\MVC.exe
debug: Microsoft.AspNetCore.Watch.BrowserRefresh.BlazorWasmHotReloadMiddleware[0]
  Middleware loaded
debug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserScriptMiddleware[0]
  Middleware loaded. Script /_framework/aspnetcore-browser-refresh.js (16491 B).
debug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserScriptMiddleware[0]
  Middleware loaded. Script /_framework/blazor-hotreload.js (799 B).
debug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserRefreshMiddleware[0]
  Middleware loaded: DOTNET_MODIFIABLE_ASSEMBLIES=debug, __ASPNETCORE_BROWSER_TOOLS=true
info: Microsoft.Hosting.Lifetime[14]
  Now listening on: http://localhost:5203
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:\Навчання\ASP.NET\Labs\lab1\MVC
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
  Failed to determine the https port for redirect.
debug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserRefreshMiddleware[1]
  Response markup is scheduled to include browser refresh script injection.
debug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserRefreshMiddleware[2]
  Response markup was updated to include browser refresh script injection.
debug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserScriptMiddleware[0]
  Script injected: /_framework/aspnetcore-browser-refresh.js
```

Рис. 2.3.2. Запуск новоствореного додатку та відображення даних про запуск в консолі

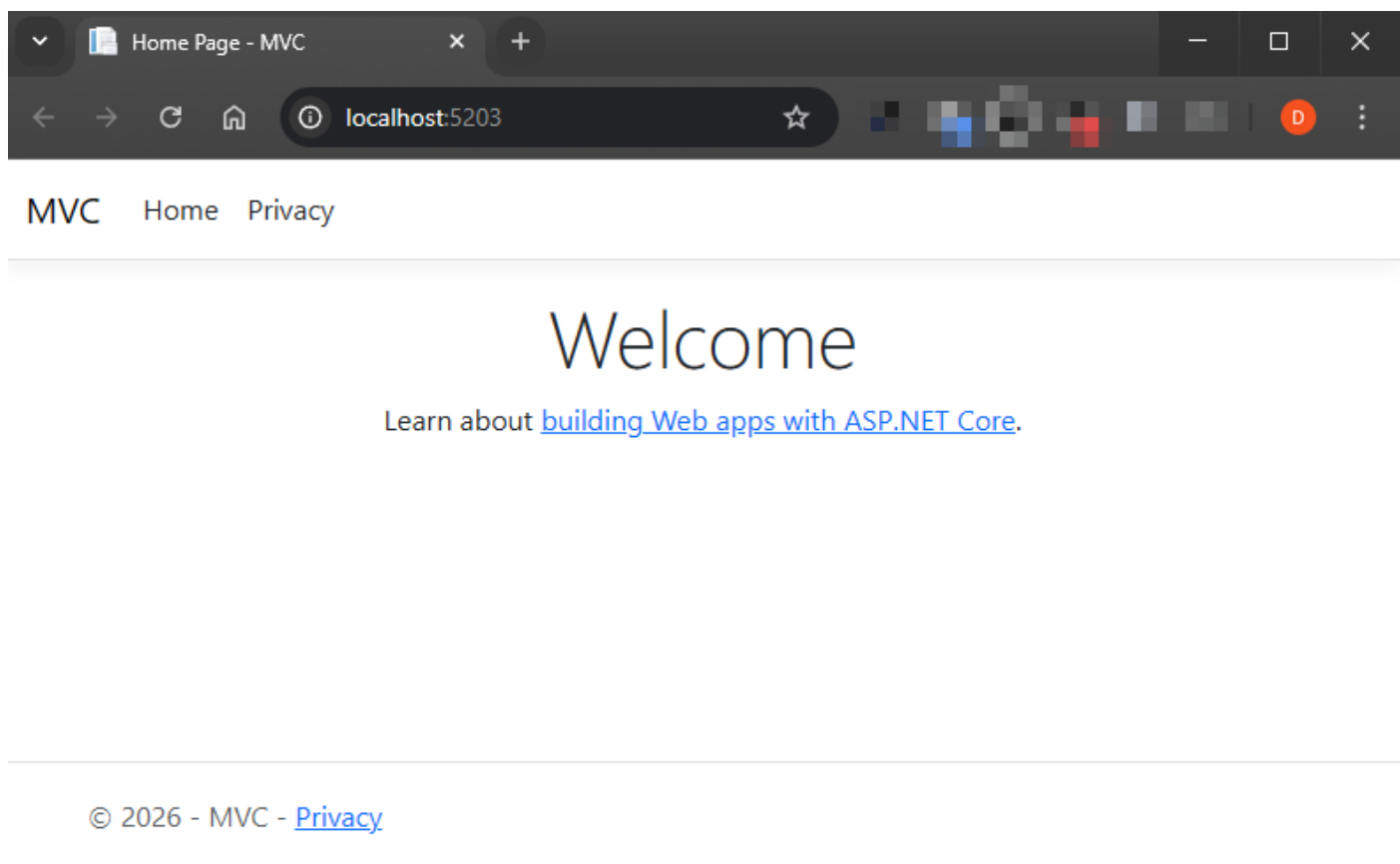
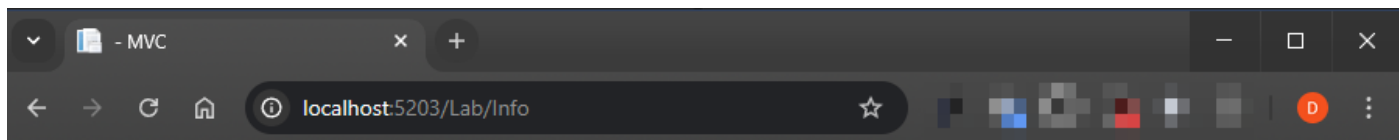


Рис. 2.3.3. Відображення створеного проекту в браузері

		Андріюк Д. Р.			ДУ «Житомирська політехніка».25.121.1.000 – Лр1	Арк.
		Українець М.О.				9
Змн.	Арк.	№ докум.	Підпис	Дата		



MVC Home Privacy

Інформація про лабораторну роботу

Параметр	Значення
Номер лабораторної роботи	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 вико-ристанням middleware.
Студент	Denis Andriiuk

© 2026 - MVC - [Privacy](#)

Рис. 2.3.4. Відображення створеного розділу за шляхом «/Lab/Info»

Завдання 3. Робота з middleware.

Створіть ASP.NET WebAPI проект з назвою MiddlewareSandbox.

Реалізуйте наступні middleware:

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

		Андрійук Д. Р.			ДУ «Житомирська політехніка».25.121.1.000 – Лр1	Арк.
		Українець М.О.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

- 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 запити можна скористатися Postman або подібними утилітами.

Для дослідження принципів роботи middleware було створено ASP.NET Web API проєкт з назвою MiddlewareSandbox. Даний проєкт використовується для експериментів із конвеєром обробки HTTP-запитів.

```

Command Prompt
D:\Навчання\ASP.NET\Labs\lab1>dotnet new webapi -n MiddlewareSandbox
The template "ASP.NET Core Web API" was created successfully.

Processing post-creation actions...
Restoring D:\Навчання\ASP.NET\Labs\lab1\MiddlewareSandbox\MiddlewareSandbox.csproj:
  Determining projects to restore...
  Restored D:\Навчання\ASP.NET\Labs\lab1\MiddlewareSandbox\MiddlewareSandbox.csproj (in 2,45 sec).
Restore succeeded.

D:\Навчання\ASP.NET\Labs\lab1>
  
```

Рис. 3.1. Створення ASP.NET WebAPI проєкту використовуючи dotnet CLI.

Було відредаговано файл конфігурації запуску застосунку з метою використання протоколу HTTP та фіксованого порта 5000. Зазначені зміни не впливали на функціональність або логіку роботи застосунку, а були виконані з навчальною метою — для ознайомлення з механізмами конфігурації та запуску ASP.NET Web API проєктів.

Файл конфігурації:

```

{
  "$schema": "https://json.schemastore.org/launchsettings.json",
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:59035",
      "sslPort": 44300
    }
  }
}
  
```

		Андріюк Д. Р.			ДУ «Житомирська політехніка». 25.121.1.000 – Лр1	Арк.
		Українець М.О.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    },
    "profiles": {
      "http": {
        "commandName": "Project",
        "dotnetRunMessages": true,
        "launchBrowser": true,
        "launchUrl": "swagger",
        "applicationUrl": "http://localhost:5000",
        "environmentVariables": {
          "ASPNETCORE_ENVIRONMENT": "Development"
        }
      },
      "https": {
        "commandName": "Project",
        "dotnetRunMessages": true,
        "launchBrowser": true,
        "launchUrl": "swagger",
        "applicationUrl": "https://localhost:7071;http://localhost:5000",
        "environmentVariables": {
          "ASPNETCORE_ENVIRONMENT": "Development"
        }
      },
      "IIS Express": {
        "commandName": "IISExpress",
        "launchBrowser": true,
        "launchUrl": "swagger",
        "environmentVariables": {
          "ASPNETCORE_ENVIRONMENT": "Development"
        }
      }
    }
  }
}

```

У межах цього завдання було реалізовано декілька middleware, кожне з яких виконує окрему функцію в процесі обробки запиту:

1. Middleware для логування HTTP-запитів. Даний middleware виводить у консоль інформацію про метод запиту (GET, POST тощо) та шлях запиту.

Лістинг middleware для логування:

```

app.Use(async (context, next) => {
    Console.WriteLine($"{DateTime.Now:HH:mm:ss} {context.Request.Method} {context.Request.Path}");
    await next();
});

```

2. Middleware для перевірки наявності API-ключа в заголовку X-API-KEY. У випадку відсутності або некоректності ключа повертається відповідь зі статусом 403 Forbidden.

		Андріюк Д. Р.			ДУ «Житомирська політехніка».25.121.1.000 – Лр1	Арк.
		Українець М.О.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг middleware для перевірки API-ключа:

```
app.Use(async (context, next) => {
    if (!context.Request.Headers.TryGetValue("X-API-KEY", out var apiKey) || apiKey !=
ServerApiKey) {
        context.Response.StatusCode = StatusCodes.Status403Forbidden;
        context.Response.ContentType = "text/plain; charset=utf-8";
        await context.Response.WriteAsync("403 Forbidden: invalid or missing X-API-
KEY.");
        return;
    }
    await next();
});
```

3. Middleware для обробки параметрів рядка запиту. Якщо у query string присутній параметр custom, middleware перехоплює запит і повертає відповідь з повідомленням про спрацювання кастомного middleware.

Лістинг middleware для перевірки параметра custom:

```
app.Use(async (context, next) => {
    if (context.Request.Query.ContainsKey("custom")) {

        context.Response.StatusCode = StatusCodes.Status200OK;
        context.Response.ContentType = "text/plain; charset=utf-8";
        await context.Response.WriteAsync("You've hit a custom middleware!\n");
        return;
    }
    await next();
});
```

4. Middleware для підрахунку кількості оброблених запитів. Кількість успішно оброблених запитів збільшується атомарно та повертається у відповіді серверу.

Лістинг middleware для підрахунку запитів:

```
app.Use(async (context, next) => {
    if (context.Request.Path == "/favicon.ico") {
        context.Response.StatusCode = StatusCodes.Status204NoContent;
        return;
    }

    var current = Interlocked.Increment(ref requestCount);

    context.Response.ContentType = "text/plain; charset=utf-8";
    await next();
    await context.Response.WriteAsync($"The amount of processed requests is {current}.");
});
```

		Андрійук Д. Р.			ДУ «Житомирська політехніка».25.121.1.000 – Лр1	Арк.
		Українець М.О.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Для тестування роботи middleware було використано утиліту Postman. У Postman було створено environment зі змінними для хоста, порта та API-ключа, що дозволяє зручно змінювати параметри запитів.

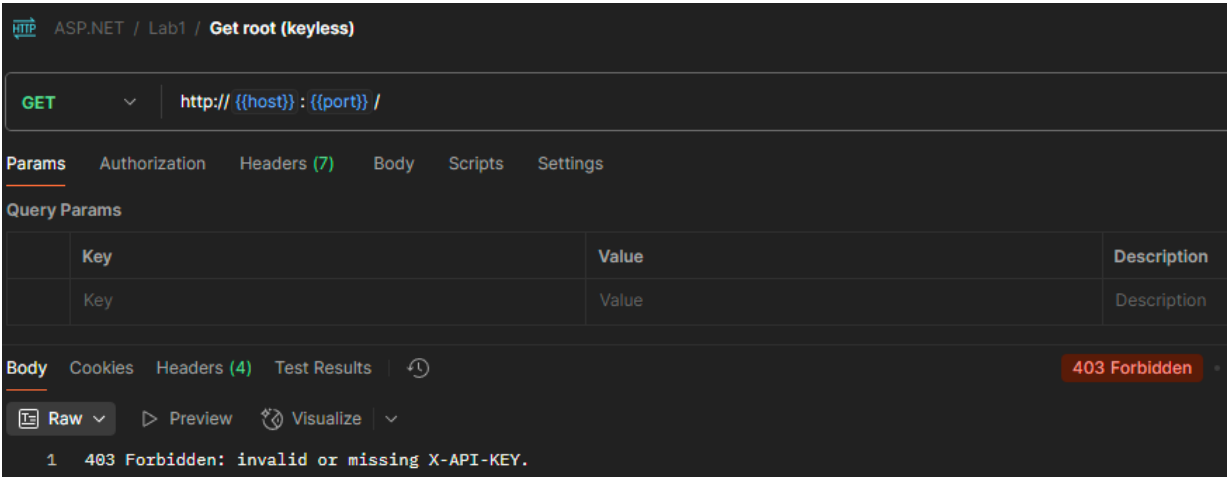


Рис. 3.2. Виконання запиту без ключа API.

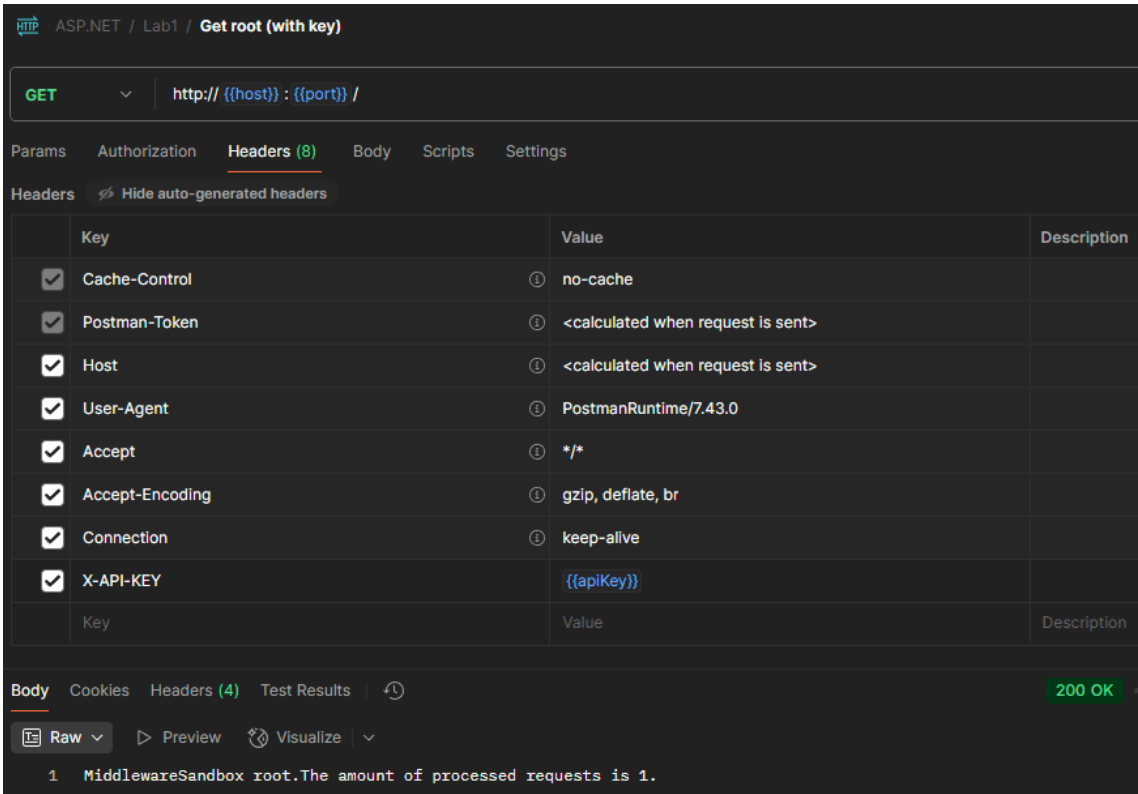


Рис. 3.3. Виконання запиту з ключем API, відображення кількості запитів (успішних).

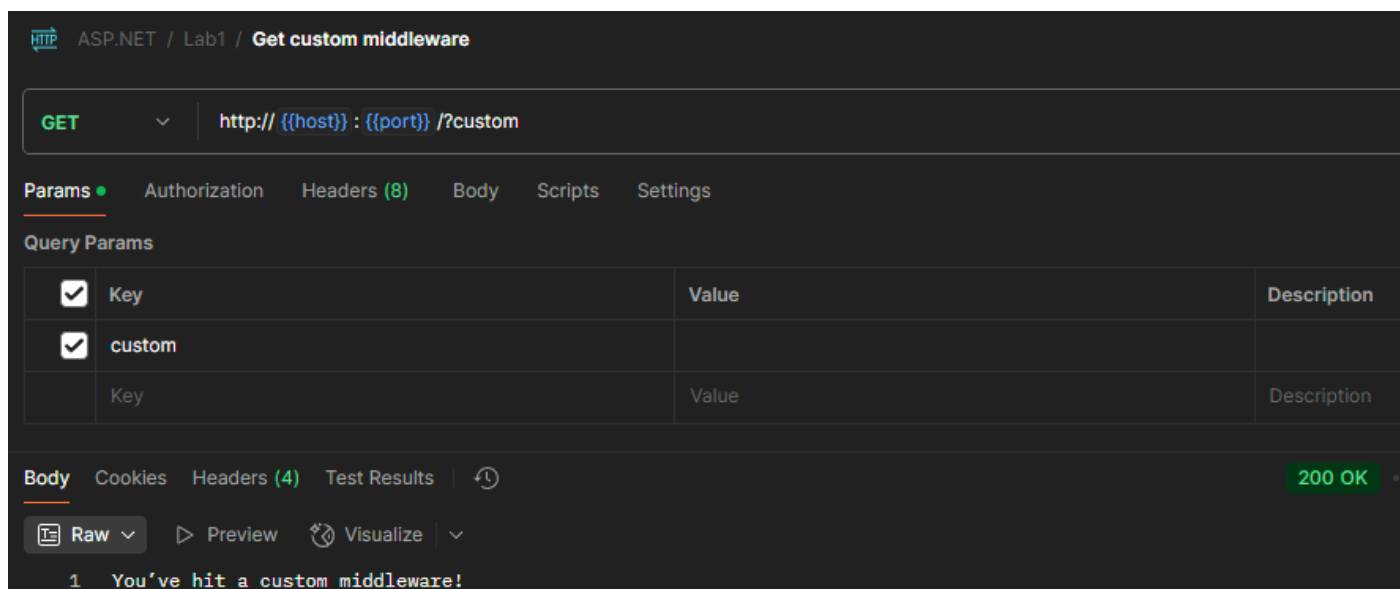


Рис. 3.4. Виконання запиту з відслідковуванням кастомного middleware.

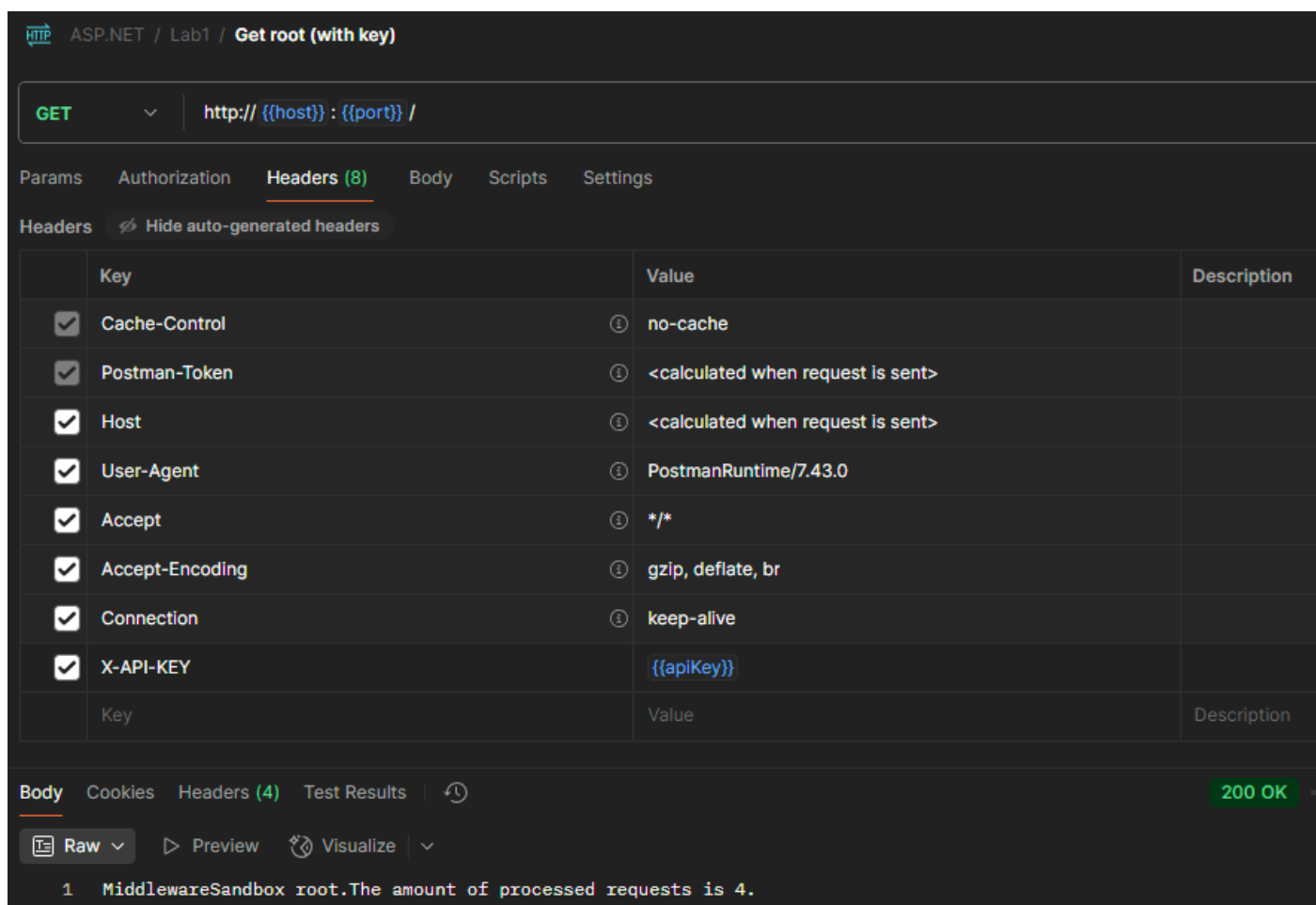


Рис. 3.5. Виконання запиту з ключем API, відображення кількості запитів після декількох викликів.

		Андріюк Д. Р.			ДУ «Житомирська політехніка».25.121.1.000 – Лр1	Арк.
		Українець М.О.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

 D:\Навчання\ASP.NET\Labs\lab1\MiddlewareSandbox\bin\Debug\net10.0\MiddlewareSandbox.exe

```
debug: Microsoft.AspNetCore.Watch.BrowserRefresh.BlazorWasmHotReloadMiddleware[0]
  Middleware loaded
debug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserScriptMiddleware[0]
  Middleware loaded. Script /_framework/aspnetcore-browser-refresh.js (16519 B).
debug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserScriptMiddleware[0]
  Middleware loaded. Script /_framework/blazor-hotreload.js (799 B).
debug: Microsoft.AspNetCore.Watch.BrowserRefresh.BrowserRefreshMiddleware[0]
  Middleware loaded: DOTNET_MODIFIABLE_ASSEMBLIES=debug, __ASPNETCORE_BROWSER_TOOLS=true
info: Microsoft.Hosting.Lifetime[14]
  Now listening on: http://localhost:5000
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:\Навчання\ASP.NET\Labs\lab1\MiddlewareSandbox
[21:07:46] GET /swagger
[21:07:46] GET /favicon.ico
[21:45:34] GET /
[21:47:22] GET /
[21:48:18] GET /
[21:49:17] GET /
```

Рис. 3.6. Запуск новоствореного додатку та відображення даних в консолі

Висновок: У ході виконання лабораторної роботи було досліджено основні підходи до створення застосунків на платформі ASP.NET Core. Було створено та запущено консольний застосунок, Web API та MVC-застосунок із використанням dotnet CLI. Особливу увагу було приділено роботі з middleware та побудові конвеєра обробки HTTP-запитів. Отримані знання дозволяють зрозуміти принципи роботи сучасних веб-застосунків та є базою для подальшого вивчення ASP.NET Core.

		Андріюк Д. Р.			ДУ «Житомирська політехніка».25.121.1.000 – Лр1	Арк.
		Українець М.О.				16
Змн.	Арк.	№ докум.	Підпис	Дата		