

## Лабораторна робота №5

**Тема:** Аутентифікація та авторизація. Робота з Web API. JSON Web Token.

**Мета:** набути навичок роботи з механізмами аутентифікації та авторизації користувачів в ASP.NET, набути навичок роботи з Web API.

Хід роботи:

### Завдання 1.

Реалізувати реєстрацію, аутентифікацію та авторизацію користувача з використанням пакету Microsoft Identity.

Для встановлення Identity, в папці з проектом слід виконати наступну команду:  
dotnet add package Microsoft.AspNetCore.Identity.EntityFrameworkCore --version 8.0.0

Далі слід створити клас AppIdentityDbContext з наступним вмістом:

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
namespace SportsStore.Models {
    public class AppIdentityDbContext : IdentityDbContext<IdentityUser> {
        public AppIdentityDbContext(
            DbContextOptions<AppIdentityDbContext> options)
            : base(options) { }
    }
}
```

Далі слід додати нову ConnectionString в appsettings.json для бази даних, де зберігатимуться дані про користувачів:

```
{
    "Logging": {
        "LogLevel": {
            "Default": "Information",
            "Microsoft.AspNetCore": "Warning"
        }
    },
    "AllowedHosts": "*",
    "ConnectionStrings": {
        "SportsStoreConnection": {
            "Server=(localdb)\\MSSQLLocalDB;Database=SportsStore;MultipleActiveResultSets=true",
            "IdentityConnection": {
                "Server=(localdb)\\MSSQLLocalDB;Database=Identity;MultipleActiveResultSets=true"
            }
        }
    }
}
```

Сконфігурувати Identity в файлі Program.cs, додавши наступні рядки:

```
...
using Microsoft.AspNetCore.Identity;

...
builder.Services.AddScoped();
builder.Services.AddScoped();
```

Змн.	Арк.	№ докум.	Підпис	Дата
Розроб.	Ганчевський О.О.			
Перевір.	Українець М.О.			
Керівник				
Н. контр.				
Зав. каф.				

ДУ «Житомирська політехніка».25.121.07.000 – Пр5

Звіт з  
лабораторної роботи №5

Літ.	Арк.	Аркушів
	1	4
ФІКТ Гр. ІПЗ-22-2[1]		

```

builder.Services.AddRazorPages();
builder.Services.AddDistributedMemoryCache();
builder.Services.AddSession();

...
builder.Services.AddDbContext(options => options.UseSqlServer(
    builder.Configuration["ConnectionStrings:IdentityConnection"])
);
builder.Services.AddIdentity().AddEntityFrameworkStores();

var app = builder.Build();
app.UseStaticFiles();
app.UseSession();

app.UseAuthentication();
app.UseAuthorization();

```

Створіть міграцію для бази даних Identity наступною командою:

```
dotnet ef migrations add Initial --context AppIdentityDbContext
```

Та застосуйте її за допомогою наступної команди:

```
dotnet ef database update --context AppIdentityDbContext
```

Після конфігурації Identity, реалізуйте наступні функціональності:

1. Реєстрація користувача (приклад)
2. встановити вимогу по унікальноті електронної адреси користувача
3. встановити вимоги до пароля: не коротший за 8 символів, містить цифри та літери, обов'язково містить хоча б одну літеру в верхньому регістрі
4. реалізувати поле для підтвердження введеного паролю
5. Автентифікація користувача (Login)
6. Вихід користувача (Logout)
7. Особистий кабінет користувача, де він матиме змогу переглядати або змінювати дані про себе.

Розробіть всі необхідні форми з підтримкою серверної валідації та розробіть відповідний контроллер. Налаштуйте доступ до Action-методів контроллерів з використанням атрибуту [Authorize] – лише залогінений користувач повинен мати можливості виконувати дії на веб-сайті, для неавтентифікованого користувача має бути доступною лише реєстрація та логін. Приклад коду контролера для виконання логіну:

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using SportsStore.Models.ViewModels;
namespace SportsStore.Controllers
{
    public class AccountController : Controller
    {
        private UserManager<IdentityUser> userManager;
        private SignInManager<IdentityUser> signInManager;
        public AccountController(UserManager<IdentityUser> userMgr,
            SignInManager<IdentityUser> signInMgr)

```

		Ганчевський О. О.			ДУ «Житомирська політехніка».25. <b>121.07.000</b> – Пр5	Арк.
		Українець М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		2

```

    {
        userManager = userMgr;
        signInManager = signInMgr;
    }
    public ViewResult Login(string returnUrl)
    {
        return View(new LoginModel
        {
            Name = string.Empty,
            Password = string.Empty,
            ReturnUrl = returnUrl
        });
    }
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Login(LoginModel loginModel)
    {
        if (ModelState.IsValid)
        {
            IdentityUser? user =
                await userManager.FindByNameAsync(loginModel.Name);
            if (user != null)
            {
                await signInManager.SignOutAsync();
                if ((await
signInManager.PasswordSignInAsync(user,
                    loginModel.Password, false, false)).Succeeded)
                {
                    return Redirect(loginModel?.ReturnUrl
                    ?? "/Admin");
                }
            }
            ModelState.AddModelError("", "Invalid name or
password");
        }
        return View(loginModel);
    }
    [Authorize]
    public async Task<RedirectResult> Logout(string returnUrl = "/")
    {
        await signInManager.SignOutAsync();
        return Redirect(returnUrl);
    }
}
}

```

## Завдання 2.

В рішенні створити новий WebAPI проект. Цей проект повинен реалізовувати всі функціональності, які наявні в MVC проекті у вигляді REST-ендпоїнтів. Для кращої організації і перевикористання коду слід перенести файли для роботи з БД (репозиторії, контексти, моделі) в окрему бібліотеку і додати посилання на неї в MVC та WebAPI проектах. Встановіть необхідні залежності для проекту бібліотеки.

Після цього налаштуйте ConnectionString'и для створених БД для WebAPI проекту в файлі appsettings.json. Таким чином, після перенесення файлів для роботи з

		Ганчевський О. О.			ДУ «Житомирська політехніка».25. <b>121.07.000 – Пр5</b>	Арк.
		Українець М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		3

БД в окрему бібліотеку ми можемо використовувати всі необхідні сервіси в обох проектах.

**Реалізуйте всі необхідні CRUD-операції в вашому WebAPI проекті, включаючи реєстрацію користувача. Перевірте працевздатність ваших ендпоїнтів за допомогою Postman, Swagger або будь-яким іншим зручним інструментом.**

### **Завдання 3.**

Реалізуйте автентифікацію та авторизацію користувача з використанням механізму JSON Web Token (JWT). Вона повинна працювати наступним чином:

1. Клієнт відправляє запит на login-ендпоїнт зі своїм логіном та паролем.
2. Якщо користувач з таким логіном та паролем існує, тоді сервер повинен повернути згенерований JSON Web Token.
3. Для отримання доступу до дій, які вимагають авторизації, клієнт відправляє в запиті заголовок Authorization зі значенням Bearer TokenValue, де TokenValue – токен, який було згенеровано під час автентифікації.
4. Якщо токен валідний – сервер виконає запит, інакше поверне помилку 401 Unauthorized.

**Висновок:** набув навичок роботи з механізмами аутентифікації та авторизації користувачів в ASP.NET, набув навичок роботи з Web API.

		Ганчевський О. О.			ДУ «Житомирська політехніка».25. <b>121.07</b> .000 – Пр5 Арк.
		Українець М.О.			
Змн.	Арк.	№ докум.	Підпис	Дата	