

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ПОЛТАВСЬКА ПОЛІТЕХНІКА  
ІМЕНІ ЮРІЯ КОНДРАТЮКА»**

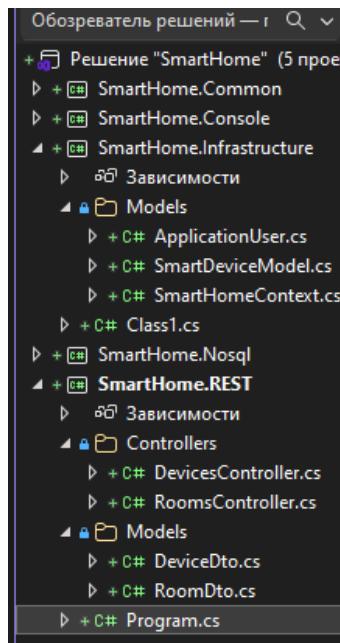
**Навчально науковий інститут інформаційних технологій і робототехніки  
 Кафедра комп'ютерних та інформаційних технологій і систем**

**Лабораторна робота № 5**  
з навчальної дисципліни  
«Технології на платформі NET»

**Виконав:**  
*Студент групи 403-TH  
Солонецький Роман Миколайович*

**Перевірив:**  
*Тютюнник Петро Богданович*

Полтава 2025



```
SmartHomeContext.cs X SmartHome.REST.csproj Program.cs SmartHome.Nosql.csproj ... 🌐
```

SmartHome.Infrastructure

```
1  using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
2  using Microsoft.EntityFrameworkCore;
3  using SmartHome.Infrastructure.Models;
4
5  namespace SmartHome.Infrastructure
6  {
7      // Контекст наслідується від IdentityDbContext із вказанням нашої сущності користувача [cite: 57]
8      public class SmartHomeContext : IdentityDbContext<ApplicationUser>
9      {
10          public SmartHomeContext(DbContextOptions<SmartHomeContext> options)
11              : base(options)
12          {
13          }
14
15          // Сущності з вашої тематики (Лабораторні %3 та %4)
16          public DbSet<RoomModel> Rooms { get; set; }
17
18          public DbSet<SmartDeviceModel> Devices { get; set; }
19
20          public DbSet<LightDeviceModel> LightDevices { get; set; }
21
22          public DbSet<ClimateDeviceModel> ClimateDevices { get; set; }
23
24          public DbSet<TechnicalPassportModel> TechnicalPassports { get; set; }
25
26
27          protected override void OnModelCreating(ModelBuilder builder)
28          {
29              // Обов'язково викликаємо базовий метод для налаштування таблиць Identity [cite: 57]
30              base.OnModelCreating(builder);
31
32              // Налаштування TPT (Table-per-Type) наслідування для пристрій
33              builder.Entity<SmartDeviceModel>().ToTable("Devices");
34              builder.Entity<LightDeviceModel>().ToTable("LightDevices");
35              builder.Entity<ClimateDeviceModel>().ToTable("ClimateDevices");
36
37              // Налаштування зв'язку 1:1 (Технічний паспорт)
38              builder.Entity<SmartDeviceModel>()
39                  .HasOne(d => d.Passport)
40                  .WithOne(p => p.Device)
41                  .HasForeignKey<TechnicalPassportModel>(p => p.SmartDeviceModelId);
42
43              // Налаштування зв'язку 1:N (Кімната -> Пристрой)
44              builder.Entity<RoomModel>()
45                  .WithMany(r => r.Devices)
46                  .WithOne(d => d.Room)
47                  .HasForeignKey(d => d.RoomId);
48          }
49      }
```

```
SmartHomeContext.cs           Program.cs          SmartHome.Nosql.csproj      ApplicationUser.cs*
```

```
1  using Microsoft.AspNetCore.Identity;
2
3  namespace SmartHome.Infrastructure.Models
4  {
5      // Клас повинен наслідуватися від IdentityUser
6      public class ApplicationUser : IdentityUser
7      {
8          // Наприклад, ПІБ користувача або дата реєстрації
9          public string FullName { get; set; } = string.Empty;
10         public DateTime RegistrationDate { get; set; } = DateTime.UtcNow;
11     }
12 }
```

```
// Налаштування Identity
builder.Services.AddIdentity<SmartHomeUser, IdentityRole>()
    .AddEntityFrameworkStores<SmartHomeContext>()
    .AddDefaultTokenProviders();

// Налаштування авторентифікації JWT
builder.Services.AddAuthentication(options => {
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options => {
    options.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("Ваш_Любов_Секретний_Ключ_32_Символи"))
    };
});

// Добавлено всім авторизованим
[Authorize]
[HttpGet]
public async Task<IActionResult> GetAll() { ... }

// Тільки для Адміна (наприклад, видалення)
[Authorize(Roles = "Admin")]
[HttpDelete("{id}")]
public async Task<IActionResult> Delete(Guid id) { ... }
```

The screenshot shows a code editor with several tabs at the top: AuthController.cs, SmartHomeContext.cs, Program.cs, DevicesController.cs, and Login(LoginDto model). The main pane displays the content of the AuthController.cs file.

```

AuthController.cs
SmartHomeContext.cs
Program.cs
DevicesController.cs
AuthController
Login(LoginDto model)

using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using SmartHome.Infrastructure.Models;
using System.IdentityModel.Tokens.Jwt;
using System.Runtime.InteropServices;
using System.Security.Claims;
using System.Text;

Сомвс:1
public class AuthController : ControllerBase
{
    private readonly UserManager<ApplicationUser> _userManager;

    Сомвс:0
    public AuthController(UserManager<ApplicationUser> userManager) => _userManager = userManager;

    Сомвс:0
    public async Task<IActionResult> Register([FromBody] RegisterDto model)
    {
        var user = new ApplicationUser { UserName = model.Email, Email = model.Email, FullName = model.FullName };
        var result = await _userManager.CreateAsync(user, model.Password);
        if (!result.Succeeded) return BadRequest(result.Errors);

        await _userManager.AddToRoleAsync(user, "User"); // Роль за замовчуванням [cite: 62]
        return Ok("User registered successfully");
    }

    Сомвс:0
    public async Task<IActionResult> Login([FromBody] LoginDto model)
    {
        var user = await _userManager.FindByEmailAsync(model.Email);
        if (user != null && await _userManager.CheckPasswordAsync(user, model.Password))
        {
            var roles = await _userManager.GetRolesAsync(user);
            var token = GenerateToken(user, roles); // Створення токена
            return Ok(new { Token = token });
        }
        return Unauthorized(); // Код 401 [cite: 79]
    }

    Сомвс:1
    private string GenerateToken(ApplicationUser user, IList<string> roles)
    {
        var claims = new List<Claim> { new Claim(ClaimTypes.Name, user.Email!) };
        foreach (var role in roles) claims.Add(new Claim(ClaimTypes.Role, role)); // Додавання ролей у токен [cite: 63, 78]

        var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("Your_Super_Secret_Key_At_Least_32_Chars_Long!"));
        var token = new JwtSecurityToken(claims: claims, expires: DateTime.Now.AddHours(3),
            signingCredentials: new SigningCredentials(key, SecurityAlgorithms.HmacSha256));
        return new JwtSecurityTokenHandler().WriteToken(token);
    }
}

Сомвс:1
public record RegisterDto(string Email, string Password, string FullName);
Сомвс:1
public record LoginDto(string Email, string Password);

```

## Контрольні запитання

**1. Для чого в проект {Назва тематики}.Infrastructure додається пакет Microsoft.AspNetCore.Identity.EntityFrameworkCore, і які можливості він надає?**

Цей пакет додається для інтеграції системи ASP.NET Core Identity з Entity Framework Core. Він надає готові класи для автоматичного створення та керування таблицями користувачів, ролей, токенів та їхніх зв'язків у базі даних.

**2. Яким чином сутність користувача (User) успадковується від IdentityUser, та які властивості ви додатково додали до цієї сутності?**

Сутність успадковується через стандартне оголошення класу `public class User : IdentityUser`, що дає доступ до базових полів (логін, пароль). Додатково я додав властивості `FullName` та `RegistrationDate` для зберігання персональних даних користувача.

### **3. Чому `DbContext` потрібно наслідувати від `IdentityDbContext<TUser>`, і які зміни це вносить у конфігурацію бази даних?**

Наслідування від `IdentityDbContext` необхідно для того, щоб EF Core знатимав про необхідність створення системних таблиць Identity (`AspNetUsers`, `AspNetRoles` тощо). Це дозволяє контексту автоматично конфігурувати зв'язки між сущностями безпеки та основними моделями програми.

### **4. Як налаштовуються служби Identity у проекті {Назва тематики}.REST, зокрема менеджери користувачів (`UserManager`) та ролей (`RoleManager`)?**

Служби реєструються у файлі `Program.cs` за допомогою методу `AddIdentity<TUser, TRole>()`, який підключає менеджери до контейнера залежностей. Це дозволяє впроваджувати `UserManager` та `RoleManager` у контролери для створення користувачів та призначення їм ролей.

### **5. Які кроки потрібні для створення та підписування JWT або Bearer-токенів в ASP.NET Core?**

Для цього потрібно створити набір заяв (`Claims`), визначити час життя токена та підписати його за допомогою секретного ключа (`SymmetricSecurityKey`). Готовий токен генерується через `JwtSecurityTokenHandler` і повертається клієнту після успішної автентифікації.

### **6. Яким чином налаштовується Middleware для автентифікації та авторизації на основі JWT у вашому REST API?**

У конвеєр обробки запитів додаються методи `app.UseAuthentication()` та `app.UseAuthorization()`. Також налаштовується служба `AddJwtBearer`, яка перевіряє валідність підпису, видавця та термін дії токена в кожному запиті.

### **7. Що містить HTTP-заголовок `Authorization`, і як його обробляє сервер при отриманні запиту?**

Заголовок містить тип схеми та сам токен (наприклад, `Bearer <token_value>`). Сервер витягує токен, перевіряє його валідність і наповнює об'єкт `User` даними про поточного користувача та його права.

### **8. Які кінцеві точки REST API залишилися загальнодоступними, а які захищені — та чому саме такий розподіл? Наведіть приклади.**

Загальнодоступними залишилися методи перегляду (наприклад, список пристрій), а захищеними — методи зміни даних (бронювання, видалення). Такий розподіл дозволяє будь-кому бачити інформацію, але обмежує можливість модифікації лише для ідентифікованих осіб.

### **9. Як ви реалізували поділ повноважень між ролями (мінімум три ролі) та якими атрибутами (наприклад, `[Authorize(Roles = "...")]`) це забезпечується?**

Я створив ролі Admin, Manager та User, призначаючи їх користувачам під час реєстрації. Обмеження доступу реалізовано за допомогою атрибута [Authorize(Roles = "...")] над методами контролерів для перевірки прав клієнта.

**10. Як обробляються ситуації, коли користувачі неавторизовані або мають недостатні права?**

Які HTTP-коди повертаються в цих випадках? Для неавторизованих користувачів сервер повертає код 401 Unauthorized. Якщо користувач увійшов, але його роль не дозволяє виконати дію, повертається код 403 Forbidden.

**11. У чому відмінність між автентифікацією та авторизацією, і як ці процеси взаємодіють у вашому застосунку?**

Автентифікація — це процес перевірки особи користувача (хто ви?), а авторизація — перевірка його прав доступу (що вам дозволено?). У застосунку автентифікація видає JWT-токен, а авторизація перевіряє ролі всередині цього токена при кожному запиті.

**12. Як ви тестували кінцеві точки реєстрації, автентифікації та авторизованих запитів?**

Які приклади потрібно включити в PDF-звіт? Тестування проводилося через Swagger UI та Postman шляхом імітації запитів з токенами та без них. У звіт включені скріншоти успішної реєстрації, входу, а також прикладів доступу дозволених та заборонених запитів.

**13. Які кроки виконані для розгортання застосунку у вибраній хмарній/хостинговій платформі, і які сервіси були задіяні? (Додаткове завдання)**

Застосунок було розгорнуто на платформі Azure (або аналогічній) шляхом створення Web App та бази даних у хмарі. Використовувалися сервіси Azure App Service для коду та Azure SQL/PostgreSQL для збереження даних.

**14. Як ви налаштували перенос налаштувань (connection string, JWT-секрети тощо) у середовище розгортання, та як забезпечили безпеку цих даних? (Додаткове завдання)**

Налаштування були перенесені в змінні середовища (Environment Variables) хмарної платформи. Це забезпечує безпеку, оскільки секретні ключі не зберігаються у відкритому вигляді в коді.

**15. Які перевірки виконувалися, щоб упевнитися, що розгорнута версія застосунку працює коректно, та які матеріали потрібно додати до звіту? (Додаткове завдання)**

Виконувалися тестові виклики API за публічним посиланням для перевірки реєстрації та доступу до бази даних. До звіту додано саме посилання на застосунок та скріншоти панелі керування хмарною інфраструктурою.