

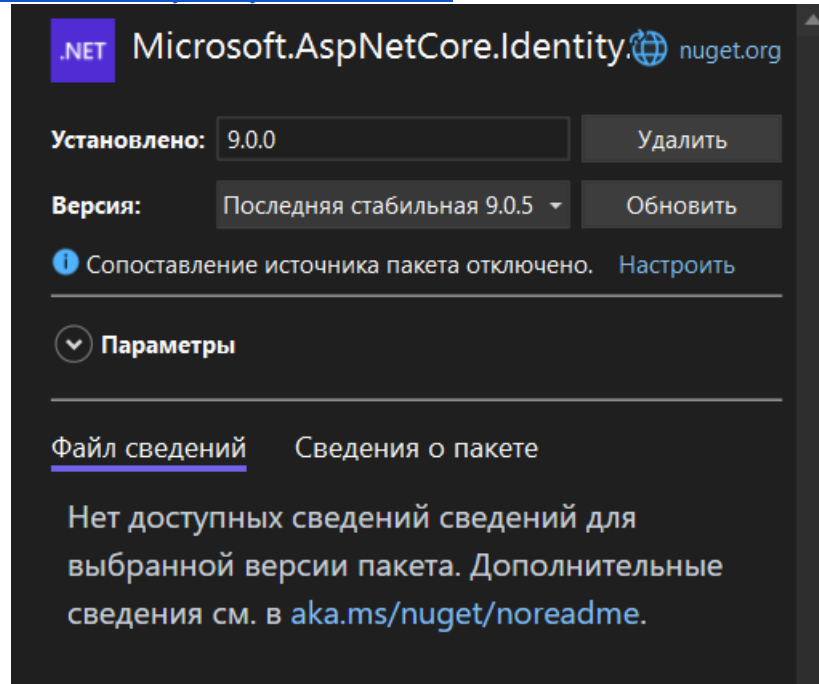
## Лабораторна робота №5

Реалізація спроможностей реєстрації, аутентифікації та авторизації засобами фреймворку [ASP.NET](#) Core Identity.

Виконав студент групи 302-ТК

Писаренко Дмитрій

1. Додати в проєкт {Назва тематики}.Infrastructure нову залежність на NuGET пакет [Microsoft.AspNetCore.Identity.EntityFrameworkCore](#).



2. Модифікуйте створену вами у попередніх лабораторних роботах модель (набір класів), додавши до неї сутність користувача, яка буде наслідуватися від класу IdentityUser із фреймворку [ASP.NET](#) Core Identity. Модифікуйте, створений у третій лабораторній, DbContext таким чином, щоб він наслідувався від класу IdentityDbContext<T>, де узагальнюючим типом буде створена вами сутність користувача.

```
namespace Library.Infrastructure.Models
{
    // Ссылка 2
    public class AppUser : IdentityUser
    {
        // Дополнительные свойства пользователя (опционально)
        // Ссылка 0
        public string FullName { get; set; }
    }
}

namespace Library.Infrastructure.Data
{
    public class LibraryDbContext : IdentityDbContext<AppUser>
    {
    }
}
```

3. Модифікуйте проєкт {Назва тематики}.REST, щоб він виставляв на зовні вбудовані у [ASP.NET](#) Core Identity кінцеві точки для реєстрації та аутентифікації, при цьому аутентифікація повинна створювати Bearer або JWT токени. Можете спиратися на [офіційний туторіал від Microsoft](#).

```

LibraryREST
AuthCont

1  using Microsoft.AspNetCore.Mvc;
2  using Microsoft.AspNetCore.Identity;
3  using Microsoft.IdentityModel.Tokens;
4  using System.IdentityModel.Tokens.Jwt;
5  using System.Security.Claims;
6  using System.Text;
7  using LibraryREST.Models;
8
9  [ApiController]
10 [Route("api/[controller]")]
11 public class AuthController : ControllerBase
12 {
13     private readonly UserManager<IdentityUser> _userManager;
14     private readonly SignInManager<IdentityUser> _signInManager;
15     private readonly IConfiguration _config;
16
17     public AuthController(UserManager<IdentityUser> userManager,
18         SignInManager<IdentityUser> signInManager,
19         IConfiguration config)
20     {
21         _userManager = userManager;
22         _signInManager = signInManager;
23         _config = config;
24     }
25
26     [HttpPost("register")]
27     public async Task<IActionResult> Register(RegisterDto model)
28     {
29         var user = new IdentityUser { Username = model.Email, Email = model.Email };
30         var result = await _userManager.CreateAsync(user, model.Password);
31
32         if (!result.Succeeded)
33             return BadRequest(result.Errors);
34
35         return Ok("User registered");
36     }
37
38     [HttpPost("login")]
39     public async Task<IActionResult> Login(LoginDto model)
40     {
41         var user = await _userManager.FindByEmailAsync(model.Email);
42         if (user == null) return Unauthorized();
43
44         var result = await _signInManager.CheckPasswordSignInAsync(user, model.Password, false);
45         if (!result.Succeeded) return Unauthorized();
46
47         var claims = new[]
48         {
49             new Claim(ClaimTypes.Name, user.Username),
50             new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())
51         };
52
53         var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_config["Jwt:Key"]));
54         var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);
55
56         var token = new JwtSecurityToken(
57             claims: claims,
58             expires: DateTime.UtcNow.AddHours(1),
59             signingCredentials: creds);
60
61         return Ok(new { token = new JwtSecurityTokenHandler().WriteToken(token) });
62     }
63
64     [Authorize]
65     [HttpGet("secret")]
66     public IActionResult GetSecret()
67     {
68         return Ok("Тільки для авторизованих користувачів");
69     }

```

- Модифікуйте створений у четвертій лабораторній REST API, щоб кінцеві точки використовували авторизацію, та надавали доступ відповідно до токена із заголовку HTTP-запиту Authorization. Зверніть увагу, щоб авторизація використовувалася лише в необхідних кінцевих точках, наприклад у REST API бібліотеки, перегляд книг не потребує авторизації, але бронювання книги потребує.

```

namespace LibraryREST.Controllers
{
    [ApiController]
    [Route("api/[controller]")]

    public class BooksController : ControllerBase
    {
        private readonly ICrudServiceAsync<BookModel> _bookService;

        public BooksController(ICrudServiceAsync<BookModel> bookService)
        {
            _bookService = bookService;
        }

        // [G] Публічний доступ
        [HttpGet]
        [AllowAnonymous]

        public async Task<ActionResult<IEnumerable<BookApiModel>>> GetAll()
        {
            var books = await _bookService.ReadAllAsync();
            var result = books.Select(book => new BookApiModel
            {
                Id = book.Id,
                Title = book.Title,
                Author = book.Author
            });

            return Ok(result);
        }

        // [G] Публічний доступ
        [HttpGet("{id:int}")]
        [AllowAnonymous]

        public async Task<ActionResult<BookApiModel>> Get(int id)
        {
            var book = await _bookService.ReadAsync(id);
            if (book == null)
                return NotFound();

            var dto = new BookApiModel
            {
                Id = book.Id,
                Title = book.Title,
                Author = book.Author
            };

            return Ok(dto);
        }

        // [G] Створення книги – авторизація обов'язкова
    }
}

```

```

[Authorize]
Семантика: 0
public async Task<IActionResult> Create([FromBody] BookCreateModel model)
{
    var newBook = new BookModel
    {
        Title = model.Title,
        Author = model.Author,
        LibraryMemberId = model.LibraryMemberId,
        BookTags = new List<BookTagModel>()
    };

    var created = await _bookService.CreateAsync(newBook);
    if (!created) return BadRequest("Could not create book.");

    await _bookService.SaveAsync();

    return CreatedAtAction(nameof(Get), new { id = newBook.Id }, newBo

// 📖 Створення книги - авторизація обов'язкова
[HttpPost("{id:int}")]
[Authorize]
Семантика: 0
public async Task<IActionResult> Update(int id, [FromBody] BookUpdateModel model)
{
    var book = await _bookService.ReadAsync(id);
    if (book == null)
        return NotFound();

    book.Title = model.Title;
    book.Author = model.Author;
    book.LibraryMemberId = model.LibraryMemberId;

    var updated = await _bookService.UpdateAsync(book);
    if (!updated) return BadRequest("Update failed.");

    await _bookService.SaveAsync();

    return NoContent();
}

// 📖 Вибір книги - авторизація обов'язкова
[HttpGet("{id:int}")]
[Authorize]
Семантика: 0
public async Task<IActionResult> Get(int id)
{
    var book = await _bookService.ReadAsync(id);
    if (book == null)
        return NotFound();

    var deleted = await _bookService.RemoveAsync(book);
    if (!deleted) return BadRequest("Delete failed.");

    await _bookService.SaveAsync();

    return NoContent();
}

// 📖 Тестовий/секретний ендпоінт
[Authorize]
[HttpGet("secret")]
Семантика: 0
public IActionResult GetSecret()
{
    return Ok("Тільки для авторизованих користувачів");
}

// 📖 Умовний ендпоінт для бронювання книги
[Authorize]
[HttpPost("{id:int}/reserve")]
Семантика: 0
public async Task<IActionResult> ReserveBook(int id)
{
    var book = await _bookService.ReadAsync(id);
    if (book == null) return NotFound();

    // Тут можна реалізувати логіку "бронювання"
    return Ok($"Книга '{book.Title}' заброньована.");
}

```

## Auth

POST	/api/Auth/register	^
POST	/api/Auth/login	^

## Books

GET	/api/Books	^
POST	/api/Books	^
GET	/api/Books/{id}	^
PUT	/api/Books/{id}	^
DELETE	/api/Books/{id}	^
GET	/api/Books/secret	^
POST	/api/Books/{id}/reserve	^

## LibraryMembers

GET	/api/LibraryMembers	^
POST	/api/LibraryMembers	^
GET	/api/LibraryMembers/{id}	^
PUT	/api/LibraryMembers/{id}	^
DELETE	/api/LibraryMembers/{id}	^

5. Додайте для створеного вебдодатку ролі, усього ролей повино бути не менше трьох. Оновіть кінцеві точки REST API, що використовують авторизацію, щоб вони виконувалися залежно від ролі клієнта, наприклад у REST API бібліотеки, забронювати книгу може

звичайний користувач, але змінити інформацію про книгу у БД лише бібліотекар.

## Auth

POST

/api/Auth/register

Parameters

No parameters

Request body

application/json

```
{  "email": "user@example.com",  "password": "User123!",  "fullName": "Ivan Ivanov"}  
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \  'http://localhost:5258/api/Auth/register' \  -H 'accept: */*' \  -H 'Content-Type: application/json' \  -d '{  "email": "user@example.com",  "password": "User123!",  "fullName": "Ivan Ivanov"}  '
```

Request URL

http://localhost:5258/api/Auth/register

Server response

Code	Details
200	<div><div>Response body</div><div>User registered successfully.</div><div>Download</div></div> <div><div>Response headers</div><div>content-type: text/plain; charset=utf-8 date: Sat, 24 May 2025 15:34:00 GMT server: Mestrel transfer-encoding: chunked</div></div>

Responses

Code	Description	Links
200	OK	No links



