

# МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. ТАРАСА ШЕВЧЕНКА  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра мережевих та інтернет технологій

## СУЧАСНІ ІНТЕРНЕТ ТЕХНОЛОГІЇ

### СТРУКТУРА ПРОЕКТУ ASP.NET CORE ВЕБ-ЗАСТОСУНКУ. РОЗДІЛЕННЯ МОНОЛІТУ НА МОДУЛІ (ШАРИ)

#### Лабораторне заняття № 1

Черкун Єви Сергіївни

#### Хід виконання роботи:

##### 1.1 Реалізація MVC шаблону та початок роботи в ASP.NET Core

Створюємо проєкт у Microsoft Visual Studio за шаблоном ASP.NET Core Web app (Model-View-Controller), де в типі автентифікації вказується Individual Accounts.

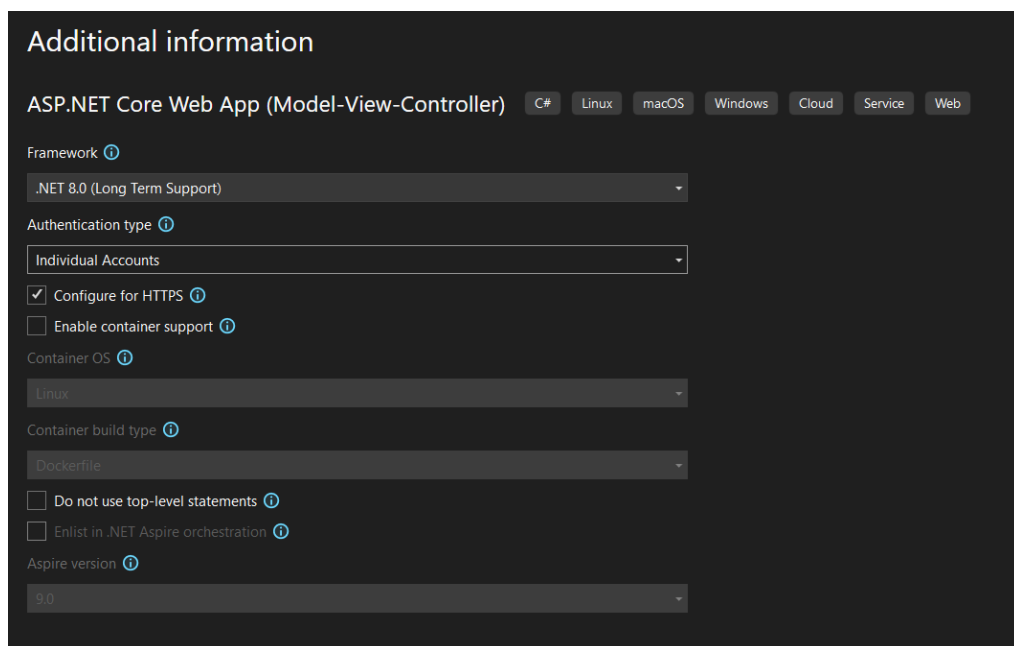


Рис 1.1.1 – Вибір типу автентифікації ASP.NET Core веб-застосунку

##### 1.2 Розділення ASP.NET Core застосунку на модулі

Для того щоб розділити застосунок ASP.NET на модулі потрібно перенести контекст даних та моделі даних в окремий проєкт в складі одного й того ж рішення. Для цього потрібно натиснути на Solution (рішення) у контекстному меню проєкту та створити новий проєкт (SlayLib).

У веб-проєкті є папки **Models** та **Data** із проєкту ASP.NET Core (WebAppCore), які треба перенести у новостворений проєкт.

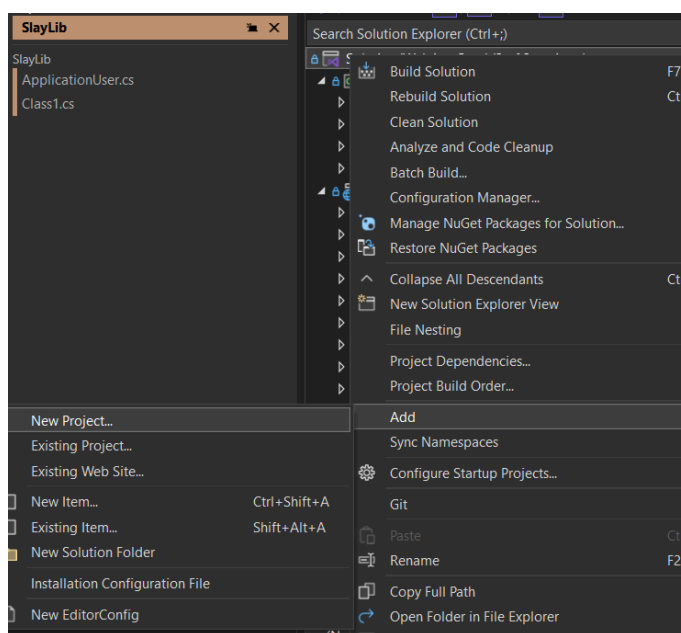


Рис 1.2.1 – Створення нового проєкту

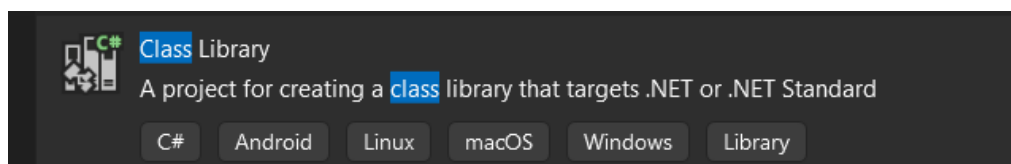


Рис 1.2.2 – Додавання нового модуля у вигляді проєкту бібліотеки класів

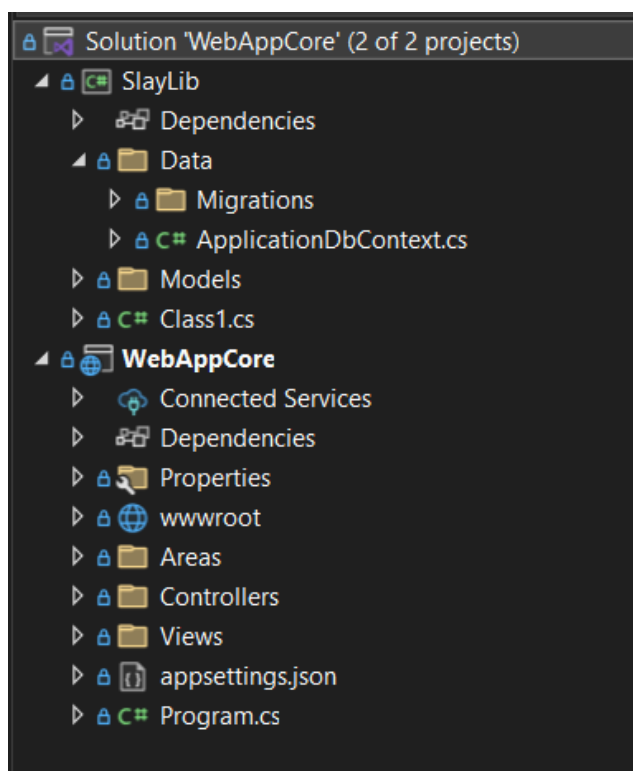


Рис 1.2.3 – Розділений ASP.NET Core застосунок на модулі

Забезпечуємо належні ідентифікатори просторів імен класів нового модуля. Корегуємо простір імен перенесених класів так, щоб вони відповідали новому розташуванню файлів.

Для класу `ApplicationDbContext`, що знаходиться в однойменному файлі папки `Data` модуля `SlayLib`, належним простором імен буде `SlayLib.Data`

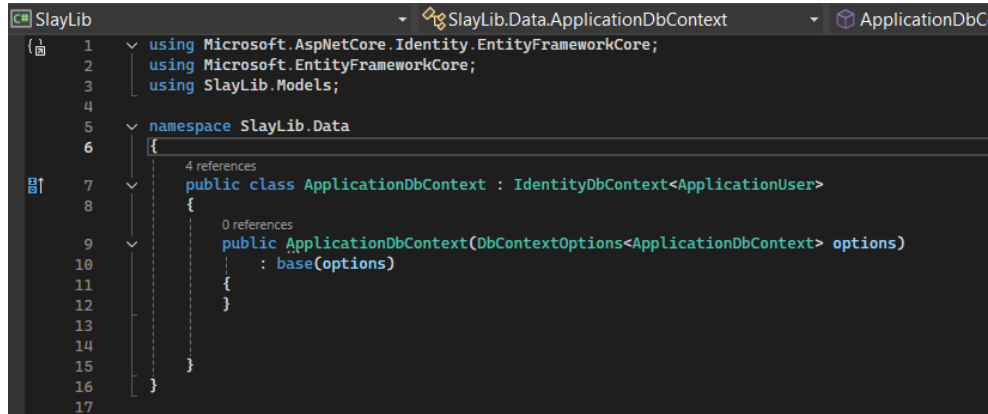


Рис 1.2.4 – Скорегований ідентифікатор просторів імен переміщених класів

Також важливо встановити необхідні бібліотеки. Встановлюємо їх менеджером пакетів `NuGet`.

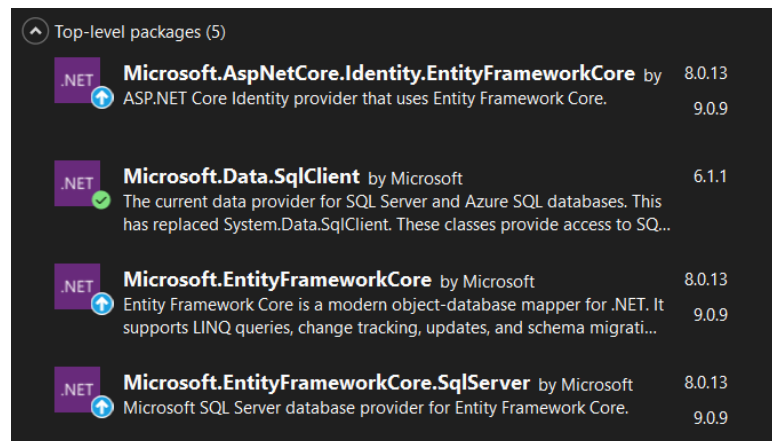


Рис 1.2.5 – Встановлення необхідних бібліотек у менеджері Nuget

Підключаємо проєкт `SlayLib` до `WebAppCore`. Для цього переходимо у каталог вебзастосунку, надисकाємо на `Dependencies` -> `Project reference` і додаємо посилання:

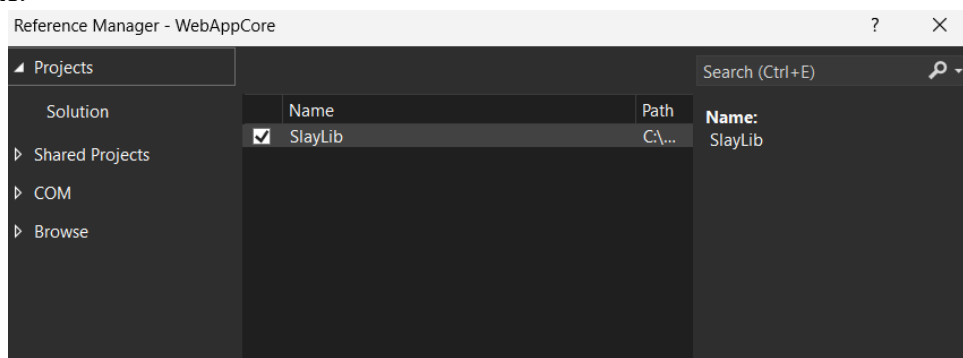


Рис 1.2.6 – Забезпечення доступності контексту та моделей у веб-застосунку

### 1.3 Розширення класу автентифікованого користувача Identity User

У папці Models створюємо клас для автентифікованого користувача та додаємо поля First Name та Last Name.

```
1 using Microsoft.AspNetCore.Identity;
2
3 namespace SlayLib.Models
4 {
5     1 reference
6     public class ApplicationUser : IdentityUser
7     {
8         0 references
9         public string FirstName { get; set; }
10        0 references
11        public string LastName { get; set; }
12    }
13 }
```

### 1.4 Виконання міграцій, перевірка та оновлення структури таблиць відповідної бази даних

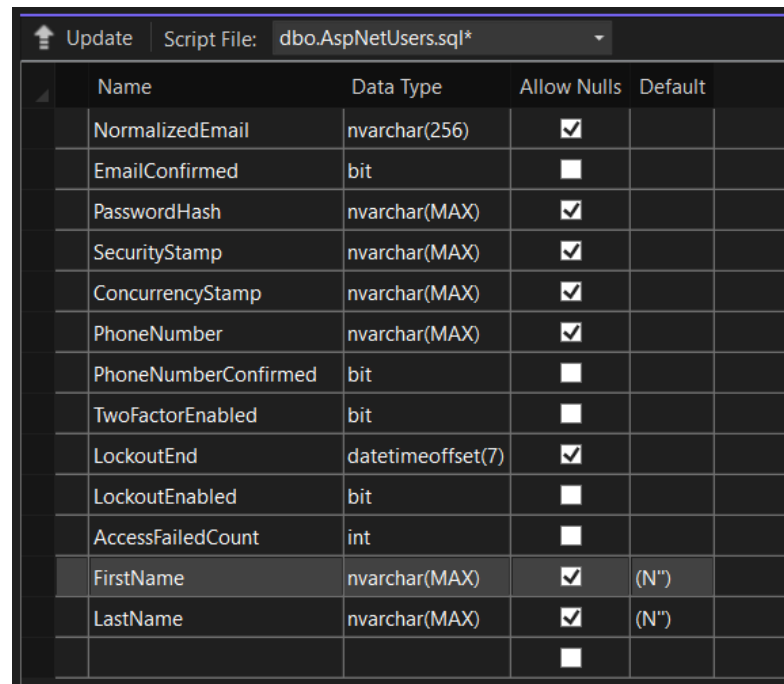
Створено міграцію для внесення змін у модель користувача (ApplicationUser) з доданими полями `FirstName` та `LastName`:

```
Package Manager Console
Package source: All
Default project: SlayLib

PM> Add-Migration InitIdentity
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM> Update-Database
Build started...
Build succeeded.
Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (17ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  SELECT 1
Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (11ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  SELECT OBJECT_ID(N'[_EFMigrationsHistory]');
Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (1ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  SELECT 1
Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (0ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  SELECT OBJECT_ID(N'[_EFMigrationsHistory]');
Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (2ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  SELECT [MigrationId], [ProductVersion]
  FROM [_EFMigrationsHistory]
  ORDER BY [MigrationId];
Microsoft.EntityFrameworkCore.Migrations[20402]
  Applying migration '20251007171151_InitIdentity'.
Applying migration '20251007171151_InitIdentity'.
Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (53ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  ALTER TABLE [AspNetUsers] ADD [FirstName] nvarchar(max) NOT NULL DEFAULT N'';
Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (3ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  ALTER TABLE [AspNetUsers] ADD [LastName] nvarchar(max) NOT NULL DEFAULT N'';
Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (7ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  INSERT INTO [_EFMigrationsHistory] ([MigrationId], [ProductVersion])
  VALUES (N'20251007171151_InitIdentity', N'8.0.13');
Done.
PM> |
```

Рис 1.4.1 – Успішне виконання міграцій

Перевіряємо таблицю бази даних та бачимо створені поля, що додали в класі:



The screenshot shows the 'Script File' window for 'dbo.AspNetUsers.sql\*'. It displays a table with the following columns: Name, Data Type, Allow Nulls, and Default. The table contains 15 rows of data.

Name	Data Type	Allow Nulls	Default
NormalizedEmail	nvarchar(256)	<input checked="" type="checkbox"/>	
EmailConfirmed	bit	<input type="checkbox"/>	
PasswordHash	nvarchar(MAX)	<input checked="" type="checkbox"/>	
SecurityStamp	nvarchar(MAX)	<input checked="" type="checkbox"/>	
ConcurrencyStamp	nvarchar(MAX)	<input checked="" type="checkbox"/>	
PhoneNumber	nvarchar(MAX)	<input checked="" type="checkbox"/>	
PhoneNumberConfirmed	bit	<input type="checkbox"/>	
TwoFactorEnabled	bit	<input type="checkbox"/>	
LockoutEnd	datetimeoffset(7)	<input checked="" type="checkbox"/>	
LockoutEnabled	bit	<input type="checkbox"/>	
AccessFailedCount	int	<input type="checkbox"/>	
FirstName	nvarchar(MAX)	<input checked="" type="checkbox"/>	(N'')
LastName	nvarchar(MAX)	<input checked="" type="checkbox"/>	(N'')
		<input type="checkbox"/>	

Рис 1.4.2 - таблиця відповідної бази даних

Після переходу на власну модель користувача ApplicationUser у веб-застосунку виникли помилки, пов'язані з реєстрацією, логіном та логаутом.

Основна причина — залишки старого типу IdentityUser у PageModels (Register, Login, Logout), через що система не могла підключити сервіси Identity (UserManager та SignInManager) для нового типу користувача. Проблему було усунуто шляхом заміни всіх згадок IdentityUser на ApplicationUser у PageModels та перевірки DI у Program.cs, а також виконання відповідних міграцій для бази даних, що забезпечило правильне збереження та аутентифікацію користувачів. Після проведених змін користувачі можуть реєструватися і логінитися:

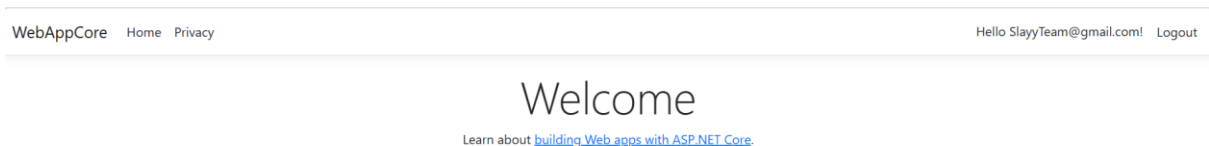


Рис 1.4.3 – Головна сторінка веб-застосунку після входу користувача

## 1.5 Створення репозиторію на GitHub проєкту

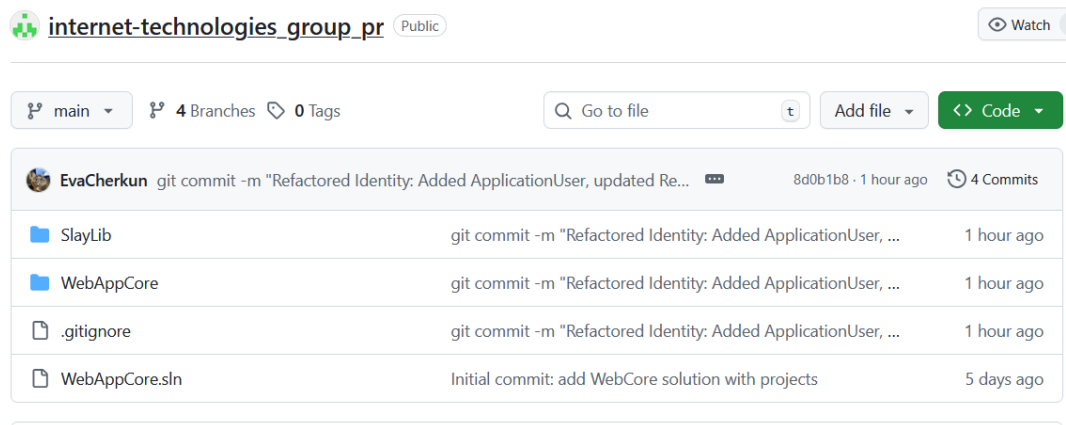


Рис – Створений репозиторій на GitHub

### Висновок:

У ході виконання роботи було створено рішення Visual Studio з проєктом ASP.NET Core та автентифікацією Individual Accounts, розширено клас IdentityUser властивостями FirstName та LastName, модульовано застосунок шляхом винесення контексту даних і моделей у окремий проєкт Slay.Lib з правильними просторами імен та забезпеченою досяжністю для веб-застосунку, виконано міграцію бази даних за допомогою Entity Framework Core та перевірено структуру таблиць, розміщено проєкт на GitHub із зафіксованими змінами та оформлено звіт із описом виконаних дій і скріншотами, що демонструє вміння працювати з ASP.NET Core Identity, EF Core, багатопроєктною структурою та системою контролю версій Git.