Лабораторна робота №4

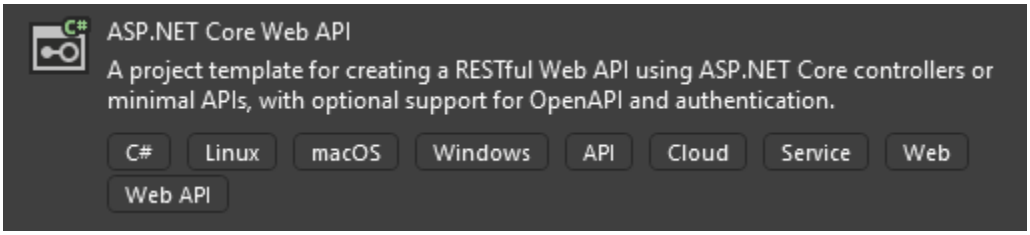
Розробка вебсерверів засобами фреймворку ASP.NET.

Виконав студент 301-пТК Тараненко Олександр

Передумова

Виконайте лабораторну роботу №3. Створіть нову гілку у раніше створеному гіт репозиторії *NUPP_NET_2025_{Номер Групи}_ТК_{Прізвище}_Lab* із назвою **lab4**, яка міститиме код із третьої лабораторної роботи.
Для здачі лабораторної роботи, необхідно буде створити пул реквест із гілки **lab4** у **master**, або якщо на момент здачі пул реквест із гілки lab3 у master не закритий - із гілки lab4 у lab3.

Завдання

1. Створити проєкт за шаблоном ASP.NET Core Web API {Назва тематики}.REST використовуючи Visual Studio або .NET SDK:
   `dotnet new webapi -n {Назва тематики}.REST`
   

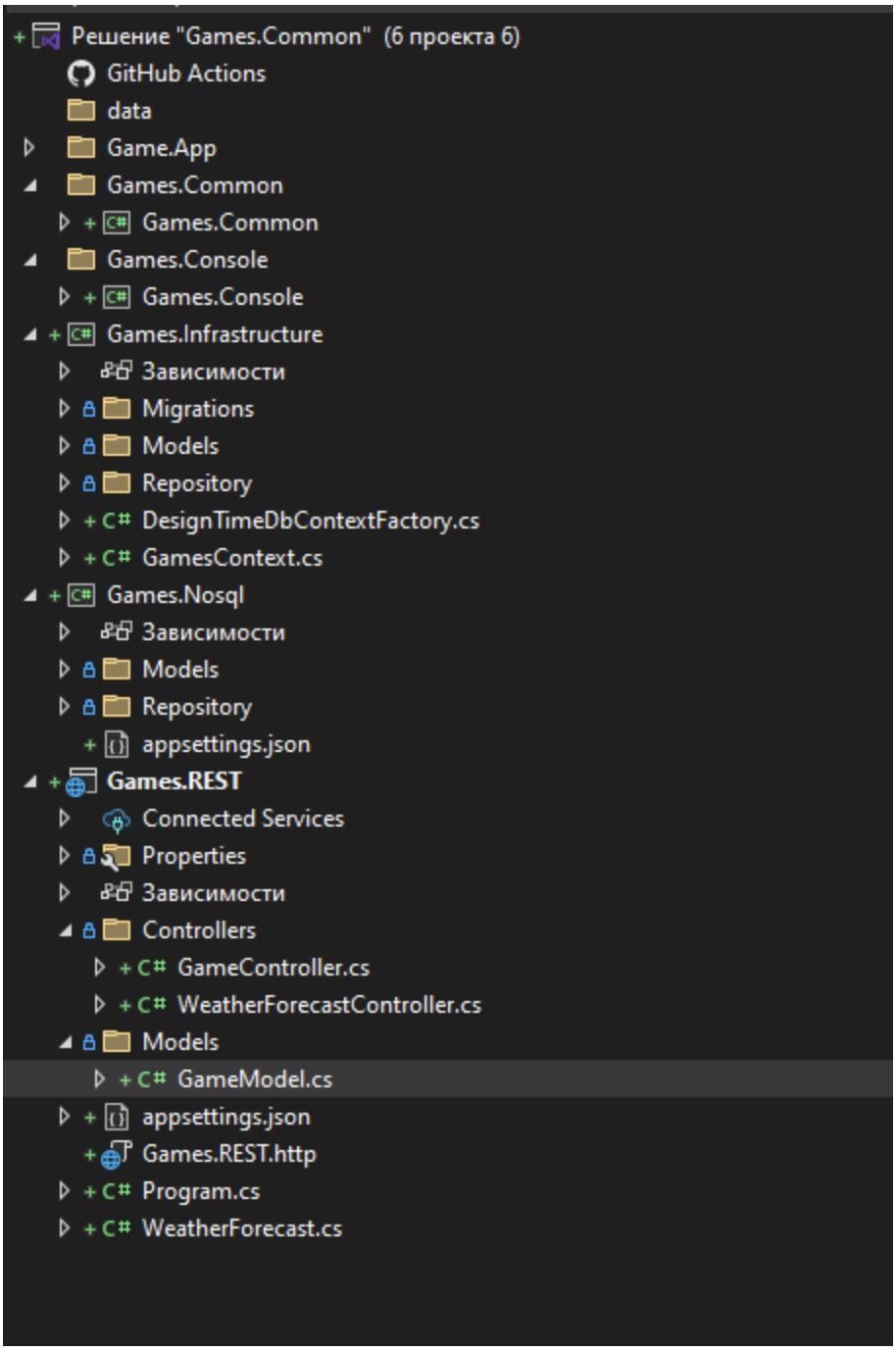
2. Створити папку Models, у якій будуть зберігатися моделі, які будуть використовуватися контролерами у проєкті {Назва тематики}.REST.
3. Створіть контролери у папці Controllers, які будуть описувати REST API для виконання CRUD операцій (створення, читання, оновлення, видалення) над моделями у вашій тематиці, які ви зберігали у базі даних у третій лабораторній, при цьому моделі, що використовуються у контролерах, повинні бути описані у проєкті {Назва тематики}.REST, щоб відповідати 3-рівневій архітектурі. Наприклад, якщо у вас є сутність Bus, що зберігається у БД, ви створите новий клас BusModel у папці Models нового проєкту, який використовуватиметься у контролерах. Також зверніть увагу, щоб моделі у контролерах, мали лишень необхідні властивості, та якщо якісь властивості зайві, для якихось із методів - створіть нову модель без цих властивостей.
4. При проєктуванні Web API зверніть увагу, щоб створений API відповідав 4-ій умові побудови REST-додатку по Філдингу - однорідності інтерфейсу (див. 31 слайд), тобто використовувалися унікальні назви сутностей, правильні HTTP методи та поверталися правильні HTTP коди у HTTP відповідях, наприклад 404 якщо сутність не знайдена, або 201 якщо нова сутність створена.

5. Використовуйте асинхрону версію дженерік CRUD сервісу, що використовує репозиторій для доступу до даних та який був реалізований у 3ій лабораторній роботі, у створених контролерах.:

```
public interface ICrudServiceAsync<T>
{
	public Task<bool> CreateAsync(T element);
	public Task<T> ReadAsync(Guid id);
	public Task<IEnumerable<T>> ReadAllAsync();
	pub
lic Task<IEnumerable<T>> ReadAllAsync(int page, int amount);
	public Task<bool> UpdateAsync(T element);
	public Task<bool> RemoveAsync(T element);
	public Task<bool> SaveAsync();
}
```

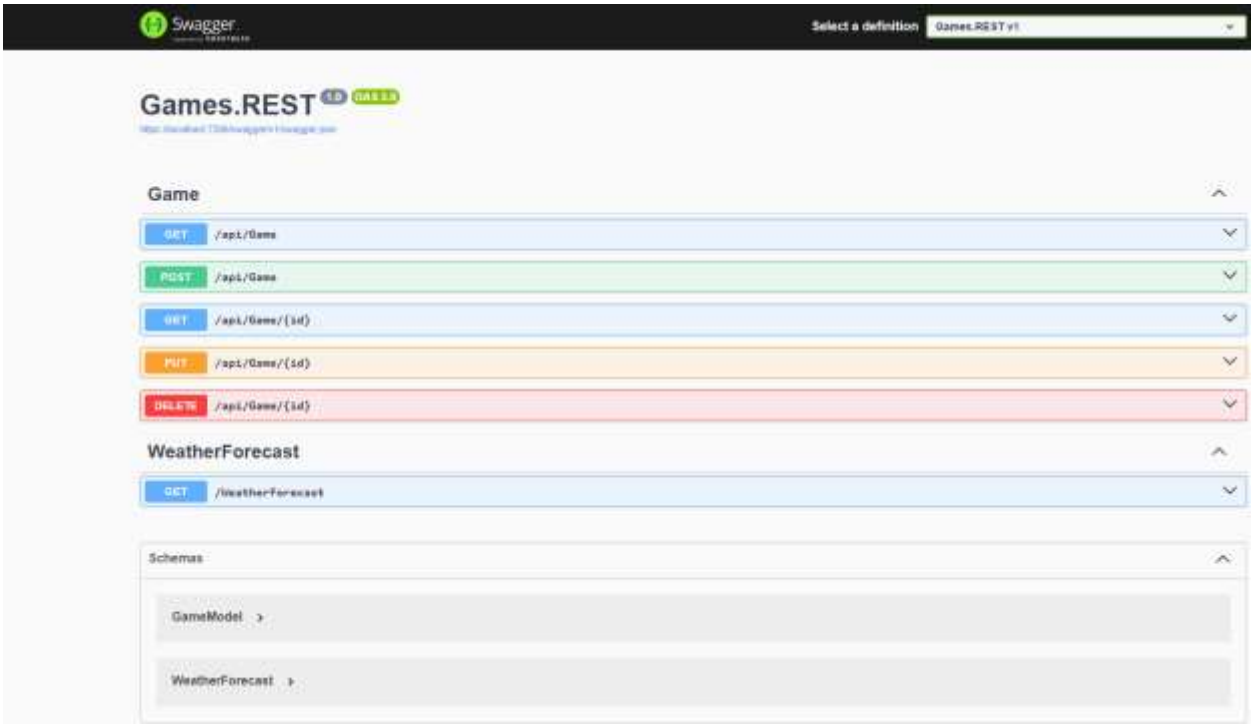
   Імплементацію CRUD сервісу, репозиторіїв та контексту, додавайте у контролери використовуючи вбудовану у ASP.NET Core підтриму впровадження залежностей(Dependency Injection).

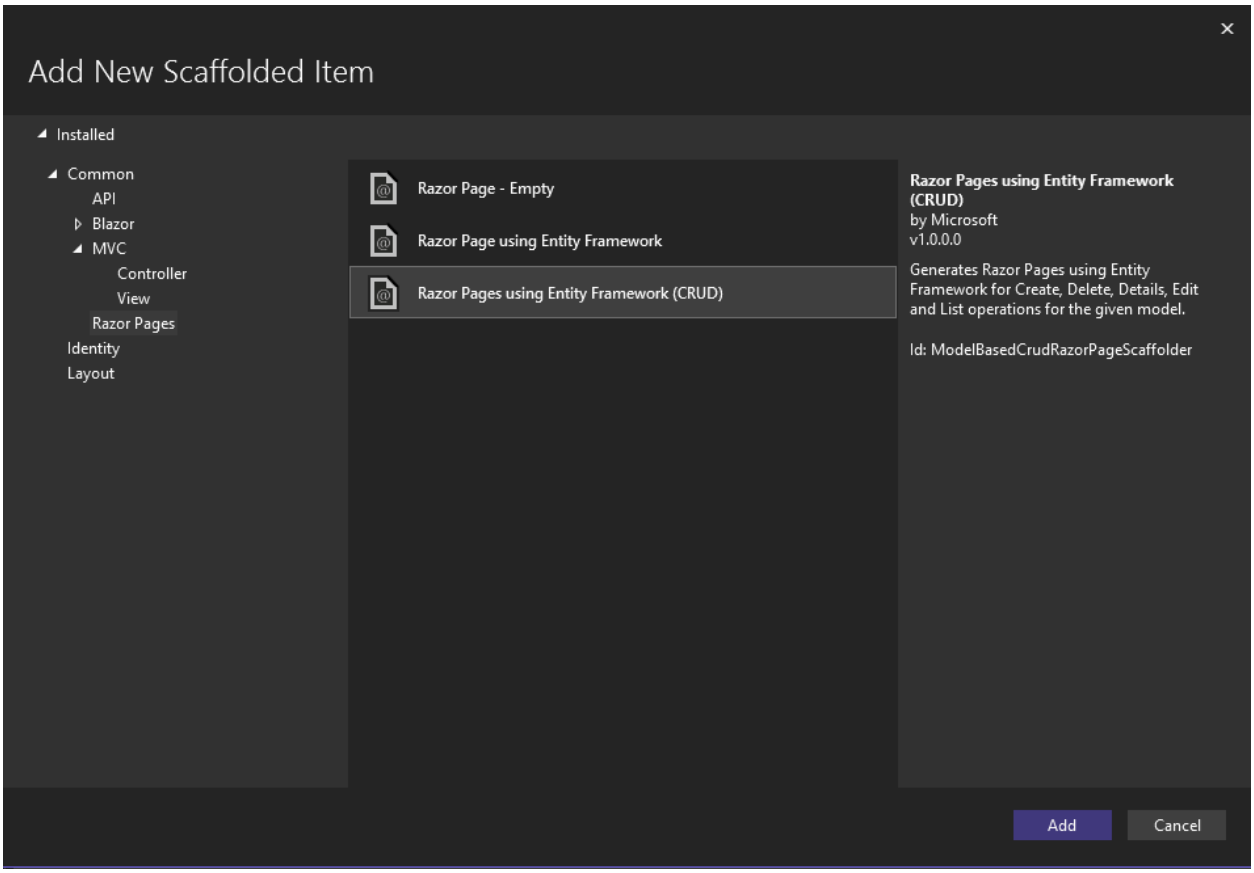
До PR готової лабораторної роботи додайте PDF файл у якому будуть результати виконання запитів, використовуючи створений REST API. Можете скористатися сторінкою Swagger, яка генерується ASP.NET застосунком, або застосунком Postman.


Решение "Games.Common" (6 проекта 6)
GitHub Actions
data
Game.App
Games.Common
Games.Common
Games.Console
Games.Console
Games.Infrastructure
Зависимости
Migrations
Models
Repository
DesignTimeDbContextFactory.cs
GamesContext.cs
Games.Nosql
Зависимости
Models
Repository
appsettings.json
Games.REST
Connected Services
Properties
Зависимости
Controllers
GameController.cs
WeatherForecastController.cs
Models
GameModel.cs
appsettings.json
Games.REST.http
Program.cs
WeatherForecast.cs

Структура

```
C:\Users\Monolit\Desktop\.NET\NUPP_NET_2025_301pTK_TK_Taranenko_Lab\Lab4\Games.Common\Games.REST\bin\Debug\net9.0\Games.REST.exe
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7286
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5155
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: C:\Users\Monolit\Desktop\.NET\NUPP_NET_2025_301pTK_TK_Taranenko_Lab\Lab4\Games.Common\Games.RES
T
```





Контрольні запитання

1. WIP

WIP — це скорочення від англійського **Work In Progress**, що означає **"робота в процесі"** або **"у процесі виконання"**.

У програмуванні / розробці — код або функція, яка ще не завершена, але над нею ведеться робота.