

Міністерство освіти і науки України  
Харківський національний університет радіоелектроніки

Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

КУРСОВА РОБОТА  
ПОЯСНЮВАЛЬНА ЗАПИСКА  
з дисципліни “ Об’єктно -орієнтоване програмування”

## ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліні “Об’єктно -орієнтоване програмування”

Керівник , Професор кафедри програмної інженерії    Бондарєв В.М.

Студент гр. ПЗПІ-22-1

Корецький І.О.

Комісія:

Проф. Бондарев В.М.,  
Ст. викл. Черепанова Ю.Ю.,  
Ст. викл. Ляпота В.М.

Ст. викл. Черепанова Ю.Ю.,

Ст. викл. Ляпота В.М.

Харків 2023

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
РАДІОЕЛЕКТРОНІКИ

Кафедра *програмної інженерії*

Рівень вищої освіти *перший (бакалаврський)*

Дисципліна *Об'єктно-орієнтоване програмування*

Спеціальність *121 Інженерія програмного забезпечення*

Освітня програма: *Програмна інженерія*

Курс 1. Група ПЗП-22 -1. Семестр 2

**ЗАВДАННЯ**

*на курсовий проект студента*

*Корецького Ігора Олександровича*

- 1 Тема проекту: Менеджер проектів
- 2 Термін здачі студентом закінченого проекту: **“16” - червня - 2023 р.**
- 3 Вихідні дані до проекту:  
Специфікація програми, методичні вказівки до виконання курсової роботи
- 4 Зміст розрахунково-пояснювальної записки:  
Вступ, опис вимог, проектування програми, інструкція користувача, висновки

## КАЛЕНДАРНИЙ ПЛАН

<i>Назва етапу</i>	<i>Термін виконання</i>
Видача теми, узгодження і затвердження теми	13.02.2023 - 14.03.2023 р.
Формулювання вимог до програми	23.05.2023 – 24.05.2023 р.
Розробка моделей	24.05.2023 – 25.05.2023 р.
Розробка функцій ...	25.05.2023 – 3.06.2023 р.
Розробка функцій зберігання та завантаження даних	3.06.2023 р.
Тестування і доопрацювання розробленої програмної системи.	4.06.2023 – 08.06.2023 р.
Оформлення пояснювальної записки, додатків, графічного матеріалу	8.06.2023 – 09.06.2023 р.
Захист	9.06.2023 р.

Студент

Корецький Ігор Олександрович

Керівник

Бондарев Володимир Михайлович

« 8 » червня 2023 р.

## РЕФЕРАТ

Пояснювальна записка до курсової роботи: 35 с., 34 рис., 1 джерело.

КЛАС, КОЛЕКЦІЯ, МЕТОД, МОВА ПРОГРАМУВАННЯ C#, ПРОЕКТИ, КОРИСТУВАЧІ, ЗАВДАННЯ, ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, КОНСОЛЬНИЙ ЗАСТОСУНОК.

Метою роботи є розробка програми «Менеджер проектів», яка буде надавати ІТ бізнесу можливість організації проектів.

В результаті отримана програма, що дозволяє зберігати проекти, задачі та користувачів. У програмі є розділення користувачей за ролями, яке надає різний рівень доступу. Програма надає можливість створювати проекти й завдання, а також редагувати їх. Є реалізація реєстрації користувачей. Є можливість додавати файли до завдання, а також слідкувати за прогресом.

В процесі розробки використано середовища Microsoft Visual Studio 2022, платформи .NET 7.0, мова програмування C#.

## ЗМІСТ

<b>ВСТУП .....</b>	<b>6</b>
<b>1 СПЕЦИФІКАЦІЯ ПРОГРАМИ.....</b>	<b>7</b>
1.1 ОСНОВНІ ФУНКЦІЇ ПРОГРАМИ .....	7
1.2 ОПИС ФУНКЦІЙ ПРОГРАМИ .....	7
1.2.1 Функція «Sign Up» .....	7
1.2.2 Функція «Sign In».....	7
1.2.3 Функції «UserMenu» .....	8
1.2.3.1 Функція «Log Out».....	8
1.2.3.2 Функція «Choose Project» .....	8
1.2.3.2.1 Функції «DutyIdentifier» .....	8
<b>2 ПРОЕКТУВАННЯ ПРОГРАМИ .....</b>	<b>11</b>
2.1 ОБ'ЄКТНА СТРУКТУРА ПРОГРАМИ .....	11
2.2 ДОКЛАДНИЙ ОПИС КЛАСІВ ТА ЇХ ВЗАЄМОДІЯ .....	13
2.2.1 Перелічувальні типи .....	14
2.2.1.1 Enum Duty .....	14
2.2.1.2 Enum Priority .....	14
2.2.1.3 Enum Status .....	15
2.2.2 Базові класи .....	16
2.2.2.1 Клас Attachment.....	16
2.2.2.2 Клас BaseEntity .....	16
2.2.2.3 Клас Project .....	17
2.2.2.4 Клас Task.....	18
2.2.2.5 Клас User .....	19
2.2.2.6 Клас UserProjectRole .....	20
2.2.3 Інтерфейс IStorage<T> .....	21
2.2.4. Інтерфейси сервісів .....	22
2.2.4.1 Інтерфейс IGenericService<T> .....	22
2.2.2.4.2 Інтерфейс IProjectService .....	23
2.2.2.4.3 Інтерфейс ITaskService .....	23
2.2.2.4.4 Інтерфейс IUserProjectRoleService .....	24
<b>3 ІНСТРУКЦІЯ КОРИСТУВАЧА .....</b>	<b>24</b>
3.1 ВСТАНОВЛЕННЯ ПРОГРАМИ .....	24

3.2	ФУНКЦІОНАЛ ПРОГРАМИ ТА ІНСТРУКЦІЯ ДО НЕЇ .....	25
3.2.1	Робота зі сторінкою «Sign Up».....	25
3.2.2	Робота зі сторінкою «Sign In» .....	26
3.2.3	Робота з меню користувача .....	26
3.2.3.1	Меню «StateManager» .....	27
3.2.3.1.1	Меню редагування проекту .....	28
3.2.3.2	Меню «Developer» та «Tester» .....	32
3.2.3.2.1	Меню взаємодії із задачею .....	33
<b>ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....</b>		<b>35</b>

## ВСТУП

Метою цієї курсової роботи є створення консольного застосунку менеджер проектів. Для цього були реалізовані наступні задачі:

- Реєстрація та авторизація користувачів;
- Створення, редагування та завершення задач;
- Відстеження прогресу виконання задач;
- Управління проектами;
- Надсилання сповіщень користувачам про зміни у статусах задач

В процесі розробки використано Microsoft Visual Studio, платформу .NET 7.0, мову програмування C#.

## 1 СПЕЦИФІКАЦІЯ ПРОГРАМИ

### 1.1 Основні функції програми

Завдяки програмі «Менеджер Проектів», користувач із тегом «StateManager» має змогу додавати користувачей до проекту, а також «StateManager» має доступ до редагувань завдань та проектів. Інші користувачі мають обмежений вплив на роботу програми. Вони не можуть редагувати ніякі параметри проектів та завдань. Проте, вони можуть додавати файли до завдань, дивитися їх, а також можуть передати проект іншому працівнику на проекті.

### 1.2 Опис функцій програми

При запуску програми, користувач обирає одну з трьох дій («Sign Up»(1), «Sign In»(2), «Exit» - вихід із програми).

#### 1.2.1 Функція «Sign Up»

Коли користувач обирає «Sign Up», запускається однойменний метод. Далі користувач має ввести всі данні, які попросить програма. У разі, якщо це перший користувач у програмі, програма також попросить створити проект, а також назначить нового користувача менеджером стану(State Manager). Після того як нового користувача буде створено, користувач автоматично перейде до меню («UserMenu»(3)).

#### 1.2.2 Функція «Sign In»

Коли користувач обирає «Sign In», запускається однойменний метод. Тут користувач має ввести адресу своєї електронної пошти, а



також пароль для входу в акаунт. У разі якщо всі введені дані правильні, буде активовано метод «UserMenu»(3).

### 1.2.3 Функції «UserMenu»

На цьому етапі, користувач побачить повідомлення (якщо вони є), а також механізм вибору наступних дій («Log Out»(1), «Choose Project»(2), «See Notifications»(знову показує повідомлення)).

#### 1.2.3.1 Функція «Log Out»

Ця опція активує метод «PerformOperationsAsync», який переносить користувача в початок програми.

#### 1.2.3.2 Функція «Choose Project»

Користувач обирає проект через метод інтерфейсу проекту «GetProjectAsync». Програма отримує тег користувача на цьому проекті, та у разі якщо всі попередні операції були виконані правильно, програма активує метод «DutyIdentifier».

##### 1.2.3.2.1 Функції «DutyIdentifier»

«DutyIdentifier» відкриває UI згідно до отриманого раніше тегу.

«DutyIdentifier» може привести до:

- Інтерфейс менеджера стану («StateManagerUI»(1));
- Інтерфейс розробнику («DeveloperUI»(2));
- Інтерфейс тестеру («TesterUI»(3));

##### 1.2.3.2.1.1 Функції «StateManagerUI»

Першим чином, користувач побачить метод «PerformOperationsAsync», який дозволить обрати один із цих варіантів:

- Додати проект («Add Project»);
- Змінити обраний проект («Edit current Project»(1));
- Вийти («Exit» - повертає до попереднього методу)

#### 1.2.3.2.1.1.1 Функції «Edit current Project»

Користувач переходить до меню, де він зможе обрати одну опцій:

- Змінити назву проекту («Edit Project Name»);
- Змінити опис проекту («Edit Project Description»);
- Створити задачу («Create Assignment»);
- Налаштувати задачі («Edit Assignments»(1));
- Налаштувати працівників («Edit Workers»(2));
- Вийти («Exit» - повертає до попереднього методу)

#### 1.2.3.2.1.1.1.1 Функції «Edit Assignments»

Користувач обирає задачу, яку він буде змінювати. Далі користувач має обрати одну з опцій:

- Назначити користувача на завдання («Assign worker to this Assignment»);
- Закрити задачу («Close Assignment»);
- Вийти («Exit» - повертає до попереднього методу)

#### 1.2.3.2.1.1.2 Функції «Edit Workers»

Користувач переходить до меню вибору взаємодії з працівниками.

- Прибрати користувача із проекту («Remove Worker from Project»);
- Додати користувача до проекту («Assign Worker for Project»);
- Дати тег («Assign Duty»);
- Вийти («Exit» - повертає до попереднього методу)

#### 1.2.3.2.1.1 Функції «DeveloperUI»

Першим чином, користувач побачить метод «PerformOperationsAsync», який дозволить обрати один із цих варіантів:

- Обрати задачу («Choose Assignment»(1));
- Вийти («Exit» - повертає до попереднього методу)

#### 1.2.3.2.1.1.1 Функції «Choose Assignment»

Працівник обирає задачу. Далі працівник може виконувати певні дії із цією задачею. Доступні дії:

- Передати задачу («Hand Assignment»);
- Завантажити файл («Upload File»);
- Відкрити у провіднику папку із файлом («Open File Folder»);
- Вийти («Exit» - повертає до попереднього методу)

## 2 ПРОЕКТУВАННЯ ПРОГРАМИ

### 2.1 Об'єктна структура програми

Програма написана із дотриманням правил архітектури застосунку. Згідно до цих правил програма поділяється на наступні компоненти: DAL (Data Access Layer) => DAL.Abstractions => BLL => BLL.Abstractions => UI. Також в цій системі передбачено додаткове відокремлення, тому в моєму коді також є Core та Helpers.

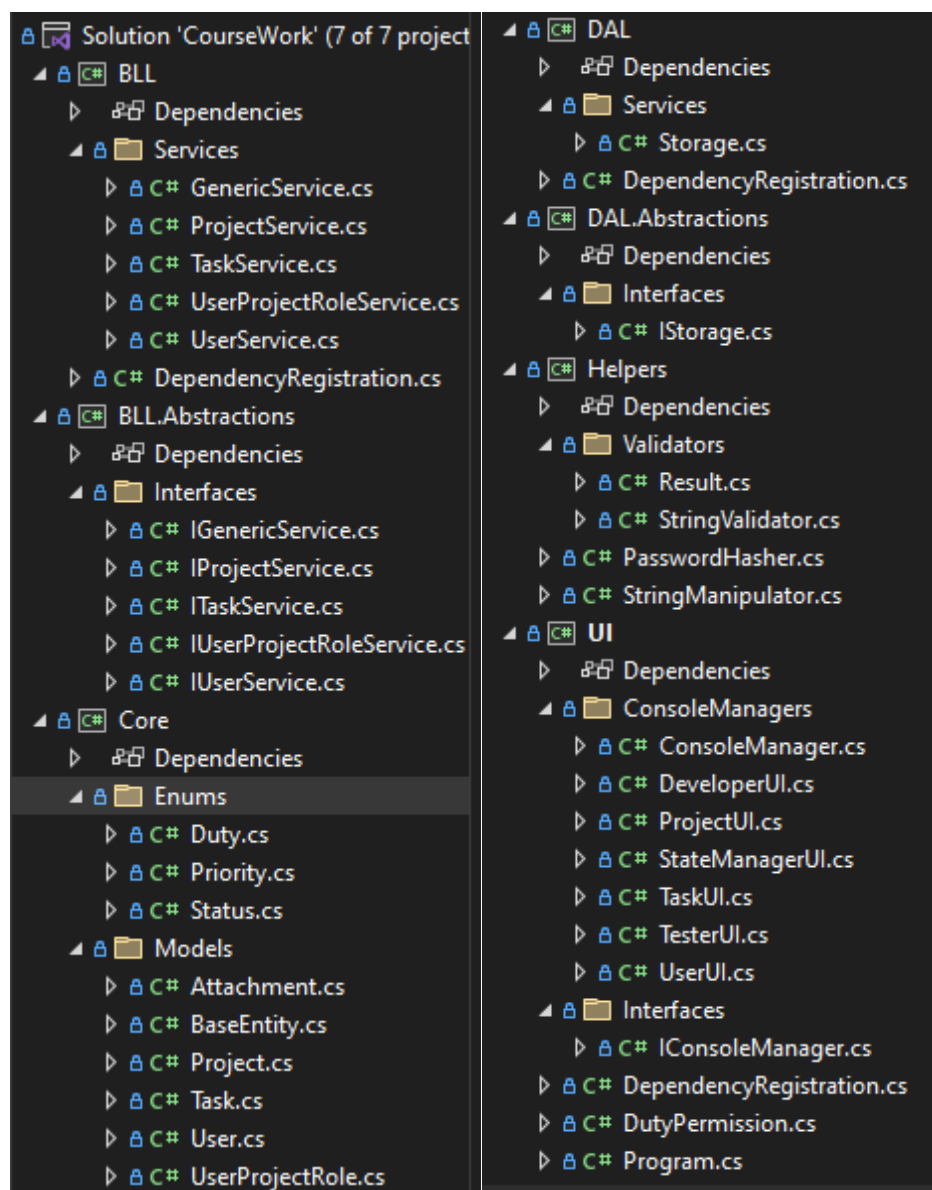


Рис. 2.1, 2.2 – Структура

Від абстрактного класу BaseEntity успадковуються всі класи в

Core.Models (окрім Attachments). Єдиною властивістю цього класу є `int` поле `Id`. Всі інші класи створюються задля використання в усьому проекті. Також в Core є папка із Enum.

В DAL знаходиться код, відповідальний за серіалізацію до Json. Всі методи для роботи програми з Json прописані в дженерік класі `Storage<T>` (`T : BaseEntity`), який успадковується від дженерік інтерфейсу `ISStorage<T>` (знаходиться у DAL. Abstractions). У роботі Storage також приймає участь клас `Result` із `Helpers.Validators`. Цей клас відповідає за обробку помилок. Таким чином для того щоб користуватись методами Storage, наступний слой BLL, користується інтерфейсом.

В BLL знаходиться логіка програми у форматі класів сервісу. Головним у BLL є абстрактний клас `GenericService<T>`, так як від нього успадковуються всі інші сервіси. Він використовується для доступу до методів роботи із Json. Сам `GenericService<T>` успадковується від інтерфейсу `IGenericService<T>`, який знаходиться в BLL. Abstractions. Всі сервісні класи, також успадковуються від своїх інтерфейсів, котрі знаходяться у BLL. Abstractions. Інтерфейси у BLL. Abstractions в свою чергу успадковуються від `IGenericService<T>`. Таким доступ до сервісів із UI буде проходити через інтерфейси.

В UI знаходяться класи, які відповідають за роботу з консоллю (у папці `ConsoleManagers`). Тут також є дженерік клас `ConsoleManager<T>`, та його інтерфейс `IConsoleManager<T>`. `ConsoleManager` використовується для взаємодії із Json. Також тут є клас `DutyPermission`, який відповідає за запуск UI відповідно до тегу користувача.

Окремо виділяю присутній на всіх рівнях програми клас `DependencyRegistration`. Клас `DependencyRegistration` відповідає за реєстрацію залежностей у контейнері залежностей. Це означає, що він вказує, які класи або компоненти залежать від інших класів або

компонентів, а також як їх створити та внедрити. На кожному рівні програми, локальний `DependencyRegistration` реєструє залежності, присутні на рівні.

## 2.2 Докладний опис класів та їх взаємодія

Core:

Перелічувальний тип даних

Enums (3):

- Duty
- Priority
- Status

Базові класи

Models (6):

- Attachment
- BaseEntity
- Project
- Task
- User
- UserProjectRole

DAL.Abstractions:

Інтерфейси

Interfaces (1):

- `IStorage<T>`

BLL.Abstractions:

Інтерфейси

Interfaces (5):

- `IGenericService<T>`
- `IProjectService`
- `ITaskService`

- IUserProjectRoleService
- IUserService

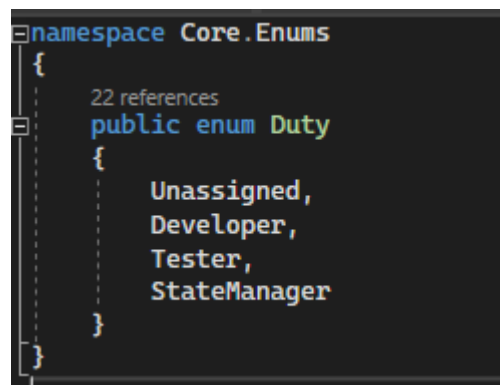
Та класи DependencyRegistration, UI та реалізацією (сервісні).

### 2.2.1 Перелічувальні типи

Ці Enum використовуються як поля для деяких моделей.

#### 2.2.1.1 Enum Duty

Enum із тегами для працівників:



```
namespace Core.Enums
{
    22 references
    public enum Duty
    {
        Unassigned,
        Developer,
        Tester,
        StateManager
    }
}
```

Рис. 2.3 – Enum Duty

Складається з:

- Unassigned
- Developer
- Tester
- StateManager

#### 2.2.1.2 Enum Priority

Enum із пріоритетом для задач:

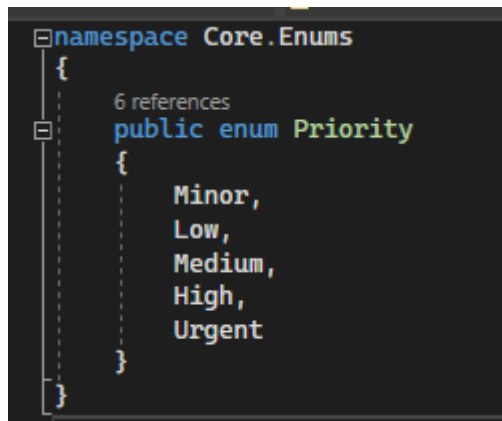


Рис. 2.4 – Enum Priority

Складається з:

- Minor
- Low
- Medium
- High
- Urgent

#### 2.2.1.3 Enum Status

Enum із статусами задач:

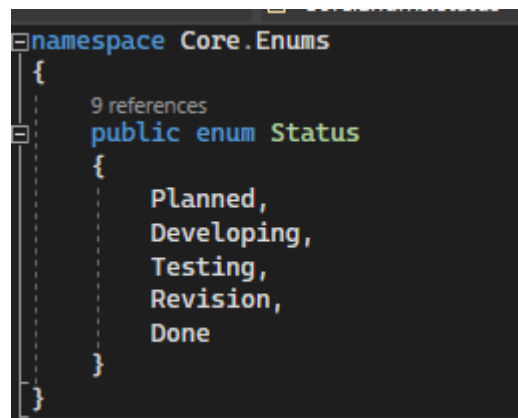


Рис. 2.5 – Enum Status

Складається з:

- Planned
- Developing
- Testing



- Revision
- Done

## 2.2.2 Базові класи

### 2.2.2.1 Клас Attachment

```
namespace Core.Models
{
    6 references
    public class Attachment
    {
        3 references
        public string FileName { get; set; }
        1 reference
        public long Size { get; set; }
        1 reference
        public byte[] Data { get; set; }
    }
}
```

Рис. 2.6 – Клас Attachment

Складається з:

- Публічної властивості FileName (string)
- Публічної властивості Size (long)
- Публічної властивості Data (byte[])

Цей клас використовується для збереження файлів, які загрузає користувач.

### 2.2.2.2 Клас BaseEntity

```
namespace Core.Models
{
    10 references
    public abstract class BaseEntity
    {
        43 references
        public int Id { get; set; }
    }
}
```

Рис. 2.7 – Клас BaseEntity

Складається з:

- Публічної властивості Id (int)

Від цього класу успадковуються майже всі інші базові моделі.

Таким чином BaseEntity надає їм Id.

### 2.2.2.3 Клас Project

```
namespace Core.Models
{
    41 references
    public class Project : BaseEntity
    {
        5 references
        public string Name { get; set; }
        3 references
        public string Description { get; set; }
        8 references
        public List<Task> Tasks { get; set; }
        5 references
        public List<User> Workers { get; set; }
    }
}
```

Рис. 2.8 – Клас Project

Складається з:

- Публічної властивості Name (string)
- Публічної властивості Description (string)
- Публічної властивості Tasks (List<Task>)
- Публічної властивості Workers (List<User>)

Цей клас потрібен для реалізації проектів, а також пов'язаних з ними взаємодій.

#### 2.2.2.4 Клас Task

```
using Core.Enums;

namespace Core.Models
{
    35 references
    public class Task : BaseEntity
    {
        8 references
        public string Name { get; set; }
        3 references
        public string Description { get; set; }
        3 references
        public DateTime EstimatedTime { get; set; }
        10 references
        public Status Status { get; set; }
        1 reference
        public Priority Priority { get; set; }
        4 references
        public List<Attachment> Attachments { get; set; }
    }
}
```

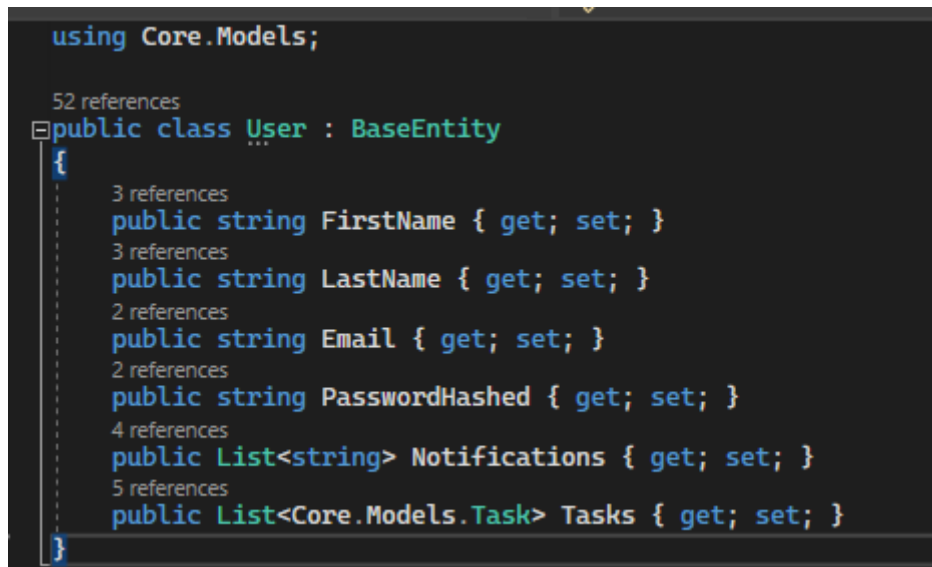
Рис. 2.9 – Клас Task

Складається з:

- Публічної властивості Name (string)
- Публічної властивості Description (string)
- Публічної властивості EstimatedTime (DateTime)
- Публічної властивості Status (Enum Status)
- Публічної властивості Priority (Enum Priority)
- Публічної властивості Attachments  
(List<Attachment>)

Цей клас використовується для реалізації задач, а також для відносин у програмі.

### 2.2.2.5 Клас User



```
using Core.Models;

52 references
public class User : BaseEntity
{
    3 references
    public string FirstName { get; set; }
    3 references
    public string LastName { get; set; }
    2 references
    public string Email { get; set; }
    2 references
    public string PasswordHashed { get; set; }
    4 references
    public List<string> Notifications { get; set; }
    5 references
    public List<Core.Models.Task> Tasks { get; set; }
}
```

Рис. 2.10 – Клас User

Складається з:

- Публічної властивості FirstName (string)
- Публічної властивості LastName (string)
- Публічної властивості Email (string)
- Публічної властивості PasswordHashed (string)
- Публічної властивості Notifications (List<string>)
- Публічної властивості Tasks (List<Task>)

Цей клас використовується для реалізації працівників у програмі.

Він також потрібен для взаємодії з об'єктами інших класів.

#### 2.2.2.6 Клас UserProjectRole

```
using Core.Enums;

namespace Core.Models
{
    13 references
    public class UserProjectRole : BaseEntity
    {
        8 references
        public int ProjectId { get; set; }
        8 references
        public int UserId { get; set; }
        4 references
        public Duty Duty { get; set; }
    }
}
```

Рис. 2.11 – Клас UserProjectRole

Складається з:

- Публічної властивості ProjectId (int)
- Публічної властивості UserId (int)
- Публічної властивості Duty (Enum Duty)

Цей клас потрібен для реалізації відносин між працівником та проектом.

### 2.2.3 Інтерфейс IStorage<T>

```
using Core.Models;
using Helpers.Validators;

namespace DAL.Abstractions.Interfaces
{
    8 references
    public interface IStorage<T> where T : BaseEntity
    {
        2 references
        Task<Result<List<T>>> GetAllAsync(int pageNumber = 1, int pageSize = 10);

        2 references
        Task<Result<T>> GetByIdAsync(int id);

        3 references
        Task<Result<T>> GetByPredicateAsync(Func<T, bool> predicate);

        2 references
        Task<Result<bool>> AddAsync(T obj);

        2 references
        Task<Result<bool>> UpdateAsync(int id, T updatedObj);

        2 references
        Task<Result<bool>> RemoveAsync(int id);
    }
}
```

Рис. 2.12 - Інтерфейс IStorage<T>

Складається з:

- Асинхронного метода GetAllAsync
- Асинхронного метода GetByIdAsync
- Асинхронного метода GetByPredicateAsync
- Асинхронного метода AddAsync
- Асинхронного метода UpdateAsync
- Асинхронного метода RemoveAsync

Цей інтерфейс використовується на рівні BLL для реалізації зберігання інформації. Клас Storage успадковується від цього інтерфейсу. Через процес серіалізації, інформація записується у Json файли. Для кожного класу, який успадковується від BaseEntity, свій Json файл.

## 2.2.4. Інтерфейси сервісів

### 2.2.4.1 Інтерфейс IGenericService<T>

```
using Core.Models;
using Task = System.Threading.Tasks.Task;

namespace BLL.Abstractions.Interfaces
{
    6 references
    public interface IGenericService<T> where T : BaseEntity
    {
        4 references
        Task Add(T obj);

        3 references
        Task Remove(int id);

        8 references
        Task<T> GetById(int id);

        8 references
        Task<List<T>> GetAll();

        2 references
        Task<T> GetByPredicate(Func<T, bool> predicate);

        8 references
        Task Update(int id, T obj);
    }
}
```

Рис. 2.13 - Інтерфейс IGenericService<T>

Складається з:

- Асинхронного метода Add
- Асинхронного метода Remove
- Асинхронного метода GetById
- Асинхронного метода GetAll
- Асинхронного метода GetByPredicate
- Асинхронного метода Update

Цей інтерфейс використовується для зв'язку BLL та функціоналу DAL.

#### 2.2.2.4.2 Інтерфейс IProjectService

```
using Core.Models;
using Task = System.Threading.Tasks.Task;

namespace BLL.Abstractions.Interfaces
{
    public interface IProjectService : IGenericService<Project>
    {
        Task<Project> CreateProject(Project project);
        Task<Project> GetProjectByTask(Core.Models.Task task);
    }
}
```

Рис. 2.14 - Інтерфейс IProjectService

Складається з:

- Асинхронного метода CreateProject
- Асинхронного метода GetProjectByTask

Цей інтерфейс надає доступ до класу реалізації проекту для рівня UI.

#### 2.2.2.4.3 Інтерфейс ITaskService

```
using Task = System.Threading.Tasks.Task;

namespace BLL.Abstractions.Interfaces
{
    public interface ITaskService : IGenericService<Core.Models.Task>
    {
        Task OpenFolder(Core.Models.Task task);
        Task CreateAttachment(string filePath, Core.Models.Task task);
        Task<bool> CheckAvailabilityForUser(User user, Core.Models.Task task);
    }
}
```

Рис. 2.15 - Інтерфейс ITaskService

Складається з:

- Асинхронного метода OpenFolder
- Асинхронного метода CreateAttachment



- Асинхронного метода CheckAvailabilityForUser

Цей інтерфейс надає доступ до класу реалізації задачі для рівня UI.

#### 2.2.2.4.4 Інтерфейс IUserProjectRoleService

```
using Task = System.Threading.Tasks.Task;
using Core.Enums;
using Core.Models;

namespace BLL.Abstractions.Interfaces
{
    public interface IUserProjectRoleService : IGenericService<UserProjectRole>
    {
        Task CreateTableRow(Project project, User user);

        Task<List<UserProjectRole>> GetTableByProjectId(int id);

        Task SetUserRole(Project project, User user, Duty duty);

        Task<UserProjectRole> GetRowByProjectUser(Project project, User user);
    }
}
```

Рис. 2.16 - Інтерфейс IUserProjectRoleService

Складається з:

- Асинхронного метода CreateTableRow
- Асинхронного метода GetTableByProjectId
- Асинхронного метода SetUserRole
- Асинхронного метода GetRowByProjectUser

Цей інтерфейс надає доступ до класу реалізації зв'язку проекту та користувача для рівня UI.

## 3 ІНСТРУКЦІЯ КОРИСТУВАЧА

### 3.1 Встановлення програми

Для запуску програми треба буде зайти у .sln файл

CourseWork.sln та запустити проект UI у Visual Studio. Всі додаткові файли та папки, програма створить у процесі взаємодії із користувачем.

### 3.2 Функціонал програми та інструкція до неї

Після виконання дій, зазначених вище, користувача зустрине консоль з ось таким написом:

```
1. Sign Up
2. Sign In
3. Exit
Enter the operation number: _
```

Рис. 3.1 – Початкове меню

Користувач має обрати одну з трьох опцій та натиснути на потрібну цифру на клавіатурі. Після цього натиснути на Enter. При натисканні на «3», програма завершить свою роботу.

#### 3.2.1 Робота зі сторінкою «Sign Up»

Після обирання опції «1», в користувача послідовно запитують Ім'я, Фамілію, Адресу електронної пошти та пароль. У разі, якщо це перший користувач, користувача також попросять ввести данні для створення першої задачі (Назву, Опис).

```
Please enter the name of the project:
Project-Test-4
Please enter the description:
-----
```

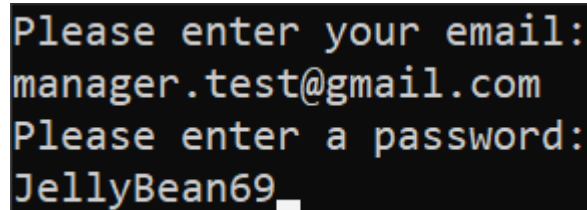
Рис. 3.2 – Меню створення проекту

В такому випадку, користувачу автоматично дадуть тег

«StateManager». У разі, якщо це не перший користувач, програма дасть тег «Unassigned» і не попросить створити проект. Після цього, буде запущено меню користувача.

### 3.2.2 Робота зі сторінкою «Sign In»

Після обирання опції «2», в користувача послідовно запитають email та пароль.



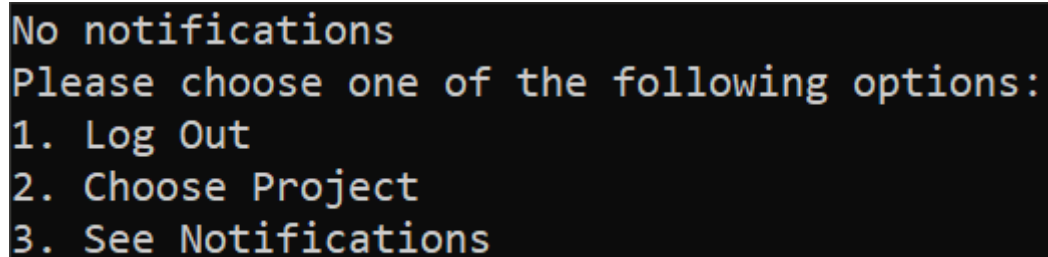
```
Please enter your email:
manager.test@gmail.com
Please enter a password:
JellyBean69_
```

Рис. 3.3 – Log In

Після вводу даних, користувач перейде до меню користувача.

### 3.2.3 Робота з меню користувача

Користувача зустрічає такий екран:



```
No notifications
Please choose one of the following options:
1. Log Out
2. Choose Project
3. See Notifications
```

Рис. 3.4 – Меню користувача

Зверху будуть виведені повідомлення, якщо вони з'являться (там де зараз написано «No notifications»).

Користувач обирає одну з трьох опцій.

- «Log Out» - переносить користувача в початок програми.
- «Choose Project» - активує меню вибору проекту.

```

2. Project-Test-2
-----Description-----
Description cool
-----Assignments-----

-----

3. Project-Test-3
-----Description-----
--
-----Assignments-----

-----

4. Project-Test-4
-----Description-----
-----
-----Assignments-----

-----

Please enter the ID of the project you want to select or 'E' to exit:
- 1: Project-Test
- 2: Project-Test-2
- 3: Project-Test-3
- 4: Project-Test-4

```

Рис. 3.5 – Меню вибору проекту

Після обрання проекту, користувач переходить до меню UI, який обирається відповідно до тегу користувача.

- «See Notifications» - додатково показує повідомлення.

### 3.2.3.1 Меню «StateManager»

Користувач бачить наступне меню:

```
Greetings!
=====
Please choose one of the following options:
1. Add Project
2. Edit current Project
3. Exit
```

Рис. 3.6 - Меню «StateManager»

Тут користувач може обрати одну з трьох опцій:

- «Add Project» - виконується той же самий код, що описано вище у «Sign Up» (див. Рис. 3.2)
- «Edit current Project» - відкриває меню редагування проекту
- «Exit» - повертається до попереднього меню

#### 3.2.3.1.1 Меню редагування проекту

Після переходу через «Edit current Project», користувач побачить наступне меню:

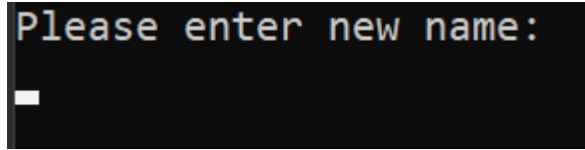
```
Please choose one of the following options:
1. Edit Project Name
2. Edit Project Description
3. Create Assignment
4. Edit Assignments
5. Edit Workers
6. Exit
```

Рис. 3.7 – Меню редагування проекту

Тут користувач обирає один з варіантів:

- Змінити назву проекту («Edit Project Name»)

Тут програма попросить користувача ввести нову назву проекту.

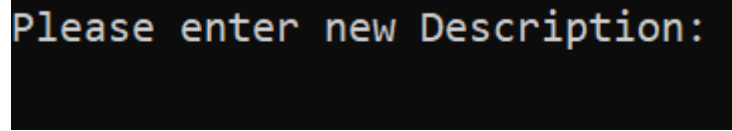


```
Please enter new name:
```

Рис. 3.8 – Зміна назви

- Змінити опис проекту («Edit Project Description»)

Тут програма попросить користувача ввести новий опис проекту.

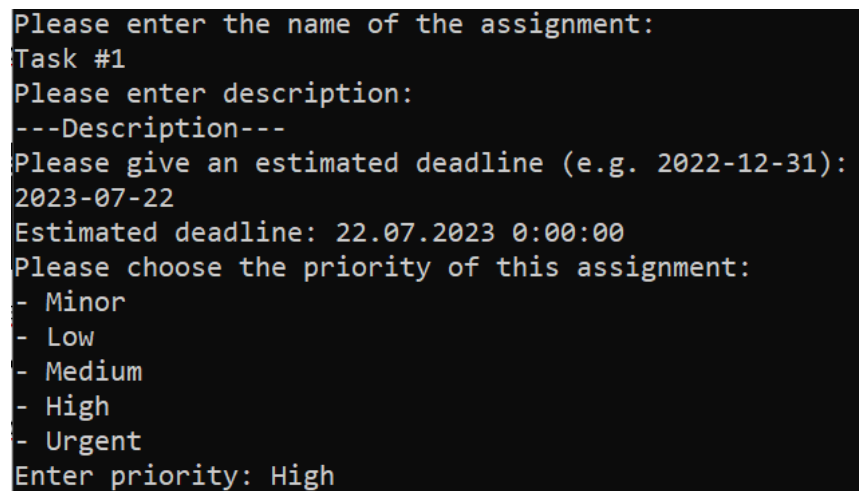


```
Please enter new Description:
```

Рис. 3.9 – Зміна опису

- Створити задачу («Create Assignment»)

Тут програма просить користувача ввести всі необхідні данні для створення задачі:



```
Please enter the name of the assignment:
Task #1
Please enter description:
---Description---
Please give an estimated deadline (e.g. 2022-12-31):
2023-07-22
Estimated deadline: 22.07.2023 0:00:00
Please choose the priority of this assignment:
- Minor
- Low
- Medium
- High
- Urgent
Enter priority: High
```

Рис. 3.10 – Створення задачі

- Налаштувати задачі («Edit Assignments»(1))
- Налаштувати працівників («Edit Workers»(2))

- Вийти («Exit» - повертає до попереднього методу)

#### 3.2.3.1.1.1 Налаштування задачі

Після переходу до меню редагування задачі, користувача попросять обрати задачу:

```
Create DB -- Planned

=====

Description:

Create a structure for a date base. It should be fully prepared, so
we could transition from file system

Estimated Time: 22.07.2023 0:00:00

Attachments:
=====Assignment #2=====

Task #1 -- Planned

=====

Description:

---Description---

Estimated Time: 22.07.2023 0:00:00

Attachments:

Please enter the ID of the assignment you want to select:

- 1: Create DB
- 2: Task #1
```

Рис. 3.11 – Меню обирання задачі

Після того як користувач натисне відповідну цифру, він побачить меню редагування задачі.

```
Please choose one of the following options:
1. Assign worker to this Assignment
2. Close assignment
3. Exit
_
```

Рис. 3.12 – Меню редагування задачі

Тут користувач може обрати одну з трьох опцій.

- Назначити користувача на завдання («Assign worker to this Assignment»). Користувач обирає одного з працівників на проєкті:

```
Please choose a worker:
- 1: Jeff Bezos || StateManager
- 2: Martin Smith || Developer
_
```

Рис. 3.13 – Меню вибору працівника

Відповідно до вибору користувача, статус програми зміниться на відповідний до нового працівника.

- Закрити задачу («Close Assignment»). Програма визначить статус задачі, як «Done»
- Вийти («Exit» - повертає до попереднього методу)

#### 3.2.3.1.1.2 Меню редагування працівників

Користувач переходить до меню вибору взаємодії з працівниками.



```
Please enter one of the following options:
1. Remove Worker from Project
2. Assign Worker for Project
3. Assign duty
4. Exit
_
```

Рис. 3.14 - Меню редагування працівників

Тут користувач може обрати одну з чотирьох опцій.

- Прибрати користувача із проекту («Remove Worker from Project»). Користувач обирає працівника з проекту (див. Рис. 3.13).
- Додати користувача до проекту («Assign Worker for Project»). Користувач обирає працівника (див. Рис. 3.13).
- Дати тег («Assign Duty»). Користувач обирає працівника (див. Рис. 3.13), а потім обирає тег:

```
Please choose the priority of this assignment:
- Unassigned
- Developer
- Tester
- StateManager
Enter duty: Developer_
```

Рис. 3.15 – Меню обирання тегу

- Вийти («Exit» - повертає до попереднього методу)

### 3.2.3.2 Меню «Developer» та «Tester»

Ці меню мають однаковий функціонал. Після обирання проекту, у разі, якщо користувач має один із вище зазначених тегів, він побачить так меню:

```
Please choose one of the following options:
1. Choose Assignment
2. Exit
```

Рис. 3.16 – Меню «Developer» та «Tester»

Користувач обирає одну з двох опцій:

- «Choose Assignment» - переносить користувача до меню обирання задачі (див. Рис. 3.11). Після цього користувач побачить меню взаємодії із задачею.
- «Exit» - повертає до попереднього методу.

#### 3.2.3.2.1 Меню взаємодії із задачею

```
Please choose one of the following options:
1. Hand Assignment
2. Upload File
3. Open File Folder
4. Exit
```

Рис. 3.17 – Меню взаємодії із задачею

Тут користувач має обрати одну з наданих опцій.

- «Hand Assignment» - ідентично до «Assign worker to this Assignment» тегу «SateManager». (див. 3.2.3.1.1.1)
- «Upload File». Програма запитає шлях до файлу.

```
Please enter the file path:
C:\Users\TheJudge\Desktop\files\test2.txt_
```

Рис. 3.18 – Напис шляху до файлу

- «Open File Folder» - відкриває провідник на комп'ютері у директорії файлів задачі.
- «Exit» - повертає до попереднього методу.

## ВИСНОВОК

У ході виконання курсової роботи на тему «Менеджер проектів» була розроблена система на платформі .NET 7.0 з консольним інтерфейсом для користувачів. Система розроблена з урахуванням принципів безпеки та захисту даних.

Система надає можливість реєстрації нових користувачів та авторизації існуючих користувачів. Також система дозволяє створювати, редагувати та закривати задачі та проекти. Користувачі можуть відстежувати прогрес виконання задач та отримувати сповіщення про зміни у статусах задач.

У подальшому систему можна удосконалити збільшенням функціоналу, а також оптимізацією деяких методів, переносом програми на більш зручний UI та використанням бази даних замість серіалізації у Json.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Microsoft .Net Fundamentals documentation – документація Microsoft. URL: <https://learn.microsoft.com/en-us/dotnet/fundamentals/>  
(дата звернення: 29.05.2023)