Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

Лабораторна робота №2

із дисципліни «*Технології розроблення програмного забезпечення*»

Тема: «Основи проектування»

Виконала:
Студент групи ІА-31
Самелюк А.С.

Перевірив:
Мягкий М.Ю.

Київ-2025

Тема: Основи проектування.

Мета: Обрати зручну систему побудови UML-діаграм та навчитися будувати діаграми варіантів використання для системи що проєктується, розробляти сценарії варіантів використання та будувати діаграми класів предметної області.

Посилання на репозиторій з проєктом:
https://github.com/Sameliuk/OnlineDiary_trpz/tree/main

Хід роботи

1. Ознайомитись з короткими теоретичними відомостями.
2. Проаналізувати тему та спроєктувати діаграму варіантів використання відповідно до обраної теми лабораторного циклу.

   Онлайн-щоденник — це веб-застосунок, який дозволяє користувачу створювати, редагувати та організовувати особисті записи.

   Основні функції:

   - створення, редагування, видалення нотаток;
   - додавання тегів до відповідних записів;
   - сортування записів за тегами чи датами;
   - пошук за ключовим словом;
   - email-нагадування про важливі записи.

   Актори:

   - Користувач
   - Система сповіщень

   Варіанти використання:

   1. Управління обліковим записом

   - Реєстрація користувача – створення нового облікового запису.
   - Авторизація користувача – вхід до системи за допомогою логіна та пароля.
   - Вихід із системи – завершення роботи користувача.

   2. Робота з нотатками

   - Створення нотатки – користувач створює новий запис.

- Редагування нотатки – внесення змін у вже існуючу нотатку.
- Видалення нотатки – повне видалення запису з бази даних.
- Перегляд нотаток – відображення списку всіх нотаток із можливістю фільтрації.

3. Організація записів

- Додавання тегів – створення тегів для нотаток.
- Прив'язка тегів до нотаток – організація записів за категоріями.
- Сортування за тегами та датою – швидка навігація між нотатками.

4. Пошук інформації

- Пошук за ключовим словом заголовку – знаходження нотаток у заголовку чи тексті.

5. Нагадування та сповіщення

- Створення нагадування – встановлення дати й часу сповіщення для нотатки.
- Отримання нагадування на email – система автоматично надсилає повідомлення через NotificationService.
- Скасування нагадування – користувач може видалити або деактивувати нагадування.

Зв'язки:

- Користувач → Реєстрація
- Користувач → Авторизація
- Користувач → Вихід із системи
- Користувач → Створення нотатки
- Користувач → Редагування нотатки
- Користувач → Видалення нотатки
- Користувач → Перегляд нотаток
- Користувач → Додавання тегів
- Користувач → Прив'язка тегів до нотаток
- Користувач → Сортування за тегами та датою
- Користувач → Пошук за ключовим словом

- Користувач → Пошук за тегом
- Користувач → Створення нагадування
- Користувач → Скасування нагадування
- Система → Отримання нагадування на email ← Email-сервіс.

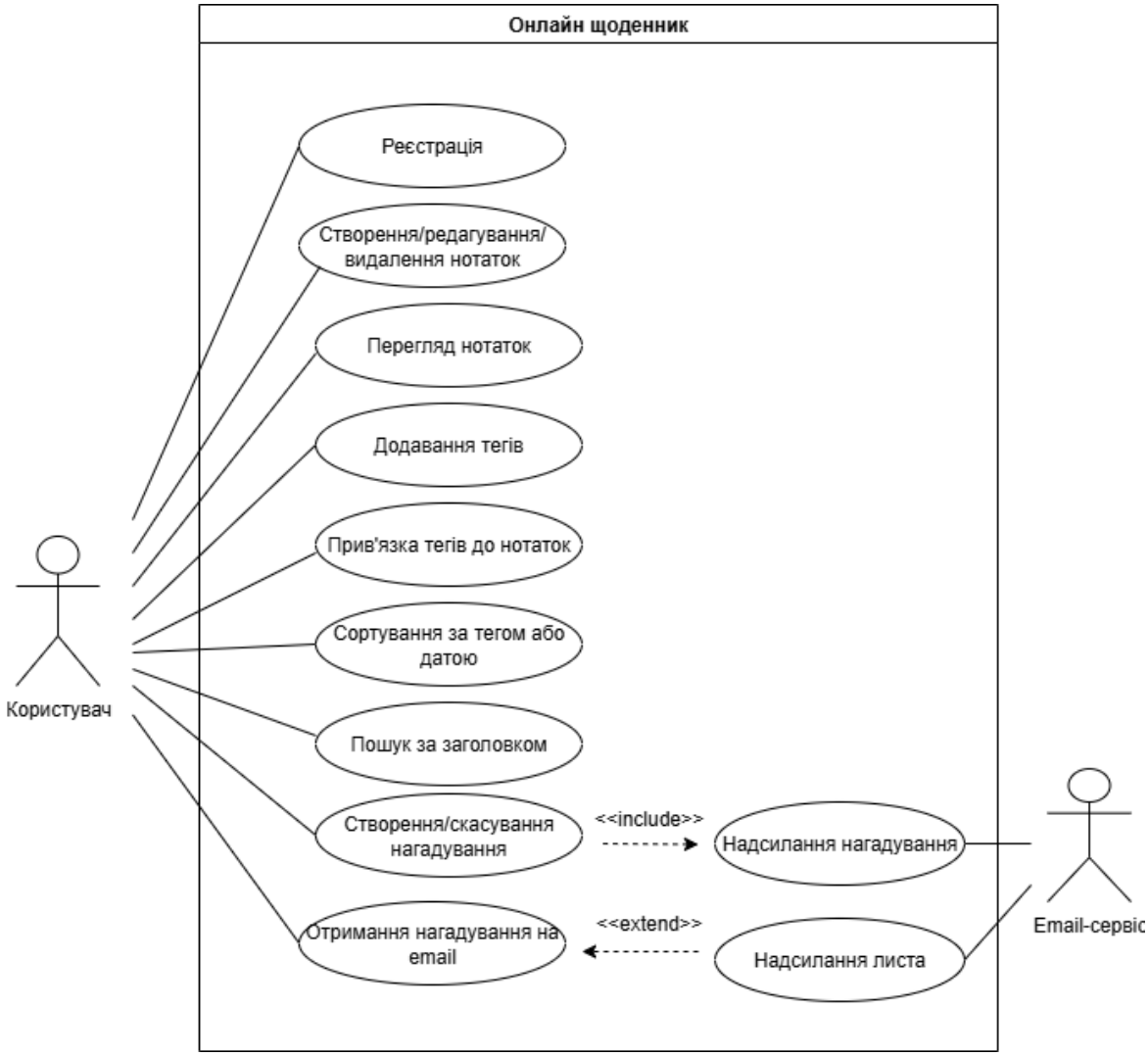
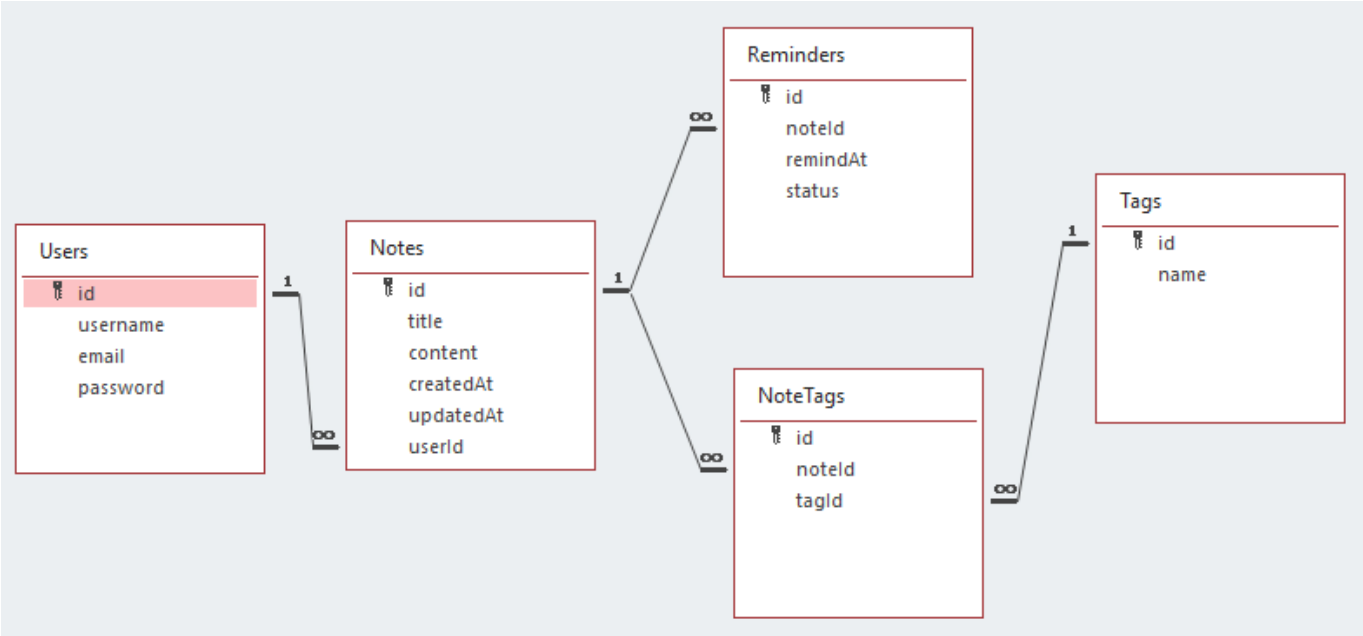


Рис.1 - Діаграму варіантів використання

3. Спроєктувати діаграму класів предметної області.



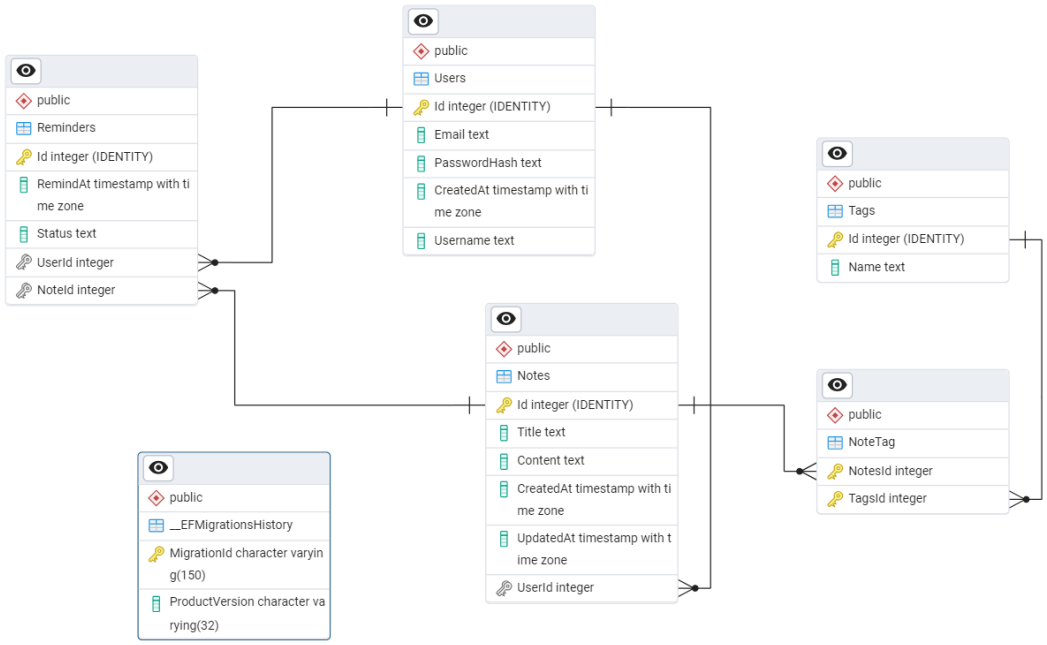


Рис.2 – Структура бази даних(MS Access та PostgreSQL)

Users

Имя поля	Тип данных
id	Счетчик
username	Короткий текст
email	Короткий текст
password	Короткий текст

Рис.3 – Опис таблиці Користувачі

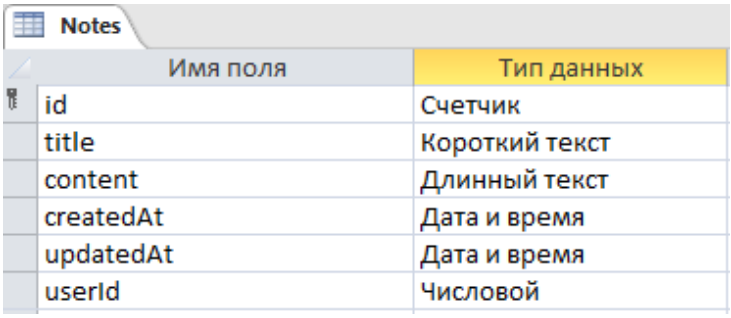


Рис.4 – Опис таблиці Нотатки

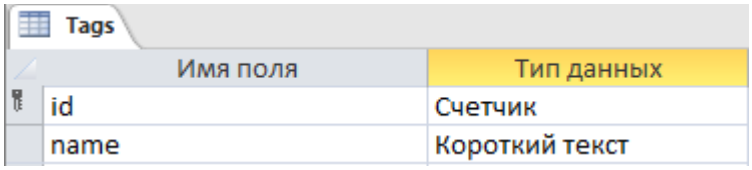


Рис.5 – Опис таблиці Теги

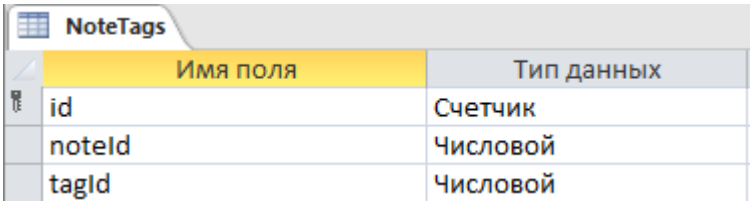


Рис.6– Опис таблиці зв’язку Теги нотаток

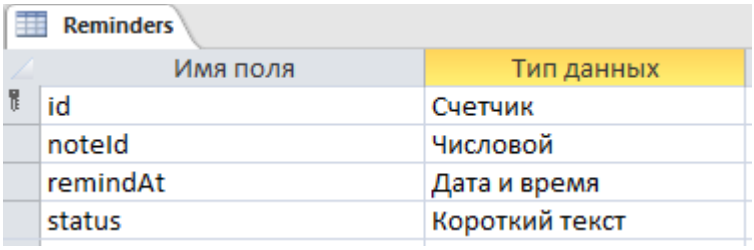


Рис.7 – Опис таблиці зв’язку Нагадування

4. Вибрати 3 варіанти використання та написати за ними сценарії використання.

Сценарій 1 – Створення нотатки

Передумови: Користувач авторизований у системі.

Постумови: У разі успішного виконання нова нотатка збережена у базі даних.

Взаємодіючі сторони: Користувач, Система.

Короткий опис: Варіант використання дозволяє користувачу створювати нові нотатки в особистому щоденнику.

Основний потік подій:

1. Користувач відкриває форму створення нотатки.
2. Система відображає форму для введення даних.
3. Користувач вводить заголовок і текст нотатки.
4. Користувач додає теги (за потреби).
5. Користувач натискає кнопку «Зберегти».
6. Система перевіряє коректність введених даних.
7. Система зберігає нотатку у базі даних.

Винятки:

- Виняток №1: Обов’язкові поля (наприклад, заголовок) залишені порожніми. Система повідомляє користувача про помилку та пропонує виправити дані.

Примітки: Відсутні.

Сценарій 2 – Пошук нотатки за ключовим словом

Передумови: У базі даних існують збережені нотатки.

Постумови: Користувач отримує список нотаток, що відповідають критерію пошуку.

Взаємодіючі сторони: Користувач, Система.

Короткий опис: Варіант використання дозволяє швидко знайти потрібну нотатку за введеним ключовим словом.

Основний потік подій:

1. Користувач вводить ключове слово у пошукове поле.
2. Користувач запускає пошук.
3. Система виконує пошук збігів у заголовках та тексті нотаток.
4. Система формує список результатів.
5. Система відображає список знайдених нотаток користувачу.

Винятки:

- Виняток №1: За заданим ключовим словом збігів не знайдено. Система повідомляє користувача, що результати відсутні.

Примітки: Можливе подальше уточнення пошуку за тегами чи датою.

Сценарій 3 – Нагадування про подію

Передумови: Користувач має створену нотатку.

Постумови: У заданий час користувач отримує email-нагадування.

Взаємодіючі сторони: Користувач, Система, Email-сервіс.

Короткий опис: Варіант використання дозволяє користувачу встановити нагадування для нотатки та отримати повідомлення на email у потрібний час.

Основний потік подій:

1. Користувач відкриває нотатку та встановлює дату і час нагадування.
2. Система зберігає дані про нагадування у базі даних.
3. Коли настає вказаний час, система активує процес надсилання.
4. Система звертається до Email-сервісу для відправки повідомлення.
5. Email-сервіс надсилає повідомлення на пошту користувача.

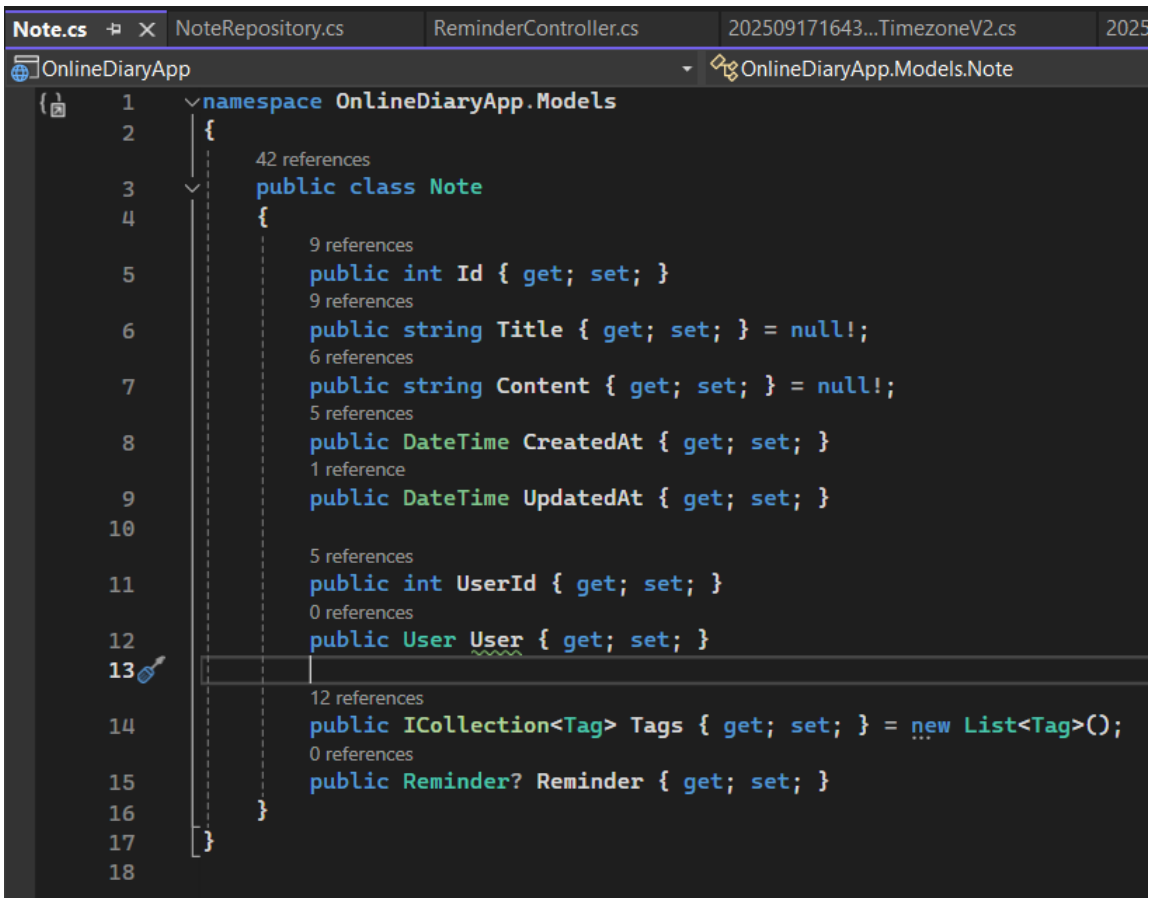
Винятки:

- Виняток №1: Email-сервіс недоступний. Система реєструє помилку, може повторити спробу надсилання пізніше.

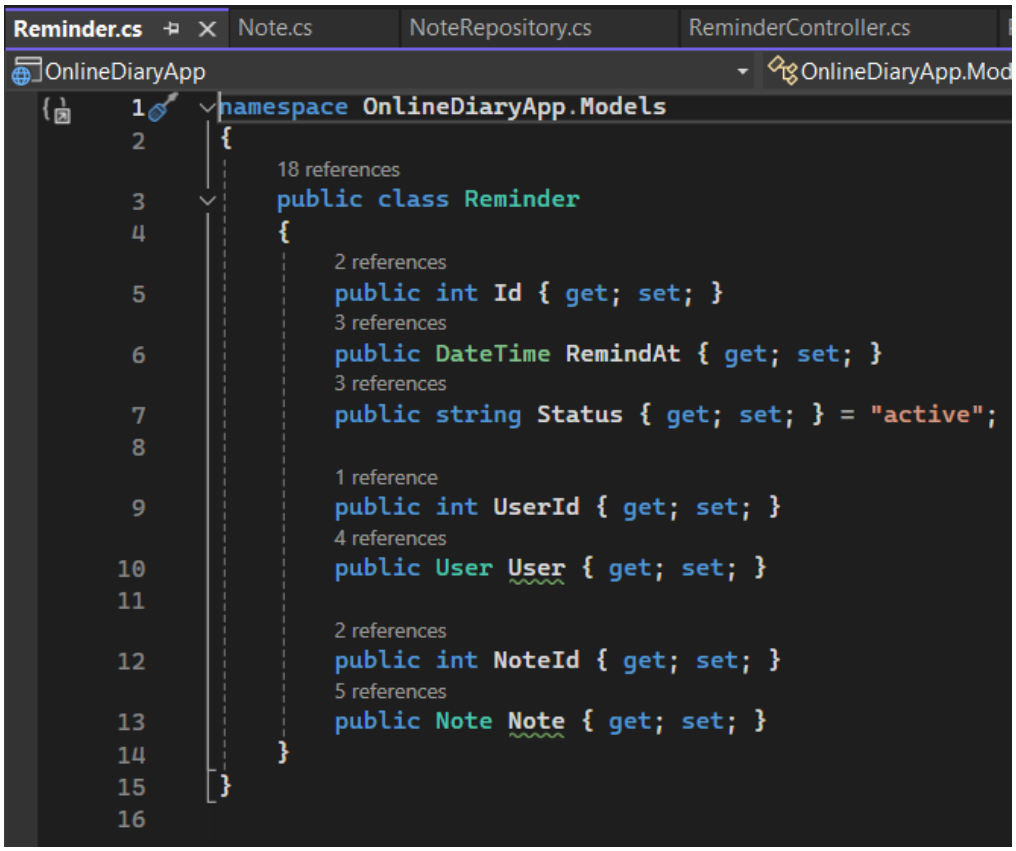
Примітки: Повідомлення може містити посилання для швидкого переходу до відповідної нотатки.

5. На основі спроєктованої діаграми класів предметної області розробити основні класи та структуру бази даних системи. Класи даних повинні реалізувати шаблон Repository для взаємодії з базою даних.

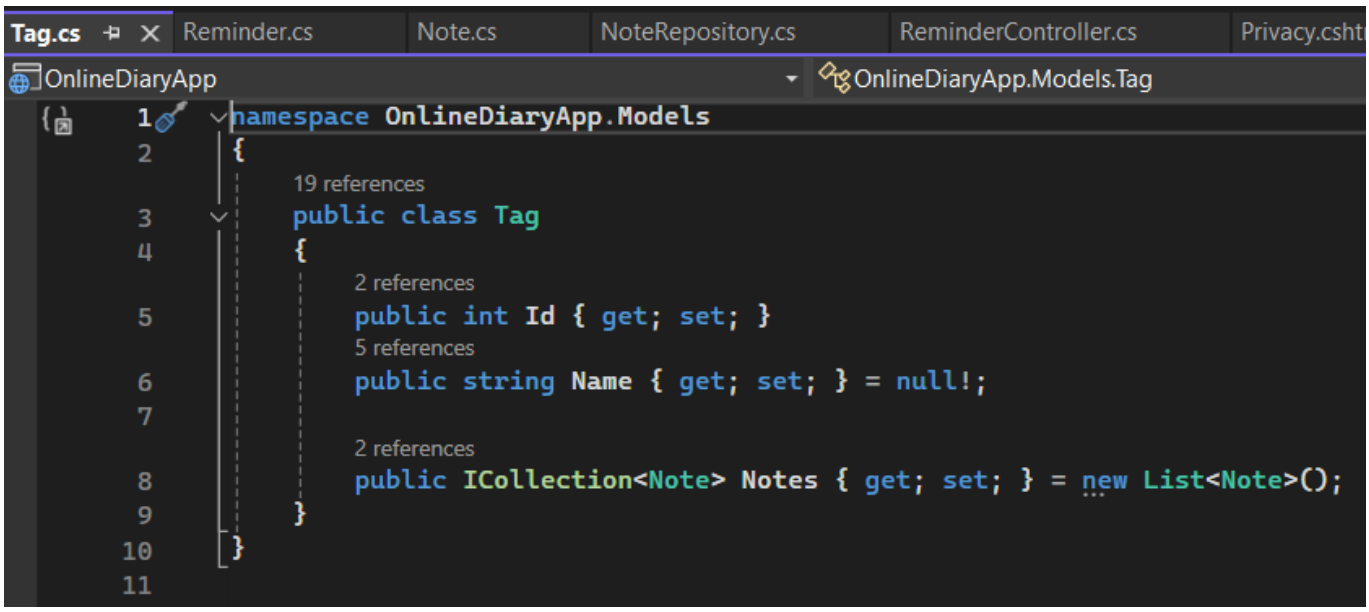
   Код реалізованих класів:



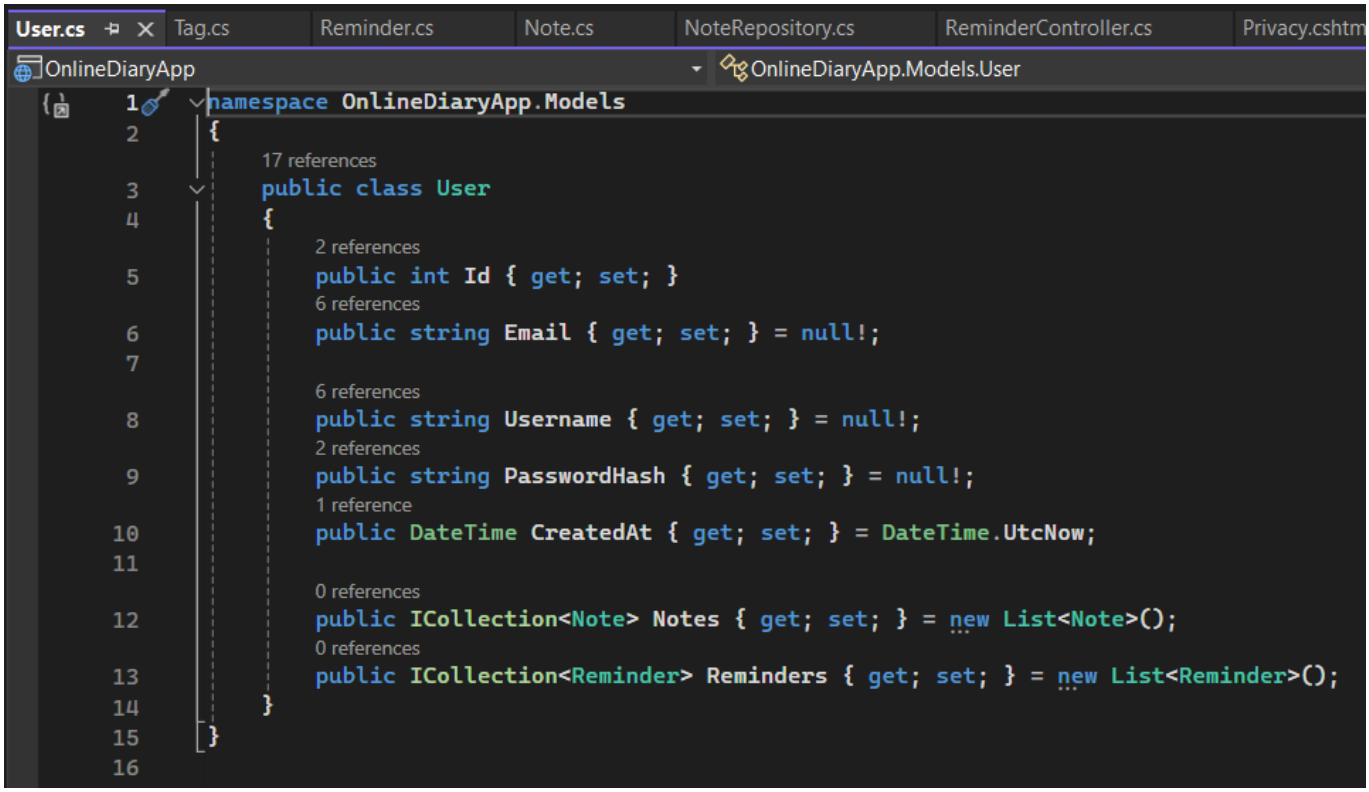
Код 1 – Note.cs (Models)



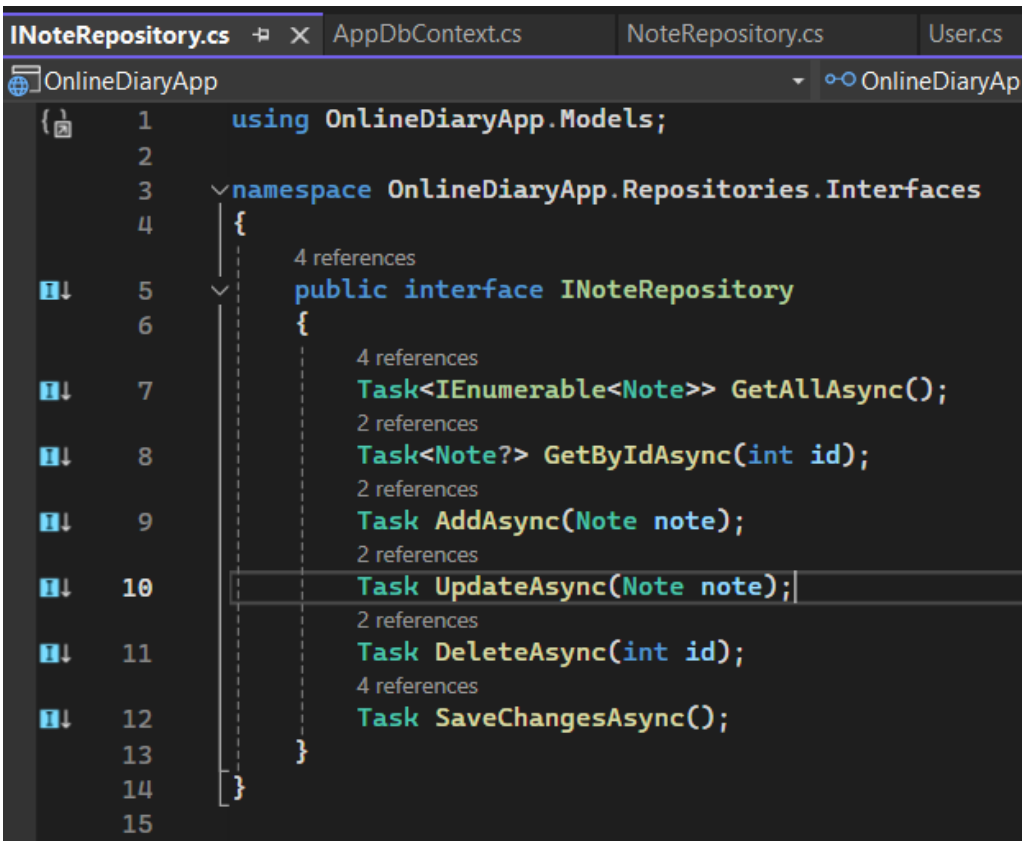
Код 2 –Reminder.cs (Models)



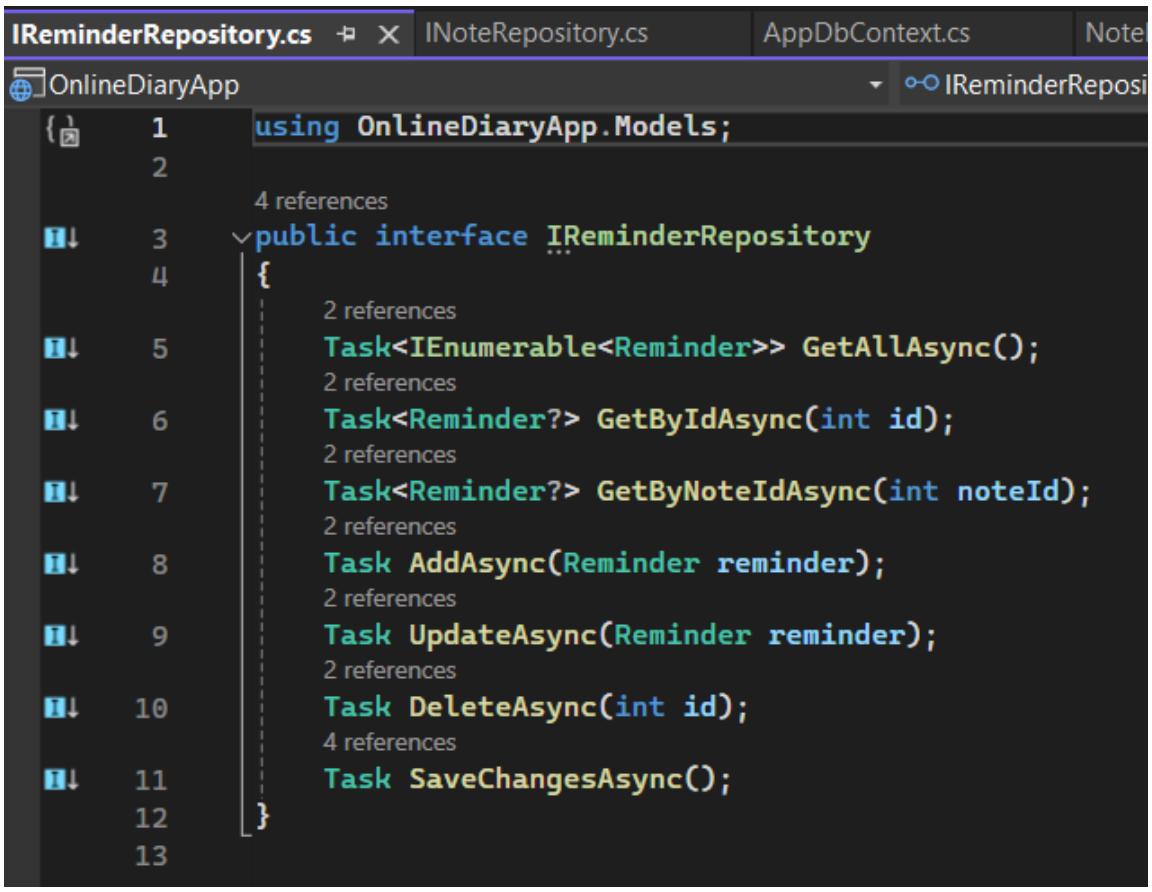
Код 3 –Tag.cs (Models)



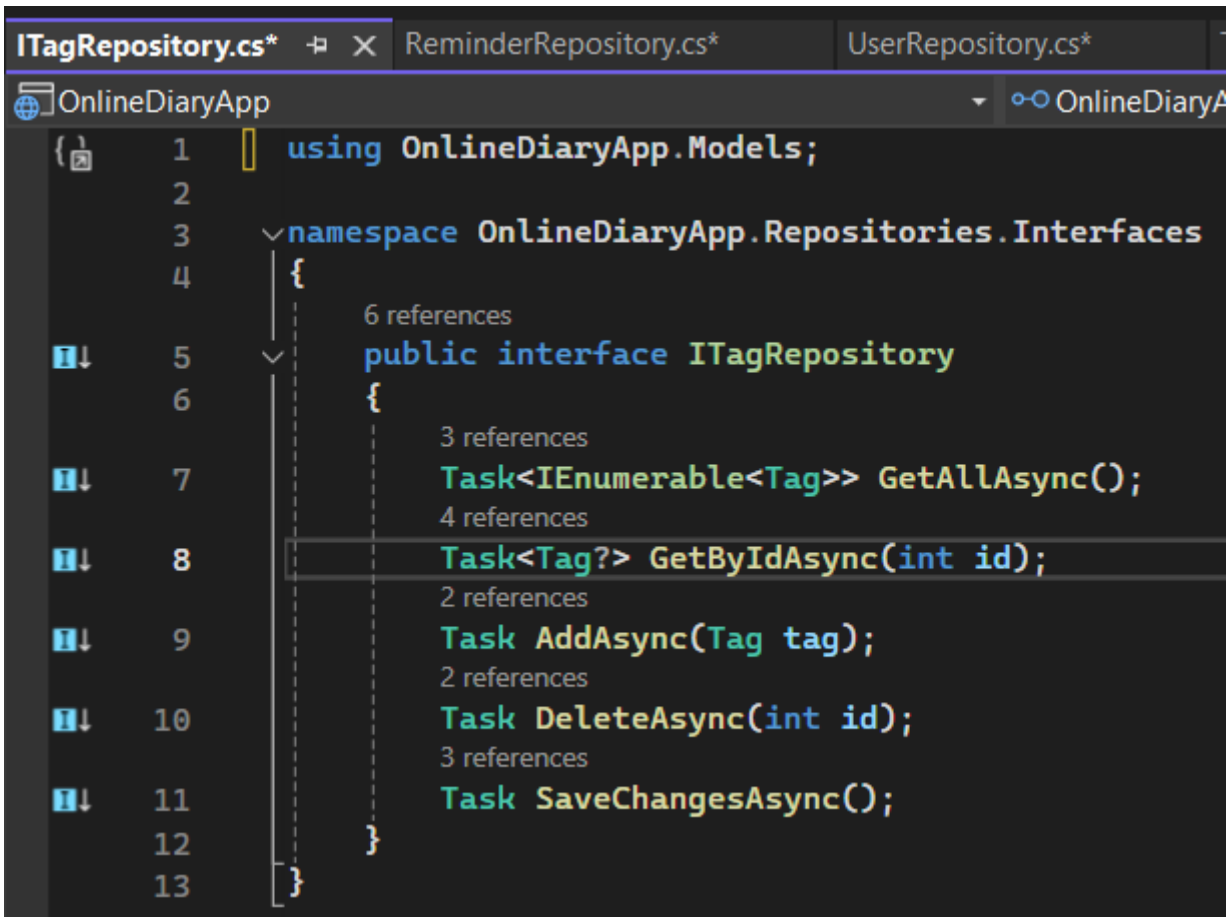
Код 4 –User.cs (Models)



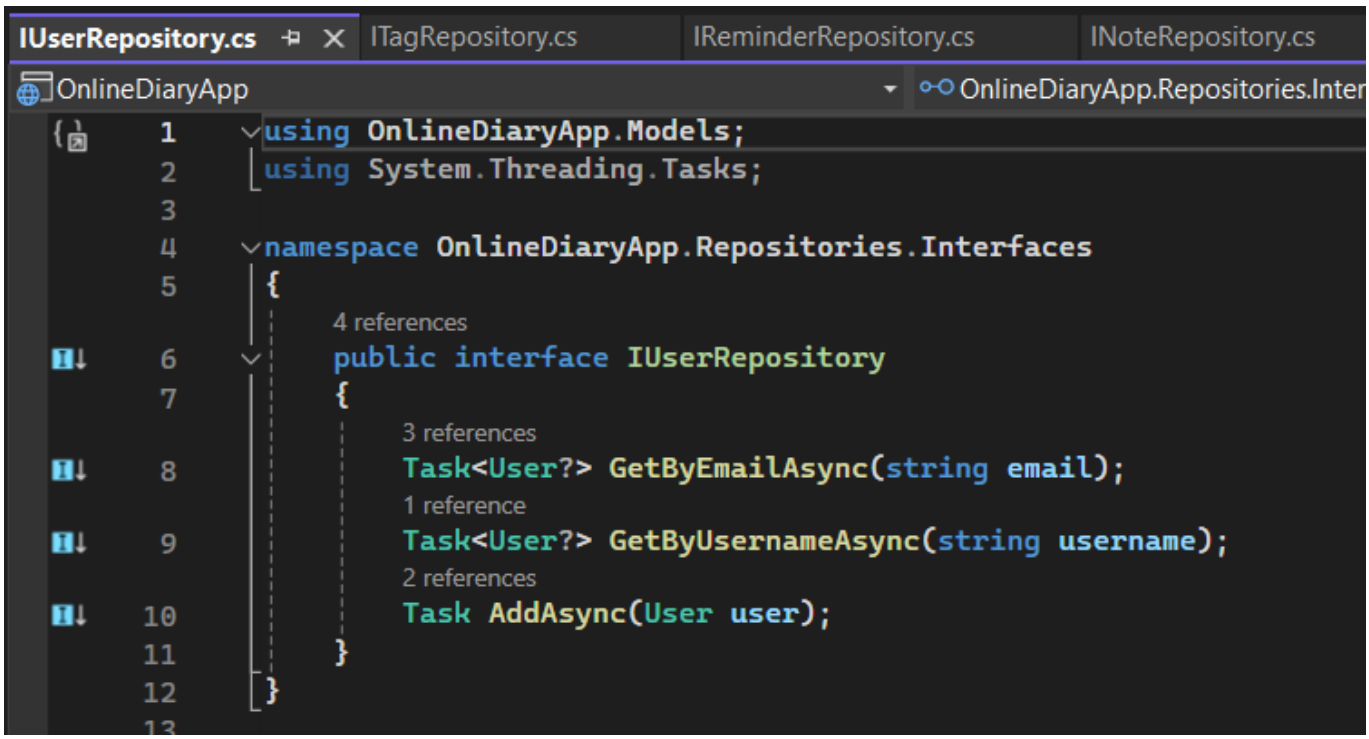
Код 5 – INoteRepository.cs (Repositories.Interfaces)



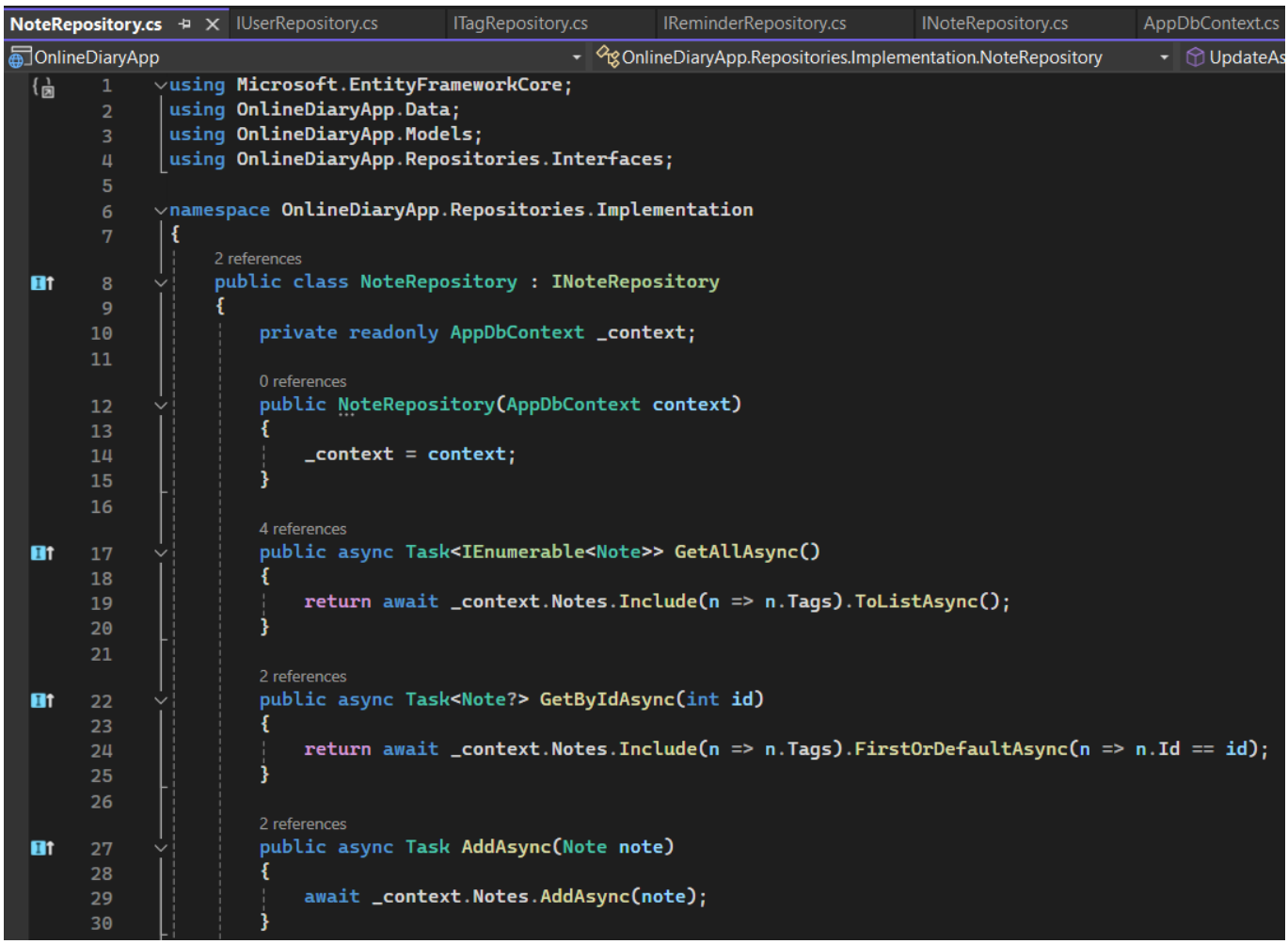
Код 6 – IReminderRepository.cs (Repositories.Interfaces)

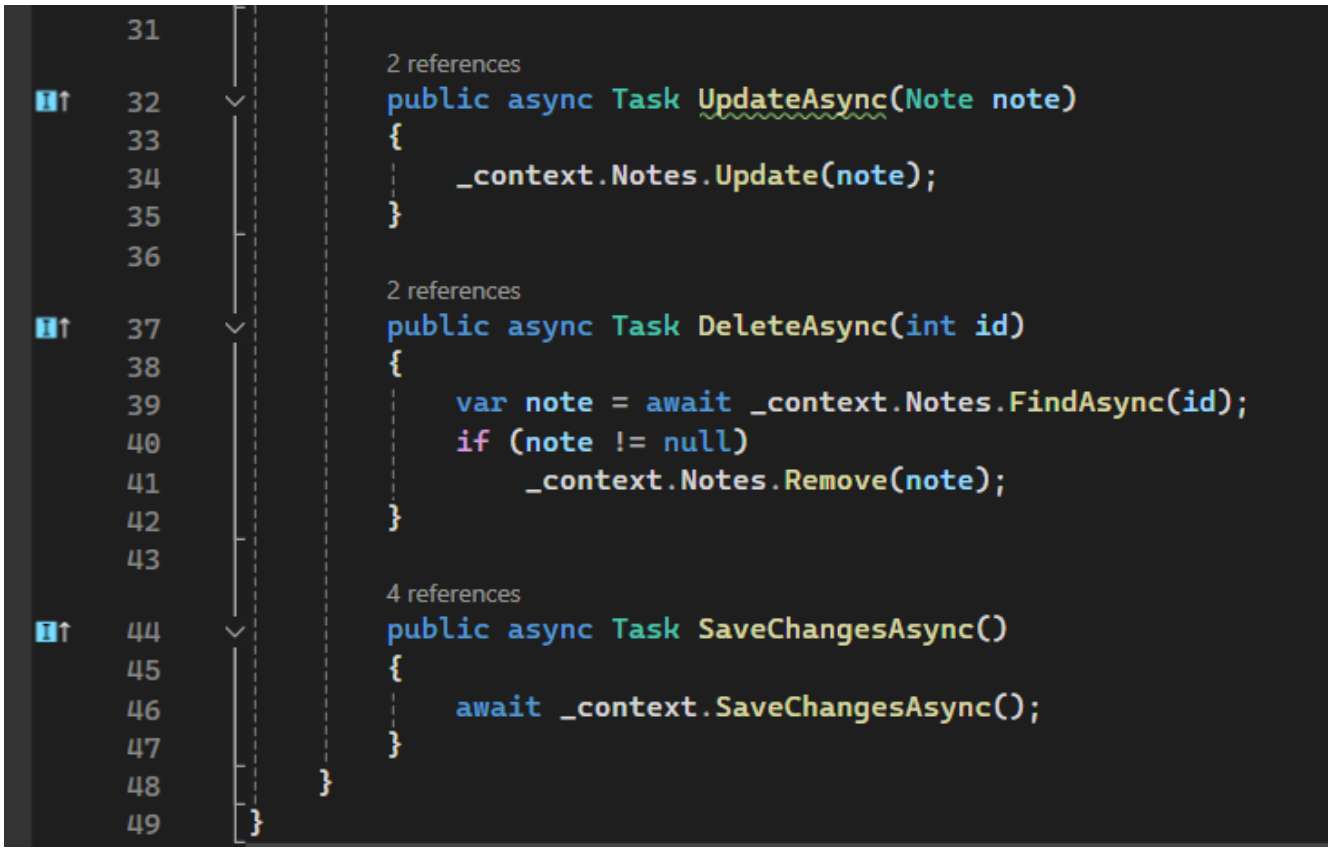


Код 7 – ITagRepository.cs (Repositories.Interfaces)



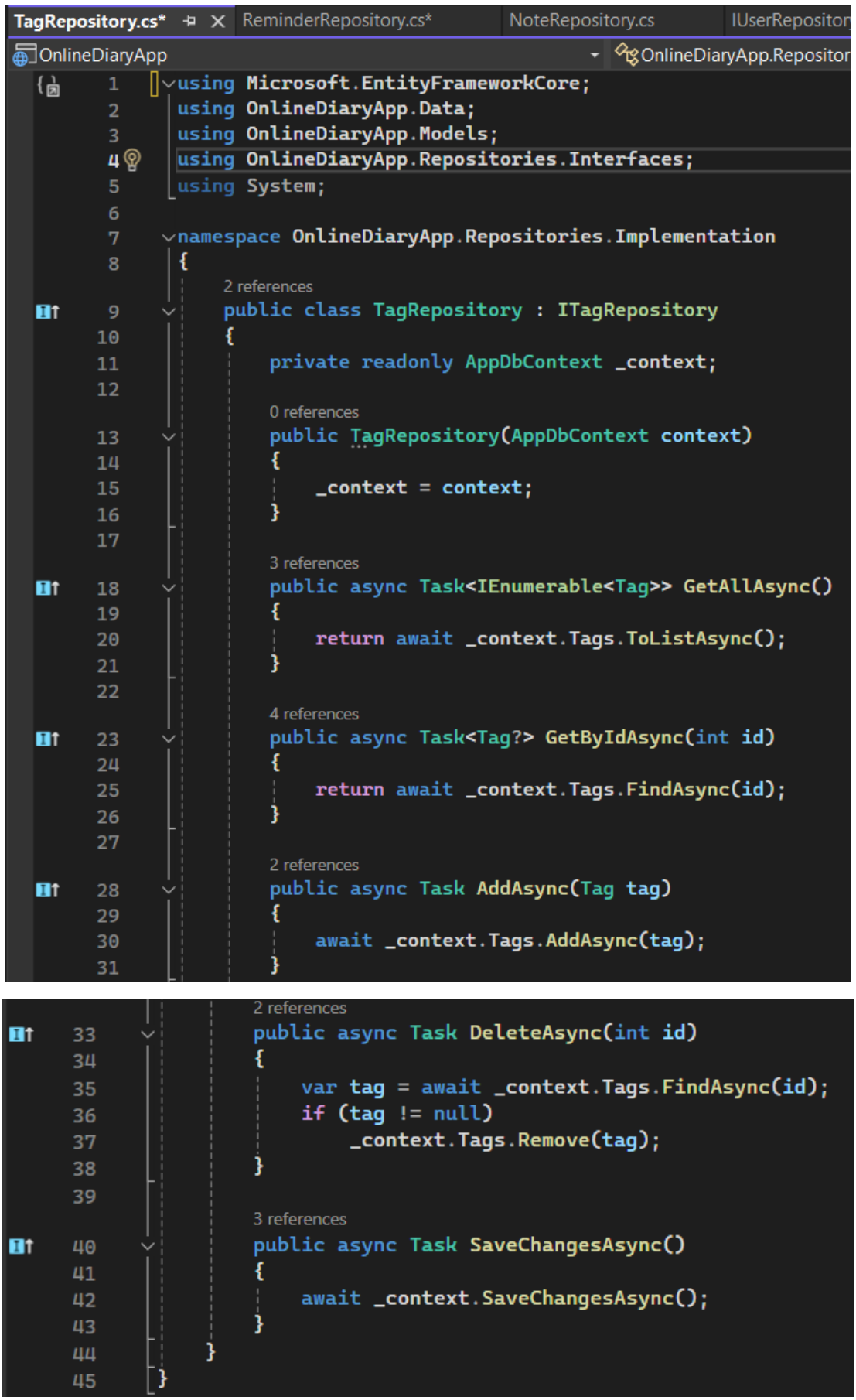
Код 8 – IUserRepository.cs (Repositories.Interfaces)





Код 9 – NoteRepository.cs (Repositories.Implementations)

Код 10 – ReminderRepository.cs (Repositories.Implementations)



Код 11 – TagRepository.cs (Repositories.Implementations)

```
UserRepository.cs*  TagRepository.cs*  ReminderRepository.cs*  NoteRepository.cs  IUserRepository.cs
OnlineDiaryApp                          OnlineDiaryApp.Repositories.Implementations.UserRepository
```

```
1   using Microsoft.EntityFrameworkCore;
2   using OnlineDiaryApp.Data;
3   using OnlineDiaryApp.Models;
4   using OnlineDiaryApp.Repositories.Interfaces;
5
6   namespace OnlineDiaryApp.Repositories.Implementations
7   {
        2 references
8       public class UserRepository : IUserRepository
9       {
10          private readonly AppDbContext _context;
11
            0 references
12          public UserRepository(AppDbContext context)
13          {
14              _context = context;
15          }
16
            3 references
17          public async Task<User?> GetByEmailAsync(string email) =>
18              await _context.Users.FirstOrDefaultAsync(u => u.Email == email);
19
            1 reference
20          public async Task<User?> GetByUsernameAsync(string username) =>
21              await _context.Users.FirstOrDefaultAsync(u => u.Username == username);
22
            2 references
23          public async Task AddAsync(User user)
24          {
25              await _context.Users.AddAsync(user);
26              await _context.SaveChangesAsync();
27          }
28      }
29  }
```

Код 12 – UserRepository.cs (Repositories.Implementations)

6. Нарисувати діаграму класів для реалізованої частини системи.

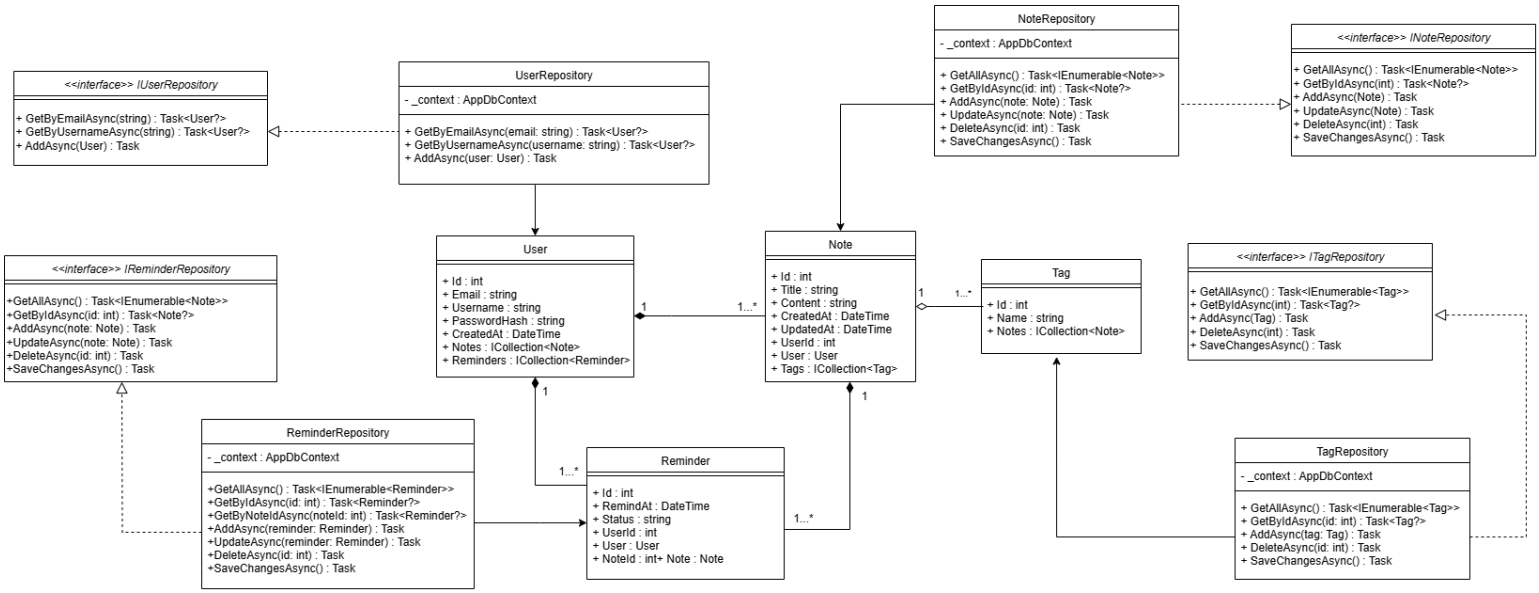


Рис. 8 – Діаграма класів

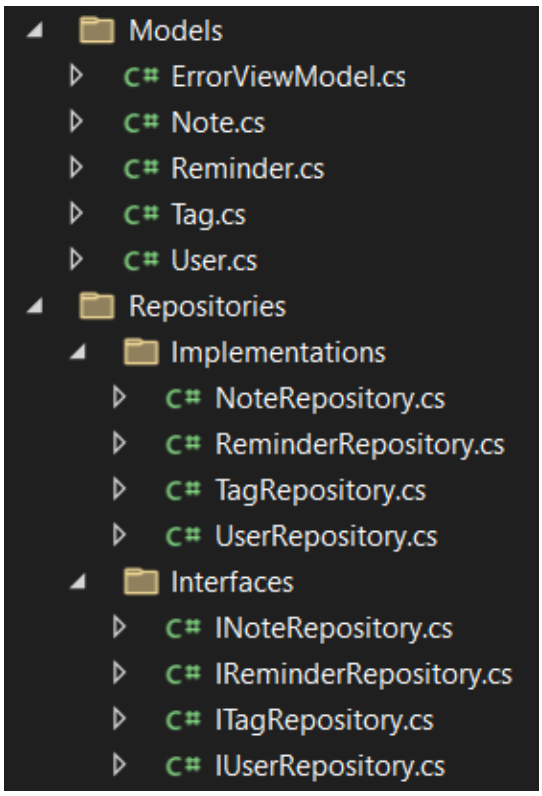


Рис. 9 – Реалізовані класи

7. Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму варіантів використання відповідно, діаграму класів системи, вихідні коди класів системи, а також зображення структури бази даних.

Висновки: Виконання лабораторної роботи дозволило ознайомитися з основними методами об’єктно-орієнтованого моделювання системи за допомогою UML. Було створено діаграму варіантів використання, яка наочно відображає взаємодію користувачів із системою та ключові функції, що вона надає. За допомогою діаграми класів було показано основні класи системи, їх атрибути та методи, а також зв’язки між класами, що дозволяє зрозуміти структурну організацію програми та логіку взаємодії об’єктів.

Відповіді на контрольні питання:

1. Що таке UML?

UML (Unified Modeling Language) — це уніфікована мова моделювання, яка використовується для графічного представлення, документування та

проектування програмних систем. Вона допомагає описати структуру, поведінку та взаємодію компонентів системи.

2. Що таке діаграма класів UML?

Діаграма класів UML — це статична діаграма, яка показує класи системи, їх атрибути, методи та зв'язки між класами. Вона використовується для моделювання структури об'єктно-орієнтованих програм.

3. Які діаграми UML називають канонічними?

Канонічні (основні) діаграми UML — це ті, що найчастіше використовуються для опису системи:

- Діаграма класів
- Діаграма варіантів використання (Use Case)
- Діаграма послідовності (Sequence)
- Діаграма станів (State)
- Діаграма активностей (Activity)

4. Що таке діаграма варіантів використання?

Діаграма варіантів використання (Use Case Diagram) показує взаємодію користувачів (акторів) з системою через варіанти використання. Вона відображає, хто і як використовує систему, без деталізації внутрішньої реалізації.

5. Що таке варіант використання?

Варіант використання (Use Case) — це послідовність дій, які система виконує для досягнення певної мети користувача. Наприклад, "Реєстрація користувача" або "Надіслати повідомлення".

6. Які відношення можуть бути відображені на діаграмі використання?

На діаграмі варіантів використання можуть бути:

- Association (асоціація) — зв'язок між актором і варіантом використання.

- Include (включення) — один варіант використання завжди виконує інший.
- Extend (розширення) — додатковий варіант, який може виконуватися у певних умовах.
- Generalization (успадкування) — актор або варіант використання наслідує інший.

7. Що таке сценарій?

Сценарій — це конкретна реалізація варіанту використання, тобто послідовність кроків, які відбуваються під час виконання дії користувачем та системою.

8. Що таке діаграма класів?

Діаграма класів — це графічне представлення класів, їх атрибутів і методів, а також зв'язків між класами. Вона описує структуру системи.

9. Які зв'язки між класами ви знаєте?

Основні типи зв'язків:

- Асоціація (Association) — загальний зв'язок між класами.
- Агрегація (Aggregation) — «має»; клас складається з інших, але частини можуть існувати окремо.
- Композиція (Composition) — сильніша агрегація; частини не можуть існувати без цілого.
- Успадкування (Generalization) — один клас наслідує властивості іншого.
- Залежність (Dependency) — один клас використовує інший тимчасово.

10. Чим відрізняється композиція від агрегації?

- Агрегація: частини можуть існувати без цілого.
- Композиція: частини не можуть існувати без цілого; знищення цілого знищує частини.

11. Чим відрізняється агрегація від композиції на діаграмах класів?

- Агрегація позначається порожнім ромбом на стороні цілого.

- Композиція позначається заповненим ромбом на стороні цілого.

12. Що являють собою нормальні форми баз даних?

Нормальні форми — це правила організації таблиць БД для уникнення надлишковості та аномалій при оновленні даних. Основні:

- 1НФ — всі атрибути атомарні.
- 2НФ — всі неключові атрибути залежать від усього первинного ключа.
- 3НФ — немає транзитивних залежностей між неключовими атрибутами.

13. Що таке фізична і логічна модель БД?

- Логічна модель — структура даних з таблицями, полями та зв’язками, незалежно від СУБД.
- Фізична модель — конкретна реалізація БД у СУБД, включає типи даних, індекси, обмеження, фізичне зберігання.

14. Який взаємозв'язок між таблицями БД та програмними класами?

Кожна таблиця БД часто відповідає класу у програмі, а рядки таблиці — об’єктам класу. Поля таблиці відображаються як атрибути класу, а зв’язки між таблицями як зв’язки між класами.