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 №3

із дисципліни «*Технології розроблення програмного забезпечення*»

Тема: «Основи проектування розгортання»

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

Перевірив:
Мягкий М.Ю.

Київ-2025

Тема: Основи проектування розгортання.

Тема проєкту: Онлайн-щоденник (strategy, adapter, observer, facade, composite, client-server). Веб-застосунок дозволяє користувачу створювати, редагувати та організовувати особисті записи з можливістю пошуку, сортування та додавання тегів, а також отримання email-нагадувань.

Мета: Навчитися проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

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

Хід роботи

1. Ознайомитись з короткими теоретичними відомостями.
2. Проаналізувати діаграми створені в попередній лабораторній роботі, а також тему системи та спроєктувати діаграму розгортання використання відповідно до обраної теми лабораторного циклу.

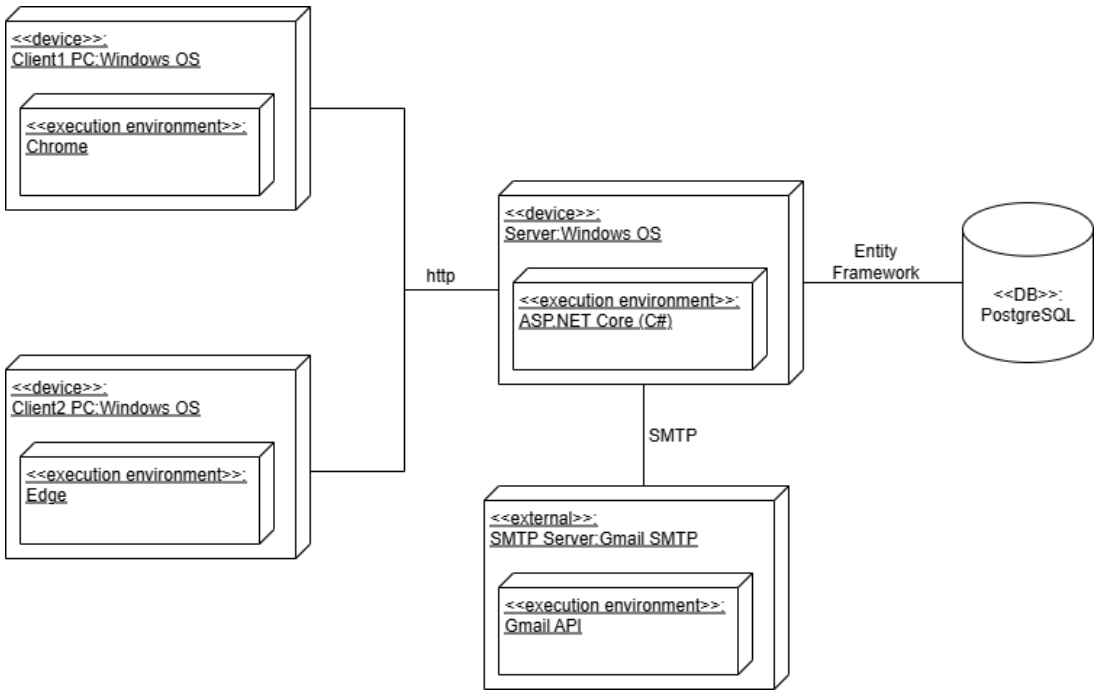


Рис.1 – Діаграма розгортання

На діаграмі розгортання зображено інфраструктуру веб-застосунку «OnlineDiary», де клієнти працюють на операційній системі Windows та взаємодіють із системою через браузери Chrome та Edge. Веб-застосунок розгорнутий на сервері під керуванням Windows OS і виконується у середовищі ASP.NET Core (C#). Для доступу до бази даних PostgreSQL застосунок використовує ORM-технологію Entity Framework. Для надсилання email-нагадувань система взаємодіє з зовнішнім SMTP-сервером (Gmail SMTP), використовуючи протокол SMTP.

3. Розробити діаграму компонентів для проєктованої системи.

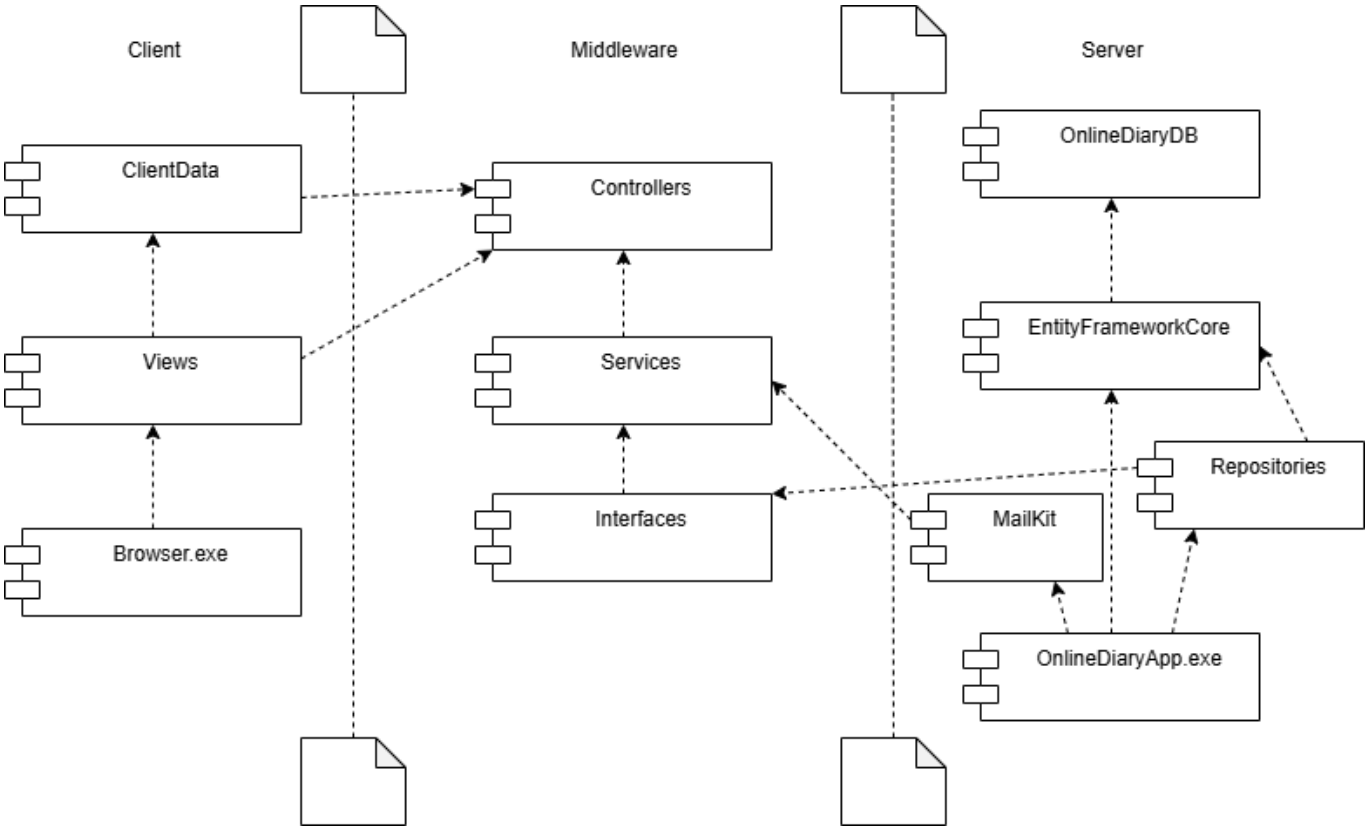


Рис.2 – Діаграма компонентів

На діаграмі зображено основні компоненти програмної системи. У лівій частині подані клієнтські файли запуску (.exe) та бібліотеки (.dll), які відповідають за інтерфейс користувача і взаємодію з веб-додатком. У правій частині показано серверні компоненти (.exe та .dll), які реалізують бізнес-логіку, роботу з базою даних через Entity Framework та інтеграцію з зовнішнім email-сервісом. У центральній частині розташовані модулі middleware, які містять загальні інтерфейси,

контракти API та сервіси, що забезпечують узгоджену взаємодію між клієнтом і сервером. Таким чином, клієнтська і серверна частини системи обмінюються інформацією через спільний набір інтерфейсів і даних, визначених у middleware.

4. Розробити як мінімум дві діаграми послідовностей для сценаріїв прописаних в попередній лабораторній роботі.

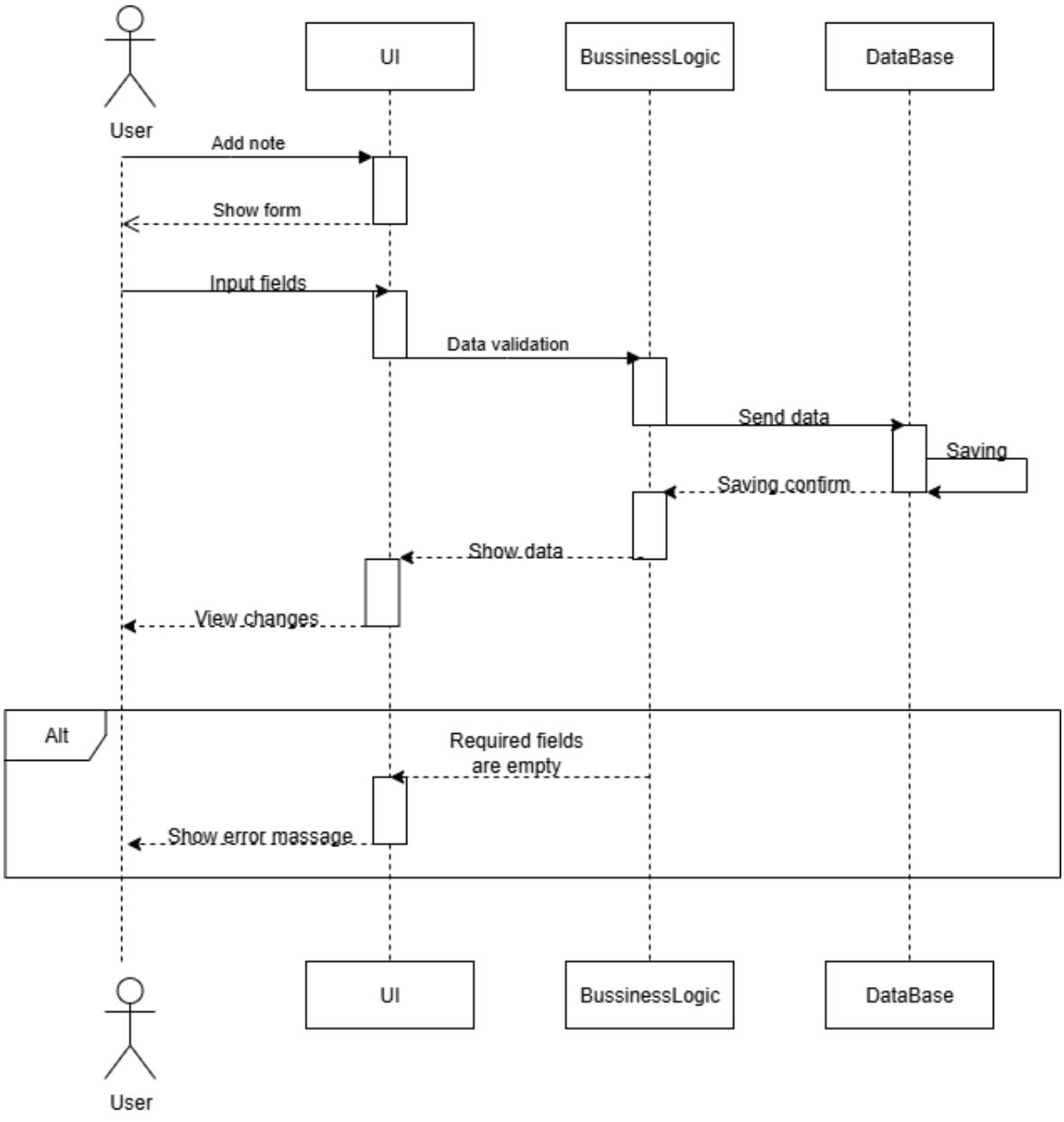


Рис.3 – Діаграма послідовності (сценарій 1)

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.

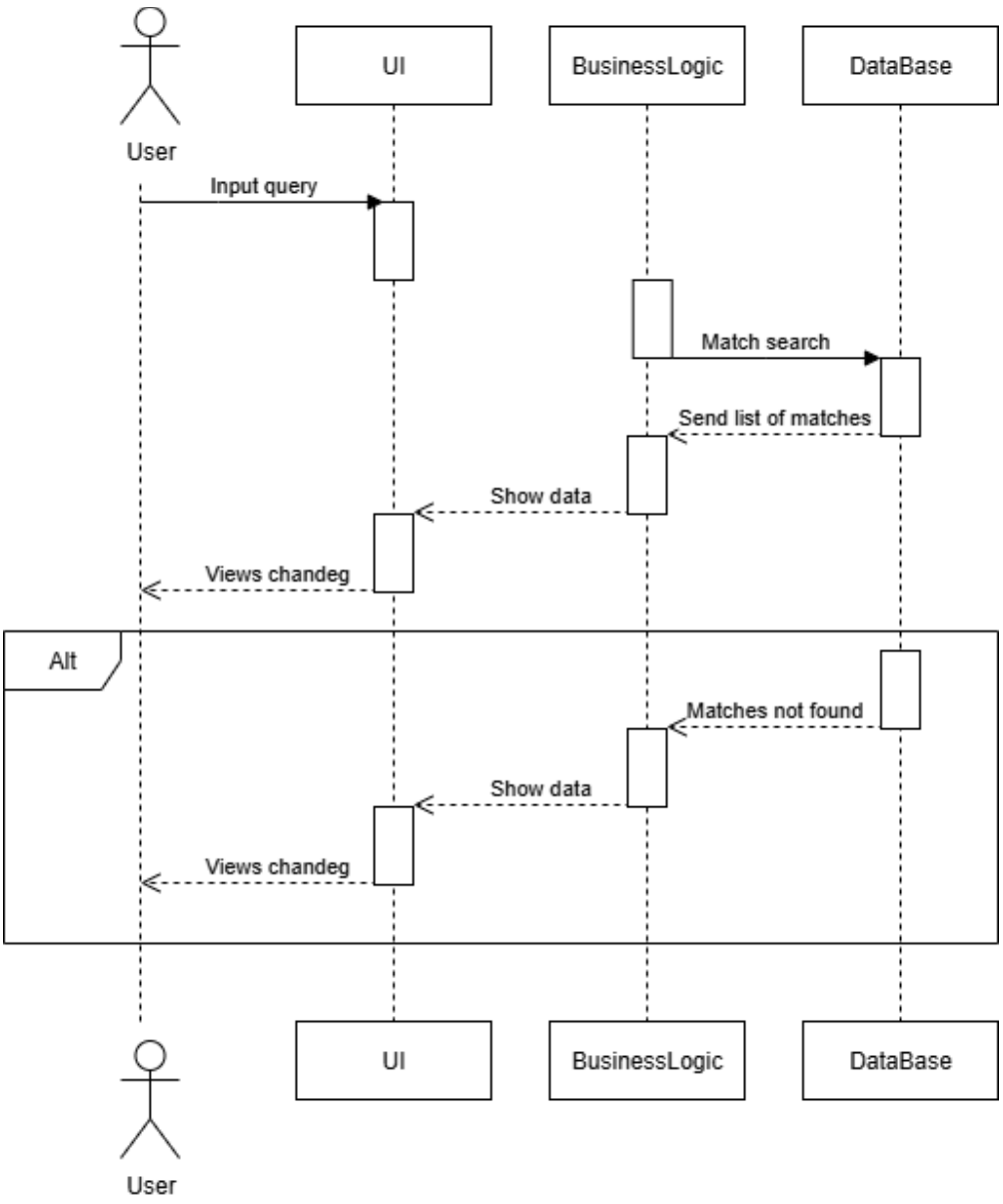


Рис.4 – Діаграма послідовності (сценарій 2)

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.

5. На основі спроєктованих діаграм розгортання та компонентів доопрацювати програмну частину системи. Реалізація системи, додатково до попередньої реалізації, повинна містити як мінімум дві візуальні форми. В системі вже повинен бути повністю реалізована архітектура (повний цикл роботи з даними від вводу на формі до збереження їх в БД і подальшій виборці з БД та відображенням на UI). Реєстрація нового користувача в системі:

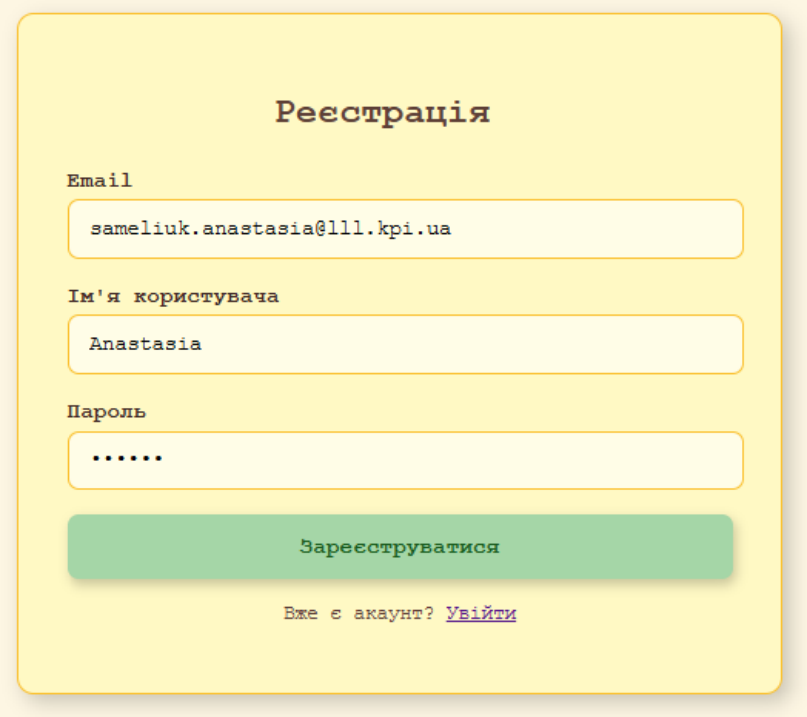


Рис.5 – Форма реєстрації

	Id [PK] integer	Email text	PasswordHash text	CreatedAt timestamp with time zone	Username text
1	1	sameliuk.anastasia@gmail.c...	/qxT8INiQB...	2025-09-17 08:43:48.531162+00	Анастасія
2	2	sameliuk.anastasia@lll.kpi.ua	/qxT8INiQB...	2025-09-22 20:01:48.443598+00	Anastasia

Рис.6 – Додано користувач в таблиці Users в базі даних

Створення нотатки:

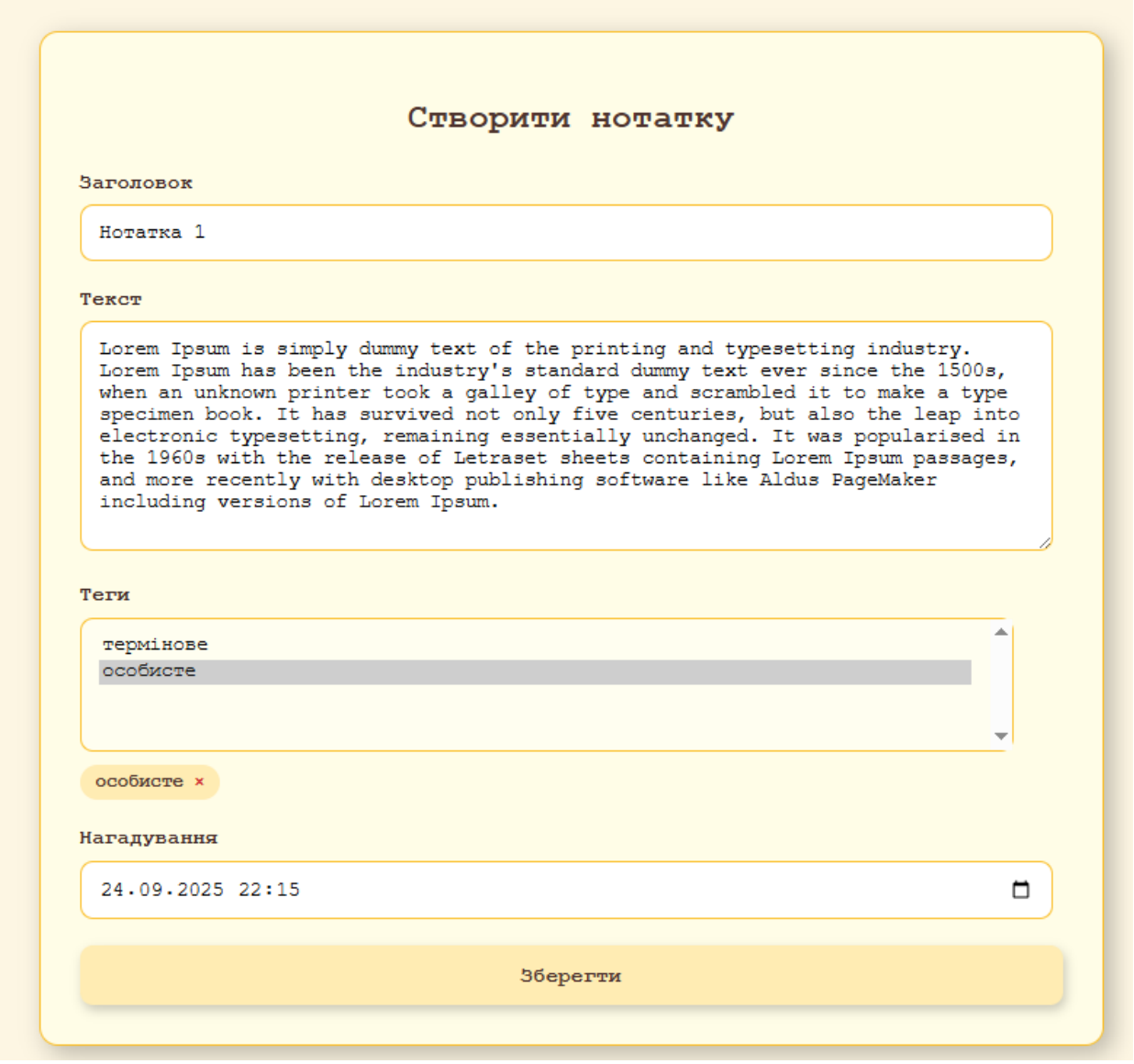


Рис.7 – Форма створення нотатки

Рис.8 – Сторінка користувача з усіма його нотатками

	Id [PK] integer	Title text	Content text	CreatedAt timestamp with time zone	UpdatedAt timestamp with time zone	UserId integer
6	7	полрдотщ	тпмлрищтшоьдльт гшщзшо т гтшз	2025-09-17 16:10:58.542067+00	2025-09-17 16:10:58.542068+00	1
7	8	мпрольдж	вртчпте	2025-09-17 16:25:21.809022+00	2025-09-17 16:25:21.809023+00	1
8	9	вапрорл	іквеанпроло	2025-09-17 16:44:05.909585+00	2025-09-17 16:44:06.150533+00	1
9	10	авапро	арпол	2025-09-17 17:05:00.753936+00	2025-09-17 17:05:00.94786+00	1
10	11	укенгш	уцкенгшщз	2025-09-17 17:05:25.215023+00	2025-09-17 17:05:25.217784+00	1
11	12	rtyui	rtryuio	2025-09-17 17:24:51.650209+00	2025-09-17 17:24:51.801885+00	1
12	13	hjhxc	efreyr6ui6olgf	2025-09-17 17:25:22.906961+00	2025-09-17 17:25:22.910011+00	1
13	14	енгшщзщ	цкеункгешнщгзшхщ	2025-09-18 20:01:48.282695+00	2025-09-18 20:01:48.36814+00	1
14	15	ваипт	івавпарп	2025-09-18 20:16:36.982996+00	2025-09-18 20:16:37.10574+00	1
15	16	уцкуенгш	кенгшщз	2025-09-18 20:33:47.471041+00	2025-09-18 20:33:47.665094+00	1
16	17	ц4у56г	ц4ну5к	2025-09-18 20:41:17.56741+00	2025-09-18 20:41:17.651086+00	1
17	18	Нотатка 1	Lorem Ipsum is simply dummy text ...	2025-09-22 20:15:40.501675+00	2025-09-22 20:15:40.674654+00	1

Рис.9 – Додано нотатку в таблиці Notes в базі даних

	Id [PK] integer	RemindAt timestamp with time zone	Status text	UserId integer	NoteId integer
1	4	2025-09-26 23:15:00+00	active	1	7
2	5	2025-09-25 19:25:00+00	active	1	8
3	9	2025-09-17 23:25:00+00	sent	1	12
4	10	2025-09-17 16:26:00+00	sent	1	13
5	11	2025-09-19 23:01:00+00	sent	1	14
6	14	2025-09-18 23:42:00+00	sent	1	17
7	15	2025-09-24 22:15:00+00	active	1	18

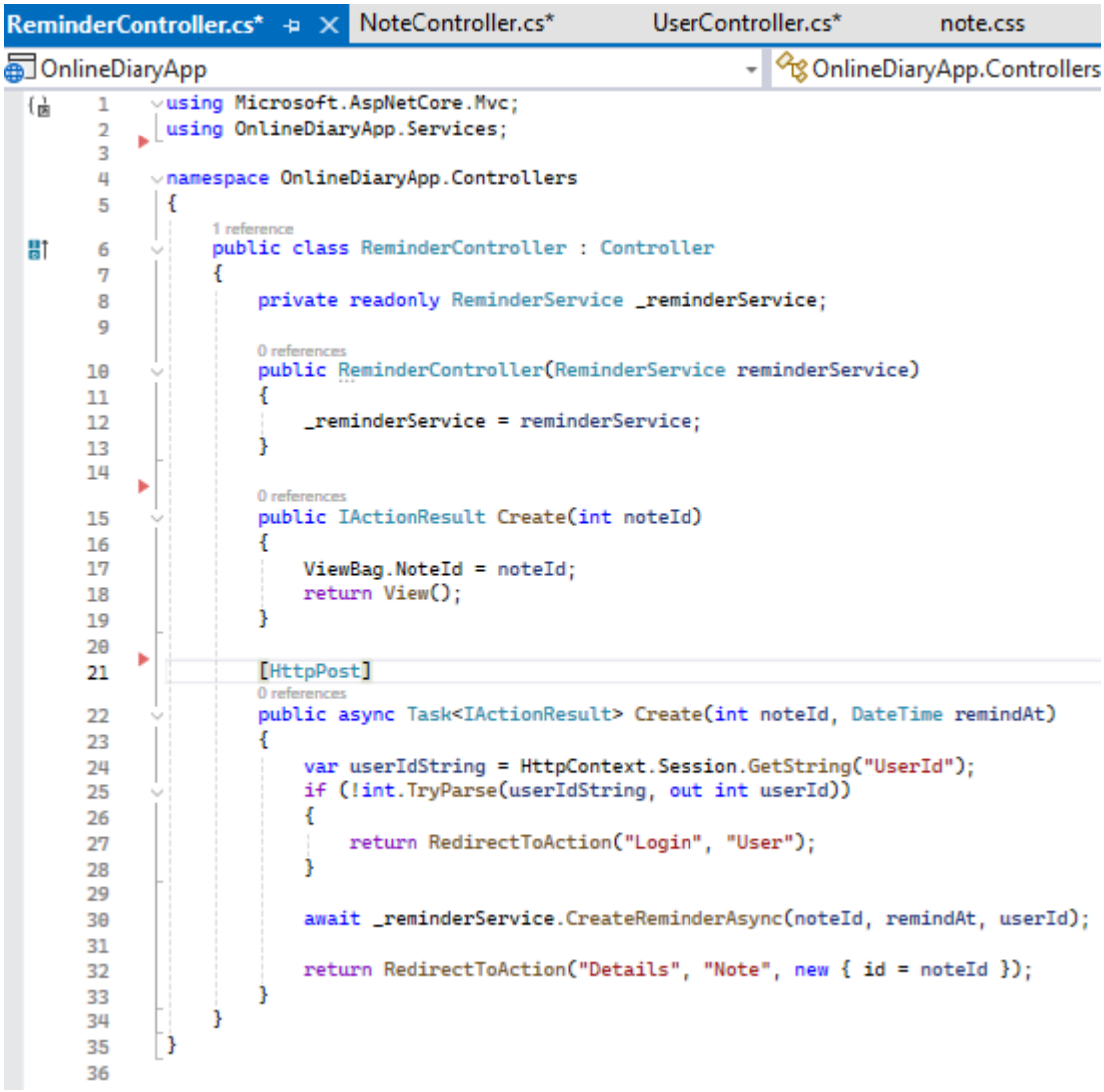
Рис.10 – Додано нагадування до нотатки в таблиці Reminders в базі даних

Код 1 – UserController.cs (controllers)

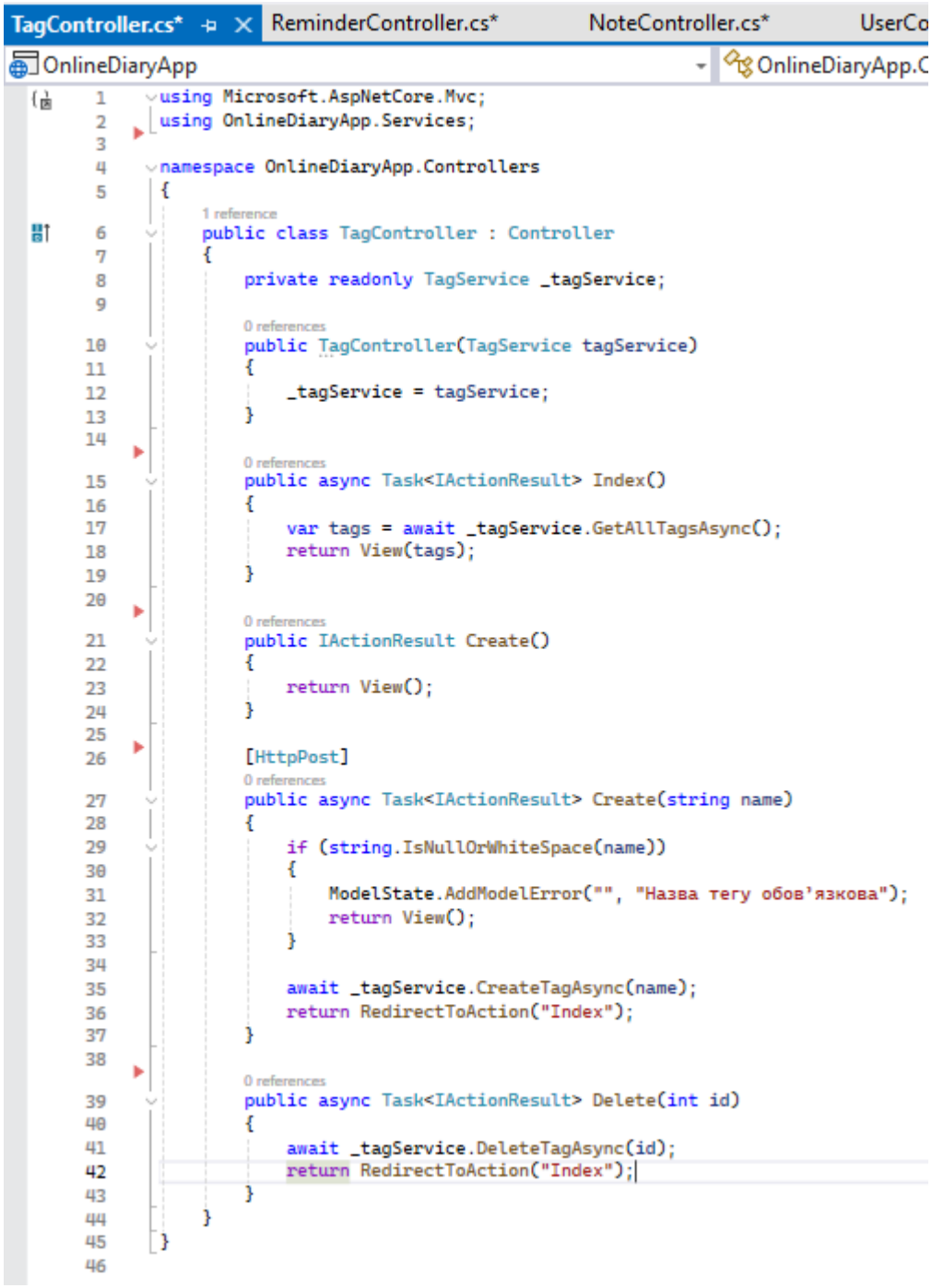
Код 2.1 – NoteController.cs (controllers)

```
78
79          [HttpPost]
            0 references
80          public async Task<IActionResult> Edit(int id, string title, string content, List<int>? tagIds, DateTime? reminderDate)
81          {
82              var note = await _noteService.GetNoteByIdAsync(id);
83              if (note == null)
84                  return NotFound();
85
86              note.Title = title;
87              note.Content = content;
88
89              await _noteService.UpdateNoteAsync(note, tagIds ?? new List<int>());
90
91              var existingReminder = await _reminderService.GetReminderByNoteIdAsync(note.Id);
92
93              if (reminderDate.HasValue)
94              {
95                  if (existingReminder != null)
96                  {
97                      await _reminderService.UpdateReminderAsync(existingReminder, reminderDate.Value);
98                  }
99                  else
100                 {
101                     await _reminderService.CreateReminderAsync(note.Id, reminderDate.Value, note.UserId);
102                 }
103             }
104
105             return RedirectToAction("Index");
106         }
107
            0 references
108         public async Task<IActionResult> Delete(int id)
109         {
110             await _noteService.DeleteNoteAsync(id);
111             return RedirectToAction("Index");
112         }
113
            0 references
114         public async Task<IActionResult> Search(string keyword)
115         {
116             var userIdString = HttpContext.Session.GetString("UserId");
117             if (!int.TryParse(userIdString, out int userId))
118                 return View(new List<Note>());
119
120             var notes = await _noteService.SearchByTitleAsync(keyword);
121             notes = notes.Where(n => n.UserId == userId);
122             return View("Index", notes);
123         }
124     }
125 }
126
```

Код 2.2 – NoteController.cs (controllers)



Код 3 – ReminderController.cs (controllers)



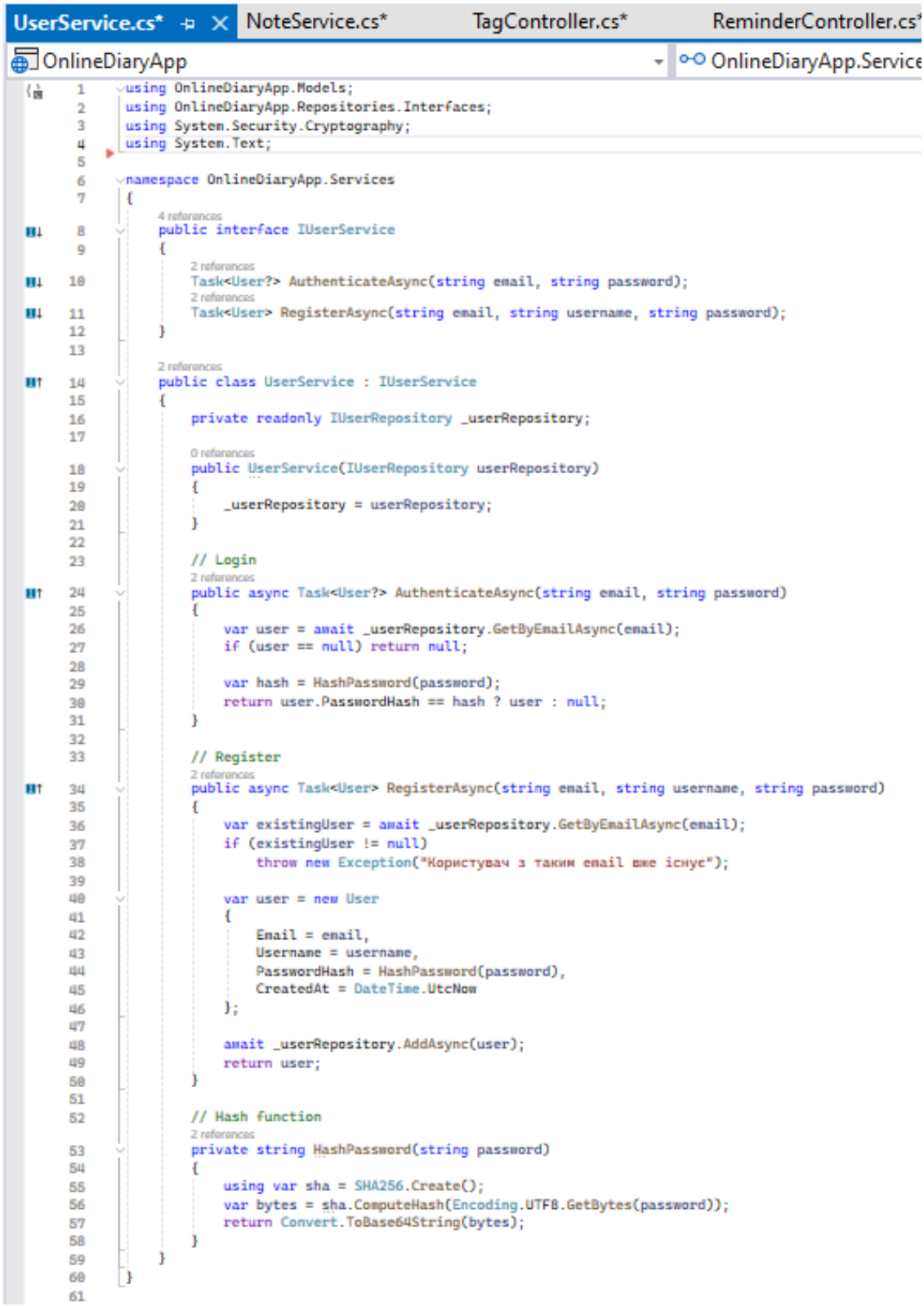
Код 4 – TagController.cs (controllers)


```
using OnlineDiaryApp.Models;
using OnlineDiaryApp.Repositories.Interfaces;

namespace OnlineDiaryApp.Services
{
    5 references
    public interface ISortStrategy
    {
        4 references
        IEnumerable<Note> Sort(IEnumerable<Note> notes);
    }

    1 reference
    public class SortByDateStrategy : ISortStrategy
    {
        3 references
        public IEnumerable<Note> Sort(IEnumerable<Note> notes) =>
            notes.OrderByDescending(n => n.CreatedAt);
    }

    2 references
    public class SortByTagStrategy : ISortStrategy
    {
        private readonly string _tag;
        1 reference
        public SortByTagStrategy(string tag) => _tag = tag;

        3 references
        public IEnumerable<Note> Sort(IEnumerable<Note> notes) =>
            notes.Where(n => n.Tags.Any(t => t.Name == _tag));
    }

    6 references
    public class NoteService
    {
        private readonly INoteRepository _noteRepository;
        private readonly ITagRepository _tagRepository;

        0 references
        public NoteService(INoteRepository noteRepository, ITagRepository tagRepository)
        {
            _noteRepository = noteRepository;
            _tagRepository = tagRepository;
        }

        1 reference
        public async Task<IEnumerable<Note>> GetAllNotesByUserAsync(int userId, ISortStrategy? strategy = null)
        {
            var notes = (await _noteRepository.GetAllAsync())
                        .Where(n => n.UserId == userId);

            return strategy != null ? strategy.Sort(notes) : notes;
        }

        1 reference
        public async Task<IEnumerable<Note>> GetAllNotesAsync(ISortStrategy? strategy = null)
        {
            var notes = await _noteRepository.GetAllAsync();
            return strategy != null ? strategy.Sort(notes) : notes;
        }

        2 references
        public async Task<Note?> GetNoteByIdAsync(int id)
        {
            return await _noteRepository.GetByIdAsync(id);
        }

        1 reference
        public async Task<Note> CreateNoteAsync(string title, string content, int userId, List<int> tagIds)
        {
            var tags = new List<Tag>();
            foreach (var id in tagIds)
            {
                var tag = await _tagRepository.GetByIdAsync(id);
                if (tag != null)
                    tags.Add(tag);
            }

            var utcNow = DateTime.UtcNow ;
```

Код 5.1 – NoteService.cs (services)

```
67
68              var note = new Note
69              {
70                  Title = title,
71                  Content = content,
72                  UserId = userId,
73                  Tags = tags,
74                  CreatedAt = utcNow
75              };
76
77              await _noteRepository.AddAsync(note);
78              await _noteRepository.SaveChangesAsync();
79
80              return note;
81          }
82
            1 reference
83          public async Task UpdateNoteAsync(Note note, List<int> tagIds)
84          {
85              var tags = new List<Tag>();
86              foreach (var id in tagIds)
87              {
88                  var tag = await _tagRepository.GetByIdAsync(id);
89                  if (tag != null)
90                      tags.Add(tag);
91              }
92
93              note.Tags = tags;
94              await _noteRepository.UpdateAsync(note);
95              await _noteRepository.SaveChangesAsync();
96          }
97
            1 reference
98          public async Task DeleteNoteAsync(int id)
99          {
100             await _noteRepository.DeleteAsync(id);
101             await _noteRepository.SaveChangesAsync();
102         }
103
            1 reference
104         public async Task<IEnumerable<Note>> SearchByTitleAsync(string keyword)
105         {
106             var notes = await _noteRepository.GetAllAsync();
107             return notes.Where(n => n.Title.Contains(keyword, StringComparison.OrdinalIgnoreCase));
108         }
109
            0 references
110         public async Task<IEnumerable<Tag>> GetAllTagsAsync()
111         {
112             return await _tagRepository.GetAllAsync();
113         }
114     }
115 }
116
```

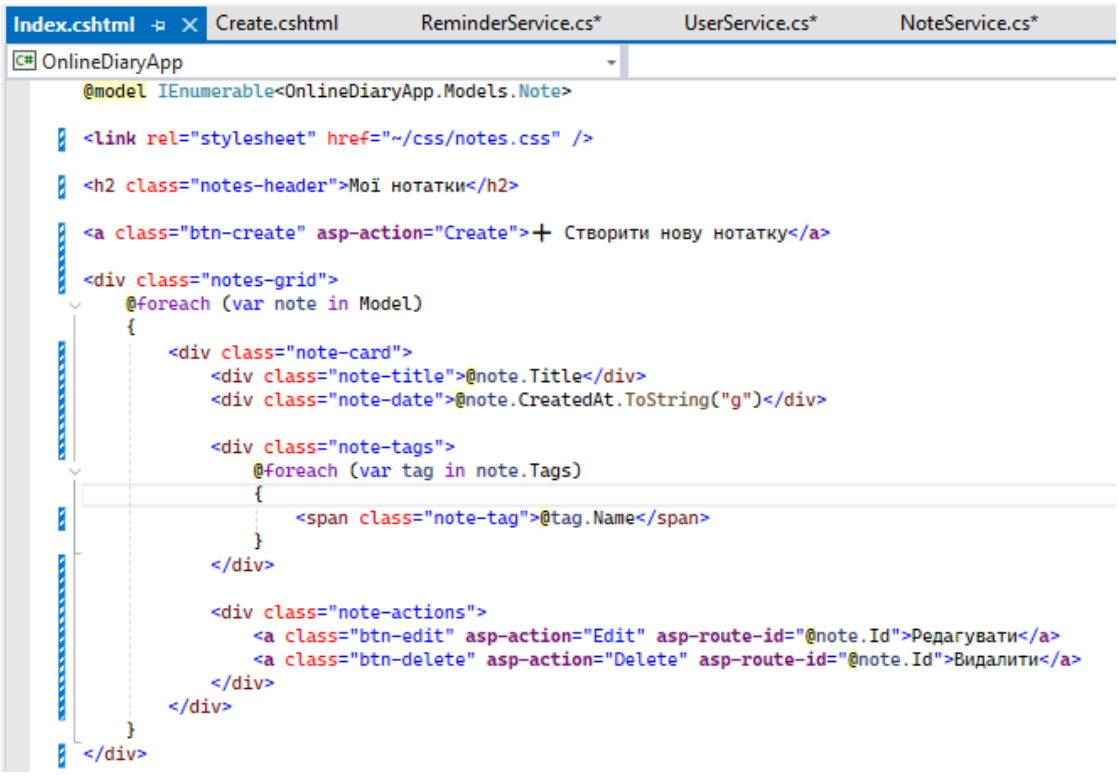
Код 5.2 – NoteService.cs (services)



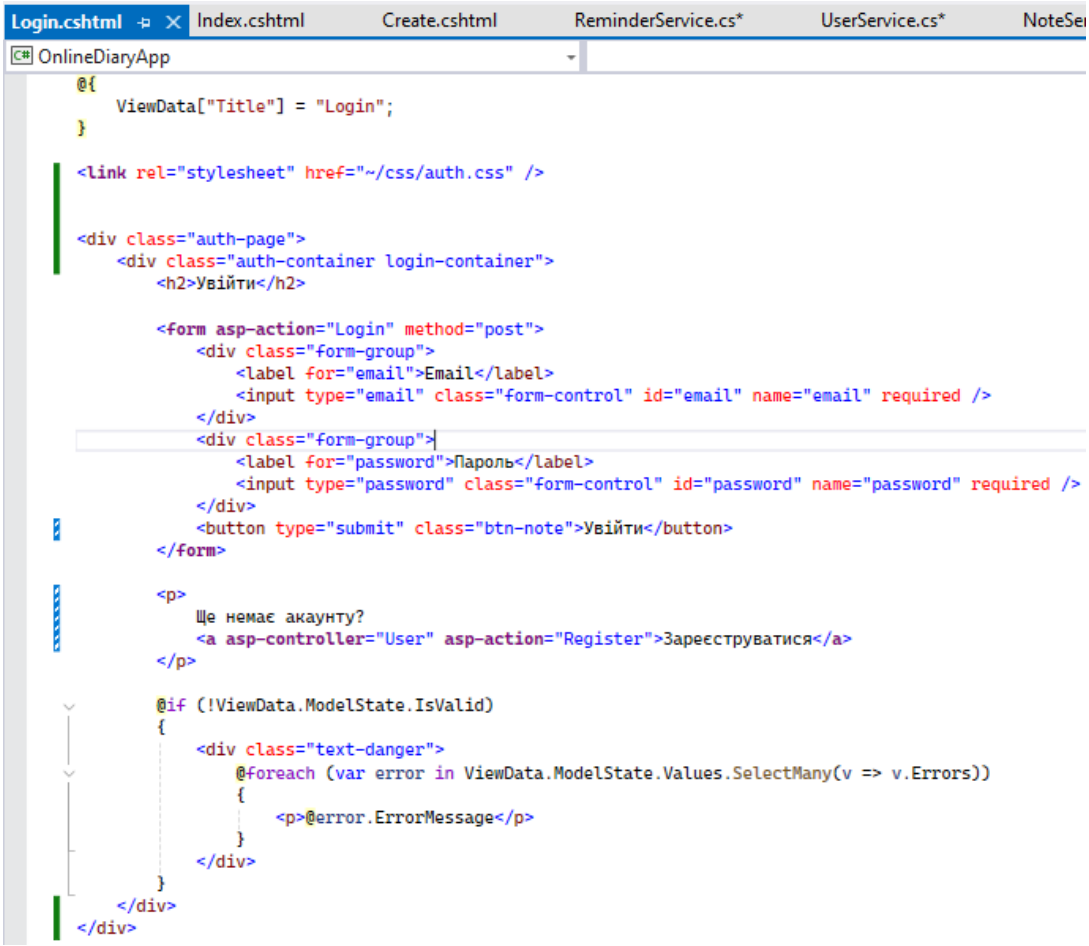
Код 6 – UserService.cs (services)

Код 7 – ReminderService.cs (services)

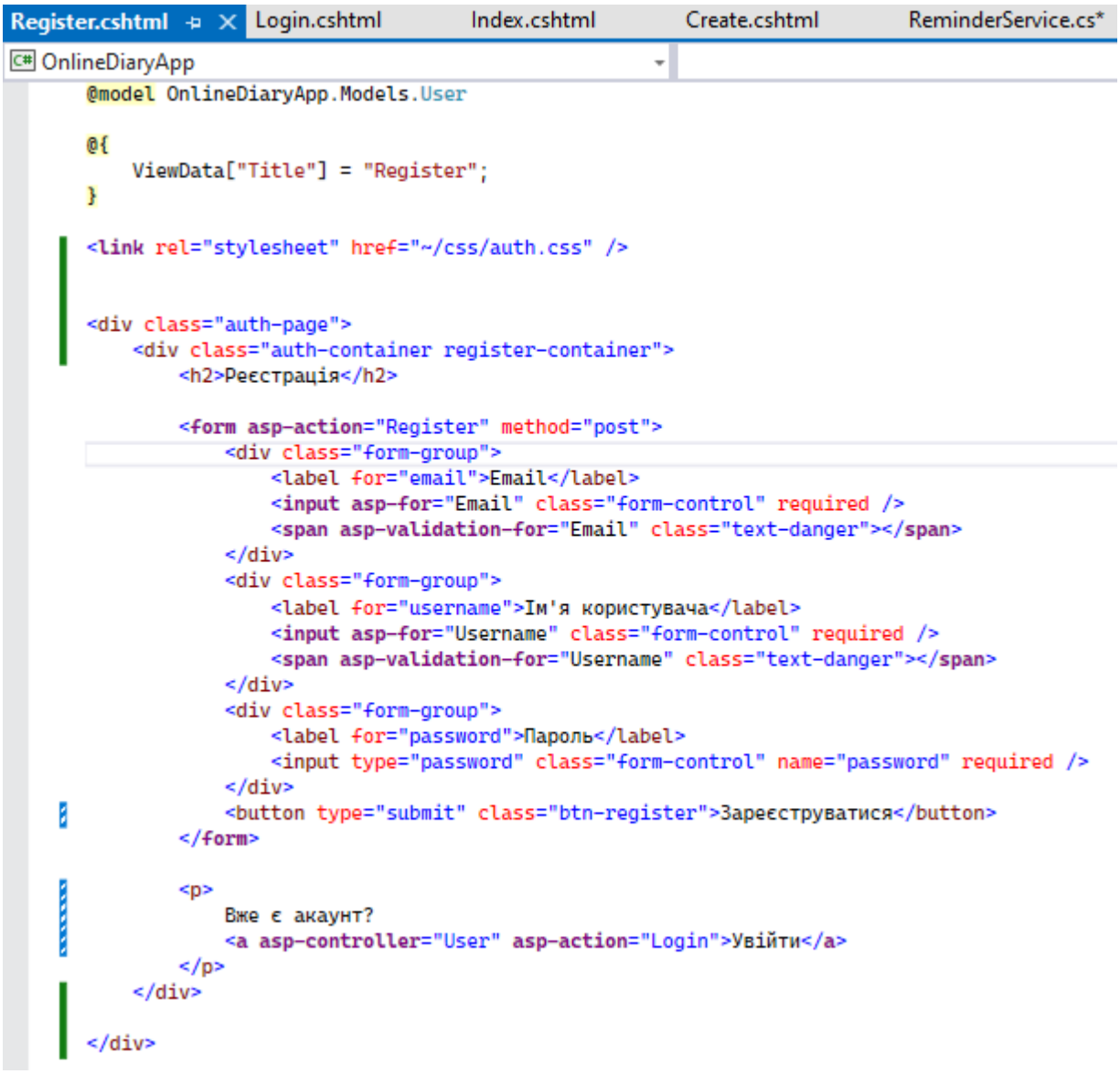
Код 8 – Create.cshtml (Views.Note)



Код 9 – Index.cshtml (Views.Note)



Код 10 – Login.cshtml (Views.User)



Код 11 – Register.cshtml (Views.User)

6. Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму розгортання з описом, діаграму компонентів системи з описом, діаграми послідовностей, а також вихідний код системи, який було додано в цій лабораторній роботі.

Висновки: У ході виконання лабораторної роботи було досягнуто поставлену мету: набуті навички проєктування діаграм розгортання та компонентів для розроблюваної системи, що дозволяє відобразити фізичну структуру системи, розміщення програмних модулів на вузлах та їх взаємозв'язки. Також було

опановано розробку діаграм взаємодії, зокрема діаграм послідовностей, на основі сценаріїв, створених у попередній лабораторній роботі. Це сприяло глибшому розумінню логіки взаємодії об’єктів системи та послідовності виконання операцій, що є важливим етапом у моделюванні програмних систем і підготовці до їхньої реалізації.

Відповіді на контрольні питання:

1. Що собою становить діаграма розгортання?

Діаграма розгортання (Deployment Diagram) показує фізичну архітектуру системи: як програмне забезпечення розміщується на апаратних вузлах, які зв’язки існують між вузлами, а також як компоненти системи розташовані на цих вузлах. Вона корисна для аналізу інфраструктури, серверів, пристроїв і мережевих з’єднань.

2. Які бувають види вузлів на діаграмі розгортання?

- Фізичні вузли (Node): апаратні пристрої або сервери.
- Програмні вузли: середовища виконання (наприклад, JVM, веб-сервер).
- Вузли баз даних: сервери баз даних або сховища.
- Компоненти всередині вузлів: програмні модулі або артефакти, що розміщені на фізичних вузлах.

3. Які бувають зв’язки на діаграмі розгортання?

- Зв’язок асоціації (Association): показує мережеве з’єднання між вузлами.
- Залежність (Dependency): вказує, що один вузол використовує інший.
- Комунікаційний канал (Communication path): фізичне з’єднання між вузлами для обміну даними.

4. Які елементи присутні на діаграмі компонентів?

- Компоненти (Component): незалежні модулі програмного забезпечення з чітко визначеними інтерфейсами.

- Інтерфейси (Interface): точки взаємодії компонентів.
- Пакети (Package): групування компонентів.
- Актори (Actor) або зовнішні системи: можуть бути присутніми для позначення джерел або отримувачів даних.

5. Що становлять собою зв'язки на діаграмі компонентів?

- Залежність (Dependency): один компонент залежить від іншого.
- Асоціація (Association): компоненти взаємодіють між собою.
- Реалізація інтерфейсу (Realization): компонент реалізує певний інтерфейс.
- Інформаційний потік (Information flow): показує рух даних між компонентами.

6. Які бувають види діаграм взаємодії?

- Діаграма послідовностей (Sequence Diagram).
- Діаграма комунікацій (Communication / Collaboration Diagram).
- Діаграма часових подій (Timing Diagram).
- Діаграма взаємодії (Interaction Overview Diagram).

7. Для чого призначена діаграма послідовностей?

Діаграма послідовностей описує динамічну взаємодію між об'єктами у вигляді послідовності повідомлень у часі. Вона використовується для аналізу логіки сценаріїв та послідовності виконання операцій у системі.

8. Які ключові елементи можуть бути на діаграмі послідовностей?

- Об'єкти / Лайнери життя (Lifeline): учасники взаємодії.
- Повідомлення (Message): обмін інформацією між об'єктами.
- Активації (Activation): період, коли об'єкт активний.
- Умовні гілки / Цикли (alt, loop, opt): відображення логічних конструкцій.
- Примітки (Note): додаткова інформація про процеси.

9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання?

Діаграми послідовностей деталізують сценарії, які описані у діаграмах варіантів використання (Use Case Diagram). Кожен варіант використання можна розкласти на кроки у вигляді повідомлень між об’єктами на діаграмі послідовностей.

10. Як діаграми послідовностей пов’язані з діаграмами класів?

Діаграми послідовностей показують об’єкти та їх взаємодію у часі, тоді як діаграми класів описують структуру цих об’єктів (класи, атрибути, методи). Іншими словами, діаграма класів визначає що існує, а діаграма послідовностей – як ці об’єкти взаємодіють.