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»

Варіант - 6

Виконав

Студент групи ІА-31: Перевірив:

Підковка Д. О. Мягкий М. Ю.

Київ 2025

Зміст

1. **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.

2. **Теоретичні відомості:**

Мова UML є загальноцільовою мовою візуального моделювання, яка розроблена для специфікації, візуалізації, проєктування та документування компонентів програмного забезпечення, бізнес-процесів та інших систем. Мова UML є досить строгим та потужним засобом моделювання, який може бути ефективно використаний для побудови концептуальних, логічних та графічних 18 моделей складних систем різного цільового призначення. Ця мова увібрала в себе найкращі якості та досвід методів програмної інженерії, які з успіхом використовувалися протягом останніх років при моделюванні великих та складних систем.

Діаграма – це графічне уявлення сукупності елементів моделі у формі зв'язкового графа, вершинам і ребрам (дугам) якого приписується певна семантика. Нотація канонічних діаграм є основним засобом розробки моделей мовою UML.

У нотації мови UML визначено такі види діаграм:

- варіантів використання (use case diagram);
- класів (class diagram);
- кооперації (collaboration diagram);
- послідовності (sequence diagram);
- станів (statechart diagram);
- діяльності (activity diagram);
- компонентів (component diagram);
- розгортання (deployment diagram).

Діаграма варіантів використання (Use-Cases Diagram) – це UML діаграма за допомогою якої у графічному вигляді можна зобразити вимоги до системи, що розробляється. Діаграма варіантів використання – це вихідна концептуальна модель проєктованої системи, вона не описує внутрішню побудову системи. Діаграма використання складається з:

- Акторів - будь-які об'єкти, суб'єкти чи системи, що взаємодіють з модельованою бізнес-системою ззовні для досягнення своїх цілей або вирішення певних завдань.
- Варіантів використання - служать для опису служб, які система надає актору.
- Відношень: асоціація, узагальнення, залежність, включення, розширення.

Діаграми класів використовуються при моделюванні програмних систем найчастіше. Вони є однією із форм статичного опису системи з погляду її проєктування, показуючи її структуру. Діаграма класів не відображає динамічної поведінки об'єктів зображених на ній класів. На діаграмах класів показуються класи, інтерфейси та відносини між ними.

Діаграма класів містить у собі класи, їхні методи та атрибути, зв’язки. Методи та атрибути мають 4 модифікатори доступу: public, package, protected, private. Зв’язки налічують у собі асоціацію, агрегацію, композицію, успадкування тощо.

3. **Завдання:**
   - Ознайомитись з короткими теоретичними відомостями.
   - Проаналізувати тему та спроєктувати діаграму варіантів використання відповідно до обраної теми лабораторного циклу.
   - Спроєктувати діаграму класів предметної області.
   - Вибрати 3 варіанти використання та написати за ними сценарії використання.
   - 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.
   - Нарисувати діаграму класів для реалізованої частини системи.
   - 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

4. **Хід роботи:**

Тема : Web-browser

Діаграма варіантів використання

Актори:

- Користувач
- Веб-сервер

Варіанти

використання:

- Навігація та перегляд
- Робота з історією
- Робота із закладками
- Технічна обробка

Зв’язки:

- Користувач -> «Ввести URL та перейти».
- Користувач -> «Керувати вкладками».
- Користувач -> «Додати закладку».
- Користувач -> «Переглянути історію».

- «Відправка HTTP-запиту» -> Веб-сервер.
- «Ввести URL та перейти» ..<<include>>..> «Завантаження сторінки».
- «Ввести URL та перейти» ..<<include>>..> «Збереження в історії».
- «Завантаження сторінки» ..<<include>>..> «Відправка HTTP-запиту».
- «Відображення помилки» ..<<extend>>..> «Завантаження сторінки» (Condition: HTTP Error).

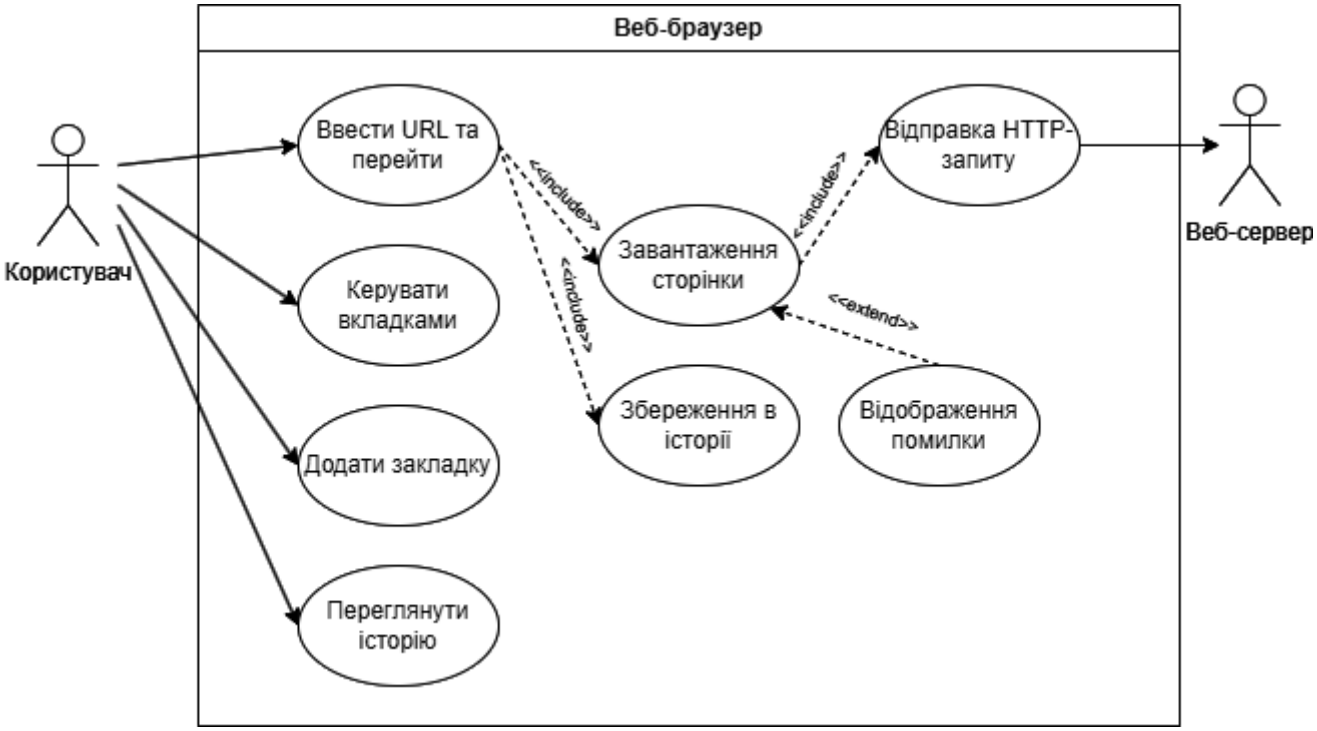

Рис. 1 - Діаграма використання

Сценарії використання:

Завантаження веб-сторінки за URL	
Передумови	Браузер запущено. Наявне активне підключення до мережі Інтернет.
Постумови	Успіх: Веб-сторінка відображена на екрані, запис про відвідування додано в Історію. Відображено сторінку з помилкою, запис в історію не додано.
Взаємодіючі сторони	Користувач (ініціатор), Веб-сервер
Короткий опис	Цей сценарій описує основний бізнес-процес браузера — навігацію в Інтернеті.
Основний потік подій	1. Користувач вводить URL-адресу в адресний рядок і натискає «Enter».
	2. Система (BrowserWindow) валідує формат URL.
	3. Система надсилає HTTP-запит (GET) до відповідного Веб-сервера.
	4. Веб-сервер обробляє запит і повертає відповідь (код 200 OK та тіло HTML).
	5. Система (WebPage) парсить отриманий HTML-код.
	6. Система відображає контент у активній вкладці (Tab).
	7. Система викликає HistoryManager для збереження запису (URL, Title, Date) у базі даних.
Винятки	Альтернативний потік А (Помилка мережі/сервера): На кроці 4 Веб-сервер повертає код помилки (404, 500, 503) або не відповідає. А1. Система ідентифікує код помилки. А2. Система генерує службову сторінку з описом помилки (наприклад, ""404 Not Found""). А3. Сценарій завершується (запис в історію додається з позначкою ""Error"" або ігнорується).
Примітки	-

Додавання сторінки в закладки	
Передумови	У активній вкладці успішно завантажено веб-сторінку. Сторінка ще не додана в закладки (опціонально).
Постумови	У таблиці bookmarks бази даних створено новий запис. Інтерфейс оновлено (наприклад, іконка ""зірочки"" стала зафарбованою).
Взаємодіючі сторони	Користувач, Система
Короткий опис	Цей сценарій описує процес збереження стану системи в базу даних (Persistent Storage).
Основний потік подій	1. Користувач натискає кнопку «Додати в закладки» на панелі інструментів.
	2. Система зчитує поточний Title та URL з активного об'єкта Tab.

	3. Система відображає модальне вікно «Редагування закладки» з полями, заповненими за замовчуванням.
	4. Користувач (опціонально) редагує назву закладки та натискає «Зберегти».
	5. Система передає дані об'єкту BookmarkManager.
	6. BookmarkManager створює об'єкт Bookmark та викликає метод репозиторію Add().
	7. Репозиторій виконує SQL-запит INSERT до БД PostgreSQL.
	8. Система закриває модальне вікно та візуально підтверджує збереження.
Винятки	Альтернативний потік В (Відміна):На кроці 4 Користувач натискає «Скасувати». В1. Система закриває вікно без змін у БД.
Примітки	-

Перегляд історії відвідувань	
Передумови	Браузер запущено. У базі даних є записи про попередні сесії.
Постумови	Користувачеві відображено список відвіданих сайтів, відсортований за датою.
Взаємодіючі сторони	Користувач, Система
Короткий опис	Цей сценарій описує основний бізнес-процес браузера — навігацію в Інтернеті.
Основний потік подій	1. Користувач відкриває головне меню та обирає пункт «Історія».
	2. Система відкриває нову вкладку (або бічну панель) для відображення історії.
	3. Система звертається до HistoryManager із запитом GetHistory().
	4. Менеджер звертається до репозиторію, який виконує SQL-запит SELECT * FROM history ORDER BY visit_date DESC.
	5. Система отримує список об'єктів HistoryItem.
	6. Система групує записи за датами (Сьогодні, Вчора, Минулий тиждень) та рендерить їх у списку.
	7. Користувач переглядає список.
Винятки	-
Примітки	Якщо історія порожня, система відображає повідомлення «Записів не знайдено».

Структура БД:

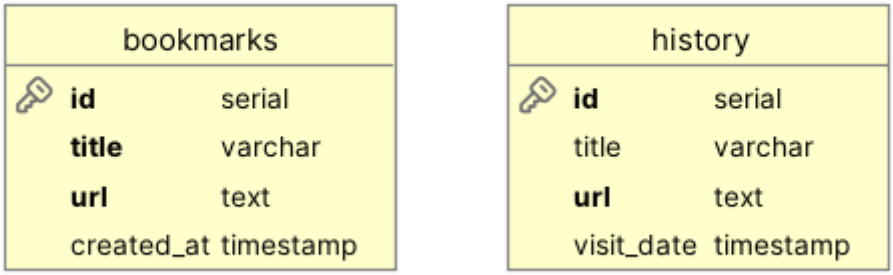

Рис. 2 - Структура БД

Діаграми класів:

Предметна область:

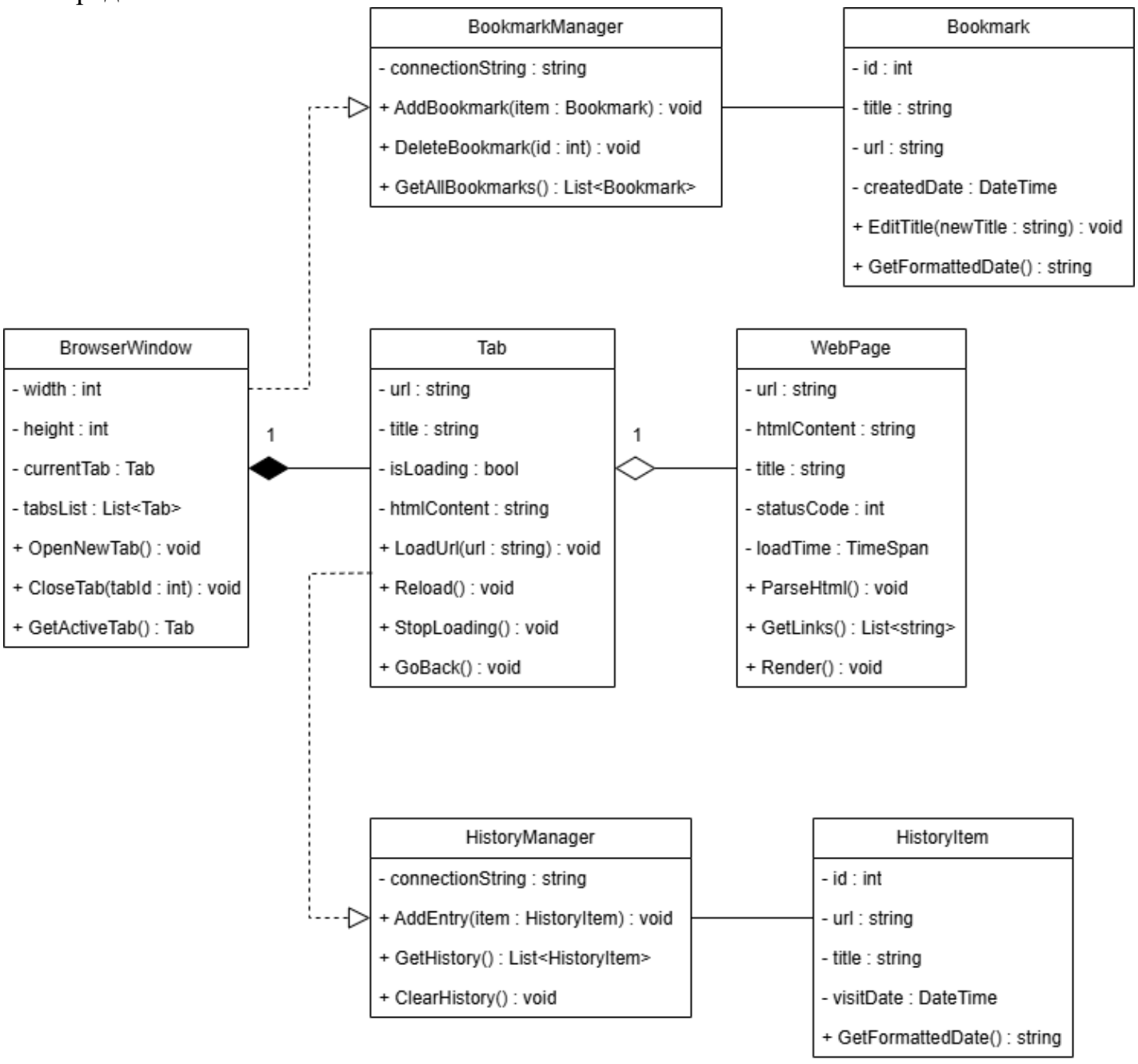

Рис. 3 - Діаграма класів предметної області

Опис класів предметної області

- Клас BrowserWindow
    - Призначення: Головний клас графічного інтерфейсу, що виступає контейнером для всього додатку. Він керує життєвим циклом вкладок та загальними параметрами вікна.
    - Поля (Атрибути):

        - width: int. Ширина вікна у пікселях.
        - height: int. Висота вікна у пікселях.
        - currentTab: Tab. Посилання на активну (обрану користувачем) вкладку.
        - tabsList: List<Tab>. Колекція всіх відкритих вкладок.
    - Методи:
        - OpenNewTab(): Створює новий екземпляр класу Tab та додає його до списку. Повертає void.
        - CloseTab(int tabId): Закриває вкладку за вказаним ідентифікатором та видаляє її з пам'яті. Повертає void.
        - GetActiveTab(): Повертає об'єкт типу Tab, з яким зараз взаємодіє користувач.

- Клас Tab
    - Призначення: Відповідає за логіку окремої вкладки браузера. Слугує "мостом" між інтерфейсом (BrowserWindow) та контентом (WebPage), а також ініціює збереження історії.
    - Поля (Атрибути):
        - url: string. Поточна адреса, введена в адресний рядок вкладки.
        - title: string. Заголовок вкладки (зазвичай береться з тегу <title>).
        - isLoading: bool. Прапорець, що показує стан завантаження (true/false).
    - Методи:
        - LoadUrl(string url): Ініціює процес завантаження сторінки за адресою. Повертає void.
        - Reload(): Перезавантажує поточну сторінку. Повертає void.

        - StopLoading(): Примусово зупиняє завантаження контенту. Повертає void.

        - GoBack(): Повертає користувача на попередню сторінку. Повертає void.

- Клас WebPage

    - Призначення: Представляє завантажений контент веб-сторінки. Відповідає за зберігання "сирих" даних, отриманих від сервера.

    - Поля (Атрибути):

        - url: string. Фактична адреса ресурсу (може відрізнятися від введеної через редіректи).

        - htmlContent: string. Текст HTML-коду сторінки.

        - title: string. Заголовок сторінки.

        - statusCode: int. HTTP-код відповіді сервера (наприклад, 200 або 404).

        - loadTime: TimeSpan. Час, витрачений на завантаження.

    - Методи:

        - ParseHtml(): Аналізує HTML-код для виділення заголовка та метаданих. Повертає void.

        - GetLinks(): Повертає список List<string> всіх гіперпосилань на сторінці.

        - Render(): Відповідає за візуалізацію HTML-коду. Повертає void.

- Клас HistoryManager

    - Призначення: Шар бізнес-логіки для керування історією відвідувань. Інкапсулює взаємодію з базою даних щодо сутностей історії.

    - Поля (Атрибути):

        - connectionString: string. Рядок підключення до бази даних PostgreSQL.

    - Методи:

        - AddEntry(HistoryItem item): Валідує та передає новий запис історії в репозиторій для збереження. Повертає void.

        - GetHistory(): Повертає повний список List<HistoryItem> відвіданих сторінок.

        - ClearHistory(): Видаляє всі записи з історії. Повертає void.

- Клас HistoryItem (Entity)

    - Призначення: Клас-сутість (DTO), що представляє один рядок таблиці історії в базі даних.

    - Поля (Атрибути):

        - id: int. Унікальний ідентифікатор запису.

        - url: string. Адреса відвіданої сторінки.

        - title: string. Назва сторінки на момент відвідування.

        - visitDate: DateTime. Точна дата та час відвідування.

    - Методи:

        - GetFormattedDate(): Повертає дату у зручному рядковому форматі (наприклад, "DD.MM.YYYY HH:mm").

- Клас BookmarkManager

    - Призначення: Шар бізнес-логіки для керування закладками користувача.

    - Поля (Атрибути):

        - connectionString: string. Рядок підключення до бази даних.

    - Методи:

        - AddBookmark(Bookmark item): Зберігає нову закладку в БД. Повертає void.
        - DeleteBookmark(int id): Видаляє закладку за її ID. Повертає void.
        - GetAllBookmarks(): Повертає список List<Bookmark> усіх збережених закладок.

- Клас Bookmark (Entity)
    - Призначення: Клас-сутість (DTO), що представляє один рядок таблиці закладок у базі даних.
    - Поля (Атрибути):
        - id: int. Унікальний ідентифікатор закладки.
        - title: string. Назва закладки, задана користувачем.
        - url: string. Адреса збереженого ресурсу.
        - createdDate: DateTime. Дата створення закладки.
    - Методи:
        - EditTitle(string newTitle): Дозволяє змінити назву закладки. Повертає void.
        - GetFormattedDate(): Повертає дату створення у рядковому форматі.

Класи реалізації:

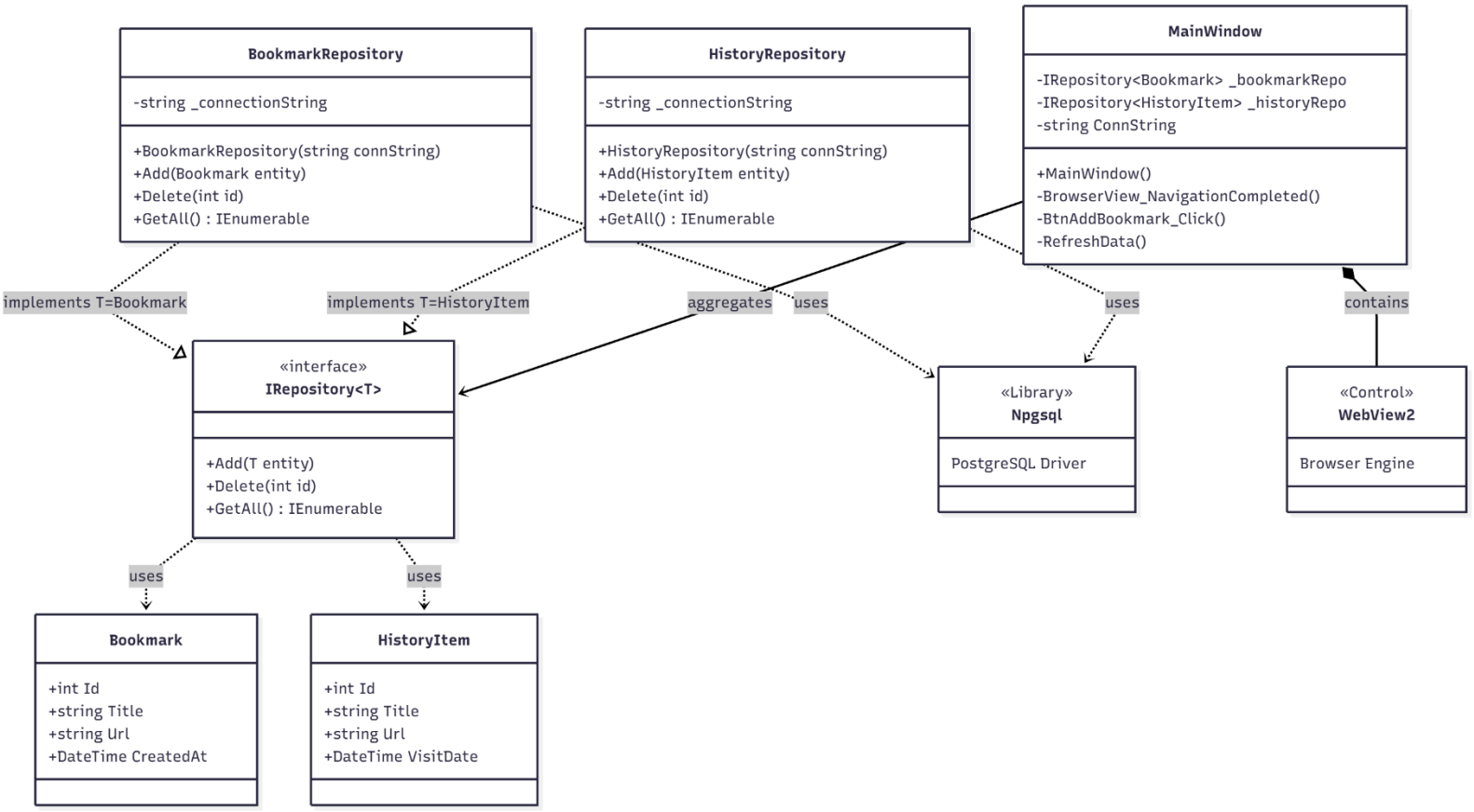

Рис. 4 - Діаграма класів даних

Опис класів реалізованої системи

- Клас MainWindow

  - Призначення: Головний клас рівня представлення (UI). Він відповідає за ініціалізацію вікна, обробку подій від користувача (кліки, введення тексту) та взаємодію з браузерним рушієм WebView2. Також він виконує роль "Composition Root", ініціалізуючи репозиторії.

  - Поля (Атрибути):

    - ConnString: const string. Зберігає рядок підключення до бази даних PostgreSQL (містить хост, користувача, пароль).

- _bookmarkRepo: IRepository<Bookmark>. Посилання на об'єкт доступу до даних закладок (через інтерфейс).

- _historyRepo: IRepository<HistoryItem>. Посилання на об'єкт доступу до даних історії (через інтерфейс).

- BrowserView: WebView2. Елемент керування, що безпосередньо відображає веб-сторінки.

- Методи:

  - MainWindow(): Конструктор. Ініціалізує компоненти XAML, створює екземпляри репозиторіїв та підписується на події.

  - BrowserView_NavigationCompleted(sender, e): Обробник події. Автоматично викликається після завантаження сторінки. Створює запис HistoryItem і зберігає його в БД.

  - BtnAddBookmark_Click(sender, e): Обробник кнопки. Зчитує поточний Title та URL, створює об'єкт Bookmark і передає його в репозиторій.

  - MapsToUrl(): Валідує введений в адресний рядок текст і змушує браузер перейти за посиланням.

  - RefreshData(): Оновлює списки в інтерфейсі (ListBox), запитуючи актуальні дані з репозиторіїв.

- Інтерфейс IRepository<T>
    - Призначення: Універсальний "контракт", який описує базові CRUD-операції. Він дозволяє класу MainWindow працювати з будь-яким типом даних (Bookmark або HistoryItem) однаковим способом, дотримуючись принципу інверсії залежностей.
    - Параметри:
        - T: Тип сутності, з якою працює репозиторій (клас даних).
    - Методи:
        - Add(T entity): Додає новий об'єкт сутності у базу даних. Повертає void.
        - Delete(int id): Видаляє запис із бази даних за унікальним ідентифікатором. Повертає void.
        - GetAll(): Виконує вибірку всіх записів з таблиці та повертає колекцію IEnumerable<T>.
- Клас BookmarkRepository
    - Призначення: Реалізація інтерфейсу IRepository конкретно для сутності Bookmark. Цей клас містить специфічний код для роботи з PostgreSQL через

бібліотеку Npgsql.

- Поля (Атрибути):
  - _connectionString: string. Рядок підключення, отриманий через конструктор.
- Методи:
  - BookmarkRepository(string connectionString): Конструктор, що приймає налаштування підключення.
  - Add(Bookmark entity): Відкриває з'єднання, формує SQL-запит INSERT INTO bookmarks... з параметрами та виконує його.
  - GetAll(): Виконує SQL-запит SELECT..., зчитує дані через NpgsqlDataReader та мапить їх у список об'єктів Bookmark.
  - Delete(int id): Виконує SQL-запит DELETE FROM bookmarks WHERE id = ...

- Клас HistoryRepository
  - Призначення: Реалізація інтерфейсу IRepository конкретно для сутності HistoryItem. Відповідає за таблицю history у базі даних.

- Поля (Атрибути):
    - _connectionString: string. Рядок підключення.
- Методи:
    - HistoryRepository(string connectionString): Конструктор.
    - Add(HistoryItem entity): Виконує SQL-запит INSERT INTO history.... Відрізняється від закладок набором полів (використовує visit_date).
    - GetAll(): Виконує вибірку історії, сортуючи записи за датою відвідування у зворотному порядку (ORDER BY visit_date DESC).
    - Delete(int id): Видаляє запис з історії.

- Зовнішня бібліотека Npgsql
    - Призначення: .NET Data Provider для PostgreSQL.
    - Використання:
        - NpgsqlConnection: Забезпечує фізичне з'єднання з сервером БД.

        - NpgsqlCommand: Дозволяє виконувати SQL-команди та процедури.

        - NpgsqlDataReader: Забезпечує швидке читання потоку рядків, отриманих від бази даних (forward-only).

- Зовнішній компонент WebView2

    - Призначення: Елемент керування WPF, що вбудовує веб-технології (HTML/CSS/JS) у нативний додаток, використовуючи рушій Microsoft Edge (Chromium).

    - Використання:

        - Source: Властивість для встановлення URL.

        - CoreWebView2.DocumentTitle: Властивість для отримання заголовка поточної сторінки.

        - NavigationCompleted: Подія, що сигналізує про завершення завантаження контенту.

Вихідний код класів:

```
using System.Collections.Generic;

namespace WebBrowser.Coursework.Repositories
{
    4 references
    public interface IRepository<T>
    {
        4 references
        void Add(T entity);
        4 references
        void Delete(int id);
        5 references
        IEnumerable<T> GetAll();
    }
}
```

Рис. 5 - Код інтерфейсу IRepository<T>

```
using System;
using System.Collections.Generic;
using Npgsql; // Обов'язково додати using
using WebBrowser.Coursework.Entities;

namespace WebBrowser.Coursework.Repositories
{
    2 references
    public class BookmarkRepository : IRepository<Bookmark>
    {
        private readonly string _connectionString;

        1 reference
        public BookmarkRepository(string connectionString)
        {
            _connectionString = connectionString;
        }

        3 references
        public void Add(Bookmark entity)
        {
            using (var conn = new NpgsqlConnection(_connectionString))
            {
                conn.Open();
                // Параметризований запит для захисту від SQL Injection
                string sql = "INSERT INTO bookmarks (title, url, created_at) VALUES (@t, @u, @d)";
                using (var cmd = new NpgsqlCommand(sql, conn))
                {
                    cmd.Parameters.AddWithValue("t", entity.Title);
                    cmd.Parameters.AddWithValue("u", entity.Url);
                    cmd.Parameters.AddWithValue("d", entity.CreatedAt);
                    cmd.ExecuteNonQuery();
                }
            }
        }
```

```
        4 references
        public IEnumerable<Bookmark> GetAll()
        {
            var list = new List<Bookmark>();
            using (var conn = new NpgsqlConnection(_connectionString))
            {
                conn.Open();
                string sql = "SELECT id, title, url, created_at FROM bookmarks ORDER BY created_at DESC";
                using (var cmd = new NpgsqlCommand(sql, conn))
                using (var reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        list.Add(new Bookmark
                        {
                            Id = reader.GetInt32(0),
                            Title = reader.GetString(1),
                            Url = reader.GetString(2),
                            CreatedAt = reader.GetDateTime(3)
                        });
                    }
                }
            }
            return list;
        }

        3 references
        public void Delete(int id)
        {
            using (var conn = new NpgsqlConnection(_connectionString))
            {
                conn.Open();
                string sql = "DELETE FROM bookmarks WHERE id = @id";
```

```
64                  string sql = "DELETE FROM bookmarks WHERE id = @id";
65                  using (var cmd = new NpgsqlCommand(sql, conn))
66                  {
67                      cmd.Parameters.AddWithValue("id", id);
68                      cmd.ExecuteNonQuery();
69                  }
70              }
71          }
72      }
73  }
```

Рис. 6,7,8 - Код класу BookmarkRepository

```
using System;
using System.Collections.Generic;
using Npgsql;
using WebBrowser.Coursework.Entities;

namespace WebBrowser.Coursework.Repositories
{
    2 references
    public class HistoryRepository : IRepository<HistoryItem>
    {
        private readonly string _connectionString;

        1 reference
        public HistoryRepository(string connectionString)
        {
            _connectionString = connectionString;
        }

        // Додавання запису про відвідування
        3 references
        public void Add(HistoryItem entity)
        {
            using (var conn = new NpgsqlConnection(_connectionString))
            {
                conn.Open();
                string sql = "INSERT INTO history (title, url, visit_date) VALUES (@t, @u, @d)";
                using (var cmd = new NpgsqlCommand(sql, conn))
                {
                    cmd.Parameters.AddWithValue("t", entity.Title ?? "No Title");
                    cmd.Parameters.AddWithValue("u", entity.Url);
                    cmd.Parameters.AddWithValue("d", entity.VisitDate);
                    cmd.ExecuteNonQuery();
                }
            }
        }

        // Отримання повної історії
        4 references
        public IEnumerable<HistoryItem> GetAll()
        {
            var list = new List<HistoryItem>();
            using (var conn = new NpgsqlConnection(_connectionString))
            {
                conn.Open();
                // Сортуємо: нові зверху
                string sql = "SELECT id, title, url, visit_date FROM history ORDER BY visit_date DESC";

                using (var cmd = new NpgsqlCommand(sql, conn))
                using (var reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        list.Add(new HistoryItem
                        {
                            Id = reader.GetInt32(0),
                            Title = reader.IsDBNull(1) ? "Без назви" : reader.GetString(1),
                            Url = reader.GetString(2),
                            VisitDate = reader.GetDateTime(3)
                        });
                    }
                }
            }
            return list;
        }

        3 references
        public void Delete(int id)
        {
            using (var conn = new NpgsqlConnection(_connectionString))
            {
```

```
                conn.Open();
                string sql = "DELETE FROM history WHERE id = @id";
                using (var cmd = new NpgsqlCommand(sql, conn))
                {
                    cmd.Parameters.AddWithValue("id", id);
                    cmd.ExecuteNonQuery();
                }
            }
        }
    }
}
```

Рис. 9,10,11 - Код класу HistoryRepository

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using Microsoft.Web.WebView2.Core; // Потрібно для подій браузера
using WebBrowser.Coursework.Entities;
using WebBrowser.Coursework.Repositories;

namespace WebBrowser.Coursework
{
    /// <summary>
    /// Логіка взаємодії для MainWindow.xaml
    /// </summary>
    2 references
    public partial class MainWindow : Window
    {
        // Рядок підключення до БД (ЗМІНІТЬ ПАРОЛЬ НА СВІЙ)
        private const string ConnString = "Host=localhost;Username=postgres;Password=YOUR_PASSWORD;Database=browser_db";

        // Оголошення репозиторіїв (Abstraction)
        private readonly IRepository<Bookmark> _bookmarkRepo;
        private readonly IRepository<HistoryItem> _historyRepo;

        0 references
        public MainWindow()
        {
            InitializeComponent();

            try
            {
                // 1. Ініціалізація доступу до даних (Dependency Injection вручну)
                _bookmarkRepo = new BookmarkRepository(ConnString);
                _historyRepo = new HistoryRepository(ConnString);

                // 2. Завантаження початкових даних з БД в інтерфейс
                RefreshData();
```

```
            // 3. Підписка на події WebView2
            // Ця подія спрацьовує, коли навігація завершена успішно або з помилкою
            BrowserView.NavigationCompleted += BrowserView_NavigationCompleted;
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Критична помилка ініціалізації: {ex.Message}\nПеревірте рядок підключення до БД.",
        }
    }

    // ========================================================
    // ЛОГІКА БРАУЗЕРА (WebView2)
    // ========================================================

    // Автоматичне збереження в історію при завершенні завантаження
    1 reference
    private void BrowserView_NavigationCompleted(object sender, CoreWebView2NavigationCompletedEventArgs e)
    {
        if (!e.IsSuccess) return; // Якщо помилка завантаження - не пишемо в історію

        try
        {
            // Оновлюємо адресний рядок актуальним URL
            string currentUrl = BrowserView.Source.ToString();
            string currentTitle = BrowserView.CoreWebView2.DocumentTitle;
            AddressBar.Text = currentUrl;

            // Створення запису історії (Сценарій UC-01)
            var historyItem = new HistoryItem
            {
                Title = string.IsNullOrEmpty(currentTitle) ? currentUrl : currentTitle,
                Url = currentUrl,
                VisitDate = DateTime.Now
            };

            // Збереження в БД через репозиторій
            _historyRepo.Add(historyItem);

            // Оновлення списку історії в інтерфейсі (можна оптимізувати, не оновлювати щоразу)
            RefreshHistoryList();
        }
        catch (Exception ex)
        {
            // Логування помилок (у реальному проєкті - у файл)
            System.Diagnostics.Debug.WriteLine($"Помилка запису історії: {ex.Message}");
        }
    }

    1 reference
    private void BtnGo_Click(object sender, RoutedEventArgs e)
    {
        NavigateToUrl();
    }

    1 reference
    private void AddressBar_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.Key == Key.Enter)
        {
            NavigateToUrl();
        }
    }

    2 references
    private void NavigateToUrl()
    {
        string url = AddressBar.Text;
        if (string.IsNullOrWhiteSpace(url)) return;

        // Проста перевірка на протокол
        if (!url.StartsWith("http://") && !url.StartsWith("https://"))
```

Рис. 12,13,14 - Код класу MainWindow

5. Питання до лабораторної роботи:

1. Що таке UML? – універсальна мова для опису та візуалізації систем.
2. Що таке діаграма класів UML? – схема з класами, їхніми полями, методами та зв’язками.
3. Які діаграми UML називають канонічними? – стандартні: класів, об’єктів, варіантів використання, послідовностей тощо.

4. Що таке діаграма варіантів використання? – показує акторів і функції, які вони виконують у системі.
5. Що таке варіант використання? – дія або функція, яку виконує користувач.
6. Які відношення можуть бути відображені на діаграмі використання? – асоціація, include, extend, узагальнення.
7. Що таке сценарій? – опис кроків взаємодії користувача з системою.
8. Що таке діаграма класів? – схема структури системи через класи та їх зв’язки.
9. Які зв’язки між класами ви знаєте? – асоціація, агрегація, композиція, наслідування, залежність.
10. Чим відрізняється композиція від агрегації? – у композиції частини не існують без цілого, в агрегації можуть.
11. Чим відрізняється зв'язки типу агрегації від зв'язків композиції на діаграмах класів? – агрегація має порожній ромб, композиція – зафарбований.
12. Що являють собою нормальні форми баз даних? – правила організації таблиць для уникнення дублювання.
13. Що таке фізична модель бази даних? Логічна? – фізична: як реально зберігається; логічна: концептуальна схема.
14. Який взаємозв'язок між таблицями БД та програмними класами? – таблиця ≈ клас, рядок ≈ об’єкт, стовпчик ≈ поле.

6. Висновок

У процесі виконання лабораторної роботи з теми "Розробка веб-браузера" було здійснено комплексний аналіз вимог та розроблено моделі системи за допомогою мови UML, включаючи діаграми варіантів використання, діаграми класів та схему бази даних.

Практичну частину реалізовано на платформі .NET (WPF) з використанням СУБД PostgreSQL. У коді застосовано сучасні підходи до проєктування, зокрема архітектурний патерн Repository, що дозволило відокремити бізнес-логіку від шару доступу до даних та забезпечити гнучкість і масштабованість архітектури.

Результатом роботи є функціональний програмний додаток, що успішно виконує поставлені завдання (навігація, збереження історії та закладок) і має потенціал для подальшого розширення в майбутньому.