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

Лабораторна робота №2

Технології розробки програмного забезпечення

«Основи проектування»

«Веб-браузер»

Виконав:
студент групи ІА-33
Мартинюк Ю.Р.

Перевірив:
Мягкий Михайло Юрійович

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.

Зміст

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.

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 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 (diagram) – 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.

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.

Діаграми варіантів використання є відправною точкою при збиранні вимог до програмного продукту та його реалізації. Дана модель будується на аналітичному етапі побудови програмного продукту (збір та аналіз вимог) і дозволяє бізнес-аналітикам отримати більш повне уявлення про необхідне програмне забезпечення та документувати його.

Діаграма варіантів використання складається з низки елементів. Основними елементами є: варіанти використання або прецеденти (use case), актор або дійова

особа (actor) та відносини між акторами та варіантами використання (relationship).

Сценарії використання – це текстові уявлення тих процесів, які відбуваються при взаємодії користувачів системи та самої системи. Вони є чітко формалізованими, покроковими інструкціями, що описують той чи інший процес у термінах кроків досягнення мети. Сценарії використання однозначно визначають кінцевий результат.

Діаграми класів використовуються при моделюванні програмних систем найчастіше. Вони є однією із форм статичного опису системи з погляду її проєктування, показуючи її структуру [3]. Діаграма класів не відображає динамічної поведінки об'єктів зображених на ній класів. На діаграмах класів показуються класи, інтерфейси та відносини між ними.

Клас – це основний будівельний блок програмної системи. Це поняття є і в мовах програмування, тобто між класами UML та програмними класами є відповідність, що є основою для автоматичної генерації програмних кодів або для виконання реінжинірингу. Кожен клас має назву, атрибути та операції. Клас на діаграмі показується як прямокутник, розділений на 3 області. У верхній міститься назва класу, у середній – опис атрибутів (властивостей), у нижній – назви операцій – послуг, що надаються об'єктами цього класу.

Логічна модель бази даних є структурою таблиць, уявлень, індексів та інших логічних елементів бази даних, що дозволяють власне програмування та використання бази даних. Процес створення логічної моделі бази даних зветься проєктування бази даних (database design). Проєктування відбувається у зв'язку з опрацюванням архітектури програмної системи, оскільки база даних створюється зберігання даних, одержуваних з програмних класів.

Тема проєкту

6. Web-browser (proxy, chain of responsibility, factory method, template method, visitor, p2p) Веб-браузер повинен мати можливість зробити наступне: мати адресний рядок для введення адреси сайту, переміщатися і відображати структуру html документа, переглядати підключений javascript та css файли,

перегляд всіх підключених ресурсів (зображень), коректна обробка відповідей з сервера (коди відповідей HTTP) – переходи при перенаправленнях, відображення сторінок 404 і 502/503.

Хід роботи

Діаграма варіантів використання

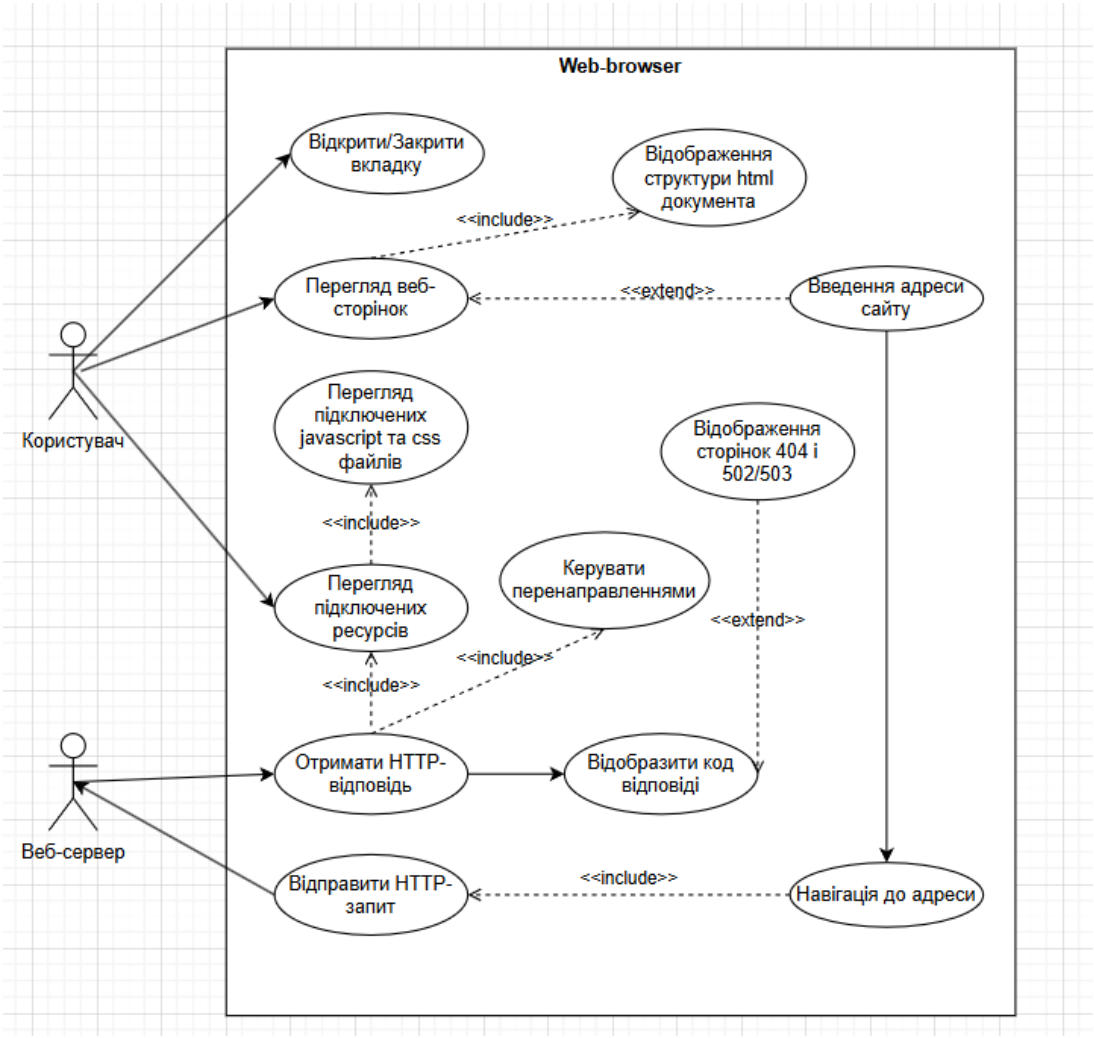

Рис. 1 – Діаграма компонентів

Сценарії використання

Сценарій 1. Навігація до адреси

Передумови:

- Браузер відкритий.
- Є стабільне підключення до мережі.

Постумови:

- Користувач отримує завантажену веб-сторінку або сторінку з

повідомленням про помилку.

Взаємодіючі сторони:

- Користувач, Браузер (система), Веб-сервер.

Короткий опис:
Користувач вводить адресу в браузері, система формує та відправляє HTTP-запит, отримує від сервера HTTP-відповідь і відображає веб-сторінку або повідомлення про помилку.

Основний перебіг подій:

1. Користувач вводить адресу сайту в адресний рядок браузера.
2. Система виконує перетворення введеного URL у HTTP-запит.
3. Система відправляє HTTP-запит на відповідний веб-сервер.
4. Веб-сервер обробляє запит і надсилає HTTP-відповідь.
5. Система перевіряє код відповіді.
6. Якщо код 200 — браузер завантажує HTML-документ та підключені ресурси.
7. Користувач бачить відображену веб-сторінку.

Винятки:

- Виняток №1: Код відповіді 3xx — система виконує перенаправлення.
- Виняток №2: Код відповіді 4xx або 5xx — система формує та показує сторінку помилки.

Примітки: Відсутні.

Сценарій 2. Перегляд підключених ресурсів

Передумови:

- Веб-сторінка вже завантажена.

Постумови:

- Користувач отримує список ресурсів і може переглянути їхній вміст.

Взаємодіючі сторони:

- Користувач, Браузер (система), Веб-сервер.

Короткий опис:
Користувач відкриває інструменти перегляду ресурсів, система відображає всі ресурси поточної сторінки та дає змогу переглядати їхні властивості чи вміст.

Основний перебіг подій:

1. Користувач відкриває інструменти перегляду ресурсів у браузері.
2. Система формує список усіх ресурсів, підключених до поточної HTML-сторінки.

3. Система відображає структуру та інформацію про ресурси.
4. Користувач обирає конкретний ресурс для перегляду.
5. Система завантажує і показує вміст або метадані цього ресурсу.

Винятки:

- Виняток №1: Якщо ресурс недоступний (код 404/502) — система показує повідомлення про помилку для цього ресурсу.

Примітки: Відсутні.

Сценарій 3. Відображення сторінки помилки (404, 502/503)

Передумови:

- Користувач намагається відкрити веб-сторінку.
- Сервер повертає помилку.

Постумови:

- Користувач бачить спеціальну сторінку з повідомленням про помилку.

Взаємодіючі сторони:

- Користувач, Браузер (система), Веб-сервер.

Короткий опис:
Коли сервер повертає код помилки (404, 502 або 503), система замість звичайного рендерингу сторінки показує повідомлення про помилку.

Основний перебіг подій:

1. Користувач вводить адресу сайту в браузері.
2. Система надсилає HTTP-запит на сервер.
3. Веб-сервер повертає HTTP-відповідь із кодом 404, 502 або 503.
4. Система аналізує код відповіді.
5. Замість завантаження HTML система формує сторінку помилки.
6. Користувач бачить повідомлення з відповідним кодом.

Винятки:

- Виняток №1: Код відповіді 200 — система переходить до завантаження та відображення сторінки.
- Виняток №2: Код відповіді 3xx — система виконує перенаправлення.

Примітки: Відсутні.

Діаграма класів

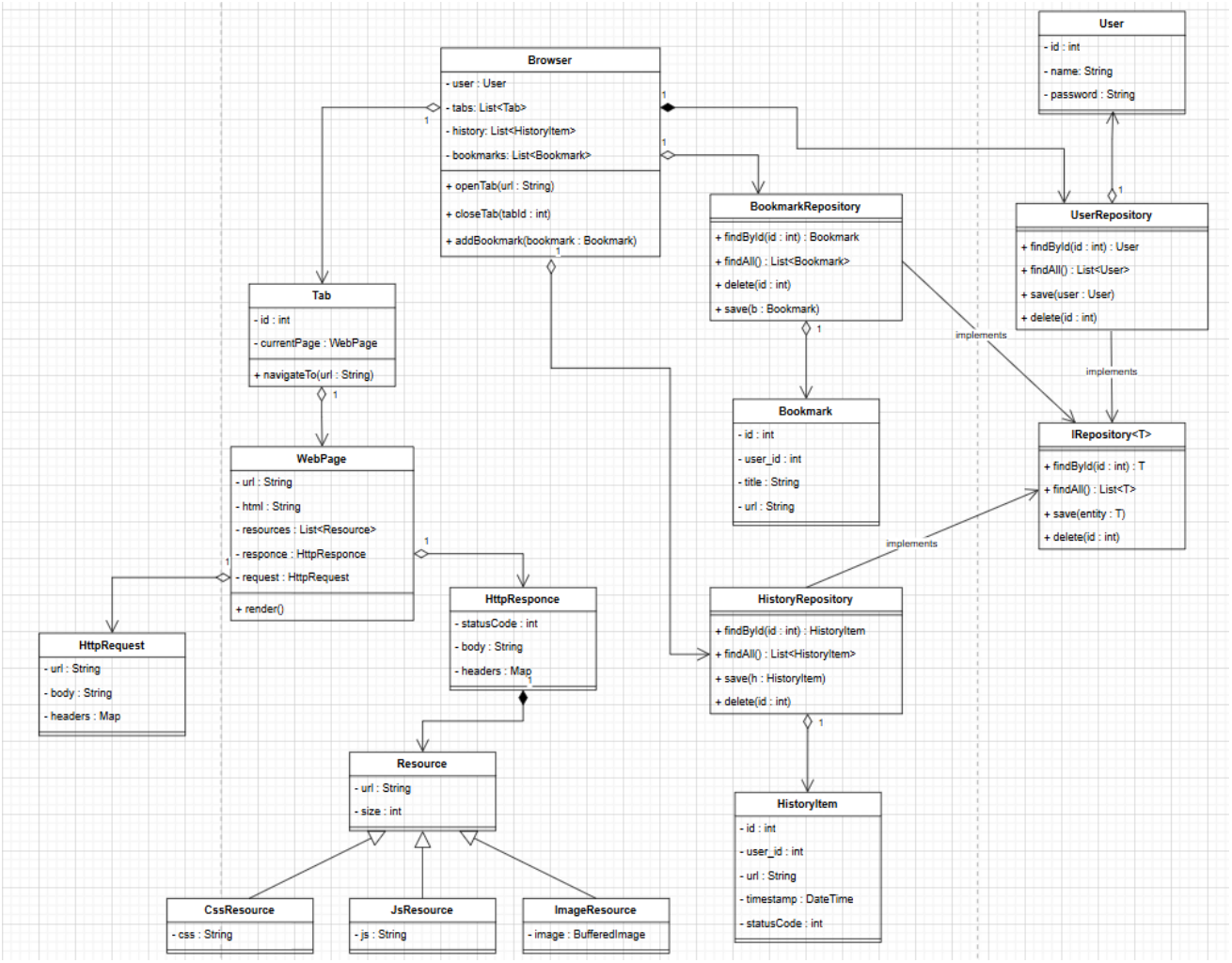

Рис. 2 – Діаграма класів

Опис зв’язків

1. Browser – Tab

- Тип відносин: Агрегація (*1 → *).
- Пояснення: Browser містить кілька вкладок (Tab). Вкладка існує тільки як частина браузера, але теоретично можна створити Tab незалежно.

2. Browser – Bookmark / HistoryItem

- Тип відносин: Агрегація (*1 → *).
- Пояснення: Browser зберігає список закладок і історію. Але закладки та історія можуть зберігатися і поза браузером (наприклад, у БД через репозиторій).

3. Browser – User

- Тип відносин: Асоціація (1 → 1).

- Пояснення: Browser належить певному користувачу. Це не композиція, бо User може існувати без Browser (наприклад, у системі є користувач, але він ще не відкрив браузер).

4. Tab – WebPage
   - Тип відносин: Композиція (1 → 1).
   - Пояснення: Tab завжди відображає одну поточну WebPage. Якщо Tab закривається — WebPage втрачає сенс.

5. WebPage – HttpRequest / HttpResponse
   - Тип відносин: Асоціація (1 → 1).
   - Пояснення: WebPage створюється в результаті HttpRequest і HttpResponse. Це просто залежність — WebPage може існувати як модель, навіть якщо реального HTTP-запиту не було (наприклад, offline rendering).

6. WebPage – Resource
   - Тип відносин: Агрегація (*1 → *).
   - Пояснення: WebPage містить набір ресурсів (CSS, JS, зображення). Ресурси можуть існувати і поза WebPage, але логічно вони пов’язані.

7. Resource – CssResource / JsResource / ImageResource
   - Тип відносин: Узагальнення (Generalization).
   - Пояснення: CssResource, JsResource і ImageResource є конкретними реалізаціями абстрактного класу Resource.

8. Repository<T> – BookmarkRepository / HistoryRepository / UserRepository
   - Тип відносин: Узагальнення (implements).
   - Пояснення: Кожен репозиторій реалізує інтерфейс Repository<T> для свого типу (Bookmark, HistoryItem, User).

9. BookmarkRepository – Bookmark / HistoryRepository – HistoryItem / UserRepository – User
   - Тип відносин: Асоціація (Repository → Entity).
   - Пояснення: Кожен репозиторій працює зі своїм типом сутності.

Проєктування БД

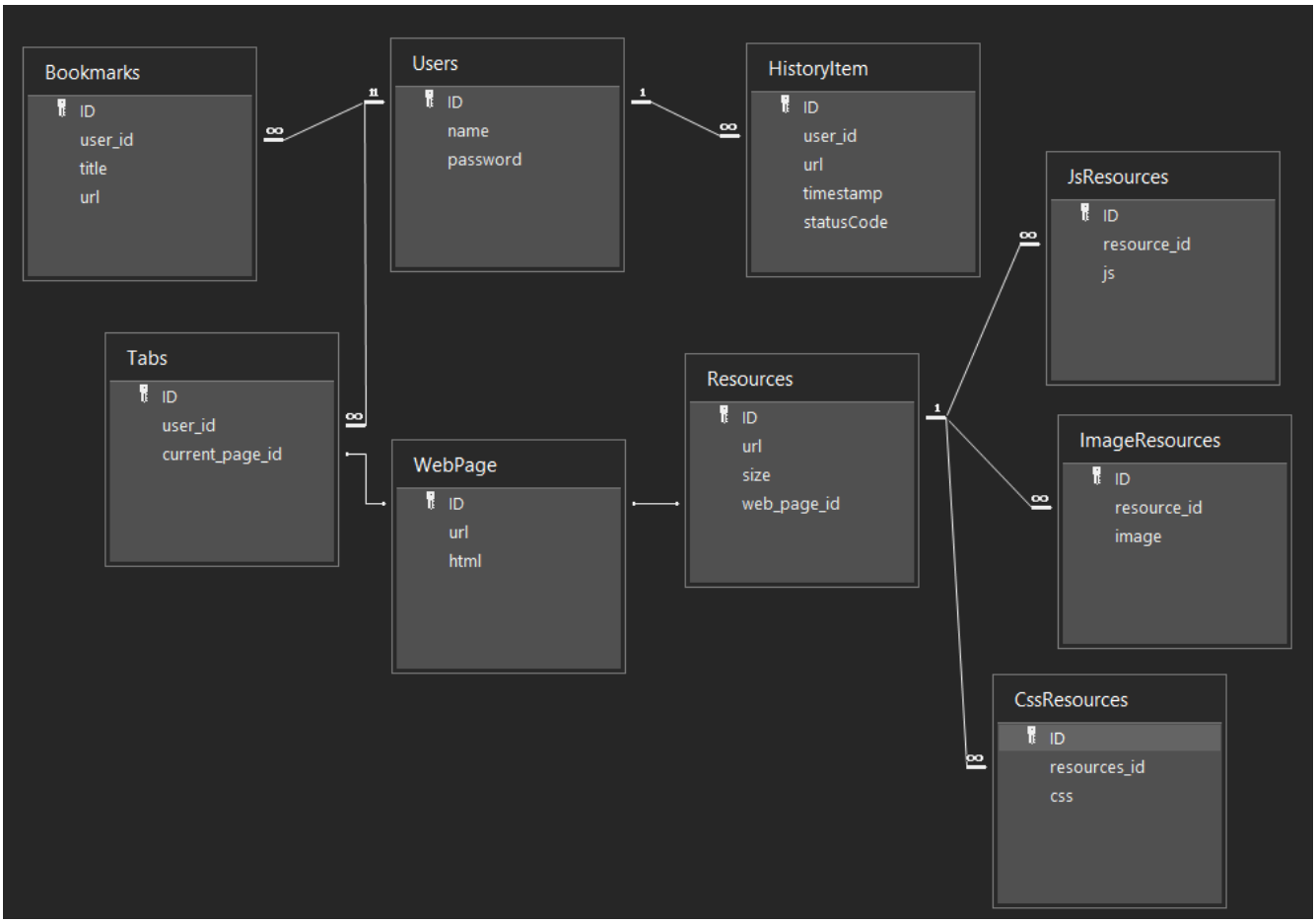

Рис. 3 – Структура БД

Опис БД

1. Таблиця Users
   - Містить інформацію про користувачів.
   - Поля:
     - ID — унікальний ідентифікатор користувача (PK).
     - name — ім’я користувача.
     - password — пароль користувача.
   - Один користувач може мати багато закладок та багато записів в історії.
2. Таблиця Bookmarks
   - Зберігає закладки користувачів.
   - Поля:
     - ID — унікальний ідентифікатор закладки (PK).
     - user_id — зовнішній ключ, що посилається на Users(ID).
     - title — назва закладки.
     - url — адреса сторінки.
   - Зв’язок: багато до одного з Users (один користувач може мати багато

закладок).

3. Таблиця HistoryItem
   - Містить історію перегляду користувача.
   - Поля:
     - ID — унікальний ідентифікатор запису (PK).
     - user_id — зовнішній ключ, що посилається на Users(ID).
     - url — відвідана адреса.
     - timestamp — час відвідування.
     - statusCode — код відповіді HTTP (наприклад, 200, 404).
   - Зв’язок: багато до одного з Users (один користувач має багато записів історії).

4. Таблиця Tabs
   - Зберігає інформацію про вкладки користувачів.
   - Поля:
     - ID — унікальний ідентифікатор вкладки (PK).
     - user_id — зовнішній ключ, що посилається на Users(ID).
     - current_page_id — зовнішній ключ, що посилається на WebPage(ID).
   - Зв’язок: багато до одного з Users (один користувач може мати багато вкладок).

5. Таблиця WebPage
   - Зберігає сторінки, відкриті у вкладках.
   - Поля:
     - ID — унікальний ідентифікатор сторінки (PK).
     - url — адреса сторінки.
     - html — HTML-код сторінки.
   - Зв’язок: багато до одного з Tabs (одна вкладка може містити багато сторінок).

6. Таблиця Resources
   - Зберігає ресурси, що належать сторінкам.
   - Поля:
     - ID — унікальний ідентифікатор ресурсу (PK).
     - url — адреса ресурсу.
     - size — розмір ресурсу.
     - web_page_id — зовнішній ключ, що посилається на WebPage(ID).
   - Зв’язок: багато до одного з WebPage (одна сторінка може мати багато

ресурсів).

7. Таблиця CssResources

- Зберігає ресурси типу CSS.
- Поля:
  - ID — унікальний ідентифікатор (PK), зовнішній ключ → Resources(ID).
  - css — CSS-код.
- Зв’язок: один до одного з Resources (кожен ресурс CSS відповідає одному запису у Resources).

8. Таблиця JsResources

- Зберігає ресурси типу JavaScript.
- Поля:
  - ID — унікальний ідентифікатор (PK), зовнішній ключ → Resources(ID).
  - js — JS-код.
- Зв’язок: один до одного з Resources (кожен ресурс JS відповідає одному запису у Resources).

9. Таблиця ImageResources

- Зберігає ресурси типу зображення.
- Поля:
  - ID — унікальний ідентифікатор (PK), зовнішній ключ → Resources(ID).
  - image — дані зображення.
- Зв’язок: один до одного з Resources (кожен ресурс зображення відповідає одному запису у Resources).

Вихідний код без реалізації на мові Java

```
class User {
    private int id;
    private String name;
    private String password;
}

class Browser {
```

```
    private User user;
    private List<Tab> tabs;
    private List<HistoryItem> history;
    private List<Bookmark> bookmarks;

    public void openTab(String url) { /* ... */ }
    public void closeTab(int tabId) { /* ... */ }
    public void addBookmark(Bookmark bookmark) { /* ... */ }
}

class Tab {
    private int id;
    private WebPage currentPage;

    public void navigateTo(String url) { /* ... */ }
}


class WebPage {
    private String url;
    private String html;
    private List<Resource> resources;
    private HttpResponse response;
    private HttpRequest request;

    public void render() { /* ... */ }
}

class HttpRequest {
    private String url;
    private String body;
    private Map<String, String> headers;
}

class HttpResponse {
```

```
    private int statusCode;
    private String body;
    private Map<String, String> headers;
}

abstract class Resource {
    protected String url;
    protected int size;
}

// Subclasses of Resource
class CssResource extends Resource {
    private String css;
}

class JsResource extends Resource {
    private String js;
}

class ImageResource extends Resource {
    private Object image; // Placeholder for BufferedImage
}

class Bookmark {
    private int id;
    private int userId;
    private String title;
    private String url;
}

class HistoryItem {
    private int id;
    private int userId;
    private String url;
    private Date timestamp;
```

```
    private int statusCode;
}

interface IRepository<T> {
    T findById(int id);
    List<T> findAll();
    void save(T entity);
    void delete(int id);
}

class UserRepository implements IRepository<User> {
    public User findById(int id) { return null; }
    public List<User> findAll() { return null; }
    public void save(User user) { }
    public void delete(int id) { }
}

class BookmarkRepository implements IRepository<Bookmark> {
    public Bookmark findById(int id) { return null; }
    public List<Bookmark> findAll() { return null; }
    public void save(Bookmark bookmark) { }
    public void delete(int id) { }
}

class HistoryRepository implements IRepository<HistoryItem> {
    public HistoryItem findById(int id) { return null; }
    public List<HistoryItem> findAll() { return null; }
    public void save(HistoryItem historyItem) { }
    public void delete(int id) { }
}
```

Контрольні запитання

4. UML — уніфікована мова моделювання, стандарт для опису та візуалізації програмних систем.
5. Діаграма класів UML — схема, що показує класи, їх атрибути, методи та

зв’язки між ними.

6. Канонічні діаграми UML — це базові діаграми, без яких неможливе повноцінне моделювання: діаграма класів, варіантів використання, послідовностей, станів, діяльності, компонентів, розгортання.
7. Діаграма варіантів використання — показує, як користувачі (актори) взаємодіють із системою через функціональні можливості (use cases).
8. Варіант використання — це опис певної функціональності системи з точки зору користувача.
9. Відношення на діаграмі використання: асоціація (актор–use case), include, extend, generalization.
10. Сценарій — послідовність кроків взаємодії користувача з системою у межах одного варіанта використання.
11. Діаграма класів — структурна діаграма, що описує класи системи та їх зв’язки.
12. Зв’язки між класами: асоціація, агрегація, композиція, наслідування (generalization), реалізація (implements), залежність.
13. Композиція vs агрегація: композиція означає повну залежність частини від цілого (життєвий цикл спільний), агрегація — слабший зв’язок, частина може існувати окремо.
14. На діаграмі класів: композиція позначається чорним ромбом, агрегація — білим ромбом.
15. Нормальні форми — правила структурування таблиць БД для усунення надлишковості та аномалій.
16. Фізична модель БД — конкретна реалізація (таблиці, індекси, типи даних). Логічна модель — концептуальна структура даних (сутності, зв’язки, атрибути).
17. Таблиці БД ↔ програмні класи: кожна таблиця часто відповідає класу, рядок таблиці — об’єкту, стовпець — атрибуту класу.

Висновки

Під час виконання лабораторної роботи, ми навчилися основам проектування, обрали зручну систему побудови UML-діаграм та навчилися будувати діаграми варіантів використання для системи що проєктується, розробляти сценарії варіантів використання та будувати діаграми класів предметної області.