Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота № 2

Технології розроблення програмного забезпечення

«Основи проектування»

20. Mind-mapping software

Виконав:
студент групи ІА–33:
Хитрова НА

Перевірив:
Мягкий МЮ

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.

2.1. 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.

2.2. 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 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.

З погляду методології ООАП (об'єктно-орієнтованого аналізу та проєктування) досить повна модель складної системи є певною кількістю взаємопов'язаних уявлень (views), кожне з яких відображає аспект поведінки або структури системи. У цьому найзагальнішими уявленнями

складної системи прийнято вважати статичне і динамічне, які у своє чергу можуть поділятися інші більш приватні.

Принцип ієрархічної побудови моделей складних систем передбачає розгляд процес побудови моделей на різних рівнях абстрагування або деталізації в рамках фіксованих уявлень.

Рівень представлення (layer) – спосіб організації та розгляду моделі на одному рівні абстракції, що представляє горизонтальний зріз архітектури моделі, тоді як розбиття представляє її вертикальний зріз.

При цьому вихідна або початкова модель складної системи має найбільш загальне уявлення та відноситься до концептуального рівня. Така модель, що отримала назву концептуальної, будується на початковому етапі проєктування і може не містити багатьох деталей та аспектів системи, що моделюється. Наступні моделі конкретизують концептуальну модель, доповнюючи її уявленнями логічного та фізичного рівня.

Загалом процес ООАП можна розглядати як послідовний перехід від розробки найбільш загальних моделей та уявлень концептуального рівня до більш приватних і детальних уявлень логічного та фізичного рівня. У цьому кожному етапі ООАП дані моделі послідовно доповнюються дедалі більше деталей, що дозволяє їм адекватно відбивати різні аспекти конкретної реалізації складної системи.

В рамках мови UML уявлення про модель складної системи фіксуються у вигляді спеціальних графічних конструкцій, що отримали назву діаграм.

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.

Хід роботи

Mind-mapping software – це система для створення інтелект-карт (mind maps). Система ментальних карт дозволяє користувачам візуалізувати думки. Основна сутність – це "Карта" (Map), яка складається з "Вузлів" (Nodes) та "Зв'язків" (Links). Користувачі можуть створювати, редагувати, зберігати, експортувати карти, ділитися ними, працювати з вузлами, темами, стилями та ін.

На основі цього аналізу було виділено ключових акторів, які взаємодіють із системою:

Гість (Guest): Неавторизований користувач. Не може використовувати систему, має зареєструватися перш ніж почати роботу.

Зареєстрований користувач (User): Основний актор. Може створювати, зберігати, редагувати карти.

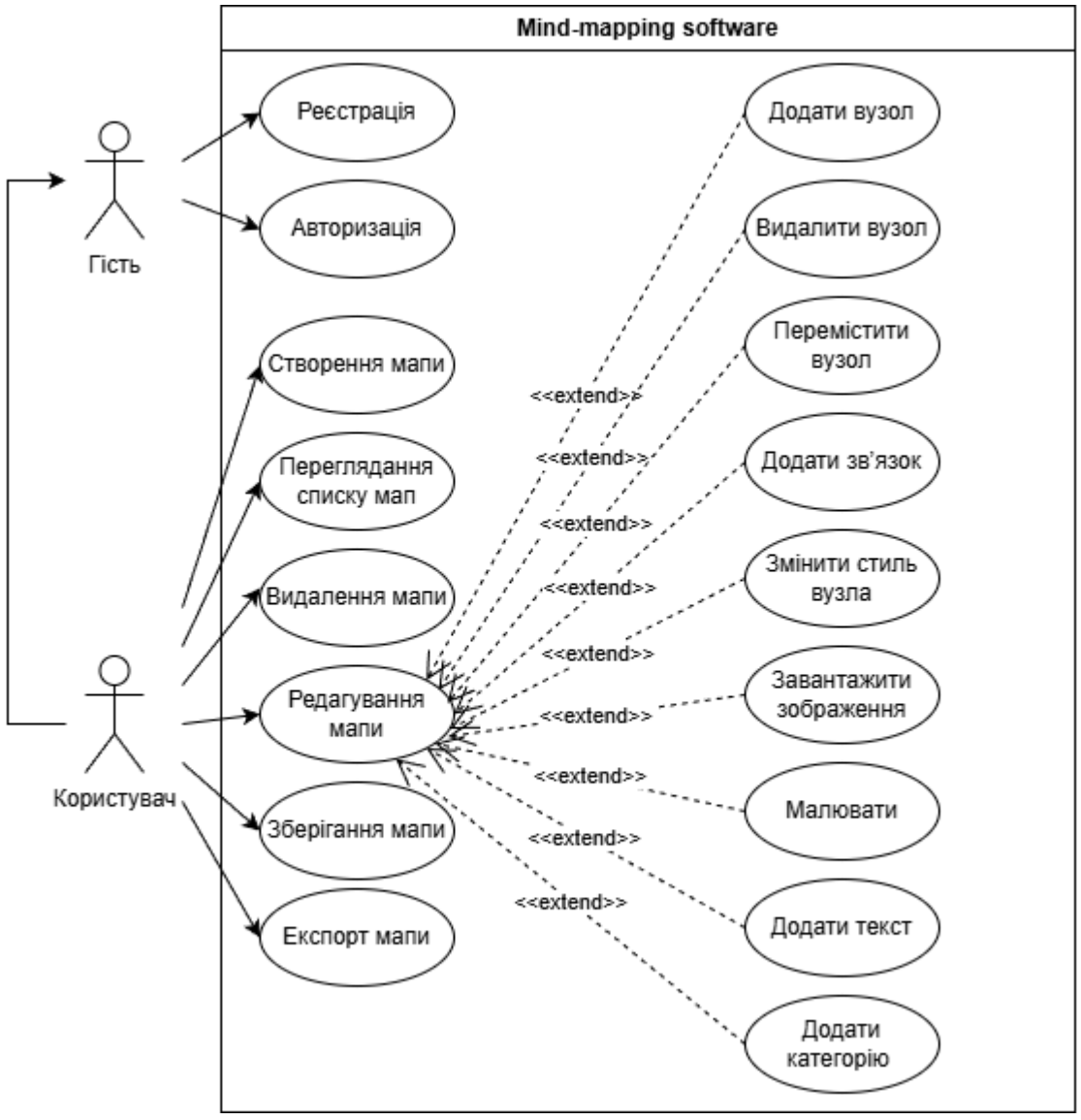


Рис 1. Use-case діаграма Mind-mapping software

Представлена діаграма моделює функціональну поведінку програмного забезпечення для створення ментальних карт (Mind-mapping software). Взаємодія із системою передбачена для двох типів акторів: «Гостя» та «Користувача». Між ними встановлено відношення

узагальнення, що вказує на те, що зареєстрований користувач потенційно володіє базовими можливостями, але отримує доступ до повного функціоналу системи лише після проходження процедур реєстрації та авторизації.

Основний робочий процес користувача охоплює повний цикл управління ментальними картами: створення нових проектів, перегляд списку власних карт, збереження прогресу, видалення мап та їх експорт у зовнішні формати.

Центральним елементом діаграми є прецедент «Редагування мапи». Оскільки цей процес є комплексним та варіативним, він деталізований за допомогою відношення розширення До базового процесу редагування приєднано низку специфічних дій: «Додати вузол», «Видалити вузол», «Перемістити вузол», «Додати зв’язок», «Змінити стиль вузла», «Завантажити зображення», «Малювати», «Додати текст» та «Додати категорію», надалі функціонал може бути розширений.

Проектування діаграми класів предметної області:

Ця діаграма відображає статичну структуру нашої системи. Вона показує, з яких класів складається програма, які дані вони зберігають та як вони пов'язані між собою. Діаграма розділена на два логічні пакети: Entities (моделі даних) та Repositories (рівень доступу до даних).

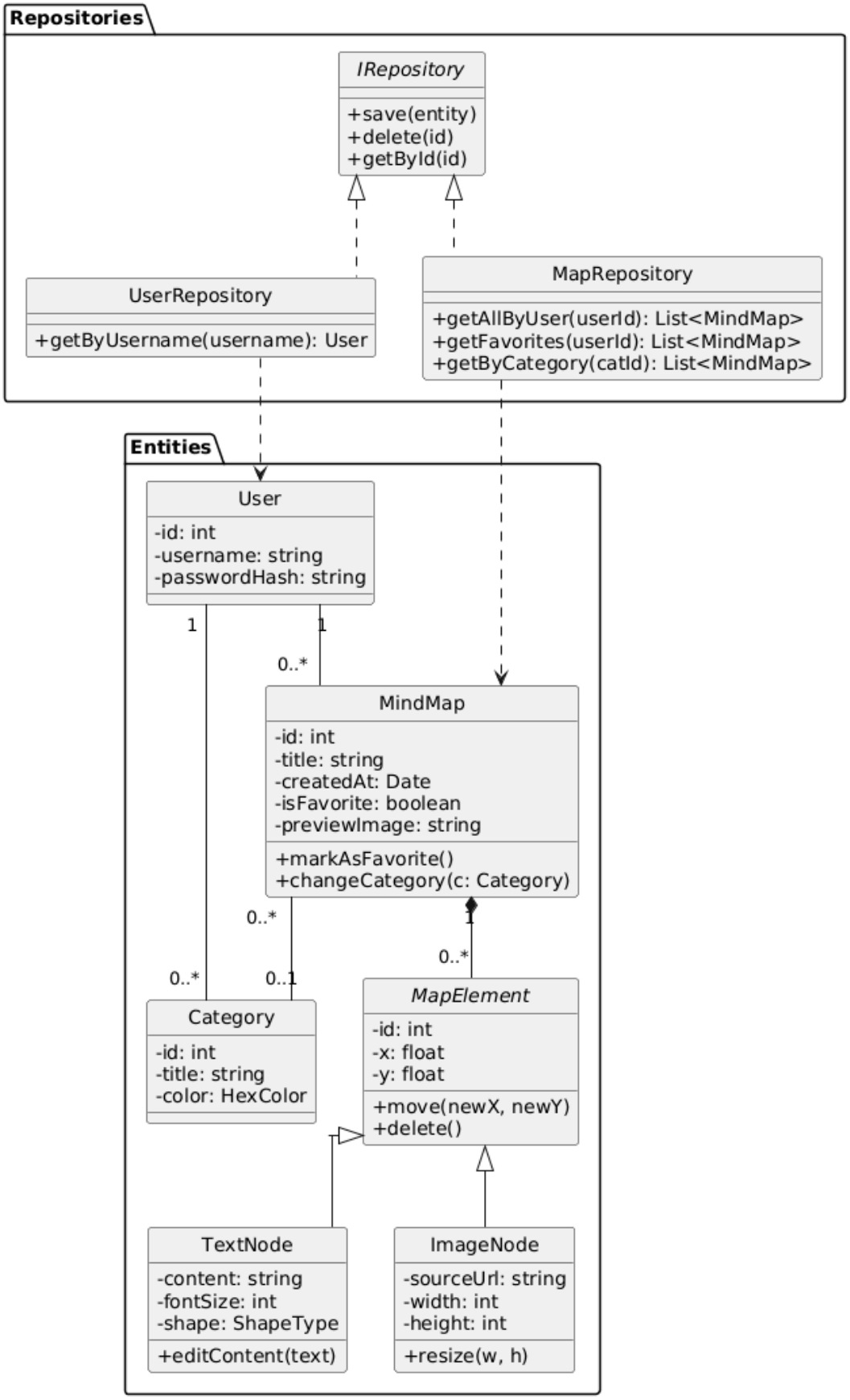


Рис 2. Діаграма класів предметної області

1. Пакет Entities (Сутності)

Це основні об'єкти, з якими працює наша бізнес-логіка.

User (Користувач): Центральна фігура системи. Ми зберігаємо лише мінімально необхідні дані: id, username (логін) та passwordHash (зашифрований пароль). Метод updateProfile() дозволяє користувачеві змінювати свої дані.

MindMap (Ментальна мапа): Це головний робочий простір. Окрім назви та дати створення, клас містить поле isFavorite. Це дозволяє реалізувати функцію "Вибране" — користувач може позначати важливі мапи, щоб мати до них швидкий доступ (метод markAsFavorite()).

Category (Категорія): Допоміжний клас для організації простору. Користувач може створювати власні категорії (наприклад, "Навчання", "Робота") і присвоювати їм кольори, щоб сортувати свої мапи.

2. Пакет Repositories (Робота з БД)

Відповідно до завдання, система реалізує шаблон проєктування Repository. Це означає, що самі класи даних (User, MindMap) не вміють зберігати себе в базу даних. Цим займаються окремі класи-посередники.

IRepository (Інтерфейс): Визначає загальний контракт для всіх репозиторіїв системи: кожен з них повинен вміти зберігати (save), видаляти (delete) та шукати по ID (getById).

UserRepository: Відповідає за таблицю користувачів. Має унікальний метод getByUsername() для авторизації.

MapRepository: Відповідає за завантаження мап. Він містить специфічні методи вибірки даних:

getAllByUser() – показати всі мапи конкретної людини.

getFavorites() – показати тільки "Вибрані" мапи.

getByCategory() – фільтрація мап по папках.

3. Пояснення зв'язків

На діаграмі використано кілька типів зв'язків, які визначають логіку взаємодії об'єктів:

User – MindMap (Асоціація 1 до 0..*): Це означає, що один користувач може володіти багатьма мапами (або не мати жодної), але кожна мапа належить суворо одному власнику.

MindMap – Category (Асоціація 0..* до 0..1): Мапа може бути прив'язана до однієї категорії (або бути без категорії), але в одній категорії може знаходитися багато мап.

MindMap – MapElement: Вузли та картинки не можуть існувати окремо від мапи. Якщо ми видаляємо об'єкт MindMap, система автоматично знищує всі пов'язані з нею об'єкти MapElement. Це забезпечує цілісність даних.

MapElement – TextNode (Узагальнення/Спадкування): Стрілка з пустим трикутником показує, що TextNode є різновидом MapElement і переймає всі його властивості.

Розробка сценаріїв варіантів використання:

Сценарій 1: Створення нової ментальної мапи

Цей сценарій описує перехід користувача від панелі керування до робочого простору.

ID: UC-01

Назва: Створення нової мапи.

Актор: Зареєстрований користувач.

Передумова: Користувач авторизований і знаходиться на головній сторінці (Dashboard).

Основний потік подій:

1. Користувач натискає кнопку «Створити нову мапу».
2. Система відображає діалогове вікно для введення параметрів нової мапи.
3. Користувач вводить назву мапи (наприклад, "План курсової").
4. Користувач підтверджує створення натисканням кнопки «Створити».
5. Система перенаправляє користувача на екран редагування мапи.

Альтернативний потік (А1): Користувач не ввів назву.

3а. Користувач залишає поле назви порожнім і натискає «Створити».

3б. Система підсвічує поле червоним та виводить повідомлення «Назва мапи обов'язкова».

3в. Сценарій повертається до кроку 3.

Постумова: Нова мапа збережена в системі, відкрито режим редагування.

Сценарій 2: Завантаження зображення на мапу

Цей сценарій деталізує один із варіантів редагування.

ID: UC-02

Назва: Додавання зображення на робочу область.

Актор: Зареєстрований користувач.

Передумова: Користувач знаходиться в режимі редагування мапи.

Основний потік подій:

1. Користувач обирає інструмент «Додати зображення» на панелі інструментів.
2. Система відкриває стандартне вікно провідника файлів операційної системи.
3. Користувач обирає графічний файл (формати JPG, PNG) на своєму пристрої.
4. Система завантажує зображення на сервер.
5. Система створює новий вузол типу "ImageNode" і відображає його на робочій області поруч із центром або в місці кліку миші.
6. Користувач перетягує зображення у потрібне місце.

Альтернативний потік (А1): Помилка формату файлу.

4а. Користувач обирає файл недопустимого формату (наприклад, .pdf або .exe).

4б. Система виводить повідомлення про помилку: «Невірний формат файлу. Будь ласка, оберіть JPG або PNG».

4в. Сценарій завершується, зображення не додається.

Постумова: Зображення відображається на мапі та збережено як частина структури мапи.

Сценарій 3: Експорт мапи

Цей сценарій описує отримання результату роботи поза системою.

ID: UC-03

Назва: Експорт мапи у графічний файл.

Актор: Зареєстрований користувач.

Передумова: Користувач відкрив мапу, завершив редагування і бажає отримати локальну копію.

Основний потік подій:

1. Користувач натискає кнопку «Експорт» у меню мапи.
2. Система пропонує обрати формат файлу (PNG або JPG).
3. Користувач обирає формат PNG і натискає «Завантажити».

4. Система генерує зображення, що охоплює всі вузли мапи.
5. Система ініціює завантаження файлу браузером користувача.
6. Система виводить повідомлення «Експорт успішний».

Альтернативний потік (А1): Порожня мапа.

2а. Система виявляє, що на мапі немає жодного вузла.

2б. Система виводить попередження: «Неможливо експортувати порожню мапу».

Постумова: Графічний файл із зображенням поточної версії мапи збережено на пристрої користувача.

На основі спроєктованої діаграми класів предметної області розробити основні класи та структуру бази даних системи:

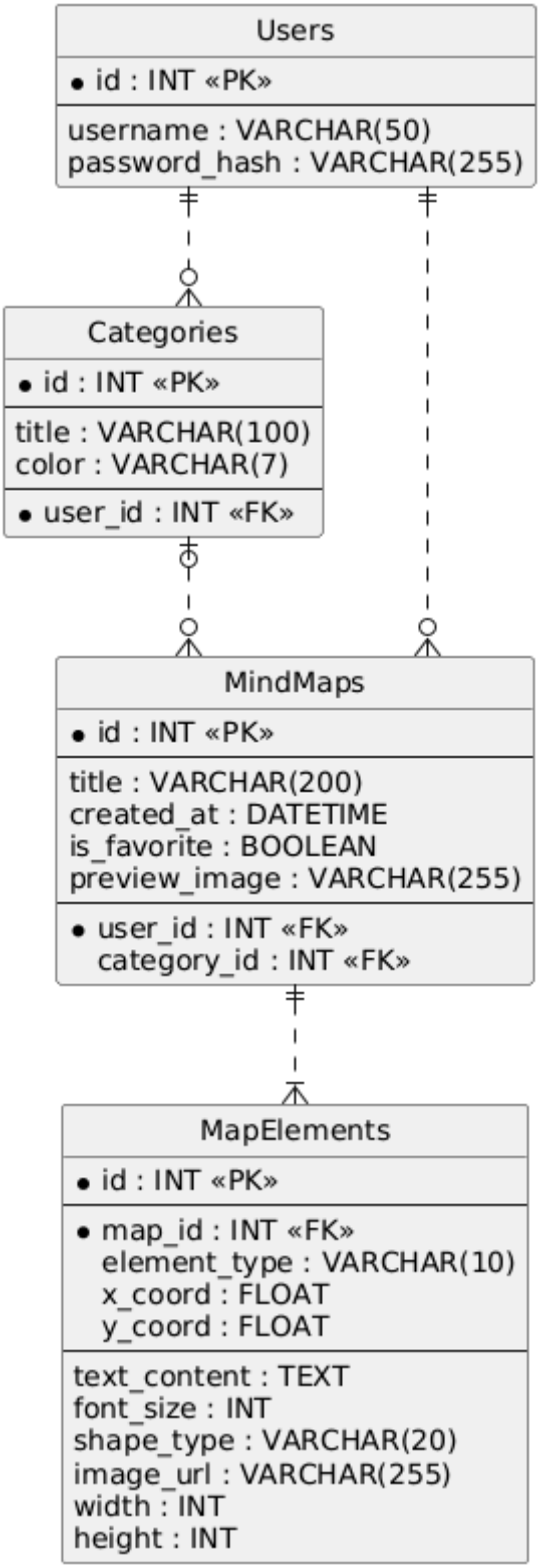


Рис. 3. Схема даних

Спроектована база даних побудована навколо таблиці користувачів, яка є центральною ланкою системи безпеки та ідентифікації власників контенту. У таблиці Users зберігаються лише необхідні облікові дані, такі

як унікальний ідентифікатор, логін та хеш пароля, що забезпечує базовий захист доступу. Кожен зареєстрований користувач отримує можливість створювати власну структуру для організації роботи, що реалізовано через таблицю Categories. Ця сутність пов’язана з користувачем через зовнішній ключ, дозволяючи зберігати назви папок та їх кольорове кодування для візуального розрізнення проєктів.

Основною робочою сутністю системи є таблиця MindMaps, яка містить метадані про кожну створену ментальну карту, включаючи її назву, дату створення, посилання на зображення попереднього перегляду та спеціальний прапорець, що визначає, чи додана карта до списку вибраних. Ця таблиця має два ключові зв’язки: обов’язковий зв’язок з таблицею користувачів, що визначає власника, та опціональний зв’язок з категоріями, що дозволяє карті або належати до певної групи, або існувати поза нею.

Найнижчий рівень ієрархії даних представлений таблицею MapElements, де зберігається весь вміст карт. Для оптимізації структури використано підхід збереження всіх типів об’єктів у одній таблиці. Це означає, що і текстові блоки, і зображення знаходяться в одній сутності, яка містить спільні поля координат розташування на полотні та спеціальне поле типу елемента. Залежно від цього типу система зчитує або текстові параметри, такі як зміст і шрифт, або параметри зображення, такі як розмір та посилання на файл, ігноруючи непотрібні поля. Такий підхід забезпечує цілісність даних, адже при видаленні карти автоматично видаляються всі пов’язані з нею елементи завдяки каскадному зв’язку.

Структура Бази Даних (SQL):

```
-- 1. Таблиця Користувачів
CREATE TABLE Users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL
);

-- 2. Таблиця Категорій
CREATE TABLE Categories (
    id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(100) NOT NULL,
```

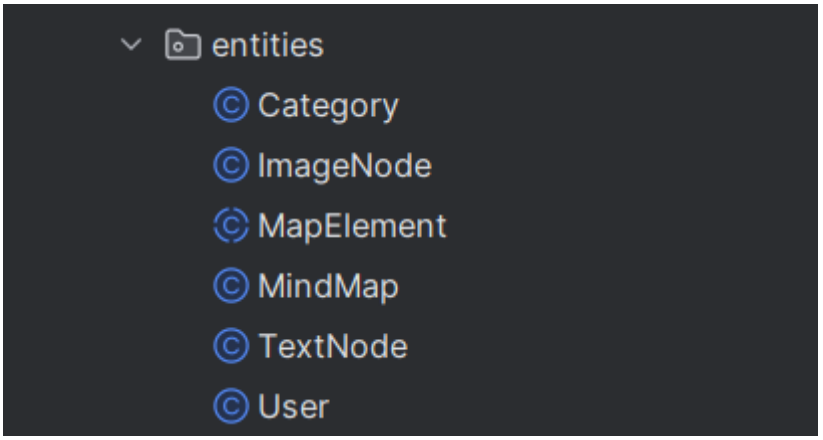
```
    color VARCHAR(7),
    user_id INT NOT NULL,
        FOREIGN KEY (user_id) REFERENCES Users(id) ON DELETE
CASCADE
);

-- 3. Таблиця Мап
CREATE TABLE MindMaps (
    id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(200) NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    is_favorite BOOLEAN DEFAULT FALSE,
    preview_image VARCHAR(255),
    user_id INT NOT NULL,
    category_id INT, -- Може бути NULL, якщо категорія не обрана
        FOREIGN KEY (user_id) REFERENCES Users(id) ON DELETE
CASCADE,
     FOREIGN KEY (category_id) REFERENCES Categories(id) ON DELETE
SET NULL
);

-- 4. Таблиця Елементів (Всі типи елементів в одній таблиці)
CREATE TABLE MapElements (
    id INT PRIMARY KEY AUTO_INCREMENT,
    map_id INT NOT NULL,
    x_coord FLOAT NOT NULL,
    y_coord FLOAT NOT NULL,

    -- Колонка-дискримінатор (визначає тип: 'TEXT' або 'IMAGE')
    element_type VARCHAR(10) NOT NULL,

    -- Поля для TextNode
    text_content TEXT,
    font_size INT,
    shape_type VARCHAR(20),

    -- Поля для ImageNode
```

```
    image_url VARCHAR(255),
    width INT,
    height INT,

    FOREIGN KEY (map_id) REFERENCES MindMaps(id) ON DELETE CASCADE
);
```

Лістинг коду реалізованих класів системи:

https://github.com/ch1fir/trpz-lab-2

Сутності, що відповідають діаграмі класів:



User.java: (модель користувача)

```java
package com.example.mindmap.entities;

public class User {  21 usages
    private int id;  3 usages
    private String username;  4 usages
    private String passwordHash;  4 usages

    public User() {  no usages
    }

    public User(int id, String username, String passwordHash) {  1 usage
        this.id = id;
        this.username = username;
        this.passwordHash = passwordHash;
    }

    public void updateProfile(String newUsername, String newPasswordHash) {  no usages
        this.username = newUsername;
        this.passwordHash = newPasswordHash;
    }

    // гетери/сетери
    public int getId() {  2 usages
        return id;
    }

    public void setId(int id) {  1 usage
        this.id = id;
    }
```

```
30
31      public String getUsername() {  2 usages
32          return username;
33      }
34
35      public void setUsername(String username) {  no usages
36          this.username = username;
37      }
38
39      public String getPasswordHash() {  no usages
40          return passwordHash;
41      }
42
43      public void setPasswordHash(String passwordHash) {  no usages
44          this.passwordHash = passwordHash;
45      }
46  }
```

MindMap.java: (ментальна карта)

```
1   package com.example.mindmap.entities;                                    ⚠ 12

2
3   import java.time.LocalDateTime;
4   import java.util.ArrayList;
5   import java.util.List;
6
7   public class MindMap {  17 usages
8       private int id;  3 usages
9       private String title;  3 usages
10      private LocalDateTime createdAt;  3 usages
11      private boolean favorite;  3 usages
12      private String previewImage;  2 usages
13
14      private User owner;  3 usages
15      private Category category;  2 usages
16      private List<MapElement> elements = new ArrayList<>();  3 usages
17
18      public MindMap() {  no usages
19          this.createdAt = LocalDateTime.now();
20      }
21
22      public MindMap(int id, String title, User owner) {  1 usage
23          this.id = id;
24          this.title = title;
25          this.owner = owner;
26          this.createdAt = LocalDateTime.now();
27      }
28
```

```java
    public void markAsFavorite() {  1 usage
        this.favorite = true;
    }

    public void unmarkFavorite() {  no usages
        this.favorite = false;
    }

    public void changeCategory(Category category) {  no usages
        this.category = category;
    }

    public void addElement(MapElement element) {  2 usages
        elements.add(element);
    }

    public void removeElement(MapElement element) {  no usages
        elements.remove(element);
    }

    // гетери/сетери
    public int getId() {  2 usages
        return id;
    }

```

```java
    public void setId(int id) {  1 usage
        this.id = id;
    }

    public String getTitle() {  1 usage
        return title;
    }

    public void setTitle(String title) {  no usages
        this.title = title;
    }

    public LocalDateTime getCreatedAt() {  no usages
        return createdAt;
    }

    public boolean isFavorite() {  1 usage
        return favorite;
    }

    public String getPreviewImage() {  no usages
        return previewImage;
    }

    public void setPreviewImage(String previewImage) {  no usages
        this.previewImage = previewImage;
    }
```

```java
    public User getOwner() {  2 usages
        return owner;
    }

    public void setOwner(User owner) {  no usages
        this.owner = owner;
    }

    public Category getCategory() {  no usages
        return category;
    }

    public List<MapElement> getElements() {  1 usage
        return elements;
    }
}
```

Category.java: (категорія карти)

```java
package com.example.mindmap.entities;                                                    ⚠ 10

public class Category {  3 usages
    private int id;  3 usages
    private String title;  3 usages
    private String color; // HEX  3 usages
    private User user;    // власник категорії  3 usages

    public Category() {  no usages
    }

    public Category(int id, String title, String color, User user) {  no usages
        this.id = id;
        this.title = title;
        this.color = color;
        this.user = user;
    }

    // гетери/сетери
    public int getId() {  no usages
        return id;
    }

    public void setId(int id) {  no usages
        this.id = id;
    }

    public String getTitle() {  no usages
        return title;
```

```java
    }

    public void setTitle(String title) {  no usages
        this.title = title;
    }

    public String getColor() {  no usages
        return color;
    }

    public void setColor(String color) {  no usages
        this.color = color;
    }

    public User getUser() {  no usages
        return user;
    }

    public void setUser(User user) {  no usages
        this.user = user;
    }
}
```

MapElement.java: (базовий абстрактний елемент карти)

```java
package com.example.mindmap.entities;                                                    10

public abstract class MapElement {  6 usages  2 inheritors
    private int id;  3 usages
    private float x;  4 usages
    private float y;  4 usages
    private MindMap map;  3 usages

    public MapElement() {  2 usages
    }

    public MapElement(int id, float x, float y, MindMap map) {  2 usages
        this.id = id;
        this.x = x;
        this.y = y;
        this.map = map;
    }

    public void move(float newX, float newY) {  no usages
        this.x = newX;
        this.y = newY;
    }

    public abstract String getType();  no usages  2 implementations

    // гетери/сетери
    public int getId() {  no usages
        return id;
    }
```

```
    public void setId(int id) {  no usages
        this.id = id;
    }

    public float getX() {  no usages
        return x;
    }

    public void setX(float x) {  no usages
        this.x = x;
    }

    public float getY() {  no usages
        return y;
    }

    public void setY(float y) {  no usages
        this.y = y;
    }

    public MindMap getMap() {  no usages
        return map;
    }

    public void setMap(MindMap map) {  no usages
        this.map = map;
    }
}
```

TextNode.java: (текстовий елемент)

```
package com.example.mindmap.entities;                                    8

public class TextNode extends MapElement {  2 usages
    private String textContent;  4 usages
    private int fontSize;  3 usages
    private String shapeType;  3 usages

    public TextNode() {  no usages
    }

    public TextNode(int id, float x, float y, MindMap map,  1 usage
                    String textContent, int fontSize, String shapeType) {
        super(id, x, y, map);
        this.textContent = textContent;
        this.fontSize = fontSize;
        this.shapeType = shapeType;
    }

    public void editContent(String newText) {  no usages
        this.textContent = newText;
    }

    @Override  no usages
    public String getType() {
        return "TEXT";
    }

    // гетери/сетери
    public String getTextContent() {  no usages
        return textContent;
```

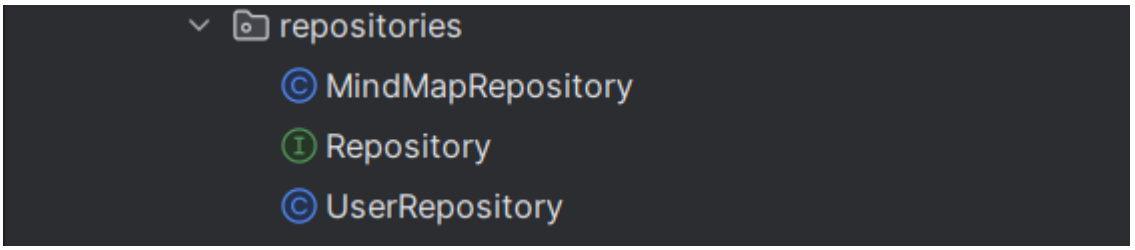
```
    }

    public void setTextContent(String textContent) {  no usages
        this.textContent = textContent;
    }

    public int getFontSize() {  no usages
        return fontSize;
    }

    public void setFontSize(int fontSize) {  no usages
        this.fontSize = fontSize;
    }

    public String getShapeType() {  no usages
        return shapeType;
    }

    public void setShapeType(String shapeType) {  no usages
        this.shapeType = shapeType;
    }
}
```

ImageNode: (графічний елемент)

```
package com.example.mindmap.entities;                                                    8

public class ImageNode extends MapElement {  2 usages
    private String imageUrl;  3 usages
    private int width;  4 usages
    private int height;  4 usages

    public ImageNode() {  no usages
    }

    public ImageNode(int id, float x, float y, MindMap map,  1 usage
                     String imageUrl, int width, int height) {
        super(id, x, y, map);
        this.imageUrl = imageUrl;
        this.width = width;
        this.height = height;
    }

    public void resize(int width, int height) {  no usages
        this.width = width;
        this.height = height;
    }

    @Override  no usages
    public String getType() {
        return "IMAGE";
    }

    // гетери/сетери
    public String getImageUrl() {  no usages
```

```java
    // гетери/сетери
    public String getImageUrl() {  no usages
        return imageUrl;
    }

    public void setImageUrl(String imageUrl) {  no usages
        this.imageUrl = imageUrl;
    }

    public int getWidth() {  no usages
        return width;
    }

    public void setWidth(int width) {  no usages
        this.width = width;
    }

    public int getHeight() {  no usages
        return height;
    }

    public void setHeight(int height) {  no usages
        this.height = height;
    }
}
```

Патерн Repository у простій in-memory версії:



Repository.java: (базовий generic interface)

```java
package com.example.mindmap.repositories;                                    4

import java.util.List;
import java.util.Optional;

public interface Repository<T, ID> {  2 usages  2 implementations
    T save(T entity);  2 usages  2 implementations
    void delete(ID id);  no usages  2 implementations
    Optional<T> getById(ID id);  no usages  2 implementations
    List<T> getAll();  no usages  2 implementations
}
```

UserRepository.java: (зберігає користувачів)

```java
package com.example.mindmap.repositories;

import com.example.mindmap.entities.User;

import java.util.*;

public class UserRepository implements Repository<User, Integer> {  2 usages

    private final Map<Integer, User> storage = new HashMap<>();  5 usages
    private int nextId = 1;  1 usage

    @Override  2 usages
    public User save(User entity) {
        if (entity.getId() == 0) {
            entity.setId(nextId++);
        }
        storage.put(entity.getId(), entity);
        return entity;
    }

    @Override  no usages
    public void delete(Integer id) {
        storage.remove(id);
    }

    @Override  no usages
    public Optional<User> getById(Integer id) {
        return Optional.ofNullable(storage.get(id));
    }

    @Override  no usages
    public List<User> getAll() {
        return new ArrayList<>(storage.values());
    }

    public Optional<User> getByUsername(String username) {  no usages
        return storage.values().stream()
                .filter( User u -> u.getUsername().equals(username))
                .findFirst();
    }
}
```

MindMapRepository.java: (зберігає карти)

```java
package com.example.mindmap.repositories;

import com.example.mindmap.entities.User;

import java.util.*;

public class UserRepository implements Repository<User, Integer> {  2 usages

    private final Map<Integer, User> storage = new HashMap<>();  5 usages
    private int nextId = 1;  1 usage

    @Override  2 usages
    public User save(User entity) {
        if (entity.getId() == 0) {
            entity.setId(nextId++);
        }
        storage.put(entity.getId(), entity);
        return entity;
    }

    @Override  no usages
    public void delete(Integer id) {
        storage.remove(id);
    }

    @Override  no usages
    public Optional<User> getById(Integer id) {
        return Optional.ofNullable(storage.get(id));
    }

    @Override  no usages
    public List<User> getAll() {
        return new ArrayList<>(storage.values());
    }

    public Optional<User> getByUsername(String username) {  no usages
        return storage.values().stream()
                .filter( User u -> u.getUsername().equals(username))
                .findFirst();
    }
}
```

Демонстрація бізнес-логіки:

App.java × User.java × Category.java .gitignore MindMap.java MapElement.java TextNode.ja

```java
package com.example.mindmap;

import com.example.mindmap.entities.*;
import com.example.mindmap.repositories.*;

public class App {
    public static void main(String[] args) {
        UserRepository userRepo = new UserRepository();
        MindMapRepository mapRepo = new MindMapRepository();

        // Створюємо користувача
        User user = new User( id: 0,  username: "student",  passwordHash: "hash123");
        userRepo.save(user);

        // Створюємо мапу
        MindMap map = new MindMap( id: 0,  title: "План курсової", user);
        mapRepo.save(map);

        // Додаємо елементи
        TextNode title = new TextNode( id: 0,  x: 100,  y: 100, map,
                textContent: "Mind-mapping software",  fontSize: 18,  shapeType: "RECTANGLE");
        map.addElement(title);

        ImageNode img = new ImageNode( id: 0,  x: 200,  y: 150, map,
                imageUrl: "/images/diagram.png",  width: 320,  height: 240);
        map.addElement(img);

        // Позначаємо як обрану
        map.markAsFavorite();

        System.out.println("Користувач: " + user.getUsername());
        System.out.println("Мапа: " + map.getTitle());
        System.out.println("Елементів на мапі: " + map.getElements().size());
    }
}
```

2.3. Питання до лабораторної роботи

1. Що таке UML?

UML – це універсальна мова моделювання, яка дозволяє описувати структуру та поведінку програмних систем за допомогою діаграм. Вона допомагає розробникам візуально планувати архітектуру та логіку роботи програм.

2. Що таке діаграма класів UML?

Діаграма класів – це схема, яка показує основні класи системи, їх атрибути, методи та зв’язки між ними. Вона відображає статичну структуру програмної системи.

3. Які діаграми UML називають канонічними?

Канонічними вважають ті діаграми, що входять до стандарту UML і найчастіше застосовуються на практиці: діаграма класів, діаграма варіантів використання, діаграма послідовностей, діаграма діяльності, діаграма станів тощо.

4. Що таке діаграма варіантів використання?

Це діаграма, що описує взаємодію користувачів (акторів) із системою через різні варіанти використання. Вона дозволяє зрозуміти, яку функціональність система повинна забезпечити.

5. Що таке варіант використання?

Варіант використання – це опис конкретної функції, яку користувач може виконати в системі. Він визначає цінну для користувача дію та реакцію системи на цю дію.

6. Які відношення можуть бути відображені на діаграмі використання?

На діаграмі використання можна зобразити відношення include (включення), extend (розширення), а також зв’язок асоціації між акторами та варіантами використання.

7. Що таке сценарій?

Сценарій – це детальний покроковий опис виконання варіанта використання: як саме користувач взаємодіє із системою і як система відповідає.

8. Що таке діаграма класів?

Діаграма класів – це модель, що показує сутності (класи) предметної області та відносини між ними. Вона допомагає зрозуміти структуру системи ще до написання коду.

9. Які зв’язки між класами ви знаєте?

Основні зв’язки: Асоціація, Агрегація, Композиція, Наслідування, Залежність, Реалізація

10. Чим відрізняється композиція від агрегації?

Композиція – це «жорсткий» зв’язок, коли один клас повністю володіє іншим і відповідає за його існування. Агрегація – «слабше» відношення, коли частина може існувати окремо від цілого.

11. Чим відрізняється зв'язки типу агрегації від зв'язків композиції на діаграмах класів?

Агрегація позначається порожнім ромбом, а композиція – заповненим ромбом. Це показує ступінь «належності» об’єктів один одному.

12. Що являють собою нормальні форми баз даних?

Нормальні форми – це правила організації даних у таблицях, які допомагають усунути дублювання інформації та забезпечують цілісність даних. Вони визначають, як правильно структурувати таблиці.

13. Що таке фізична модель бази даних? Логічна?

Логічна модель – це опис сутностей, атрибутів і зв’язків між ними без прив’язки до конкретної СУБД.

Фізична модель – це реалізація цієї логіки у вигляді таблиць, індексів, типів даних у конкретній СУБД.

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

Кожна таблиця зазвичай відповідає окремому класу у програмі. Рядки таблиці – це об’єкти класу, а стовпці – його поля. Зв’язки між таблицями відображаються як асоціації між класами.

Висновок: У ході виконання лабораторної роботи було засвоєно практичні навички об'єктно-орієнтованого проектування програмного забезпечення на прикладі системи для створення ментальних карт (Mind-mapping software). Було проведено детальний аналіз предметної області, в результаті якого виділено основні ролі (Гість, Користувач) та сценарії їхньої взаємодії з системою, що було візуалізовано на діаграмі варіантів використання. Особливу увагу приділено проектуванню функціоналу редагування мапи. Отримані діаграми, сценарії та структура бази даних є готовою основою для подальшої програмної реалізації проєкту.