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 № 3

Технології розроблення програмного забезпечення

«Основи проектування розгортання»

20. Mind-mapping software

Виконав:
студент групи ІА–33:
Хитрова НА

Перевірив:
Мягкий МЮ

Київ 2025

Тема: Основи проектування розгортання.

Мета: Навчитися проектувати 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.

3.1. Завдання

• Ознайомитись з короткими теоретичними відомостями.

• Проаналізувати діаграми створені в попередній лабораторній роботі а також тему системи та спроєктувати діаграму розгортання використання відповідно до обраної теми лабораторного циклу.

• Розробити діаграму компонентів для проектованої системи.

• Розробити діаграму розгортання для проектованої системи.

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

• На основі проектованих 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).

• 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.

3.2. Теоретичні відомості

Діаграми розгортання представляють фізичне розташування системи, показуючи, на якому фізичному обладнанні запускається та чи інша складова програмного забезпечення [3].

Головними елементами діаграми є вузли, пов'язані інформаційними шляхами. Вузол (node) – це те, що може містити програмне забезпечення. Вузли бувають двох типів. Пристрій (device) – це фізичне обладнання: комп'ютер або пристрій, пов'язаний із системою. Середовище виконання (execution environment) – це програмне забезпечення, яке саме може включати інше програмне забезпечення, наприклад операційну систему або процес-контейнер (наприклад, веб сервер).

Між вузлами можуть стояти зв'язки, які зазвичай зображують у вигляді прямої лінії. Як і на інших діаграмах, у зв'язків можуть бути атрибути множинності (для показання, наприклад, підключення 2х і більше клієнтів до одного сервера) і назва. У назві, як правило, міститься спосіб зв'язку між двома вузлами – це може бути назва протоколу (HTTP, IPC) або технологія, що використовується для забезпечення взаємодії вузлів (.NET Remoting, WCF).

Вузли можуть містити артефакти (artifacts), які є фізичним уособленням програмного забезпечення; зазвичай це файли. Такими файлами можуть бути виконувані файли (такі як файли .exe, двійкові файли, файли DLL, файли JAR, збірки або сценарії) або файли даних, конфігураційні файли, HTML-документи тощо. Перелік артефактів усередині вузла вказує на те, що на даному вузлі артефакт розгортається в систему, що запускається.

Артефакти можна зображати у вигляді прямокутників класів або перераховувати їхні імена всередині вузла. Якщо ви показуєте ці елементи у вигляді прямокутників класів, то можете додати значок документа або ключове слово «artifact». Можна супроводжувати вузли або артефакти значеннями у вигляді міток, щоб вказати різну цікаву інформацію про вузол, наприклад постачальника, операційну систему, місце розташування – загалом, усе, що спаде вам на думку.

Діаграма компонентів UML є представленням проектованої системи, розбитої на окремі модулі [3]. Залежно від способу поділу на модулі розрізняють три види діаграм компонентів:

- логічні;
- фізичні;
- виконувані.

Коли використовують логічне розбиття на компоненти, то у такому разі проектовану систему віртуально уявляють як набір самостійних, автономних модулів (компонентів), що взаємодіють між собою.

Діаграма послідовностей (Sequence Diagram) – це один із типів діаграм у моделюванні UML (Unified Modeling Language), який використовується для моделювання взаємодії між об'єктами системи у певній послідовності часу. Вона відображає, як об'єкти обмінюються повідомленнями, показуючи порядок і логіку виконання операцій.

Хід роботи

Діаграму розгортання для проектованої системи:

Мета діаграми розгортання показати фізичну архітектуру системи, на якому обладнанні працює програма і як вона зв’язується з даними.

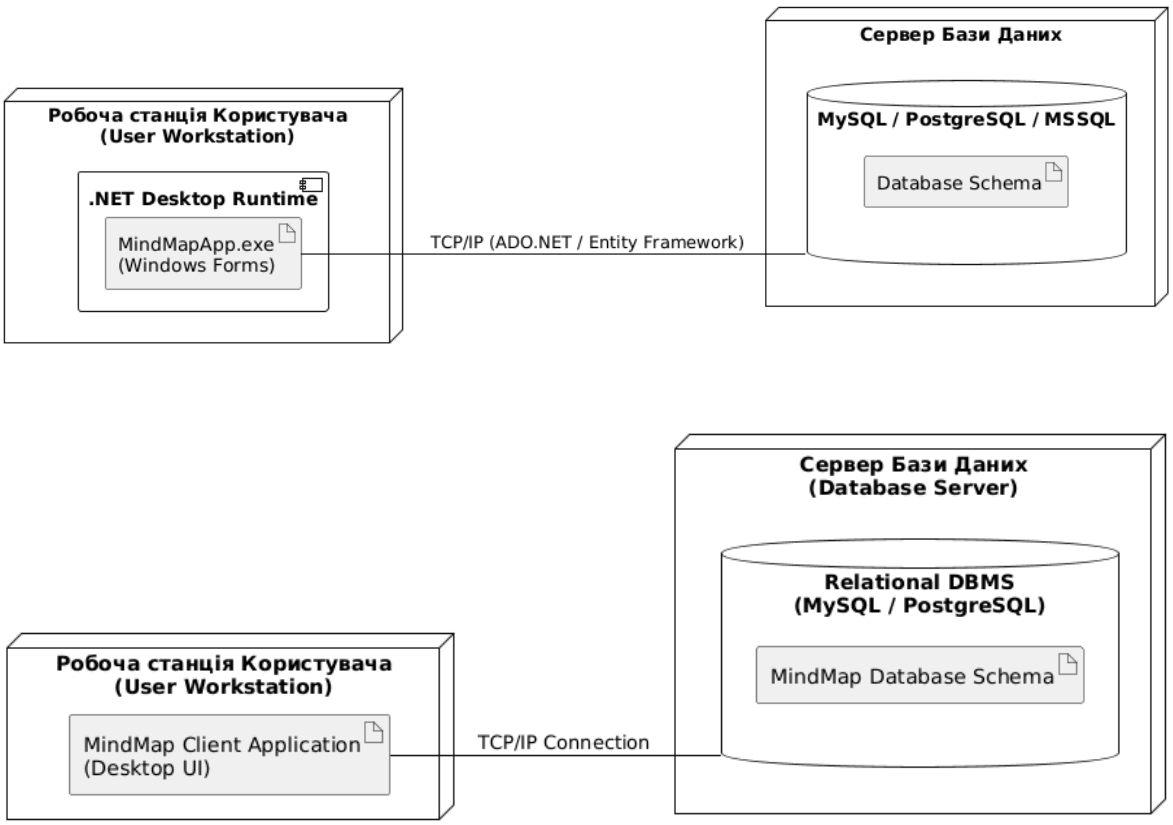


Рис 1. Діаграма розгортання

Діаграма розгортання демонструє фізичну архітектуру системи та показує, на яких апаратних і програмних вузлах розміщуються окремі компоненти застосунку. Система побудована за клієнт-серверною моделлю, де робоча станція користувача виконує роль клієнта, а сервер бази даних забезпечує централізоване зберігання даних.

Першим основним вузлом є Робоча станція користувача (User Workstation). Це персональний комп’ютер або ноутбук, на якому запускається клієнтський застосунок – MindMap Client Application. Даний модуль відповідає за взаємодію з користувачем, відображення інтерфейсу, роботу з ментальною картою, обробку подій (додавання вузлів, редагування, переміщення елементів тощо) та виконання бізнес-логіки на

стороні клієнта. Саме на робочій станції виконується основна частина функціонального навантаження програми, включаючи рендеринг графічних компонентів та управління локальним станом карти до моменту збереження.

Другим важливим вузлом є Сервер бази даних (Database Server). На ньому розгорнута система керування реляційними базами даних (наприклад, MySQL або PostgreSQL). У межах цього вузла розташована логічна схема бази даних MindMap Database Schema, яка містить усі сутності, необхідні для роботи системи: користувачів, категорії, ментальні карти та елементи карт. База даних забезпечує надійне довгострокове зберігання інформації, її цілісність та доступність.

Комунікація між клієнтом і сервером здійснюється через стандартне мережеве з'єднання TCP/IP. Клієнтський застосунок надсилає SQL-запити на сервер бази даних і отримує відповіді у вигляді результатів вибірки. Такий підхід дає змогу організувати повний цикл роботи з даними: від введення інформації користувачем у графічному інтерфейсі — до її збереження у віддаленій базі та подальшого відображення при повторному відкритті мапи.

Таким чином, діаграма розгортання описує логічно зрозумілу та легко масштабовану інфраструктуру системи: клієнтська частина відповідає за взаємодію та візуалізацію, а серверна — за структуроване зберігання даних. Такий варіант архітектури забезпечує простоту розгортання, чітке розмежування відповідальності та можливість подальшого розширення функціональності.

Діаграма компонентів для проектованої системи:

Діаграма компонентів відображає внутрішню архітектуру десктопного застосунку Mind-mapping software, який побудований за трирівневою (трьохшаровою) моделлю. Кожен шар має чітко визначену відповідальність, що підвищує гнучкість, масштабованість та легкість супроводу системи.

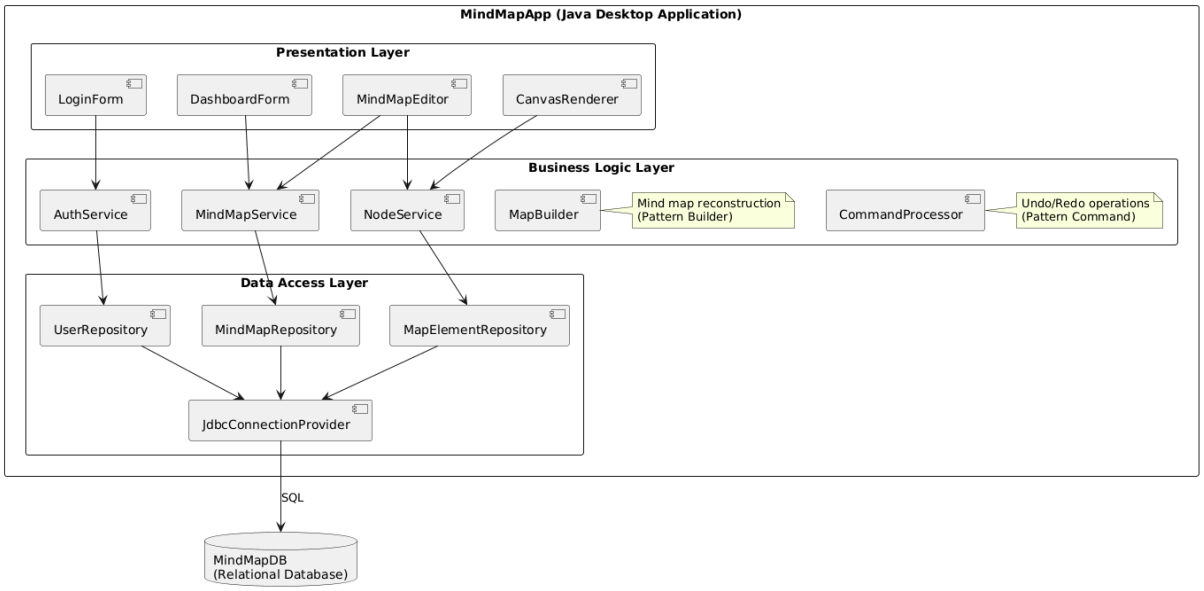


Рис 2. Діаграма компонентів

Шар представлення (Presentation Layer) містить графічні компоненти LoginForm, DashboardForm та MindMapEditor. Саме тут користувач взаємодіє із системою: виконує авторизацію, переглядає список ментальних карт і редагує структуру обраної карти. Компонент CanvasRenderer відповідає за відображення вузлів та зображень на полотні.

Шар бізнес-логіки (Business Logic Layer) інкапсулює основні алгоритми програми та використовує шаблони проектування. Сервіси (AuthService, MindMapService, NodeService) реалізують сценарії роботи користувача, не залежачи від деталей збереження даних. Компонент CommandProcessor реалізує механізм Undo/Redo на основі патерну Command, а MapBuilder використовує патерн Builder для реконструкції складних ментальних карт під час завантаження.

Шар доступу до даних (Data Access Layer) відповідає за роботу з базою даних. Репозиторії реалізують шаблон Repository і забезпечують стандартні CRUD-операції. Компонент JdbcConnectionProvider створює з’єднання з базою даних та передає SQL-запити.

Усі репозиторії взаємодіють лише через клас JdbcConnectionProvider, який, у свою чергу, обмінюється даними з реляційною базою MindMapDB. Така архітектура забезпечує повне розділення UI, бізнес-логіки та інфраструктурного коду, що відповідає сучасним принципам побудови клієнтських застосунків.

Діаграми послідовностей для сценаріїв прописаних в попередній лабораторній роботі:

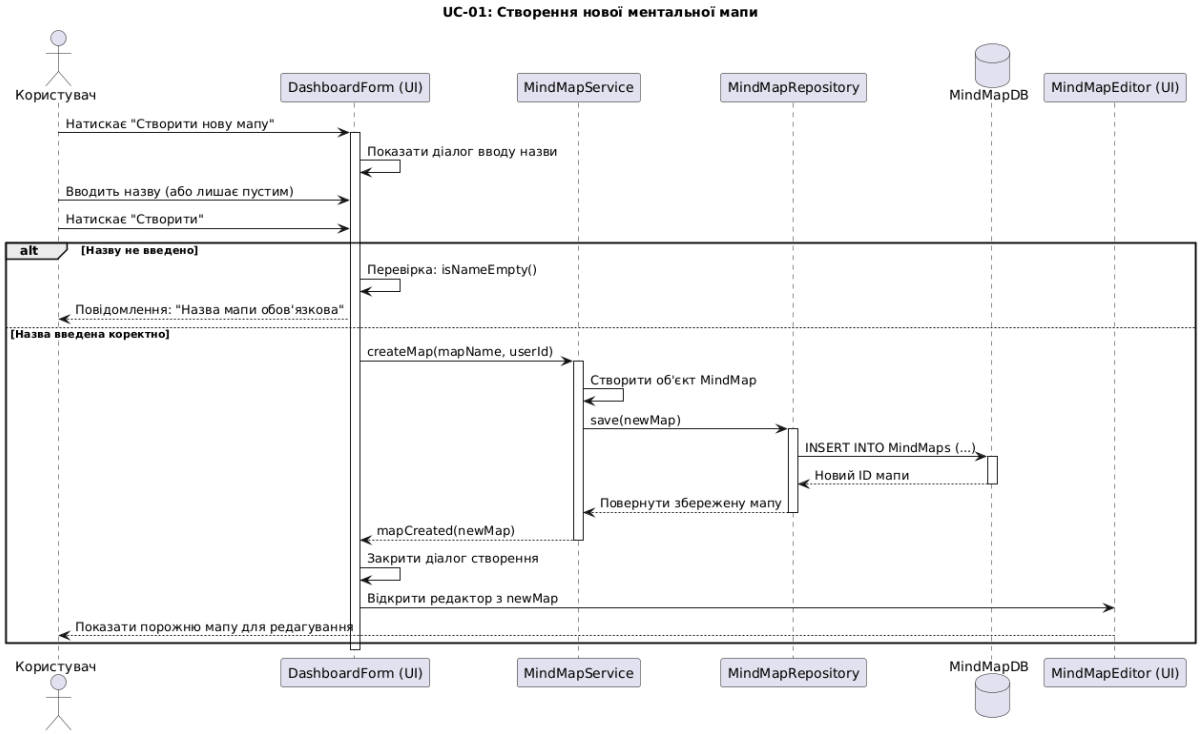


Рис 3.1 Діаграма послідовностей для сценарія UC-01

Діаграма послідовності для сценарію створення нової ментальної мапи демонструє взаємодію між користувачем, інтерфейсом програми та бізнес-логікою під час ініціалізації нового проєкту. Сценарій починається з того, що зареєстрований користувач перебуває на головній панелі (DashboardForm) та натискає кнопку «Створити нову мапу». Інтерфейс відображає діалогове вікно для введення назви, після чого користувач підтверджує створення.

Далі система виконує перевірку: якщо поле назви порожнє, інтерфейс виводить повідомлення про помилку і пропонує повторити введення. Це відповідає альтернативному потоку A1. У випадку, коли назву введено коректно, DashboardForm передає її до компонента бізнес-логіки MindMapService. Сервіс створює новий об'єкт мапи (MindMap), після чого звертається до MindMapRepository для збереження даних у базі.

Репозиторій виконує SQL-операцію INSERT через підключення до бази даних і повертає ідентифікатор створеного запису. Отримавши відповідь, MindMapService передає успішний результат назад до інтерфейсу.

Після цього головна форма закриває діалог та відкриває редактор мап (MindMapEditor), куди передає щойно створений об’єкт. У результаті користувачу показується порожня мапа, готова до редагування.

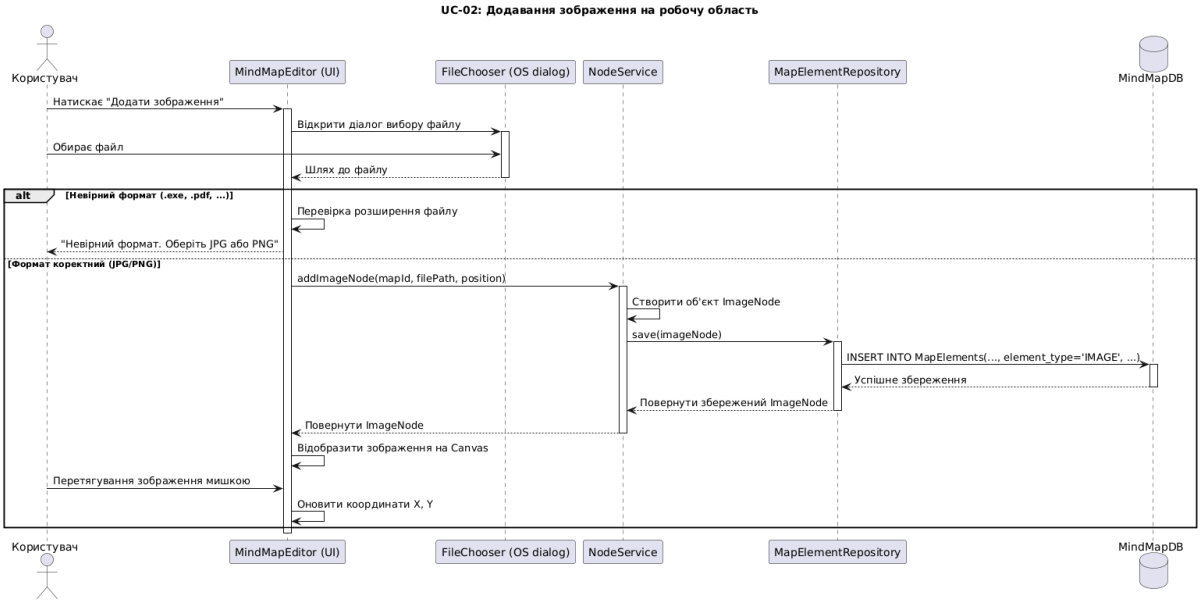


Рис 3.2 Діаграма послідовностей для сценарія UC-02

У цьому сценарії зображено детальний процес додавання графічного файлу в поточну ментальну мапу. Користувач перебуває у редакторі (MindMapEditor) та натискає інструмент “Додати зображення”. Інтерфейс викликає стандартне системне вікно вибору файлу, де користувач обирає локальне зображення.

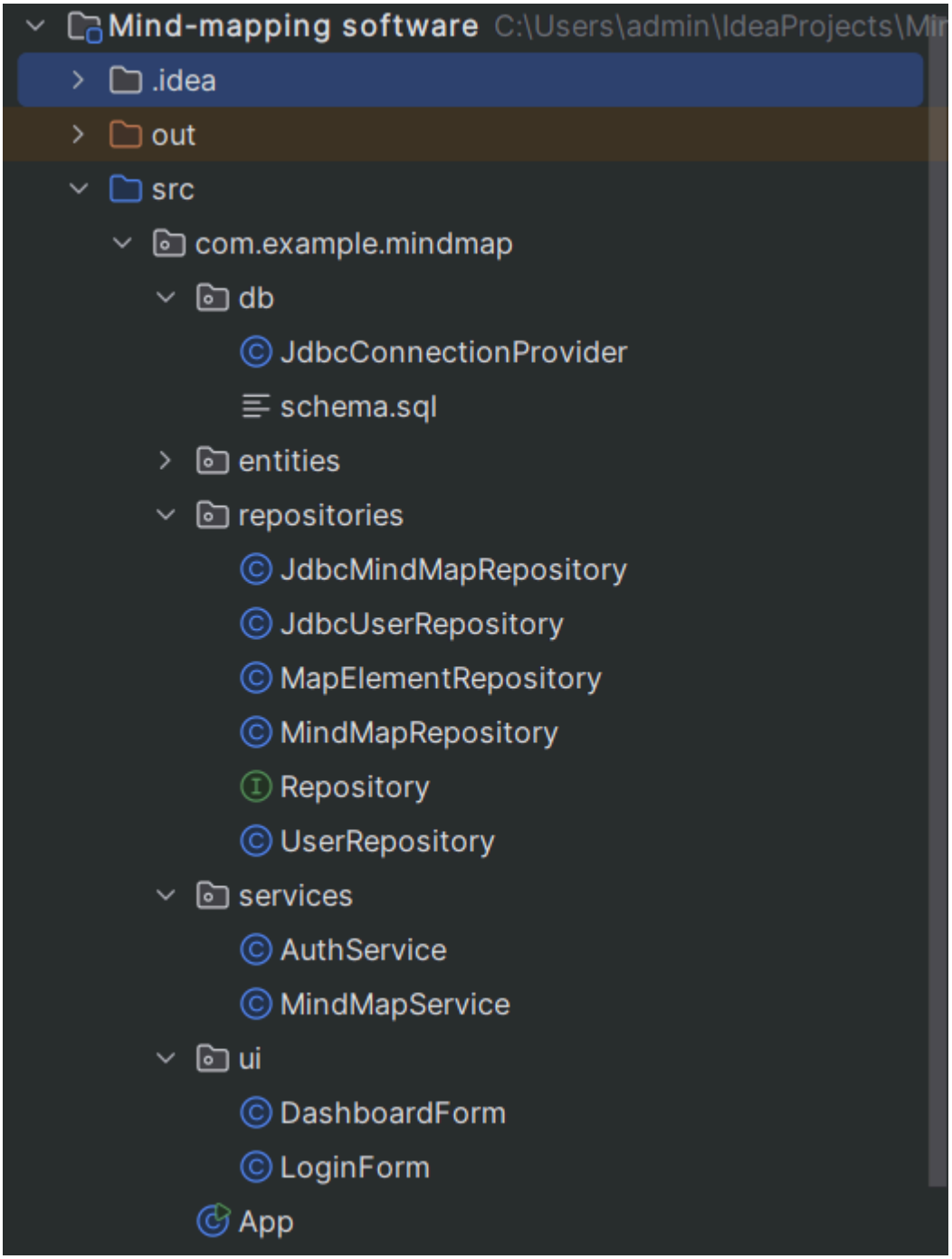
Після отримання шляху до файлу система перевіряє формат. Якщо користувач вибрав непідтримуваний тип файлу (наприклад .exe, .pdf), редактор відображає попередження і операція завершується без створення вузла. Це описано як альтернативний потік.

У разі, якщо формат коректний (PNG або JPG), MindMapEditor передає шлях до файлу сервісу NodeService, який відповідає за створення елементів карти. Сервіс формує новий об’єкт ImageNode з початковими координатами. Далі NodeService звертається до репозиторію MapElementRepository, який зберігає новий елемент у базі даних.

Після успішного збереження репозиторій повертає інформацію про створений вузол, а сервіс надсилає її назад інтерфейсу. MindMapEditor відображає зображення на полотні (Canvas) та дає користувачу можливість перетягувати його мишкою. Після переміщення система оновлює координати X і Y в локальному стані.

У підсумку зображення стає частиною структури мапи та синхронізується з базою даних, забезпечуючи його збереження під час наступних відкриттів проекту.

Лістинг вихідного код системи, який було додано в цій лабораторній роботі:
https://github.com/ch1fir/trpz-lab-3



App.java:

```java
package com.example.mindmap;

import com.example.mindmap.db.JdbcConnectionProvider;
import com.example.mindmap.repositories.JdbcMindMapRepository;
import com.example.mindmap.repositories.JdbcUserRepository;
import com.example.mindmap.services.AuthService;
import com.example.mindmap.services.MindMapService;
import com.example.mindmap.ui.LoginForm;

import javax.swing.*;

public class App {
    public static void main(String[] args) {

        JdbcConnectionProvider connectionProvider = new JdbcConnectionProvider();
        JdbcUserRepository userRepo = new JdbcUserRepository(connectionProvider);
        JdbcMindMapRepository mapRepo = new JdbcMindMapRepository(connectionProvider);

        AuthService authService = new AuthService(userRepo);
        MindMapService mindMapService = new MindMapService(mapRepo);

        SwingUtilities.invokeLater(() -> {
            new LoginForm(authService, mindMapService).setVisible(true);
        });
    }
}
```

JdbcUserRepository.java:

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

import com.example.mindmap.db.JdbcConnectionProvider;
import com.example.mindmap.entities.User;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

public class JdbcUserRepository implements Repository<User, Integer> {  6 usages

    private final JdbcConnectionProvider connectionProvider;  7 usages

    public JdbcUserRepository(JdbcConnectionProvider connectionProvider) {  1 usage
        this.connectionProvider = connectionProvider;
    }

    @Override  2 usages
    public User save(User entity) {
        if (entity.getId() == 0) {
            // INSERT
            String sql = "INSERT INTO Users(username, password_hash) VALUES(?, ?)";
            try (Connection conn = connectionProvider.getConnection();
                 PreparedStatement ps = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {

                ps.setString( parameterIndex: 1, entity.getUsername());
                ps.setString( parameterIndex: 2, entity.getPasswordHash());
```

```
                ps.executeUpdate();

                try (ResultSet rs = ps.getGeneratedKeys()) {
                    if (rs.next()) {
                        entity.setId(rs.getInt( columnIndex: 1));
                    }
                }
            } catch (SQLException e) {
                throw new RuntimeException(e);
            }
        } else {
            // UPDATE
            String sql = "UPDATE Users SET username = ?, password_hash = ? WHERE id = ?";
            try (Connection conn = connectionProvider.getConnection();
                 PreparedStatement ps = conn.prepareStatement(sql)) {

                ps.setString( parameterIndex: 1, entity.getUsername());
                ps.setString( parameterIndex: 2, entity.getPasswordHash());
                ps.setInt( parameterIndex: 3, entity.getId());
                ps.executeUpdate();
            } catch (SQLException e) {
                throw new RuntimeException(e);
            }
        }
        return entity;
    }
```

```
    @Override  no usages
    public void delete(Integer id) {
        String sql = "DELETE FROM Users WHERE id = ?";
        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt( parameterIndex: 1, id);
            ps.executeUpdate();
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }

    @Override  no usages
    public Optional<User> getById(Integer id) {
        String sql = "SELECT id, username, password_hash FROM Users WHERE id = ?";
        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setInt( parameterIndex: 1, id);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
                    User u = new User(
                            rs.getInt( columnLabel: "id"),
                            rs.getString( columnLabel: "username"),
                            rs.getString( columnLabel: "password_hash")
                    );
```

```
                    return Optional.of(u);
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return Optional.empty();
    }

    @Override  no usages
    public List<User> getAll() {
        String sql = "SELECT id, username, password_hash FROM Users";
        List<User> users = new ArrayList<>();
        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql);
             ResultSet rs = ps.executeQuery()) {

            while (rs.next()) {
                users.add(new User(
                        rs.getInt( columnLabel: "id"),
                        rs.getString( columnLabel: "username"),
                        rs.getString( columnLabel: "password_hash")
                ));
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
```

```
        }
        return users;
    }

    public Optional<User> getByUsername(String username) {  1 usage
        String sql = "SELECT id, username, password_hash FROM Users WHERE username = ?";
        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setString( parameterIndex: 1, username);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
                    return Optional.of(new User(
                            rs.getInt( columnLabel: "id"),
                            rs.getString( columnLabel: "username"),
                            rs.getString( columnLabel: "password_hash")
                    ));
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return Optional.empty();
    }
}
```

JdbcMindMapRepository.java:

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

import com.example.mindmap.db.JdbcConnectionProvider;
import com.example.mindmap.entities.MindMap;
import com.example.mindmap.entities.User;

import java.sql.*;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

public class JdbcMindMapRepository implements Repository<MindMap, Integer> {  6 usages

    private final JdbcConnectionProvider connectionProvider;  6 usages

    public JdbcMindMapRepository(JdbcConnectionProvider connectionProvider) {  1 usage
        this.connectionProvider = connectionProvider;
    }

    @Override  2 usages
    public MindMap save(MindMap entity) {
        if (entity.getId() == 0) {
            String sql = "INSERT INTO MindMaps(title, created_at, is_favorite, preview_image, user_id) " +
                    "VALUES(?, ?, ?, ?, ?)";
            try (Connection conn = connectionProvider.getConnection();
                 PreparedStatement ps = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {
                ps.setString( parameterIndex: 1, entity.getTitle());
                ps.setTimestamp( parameterIndex: 2, Timestamp.valueOf(entity.getCreatedAt()));
                ps.setBoolean( parameterIndex: 3, entity.isFavorite());
                ps.setString( parameterIndex: 4, entity.getPreviewImage());
                ps.setInt( parameterIndex: 5, entity.getOwner().getId());
                ps.executeUpdate();

                try (ResultSet rs = ps.getGeneratedKeys()) {
                    if (rs.next()) {
                        entity.setId(rs.getInt( columnIndex: 1));
                    }
                }
            } catch (SQLException e) {
                throw new RuntimeException(e);
            }
        } else {
            String sql = "UPDATE MindMaps SET title = ?, is_favorite = ?, preview_image = ? WHERE id = ?";
            try (Connection conn = connectionProvider.getConnection();
                 PreparedStatement ps = conn.prepareStatement(sql)) {

                ps.setString( parameterIndex: 1, entity.getTitle());
                ps.setBoolean( parameterIndex: 2, entity.isFavorite());
                ps.setString( parameterIndex: 3, entity.getPreviewImage());
                ps.setInt( parameterIndex: 4, entity.getId());
                ps.executeUpdate();
```

```
            } catch (SQLException e) {
                throw new RuntimeException(e);
            }
        }
        return entity;
    }

    @Override  no usages
    public void delete(Integer id) {
        String sql = "DELETE FROM MindMaps WHERE id = ?";
        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt( parameterIndex: 1, id);
            ps.executeUpdate();
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }

    @Override  no usages
    public Optional<MindMap> getById(Integer id) {
        String sql = "SELECT id, title, created_at, is_favorite, preview_image, user_id FROM MindMaps WHERE id = ?";
        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setInt( parameterIndex: 1, id);
```

```
            ps.setInt( parameterIndex: 1, id);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
                    MindMap map = new MindMap();
                    map.setId(rs.getInt( columnLabel: "id"));
                    map.setTitle(rs.getString( columnLabel: "title"));
                    Timestamp ts = rs.getTimestamp( columnLabel: "created_at");
                    map.setOwner(new User(rs.getInt( columnLabel: "user_id"),  username: null,  passwordHash: null));
                    if (ts != null) {
                        // createdAt ми ставили в конструктор, але тут можемо перезаписати
                        // (потрібен сеттер, який ми вже додали)
                        // в нашому класі createdAt тільки геттер, можна додати сеттер або пропустити.
                    }
                    if (rs.getBoolean( columnLabel: "is_favorite")) {
                        map.markAsFavorite();
                    }
                    map.setPreviewImage(rs.getString( columnLabel: "preview_image"));
                    return Optional.of(map);
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return Optional.empty();
    }
```

```java
    @Override  no usages
    public List<MindMap> getAll() {
        throw new UnsupportedOperationException("Not needed yet");
    }

    public List<MindMap> getAllByUser(User user) {  1 usage
        String sql = "SELECT id, title, created_at, is_favorite, preview_image " +
                "FROM MindMaps WHERE user_id = ?";
        List<MindMap> maps = new ArrayList<>();
        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setInt( parameterIndex: 1, user.getId());
            try (ResultSet rs = ps.executeQuery()) {
                while (rs.next()) {
                    MindMap map = new MindMap();
                    map.setId(rs.getInt( columnLabel: "id"));
                    map.setTitle(rs.getString( columnLabel: "title"));
                    map.setOwner(user);
                    Timestamp ts = rs.getTimestamp( columnLabel: "created_at");
                    if (ts != null) {
                        // можна проігнорити або додати сеттер для createdAt
                    }
                    if (rs.getBoolean( columnLabel: "is_favorite")) {
                        map.markAsFavorite();
                    }
                    map.setPreviewImage(rs.getString( columnLabel: "preview_image"));
                    maps.add(map);
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return maps;
    }
}
```

MapElementRepository.java:

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

public class MapElementRepository {  no usages
}
```

(Пізніше буде доповнений)

JdbcConnectionProvider.java:

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

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class JdbcConnectionProvider {  9 usages

    private static final String URL = "jdbc:mysql://localhost:3306/mindmapdb";  1 usage
    private static final String USER = "root";  1 usage
    private static final String PASSWORD = ""; // АБО твій реальний пароль, якщо він є  1 usage

    static {
        try {
            Class.forName( className: "com.mysql.cj.jdbc.Driver"); // драйвер вже працює
        } catch (ClassNotFoundException e) {
            throw new RuntimeException("JDBC Driver not found", e);
        }
    }

    public Connection getConnection() throws SQLException {  11 usages
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

AuthService.java:

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

import com.example.mindmap.entities.User;
import com.example.mindmap.repositories.JdbcUserRepository;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Optional;

public class AuthService {  6 usages

    private final JdbcUserRepository userRepository;  3 usages

    public AuthService(JdbcUserRepository userRepository) {  1 usage
        this.userRepository = userRepository;
    }

    // ЛОГІН
    public Optional<User> login(String username, String password) {  1 usage
        String hashed = hashPassword(password);

        return userRepository.getByUsername(username)
                .filter( User u -> u.getPasswordHash().equals(hashed));
    }

    // РЕЄСТРАЦІЯ
    public User register(String username, String password) {  1 usage
        String hashed = hashPassword(password);          // тут хешуємо
```

```
        User user = new User( id: 0, username, hashed);      // зберігаємо вже хеш
        return userRepository.save(user);
    }

    // ПРИВАТНИЙ МЕТОД ДЛЯ ХЕШУВАННЯ
    private String hashPassword(String plainText) {  2 usages
        if (plainText == null) {
            return null;
        }
        try {
            MessageDigest digest = MessageDigest.getInstance( algorithm: "SHA-256");
            byte[] hashBytes = digest.digest(plainText.getBytes());
            StringBuilder sb = new StringBuilder();
            for (byte b : hashBytes) {
                sb.append(String.format("%02x", b)); // перетворюємо байти в HEX-рядок
            }
            return sb.toString();
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException("SHA-256 not supported", e);
        }
    }
}
```

MindMapService.java:

```
package com.example.mindmap.services;                                    ⚠1 ⚠1 ^

import com.example.mindmap.entities.MindMap;
import com.example.mindmap.entities.User;
import com.example.mindmap.repositories.JdbcMindMapRepository;

import java.util.List;

public class MindMapService {  9 usages

    private final JdbcMindMapRepository mapRepository;  3 usages

    public MindMapService(JdbcMindMapRepository mapRepository) {  1 usage
        this.mapRepository = mapRepository;
    }

    public MindMap createMap(String title, User owner) {  1 usage
        MindMap map = new MindMap( id: 0, title, owner);
        return mapRepository.save(map);
    }

    public List<MindMap> getMapsByUser(User owner) {  1 usage
        return mapRepository.getAllByUser(owner);
    }
}
```

LoginForm.java:

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

import com.example.mindmap.entities.User;
import com.example.mindmap.services.AuthService;
import com.example.mindmap.services.MindMapService;

import javax.swing.*;
import java.awt.*;
import java.util.Optional;

public class LoginForm extends JFrame {  2 usages

    private final AuthService authService;  3 usages
    private final MindMapService mindMapService;  2 usages

    private JTextField usernameField;  4 usages
    private JPasswordField passwordField;  4 usages

    public LoginForm(AuthService authService, MindMapService mindMapService) {  1 usage
        super( title: "Mind Map - Login");
        this.authService = authService;
        this.mindMapService = mindMapService;

        initComponents();
    }

    private void initComponents() {  1 usage
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        setSize( width: 320,  height: 180);
        setLocationRelativeTo(null);

        JPanel panel = new JPanel(new GridLayout( rows: 3,  cols: 2,  hgap: 5,  vgap: 5));
        panel.setBorder(BorderFactory.createEmptyBorder( top: 10,  left: 10,  bottom: 10,  right: 10));

        panel.add(new JLabel( text: "Username:"));
        usernameField = new JTextField();
        panel.add(usernameField);

        panel.add(new JLabel( text: "Password:"));
        passwordField = new JPasswordField();
        panel.add(passwordField);

        JButton loginBtn = new JButton( text: "Login");
        JButton registerBtn = new JButton( text: "Register");

        panel.add(loginBtn);
        panel.add(registerBtn);

        add(panel);

        loginBtn.addActionListener( ActionEvent e -> onLogin());
        registerBtn.addActionListener( ActionEvent e -> onRegister());
    }
```

```
    private void onLogin() {  1 usage
        String username = usernameField.getText();
        String password = new String(passwordField.getPassword());

        Optional<User> userOpt = authService.login(username, password);
        if (userOpt.isPresent()) {
            User user = userOpt.get();
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Login successful");

            SwingUtilities.invokeLater(() -> {
                new DashboardForm(user, mindMapService).setVisible(true);
            });
            dispose();
        } else {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Invalid username or password");
        }
    }

    private void onRegister() {  1 usage
        String username = usernameField.getText();
        String password = new String(passwordField.getPassword());

        if (username.isBlank() || password.isBlank()) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Username and password are required");
            return;
        }
```

```
        User created = authService.register(username, password);
        JOptionPane.showMessageDialog( parentComponent: this,
                message: "User created with id = " + created.getId());
    }
}
```

DashboardForm.java:

```
package com.example.mindmap.ui;

import com.example.mindmap.entities.MindMap;
import com.example.mindmap.entities.User;
import com.example.mindmap.services.MindMapService;

import javax.swing.*;
import java.awt.*;
import java.util.List;

public class DashboardForm extends JFrame {  1 usage

    private final User user;  4 usages
    private final MindMapService mindMapService;  3 usages

    private DefaultListModel<MindMap> listModel;  4 usages
    private JList<MindMap> mapsList;  2 usages

    public DashboardForm(User user, MindMapService mindMapService) {  1 usage
        super( title: "Mind Map - Dashboard");
        this.user = user;
        this.mindMapService = mindMapService;

        initComponents();
        loadMaps();
    }
```

```
28      private void initComponents() {  1 usage
29          setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
30          setSize( width: 500,  height: 320);
31          setLocationRelativeTo(null);
32
33          JPanel top = new JPanel(new BorderLayout());
34          top.setBorder(BorderFactory.createEmptyBorder( top: 5,  left: 5,  bottom: 5,  right: 5));
35          top.add(new JLabel( text: "Logged in as: " + user.getUsername()), BorderLayout.WEST);
36
37          JButton createBtn = new JButton( text: "Create new map");
38          top.add(createBtn, BorderLayout.EAST);
39
40          listModel = new DefaultListModel<>();
41          mapsList = new JList<>(listModel);
42
43          add(top, BorderLayout.NORTH);
44          add(new JScrollPane(mapsList), BorderLayout.CENTER);
45
46          createBtn.addActionListener( ActionEvent e -> onCreateMap());
47      }
48
49      private void loadMaps() {  2 usages
50          listModel.clear();
51          List<MindMap> maps = mindMapService.getMapsByUser(user);
52          for (MindMap m : maps) {
53              listModel.addElement(m);
54          }
```

```
54          }
55      }
56
57      private void onCreateMap() {  1 usage
58          String title = JOptionPane.showInputDialog( parentComponent: this,  message: "Enter map title:");
59          if (title == null) {
60              return; // користувач натиснув Cancel
61          }
62          if (title.isBlank()) {
63              JOptionPane.showMessageDialog( parentComponent: this,  message: "Назва мапи обов'язкова");
64              return;
65          }
66
67          mindMapService.createMap(title, user);
68          loadMaps();
69      }
70  }
```

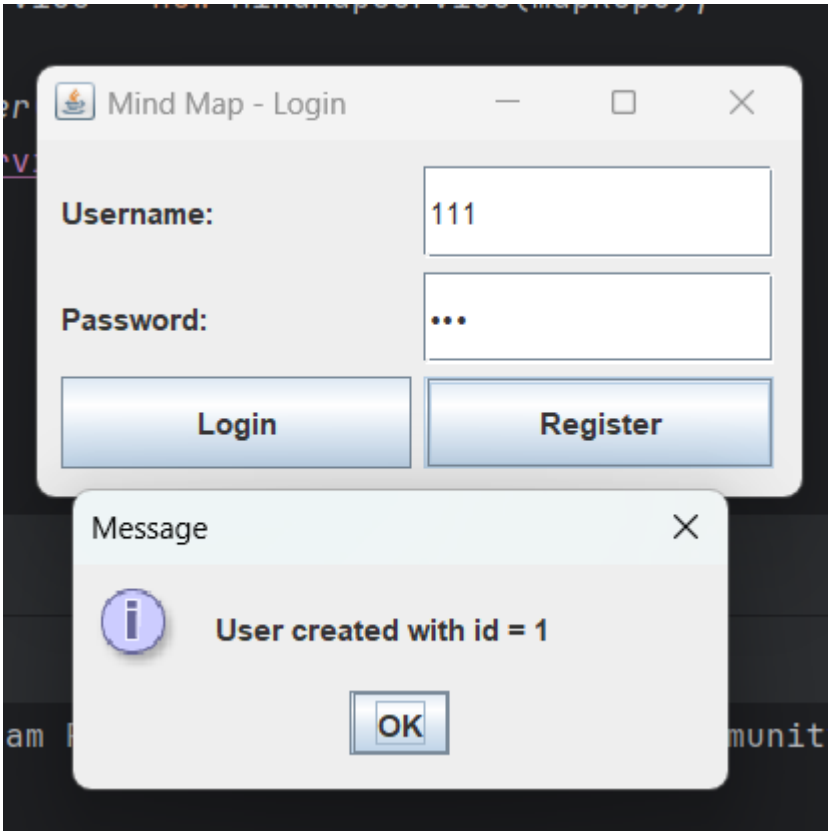
```sql
CREATE DATABASE IF NOT EXISTS mindmapdb CHARACTER SET utf8mb4;
USE mindmapdb;

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

-- 2. Таблиця категорій
CREATE TABLE IF NOT EXISTS 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 IF NOT EXISTS 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 мапи
CREATE TABLE IF NOT EXISTS 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
);
```

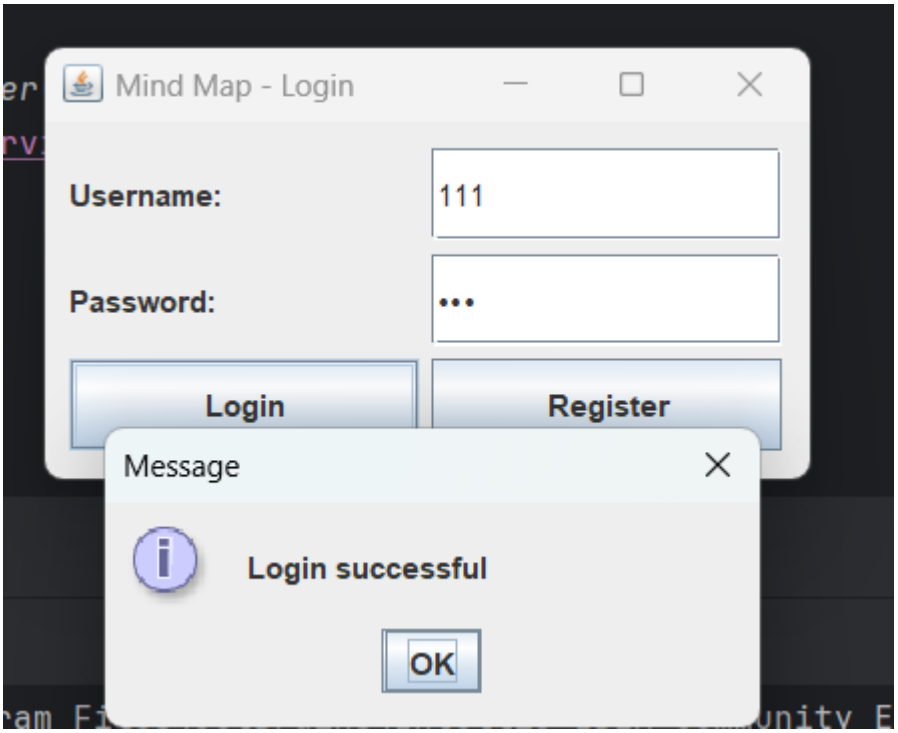
Output

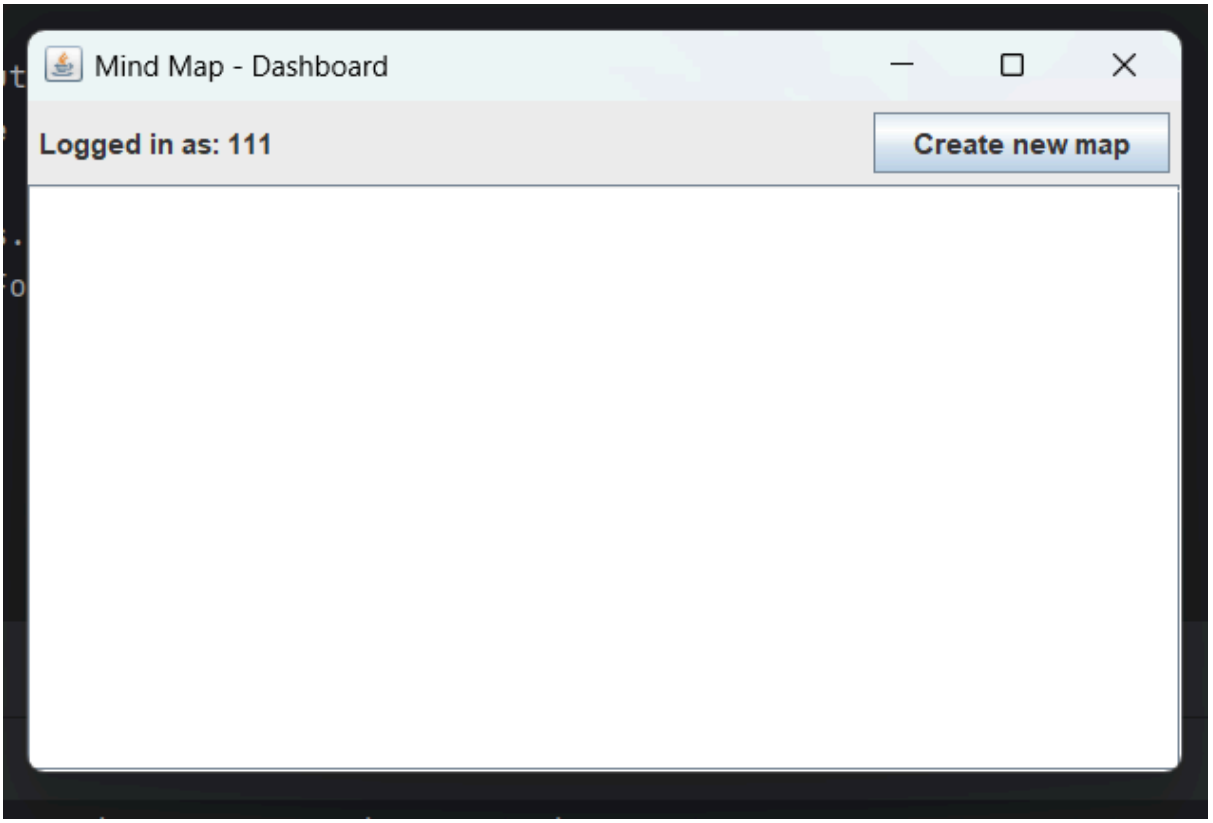
Action Output

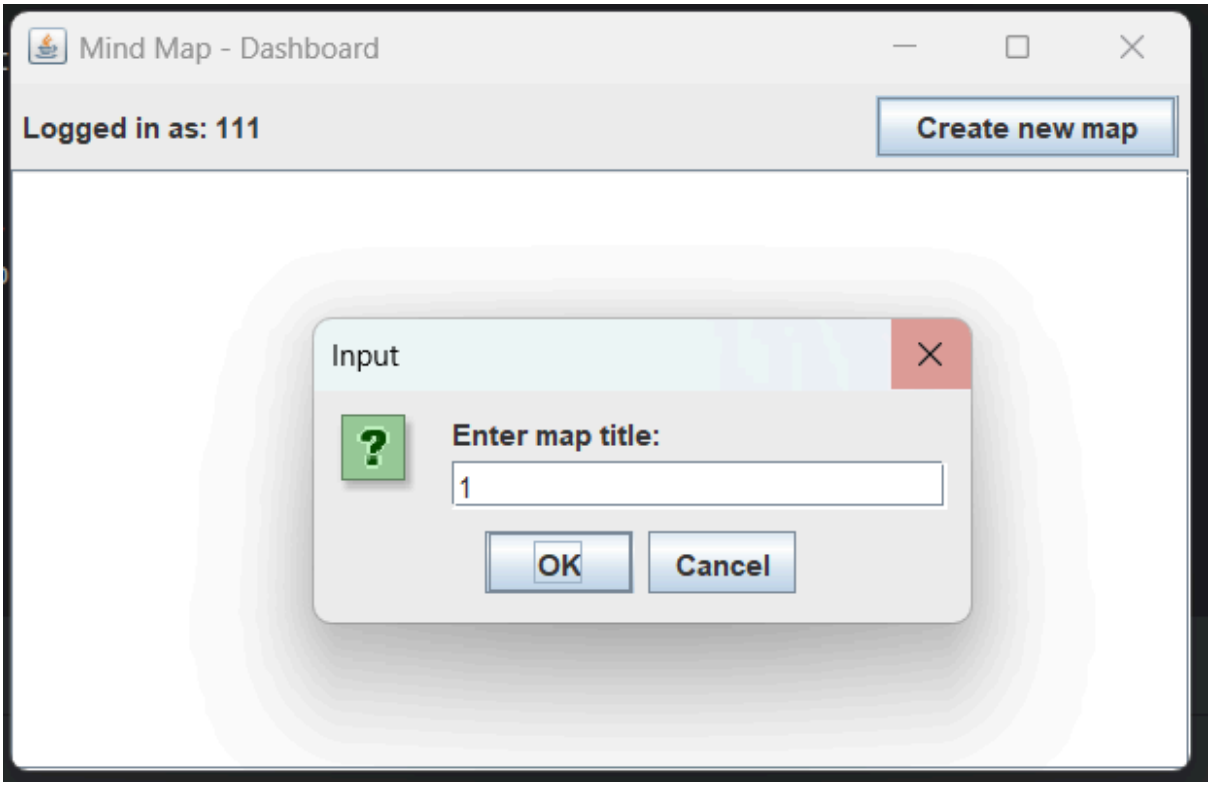
#	Time	Action	Message	Duration / Fetch
1	17:25:59	CREATE DATABASE IF NOT EXISTS mindmapdb CHARACTER SET utf8mb4	1 row(s) affected	0.000 sec
2	17:25:59	USE mindmapdb	0 row(s) affected	0.000 sec
3	17:25:59	CREATE TABLE IF NOT EXISTS Users (id INT PRIMARY KEY AUTO_INCREMENT, username VA...	0 row(s) affected	0.063 sec
4	17:25:59	CREATE TABLE IF NOT EXISTS Categories (id INT PRIMARY KEY AUTO_INCREMENT, title VAR...	0 row(s) affected	0.031 sec
5	17:25:59	CREATE TABLE IF NOT EXISTS MindMaps (id INT PRIMARY KEY AUTO_INCREMENT, title VAR...	0 row(s) affected	0.031 sec
6	17:25:59	CREATE TABLE IF NOT EXISTS MapElements (id INT PRIMARY KEY AUTO_INCREMENT, map_i...	0 row(s) affected	0.031 sec

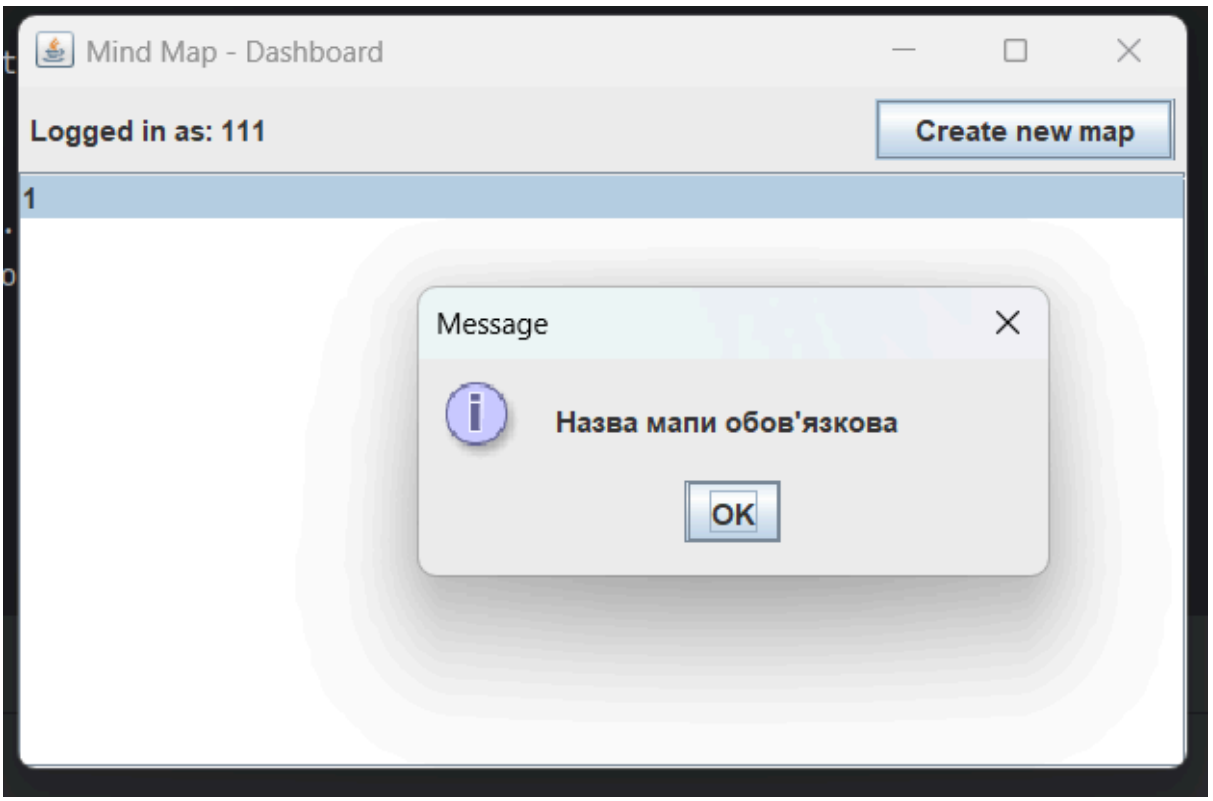
Початки нашої системи, базова реалізація реєстрації:






Mind Map - Dashboard
Logged in as: 111
Create new map


Mind Map - Dashboard
Logged in as: 111
Create new map
Input
Enter map title:
1
OK
Cancel



3.3. Питання до лабораторної роботи

1. Що собою становить діаграма розгортання?

Діаграма розгортання показує, як програмна система фізично розміщується на обладнанні. Вона відображає сервери, пристрої, мережі та програмні компоненти, що на них запускаються.

2. Які бувають види вузлів на діаграмі розгортання?

Вузли бувають фізичні (сервери, комп'ютери, мобільні пристрої) і віртуальні (контейнери, віртуальні машини, середовища виконання). Вони показують, де саме працює система.

3. Які бувають зв'язки на діаграмі розгортання?

Основний тип зв'язку – комунікаційні зв'язки, які показують, як вузли обмінюються даними. Вони позначають мережеві канали, протоколи та взаємодію між частинами системи.

4. Які елементи присутні на діаграмі компонентів?

На діаграмі компонентів присутні це: Компоненти (частини системи з чіткими функціями), Інтерфейси, Залежності між компонентами, Пакети, що групують логічну структуру.

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

Зв'язки показують залежності між компонентами: хто використовує чий інтерфейс, які модулі взаємодіють і як саме система логічно побудована.

6. Які бувають види діаграм взаємодії?

Є два основні типи:

Діаграма послідовностей – показує порядок викликів.

Діаграма комунікацій – показує, як об'єкти пов’язані й взаємодіють.

Обидві описують динамічну поведінку системи.

7. Для чого призначена діаграма послідовностей?

Діаграма послідовностей відображає, у якій послідовності об'єкти надсилають один одному повідомлення під час виконання певного сценарію. Вона фіксує логіку процесів у часі.

8. Які ключові елементи можуть бути на діаграмі послідовностей?

Основні елементи: Об’єкти (учасники), Лінії життя, Повідомлення та виклики методів, Активності, Умовні або циклічні фрагменти. Вони показують, як виконується сценарій покроково.

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

Діаграма послідовностей деталізує конкретний сценарій варіанта використання. Якщо діаграма використання показує що робить система, то діаграма послідовностей показує як саме це відбувається.

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

Діаграми послідовностей демонструють взаємодію об’єктів, які належать до класів, зображених на діаграмі класів. Таким чином, вони показують динамічну поведінку статичної структури, описаної класами.

Висновок: У цій лабораторній роботі було здійснено розширення та структуризацію коду відповідно до попередньо спроектованих діаграм компонентів, розгортання та діаграм послідовностей. Основною метою було показати реальне втілення архітектури системи у програмному коді, забезпечити повний цикл взаємодії з базою даних та реалізувати перші елементи інтерфейсу користувача.