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

із дисципліни «**Технології розроблення програмного забезпечення**»

Тема «**Діаграма варіантів використання. Сценарії варіантів використання. Діаграма UML. Діаграми класів. Концептуальна модель системи**»

Виконав:

студент групи ІА–24

Яблонський Д.Б.

Перевірив:

Мягкий М.Ю.

Київ 2024

Зміст

Мета: Вивчення основних принципів побудови та проектування програмних систем з використанням схем прецедентів, діаграм класів та структури бази даних.

Теоретичні відомості

Прецеденти (Use Case Diagram)

Діаграма прецедентів використовується для візуалізації функціональних вимог до системи. Вона показує, як користувачі (актор або актори) взаємодіють із системою через певні сценарії використання (прецеденти). Основними елементами діаграми є актори, прецеденти, а також зв'язки між ними. Прецеденти допомагають виявити основні функції системи та забезпечують їх розуміння на високому рівні.

Діаграма класів (Class Diagram)

Діаграма класів використовується для моделювання статичної структури системи. Вона показує класи, атрибути класів, методи (операції), а також зв'язки між класами, такі як асоціації, агрегації, композиції та успадкування. Класи представляють основні компоненти системи, їхні характеристики (атрибути) та поведінку (методи). Зв'язки показують, як ці класи взаємодіють між собою.

База даних та її структура

База даних — це організований набір даних, які зберігаються в структурованому вигляді, зазвичай у вигляді таблиць. Таблиці в базі даних складаються з рядків (записів) і стовпців (полів), що містять атрибути даних. Структура бази даних визначає, як дані взаємопов'язані між собою. Основними елементами є таблиці, ключі (первинні та зовнішні) і зв'язки між таблицями (один-до-одного, один-до-багатьох, багато-до-багатьох).

Шаблон Репозиторію (Repository Pattern)

Шаблон Репозиторію використовується для абстрагування доступу до даних. Він дозволяє взаємодіяти з базою даних через клас-репозиторій, що інкапсулює всі операції збереження, отримання, оновлення та видалення даних. Це забезпечує слабку залежність між бізнес-логікою та логікою доступу до даних, роблячи систему більш гнучкою для змін або модернізацій.

Вибір прецедентів та аналіз

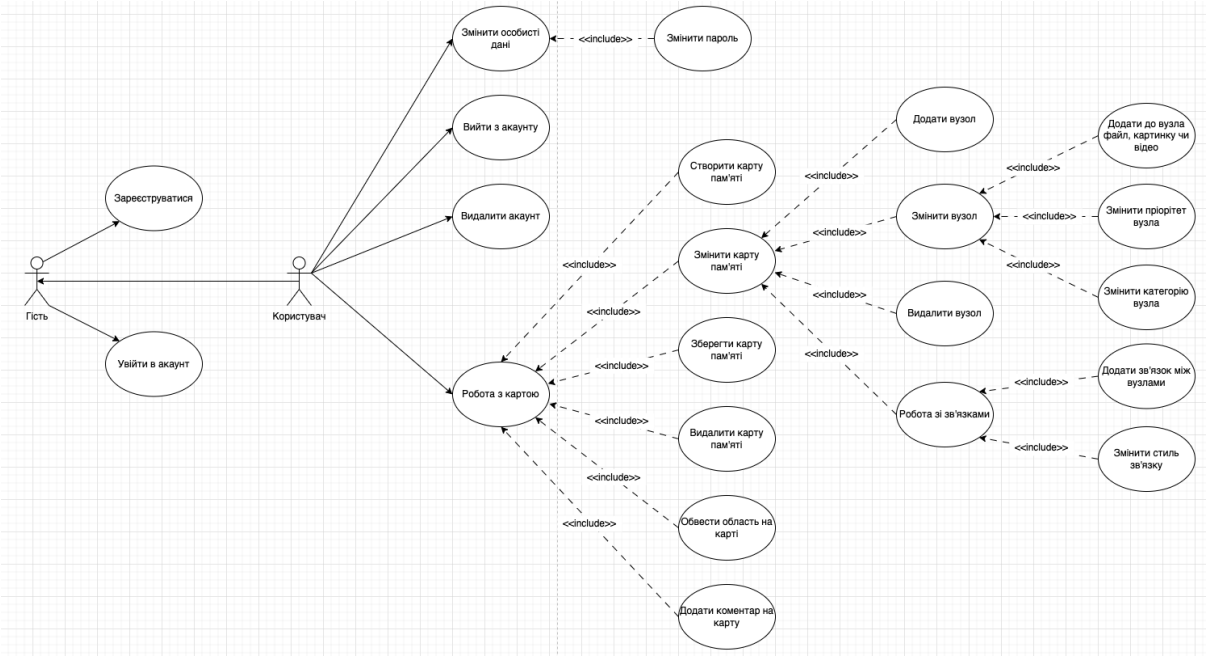
Прецеденти дозволяють ідентифікувати сценарії використання, що найбільш підходять для вашої системи. Важливо вибрати і проаналізувати кілька подібних систем, щоб побачити, як їх функціональність відповідає вимогам вашого проекту. Це допомагає розробити найбільш ефективні та зручні рішення для користувачів.

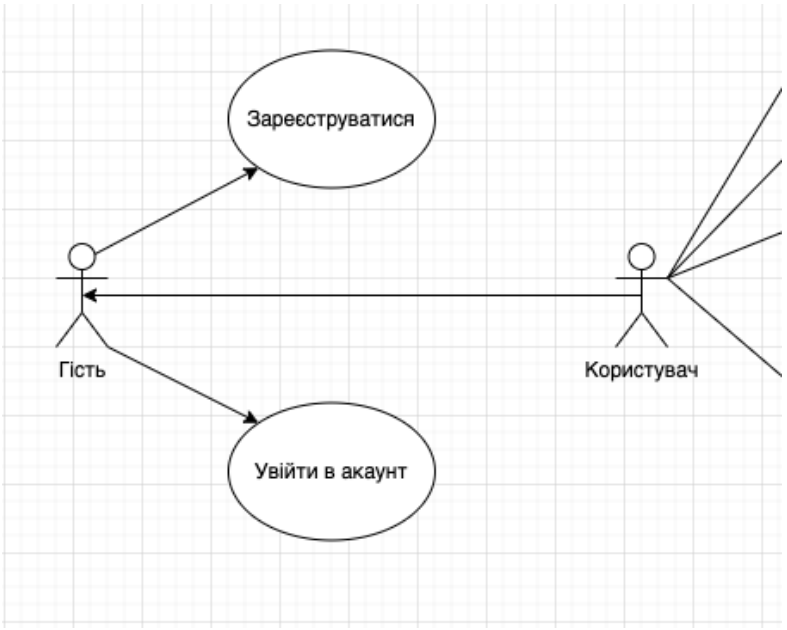
Хід роботи

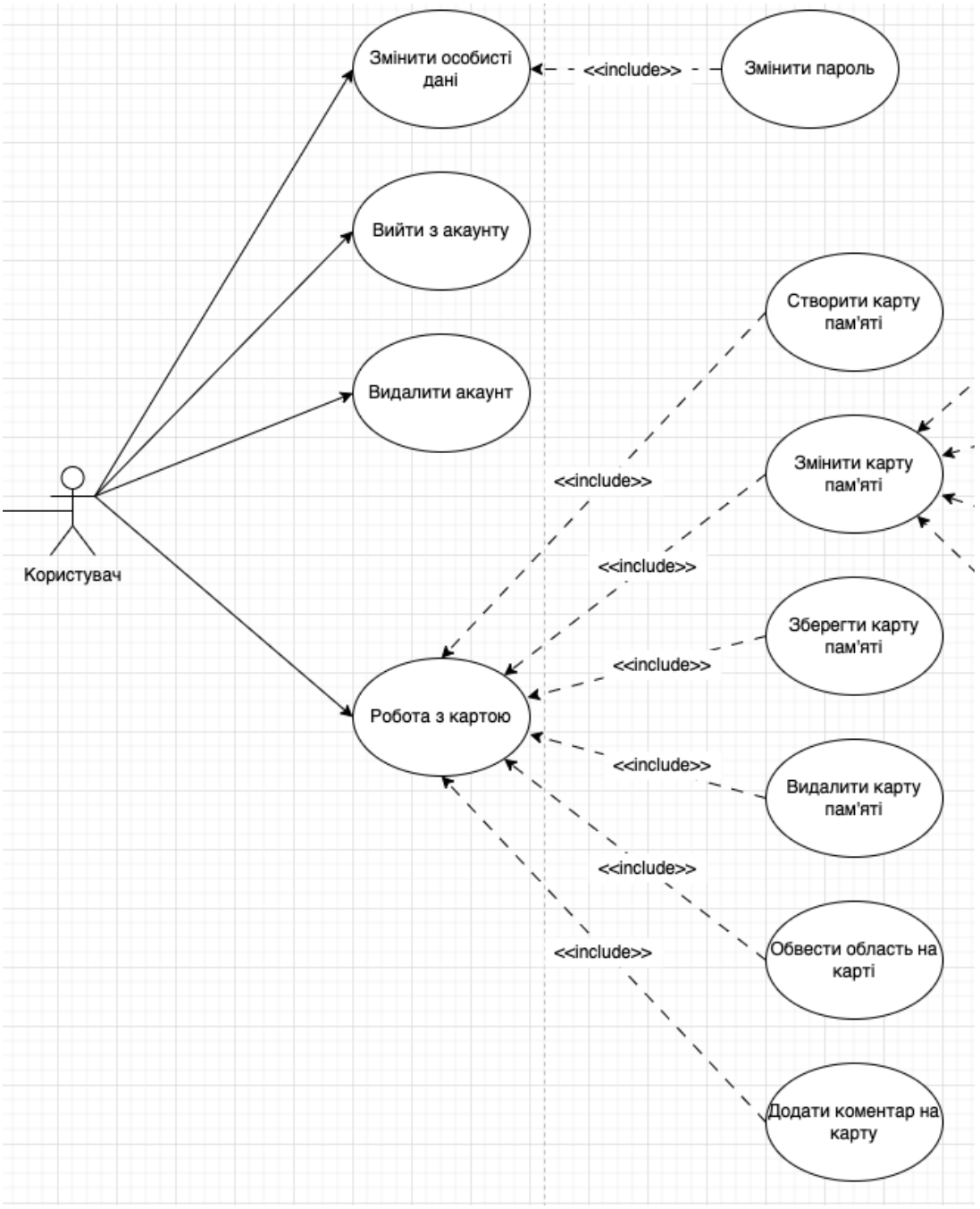
1. Ознайомитися з короткими теоретичними відомостями.
2. Проаналізуйте тему та намалюйте схему прецеденту, що відповідає обраній темі лабораторії.
3. Намалюйте діаграму класів для реалізованої частини системи.
4. Виберіть 3 прецеденти і напишіть на їх основі прецеденти.
5. Розробити основні класи і структуру системи баз даних.
6. Класи даних повинні реалізувати шаблон Репозиторію для взаємодії з базою даних.

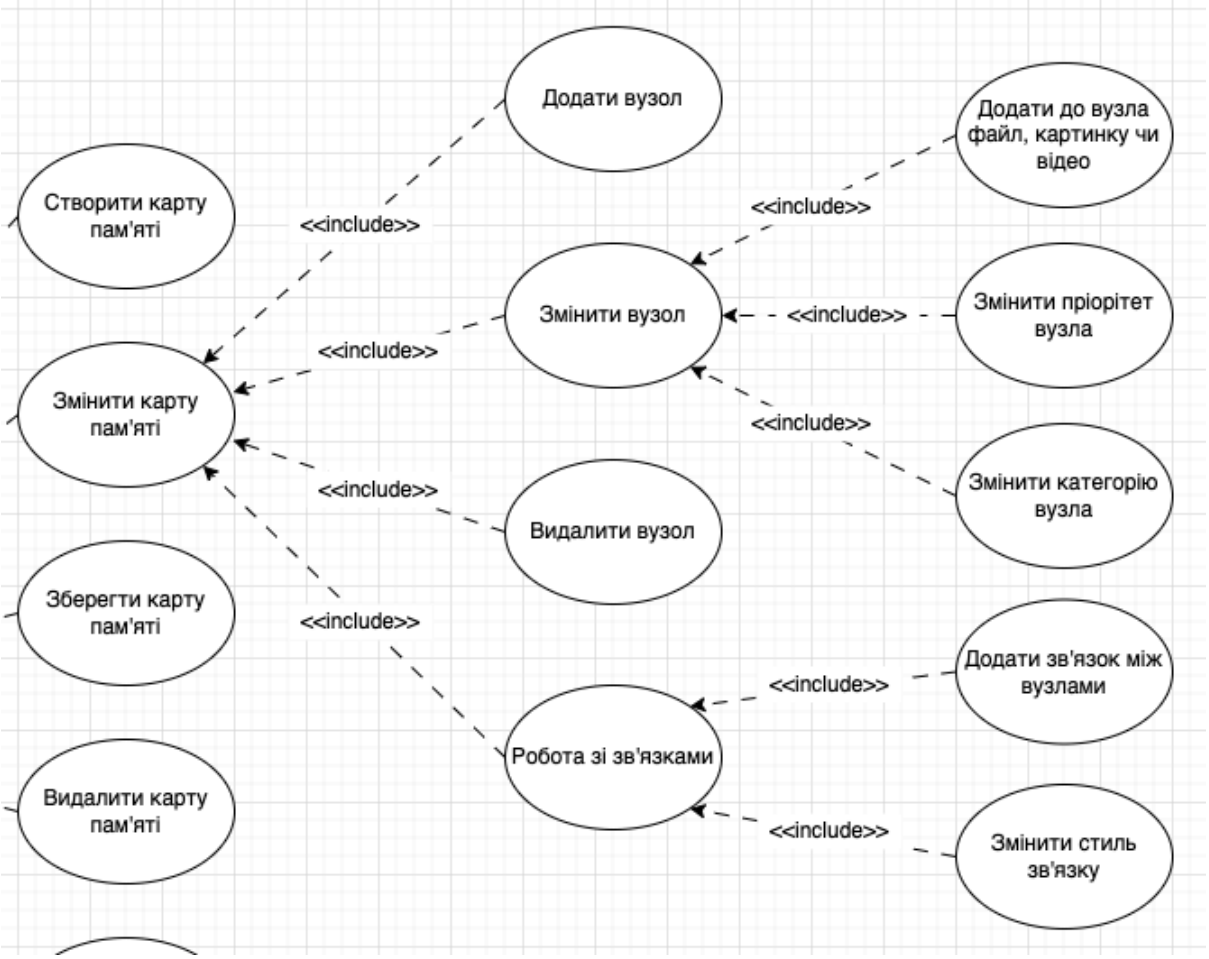
Тема: Mind-mapping software

Діаграма прецедентів




Зареєструватися
Гість
Користувач
Увійти в акаунт


Змінити особисті дані
<<include>>
Змінити пароль
Вийти з акаунту
Створити карту пам'яті
Видалити акаунт
Змінити карту пам'яті
<<include>>
<<include>>
Користувач
Зберегти карту пам'яті
<<include>>
Робота з картою
<<include>>
Видалити карту пам'яті
<<include>>
<<include>>
Обвести область на карті
Додати коментар на карту



Прецедент 1: Гість реєструється

Передумови:

- Гість не має існуючого акаунта в системі.
- Гість знаходиться на сторінці реєстрації.

Післяумови:

- Гість стає зареєстрованим користувачем.
- Створено новий запис у базі даних користувачів.
- Гість автоматично увійшов у систему після реєстрації.

Основний хід подій:

1. Гість відкриває форму реєстрації.
2. Гість заповнює необхідні поля (ім'я, email, пароль).
3. Гість натискає кнопку "Зареєструватися".
4. Система перевіряє введені дані на коректність і унікальність email.
5. Якщо все коректно, система створює новий акаунт і авторизує користувача.
6. Гість стає зареєстрованим користувачем і отримує доступ до

функцій, доступних тільки для зареєстрованих користувачів.

Виключення:

- Якщо email уже використовується, система видає помилку про існуючий акаунт і пропонує увійти в систему або використати інший email.
- Якщо введені дані некоректні (наприклад, неправильний формат email або слабкий пароль), система повідомляє про помилку і пропонує виправити дані.

Прецедент 2: Користувач змінює особисті дані

Передумови:

- Користувач вже зареєстрований і увійшов у систему.
- Користувач знаходиться на сторінці налаштувань свого акаунта.

Післяумови:

- Особисті дані користувача (ім'я, email, пароль) успішно оновлені.
- Зміни відображаються в системі під час наступного входу користувача.

Основний хід подій:

1. Користувач заходить на сторінку налаштувань акаунта.
2. Користувач змінює свої особисті дані.
3. Користувач натискає кнопку "Зберегти зміни".
4. Система перевіряє коректність нових даних.
5. Якщо дані правильні, система оновлює профіль користувача і підтверджує успішне збереження.

Виключення:

- Якщо новий email уже зайнятий іншим користувачем, система видає повідомлення про помилку і просить вибрати інший email.
- Якщо введені дані некоректні (наприклад, помилки в форматі email), система відображає повідомлення з проханням виправити помилки.

Прецедент 3: Користувач додає коментар на карті

Передумови:

- Користувач увійшов у систему.
- Користувач працює з інтерфейсом карти, яка відображається на екрані.
- Карта активна і доступна для редагування.

Післяумови:

- Коментар успішно доданий на карту.
- Коментар зберігається в базі даних і стає доступним для перегляду на цій карті в майбутньому.

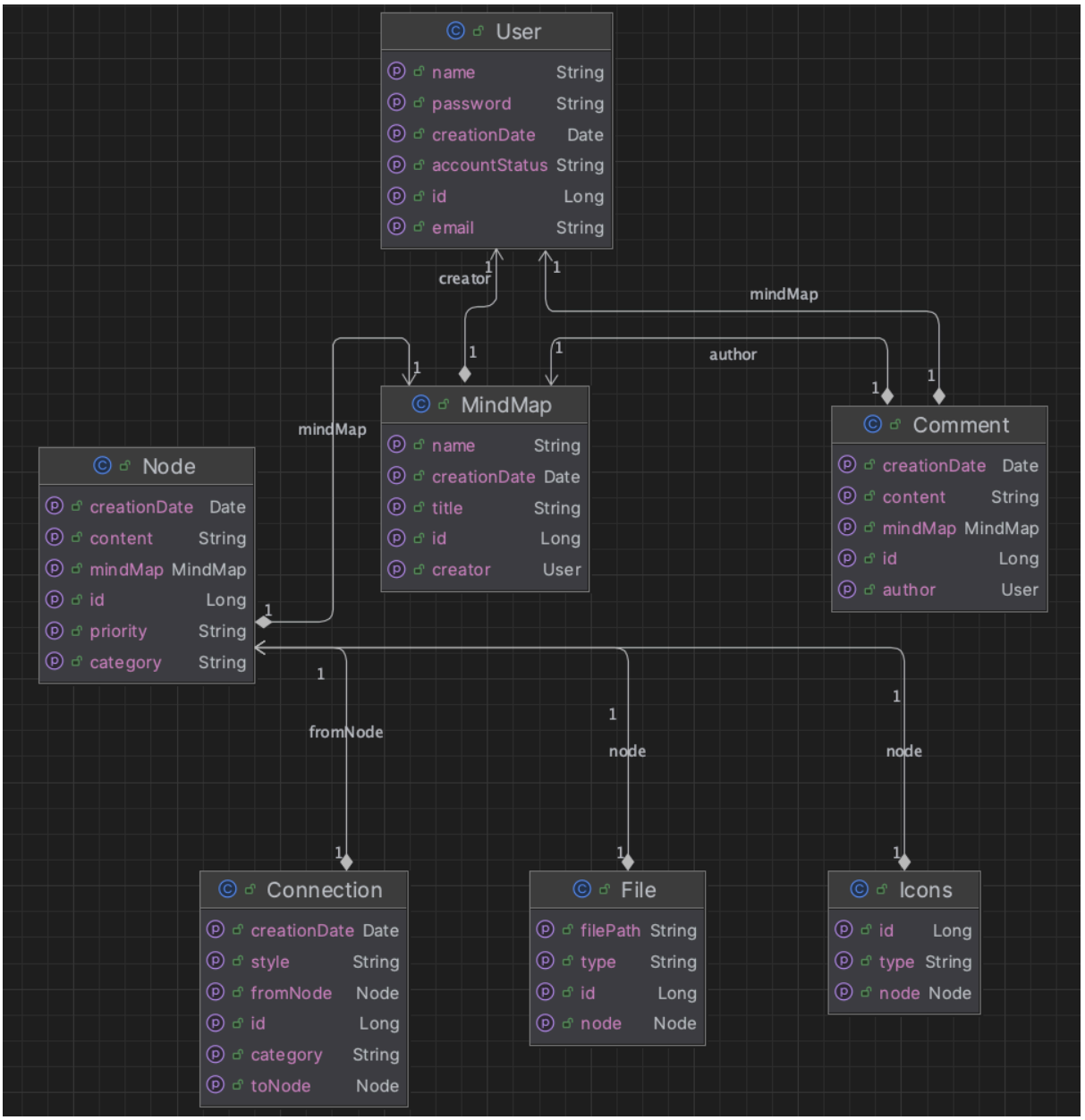
Основний хід подій:

1. Користувач відкриває карту для редагування.
2. Користувач натискає кнопку "Додати коментар".
3. Користувач вибирає точку на карті, де він хоче залишити коментар.
4. Система відкриває поле для введення тексту коментаря.
5. Користувач вводить текст коментаря.
6. Користувач натискає кнопку "Зберегти"
7. Система зберігає коментар і відображає його на вибраній точці на карті.

Виключення:

- Якщо користувач залишає порожнє поле для коментаря, система видає повідомлення про необхідність заповнити текст перед збереженням.

Діаграма класів



User (Користувач)

- ID: Унікальний ідентифікатор користувача (первинний ключ).
- username: Логін користувача.
- password: Пароль користувача.
- email: Електронна пошта.
- accountStatus: Статус облікового запису (наприклад, "active" або "deleted").
- creationData: Дата створення облікового запису.

MindMap (Карта пам'яті)

- ID: Унікальний ідентифікатор карти пам'яті.

- creator: Автор карти.
- title: Назва карти пам'яті.
- creationData: Дата створення карти пам'яті.

Node (Вузол на карті)

- ID: Унікальний ідентифікатор вузла.
- mindMap: Карта пам'яті, до якої належить вузол.
- content: Вміст вузла (текст або дані).
- priority: Пріоритет вузла.
- category: Категорія вузла.
- creationData: Дата створення вузла.

Connection (Зв'язок)

- ID: Унікальний ідентифікатор зв'язку.
- fromNode: Початковий вузол.
- toNode: Кінцевий вузол.
- style: Стиль зв'язку (тип лінії, пунктир тощо).
- category: Категорія зв'язку.
- creationData: Дата створення зв'язку.

File (Файл)

- ID: Унікальний ідентифікатор файлу.
- node: Вузол, до якого прикріплено файл
- filePath: Шлях до файлу.
- type: Тип файлу (зображення, відео).

Icon (Іконка)

- ID: Унікальний ідентифікатор іконки.
- node: Вузол, до якого прикріплено іконку.
- iconType: Тип іконки (категорія або терміновість).

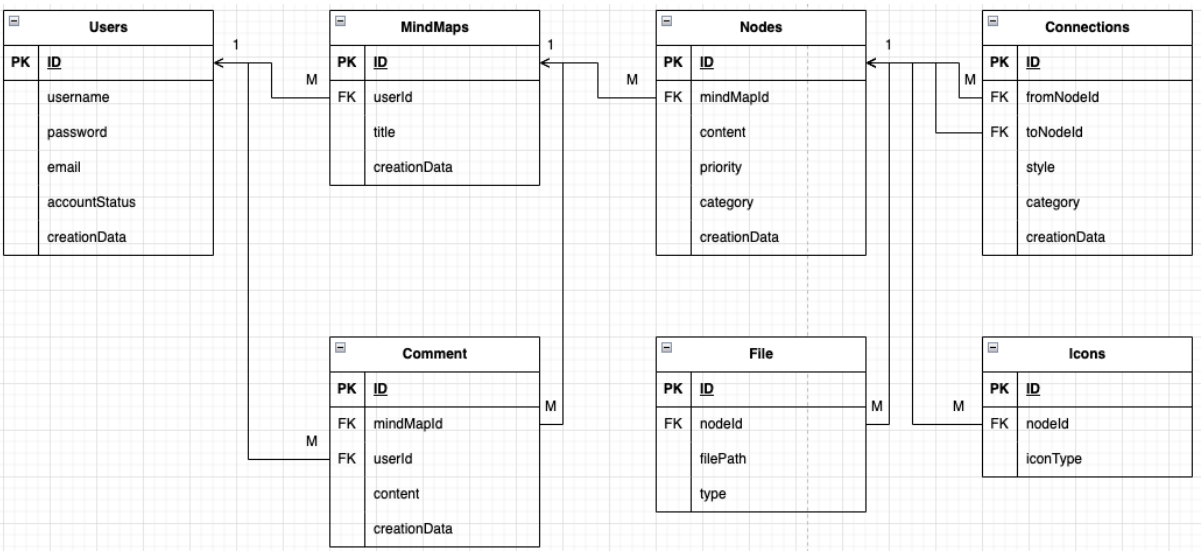
Comment (Коментар)

- ID: Унікальний ідентифікатор коментаря.
- mindMap: Карта пам'яті, на якій залишено коментар
- author: Користувач, який залишив коментар.
- content: Текст коментаря.
- creationData: Дата створення коментаря.

Зв'язки між сутностями:

- User має зв'язок один до багатьох з MindMap (один користувач може створити кілька карт пам'яті).
- MindMap має зв'язок один до багатьох з Node (одна карта пам'яті може мати кілька вузлів).
- MindMap має зв'язок один до багатьох з Comment (одна карта може мати кілька коментарів).
- Node має зв'язок один до багатьох з File (один вузол може мати кілька прикріплених файлів).
- Node має зв'язок один до багатьох з Icon (один вузол може мати кілька іконок).
- Node має зв'язок один до багатьох з Connection (один вузол може мати кілька з'єднань з іншими вузлами).

Структура бази даних



Класи та репозиторії можна знайти в директорії за таким шляхом: TRPZ_labs_yablonskyi_ia-24/MindMappingSoftware/src/main/java/org/example/mindmappingsoftware

Висновок: У ході виконання даної лабораторної роботи мною було опрацьовано основи моделювання програмних систем з використанням діаграм прецедентів та діаграм класів. Я розробив структуру бази даних для системи карт пам'яті, яка включає сутності користувачів, карт пам'яті, вузлів, зв'язків між ними, файлів, іконок та коментарів.

В цій роботі я створив діаграми прецедентів, які описують взаємодію користувача із системою та побудував діаграму класів, що відображають зв'язки між сутностями бази даних.

Отримані результати дозволяють краще зрозуміти процес моделювання програмного забезпечення і забезпечити ефективну взаємодію між різними компонентами системи. Такий підхід допомагає створювати гнучкі, масштабовані та підтримувані рішення для управління даними.