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

Технологія розробки програмного забезпечення

«ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЇ ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ КЛАСІВ. КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ»

Виконала студентка
групи ІА-23:
Шрубович Н. С.

Перевірив:
Мягкий М. Ю.

Київ 2024

Зміст

Тема: Діаграма варіантів використання. Сценарії варіантів використання. Діаграми UML. Діаграми класів. Концептуальна модель системи

Завдання**:**

1. Ознайомитися з короткими теоретичними відомостями.
2. Проаналізуйте тему та намалюйте схему прецеденту.
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.
7. Підготувати звіт про хід виконання лабораторних робіт. Звіт, що подається повинен містити: діаграму прецедентів, діаграму класів системи, вихідні коди класів системи, а також зображення структури бази даних.

..3 Текстовий редактор (strategy, command, observer, template method, flyweight, SOA)

Текстовий редактор повинен вміти розпізнавати текстові файли в будь-якій кодуванні, мати розширені функції редагування: макроси, сніппети, підказки, закладки, перехід на рядок / сторінку, підсвічування синтаксису (для однієї мови програмування або розмітки на розсуд студента).

Теоретичні відомості

• Діаграма варіантів використання (Use Case Diagram) – це тип діаграми UML, що описує функціональність системи з точки зору її користувачів і взаємодії між ними та системою. Вона показує, які дії (варіанти використання) можуть виконуватися користувачами, але не вдається у внутрішні механізми їх реалізації.

• Сценарії варіантів використання (Use Case Scenarios) – це текстовий опис варіантів використання, де детально викладається, як система повинна реагувати на дії користувачів у кожній конкретній ситуації. Включає в себе основний потік подій та альтернативні шляхи розвитку сценарію.

• Діаграма класів (Class Diagram) – це структура, яка моделює класи системи, їх властивості, методи, а також зв'язки між ними. Класи представляють основні об'єкти системи, які мають атрибути та операції, а також відображають взаємодію між різними компонентами.

• Концептуальна модель системи – це абстрактне представлення об'єктів та зв'язків між ними, що відображає ключові аспекти системи з точки зору бізнесу або предметної області. Вона описує основні компоненти, їх взаємодію та структуру, але не деталізує технічну реалізацію. Ці діаграми дозволяють аналізувати вимоги до системи та планувати її розробку.

Хід Роботи

Діаграма прецендентів

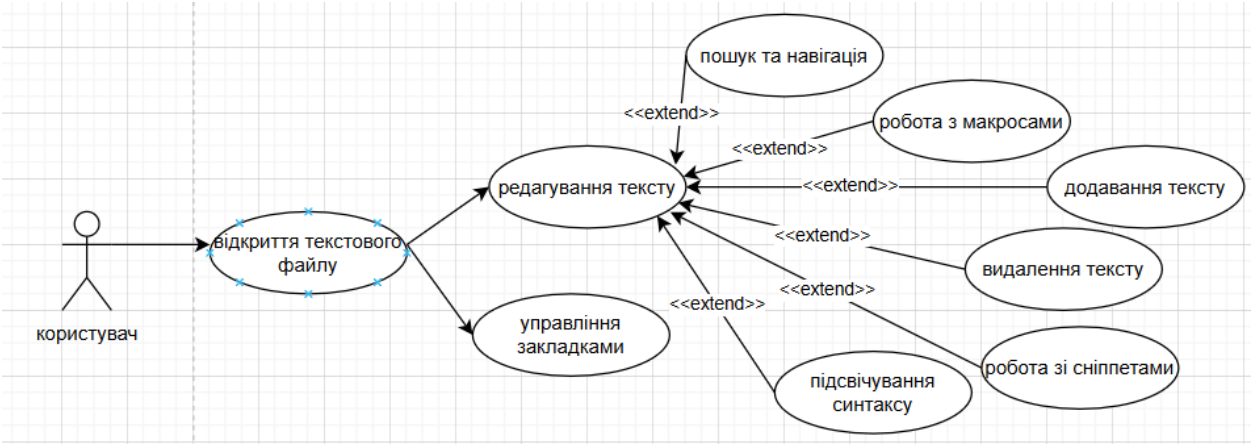

Рис 1. Діаграма прецендентів

Користувач запускає текстовий редактор та може:

- Відкрити текстовий файл (розпізнати кодування тексту та завантажити його у редактор).
- Редагувати текст (додавати новий текст, видаляти або змінювати текст, працювати зі сніппетами, записувати макроси, вставляти закладки, змінювати параметри підсвічування синтаксису).
- Шукати та навігувати по документу (шукати ключові слова, переходити на рядок або сторінку).

Прецеденти на основі трьох прецедентів:

1. Відкриття текстового файлу

- **Передумови:** Користувач має доступ до текстового файлу на диску.
- **Постумови:** Файл успішно відкрито в редакторі, а його кодування правильно розпізнано. У разі помилки система повідомляє користувача.
- **Сторони взаємодії:** Користувач, текстовий редактор.
- **Короткий опис:** Користувач відкриває текстовий файл, який автоматично завантажується в редактор з урахуванням його кодування.

Основний потік подій:

1. Користувач запускає редактор.
2. Обирає опцію відкриття файлу.
3. Система відображає діалог вибору файлу.
4. Користувач обирає файл і натискає "Відкрити".
5. Система розпізнає кодування файлу та завантажує його в редактор.

6. Успішно відкритий файл відображається у вікні редактора.

Винятки:

- **Неправильне кодування:** Якщо система не може розпізнати кодування, вона пропонує користувачу обрати його вручну.
- **Файл не знайдено:** Система повідомляє про помилку доступу до файлу.

2. Редагування тексту

- **Передумови:** Користувач має доступ до відкритого текстового файлу.
- **Постумови:** Зміни в тексті збережено або оновлено у файлі. У разі помилки користувач отримує відповідне повідомлення.
- **Сторони взаємодії:** Користувач, текстовий редактор.
- **Короткий опис:** Користувач вносить зміни до тексту (додає текст, видаляє, змінює), використовуючи додаткові функції, такі як макроси чи сніппети.

Основний потік подій:

1. Користувач відкриває файл для редагування.
2. Вносить зміни до тексту.
3. Використовує додаткові функції:
    - Додає сніппети.
    - Додає текст.
    - Видаляє текст.
    - Використовує макроси.
    - Використовує пошук та навігацію.
    - Змінює параметри підсвічування синтаксису.
4. Система зберігає зміни після натискання "Зберегти".

Винятки:

- **Некоректний формат тексту:** Система повідомляє про помилку форматування.
- **Помилка збереження:** Якщо система не може зберегти файл, користувачу пропонується обрати інше місце або виправити проблему.

3. Управління закладками

- **Передумови:** Користувач має доступ до відкритого текстового файлу.
- **Постумови:** Закладки додаються, видаляються або використовуються для переходу до позначеного місця у документі. У разі помилки користувач отримує відповідне повідомлення.
- **Сторони взаємодії:** Користувач, текстовий редактор.
- **Короткий опис:** Користувач може додавати закладки до рядків тексту, видаляти їх або використовувати для швидкої навігації.

Основний потік подій:

1. Користувач відкриває текстовий файл.
2. Вибирає місце у документі, яке потрібно позначити закладкою.
3. Активує функцію додавання закладки (наприклад, через контекстне меню або комбінацію клавіш).
4. Система додає закладку до обраного місця та відображає її у списку закладок.
5. Користувач може:
    - Перейти до існуючої закладки зі списку.
    - Видалити закладку.

Винятки:

- **Закладка вже існує:** Якщо користувач намагається додати закладку в те саме місце, система повідомляє про це.
- **Некоректний вибір:** Якщо користувач обирає неіснуючу або пошкоджену закладку, система повідомляє про помилку.

Діаграма класів

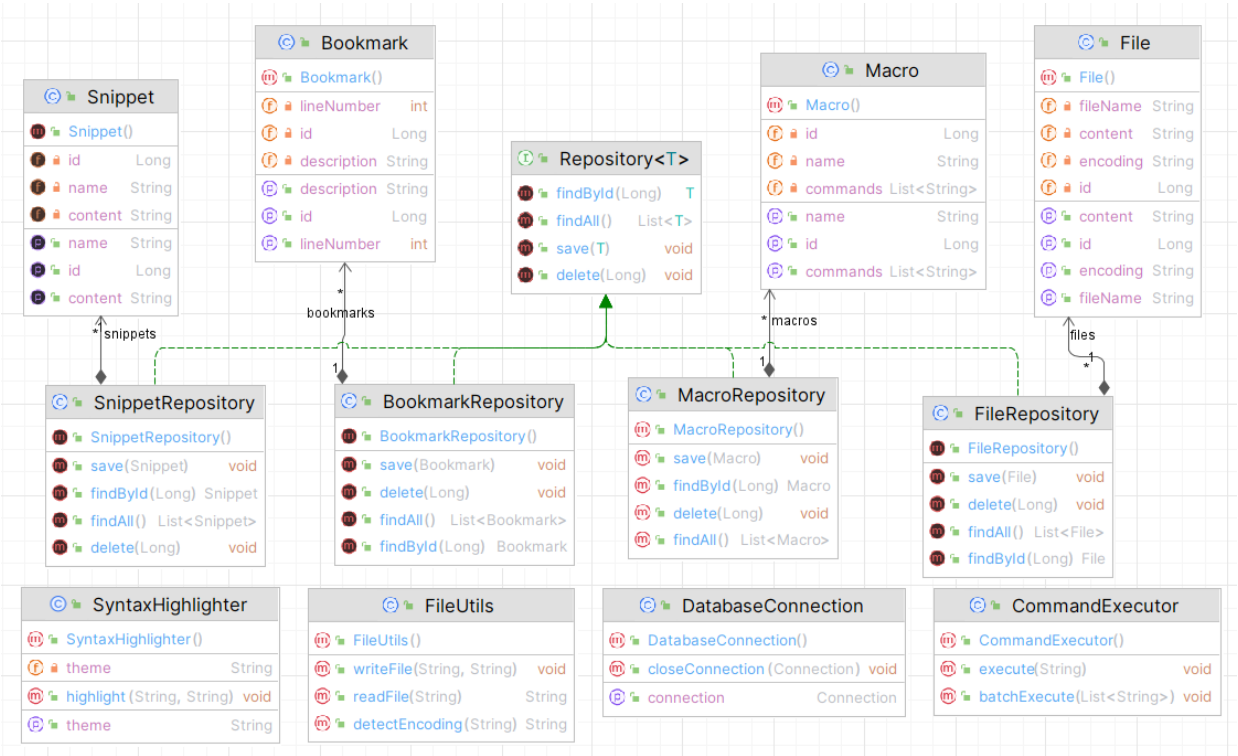

Рис 2. Діаграма класів

Основні компоненти діаграми:

1. Repository Pattern

- Repository<T> – базовий інтерфейс для CRUD-операцій:
  - save(T t) – для збереження об'єкта.
  - findById(Long id) – для пошуку об'єкта за ідентифікатором.
  - delete(Long id) – для видалення об'єкта.
  - findAll() – для отримання списку всіх об'єктів.
  - Забезпечує розділення бізнес-логіки додатка і логіки роботи з базою даних.

2. Моделі

- File – модель, яка представляє текстовий файл. Поля:
  - id (Long) – ідентифікатор файлу.
  - fileName (String) – назва файлу.
  - encoding (String) – кодування файлу.
  - content (String) – вміст файлу.
- Snippet – модель для опису сніппета. Поля:
  - id (Long) – ідентифікатор сніппета.
  - name (String) – назва сніппета.
  - content (String) – вміст сніппета.
- Macro – модель для опису макросів. Поля:
  - id (Long) – ідентифікатор макросу.
  - name (String) – назва макросу.
  - commands (List<String>) – список команд макросу.
- Bookmark – модель для опису закладок. Поля:
  - id (Long) – ідентифікатор закладки.
  - lineNumber (int) – номер рядка.
  - description (String) – опис закладки.

3. Зв'язки між класами

- File і Snippet – сніппети можуть бути використані у файлі для швидкої вставки шаблонного тексту.
- File і Macro – макроси можуть бути застосовані для редагування тексту.
- File і Bookmark – закладки дозволяють зберігати важливі позиції у тексті.
- File і SyntaxHighlighter – кожен файл може мати підсвічування синтаксису.
- Macro і Command – макрос складається з набору команд.

4. Репозиторії

- FileRepository – забезпечує CRUD-операції для моделі File:
  - save(File file)
  - findById(Long id)
  - delete(Long id)
  - findAll()
- SnippetRepository – CRUD-операції для моделі Snippet:
  - save(Snippet snippet)
  - findById(Long id)
  - delete(Long id)
  - findAll()
- MacroRepository – CRUD-операції для моделі Macro:
  - save(Macro macro)
  - findById(Long id)
  - delete(Long id)
  - findAll()
- BookmarkRepository – CRUD-операції для моделі Bookmark:
  - save(Bookmark bookmark)
  - findById(Long id)
  - delete(Long id)
  - findAll()

5. Utility-класи

- FileUtils – утиліти для роботи з файлами:
  - detectEncoding(String path): String – визначає кодування файлу.
  - readFile(String path): String – читає файл із заданого шляху.
  - writeFile(String path, String content): void – записує текст у файл.
- SyntaxHighlighter – клас для підсвічування синтаксису:
  - highlight(String language, String content): String – підсвічує текст для заданої мови програмування.
  - setTheme(String theme): void – змінює тему підсвічування.
- CommandExecutor – утиліти для виконання команд:
  - execute(String command): void – виконує одну команду.
  - batchExecute(List<String> commands): void – виконує список команд.

6. База даних та з'єднання

- DatabaseConnection – клас для управління з'єднанням із базою даних:
  - getConnection() – встановлює та повертає з'єднання з базою даних.
  - closeConnection(Connection connection) – закриває активне з'єднання.
  -

Загальна структура

1. Моделі – описують основні об’єкти текстового редактора, такі як файли, сніппети, макроси та закладки.
2. Репозиторії – забезпечують CRUD-операції для моделей, забезпечуючи розділення логіки доступу до даних і бізнес-логіки.
3. Utility-класи – допоміжні класи для виконання операцій з файлами, підсвічуванням синтаксису та виконанням команд.
4. Зв’язки – моделі інтегровані між собою, забезпечуючи повну функціональність текстового редактора.

Структура бази даних

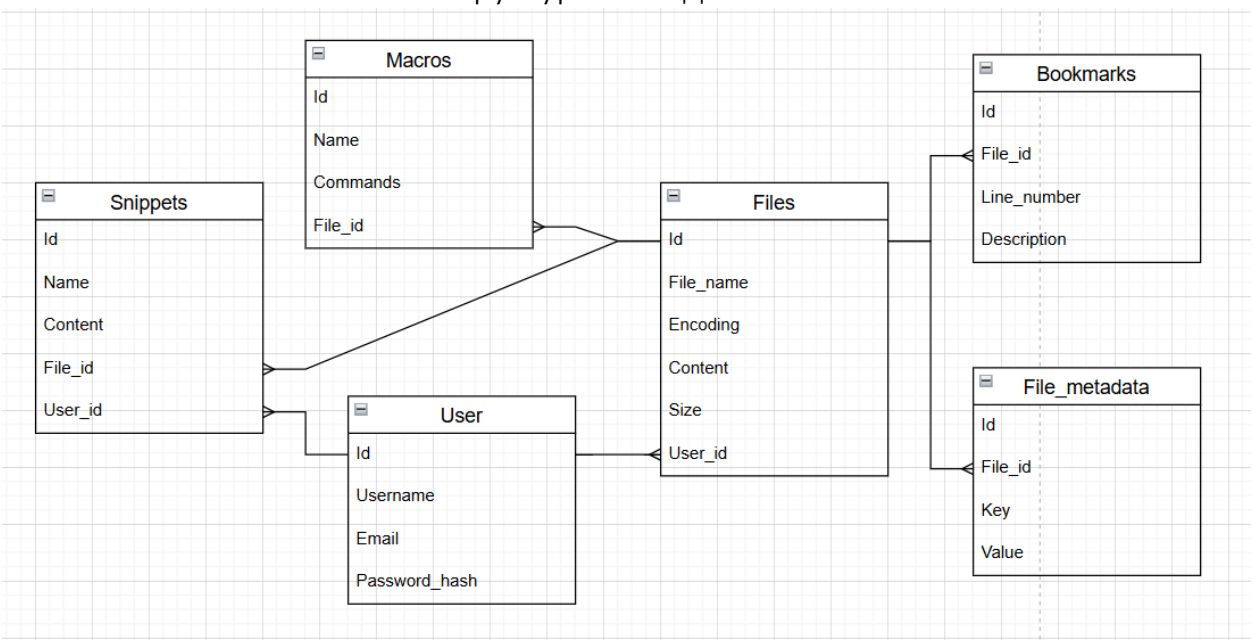

Рис 3. Структура бази даних

Висновок**:** Під час виконання лабораторної роботи я ознайомилася з теоретичними відомостями та розробила прецеденти, діаграми класів та структуру бази даних для системи керування завданнями. Діаграма прецедентів була створена для відображення основних сценаріїв використання системи, визначення ключових взаємодій між користувачами та системою, а також для забезпечення чіткого розуміння функціональних вимог. Діаграма класів була розроблена для моделювання основних об’єктів системи, їхніх властивостей та зв’язків між ними, що дозволяє структуровано підходити до проєктування системи. Структура бази даних забезпечила визначення необхідних таблиць, їхніх полів та зв’язків, що дозволяє ефективно зберігати та обробляти дані системи. У звіт включено всі необхідні компоненти, що відображають логіку, архітектуру та структуру даної системи, забезпечуючи її подальшу реалізацію.