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

Лабораторна робота № 2

Технології розроблення програмного забезпечення

«Основи проектування»

Виконала:
студентка групи ІА-32
Чайковська С. В.

Перевірив:
Мягкий М. Ю.

Київ 2025

Зміст

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.

1 Теоретичні відомості

Діаграма варіантів використання

Мова UML – це універсальна мова для візуального моделювання, яка використовується для специфікації, візуалізації, проєктування та документування компонентів програмного забезпечення, бізнес-процесів та інших систем. UML дозволяє будувати концептуальні, логічні та графічні моделі складних систем, застосовуючи найкращі методи програмної інженерії.

В рамках UML моделі складних систем представлені у вигляді графічних конструкцій, званих діаграмами. Серед основних видів діаграм UML:

- діаграма варіантів використання (use case diagram),
- діаграма класів (class diagram),
- діаграма послідовностей (sequence diagram),
- діаграма станів (statechart diagram),
- діаграма діяльності (activity diagram),
- діаграма компонентів (component diagram),
- діаграма розгортання (deployment diagram).

Діаграма варіантів використання є початковою концептуальною моделлю системи. Вона відображає загальні вимоги до функціональності системи і не деталізує її внутрішню структуру.

Основні завдання діаграми варіантів використання:

1. Визначення меж функціональності системи.
2. Формулювання загальних вимог до функціональної поведінки.
3. Розробка початкової концептуальної моделі системи.
4. Створення основи для подальшого аналізу, проєктування та тестування.

Елементи діаграми:

- Актор (actor) – об'єкт або користувач, який взаємодіє із системою.
- Варіант використання (use case) – дії або послуги, які система надає актору.
- Зв’язки (relationships) – відношення між акторами та варіантами використання.

Діаграми UML. Діаграми класів. Концептуальна модель системи

Діаграми класів найчастіше використовуються при моделюванні програмних систем (ПС). Вони є формою статичного опису системи, показуючи її структуру, але не відображають динамічну поведінку об’єктів. На діаграмах класів зображуються класи, інтерфейси та відносини між ними.

Представлення класів

Клас – це основний будівельний блок ПС. Клас має назву, атрибути та операції. На діаграмі клас показується у вигляді прямокутника, розділеного на три області:

- Верхня область – назва класу.
- Середня область – опис атрибутів (властивостей).
- Нижня область – назви операцій (послуг), що надаються об’єктами цього класу.

Атрибути класу визначають структуру даних, які зберігаються в об'єктах. Кожен атрибут має ім’я та тип, що визначає його значення в програмі.

Видимість атрибутів

Для атрибутів класу можна задати видимість:

- Відкритий (public) – доступний для будь-якого класу.
- Захищений (protected) – доступний лише для нащадків.
- Закритий (private) – недоступний ззовні і використовується тільки в класі.

Це дозволяє реалізувати інкапсуляцію даних, забезпечуючи захист від несанкціонованого доступу.

Операції класу

Клас містить визначення операцій, які об'єкти цього класу повинні виконувати. Кожна операція має сигнатуру з іменем, типом повернення та списком параметрів. Закриті операції є внутрішніми для об'єктів класу, в той час як відкриті формують інтерфейс класу.

Відносини на діаграмах класів

На діаграмах класів зазвичай показуються асоціації та об'єднання (наслідування):

1. Асоціація (Association) – відношення між об’єктами. Вона може мати назву та характеристику, таку як множинність, що показує, скільки об’єктів кожного класу може брати участь у зв’язку.
2. Об'єднання (Generalization) – показує зв'язок між класом-родителем та класом-нащадком. Цей зв'язок використовується для виявлення спільних характеристик кількох класів, які об’єднуються у батьківський клас.

Асоціації можуть також мати свої атрибути та операції, у цьому випадку вони називаються клас-асоціацією.

Взагалі, асоціація є загальним видом зв’язку між класами, відображаючи використання одного класу іншим.

Логічна структура бази даних

Існує дві моделі баз даних: фізична та логічна. *Фізична модель* зберігає дані у вигляді бінарних файлів, оптимізованих для зберігання та отримання інформації. *Логічна модель* відображає структуру таблиць, представлень, індексів та інших елементів для програмування і використання бази даних.

Процес проєктування бази даних полягає в побудові зв'язків між програмними класами та таблицями. Основою для проєктування таблиць є нормальні форми, що допомагають уникнути надмірності та аномалій оновлення.

Нормальні форми:

1. 1НФ – кожен атрибут відношення має одне значення.
2. 2НФ – всі неключові атрибути залежать від ключа.
3. 3НФ – немає транзитивних залежностей між неключовими атрибутами.
4. НФ Бойса-Кодда – кожна функціональна залежність базується на ключі.

2 Тема

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).

3 Діаграма варіантів використання

Діаграма варіантів використання зображена на рисунку 1:

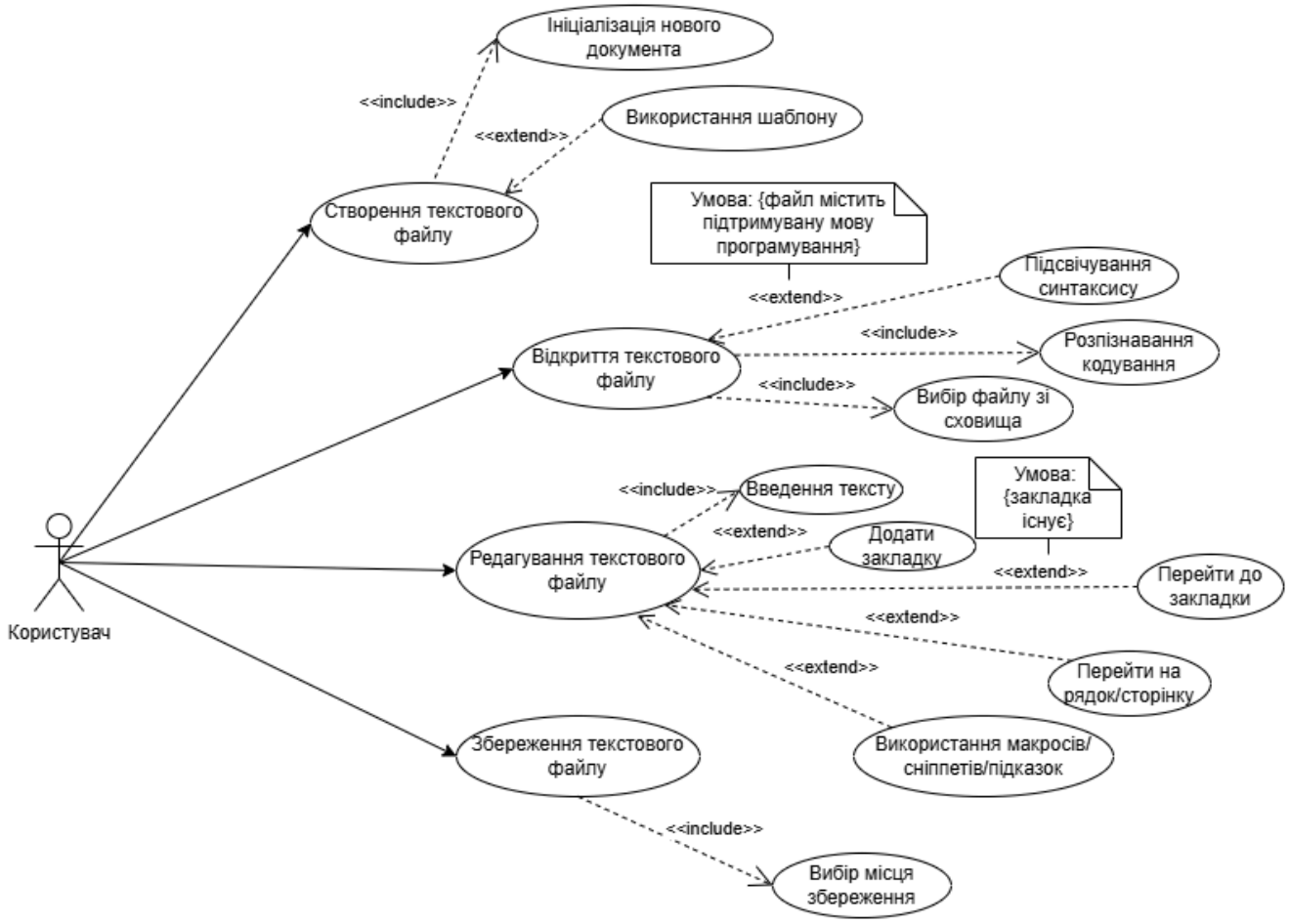


Рис. 1 – Діаграма варіантів використання

Ця діаграма є діаграмою варіантів використання (use case diagram) для текстового редактора. Вона відображає взаємодію користувача з основними функціями програми.

Основні елементи діаграми:

1. Актор:
    - Користувач – взаємодіє з текстовим редактором, ініціює всі основні дії.
2. Варіанти використання (Use Cases):
    - Створення текстового файлу
        - Ініціалізація нового документа (*include*)
        - Використання шаблону (*extend*)

- Відкриття текстового файлу
  - Вибір файлу зі сховища (*include*)
  - Розпізнавання кодування (*include*)
  - Підсвічування синтаксису (*extend*, якщо файл містить мову програмування)
- Редагування текстового файлу
  - Введення тексту (*include*)
  - Додавання закладки (*extend*)
    - Перехід до закладки (*extend*, якщо закладка існує)
  - Перехід на рядок/сторінку (*extend*)
  - Використання макросів/сніппетів/підказок (*extend*)
- Збереження текстового файлу
  - Вибір місця збереження (*include*)

3. Зв’язки між прецедентами:
   - <<include>> – обов’язкові дії, що завжди виконуються разом з основним прецедентом.
   - <<extend>> – додаткові дії, які можуть виконуватися за певних умов.

4 Сценарії використання для трьох прецедентів

Варіант використання 1:

Назва: Відкриття текстового файлу.

Передумови: Користувач запустив текстовий редактор.

Постумови: Файл відкритий у редакторі та готовий до перегляду/редагування.

Взаємодіючі сторони: Користувач, Текстовий редактор.

Короткий опис: Користувач відкриває існуючий текстовий файл із файлової системи.

Основний перебіг подій:

1. Користувач обирає опцію «Відкрити файл».

2. Система відкриває діалог вибору файлу.
3. Користувач вибирає потрібний файл.
4. Система визначає кодування файлу.
5. Система відкриває файл у вікні редактора.

Винятки:

- Виняток №1: Файл не знайдено→ система виводить повідомлення про помилку.
- Виняток №2: Невідоме кодування → система пропонує вибрати кодування самостійно.

Примітки: Редактор підтримує автоматичне розпізнавання різних систем кодування (UTF-8, Windows-1251 тощо).

Варіант використання 2:

Назва: Збереження текстового файлу.

Передумови: У редакторі відкрито документ (створений або відредагований).

Постумови: Файл збережено у сховищі у вибраному місці.

Взаємодіючі сторони: Користувач, Текстовий редактор.

Короткий опис: Користувач зберігає зміни у файлі.

Основний перебіг подій:

1. Користувач натискає кнопку "Зберегти".
2. Якщо файл уже збережений:
    - Зміни автоматично зберігаються без додаткових дій.
3. Якщо файл новий (без назви):
    - Відкривається діалогове вікно для введення назви файлу.
    - Користувач вводить ім'я файлу та натискає "Зберегти".

Винятки:

- Виняток №1: Недостатньо прав для запису у вибрану папку – система виводить повідомлення і пропонує інше місце збереження.
- Виняток №2: Виникла помилка при збереженні – система повідомляє користувача та пропонує повторити дію.

Примітки: Можливість збереження у різних форматах (TXT, інші).

Варіант використання 3:

Назва: Використання сніппетів.

Передумови: У редакторі відкрито текстовий файл для редагування.

Постумови: У текст додано готовий фрагмент.

Взаємодіючі сторони: Користувач, Текстовий редактор.

Короткий опис: Користувач вставляє готовий фрагмент коду або тексту (сніппет) для прискорення редагування.

Основний перебіг подій:

1. Користувач викликає меню сніппетів.
2. Система відображає список доступних сніппетів.
3. Користувач обирає потрібний сніппет.
4. Система вставляє обраний фрагмент у текст у поточній позиції курсора.

Винятки:

- Виняток №1: Список сніппетів порожній – система повідомляє користувача.
- Виняток №2: Вставка неможлива (наприклад, файл тільки для читання) – система виводить повідомлення про помилку.

Примітки: Відсутні

5 Діаграма класів

Діаграма класів зображена на рисунку 2:

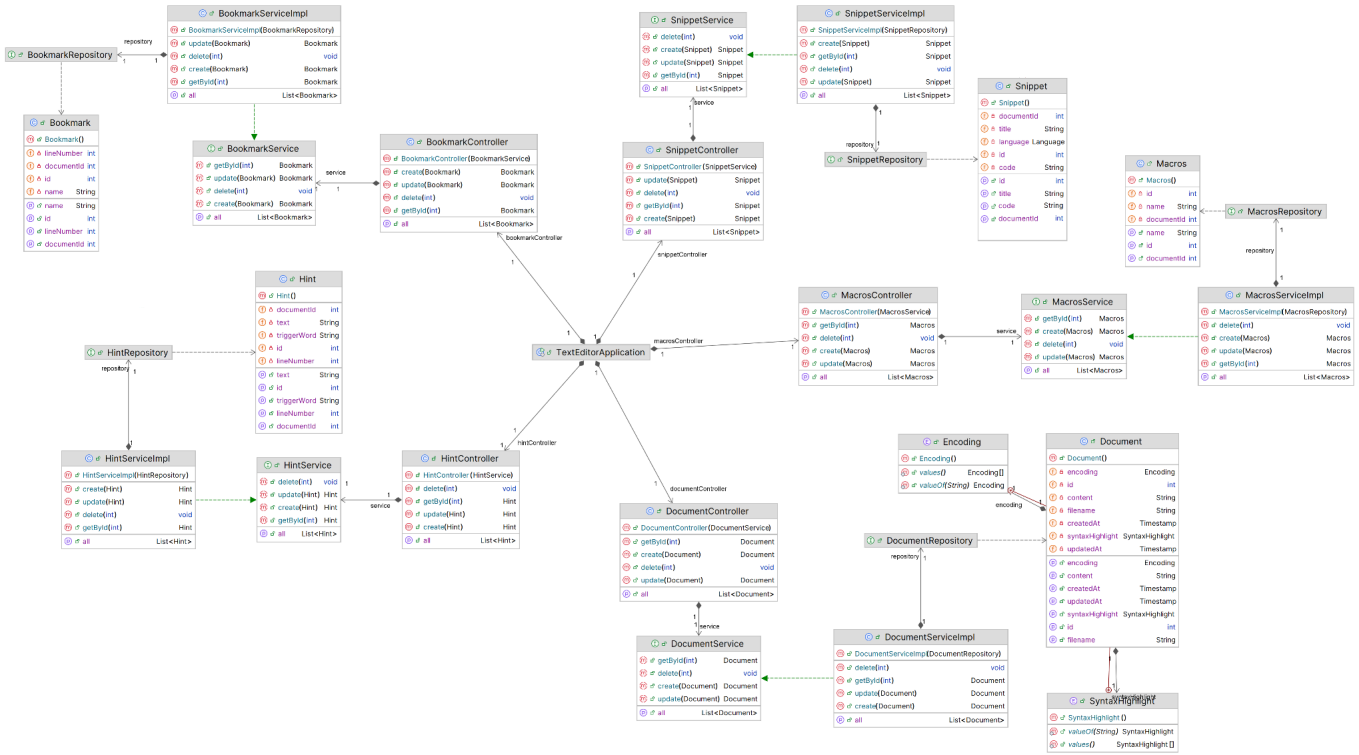


Рис. 2 – Діаграма класів

Ця діаграма класів відображає архітектуру текстового редактора з використанням кількох компонентів для управління документами, підказками, закладками, макросами та фрагментами коду. Діаграма містить такі основні елементи:

1. *DocumentController, HintController, BookmarkController, MacrosController, SnippetController* – це контролери, які керують відповідними функціональними частинами текстового редактора, такими як документи, підказки, закладки, макроси та фрагменти коду. Вони взаємодіють із сервісами для виконання CRUD-операцій (створення, читання, оновлення та видалення).

2. *DocumentServiceImpl, HintServiceImpl, BookmarkServiceImpl, MacrosServiceImpl, SnippetServiceImpl* – це реалізації сервісних класів, які

виконують основну бізнес-логіку для кожної з відповідних сутностей. Вони використовують репозиторії для зберігання та отримання даних.

3. *DocumentService, HintService, BookmarkService, MacrosService, SnippetService* – це сервіси, що визначають основні методи для маніпулювання даними (документи, підказки, закладки, макроси, фрагменти коду), такі як створення, оновлення, видалення, отримання за ідентифікатором та отримання всіх записів.

4. *DocumentRepository, HintRepository, BookmarkRepository, MacrosRepository, SnippetRepository* – це репозиторії, які забезпечують доступ до бази даних для кожної з сутностей.

5. *Document, Hint, Bookmark, Macros, Snippet* – це моделі даних, що описують властивості відповідних сутностей текстового редактора. Вони містять поля, які зберігають ключову інформацію (наприклад, id, name, documentId, code, lineNumber тощо).

6. *Encoding, SyntaxHighlight* – це допоміжні перерахування (enum), які забезпечують вибір типів кодування та підсвічування синтаксису.

7. *TextEditorApplication* – головний клас програми з методом main, що ініціалізує текстовий редактор і всі його компоненти.

Взаємозв'язки між контролерами, сервісами та репозиторіями показують потік даних та залежності між компонентами програми. Це забезпечує чітке розділення відповідальностей та модульність архітектури.

Шаблон Repository зображено на рисунку 3:

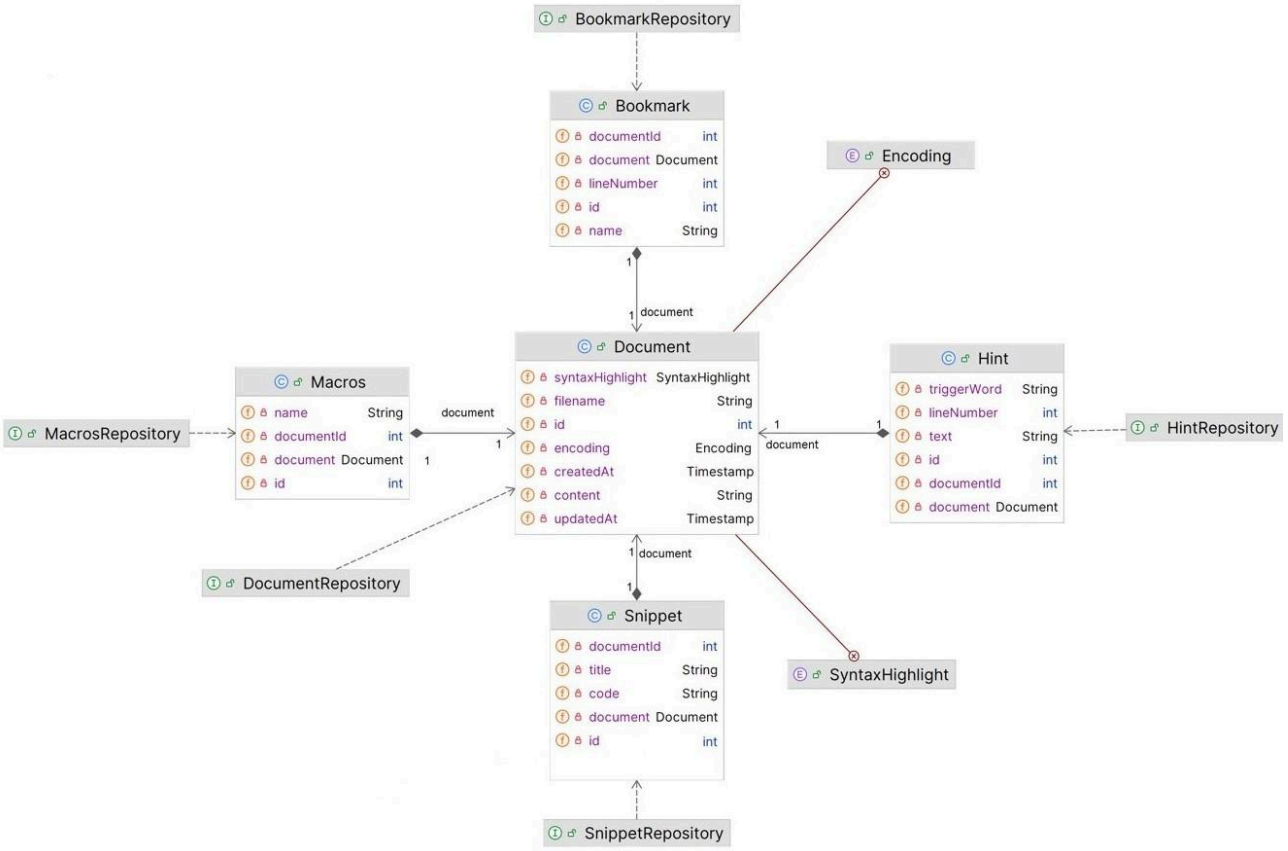


Рис. 3 – Шаблон Repository

Цей шаблон Repository представляє модель текстового редактора, що включає кілька основних сутностей та їх взаємозв'язки:

1. *Document* – центральна сутність, яка представляє документ, що редагується в текстовому редакторі. Зберігає інформацію про вміст, ім’я файлу, дату створення та оновлення, а також параметри підсвічування синтаксису та кодування. Всі інші сутності мають зв'язок із документом.

2. *Bookmark* – зберігає закладки, які пов'язані з певними рядками у документі. Кожна закладка належить одному документу.

3. *Hint* – підказки, що містять ключові слова, текст і номер рядка. Використовуються для допомоги при редагуванні. Кожна підказка належить одному документу.

4. *Macros* – макроси, які можуть бути застосовані до документа для автоматизації повторюваних дій. Кожен макрос прив’язаний до одного документа.

5. *Snippet* – фрагменти коду або тексту, що зберігаються для швидкого повторного використання. Кожен сніпет належить одному документу.

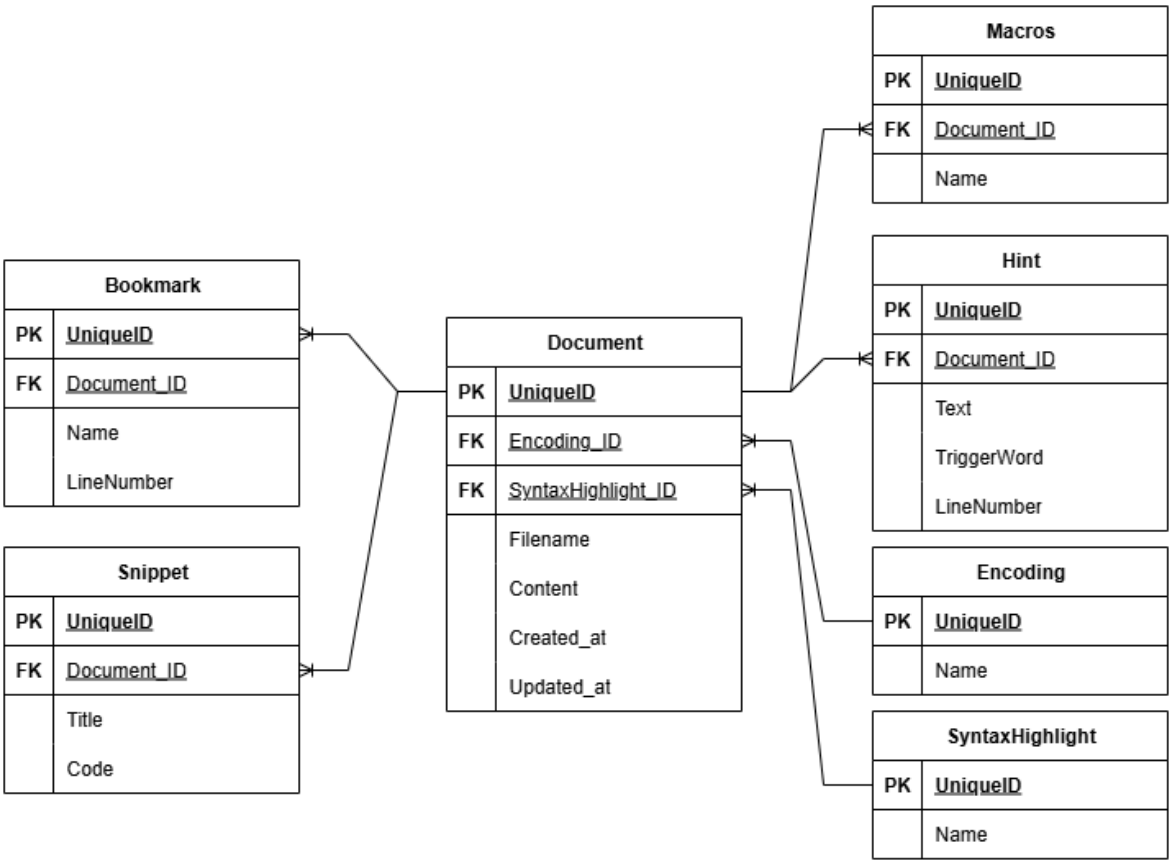
6. *Encoding* – визначає кодування, що використовується у документі. Кожен документ має одне конкретне кодування.

7. *SyntaxHighlight* – описує параметри підсвічування синтаксису, які застосовуються до документа.

8. Репозиторії *(DocumentRepository, BookmarkRepository, HintRepository, MacrosRepository, SnippetRepository)* – відповідають за управління і збереження відповідних сутностей у системі.

6 Структура бази даних

Структура бази даних зображена на Рисунку 4:



База даних складається з кількох основних таблиць, які мають тісні зв’язки між собою, що забезпечує ефективне управління документами та їх редагуванням.

1. *Document:* Це центральна таблиця, яка містить основну інформацію про текстові файли. У ній зберігаються назва файлу, вміст, параметри кодування та підсвічування синтаксису. Також є поля для фіксації дати створення та останнього оновлення. Всі інші таблиці пов’язані з нею через зовнішній ключ Document_ID.

2. *Macros:* Таблиця зберігає макроси, що дозволяють автоматизувати повторювані дії над документами. Кожен макрос має власний унікальний ідентифікатор і пов'язаний з конкретним документом за допомогою Document_ID.

3. *Snippet:* Використовується для збереження повторно використовуваних текстових або кодових фрагментів. Кожен сніпет має заголовок і код, а також прив'язується до конкретного документа через зовнішній ключ Document_ID.

4. *Bookmark:* У цій таблиці зберігаються закладки для документів, які дозволяють швидко знаходити певні рядки. Вона містить назву закладки та номер рядка, а також пов’язана з таблицею Document.

5. *Hint:* Таблиця призначена для зберігання підказок у вигляді ключових слів, тексту та номера рядка. Всі підказки відносяться до конкретного документа і допомагають користувачу під час редагування.

6. *Encoding:* Зберігає дані про можливі варіанти кодувань. Кожен документ пов’язаний з певним кодуванням через Encoding_ID.

7. *SyntaxHighlight:* Містить інформацію про режими підсвічування синтаксису, що можуть застосовуватись до документів. Зв’язок із таблицею Document встановлюється через SyntaxHighlight_ID.

Загалом, така структура бази даних дозволяє ефективно організувати та управляти документами, закладками, макросами, сніпетами та підказками. Вона

забезпечує зручність використання текстового редактора, цілісність даних та швидкий доступ до потрібної інформації.

Посилання на GitHub: https://github.com/chaikovsska/TextEditor

7 Висновки

У процесі виконання лабораторної роботи ми ознайомилися з теоретичним матеріалом, проаналізували тему, намалювали діаграму варіантів використання, діаграму класів, розробили основні класи та структуру бази даних.

8 Відповіді на питання

1. Що таке UML?

UML (Unified Modeling Language) – це універсальна мова візуального моделювання, яка використовується для опису, проєктування та документування програмних систем, бізнес-процесів та інших складних систем.

2. Що таке діаграма класів UML?

Діаграма класів – це статичне подання структури системи, яке показує класи, їх атрибути, операції та відносини між ними.

3. Які діаграми UML називають канонічними?

До канонічних діаграм UML належать:

- варіантів використання (use case diagram);
- класів (class diagram);
- кооперації (collaboration diagram);
- послідовності (sequence diagram);
- станів (statechart diagram);
- діяльності (activity diagram);
- компонентів (component diagram);
- розгортання (deployment diagram).

4. Що таке діаграма варіантів використання?

Це концептуальна модель системи, яка показує зовнішніх користувачів (акторів) та функціональність, яку надає система (варіанти використання).

5. Що таке варіант використання?

Варіант використання – це опис дії або послуги, яку система надає актору.

6. Які відношення можуть бути відображені на діаграмі використання?

На діаграмі використання відображаються відносини між акторами та варіантами використання (зв’язки асоціації).

7. Що таке сценарій?

Сценарій – це конкретна послідовність дій та подій, що описує виконання певного варіанту використання.

8. Що таке діаграма класів?

Діаграма класів – це графічне відображення класів системи, їх атрибутів, методів та зв’язків між ними.

9. Які зв’язки між класами ви знаєте?

Основні зв’язки:

- асоціація,
- агрегація,
- композиція,
- узагальнення (наслідування).

10. Чим відрізняється композиція від агрегації?

Агрегація – це відношення типу “частина–ціле” зі слабким зв’язком, коли об’єкт-частина може існувати незалежно від об’єкта-цілого, тоді як композиція – це сильніший зв’язок, при якому частина не може існувати без цілого.

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

Зв’язки типу агрегації на діаграмах класів відображають відношення частина–ціле зі слабким зв’язком, коли об’єкти одного класу входять до складу іншого (наприклад, список і його елементи) і зазвичай реалізуються як

посилання на об’єкт, тоді як композиція також позначає відношення частина–ціле, але з сильнішим зв’язком, коли частина не може існувати без цілого, і в коді зазвичай реалізується як змінна типу структури.

12. Що являють собою нормальні форми баз даних?

Це правила проєктування таблиць, що забезпечують відсутність надмірності та аномалій:

- 1НФ – кожен атрибут має одне значення,
- 2НФ – усі неключові атрибути залежать від ключа,
- 3НФ – немає транзитивних залежностей,
- НФ Бойса-Кодда – усі функціональні залежності базуються на ключах.

13. Що таке фізична модель бази даних? Логічна?

- Фізична модель – це спосіб збереження даних у вигляді бінарних файлів, оптимізованих для продуктивності.
- Логічна модель – це опис таблиць, атрибутів, зв’язків, індексів і представлень, які використовуються у програмуванні.

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

Програмні класи відображаються у таблицях БД: класи з атрибутами відповідають таблицям із полями, а зв’язки між класами реалізуються через зовнішні ключі в таблицях.