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

Тема: «Основи проектування»

Виконала:
Студентка групи ІА-34
Сизоненко А.О
Перевірив:
Мягкий Михайло Юрійович

Київ-2025

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.

Тема роботи: 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

UML (Unified Modeling Language) — це універсальна мова візуального моделювання для специфікації, візуалізації, проєктування та документування систем. Вона дозволяє будувати концептуальні, логічні та фізичні моделі складних систем, враховуючи об'єктно-орієнтований аналіз та проєктування (ООАП). Моделі поділяються на уявлення (views): статичні та динамічні, які деталізуються на різних рівнях абстракції (концептуальному, логічному, фізичному).

Діаграми UML — основний спосіб представлення моделей графічно. Основні види діаграм: варіанти використання (use case), класи (class), кооперації (collaboration), послідовності (sequence), станів (statechart), діяльності (activity), компонентів (component), розгортання (deployment).

(Use Case Diagram)

Діаграма варіантів використання показує вимоги до системи та її функціональні можливості. Вона визначає межі системи та є вихідною концептуальною моделлю, яка потім використовується для створення інших діаграм (наприклад, діаграми класів).

Основні елементи:

Актор (actor) — зовнішній суб'єкт, який взаємодіє з системою (людина, програма, пристрій).

Варіант використання (use case) — послідовність дій, що система виконує для актора, зображується еліпсом.

Відносини визначають взаємозв'язки між елементами:

Асоціація — звичайний зв'язок між актором і варіантом використання (суцільна лінія, може бути спрямована або ненаправлена).

Узагальнення (generalization) — успадкування властивостей одного елемента іншим.

Залежність (dependency) — зміна одного елемента впливає на інший;

Include — один варіант використання включає інший;

Extend — один варіант розширює базовий за певних умов.

Сценарії використання — покрокові текстові описи процесів взаємодії користувачів і системи. Містять передумови, постумови, основний потік подій, винятки та примітки.

Діаграми класів

Діаграми класів показують статичну структуру системи, включаючи класи, атрибути, операції та відносини між ними.

Атрибути мають видимість: public (+), package (~), protected (#), private (-).

Відносини:

1. **Асоціація** — зв’язок між класами, може мати назву та множинність;
2. **Узагальнення** — спадкування властивостей;
3. **Агрегація** — відношення частина-ціле (слабке);
4. **Композиція** — відношення частина-ціле (тісне, елементи не існують без цілого).

Діаграми класів використовуються для моделювання даних, архітектури, логіки програмних компонентів і навігації екранів.

Логічна структура бази даних

Бази даних мають **логічну** та **фізичну** моделі:

Фізична модель — організація даних у файлах на диску;

Логічна модель — структура таблиць, зв’язків, індексів для програмного використання.

Нормальні форми забезпечують мінімальну надмірність та уникнення логічних помилок:

1. 1НФ — кожен атрибут містить одне значення;
2. 2НФ — кожен неключовий атрибут повністю залежить від ключа;
3. 3НФ — відсутні транзитивні функціональні залежності неключових атрибутів від ключових;

Хід роботи

1. Діаграма варіантів використання. Use Case diagram

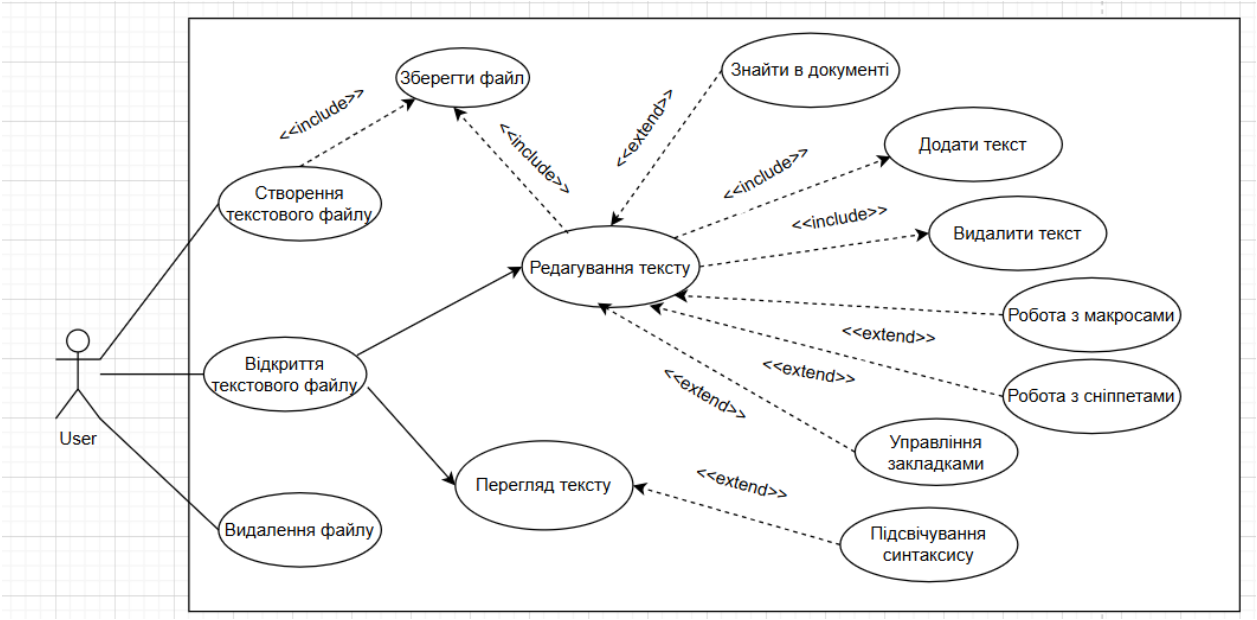


Рисунок 1 – Use Case діаграма

2. Розробка діаграми класів для предметної області.

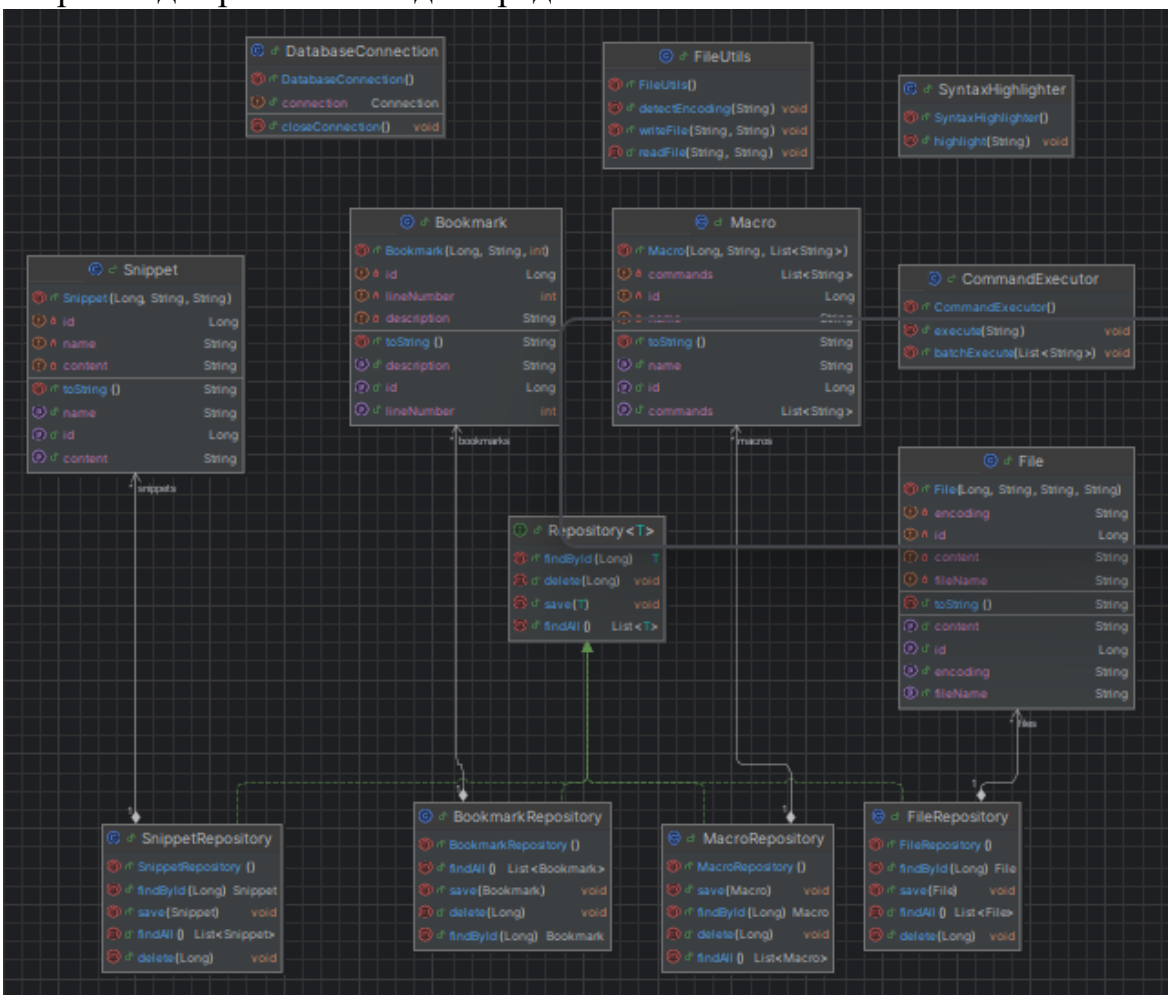


Рисунок 2 –діаграма класів

3. Сценарії використання

Сценарій 1 “Редагування текстового файлу”

Сторони взаємодії: користувач, текстовий редактор
Передумови: користувач має доступ до редагування текстового файлу.
Основний сценарій подій:

1. Користувач відкриває або створює файл.
2. Вносить зміни до тексту (додає, видаляє, додає закладки)
3. Використовує додаткові функції (додає сніппети, використовує макроси або пошук по документу)
4. Зберігає внесені до файлу зміни.

Постумови: Файл містить відредагований текст.

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.

Сценарій 2 “Робота з макросами”
Сторони взаємодії: користувач, текстовий редактор
Передумови: користувач відкрив документ для редагування.
Основний сценарій:

1. Користувач запускає функцію «Робота з макросами».
2. Записує послідовність дій (наприклад, вставку шаблонного тексту).
3. Зберігає макрос.
4. Виконує макрос у документі.

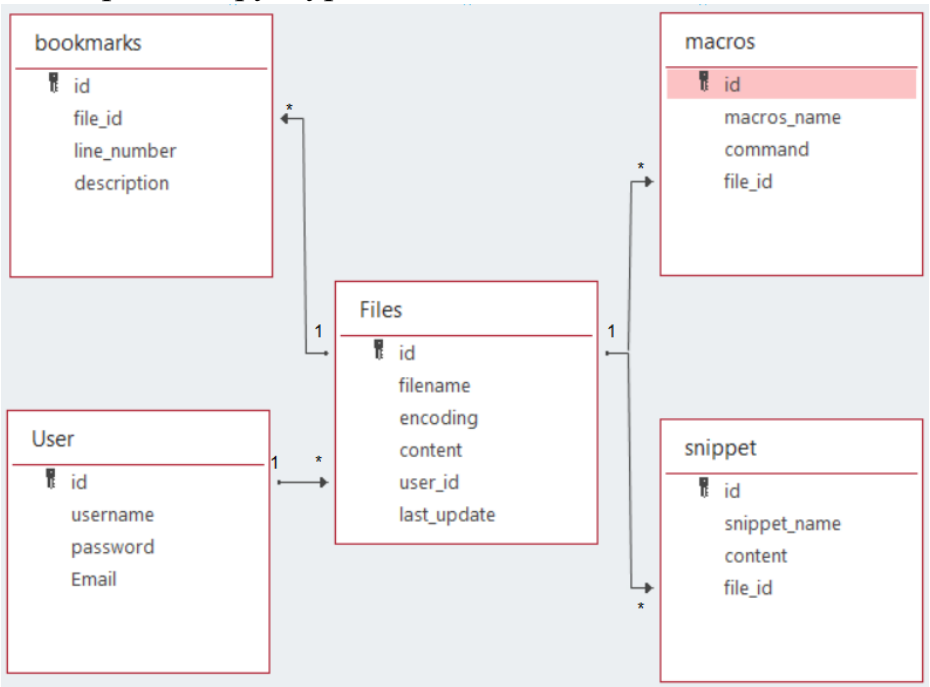
Альтернативи: Користувач редагує/видаляє існуючий макрос.
Постумови: Збережено макроси, які можна застосовувати для автоматизації подальшої роботи.

Сценарій 3 “Управління закладками”
Сторони взаємодії: користувач, текстовий редактор
Передумови: користувач відкрив документ для редагування.
Основний сценарій:

1. Користувач переходить до потрібного місця тексту.
2. Активує функцію додавання закладок через контекстне меню.
3. Додає закладку.
4. Користувач переходить до існуючої закладки.

Альтернативи: Користувач редагує/видаляє існуючу закладку.
Постумови: Закладку збережено та можна використати для переходу до конкретного місця в документі.

4. Розробка структури бази даних.



User

id — унікальний ідентифікатор користувача (PK).
username — ім’я користувача.
password — хеш пароля.
email — електронна пошта.

Files

id — унікальний ідентифікатор файлу (PK).
filename — назва файлу.
encoding — кодування файлу.
content — вміст файлу (текст).
user_id — зовнішній ключ на User.id (власник/автор).
last_update — дата/час останнього редагування.

bookmarks

id — унікальний ідентифікатор закладки (PK).
file_id — зовнішній ключ на Files.id (до якого файлу прив’язана).
line_number — номер рядка в файлі.
description — короткий опис або примітка.

snippet

id — унікальний ідентифікатор сніпета (PK).
snippet_name — назва сніпета.
content — текст або код сніпета.
file_id — зовнішній ключ на File.id.

macros

id — унікальний ідентифікатор макроса (PK).
macros_name — назва макроса.
command — команда або опис дії, що виконує макрос.

file_id — зовнішній ключ на File.id.

Опис зв’язків

User 1 → N

File — один користувач може мати багато файлів.

File 1 → N

bookmarks — у файлі може бути багато закладок.

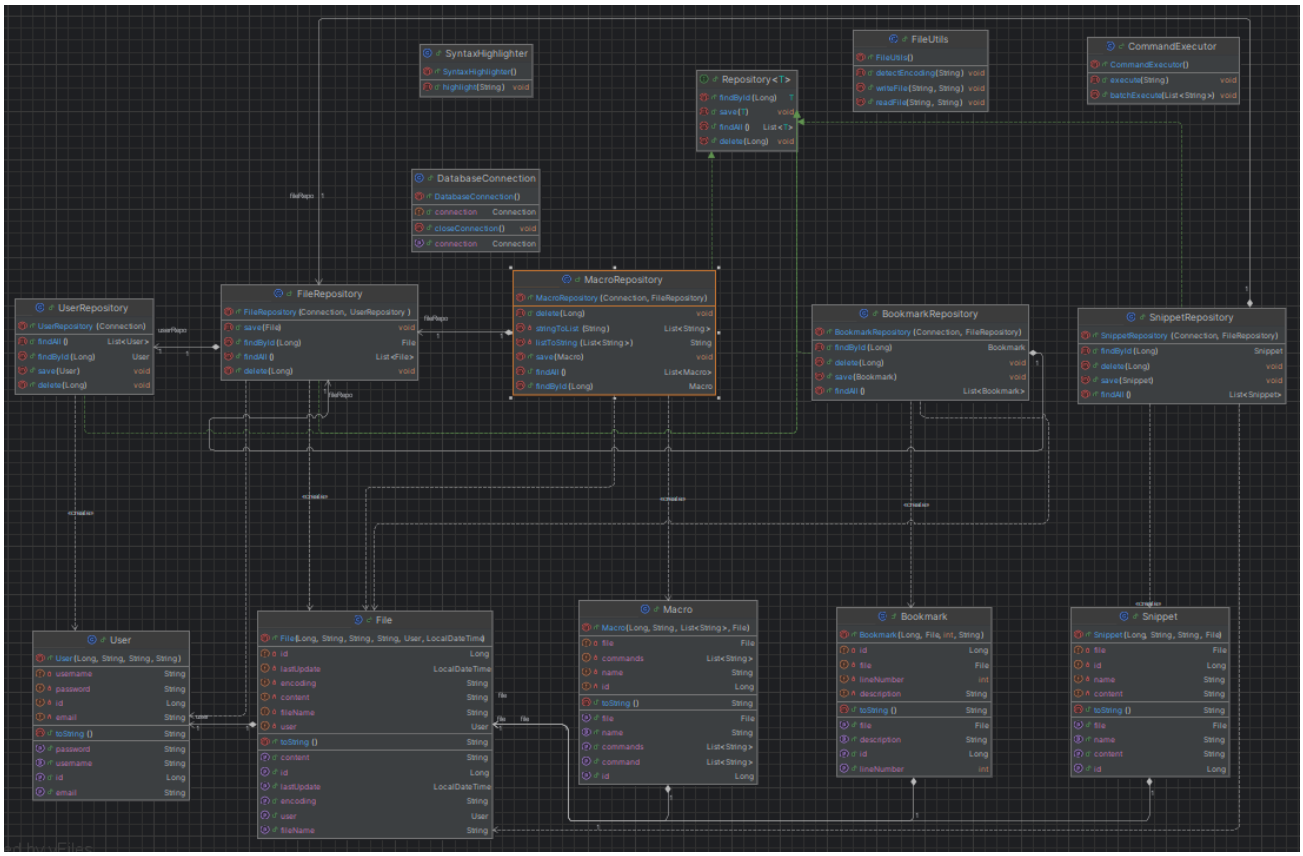
File 1 → N

snippet — у файлі можуть бути багато сніпетів.

File 1 → N

macros — у файлі можуть бути багато макросів.

5. Діаграма класів для реалізованої частини системи.



Система складається з користувачів, файлів та допоміжних сутностей. Користувач має множину файлів, тоді як кожен файл належить лише одному користувачу. Файл може містити множину макросів, закладок і сніпетів, причому кожен із цих елементів прив’язаний до конкретного файлу. Для управління сутностями передбачені окремі репозиторії, що забезпечують операції створення, пошуку та збереження даних.

Користувач → Файл: один користувач має багато файлів; файл належить одному користувачу.

Файл → Макроси: один файл містить багато макросів; макрос належить одному файлу.

Файл → Закладки: один файл містить багато закладок; закладка належить одному файлу.

Файл → Сніпети: один файл містить багато сніпетів; сніпет належить одному файлу.

Висновок: У лабораторній роботі я опрацювала теоретичні відомості та створила діаграму прецедентів, діаграми класів і структуру бази даних для системи. Діаграма прецедентів відображають основні сценарії та взаємодію користувачів із системою, діаграми класів показують об’єкти та їхні зв’язки, а структура бази даних визначає таблиці та відносини для зберігання даних. Це забезпечує чітку основу для реалізації системи.