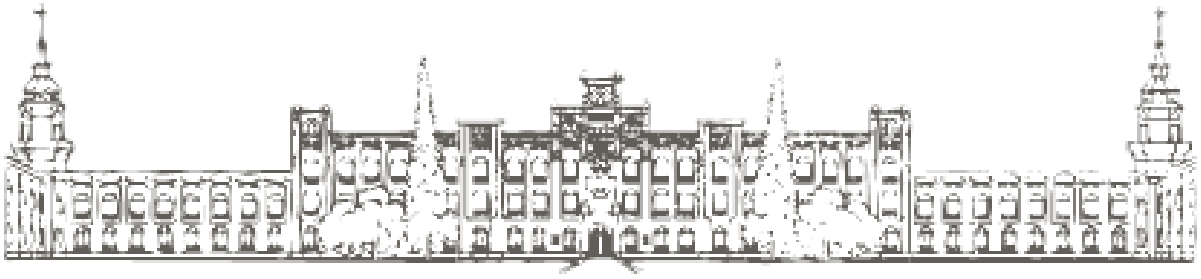


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Звіт лабораторної роботи №2

з курсу

«Технології розроблення програмного забезпечення»

Виконавець:
Нікітченко Наталя Олегівна
студентка групи ІА-33
залікова книжка № ІА-3318

«25» 10 2025 р.

Перевірив: **Мягкий М. Ю.**

Київ – 2025

2. ЛАБОРАТОРНА РОБОТА № 2

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.

Завдання:

• Ознайомитись з короткими теоретичними відомостями.

• Проаналізувати тему та спроєктувати діаграму варіантів використання відповідно до обраної теми лабораторного циклу.

• Спроєктувати діаграму класів предметної області.

• Вибрати 3 варіанти використання та написати за ними сценарії використання.

• На основі спроєктованої діаграми класів предметної області розробити основні класи та структуру бази даних системи. Класи даних повинні реалізувати шаблон Repository для взаємодії з базою даних.

• Нарисувати діаграму класів для реалізованої частини системи.

• Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму варіантів використання відповідно, діаграму класів системи, вихідні коди класів системи, а також зображення структури бази даних.

Теоретичні відомості:

Діаграма варіантів використання (Use-Cases Diagram)

Діаграма варіантів використання (Use-Cases Diagram) – це UML діаграма за допомогою якої у графічному вигляді можна зобразити вимоги до системи, що розробляється .

Діаграма варіантів використання – це вихідна концептуальна модель проєктованої системи, вона не описує внутрішню побудову системи.
Діаграми варіантів використання призначені для:

• визначення загальної межі функціональності проєктованої системи;

• формулювання загальних вимоги до функціональної поведінки проєктованої системи;

• подальшої розробка вихідної концептуальної моделі системи (діаграми класів);

• створення основи для виконання аналізу, проєктування, розробки та тестування.

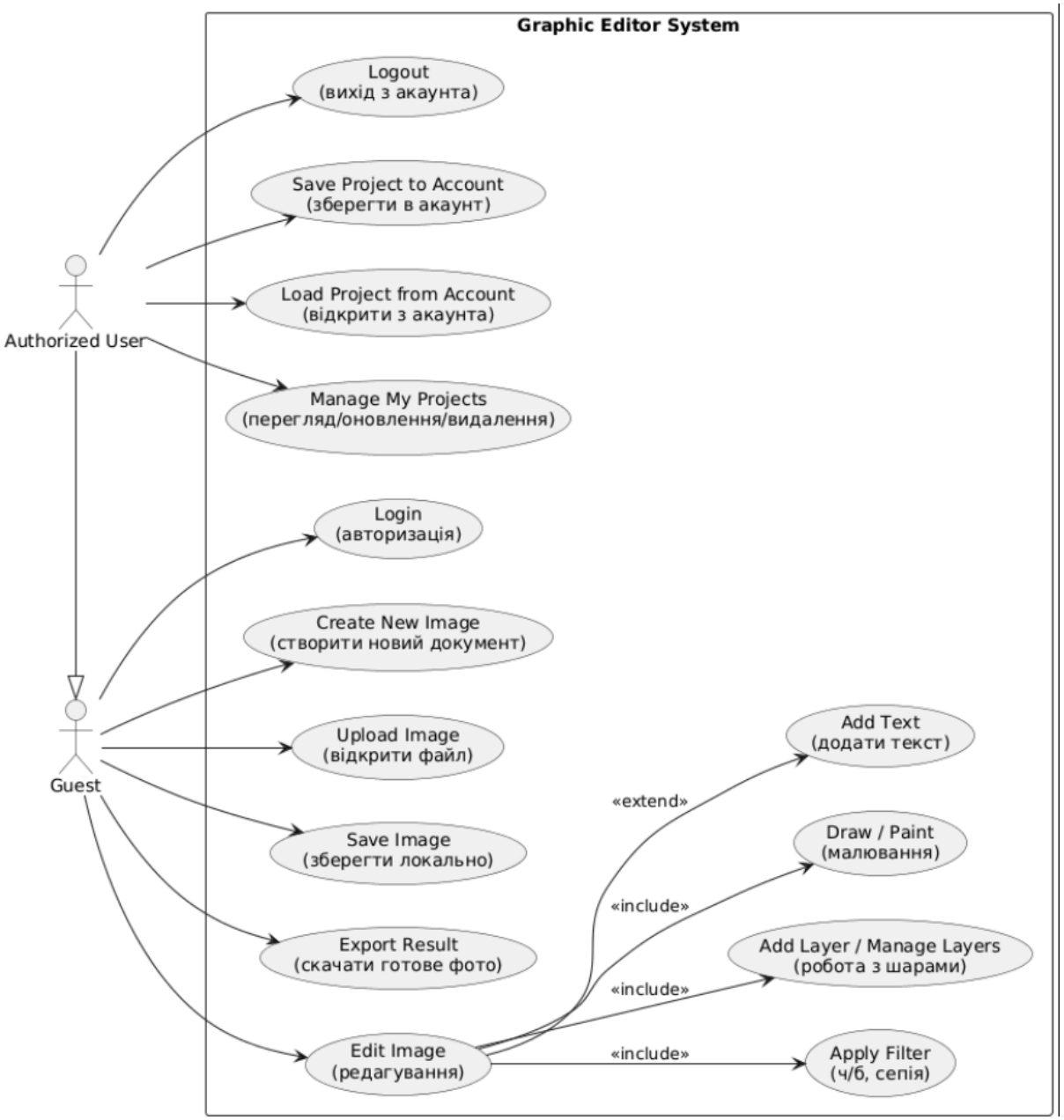
Діаграми варіантів використання є відправною точкою при збиранні вимог до програмного продукту та його реалізації. Дана модель будується на аналітичному етапі побудови програмного продукту (збір та аналіз вимог) і дозволяє бізнес-аналітикам отримати більш повне уявлення про необхідне програмне забезпечення та документувати його. Діаграма варіантів використання складається з низки елементів. Основними елементами є: варіанти використання або прецеденти (use case), актор або дійова особа (actor) та відносини між акторами та варіантами використання (relationship).

Діаграми класів

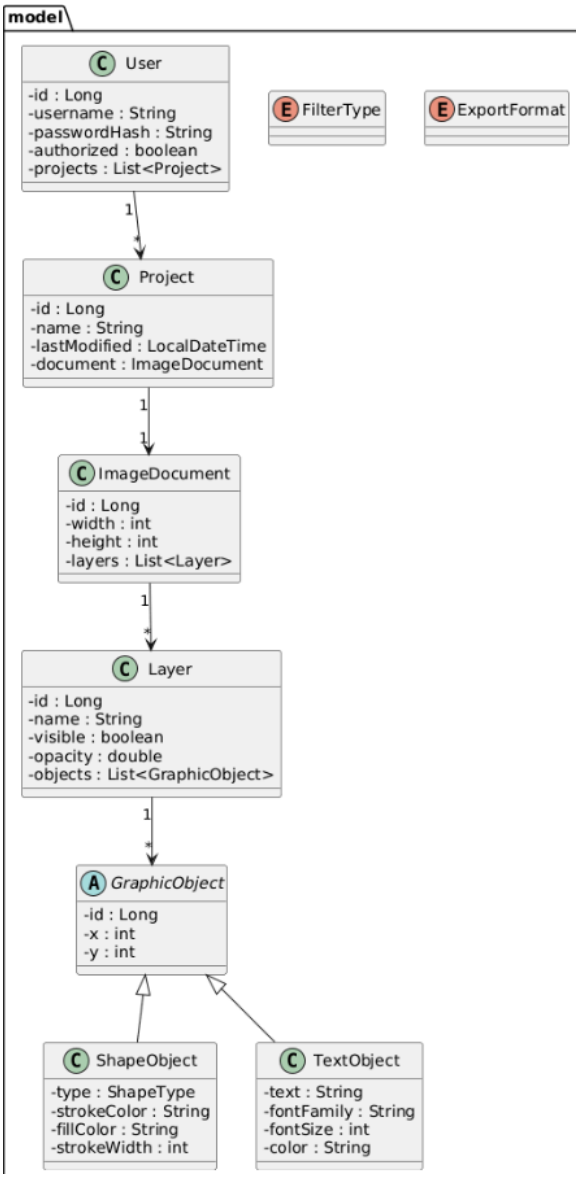
Діаграми класів використовуються при моделюванні програмних систем найчастіше. Вони є однією із форм статичного опису системи з погляду її проєктування, показуючи її структуру. Діаграма класів не відображає динамічної поведінки об'єктів зображених на ній класів. На діаграмах класів показуються класи, інтерфейси та відносини між ними.

Хід роботи:

Діаграма варіантів використання **(Use-Cases Diagram):**



Діаграма класів предметної області**:**



Сценарії використання:

Сценарій 1:

Створення нового проєкту та полотна

Передумови: користувач (User) авторизований у системі.

Постумови: у користувача з’явився новий Project із прив’язаним ImageDocument; створено стартовий Layer (наприклад "Background" або "Layer 1"); проєкт стає доступним у списку користувача.

Короткий опис: користувач ініціює створення нового проєкту, задає параметри майбутнього зображення (розміри полотна), система створює структуру об’єктів (Project → ImageDocument → Layer) і відкриває робочу область для редагування.

Основний перебіг подій:

1. Варіант використання починається, коли користувач натискає «Створити новий проєкт» / «New Image».
2. Система запитує базові параметри документа: ім’я проєкту, ширину та висоту зображення.
3. Користувач вводить назву (наприклад "Лабораторна 3") і задає розміри полотна (наприклад, 800×600 px).
4. Система створює новий об’єкт Project і прив’язує його до поточного користувача (User → Project).
5. Система створює порожній ImageDocument з указаними width/height і прив’язує його до Project.
6. Система автоматично створює перший Layer (видимий, opacity = 1.0) і додає його до ImageDocument.
7. Система відкриває інтерфейс редагування: полотно, панель шарів, панель інструментів (фігури, текст, тощо).

8. Проєкт позначається як "активний" для подальшої роботи (його lastModified фіксується поточним часом).

Винятки:

Виняток №1: Некоректні параметри документа.

Якщо користувач задав неможливі розміри (наприклад 0×0 або занадто велике значення, що не підтримується), система виводить повідомлення про помилку й просить виправити значення перед створенням.

Виняток №2: Користувач неавторизований.

Якщо користувач гостьовий (не збережений у User як authorized = true), система може дозволити створити тимчасовий документ без збереження в Project (наприклад "Untitled") і попереджає, що дані можуть не зберегтися після закриття.

Сценарій 2:

Редагування шару: додавання фігури та тексту

Передумови: існує активний Project з ImageDocument і хоча б одним Layer; користувач знаходиться в інтерфейсі редагування цього документа.

Постумови: у вибраний Layer додано нові GraphicObject — фігура (ShapeObject) і текст (TextObject); зміни відмічені як несохранені (dirty state), lastModified оновлено.

Короткий опис: користувач працює з шаром, додає графічні елементи (прямокутник, текстовий підпис), змінює їх властивості й бачить результат на полотні.

Основний перебіг подій:

1. Система відображає поточний ImageDocument (полотно) і список шарів. Активний шар позначений як вибраний.
2. Користувач обирає інструмент «Фігура» (Rectangle / Ellipse / Line).
3. Користувач тягне мишкою по полотну — система створює новий ShapeObject:
   – type (наприклад RECTANGLE),
   – координати x,y (де користувач намалював),
   – strokeColor, fillColor, strokeWidth.
4. Система додає цей ShapeObject у поточний Layer (Layer.objects → List<GraphicObject>).
5. Користувач обирає інструмент «Текст» і клацає по полотну.
6. Система пропонує ввести текст, параметри шрифту (fontFamily, fontSize) і колір.
7. Користувач вводить текст (наприклад "Звіт ЛР3") і підтверджує.
8. Система створює TextObject з цими властивостями і теж додає його до того ж Layer.
9. Система оновлює прев'ю полотна з урахуванням нового ShapeObject і TextObject.
10. Система позначає Project як змінений (оновлює поле lastModified).

Додаткові дії (опційно, якщо є фільтри):

11. Користувач може вибрати шар цілком і застосувати ефект (наприклад GRAYSCALE або INFRARED_STYLE).
12. Система фіксує застосований фільтр і оновлює відображення шару.

Винятки:

Виняток №1: Шар невидимий.

Якщо вибраний активний шар має visible = false, система або не дозволяє редагувати саме його, або автоматично робить його visible = true і показує

попередження.

Виняток №2: Заборона редагування.

Якщо шар заблокований (у майбутньому можна мати флаг “locked”), система показує повідомлення і не додає об’єкт.

Виняток №3: Вихід за межі полотна.

Якщо користувач намагається створити об’єкт за межами width/height ImageDocument, система або обрізає координати, або показує попередження.

Сценарій 3:

Експорт зображення у файл

Передумови: у поточного користувача відкритий Project з ImageDocument; у документі є хоча б один Layer з об’єктами.

Постумови: система згенерувала растрове зображення з урахуванням усіх видимих шарів і зберегла його у вибраному користувачем форматі (PNG/JPG/BMP). Копія файлу потрапляє на диск користувача. У проєкті зберігається оновлений lastModified.

Короткий опис: користувач зберігає результат роботи не як "проєкт з шарами", а як готове фінальне зображення для використання поза редактором.

Основний перебіг подій:

1. Варіант використання починається, коли користувач виконує команду «Експорт» або «Зберегти як зображення».
2. Система пропонує вибрати вихідний формат файлу (ExportFormat): PNG, JPG або BMP.
3. Користувач вибирає формат і місце на диску.

4. Система проходить по всіх шарах ImageDocument у правильному порядку (знизу вгору), бере тільки ті, що visible = true, враховує opacity кожного шару.
5. Система рендерить усі GraphicObject (і ShapeObject, і TextObject) у спільне растрове зображення.
6. Система записує результатний файл у вибрану локацію.
7. Система показує повідомлення «Експорт успішно завершено».
8. Project.lastModified оновлюється поточним часом.

Винятки:

Виняток №1: Неприпустимі параметри експорту.

Якщо користувач вибрав невідомий формат (не PNG/JPG/BMP) або не має прав на запис у вибрану директорію, система показує повідомлення про помилку і просить змінити формат або шлях.

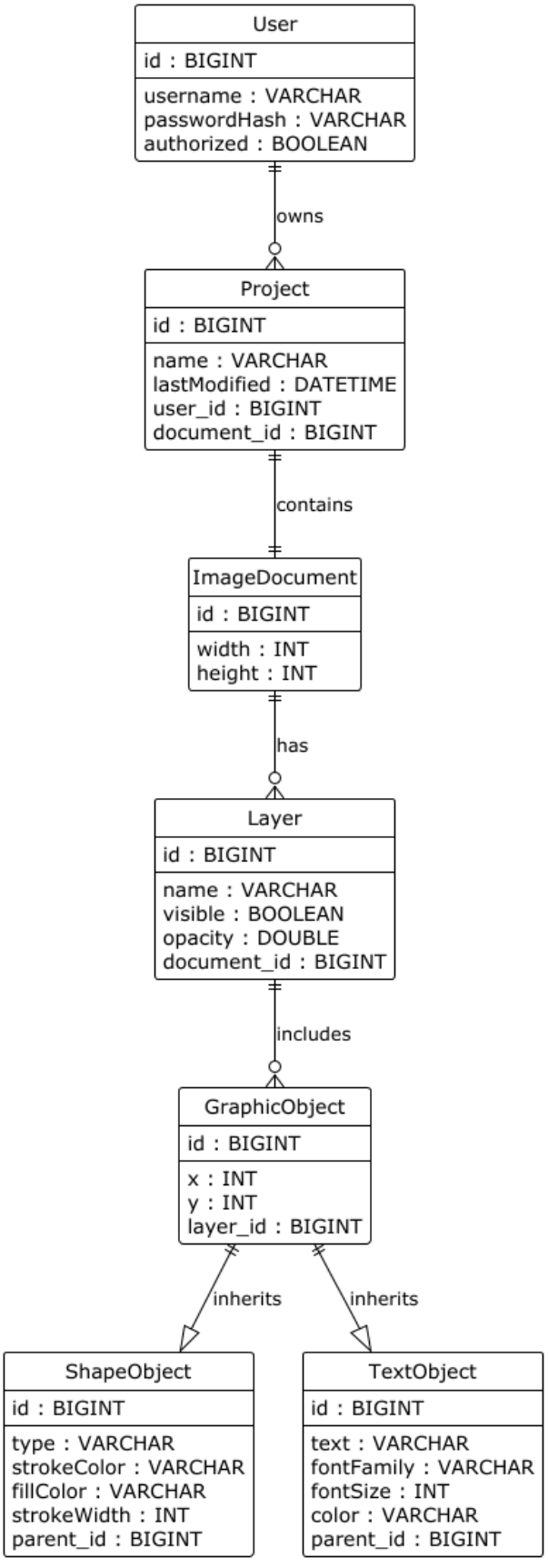
Виняток №2: Порожній документ.

Якщо в документі всі шари невидимі / порожні (нема об’єктів), система попереджає, що буде збережено порожнє зображення.

Виняток №3: Конфлікт імені файлу.

Якщо файл з такою назвою вже існує, система питає: «Перезаписати?»

Зображення структури бази даних:



Лістинг коду реалізованих класів системи:
https://github.com/niknat007/Graphiceditor

Висновки: у ході даної лабораторної роботи було проаналізовано тему та спроєктовано діаграму варіантів використання відповідно до обраної теми лабораторного циклу. Також, було спроєктовано діаграму класів предметної області. До діаграми варіантів використання системи було прописано 3 сценарії варіантів використання. Було візуалізовано структуру бази даних проєкту. Пророблена робота надала необхідні навички для роботи з документацією та плануванням. У результаті отримано модель, за якою можна працювати та покращувати проєкт.

Контрольні питання:

1. Що таке UML?

UML (Unified Modeling Language) — це уніфікована мова моделювання, що використовується для опису, проєктування та документування об’єктно-орієнтованих систем у вигляді діаграм.

2. Що таке діаграма класів UML?

Діаграма класів — це структурна діаграма UML, що показує класи системи, їх атрибути, методи та зв’язки (асоціації, наслідування, агрегації, композиції).

3. Які діаграми UML називають канонічними?

Канонічні (основні) діаграми UML — це стандартні діаграми, визначені UML-специфікацією: діаграма класів, варіантів використання, послідовностей, станів, діяльності, компонентів, розгортання.

4. Що таке діаграма варіантів використання?

Це поведінкова діаграма, що показує взаємодію між користувачами (акторами) і системою через варіанти використання (Use Cases).

5. Що таке варіант використання?

Варіант використання (Use Case) — це опис конкретної функції, яку система виконує у відповідь на дію користувача або іншої системи.

6. Які відношення можуть бути відображені на діаграмі використання?

Association (асоціація), Include (включення), Extend (розширення), Generalization (узагальнення).

7. Що таке сценарій?

Сценарій (use case scenario) — це послідовність кроків, що описує конкретний перебіг подій під час виконання варіанта використання.

8. Що таке діаграма класів?

Це графічне представлення структури системи у вигляді класів, їх атрибутів, операцій та зв’язків.

9. Які зв’язки між класами ви знаєте?

Асоціація, агрегація, композиція, наслідування, залежність, реалізація.

10. Чим відрізняється композиція від агрегації?

Агрегація — слабкий зв’язок «ціле-частина», частина може існувати окремо. Композиція — сильний зв’язок, частина не може існувати без цілого.

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

Агрегація позначається порожнім ромбом (◇), композиція — заповненим (◆). У композиції частина знищується разом із цілим.

12. Що являють собою нормальні форми баз даних?

Нормальні форми (НФ) — це правила структуризації таблиць для усунення надлишковості й забезпечення цілісності даних (1НФ, 2НФ, 3НФ, BCNF тощо).

13. Що таке фізична модель бази даних? Логічна?

Логічна модель описує структуру даних (таблиці, зв’язки), фізична — як ці дані зберігаються у конкретній СУБД (типи полів, індекси).

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

Кожна таблиця відповідає класу програми, а рядок таблиці — об’єкту. Це принцип ORM (Object-Relational Mapping).