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 №3

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Основи проектування розгортання»

Тема роботи: 7. Редактор зображень

Виконав
студент групи ІА–33
Марченко Вадим Олександрович

Київ 2025

Тема: Основи проектування розгортання.

Мета: Навчитися проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

Посилання на репозиторій: https://github.com/nwu1015/ImageEditor

Короткі теоретичні відомості

Діаграма розгортання (Deployment Diagram)

Діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонентів.

Діаграма компонентів (Component diagram)

Діаграма компонентів – в UML, діаграма, на якій відображаються компоненти, залежності та зв'язки між ними. Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись. Модуль програмного забезпечення може бути представлено як компоненту. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонентів відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Діаграма послідовностей (Sequence diagram)

Діаграма послідовностей Діаграма послідовності – різновид діаграми в UML. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність надісланих повідомлень.

Діаграма розгортання

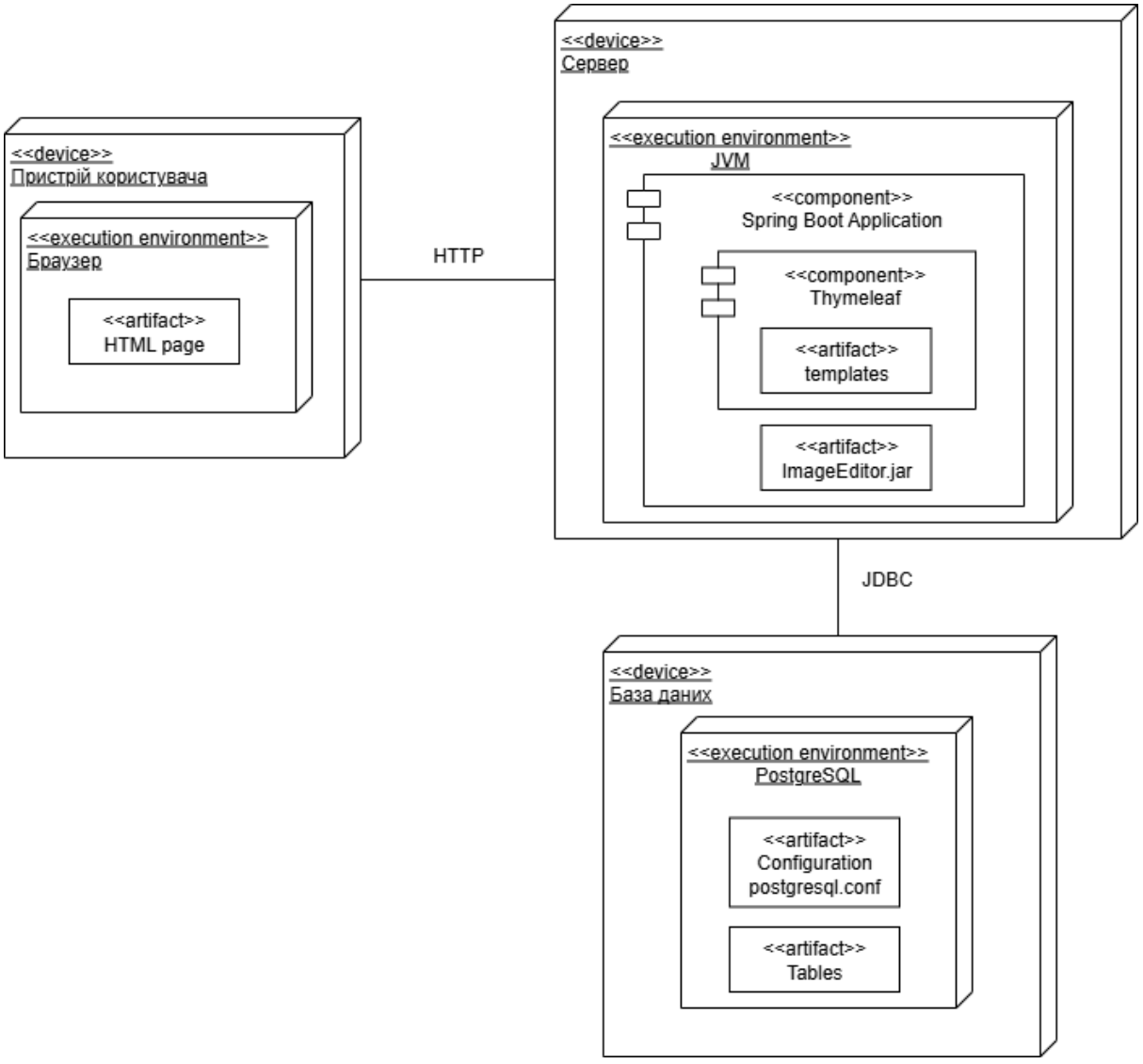

Рисунок 1. - Діаграма розгортання застосунку

Пристрій користувача:

На стороні користувача розташований веб-браузер як середовище виконання. Він відображає згенеровані сервером веб-сторінки (HTML, CSS, JavaScript) та надає інтерфейс для взаємодії з користувачем.

Фактично у браузері виконуються лише готові ресурси (артефакти), які надіслав сервер, – форми авторизації, вікно редагування зображень, інтерфейс застосування ефектів тощо.

Безпосередньо жодної бізнес-логіки чи шаблонів у браузері немає, він виступає лише у ролі клієнта.

Сервер додатку:

На сервері розташоване середовище виконання JVM (Java Virtual Machine), у якому працює компонент Spring Boot Application.

Цей застосунок реалізує всю бізнес-логіку редактора:

- завантаження та збереження зображень,
- застосування ефектів (поворот, масштабування, кадрування),
- створення колажів,
- керування історією змін,
- копіювання станів зображень.

У межах Spring Boot працює компонент Thymeleaf Engine, який відповідає за генерацію HTML-сторінок із шаблонів (templates/*.html). Завдяки цьому користувач отримує динамічний інтерфейс у браузері.

Основним артефактом, що розгортається на сервері, є ImageEditor.jar – скомпільований застосунок, який містить бізнес-логіку та всі необхідні модулі.

Сервер бази даних:

Третій вузол – це сервер бази даних із середовищем виконання PostgreSQL.

У ньому зберігаються два типи артефактів:

- Configuration – конфігураційні файли БД (postgresql.conf), які визначають параметри підключення та доступу;
- Tables – власне база даних image_editor_db, що містить таблиці з користувачами, зображеннями, історією редагувань і метаданими для створення колажів.

Взаємодії:

1. Пристрій користувача та сервер додатку:
- Протокол взаємодії: HTTP
- Призначення: Користувач через браузер надсилає запити на сервер (наприклад відкриття зображення, застосування ефектів).
- Дані, що передаються: 1) введені користувачем дані 2) параметри редагування (наприклад, обраний ефект, координати кадрування) 3) зображення для завантаження 4) відповіді від сервера у вигляді HTML-сторінок, JSON-даних чи готових зображень після обробки.

2. Сервер додатку та база даних:
- Протокол: PostgreSQL (TCP) через JDBC/JPA у Spring Boot.
- Призначення: Збереження та отримання даних, необхідних для роботи системи.
- Дані, що передаються: 1) SQL-запити від Spring Boot до PostgreSQL (наприклад, «отримати дані користувача», «зберегти інформацію про редаговане зображення») 2) результати виконання SQL-запитів (рядки з таблиць, статуси виконання).

Діаграма компонентів

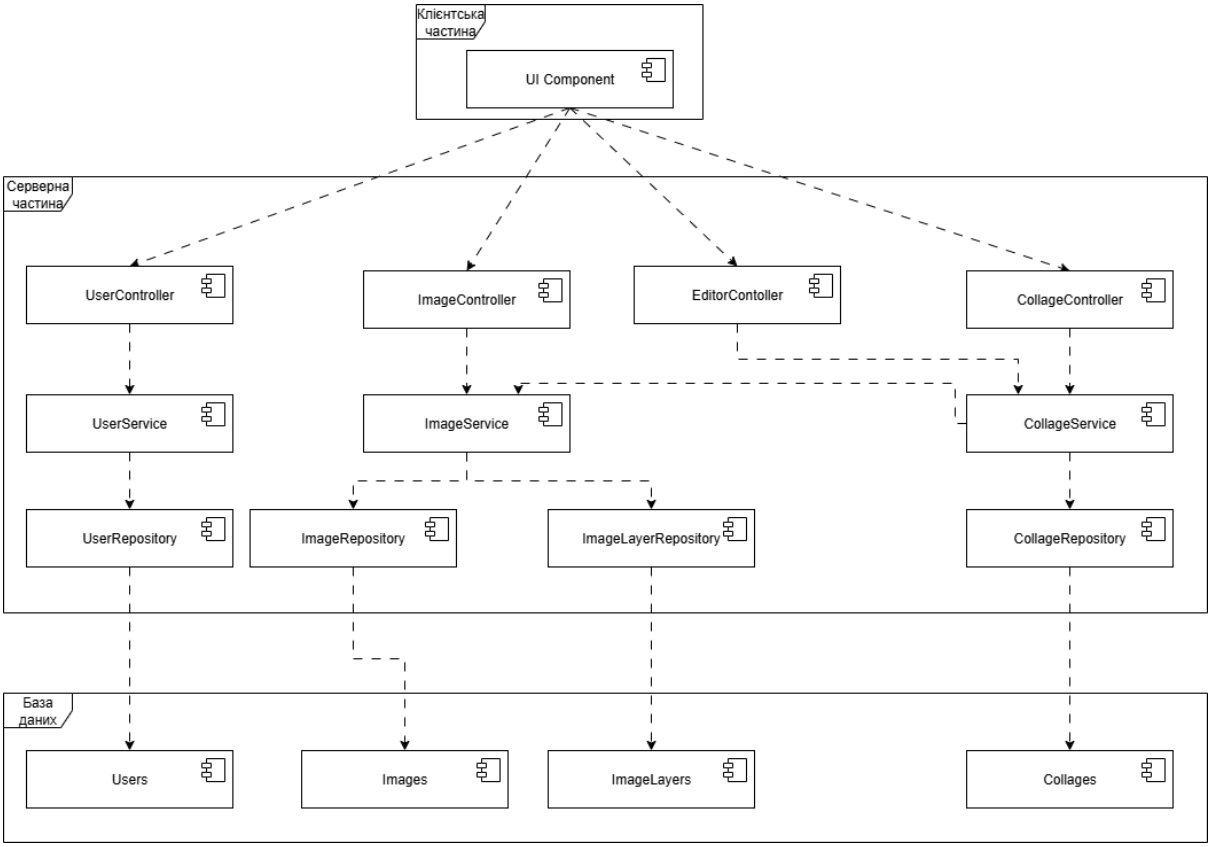

Рисунок 2. - Діаграма компонентів застосунку

У даній діаграмі компонентів представлена архітектура системи редагування зображень, яка побудована за трирівневою моделлю. Перший рівень – це користувач та інтерфейс, через який він взаємодіє із системою. Інтерфейс реалізовано у вигляді веб-сторінок, що відображаються у браузері. Користувач може завантажувати та переглядати зображення, створювати колажі, застосовувати ефекти, зберігати результати своєї роботи. Усі ці дії відправляються у вигляді запитів до контролерів, які виступають посередниками між інтерфейсом і бізнес-логікою.

Другий рівень – це серверна частина, де реалізована основна логіка роботи програми. Контролери викликають сервіси, які відповідають за обробку зображень, застосування ефектів, роботу з колажами чи користувачами. На цьому рівні також зберігаються сутності, що описують

користувача, зображення, колажі та ефекти. Для доступу до даних сервіси використовують репозиторії, які здійснюють запити до бази даних.

Третій рівень – це база даних, яка забезпечує зберігання всієї інформації. Тут містяться дані про користувачів, зображення, колажі, застосовані ефекти та історію змін. Репозиторії серверної частини напряму звертаються до таблиць бази даних і повертають результати у вигляді сутностей, з якими далі працюють сервіси.

Діаграма демонструє класичну модель взаємодії між клієнтом, сервером та базою даних. Користувач працює через інтерфейс, його дії обробляються на сервері, а дані зберігаються у базі. Такий поділ дозволяє чітко розмежувати відповідальність між рівнями та робить систему зрозумілою і гнучкою для подальшого розвитку.

Діаграма послідовностей

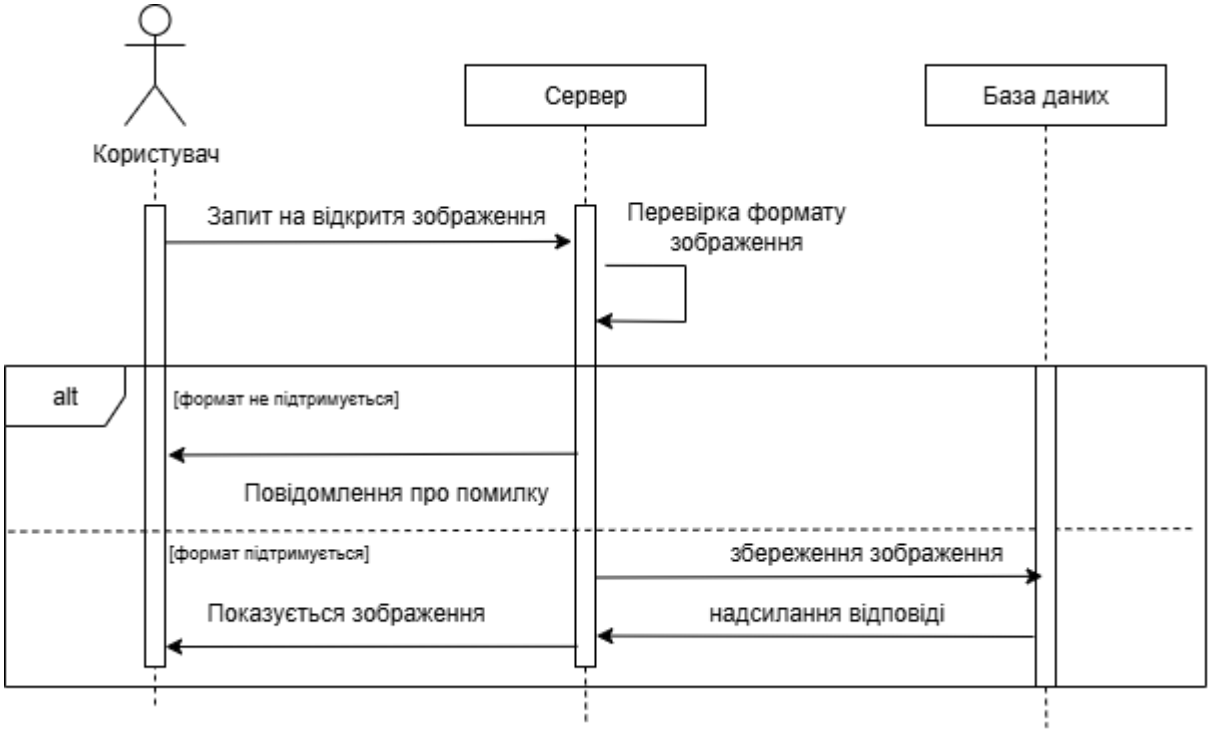

Рисунок 3. - Діаграма послідовностей для сценарію «Відкрити зображення»

Діаграма послідовностей для сценарію «Відкрити зображення» ілюструє взаємодію між трьома основними учасниками: користувачем, сервером та базою даних. Користувач ініціює процес, надсилаючи через інтерфейс запит на відкриття зображення. Сервер обробляє цей запит, зокрема перевіряє коректність формату файлу, який може бути одним із підтримуваних типів (JPG, PNG, GIF, BMP, TIFF). Якщо формат виявляється неприпустимим, сервер одразу повідомляє користувача про помилку.

У випадку правильного формату сервер завантажує вибране зображення у робочу область програми, зберігаючи його тимчасово в оперативній пам’яті. Якщо користувач є авторизованим, додатково виконується звернення до бази даних для збереження інформації про відкритий файл як окремого проєкту. База даних у відповідь підтверджує успішність операції. Після цього сервер передає результат користувачу, і зображення стає доступним у редакторі для подальшої роботи.

Діаграма також враховує виняткові ситуації. Якщо файл недоступний для завантаження або має невірний формат, система інформує про це користувача відповідним повідомленням. Таким чином, діаграма демонструє базову логіку роботи механізму відкриття зображення, а також підкреслює характерні зв’язки: безпосередня взаємодія користувача із сервером через веб-інтерфейс та обмін даними між сервером і базою даних без прямого доступу користувача до сховища.

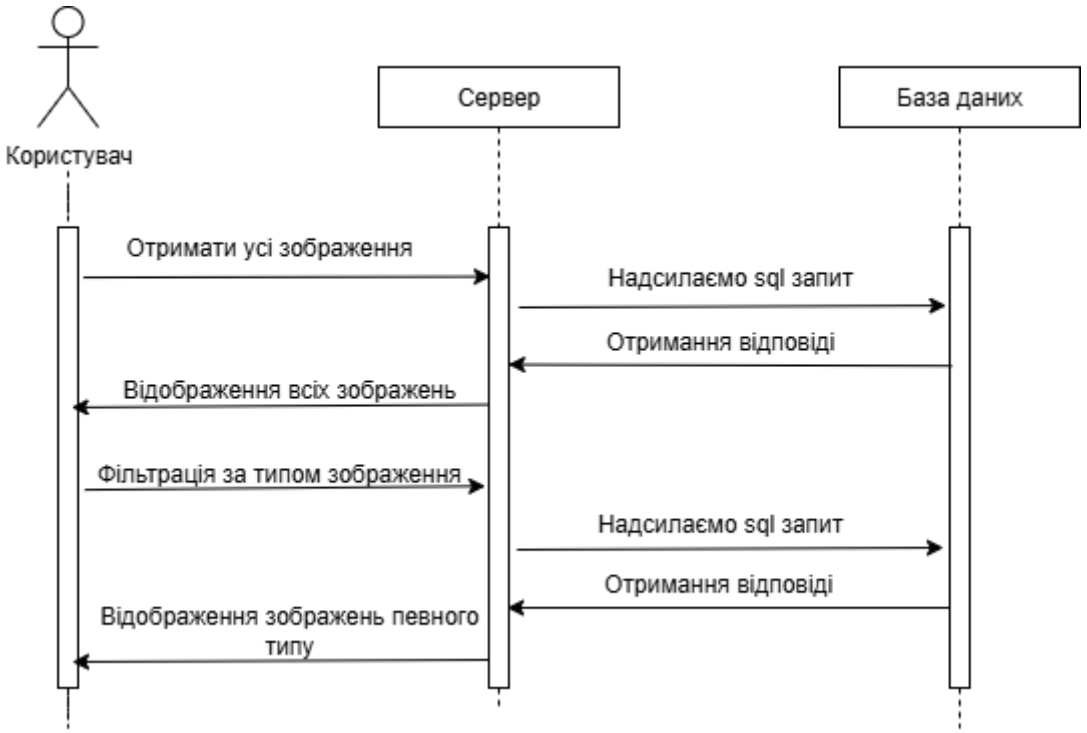

Рисунок 4. - Діаграма послідовностей для сценарію «Фільтрація зображень за типом»

Діаграма послідовностей для сценарію «Фільтрація зображень за типом» відображає взаємодію між трьома основними компонентами системи: користувачем, сервером та базою даних. Користувач, перебуваючи у своєму обліковому записі, відкриває розділ зі збереженими зображеннями, після чого сервер звертається до бази даних, щоб отримати список усіх файлів. Отримані результати повертаються на сервер і відображаються користувачу у вигляді повного списку зображень.

Далі користувач ініціює фільтрацію, обираючи певний формат файлу, наприклад PNG. Сервер надсилає відповідний запит до бази даних, де відбувається вибірка лише тих зображень, які відповідають вказаному типу. Відфільтровані дані повертаються на сервер, який відображає їх користувачу на екрані.

У випадку, якщо під час взаємодії з базою даних виникає проблема доступу, користувач отримує повідомлення про неможливість

завантаження списку. Якщо ж для вибраного формату не знайдено жодного файлу, система повідомляє про відсутність результатів та пропонує обрати інший тип. Таким чином, діаграма демонструє типовий обмін повідомленнями між компонентами системи під час фільтрації, а також враховує можливі помилкові ситуації.

Висновок: В ході виконання даної лабораторної роботи було досягнуто поставлену мету: спроєктовано та доопрацьовано архітектуру програмної системи "Редактор зображень" з використанням UML-діаграм та реалізовано повний цикл обробки даних.

На основі аналізу предметної області та вимог до системи було розроблено ключові діаграми моделювання:

- Діаграма компонентів була спроєктована для візуалізації архітектури додатку. Вона чітко продемонструвала багаторівневу структуру системи та визначила залежності між основними компонентами, а також їх взаємодію з базою даних PostgreSQL та файловим сховищем.
- Діаграма розгортання дозволила описати фізичну архітектуру системи. Вона показала, як компоненти програми розміщуються на апаратному забезпеченні: клієнтський веб-браузер, сервер додатків (де виконується Spring Boot застосунок) та сервер бази даних (PostgreSQL), що забезпечує чітке розуміння інфраструктури проєкту

Контрольні запитання

1. Що собою становить діаграма розгортання?

Діаграма розгортання — це тип діаграми UML, який показує фізичне розміщення програмних компонентів системи на апаратних пристроях.

Вона відображає, на яких серверах або комп’ютерах розгорнуті частини програми та як між ними здійснюється зв’язок. Така діаграма використовується для опису архітектури системи й показує, де саме працюють окремі компоненти.

2. Які бувають види вузлів на діаграмі розгортання?

Пристрій — це фізичний елемент системи, наприклад комп’ютер, сервер, смартфон або маршрутизатор. Він представляє апаратне забезпечення, на якому може виконуватись програмне забезпечення.

Виконавче середовище — це програмна платформа, яка працює всередині пристрою й забезпечує виконання програм. Прикладом може бути операційна система, віртуальна машина Java або контейнер.

3. Які бувають зв’язки на діаграмі розгортання?

Зв’язок комунікації показує мережеву взаємодію між вузлами — наприклад, обмін даними між сервером і клієнтом. Такий зв’язок відображає, як пристрої або середовища з’єднані між собою.

Зв’язок розміщення показує, який програмний компонент або артефакт розгорнуто на певному вузлі. Він відображає фізичне розташування програмних частин системи на пристроях.

4. Які елементи присутні на діаграмі компонентів?

Компонент — це основний елемент діаграми, який представляє окрему частину системи, наприклад модуль, бібліотеку або службу.

Інтерфейс показує, які функції або сервіси надає чи використовує компонент

Залежність відображає відношення між компонентами, коли один із них потребує функцій іншого.

З’єднання показують взаємодію між компонентами через їхні інтерфейси.

5. Що становлять собою зв’язки на діаграмі компонентів?

Зв’язки на діаграмі компонентів показують взаємозалежність і взаємодію між різними компонентами системи. Вони відображають, як один компонент використовує або надає функціональність іншому.

Найпоширенішим типом зв’язку є залежність, яка показує, що один компонент потребує інтерфейс або сервіс іншого для своєї роботи. Також можуть бути з’єднання через інтерфейси, які відображають реальну взаємодію між компонентами під час виконання програми.

6. Які бувають види діаграм взаємодії?

Діаграма послідовності показує обмін повідомленнями між об’єктами у часі — тобто хто, коли і в якій послідовності викликає певні операції.

Діаграма кооперації (комунікації) відображає ті самі взаємодії, але зосереджується не на часовій послідовності, а на структурних зв’язках між об’єктами, які беруть участь у взаємодії.

7. Для чого призначена діаграма послідовностей?

Діаграма послідовностей призначена для відображення процесу взаємодії між об’єктами у часі. Вона показує, які об’єкти беруть участь у певному сценарії системи, які повідомлення вони надсилають одне одному та в якій послідовності це відбувається.

За допомогою такої діаграми можна наочно простежити логіку виконання операцій, порядок викликів методів і реакцію системи на певні події. Діаграма послідовностей використовується під час аналізу або проєктування програм, щоб описати поведінку системи у динаміці.

8. Які ключові елементи можуть бути на діаграмі послідовностей?

На діаграмі послідовностей основними елементами є об’єкти, лінії життя, повідомлення та активації.

Об’єкти представляють учасників взаємодії, які надсилають або отримують повідомлення.

Лінія життя показує існування об’єкта протягом часу — вона зображується вертикально під об’єктом.

Повідомлення відображають виклики методів або обмін даними між об’єктами та показуються стрілками.

Активація позначає період, коли об’єкт виконує певну дію у відповідь на отримане повідомлення.

9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання?

Діаграми послідовностей тісно пов’язані з діаграмами варіантів використання, оскільки вони деталізують поведінку системи, описану у варіантах використання.

Діаграма варіантів використання показує, які функції виконує система та які актори взаємодіють із нею. Діаграма послідовностей конкретизує ці сценарії, показуючи покрокову взаємодію об’єктів і обмін повідомленнями, необхідними для реалізації певного варіанту використання.

10. Як діаграми послідовностей пов’язані з діаграмами класів?

Діаграми послідовностей пов’язані з діаграмами класів тим, що вони описують взаємодію об’єктів конкретних класів у часі.

Діаграма класів визначає структуру системи, показуючи класи, їхні атрибути та методи, а також зв’язки між класами. Діаграма послідовностей показує, як об’єкти цих класів взаємодіють один з одним під час виконання певного сценарію, які методи викликаються і в якому порядку.

Робота програми

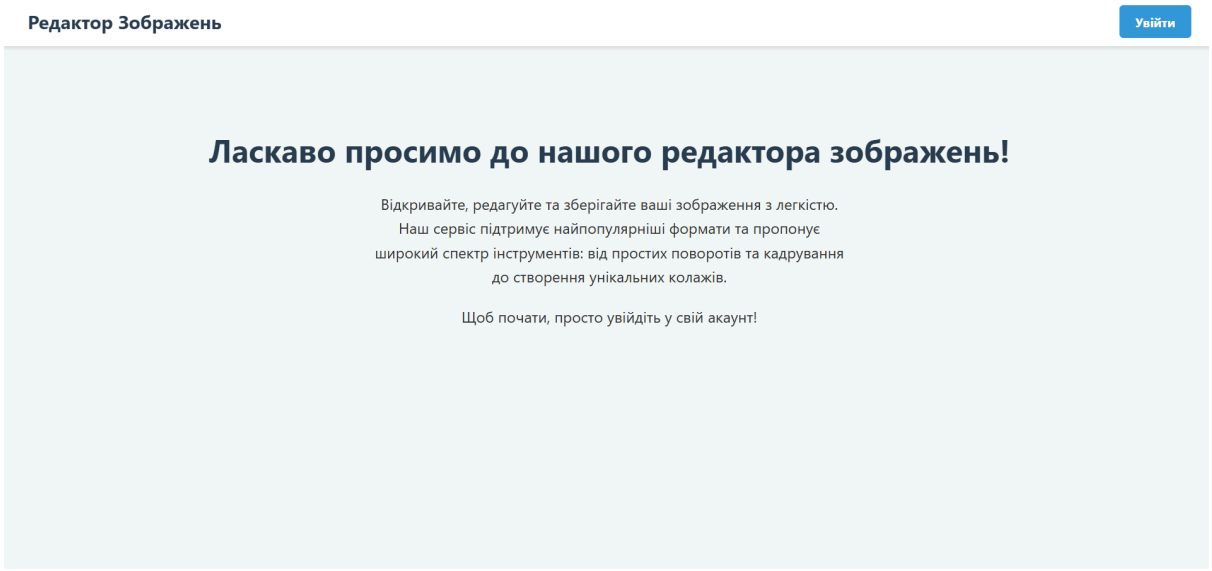

Рисунок 5. - Головна сторінка

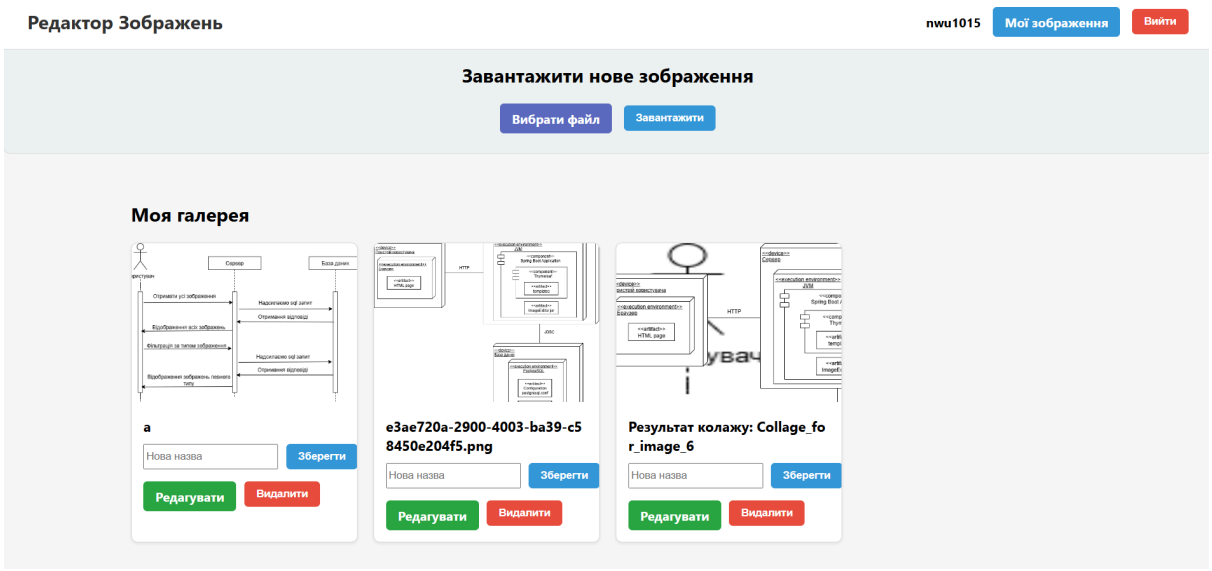

Рисунок 6. - Мої зображення

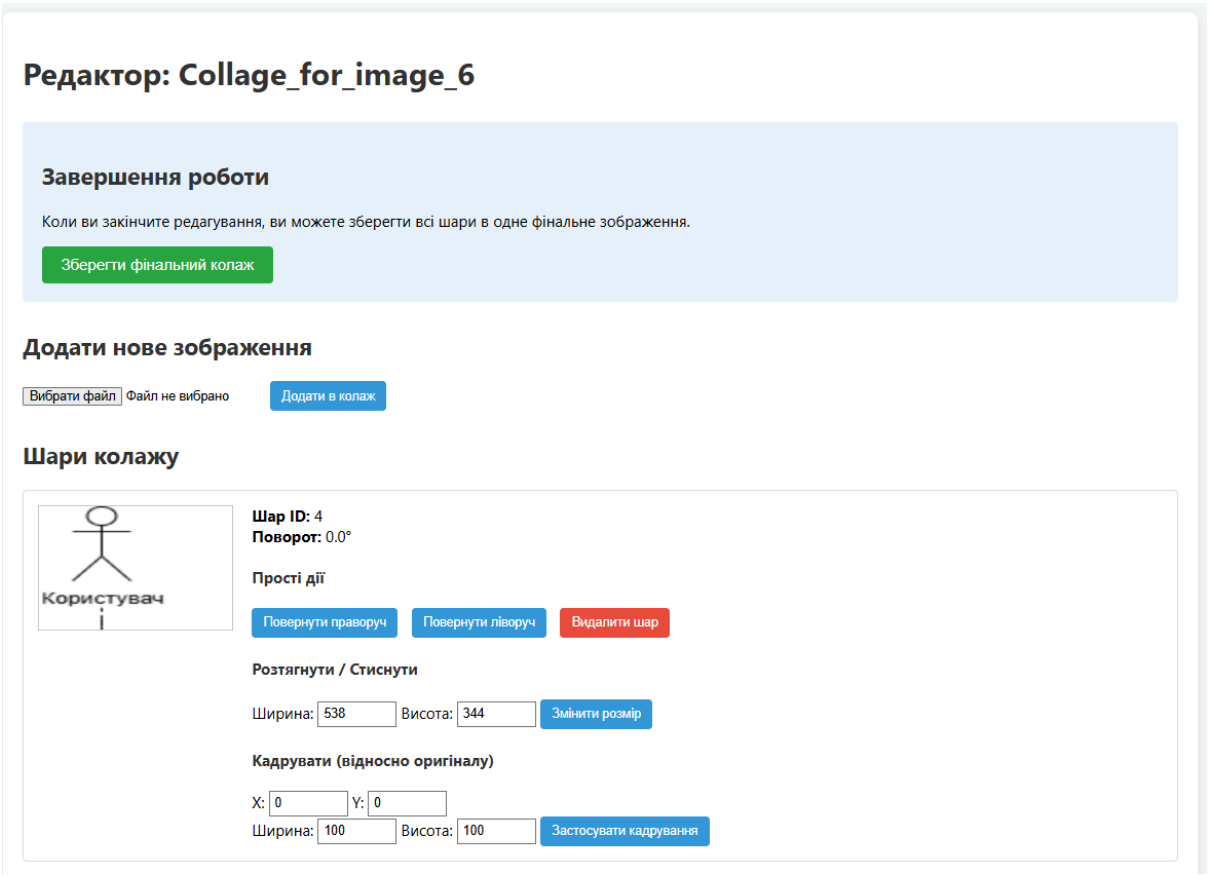

Рисунок 7. - Редагування зображення