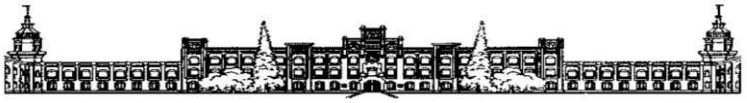


Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського” Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №3

із дисципліни *«Технології розроблення програмного забезпечення»*

Тема: « ДІАГРАМА РОЗГОРТАННЯ. ДІАГРАМА КОМПОНЕНТІВ. ДІАГРАМА ВЗАЄМОДІЙ ТА ПОСЛІДОВНОСТЕЙ »

Виконав:

Студент групи ІА-23 Хохол М.В.

Перевірив:

Мягкий М.Ю.

2024

Варіант №7

..7 Редактор зображень (state, prototype, memento, facade, composite, client-server)

Редактор зображень має такі функціональні можливості: відкриття/збереження зображень у найпопулярніших форматах (5 на вибір студента), застосування ефектів, наприклад поворот, розтягування, стиснення, кадрування зображення, можливість створення колажів шляхом «нашарування» зображень.

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.

2. Розробити діаграму розгортання для проектованої системи.

3. Розробити діаграму компонентів для проектованої системи.

4. Розробити діаграму послідовностей для проектованої системи.

5. Скласти звіт про виконану роботу

Зміст

Хід роботи:

Крок 1. Короткі теоретичні відомості

Діаграма розгортання (Deployment Diagram) використовується для того, щоб показати, як фізично розташована система на пристроях, тобто де саме і на якому обладнанні працюють її частини. Вона дозволяє розробникам зрозуміти, як система працює на реальних пристроях і як різні частини взаємодіють одна з одною. Основні елементи діаграми — це узли (пристрої або середовища виконання) та артефакти (виконувані файли, конфігураційні файли тощо).

Діаграма компонентів (Component Diagram) показує структуру системи, поділяючи її на компоненти, що можуть взаємодіяти один з одним. Вона дає змогу побачити, як система розбита на частини і як ці частини взаємодіють між собою, візуалізуючи структуру коду та залежності між компонентами.

Діаграма взаємодії показує, як компоненти або об'єкти системи взаємодіють один з одним під час виконання певного процесу. Вона використовується для представлення послідовності обміну повідомленнями між об'єктами в конкретному контексті. На такій діаграмі відображається порядок дій та логіка їх виконання. Аналітик зосереджується на переходах від однієї активності до іншої та результатах цих переходів. Зазвичай результати дій можуть призводити до зміни стану системи або повернення певного значення. Основні елементи діаграми взаємодії включають об'єкти або учасників, лінії життя (що показують, як довго об'єкт існує протягом процесу), повідомлення між об'єктами, а також активності, які вони виконують, і можливі обмеження чи умови.

Діаграма послідовностей (Sequence Diagram) показує взаємодію об'єктів у системі у часі. Вона демонструє, як відбувається обмін повідомленнями між учасниками (користувачами, сервером, базою даних) для виконання конкретних задач. Основні елементи діаграми — це об'єкти, лінії життя, фокус управління та повідомлення, які відображають передачу даних або виклик методів у певній послідовності.

Крок 2. Діаграма розгортання

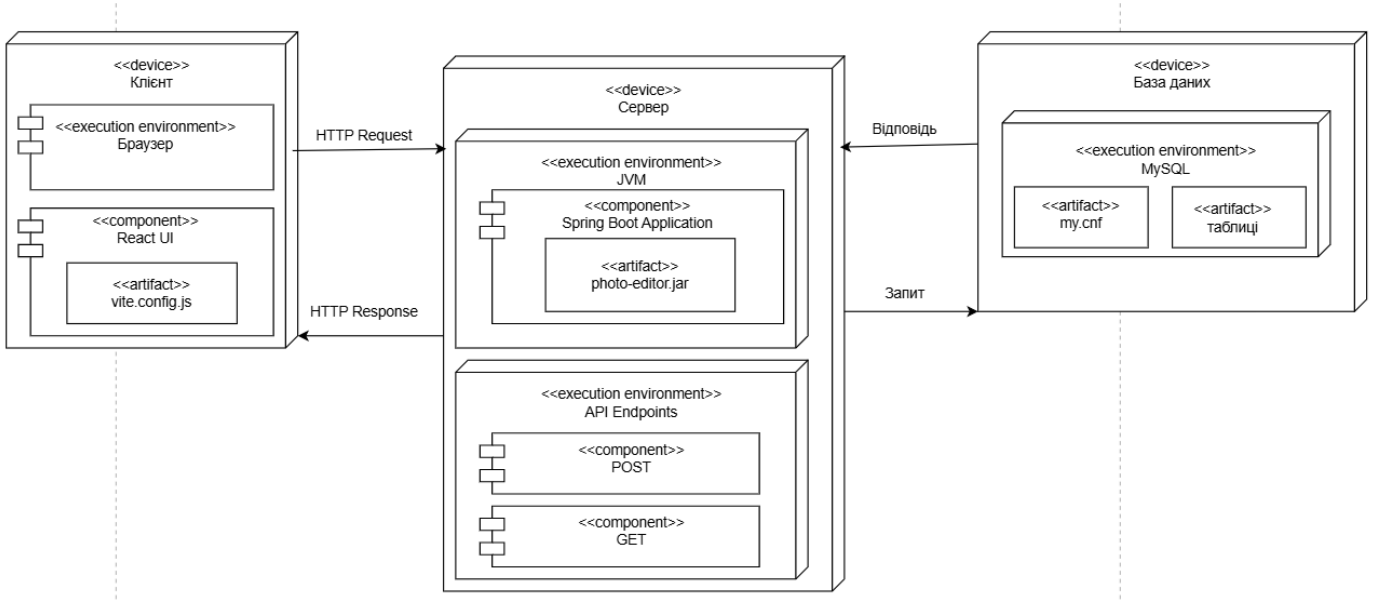


Рис1. Діаграма розгортання

Пристрої (Devices):

Клієнт (<<device>>): Це користувацький пристрій, з якого взаємодіють із системою через браузер. На клієнтському пристрої розгорнутий:

- **Середовище виконання** (<<execution environment>>) — **Браузер**, що використовується для відображення інтерфейсу користувача.
- **Компонент** (<<component>>) — **React UI**, який відповідає за фронтенд частину застосунку. React забезпечує відображення та взаємодію з користувачем.
- **Артефакт** (<<artifact>>) — **vite.config.js**, який вказує на конфігураційний файл для зборки застосунку за допомогою інструмента Vite.

Сервер (<<device>>): Це пристрій, де розгорнута серверна частина системи. На сервері розгорнуті:

- **Середовище виконання JVM** (<<execution environment>>) — **JVM** (Java Virtual Machine), де працює Spring Boot Application.
- **Компонент** (<<component>>) — **Spring Boot Application**, що відповідає за серверну логіку обробки запитів.
- **Артефакт** (<<artifact>>) — **photo-editor.jar**, який представляє скомпільовану програму, що реалізує серверну частину застосунку.

- Додатково розгорнуті **API Endpoints** як середовище виконання, яке містить компоненти:

- POST та **GET** компоненти, які представляють точки доступу (API) для обробки HTTP запитів, відповідно для додавання нових даних та отримання існуючих.

База даних (<<device>>): Це пристрій або сервер бази даних, де зберігаються всі дані системи.

- Середовище виконання (<<execution environment>>) — **MySQL**, що представляє систему керування базами даних.

- Артефакт (<<artifact>>) — **my.cnf**, який є конфігураційним файлом для MySQL, що містить налаштування роботи сервера бази даних.

- Артефакт (<<artifact>>) — **Таблиці**, що зберігають дані, необхідні для роботи застосунку, як-от інформація про користувачів, зображення тощо.

Крок 3. Діаграма компонентів

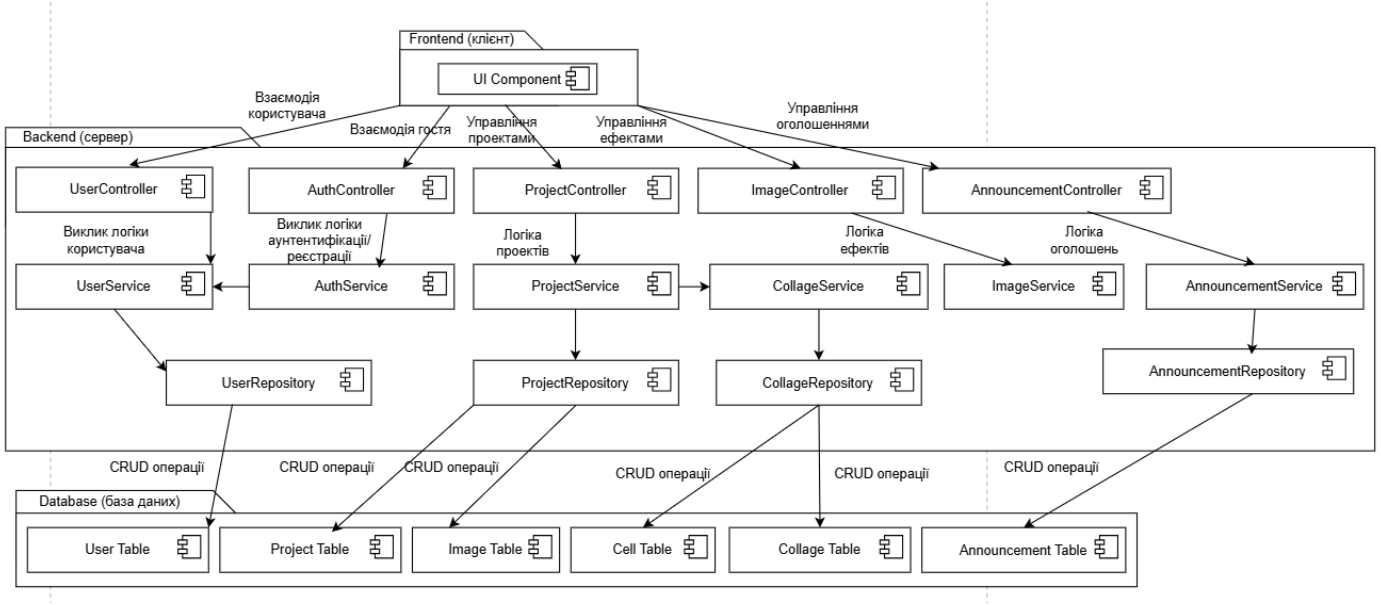


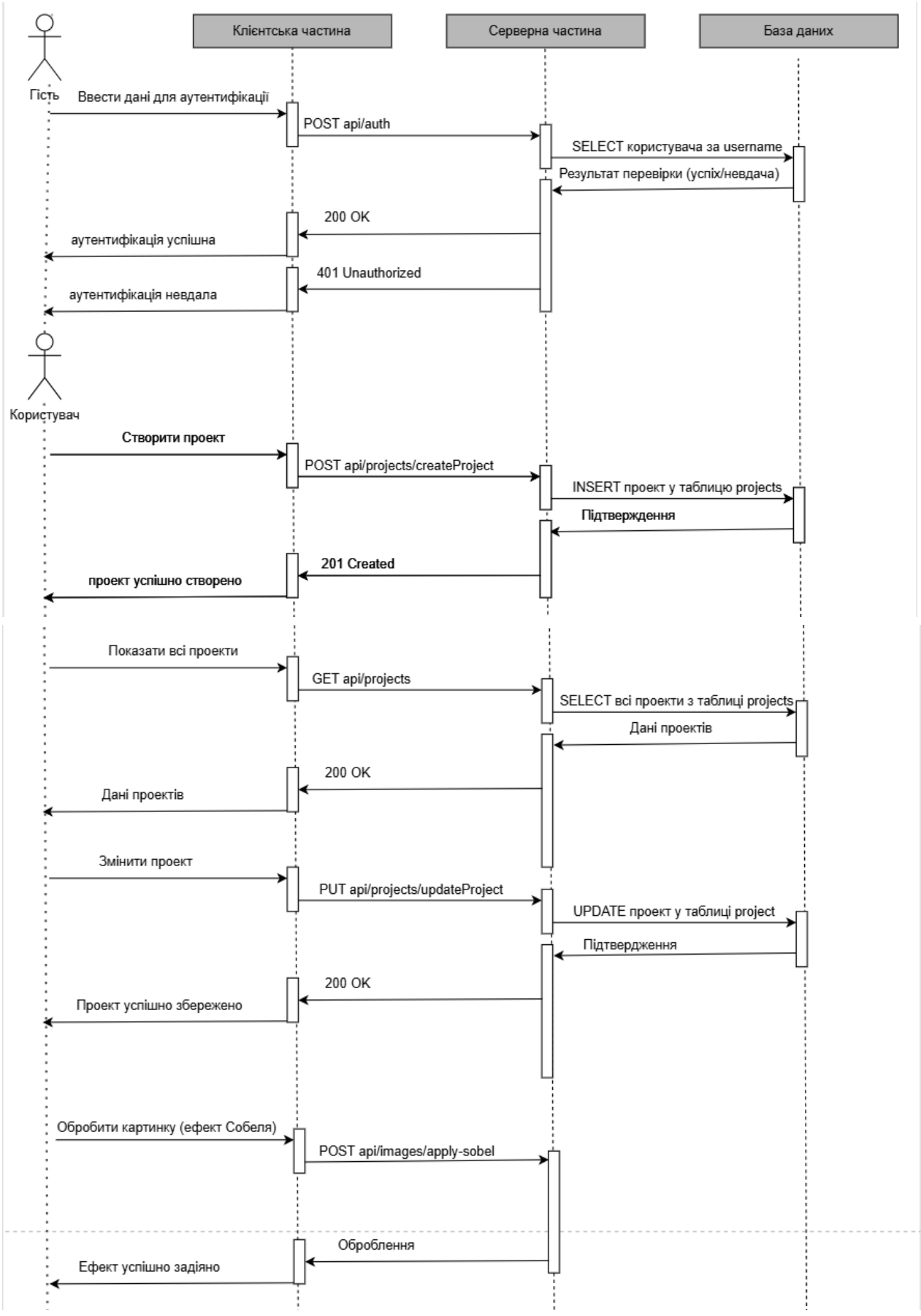
Рис1. Діаграма компонентів

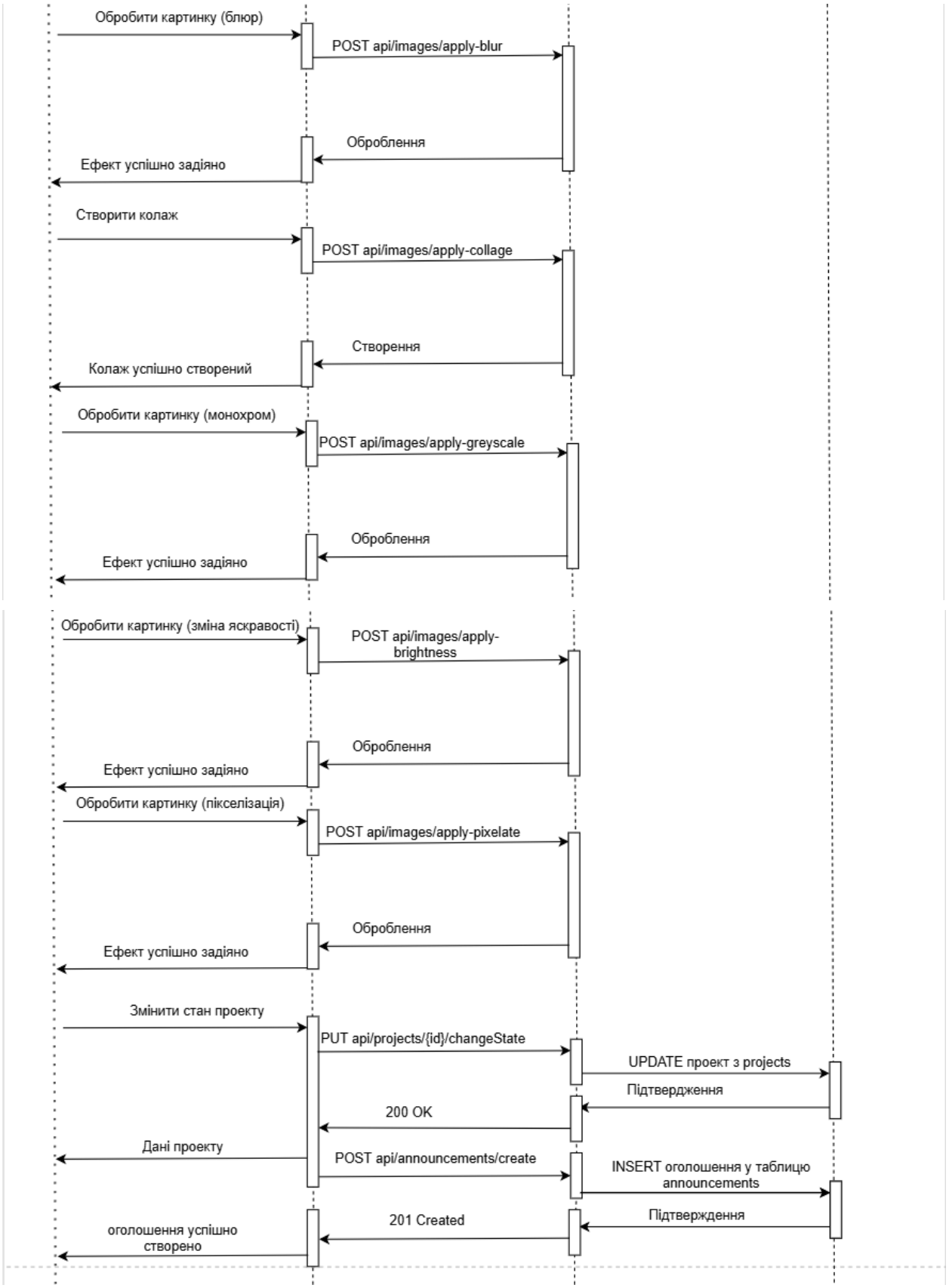
- **UI Component**: Цей компонент відповідає за інтерфейс користувача, тобто те, що бачить і з чим взаємодіє користувач. Він надсилає запити до серверної частини і отримує відповіді для відображення даних на екрані.
- **UserController**: Контролер для обробки запитів, пов’язаних з користувачами. Він приймає запити з інтерфейсу, наприклад, для отримання інформації про користувача або оновлення його профілю.
- **UserService**: Сервіс, який містить логіку роботи з користувачами. Виконує такі операції, як створення нового користувача, редагування його даних тощо.
- **UserRepository**: Репозиторій для взаємодії з таблицею користувачів у базі даних. Виконує операції збереження, оновлення, видалення та пошуку інформації про користувачів.
- **AuthController**: Контролер для обробки запитів, пов'язаних з аутентифікацією користувачів, зокрема для входу в систему або реєстрації нового користувача.
- **AuthService**: Сервіс, який реалізує логіку аутентифікації та реєстрації. Перевіряє облікові дані користувачів, генерує токени та забезпечує безпеку доступу.
- **ProjectController**: Контролер для обробки запитів, що стосуються управління проектами. Отримує запити з інтерфейсу користувача для створення, редагування або видалення проектів.
- **ProjectService**: Сервіс для управління логікою проектів. Реалізує бізнес-логіку, пов'язану зі створенням, оновленням та видаленням проектів.
- **ProjectRepository**: Репозиторій, який забезпечує взаємодію з таблицею проектів у базі даних. Відповідає за CRUD-операції з проектами.
- **ImageController**: Контролер для обробки запитів, що стосуються зображень. Наприклад, для додавання нових зображень до проекту або для редагування існуючих.
- **ImageService**: Сервіс, який містить логіку роботи із зображеннями, таку як завантаження, обробка та збереження зображень.
- **CollageService**: Сервіс, який відповідає за управління колажами, зокрема їх створення, зміни та збереження. Реалізує логіку щодо комбінування зображень у колажі.

- **CollageRepository**: Репозиторій, що забезпечує взаємодію з таблицею колажів у базі даних, для виконання CRUD-операцій з колажами.
- **AnnouncementController**: Контролер для управління оголошеннями. Приймає запити на створення, зміну або видалення оголошень, а також на отримання їх списку.
- **AnnouncementService**: Сервіс для логіки, пов'язаної з оголошеннями. Виконує такі дії, як створення нових оголошень, відправка сповіщень тощо.
- **AnnouncementRepository**: Репозиторій, що забезпечує збереження, оновлення та видалення даних оголошень у базі даних.
- **User Table**: Таблиця у базі даних, що зберігає дані користувачів, наприклад, імена, електронні адреси та паролі.
- **Project Table**: Таблиця у базі даних для збереження інформації про проекти, такі як назва проекту, опис та інші дані.
- **Image Table**: Таблиця для збереження інформації про зображення, включаючи їх метадані та посилання на файли.
- **Cell Table**: Таблиця, що зберігає інформацію про клітинки колажу, наприклад, їх розташування та інші властивості.
- **Collage Table**: Таблиця у базі даних, що зберігає дані про колажі, включаючи їх структуру та зв'язки із зображеннями.
- **Announcement Table**: Таблиця для збереження оголошень, які можуть бути створені користувачами або системою для інформування інших користувачів про зміни у проектах.

Ці файли та компоненти разом складають архітектуру застосунку, де фронтенд спілкується з бекендом через контролери, сервіси обробляють бізнес-логіку, а репозиторії взаємодіють з базою даних для збереження та отримання інформації.

Крок 3. Діаграма послідовності


Клієнтська частина
Серверна частина
База даних
Гість
Ввести дані для аутентифікації
POST api/auth
SELECT користувача за username
Результат перевірки (успіх/невдача)
200 OK
аутентифікація успішна
401 Unauthorized
аутентифікація невдала
Користувач
Створити проект
POST api/projects/createProject
INSERT проект у таблицю projects
Підтвердження
201 Created
проект успішно створено
Показати всі проекти
GET api/projects
SELECT всі проекти з таблиці projects
Дані проектів
200 OK
Дані проектів
Змінити проект
PUT api/projects/updateProject
UPDATE проект у таблиці project
Підтвердження
200 OK
Проект успішно збережено
Обробити картинку (ефект Собеля)
POST api/images/apply-sobel
Оброблення
Ефект успішно задіяно


Обробити картинку (блюр)
POST api/images/apply-blur
Оброблення
Ефект успішно задіяно
Створити колаж
POST api/images/apply-collage
Створення
Колаж успішно створений
Обробити картинку (монохром)
POST api/images/apply-greyscale
Оброблення
Ефект успішно задіяно
Обробити картинку (зміна яскравості)
POST api/images/apply-brightness
Оброблення
Ефект успішно задіяно
Обробити картинку (пікселізація)
POST api/images/apply-pixelate
Оброблення
Ефект успішно задіяно
Змінити стан проекту
PUT api/projects/{id}/changeState
UPDATE проект з projects
Підтвердження
200 OK
Дані проекту
POST api/announcements/create
INSERT оголошення у таблицю announcements
Підтвердження
201 Created
оголошення успішно створено

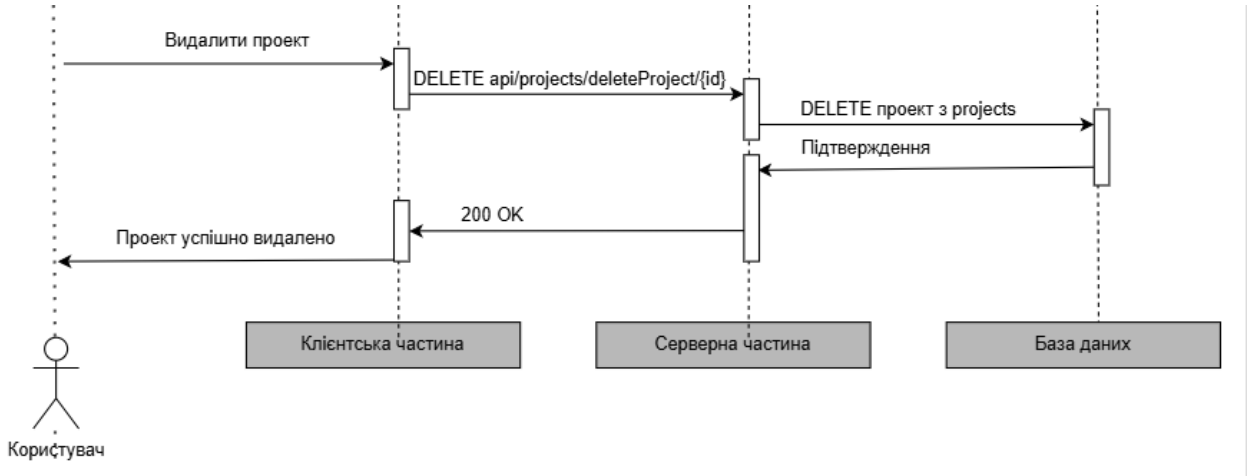


Рис3. Діаграма послідовностей

1. Аутентифікація: Користувач вводить свої дані для входу. Сервер перевіряє їх у базі даних і надає доступ у разі успіху.

2. Створення проекту: Після успішної авторизації користувач може створити новий проект, який зберігається у базі даних.

3. Перегляд проектів: Користувач може отримати список всіх своїх проектів.

4. Редагування проекту: Користувач може змінювати існуючі проекти, оновлюючи їх у базі даних.

5. Застосування ефектів до зображень: Користувач має можливість застосовувати різні ефекти до зображень (розмиття, пікселізація, яскравість тощо).

6. Створення колажів: Користувач може створювати колажі з кількох зображень.

7. Зміна стану проекту: Користувач може змінити стан проекту, наприклад, зробити його закритим.

8. Видалення проекту: Користувач може видалити проект, що буде видалено з бази даних.

Висновок: У рамках виконання даної лабораторної роботи було розроблено архітектуру системи "Редактор зображень", застосувавши комплексний підхід до проектування та моделювання. На початкових етапах проведено теоретичний аналіз, що стосується важливості використання UML-діаграм для проектування

інформаційних систем, зокрема, їх значення для візуалізації та документування архітектури програмного забезпечення. Створені UML-діаграми заклали основу для реалізації системи, що сприятиме ефективному управлінню проектом