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

Розробка структури застосунку

Виконали:
Палінчак І.В.
Хохол М.В.

Перевірив:
Нечай Д.О.

Київ 2024

Завдання:

1. Розділити додаток на основні компоненти/модулі, описати їх взаємодію відомими вам методами (наприклад діаграма компонентів ПЗ)
2. Описати дані та їх зв’язки (наприклад ER діаграма)
3. Описати як дані оновлюються\змінюються\агрегуються на основі ключових сценаріїв, які буде виконувати додаток

Хід роботи

Діаграма компонентів в UML використовується для моделювання логічної структури системи, зокрема її компонентів та взаємодій між ними. Вона показує, як різні частини програмного забезпечення (компоненти) з'єднані між собою і як вони взаємодіють через інтерфейси.

Основні елементи:

- Компоненти: основні частини системи (наприклад, класи, модулі, бібліотеки).
- Інтерфейси: визначають точки взаємодії між компонентами.
- Зв'язки: показують залежності чи комунікацію між компонентами.

Призначення:

- Опис архітектури програмної системи.
- Визначення залежностей між компонентами.
- Підтримка розуміння організації та структури програмного забезпечення.

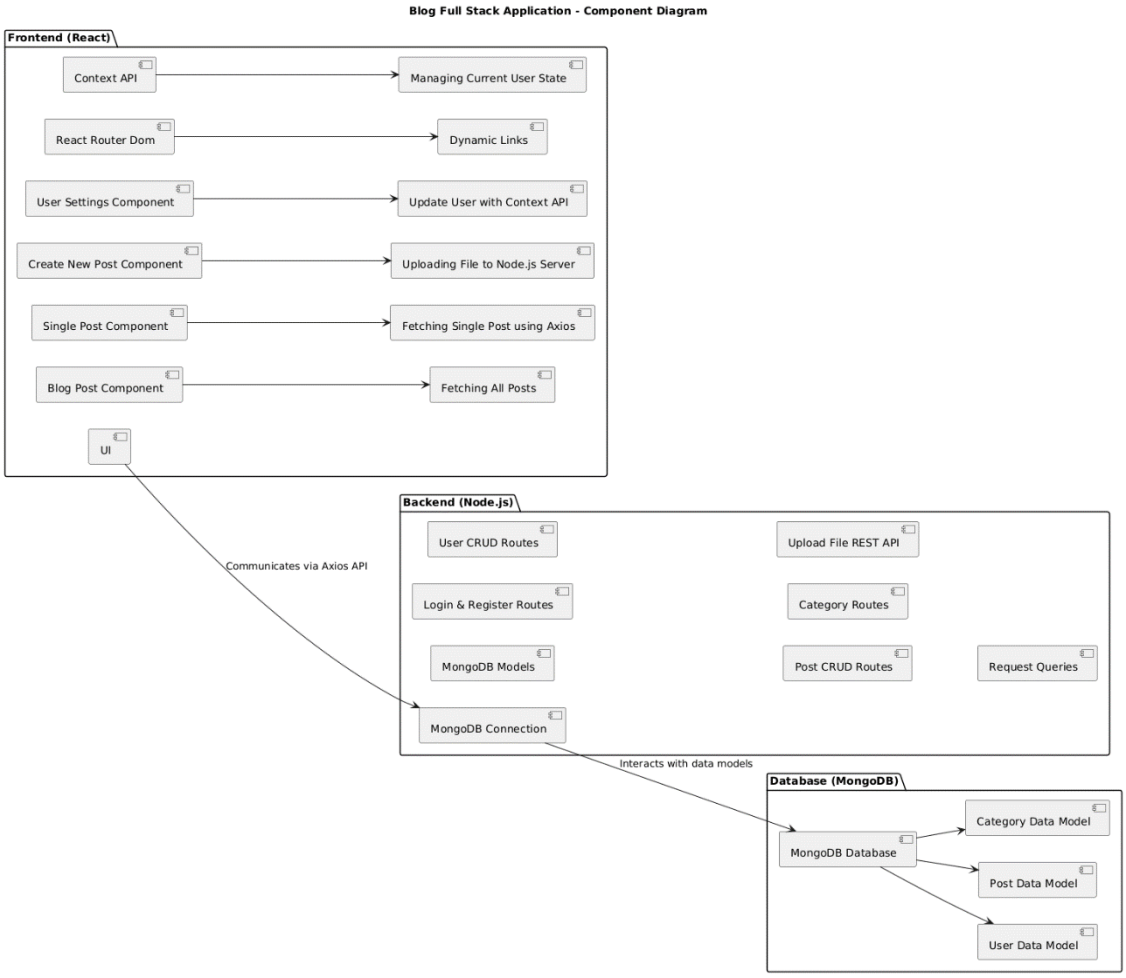

Рисунок 1 – Діаграма компонентів проєктуємого застосунку

Ця діаграма компонентів ілюструє архітектуру повного стеку додатку для блогу, побудованого за допомогою технологій MERN (MongoDB, Express, React, Node.js). Вона розділяє систему на три основні пакети:

1. **Frontend (React)**:
    - Це частина додатку, яка взаємодіє безпосередньо з користувачем. Вона включає UI-компоненти, такі як:
        - Blog Post Component: відображає всі пости блогу.
        - Single Post Component: показує один пост при його виборі.
        - Create New Post Component: дозволяє створювати нові пости.
        - User Settings Component: для редагування налаштувань користувача.
        - Login and Register Page: сторінки для реєстрації та входу.
        - React Router Dom: використовується для навігації по додатку з динамічними маршрутами.
        - Context API: управління станом поточного користувача (наприклад, для автентифікації).

    Ці компоненти взаємодіють з сервером через Axios API для отримання даних та виконання запитів (наприклад, отримання постів або створення нових).

Наприклад, Blog Post Component отримує всі пости, а Single Post Component отримує один пост за допомогою Axios.

2. **Backend (Node.js)**:
    - Це серверна частина додатку, яка працює з базою даних і виконує основну логіку бізнес-процесів:
        - MongoDB Connection: з'єднання з базою даних MongoDB.
        - MongoDB Models: визначення моделей для взаємодії з даними (наприклад, користувачі, пости, категорії).
        - Login & Register Routes: маршрути для реєстрації та входу користувачів.
        - User CRUD Routes: маршрути для створення, читання, оновлення та видалення користувачів.
        - Post CRUD Routes: маршрути для маніпуляцій з постами.
        - Category Routes: маршрути для роботи з категоріями постів.
        - Upload File REST API: маршрути для завантаження файлів на сервер.
        - Request Queries: обробка запитів до бази даних.

    Сервер взаємодіє з базою даних через MongoDB Connection, що забезпечує доступ до даних користувачів, постів і категорій.

3. **Database (MongoDB)**:
    - Це база даних, де зберігаються всі дані додатку:
        - User Data Model: модель даних користувачів (наприклад, ім'я, email, пароль).
        - Post Data Model: модель даних постів (заголовок, контент, автор).
        - Category Data Model: модель категорій, до яких відносяться пости (наприклад, технології, новини).

Основний потік даних:

- Користувачі взаємодіють з UI на фронтенді, де вони можуть переглядати пости, реєструватися, входити в систему або створювати нові пости.
- Фронтенд через Axios здійснює запити до серверної частини на Express Server, який обробляє запити і взаємодіє з базою даних MongoDB.
- Сервер на Node.js взаємодіє з MongoDB для отримання та збереження даних про користувачів, пости та категорії.

Ця діаграма дає чітке уявлення про архітектуру та взаємодію компонентів у MERN додатку, де кожен компонент має чітко визначену роль у процесі обробки даних і взаємодії з іншими частинами системи.

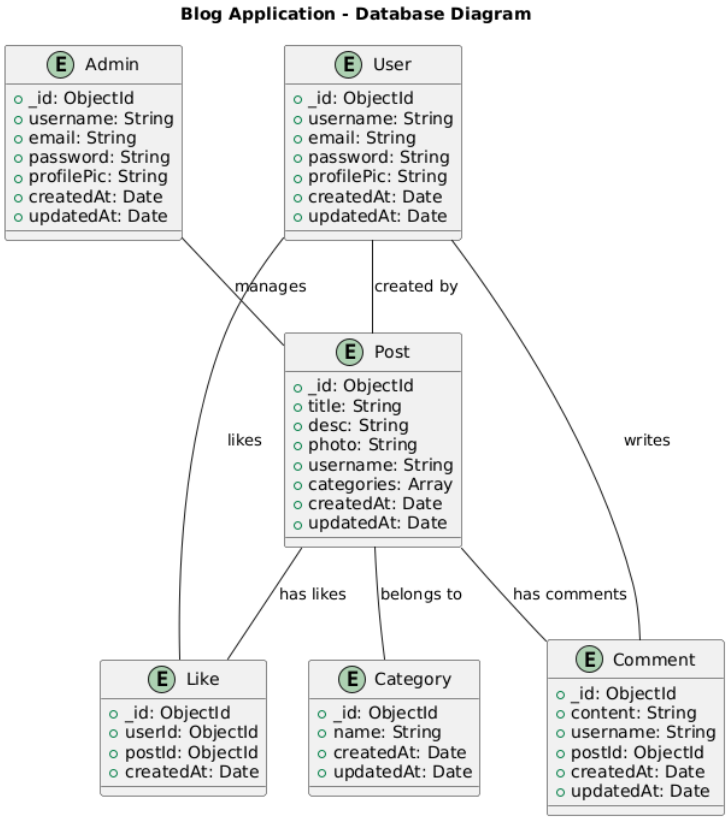

Рисунок 2 – Діаграма що ілюструє структуру баз даних

Структура бази даних та їх призначення

1.1. Users (Користувачі)
Зберігає інформацію про зареєстрованих користувачів: імена, email, паролі, аватарки тощо.

1.2. Admins (Адміністратори)
Містить дані про адміністраторів, які можуть керувати контентом (пости, коментарі тощо).

1.3. Posts (Пости)
Зберігає інформацію про публікації користувачів, включаючи заголовки, текст та автора.

1.4. Categories (Категорії)
Список категорій для класифікації постів, наприклад, "Технології", "Новини" тощо.

1.5. Comments (Коментарі)
Містить дані про коментарі, які залишають користувачі під постами.

1.6. Likes (Лайки)
Фіксує інформацію про лайки, поставлені користувачами на пости.

1.7. PostCategories (Зв’язок постів і категорій)
Забезпечує зв’язок між постами та категоріями, оскільки один пост може належати до кількох категорій.

Діаграма випадків використання (Use Case Diagram) — це графічна модель, яка ілюструє, як користувачі (актори) взаємодіють із системою для досягнення певних цілей. Вона дає змогу зрозуміти функціональність системи, показуючи, які дії можуть виконувати різні користувачі, і як ці дії пов'язані між собою.

Основні елементи:

1. **Актори (Actors)**:
   - Представляють користувачів або зовнішні системи, що взаємодіють із системою.
   - Зображуються у вигляді фігури людини.
2. **Випадки використання (Use Cases)**:
   - Це функції або сценарії, які система виконує для досягнення мети.
   - Зображуються у вигляді овалів.
3. **Зв'язки (Relationships)**:
   - Показують, як актори взаємодіють із випадками використання, і як ці випадки пов'язані між собою.
   - Основні типи зв'язків:
     - Асоціація (Association): Лінія, що з'єднує актора і випадок використання.
     - Include: Один випадок використання обов'язково викликає інший.
     - Extend: Один випадок використання може розширювати інший за певних умов.

Призначення:

- Візуалізує ключові функції системи і дозволяє побачити, хто і як їх використовує.
- Допомагає зрозуміти вимоги до системи.
- Використовується для комунікації між розробниками, аналітиками і зацікавленими сторонами.

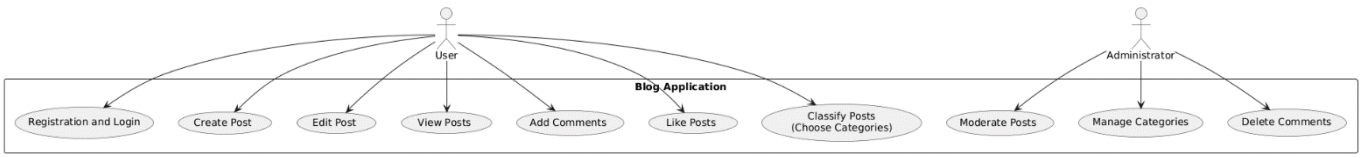

Рисунок 3 – Діаграма що ілюструє структуру баз даних

Ключові моменти:
Користувач:

- Може зареєструватися та увійти в систему.
- Створювати, редагувати та переглядати дописи.
- Додавати коментарі та вподобання до дописів.

- Класифікувати дописи, обираючи категорії.

Адміністратор:

- Може модерувати дописи.
- Керувати категоріями в системі.
- Видаляти недоречні або небажані коментарі.

Висновок:

Виконано розподіл додатка на основні компоненти, створено діаграму компонентів, яка демонструє взаємодію між модулями системи блогу (інтерфейс користувача, серверна логіка, база даних). Створено ER-діаграму, яка відображає структуру даних, зв’язки між таблицями бази даних, зокрема авторів, пости, коментарі, та їх атрибути. Розроблено Use Case Diagram, що ілюструє ключові сценарії взаємодії акторів (Автор та Читач) із системою: створення, редагування та перегляд постів, коментування. Ці діаграми забезпечують комплексний огляд архітектури, структури даних та основних функціональних сценаріїв блогу.