МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри_____ Є. О. Давиденко
підпис
«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

РОЗРОБКА СИСТЕМИ КЕРУВАННЯ ВМІСТОМ ДЛЯ ОРГАНІЗАЦІЇ БЛОГУ

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409.22011017

Здобувач: _____А. Д. Нищоцький
підпис
«__» _____ 2024 р.

Керівник: PhD, ст. викладач _____К. О. Антіпова
підпис
«__» _____ 2024 р.

Консультант: канд. техн. наук.,
доцент кафедри екології _____А.О. Алексєєва
підпис
«__» _____ 2024 р.

м. Миколаїв – 2024 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп’ютерних наук

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри _____ Є. О. Давиденко

«_____» _____ 2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано здобувачу групи 409 факультету комп’ютерних наук

Нищоцькому Артуру Дмитровичу

(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи

Розробка системи керування вмістом для організації блогу

Затверджена наказом по ЧНУ від «22» грудня 2023 р. № 269

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є розроблена система керування вмістом для блогу

4. Перелік питань, що підлягають розробці:

- дослідження предметної області та аналіз існуючих аналогів;
- формування специфікації вимог до програмного забезпечення;
- визначення архітектури для проєктування програмного забезпечення;
- моделювання та проєктування програмного забезпечення;
- розробка програмного забезпечення;
- здійснення тестування роботи програмного забезпечення;

5. Перелік графічних матеріалів

Презентація

6. Завдання до спеціальної частини

Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алексєєва А.О., канд. Техн. наук, доцент (б. в. з.)	Кафедра екології Медичного інституту ЧНУ ім. Петра Могили	Спеціальна частина з охорони праці

Керівник роботи _____ PhD, ст. викладач Антіпова К.О._____
(посада, прізвище, ім’я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

_____Нищоцький Артур Дмитрович_____
(прізвище, ім’я, по батькові)

_____
(підпис)

Дата видачі завдання «____» _____ 2024р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: «Розробка системи керування вмістом для організації блогу»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КРБ	20.02.2024	23.02.2024	Виконано
2	Огляд літератури за темою роботи	24.02.2024	26.02.2024	Виконано
3	Складання календарного плану КРБ	26.02.2024	27.02.2024	Виконано
4	Аналіз предметної області	28.02.2024	01.03.2024	Виконано
5	Розробка проєктних рішень	04.03.2024	10.03.2024	Виконано
6	Моделювання та конструювання ПЗ	22.03.2024	26.03.2024	Виконано
7	Кодування ПЗ, тестування та апробація розробленого ПЗ, аналіз результатів тестування	28.03.2024	22.05.2024	Виконано
8	Розробка спеціальної частини з охорони праці	15.04.2024	21.04.2024	Виконано
9	Оформлення КРБ та презентації	22.04.2024	29.04.2024	Виконано
10	Відгук керівника КРБ	06.05.2024	08.05.2024	Виконано
11	Завершення оформлення КРБ та презентації	10.05.2024	23.05.2024	Виконано
12	Попередній захист	05.06.2024	05.06.2024	Виконано
13	Рецензування	16.06.2024	17.06.2024	Виконано
14	Захист кваліфікаційної роботи	25.06.2024	25.06.2024	Виконано

Розробив здобувач Нищоцький Артур Дмитрович
(прізвище, ім’я, по батькові) *(підпис)*
«24» січня 2024 р.

Керівник роботи PhD, ст. викладач Антіпова Катерина Олександрівна
(посада, прізвище, ім’я, по батькові) *(підпис)*
«24»січня 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Розробка системи керування вмістом для організації блогу»

Здобувач 409 гр,: Нищоцький Артур Дмитрович

Керівник: PhD, ст. викладач Антіпова Катерина Олександрівна

Кваліфікаційна робота присвячена розробці системи керування вмістом для організації блогу на освнові фреймворку Laravel(далі CMS).

Актуальність даної роботи обумовлена необхідністю забезпечення ефективного та зручного керування вмістом сайтів за допомогою створення спрощенної та більш автоматизованної CMS.

Об'єктом кваліфікаційної роботи є процес розробки та впровадження CMS для вебсайтів.

Предметом кваліфікаційної роботи є засоби розробки та впровадження системи управління контентом з використанням Laravel.

Метою кваліфікаційної роботи є розробка системи управління контентом для оптимізації процесу створення та управління контентом на вебсайтах.

У першому розділі кваліфікаційної роботи бакалавра представлено аналіз предметної області, аналіз готових рішень та специфікація вимог до ПЗ.

У другому розділі спроєктовано декілька UML діаграм для CMS, а саме діаграму варіантів використання, діаграму класів, діаграму послідовностей та мокап системи.

У третьому розділі описані методи розробки CMS та описано базу даних.

У четвертому розділі описані результати розробки системи. КРБ викладена на 74 сторінки, вона містить 4 розділи, 30 ілюстрацій, 16 таблиць, 17 джерел в переліку посилань

Ключові слова: вебсайт, система керування вмістом, Laravel, PHP, база даних, MySQL, CMS.

ABSTRACT

of the Bachelor's Thesis

«Development of a content management system for organizing a blog»

Student of group 409: Nyshchotskyi Artur Dmytrovych

PhD, Senior Lecturer Antipova Kateryna Oleksandrivna

This work is dedicated to the development of a browser system along with the organization of a blog based on the Laravel framework (also called CMS).

The relevance of this work is due to the need to ensure effective and convenient management of site content by creating a simplified and more automated CMS.

The subject of investigation is the process of development and implementation of CMS for websites.

The subject is the development and implementation of a content management system using Laravel wikis.

The purpose of this work is to develop a content management system to optimize the process of creating and managing content on websites.

The first section of the bachelor's degree includes analysis of the subject area, analysis of ready-made solutions, problem statement and specification.

Second section has designed a UML diagram for CMS, as well as a choice diagram, a class diagram, a sequence diagram and a system mockup.

The third section describes the CMS development method and describes the database. The fourth section describes the results of the system development.

KRB is laid out on 74 pages, it contains 4 sections, 30 illustrations, 16 tables, 17 sources in the list of references

Keywords: website, content management system, Laravel, PHP, database, MySQL, CMS.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ДсанПіН – Державні санітарні норми і правила;

КЗпП – Кодекс законів про працю;

ДНАОП – Державні нормативні акти про охорону праці.

ПА – Панель адміністратора;

CMS – Content Management System – Система управління контентом;

UML (Unified Modeling Language) – уніфікована мова моделювання;

HTML (HyperText Markup Language) – мова розмітки гіпертексту;

CSS (Cascading Style Sheets) – каскадні таблиці стилів;

PHP (Hypertext Preprocessor) – препроцессор гипертекста;

MYSQL – Система керування реляційними базами даних.

ВСТУП

У час, коли інтернет давно став вже дуже потужною та масштабною системою, користувачі почали розуміти, що інформацію важливо фільтрувати та керувати нею для економії часу та продуктивності виконання поставлених задач. Тому, спеціалізованне срощення управління контентом на вебсайтах є сучасною особливістю , яка дозволить не витрачати зайвих зусиль. Досі є зростаюча потреба у зручних та ефективних інструментах для створення систем керування контентом, яка вимагає розробки нових інноваційних та доступних для більшості людей рішень. В такому випадку, вдалим рішенням є використання фреймворку Laravel для створення системи керування вмістом (CMS), що дозволяє забезпечити високу продуктивність та зручність використання, за невеликий час.

Метою кваліфікаційної роботи є розробка системи керування вмістом для оптимізації процесу створення та управління контентом на вебсайтах та блогах, яка буде важити менше, а також працювати швидше.

Об'єктом кваліфікаційної роботи є системи управління контентом для вебсайтів.

Предметом кваліфікаційної роботи є засоби розробки та впровадження системи керування контентом з використанням фреймворку Laravel.

Актуальність даної роботи обумовлена необхідністю забезпечення ефективного та зручного керування вмістом сайтів за допомогою створення спрощенної та більш автоматизованної CMS.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СПЕЦИФІКАЦІЯ ВИМОГ ДО ПЗ

1.1 Аналіз предметної області системи керування вмістом

Системи керування вмістом (Content Management Systems, CMS) є програмними продуктами, що дозволяють користувачам створювати, редагувати, публікувати та керувати цифровим контентом на веб-сайтах. Основною метою використання CMS є забезпечення зручного та інтуїтивно зрозумілого інтерфейсу для роботи з контентом, що дозволяє зменшити залежність від технічних навичок користувачів і зосередитися на змісті.

Перші системи керування вмістом з'явилися у 1990-х роках як відповідь на зростаючу потребу в зручному інструменті для управління веб-контентом. Однією з перших комерційних CMS була Vignette StoryServer, яка з'явилася у 1995 році. Вона дозволяла компаніям створювати та управляти контентом для своїх веб-сайтів без необхідності залучення програмістів для кожної зміни. З тих пір CMS постійно розвивалися, і сьогодні існує багато різних систем, як комерційних, так і з відкритим вихідним кодом.

Основні функції CMS у себе включають:

- Створення контенту. CMS надають користувачам інструменти для створення різних типів контенту, таких як текстові статті, зображення, відео та аудіо файли. Це дозволяє створювати багатий мультимедійний досвід для відвідувачів веб-сайту.

- Редагування контенту. Системи керування вмістом зазвичай включають візуальні редактори, які дозволяють користувачам редагувати контент без необхідності знань HTML або інших мов розмітки.

- Управління користувачами. CMS дозволяють керувати ролями та правами доступу користувачів, що забезпечує контроль над тим, хто може створювати, редагувати або публікувати контент.

- Публікація контенту. Однією з ключових функцій CMS є можливість публікації контенту на веб-сайті з мінімальними зусиллями. CMS автоматично обробляє форматування та структурування контенту відповідно до заздалегідь визначених шаблонів.

- Організація та класифікація. CMS дозволяють організовувати контент у категорії та тегувати його, що полегшує навігацію та пошук інформації на сайті.

- Зберігання та управління версіями. Системи керування вмістом зберігають історію змін контенту, дозволяючи відновлювати попередні версії та відстежувати, хто і коли вносив зміни.

- Пошукова оптимізація (SEO). Багато CMS включають інструменти для оптимізації контенту під пошукові системи, що допомагає поліпшити видимість веб-сайту у пошукових результатах.

- Інтеграція з іншими системами. Сучасні CMS можуть інтегруватися з різними зовнішніми сервісами та додатками, такими як соціальні мережі, аналітичні інструменти, CRM-системи та інші.

З кожним роком розвиток технологій змінює підходи до управління контентом на вебсайтах. Системи управління контентом (CMS) дозволяють автоматизувати процеси створення, редагування та публікації контенту, забезпечуючи зручність для користувачів. Використання фреймворку Laravel для розробки CMS надає розробникам потужні інструменти для швидкої та ефективної розробки вебзастосунків.

Перевагами використання Laravel для розробки CMS є його гнучкість, висока продуктивність та зручність у використанні. Laravel надає багатий набір функцій, таких як маршрутизація, автентифікація, робота з базами даних та інші, що дозволяють створювати потужні та масштабовані застосунки.

Однак, існують і певні недоліки використання Laravel для розробки CMS. Наприклад Laravel може вимагати значних ресурсів сервера, особливо при обробці великої кількості запитів, що може призвести до необхідності додаткових витрат на серверне обладнання або хостинг. І хоча Laravel має велику кількість вбудованих функцій, налаштування деяких з них може бути складним, особливо для новачків. Це може вимагати додаткового часу та зусиль для налаштування та оптимізації для розробників. Ститистика показує, що Laravel є популярним фреймворком у вебсегменті, із великою спільнотою та документацією, але він все ще може мати відносно високу кривизну навчання для новачків у веброзробці або для тих, хто не знайомий з фреймворками PHP. Незважаючи на те, що Laravel добре підходить для багатьох типів проєктів, при дуже великому масштабі (наприклад, для високонавантажених систем) можуть виникати проблеми з продуктивністю, що вимагатиме додаткових оптимізацій або переходу на більш масштабовані рішення.

Суть роботи системи керування контентом (CMS) полягає в тому, щоб надати користувачеві можливість вибрати заздалегідь підготовлений шаблон для оформлення сторінки та заповнити його необхідною інформацією. Велика кількість систем керування вмістом використовують візуальний редактор (WYSIWYG - від англ. What You See Is What You Get - «що бачиш, те й отримаєш»), який дозволяє за допомогою інтуїтивно зрозумілого інтерфейсу додавати або змінювати контент на сайті.

Важливо зазначити, що сайт не є простою сукупністю статичних сторінок, а формується динамічно. Доданий контент зберігається в базі даних, наприклад, MySQL, і використовується для генерації сторінки під час отримання відповідного запиту з боку користувача.

Зазвичай CMS використовуються для таких типів сайтів:

− інтернет-магазини (Magento, OpenCart, osCommerce);

− корпоративні сайти (Joomla, Drupal);

− блоги та форуми (WordPress, phpBB, vBulletin);

− персональні сайти (WordPress, Monstra);

− портали (DLE, Drupal).

− соціальні мережі (InstantCMS, Social Engine);

Безумовно, більшість систем керування вмістом (CMS) мають гнучкі налаштування і можуть бути використані для створення сайтів різного спрямування. Наприклад, найбільш популярною і універсальною є WordPress, яка дозволяє створювати практично будь-який проєкт: від особистого блогу до великого порталу або інтернет-магазину.

Переваги CMS:

− легкість і зручність у використанні;

− широкий набір функцій завдяки плагінам, темам і розширенням;

− швидке створення сайту;

− наявність великої кількості документації.

Недоліки:

− високе споживання ресурсів, особливо при використанні додаткових модулів;

− необхідність постійного оновлення CMS і сумісності версій плагінів;

− популярні CMS часто уразливі;

− не підходять для специфічних завдань.

Бувають також CMS, розроблені на замовлення під спеціальний проєкт, що робить їх функціонал більш вузьким, ніж у масових систем, але вони максимально відповідають поставленим завданням і не містять зайвих інструментів. Індивідуальні CMS використовуються для створення складних і масштабних сайтів зі специфічними вимогами.

Переваги:

− менше навантажує сервер, оскільки не містить зайвих функцій;

− адаптована під конкретні потреби проєкту;

− CMS більш надійна і менш схильна до уразливостей.

Недоліки:

− надається на платній основі, що часто дорожче, ніж придбання ліцензії на одну з популярних CMS;

− для розширення функціоналу або вирішення технічних проблем необхідно звертатися до розробника.

1.2 Аналіз готових рішень

1) WordPress**(https://wordpress.com)** – це готова система керування вмістом (CMS) для створення блогів та невеликих сайтів. WordPress – це найпопулярніша у світі CMS для створення як невеликих сайтів, так і повноцінних блогів.

Переваги:

− Багатофункціональна панель адміністратора. Інтуїтивно зрозумілий інтерфейс, що дозволяє легко керувати вмістом сайту навіть користувачам без технічних знань.

− Велика кількість тем для швидкого створення сайту. Сотні безкоштовних та преміум тем, що дозволяють швидко створити унікальний вигляд сайту без необхідності написання коду.

− Можливість розширення функціоналу через додавання плагінів або створення власних, а також можливість доповнення функціоналу за допомогою коду в самій темі.

Недоліки:

− Великий обсяг пам'яті. WordPress може займати значну кількість дискового простору, особливо якщо встановлено багато плагінів і тем.

− Дуже низька швидкість завантаження сторінок без оптимізації. : Без належної оптимізації сайти на WordPress можуть завантажуватись повільно. Це може бути викликано великою кількістю встановлених плагінів або недостатньо оптимізованими зображеннями.

− Недостатній рівень захисту. Через свою популярність WordPress часто стає мішенню для хакерів. Базова установка WordPress може бути вразливою без додаткових заходів безпеки, таких як регулярне оновлення, використання плагінів для захисту та налаштування брандмауера.

− Залежність від плагінів. Багато функцій потребують встановлення додаткових плагінів, що може призвести до конфліктів між ними або уповільнення роботи сайту.

− Потреба в регулярному обслуговуванні. WordPress вимагає регулярного оновлення ядра системи, тем та плагінів, щоб забезпечити безпеку та стабільність роботи сайту.

CMS WordPress представлена на рисунку 1.1.

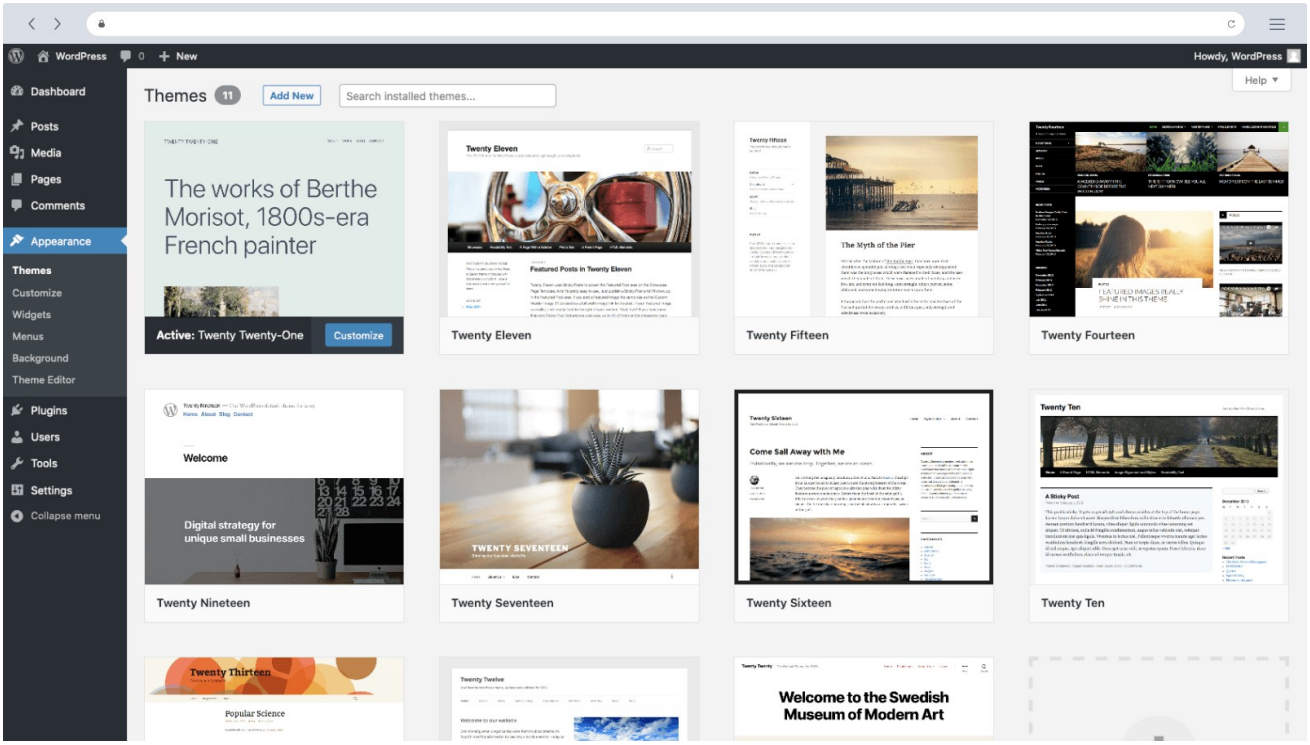

Рисунок 1.1 – CMS WordPress

На рисунку панель керування темами в системі керування вмістом WordPress. Це розділ, де користувачі можуть переглядати, активувати, додавати нові та налаштовувати існуючі теми для свого сайту. Зліва розташована панель навігації, яка містить такі пункти, як Dashboard (Панель керування), Posts (Публікації), Media (Медіа), Pages (Сторінки), Comments (Коментарі), Appearance (Зовнішній вигляд), Plugins (Плагіни), Users (Користувачі), Tools (Інструменти) та Settings (Налаштування). В основній області відображаються встановлені теми, які можна активувати або налаштовувати. Активна тема показана з позначкою "Active" і на зображенні це Twenty Twenty-One. Крім активної теми, представлений список інших встановлених тем, таких як Twenty Eleven, Twenty Fifteen, Twenty Fourteen та інші. Ця панель є важливим інструментом для налаштування зовнішнього вигляду сайту на WordPress. Вона дозволяє легко змінювати теми, експериментувати з дизайном та

налаштовувати кожен аспект зовнішнього вигляду без необхідності знання коду.

2) Drupal (https://www.drupal.org) – це потужна система керування вмістом (CMS) з відкритим вихідним кодом, яка підходить для створення складних веб-сайтів та веб-додатків. Drupal часто використовується для розробки корпоративних сайтів, державних порталів, та інтернет-магазинів завдяки своїй гнучкості та потужним можливостям.

Переваги:

– Гнучкість та масштабованість. Drupal дозволяє створювати як невеликі сайти, так і масштабні корпоративні портали з великим обсягом вмісту та трафіку.

– Високий рівень безпеки. Вважається однією з найбільш безпечних CMS завдяки регулярним оновленням та вбудованим механізмам безпеки, що робить її популярною серед урядових установ та великих підприємств.

– Потужний інструментарій для розробників. Drupal пропонує безліч модулів та API для розширення функціоналу, що дозволяє створювати складні та кастомізовані рішення.

– Гнучка система таксономії. Вбудовані засоби для організації та структурування контенту, що особливо корисно для сайтів з великою кількістю інформації.

– Мультимовність. Повноцінна підтримка створення багатомовних сайтів без необхідності встановлення додаткових плагінів.

– Активна спільнота. Велика міжнародна спільнота розробників, яка постійно працює над поліпшенням платформи, а також надає підтримку та обмінюється досвідом.

Недоліки:

– Складність налаштування та використання. Drupal має круту криву навчання, особливо для новачків. Для ефективного використання необхідні певні технічні знання.

– Ресурсоємність. Для забезпечення високої продуктивності та швидкості роботи сайтів на Drupal часто потрібні потужні сервери та оптимізація.

– Відсутність великої кількості готових тем. На відміну від WordPress, кількість готових тем для Drupal менша, що може вимагати більше часу та ресурсів на створення унікального дизайну.

– Вартість розробки та підтримки. Через свою складність Drupal може потребувати залучення досвідчених розробників, що підвищує вартість розробки та підтримки сайту.

Інтерфейс користувача CMS Drupal представлено на рисунку 1.2. Це сторінка створення нового користувача в адміністративній панелі. У лівій частині екрана знаходиться меню навігації, яке включає розділи для додавання користувачів, управління користувачами, ролями та дозволами. У центральній частині показано форму для створення нового користувача, де можна ввести ім'я користувача, електронну пошту, пароль та завантажити зображення профілю. Справа розташовані додаткові налаштування, такі як статус, ролі, локальні налаштування та контактні налаштування.

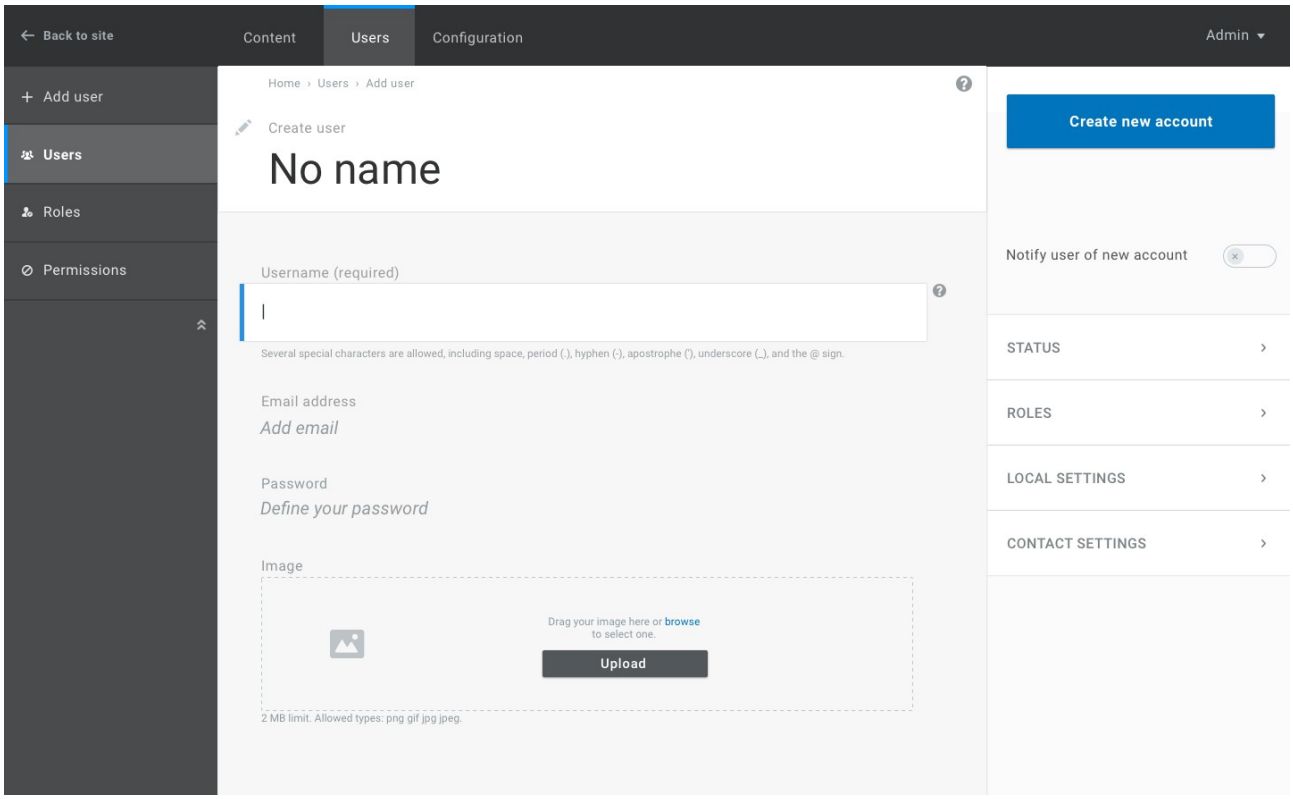

Рисунок 1.2 – Інтерфейс користувача Drupal

Цей інтерфейс дозволяє адміністраторам сайту легко створювати та керувати обліковими записами користувачів. Інтуїтивний інтерфейс адміністрування допомагає швидко та ефективно виконувати задачі з управління користувачами, забезпечуючи високий рівень контролю над доступом до різних частин сайту.

3) Joomla (https://www.joomla.org) – також популярна система керування вмістом (CMS) з відкритим вихідним кодом, яка використовується для створення веб-сайтів різної складності, від простих до корпоративних порталів. Joomla займає проміжне місце між WordPress і Drupal, поєднуючи в собі легкість використання та гнучкість.

Переваги:

– Зручний інтерфейс адміністратора. Joomla пропонує інтуїтивно зрозумілий інтерфейс для керування вмістом, що полегшує роботу навіть користувачам без технічних знань.

– Модульна структура. Система дозволяє легко додавати та налаштовувати модулі для розширення функціоналу сайту.

– Широкі можливості для розробників. Joomla надає потужний інструментарій для розробників, що дозволяє створювати кастомні рішення та інтеграції.

– Підтримка мультимовності. Вбудовані засоби для створення багатомовних сайтів без необхідності встановлення додаткових плагінів.

– Велика кількість розширень. Joomla має велику бібліотеку розширень (плагінів), що дозволяють додавати нові функції та покращувати роботу сайту.

– Спільнота та підтримка. Активна спільнота користувачів і розробників, яка постійно працює над покращенням платформи та надає підтримку.

Недоліки:

– Складність для новачків. Незважаючи на інтуїтивний інтерфейс, Joomla має круту криву навчання для новачків, особливо тих, хто не має технічних знань.

– Вимоги до серверів. Для ефективної роботи сайтів на Joomla можуть знадобитися більш потужні сервери та оптимізація.

– Обмежена кількість готових тем Хоча Joomla пропонує достатню кількість тем, їх менше, ніж для WordPress, що може ускладнити створення унікального дизайну.

– Проблеми сумісності розширень. Деякі розширення можуть конфліктувати між собою, що може призводити до проблем у роботі сайту.

CMS Joomla представлена на рисунку 1.3.

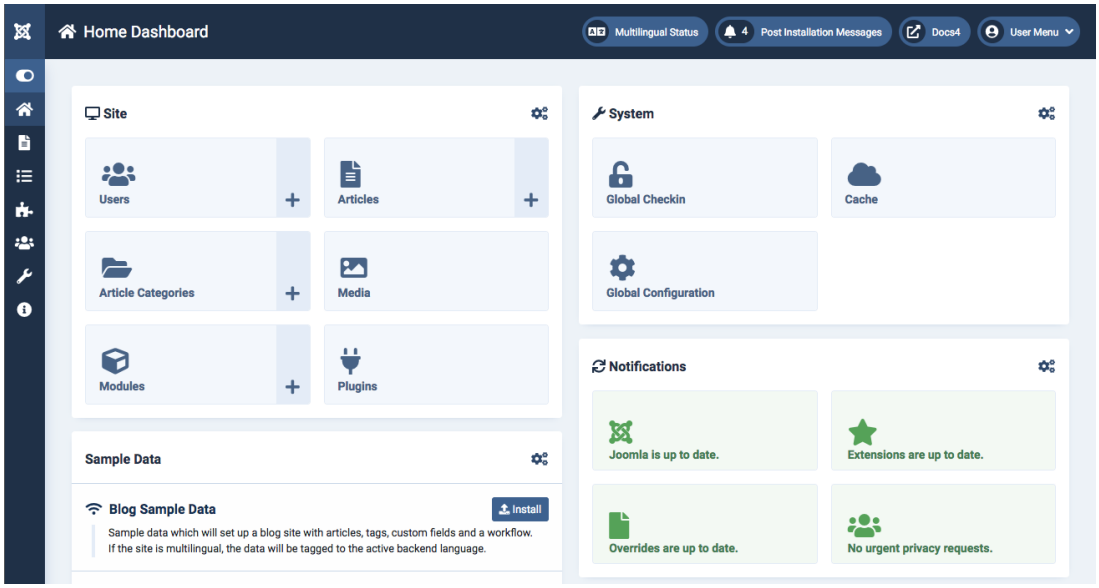

Рисунок 1.3 – Інтерфейс користувача Joomla

На зображенні головна панель керування в системі керування вмістом Joomla. Цей інтерфейс містить різні блоки, які дозволяють адміністраторам сайту легко керувати контентом та налаштуваннями системи. У лівій частині екрану розташоване головне меню навігації, яке включає такі розділи, як Dashboard (Панель керування), Content (Контент), Users (Користувачі), Menus (Меню), Components (Компоненти), Templates (Шаблони), і System (Система). В основній області панелі є кілька блоків, що дозволяють швидкий доступ до найважливіших функцій. Блок "Site" містить швидкі посилання для управління користувачами, статтями, категоріями статей, медіафайлами, модулями та

плагінами. Блок "System" включає такі функції, як глобальний чекін, очищення кешу та глобальні налаштування. Праворуч відображаються повідомлення та статус системи, такі як оновлення Joomla, розширення та інші важливі сповіщення. Цей інтерфейс є зручним і дозволяє адміністраторам легко орієнтуватися та виконувати основні задачі з управління сайтом.

1.3 Постанова задачі

Розробити власну CMS систему для керування невеликим блогом.

Функції сайту для користувача:

− перегляд стрічки постів;

− перегляд окремого поста;

− перегляд категорій;

− перегляд постів за категоріями;

− перегляд постів за тегами;

− авторизація;

− реєстрація;

− управління профілем користувача;

− підписка на нові публікації;

− перегляд популярних постів;

− перегляд відмічених постів;

− можливість залишати коментарі під постами.

Функції сайту для адміністратора:

− перехід до панелі адміністратора;

− управління постами;

− помітка поста як рекомендованого;

− помітка поста як чернетки;

− управління категоріями;

− управління тегами;

− модерування коментарів під постами (відображення коментаря, видалення або приховування);

− управління користувачами;

− призначення користувача адміністратором;

− можливість мінімального налаштування адміністратора.

Вхідні дані:

− інформація про пост;

− інформація про користувача;

− інформація про категорії;

− інформація про теги;

− інформація про коментарі.

Вихідні дані:

− дані про пост;

− сторінка з постами;

− вибірка постів за категоріями;

− вибірка постів за тегами;

− дані користувача;

− список користувачів в панелі адміністратора (ПА);

− список постів в ПА;

− список тегів в ПА;

− список категорій в ПА;

− список коментарів в ПА.

Розробка вебсайту виконується на основі технічного завдання, в якому детально описано вимоги до сайту. Технічне завдання на розробку вебсайту наведено в додатку А.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи бакалавра розглянуто предметну область систем керування вмістом (CMS) та специфікацію вимог до програмного забезпечення. Проведено детальний аналіз сучасних технологій, що використовуються для управління контентом на вебсайтах, зокрема фреймворку Laravel, який надає потужні інструменти для швидкої та ефективної розробки вебзастосунків.

Аналіз переваг і недоліків використання Laravel для розробки CMS показав, що цей фреймворк забезпечує високу гнучкість, продуктивність та зручність у використанні, але може вимагати значних ресурсів сервера та має відносно високу кривизну навчання для новачків. Було розглянуто принцип роботи систем керування вмістом, які надають користувачам можливість вибирати заздалегідь підготовлені шаблони для оформлення сторінок та заповнювати їх необхідною інформацією.

Специфікація вимог до програмного забезпечення включає функції для користувачів (перегляд постів, категорій, тегів, авторизація, реєстрація, коментування) та функції для адміністраторів (управління постами, категоріями, тегами, користувачами, модерування коментарів). Даний розділ заклав основу для подальшої розробки власної CMS, визначивши необхідні функціональні можливості та вимоги до програмного забезпечення.

2 МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ

2.1 Вибір засобів моделювання

Створення системи керування вмістом (CMS) для блогу включає низку ключових етапів, серед яких дизайн макету, розробка фронтенду та бекенду. Розробка CMS являє собою написання індивідуального програмного коду, проте, в зв’язку з тим, що усі CMS мають досить схожу структуру, та в більшості випадків діляться на чіткі вимоги це надає простір для використання готових ідей, які можуть допомогти створити необхідний продукт з базовими функціями під власниу необхідність.

Дизайн передбачає створення візуального вигляду сайту, включаючи вибір шрифтів, кольорової гами, макетів сторінок та логотипу. Це важливий етап, оскільки зовнішній вигляд сайту визначає перше враження користувачів. Спочату планується розробити макет застосунку у редакторі Figma, що допоможе коректно та своєчасно приступити до розробки інтерфейсу без зайвого редагування. У цей етап буде входити також розробка оригінального стилю, вибір колірного поєднання, логотип, типи шрифтів та інше.

Фронтенд розробка включає створення інтерфейсу користувача, використовуючи технології HTML, CSS та Bootstrap вже маючи макетні заготовки майбутнього застосунку. У меті - забезпечити інтерактивність та зручність використання сайту.

Бекенд розробка зосереджується на серверній частині застосунку, що включає управління базами даних, аутентифікацію користувачів, обробку запитів та інші логічні операції. Тут використовується достатньо класична та популярна серверна мова програмування PHP, а також фреймворк Laravel, який забезпечує зручну структуру та численні інструменти для швидкої розробки.

Laravel пропонує такі можливості, як маршрутизація, ORM (Eloquent), система міграцій бази даних, вбудовані засоби для автентифікації та авторизації, а також підтримку RESTful API.

У проєкті буде проводитися перевірка коректної робоит застосунку. Це включатиме перевірку основних функцій, таких як авторизація, створення постів, видалення постів, створенні тегів та категорій, видалення тегів та категорій, робота адміністративної у панелі, взаємодія між компонентами, тощо. Після цього буде виконано огляд коду, дизайну та документації, щоб переконатися у відповідності вимогам. Завершальний етап включає розгортання системи в експлуатацію, моніторинг її роботи та забезпечення технічної підтримки для користувачів.

Найбільш відомими та поширеними візуальними моделями для проєктування комп'ютерних систем та їхнього програмного забезпечення є діаграми, що використовують мову UML. За допомогою UML можна візуалізувати, специфікувати, конструювати та документувати артефакти будь-яких програмних систем: від інформаційних систем підприємства до розподілених вебзастосунків і навіть вбудованих систем реального часу. Для проєктування CMS-системи доцільно використовувати мову моделювання UML щоб спроєктувати архітектуру застосунку.

Використовуються такі діаграми як: діаграма послідовності та кооперації, діаграма компонентів, діаграма розгортання, діаграма варіантів використання, діаграма класів, діаграма станів, діаграма діяльності.

2.2 Розробка контекстної моделі системи

Проєктування програмного забезпечення починається з розробки контекстної діаграми. Контекстна діаграма є початковою моделлю, яка описує

загальні функції системи та її зв'язки з навколишнім середовищем. Для подальшого проєктування необхідно виділити основних користувачів системи:

− користувач;

− адміністратор.

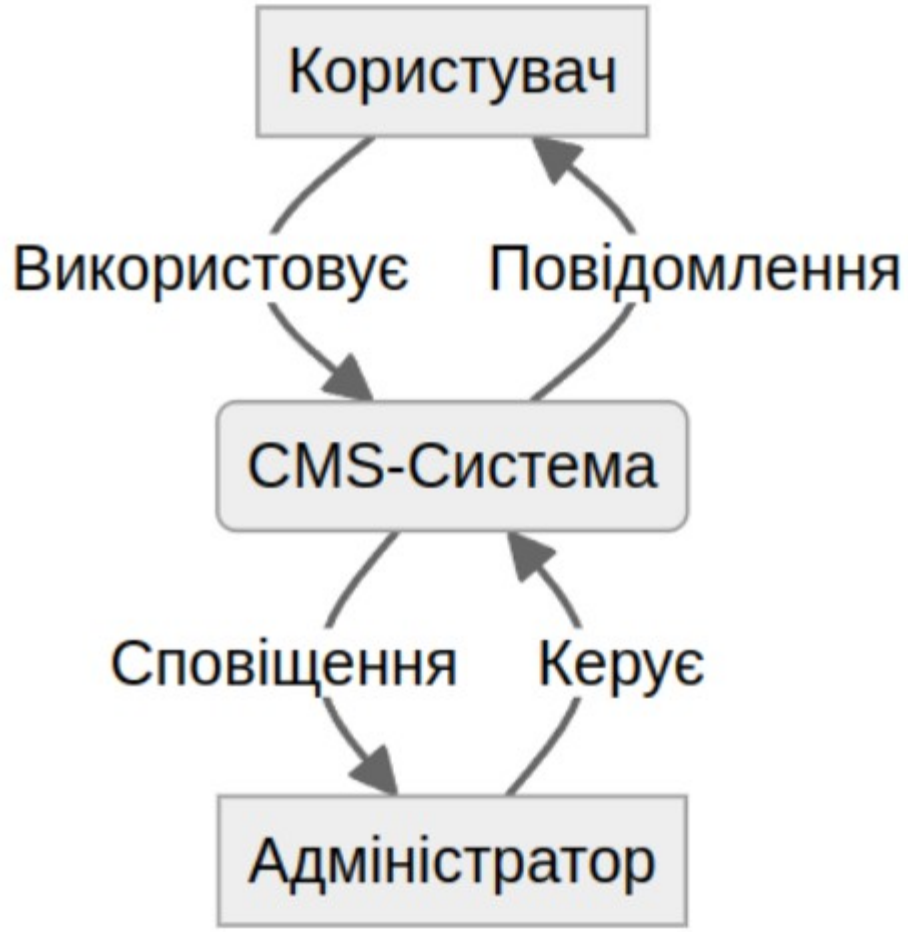

Рисунок 2.1 – Контекстна діаграма

Користувач (А):

- Використовує систему CMS для виконання певних завдань. Це наприклад такі дії як перегляд контенту, редагування особистих даних тощо.

- Отримує повідомлення від системи CMS. Це можуть бути повідомлення про успішне виконання дій, помилки або інша інформація.

CMS-Система (B): Це центральна частина діаграми, яка обробляє всі дії та взаємодії з користувачами та адміністраторами. Система взаємодіє з користувачами, надаючи їм доступ до необхідних функцій та інформації. Система також забезпечує адміністраторів необхідними інструментами та сповіщеннями для ефективного управління системою.

Адміністратор(C):

- Керує системою CMS. Це включає в себе адміністрування користувачів, управління контентом, налаштування системи та інші адміністративні завдання.

Отримує сповіщення від системи CMS. Сповіщення можуть включати в себе важливі системні повідомлення, інформацію про помилки, зміни в системі тощо.

Взаємодії:

- Користувач взаємодіє з CMS-Системою, використовуючи її для своїх потреб.

- Адміністратор взаємодіє з CMS-Системою, керуючи нею та отримуючи необхідні сповіщення для ефективного управління

- CMS-Система відправляє повідомлення користувачу та сповіщення адміністратору для інформування про важливі події та дії.

2.2 Розробка діаграми Use case

При розробці будь-якого програмного забезпечення визначаються вимоги, які повинна виконувати система. Діаграми варіантів використання (use case) показують функціональні можливості системи або те, що система має

робити. Сценарії – це можливі послідовності дій для кожного варіанту використання.

Ця діаграма варіантів використання (use case diagram) для CMS-застосунку відображає взаємодію двох основних типів користувачів: звичайного користувача (User) та адміністратора (Admin) із системою. Вона допомагає візуалізувати та організувати функціональні вимоги до застосунку, полегшуючи розуміння того, як різні ролі взаємодіють із системою.

Діаграма варіантів використання CMS-системи для управління блогом наведена на рисунку 2.2.

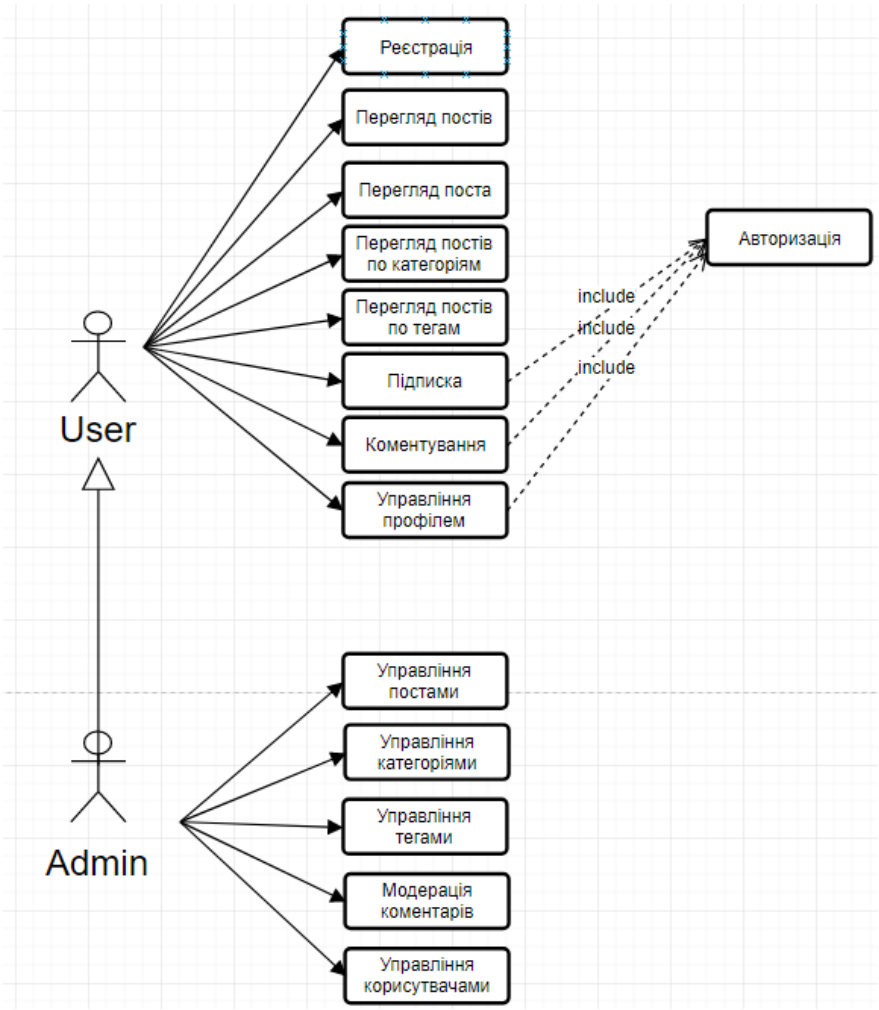

Рисунок 2.2 – Діаграма варіантів використання

Користувач (User) має доступ до наступних функцій: Реєстрація дозволяє користувачеві створити обліковий запис для доступу до функцій, недоступних незареєстрованим користувачам. Перегляд постів дає змогу користувачеві переглядати стрічку постів на головній сторінці. Користувач може переглядати окремі пости, а також фільтрувати їх за категоріями та тегами. Підписка дозволяє користувачеві отримувати сповіщення про оновлення блогу та нові пости. Користувач може залишати коментарі до постів та керувати своїм профілем, редагуючи персональні дані. Для доступу до всіх цих функцій необхідна авторизація.

Адміністратор (Admin) має доступ до розширених функцій для управління системою. Адміністратор може створювати, редагувати та видаляти пости, категорії та теги, забезпечуючи структуру та наповнення контенту. Модерація коментарів дозволяє адміністратору видаляти, приховувати або схвалювати коментарі користувачів, підтримуючи відповідний рівень взаємодії на сайті. Адміністратор також має можливість керувати користувачами системи, надаючи або обмежуючи права доступу, що забезпечує належне функціонування та безпеку системи.

За допомогою цієї діаграми було продемонстровано як користувачі взаємодіють із системою, і які основні дії можуть виконувати. Всі дії користувача передбачають попередню авторизацію, що забезпечує захист даних та доступу до функцій системи. Адміністратор має розширені права доступу для управління контентом та користувачами, забезпечуючи належне функціонування системи.

Наступним кроком буде описано усі елементи діаграми варіантів використання. Специфікації варіантів використання CMS-системи для управління блогом представлені в таблицях 2.1-2.14.

Таблиця 2.1 – Специфікація прецеденту «Реєстрація»

Характеристика	Опис
Короткий опис	Цей варіант використання дозволяє користувачу створити власний профіль для подальшої роботи з функціями, що недоступні для незареєстрованих користувачів.
Учасники	Користувачі
Передумови	Відсутні
Основний потік подій	Користувач ініціює завантаження сторінки реєстрації системою. Вводить необхідні дані. Система перевіряє інформацію на наявність та на коректність, якщо все правильно – зберігає дані.
Альтернативний потік подій	Введені некоректні дані. Користувач вже є в системі. Надання можливості повторного введення даних.

Таблиця 2.2 – Специфікація прецеденту «Авторизація»

Характеристика	Опис
Короткий опис	Цей варіант використання дозволяє користувачу увійти на сайт за допомогою пошти та паролю, які були вказані під час реєстрації, або зміни у профілі.
Учасники	Користувачі
Передумови	Відсутні
Основний потік подій	Користувач ініціює завантаження сторінки авторизації системою. Вводить необхідні дані. Система перевіряє інформацію на наявність та на коректність, якщо все правильно – дає доступ до функцій авторизованого користувача.
Альтернативний потік подій	Введені некоректні дані. Надання можливості повторного введення даних.

Окрім того, для підвищення зручності користувачів можна додати функції, такі як запам'ятовування користувача на пристрої (Remember Me), відновлення паролю через електронну пошту та інтеграцію з соціальними мережами для швидкої авторизації.

Таблиця 2.3 – Специфікація прецеденту «Перегляд постів»

Характеристика	Опис
Короткий опис	Цей варіант використання дозволяє користувачеві переглядати стрічку постів на головній сторінці сайту.
Учасники	Користувачі
Передумови	Відсутні
Основний потік подій	Користувач ініціює завантаження головної сторінки системою. Система перевіряє інформацію на наявність та коректність, якщо все правильно – виводить пости.
Альтернативний потік подій	Виведення повідомлення «Немає постів».

Таблиця 2.4 – Специфікація прецеденту «Перегляд поста»

Характеристика	Опис
Короткий опис	Цей варіант використання дозволяє користувачеві переглядати сторінку конкретного поста.
Учасники	Користувачі
Передумови	Відсутні
Основний потік подій	Користувач ініціює завантаження сторінки одного посту системою. Система перевіряє інформацію на наявність та коректність, якщо все правильно – виводить контент посту.
Альтернативний потік подій	Редирект на головну сторінку з повідомленням «Помилка».

Для покращення взаємодії користувачів можна додати функції, такі як кнопки навігації до попереднього або наступного посту, можливість лайкнути або поділитися постом, а також відображення пов'язаних постів.

Таблиця 2.5 – Специфікація прецеденту «Перегляд постів по категоріям»

Характеристика	Опис
Короткий опис	Цей варіант використання дозволяє користувачеві переглядати сторінку постів за категоріями.
Учасники	Користувачі
Передумови	Відсутні
Основний потік подій	Користувач ініціює завантаження сторінки постів за категорією системою. Система перевіряє інформацію на наявність та коректність, якщо все правильно – виводить стрічку з постами.
Альтернативний потік подій	Редирект на головну сторінку з повідомленням «Помилка».

Таблиця 2.6 – Специфікація прецеденту «Перегляд постів по тегам»

Характеристика	Опис
Короткий опис	Цей варіант використання дозволяє користувачеві переглядати сторінку постів за тегами.
Учасники	Користувачі
Передумови	Відсутні
Основний потік подій	Користувач ініціює завантаження сторінки постів за тегом системою. Система перевіряє інформацію на наявність та коректність, якщо все правильно – виводить стрічку з постами.
Альтернативний потік подій	Виведення повідомлення «Немає постів».

Для покращення взаємодії користувачів можна додати функції, такі як сортування та фільтрація постів за тегами, відображення популярних тегів, а також пропозиції схожих тегів.

Таблиця 2.7 – Специфікація прецеденту «Підписка»

Характеристика	Опис
Короткий опис	Цей варіант використання дозволяє користувачеві підписуватися на оновлення блогу, а також на додавання нових постів.
Учасники	Користувачі
Передумови	Виконання варіанту використання «Авторизація»
Основний потік подій	Користувач ініціює завантаження головної сторінки. Вводить у відповідну форму дані. Система перевіряє дані на наявність та на коректність, якщо все правильно – зберігає інформацію.
Альтернативний потік подій	Введені некоректні дані. Надання можливості повторного введення даних.

Таблиця 2.8 – Специфікація прецеденту «Коментування»

Характеристика	Опис
Короткий опис	Цей варіант використання дозволяє користувачеві коментувати записи в блозі.
Учасники	Користувачі
Передумови	Виконання варіанту використання «Авторизація». Виконання варіанту використання «Перегляд поста»
Основний потік подій	Користувач ініціює завантаження сторінки одного посту системою. Вводить текст коментаря у відповідне поле. Система перевіряє інформацію на коректність, якщо все правильно – додає коментар до посту.
Альтернативний потік подій	Введені некоректні дані. Надання можливості повторного введення даних.

Додатково можна також додати користувачам можливість отримувати сповіщення про відповіді на їхні коментарі, що сприяє активнішій участі у дискусіях.

Таблиця 2.9 – Специфікація прецеденту «Управління профілем»

Характеристика	Опис
Короткий опис	Цей варіант використання дозволяє користувачеві переглядати та редагувати свій профіль користувача.
Учасники	Користувачі
Передумови	Виконання варіанту використання «Авторизація»
Основний потік подій	Користувач ініціює завантаження сторінки профілю системою. Система перевіряє інформацію на коректність та наявність, якщо все правильно – виводить дані користувача.
Альтернативний потік подій	Редирект на головну сторінку з повідомленням «Помилка».

Таблиця 2.10 – Специфікація прецеденту «Управління постами»

Характеристика	Опис
Короткий опис	Цей варіант використання дозволяє адміністратору створювати, редагувати та видаляти пост у ПА.
Учасники	Адміністратор
Передумови	Виконання варіанту використання «Авторизація»
Основний потік подій	Адміністратор ініціює завантаження сторінки створення поста системою. Вводить у спеціальні поля дані. Система перевіряє інформацію на коректність, якщо все правильно – створює пост.
Альтернативний потік подій	Введені некоректні дані. Надання можливості повторного введення даних.

Також, додатково можна впровадити функцію автозбереження для запобігання втраті даних під час редагування та додати функцію для запланованої публікації постів на визначену дату та час.

Таблиця 2.11 – Специфікація прецеденту «Управління категоріями»

Характеристика	Опис
Короткий опис	Цей варіант використання дозволяє адміністратору створювати, редагувати та видаляти категорію у ПА.
Учасники	Адміністратор
Передумови	Виконання варіанту використання «Авторизація»
Основний потік подій	Адміністратор ініціює завантаження сторінки створення категорії системою. Вводить у спеціальні поля дані. Система перевіряє інформацію на коректність, якщо все правильно – створює категорію.
Альтернативний потік подій	Введені некоректні дані. Надання можливості повторного введення даних.

Таблиця 2.12 – Специфікація прецеденту «Управління тегами»

Характеристика	Опис
Короткий опис	Цей варіант використання дозволяє адміністратору створювати, редагувати та видаляти тег у ПА.
Учасники	Адміністратор
Передумови	Виконання варіанту використання «Авторизація»
Основний потік подій	Адміністратор ініціює завантаження сторінки створення тегу системою. Вводить у спеціальні поля дані. Система перевіряє інформацію на коректність, якщо все правильно – створює тег.
Альтернативний потік подій	Введені некоректні дані. Надання можливості повторного введення даних.

Таблиця 2.13 – Специфікація прецеденту «Модерація коментарів»

Характеристика	Опис
Короткий опис	Цей варіант використання дозволяє адміністратору модерувати коментарі (блокувати або відображати у постах) у ПА.
Учасники	Адміністратор
Передумови	Виконання варіанту використання «Авторизація»
Основний потік подій	Адміністратор ініціює завантаження сторінки з усіма коментарями системою. Натискає на кнопку блокування, якщо коментар неприпустимий. Натискає на кнопку схвалення, якщо коментар прийнятний. Система перевіряє дані на наявність, якщо все правильно – виконує дію відповідної кнопки.
Альтернативний потік подій	Повідомлення з написом «Помилка».

Таблиця 2.14 – Специфікація прецеденту «Управління користувачами»

Характеристика	Опис
Короткий опис	Цей варіант використання дозволяє адміністратору переглядати, редагувати, створювати та видаляти користувачів у ПА.
Учасники	Адміністратор
Передумови	Виконання варіанту використання «Авторизація»
Основний потік подій	Адміністратор ініціює завантаження сторінки зі списком користувачів системою. Система перевіряє інформацію на наявність, якщо все правильно – відображає список користувачів.
Альтернативний потік подій	Редирект із повідомленням «Помилка».

2.3 Концептуальна модель

Концептуальна модель – це абстрактне представлення системи або предметної області, яке включає перелік взаємопов'язаних понять, що використовуються для опису цієї області. Вона містить властивості та характеристики цих понять, їх класифікацію за типами, ситуаціями та ознаками, а також закони, що визначають процеси в цій області. Концептуальна модель допомагає зрозуміти структуру і функції системи на високому рівні абстракції, без деталізації технічних аспектів.

Основні характеристики концептуальної моделі:

1. Поняття та об'єкти. Включає основні поняття та об'єкти, які є ключовими для даної системи або предметної області.

2. Зв'язки та відносини. Визначає взаємозв'язки між різними поняттями та об'єктами.

3. Класифікація та властивості. Описує класифікацію об'єктів за певними критеріями, їхні властивості та характеристики.

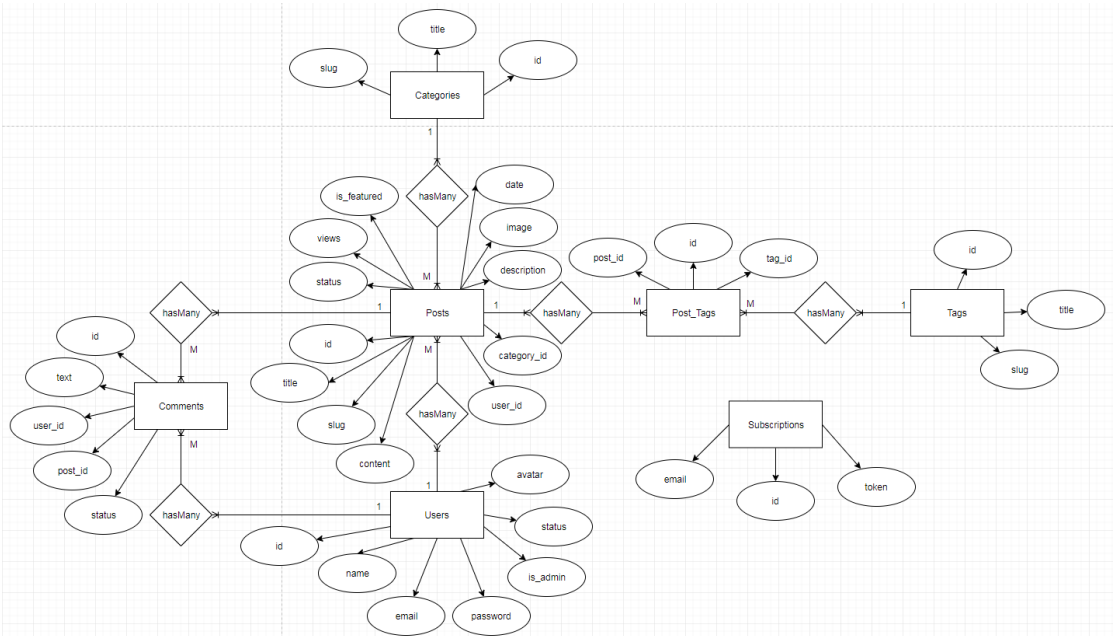

Рисунок 2.2 – Концептуальна модель CMS-системи

Ця концептуальна діаграма відображає основні сутності та їхні взаємозв'язки у системі керування вмістом (CMS) для блогу. Основними сутностями є Users, Posts, Comments, Categories, Tags, Post_Tags, та Subscriptions.

Users представляють користувачів системи, маючи атрибути, такі як id, name, email, password, avatar, status, і is_admin. Користувач може мати багато постів та коментарів. Posts є центральною сутністю, яка має атрибути, як-от id, title, slug, content, views, status, date, image, description, category_id, user_id, і is_featured. Пост може мати багато коментарів, тегів, і належати до однієї категорії.

Comments відображають коментарі, що користувачі залишають під постами, з атрибутами id, text, user_id, post_id, та status. Один пост може мати багато коментарів. Categories представляють категорії, які мають атрибути id, title, і slug, і можуть містити багато постів.

Tags відображають теги з атрибутами id, title, і slug. Один тег може бути асоційований з багатьма постами через таблицю Post_Tags, яка містить атрибути id, post_id, і tag_id. Subscriptions представляють підписки користувачів на оновлення, з атрибутами id, email, і token.

Зв'язки між сутностями включають відносини "один до багатьох" (hasMany) між користувачами та постами, постами та коментарями, категоріями та постами, а також "багато до багатьох" (M) між постами і тегами через проміжну таблицю Post_Tags. Таким чином ця діаграма допомагає зрозуміти структуру бази даних, зв'язки між даними та забезпечує основи для подальшої розробки системи.

У таблиці 2.15 наведена схема зв`язків між сутностями.

Таблиця 2.15 – Зв'язки між сутностями

Сутності	Зв'язок	Пояснення
Posts Categories	M:1	Одна категорія містить багато постів
Posts Comments	1:M	Один пост має багато коментарів
Users Comments	1:M	Один користувач може писати багато коментарів
Users - Posts	1:M	Один користувач може створювати багато постів
Posts - Tags	M:M	Один пост має багато тегів Один тег містить багато постів

Ця таблиця описує зв'язки між основними сутностями у системі керування вмістом (CMS) для блогу. Вона допомагає зрозуміти, як різні об'єкти в системі взаємодіють між собою, показуючи типи зв'язків та пояснення до них.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи бакалавра розглянуто етапи розробки системи керування вмістом (CMS) для блогу, починаючи від вибору засобів моделювання та розробки контекстної моделі системи. Основним акцентом було надано проектуванню архітектури застосунку, плануванню створення макетів та інструментів для розробки фронтенду і бекенду.

Розглянуто вибір засобів моделювання, таких як UML-діаграми, які допомагають візуалізувати, специфікувати, конструювати та документувати різні аспекти програмного забезпечення. Використання UML діаграм дозволило чітко визначити функціональні можливості системи, взаємодії між

користувачами та системою, а також основні процеси, що відбуваються в системі.

Розробка контекстної діаграми та діаграми варіантів використання показала загальну структуру системи та взаємозв'язки між користувачами (користувачами та адміністраторами) та системою CMS. Контекстна діаграма допомогла визначити загальні функції системи та її зв'язки з навколишнім середовищем, а діаграма варіантів використання продемонструвала основні дії користувачів і адміністраторів у системі.

Розробка концептуальної моделі та діаграм зв'язків між сутностями (ER-діаграм) надала детальне уявлення про структуру бази даних і взаємозв'язки між основними об'єктами системи. Ця інформація є необхідною для подальшої реалізації системи, оскільки забезпечує розробникам чітке розуміння архітектури даних та логіки взаємодії між компонентами системи.

Таким чином, у другому розділі було закладено основи моделювання системи для подальшої розробки системи керування вмістом (CMS) для блогу.

3 АРХІТЕКТУРА, МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ЗАСТОСУНКУ

3.1 Архітектура системи керування вмістом (CMS) для блогу

Розробка архітектури для системи керування вмістом (CMS) є одним із найважливіших етапів у процесі створення програмного забезпечення. Архітектура визначає основну структуру системи, її компоненти, зв'язки між ними, а також принципи та стандарти, які будуть використані при розробці та підтримці системи. Це допомагає розробникам планувати та організовувати роботу над проектом. Чітке визначення структури системи дозволяє краще розподілити завдання між членами команди, визначити необхідні ресурси та терміни виконання робіт. Це сприяє більш ефективному використанню часу та ресурсів.

Добре спроектована архітектура забезпечує масштабованість системи, дозволяючи легко додавати нові функції та компоненти без значних змін у базовій структурі. Це також важливо для проектів, які можуть розширюватися та розвиватися з часом, відповідно до зростаючих вимог бізнесу та користувачів.

Архітектура визначає стандарти та принципи розробки, що сприяє забезпеченню якості та надійності програмного забезпечення. Використання відомих архітектурних патернів та підходів дозволяє уникнути типових помилок та забезпечує стабільну роботу системи у різних умовах.

Архітектура системи керування вмістом (CMS) для блогу базується на фреймворку Laravel, який є потужним і гнучким інструментом для створення веб-застосунків. Система складається з декількох основних компонентів,

кожен з яких відіграє важливу роль у забезпеченні функціональності та зручності використання.

Контролери

Контролери відповідають за обробку запитів та взаємодію з моделями і видами. Наприклад, контролери для роботи з категоріями, коментарями, постами, тегами та користувачами реалізують основні CRUD операції (створення, читання, оновлення, видалення).

Middleware

Middleware використовується для перевірки автентифікації користувачів, авторизації доступу до певних функцій та інших проміжних операцій, необхідних для безпечної та коректної роботи системи.

Сервіси

Сервісні класи дозволяють відокремити логіку бізнес-процесів від контролерів, забезпечуючи більш чистий та зрозумілий код.

3.2 Діаграма станів

Діаграма станів демонструє, як об’єкт переходить з одного стану в інший та служить для моделювання динамічних аспектів системи. Проаналізувавши, CMS-систему для управління блогом, розглянув всі можливі умови, при виконанні яких об’єкт або система буде виконувати певні дії чи перебувати в очікуванні появи іншої події. Діаграма станів CMS-системи для адміністратора продемонстрована на рисунках 3.1 та 3.2.

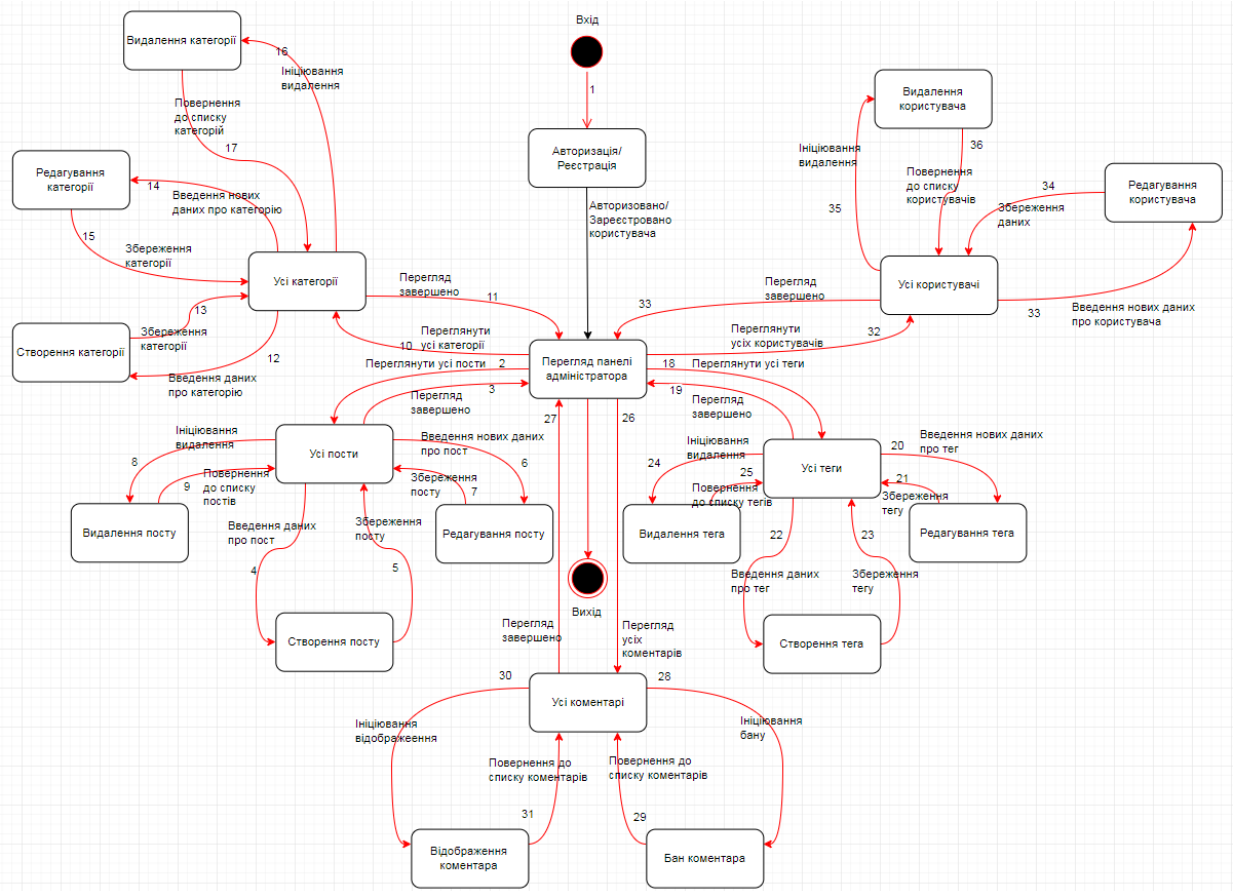

Рисунок 3.1 – Діаграма переходів станів CMS-системи для адміністратора

Ця діаграма представляє процеси управління адміністративною панеллю вебзастосунку. Розпочинається все з авторизації/реєстрації користувача (стан 1). Після успішної авторизації користувач переходить до перегляду панелі адміністратора (стан 2). Звідси є кілька шляхів для подальших дій.

Користувач може переглянути всі категорії (стан 10), усі пости (стан 11), усі коментарі (стан 12), усі теги (стан 19) або всіх користувачів (стан 32).

У стані "Усі категорії" (стан 10) можна створити категорію (стан 13), редагувати категорію (стан 14) або видалити категорію (стан 15). Після

виконання цих дій відбувається збереження даних або повернення до списку категорій.

У стані "Усі пости" (стан 11) можливі дії: створення посту (стан 5), редагування посту (стан 6) або видалення посту (стан 8). Введення або редагування даних посту супроводжується збереженням, після чого користувач повертається до списку постів.

У стані "Усі коментарі" (стан 12) користувач може ініціювати відображення конкретного коментаря (стан 30), або заблокувати коментар (стан 29). Відображення коментаря приводить до повернення до списку коментарів.

У стані "Усі теги" (стан 19) можна створити тег (стан 20), редагувати тег (стан 21) або видалити тег (стан 22). Після кожної дії користувач повертається до списку тегів.

У стані "Усі користувачі" (стан 32) можливе редагування даних користувача (стан 33) або видалення користувача (стан 35). Після цих дій користувач повертається до списку користувачів. З кожного стану можна повернутись назад до попереднього стану або до панелі адміністратора, якщо перегляд завершено. Основні процеси завершуються поверненням до відповідних списків об'єктів (категорій, постів, коментарів, тегів, користувачів).

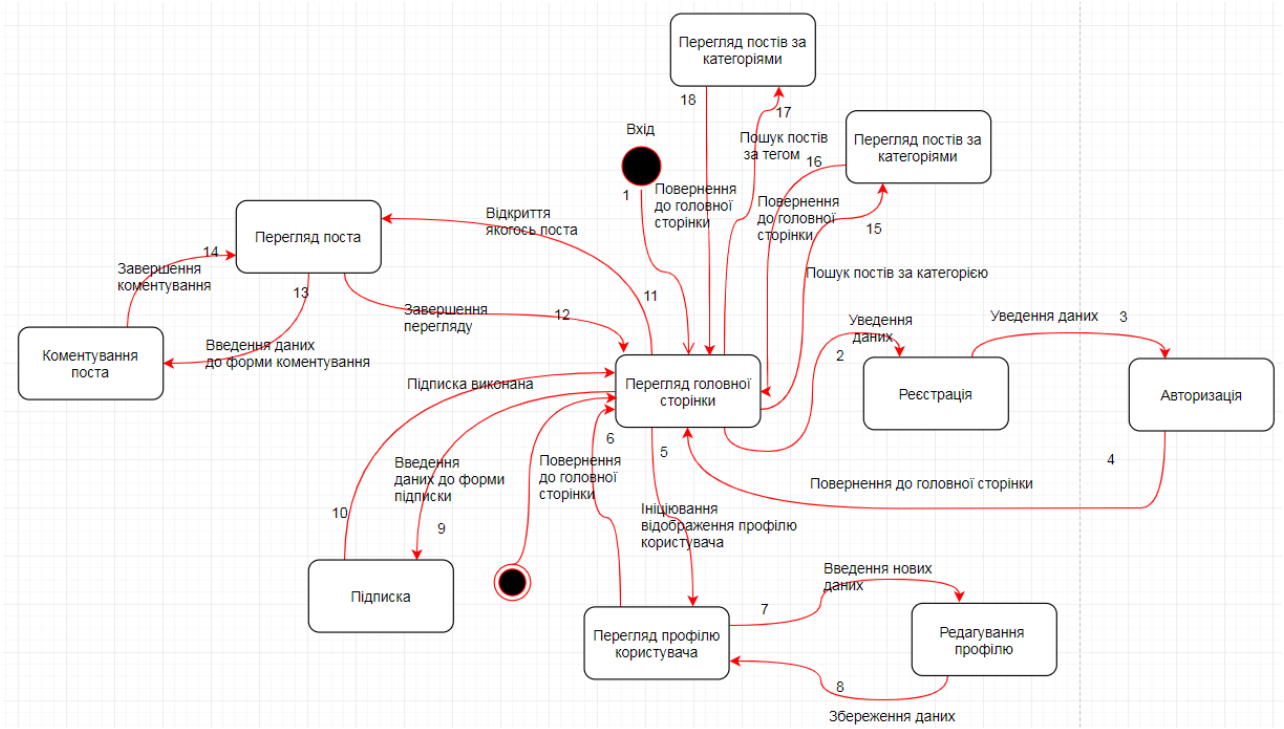

Рисунок 3.2 – Діграма переходів станів CMS-системи для користувача

Ця діаграма станів представляє процеси взаємодії користувача з вебзастосунком, починаючи від авторизації та реєстрації до перегляду постів і коментування. Користувач починає з авторизації (стан 1) або реєстрації (стан 2). Після введення даних (стан 3), він переходить до перегляду головної сторінки (стан 11). З головної сторінки є кілька варіантів дій.

Користувач може ініціювати відображення профілю (стан 5), де може редагувати профіль (стан 7) з подальшим збереженням даних (стан 8) і поверненням до головної сторінки.

Також користувач може здійснити підписку (стан 9), вводячи дані до форми підписки (стан 10), і після завершення підписки (стан 12) повертається до головної сторінки. Користувач має можливість переглядати пости за категоріями (стан 15) або тегами (стан 16).

Пошук постів за категоріями (стан 17) також приводить до перегляду постів за категоріями (стан 18) і повернення до головної сторінки.

При перегляді поста (стан 13), користувач може коментувати його (стан 14), вводячи дані до форми коментування (стан 13), і завершувати коментування (стан 14).

Також можливе відкриття будь-якого поста з головної сторінки (стан 11) для його перегляду (стан 13). Завершення перегляду поста (стан 12) також повертає користувача до головної сторінки.

3.3 Діаграма послідовності

Діаграма послідовності (sequence diagram) є основним способом відображення взаємодії об'єктів в часі. Вона потрібна для візуалізації процесу взаємодії між різними об'єктами в системі протягом певного сценарію. Вона підвищує розуміння процесу, чітко показуючи, які об'єкти взаємодіють між собою і в якому порядку, що допомагає всім учасникам проекту (розробникам, тестувальникам, аналітикам) краще зрозуміти логіку роботи системи.

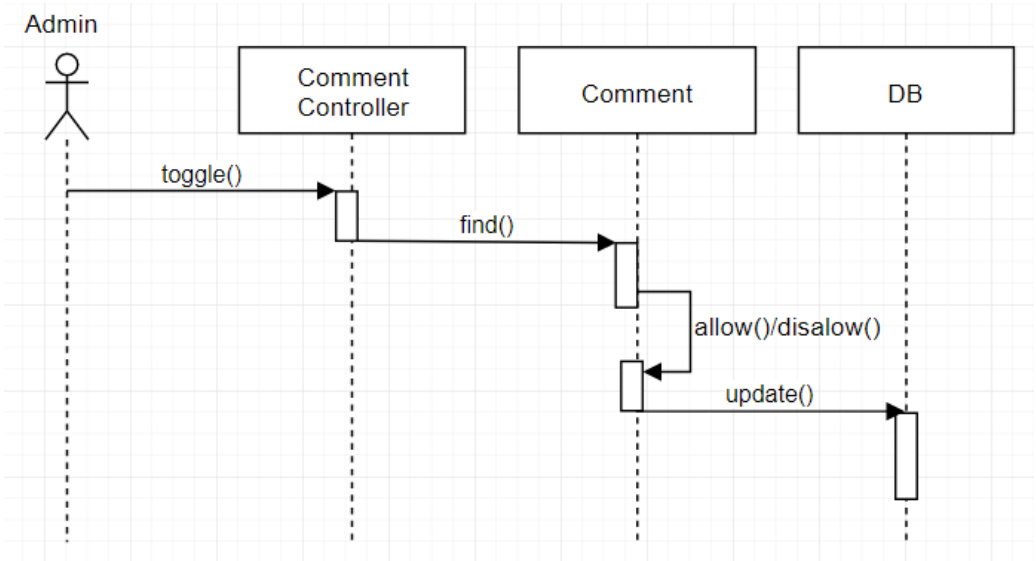

Рисунок 3.3 – Діаграма послідовностей для варіанту використання «Модерування коментара»

Ця діаграма послідовностей ілюструє процес, в якому адміністратор змінює статус коментаря. Адміністратор ініціює дію, викликаючи метод toggle() у Comment Controller. Comment Controller виконує метод find(), щоб знайти відповідний коментар у системі. Після цього об'єкт Comment виконує методи allow() або disallow() для зміни свого статусу. Змінений статус коментаря зберігається у базі даних через метод update().

3.4 Діаграма кооперації

Поняття кооперації (collaboration) є одним з фундаментальних понять у мові UML. Воно служить для позначення безлічі взаємодіючих з певною метою об'єктів в загальному контексті модельованої системи. Мета самої кооперації полягає в тому, щоб специфікувати особливості реалізації окремих найбільш значущих операцій в системі. Кооперація визначає структуру поведінки системи в термінах взаємодії учасників цієї кооперації. Діаграма кооперації для варіанту використання «Створення поста» представлена на рисунку 2.8.

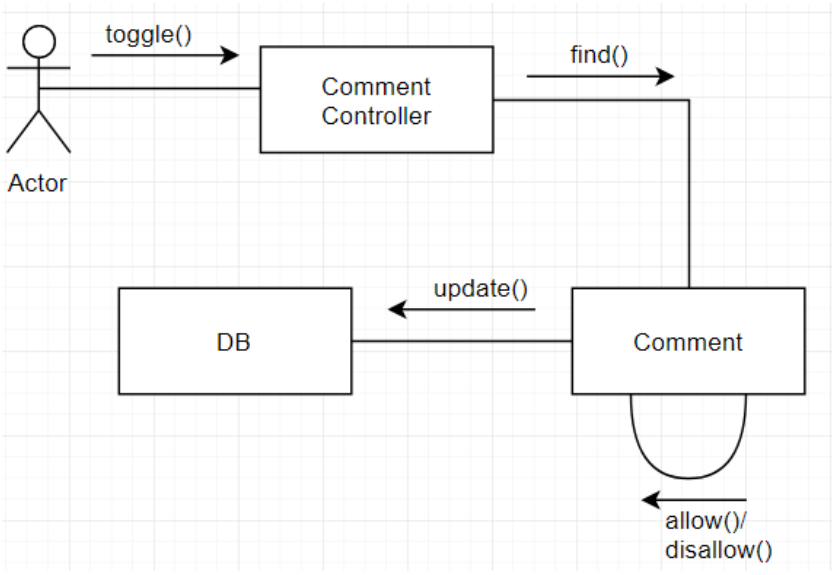

Рисунок 3.4 - Діаграма кооперації для варіанту використання «Модерування коментара»

Наведена діаграма кооперації для варіанту використання "Модерування коментара" ілюструє процес, в якому адміністратор ініціює зміну статусу коментаря. Адміністратор викликає метод toggle() у контролері коментарів (Comment Controller). Контролер коментарів знаходить потрібний коментар за допомогою методу find(). Після цього коментар (Comment) перевіряється на можливість зміни статусу, викликаючи методи allow() або disallow(). Змінений статус коментаря передається до бази даних (DB) через метод update(). Діаграма демонструє послідовність і взаємодію між об'єктами адміністратор, контролер коментарів, коментар і база даних для успішного модерації коментаря.

3.5 Діаграми діяльності

При моделюванні поведінки системи виникає необхідність деталізувати особливості алгоритмічної та логічної реалізації операцій. Традиційно з цією
метою використовують блок-схеми або структурні схеми алгоритмів. Кожнатака схема акцентує увагу на послідовності виконання певних дій (або елементарних операцій), які в сукупності спричинюють до отримання бажаногорезультату. Для моделювання процесу виконання операцій у мові UMLвикористовують діаграму діяльності, яка зображається графом, вершинами якого є стани (дій і/або видів діяльності), а дугами – переходи від одного стану (дій/виду діяльності) до іншого стану (дій/виду діяльності). На рисунку 3.5 зображена діаграма діяльності для варіанту використання «Модерування коментара».

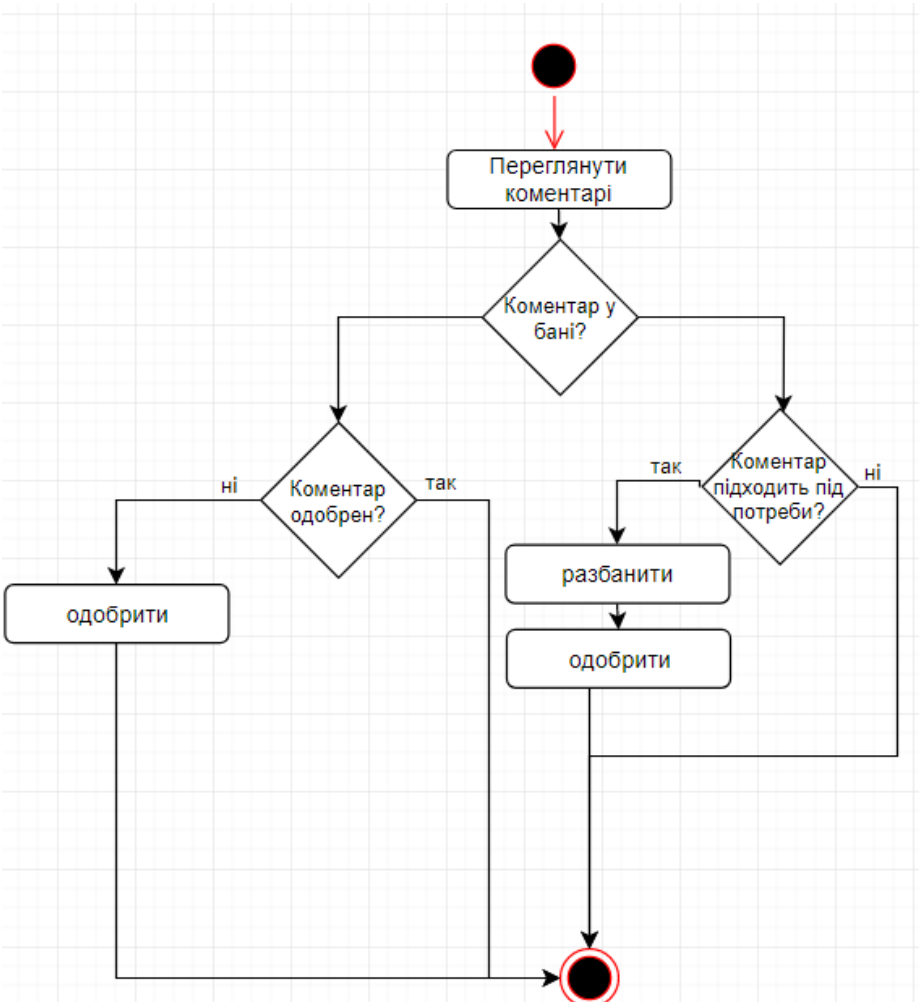

Рисунок 3.6 – Діаграма діяльності для варіанту використання «Модерування коментара»

Ця діаграма діяльності описує процес модерації коментарів. Процес починається з перегляду коментарів. Якщо коментар не в бані, модератор перевіряє, чи він уже схвалений. Якщо так, процес завершується; якщо ні, коментар схвалюється. Якщо коментар у бані, перевіряється, чи підходить він під потреби. Якщо так, коментар розблоковується і схвалюється, якщо ні — процес завершується.

3.6 Діаграма класів

Діаграма класів є важливим інструментом в розробці програмного забезпечення, оскільки вона дозволяє розібрати систему на окремі компоненти і зрозуміти, як вони взаємодіють між собоювізуалізує структуру системи, показуючи класи, їхні атрибути, методи та зв’язки між ними. Кожен клас на діаграмі класу має назву, яка ідентифікує його. Назви класів зазвичай вказують на їх роль або суть в системі..

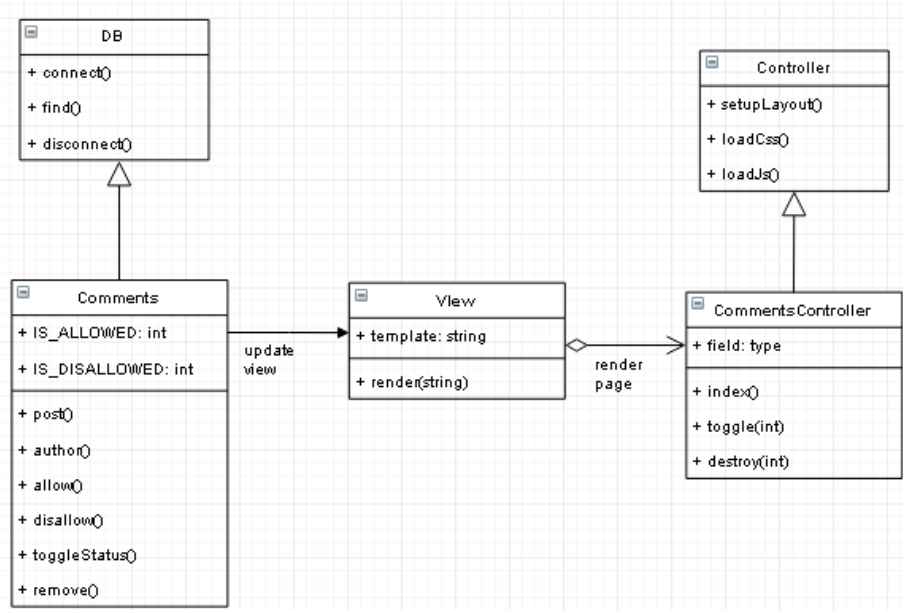

Рисунок 3.7 – Діаграмма класів для прецеденту «Модерування коментара»

Наведена діаграма класів описує процес модерації коментарів у блозі. Клас DB забезпечує з'єднання з базою даних для доступу до коментарів. Клас Comments управляє коментарями, дозволяючи їх додавання, видалення та зміну статусу. Після зміни статусу коментарів, клас View оновлює відображення сторінки, використовуючи шаблони для рендерингу. Клас

Controller налаштовує загальний макет сторінки, завантажує CSS-стилі та JavaScript-скрипти. Клас CommentsController керує логікою відображення списку коментарів, змінює їх статус і видаляє коментарі, взаємодіючи з класом View для рендерингу сторінок.

Таблиця 3.1 для діаграми класів, що стосується прецеденту «Модерування коментарів»:

Клас	Атрибути	Методи	Взаємодії
DB (База Даних)		connect(), find(), disconnect()	Використовується класом Comments для доступу до даних.
Comments (Коментарі)	IS_ALLOWED: int, IS_DISALLOWED: int	post(), author(), allow(), disallow(), toggleStatus(), remove()	Оновлює клас View після зміни статусу коментарів.
View (Відображення)	template: string	render(string)	Оновлюється класом Comments. Рендерить сторінку для класу CommentsController.
Controller (Контролер)		setupLayout(), loadCss(), loadJs()	Використовується класом CommentsController для налаштування сторінки.
CommentsController (Контролер Коментарів)	field: type	index(), toggle(int), destroy(int)	Використовує клас View для рендерингу сторінок.

3.7 Розробка фізичної моделі

Фізична модель даних — це детальний план структури бази даних, який включає інформацію про таблиці, стовпці, індекси, обмеження, типи даних і взаємозв'язки між таблицями. Вона потрібна для визначення структури

даних, оптимізації продуктивності, реалізації логічної моделі, управління та підтримки бази даних, а також забезпечення цілісності даних.

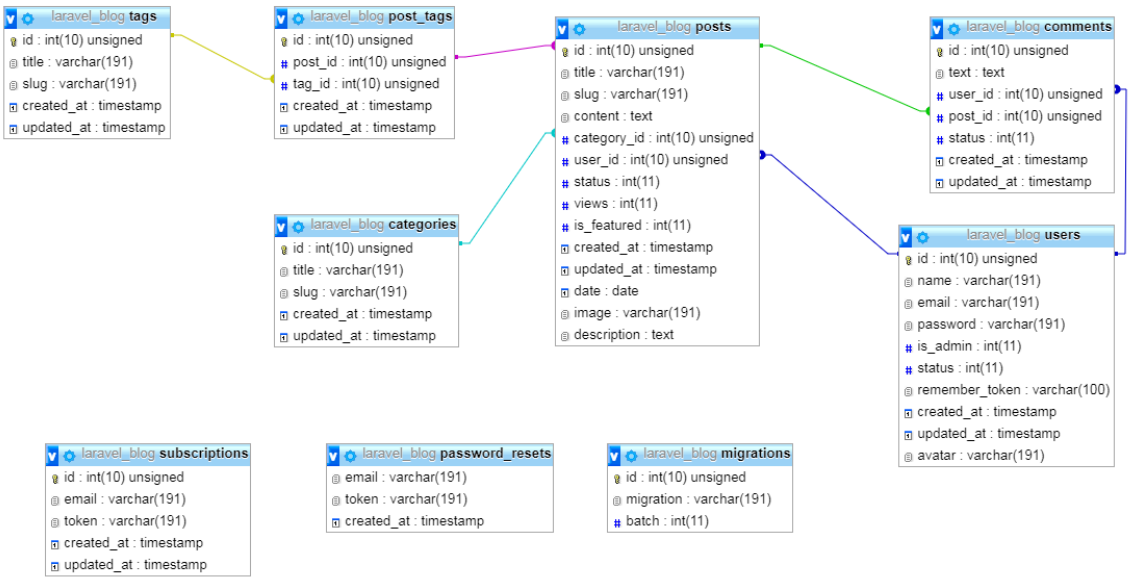

Рисунок 3.8 – Фізична модель CMS-системи

На представленій діаграмі відображено фізичну модель бази даних для блогу, яка включає кілька таблиць:

- Таблиця laravel_blog_tags зберігає інформацію про теги з такими атрибутами: id, title, slug, created_at, updated_at.

- Таблиця laravel_blog_post_tags використовується для зв'язку постів з тегами і містить атрибути: id, post_id, tag_id, created_at, updated_at.

- Таблиця laravel_blog_posts зберігає інформацію про пости з атрибутами: id, title, slug, content, category_id, user_id, status, views, is_featured, created_at, updated_at, date, image, description.

- Таблиця laravel_blog_categories зберігає категорії з атрибутами: id, title, slug, created_at, updated_at.

- Таблиця laravel_blog_comments зберігає коментарі з атрибутами: id, text, user_id, post_id, status, created_at, updated_at.

- Таблиця laravel_blog_users зберігає інформацію про користувачів з атрибутами: id, name, email, password, is_admin, status, remember_token, created_at, updated_at, avatar. Таблиця laravel_blog_subscriptions містить підписки з атрибутами: id, email, created_at, updated_at.

- Таблиця laravel_blog_password_resets зберігає інформацію для скидання паролів з атрибутами: email, token, created_at.

- Таблиця laravel_blog_migrations зберігає міграції з атрибутами: id, migration, batch.

Ця модель визначає точні структури таблиць, включаючи їх атрибути та взаємозв'язки, що полегшує управління базою даних, її підтримку, оптимізацію та забезпечення цілісності даних.

3.8 Розробка діаграми розгортання

Діаграма розгортання (Deployment Diagram) є одним з типів діаграм в мові UML (Unified Modeling Language), яка відображає фізичну архітектуру системи, включаючи обчислювальні вузли, компоненти програмного забезпечення, що виконуються на цих вузлах, і зв'язки між ними. Вона показує, як програмні компоненти розгортаються на фізичних пристроях та серверних ресурсах. Основна мета діаграми розгортання – це візуалізувати, яким чином програмне забезпечення розгортається на інфраструктурі, щоб забезпечити правильну роботу системи. Це допомагає зрозуміти, як компоненти системи взаємодіють один з одним на фізичному рівні, які апаратні ресурси використовуються та які мережеві з'єднання необхідні.Діаграма розгортання також може відображати важливі аспекти конфігурації та налаштування програмного забезпечення, такі як налаштування серверів, баз даних, мережевих налаштувань тощо.

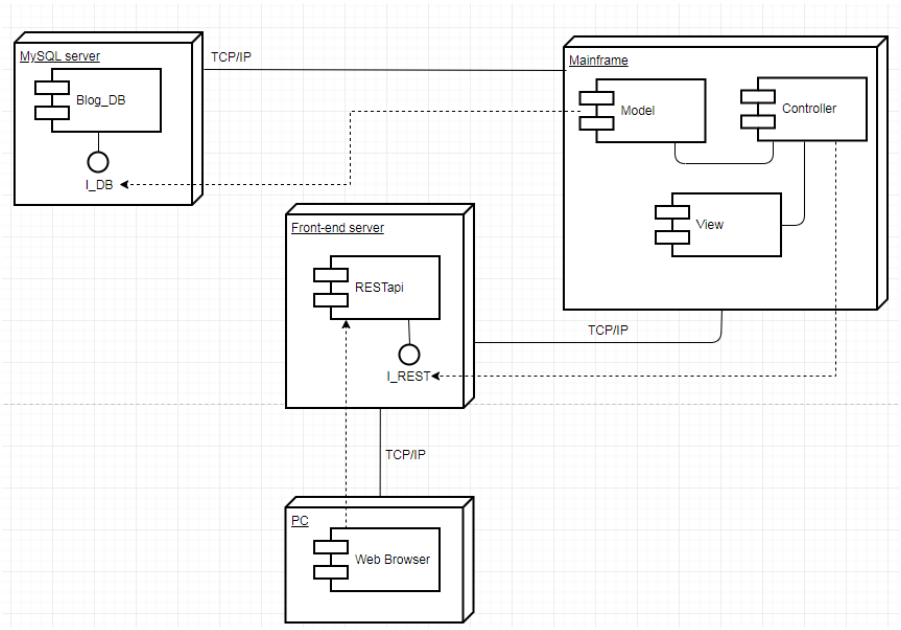

Рисунок 3.9 – Діаграма розгортання CMS-системи

Ця діаграма розгортання відображає архітектуру системи керування вмістом (CMS) для блогу, яка включає кілька ключових компонентів. На діаграмі показані взаємозв'язки між компонентами та фізичними пристроями, на яких вони розгорнуті. MySQL сервер з базою даних Blog_DB відповідає за зберігання всього контенту блогу, включаючи пости, коментарі, користувачів та інші дані. Front-end сервер містить RESTful API, який обробляє запити від користувачів та передає їх до відповідних модулів системи. Основний компонент системи (Mainframe) складається з моделі, контролера та виду (MVC архітектура), які відповідають за логіку застосунку, управління даними та відображення інтерфейсу користувача. Веб-браузер на ПК користувача взаємодіє з Front-end сервером через протокол TCP/IP, надсилаючи запити та отримуючи відповіді у вигляді веб-сторінок.

3.9 Проєктування інтерфейсу

Інтерфейс користувача (UI – User Interface) визначає, як виглядає та функціонує взаємодія між користувачем і програмним забезпеченням. Це включає в себе всі візуальні елементи та фізичні характеристики, які впливають на зручність використання системи. Інтерфейс користувача визначає кольорову схему, розташування кнопок, читабельність тексту та інші аспекти, які впливають на взаємодію з системою.

Інтерфейс користувача забезпечує передачу інформації між користувачем і програмно-апаратними компонентами системи. Він складається з різних засобів і методів, які забезпечують ефективну взаємодію.

Засоби інтерфейсу:

Виведення інформації. Інформація з пристрою передається користувачу за допомогою різноманітних засобів впливу на організм людини, таких як зорові (екрани, дисплеї, проєктори), слухові (динаміки, зумери, сирени) та тактильні (вібромотори) засоби.

Введення інформації. Користувач вводить інформацію або команди через різні пристрої, такі як кнопки, перемикачі, потенціометри, датчики положення та руху, сервоприводи, жести руками і обличчям, а також технології для знімання мозкової активності.

Методи інтерфейсу:

Логічний інтерфейс. Набір правил, встановлених розробником, визначає, як сукупність дій користувача призводить до необхідної реакції пристрою та виконання поставлених завдань. Це включає правила та алгоритми, які забезпечують логічну та передбачувану поведінку системи у відповідь на дії користувача.

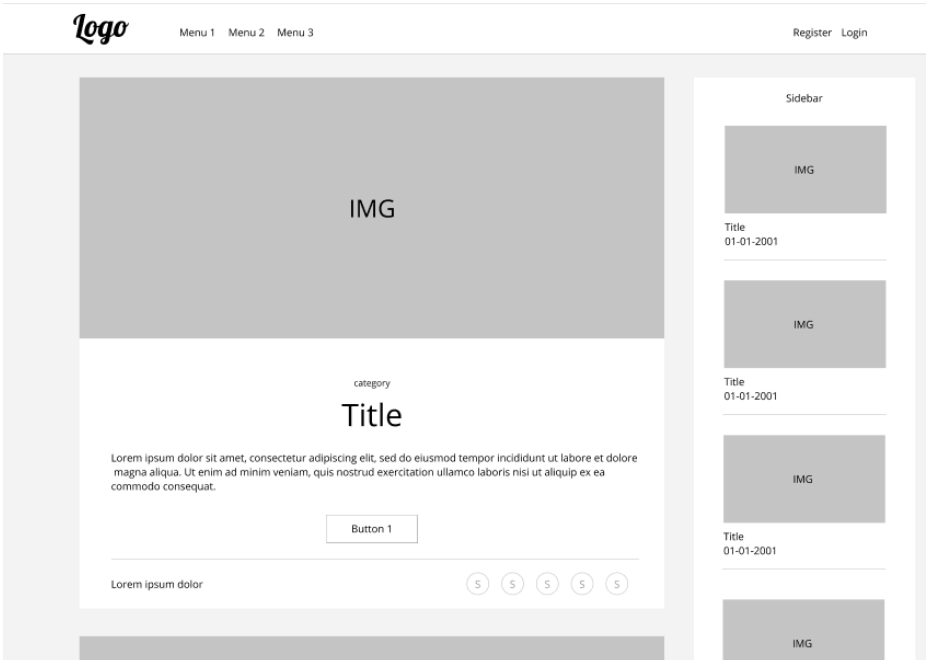

Рисунок 3.10 – Прототип головної сторінки

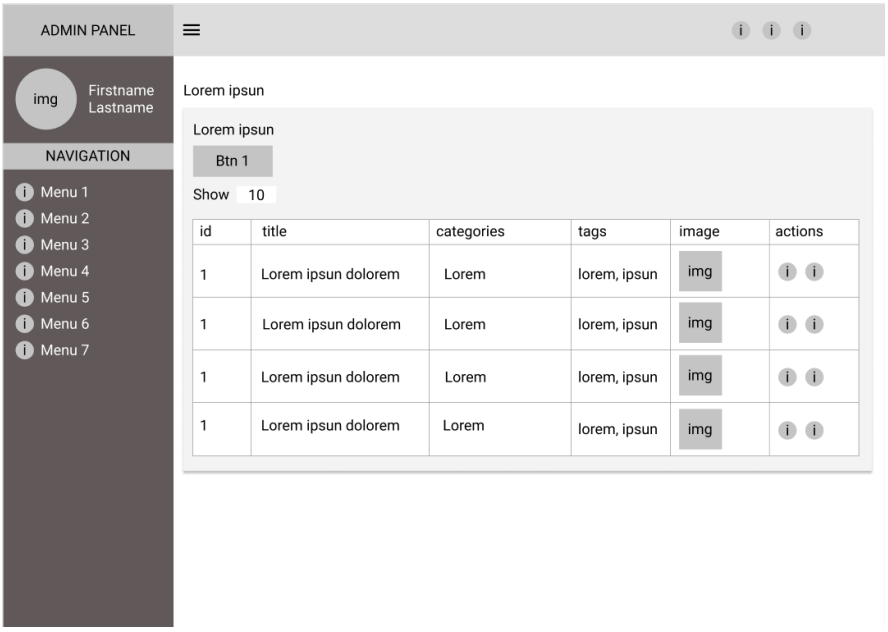

Рисунок 3.11 – Прототип ПА

Висновки до розділу 3

У третьому розділі було розглянуто архітектуру, моделювання та проєктування системи керування вмістом (CMS) для блогу. Були виділені основні компоненти архітектури, такі як контролери, middleware та сервіси, які відповідають за обробку запитів, перевірку автентифікації та авторизації, а також за реалізацію бізнес-логіки.

Діаграми станів допомогли візуалізувати динамічні аспекти системи, показуючи, як об'єкти переходять з одного стану в інший залежно від дій користувача або адміністратора.

Діаграми послідовностей і кооперації дали змогу детально описати порядок взаємодії між різними об'єктами системи під час виконання певних сценаріїв.

Діаграми діяльності показали алгоритмічну та логічну реалізацію операцій, деталізуючи послідовність виконання певних дій.

Діаграма класів дала змогу візуалізувати структуру системи, показавши класи, їхні атрибути, методи та зв’язки між ними. Це забезпечило розуміння того, як різні компоненти системи взаємодіють між собою на рівні коду.

Фізична модель даних була розроблена для визначення структури бази даних, включаючи таблиці, стовпці, індекси, обмеження та взаємозв'язки між таблицями.

Діаграма розгортання відобразила фізичну архітектуру системи, показавши, як програмні компоненти розгортаються на інфраструктурі.

4 РОЗРОБКА CMS, КОДУВАННЯ ТА АПРОБАЦІЯ ЗАСТОСУНКУ

4.1 Засоби розробки

Для розробки були використані наступні засоби:

– Мови програмування: PHP 7.1, JavaScript (ES6), HTML5, CSS3;

– Фреймворки: Laravel 9, jQuery 3.4.1, Bootstrap 5;

– IDE: PHPStorm 2024;

– Сервер: OpenServer (Apache);

– База даних: MySQL 8

– Засоби тестування : RESTapi кліент “Insomnia”;

PHP – це скриптова, серверна мова програмування для розробки вебзастосунків. За допомогою неї можна працювати із базою данних, обробляти дані на сервері. Мова інтепритується за допомогою Zend Engine.

HTML5 – гипертекстова мова розмітки вебсторінок. За допомогою неї розробляють каркас вебсторінки.

CSS3 – каскадна таблиця стилів, потрібна для стилізації та зміни зовнішнього вигляду вебсторінки.

JS(ES6) – скриптова мова, яка використовується, як інтегруюча мова длядоступу до об`єктів застосунків, надання їм інтекрактивності тощо.

Bootstrap 5 – це CSS фреймворк, який має величезний набір інструментів
для адаптації, верстки та інших Front-end можливостей для розробки.

jQuery 3.4.1 – бібліотека, яка фокусується на взаємодії HTML та JS, дозволяє легко працювати з DOM та маніпулювати ним.

Laravel 9 – це PHP фреймворк побудований на архітектурі MVC із використанням паттернів ServiceProvider, Singleton, Facades, а також стандартизований згідно стандартів PSR.

Сервер Apache – використовується для розробки на мові PHP. PHPStorm – IDE для веброзробки із можливістю підключення до БД, Github, удаленних серверів тощо.

MySQL 8 – реляційна БД для створення невеликих проєктів. Робота з базою проводилась через графічний клієнт PHPMyAdmin, а також через технологію міграцій до БД за допомогою Laravel.

Insomnia – графічний клієнт для тестування RESTapi. Імітує HTTP-клієнт, для роботи з RESTapi вебзастосунку.

PHP та Laravel залишаються одними з найкращих виборів для розробки вебзастосунків у 2024 році з кількох причин:

- Популярність та підтримка спільноти. PHP є однією з найпоширеніших мов програмування у світі. Величезна кількість вебсайтів, включаючи популярні платформи як WordPress, використовують PHP. Це забезпечує постійну підтримку та розширення можливостей мови.

- Ефективність та продуктивність. Laravel є одним із найпопулярніших PHP фреймворків, який дозволяє швидко та ефективно розробляти вебзастосунки. Завдяки його модульній структурі та вбудованим інструментам, розробка стає більш продуктивною.

- Вбудовані функції. Laravel пропонує безліч вбудованих функцій, таких як маршрутизація, автентифікація, робота з базами даних через ORM Eloquent, підтримка RESTful API, система міграцій баз даних та багато іншого. Це дозволяє зосередитися на логіці застосунку, а не на рутинних завданнях.

- Стандартизація та безпека. Laravel дотримується стандартів PSR, що забезпечує чистоту та підтримуваність коду. Крім того, фреймворк надає вбудовані засоби для забезпечення безпеки, такі як захист від SQL-ін'єкцій, XSS та CSRF.

- Швидкість розробки. Завдяки використанню Laravel, час на розробку скорочується завдяки використанню готових компонентів та шаблонів. Це дозволяє зосередитися на розробці унікальних функцій та логіки вашого застосунку.

- Інструменти для тестування. Laravel має вбудовані інструменти для тестування, що дозволяє забезпечити високу якість коду та стабільність роботи системи. Використання Insomnia для тестування REST API також допомагає в швидкій перевірці коректності роботи серверної частини.

4.2 Кодування

Кодування – це процес перетворення проєкту на реальність за допомогою написання коду. У цій системі керування вмістом (CMS) для блогу, основні частини коду відповідають за реалізацію CRUD-операцій (створення, читання, оновлення, видалення), аутентифікацію користувачів, управління контентом та інші функціональні можливості системи.

Основні компоненти коду: Контролери, Моделі, Роутери, Middleware, Сервіси. Далі буде детально роглянуто кожен з них:

1. Користувачі (Users). Модель User визначає зв'язки з іншими сутностями:

```
<?php

class User extends Authenticatable {
```

```
use Notifiable, SoftDeletes;

protected $fillable = ['name', 'email', 'password', 'avatar', 'is_admin'];
protected $hidden = ['password', 'remember_token'];
protected $casts = ['email_verified_at' => 'datetime'];

public function posts(){
return $this->hasMany(Post::class);
}

public function comments(){
return $this->hasMany(Comment::class);
}

public function isAdmin(){
return $this->is_admin;
}
}
```

Користувачі є основними учасниками системи, які можуть створювати, редагувати та видаляти контент. Кожен користувач має атрибути, такі як id, name, email, password, avatar, status, та is_admin.

2. Пости (Posts). Модель Post визначає зв'язки з категоріями, коментарями та тегами:

```
<?php
class Post extends Model {
use SoftDeletes;

protected $fillable = ['title', 'slug', 'content', 'category_id', 'user_id', 'status', 'views',
'is_featured'];

public function category() {
return $this->belongsTo(Category::class);
```

```
}

public function comments() {
return $this->hasMany(Comment::class);
}

public function tags() {
return $this->belongsToMany(Tag::class, 'post_tags');
}
}
```

Пости є центральною сутністю системи. Вони містять основний контент блогу. Кожен пост має атрибути, такі як id, title, slug, content, category_id, user_id, status, views, is_featured.

3. Категорії (Categories). Модель Category визначає зв'язок з постами:

```
<?php
class Category extends Model {
   use SoftDeletes;

   protected $fillable = ['title', 'slug'];

   public function posts() {
      return $this->hasMany(Post::class);
   }
}
```

Категорії допомагають організувати пости у логічні групи. Кожна категорія має атрибути, такі як id, title, slug.

4. Теги (Tags). Модель Tag визначає зв'язок з постами через проміжну таблицю post_tags:

```
<?php
class Tag extends Model {
   use SoftDeletes;
```

```
    protected $fillable = ['title', 'slug'];

    public function posts() {
        return $this->belongsToMany(Post::class, 'post_tags');
    }
}
```

Теги дозволяють додатково категоризувати пости, додаючи їм ключові слова для кращої організації та пошуку.

5. Коментарі (Comments). Модель Comment визначає зв'язок з постами та користувачами:

```
<?php
class Comment extends Model {
    use SoftDeletes;

    protected $fillable = ['text', 'user_id', 'post_id', 'status'];

    public function post() {
        return $this->belongsTo(Post::class);
    }

    public function user() {
        return $this->belongsTo(User::class);
    }
}
```

Коментарі дозволяють користувачам взаємодіяти з постами, залишаючи свої відгуки або питання.

6. Зв'язки між сутностями. Міграція створення таблиці post_tags:

```
<?php
Schema::create('post_tags', function (Blueprint $table) {
$table->increments('id');
$table->integer('post_id');
```

```
$table->integer('tag_id');
$table->timestamps();
});
```

Для організації зв'язків між постами та тегами використовується проміжна таблиця post_tags.

Контролери

Контролери відповідають за обробку HTTP-запитів, взаємодію з моделями та видами, а також виконання основних бізнес-логік.

```php
<?php
class PostsController extends Controller
{
    public function index()
    {
        $posts = Post::all();
        return view('posts.index', compact('posts'));
    }

    public function create()
    {
        return view('posts.create');
    }

    public function store(Request $request)
    {
        $validatedData = $request->validate([
            'title' => 'required|max:255',
            'content' => 'required',
        ]);

        $post = new Post();
        $post->title = $validatedData['title'];
        $post->content = $validatedData['content'];
        $post->save();

        return redirect()->route('posts.index');
    }

    public function show($id)
    {
        $post = Post::findOrFail($id);
        return view('posts.show', compact('post'));
    }
```

Рисунок 4.1 — Приклад коду з контролера PostsController.php

Сервіси

Сервіси містять бізнес-логіку, відокремлюючи її від контролерів для покращення читабельності коду.

```php
<?php
class PostService
{
    public function create(array $data)
    {
        $post = new Post();
        $post->title = $data['title'];
        $post->content = $data['content'];
        $post->save();

        return $post;
    }

    public function update(Post $post, array $data)
    {
        $post->title = $data['title'];
        $post->content = $data['content'];
        $post->save();

        return $post;
    }
}
```

Рисунок 4.2 — Приклад коду з PostService.php

Використання контролерів, моделей, роутерів, middleware та сервісів дозволяє створити гнучку та масштабовану систему, яка легко підтримується та розширюється. Кодування є важливим етапом розробки програмного забезпечення, що забезпечує реалізацію проекту відповідно до вимог та специфікацій.

4.3 Результат розробки

Результатом розробки вебзастосунку є розроблена система керування вмістом для організації блогу блогом. Далі буде продемонстровано запуск cms та перевірка ключових функцій задля остаточної перевірки коректної роботи та переконатися, що поставлені цілі було успішно реалізовано під час виконання цього проєкту.

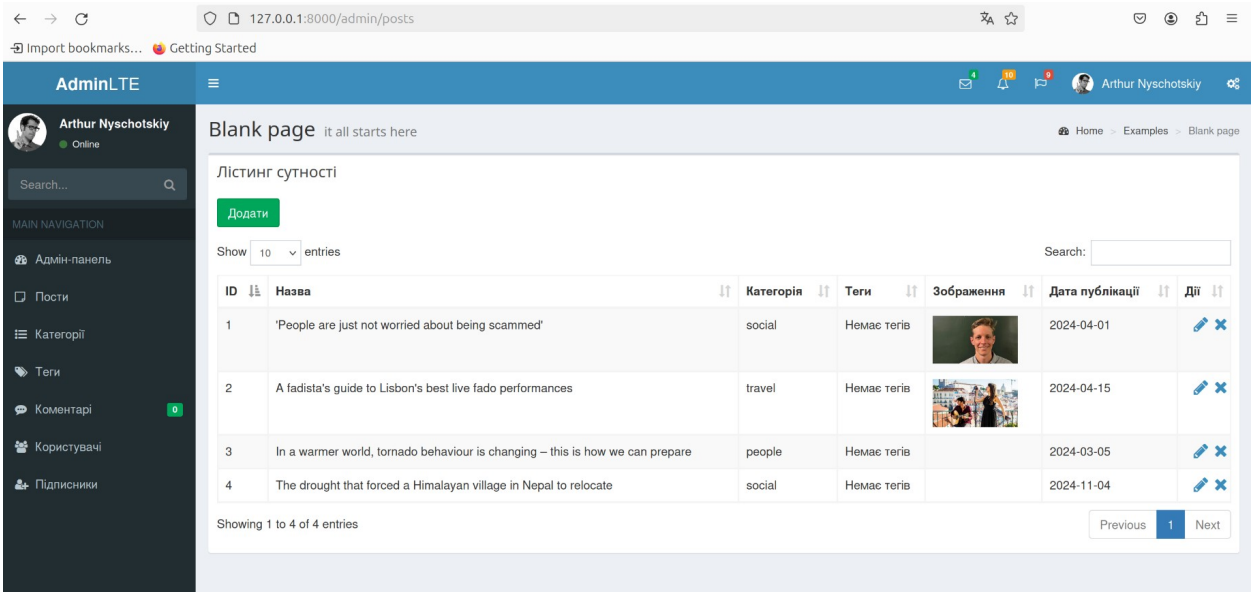

Рисунок 4.3 – Лістинг постів у панелі адміністратора

На цьому рисунку зображено інтерфейс адміністративної панелі, де відображені записи постів у табличному форматі. Кожен запис містить такі поля: ID, Назва, Категорія, Теги, Зображення, Дата публікації та Дії.

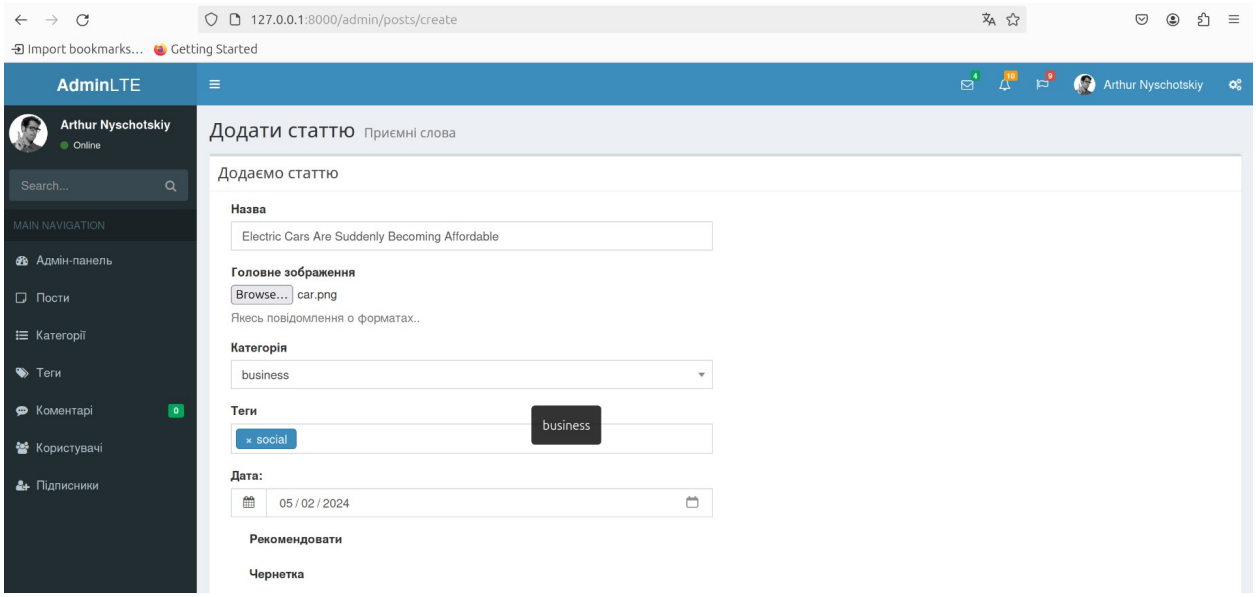

Рисунок 4.4 – Створення поста у панелі адміністратора

На зображенні представлена форма для створення нового поста в адміністративній панелі. Форма містить поля для введення назви поста, завантаження головного зображення, вибору категорії, додавання тегів, встановлення дати публікації, а також опції для рекомендації та збереження поста як чернетки.

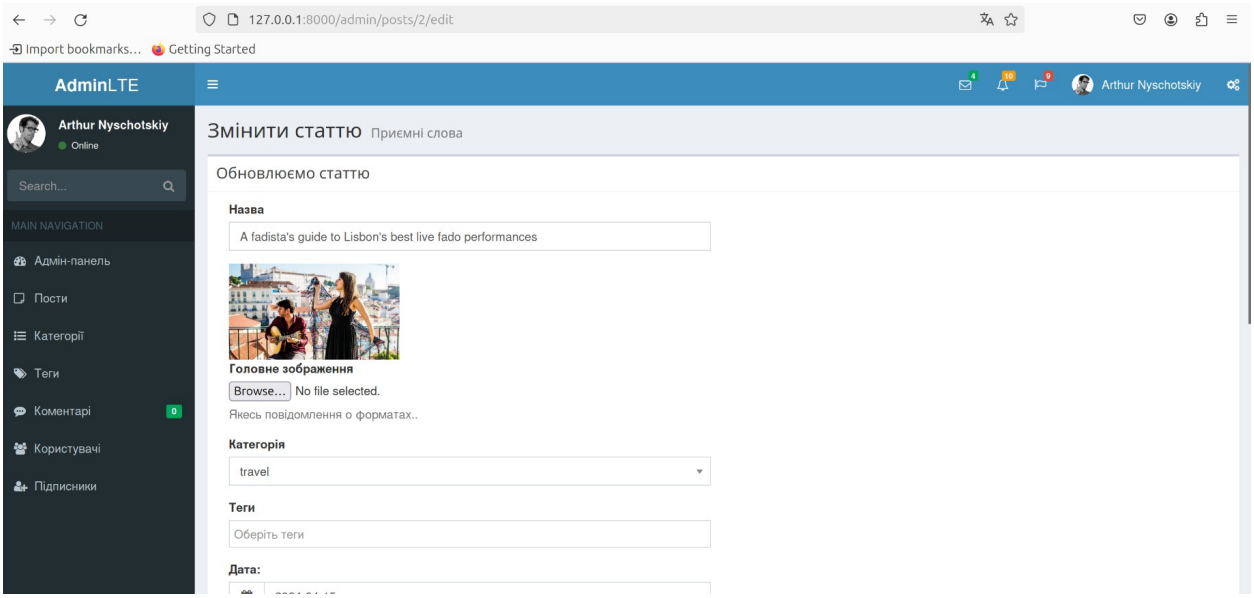

Рисунок 4.5 – Редагування поста у панелі адміністратора

На рисунку представлена форма для редагування існуючого поста в адміністративній панелі. Форма містить поля для редагування назви поста, завантаження або зміни головного зображення, вибору категорії, додавання тегів, встановлення дати публікації та інших параметрів поста.

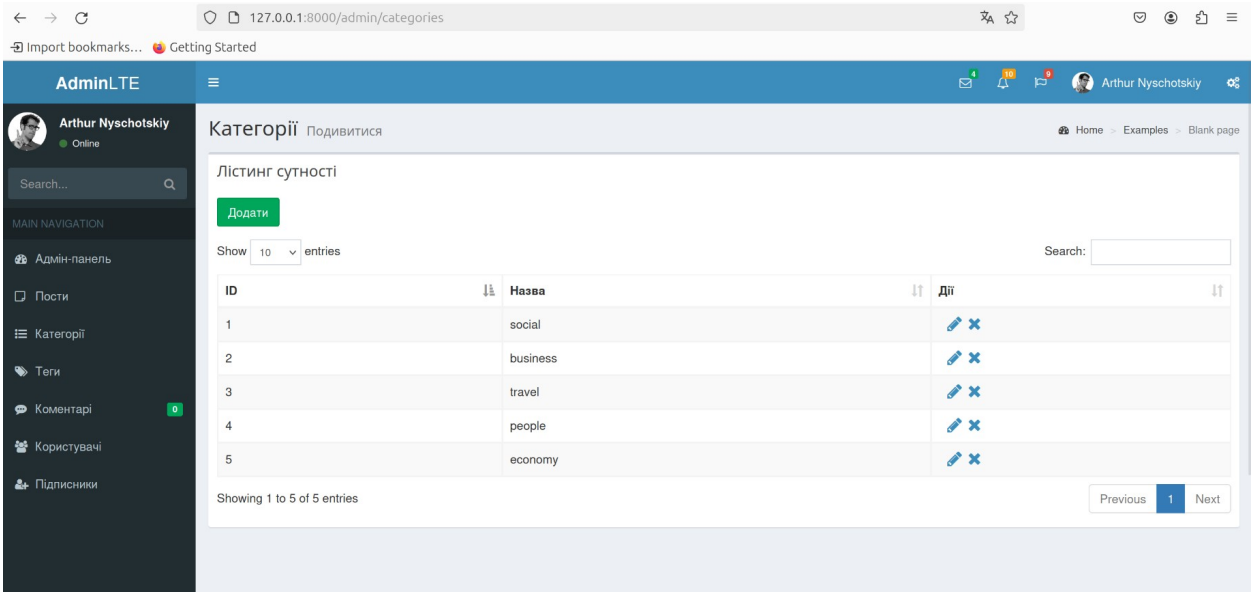

Рисунок 4.6 – Лістинг категорій у панелі адміністратора

На рисунку показаний список категорій у адміністративній панелі, який включає ідентифікатор (ID), назву категорії та доступні дії (редагування, видалення). Ця форма дозволяє адміністратору переглядати всі наявні категорії, а також виконувати операції з їх редагування та видалення.

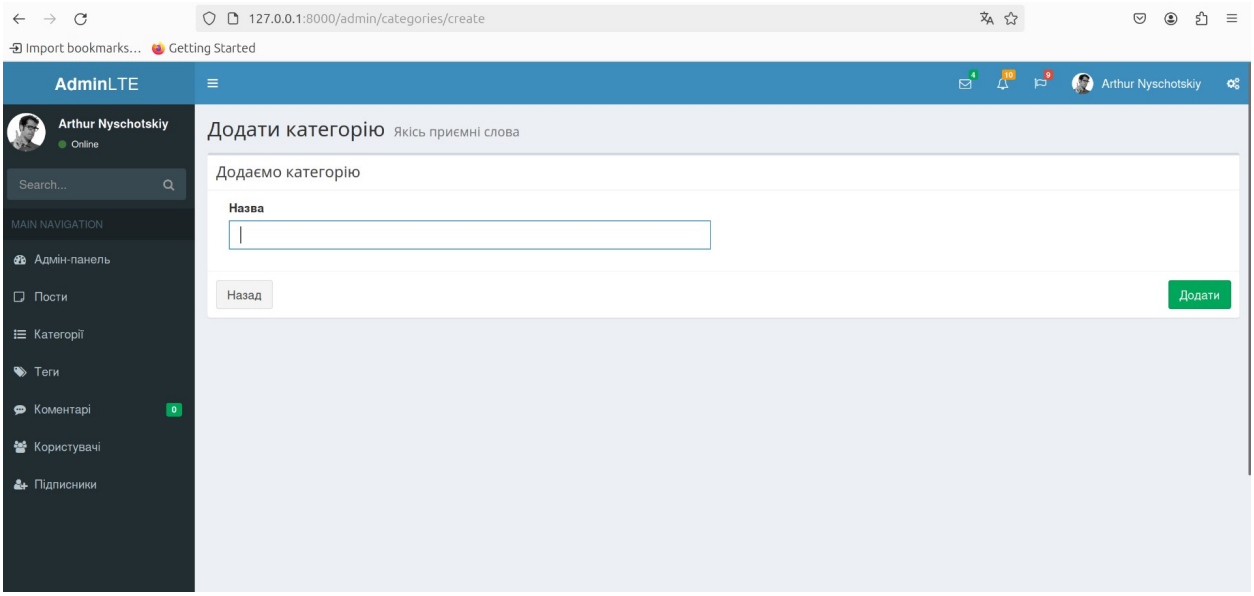

Рисунок 4.7 – Додавання категорії у ПА

На рисунку показана форма для додавання нової категорії в адміністративній панелі. Форма містить поле для введення назви нової категорії та кнопку "Додати", яка зберігає введені дані в базі даних.

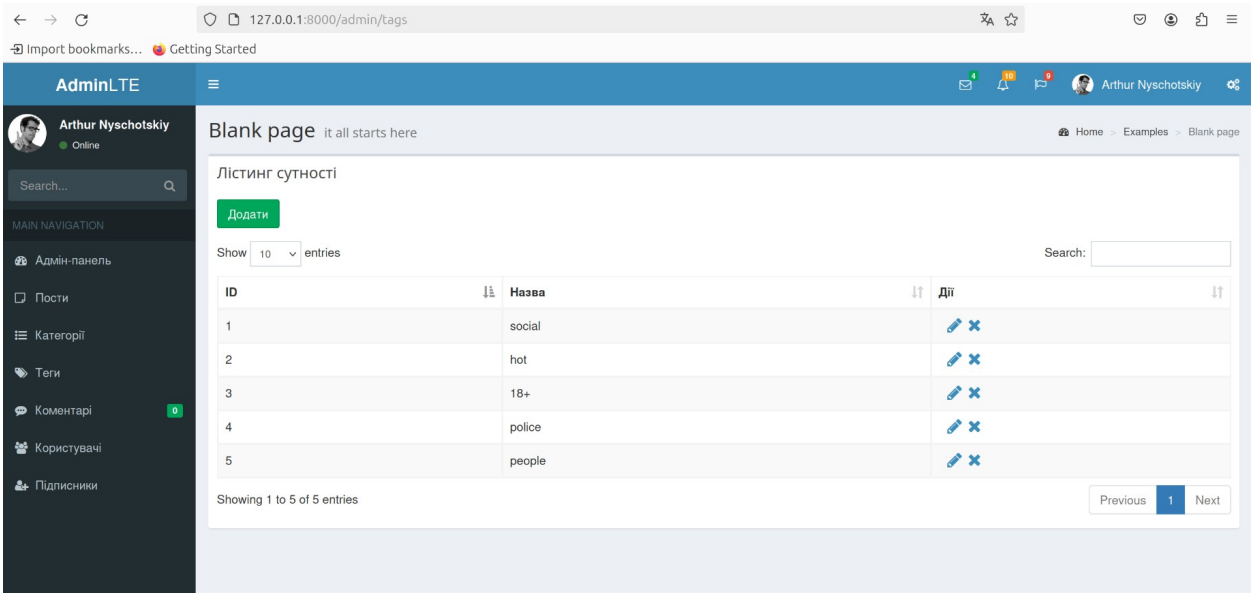

Рисунок 4.8 – Лістинг тегів у панелі адміністратора

На зображенні показаний список тегів у адміністративній панелі, який включає ідентифікатор (ID), назву тегу та доступні дії (редагування, видалення). Ця форма дозволяє адміністратору переглядати всі наявні теги, а також виконувати операції з їх редагування та видалення.

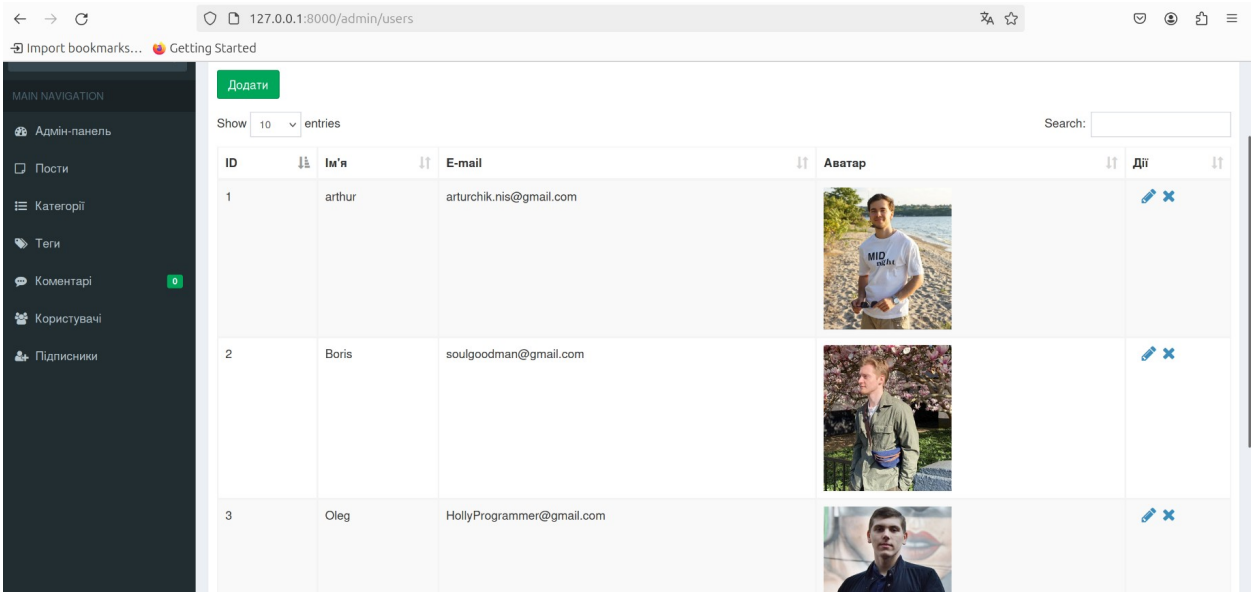

Рисунок 4.9 – Лістинг користувачів у панелі адміністратора

На рисунку представлений список користувачів у адміністративній панелі, що включає ідентифікатор (ID), ім'я, email, аватар користувача та доступні дії (редагування, видалення). Ця форма дозволяє адміністратору переглядати всі наявні акаунти користувачів, а також виконувати операції з їх редагування та видалення.

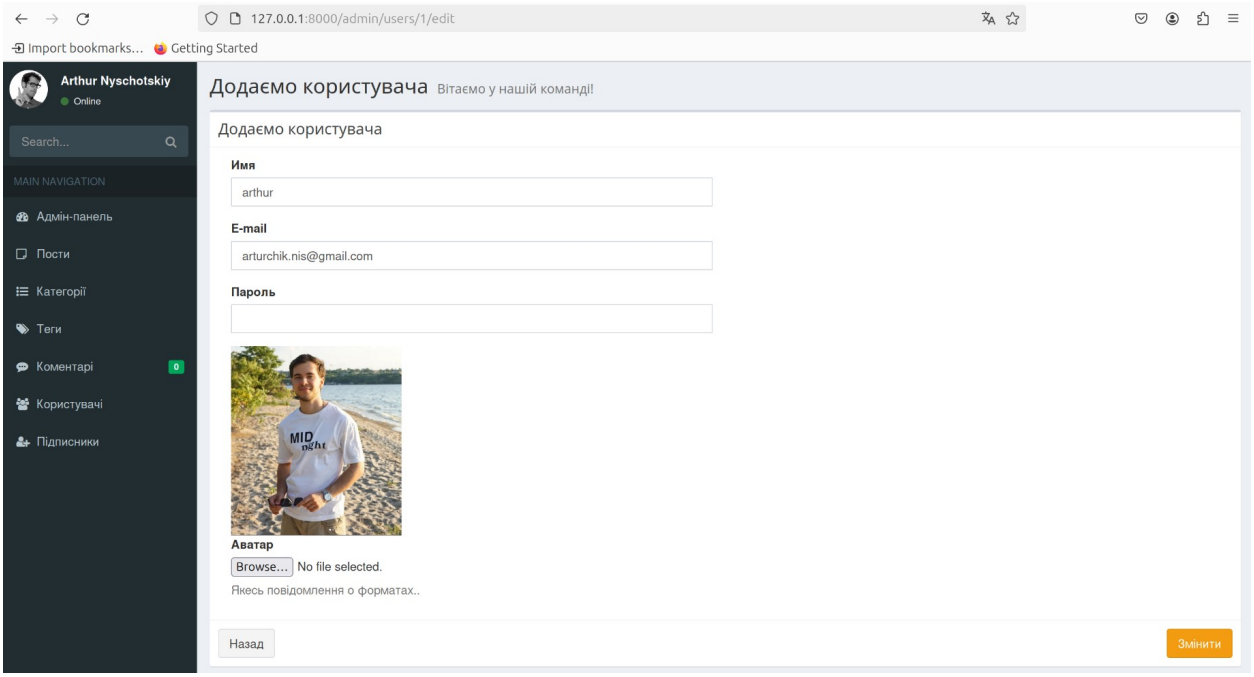

Рисунок 4.10 – Редагування користувачів у панелі адміністратора

На зображенні рисунку зображена форма редагування ПА. Форма дозволяє адміністратору змінювати ім'я, email, пароль та аватар користувача. Після внесення змін адміністратор може зберегти їх, натиснувши кнопку "Змінити".

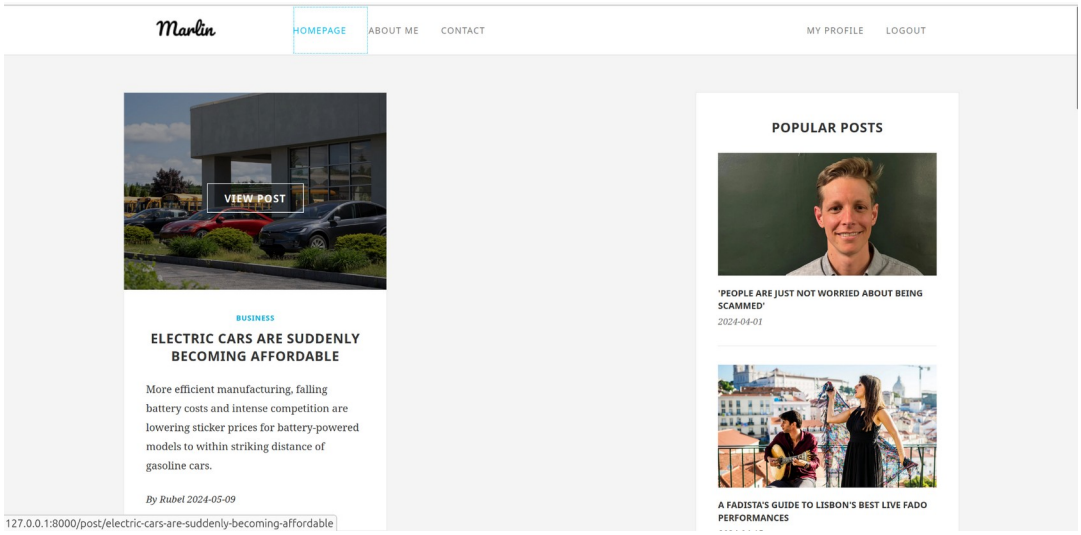

Рисунок 4.11 – Головна сторінка блогу

На рисунку показана головна сторінка блогу, яка містить останні опубліковані статті та популярні пости. Користувач може переглянути деталі статті, натиснувши на заголовок або зображення поста. Верхня навігаційна панель дозволяє користувачам переміщатися між різними розділами сайту, такими як головна сторінка, сторінка "Про мене", контактна інформація, профіль користувача та вихід з облікового запису.

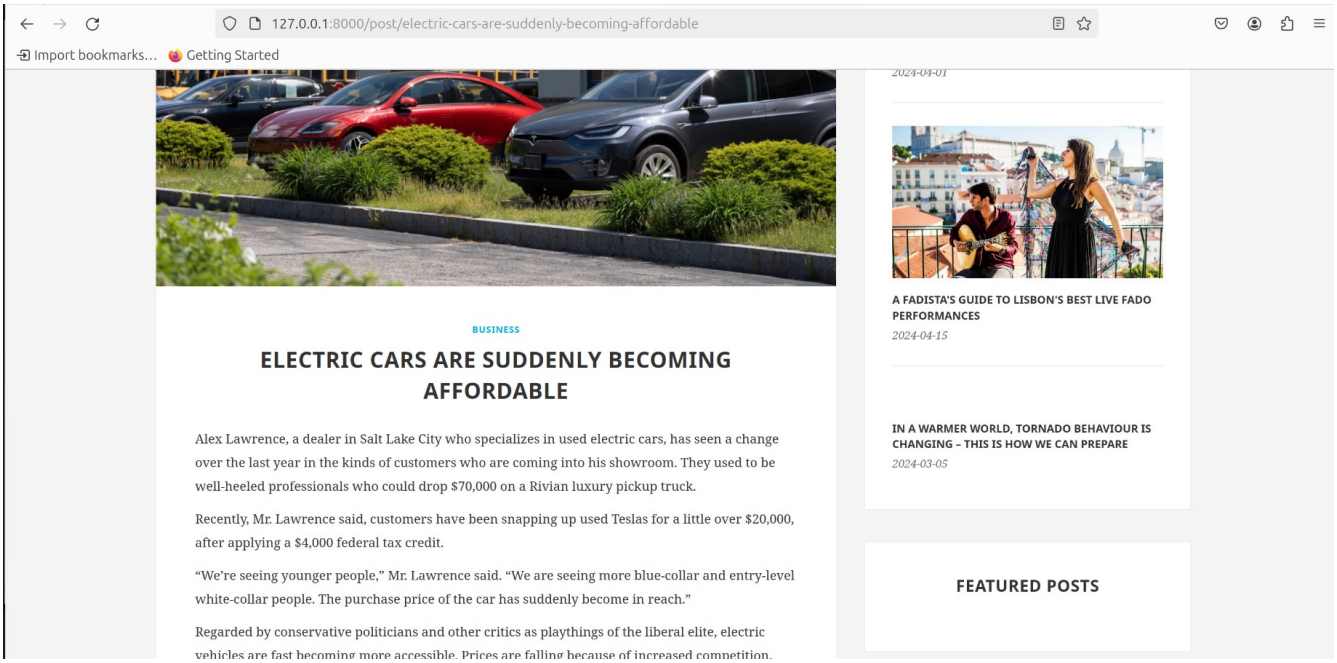

Рисунок 4.12 – Сторінка одиночного поста

На рисунку показана сторінка з детальною інформацією про окремий пост блогу. Включає заголовок, дату публікації, основний текст статті та зображення. Праворуч розміщені популярні пости, які можна переглянути, натиснувши на заголовок або зображення.

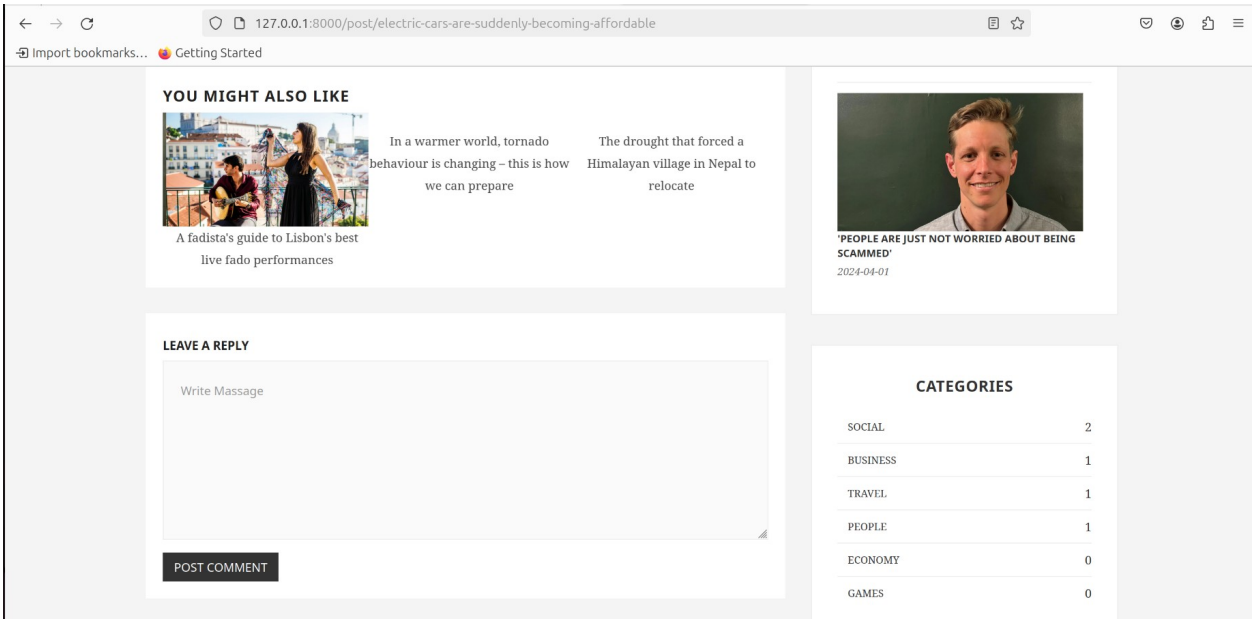

Рисунок 4.12 – Розділ пошуку за категорією

На рисунку показано розділ, який дозволяє здійснювати пошук постів за категорією, які знаходяться у правому нижньому куті головної сторінки.

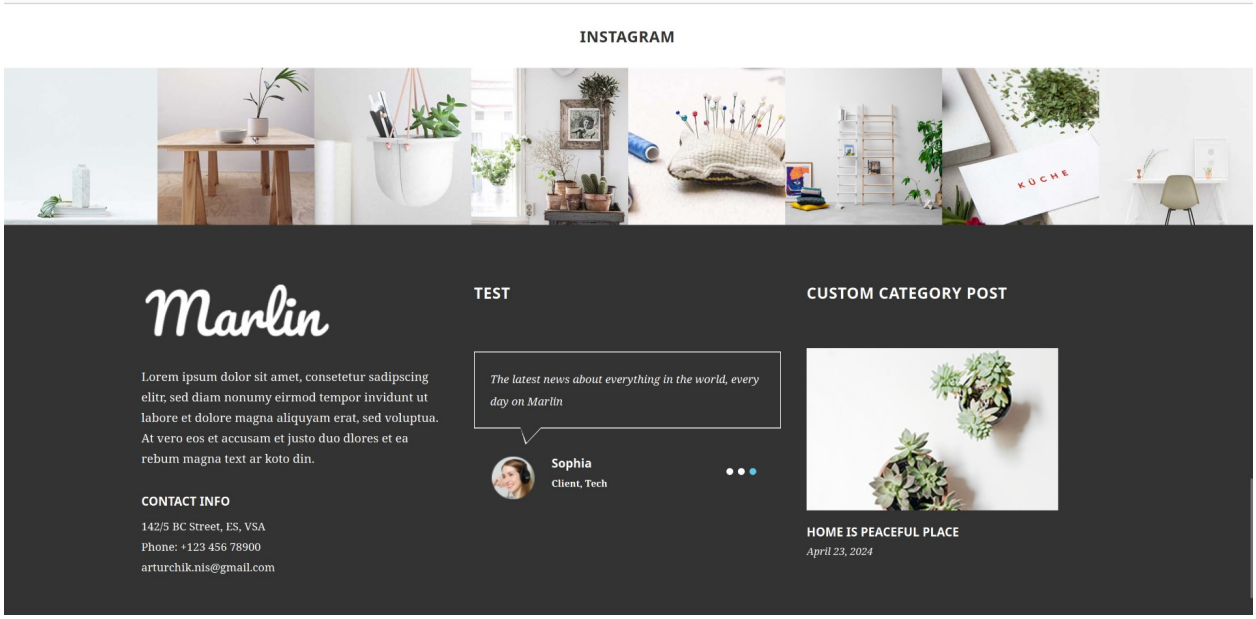

Рисунок 4.13 – Розділ контакти

На рисунку показаний розділ сайту з контактною інформацією.

Висновки до розділу 4

У даному розділі було проведено ряд важливих етапів, необхідних для успішного створення та впровадження системи керування вмістом (CMS) для блогу. Основними завданнями цього розділу були: визначення засобів розробки, написання коду, тестування та апробація системи.

Спочатку, було обрано та налаштовано необхідні засоби розробки.

Далі, було реалізовано кодування основних функцій системи. Написано контролери для роботи з постами, категоріями, тегами, коментарями та користувачами, які забезпечують основні CRUD-операції. Наприклад, контролер PostsController реалізує методи для створення, редагування, перегляду та видалення постів. Крім того, було реалізовано middleware для автентифікації та авторизації користувачів, а також сервіси для відокремлення бізнес-логіки від контролерів.

Також, було проведено тестування та апробацію системи: контрольний запуск та перевірку коректності роботи застосунку, яка відповідає очікуванням.

Результатом виконання розділу 4 стало створення функціональної та надійної CMS-системи для блогу, яка відповідає вимогам користувачів та забезпечує зручний інтерфейс для управління вмістом. Завдяки використанню сучасних технологій та інструментів, система є масштабованою, легко розширюваною та підтримуваною. Успішна апробація системи підтвердила її готовність до впровадження та використання у реальних умовах.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було розроблено систему керування вмістом (CMS) для блогу, яка відповідає сучасним вимогам веб-розробки та забезпечує високу функціональність і зручність використання для адміністратора та кінцевих користувачів.

Розробка включала в себе кілька ключових етапів:

1. Проєктування архітектури системи - визначення основних компонентів, їх взаємодії та структури, що забезпечило чіткий розподіл функцій і полегшило подальшу розробку.

2. Моделювання та створення діаграм - були розроблені діаграми станів, послідовностей, кооперації та діяльності, що дозволило візуалізувати динамічні аспекти роботи системи та покращити розуміння процесів серед учасників проекту.

3. Кодування - розробка основного функціоналу системи, включаючи контролери для управління постами, категоріями, тегами, коментарями та користувачами.

4. Розробка інтерфейсу користувача - створення зручного та інтуїтивно зрозумілого інтерфейсу для адміністратора та кінцевих користувачів.

5. Перевірка коректності роботи застосунку.

Отриманий результат – це функціональний вебзастосунок, що дозволяє легко управляти контентом блогу, додавати нові пости, категорії, теги, коментарі та користувачів. Система забезпечує надійну роботу, безпеку даних та зручність використання як для адміністратора, так і для кінцевих користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Laravel Documentation. URL: https://laravel.com/docs/9.x (дата звернення: 24.02.2024).

2. Bootstrap Documentation. URL: https://getbootstrap.com/docs/5.0/getting-started/introduction/ (дата звернення: 24.02.2024).

3. jQuery Documentation. URL: https://api.jquery.com/ (дата звернення: 24.02.2024).

4. PHP: Hypertext Preprocessor. URL: https://www.php.net/ (дата звернення: 28.02.2024).

5. MySQL Documentation. URL: https://dev.mysql.com/doc/ (дата звернення: 28.02.2024).

6. W3Schools - HTML5 Tutorial. URL: https://www.w3schools.com/html/html5_intro.asp (дата звернення: 04.03.2024).

7. W3Schools - CSS3 Tutorial. URL: https://www.w3schools.com/css/css3_intro.asp (дата звернення: 04.03.2024).

8. ES6 Features. URL: https://www.ecma-international.org/ecma-262/6.0/ (дата звернення: 04.03.2024).

9. REST API Tutorial. URL: https://restfulapi.net/ (дата звернення: 15.04.2024).

10. UML Diagrams. URL: https://www.uml-diagrams.org/ (дата звернення: 25.04.2024).

11. Офіційна документація Laravel. URL: https://laravel.com/docs/9.x (дата звернення: 25.04.2024).

12. Документація Bootstrap. URL: https://getbootstrap.com/docs/5.0/getting-started/introduction/ (дата звернення: 03.05.2024).

13. Visual Paradigm - UML Deployment Diagram Tutorial. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/ (дата звернення: 09.05.2024).

14. PHP: Hypertext Preprocessor. URL: https://www.php.net/ (дата звернення: 03.05.2024).

15. Документація MySQL. URL: https://dev.mysql.com/doc/ (дата звернення: 07.05.2024).

16. ES6 Features. URL: https://www.ecma-international.org/ecma-262/6.0/ (дата звернення: 08.05.2024).

17. REST API Підручник. URL: https://restfulapi.net/ (дата звернення: 10.05.2024).