МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ФОТОБЛОГУ З ВИКОРИСТАННЯМ LARAVEL»**

Виконав: студент 4 курсу, групи 6.1211-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
О.Д. Деменський
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н. Кривохата А.Г.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Решевська К.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Деменському Олексію Дмитровичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка фотоблогу з використанням Laravel

керівник роботи Кривохата Анастасія Григорівна, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 02.06.2025 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі, аналіз предметної області.
2. Проєктування.
3. Реалізація та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 26.12.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	13.01.2025	
2.	Збір вихідних даних.	27.01.2025	
3.	Обробка методичних та теоретичних	17.02.2025	
	джерел.		
4.	Розробка першого та другого розділу.	31.03.2025	
5.	Розробка третього розділу.	19.05.2025	
6.	Оформлення та нормоконтроль	26.05.2025	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	16.06.2025	

Студент _____ О.Д. Деменський
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.Г. Кривохата
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка фотоблогу з використанням Laravel»: 37 с., 23 рис., 7 джерел.

БАЗА ДАНИХ, ВЕБЗАСТОСУНОК, ФОТОБЛОГ, BACKEND, FRONTEND, LARAVEL, MYSQL, PHP.

Об’єкт дослідження – процес розробки фотоблогу.

Предмет дослідження – інструменти фреймворку Laravel, засоби взаємодії з користувачем.

Мета роботи: розробити фотоблог з використанням Laravel.

Метод дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання, проєктування, конструювання та тестування програмного забезпечення.

При розробці вебдодатку був проведений аналіз предметної області, обрано та спроєктовано архітектуру системи за допомогою UML у вигляді наборів діаграм варіантів використання, послідовностей, класів, компонентів та розгортання. Реалізовано з використанням фреймворку Laravel.

В результаті роботи створено вебзастосунок фотоблог.

SUMMARY

Bachelor’s qualifying paper “Photoblog Development Using Laravel”: 37 pages, 23 figures, 7 references.

DATABASE, WEB APPLICATION, PHOTOBLOG, BACKEND, FRONTEND, LARAVEL, MYSQL, PHP.

The object of the study is the process of developing a photo blog.

The subject of the research is the tools provided by the Laravel framework and user interaction mechanisms.

The aim of the study is to develop a photo blog using the Laravel framework.

The research methods include methods for collecting and analyzing software requirements, as well as modeling, designing, implementing, and testing software.

During the development of the web application, an analysis of the subject area was carried out, and the system architecture was selected and designed using UML, presented through sets of use case, sequence, class, component, and deployment diagrams. The system was implemented using the Laravel framework.

As a result of the work, a functional web application in the form of a photo blog was developed.

ЗМІСТ

Завдання на кваліфікаційну роботу ……2
Реферат 4
Summary 5
Вступ 7
1 Технічне завдання 8
1.1 Терміни та визначення 8
1.2 Опис предметної області 8
1.3 Функціональні вимоги 10
1.4 Нефункціональні вимоги 10
1.5 Вибір технологій 11
2 Проєктування 13
2.1 Діаграма прецедентів 13
2.2 Діаграма послідовності 14
2.3 Діаграма класів 16
2.4 Структура бази даних 17
2.5 Діаграма компонентів 22
2.6 Діаграма розгортання 24
3 Реалізація та тестування 26
3.1 Підготовка середовища розробки 26
3.2 Створення та підключення бази даних 26
3.3 Керівництво користувача 31
Висновки 36
Перелік посилань 37

ВСТУП

У сучасну цифрову епоху фотоблоги стали популярним засобом для візуального самовираження, презентації творчості та ведення візуальних щоденників. З розвитком вебтехнологій створення власного фотоблогу стало доступним навіть для невеликих команд або окремих розробників. Особливої актуальності набуває використання сучасних фреймворків, таких як Laravel, який забезпечує високу продуктивність, безпеку та зручність розробки.

Актуальність теми зумовлена стрімким розвитком вебтехнологій, зростаючим попитом на візуальний контент в інтернеті, а також потребою у зручних та функціональних платформах для ведення фотоблогів.

Метою даної роботи є розробка функціонального вебдодатку фотоблогу з використанням фреймворку Laravel, який дозволяє користувачам публікувати, переглядати та коментувати фотографії.

Для досягнення поставленої мети було визначено такі завдання:

– сформувати вимоги до системи;
– спроєктувати та побудувати архітектуру системи;
– реалізувати вебдодаток фотоблог;
– протестувати роботу системи.

Об’єктом дослідження є процес розробки вебдодатків з використанням PHP-фреймворків.

Предмет дослідження: технології та інструменти Laravel, які забезпечують створення фотоблогу.

Методи дослідження: аналіз, моделювання, проєктування, конструювання та тестування програмного забезпечення.

Перший розділ присвячено збору та аналізу вимог до системи, огляду інструментів розробки та опису системи. У другому розділі розглянуто етапи проєктування системи, наведено структуру бази даних. Третій розділ присвячено реалізації та тестуванню роботи системи вебзастосунку.

1 ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Терміни та визначення

Laravel – це безкоштовний PHP-фреймворк, який надає безліч корисних інструментів для розробки вебдодатків. З моменту свого запуску він став найпопулярнішим на ринку завдяки своїй простоті, безпеці, великому набору інструментів, підтримці архітектури MVC та обширній документації.

MySQL – це система управління базами даних, яка використовується для зберігання, організації та управління даними.

БД – база даних, один або кілька файлів даних, призначених для зберігання, зміни й обробки інформації.

MVC (Model-View-Controller) – архітектурний шаблон, який використовується під час проєктування та розробки програмного забезпечення.

1.2 Опис предметної області

Предметною областю є фотоблог. У сучасну цифрову епоху фотоблоги стали популярним засобом для візуального самовираження, презентації творчості та ведення фото щоденників. З розвитком вебтехнологій створення власного вебдодатку із медіа-контентом стало доступним навіть для невеликих команд або окремих розробників. Значно вплинули на це фреймворки. Laravel, як один із найпотужніших PHP-фреймворків, надає розробникам інструменти для швидкого створення безпечних, гнучких та масштабованих вебзастосунків.

Розроблена система – це вебзастосунок «Фотоблог», який дозволяє користувачам створювати особисті фотоблоги, завантажувати фотографії,

додавати до них описи, переглядати публікації інших користувачів і залишати коментарі. Система забезпечує базову взаємодію між користувачами та контентом, а також має функціонал для аутентифікації та керування профілем.

Користувачі поділяються на неавторизованих, авторизованих та з правами адміністратора.

Гість, або неавторизований користувач, бачить головну сторінку, фотоблоги, коментарі, але не може додавати коментарі чи створювати блоги, поки не авторизується. Коли користувач увійде до свого акаунту, він матиме доступ до коментування фотоблогів інших користувачів, а також до створення або видалення своїх публікацій. Адміністратор додатково має змогу переглянути список зареєстрованих користувачів та видаляти коментарі або публікації, що порушують правила.

Застосунок повинен надавати користувачам можливість ведення та перегляду фотоблогів. Процес взаємодії з системою проходить наступним чином. Користувач, повинен ввести адресу сайту в браузері, система відобразить головну сторінку. Після переходу на сайт користувач має можливість взаємодіяти з такими розділами як: створення фотоблогу, коментування.

Процес створення свого фотоблогу проходить таким чином. Користувач натискає кнопку «Створити фотоблог», після чого система перенаправляє його на сторінку створення нового допису. На цій сторінці користувач заповнює форму, що містить такі поля, як назва фотоблогу, опис і завантаження зображення. Після заповнення форми та натискання кнопки «Опублікувати», інформація зберігається у базі даних, а фотоблог буде відображатися на головній сторінці або в окремому профілі користувача.

Основні можливості системи:

– система реєстрації та авторизації користувачів;
– створення/видалення власного фотоблогу;
– завантаження своїх фотографій;
– перегляд та коментування публікацій користувачів.

Система орієнтована на зручність користувачів, тому має простий, інтуїтивно зрозумілий інтерфейс з адаптивною верстkою. Використання Laravel дозволяє реалізувати надійну серверну логіку, захистити дані користувачів та забезпечити ефективну роботу з базою даних.

1.3 Функціональні вимоги

Функціональне призначення системи – реалізувати функціональні можливості фотоблогу, що забезпечують взаємодію користувача з інтерфейсом вебзастосунку через браузер.

Експлуатаційне призначення системи: система може експлуатуватися користувачем системи.

Мета створення системи – розробка фотоблогу на сайті.

Система має надавати всім користувачам такі можливості:

– реєстрація та авторизація користувачів;
– створення власного фотоблогу;
– перегляд усіх фотографій;
– коментування фотоблогів користувачів;
– завантаження своїх фотографій;
– видалення фотоблогу.

Система має надавати адміністраторам такі можливості:

– перегляд зареєстрованих користувачів;
– видалення публікацій або коментарів, що порушують правила.

1.4 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи, що впливають на зручність використання, продуктивність, безпеку, масштабованість та сумісність з різними платформами. Вони не описують

безпосередньо функціональність, а встановлюють обмеження і критерії, які система повинна задовольняти під час її експлуатації.

Розглянемо **нефункціональні вимоги** до розроблюваної системи.

Адаптивність інтерфейсу: система повинна відображати коректно інтерфейс користувача на будь-якому пристрої.

Кросбраузерність: система повинна працювати та правильно відображатися для наступних браузерів останніх версій:

– Mozilla Firefox;
– Google Chrome;
– Safari;
– Opera.

Швидкість завантаження сторінки: будь-яка сторінка повинна завантажуватися не довше, ніж за 2 секунди.

Швидкість обробки дій користувача: система повинна обробляти дії користувача (наприклад, надсилання коментаря або створення блогу) і відправляти відповідні повідомлення не довше, ніж за 2 секунди.

Безпека доступу до адміністративної панелі: інтерфейс адміністратора має бути доступним виключно для авторизованих користувачів з адміністративними правами. Інші користувачі не повинні мати до нього фізичний чи програмний доступ.

Контроль доступу до даних: система повинна забезпечувати захист даних – неавторизовані користувачі не повинні мати доступу до особистої інформації, паролів або будь-яких внутрішніх даних системи.

1.5 Вибір технологій

Під час розробки вебзастосунку фотоблогу було використано сучасні, зручні та ефективні технології, що забезпечують швидку, масштабовану й зручну розробку як з точки зору інтерфейсу, так і з боку серверної частини.

Обраний інструментарій забезпечує ефективну реалізацію функціональних вимог, високий рівень продуктивності та зрозумілу підтримку в майбутньому.

Для написання коду було обрано середовище Visual Studio Code. Цей редактор надає широкі можливості завдяки підтримці розширень, автодоповнення коду та інтеграції з терміналом. У межах даної роботи VS Code використовувався для розробки Blade-шаблонів, стилів Tailwind CSS, скриптів мовою JavaScript, TypeScript, PHP-коду, а також роботи з консоллю Artisan та керування міграціями Laravel.

Фреймворк Laravel було обрано як основу для створення серверної частини застосунку. Laravel забезпечує простоту, масштабованість, структуровану архітектуру MVC, безпеку, високу швидкість завантаження сторінок, вбудовані механізми аутентифікації, широку документацію, інструменти тестування, зручну й безпечну роботу з базами даних [1].

Фреймворк React було обрано для реалізації інтерактивних компонентів на стороні клієнта. React дозволяє створювати багаторазові UI-компоненти, оновлювати DOM без повного перезавантаження сторінки, що покращує продуктивність і взаємодію з користувачем [2, 6].

Для оформлення інтерфейсу використовується Tailwind CSS. Він дозволяє зменшити кількість кастомного CSS-коду та забезпечує уніфіковану стилізацію [7].

У якості системи управління базами даних (СУБД) використовується MySQL, яка є однією з найпоширеніших реляційних СУБД з відкритим кодом. MySQL забезпечує високу продуктивність, масштабованість і просту інтеграцію з Laravel.

Для локального розгортання серверного середовища обрано WAMP – програмний пакет, який включає Apache, MySQL та PHP. Його було використано для тестування функціоналу без необхідності розгортання на хостингу, а також для забезпечення взаємодії між PHP-кодом Laravel і сервером бази даних MySQL.

2 ПРОЄКТУВАННЯ

2.1 Діаграма прецедентів

Діаграма варіантів використання (діаграма прецедентів, use-case diagram) – діаграма, за допомогою якої видно відношення між акторами та прецедентами в системі, де актор – хтось або щось, що взаємодіє з системою, прецедент – певна дія, функція, завдання системи [4].

Діаграма прецедентів розроблюваної системи фотоблог наведена на рисунку 2.1.

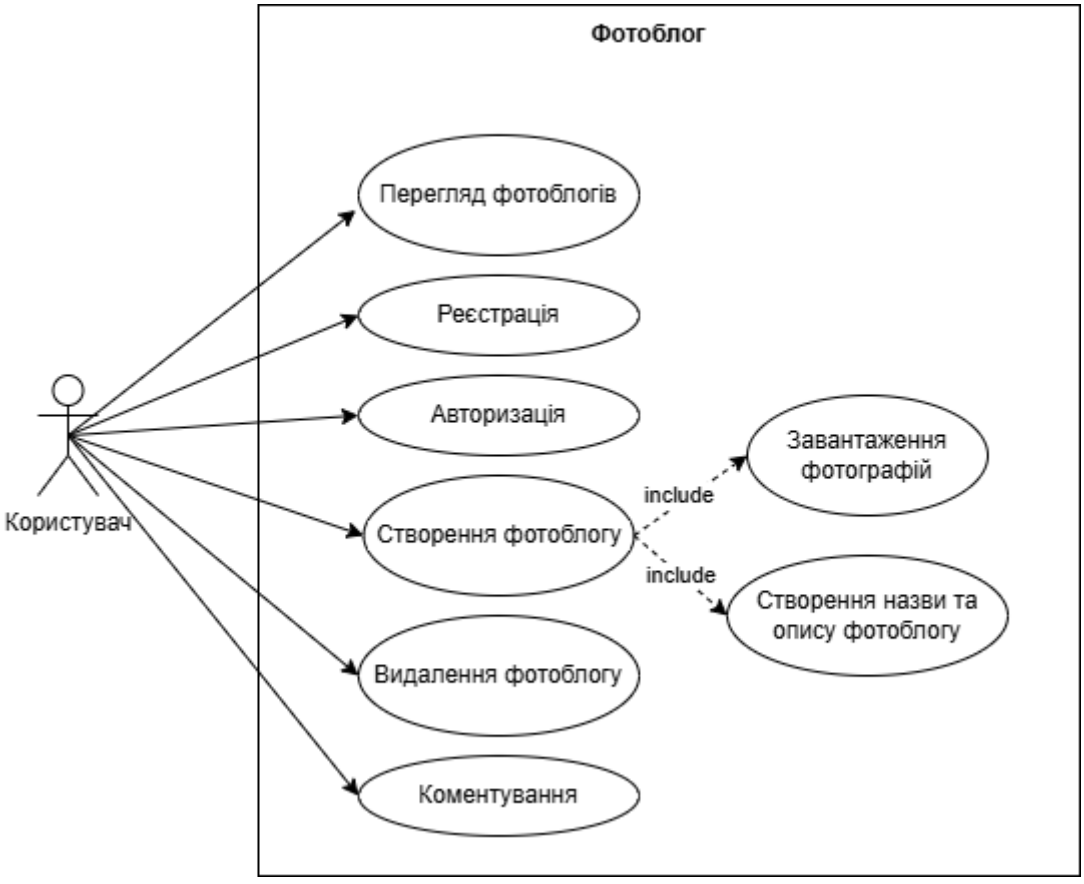


Рисунок 2.1 – Діаграма прецедентів вебзастосунку фотоблог

З вищенаведеної діаграми прецедентів можна зробити висновок, що розроблювана система надає користувачеві базовий набір функцій для роботи з фотоблогами. Основним актором є «Користувач», який має змогу

переглядати, створювати та видаляти, коментувати фотоблоги, створювати новий обліковий запис та входити у систему.

2.2 Діаграма послідовності

Діаграма послідовності (sequence diagram) – діаграма, яка показує часові особливості передачі і прийому повідомлень між об’єктами. У вигляді вертикальних ліній зображено процеси або об’єкти, що існують водночас. Надіслані повідомлення візуалізовано у вигляді горизонтальних ліній зі стрілкою, в порядку їх відправлення [4].

З діаграми послідовності можна простежити логіку взаємодії користувача із системою під час реєстрації та авторизації. У процесі беруть участь чотири основні компоненти: користувач, інтерфейс вебзастосунку, контролер і база даних.

Процес реєстрації починається з переходу на форму авторизації, де користувач обирає реєстрацію і заповнює необхідні поля.

Після надсилання форми дані надходять до контролера, який перевіряє унікальність електронної пошти. Якщо конфліктів не виявлено, інформація зберігається у базі даних, а система повідомляє про успішне створення облікового запису.

Для входу користувач вводить свою електронну адресу й пароль, після чого контролер здійснює перевірку. У разі правильності даних система надає доступ до функціоналу вебзастосунку.

Таким чином, діаграма ілюструє послідовність дій, необхідних для безпечної аутентифікації, та демонструє узгоджену взаємодію між компонентами системи.

На рисунку 2.2 зображена діаграма послідовності реєстрації та авторизації.

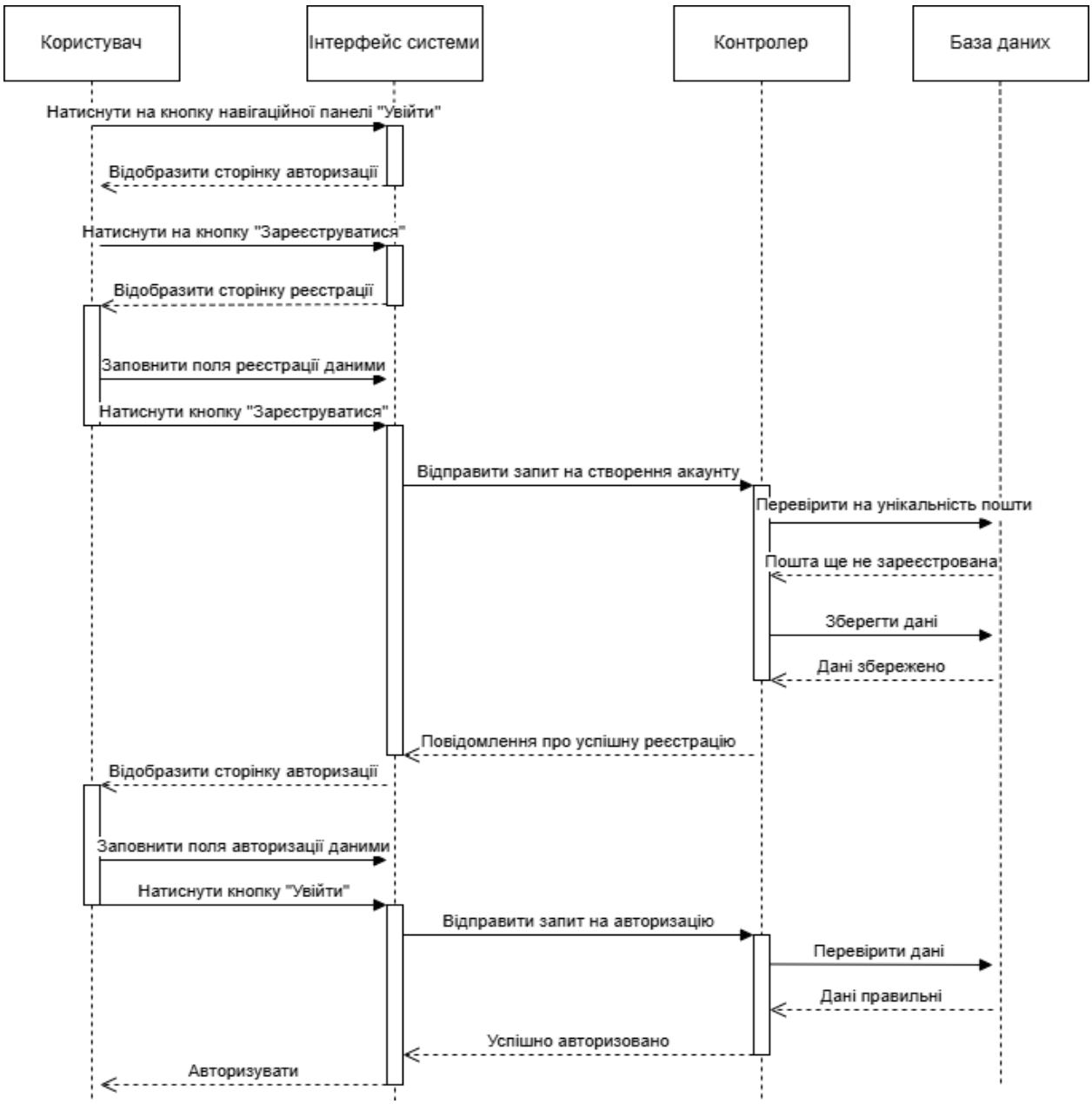


Рисунок 2.2 – Діаграма послідовності реєстрації та авторизації

Після успішної авторизації користувач отримує доступ до функціоналу створення власного фотоблогу. На рисунку 2.3 представлено діаграму послідовності, що ілюструє процес додавання нового блогу.

З діаграми видно, що ініціатором взаємодії виступає користувач, який натискає кнопку «Створення фотоблогу». У відповідь система відображає сторінку з відповідною формою, яку користувач заповнює й надсилає. Після цього дані передаються контролеру, який обробляє запит та надсилає інформацію до бази даних. Після успішного збереження система повертає повідомлення про успішне створення блогу.

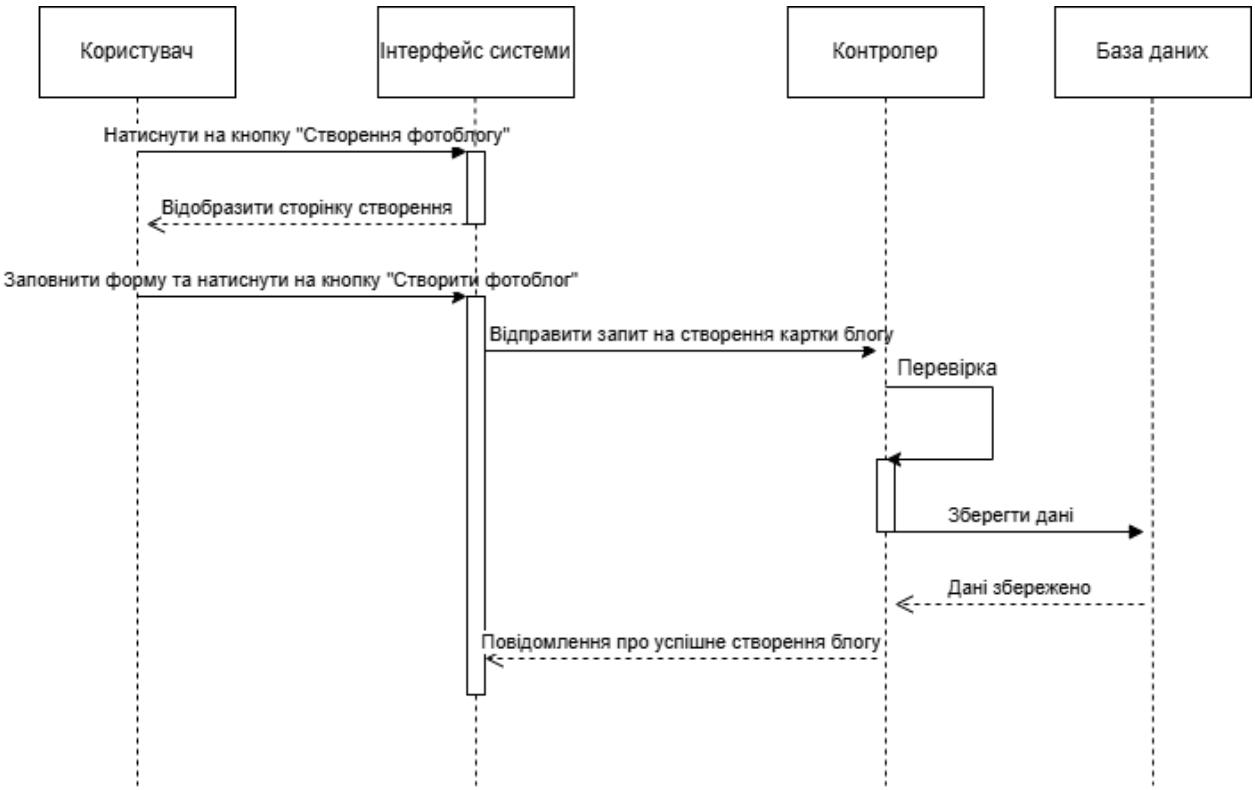


Рисунок 2.3 – Діаграма послідовності створення фотоблогу

2.3 Діаграма класів

Діаграма класів відображає статичні елементи, такі як: класи, типи даних, їх зміст та відношення.

На рисунку 2.4 представлено діаграму класів, яка моделює структуру основних об’єктів системи фотоблогу та взаємозв’язки між ними.

У центрі моделі знаходиться клас User, який представляє зареєстрованого користувача системи.

Кожен користувач має унікальний ідентифікатор, ім’я, адресу електронної пошти, пароль, аватар і опис. До основних операцій, що виконує користувач, належать: реєстрація, авторизація, вихід із системи, редагування аватару та опису в профілі.

Клас Blog містить основну інформацію про фотоблог: унікальний ідентифікатор, заголовок, опис, зображення, дату створення та дату останнього оновлення.

Клас Comment містить текст коментаря, дату створення та прив’язаний як до користувача, що його написав, так і до блогу, до якого коментар стосується.

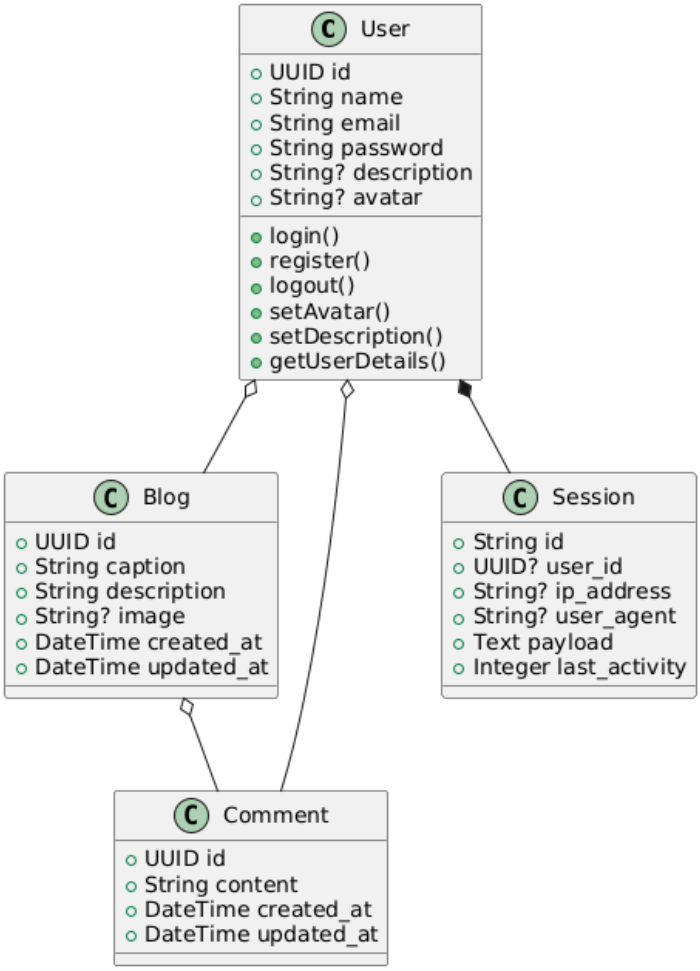


Рисунок 2.4 – Діаграма класів вебзастосунку фотоблог

Окрім цього, система зберігає інформацію про активні сесії користувачів в класі Session. Це дозволяє реалізувати механізм авторизації та відстеження активності.

2.4 Структура бази даних

Проєктування бази даних складається з концептуального, логічного та фізичного проєктування.

Під час концептуального проєктування створюється загальна модель вебзастосунку фотоблог, яка відображає ключові об’єкти системи, їхні характеристики та зв’язки між ними. Така модель дозволяє зрозуміти, як взаємодіють основні складові без деталізації технічної реалізації.

На цьому етапі зазвичай створюється ER-діаграма (діаграма «сутність-зв’язок»), яка відображає сутності, їх атрибути та типи зв’язків між ними.

Основна мета цього етапу – зрозуміти інформаційні потреби системи та відобразити їх у зручному вигляді для подальшого аналізу.

Можна виділити такі основні сутності системи:

– User (id, name, email, password, description, avatar…) – сутність «Користувач»;
– Blog (id, caption, description, image…) – сутність «Фотоблог»;
– Comment (id, content…) – сутність «Коментар»;
– Session (id, user_id…) – сутність «Сесія».

Між сутностями існують зв’язки:

– «Користувач» створює «Фотоблог» та «Коментар»;
– «Користувач» має «Сесію»;
– «Фотоблог» має «Коментар».

Модель «сутність-зв’язок» (ER-модель) (entity-relationship diagram) – модель даних, яка дозволяє описувати концептуальні схеми розроблюваної системи фотоблог за допомогою узагальнених конструкцій блоків (рис.2.5).

Зв’язок «Створює» є зв’язком типу один до багатьох (One-to-Many), так як один користувач може створювати багато фотоблогів та багато коментарів. Атрибути зв’язку: id користувача.

Зв’язок «Має» є зв’язком типу один до багатьох (One-to-Many), так як один користувач може мати багато активних сесій. Атрибути зв’язку: id користувача.

Зв’язок «Залишає» є зв’язком типу один до багатьох (One-to-Many), так як один користувач може залишити багато коментарів. Атрибути зв’язку: id користувача.

Зв’язок «Містить» є зв’язком типу один до багатьох (One-to-Many), так як один фотоблог може містити багато коментарів. Атрибути зв’язку: id фотоблогу.

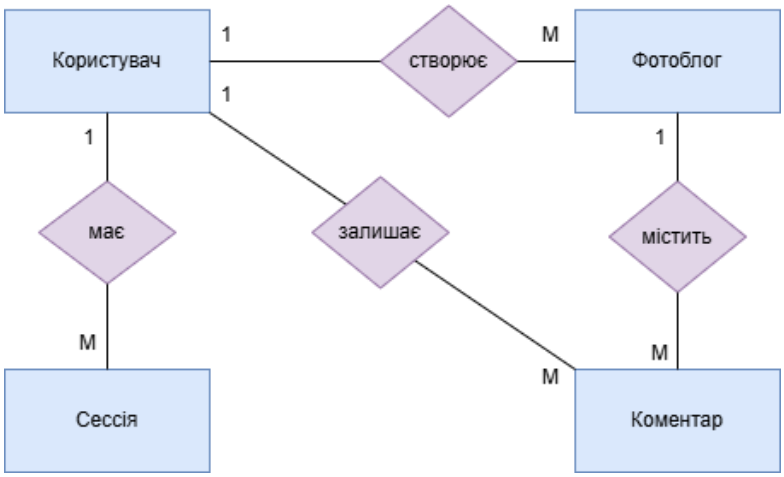


Рисунок 2.5 – ER-модель фотоблогу

Для логічного проєктування створюється схема бази даних на основі отриманих даних під час попереднього етапу. Для реляційної моделі даних – це набір схем відносин, зазвичай із зазначенням первинних ключів, а також «зв’язків» між відносинами, що представляють собою зовнішні ключі.

Розглянемо детальніше логічну модель для фотоблогу.

Користувачі (Users):

– id (PRIMARY KEY) – ідентифікатор користувача;
– name (VARCHAR) – ім’я користувача;
– email (VARCHAR, UNIQUE) – електронна пошта (унікальна);
– password (VARCHAR) – пароль;
– description (TEXT) – опис профілю;
– avatar (VARCHAR) – аватар профілю;
– created_at (DATETIME) – дата створення користувача;
– updated_at (DATETIME) – дата оновлення користувача.

Блоги (Blogs):

– id (PRIMARY KEY) – ідентифікатор фотоблогу;
– caption (VARCHAR) – заголовок;

– user_id (FOREIGN KEY) – зовнішній ключ до користувача, вказує хто створив блог;
– description (TEXT) – опис;
– image (VARCHAR) – зображення;
– created_at (DATETIME) – дата створення блогу;
– updated_at (DATETIME) – дата оновлення блогу.

Коментарі (Comments):

– id (PRIMARY KEY) – ідентифікатор коментаря;
– user_id (FOREIGN KEY) – зовнішній ключ до користувача, вказує хто залишив коментар;
– blog_id (FOREIGN KEY) – зовнішній ключ, вказує до якого блогу відноситься коментар;
– content (TEXT) – текст коментаря;
– created_at (DATETIME) – дата створення коментаря;
– updated_at (DATETIME) – дата оновлення блогу.

Сесія (Session):

– id (PRIMARY KEY) – ідентифікатор сесії;
– user_id (FOREIGN KEY) – зовнішній ключ до користувача, якому належить сесія;
– ip_address (VARCHAR (45)) – ip-адреса, з якої користувач звертається до системи;
– user_agent (TEXT) – клієнтський браузер і пристрій користувача;
– payload (LONGTEXT) – зашифровані дані сесії, що зберігають її поточний стан;
– last_activity (INTEGER) – часовий маркер останньої активності в сесії.

Фізичне проєктування бази даних полягає у перетворенні логічної моделі даних у реалізацію, сумісну з конкретною системою управління базами даних (СУБД), наприклад, MySQL. На цьому етапі створюється схема бази даних, обираються відповідні типи даних для кожного поля, встановлюються

первинні та зовнішні ключі, створюються індекси, а також реалізуються обмеження цілісності.

Особливістю реалізації фізичної моделі в MySQL є вбудована підтримка зовнішніх ключів, індексації та блокування на рівні окремих записів. Це дозволяє зберігати взаємозв’язки між таблицями, забезпечувати контроль над змінами даних, підвищувати цілісність і узгодженість інформації в системі.

На рисунку 2.6 зображено фізичну схему бази даних вебзастосунку фотоблог, реалізовану для СУБД MySQL. У моделі представлено чотири таблиці: User, Blog, Comment та Session, між якими встановлено зв’язки через зовнішні ключі.

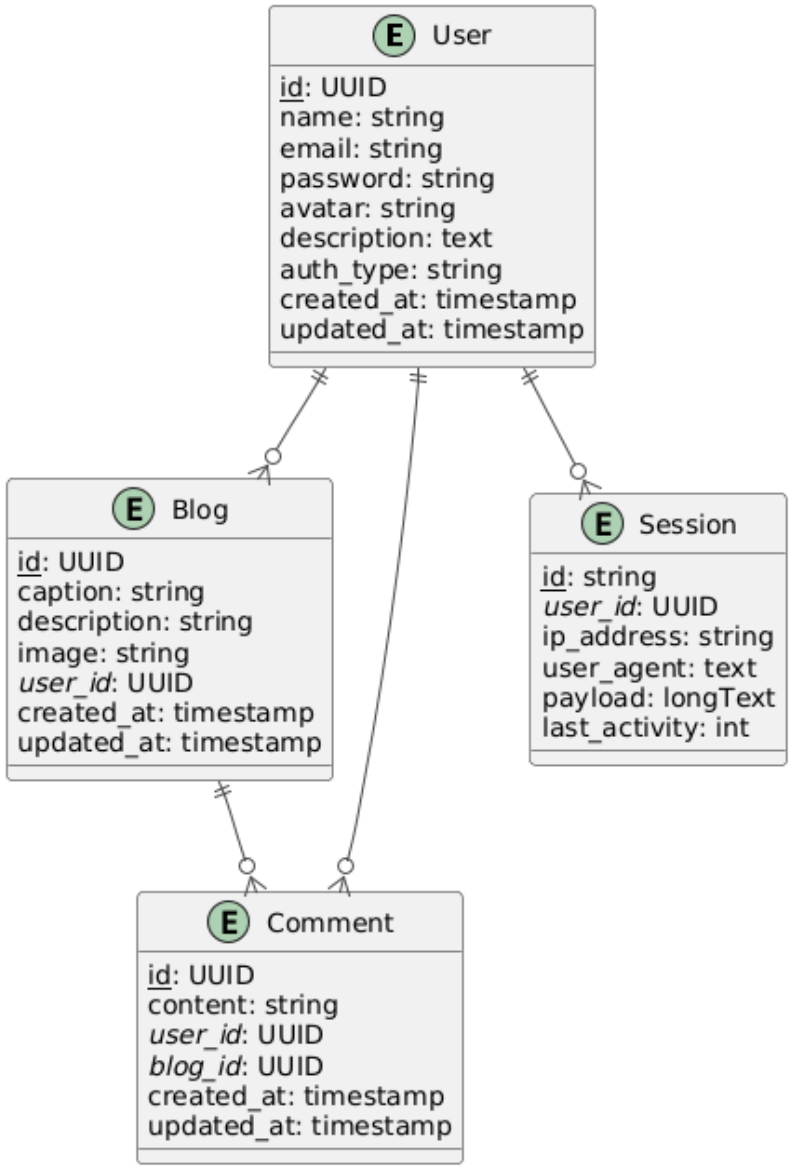


Рисунок 2.6 – Схема бази даних вебзастосунку фотоблог

Таблиця User містить основну інформацію про користувачів системи: унікальний ідентифікатор (UUID), ім’я, електронну пошту, пароль, аватар, опис, тип авторизації (auth_type), а також мітки часу створення та оновлення запису. Вона є центральною сутністю, з якою пов’язано інші таблиці.

Таблиця Blog зберігає дані про фотоблоги: унікальний ідентифікатор, заголовок, опис, шлях до зображення, зовнішній ключ user_id, дати створення та останнього оновлення блогу.

Таблиця Comment описує коментарі до фотоблогів. Вона містить ідентифікатор, текст, зовнішні ключі user_id та blog_id, а також мітки часу. Таким чином забезпечується зв’язок між користувачем, блогом і коментарем.

Таблиця Session реалізує механізм збереження активних сесій користувачів. Вона включає ідентифікатор сесії, зовнішній ключ user_id, IP-адресу, дані про клієнтський застосунок, структуру збережених даних та останню активність у вигляді цілого числа.

2.5 Діаграма компонентів

Компонентна діаграма (Component diagram) – це структурна діаграма, яка відображає основні програмні компоненти системи, а також взаємозв’язки та залежності між ними. Вона дозволяє зрозуміти, як логічно організовані частини застосунку та як вони взаємодіють між собою [4].

На рисунку 2.7 зображено діаграму компонентів вебзастосунку фотоблог.

Діаграма компонентів демонструє архітектуру вебзастосунку «Фотоблог», розділену на клієнтську частину, серверну логіку та зовнішній сервіс – базу даних.

Клієнтська частина (Front-end) створена з використанням фреймворку React. React UI – це єдиний сторінковий застосунок (Single Page Application). Він відповідає за візуальне представлення, інтерфейс взаємодії з користувачем, а також за надсилання HTTP-запитів до серверного API [2, 6].

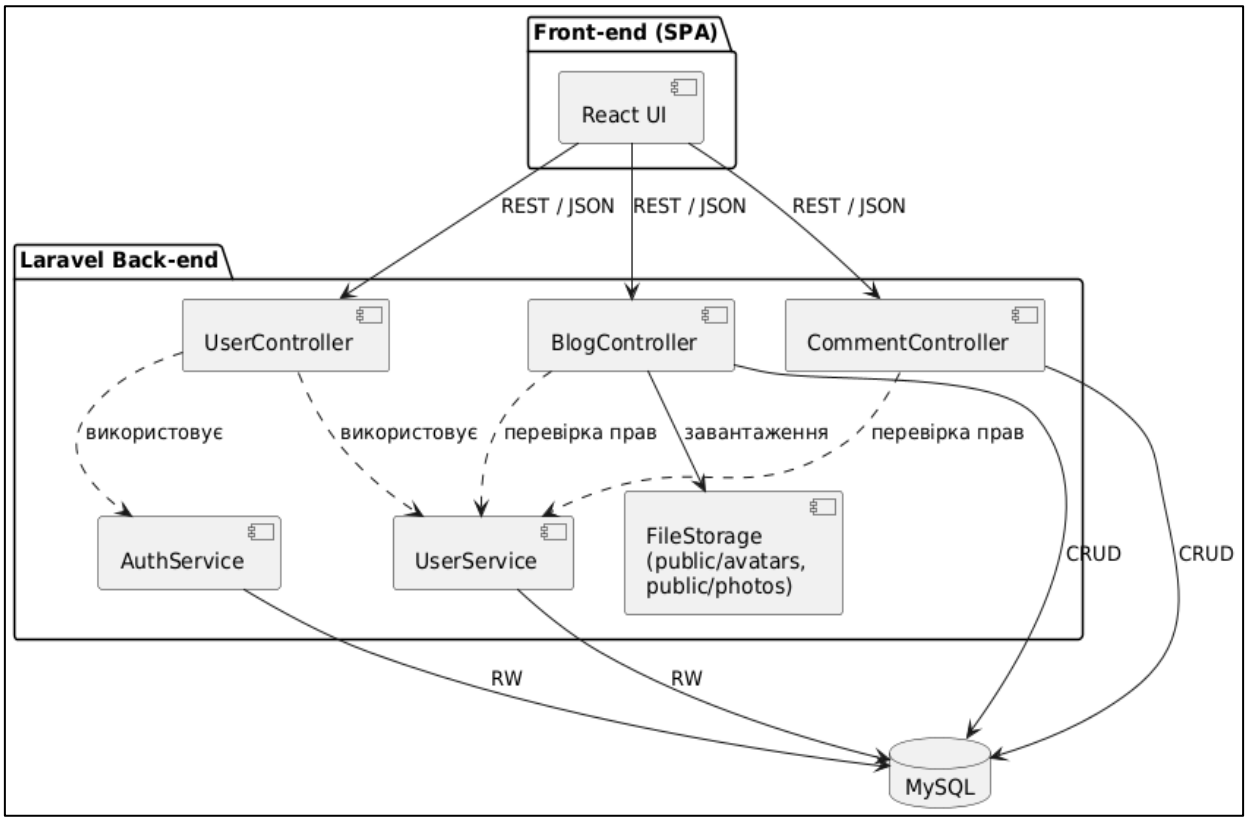


Рисунок 2.7 – Діаграма компонентів вебзастосунку фотоблог

Серверна частина (Laravel Back-end) створена з використанням фреймворку Laravel. У цій частині знаходиться декілька компонентів: UserController, BlogController, CommentController, AuthService, UserService та FileStorage.

UserController – контролер, що обробляє запити, пов’язані з автентифікацією та користувачами (реєстрація, логін, оновлення профілю тощо). Він не містить бізнес-логіки напряму, а делегує її відповідним сервісам (AuthService та UserService).

BlogController – компонент, який відповідає за створення, перегляд, редагування та видалення фотоблогів. Виконує CRUD-операції (Create, Read, Update, Delete) з таблицею блогів та взаємодіє з файловим сховищем для обробки завантажень зображень.

CommentController – контролер, що відповідає за логіку створення, перегляду та видалення коментарів до фотоблогів.

AuthService – сервіс, що реалізує логіку автентифікації користувачів: перевірку облікових даних, створення сесій, керування токенами.

UserService – сервіс, який надає методи для взаємодії з користувачами: перевірка прав доступу, оновлення профілю, зміна аватара тощо.

FileStorage (public/avatars, public/photos) – абстракція над файловою системою, яка зберігає мультимедійні дані. Сховище обробляє завантаження аватарів користувачів, а також зображень для блогів. Залежно від налаштувань Laravel може бути локальним або мережевим.

База даних (MySQL) – реляційна СУБД, де зберігаються всі структуровані дані: користувачі, блоги, коментарі, сесії тощо. До системи бази даних мають доступ майже усі компоненти серверної частини: AuthService перевіряє облікові дані; UserService звертається для того, щоб змінити або отримати інформацію про користувача; BlogController, CommentController CRUD-операції з відповідними моделями.

2.6 Діаграма розгортання

Діаграма розгортання (Deployment diagram) – діаграма, яка відображає фізичну структуру на якій розгорнуто вебзастосунок [4].

На рисунку 2.8 зображена діаграма розгортання вебзастосунку фотоблог.

User Device (браузер) – фізичний пристрій користувача, з якого здійснюється доступ до вебзастосунку. Вузол містить компонент Web Browser, який завантажує React-застосунок, що виконується на стороні клієнта. Уся взаємодія відбувається через інтерфейс користувача.

Web Server (Windows + Apache + PHP) – середовище, де розгорнуто серверну частину застосунку на Laravel. Тут розміщено сукупність серверного коду, який обробляє запити, взаємодіє з базою даних і файловим сховищем. Працює в середовищі WAMP.

MySQL Server – вузол, що надає реляційну базу даних. Він обслуговує запити на зчитування та запис даних із Laravel-додатку.

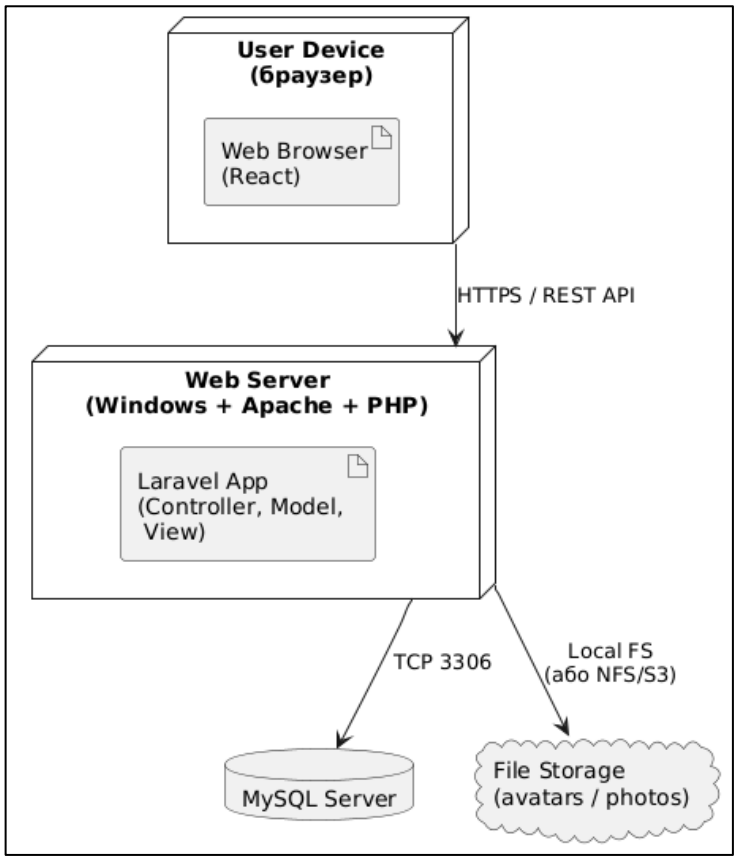


Рисунок 2.8 – Діаграма розгортання вебзастосунку фотоблог

File Storage – зовнішнє файлове сховище, де застосунок зберігає завантажені аватари й зображення до блогів.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Підготовка середовища розробки

На початковому етапі реалізації вебзастосунку потрібно здійснити налаштування середовища розробки.

Першим кроком встановлено локальний вебсервер WAMP. Для завантаження актуальної версії WAMP було використано офіційний сайт: https://www.wampserver.com/en/.

Після встановлення вебсерверу створено Laravel проєкт у теці wamp64\www [3]. Саме в цій директорії зберігаються всі вебзастосунки, які запускаються локально через Apache.

Для реалізації клієнтської частини системи фотоблог встановлено React.js через термінал Visual Studio Code. Підключено до фреймворку Laravel за допомогою Vite – сучасного інструменту для збірки фронтенд-проєктів.

3.2 Створення та підключення бази даних

Для того, щоб створити таблиці у базі даних MySQL потрібно налаштувати підключення із Laravel-проєктом у конфігураційному файлі .env (рис. 3.1).

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel_photoblog
DB_USERNAME=root
DB_PASSWORD=
```

Рисунок 3.1 – Налаштування для підключення бази даних

Далі, щоб відтворити структуру спроєктовану у другому розділі, було використано вбудований механізм міграцій Laravel. Міграції дозволяють зберігати схему бази даних у вигляді PHP-скриптів та виконувати їх однією командою [1, 5].

Міграції таблиць users, sessions, blogs та comments будуть мати такий вигляд, як на рисунках 3.2 – 3.4.

Таблиця users (див. рис. 3.2) має унікальний первинний ключ з типом UUID; поля name, email, password, які є обов'язковими до заповнення; description та avatar – необов'язковими; auth_type, який за замовчуванням містить значення, що дозволяє у майбутньому підтримувати інші способи авторизації; created_at, updated_at – традиційні мітки часу, які створюються автоматично.

Таблиця sessions (див. рис. 3.2) має первинний ключ з типом string, зовнішній ключ user_id, ip-адресу, user_agent, payload, last_activity.

```php
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 'users', callback: function (Blueprint $table): void {
            $table->uuid(column: 'id')->primary()->unique();
            $table->string(column: 'name');
            $table->text(column: 'description')->nullable();
            $table->string(column: 'email')->unique();
            $table->string(column: 'password');
            $table->string(column: 'auth_type')->default(value: 'password'); // Тип авт
            $table->string(column: 'avatar')->nullable();
            $table->timestamps();
        });


        Schema::create(table: 'sessions', callback: function (Blueprint $table): void {
            $table->string(column: 'id')->primary();
            $table->foreignId(column: 'user_id')->nullable()->index();
            $table->string(column: 'ip_address', length: 45)->nullable();
            $table->text(column: 'user_agent')->nullable();
            $table->longText(column: 'payload');
            $table->integer(column: 'last_activity')->index();
        });
    }
```

Рисунок 3.2 – Міграції таблиць users та sessions

Таблиця на рисунку 3.3 має унікальний первинний ключ з типом UUID, два обов’язкових поля caption та description, image, вторинний ключ user_id прив’язаний до таблиці users та мітки часу.

```php
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 'blogs', callback: function (Blueprint $table): void {
            $table->uuid(column: "id")->primary()->unique();
            $table->string(column: "caption");
            $table->string(column: "description");
            $table->string(column: "image")->nullable();
            $table->foreignUuid(column: "user_id")->constrained(table: "users");
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists(table: 'blogs');
    }
};
```

Рисунок 3.3 – Міграція таблиці blogs

Таблиця comments (див. рис. 3.4) має унікальний первинний ключ з типом UUID, обов’язкове поле content, зовнішній ключ user_id, зовнішній ключ blog_id та мітки часу.

Для взаємодії з базою даних у Laravel застосовується ORM-система Eloquent, що дозволяє працювати з таблицями бази даних як з об’єктами у PHP. Кожна таблиця має відповідну модель, яка реалізує логіку доступу, зв’язків і обробки даних.

У системі було створено три основні моделі:

– Blog.php;

– Comment.php;

– User.php.

Зміст файлів наведено на рисунках 3.5 – 3.7.

```php
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 'comments', callback: function (Blueprint $table): void {
            $table->uuid(column: "id")->primary()->unique();
            $table->string(column: "content");
            $table->foreignUuid(column: "user_id")->constrained(table: "users");
            $table->foreignUuid(column: "blog_id")->constrained(table: "blogs");
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists(table: 'comments');
    }
};
```

Рисунок 3.4 – Міграція таблиці comments

Модель Blog (див. рис. 3.5) описує сутність фотоблогу. У ній визначено зв’язок із моделлю користувача через метод user() (відношення «належить до»), а також зв’язок з коментарями через метод comments() (відношення «має багато»).

```php
class Blog extends Model
{
    use HasFactory, HasUuids;

    0 references
    protected $fillable = [
        "caption",
        "description",
        "image"
    ];

    0 references | 0 overrides
    public function user(): BelongsTo
    {
        return $this->belongsTo(related: User::class);
    }

    0 references | 0 overrides
    public function comments(): HasMany
    {
        return $this->hasMany(related: Comment::class);
    }
}
```

Рисунок 3.5 – Модель фотоблогу Blog.php

Модель Comment (див. рис. 3.6) реалізує сутність коментаря до блогу. Тут міститься посилання на користувача та блог. Відповідні методи визначають реляційні зв’язки типу належить до.

```php
class Comment extends Model
{
    use HasFactory;
    use HasUuids;

    0 references
    protected $fillable = [
        "user_id",
        "blog_id",
        "content"

    ];

    0 references | 0 overrides
    public function user(): BelongsTo
    {
        return $this->belongsTo(related: User::class);
    }

    0 references | 0 overrides
    public function blog(): BelongsTo
    {
        return $this->belongsTo(related: Blog::class);
```

Рисунок 3.6 – Модель коментаря Comment.php

За допомогою моделі User (див. рис. 3.7) у програмі обробляються всі дані, пов’язані з користувачами. Модель містить перелік полів, які призначені для масового призначення (fillable) та перелік полів, які потрібно приховати під час серіалізації (hidden).

```php
class User extends Authenticatable
    use HasFactory, Notifiable, HasUuids, HasApiTokens;

    /**
     * The attributes that are mass assignable.
     *
     * @var list<string>
     */
    0 references
    protected $fillable = [
        'name',
        'email',
        'password',
        'description',
        'avatar',
        'auth_type'
    ];

    /**
     * The attributes that should be hidden for serialization
     *
     * @var list<string>
     */
    0 references
    protected $hidden = [
        'password',
        'remember_token',
    ];
```

Рисунок 3.7 – Модель користувача User.php

3.3 Керівництво користувача

У цьому пункті наведено приклади використання реалізованого вебзастосунку «Фотоблог» з точки зору кінцевого користувача. Користувач має змогу зареєструвати обліковий запис, увійти в систему, створювати фотоблоги, переглядати записи інших користувачів і залишати коментарі. Для демонстрації роботи інтерфейсу наведено серію скріншотів з відповідними поясненнями.

При вході на сайт, система відображає головну сторінку (рис. 3.8).

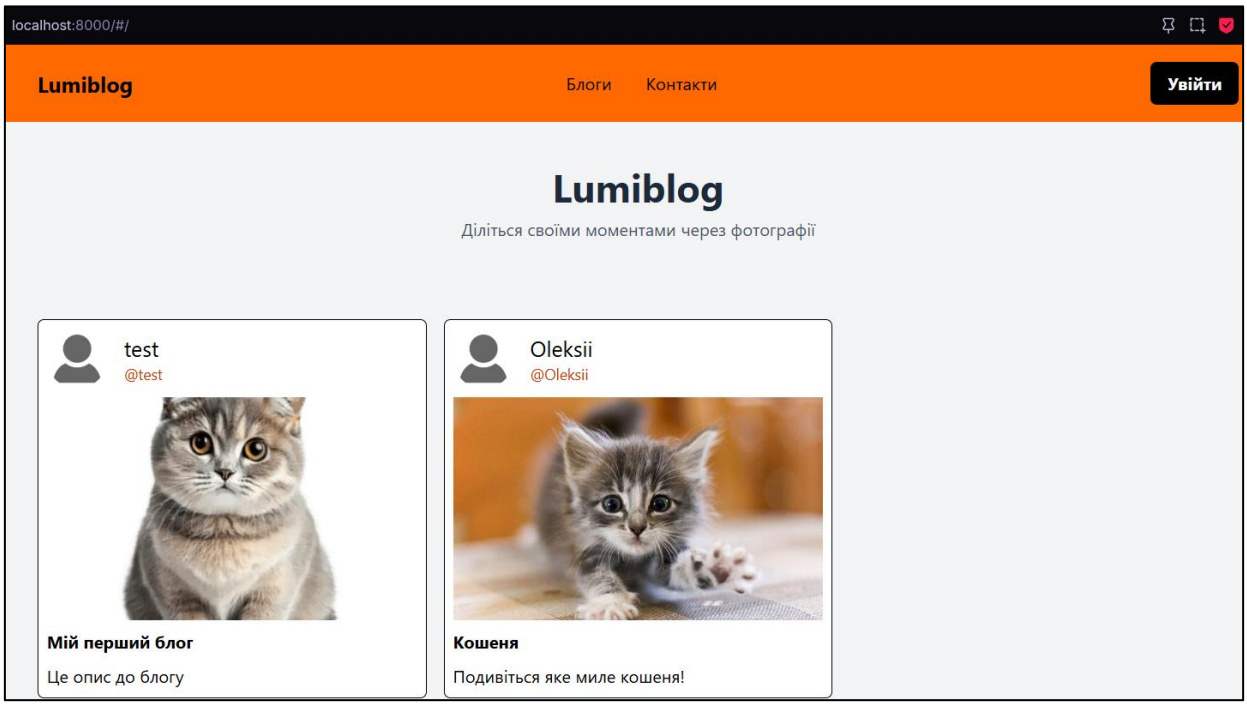


Рисунок 3.8 – Головна сторінка фотоблогу

На головній сторінці користувач може переглядати всі доступні фотоблоги.

Але для створення блогів користувачеві потрібно пройти авторизацію. Кнопка «Увійти» знаходиться у правому верхньому куті. При натисканні, система відобразить сторінку авторизації (див. рис. 3.9).

Для реєстрації потрібно натиснути кнопку «Зареєструватися». Після цього система відобразить сторінку реєстрації (див. рис. 3.10).

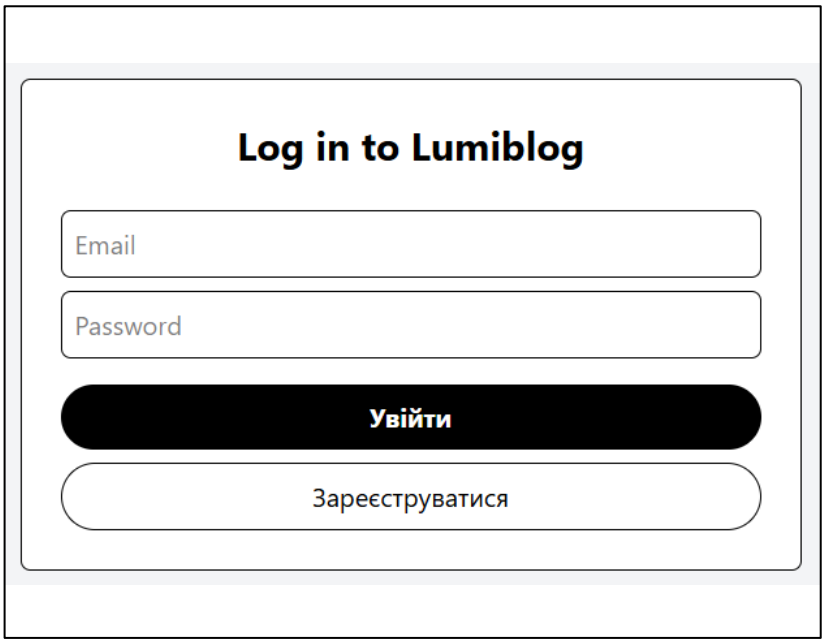


Рисунок 3.9 – Сторінка авторизації вебзастосунку

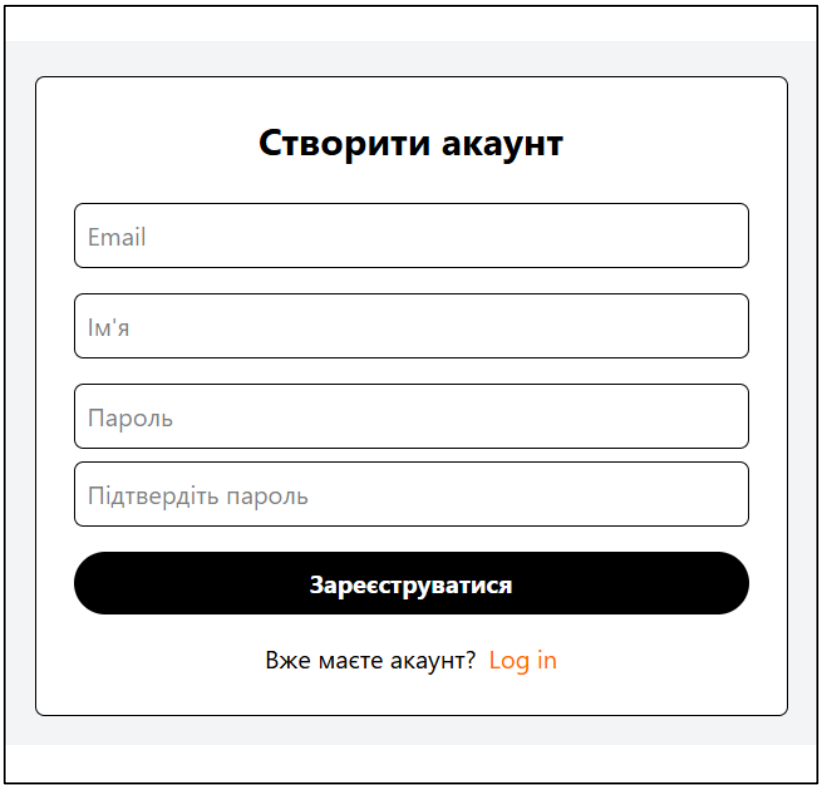


Рисунок 3.10 – Сторінка реєстрації вебзастосунку

Якщо дані вірні, то система без помилок зареєструє користувача та перенаправить його на сторінку авторизації.

Якщо нічого не вводити, то система відобразить таку помилку (див. рис. 3.11).

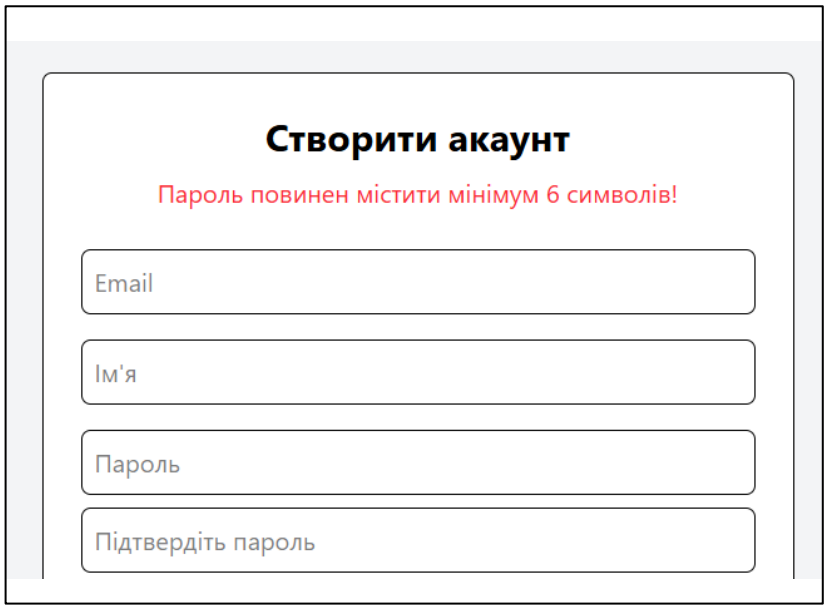


Рисунок 3.11 – Відсутні дані у формі реєстрації

Якщо ввести вже існуючу електронну пошту у базі даних вебзастосунку, то система сповістить про помилку (див. рис. 3.12).

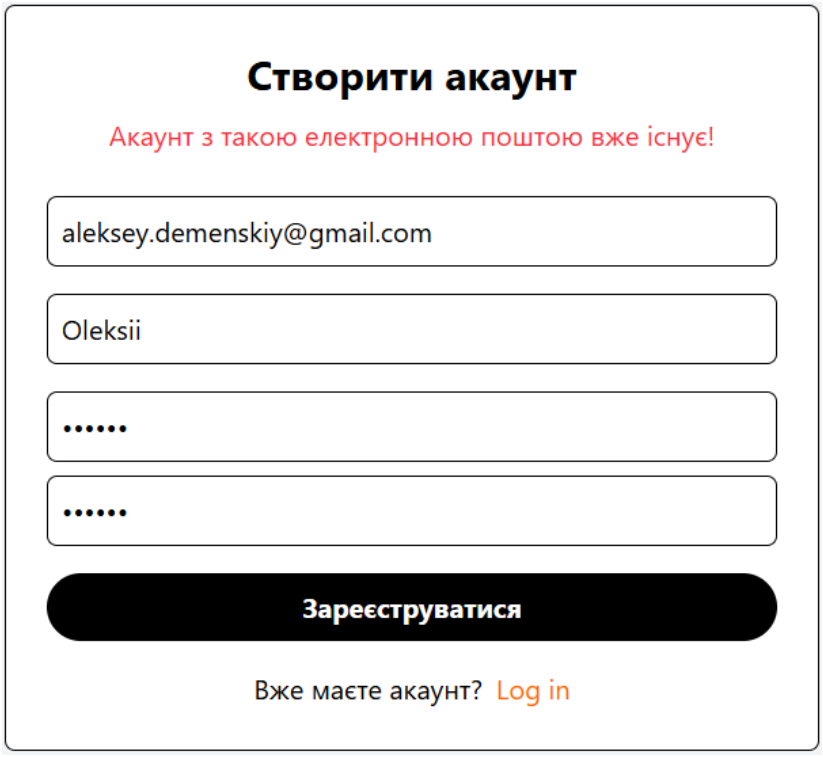


Рисунок 3.12 – Існуюча електронна пошта у формі реєстрації

Якщо не співпадають паролі у полях «Пароль» і «Підтвердіть пароль», то система виведе повідомлення про відповідну помилку (див. рис. 3.13).

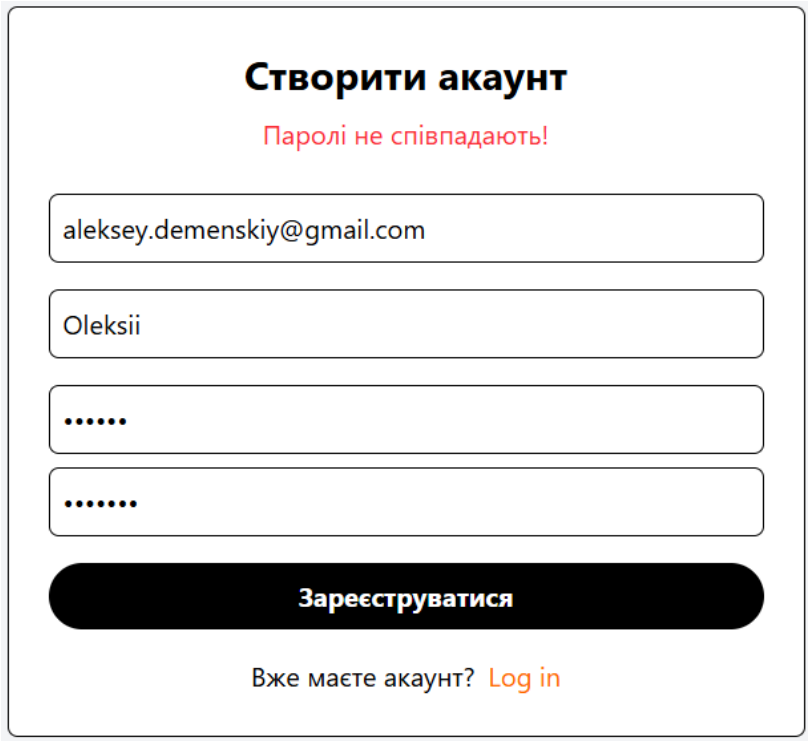


Рисунок 3.13 – Розбіжність паролів у формі реєстрації

Після вдалої реєстрації система перенаправляє користувача до форми авторизації. Успішний вхід до системи у свою чергу відправить користувача до головної сторінки фотоблогу, де зможе створити свій блог. Також у правому верхньому кутку з’явиться кнопка «Вийти» (рис. 3.14).

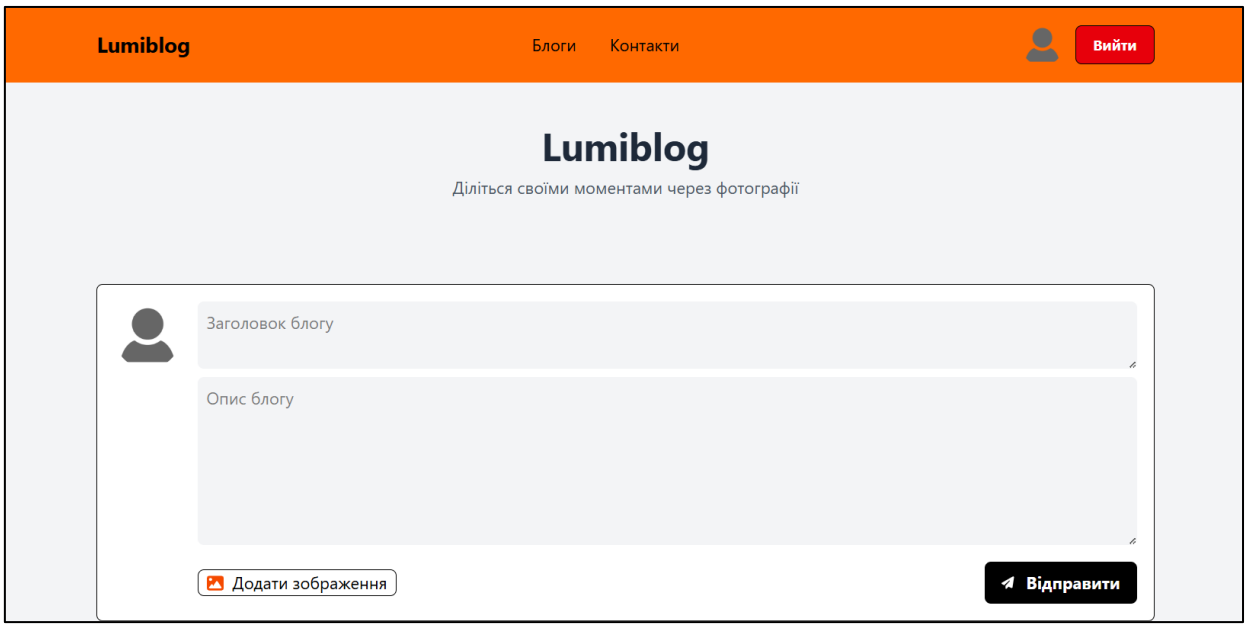


Рисунок 3.14 – Вигляд головної сторінки авторизованого користувача у системі

Тепер користувач заповнює поля із заголовком та описом, додає зображення, натиснувши кнопку «Додати зображення», і коли він натисне на кнопку «Відправити» буде створено новий фотоблог. Також авторизований користувач може додавати коментарі.

На рисунку 3.15 показано приклад додавання коментаря до блогу.

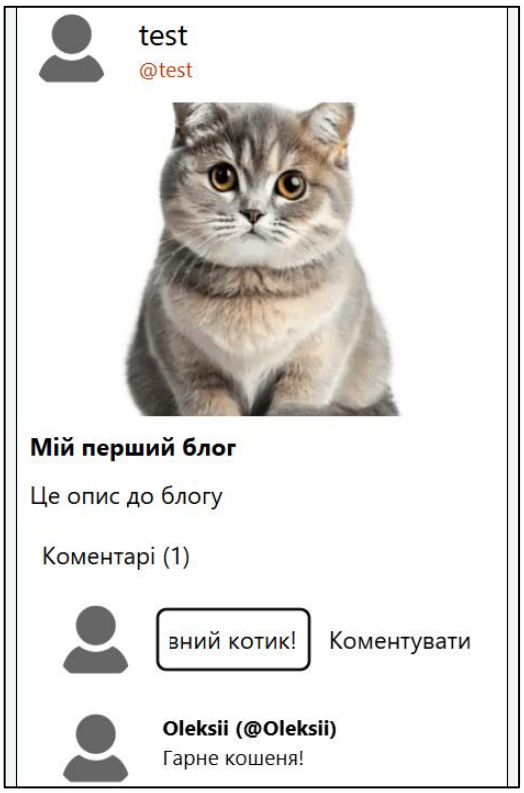


Рисунок 3.15 – Додавання коментаря до блогу

Після натискання на кнопку «Коментувати» коментар буде додано до блогу.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було сформовано технічне завдання на створення вебзастосунку фотоблог. Для реалізації системи були використані сучасні фреймворки Laravel та React.

Відповідно до поставленої мети розроблено функціональний вебдодаток що дає змогу користувачам реєструватися, авторизовуватися, створювати фотоблоги та залишати під ними коментарі.

У відповідності з поставленими завданнями було виконано такі етапи створення системи:

– сформовано вимоги до системи (визначено основну функціональність фотоблогу, його модулі, обрано стек технолгій і методи реалізації);
– спроєктовано архітектуру вебзастосунку (побудовано модель бази даних, створено діаграми варіантів використання, послідовності, класів, компонентів, розгортання);
– реалізовано вебдодаток фотоблог: розроблено клієнтську частину на основі React, яка взаємодіє з сервером, та серверну частину на Laravel, що обробляє запити, керує користувачами, базою даних і файловим сховищем;
– здійснено тестування системи: перевірено коректну роботу авторизації, реєстрації, створення блогів, перегляду публікацій і залишення коментарів, проведено тестування введення вірних та невірних даних.

Загалом, робота досягла поставленої мети – створено сучасний вебзастосунок для ведення фотоблогу з базовою функціональністю, масштабованою архітектурою та зручним інтерфейсом.

У подальшому проєкт може бути розширений шляхом додавання можливостей редагування публікацій, системи оцінювання (лайків), Google авторизації, пошуку і фільтрації блогів та категорії фотоблогів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Minasian J. Master Laravel for Beginners : Get from Zero to Proficiency in the Laravel Framework: Laravel Book. Independently Published, 2021. 109 p.

2. Narayn H. Just React! : Learn React the React Way. New York : Apress L. P., 2022. 369 p.

3. Laravel 12.x Documentation : The PHP Framework for Web Artisans. *Laravel*. URL: https://laravel.com/docs/12.x (дата звернення: 14.04.2025).

4. Rumbaugh J., Jacobson I., Booch G. The unified modeling language reference manual. Reading, Mass : Addison-Wesley, 1998. 550 p.

5. Vickler A. PHP : This Book Includes: PHP Basics for Beginners + PHP Security and Session Management + Advanced PHP Functions. Ladoo Publishing LLC, 2022. 576 p.

6. Kumar T. Fluent React: Build Fast, Performant, and Intuitive Web Applications. Sebastopol, CA : O'Reilly Media, 2024. 334 p.

7. Rappin N. Modern CSS with Tailwind: Flexible Styling Without the Fuss. Raleigh, NC : Pragmatic Bookshelf, 2021. 92 p.