МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ Економіки та бізнес-освіти

Кафедра Економіки та цифрового бізнесу

Спеціальність 122 «Комп'ютерні науки»

Форма навчання Денна

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Лісовець Валерії Сергіївни
(прізвище, ім'я, по батькові здобувача)

на тему Розробка вебзастосунку для публікації та обміну фото-контентом
(повна назва теми)

за матеріалами

(повна назва бази дослідження)

науковий керівник к.т.н. _____ Селезньов М.Є.
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 12 червня 2026 р. № 15
Завідувач кафедри _____
(підпис)
к.е.н., доцент Радько В.М.
наук. ступень, вчене звання прізвище, ініціали

Кривий Ріг – 2026

ЗАТВЕРДЖЕНО
Наказ Міністерства освіти і науки, молоді та спорту України
29 березня 2012 року № 384

Форма № Н-9.01

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ ТА БІЗНЕС-ОСВІТИ
(повне найменування вищого навчального закладу)

Кафедра економіки та цифрового бізнесу
Освітній ступінь бакалавр
Спеціальність 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри _____ В.М. Радько

"30" березня 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ ЗДОБУВАЧУ

Лісовець Валерії Сергіївні

1. Тема роботи Розробка вебзастосунку для публікації та обміну фото-контентом

науковий керівник роботи Селезньов Максим Євгенович, к.т.н.
затверджені наказом вищого навчального закладу від «23» березня 2026 р. № 193-ст

2. Строк подання здобувачем роботи 29.05.2026р.

3. Зміст кваліфікаційної бакалаврської роботи, об'єкт, предмет та мета дослідження:
Розділ 1 Теоретичні аспекти розробки вебзастосунку для публікації та обміну фото-контентом
Розділ 2 Проєктування та розробка вебзастосунку для публікації та обміну фото-контентом
Розділ 3 Функціонування та елементи інтерфейсу вебзастосунку для публікації та обміну фото-контентом

Об'єкт дослідження – процеси публікації та обміну фото-контентом у сучасному вебсередовищі

Предмет дослідження - методи, програмні засоби, технології та архітектурні підходи до розробки вебзастосунку для публікації та обміну фото-контентом

Мета кваліфікаційної бакалаврської роботи – розробка вебзастосунку для публікації та обміну фото-контентом, який забезпечує зручне керування зображеннями, взаємодію між користувачами, безпечне завантаження файлів та можливість подальшого розширення функціональності

4. Дата видачі завдання 03.04.2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	до 17.04.2026р.	16.04.2026р.
2	Підготовка розділу 2	до 08.05.2026р.	07.05.2026р.
3	Підготовка розділу 3	до 25.05.2026р.	24.05.2026р.
4	Реєстрація завершеної кваліфікаційної роботи	до 29.05.2026р.	28.05.2026р.
5	Отримання відгуку від наукового керівника	04.06.2026р.	04.06.2026р.
6	Отримання зовнішньої рецензії	05.06.2026р.	05.06.2026р.
7	Попередній захист кваліфікаційної роботи на кафедрі	08.06.2026р.	08.06.2026р.
8	Перевірка кваліфікаційної роботи на плагіат	12.06.2026р.	12.06.2026р.
9	Допуск кафедрою кваліфікаційної роботи до захисту	12.06.2026р.	12.06.2026р.
10	Підготовка студента до захисту в ЕК	до 19.06.2026р.	18.06.2026р.

Завдання підготував науковий керівник _____ Селезньов М. Є.

Завдання одержав здобувач _____ Лісовець В. С.

РЕФЕРАТ

Робота містить 55 сторінок, 16 рисунків, 1 таблицю, 45 джерел, 1 додаток.

Об’єкт дослідження: процеси публікації та обміну фото-контентом у сучасному вебсередовищі.

Предмет дослідження: методи, програмні засоби, технології та архітектурні підходи до розробки вебзастосунку для публікації та обміну фото-контентом.

Мета роботи: розробка вебзастосунку для публікації та обміну фото-контентом, який забезпечує зручне керування зображеннями, взаємодію між користувачами, безпечне завантаження файлів та можливість подальшого розширення функціональності.

У результаті дослідження було проаналізовано сучасні тенденції розвитку сервісів для публікації та обміну фото-контентом, виконано аналіз існуючих платформ, обґрунтовано вибір технологічного стеку та архітектури вебзастосунку. Було спроєктовано інформаційну модель системи та структуру бази даних, реалізовано механізми реєстрації та автентифікації користувачів, підтвердження електронної пошти, створення та редагування профілів, публікації постів із кількома зображеннями, використання тегів, системи вподобань і підписок. Також реалізовано стрічку публікацій, механізми роботи з медіафайлами та засоби забезпечення безпеки під час завантаження контенту.

Область застосування: результати роботи можуть бути використані під час створення соціальних вебплатформ, сервісів для публікації та обміну фото-контентом, навчальних проєктів у галузі веброзробки, а також як основа для подальшого розвитку багатокористувацьких інформаційних систем.

DJANGO, PYTHON, АВТЕНТИФІКАЦІЯ, БАЗА ДАНИХ, ВЕБЗАСТОСУНОК, ВЕБРОЗРОБКА, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, ПУБЛІКАЦІЯ ЗОБРАЖЕНЬ, СОЦІАЛЬНА МЕРЕЖА, ФОТО-КОНТЕНТ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API – Application Programming Interface (інтерфейс прикладного програмування)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

DBMS – Database Management System (система керування базами даних)

HTML – HyperText Markup Language (мова розмітки гіпертексту)

HTTP – HyperText Transfer Protocol (протокол передавання гіпертексту)

HTTPS – HyperText Transfer Protocol Secure (захищений протокол передавання гіпертексту)

JSON – JavaScript Object Notation (текстовий формат обміну даними)

MVC – Model-View-Controller (архітектурний шаблон проєктування)

MTV – Model-Template-View (архітектурний шаблон Django)

ORM – Object-Relational Mapping (об’єктно-реляційне відображення)

PostgreSQL – Object-Relational Database Management System (об’єктно-реляційна система керування базами даних)

REST – Representational State Transfer (архітектурний стиль взаємодії вебсервісів)

SQL – Structured Query Language (мова структурованих запитів)

UI – User Interface (користувацький інтерфейс)

URL – Uniform Resource Locator (уніфікований покажчик ресурсу)

UX – User Experience (користувацький досвід)

WWW – World Wide Web (всесвітня павутина)

ВСТУП

Сучасний розвиток цифрового середовища супроводжується зростанням ролі візуального контенту в комунікації, самопрезентації та повсякденній взаємодії користувачів. Фото, зображення та інші графічні матеріали стали одним із найпоширеніших форматів передавання інформації в мережі, а сервіси для їх публікації та обміну – важливим елементом цифрової інфраструктури. Поширення мобільних пристроїв, активне використання соціальних мереж і розвиток вебтехнологій сприяли тому, що створення, зберігання та поширення фото-контенту нині є звичною частиною цифрового життя користувачів.

Актуальність теми бакалаврської роботи зумовлена тим, що вебзастосунки для публікації та обміну фото-контентом поєднують одразу кілька важливих напрямів сучасної ІТ-галузі: обробку користувацького контенту, організацію цифрової взаємодії, побудову зручного інтерфейсу, забезпечення безпеки завантаження файлів і підтримку належної продуктивності вебресурсу.

Користувачі очікують від таких систем не лише можливості додавання фотографій, а й швидкого доступу до матеріалів, зручного перегляду, структурованого зберігання, обміну між користувачами, гнучкого керування приватністю та коректної роботи на різних пристроях. Оскільки зображення є одним із найбільш ресурсоємних видів вмісту на вебсторінках, їх використання потребує додаткової уваги до оптимізації, адаптивності та швидкодії. Водночас фото-контент пов’язаний із питаннями конфіденційності, контролю доступу, авторського права та модерації користувацьких матеріалів, що надає обраній темі не лише технічної, а й соціальної значущості.

Проблематика публікації та обміну фото-контентом достатньо широко представлена у вітчизняних і зарубіжних дослідженнях, однак здебільшого висвітлюється через окремі аспекти. У вітчизняному науковому дискурсі увага приділяється ролі візуального контенту в соціальних мережах, особливостям його сприйняття та впливу на аудиторію. У зарубіжних дослідженнях значний акцент зроблено на соціальній природі фотообміну, поведінці користувачів,

прийнятті рішень щодо публікації зображень, а також на проблемах приватності й безпеки в середовищі цифрових платформ. Окремий напрям досліджень стосується користувацького контенту та модерації матеріалів, що розміщуються на онлайн-ресурсах. Це особливо важливо для сервісів обміну фото, оскільки їх основне наповнення формується безпосередньо користувачами.

Паралельно у працях, присвячених сучасним вебзастосункам, розглядаються питання архітектури, класифікації, етапів розробки та побудови функціональних і масштабованих вебсистем. Проте, попри наявність значної кількості джерел, у багатьох із них відсутній комплексний підхід, який поєднував би теоретичне осмислення ролі фото-контенту з практичним проєктуванням і реалізацією вебзастосунку, орієнтованого на публікацію, обмін, безпечне завантаження та зручне керування зображеннями. Саме це зумовлює доцільність подальшого дослідження теми.

Специфіка джерельної бази роботи полягає в її комплексному характері. Інформаційну основу дослідження становлять наукові праці, присвячені візуальному контенту, фотообміну та особливостям функціонування соціальних медіа; дослідження з питань конфіденційності, безпеки, поведінки користувачів і модерації користувацького контенту; матеріали, що розкривають загальні принципи розробки вебзастосунків та сучасні підходи до побудови їх архітектури; а також офіційна технічна документація сучасних програмних засобів, які можуть бути використані для реалізації вебзастосунку. Поєднання наукових і технічних джерел дає змогу обґрунтувати не лише теоретичні засади дослідження, а й практичні рішення, прийняті в процесі розробки. Водночас частина джерел більшою мірою розкриває соціально-комунікаційний бік фото-контенту, тоді як питання конкретної технічної реалізації часто подаються фрагментарно.

Об’єктом дослідження є процеси публікації та обміну фото-контентом у сучасному вебсередовищі. Предметом дослідження є методи, програмні засоби, технології та архітектурні підходи до розробки вебзастосунку для публікації та обміну фото-контентом.

Метою бакалаврської роботи є проєктування та розробка вебзастосунку для публікації та обміну фото-контентом, який забезпечує зручне керування зображеннями, взаємодію користувачів, безпечне завантаження файлів, належний рівень продуктивності та можливість подальшого розширення функціональності.

Для досягнення поставленої мети в роботі необхідно вирішити такі завдання: проаналізувати сучасні тенденції в галузі публікації та обміну фото-контентом; дослідити наявні програмні засоби й сервіси цього призначення та визначити їхні переваги й недоліки; обґрунтувати вибір технологічного стеку й архітектури вебзастосунку; спроєктувати структуру системи, основні сутності та механізми взаємодії між клієнтською та серверною частинами; реалізувати ключовий функціонал вебзастосунку; забезпечити вимоги до безпеки, зручності користування та швидкодії; оцінити практичну придатність розробленого рішення.

Методологічною основою дослідження є сукупність загальнонаукових і спеціальних методів. У роботі використано аналіз, синтез і узагальнення наукових джерел для вивчення теоретичних підходів до організації візуального контенту у вебсередовищі; порівняльний метод – для дослідження функціональних можливостей існуючих сервісів публікації та обміну фотографіями; системний аналіз – для формування вимог до вебзастосунку; методи проєктування програмних систем – для вибору архітектури та логіки взаємодії компонентів; а також практичні методи програмної реалізації, тестування та перевірки працездатності створеного програмного продукту.

Наукова новизна одержаних результатів полягає в узагальненні теоретичних і прикладних підходів до створення вебзастосунку для публікації та обміну фото-контентом та в обґрунтуванні програмного рішення, яке поєднує функції завантаження, публікації, організації, перегляду та обміну зображеннями в межах єдиної системи. На відміну від частини існуючих платформ, що орієнтуються переважно або на архівування фотографій, або на інтенсивну соціальну взаємодію, у межах даної роботи акцент зроблено на поєднанні

функціональності, зручності користування, безпеки завантаження файлів та придатності системи до подальшого розширення.

Практична значущість роботи полягає в тому, що розроблений вебзастосунок може бути використаний як основа для створення сервісу обміну фото-контентом у навчальній, персональній або комерційній сфері. Практичну цінність мають також запропоновані підходи до організації клієнт-серверної взаємодії, структурування даних, побудови API, оптимізації зображень та реалізації базових заходів безпеки при роботі із завантаженням файлів. Результати дослідження можуть бути використані під час створення аналогічних вебсистем і слугувати основою для подальшого розвитку функціональності застосунку.

Отже, тема бакалаврської роботи є своєчасною та значущою як з теоретичного, так і з практичного погляду. Її дослідження дозволяє поєднати аналіз ролі візуального контенту в сучасному цифровому просторі з прикладними завданнями веброзробки та створити програмне рішення, орієнтоване на актуальні потреби користувачів у сфері публікації та обміну фото-контентом.

РОЗДІЛ 1
ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ ВЕБЗАСТОСУНКУ ДЛЯ ПУБЛІКАЦІЇ ТА ОБМІНУ ФОТО-КОНТЕНТОМ

1.1 Огляд сучасних тенденцій в галузі публікації та обміну фото-контентом

У сучасному цифровому середовищі фото-контент посідає одне з провідних місць серед форм онлайн-комунікації. Поширення мобільних пристроїв, розвиток соціальних мереж і зростання доступності мережевих сервісів зумовили активне використання зображень як засобу передавання інформації, самовираження та взаємодії між користувачами. Звіти DataReportal свідчать про подальше зростання цифрової активності, соціальних мереж та візуально орієнтованих форматів взаємодії у глобальному цифровому середовищі [1].

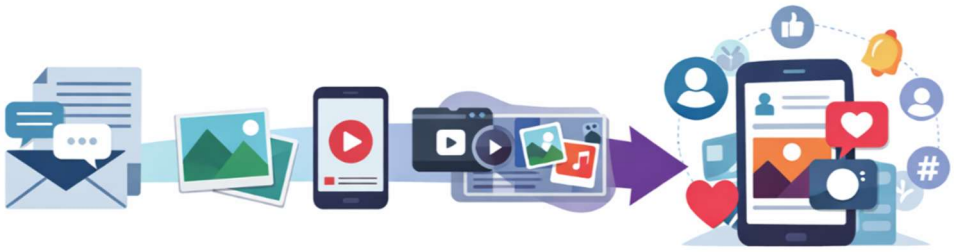

Рис. 1.1. Еволюція цифрової комунікації до візуального формату
(розроблено автором)

Однією з ключових тенденцій є зміщення акценту від традиційних текстових форматів до візуально орієнтованої комунікації. Якщо раніше мережеве спілкування переважно ґрунтувалося на текстових повідомленнях, то нині користувачі активно обмінюються фотографіями, короткими відео, графічними добірками та іншими мультимедійними матеріалами. У цьому

середовищі фотографія виконує не лише ілюстративну функцію, а й стає окремою одиницею цифрової взаємодії, через яку користувач демонструє власні інтереси, спосіб життя та соціальні зв’язки. Популярність Instagram та подібних платформ підтверджує зростання ролі візуального контенту в сучасній комунікації [2].

Ще однією помітною тенденцією є перехід від простих фотохостингів до багатофункціональних вебплатформ із розвиненими механізмами взаємодії. Сучасні сервіси для обміну фото-контентом, як правило, не обмежуються лише завантаженням та переглядом зображень. Вони включають персональні профілі, стрічку оновлень, реакції, коментарі, обмін повідомленнями, а також інші інструменти соціальної взаємодії. Офіційні матеріали Instagram демонструють, що функції публікації контенту тісно поєднуються з інструментами комунікації та персоналізації користувацького досвіду [3].

Для сфери публікації фото-контенту характерною є також тенденція до поглиблення персоналізації. Важливого значення набувають формування індивідуальної стрічки, відображення контенту відповідно до інтересів користувача та його взаємодії з іншими акаунтами. Зокрема, Instagram прямо визначає стрічку як простір, у якому користувач взаємодіє з людьми та темами, що становлять для нього інтерес, що свідчить про орієнтацію сучасних платформ на персональний інформаційний простір [4].

Окремої уваги заслуговує тенденція до ускладнення структури самого фото-контенту. Якщо на ранніх етапах розвитку подібних сервісів основним об’єктом була окрема фотографія, то нині публікація може містити кілька зображень, текстовий опис, позначки, теги та інші додаткові атрибути. Це зумовлює необхідність переходу від простих моделей зберігання файлів до більш складних структур даних, у яких один пост пов’язується з кількома зображеннями та метаданими. Такий підхід є логічним наслідком розвитку сучасних платформ, де візуальний контент інтегрується в ширшу систему взаємопов’язаних сутностей [3].

Важливою тенденцією у сфері фотообміну є посилення уваги до конфіденційності та безпеки. Поширення зображень у мережі пов’язане з ризиками несанкціонованого доступу, небажаного копіювання, витоку персональних даних та завантаження небезпечних файлів. Рекомендації OWASP прямо вказують на необхідність валідації типу файлів, обмеження дозволених форматів, контролю розміру файлів та перевірки прав доступу під час реалізації функції завантаження [5]. Додатково в рекомендаціях OWASP наголошується на доцільності обмеження розміру файлів, перевірки вмісту завантажень, контролю імен файлів та ізоляції файлового сховища від прямого небезпечного доступу [42].

Не менш значущою є тенденція до підвищення вимог до продуктивності вебзастосунків, що працюють із зображеннями. Зображення часто є одними з найважчих ресурсів вебсторінки, а тому їх оптимізація суттєво впливає на швидкість завантаження та загальний користувацький досвід. У матеріалах web.dev зазначається, що оптимізація зображень дозволяє скоротити обсяг переданих даних і покращити продуктивність вебресурсу, що особливо важливо для сервісів, орієнтованих на активний обмін фото-контентом [6]. Практики оптимізації зображень включають зменшення розміру файлів, використання адаптивних форматів, налаштування коректних розмірів медіа та зниження обсягу переданих даних, що безпосередньо впливає на швидкість завантаження сторінок [43]. Окремо в документації MDN підкреслюється важливість використання адаптивних зображень, що дає змогу відображати різні варіанти графічного контенту залежно від розміру екрана та особливостей пристрою користувача [44].

Ще однією характерною тенденцією є підвищення вимог до продуктивності вебзастосунків, що працюють із зображеннями. Фотографії, як правило, займають значний обсяг пам’яті та істотно впливають на швидкість завантаження сторінок. Саме тому сучасні підходи до розробки передбачають оптимізацію зображень, використання адаптивних форматів, встановлення коректних розмірів, попередній перегляд, генерацію прев’ю та відкладене

завантаження медіа. У матеріалах web.dev та MDN зазначається, що зображення є одними з найважчих ресурсів на вебсторінках, а їх оптимізація безпосередньо впливає на якість користувацького досвіду. Для застосунків, орієнтованих на активний обмін фото, це особливо важливо, оскільки від швидкості завантаження та відображення контенту залежить зручність користування сервісом у цілому.

Окрім технічних і соціальних аспектів, сучасна сфера публікації фото-контенту демонструє тенденцію до інтеграції різних технологічних компонентів у межах єдиного вебрішення. Типовий сервіс у цій галузі включає механізми автентифікації, керування профілем, створення публікацій, збереження мультимедійних файлів, побудову стрічки та інструменти взаємодії між користувачами. Такий підхід формує вимоги до вебзастосунку як до цілісної системи, а не лише як до інтерфейсу для розміщення зображень [1; 3].

Отже, сучасні тенденції в галузі публікації та обміну фото-контентом свідчать про перехід від простого розміщення зображень до створення складних інтерактивних вебзастосунків, у яких поєднуються візуальна комунікація, соціальна взаємодія, персоналізація, безпека та висока продуктивність. Фото-контент у сучасному цифровому середовищі виконує не лише інформаційну, а й соціальну, комунікативну та організаційну функції. Саме тому розробка вебзастосунку для публікації та обміну фото-контентом є актуальним напрямом, що потребує врахування як загальних тенденцій розвитку цифрових платформ, так і конкретних технічних вимог до реалізації подібної системи.

1.2 Аналіз існуючих засобів для публікації та обміну фото-контентом

Для визначення вимог до вебзастосунку для публікації та обміну фото-контентом доцільно проаналізувати вже існуючі цифрові сервіси, які реалізують подібну функціональність. Серед найбільш відомих і поширених засобів у цій галузі можна виокремити Instagram, Flickr, Google Photos та Pinterest. Кожен із них орієнтований на роботу з візуальним контентом, однак відрізняється за

функціональним призначенням, моделлю взаємодії з користувачем і способом організації матеріалів [7; 8; 9; 10].

Instagram є одним із найпопулярніших прикладів платформи, що поєднує публікацію фото- та відеоконтенту із соціальною взаємодією. Сервіс надає можливість створювати публікації, вести особистий профіль, використовувати Stories і Direct, взаємодіяти з іншими користувачами через вподобання, коментарі та підписки. Сильною стороною цієї платформи є висока інтеграція механізмів обміну контентом і комунікації. Разом із тим Instagram значною мірою орієнтований на алгоритмічну стрічку, активну соціальну взаємодію та публічність, що не завжди є оптимальним для користувачів, яким важливіша структурованість контенту або більш контрольований доступ до нього [11].

Flickr є сервісом, який традиційно асоціюється з розміщенням, зберіганням і систематизацією фотографій. Його функціональність більшою мірою орієнтована на створення фотоальбомів, організацію зображень і взаємодію всередині тематичних спільнот. Перевагою Flickr є зручність роботи з великими колекціями фото та акцент на візуальному архівуванні матеріалів. Водночас рівень соціальної взаємодії в межах платформи є менш вираженим порівняно із сучасними соціальними мережами, а інтерфейс і модель використання більшою мірою підходять для зберігання та каталогізації, ніж для динамічної стрічки новин [8].

Google Photos реалізує інший підхід до роботи з фото-контентом. Основне призначення цього сервісу полягає у зберіганні, резервному копіюванні, синхронізації та впорядкуванні фотографій користувача. Платформа зручна для персонального використання, оскільки забезпечує доступ до зображень із різних пристроїв, дозволяє організовувати медіафайли та ділитися ними. Проте Google Photos меншою мірою орієнтований на соціальну складову, публічну взаємодію між користувачами та формування стрічки новин, що обмежує його відповідність концепції соціального вебзастосунку для обміну фото-контентом [9].

Pinterest також належить до платформ, пов’язаних із візуальним контентом, однак його головна функція полягає не стільки у класичному фотообміні, скільки у візуальному пошуку, збереженні та систематизації ідей. Користувачі можуть створювати тематичні добірки, зберігати зображення та використовувати платформу як інструмент навігації за інтересами. Перевагою Pinterest є зручна візуальна класифікація контенту та орієнтація на пошук за темами, проте сервіс не зосереджується на повноцінній моделі соціальної взаємодії між користувачами у тому вигляді, у якому це реалізовано на фото-соціальних платформах [10; 12].

Порівняння наведених сервісів показує, що кожен із них реалізує лише частину функціональних можливостей, які є важливими для сучасного вебзастосунку в досліджуваній предметній області. Instagram забезпечує активну соціальну взаємодію, Flickr – структуроване зберігання фотографій, Google Photos – зручне персональне керування медіаархівом, а Pinterest – візуальну категоризацію та пошук контенту [7; 8; 9; 10]. Водночас жоден із розглянутих засобів не є універсальним прикладом системи, яка однаково збалансовано поєднує публікацію фото, багатозображувальні пости, теги, підписки, стрічку новин, реакції та контроль доступу до профілю в межах спрощеної, чітко визначеної архітектури навчального вебзастосунку.

Таблиця 1.1

Порівняльний аналіз існуючих засобів для публікації та обміну фото-контентом

Основне призначення	Переваги	Недоліки
Соцмережа для фото	Підписки, вподобання, активна взаємодія	Залежність від алгоритмічної стрічки
Фотохостинг	Альбоми, архівування, фотоколекції	Слабша соціальна складова
Хмарне фото-сховище	Синхронізація, резервне копіювання	Обмежена соціальна взаємодія
Візуальний пошук ідей	Тематична класифікація, навігація	Не є класичною фото-соцмережею

Отже, аналіз існуючих засобів для публікації та обміну фото-контентом свідчить про наявність значної кількості платформ із розвиненими можливостями роботи із зображеннями, однак кожна з них реалізує власну модель використання та робить акцент на окремих функціональних аспектах. Це дає підстави для розробки власного вебзастосунку, який поєднуватиме основні переваги сучасних сервісів: реєстрацію користувачів, ведення профілю, створення публікацій із кількома зображеннями, додавання тегів, стрічку новин, систему підписок і механізм вподобань, а також обмеження доступу для неавторизованих користувачів.

1.3 Вибір технологічного стеку та архітектури вебзастосунку

Вибір технологічного стеку для вебзастосунку, призначеного для публікації та обміну фото-контентом, має ґрунтуватися на вимогах до функціональності, безпеки, зручності розробки та можливості подальшого розширення системи. Оскільки розроблюваний застосунок передбачає реєстрацію користувачів, автентифікацію, завантаження зображень, створення профілів, формування стрічки публікацій, використання тегів, вподобань і підписок, доцільно обрати стек технологій, який забезпечує як швидку реалізацію базових можливостей, так і підтримку складнішої бізнес-логіки. Для такого типу систем доцільним є використання мови програмування Python і фреймворку Django, який надає вбудовані засоби для роботи з моделями даних, шаблонами, формами, маршрутизацією, автентифікацією, адміністративною панеллю та безпекою вебзастосунку [13].

Використання Django є доцільним також з огляду на його архітектурну модель. У межах цього фреймворку застосовується підхід MTV, у якому логіка обробки запитів, робота з даними та формування інтерфейсу розділяються між окремими компонентами. Такий підхід спрощує підтримку коду, полегшує декомпозицію системи на модулі та дозволяє організувати проєкт у вигляді окремих застосунків, наприклад для користувачів, публікацій, підписок,

вподобань і тегів. Для вебзастосунку соціального типу це є важливою перевагою, оскільки функціональність системи зростає поступово, а модульна структура дає змогу розширювати її без істотного порушення вже реалізованих компонентів [13].

Для взаємодії з серверною частиною доцільно використати класичну клієнт-серверну архітектуру. У такій моделі браузер користувача виступає клієнтом, який надсилає запити та отримує HTML-сторінки, стилі, скрипти або структуровані дані, а серверна частина виконує обробку запитів, взаємодіє з базою даних, керує автентифікацією та доступом до ресурсів. Для частини функціональності, пов’язаної з асинхронною взаємодією, наприклад завантаженням даних без повного перезавантаження сторінки, обробкою реакцій користувача або окремими динамічними елементами інтерфейсу, доцільним є використання JavaScript. У разі потреби побудови API або реалізації окремих програмних інтерфейсів може бути застосований Django REST framework, який надає засоби для серіалізації даних, побудови класових представлень і маршрутизації API-ресурсів [16]. Використання OAuth-підходів у такому випадку є доцільним також для подальшої інтеграції з зовнішніми сервісами та програмними інтерфейсами, що підтверджується документацією GitHub щодо автентифікації через OAuth-додатки для доступу до API [38].

Для зберігання даних у вебзастосунку такого типу доцільно використовувати реляційну систему керування базами даних. Оскільки застосунок передбачає наявність пов’язаних сутностей, зокрема користувачів, профілів, публікацій, зображень, тегів, підписок і вподобань, важливими є надійність зберігання, підтримка зв’язків між таблицями та цілісність даних. Як основне рішення доцільно розглядати PostgreSQL, оскільки ця система керування базами даних характеризується надійністю, розширюваністю та підтримкою широкого набору можливостей для роботи зі структурованими даними. Водночас на етапі локальної розробки та початкового тестування може використовуватися і вбудована база SQLite, яка спрощує старт проєкту [17; 18].

Окрему увагу в межах обраного стеку слід приділити роботі з медіафайлами, оскільки саме фото є центральним типом контенту в системі. Для реалізації завантаження зображень у Django доцільно використовувати ImageField, який дозволяє зберігати шляхи до файлів у базі даних і працювати з ними через файлове сховище. Для перевірки та обробки зображень застосовується бібліотека Pillow, яка використовується разом із ImageField. Такий підхід дає змогу реалізувати зберігання аватарів користувачів, фото в публікаціях і, за потреби, попередню обробку або валідацію графічних файлів [15; 19; 20]. Оскільки користувач завантажує контент самостійно, важливими також є перевірка введених даних, обмеження типів файлів і загальні механізми захисту від некоректного або небезпечного вмісту [5].

Для реалізації користувацького інтерфейсу доцільно застосувати HTML, CSS, JavaScript і фреймворк Bootstrap. Використання Bootstrap є виправданим завдяки наявності адаптивної сітки, готових компонентів і допоміжних класів, що прискорює побудову інтерфейсу та забезпечує коректне відображення сторінок на різних типах пристроїв. Для вебзастосунку, орієнтованого на публікацію фото-контенту, адаптивність має особливе значення, оскільки користувачі часто працюють із такими системами як із персональних комп’ютерів, так і з мобільних пристроїв [21; 22; 23].

Для підсистеми користувачів доцільно використати вбудовані механізми Django або власну розширену модель користувача. Це є важливим для системи, у якій користувач має не лише стандартні облікові дані, а й додаткові атрибути, зокрема повне ім’я, біографію, аватар та інші елементи профілю. Крім того, функціональність застосунку передбачає підтвердження реєстрації через електронну пошту, тому логічним є використання засобів Django для надсилання електронних повідомлень. Якщо ж у майбутньому виникне потреба у вході через зовнішні сервіси, наприклад Google або GitHub, для цього можуть бути інтегровані спеціалізовані модулі соціальної автентифікації, зокрема social-auth-app-django [24; 25; 26]. Для вебзастосунків такого типу важливим є використання стандартизованих протоколів авторизації, зокрема OAuth 2.0, що широко

застосовується для реалізації входу через зовнішні сервіси [39]. Документація Google також підтверджує доцільність використання OAuth 2.0 у вебзастосунках, де необхідно забезпечити безпечний вхід користувача через зовнішній обліковий запис і контроль доступу до пов’язаних ресурсів [40].

Отже, для розробки вебзастосунку для публікації та обміну фото-контентом доцільно обрати стек, до якого входять Python, Django, HTML, CSS, JavaScript, Bootstrap і реляційна база даних, насамперед PostgreSQL, із можливістю використання SQLite на локальному етапі розробки. Найбільш доцільною архітектурою для такого рішення є клієнт-серверна архітектура з модульною організацією проєкту та використанням підходу MTV. Обраний стек забезпечує достатню гнучкість, зручність реалізації основного функціоналу, підтримку роботи з медіафайлами, механізми автентифікації, безпеки та можливість подальшого розширення системи [13; 17; 21; 24].

Висновки до розділу 1

У першому розділі було розглянуто теоретичні аспекти розробки вебзастосунку для публікації та обміну фото-контентом. Проведений аналіз сучасних тенденцій у цій галузі показав, що фото-контент посідає важливе місце у цифровій комунікації та є основою багатьох сучасних онлайн-платформ. Встановлено, що розвиток вебзастосунків такого типу відбувається у напрямі посилення соціальної взаємодії, персоналізації контенту, ускладнення структури публікацій, підвищення вимог до безпеки, конфіденційності та продуктивності систем.

У процесі аналізу існуючих засобів для публікації та обміну фото-контентом було визначено, що сучасні платформи, зокрема Instagram, Flickr, Google Photos та Pinterest, реалізують різні підходи до роботи з візуальним контентом. Кожен із розглянутих сервісів має власні переваги та функціональні особливості, однак жоден із них не є повністю універсальним рішенням у контексті поставленого завдання. Це підтвердило доцільність розробки власного

вебзастосунку, який поєднуватиме основні можливості сучасних платформ у межах єдиної системи.

Також було обґрунтовано вибір технологічного стеку та архітектури майбутнього вебзастосунку. Визначено, що для реалізації поставлених функціональних вимог доцільно використати мову програмування Python, фреймворк Django, клієнтські технології HTML, CSS, JavaScript і Bootstrap, а також реляційну базу даних із використанням SQLite на етапі локальної розробки та можливістю подальшого переходу на PostgreSQL. Встановлено, що найбільш придатною для такого рішення є клієнт-серверна архітектура з модульною організацією проєкту на основі підходу MTV.

Отже, результати першого розділу дали змогу визначити особливості предметної області, проаналізувати існуючі аналоги та обґрунтувати технологічну основу розробки вебзастосунку для публікації та обміну фото-контентом. Одержані висновки є підґрунтям для подальшого проєктування структури системи, моделювання її компонентів і практичної реалізації програмного продукту.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ПУБЛІКАЦІЇ ТА ОБМІНУ ФОТО-КОНТЕНТОМ

2.1 Проєктування інформаційної моделі та бази даних вебзастосунку

Проєктування інформаційної моделі є одним із ключових етапів розробки вебзастосунку для публікації та обміну фото-контентом, оскільки саме на цьому етапі визначаються основні сутності системи, зв’язки між ними та структура зберігання даних.

Для застосунку типу DjangoGramm коректно спроєктована модель даних має забезпечувати реалізацію реєстрації користувачів, підтвердження електронної пошти, ведення профілю, створення публікацій із кількома зображеннями, додавання тегів, взаємодію через вподобання та механізм підписок між користувачами (див. рис. 2.1).

Для зберігання даних у проєкті обрано реляційний підхід, оскільки він є найбільш доцільним для системи, де між об’єктами існує значна кількість логічних зв’язків. У вебзастосунку наявні як прості зв’язки типу «один до одного» та «один до багатьох», так і складніші зв’язки типу «багато до багатьох». Використання реляційної бази даних дозволяє забезпечити цілісність даних, уникнути дублювання інформації та підтримувати коректну взаємодію між окремими компонентами системи [27; 28].

Центральною сутністю інформаційної моделі є користувач. У проєкті використовується розширена модель користувача, побудована на основі AbstractUser, у якій як основний ідентифікатор використовується адреса електронної пошти замість стандартного імені користувача. Таке рішення є логічним для застосунку, у якому реєстрація та подальша автентифікація орієнтовані саме на email. Це також спрощує реалізацію механізму підтвердження облікового запису через електронний лист [27].

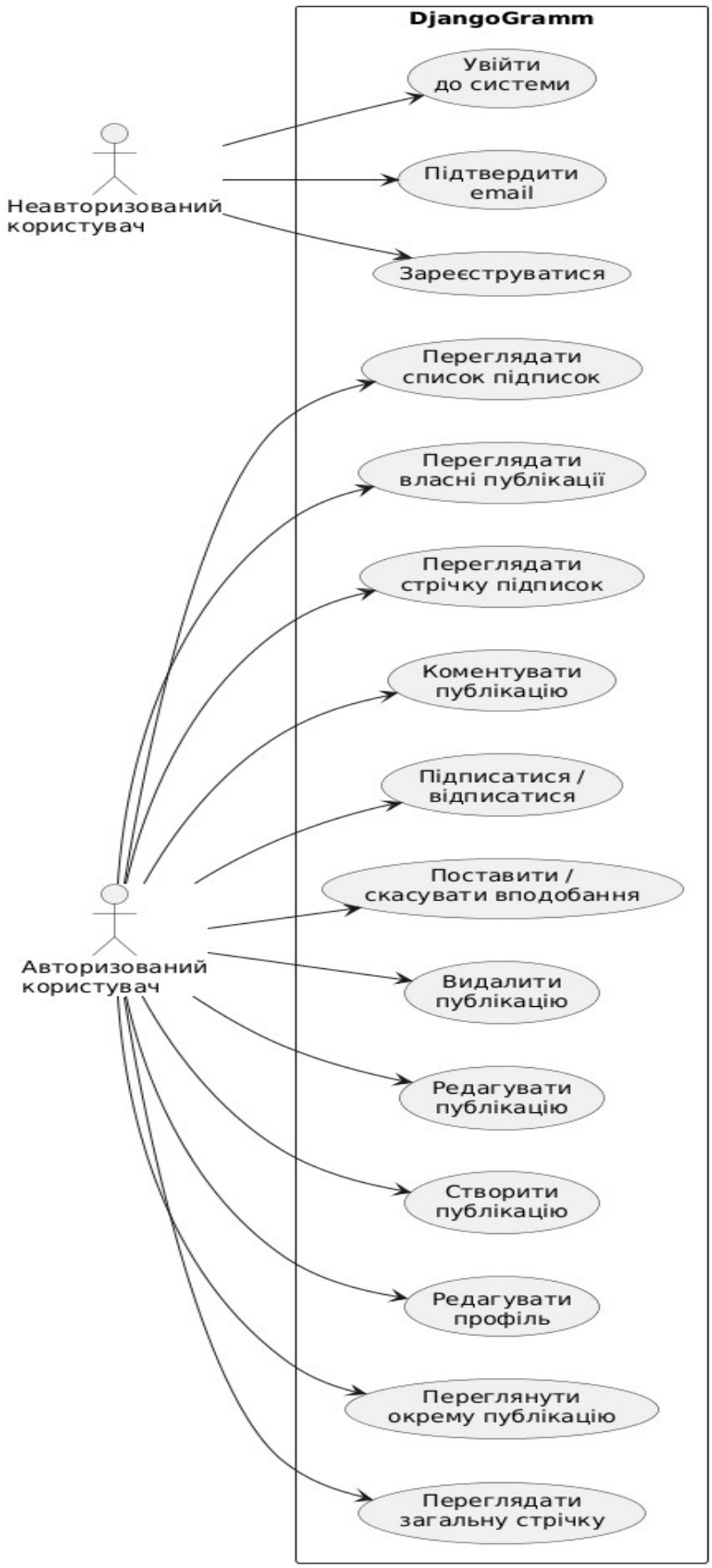

Рис. 2.1. UML-діаграма прецедентів вебзастосунку DjangoGramm

(розроблено автором)

Окремою сутністю, пов’язаною з користувачем, є профіль користувача. Профіль містить додаткові персональні дані, які не входять до базового набору облікових атрибутів, а саме повне ім’я, біографію та аватар. Між сутностями користувача і профілю реалізовано зв’язок типу «один до одного», що дозволяє розділити автентифікаційні дані та інформацію, призначену для відображення у вебінтерфейсі. Такий підхід є зручним як з точки зору структури бази даних, так і з точки зору подальшого розширення функціональності облікового запису [28].

Для реалізації підтвердження електронної пошти в інформаційну модель введено окрему сутність токена підтвердження. Вона пов’язана з користувачем через зв’язок «один до багатьох», оскільки для одного користувача в різні моменти часу можуть створюватися кілька токенів підтвердження. Ця сутність містить хеш токена, дату завершення його дії, час використання та дату створення. Така структура дозволяє забезпечити безпечне підтвердження облікового запису, контролювати строк дії посилання та запобігати повторному використанню одного й того самого токена. Загальна структура основних сутностей та зв’язків між ними наведена на рис. 2.2

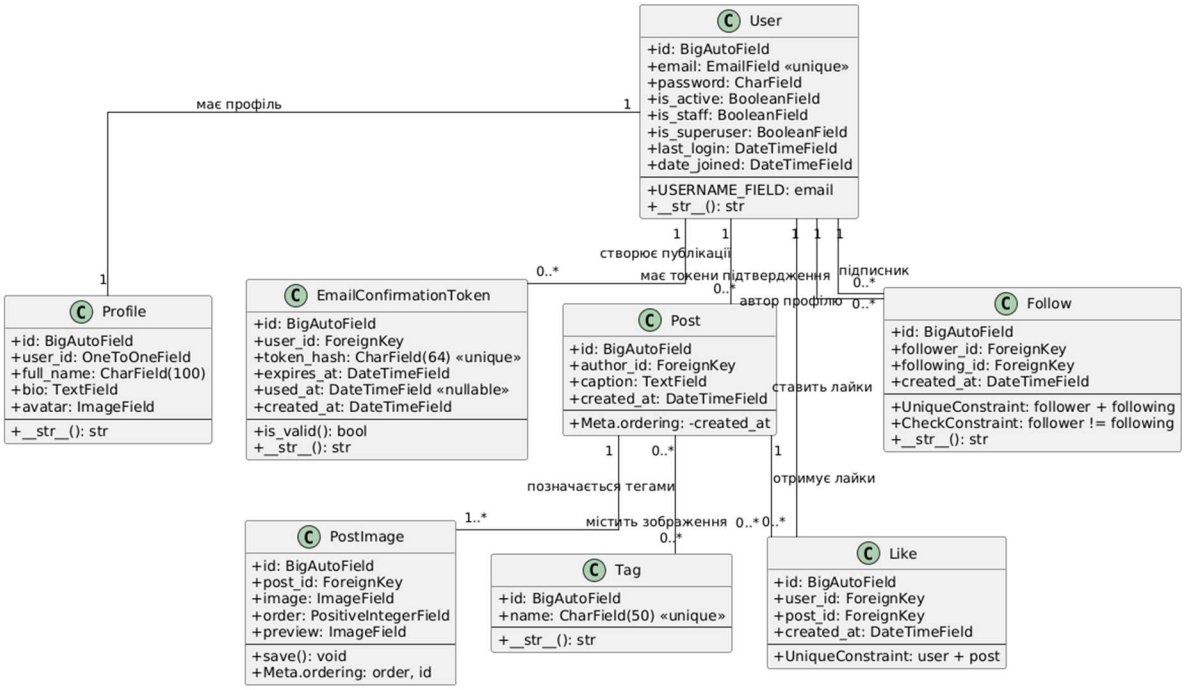

Рис. 2.2. ER-діаграма бази даних вебзастосунку DjangoGramm

(розроблено автором)

Основною сутністю, що представляє контент у системі, є публікація. Кожна публікація пов’язана з автором, містить текстовий опис та дату створення. З користувачем публікація зв’язана відношенням «один до багатьох», оскільки один користувач може створити багато постів, але кожен пост має лише одного автора. Саме сутність публікації є центральною для подальшої взаємодії з фото, тегами, вподобаннями та стрічкою новин.

Оскільки за умовами проєкту один пост може містити кілька зображень, у моделі даних передбачено окрему сутність для зберігання зображень публікації. Вона пов’язана з постом зв’язком «один до багатьох». Кожен об’єкт такого типу містить шлях до основного зображення, порядковий номер відображення та шлях до попереднього перегляду. Винесення зображень в окрему таблицю є принципово важливим рішенням, оскільки воно забезпечує підтримку багатозображувальних публікацій, дозволяє керувати порядком фото в межах поста та спрощує подальшу обробку файлів [29].

Для реалізації тематичної класифікації контенту в системі використовується сутність тегу. Вона містить унікальну назву тегу та пов’язана з публікаціями зв’язком «багато до багатьох». Це означає, що одна публікація може мати кілька тегів, а один і той самий тег може використовуватися в багатьох публікаціях. Такий підхід є доцільним, оскільки забезпечує гнучкість категоризації контенту, підтримує повторне використання тегів і спрощує пошук або групування постів за тематикою.

Для відображення соціальної взаємодії між користувачами в інформаційній моделі передбачено сутність вподобання. Вона пов’язує користувача та публікацію, фіксуючи факт поставленої реакції та час її створення. Це проміжна таблиця між користувачами та постами, яка реалізує зв’язок «багато до багатьох» із додатковими атрибутами. У моделі також передбачено обмеження унікальності, що не дозволяє одному користувачу поставити вподобання одній і тій самій публікації більше одного разу. Це забезпечує коректність логіки лайків і спрощує реалізацію механізму скасування вподобання.

Ще однією важливою соціальною сутністю є підписка між користувачами. У цьому випадку один користувач може підписуватися на багатьох інших користувачів, а також мати багато підписників. Для реалізації такого механізму використовується окрема сутність, у якій зберігаються користувач-підписник, користувач, на якого оформлено підписку, та дата створення зв’язку. У моделі передбачено обмеження, які забороняють дублювання однакових підписок і виключають можливість підписки користувача на самого себе. Така структура є необхідною для формування персоналізованої стрічки новин і реалізації логіки соціальних зв’язків усередині системи.

Узагальнюючи спроєктовану інформаційну модель, можна виділити такі основні сутності: користувач, профіль, токен підтвердження електронної пошти, публікація, зображення публікації, тег, вподобання та підписка. Між ними реалізовано сукупність зв’язків, що повністю відповідає функціональним вимогам проєкту. Зокрема, модель підтримує реєстрацію та підтвердження облікового запису, редагування профілю, створення постів із кількома зображеннями, роботу з тегами, механізми вподобань та підписок, а також формування персоналізованої стрічки.

Отже, спроєктована інформаційна модель та структура бази даних забезпечують цілісне представлення предметної області вебзастосунку для публікації та обміну фото-контентом. Вона є достатньо гнучкою для реалізації основного функціоналу системи, підтримує розширення в майбутньому та створює надійну основу для подальшої програмної реалізації серверної логіки й користувацького інтерфейсу.

2.2 Опис основних алгоритмів та методів обробки та збереження даних

У межах вебзастосунку DjangoGramm реалізовано сукупність алгоритмів (рис. 2.2.), що забезпечують реєстрацію користувачів, підтвердження електронної пошти, створення та редагування профілю, публікацію постів, обробку тегів, механізми вподобань і підписок, а також формування стрічки

новин. Якщо в попередньому підпункті було розглянуто інформаційну модель системи, то в даному підпункті увагу зосереджено саме на логіці виконання основних процесів, тобто на послідовності дій, які виконує система під час взаємодії з користувачем.

Одним із базових у застосунку є алгоритм реєстрації користувача. На цьому етапі користувач заповнює форму, вказуючи адресу електронної пошти та пароль. Після надсилання форми система перевіряє коректність введених даних: валідність адреси електронної пошти, унікальність облікового запису, збіг паролів і відповідність пароля встановленим правилам безпеки [30; 31]. Якщо перевірка проходить успішно, створюється новий обліковий запис, однак до моменту підтвердження електронної пошти він залишається неактивним. Такий підхід підвищує безпеку системи та зменшує ризик появи помилкових або фіктивних реєстрацій.

Після реєстрації виконується алгоритм підтвердження електронної пошти. Для цього система генерує унікальний токен, обчислює його хеш і формує посилання, яке надсилається користувачу листом. Далі користувач переходить за отриманим посиланням, після чого система перевіряє, чи існує відповідний токен, чи не був він використаний раніше та чи не завершився строк його дії [31]. Якщо всі умови виконано, обліковий запис активується, а токен позначається як використаний. У протилежному випадку система повідомляє про помилку підтвердження. Такий алгоритм дозволяє реалізувати безпечне підтвердження реєстрації без збереження токена у відкритому вигляді.

Наступним важливим процесом є алгоритм створення та редагування профілю користувача. Після активації облікового запису користувач переходить до заповнення профілю, де може вказати повне ім’я, короткий опис і завантажити аватар. Система приймає введені дані, виконує їх перевірку та зберігає у відповідній структурі профілю. Якщо профіль для цього користувача ще не був створений, система формує його автоматично. Такий алгоритм дозволяє відокремити облікові дані від додаткової інформації профілю та забезпечує подальше відображення персональних відомостей у стрічках і публікаціях.

Ключовим для даного вебзастосунку є алгоритм створення публікації. Спочатку користувач вводить текстовий опис, за потреби додає теги та завантажує одне або кілька зображень. Після надсилання форми система виконує перевірку кількості завантажених файлів і правильності введених даних. Якщо кількість зображень не відповідає встановленим обмеженням або форма містить помилки, користувач отримує відповідне повідомлення. Якщо ж усі умови виконано, система послідовно створює запис публікації, обробляє теги та додає пов’язані зображення із зазначенням їх порядку. Збереження виконується як єдиний узгоджений процес, що дає змогу уникнути часткового запису даних у разі помилки на одному з етапів [32; 33] Послідовність взаємодії користувача, вебінтерфейсу, серверної частини, бази даних та файлового сховища під час створення публікації наведено на рис. 2.3, а деталізовану блок-схему алгоритму створення публікації з кількома зображеннями і тегами – на рис. 2.4.

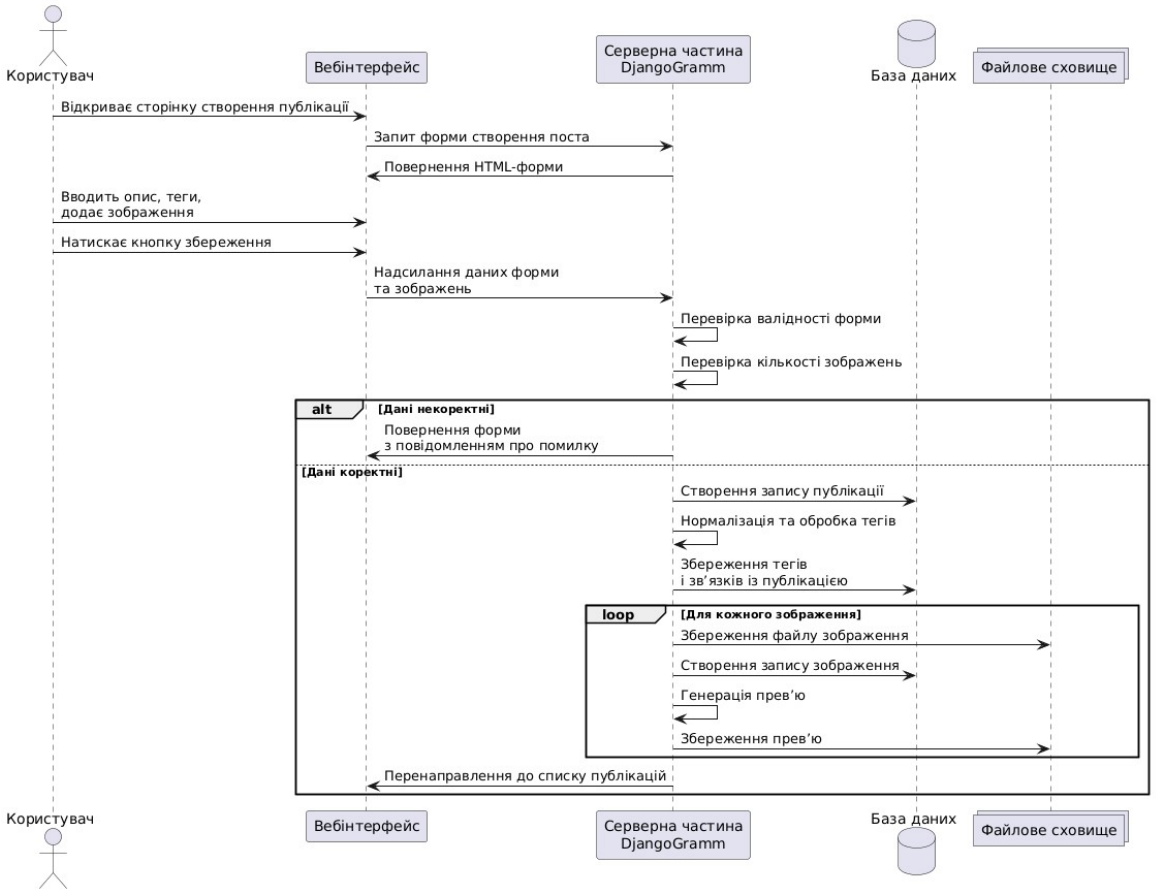

Рис. 2.3. Діаграма послідовності прецедента “Створення публікації”

(розроблено автором)

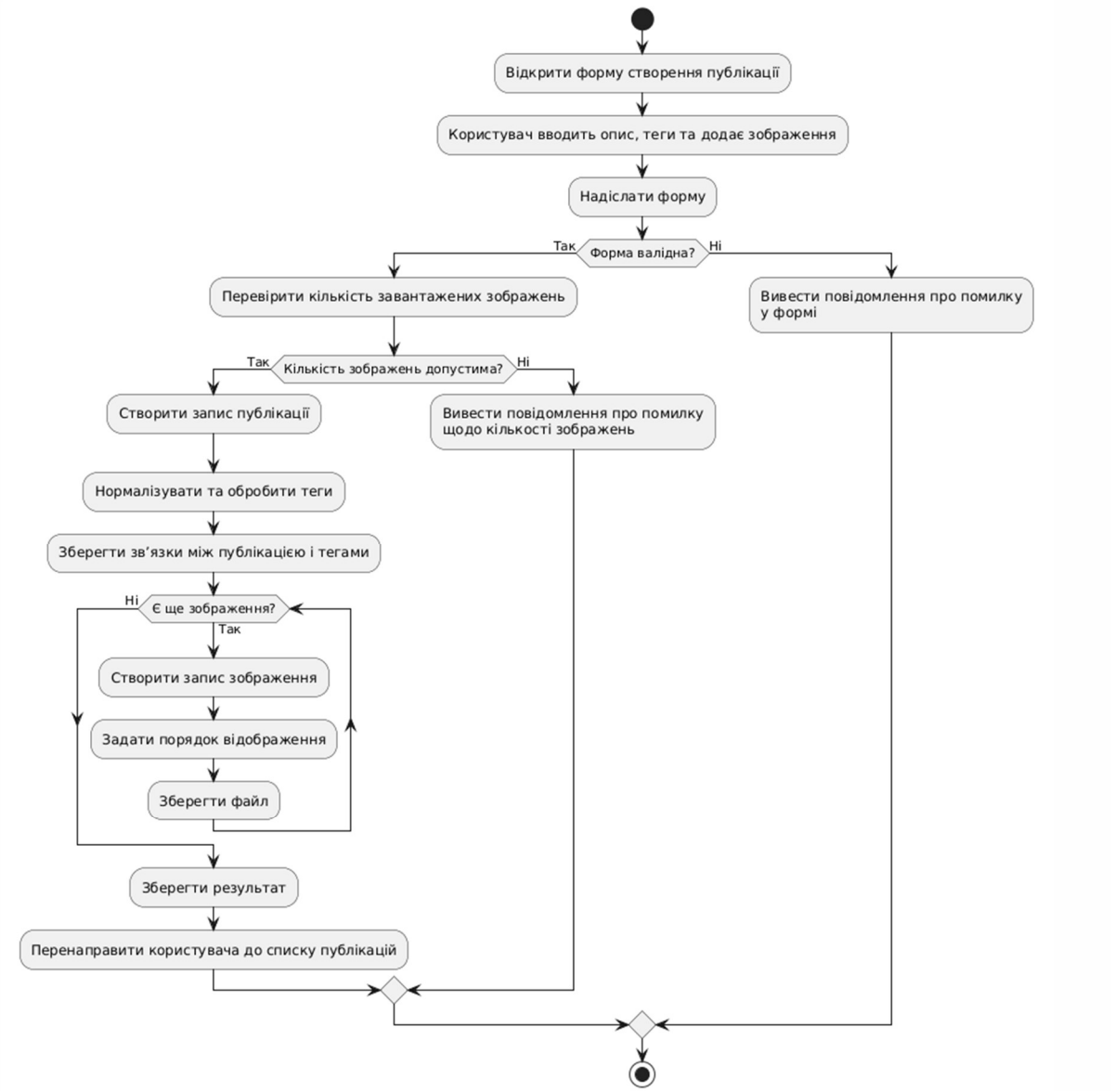

Рис. 2.4. Блок-схема алгоритму створення публікації з кількома зображеннями і тегами

(розроблено автором)

Окремий алгоритм пов'язаний з обробкою тегів. Користувач вводить теги як текстовий список, розділений комами. Після цього система нормалізує отриманий рядок: вилучає зайві пробіли, прибирає порожні значення, уніфікує роздільники та виключає дублікати. На основі отриманого списку система або знаходить уже наявні теги, або формує нові, після чого встановлює зв'язок між тегами та публікацією. Такий підхід забезпечує повторне використання тегів, полегшує категоризацію постів і спрощує подальший пошук контенту.

Важливим алгоритмом є обробка зображень публікації. Після завантаження фотографії система зберігає основний файл і, за потреби, автоматично формує його зменшену копію для попереднього перегляду. Під час цього процесу зображення відкривається, за необхідності перетворюється до потрібного кольорового формату, масштабується до визначеного розміру та зберігається в оптимізованому вигляді [29; 34]. Такий метод знижує навантаження на інтерфейс і пришвидшує відображення списків публікацій, оскільки для попереднього перегляду використовуються менші за розміром файли.

Подібна логіка застосовується і в алгоритмі редагування публікації. У цьому випадку система приймає оновлені текстові дані, змінений набір тегів та зміни у складі зображень. Після перевірки коректності введеної інформації система оновлює текстову частину поста, переобчислює набір тегів і зберігає змінений набір вкладених зображень. Завдяки цьому забезпечується узгодженість між описом, тегами та медіафайлами після внесення змін.

Алгоритм обробки вподобань реалізовано за принципом перемикання стану. Коли авторизований користувач взаємодіє з кнопкою вподобання, система перевіряє, чи існує вже відповідний зв’язок між користувачем і публікацією. Якщо такого зв’язку немає, він створюється, а публікація вважається вподобаною. Якщо зв’язок уже існує, він видаляється, тобто виконується скасування вподобання. У результаті система оновлює стан реакції та кількість отриманих вподобань. Такий алгоритм є простим і ефективним для реалізації механізму like/unlike.

Схожим чином працює алгоритм підписки на користувачів. Під час спроби оформити підписку система спочатку перевіряє, чи не намагається користувач підписатися на самого себе. Якщо така ситуація виникає, операція скасовується. Якщо ж підписка між двома користувачами ще не існує, вона створюється; якщо існує, вона видаляється. У такий спосіб один і той самий механізм забезпечує як підписку, так і відписку, а також підтримує коректність зв’язків у системі.

Ще одним важливим алгоритмом є формування загальної та персоналізованої стрічки новин. Для загального списку система вибирає всі наявні публікації, впорядковує їх за датою створення та передає для відображення у вигляді сторінок із пагінацією. Для персоналізованої стрічки додатково враховуються підписки поточного користувача: до вибірки включаються лише публікації тих авторів, на яких оформлено підписку. Крім того, система визначає поточний стан взаємодії користувача з контентом, зокрема перелік уже вподобаних постів та авторів, на яких він підписаний. Використання пагінації дозволяє зменшити навантаження на інтерфейс і зробити навігацію зручнішою [35].

Методи збереження даних у межах розглянутих алгоритмів базуються на використанні об’єктної моделі доступу до даних, що дає змогу працювати з сутностями системи без безпосереднього написання SQL-запитів. Для підтримання цілісності даних використовуються перевірки унікальності, контроль зв’язків між об’єктами та узгоджене виконання багатокрокових операцій [32; 33]. Це забезпечує надійну обробку, збереження та оновлення інформації в межах вебзастосунку.

Отже, у вебзастосунку DjangoGramm реалізовано основні алгоритми, необхідні для підтримки реєстрації та підтвердження користувачів, створення й редагування профілів, публікації постів із кількома зображеннями, роботи з тегами, механізмів вподобань і підписок, а також формування загальної та персоналізованої стрічки. Використані алгоритми та методи збереження даних забезпечують цілісність інформації, зручність користування системою та можливість подальшого розширення функціональності вебзастосунку.

2.3 Реалізація серверної та клієнтської частини вебзастосунку

Реалізація вебзастосунку DjangoGramm виконана на основі поділу системи на серверну та клієнтську частини, які взаємодіють у межах класичної клієнт-серверної архітектури. Серверна частина відповідає за обробку запитів, роботу з

базою даних, автентифікацію користувачів, бізнес-логіку та керування медіаконтентом, тоді як клієнтська частина забезпечує відображення сторінок, взаємодію користувача з інтерфейсом і подання результатів обробки даних у зручному вигляді. Основу проєкту становить Django-проєкт djangogramm, у межах якого реалізовано окремі застосунки accounts і posts, що відповідають за підсистему користувачів та підсистему публікацій відповідно [13; 21].

Серверна частина вебзастосунку організована за модульним принципом. У застосунку accounts реалізовано функціональність реєстрації (рис. 2.5.), підтвердження електронної пошти, входу до системи, виходу з неї та редагування профілю користувача. Для цього використовуються модель користувача, модель профілю та модель токена підтвердження, форми для реєстрації й заповнення профілю, а також окремий сервісний модуль, у якому зосереджено логіку генерації токенів і підтвердження email [27; 30; 31].

```
class UserRegistrationForm(forms.Form):  3 usages  Valeriia
    email = forms.EmailField(label="Email")
    password1 = forms.CharField(label="Password", widget=forms.PasswordInput)
    password2 = forms.CharField(label="Repeat password", widget=forms.PasswordInput)

    def clean_email(self):  Valeriia
        email = self.cleaned_data["email"].strip()
        if User.objects.filter(email__iexact=email).exists():
            raise ValidationError("This email is already registered.")
        return email

    def clean_password1(self):  Valeriia
        password1 = self.cleaned_data.get("password1")
        if password1:
            validate_password(password1)
        return password1

    def clean(self):  Valeriia
        cleaned = super().clean()
        p1 = cleaned.get("password1")
        p2 = cleaned.get("password2")
        if p1 and p2 and p1 != p2:
            self.add_error( field: "password2",  error: "Passwords do not match.")
        return cleaned

    def save(self):  Valeriia
        email = self.cleaned_data["email"].strip()
        password = self.cleaned_data["password1"]
        return User.objects.create_user(email=email, password=password, is_active=False)
```

Рис. 2.5. Скрипт, що відповідає за реєстрацію користувача

(розроблено автором)

Функціональність автентифікації реалізовано як для стандартного входу за електронною поштою або ім’ям користувача, так і для входу через зовнішні сервіси. Для цього в проєкті підключено модуль social_django, що забезпечує підтримку авторизації через GitHub і Google [24; 25; 26]. Такий підхід підвищує зручність користування системою та розширює можливості доступу до неї. Коректна робота такого механізму потребує також налаштування клієнтів OAuth, дозволених URI перенаправлення та параметрів взаємодії із зовнішнім постачальником автентифікації [41].

У модулі posts реалізовано основну функціональність роботи з публікаціями. До неї належать перегляд усіх постів, формування персоналізованої стрічки підписок, перегляд окремої публікації, створення та редагування постів, видалення, постановка і скасування вподобань, а також підписка на інших користувачів і відписка від них. Для цього в модулі представлень реалізовано відповідні функції обробки запитів, а доступ до більшості з них обмежується декоратором login_required, що відповідає функціональній вимозі про недоступність основного контенту для неавторизованих користувачів [31].

Маршрутизація в проєкті побудована через окремі URL-конфігурації для застосунків accounts і posts, які підключаються у головному файлі маршрутизації проєкту. У системі виділено маршрути для входу, виходу, реєстрації, підтвердження електронної пошти (рис. 2.6.), редагування профілю, перегляду списку постів, створення нової публікації, детального перегляду поста, редагування, видалення, вподобання та підписки. У режимі локальної розробки також налаштовано обслуговування медіафайлів, що забезпечує коректну роботу з аватарами та зображеннями публікацій.

Для роботи з публікаціями на серверному рівні використано форми для створення та редагування постів, а для вкладених зображень – пов’язаний набір форм. Такий підхід дозволяє поєднати обробку текстового опису, тегів і кількох зображень у межах єдиної серверної логіки [29; 30]. Окремо реалізовано механізм опрацювання рядка тегів, який передбачає нормалізацію введення,

усунення дублікатів і формування списку унікальних значень. Це забезпечує гнучку роботу з багатозображувальними публікаціями та тематичною категоризацією контенту.

```python
def generate_token(user: User) -> str:  5 usages  Valeriia
    raw = secrets.token_urlsafe(32)
    h = token_hash(raw)
    expires = timezone.now() + timedelta(hours=24)
    EmailConfirmationToken.objects.filter(user=user, used_at__isnull=True).update(used_at = timezone.now())
    EmailConfirmationToken.objects.create(user=user, token_hash=h, expires_at=expires)
    return raw

def token_hash(raw: str) -> str:  2 usages  Valeriia
    return hashlib.sha256(raw.encode('utf-8')).hexdigest()

@transaction.atomic  2 usages  Valeriia
def confirm_email(raw_token: str, request) -> bool:
    h = token_hash(raw_token)
    token = (EmailConfirmationToken.objects.select_related('user').filter(token_hash=h).first())

    if token is None:
        return False
    if not token.is_valid():
        return False

    user = token.user
    user.is_active = True
    user.save(update_fields=['is_active'])

    token.used_at = timezone.now()
    token.save(update_fields=['used_at'])
    login(request, user, backend="django.contrib.auth.backends.ModelBackend")
    return True
```

Рис. 2.6. Фрагмент коду, що відповідає за генерацію та підтвердження email-токена

(розроблено автором)

У серверній частині проєкту реалізовано також обробку асинхронної взаємодії. У механізмах вподобань і підписок система може повертати або звичайне перенаправлення, або структуровану відповідь для динамічного оновлення інтерфейсу. Крім того, для оптимізації доступу до бази даних у вибірках використовуються засоби попереднього підвантаження пов’язаних об’єктів, а для розбиття списків публікацій застосовується пагінація [33; 35]. Це дозволяє підвищити продуктивність системи та покращити користувацький досвід під час роботи зі стрічкою публікацій.

Клієнтська частина вебзастосунку побудована на основі шаблонів Django з використанням HTML, Bootstrap 5 і JavaScript. Базовий шаблон base.html визначає загальну структуру сторінок, містить навігаційну панель, блок відображення системних повідомлень і підключення клієнтського JavaScript-бандлу [21; 22; 23]. Навігація змінюється залежно від стану автентифікації користувача: для авторизованих користувачів відображаються посилання на стрічку, список постів, створення нової публікації, підписки та профіль, а для неавторизованих – кнопки входу та реєстрації. Така організація забезпечує єдиний стиль інтерфейсу в межах усього застосунку.

Сторінка створення поста реалізована у шаблоні post_form.html. Вона містить поля для введення текстового опису, списку тегів (рис. 2.7) та елемент для завантаження кількох зображень одночасно. У шаблоні також передбачено відображення повідомлень про помилки, підказки щодо введення тегів і допоміжні елементи інтерфейсу. Використання стилів Bootstrap забезпечує зручне сприйняття форми та коректне відображення елементів на різних типах пристроїв [22; 23].

```
if form.is_valid() and files_ok:
    try:
        with transaction.atomic():
            post = form.save(commit=False)
            post.author = request.user
            post.save()

            for name in form.parsed_tags():
                tag, _ = Tag.objects.get_or_create(name=name)
                post.tags.add(tag)

            for idx, f in enumerate(files, start=1):
                PostImage.objects.create(post=post, image=f, order=idx)

        return redirect("posts:list")
```

Рис. 2.7. Фрагмент коду, який відповідає за створення публікації з кількома зображеннями і тегами

(розроблено автором)

Сторінка списку всіх публікацій (рис. 2.8.) реалізована у шаблоні post_list.html. Для кожного поста відображаються автор, дата створення, текстовий опис, набір зображень, теги та кнопки дій. Якщо автором поста є поточний користувач, йому доступні операції відкриття, редагування та видалення власної публікації. Для чужих постів доступні механізми вподобання та підписки на автора. У шаблоні також реалізовано пагінацію та умовне відображення кнопок залежно від поточного стану взаємодії користувача з постом [21; 23; 35].

```
@login_required  1 usage  Valeriia
def feed(request):
    following_ids = Follow.objects.filter(follower=request.user).values_list( *fields: "following_id", flat=True)

    qs = (
        Post.objects.filter(author__id__in=following_ids)
        .select_related("author")
        .prefetch_related( *lookups: "images", "tags")
        .order_by("-created_at")
    )

    page_obj = Paginator(qs,  per_page: 10).get_page(request.GET.get("page"))

    liked_post_ids = set(
        Like.objects.filter(user=request.user, post__in=page_obj.object_list)
        .values_list( *fields: "post_id", flat=True)
    )

    followed_ids = set(following_ids)

    return render(
        request,
        template_name: "posts/feed.html",
        context: {
            "page_obj": page_obj,
            "liked_post_ids": liked_post_ids,
            "followed_ids": followed_ids,
        },
    )
```

Рис. 2.8. Фрагмент коду, що відповідає за формування стрічки новин

(розроблено автором)

Для детального перегляду публікації реалізовано окремий шаблон сторінки поста, у якому відображаються зображення, текстовий опис, теги, коментарі та засоби взаємодії з контентом. Додатково реалізовано відкриття

зображень у модальному вікні, що підвищує зручність перегляду фотоконтенту без переходу на іншу сторінку. У клієнтській частині також передбачено JavaScript-логіку для асинхронного оновлення стану вподобань і підписок, що робить інтерфейс більш динамічним та наближеним до сучасних соціальних платформ.

Для роботи з обліковими записами на клієнтському рівні реалізовано окремі шаблони входу, реєстрації, підтвердження надсилання листа та редагування профілю. Вони побудовані на базі спільного шаблону, що забезпечує єдність зовнішнього вигляду та логіки навігації в межах усього вебзастосунку. Така організація інтерфейсу дозволяє зробити взаємодію користувача із системою цілісною та зрозумілою.

Загалом реалізація серверної та клієнтської частини вебзастосунку DjangoGramm побудована таким чином, щоб забезпечити цілісність системи, розподіл відповідальності між окремими модулями та зручність подальшої підтримки. Серверна частина реалізує бізнес-логіку, валідацію, обробку та збереження даних, а клієнтська частина відповідає за подання функціональних можливостей у доступному та адаптивному інтерфейсі. Такий підхід дозволив створити вебзастосунок, який відповідає визначеним функціональним вимогам і підтримує реєстрацію та підтвердження користувачів, ведення профілю, створення постів із кількома зображеннями, роботу з тегами, лайками, підписками та переглядом стрічки публікацій [13; 21; 24].

Висновки до розділу 2

У ході проєктування інформаційної моделі вебзастосунку DjangoGramm визначено основні сутності системи, а саме користувача, профіль, токен підтвердження електронної пошти, публікацію, зображення публікації, тег, вподобання та підписку. Обрана структура бази даних і зв’язки між сутностями повністю відповідають функціональним вимогам проєкту та забезпечують цілісне представлення предметної області.

У межах опису основних алгоритмів та методів обробки і збереження даних проаналізовано логіку роботи системи на ключових етапах її функціонування. Розглянуто алгоритми реєстрації користувача, підтвердження електронної пошти, створення та редагування профілю, публікації постів із кількома зображеннями, обробки тегів, постановки і скасування вподобань, підписки та формування стрічки новин. При цьому використання валідації, транзакцій, обмежень цілісності та механізмів ORM дозволяє забезпечити надійність обробки даних і коректну роботу застосунку.

Також, було охарактеризовано реалізацію серверної та клієнтської частини вебзастосунку. Зокрема, серверна частина, побудована на основі Django, забезпечує маршрутизацію, обробку запитів, автентифікацію, роботу з моделями, формами та сервісною логікою. Клієнтська частина, реалізована за допомогою шаблонів Django, HTML, CSS, JavaScript і Bootstrap, забезпечує зручну взаємодію користувача з основним функціоналом системи. Модульна організація проєкту спрощує підтримку коду та створює основу для подальшого розширення можливостей застосунку.

Отже, сформовано практичну основу вебзастосунку DjangoGramm: спроєктовано структуру даних, визначено алгоритми роботи системи та описано реалізацію її серверної і клієнтської частин. Отримані результати підтверджують працездатність обраних технічних рішень і створюють підґрунтя для подальшого функціонування вебзастосунку та його інтерфейсних елементів.

РОЗДІЛ 3
ФУНКЦІОНУВАННЯ ТА ЕЛЕМЕНТИ ІНТЕРФЕЙСУ ВЕБЗАСТОСУНКУ ДЛЯ ПУБЛІКАЦІЇ ТА ОБМІНУ ФОТО

3.1 Огляд реалізованого функціоналу розробленого вебзастосунку

У межах виконання кваліфікаційної роботи було розроблено вебзастосунок DjangoGramm, призначений для публікації та обміну фотоконтентом між зареєстрованими користувачами. Реалізована система забезпечує базовий набір функцій, характерний для сучасних соціально-орієнтованих вебплатформ, та дозволяє користувачам створювати власний контент, переглядати матеріали інших авторів, взаємодіяти з публікаціями та керувати персональним профілем.

Основою роботи застосунку є система автентифікації та реєстрації користувачів. У застосунку реалізовано створення облікового запису з використанням електронної пошти, а також механізм підтвердження email-адреси через спеціальний токен (рис.3.1.), який надсилається користувачу листом. Після підтвердження облікового запису користувач отримує доступ до внутрішнього функціоналу системи.

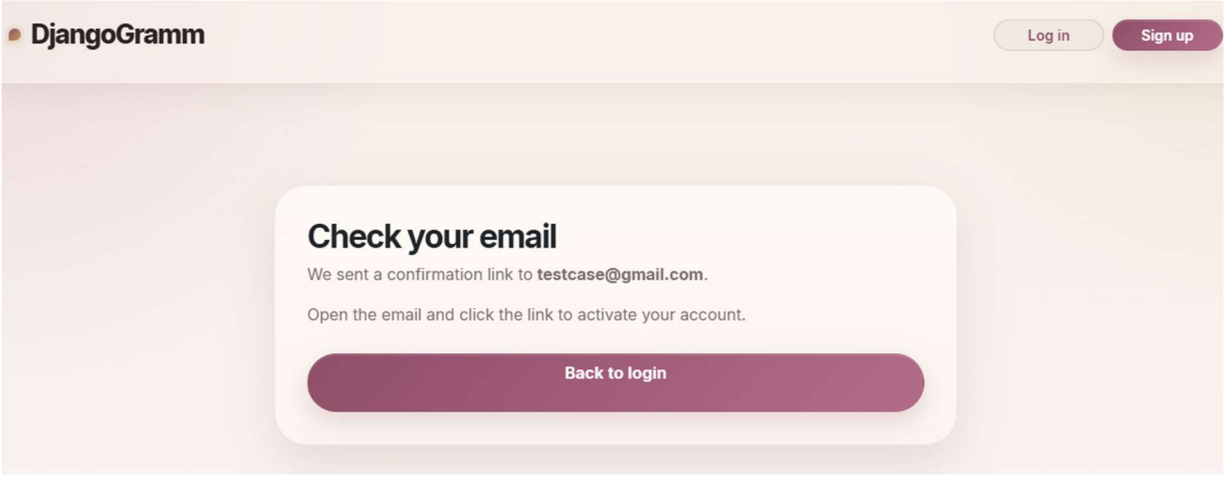

Рис. 3.1. Підтвердження email-адреси через спеціальний токен

(розроблено автором)

Окрім стандартного входу за логіном і паролем, підтримано автентифікацію за допомогою сторонніх сервісів, зокрема GitHub та Google (рис.3.2.) , що спрощує процес входу до системи. Такий підхід відповідає сучасній практиці використання OAuth-додатків для спрощення входу до вебсистем і делегування перевірки облікових даних зовнішньому сервісу [36]. Водночас реалізація такого входу потребує дотримання правил авторизації, керування дозволами та коректного обміну даними між вебзастосунком і зовнішнім сервісом [37].

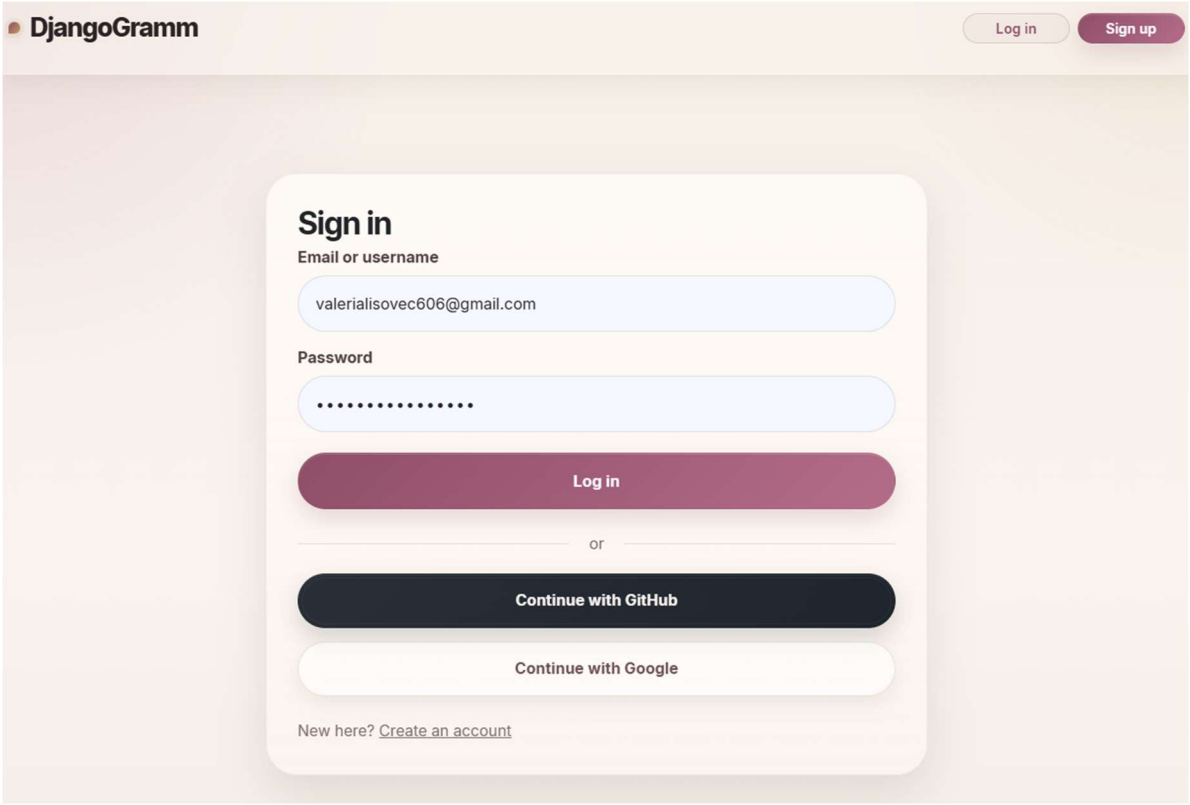

Рис. 3.2. Автентифікація за допомогою сторонніх сервісів

(розроблено автором)

Після успішної авторизації користувач отримує можливість працювати з власним профілем (рис. 3.3). У межах профілю передбачено редагування основних персональних даних, зокрема імені, короткого опису та аватару. Наявність персоналізованого профілю підвищує зручність взаємодії з платформою та дозволяє ідентифікувати автора публікацій у стрічці.

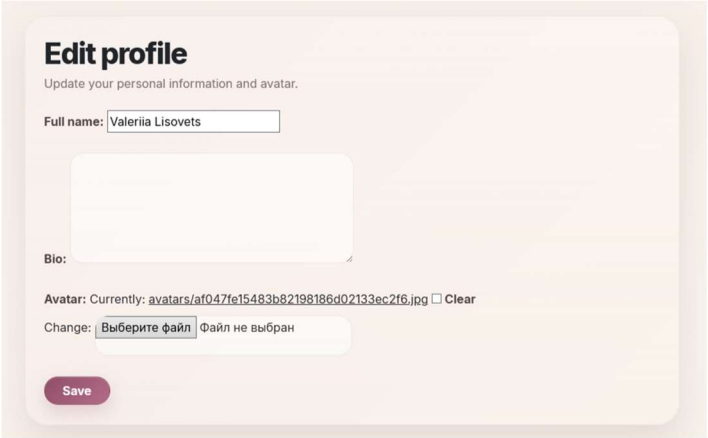

Рис. 3.3. Редагування профілю користувача

(розроблено автором)

Центральним елементом вебзастосунку є публікація постів. Користувач може створювати нові записи (рис.3.4), додаючи текстовий опис та одне або кілька зображень.

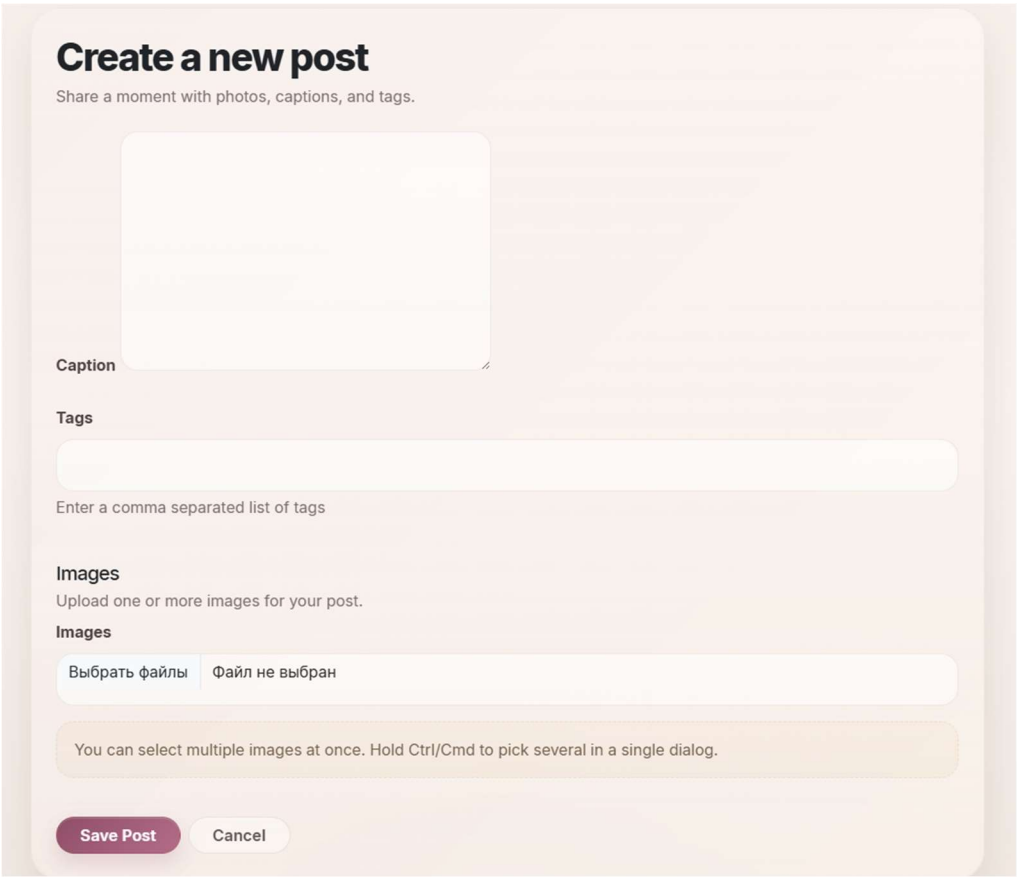

Рис. 3.4. Форма створення нової публікації

(розроблено автором)

Для публікацій підтримується робота з тегами, що дозволяє структурувати контент та підвищує зручність його подальшого перегляду. Під час завантаження зображень система зберігає файли та формує прев'ю-версії, що оптимізує відображення контенту в інтерфейсі. Також реалізовано можливість редагування створених постів і видалення непотрібних записів.

Для перегляду контенту в застосунку передбачено кілька режимів. Зокрема, користувач може відкрити загальну стрічку всіх постів, де відображаються всі доступні публікації, а також персоналізовану стрічку підписок, у якій показуються лише записи авторів, на яких підписаний поточний користувач. Окремо реалізовано сторінку з власними публікаціями, що дає можливість швидко переглядати, редагувати та видаляти створений користувачем контент.

У системі реалізовано механізми соціальної взаємодії між користувачами. Зокрема, доступна можливість підписки на інших авторів та скасування підписки. Для зручності користувача реалізовано окремий розділ зі списком акаунтів, на які він підписаний (рис.3.5). Це дозволяє не лише формувати персоналізовану стрічку, а й керувати власними підписками в одному місці.

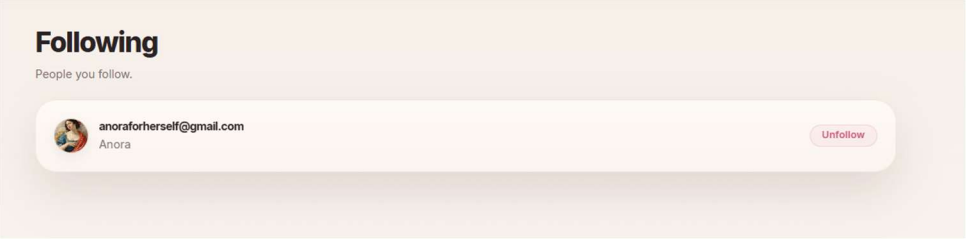

Рис. 3.5. Список акаунтів, на які підписаний користувач

(розроблено автором)

Ще одним важливим елементом взаємодії є система вподобань та коментарів. Користувачі можуть позначати публікації як уподобані, а також переглядати кількість отриманих вподобань. Оновлення стану лайка та його лічильника виконується динамічно без повного перезавантаження сторінки, що покращує загальний користувацький досвід. Додатково до цього на сторінці

окремого поста реалізовано секцію коментарів (рис.3.6), де користувачі можуть залишати текстові повідомлення та переглядати коментарі інших користувачів.

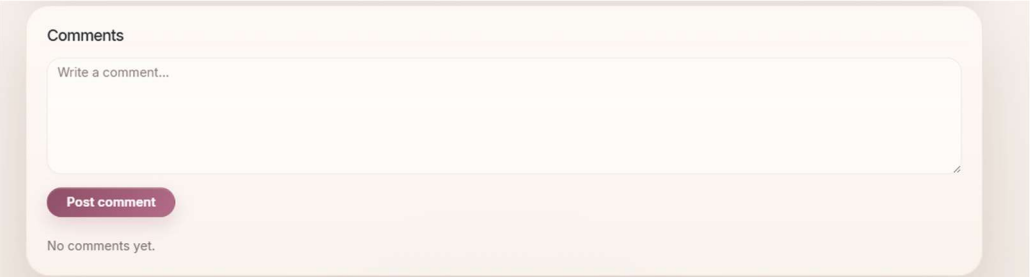

Рис. 3.6. Секція коментарів на сторінці публікації

(розроблено автором)

Для покращення зручності роботи з фотоконтентом у застосунку реалізовано перегляд зображень у модальному вікні (рис.3.7). При натисканні на зображення воно відкривається у збільшеному форматі поверх основного інтерфейсу, що дозволяє розглядати контент без переходу на іншу сторінку. Такий підхід відповідає сучасним тенденціям побудови інтерфейсів соціальних платформ та робить взаємодію з візуальним контентом більш інтуїтивною.

Рис. 3.7. Перегляд зображення у модальному вікні

(розроблено автором)

Інтерфейс застосунку було стилізовано з урахуванням принципів сучасного вебдизайну. Реалізовано світлу візуальну тему з акцентами, адаптованими картками публікацій, стилізованими кнопками взаємодії, аватарами користувачів, а також логічною системою навігації між ключовими розділами платформи. Окрему увагу приділено оформленню сторінок входу, реєстрації, стрічки постів, сторінки профілю та перегляду окремої публікації.

Отже, у розробленому вебзастосунку реалізовано повний базовий цикл взаємодії користувача з платформою фотоконтенту: від реєстрації та підтвердження облікового запису до створення, редагування, перегляду й соціальної взаємодії з публікаціями. Наявний функціонал дозволяє розглядати створений застосунок як працездатну основу для подальшого розширення та вдосконалення.

3.2 Сценарії взаємодії користувача з розробленим вебзастосунком

Взаємодія користувача з вебзастосунком DjangoGramm відбувається як послідовність логічно пов’язаних дій, що забезпечують доступ до основних можливостей платформи. З огляду на призначення розробленої системи, доцільно розглянути типові сценарії роботи користувача, які відображають практичне використання реалізованого функціоналу.

Початковим сценарієм є створення нового облікового запису. Користувач переходить на сторінку реєстрації, вводить необхідні дані та надсилає форму. Після цього система формує токен підтвердження та надсилає його на вказану електронну адресу. Для завершення реєстрації користувач повинен перейти за отриманим посиланням, після чого обліковий запис активується. Таким чином забезпечується перевірка коректності email-адреси та підтвердження належності облікового запису конкретному користувачеві.

Після активації облікового запису реалізується сценарій входу до системи. Для цього користувач може використати стандартну форму авторизації або скористатися сторонніми сервісами, зокрема GitHub чи Google. У результаті

успішної автентифікації користувач отримує доступ до внутрішніх сторінок застосунку та можливість працювати з персоналізованим контентом.

Наступним сценарієм є первинне налаштування профілю. Після входу користувач переходить до сторінки профілю, де може вказати ім’я, коротку інформацію про себе та завантажити аватар. Цей етап є важливим для подальшої ідентифікації користувача в системі, оскільки відомості профілю відображаються у стрічках публікацій, списках підписок і на сторінках окремих постів.

Одним із центральних сценаріїв є створення публікації. Для цього користувач відкриває форму додавання поста, заповнює текстовий опис, додає одне або кілька зображень і, за потреби, вказує тематичні теги. Після збереження новий пост стає доступним для перегляду у загальній стрічці, на сторінці власних публікацій, а також у стрічках підписників автора. У разі потреби користувач може повторно відкрити свій пост, змінити його вміст або видалити.

Подальший сценарій пов’язаний із переглядом контенту. Користувач має змогу переглядати загальну стрічку всіх публікацій, де зібрано весь доступний контент, або перейти до персоналізованої стрічки підписок, яка формується на основі обраних авторів. Крім того, для автора передбачено окрему сторінку власних постів, що використовується для швидкого доступу до особистого контенту та керування ним.

Важливою частиною взаємодії є формування кола підписок. Під час перегляду стрічки користувач може підписуватися на авторів, чиї публікації його цікавлять. Після оформлення підписки нові записи відповідного автора починають відображатися у персоналізованій стрічці. Також користувач може відкрити окремий список підписок, переглянути акаунти, на які він підписаний, і за потреби скасувати підписку.

Окремий сценарій стосується взаємодії з конкретною публікацією. Перейшовши на сторінку окремого поста, користувач отримує змогу детально переглянути зображення, ознайомитися з описом і тегами, поставити вподобання або залишити коментар. Для зручності перегляду фотоконтенту зображення

відкривається у модальному вікні без переходу на іншу сторінку. Такий сценарій забезпечує більш глибоку взаємодію з контентом і підвищує зручність користування платформою.

Для автора реалізовано сценарій керування власними публікаціями. Він передбачає перегляд списку створених постів, відкриття окремого запису, його редагування або видалення. Це дозволяє користувачеві контролювати власний контент, підтримувати його актуальність і в разі потреби змінювати структуру або зміст уже опублікованих матеріалів.

Отже, сценарії взаємодії, реалізовані у вебзастосунку DjangoGramm, охоплюють повний цикл роботи користувача із системою: від реєстрації та входу до створення, перегляду, редагування й оцінювання публікацій, а також організації власного інформаційного середовища через механізм підписок. Це свідчить про цілісність логіки побудови застосунку та практичну придатність реалізованого рішення до повсякденного використання.

3.3 Аналіз ефективності розробленого вебзастосунку та можливих обмежень

Розроблений вебзастосунок DjangoGramm у межах реалізованого функціоналу можна вважати працездатним рішенням для публікації та обміну фотоконтентом між зареєстрованими користувачами. Ефективність застосунку визначається не лише наявністю основних функцій, а й цілісністю їх взаємодії, зручністю користування та відповідністю поставленим вимогам.

Насамперед ефективність розробленого рішення проявляється у тому, що застосунок забезпечує повний базовий цикл роботи користувача з платформою. Користувач має можливість створити обліковий запис, підтвердити електронну адресу, виконати вхід до системи, налаштувати профіль, опублікувати пост, переглядати стрічки публікацій, взаємодіяти з контентом інших користувачів та керувати власними записами. Така завершеність функціонального ланцюга

свідчить про практичну придатність розробленого вебзастосунку до реального використання.

Важливою перевагою застосунку є поєднання стандартної та зовнішньої автентифікації. Підтримка входу як за допомогою логіна і пароля, так і через сторонні сервіси GitHub і Google підвищує гнучкість системи та спрощує доступ до неї для користувачів. Додаткове підтвердження email-адреси забезпечує базовий рівень перевірки достовірності облікового запису та підвищує надійність механізму реєстрації.

З позиції користувацького досвіду ефективність застосунку також забезпечується структурованою організацією контенту. Наявність окремих режимів перегляду, зокрема загальної стрічки, персоналізованої стрічки підписок, сторінки власних публікацій та списку підписок, дозволяє користувачеві швидко знаходити потрібний тип інформації. Такий підхід робить роботу з системою більш передбачуваною та логічною, оскільки різні типи контенту не змішуються в одному інтерфейсному просторі.

Окремої уваги заслуговує реалізація взаємодії з публікаціями. Користувачі можуть ставити вподобання, переглядати їх кількість, залишати коментарі, а також відкривати зображення у модальному вікні для детальнішого перегляду. Частина взаємодій, зокрема лайки та підписки, реалізована без повного перезавантаження сторінки, що позитивно впливає на швидкість сприйняття інтерфейсу та робить застосунок більш наближеним до сучасних вебплатформ. Така поведінка інтерфейсу відповідає сучасному підходу до асинхронної клієнт-серверної взаємодії, зокрема із застосуванням Fetch API для надсилання запитів і динамічного оновлення окремих елементів сторінки [45].

Ефективність застосунку проявляється і в можливості персоналізації профілю користувача. Додавання аватара, імені та короткого опису покращує візуальне сприйняття платформи та сприяє формуванню індивідуального користувацького простору. Це особливо важливо для соціально орієнтованих вебзастосунків, у яких значну роль відіграє не лише сам контент, а й його автор.

Разом із тим розроблений вебзастосунок має низку обмежень, які зумовлені як рамками кваліфікаційної роботи, так і поточним рівнем реалізації. Зокрема, у системі відсутній розширений механізм пошуку публікацій або користувачів, що обмежує зручність навігації у випадку збільшення обсягу контенту. Також не реалізовано розподіл ролей користувачів або адміністративну панель, орієнтовану на модерування контенту в межах окремого інтерфейсу платформи.

До обмежень можна віднести і те, що реалізований застосунок орієнтований передусім на базовий функціональний сценарій, без додаткових соціальних механізмів, характерних для великих платформ. Зокрема, не передбачено системи приватних повідомлень, сповіщень про нові лайки чи коментарі, рекомендацій контенту, статистики активності користувача або алгоритмічного формування стрічки. Усі ці можливості можуть бути предметом подальшого розширення системи.

Певні обмеження стосуються також технічної масштабованості. За поточної реалізації застосунок придатний для демонстрації основних функцій та роботи з помірним навантаженням, однак для використання в умовах значної кількості користувачів може виникнути потреба в оптимізації роботи з файлами, удосконаленні системи зберігання медіаконтенту, кешуванні запитів, підвищенні продуктивності бази даних та розширенні засобів захисту.

Незважаючи на зазначені обмеження, результати розробки свідчать про те, що створений вебзастосунок успішно виконує поставлені завдання. Реалізоване рішення дозволяє користувачам публікувати фотоконтент, взаємодіяти між собою та керувати власним профілем у межах єдиного вебсередовища. Це дає підстави вважати розроблений застосунок функціонально завершеною базовою платформою, яка може бути надалі вдосконалена та розширена відповідно до потреб користувачів і нових технічних вимог.

Висновки до розділу 3

Розглянуто особливості функціонування та елементи інтерфейсу розробленого вебзастосунку для публікації та обміну фотоконтентом. Проаналізовано реалізований функціонал системи, основні сценарії взаємодії користувача з платформою, а також оцінено ефективність створеного рішення й визначено його поточні обмеження.

У результаті проведеного аналізу встановлено, що розроблений вебзастосунок забезпечує виконання ключових завдань, поставлених у межах кваліфікаційної роботи. Реалізоване рішення підтримує реєстрацію та авторизацію користувачів, у тому числі з застосуванням сторонніх сервісів, підтвердження електронної адреси, редагування профілю, створення, перегляд, редагування та видалення публікацій, механізми підписок, вподобань і коментування, а також перегляд зображень у модальному вікні.

Інтерфейс вебзастосунку побудовано відповідно до принципів логічної навігації та зручності користування. Основні користувацькі сценарії є послідовними, зрозумілими та охоплюють повний базовий цикл роботи з платформою. Окремою перевагою є поєднання функціональних можливостей із сучасним стилізованим інтерфейсом, що підвищує загальну зручність роботи з системою.

В той же час, поточна версія застосунку має низку обмежень, пов’язаних із відсутністю розширеного пошуку, системи повідомлень, сповіщень, рекомендацій контенту, розвинених засобів модерації та додаткових інструментів масштабування. Проте виявлені обмеження не знижують цінності отриманого результату, а радше визначають напрями подальшого розвитку системи.

Таким чином розроблений вебзастосунок є функціонально цілісним програмним продуктом, придатним для практичного використання як базова платформа для публікації та обміну фотоконтентом.

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто теоретичні та практичні аспекти розробки вебзастосунку для публікації та обміну фотоконтентом. У ході виконання роботи досліджено сучасні тенденції у сфері онлайн-платформ візуального контенту, проаналізовано існуючі засоби для публікації та обміну фотографіями, а також обґрунтовано вибір технологічного стеку й архітектурних рішень для створення програмного продукту.

Встановлено, що вебзастосунки для роботи з фотоконтентом є важливою частиною сучасного цифрового середовища, оскільки поєднують функції зберігання, публікації, персоналізації та соціальної взаємодії користувачів. Аналіз існуючих платформ дозволив визначити основні функціональні вимоги до системи такого типу, а також сформувати бачення майбутнього програмного рішення. У результаті було обрано доцільний стек технологій, який забезпечує можливість створення повноцінного вебзастосунку з клієнтською та серверною частинами.

Здійснено проєктування структури вебзастосунку, інформаційної моделі та бази даних, визначено основні сутності системи та зв’язки між ними. У процесі розробки реалізовано серверну частину застосунку на базі Django, механізми обробки та збереження даних, роботу з файлами зображень, автентифікацію користувачів, а також клієнтську частину, яка забезпечує зручну взаємодію з функціоналом системи через вебінтерфейс.

Проаналізовано функціонування створеного вебзастосунку та розглянуто ключові елементи його інтерфейсу. У результаті реалізації вдалося створити застосунок, який підтримує реєстрацію та авторизацію користувачів, включаючи підтвердження електронної адреси та вхід через зовнішні сервіси GitHub і Google, редагування профілю, публікацію та редагування постів, додавання фотографій і тегів, формування стрічок публікацій, підписки на інших користувачів, вподобання, коментарі та перегляд зображень у модальному вікні.

Також розглянуто основні сценарії взаємодії користувача з системою та визначено обмеження поточної реалізації.

У результаті виконання роботи поставлену мету досягнуто, а основні завдання виконано в повному обсязі. Розроблений вебзастосунок є функціонально завершеним базовим рішенням для публікації та обміну фотоконтентом, придатним для практичного використання. Створена система демонструє можливість поєднання сучасних засобів веброзробки для реалізації соціально-орієнтованої платформи з інтуїтивним інтерфейсом та базовими механізмами взаємодії між користувачами.

Практична цінність розробленого вебзастосунку полягає в тому, що він може бути використаний як основа для подальшого розширення. Перспективами розвитку системи є впровадження розширеного пошуку, системи повідомлень і сповіщень, інструментів модерації, рекомендаційного механізму, покращення масштабованості, а також розширення аналітичних і соціальних можливостей платформи.

Таким чином було успішно реалізовано вебзастосунок для публікації та обміну фотоконтентом, який відповідає поставленим вимогам, демонструє працездатність обраних технічних рішень і може слугувати основою для подальшого вдосконалення та практичного використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kemp S. Digital 2025: Global Overview Report [Електронний ресурс] // DataReportal. URL: https://datareportal.com/reports/digital-2025-global-overview-report

2. Kemp S. Digital 2025: top social platforms in 2025 [Електронний ресурс] // DataReportal. URL: https://datareportal.com/reports/digital-2025-sub-section-top-social-platforms

3. Kemp S. Instagram Users, Stats, Data & Trends for 2025 [Електронний ресурс] // DataReportal. URL: https://datareportal.com/essential-instagram-stats

4. Kemp S. Digital 2026: Global Overview Report [Електронний ресурс] // DataReportal. URL: https://datareportal.com/reports/digital-2026-global-overview-report

5. Social Media Trends 2026 [Електронний ресурс] // Hootsuite. URL: https://www.hootsuite.com/research/social-trends

6. The 18 social media trends to shape your 2026 strategy [Електронний ресурс] // Hootsuite Blog. URL: https://blog.hootsuite.com/social-media-trends/

7. 60+ social media statistics marketers need to know in 2026 [Електронний ресурс] // Hootsuite Blog. URL: https://blog.hootsuite.com/social-media-statistics/

8. Django documentation contents [Електронний ресурс] // Django documentation. URL: https://docs.djangoproject.com/en/6.0/contents/

9. Models [Електронний ресурс] // Django documentation. URL: https://docs.djangoproject.com/en/6.0/topics/db/models/

10. Model field reference [Електронний ресурс] // Django documentation. URL: https://docs.djangoproject.com/en/6.0/ref/models/fields/

11. Working with forms [Електронний ресурс] // Django documentation. URL: https://docs.djangoproject.com/en/6.0/topics/forms/

12. Form handling with class-based views [Електронний ресурс] // Django documentation. URL: https://docs.djangoproject.com/en/6.0/topics/class-based-views/generic-editing/

13. File uploads [Електронний ресурс] // Django documentation. URL: https://docs.djangoproject.com/en/6.0/topics/http/file-uploads/

14. Using the Django authentication system [Електронний ресурс] // Django documentation. URL: https://docs.djangoproject.com/en/6.0/topics/auth/default/

15. Making queries [Електронний ресурс] // Django documentation. URL: https://docs.djangoproject.com/en/6.0/topics/db/queries/

16. Paginator [Електронний ресурс] // Django documentation. URL: https://docs.djangoproject.com/en/6.0/ref/paginator/

17. Database transactions [Електронний ресурс] // Django documentation. URL: https://docs.djangoproject.com/en/6.0/topics/db/transactions/

18. Get started with Bootstrap 5.3 [Електронний ресурс] // Bootstrap Documentation. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/

19. Navbar [Електронний ресурс] // Bootstrap Documentation. URL: https://getbootstrap.com/docs/5.3/components/navbar/

20. Cards [Електронний ресурс] // Bootstrap Documentation. URL: https://getbootstrap.com/docs/5.3/components/card/

21. Modal [Електронний ресурс] // Bootstrap Documentation. URL: https://getbootstrap.com/docs/5.3/components/modal/

22. Forms [Електронний ресурс] // Bootstrap Documentation. URL: https://getbootstrap.com/docs/5.3/forms/overview/

23. Grid system [Електронний ресурс] // Bootstrap Documentation. URL: https://getbootstrap.com/docs/5.3/layout/grid/

24. Ratios [Електронний ресурс] // Bootstrap Documentation. URL: https://getbootstrap.com/docs/5.3/helpers/ratio/

25. Bootstrap 5.3.3 [Електронний ресурс] // Bootstrap Blog. URL: https://blog.getbootstrap.com/2024/02/20/bootstrap-5-3-3/

26. Image module [Електронний ресурс] // Pillow documentation. URL: https://pillow.readthedocs.io/en/stable/reference/Image.html

27. Tutorial [Електронний ресурс] // Pillow documentation. URL: https://pillow.readthedocs.io/en/stable/handbook/tutorial.html

28. Image file formats [Електронний ресурс] // Pillow documentation. URL: https://pillow.readthedocs.io/en/stable/handbook/image-file-formats.html

29. Pillow documentation (PDF edition) [Електронний ресурс] // Pillow documentation. URL: https://pillow.readthedocs.io/_/downloads/en/stable/pdf/

30. Pillow 7.0.0 release notes [Електронний ресурс] // Pillow documentation. URL: https://pillow.readthedocs.io/en/stable/releasenotes/7.0.0.html

31. Python Social Auth documentation [Електронний ресурс] // Read the Docs. URL: https://python-social-auth.readthedocs.io/en/latest/

32. Django Framework [Електронний ресурс] // Python Social Auth documentation. URL: https://python-social-auth.readthedocs.io/en/latest/configuration/django.html

33. GitHub backend [Електронний ресурс] // Python Social Auth documentation. URL: https://python-social-auth.readthedocs.io/en/latest/backends/github.html

34. Google backend [Електронний ресурс] // Python Social Auth documentation. URL: https://python-social-auth.readthedocs.io/en/latest/backends/google.html

35. Use cases [Електронний ресурс] // Python Social Auth documentation. URL: https://python-social-auth.readthedocs.io/en/latest/use_cases.html

36. Creating an OAuth app [Електронний ресурс] // GitHub Docs. URL: https://docs.github.com/en/apps/oauth-apps/building-oauth-apps/creating-an-oauth-app

37. Authorizing OAuth apps [Електронний ресурс] // GitHub Docs. URL: https://docs.github.com/en/apps/oauth-apps/building-oauth-apps/authorizing-oauth-apps

38. Authenticating to the REST API with an OAuth app [Електронний ресурс] // GitHub Docs. URL: https://docs.github.com/en/apps/oauth-apps/building-oauth-apps/authenticating-to-the-rest-api-with-an-oauth-app

39. Using OAuth 2.0 for Web Server Applications [Електронний ресурс] // Google forDevelopers.URL:https://developers.google.com/identity/protocols/oauth2/web-server

40. Using OAuth 2.0 to Access Google APIs [Електронний ресурс] // Google for Developers. URL: https://developers.google.com/identity/protocols/oauth2

41. Manage OAuth Clients [Електронний ресурс] // Google Cloud Support. URL: https://support.google.com/cloud/answer/15549257

42. File Upload Cheat Sheet [Електронний ресурс] // OWASP Cheat Sheet Series.
URL:https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html

43. Image performance [Електронний ресурс] // web.dev. URL: https://web.dev/learn/performance/image-performance

44. Using responsive images in HTML [Електронний ресурс] // MDN Web Docs. URL:https://developer.mozilla.org/enUS/docs/Web/HTML/Guides/Responsive_images

45. Fetch API [Електронний ресурс] // MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

Додаток А

Фрагмент програмної реалізації вебзастосунку DjangoGramm

```
from io import BytesIO

from PIL import Image

from django.conf import settings
from django.core.files.base import ContentFile
from django.db import models
from django.db.models import F, Q


class Tag(models.Model):
    name = models.CharField(max_length=50, unique=True,
db_index=True)

    def __str__(self):
        return self.name

class Post(models.Model):
    author = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, related_name='posts')
    caption = models.TextField(blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    tags = models.ManyToManyField('Tag', related_name='posts',
blank=True)

    class Meta:
        ordering = ['-created_at']

class PostImage(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE,
related_name='images')
```

```
    image = models.ImageField(upload_to='posts/')
    order = models.PositiveIntegerField(default=0)
    preview = models.ImageField(upload_to='posts/previews/',
blank=True)

    class Meta:
        ordering = ['order', 'id']

    def save(self, *args, **kwargs):
        super().save(*args, **kwargs)

        if self.image and not self.preview:
            try:
                self.image.file.seek(0)
                img = Image.open(self.image)

                if img.mode != 'RGB':
                    img = img.convert('RGB')

                img.thumbnail((400, 400))

                buffer = BytesIO()
                img.save(buffer, format='JPEG', quality=85,
optimize=True)
                buffer.seek(0)

                preview_filename = f"{self.pk}_preview.jpg"

                self.preview.save(preview_filename,
ContentFile(buffer.getvalue()), save=False)
                super().save(update_fields=['preview'])

            except Exception as e:
                print(f'Error creating preview: {e}')
```

```
    def delete(self, *args, **kwargs):
        if self.image:
            self.image.delete(save=False)
        if self.preview:
            self.preview.delete(save=False)
        super().delete(*args, **kwargs)

class Like(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, related_name='likes')
    post = models.ForeignKey(Post, on_delete=models.CASCADE,
related_name='likes')
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        constraints = [models.UniqueConstraint(fields=['user',
'post'], name='uq_like_user_post')]

class Follow(models.Model):
    follower = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, related_name='following')
    following = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, related_name='followers')

    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        db_table = 'posts_follow'
        constraints = [
            models.UniqueConstraint(fields=['follower',
'following'], name='uq_follow'),

models.CheckConstraint(check=~Q(follower=F("following")),
name="no_self_follow"),
```

```
        ]
    def __str__(self):
        return f'{self.follower} -> {self.following}'

class Comment(models.Model):
    post = models.ForeignKey("Post", on_delete=models.CASCADE,
related_name="comments")
    author = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, related_name="comments")
    text = models.TextField(max_length=500)
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ["created_at"]

    def __str__(self):
        return f"{self.author} → {self.post_id}"
```

ЗГОДА здобувачки вищої освіти

Державного університету економіки і технологій про
перевірку кваліфікаційної роботи на прояви
академічного плагіату
та розміщення в Репозитарії Університету

Я, **Лісовець Валерія Сергіївна**,

підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота

«Розробка вебзастосунку для публікації та обміну фото-контентом»

виконана самостійно та не містить академічного плагіату. Я не надавала і не одержувала недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлена. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформована, що відповідно до «Положення про Репозитарій (електронну базу даних) Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету (Репозитарії ДУЕТ). З умовами такого розміщення ознайомлена.

15.06.26 р  Лісовець В. С.