ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО «ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«МІЖРЕГІОНАЛЬНА АКАДЕМІЯ УПРАВЛІННЯ ПЕРСОНАЛОМ»»

Факультет комп’ютерно-інформаційних технологій

Кафедра комп’ютерних інформаційних систем і технологій

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Тема «Розробка веб-орієнтованої інформаційної системи управління контентом блог-платформи»

Виконав студент групи ІК-9-22-Б1ІПЗ(4.0д) Купченко Я.Ю.

Керівник: викладач Яременко Д.М.

Допущено до захисту

Завідувач кафедри _____ д.е.н., професор Кавун С.В.

(підпис)

Київ, 2026

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи складається з 92 сторінок, містить 38 рисунків, 2 таблиці, 2 додатки, 31 джерело

ВЕБ-ОРІЄНТОВАНА ІНФОРМАЦІЙНА СИСТЕМА, УПРАВЛІННЯ КОНТЕНТОМ, БЛОГ-ПЛАТФОРМА, ПРОГРАМУВАННЯ, PHP, JAVASCRIPT, BOOTSTRAP 5, MySQL, DOCKER, CMS.

Об'єкт дослідження – це вебзастосунок для керування контентом на блог-платформі, присвяченій програмуванню.

Завданням роботи стало створення такої IT-спільнотової блог-платформи, котра б надавала змогу розміщувати, змінювати й перевіряти програмні публікації, реалізовувала реєстраційний та допуск (авторизаційний) механізми для відвідувачів, мала функціонал залишати відгуки, дозволяла б категоризувати матеріали відповідно до IT-тематик, а також забезпечувала б повне керування системою через адміністративну сторінку. Увесь цей функціонал має бути доступним коректно як на стаціонарних ПК, так і на портативних ґаджетах.

Для створення цього рішення застосовувалися такі технології: PHP, JavaScript, Bootstrap 5, база даних MySQL, а також технологія контейнеризації Docker.

Підсумком праці стала готова платформа для розміщення технічного контенту, яка містить розмежування доступу для різних ролей користувачів, адаптивний візуальний вигляд та інфраструктуру на базі контейнерів. Такий продукт цілком придатний до застосування як засіб поширення фахових знань серед технічної спільноти.

WEB-BASED INFORMATION SYSTEM, CONTENT MANAGEMENT, BLOG PLATFORM, PROGRAMMING, PHP, JAVASCRIPT, BOOTSTRAP 5, MySQL, DOCKER, CMS.

The subject of this study is a web application for managing content on a blogging platform dedicated to programming.

The objective of this project was to create an IT community blogging platform that would allow users to post, edit, and review software-related posts, implement registration and authorization mechanisms for visitors, include a feature for leaving comments, allow for categorizing content according to IT topics, and provide full system management via an administrative dashboard. All of this functionality must work correctly on both desktop PCs and mobile devices.

The following technologies were used to create this solution: PHP, JavaScript, Bootstrap 5, a MySQL database, and Docker containerization technology.

The result of this work is a ready-to-use platform for hosting technical content, featuring access restrictions for different user roles, a responsive design, and a container-based infrastructure. This product is well-suited for use as a tool for disseminating professional knowledge within the technical community.

ЗМІСТ

ВСТУП

Бурхливий розвиток інформаційних технологій неухильно підвищує потребу в спеціалізованих ресурсах для обміну технічними знаннями. Як розробникам, так і студентам IT-спеціальностей та технічним фахівцям потрібні зручні майданчики, де вони могли б розміщувати матеріали з програмування, обговорювати технічні рішення та ділитися досвідом. Наявні універсальні системи керування контентом — такі як WordPress, Squarespace чи Ghost — не враховують специфіки IT-матеріалів і не мають необхідної тематичної структуризації за мовами програмування, технологіями чи напрямами розробки. З огляду на це, розробка спеціалізованої блог-платформи для IT-спільноти є на часі.

Завданням дипломної роботи є проєктування та створення повнофункціональної тематичної блог-платформи у сфері програмування.

Для досягнення цієї мети необхідно вирішити такі кроки:

- провести аналіз предметної сфери та огляд існуючих ресурсів для розміщення IT-контенту;
- дослідити та порівняти технологічні стеки для серверного та клієнтського боку вебзастосунків;
- спроєктувати архітектурну модель системи та структуру бази даних на базі MySQL;
- розробити функціональні модулі, відповідальні за реєстрацію, авторизацію, керованість публікаціями, коментування та IT-категоризацію;
- впровадити механізм розмежування прав доступу для різних груп користувачів;
- забезпечити кросплатформну адаптивність зовнішнього вигляду за допомогою Bootstrap 5;

- здійснити розгортання системи із застосуванням технології Docker-контейнеризації;
- виконати тестування та оцінку ефективності розробленого рішення.

Об'єкт дослідження: процес створення веб-орієнтованої системи для керування контентом, призначеної для блог-платформи у галузі програмування та інформаційних технологій.

Предмет дослідження: методики та інструментарій, що застосовуються при проєктуванні й реалізації веб-платформ для управління контентом.

Новизна отриманих результатів полягає у створенні блог-платформи, цільово орієнтованої на IT-тематику.

Практична цінність полягає у можливості використання розробленої платформи для публікації технічних статей, освітніх матеріалів та налагодження обміну знаннями у IT-сфері. Адаптивний дизайн гарантує зручність роботи як на персональних комп'ютерах, так і на портативних пристроях.

Сфера застосування: IT-спільноти, освітні заклади, команди розробників та окремі фахівці, залучені до програмування та інформаційних технологій.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ У СФЕРІ ІТ-БЛОГІНГУ

1.1 Актуальність розробки тематичної блог-платформи для ІТ-спільноти

Ведення блогу як форми онлайн-публікацій має вже кілька десятиліть історії, трансформувавшись із особистих записників на дієвий механізм для поширення знань та професійного зростання. Блог являє собою ресурс, що регулярно оновлюється, де автори розміщують текстові матеріали, зображення та відео . Така організація допомагає читачам передусім бачити актуальну інформацію, що є надзвичайно важливим у сферах, які швидко змінюються, зокрема у сфері інформаційних технологій.

Сектор ІТ належить до найбільш динамічних: постійно виникають нові технологічні рішення, програмні каркаси, мови кодування та інструментарій. Фахівцям-розробникам, інженерам та іншим ІТ-професіоналам, життєво необхідне своєчасне та якісне джерело відомостей. Саме тому спеціалізовані блог-майданчики для ІТ-спільноти набувають дедалі більшої значущості, вони дають можливість професіоналам обмінюватися набутим досвідом, публікувати технічні статті, посібники та огляди, утворюючи, таким чином, живу енциклопедію цієї галузі.

Серед усього розмаїття блогів виділяють окремий пласт — технічні блоги, що охоплюють теми, пов'язані з програмним забезпеченням, практичними інструкціями. Однак більшість нинішніх універсальних платформ не враховують особливих потреб ІТ-аудиторії: відсутнє коректне виділення синтаксису коду, бракує зручної системи класифікації за технологіями, а також інструментарію для технічного аналізу публікацій.

Більше того, ведення тематичного блогу несе відчутну користь як для окремого експерта, так і для всієї спільноти. Публікація матеріалів, що демонструють експертизу, дає змогу автору закріпитися як визнаному фахівцю, розвиваючи довіру серед колег та потенційних наймачів. Для ІТ-спеціаліста

особистий ресурс чи корпоративний блог по суті функціонує як відкрите портфоліо його знань.

Не менш вагомими є комерційний та навчальний аспекти. Цінний контент сприяє покращенню видимості ресурсу у пошукових системах, приваблює релевантну аудиторію та стимулює формування активного ком'юніті навколо платформи [1]. Стосовно IT-блогінгу це трансформується у природне розширення бази читачів програмістів, студентів технічних напрямків та IT-керівників, які шукають конкретні відповіді на технічні питання.

Актуальність релевантного контенту підтверджується і практиками контент-маркетингу: матеріали, що відповідають на реальні запити цільової аудиторії, значно ефективніше формують прихильність та довготривалі зв'язки з читачем [2]. Для IT-ресурсу це критично важливо, оскільки технічна аудиторія вимоглива: вона миттєво розрізняє поверховий текст від того, що є глибоким та прикладним.

1.2 Аналіз існуючих платформ для публікації IT-контенту

Здійснено аналіз існуючих ресурсів для розміщення технічних матеріалів перед тим, як приступити до створення власного продукту, аби чітко зрозуміти їхній функціонал та існуючі прогалини.

– DEV.to, інакше відома як DEV Community, являє собою відчинений блогінговий ресурс із відкритим кодом, який від початку свого існування був спрямований винятково на розробників та технічних фахівців. На відміну від більшості блог-хостингів, які намагаються залучити широке коло користувачів, Dev.to чітко концентрується на одній спільноті, людях, залучених до розробки програмного забезпечення, незалежно від рівня їхньої кваліфікації. Тут розміщують технічні дописи, навчальні гайди, ставлять питання та проводять дискусії навколо актуальних тем у сфері IT [5]. Цей майданчик функціонує на базі Forem, що є відкритим програмним забезпеченням для формування онлайн-спільнот.

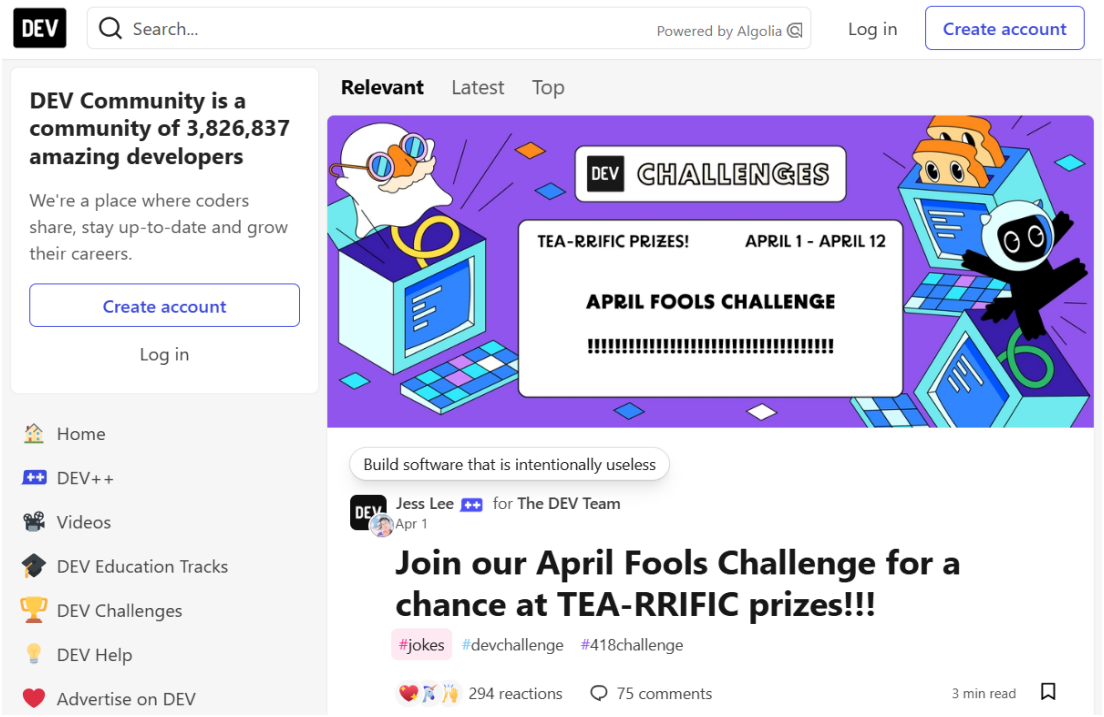

Рисунок 1.1 – Головна сторінка платформи Dev.to

Переваги. Найбільш значущим позитивним аспектом є його абсолютна безкоштовність. Для створення облікового запису та публікації матеріалів не потрібна жодна фінансова витрата, що робить платформу доступною для будь-кого. Це особливо цінно для тих, хто навчається, та новачків, які бажають розпочати ведення техблогу, не вкладаючи власних коштів.

Є суттєва зручність для створення технічного контенту. Редактор підтримує синтаксис Markdown, що дозволяє з легкістю вбудовувати та оформлювати приклади коду безпосередньо у тексті статті. Для автора технічного спрямування це надзвичайно практично, адже відпадає потреба у використанні зовнішніх інструментів чи встановленні додаткових модулів.

Крім того, профіль на цьому ресурсі фактично функціонує як частина технічного резюме (портфоліо); потенційні роботодавці мають можливість оцінити, про що пише фахівець і як він міркує.

Приємним доповненням є функція перенесення матеріалів з інших джерел. Якщо автор уже веде блог на Hashnode чи іншому особистому сайті, він може без

зайвих ускладнень імпортувати свої дописи на Dev.to, зберігаючи при цьому посилання на першоджерело [5]. Це вигідна опція для тих, хто прагне розширити свою читацьку аудиторію, не дублюючи роботу.

Найбільша помітна риса, є однакове ставлення до всіх авторів. На Dev.to відсутня така штучна ієрархія популярності, де б публікації нових дописувачів мали б меншу прохідність, аніж матеріали вже визнаних експертів [3]. Алгоритмічна система не гальмує новачків, віддаючи перевагу «зіркам» майданчика, тому навіть новозареєстрований користувач має цілком реальний шанс здобути значну аудиторію вже з першої своєї статті.

Недоліки. Незважаючи на усі переваги, цей ресурс має і доволі суттєві вади. Основна з них — це відсутність розгорнутої системи поділу на категорії чи тематичні секції. Увесь наявний контент угруповується виключно за мітками (тегами), і якщо виникає потреба відшукати матеріали, присвячені певній темі, єдиний шлях, це пошук саме за цими тегами [3]. Для читача, що прагне планомірно опановувати обраний напрямок, такий підхід є не надто зручним: немає чітко визначених розділів на кшталт «Backend», «Frontend», «DevOps» та подібних, усе розкидане по тегах, які самі автори можуть присвоювати доволі довільно.

Також слід звернути увагу на те, що активність спільноти розподіляється нерівномірно. Хоча платформа і позиціонується як жвавий осередок для програмістів, деякі користувачі вказують на те, що справжній рівень залученості в коментарях та дискусіях доволі низька [5]. Щоденно з'являються тисячі дописів, проте ґрунтовних обговорень та зворотного зв'язку під кожною публікацією трапляється значно менше, ніж логічно було б припустити для платформи такого розміру.

– Hashnode — це платформа для технічних блогів, яка постала на ринку порівняно нещодавно, позаду своїх попередників. Проте, за нетривалий період вона змогла завоювати неабияку прихильність серед програмістів, завдяки сфокусованості на потребах саме IT-спільноти. На противагу блог-сервісам широкого профілю, Hashnode від самого старту проєктувалася з огляду на

особливості технічного наповнення: публікації із програмними кодами, рецензії на інструментарій, навчальні посібники та архітектурні концепції.

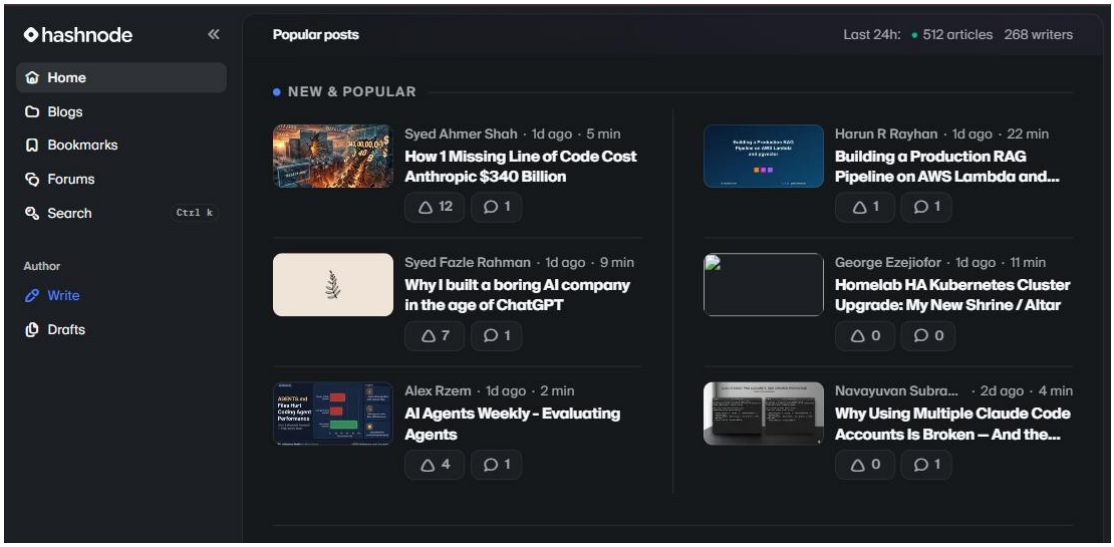

Рисунок 1.2 – Головна сторінка платформи Hashnode

Серед сильних сторін Hashnode виділяється значно краща реалізація підтримки Markdown, яка пропонує гнучкість у форматуванні коду та легке додавання медіафайлів, що явно переважає функціонал DEV Community. Додатковим плюсом є прив'язка до GitHub, що дає змогу тримати публікації у синхроні із репозиторієм. Фокус на аудиторії, яка складається виключно з розробників, гарантує високу якість відгуків на технічні матеріали.

Говорячи про недолікі Hashnode, слід наголосити, що значний обсяг розширених можливостей, зокрема, глибока аналітика, розширені опції налаштування, повноцінне використання власного доменного імені та пріоритетне обслуговування, відкриваються лише після оформлення платного доступу, що суттєво звужує простір для авторів, які тільки починають свій шлях [3].

– WordPress незмінно лишається найпопулярнішим вибором для будь-якого типу інтернет-ресурсу та персональних щоденників. Ця платформа була започаткована у 2003 році, коли розробники Метт Мулленвег та Майк Літтл представили інструмент, сфокусований перш за все на ведення блогів. Проте з

плином часу можливості цієї системи значно розширилися, і сьогодні вона слугує фундаментом для сайтів найрізноманітнішої спрямованості, від особистих заміток до великих міжнародних корпоративних порталів.

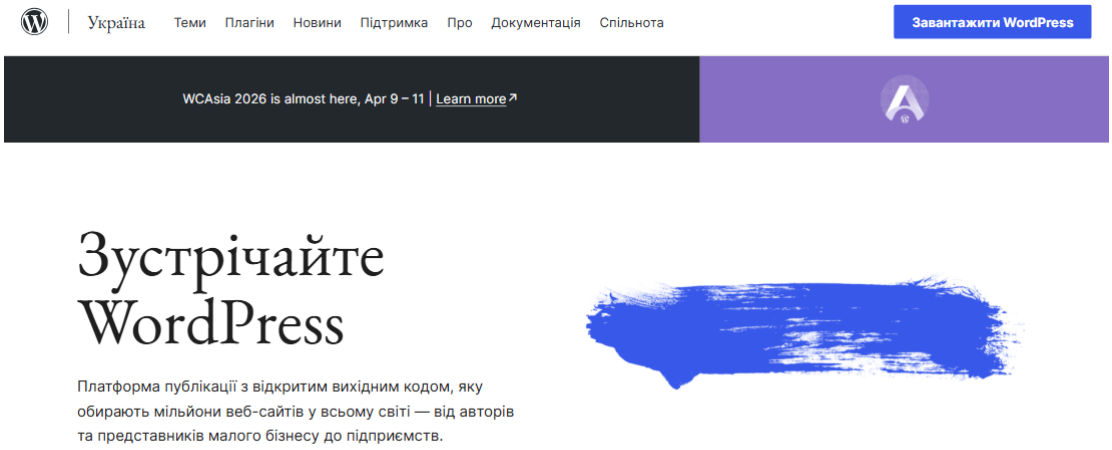

Рисунок 1.3 – Головна сторінка платформи WordPress

Ключовим моментом є необхідність розрізняти два окремі продукти, що експлуатують спільний бренд. WordPress.org – це програмне забезпечення з відкритим вихідним кодом, яке вимагає встановлення на власний сервер і надає користувачеві повний контроль над ресурсом. Натомість WordPress.com є послугою, розміщеною у хмарі, з певними обмеженнями у безкоштовному плані: власник не має права розміщувати сторонню рекламу, його вибір доменних імен обмежений без переходу на платний тариф, і відсутні повноцінні інструменти для електронної комерції. Для серйозних IT-проєктів чи блогів, як правило, більш виправданим є використання саме WordPress.org.

Сильні сторони цієї платформи. Однією з найвагоміших переваг WordPress вважається його надзвичайно розвинена екосистема доповнень (плагінів) та візуальних оформлень (тем). В офіційному сховищі системи налічується близько 57 тисяч безкоштовних плагінів, і кількість комерційних модулів невпинно зростає. Завдяки цим розширенням можливо вирішити практично будь-яке

завдання: від тонкого налаштування SEO-оптимізації та захисту від небажаної пошти до побудови функціонального інтернет-магазину.

Серед головних переваг WordPress варто окреслити декілька аспектів. Програмне забезпечення WordPress.org поширюється безкоштовно, що суттєво знижує фінансовий поріг входу для нових авторів та невеликих редакцій. Навколо WordPress сформувалася потужна спільнота як користувачів, так і розробників; як наслідок, у мережі доступна величезна база навчальних матеріалів, офіційної документації, відеоінструкцій та форумів для отримання техпідтримки. Також вагомим плюсом є простота інсталяції та оновлення системи: більшість хостинг-провайдерів пропонують інсталювати WordPress автоматично у кілька кліків, а оновлення ядра чи плагінів здійснюється через адмінпанель без потреби редагування коду [6]. Ця платформа вирізняється гнучкістю налаштувань і може бути використана для створення найрізноманітніших типів вебсайтів, від простих блогів до повноцінних торгових майданчиків, що є серйозним аргументом при виборі інструменту для розвитку IT-спільнот [4].

Слабкі місця платформи. Незважаючи на очевидні позитивні моменти, WordPress має певні обмеження, які необхідно врахувати ще на етапі планування проєкту. Перш за все, це стосується складності освоєння для людей із мінімальним технічним досвідом: хоча інтерфейс адмінчастини є зрозумілим, відповідальність за питання, пов'язані з хостингом, регулярними оновленнями, створенням копій та безпекою, повністю лягає на плечі власника ресурсу.

Наступний недолік — це потенційне зниження швидкодії: велика кількість одночасно активних розширень збільшує навантаження на сервер та сповільнює завантаження сторінок, що є особливо критичним для сайтів із високою читацькою активністю [6]. Насамкінець, хоча базовий функціонал є безкоштовним, реальна вартість утримання повноцінного проєкту на WordPress може виявитися досить значною: купівля преміум-тем, необхідних платних плагінів, високоякісний хостинг та можлива залученість розробника формують сукупний бюджет, який потрібно планувати заздалегідь [4, 6].

– Ghost являє собою досить нову, проте вже встигну набрати авторитет систему керування контентом, яка від початкової ідеї проєкту розроблялася з акцентом на потреби тих, хто створює та публікує матеріали. На відміну від більш універсальних рішень, Ghost сфокусований саме на ведення блогів, email-розсилках та медійних проєктах. Це помітно як на загальній простоті інтерфейсу, так і на наявності потужних інструментів для роботи з контентом.

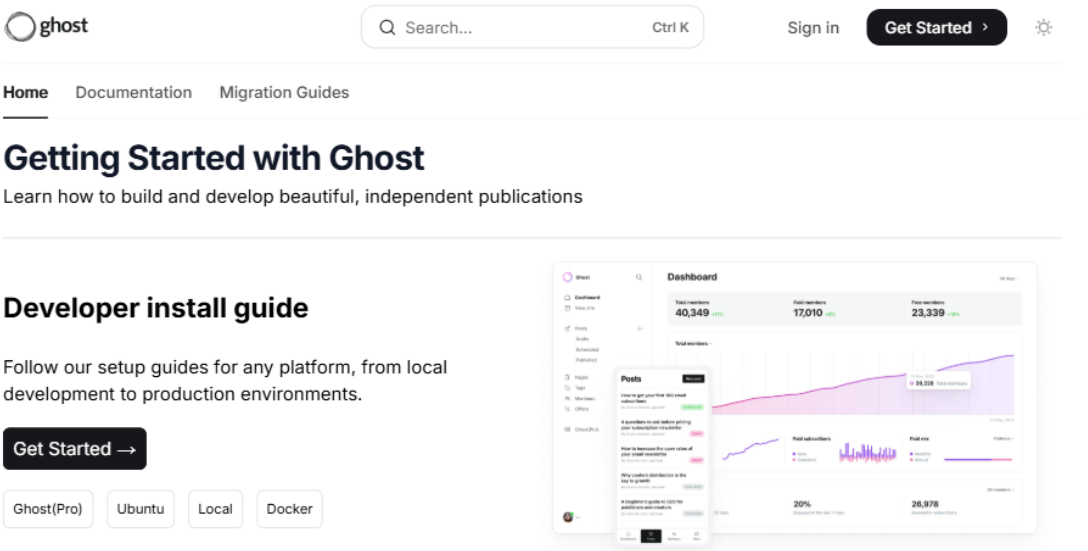

Рисунок 1.4 – Головна сторінка платформи Ghost

Платформа доступна у двох форматах. По-перше, це версія Ghost з відкритим кодом (Open-Source), яку можна завантажити безкоштовно та розмістити на власному сервері. Такий підхід гарантує повний доступ до програмного коду та можливість глибоких змін під будь-які потреби. Проте, він зобов'язує власника мати достатній технічний досвід для адміністрування та забезпечення безперебійної роботи. Другий варіант — Ghost Pro, це платний хмарний сервіс, де всі проблеми з інфраструктурою, оновленнями та безпекою бере на себе команда розробників [7]. Таким чином, користувачі можуть обрати

Ghost як послугу розміщення або як програмне забезпечення, яке інсталюється на обраному користувачем хостингу [4].

Переваги системи. Ключовою перевагою Ghost виступає надзвичайно висока швидкість завантаження вебсторінок. Система здатна обробляти понад 200 мільйонів звернень щомісяця та впевнено справляється з несподіваними піками відвідуваності. Варто зазначити, що сторінки завантажуються менш ніж за три секунди навіть без тонкого налаштування механізмів кешування, що є великим плюсом як для позицій у пошукових системах, так і для зручності самих читачів.

Редактор контенту в Ghost побудований на модульному (блочному) принципі. Він підтримує інтеграцію зображень, відео, форматування Markdown, цитат, а також вбудовування контенту з популярних платформ, як-от YouTube, Instagram чи SoundCloud. Платформа також дає змогу призначати публікації на майбутній час та попередньо переглядати, як вони виглядатимуть після виходу, що значно оптимізує роботу редакції.

Ghost має вбудований функціонал для email-розсилок: після випуску нового допису всі підписники автоматично отримують його на свою пошту без потреби додаткових конфігурацій з боку автора. Для забезпечення злагодженої роботи команди передбачено цілу ієрархію прав доступу — власник, адміністратор, редактор, автор та учасник, що дозволяє кільком особам паралельно керувати контентом, уникаючи конфліктів прав доступу.

Недоліки платформи. Незважаючи на вагомі позитивні аспекти, Ghost має певні мінуси, які слід брати до уваги при виборі. Насамперед, модель ціноутворення Ghost Pro має підвищувальний характер і безпосередньо корелює з кількістю аудиторії. При зростанні читацької бази щомісячні платежі можуть відчутно збільшуватися, що робить сервіс не надто вигідним для авторів із великою спільнотою, які, однак, не використовують платні функції монетизації.

Складність самостійного впровадження стосується версії Open-Source. Безкоштовний варіант Ghost вимагає від користувача не лише навичок серверного адміністрування, а й розуміння специфіки середовища Node.js, на

якому розроблена ця система. На відміну від WordPress, що встановлюється на багатьох хостингах фактично в один клік через типові інсталятори, Ghost потребує ручного налаштування сервера, що суттєво обмежує коло осіб, здатних розгорнути його самостійно без залучення фахівців [7].

– Конструктори сайтів, як Wix чи Squarespace, дозволяють швидко запустити блог за лічені години, не маючи жодних спеціальних технічних знань. Але такий підхід більше підходить для персональних сторінок або комерційних візитівок, ніж для повноцінного технічного ресурсу: можливості персоналізації обмежені, базові SEO-інструменти доволі слабкі, а залежність від материнської платформи є дуже високою [4].

Зроблений зіставний розгляд існуючих майданчиків, призначених для розміщення IT-матеріалів, демонструє, що кожна з оглянутих систем володіє як своїми унікальними рішеннями, так і вагомими вадами. Платформи DEV.to разом із Hashnode сфокусовані для розробників; однак DEV.to страждає від браку системи логічного структурування, а Hashnode накладає обмеження на використання важливого функціоналу, якщо не оформити преміум-доступ. WordPress славиться своєю адаптивністю та широкою мережею доповнень, але потребує чималих ресурсів для обслуговування та управління. Ghost пропонує відмінну швидкість роботи та інтуїтивно зрозумілий редактор, проте його хмарне розміщення стає затратним із примноженням читацької бази, тоді як самостійне інсталювання вимагає ґрунтовної технічної фаховості. Сервіси для створення сайтів типу Wix та Squarespace дають змогу миттєво запустити проєкт, але вони не відповідають потребам повноцінного технічного блогу. Наведене вище доводить доцільність створення власного інструменту для IT-контенту, що може усунути недоопрацювання, виявлені в існуючих рішеннях.

1.3 Аналіз вимог до функціональності системи

Формулювання функціональних вимог — це один із ключових етапів на старті роботи над будь-яким програмним забезпеченням. Під функціоналом, що

має бути реалізований, ми розуміємо чітке визначення того, які саме дії система повинна виконувати: які завдання обробляти, які сервіси пропонувати різним групам користувачів та як відповідати на їхні запити. Як вказують К. Вігерс та Дж. Біті у своїй праці "Software Requirements", саме повнота та бездоганна точність сформованих вимог лягають в основу якості фінального програмного виробу [9]. Нечітке чи неповне окреслення вимог неминуче тягне за собою хибні рішення на етапі проєктування, необхідність повторного виконання вже виконаних робіт і, відповідно, зростання часових і фінансових витрат на розробку.

Предметом дослідження у цій роботі виступає веб-додаток, призначений для ведення блогу, сфокусований на темах програмування та спрямований на IT-спільноту. Для встановлення необхідного обсягу функціональних можливостей, які підлягають реалізації, було проведено аналіз існуючих практичних моделей побудови платформ для ведення блогів [8] та класифіковано вимоги відповідно до головних аспектів функціонування системи. Усі визначені вимоги систематизовано та розподілено за відповідними групами, опис яких наведено далі.

– Аспекти керування матеріалами. Публікація та редагування матеріалів є центральним завданням будь-якої системи, що орієнтована на розміщення контенту. Запропонований механізм має забезпечувати повний життєвий цикл роботи з текстами: їх створення, редагування, зміну відповідного статусу та повне видалення. Кожна окрема публікація обов’язково повинна мати свій заголовок, основний текст, а також бути асоційованою з певною тематичною категорією. Якісне оформлення та логічна розбудова поданого матеріалу прямо пропорційно впливають на легкість його засвоєння тими, хто його читає [10]. Групування контенту за тематиками є невід’ємною частиною інтуїтивно зрозумілої навігаційної системи: коли статті згруповані за напрямками (скажімо, веб-розробка, методи обчислень, бази даних, мови програмування), це дає змогу користувачеві оперативно віднайти потрібні йому публікації, не прокручуючи весь масив інформації.

Окрім пошуку через категорії, система підтримує функціонал знаходження публікацій, ґрунтуючись на введених ключових словах. Цей інструмент доступний як для звичайних відвідувачів, так і для користувачів, які пройшли авторизацію. Ключові слова слугують фундаментальним засобом, через який відвідувач формулює свій запит і отримує відповідні, релевантні результати [11]. Користувач вписує пошуковий запит, після чого система видає перелік статей, у заголовку чи безпосередньо у змісті яких знайдено збіги. Внаслідок цього, знаходження необхідного матеріалу відбувається миттєво, навіть якщо читач не знає точної класифікації публікації.

Окремо передбачено можливість оновлення стану публікації, переключаючи її між станами «викладена» та «чернетка». Такий підхід дає змогу попередньо опрацювати матеріал, не виводячи його у загальний доступ відразу, а розміщуючи тоді, коли це буде найбільш доречно. У запровадженій архітектурі всі операції зі створення та управління публікаціями сконцентровані винятково в адміністративній зоні, що повністю відповідає обраній моделі розмежування прав доступу.

– Визначення для реєстрації та перевірки особи. Автентифікація являє собою комплекс заходів, спрямованих на підтвердження особистості користувача, що дає змогу користувачу використовувати певний функціонал системи відповідно до її призначеної ролі [12]. У цій розробленій системі впроваджено структуру доступу з двома рівнями: стандартний зареєстрований користувач та особа з правами адміністратора.

Користувачі, які не пройшли реєстрацію, можуть лише ознайомлюватися з опублікованими матеріалами, не маючи змоги взагалі коментувати чи виконувати будь-які інші дії. Для розширення наданих можливостей обов'язковим є процес реєстрації, який вимагає зазначення імені користувача, електронної пошти та встановлення пароля. Після успішного проходження процедури входу, відвідувач, який зареєструвався, набуває права додавати коментарі до статей.

Особа з правами адміністратора отримує повний доступ до керуючої панелі та усіх інструментів для управління системою. Розподіл прав доступу між визначеними ролями здійснюється на рівні внутрішньої логіки сервера: під час обробки кожного окремого запиту відбувається верифікація, чи є поточний користувач авторизованим та яку саме роль він займає.

Варто підкреслити, що в нинішньому стані системи відсутня проміжна роль, яку можна було б назвати «автор». Подібна роль надала б зареєстрованій особі можливість самостійно генерувати публікації без необхідності мати адміністраторські повноваження. Відсутність цієї ролі була свідомим вибором архітектури в контексті поставлених завдань: функціонал створення публікацій наразі покладено виключно на адміністратора. При подальшому розширенні можливостей платформи, ця роль може бути інтегрована без необхідності кардинальних трансформацій у загальній будові системи.

– Особливості системи, що відповідає за коментарі. Можливість залишати відгуки до публікацій є ключовою складовою будь-якої системи для ведення блогів. Коментарі слугують каналом для комунікації між тими, хто створює контент, та читачами, сприяють жвавим дискусіям, максимізуючи залучення наявної аудиторії. Якісно спроектована реалізація цього функціоналу має прямий вплив на динаміку спільноти ресурсу [13].

У втіленій архітектурі користувач, який пройшов процедуру аутентифікації, отримує право додавати свої зауваження під будь-якою статтею, що вже була опублікована. Кожен такий внесок реєструється з прив'язкою до профілю користувача, який його створив, та до відповідної публікації. Водночас, особа з адміністративними правами може інспектувати весь масив коментарів і видаляти ті з них, що визнані недоречними чи такими, що не відповідають встановленому регламенту майданчика. Отже, первинне управління контентом шляхом модерації здійснюється саме на рівні адміністративного доступу.

– Вимоги до Адміністративного Інтерфейсу. Адмін-панель слугує головним вузлом керування у всій системі. Вона забезпечує особі, відповідальній за

адміністрування, інтуїтивно зрозумілий простір для здійснення усіх важливих дій, при цьому усувається потреба у прямому втручанні у сховище даних або програмний код. Наявність простої у освоєнні та багатофункціональної адмінзони вважається однією з фундаментальних вимог для сучасних інтернет-рішень [14].

У запропонованій архітектурі адміністративний розділ охоплює чотири головні сфери діяльності. Насамперед, це робота з контентом: адміністратор має змогу генерувати свіжі матеріали, вносити правки до існуючих, змінювати їхній стан та видаляти, якщо це необхідно. По-друге, керування категоріями: надається функціонал для запровадження нових тематичних розділів, коригування їхніх найменувань та вилучення тих, чия актуальність минула. По-третє, нагляд за коментарями: адміністратор отримує повний огляд усіх повідомлень, залишених користувачами, та можливість видалити будь-яке з них. І, нарешті, четверте — перегляд профілів: адміністративний персонал може бачити реєстр зареєстрованих осіб разом із ключовою інформацією про них.

Створений інтерфейс є достатнім мінімумом для всебічного контролю над платформою. Його дизайн продумано таким чином, щоб будь-яку типову операцію можна було завершити за мінімальну кількість кроків, уникаючи будь-якої надмірної заплутаності.

– Критерії адаптивності користувацького середовища. Нинішня аудиторія мережевих сервісів використовує широкий спектр апаратних засобів: стаціонарні ПК, ноутбуків, планшетні пристрої та мобільні телефони. Відтак, належне представлення візуального оформлення на дисплеях різної діагоналі є безумовним стандартом для будь-якої інтернет-платформи. Гнучкий (адаптивний) підхід до проєктування гарантує комфорт взаємодії, незалежно від типу апаратури, з якої здійснюється підключення.

Для втілення цієї настанови у проєкті задіяно бібліотеку Bootstrap версії 5. Вона постачає готову сіткову структуру, адаптивні компоненти та допоміжні інструменти, завдяки чому елементи інтерфейсу автоматично перебудовуються відповідно до поточної ширини дисплея [15]. Цей підхід дав змогу забезпечити

правильне відображення системи як на екранах з великою роздільною здатністю, так і на компактних гаджетах, усуваючи потребу в створенні індивідуальних медіа-запитів від початку.

– Технічні аспекти впровадження. Разом із функціональними критеріями, під час розробки системи враховувалися й суто технічні аспекти – тобто ліміти та умови, обумовлені вибором технологічного набору інструментів та оточення для життєдіяльності проєкту. Бекенд платформи був створений на базі PHP, що є однією з найбільш уживаних мов у сфері веб-програмування та добре підходить для конструювання динамічних веб-ресурсів із обробкою запитів на стороні сервера. Фронтенд використовує JavaScript для втілення інтерактивних елементів інтерфейсу та Bootstrap 5 для гарантування відповідності вигляду на різних пристроях. Для зберігання даних застосовується реляційна база даних MySQL, яка гарантує стабільне зберігання та швидке витягування впорядкованих відомостей, таких як дописи, категорії, коментарі та профілі користувачів [16].

Розгортання системи в роботу відбувається у середовищі, організованому за допомогою контейнеризації Docker. Такий підхід дає змогу оперувати платформою у відокремленому середовищі, незалежному від операційної системи чи налаштувань машини-хоста, що значно полегшує конфігурування та міграцію розробки між різними серверами [17].

Визначені функціональні та технічні вимоги охоплюють усі ключові аспекти функціонування платформи й є фундаментом для її подальшого архітектурного опрацювання та кодування.

1.4 Висновки до першого розділу

У першому розділі проведено аналіз предметної області та визначено загальні напрями розробки веб-орієнтованої інформаційної системи управління контентом блог-платформи, орієнтовану на IT-спільноту.

Було з'ясовано, що профільні блогові сервіси для IT-фахівців становлять значущий засіб для поширення технічних відомостей. Інтенсивне технологічне зростання спричиняє стійкий інтерес до спеціалізованих порталів, де розробникам та тим, хто навчається IT-дисциплін, була б надана можливість ділитися набутим досвідом, оприлюднювати навчальні матеріали та проводити технічний аналіз.

Проведено зіставлення вже існуючих рішень — Dev.to, Hashnode, WordPress та Ghost. Встановлено, що кожна з платформ має свої сильні боки, проте жодна не гарантує ідеального поєднання логічної організації матеріалів, легкості конфігурування та здатності до повного управління системними функціями. Цей факт обґрунтовує необхідність розробки власного продукту, сфокусованого на виправленні помічених недоліків.

Визначено функціональні вимоги, які система мусить задовольняти. Вони охоплюють управління публікаціями, механізми реєстрації та підтвердження особистості користувачів, можливість залишати коментарі, наявність панелі керування для адміністратора і забезпечення коректного вигляду на різноманітних пристроях. Зазначені вимоги є достатніми умовами для належного функціонування платформи згідно з потребами цільової групи.

Отримані висновки створюють необхідну базу для вибору технологічних інструментів для розробки та подальшого архітектурного планування системи, що буде розглянуто у наступних розділах цієї роботи.

РОЗДІЛ 2. ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РОЗРОБКИ СИСТЕМИ

2.1 Засоби серверної розробки: PHP

PHP (Hypertext Preprocessor) — це інтерпретована скриптова мова програмування широкого вжитку, була призначена переважно для генерації динамічних вебсторінок та інтернет-застосунків. Ця мова функціонує на серверному боці: коли надходить HTTP-запит від користувача (клієнта), сервер здійснює виконання PHP-скрипт, формує відповідь у форматі HTML та передає її браузеру [18]. Таким чином, PHP є фундаментальним елементом архітектури "клієнт-сервер", де уся робота з обробки суті бізнес-задач залишається невидимою для того, хто кінцево користується продуктом.

Мову було створено Расмус Лердорф, який започаткував її у 1994 році як набір утиліт для моніторингу трафіку на власному сайті. Протягом часу PHP розвинувся до статусу повноцінної об'єктно-орієнтованої мови з розвиненим оточенням (екосистемою). Як вказано у офіційній документації , на цей день PHP належить до мов, що домінують у сфері серверної веб-розробки по всьому світу, його підтримує більша частка постачальників хостингових послуг і він є невід'ємною частиною основного набору LAMP (Linux, Apache, MySQL, PHP) [18]. Зважаючи на легкість доступу та простоту впровадження, PHP історично позиціонується як першочерговий інструмент для створення веб-платформ, систем керування контентом та блогінгових рішень.

Серед основних рис PHP, які зумовлюють його активне використання у бекенд-розробці, слід виділити наступні аспекти. Насамперед, ця мова пропонує вбудовані інструменти для опрацювання HTTP-запитів із форм, управління сесіями та куками, що уможливлює впровадження механізмів входу й прав доступу користувачів без залучення зовнішніх модулів [20]. По-друге, PHP володіє багатогранним API для взаємодії з реляційними сховищами даних: зокрема, розширення MySQLi та PDO гарантують надійне з'єднання з MySQL та захист від атак SQL-ін'єкцій завдяки застосуванню підготовлених виразів

(prepared statements). По-третє, наявність вбудованих функцій для роботи з регулярними виразами, текстовими даними, файловою інфраструктурою та мережею робить PHP самодостатнім середовищем, здатним реалізувати серверні бізнес-правила без необхідності імпорту додаткових сторонніх пакетів.

Від версії PHP 5 мова отримала цілковиту підтримку об'єктно-орієнтованого підходу: запроваджено класи, інтерфейси, абстрактні класи, трейти, зони видимості (простори імен) та механізми ловіння помилок (обробка винятків) [21]. Версія PHP 7 суттєво поліпшила швидкість роботи завдяки новому рушію Zend Engine 3 та впровадженню декларування суворих типів (scalar type declarations). Найновіша гілка, PHP 8, представила можливість використання іменованих параметрів, переліків (enums), метаданих (атрибутів) та компілятора JIT, що додатково оптимізує швидкість виконання скриптів. Завдяки цим нововведенням сучасний PHP дозволяє створювати код з гарною архітектурою, надійним типом і можливістю легкого тестування та подальшого розширення [22].

Однією з ключових переваг PHP слугує його вже сформована екосистема. Фактично, усі провідні інструменти та CMS для створення вебдодатків базуються саме на цій мові: йдеться про такі системи, як Laravel, Symfony, а також платформи WordPress, Drupal та Joomla [19]. Цей факт означає, що розробнику відкритий доступ до величезного арсеналу готових рішень, бібліотек та навчальних матеріалів, документації. Менеджер залежностей Composer надає можливість декларативного управління пакетами та їхніми версіями через файл composer.json, що значно спрощує супровід проєкту. Для платформ, призначених для ведення блогів, особливо тих, що орієнтовані на IT-спільноту, PHP є логічним вибором: він забезпечує легку взаємодію з MySQL, зручний механізм формування HTML-відповідей та вбудовану функціональність для завантаження файлів, усе потрібне для ефективного управління контентом публікацій [19].

Щоб обґрунтувати вибір PHP як технології для серверної частини у цьому проекті, варто провести порівняння з Python — мовою, що також активно

використовується у серверній розробці та часто розглядається як її можлива заміна.

Python — це високорівнева мова програмування загального призначення, що вирізняється мінімалістичним і чітким синтаксисом, який полегшує читання коду. Вона підтримує принципи модульності та багаторазового використання коду: наприклад, елемент форми, написаний один раз, може бути задіяний на різних сторінках застосунку без необхідності його дублювання. Python відомий своєю доступністю для опанування навіть новачками, а також потужною підтримкою інструментів для автоматизації рутинних завдань. Ця мова є справді універсальною, знаходячи широке застосування у великих комерційних проєктах, системах аналізу даних, застосунках машинного навчання та штучного інтелекту. Серед її вагомих плюсів, є висока продуктивність виконання коду, чиста програмна структура та здатність до злиття з іншими мовами (такими як Java, C, C++). У сфері веброзробки Python пропонує такі фреймворки, як Django та Flask, однак жоден із них не є складовою самої мови, вони потребують окремого встановлення та конфігурування [23].

Незважаючи на очевидні переваги Python, при вирішенні про серверний стек для блогу, що оперує даними MySQL, необхідно зважати на фундаментальні відмінності між цими двома мовами. Насамперед, PHP створювалася суто для вебу, забезпечуючи вбудовану підтримку усіх необхідних HTTP-протоколів, тоді як Python для обробки навіть найпростіших запитів вимагає залучення додаткового програмного фреймворку. По-друге, PHP безпосередньо та природно зв'язується з MySQL завдяки вбудованим розширенням, тоді як для взаємодії Python з реляційними базами даних потрібні зовнішні залежності, як-от PyMySQL чи SQLAlchemy. По-третє, полегшення доступу до PHP у хостингу є значним плюсом: більшість стандартних планів спільного хостингу вже включають PHP із початковими налаштуваннями, тоді як розгортання програм на Python зазвичай тягне за собою необхідність оренди виділеного сервера чи налаштування WSGI-сумісного сервера. В умовах, коли блог-платформа має бути легко відтворюваною та нескладною в експлуатації, це слугує вагомим

практичним фактором. Насамкінець, справжня потужність Python розкривається в завданнях, пов'язаних з аналізом великих обсягів інформації, машинним навчанням та високопродуктивними науковими розрахунками, тобто там, де ці функції дійсно незамінні [19]. Для типової інтернет-платформи, головний фокус якої полягає у керуванні контентом, автентифікації користувачів та взаємодії з базою даних, це означає зайве обтяження функціоналом без реальної користі.

Огляд ключових аспектів порівняння PHP та Python у сфері розробки серверної частини вебзастосунків представлено у Таб. 2.1.

Критерій	**PHP**	**Python**
Основне призначення	Серверна веброзробка	Загальне призначення, ML, наука про дані
Вбудована веб-інтеграція	Нативна (HTTP, сесії, форми)	Потребує фреймворку (Django, Flask)
Інтеграція з MySQL	Вбудована (MySQLi, PDO)	Через зовнішні бібліотеки (PyMySQL, SQLAlchemy)
Хостинг-підтримка	Підтримується більшістю хостингів	Потребує VPS або спеціалізованого хостингу
CMS-екосистема	WordPress, Drupal, Joomla (великий вибір)	Обмежена (Django CMS)
Продуктивність (PHP 8 / JIT)	Висока для вебзапитів	Висока, але переваги — в обчисленнях
Складність вивчення для веброзробки	Низька — проектувалась для вебу	Помірна — потребує вивчення фреймворку

Таблиця 2.1 — Порівняння PHP та Python для серверної веброзробки

Згідно з даними, наведеними у Таблиці 2.1, PHP демонструє більшу спрямованість та природність у контексті веб-розробки. Натомість Python має переваги у тих проєктах, де критично важливим є обробка даних з великими обсягами або впровадження алгоритмів машинного навчання [23]. З огляду на те, що цільова аудиторія блог-платформи – це технічна IT-спільнота, ключовими вимогами стають легкість управління контентом, функціонал авторизації користувачів та стійка робота з реляційними базами даних. Саме за цими параметрами PHP виявляється найбільш вдалим вибором.

Необхідно також підкреслити аспекти безпеки, притаманні PHP. Актуальні версії цієї мови програмування включають нативні інструменти для протидії типовим загрозам у веб-середовищі: передбачені механізми для валідації та очищення вхідних відомостей (наприклад, `filter_input`, `filter_var`), надійне хешування паролів за допомогою bcrypt через пари функцій `password_hash`/`password_verify`, а також захист від міжсайтового підроблення запитів (CSRF) завдяки функціоналу сесій. Комбінація цих вбудованих засобів із коректним застосуванням підготовлених виразів PDO дає змогу сформувати надійний щит для захисту особистих даних користувачів ресурсу[21].

Рішення обрати PHP як основний стек для серверної частини розробки цієї блог-платформи грунтується на комплексі факторів: легка сумісність із MySQL, вбудована підтримка всіх необхідних функціональних елементів для вебу (управління сесіями, реєстрація/вхід, завантаження файлів), широка база готових рішень та бібліотек (включно з CMS), висока адаптивність до різних хостинг-середовищ та підтверджена ефективність у розробці аналогічних проєктів. Використання сучасного PHP 8 сприяє написанню чистого коду, реалізованого на принципах ООП, що цілком відповідає критеріям, висунутим до програмного продукту, який має академічне значення [18][22].

2.2 Засоби клієнтської розробки: JavaScript та Bootstrap 5

Фронтенд нашого проєкту блог-платформи побудована із застосуванням двох взаємодоповнюючих технологій: мови JavaScript та CSS-фреймворку Bootstrap 5. Кожна з них несе свою частину відповідальності за вигляд сторінки: JavaScript опікується динамікою та взаємодією користувача, тоді як Bootstrap забезпечує стильове каркасу та коректне відображення на різних девайсах.

JavaScript, будучи інтерпретованою мовою з динамічною системою типів, спершу була задумана лише для роботи у браузерах, але згодом її сфера застосування суттєво розширилась [25]. Нині це єдина мова, яку браузери здатні виконати напряму, що закріплює її позиції як незамінну складову у створенні інтерактивних рішень для мережі.

Мова підтримує одночасне використання кількох парадигм: функціональної та об'єктно-орієнтованої, що надає велику маневреність у вирішенні різнорідних завдань [26]. Виконання коду відбувається прямо у вікні браузера без потреби попередньої компіляції, що спрощує етапи розробки та тестування: будь-які правки видно миттєво після оновлення вмісту сторінки.

Важливою рисою цієї мови є те, що функції трактуються як об'єкти першого класу: їх можна передавати як аргументи, повертати як результат роботи інших функцій та призначати змінним. Це відкриває двері до потужних можливостей функціонального програмування. Окрім того, JavaScript має потужну реалізацію для роботи з асинхронністю завдяки інструментам Promise та async/await, що дає змогу організовувати неблокуючий обмін даними із сервером, не примушуючи користувача очікувати повного перезавантаження сторінки [24].

Виняткова гнучкість мови JavaScript лежить в основі її широкого поширення. Завдяки інтеграції Node.js та розмаїттю фреймворків, як-от React, Angular, Vue, а також численних бібліотек, один і той самий інструмент охоплює як фронтенд, так і бекенд розробку [27]. Аналізуючи щорічні дослідження Stack

Overflow, помітно, що JavaScript протягом останніх понад десять років незмінно утримує лідерські позиції серед найбільш затребуваних мов програмування.

Водночас, у процесі створення програм на JavaScript необхідно усвідомлювати ряд його специфічних рис. Динамічна типізація, хоча й сприяє швидкому написанню логіки, водночас несе ризик появи прихованих дефектів, якщо розробник не підтримує належну строгість у поводженні з типами [27]. Також варто пам'ятати, що можуть існувати несуттєві розбіжності у функціонуванні певних програмних інтерфейсів (API) між різними браузерами, що зумовлює необхідність тестування сумісності під час роботи [26]. Попри це, наявність деталізованої довідкової літератури, зокрема ресурсів MDN Web Docs та The Modern JavaScript Tutorial, значно полегшує старт роботи та забезпечує оперативне знаходження рішень для робочих завдань [24][25].

Bootstrap являє собою відкритий CSS-каркас, який пропонує розробникам вже підготовлений арсенал оформлень, елементів інтерфейсу та допоміжних засобів для створення гнучких (адаптивних) веб-сторінок. П'ята версія цього каркасу, що була актуальною під час створення проєкту, продемонструвала суттєвий прогрес, якщо порівнювати її з попередніми версіями.

Фундаментальною ідеєю, закладеною у проєктування Bootstrap, є концепція "спочатку мобільні" (mobile first). Це означає, що стандартні стилі налаштовані насамперед для портативних ґаджетів, а розширення видимості до більших дисплеїв реалізується завдяки механізму контрольних точок (breakpoints). Такий підхід разюче контрастує зі застарілою методологією, коли макет спершу верстали для стаціонарних ПК, а потім "підганяли" його під менші розміри. Внаслідок цього рішення, інтерфейси на базі Bootstrap забезпечують належне відображення абсолютно на будь-якому пристрої — від мобільного телефону до монітора з великим екраном.

Базою для структурування контенту слугує система сітки (grid system), збудована на технології CSS Flexbox, яка умовно розділяє простір сторінки на дванадцять вертикальних смуг. Програміст вказує, скільки таких смуг займатиме певний фрагмент залежно від поточної ширини дисплея, а Bootstrap сам

організовує перекомпонування [15]. Окрім власне сітки, цей каркас містить велику добірку готових елементів: панелей для навігації, блоків-карток, полів для введення даних, кнопок, спливаючих вікон, систем розбиття на сторінки, іконок тощо. Усі ці складові вже мають єдиний візуальний стиль і не потребують ручного прописування CSS-правил з нуля.

П'ятка версія Bootstrap внесла низку вагомих оновлень. Ключовою серед них стала повна ліквідація залежності від jQuery: усі динамічні елементи тепер функціонують на нативному JavaScript, що тягне за собою зменшення кількості необхідних підключень. Крім того, у Bootstrap 5 реалізовано розширену роботу з CSS-змінними, завдяки чому зміна колірної схеми та інших аспектів стає значно легшою, не вимагаючи редагування первинного коду фреймворку. А оновлений API для утиліт надає змогу гнучко створювати персоналізовані класи-помічники без необхідності писати окремі CSS-правила.

Популярність Bootstrap серед розробників зумовлена насамперед помітним прискоренням процесу творення веб-сторінок: стандартні інтерфейсні рішення досягаються завдяки вже готовим компонентам та стильовим класам, замість ручного прописування оформлення [28]. Це особливо цінно в тих розробках, де головним є роботоздатність, а не унікальність візуального оформлення. Водночас, Bootstrap зовсім не накладає обмежень на індивідуалізацію: за бажанням, будь-який елемент можна перевизначити, застосувавши власні стилі CSS.

Зв'язок між JavaScript та Bootstrap 5 є доволі органічним. Деякі елементи Bootstrap: а саме, модальні вікна, меню, що випадають, та спливаючі підказки функціонують завдяки JavaScript, і для цього сам фреймворк пропонує свій власний збірний JS-файл [15]. Водночас, програмний код на JavaScript, що належить застосунку, має можливість взаємодіяти з елементами Bootstrap або через їхній програмний інтерфейс (API), або шляхом додавання/видалення CSS-класів.

Такий розподіл функцій є загальноприйнятим підходом у сучасній розробці на фронтенді: Bootstrap відповідає за каркас та візуальне оформлення,

тоді як JavaScript займається логікою роботи та динамічними змінами у відображенні. Комплексне використання цих двох технологій допомагає значно зменшити обсяг коду, який потрібно написати, гарантує узгоджену роботу в різних браузерах та дає можливість створити адаптивний інтерфейс без необхідності підключати додаткові програмні платформи [24][25][28].

2.3 Засоби зберігання даних: MySQL

MySQL являє собою систему керування базами даних реляційного типу з відкритим вихідним кодом, яка наразі зайняла одну з найміцніших позицій у сфері веб-розробки. Відлік історії цього програмного забезпечення розпочався у 1995 році, коли шведська фірма MySQL AB представила громадськості його перший реліз. Назва продукту утворилася з імені доньки одного із засновників – My – та загальновідомого скорочення SQL. У 2008 році відбулася подія: Sun Microsystems викупила MySQL AB, а вже у 2010 році, разом із Sun, контроль над системою перебрала Oracle Corporation. Попри зміну власників та певні занепокоєння серед спільноти щодо подальшого статусу відкритості проєкту, безкоштовна версія MySQL Community Edition не тільки збереглася, але й активно розвивається дотепер.

Базою для побудови системи слугує реляційна модель даних, концепцію якої розробив Едгар Кодд ще у 1970 році. Основний принцип цієї моделі полягає у організації усіх даних у форматі двовимірних таблиць, які називають відношеннями. Кожна така таблиця складається з вертикальних позицій (стовпців чи атрибутів) та горизонтальних записів (рядків). Зв'язки між цими таблицями формуються завдяки використанню ключових полів: первинний ключ однієї таблиці може слугувати зовнішнім ключем в іншій, що дає змогу створювати складні структури даних без необхідності їхнього надмірного копіювання. Саме ця структурованість, що відповідає принципам нормалізації, гарантує збереження цілісності та узгодженості даних у рамках системи.

Робота з MySQL ведеться за допомогою мови SQL — Structured Query Language. Це мова декларативного типу, де фахівець вказує, які саме відомості йому потрібні, а не описує послідовність кроків для їх отримання. SQL є мовою, стандартизованою комітетами ISO та ANSI, внаслідок чого фундаментальні команди залишаються незмінними у переважній більшості систем керування реляційними базами даних (СКБД). MySQL володіє підтримкою усіх ключових операторів: SELECT призначений для зчитування інформації, INSERT — для введення нових записів, UPDATE — для редагування наявних, DELETE — для стирання даних, окрім того, присутні операції DDL для формування структури таблиць, як-от CREATE TABLE, ALTER TABLE, DROP TABLE.

З погляду архітектури, MySQL функціонує за усталеною клієнт-серверною моделлю. Серверний компонент MySQL являє собою окремий процес, який працює постійно у фоновому режимі; він чекає на підключення від клієнтських програм через визначений мережевий порт та виконує отримані запити. Роль клієнта може виконувати як утиліта командного рядка mysql, так і будь-яка програма, що задіює відповідний програмний інтерфейс (драйвер). Якщо йдеться про веб-рішення, клієнтом виступає код, написаний на PHP: він ініціює зв'язок із сервером бази даних MySQL, передає SQL-виконавчі команди та отримує у відповідь набори даних, які потім оформлюються і демонструються у вікні браузера кінцевого користувача. Така конфігурація ідеально відповідає звичній тришаровій структурі веб-додатків: рівень відображення (HTML у браузері) — рівень обробки бізнес-логіки (PHP) — рівень зберігання інформації (MySQL) [16].

Однією з важливих характеристик архітектури MySQL є здатність підтримувати різні механізми (рушії) зберігання даних. Технічно, кожна таблиця потенційно може використовувати свій унікальний рушій, хоча на практиці зазвичай для усієї бази обирається один домінуючий механізм. Найбільш значущим і широко вживаним є рушій InnoDB, який став стандартним, починаючи з версії MySQL 5.5. Саме InnoDB надає можливість реалізації транзакцій, керування зовнішніми ключами та механізмів блокування на рівні

окремих рядків, що робить його найкращим вибором для складних, критично важливих комерційних застосувань.

Транзакції у всьому їхньому виконанні в InnoDB підпорядковуються принципам ACID — це комплекс критеріїв, що засвідчує надійність усіх операцій із даними. Атомарність диктує, що транзакція або виконується стовідсотково, або ж зовсім не починається: не буває жодних напівзаходів. У випадку будь-якого збою під час виконання низки дій, усі зроблені корективи автоматично повертаються до свого первісного стану. Узгодженість (Consistency) гарантує, що по завершенню операції база даних переходить лише у валідний стан, який не порушує жодних попередньо встановлених обмежень. Ізоляція гарантує, що одночасно запущені транзакції не бачать одна одної на етапі незавершеного виконання. Довговічність (Durability) означає, що після фінального підтвердження (COMMIT) усі зміни фіксуються, і вони залишаться незмінними навіть після припинення живлення чи краху системи.

Система зовнішніх ключів дає змогу явно прописати взаємозв'язки між різними таблицями, перекладаючи обов'язок моніторингу їхнього дотримання на саму СУБД. Наприклад, якщо у таблиці з публікаціями присутній стовпець для ідентифікатора автора, зовнішній ключ забезпечить, що цей ID завжди відповідатиме дійсному користувачу в таблиці з користувачами. Якщо ж буде зроблена спроба видалити користувача, на якого посилаються якісь публікації, СУБД або заблокує цю дію, або ж виконає каскадне видалення всіх пов'язаних записів — залежно від заздалегідь визначеного правила (це може бути RESTRICT, CASCADE або SET NULL).

Індексація є ще одним життєво важливим інструментом у MySQL, що прямо впливає на швидкість виконання запитів. Індекс — це спеціальна допоміжна структура, розроблена для пошуку потрібних записів без необхідності послідовного перегляду всієї таблиці. Первинний ключ автоматично отримує індексацію, що забезпечує надшвидкий пошук за його значенням. Для полів, які часто залучаються у ваші запити на фільтрацію чи об'єднання (наприклад, 'slug' для статті, 'email' для користувача, або категорійний

ID), можна створювати додаткові індекси. Без використання індексів кожен пошуковий запит змушений опрацьовувати кожен рядок таблиці, що катастрофічно знижує продуктивність зі зростанням обсягу бази даних [30].

У MySQL реалізовано великий арсенал засобів для представлення даних: від цілочисельних типів різних габаритів (таких як TINYINT, INT, BIGINT) до чисел з рухомою комою (FLOAT, DOUBLE, DECIMAL). Крім того, є опції для рядків фіксованої та змінної довжини (CHAR, VARCHAR), великих обсягів тексту (TEXT, MEDIUMTEXT, LONGTEXT), часових міток (DATE, TIME, DATETIME, TIMESTAMP) та бінарних масивів (BLOB). Умілий підбір відповідного формату для кожної колонки сприяє економії місця на диску та покращує швидкість роботи індексів [16].

PostgreSQL являє собою систему керування базами даних (СКБД) з відкритим кодом, що належить до об'єктно-реляційного типу. Її розробка здійснюється незалежною спільнотою розробників з 1986 року, а сама СУБД є прямим нащадком дослідницької роботи POSTGRES, що проводилася в Берклі (Каліфорнійський університет). Ключова відмінність від MySQL полягає в тому, що PostgreSQL від початкових етапів розробки орієнтувалася на жорстке дотримання стандартів SQL та високу розширюваність.

Спектр вбудованих типів даних у PostgreSQL є більш насиченим, ніж у MySQL: тут є підтримка масивів, діапазонних типів, ідентифікаторів UUID, сховища ключ-значення hstore, і, що особливо важливо, формату JSONB — індексованого варіанту JSON у бінарному вигляді. Наявність JSONB дозволяє PostgreSQL ефективно оперувати напівструктурованими даними, об'єднуючи переваги як реляційних, так і документо-орієнтованих підходів. Додатково, PostgreSQL надає нативну підтримку геопросторових даних через розширення PostGIS, що робить її ідеальним рішенням для додатків, пов'язаних із картографією та геоінформаційних застосунків.

Щодо обробки запитів, PostgreSQL розглядається як більш потужний інструмент для комплексних аналітичних завдань. Його планувальник запитів перебуває на вищому рівні, ефективніше обробляючи з'єднання багатьох таблиць

та надаючи можливість паралельного опрацювання запитів на багатоядерних архітектурах. Вбудований повнотекстовий пошук у PostgreSQL також демонструє значно більшу гнучкість та функціональність. Додатково, PostgreSQL демонструє більш суворе дотримання стандартів SQL, коректніше обробляючи певні граничні ситуації, де MySQL може відхилятися від стандарту.

Однак, з практичного погляду, PostgreSQL має й певні мінуси. Початкове розгортання та подальше адміністрування є більш заплутаним. Обсяг доступних навчальних ресурсів та готових рішень, особливо у сфері PHP-розробки, є меншим, ніж у випадку з MySQL. Для фахівця, який вперше знайомиться із СУБД у веб-середовищі, PostgreSQL вимагатиме глибшого занурення у тонкощі налаштування та архітектури системи.

Зіставлення систем є перевіреними часом, стабільними та безкоштовними рішеннями, які підтримуються великими спільнотами. Кінцевий вибір між ними визначається специфічними потребами проєкту [29]. Нижче подано порівняння за важливими аспектами MySQL та PostgreSQL у Таб. 2.2.

Критерій	**MySQL**	**PostgreSQL**
Тип системи	Реляційна СУБД	Об'єктно-реляційна СУБД
Відкритий код	Так (Community Edition)	Так (повністю відкрита)
Дотримання стандарту SQL	Часткове	Суворе
Типи даних	Стандартні реляційні	Розширені (масиви, JSONB, UUID, геодані)
Продуктивність на читання	Дуже висока	Висока
Продуктивність запису	Висока	Висока
Поріг входження	Низький	Середній / Високий

Інтеграція з PHP	Відмінна, багато прикладів	Добра, менше прикладів
Розгортання у Docker	Просте	Просте
Паралельне виконання запитів	Обмежено	Так, повноцінна підтримка
Підтримка JSON	Базова	Повноцінна (JSONB з індексуванням)

Таблиця 2.2 — Порівняння MySQL та PostgreSQL засобів зберігання даних

Здійснивши порівняльний аналіз обох систем, можна дійти висновку, що MySQL є більш обґрунтованим рішенням для новоствореної блог-платформи. Розглянемо аргументацію детальніше.

Модель даних, притаманна цій платформі, вирізняється відносною простотою та ідеально вписується в традиційну реляційну модель: нам потрібні таблиці для обліку користувачів, дописів, тематичних постів та коментарів. Усі взаємозв'язки між цими сутностями відповідають загальноприйнятим зв'язкам типу «один-до-багатьох». Специфічні можливості PostgreSQL, як-от робота з масивами, форматом JSONB чи геопросторовими даними, у межах цього проєкту не є необхідними. Таким чином, наявні технічні переваги PostgreSQL у цьому конкретному випадку не мають де проявитися [29].

Функціонал, який виконуватиме платформа, обмежується стандартним набором операцій CRUD: створення, отримання, зміна та видалення записів. Складні аналітичні запити, потреба в агрегації значних обсягів інформації чи високопаралельна обробка даних не плануються. З огляду на характер очікуваного навантаження, MySQL демонструє чудові показники швидкодії та повністю закриває всі потреби системи [16].

Ключовою практичною перевагою є те, що комбінація PHP та MySQL являє собою класичним стеком технологій для веб-розробки, який накопичив багаторічний досвід. Завдяки цьому існує незліченна кількість задокументованих рішень, типових прикладів, готових бібліотек та допоміжних інструментів. Коли виникає будь-яке технічне питання, знайти розв'язання у довідковій літературі чи серед спільноти набагато легше, ніж у ситуації з PostgreSQL, якщо мова йде про використання його в оточенні PHP. Для навчального проєкту це має першочергове значення, адже швидкість подолання труднощів прямо корелює зі швидкістю самого процесу створення продукту [29].

На додаток, запуск MySQL у середовищі Docker є процесом, який детально описаний і не викликає складнощів. Офіційний образ mysql:8.0 вільно доступний на Docker Hub, він адекватно реагує на налаштування через змінні оточення та без проблем вбудовується у конфігурацію docker-compose поряд із контейнером, де працює PHP [16].

Підводячи підсумки порівняння двох флагманських реляційних систем керування базами даних (СКБД): MySQL та PostgreSQL — для втілення частини, що відповідає за збереження інформації блог-платформи, вибір було зроблено на користь MySQL. Ця система цілком задовольняє всі функціональні потреби розробки: вона гарантує стабільне реляційне зберігання даних, забезпечує підтримку транзакцій з властивостями ACID, дозволяє використовувати механізми зовнішніх зв'язків та індексації, природно взаємодіє з PHP завдяки розширенню PDO та легко розгортається у відповідному Docker-контейнері.

Хоча PostgreSQL, безперечно, має вагомі технічні козирі в арсеналі завдяки розширеним можливостям типів даних, потужним інструментам для аналізу даних та неухильному дотриманню вимог стандарту SQL, він виявляється надмірним вибором для проєкту, що оперує простою структурою даних та не вимагає складних функціональних рішень. Своєю чергою, підвищена заплутаність у конфігурації та менший обсяг готових рішень, доступних саме для середовища PHP, роблять PostgreSQL менш вигідним для реалізації конкретно цього завдання.

MySQL постає як найбільш раціональне та цілком достатнім засобом сховища даних для реалізованого блог-сервісу. Цей вибір обґрунтовується як ретельним технічним оглядом вимог, поставлених перед проєктом, так і тривалим досвідом успішного застосування цієї системи керування базами даних у веброзробці загалом [16][29][30].

2.4 Засоби розгортання: Docker

Сучасне створення вебресурсів все частіше схиляється до парадигм, які гарантують передбачуваність робочого оточення та не залежать від специфічного обладнання. Серед найактуальніших інструментів, що відповідають цим критеріям, виділяється Docker — платформа для контейнеризації додатків у контейнери. У цьому підпункту ми проаналізуємо суть технології Docker, її ключові функції, а також проведемо зіставлення з іншим популярним рішенням — Kubernetes.

Сутність Docker та його призначення. Процес розробки вебдодатку неминуче пов'язаний з труднощами, що виникають через розбіжності між робочими середовищами: програма, яка ідеально функціонує на особистому комп'ютері, може дати збій на сервері через неузгодженість версій PHP, параметрів бази даних MySQL чи налаштувань вебсервера. Docker пропонує радикальне вирішення цієї дилеми, даючи можливість "упакувати" застосунок разом з усією сукупністю його залежностей у контейнери.

Контейнер являє собою відокремлений програмний простір, який інкапсулює все необхідне для безперебійної роботи окремого застосунку чи його компонента: вихідний код, необхідні бібліотеки, змінні середовища та файли конфігурації. На відміну від класичних віртуальних машин, контейнери не займаються симуляцією апаратної частини чи запуском повноцінної гостьової ОС всередині себе; натомість вони використовують ядро операційної системи хоста спільно, що значно підвищує їхню швидкодію та зменшує навантаження.

Основоположним елементом у Docker-середовищі слугує образ (image) — незмінний шаблон, із якого генеруються контейнери. Створення таких образів здійснюється через спеціальний файл, відомий як Dockerfile, де послідовно вказано, яке базове оточення необхідне, які потрібні для встановлення залежності та яку саме команду слід виконати під час старту. Такий механізм інтегрує конфігурацію середовища безпосередньо у вихідний код проєкту, дозволяючи зберігати її у системі керування версіями поруч із самим кодом [17].

Ще одним вагомим інструментарієм є Docker Compose — допоміжна програма, призначена для координування роботи декількох контейнерів у межах однієї робочої одиниці. Застосовуючи єдиний YAML-файл, можна декларувати сукупність сервісів: приміром, вебсервер, середовище виконання PHP, а також сервер для керування базами даних, і запустити їх усі одночасно єдиним викликом. Це значно згладжує процес розгортання застосунків зі складною архітектурою та робить його зрозумілим навіть для фахівців, які вперше стикаються з цим конкретним проєктом.

Серед практичних здобутків використання Docker виділяються такі ключові моменти. Насамперед, це переносимість: середовище, описане одноразово, забезпечить ідентичний запуск на операційних системах Windows, macOS, Linux, а також на хмарних ресурсах. По-друге, ізоляція: кожен контейнер функціонує автономно від інших, що повністю усуває ризик виникнення колізій через різні версії бібліотек між проєктами на одній апаратній платформі. По-третє, швидкість: контейнери ініціалізуються за лічені секунди, тоді як для підняття повноцінної віртуальної машини може знадобитися кілька хвилин. По-четверте, повторюваність результату: наявність Dockerfile гарантує повну відповідність між середовищем розробки та середовищем експлуатації.

Крім того, Docker оперує власною схованкою (реєстром) образів, відомою як Docker Hub, де містяться затверджені образи для поширених технологій, таких як PHP, MySQL, Nginx, Apache та багато інших. Як наслідок, розгортання стандартного набору компонентів для PHP-проєкту займає мінімум часу:

достатньо лише визначити необхідні образи у конфігураційному файлі та згрупувати їх разом за допомогою Docker Compose.

Зіставлення Docker та Kubernetes. Часто в обговореннях контейнеризації згадують як про Docker, так і про Kubernetes, що іноді призводить до помилкового ототожнення цих рішень або їх некоректного протиставлення. Насправді коректно розглядати їх як інструменти, що оперують на різних рівнях абстракції й призначені для виконання різних функцій, хоча обидва є частинами спільної технологічної платформи [31].

Docker — це перш за все інструмент для формування та активації контейнерів. Спрямований на програміста, він ідеально підходить для локального оточення, невеликих розробок та сценаріїв, де перелік сервісів є невеликий і не вимагає заплутаної взаємодії. Docker Compose доводить його можливості до рівня, достатнього для переважної більшості середніх інтернет-продуктів.

Натомість Kubernetes являє собою цілісну платформу для оркестрування контейнерних одиниць, розроблену " Google " і надану у відкритий доступ. Його призначення — керування значною кількістю контейнерів, що розкидані по багатьох серверах або навіть групах дата-центрів. Kubernetes автоматично розподіляє робоче навантаження між вузлами кластера, відновлює роботу контейнерів після збою, динамічно змінює число активних прикладів відповідно до поточної завантаженості й гарантує безперервний реліз без припинення роботи сервісу.

Отже, Kubernetes займається зовсім іншими викликами. Якщо Docker дає відповідь на запитання «яким чином завести цей додаток у контейнері», то Kubernetes відповідає на вимогу «як підтримувати тисячі контейнерів у стані сталої працездатності на великій сукупності серверів».

Проте, за цією потужністю ховається неабияка заплутаність. Конфігурування кластера Kubernetes, створення маніфестів для кожного компонента, опанування таких засадничих понять як Pod, Deployment, Service, Ingress та іншої термінології забирає чимало часу на навчання та практичне

застосування. Для невеликих розробок та команд із обмеженим бюджетом це може видатися надмірним тягарем [31].

Якщо ж говорити про потреби у ресурсах, Docker виходить уперед. Для функціонування Docker може вистачити звичайного ПК або бюджетного віртуального сервера. Натомість Kubernetes у повноцінному виконанні потребує мінімум кількох серверних машин, які мають достатній обсяг оперативної пам'яті та обчислювальної потужності, що значно збільшує витрати на технічне забезпечення.

Існує також відмінність у підході до розгортання. Docker Compose дозволяє описати середовище розробки в одному файлі, що є інтуїтивно зрозумілим для будь-якого програміста. Kubernetes же оперує цілим набором YAML-файлів, які деталізують бажаний стан системи, і вимагають окремого розуміння того, як працюють керуючі механізми та планувальник. Хоча, варто зазначити, у світі Kubernetes є рішення типу Helm, які полегшують роботу з налаштуваннями, але водночас вводять додатковий рівень умовностей.

Слід пам'ятати, що ці два інструменти не виключають один одного. Насправді, Docker зазвичай застосовується для створення та валідації образів, тоді як Kubernetes відповідає за їхнє впровадження у виробничих умовах. Власне, Docker-образи слугують базовими елементами, з якими працює Kubernetes, запускаючи контейнери у кластері. Отже, у серйозних масштабних розробках вони доповнюють функціонал один одного [31].

Для проєктної платформи блогу, створеної у рамках цієї кваліфікаційної роботи, технологія Docker є цілком обґрунтованим і доцільним варіантом. Ця розробка являє собою монолітний вебдодаток, структурований із кількох компонентів: вебсервера, середовища виконання PHP та СУБД MySQL. Жоден із цих елементів не вимагає горизонтального збільшення потужності чи автоматичного розподілу навантаження між кількома вузлами.

Використання Docker у зв'язці з Docker Compose допомагає реалізувати основні потреби: гарантувати однакове середовище як для розробки, так і для продакшену, спростити старт проєкту на новому робочому місці до виконання

єдиної команди, а також зафіксувати всю інфраструктурну конфігурацію безпосередньо у файлах налаштувань. При цьому рівень заплутаності системи залишається прийнятним для інженера-одинака, що критично важливо в умовах обмеженого часового ресурсу на виконання завдання.

У цьому випадку Kubernetes став би надмірним інструментом, оскільки його переваги розкриваються при роботі з розподіленими архітектурами, високим рівнем трафіку та наявністю команд, сфокусованих на DevOps. Більш глибокий розгляд обраної конфігурації та специфіки взаємодії Docker з частинами розробленої платформи буде надано у третьому розділі дослідження.

2.5 Висновки до другого розділу

У межах другого розділу ми зосередилися на виборі та аргументації технологічного арсеналу, необхідного для створення платформи для ведення блогів. Щодо бекенду, було вирішено зупинитися на PHP, враховуючи його природну спорідненість з веб-технологіями та вбудовану функціональність для маніпуляцій із базами даних. Фронтенд формується за допомогою JavaScript у поєднанні з фреймворком Bootstrap версії 5, що гарантує як інтерактивність користувацького інтерфейсу, так і коректне відображення на будь-яких гаджетах.

Для управління даними обрано MySQL, перевірену часом реляційну систему керування базами даних, яка надійно підтримує транзакції та роботу із зовнішніми зв'язками. Цей вибір є логічним з огляду на легкість її інтеграції з PHP, особливо актуально для проєкту, чия модель даних не відзначається надмірною складністю. Процес розгортання інфраструктури забезпечено за допомогою Docker та Docker Compose, що дозволяє створити ізольоване та повністю відтворюване середовище без додаткових ускладнень. Таке поєднання технологічних рішень повністю покриває всі вимоги розробки та закладає міцний фундамент для подальшого розвитку системи.

РОЗДІЛ 3. РОЗРОБКА БЛОГ-ПЛАТФОРМИ

3.1 Проєктування розробки системи

Архітектура цієї системи відповідає стандартній трирівневій моделі "клієнт-сервер". Роль клієнтської частини виконує веб-інтерфейс, розроблений із застосуванням Bootstrap 5 та JavaScript, що гарантує гнучкість відображення на будь-яких пристроях. На другому рівні розташований серверний компонент – застосунок на PHP, відповідальний за прийом HTTP-запитів, виконання бізнес-правил та керування процесами ідентифікації користувачів. Третій шар, рівень сховища, формується системою управління базою даних MySQL, де зберігає дані щодо суб'єктів (користувачів), тем, контенту та зворотного зв'язку (коментарів).

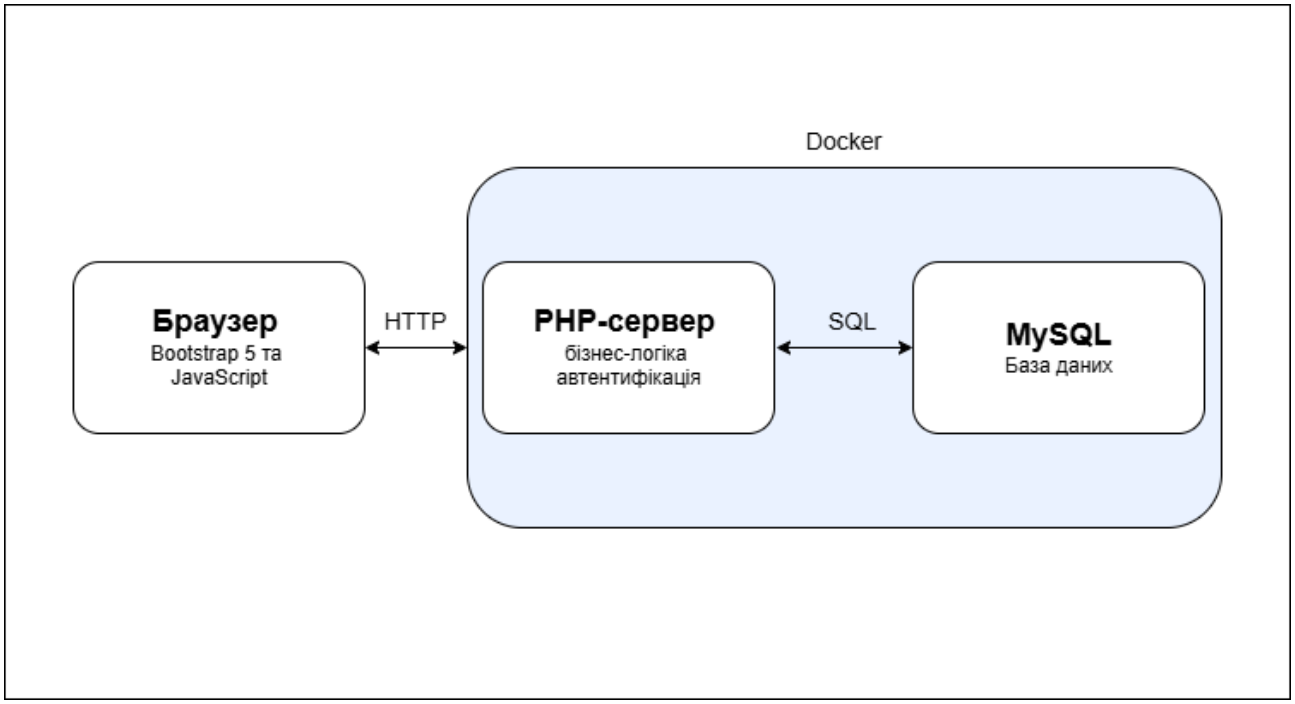

Рисунок 3.1 – Архітектура веб-орієнтованої інформаційної системи

Взаємодія між цими елементами відбувається наступним чином: клієнтський браузер відправляє запити на PHP-сервер, далі сервер генерує відповідні SQL-запити до бази даних MySQL, а потім клієнту повертається

фінальна HTML-сторінка. Для забезпечення повної ізоляції середовища, сталості налаштувань та полегшення процесу імплементації, вся ця інфраструктура розгорнута всередині Docker-контейнерів.

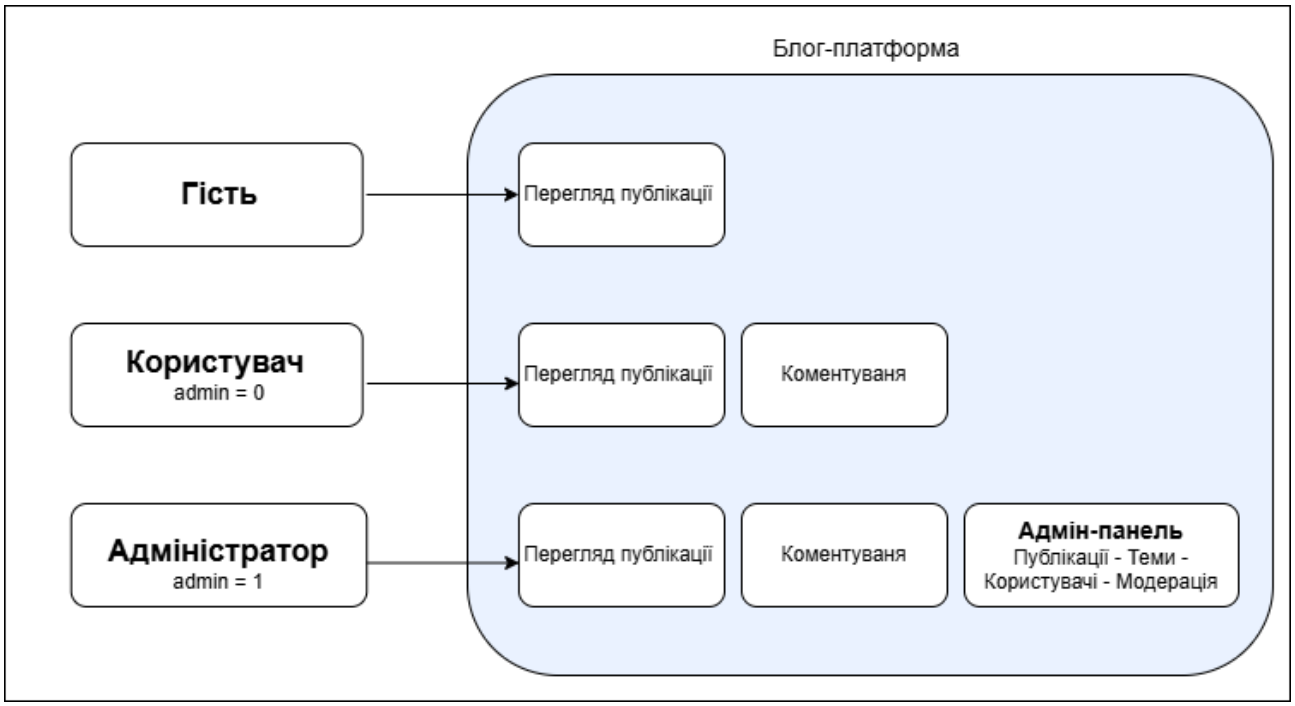

Рисунок 3.2 – Діаграма ролей користувачів платформи

Щодо розмежування прав доступу, система передбачає три рівні взаємодії. Особа, яка не має облікового запису (гість), спроможна лише ознайомлюватися з оприлюдненими даними, не маючи жодного права на публікацію коментарів чи здійснення інших операцій. Користувач, який пройшов реєстрацію та автентифікацію (з показником admin = 0 у базі), отримує можливість як перегляду публікацій, так і залишення власних коментарів. Особі з правами адміністратора (admin = 1) надається додатковий доступ до спеціалізованого інтерфейсу управління, через який вона може контролювати публікації, категорії тем, облікові записи користувачів, а також здійснювати модерацію коментарів. Конкретна структура бази даних буде розглянуто трохи далі.

3.1.1 Структура бази даних MySQL

Платформна база даних була збудована на базі СУБД MySQL і охоплює чотири сутності-таблиці: users, topics, posts та comments [16]. Проєктування виконувалося з дотриманням норм третьої нормальної форми, що не лише зменшує дублювання інформації, але й гарантує коректність взаємозв'язків між основними сутностями.

В таблиці `users` зберігаються дані про тих, хто пройшов реєстрацію. Ідентифікатор `id` слугує первинним ключем із автоматичним зростанням. Поле `admin`, представлене як `TINYINT`, встановлює рівень доступу: нуль (0) позначає стандартного юзера, одиниця (1) — адміністративну особу. Інформація для входу включає `username`, `email` та захешований рядок для `password`. Для реалізації функції відновлення доступу слугують поля `reset_token` та `token_expire`, які можуть залишатися незаповненими (NULL). Структуру таблиці `users` наведено на рис. 3.3.

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id** 🔑	int			Ні	*Немає*		AUTO_INCREMENT
2	**admin**	tinyint			Ні	*Немає*		
3	**username**	varchar(255)	utf8mb4_0900_ai_ci		Ні	*Немає*		
4	**email**	varchar(255)	utf8mb4_0900_ai_ci		Ні	*Немає*		
5	**password**	varchar(255)	utf8mb4_0900_ai_ci		Ні	*Немає*		
6	**created**	timestamp			Ні	CURRENT_TIMESTAMP		DEFAULT_GENERATED
7	**reset_token**	varchar(255)	utf8mb4_0900_ai_ci		Так	*NULL*		
8	**token_expire**	datetime			Так	*NULL*		

Рисунок 3.3 – Структура таблиці `users`.

Сутність `topics` зберігає зведення про різні напрямки чи рубрики, представлені у публікаціях. Кожна рубрика має свій неповторний ідентифікатор `id`, стисле ім'я `name`, а також розширений текст `description` типу `TEXT`. Структуру таблиці `topics` наведено на рис. 3.4.

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int			Ні	*Немає*		AUTO_INCREMENT
2	**name**	varchar(121)	utf8mb4_0900_ai_ci		Ні	*Немає*		
3	**description**	text	utf8mb4_0900_ai_ci		Ні	*Немає*		

Рисунок 3.4 – Структура таблиці `topics`

Центральним елементом цієї системи є сутність, представлена таблицею `posts`. Поле `id_user` слугує зовнішнім ключем, вказуючи на запис у таблиці `users`, тоді як `id_topic` є посиланням на таблицю `topics`. Варто зазначити, що `id_topic` може не мати значення (NULL), якщо допис жодній рубриці не належить. Стан публікації передається через поле `status`, що має тип `TINYINT`; на прикладі, воно може вказувати, чи є матеріал чернеткою, чи вже опублікований. Про шлях до ілюстрації-прев'ю свідчить поле `img`, а основна текстова частина зберігається у полі `content` типу `TEXT`. Структуру таблиці `posts` наведено на рис. 3.5.

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int			Ні	*Немає*		AUTO_INCREMENT
2	**id_user**	int			Ні	*Немає*		
3	**title**	varchar(255)	utf8mb4_0900_ai_ci		Ні	*Немає*		
4	**img**	varchar(255)	utf8mb4_0900_ai_ci		Ні	*Немає*		
5	**content**	text	utf8mb4_0900_ai_ci		Ні	*Немає*		
6	**status**	tinyint			Ні	*Немає*		
7	**id_topic**	int			Так	*NULL*		
8	**created_date**	datetime			Ні	CURRENT_TIMESTAMP		DEFAULT_GENERATED

Рисунок 3.5 – Структура таблиці `posts`

Для реалізації багаторівневої системи відповідей слугує таблиця `comments`. Поле `page` фіксує, до якого саме допису приєднано коментар. Автор коментаря ідентифікується через поле `user_id`. Структура відповідей підтримується завдяки полю `parent_id`, яке може бути відсутнім (NULL);

наявність значення тут означає, що цей коментар є відповіддю на інший. Модерація коментарів здійснюється через поле `status`, де, наприклад, нульове значення (0) вказує на те, що коментар потребує перегляду або наразі є прихованим. Структуру таблиці `comments` наведено на рис. 3.6.

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**id**	int			Ні	*Немає*		AUTO_INCREMENT
2	**status**	tinyint			Ні	0		
3	**page**	int			Ні	*Немає*		
4	**user_id**	int			Ні	*Немає*		
5	**parent_id**	int			Так	*NULL*		
6	**comment**	text	utf8mb4_0900_ai_ci		Ні	*Немає*		
7	**created_date**	datetime			Ні	CURRENT_TIMESTAMP		DEFAULT_GENERATED

Рисунок 3.6 – Структура таблиці `comments`

Відносини між таблицями побудовані за моделлю "один проти багатьох" [16]: множина дописів може належати одному користувачеві (зв'язок `users` до `posts`); так само, багато дописів можуть бути згруповані в одній темі (`topics` до `posts`). Кожен окремий допис допускає існування численних коментарів (`posts` до `comments`), і так само, один і той самий користувач може залишити безліч коментарів (`users` до `comments`). Схема зв’язків наведена на рис. 3.7.

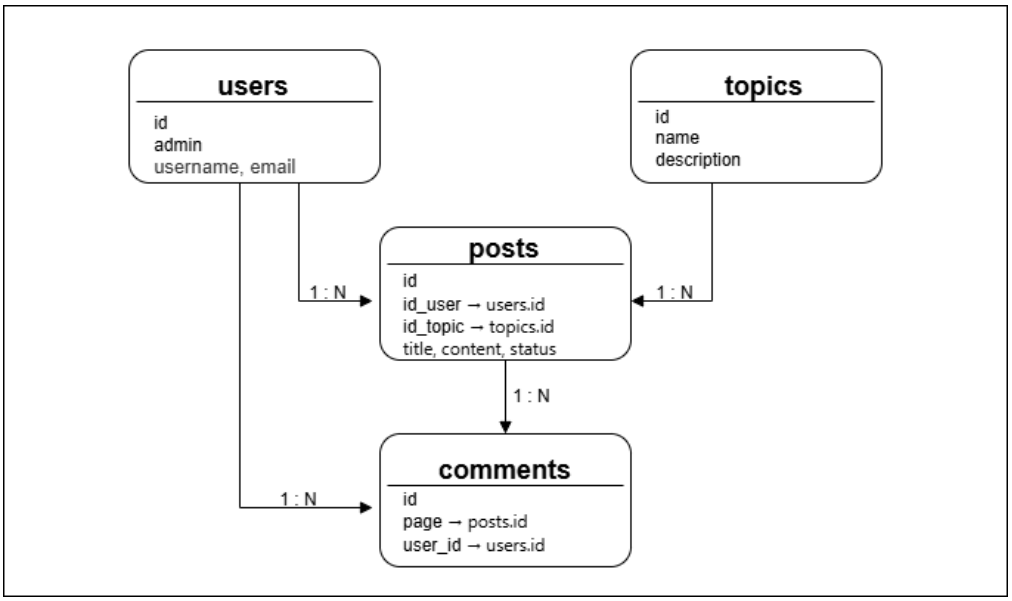

Рисунок 3.7 – Схема зв'язків між таблицями бази даних

База даних, що була розроблена, дає змогу платформі працювати у повному обсязі: вона зберігає облікові дані користувачів, організовує публікації за відповідними тематичними категоріями, а також забезпечує функціональність для багатошарової системи коментарів, включаючи можливість модераторського втручання. Забезпечення відповідності Третій нормальній формі та ретельно продумані зв'язки «один-до-багатьох» між окремими таблицями слугують запорукою як цілісності внесених даних, так і загальної масштабованості усієї системи.

3.1.2 Реалізація адміністративної панелі

Для забезпечення доступу до функціоналу, адаптованого під кожного окремого користувача блог-платформи, були запроваджені механізми реєстраційних процедур та входу в систему. Ці етапи критично важливі для підтримки заходів безпеки та однозначної ідентифікації особи, яка взаємодіє із сервісом. Реалізацію механізму авторизації та реєстрації користувачів наведено у додатку Б (лістинг Б.1).

Процедура реєстрації активується через відповідний інтерфейс на вебсторінці, де користувач має заповнити поля для визначення свого імені (логіна), адреси електронної пошти та секретного коду (пароля) із обов'язковим повторенням останнього. Під час обробки наданої інформації здійснюється її верифікація: перевіряється заповненість усіх обов'язкових полів, мінімальна прийнятна довжина логіна, а також точна відповідність двох записів пароля. Додатково проводиться аналіз бази даних для гарантування унікальності вказаної електронної адреси. Якщо всі критерії валідації виконано успішно, пароль користувача трансформується у захищений хеш, використовуючи вбудовані інструменти мови програмування, після чого запис фіксується у

сховищі даних. Завершення реєстрації автоматично призводить до входу користувача у систему (рис. 3.8).

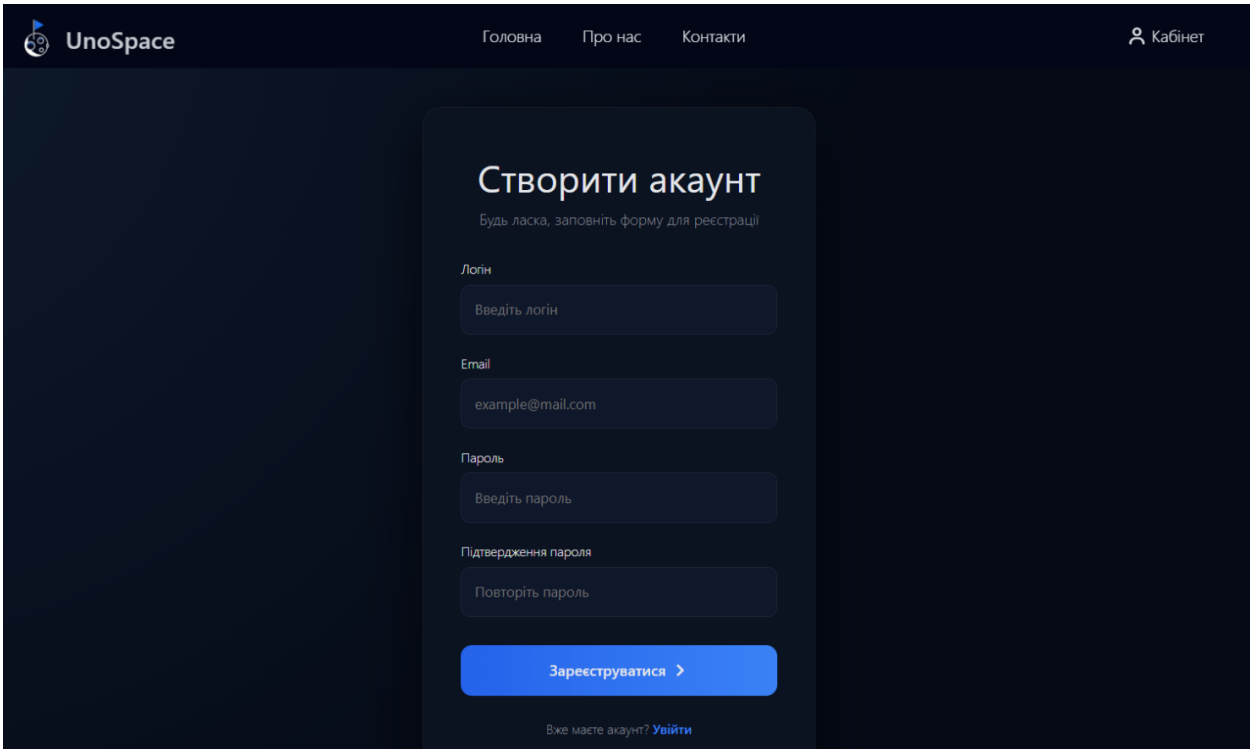

Рисунок 3.8 – Форма реєстрації користувача

Вхід до облікового запису (авторизація) оформлено через окрему форму, що вимагає введення електронної пошти та пароля. Після відправлення цих даних система шукає відповідний запис у сховищі та проводить зіставлення введеного пароля із збереженим хешем. У разі успішної ідентифікації ініціюється сесія, де зберігається ключова інформація (унікальний ідентифікатор, логін, пошта та надана роль). На основі визначеної ролі відбувається перенаправлення: адміністраторам відкривається доступ до керуючої панелі, тоді як звичайні користувачі потрапляють до основного контенту ресурсу (рис. 3.9).

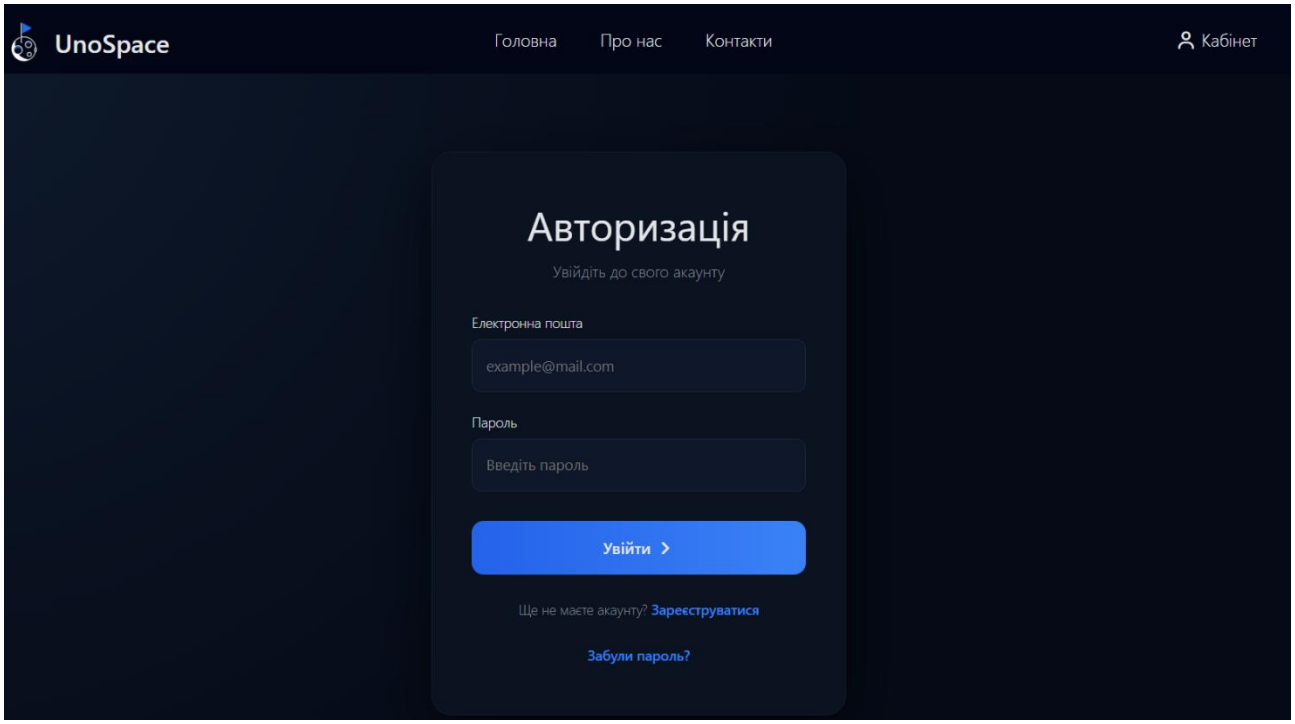

Рисунок 3.9 – Форма авторизації користувача

Якщо в процесі виникають будь-які збої (наприклад, некоректне введення даних або неіснуючий обліковий запис), система відображає повідомлення, що пояснюють причину невдалої спроби автентифікації чи реєстрації. Це значно покращує досвід взаємодії, дозволяючи оперативно усунути помилки введення.

Загальна будова адміністративної оболонки. Адміністративна частина системи збудована як окремий, захищений модуль, доступний лише тим обліковим записам, що мають адміністративні повноваження. Візуальне оформлення базується на стилях Bootstrap 5 і використовує єдиний шаблон верстки: спільний верхній блок (шапка, файл header-admin.php), бічний навігаційний елемент (сайдбар, файл sidebar-admin.php) та динамічну область для відображення вмісту. Цей уніфікований підхід забезпечує однаковий вигляд для всіх сторінок адмін-панелі та спрощує подальшу технічну підтримку.

– Керування обліковими записами. Блок керування користувачами складається з трьох взаємопов'язаних частин: сторінки для перегляду списку, форми для додавання нового користувача та форми для його модифікації.

Основна логіка опрацювання внесених даних зосереджена у файлі контролера users.php, який включається на початку кожної відповідної сторінки за допомогою `require_once`.

На головній сторінці цього розділу відображається таблиця з усіма зареєстрованими суб'єктами, що містить їхні ідентифікатори, логіни, електронні адреси та призначені ролі. Статус ролі визначається станом поля `admin` у сховищі даних: якщо це значення "1", відображається позначка «Admin», в усіх інших випадках — «User». Для кожного запису у списку передбачені активні посилання для переходу до редагування або для видалення (рис. 3.10).

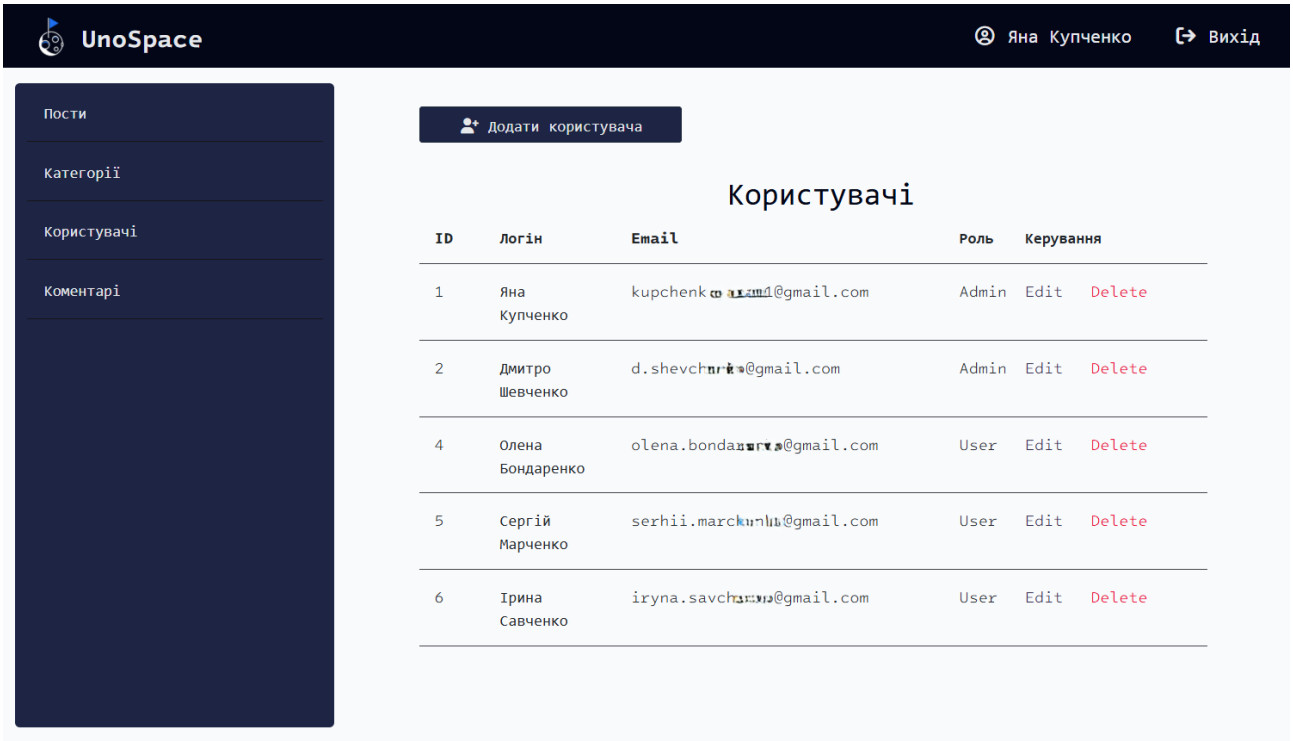

Рисунок 3.10 – Відображення списку користувачів у адміністративній панелі

Для отримання даних про користувачів контролер задіює службову функцію `selectAll('users')`, і повернутий масив даних ($users) передається у відповідний файл представлення. Видалення облікового запису ініціюється через HTTP GET-запит із параметром `delete_id`. Контролер перехоплює цей

ідентифікатор, викликає функцію `delete('users', $id)` та повертає адміністратора назад до таблиці зі списком.

Форма для реєстрації нового користувача вимагає введення логіну, електронної пошти, пароля та його повторення, а також містить перемикач для присвоєння статусу адміністратора. Перед тим, як вносити дані до сховища, контролер проводить серію валідаційних перевірок: підтверджує заповнення всіх обов'язкових полів, мінімальну довжину логіну (не менше двох літер), збіг введених паролів, а також унікальність поданої електронної адреси. У разі виявлення невідповідностей, помилки накопичуються в масиві `$errMsg` та виводяться адміністратору через підключений фрагмент файлу errorInfo.php. Лише після успішного проходження всіх перевірок пароль трансформується за допомогою хешування через `password_hash()` із стандартом `PASSWORD_DEFAULT`, і лише тоді новий запис зберігається у базі за допомогою функції `insert('users', $user)` (рис. 3.11).

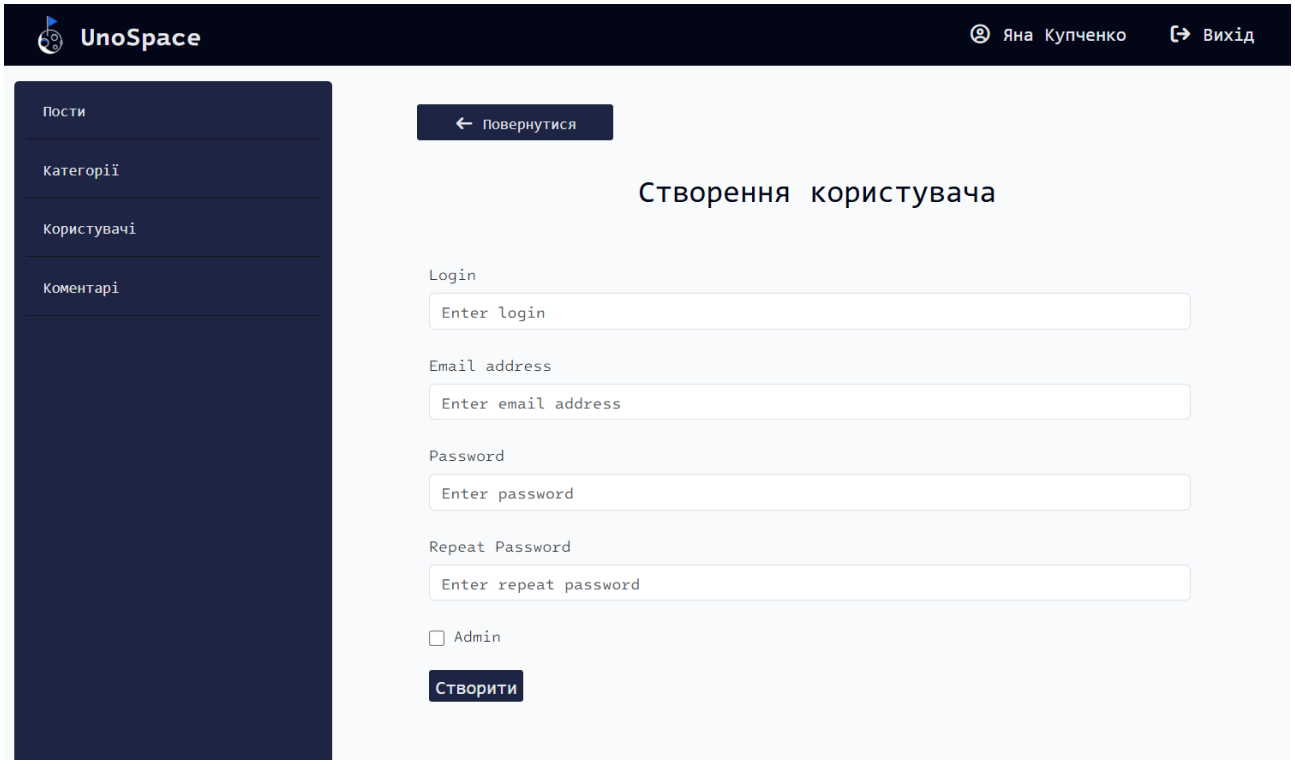

Рисунок 3.11 – Форма створення нового користувача

Форма для зміни даних активується за наявності GET-запиту з параметром `edit_id`. Контролер у цьому випадку витягує з бази повну інформацію про конкретного користувача та передає її в представлення для попереднього заповнення полів форми. Поле електронної пошти встановлено як незмінне (`readonly`), оскільки зміна цієї адреси не дозволена архітектурою. Поля пароля є необов'язковими: якщо адміністратор залишає їх порожніми, оновлення пароля не відбувається. Якщо ж новий пароль вказано, він проходить ідентичний процес валідації, що й при створенні, а після успішної перевірки хешується перед фінальним збереженням змін за допомогою функції `update('users', $id, $user)` (рис. 3.12).

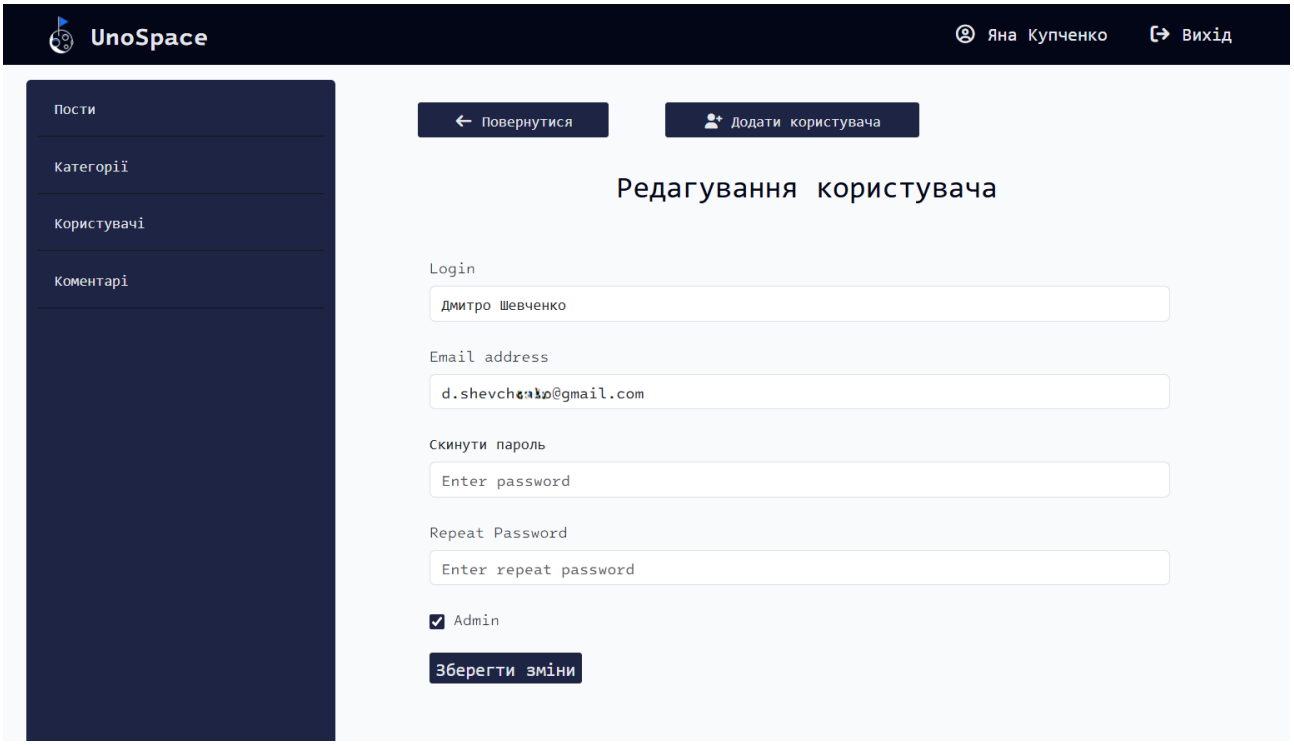

Рисунок 3.12 – Форма редагування даних користувача

– Керування категоріями. Модуль, відповідальний за роботу з категоріями, забезпечує повний ланцюжок дій: переглядати, створювати, змінювати та видаляти їх. Для доступу до переліку всіх категорій використовується команда `selectAll('topics')`, а результат відображається у формі таблиці, де присутні порядковий номер та власне найменування розділу (рис. 3.13).

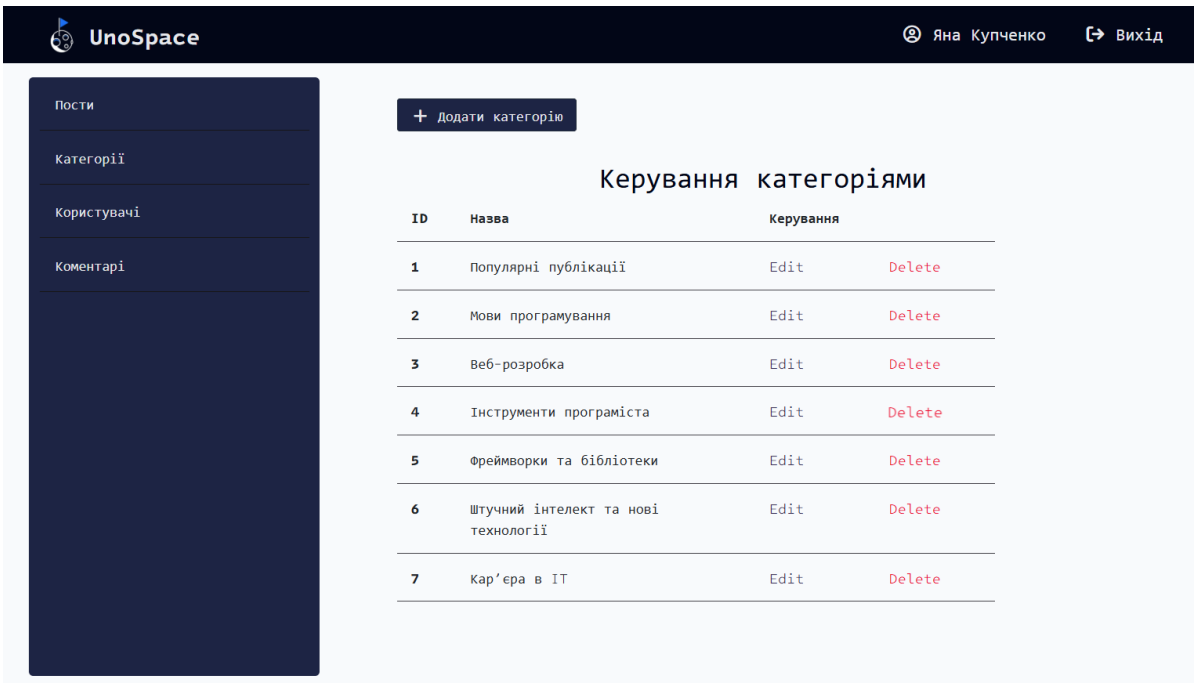

Рисунок 3.13 – Відображення списку категорій в адміністративній панелі

Форма для додавання нової категорії має дві складові: поле для введення назви та поле для детального опису. Відповідальний модуль (контролер) спершу перевіряє, чи обидва поля заповнені, а також чи має назва достатню мінімальну довжину (щонайменше два знаки). Окремо здійснюється контроль на неповторність: якщо категорія із таким самим найменуванням уже є у сховищі даних, процедура додавання зупиняється із виведенням відповідного попередження. У разі успішного збереження система автоматично перенаправляє користувача до загального переліку категорій (рис. 3.14).

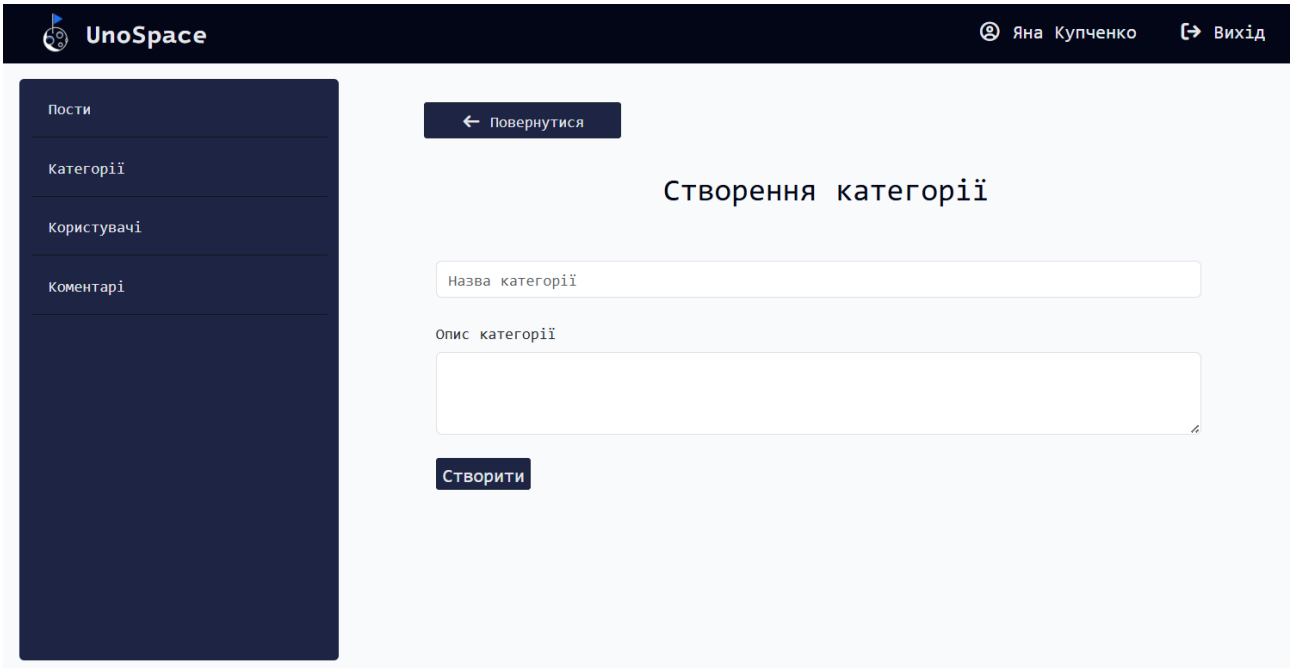

Рисунок 3.14 – Форма створення нової категорії

Можливість внесення змін активується через GET-запит, якому передається ідентифікатор (`id`). Контролер у цьому випадку витягує поточні дані з полів і передає їх на форму для візуалізації. Після відправлення POST-запиту, позначеного як `topic-edit`, виконуються валідаційні кроки, схожі на ті, що були під час створення. Якщо всі перевірки пройшли успішно, інформація оновлюється у сховищі за допомогою функції `update('topics', $id, $topic)` (рис. 3.15).

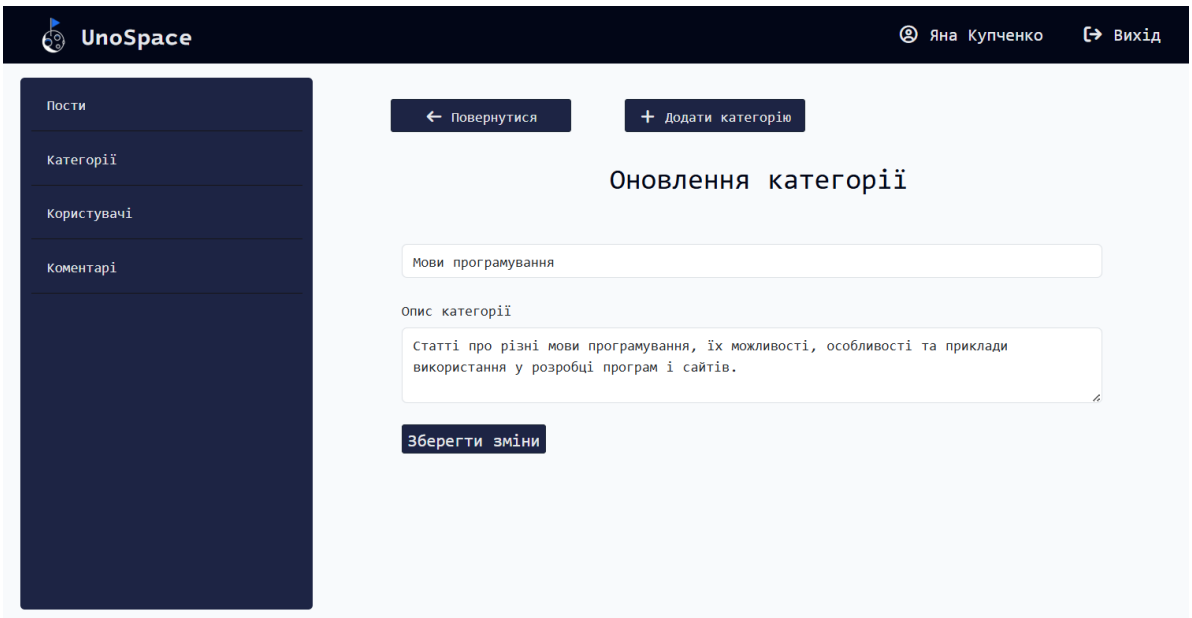

Рисунок 3.15 – Форма редагування категорії

Видалення вибраної категорії здійснюється за допомогою GET-запиту, до якого додається ідентифікатор на видалення (`del_id`). Після завершення цієї операції користувач із правами адміністратора отримує список категорій.

– Керування Публікаціями. Блок, що займається публікаціями, є найбільш насиченим функціоналом. Щоб вивести перелік дописів, задіяна специфічна функція `selectAllFromPostsWithUsers('posts', 'users')`. Вона здійснює злиття таблиць, додаючи до даних самого допису ім'я його автора. У табличному вигляді відображаються послідовний номер, заголовок (скорочений до 60 символів), ім'я автора та відповідна група навігаційних посилань для дій (рис. 3.16).

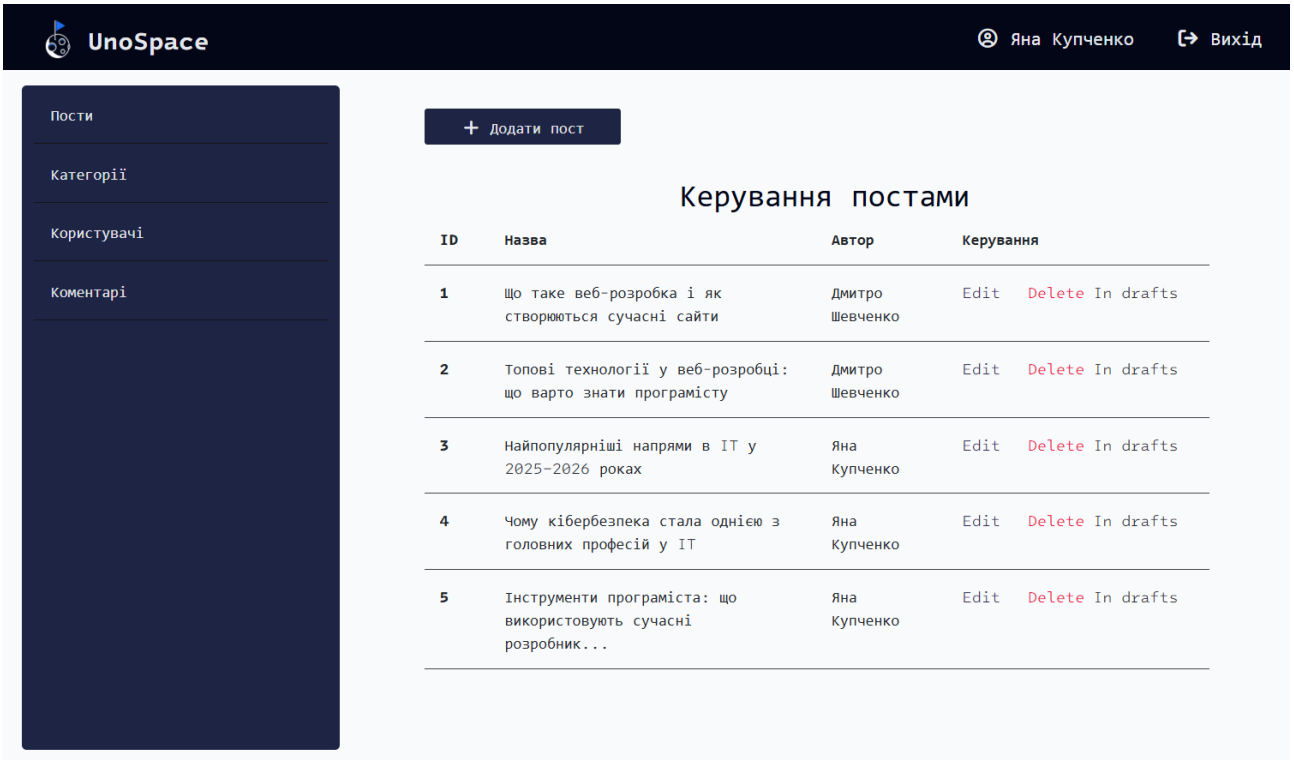

Рисунок 3.16 – Відображення списку публікацій

Окремим чином спроектовано механізм для зміни статусу публікації, який не вимагає відкриття форми для редагування. Поточний стан визначає, який саме

контрольний елемент буде показано: коли допис активний, відображається посилання «In drafts» (У чернетки), а коли він у стані чернетки — «Publish» (Опублікувати). При натисканні на відповідне посилання, контролер отримує параметри `pub_id` та `$publish` і викликає функцію оновлення `update('posts', $id, ['status' => $publish])`, після чого користувача повертають до списку.

Форма для створення нового допису містить поля під заголовок, основний текст (який реалізовано через `<textarea>`), опцію для завантаження зображення, селектор для вибору категорії зі спадного списку та прапорець для позначення публікації. Для завантаженого файлу зображення передбачена багаторівнева перевірка: аналіз MIME-типу файлу, контроль розмірів у пікселях (у межах від 300×300 до 2000×2000) та обмеження загального розміру файлу (не більше 2 МБ). У разі успішного проходження всіх перевірок, файл переміщується до каталогу `assets/images/posts/`, при цьому до імені додається мітка часу для гарантування його унікальності (рис. 3.17).

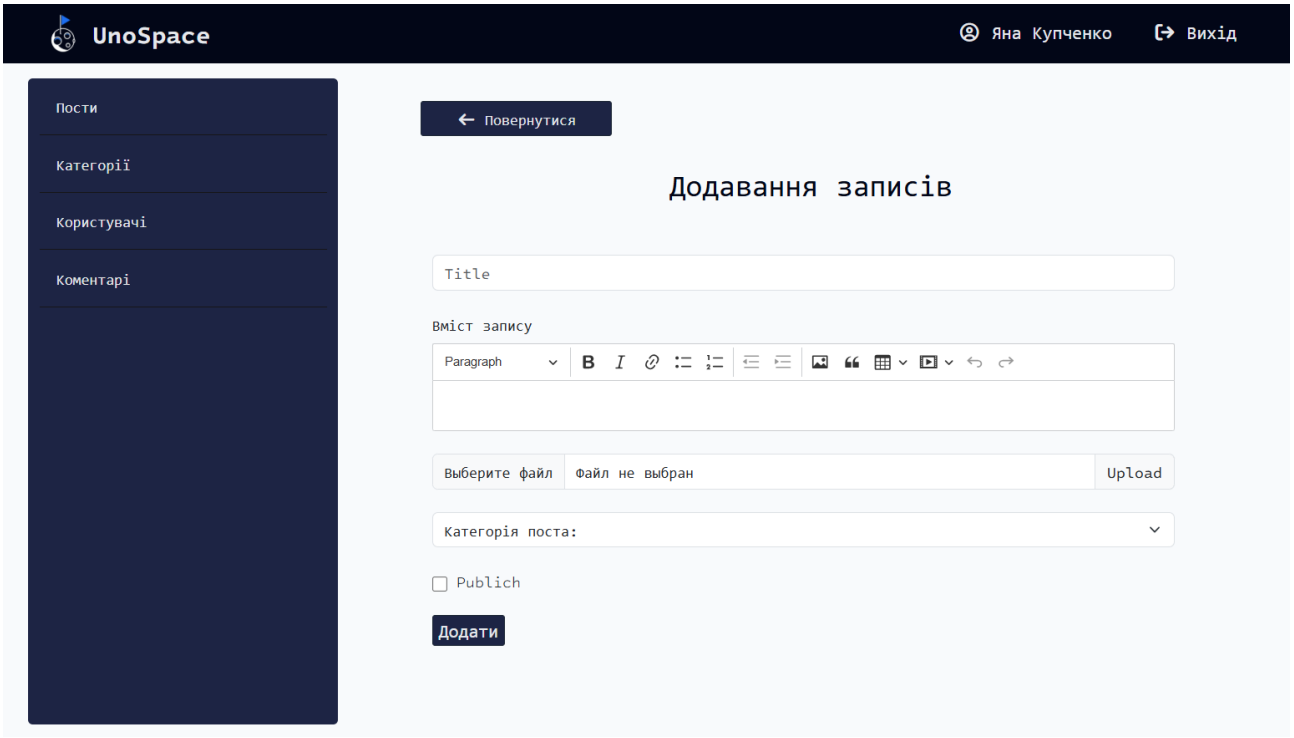

Рисунок 3.17 – Форма створення нової публікації

Якщо під час редагування допису нове зображення не завантажується, контролер забезпечує збереження посилання на попередню картинку (`$_POST['img'] = $post['img']`), що унеможливлює її втрату. Оновлення всіх інших полів працює за аналогічним принципом, як і процедура створення нового запису (рис. 3.18). Реалізацію функціоналу створення та обробки постів наведено у додатку Б (лістинг Б.2).

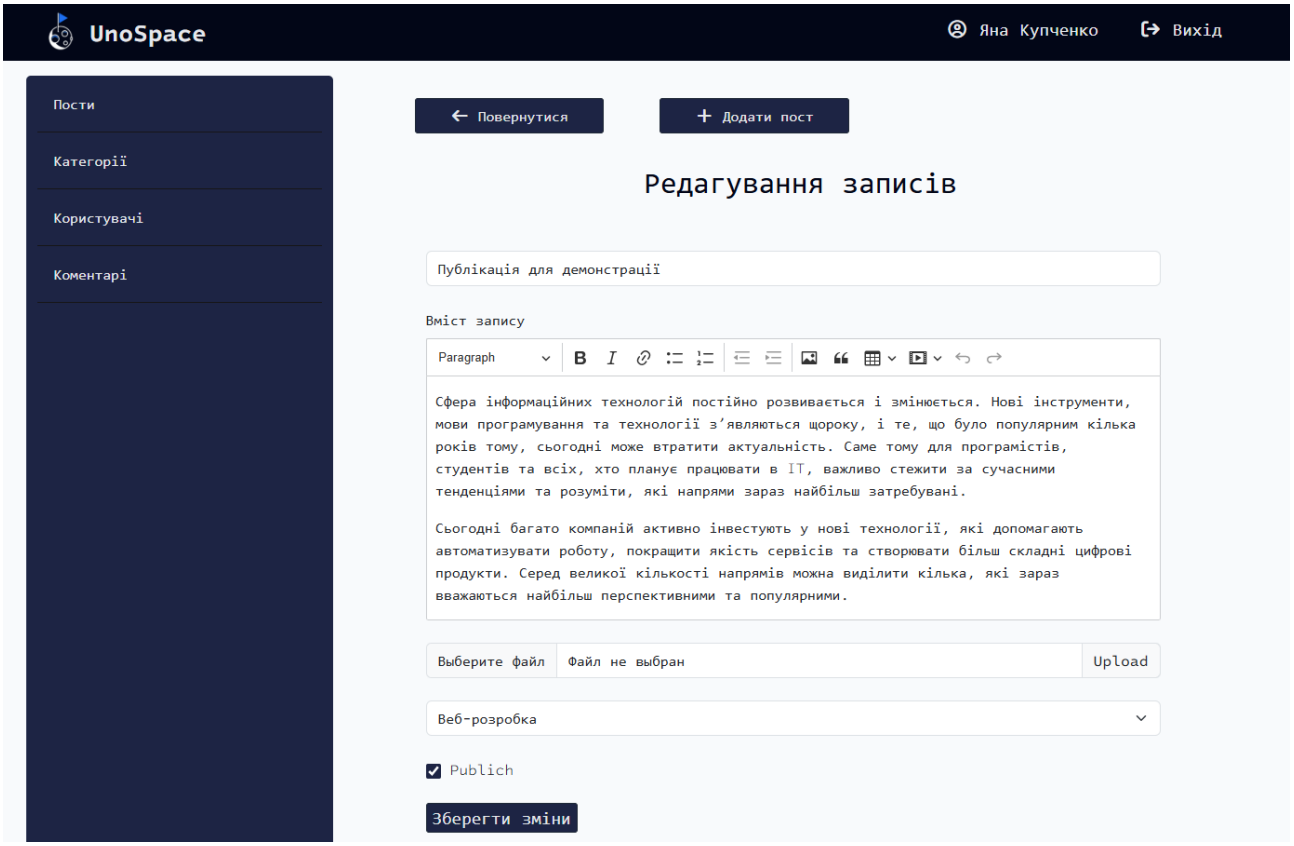

Рисунок 3.18 – Форма редагування публікації

– Адміністрування коментарів. Даний модуль системи пропонує адміністративному персоналу необхідний арсенал засобів для інспектування, трансформації, встановлення видимості та усунення коментарів. Для вилучення потрібної інформації задіяні спеціально розроблені методики: `getCommentsForAdmin()` та `getCommentsWithUsers()`. Вони забезпечують отримання коментарів у сукупності з даними про авторів (їхні імена) та заголовками статей, до яких ці коментарі належать.

У візуалізації даних, представленій у табличному вигляді, текст коментаря відображається у стислому форматі — лише перші 20 символів. Повний вміст тексту стає доступним при наведенні курсора завдяки спливаючій підказці (реалізованій через Bootstrap tooltip). Крім того, у таблиці демонструються прізвища осіб, які залишили коментар, гіперпосилання на першоджерело (публікацію) та панель інструментів для управління. Аби убезпечити систему від XSS-атак, усі текстові дані проходять обробку за допомогою функції `htmlspecialchars()` (рис. 3.19).

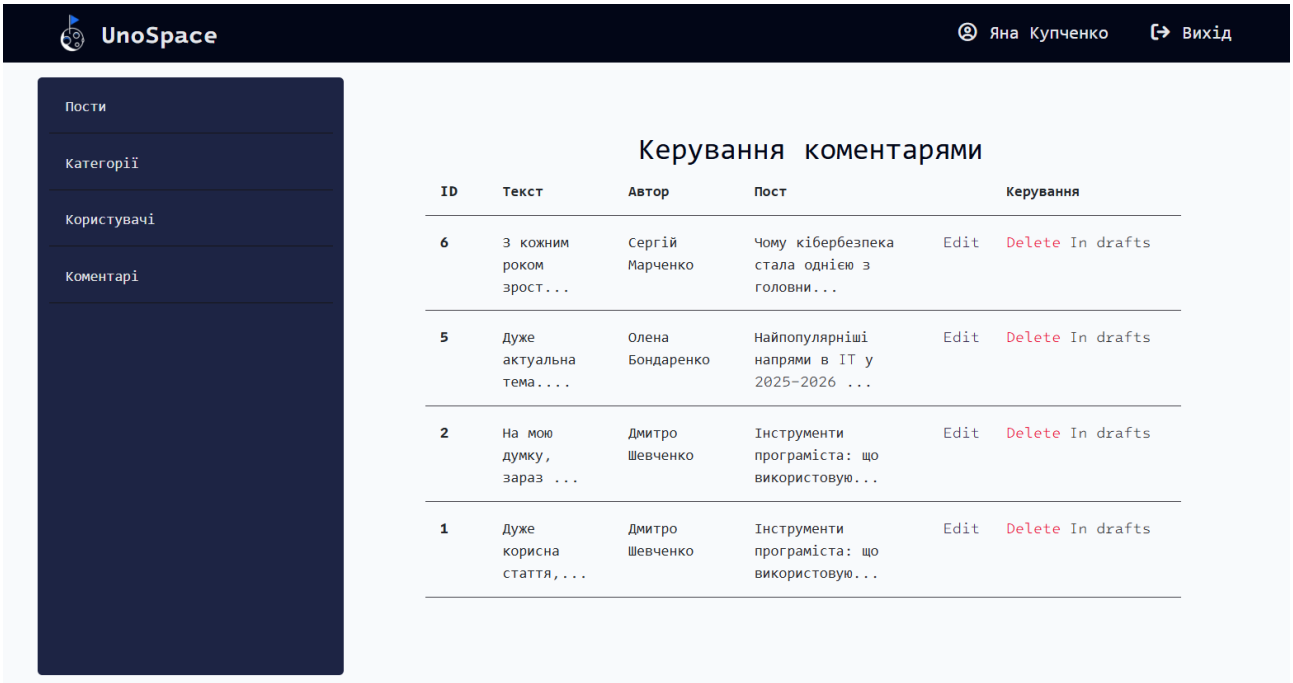

Рисунок 3.19 – Відображення списку коментарів

Інтерфейс для зміни коментаря дає змогу модифікувати його текстовий вміст та оновити статус опублікованості. Поле, що фіксує ім'я автора, є незмінним (лише для перегляду) і не підлягає жодним змінам. Функціонал оновлення статусу працює ідентично механізму, застосованому для статей: зміна виконується через передачу параметрів `pub_id` та `publish` у GET-запиті, що усуває необхідність у відкритті форми редагування. Операція ж видалення ініціюється GET-запитом, що містить ідентифікатор `delete_id`, після чого

система автоматично повертає користувача до загального списку коментарів (рис. 3.20).

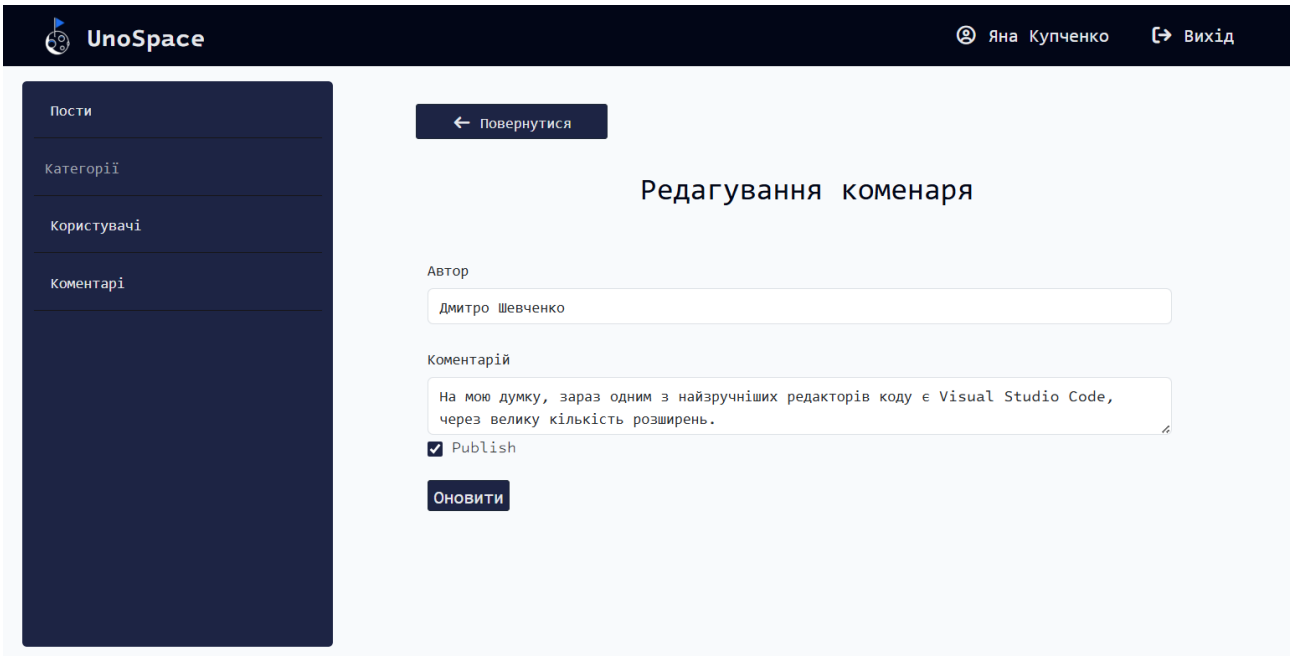

Рисунок 3.20 – Форма редагування коментаря

Запроваджені механізми реєстрації та авторизації у системі дають змогу якісно убезпечити доступ до можливостей платформи, а також гарантують правильне встановлення особи користувачів. Це, своєю чергою, посилює загальну захищеність системи та формує комфортне середовище для персоналізованого користування інтернет-ресурсом.

3.1.3 Реалізація публічної частини платформи

Загальнодоступна частина платформи вміщує усі сторінки, які може переглянути гість, не маючи жодних адміністративних привілеїв: головну вкладку зі списком контенту, сторінку із окремим дописом, огляд матеріалів за їхніми групами, пошуковий функціонал та систему коментування. Кожна з цих

частин реалізована як окремий PHP-файл, до якого долучаються уніфіковані файлові шаблони для верхньої (шапки) та нижньої (підвалу) частин сторінки.

– Стартова сторінка та механізм розбиття на сторінки. Головна сторінка слугує точкою входу для тих, хто завітав на ресурс. На самому початку даного файлу спершу підключаються необхідні допоміжні файли контролерів, після чого здійснюється підготовка інформації. Зокрема, звідси завантажується повний перелік усіх тематичних розділів, а потім формується хеш-масив $topicsMap, де ідентифікатор розділу є ключем, а його назва — відповідним значенням. Це дозволяє надалі показувати назву категорії для будь-якої публікації без необхідності робити додаткові звернення до бази даних.

На цій початковій сторінці, перед тим як демонструвати основний список статей, інстальовано інтерактивний банер у форматі "каруселі". Зміст для неї формується окремим запитом, який витягує публікації з найбільш затребуваних тем. Карусель збудована за допомогою конструкцій Bootstrap 5, використовуючи режим carousel-fade, і забезпечена автоматичною зміною слайдів кожні чотири секунди. Кожен слайд містить фонове зображення допису, шар із легким затемненням (напівпрозорий), заголовок (обмежений до 80 символів), короткий вступний текст (до 140 знаків, без HTML-тегів) та кнопку "Прочитати далі". Перший елемент у цьому наборі отримує клас CSS 'active', щоб карусель стартувала коректно з першого зображення. Керування перемиканням вручну реалізовано за допомогою стилізованих стрілок, створених через іконки Font Awesome (рис. 3.21).

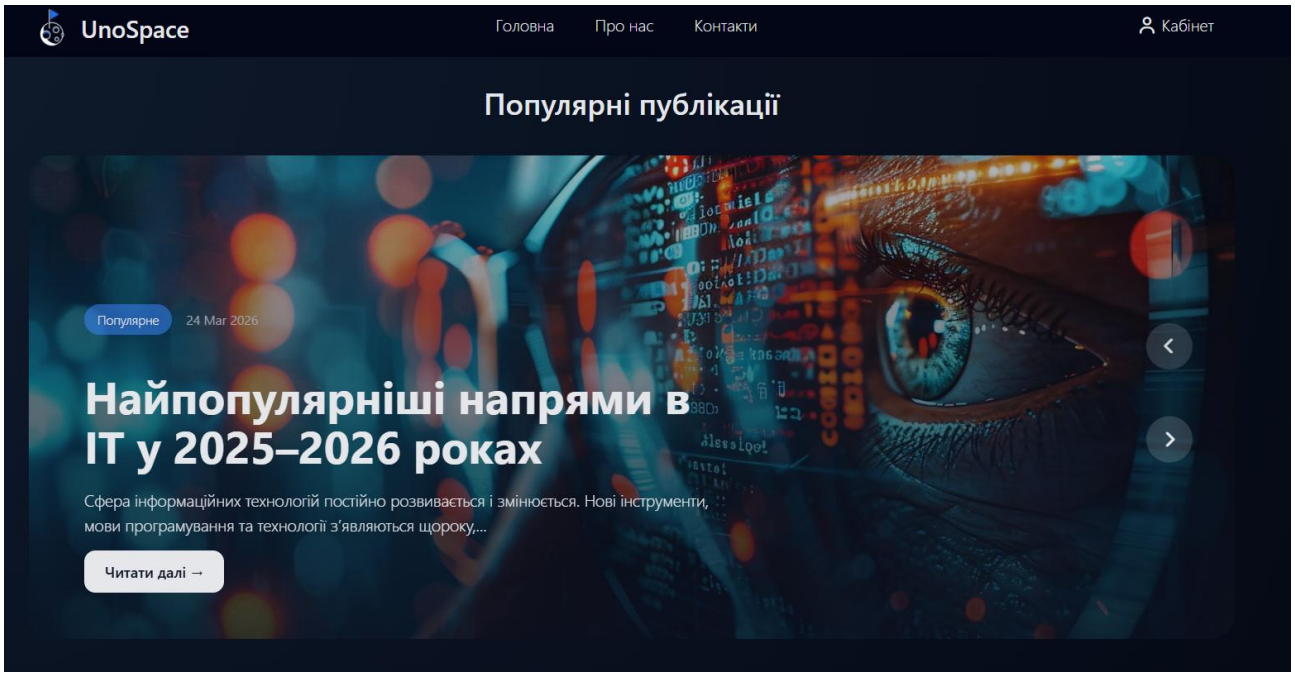

Рисунок 3.21 – Головна сторінка платформи: карусель популярних публікацій

Для забезпечення посторінкового відображення контенту використовується класична система пагінації. Актуальний номер сторінки зчитується з Get-параметра під назвою 'page', тоді як кількість матеріалів, що відображаються на одній сторінці, є фіксованою — саме чотири записи. Зміщення для вибірки ($offset) розраховується як добуток ліміту на номер попередньої сторінки, а загальна кількість сторінок визначається діленням загальної кількості записів у таблиці на обсяг сторінки із застосуванням округлення до більшого цілого. Самі ж публікації отримуються за допомогою спеціальної функції, яка виконує SQL-запит із об'єднанням (JOIN) таблиць дописів та користувачів, повертаючи, зокрема, ім'я автора (рис. 3.22).

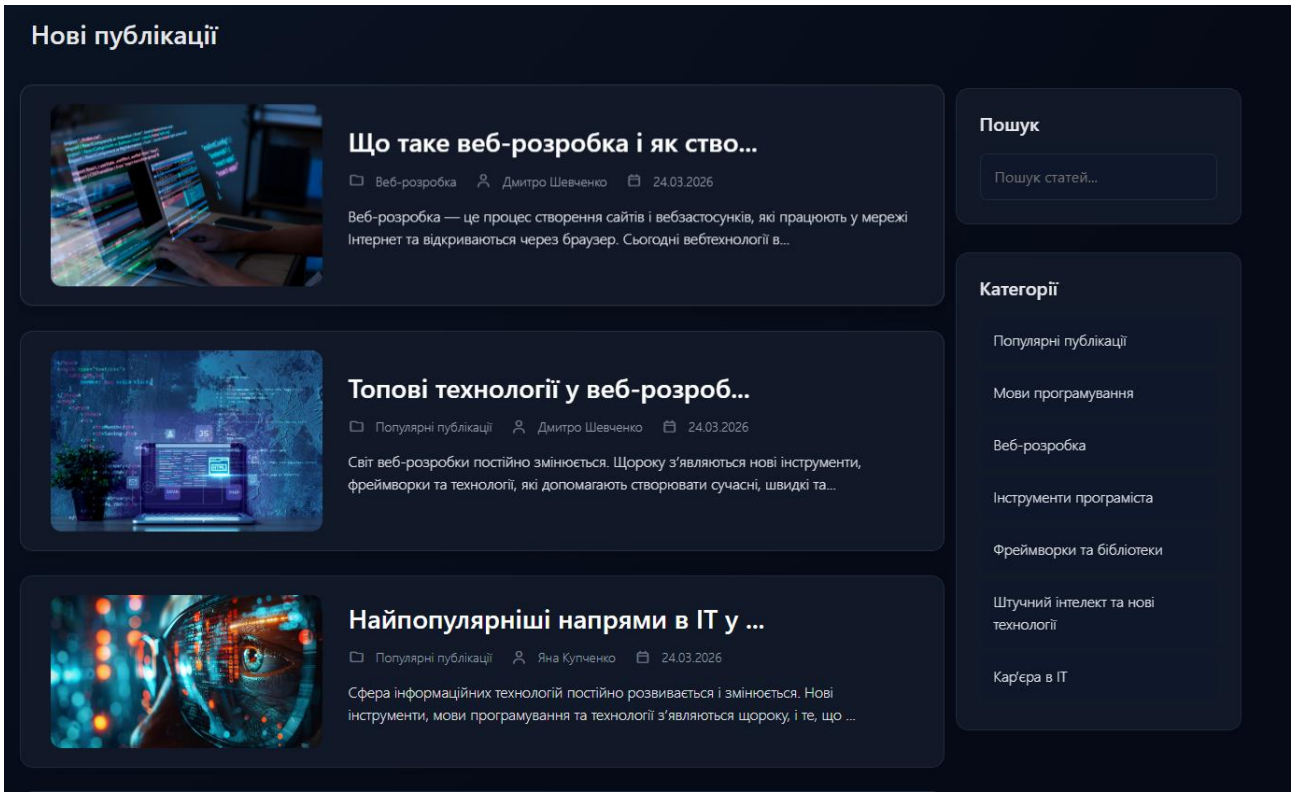

Рисунок 3.22 – Блок публікацій із бічною панеллю пошуку та категорії на головній сторінці

Для відображення навігаційних кнопок між сторінками залучається окремий шаблон пагінації. Логіка цього модуля обчислює "вікно" з посиланнями на номери сторінок, яке має фіксовану ширину навколо поточної сторінки. Якщо поточна позиція розташована близько до початку або кінця загального переліку, межі цього вікна коригуються так, щоб видима кількість кнопок завжди залишалася незмінною. Окремо виводяться посилання на перший і останній номер сторінки, розділені багатокрапкою та вікном, якщо між ними є розрив, а також кнопки для переходу на попередній чи наступний розділ, які з'являються лише тоді, коли такий перехід є технічно можливим (рис. 3.23).

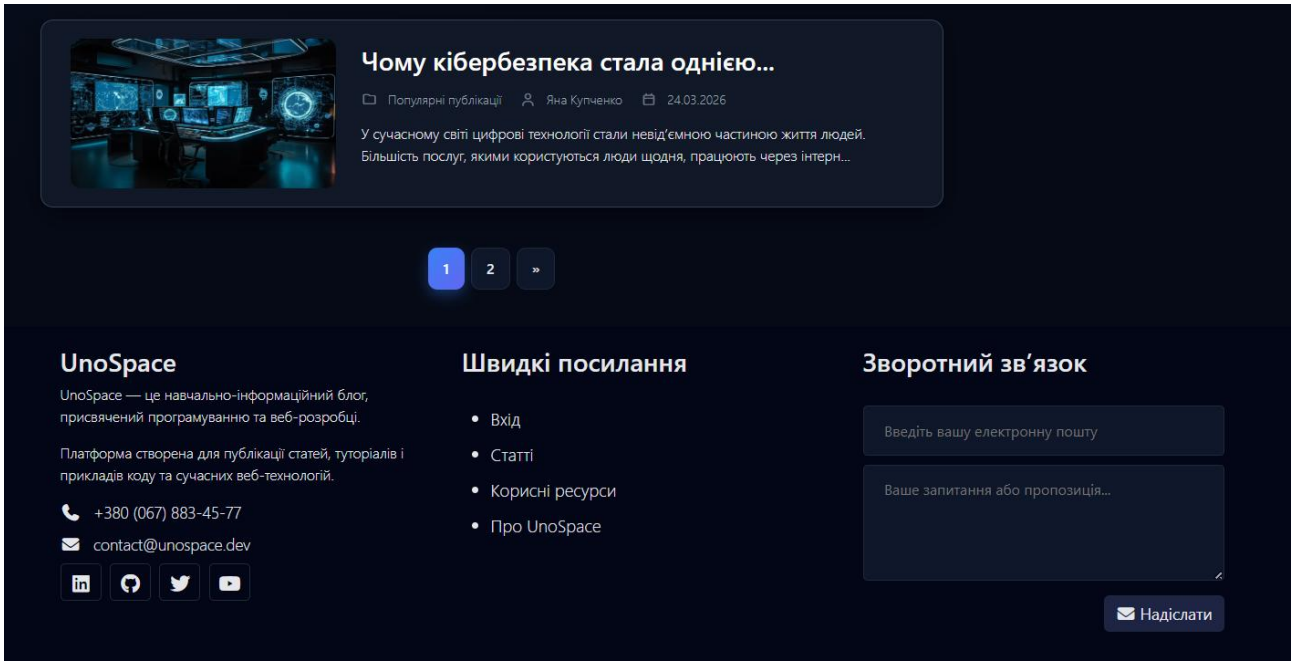

Рисунок 3.23 – Нижня частина головної сторінки: остання публікація, пагінація та підвал платформи

– Перегляд окремого допису. Для сторінки демонстрації обраної статті ідентифікатор запису забирається з GET-параметра, після чого виконується запит до бази даних, який поєднує таблиці дописів та даних про користувачів. На сторінці відображається заголовок, повноцінне зображення, атрибути (рубрика, автор, дата випуску) та весь вміст статті у форматі HTML. Оскільки адміністратор може вставляти розмітку безпосередньо через редактор, контент виводиться без очищення тегів; безпека гарантується на етапі збереження даних у адмін-панелі. Справа від основного контенту розташована бічна колонка, яка містить форму пошуку та список рубрик — функціонально аналогічна до тієї, що представлена на головній сторінці та в розділі рубрик (рис. 3.24).

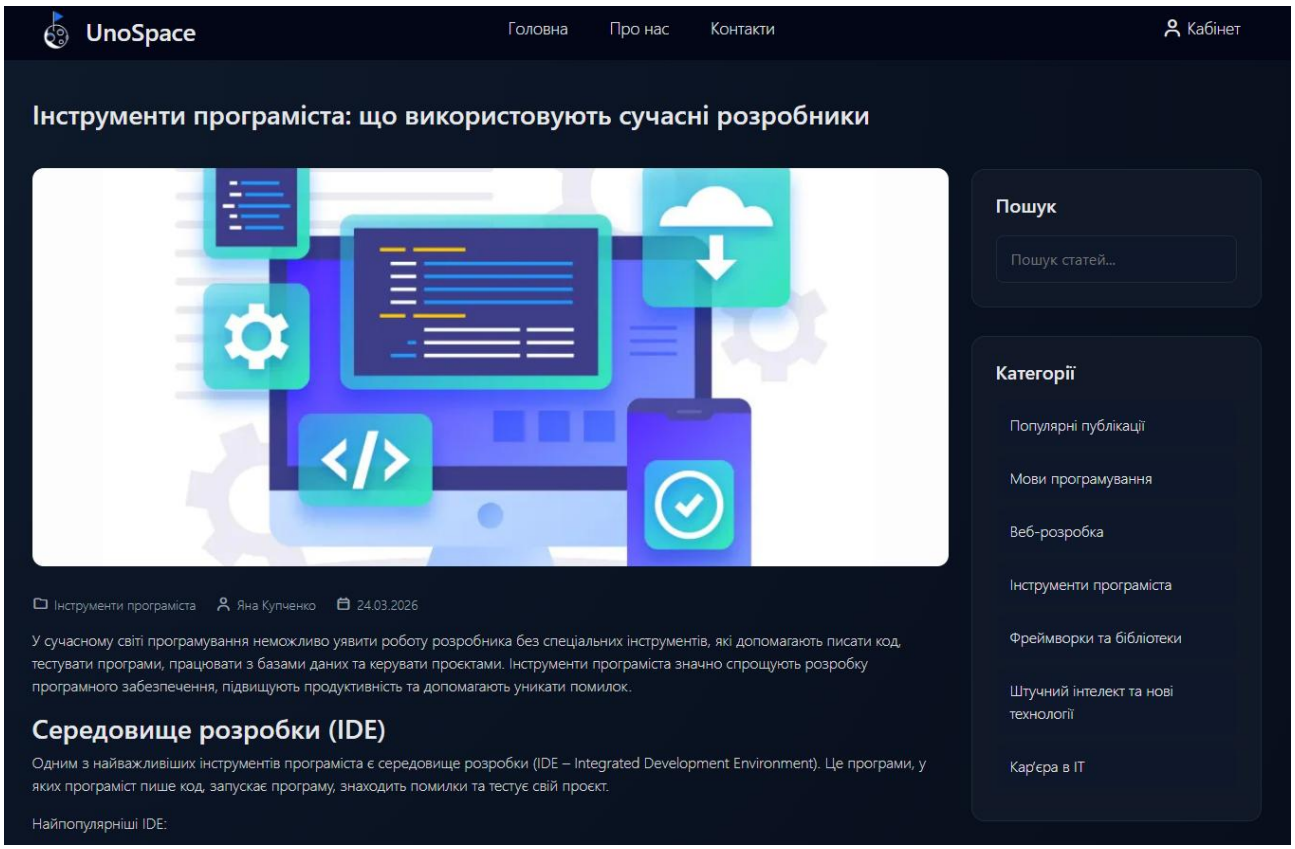

Рисунок 3.24 – Інтерфейс сторінки перегляду окремого допису

– Модуль коментарів. Секція коментарів інтегрується до сторінки допису окремим файлом шаблону. Вона підтримує двоярусну структуру: первинні коментарі та відповіді на них. Можливість додавання коментарів надається виключно аутентифікованим користувачам, що перевіряється наявністю запису в $_SESSION['id']. Якщо сесія неактивна, замість форми додавання виводиться сповіщення із закликом до входу в обліковий запис.

Форма містить приховані елементи: ідентифікатор поточного допису та `parent_id` — яким позначається ID коментаря, на який дається відповідь. Значення `parent_id` за замовчуванням встановлюється як фіктивний рядок і динамічно коригується за допомогою JavaScript після натискання кнопки «Відповісти». Відображення вже збережених коментарів реалізовано через рекурсивну PHP-функцію `showReplies()`, яка приймає сукупність усіх коментарів та Ідентифікатор батьківського елемента, фільтрує безпосередні

нащадки та виводить їх із відповідним відступом, при цьому вказуючи ім'я автора коментаря, на який відповідають, міткою «відповідає @...». Ця функція викликає сама себе для кожного ідентифікованого дочірнього коментаря, забезпечуючи повний обхід структури відповідей. Для захисту від XSS-атак текст усіх коментарів обробляється за допомогою `htmlspecialchars()` (рис. 3.25).

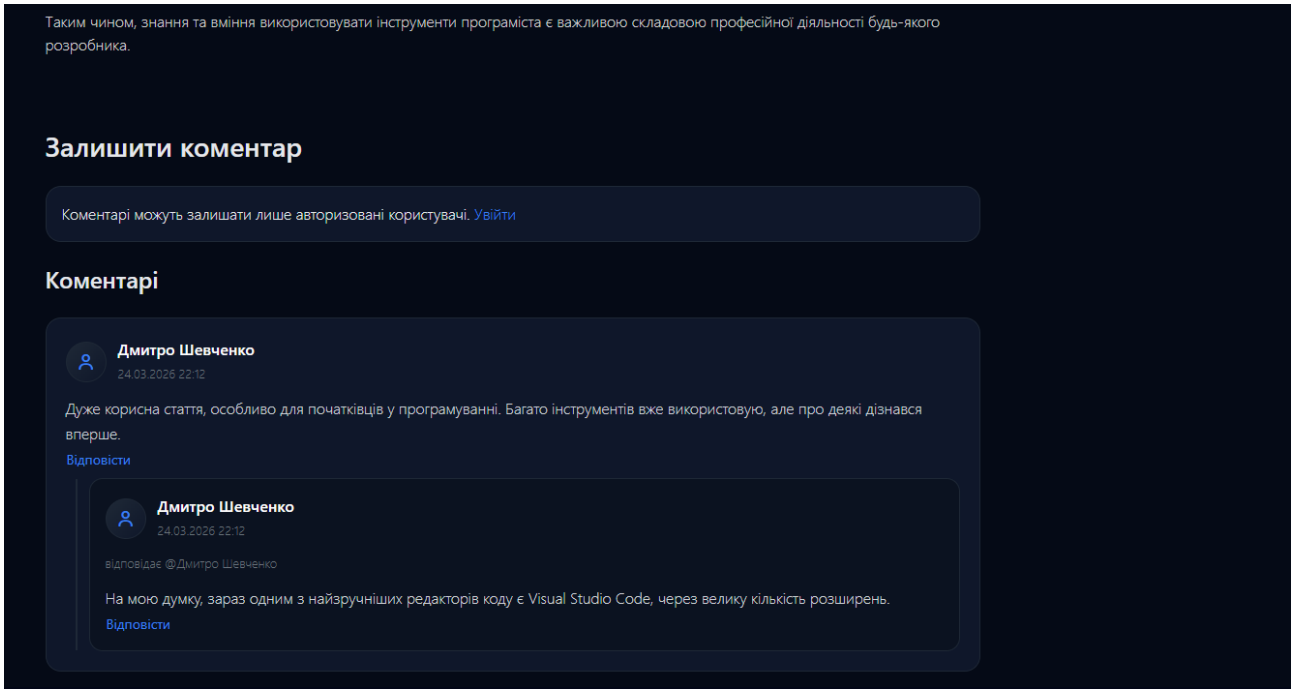

Рисунок 3.25 – Модуль коментарів на сторінці допису

– Модуль категорій та пошукові операції. На сторінці, присвяченій певній тематиці, зчитується ідентифікатор обраного розділу, отриманий з URL-параметра `id`. На основі цього ідентифікатора завантажується назва цієї тематики та повний перелік усіх матеріалів, асоційованих з нею. Кожен елемент (картка) матеріалу демонструє прев'ю-зображення, заголовок, ключову інформацію (метадані) та стислий виклад тексту, доповнений гіперпосиланням на повний текст статті. Візуальне оформлення цих карток повністю дублює стиль, застосований на головній сторінці, що забезпечує єдиний вигляд усього ресурсу (рис. 3.26).

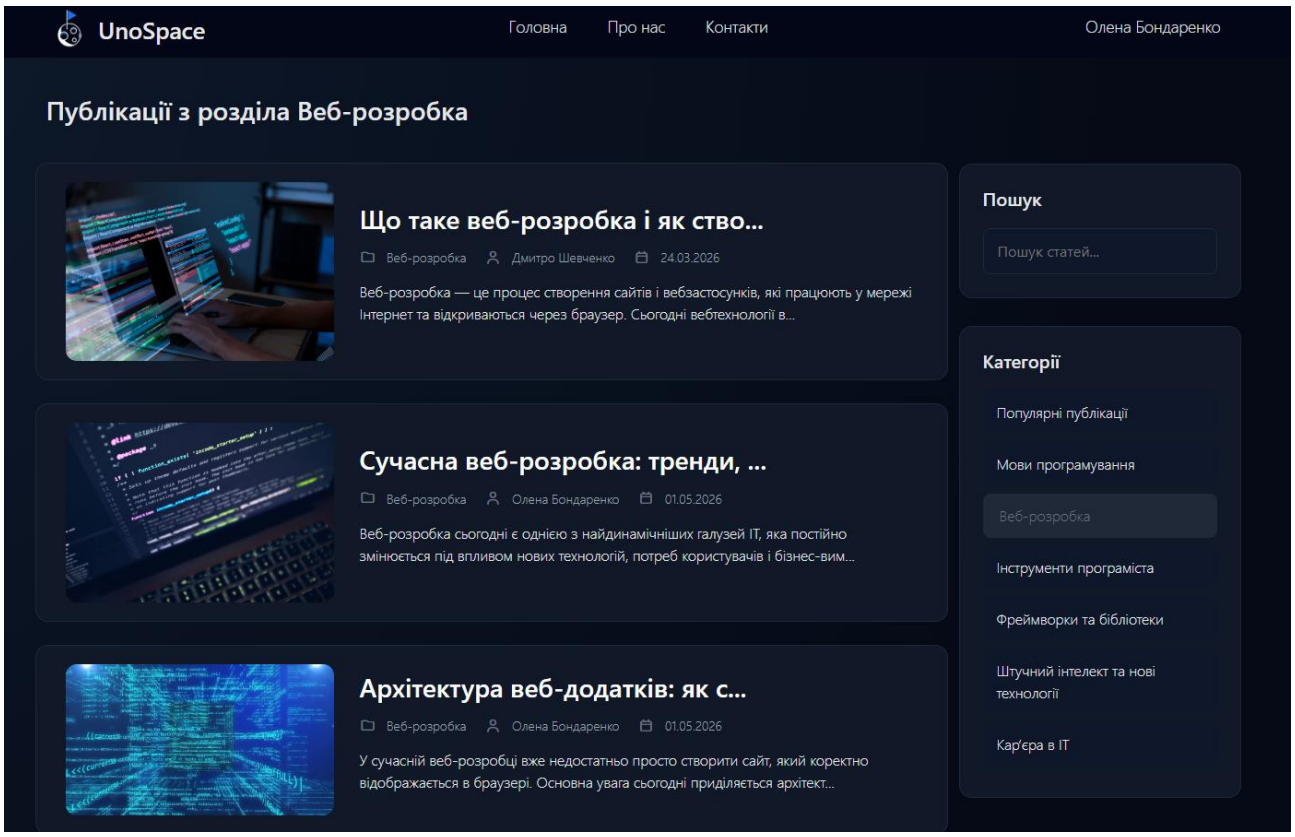

Рисунок 3.26 – Інтерфейс сторінки перегляду матеріалів за категорією

Сторінка пошуку обробляє дані, надіслані методом POST, а саме ключовий рядок пошуку, експортований з форми у бічній панелі. Якщо введений запит не є пустим, виконується виклик функції повнотекстового пошуку, котра шукає відповідності у заголовках та вмісті усіх опублікованих матеріалів. Результати пошуку демонструються у тому самому форматі карток, що й звичайний контент. Якщо збігів не виявлено, користувачеві показується відповідне повідомлення про відсутність результатів. Слово, яке використовувалося для пошуку, фіксується у змінній `$searchTerm` і повертається у поле введення форми через атрибут `value`, при цьому обов'язково здійснюється його екранування через `htmlspecialchars()` (рис. 3.27).

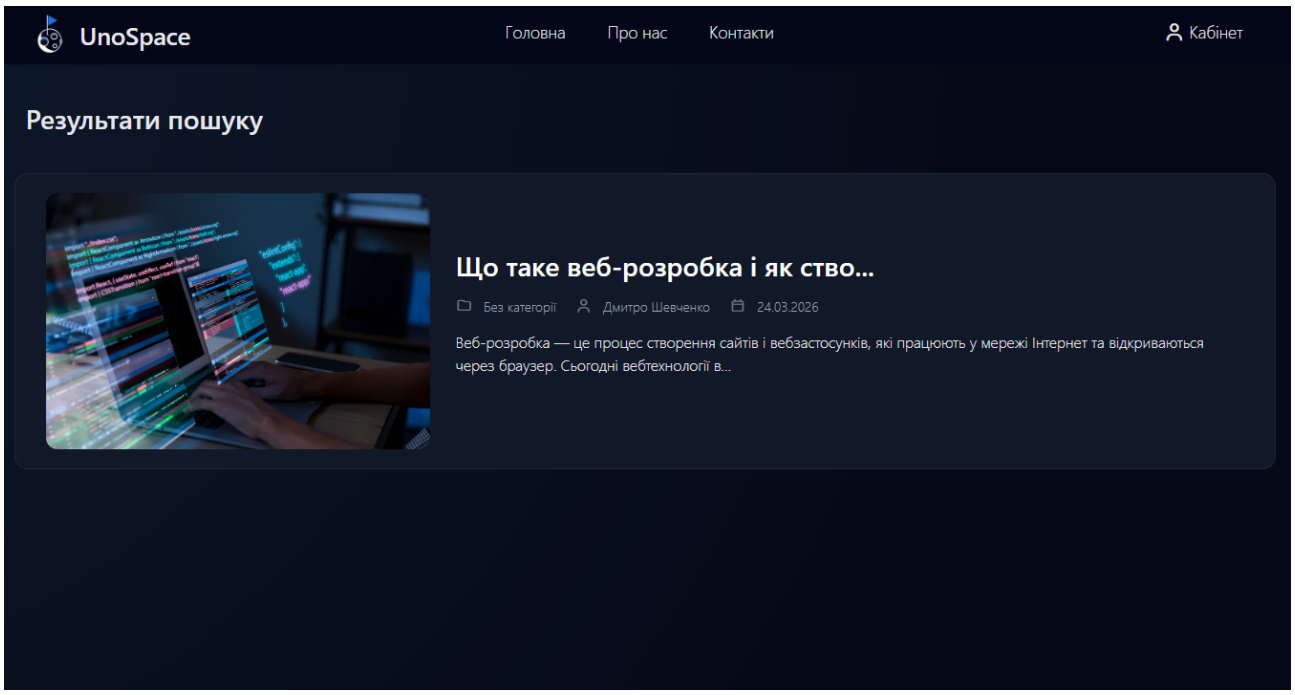

Рисунок 3.27 – Інтерфейс сторінки відображення результатів пошуку

– Універсальні складові: верхній блок, нижній блок та сторінка помилки. Верхній елемент навігації сайту (шапка) інтегрується на кожній сторінці за допомогою команди `include`. Він містить логотип, що веде на головну сторінку, основне меню навігації та секцію для управління сесією користувача. Приклади допоміжних сторінок («Про нас», «Контакти») наведено у додатку А. Зміст останнього пункту меню динамічно змінюється залежно від статусу сесії: для верифікованого користувача відображається його логін із випадаючим підменю, до якого включено доступ до адміністративної панелі (якщо користувач має відповідну позначку) і посилання для виходу; для незареєстрованого відвідувача пропонуються посилання на форми входу та реєстрації. Нижній блок (підвал) містить коротку довідку про платформу, контактні відомості, іконки соціальних мереж, набір швидких посилань та форму для зворотного зв'язку.

Окремим чином реалізовано сторінку індикації помилки 404. Вона активується при спробі доступу до неіснуючого URL-адреси. На цій сторінці також використовуються ті самі шаблони шапки та підвалу, а контент обмежується лаконічним текстом, що інформує про те, що запитувана веб-

сторінка не була знайдена, при цьому зберігається загальне стилістичне оформлення проєкту (рис. 3.28).

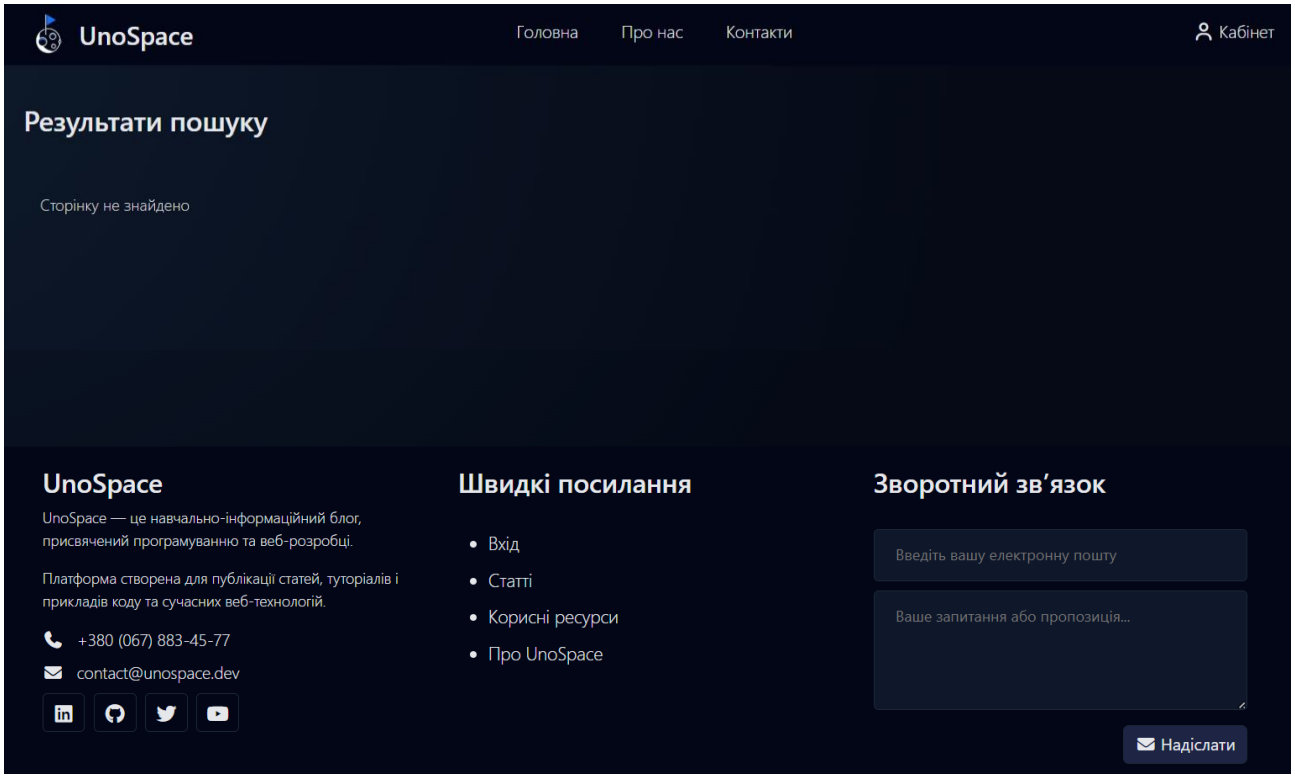

Рисунок 3.28 – Універсальні складові інтерфейсу: шапка, підвал та сторінка не знайдена

Клієнтська складова системи містить перелік усього потрібного функціоналу: починаючи від можливості бачити дописи й залишати відгуки, і закінчуючи механізмами пошуку, сортування згідно з категоріям та правильним реагуванням на некоректні посилання. Усі розділи мають уніфікований вигляд завдяки спільним макетам для верхньої та нижньої частин сайту, що гарантує логічну зв'язність усього інтерфейсу розробки.

3.2 Адаптивний інтерфейс на основі Bootstrap 5

Для сучасної платформи критично необхідним є коректне відображення інтерфейсу на різноманітних ґаджетах. Адаптивність нашої платформи

досягнута інтеграцією функціоналу Bootstrap 5 та застосуванням персональних CSS-правил, базованих на медіазапитах. Цей метод дав змогу втілити повну адаптивність без залучення додаткових бібліотек, водночас гарантуючи уніфікований вигляд на будь-якому девайсі.

– Структура сторінок та сітка. Розміщення елементів на усіх сторінках платформи ґрунтується на дванадцятиколонковій сітковій системі Bootstrap 5. Використання класів з префіксом `col-md-` дає змогу гнучко налаштовувати, як елементи реагують на зміну ширини дисплея. Так, на головній, сторінці категорій та сторінці новин основний вміст займає дев'ять колонок, а бічна панель — три. Коли ж користувач переходить на смартфон, ці блоки самі розширюються на всю доступну ширину і стають один під одним: спершу контент, потім бічна панель. Картки з публікаціями мають схожу логіку: на великих екранах зображення отримує чотири колонки, а текстовий блок — вісім, тоді як на мобільних пристроях вони обидва займають усю ширину, шикуючись одне за одним. Така організація гарантує комфортне споживання інформації незалежно від розміру екрана (рис. 3.29).

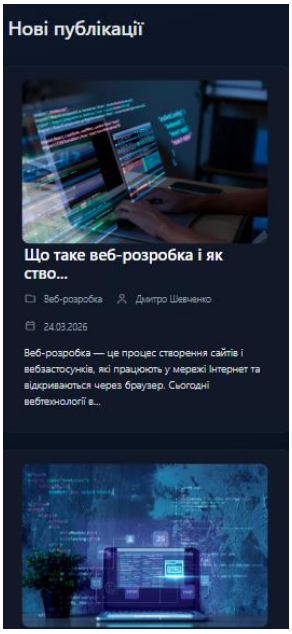

Рисунок 3.29 – Сторінка публікацій на мобільному пристрої

– Гнучке керування орієнтацією з боковим керуванням (offcanvas). Однією з найважливіших складових інтерфейсу, що пристосовується, є панель навігації, впровадження якої кардинально відрізняється залежно від типу пристрою, що використовується. На великих дисплеях ми бачимо стандартний горизонтальний блок меню, який містить прямі посилання на головні розділи ресурсу, а також висувне підменю для аутентифікації (вхід, реєстрація) або виходу, залежно від поточного статусу сесії користувача. Додатково, якщо особа має права адміністратора, у цьому ж підменю з'являється опція доступу до адміністративної вкладки. Для портативних гаджетів горизонтальний набір елементів повністю ховається, що досягається за допомогою медіа-запиту з межею поділу 768 пікселів; замість нього у верхній правій частині заголовка викликається "іконка-бургер", що являє собою три паралельні лінії. При активуванні цієї іконки, з правого краю екрана висувається бічне меню завширшки 280 пікселів. У цей же момент, основний вміст сторінки частково затемнюється напівпрозорим шаром, що слугує візуальним акцентом на активності меню та відокремлює його від решти інтерфейсу. Деактивувати меню можна або натиснувши на спеціальну кнопку закриття всередині панелі, або ж торкнувшись затемненої площі. Плавність цього процесу відкриття/закриття досягається завдяки застосуванню CSS- перехід до властивості `right`, що забезпечує візуально м'який і природний рух (рис. 3.30).

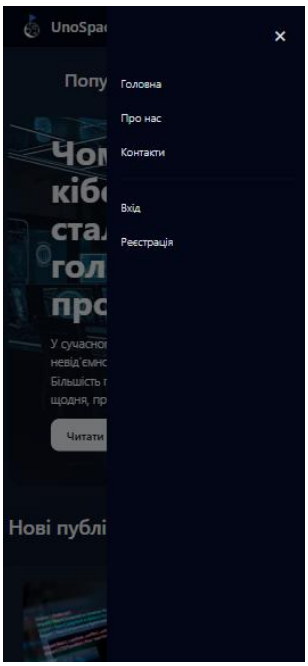

Рисунок 3.30 – Відкрите burger-меню на мобільному пристрої

– Бічна навігаційна панель. На екранах стаціонарних ПК бічна панель, де розташовані блоки пошуку та списку категорій, залишається закріпленою поруч із основним вмістом завдяки використанню властивості `position: sticky` зі зміщенням від верхнього краю вікна. При роботі з мобільними гаджетами, панель переміщується під основний контент, займаючи усю доступну горизонтальну площу. Такий підхід не перешкоджає перегляду матеріалів і підтримує чистий та інтуїтивно зрозумілий вигляд інтерфейсу (рис. 3.31).

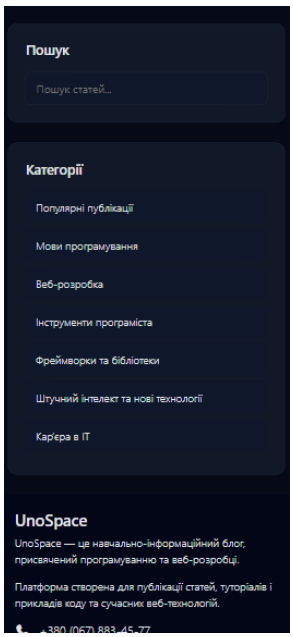

Рисунок 3.31 – Відображення бічної навігаційної панелі на мобільному пристрої

– Адаптивність форм та елементів керування. Усі форми, присутні на ресурсі — як-от форма для пошуку у боковій області, форма для залишення відгуків під публікаціями, чи форма контактів у нижньому колонтитулі — розроблені з використанням компонентів Bootstrap 5, що забезпечує їхнє розширення на всю ширину доступного простору на будь-якому типі екрана. Поля для введення даних та кнопки контролю автоматично масштабуються, зберігаючи зручність взаємодії, незалежно від того, чи використовується маніпулятор (миша) чи сенсорний екран. Пошукове поле додатково обладнане функцією очищення, яка реалізована як інтегрована кнопка, розташована всередині поля за допомогою абсолютного позиціонування, і залишається легко доступною незалежно від габаритів дисплея.

– Оформлення тексту та збереження візуальної єдності. Усі текстові елементи на проєкті використовують відносні одиниці вимірювання `rem`, що дає змогу коректно адаптувати розмір шрифтів відповідно до індивідуальних налаштувань браузера користувача. Заголовки, службова інформація про публікації (метадані) та фрагменти тексту попереднього перегляду підлаштовуються під наявну ширину, не виходячи за межі визначеної зони. Метадані, що включають категорію, автора та дату публікації, оформлені як гнучкий контейнер з опцією перенесення рядків (`flex-wrap: wrap`), що дозволяє цим елементам коректно займати один або кілька рядків залежно від вільного місця.

Таким чином, ми отримали інтерфейсне рішення, яке демонструє однаково коректне відображення як на потужних робочих станціях, так і на портативних пристроях, не поступаючись ані у функціональності, ані у візуальній цілісності.

3.3 Розгортання системи засобами Docker

Щоб забезпечити можливість відтворювання та розгортання платформи незалежно від середовища, ми інтегрували технологію контейнеризації Docker. Завдяки контейнеризації, кожен елемент системи отримує власне середовище виконання, що усуває прив'язку до налаштувань хост-машини та значно спрощує запуск на будь-якій наявній інфраструктурі.

– Архітектура контейнерного середовища. Дана платформа збудована на базі трьох відокремлених сервісів, які визначені у файлі конфігурації `docker-compose.yml`: це контейнер вебсервера із задіянням PHP, окремий сервер баз даних MySQL, а також інструмент для керування даними, phpMyAdmin. Хоча кожен сервіс працює у власному ізольованому контейнері, вони ефективно комунікують між собою через внутрішню мережу, створену Docker.

– Специфікація образу вебсервера через Dockerfile. Для формування образу, що обслуговує вебзастосунок, був створений індивідуальний `Dockerfile`, що базується на офіційному образі `php:8.2-apache`. Ця базова збірка вже містить інтерпретатор PHP версії 8.2 разом із інтегрованим вебсервером Apache, що повністю покриває потреби обслуговування PHP-додатку без потреби у додатковій інсталяції програмного забезпечення.

На етапі створення образу відбувається інсталяція розширень `pdo` та `pdo_mysql`. Вони необхідні для того, аби PHP-додаток міг встановити зв'язок із сервером MySQL, використовуючи інтерфейс PDO. Крім того, ми активуємо модуль Apache `mod_rewrite`, який є критично важливим для коректної обробки посилань у форматі ЧПУ (зрозумілих для користувача URL) через файл `.htaccess`.

Щоб надати вебсерверу повноваження на інтерпретацію налаштувань, прописаних у `.htaccess` у кореневій директорії додатку, ми створюємо файл конфігурації `override.conf`. У ньому задаємо директиву `AllowOverride All`. Цей файл конфігурації потім підключається до Apache за допомогою команди `a2enconf`.

– Складові частини docker-compose. Компонент `db` функціонує на базі офіційного образу mysql:8 та відповідає за фіксацію усієї інформації платформи. На старті, якщо це перше використання, буде автоматично створена база даних `unospace` із заданим адміністративним паролем. Для започаткування структури бази даних задіюється механізм точки входу MySQL: файл `db.sql`, що міститься у головній теці проєкту, монтується до шляху `/docker-entrypoint-initdb.d/` всередині контейнера. У результаті, при першому запуску цього компонента, схема таблиць та початкові записи імпортуються автоматично, не вимагаючи ручних дій. Збереження даних між перезапусками контейнера забезпечується постійним сховищем, що зветься `db_data`, яке зберігається на машині-хості. Порт 3306, що використовується контейнером, відображається на порт 3307 хоста. Це дає змогу зовнішнім клієнтам підключатися до СУБД, уникаючи конфліктів із будь-яким локально встановленим MySQL.

Компонент `web` будується із застосуванням власного Dockerfile, опис якого наведено раніше. Початковий код програми, що знаходиться у папці `./app/UnoSpace`, монтується безпосередньо у кореневий каталог вебсервера `/var/www/html`. Такий підхід дозволяє вносити правки у код без необхідності повторного створення образу, що значно прискорює цикл розробки. Цей компонент буде доступний через порт 8080 машини-хоста. Інструкція `depends_on` гарантує, що контейнер вебсервера почне роботу лише після того, як контейнер бази даних буде повністю запущено.

Компонент `phpmyadmin` розгортається, використовуючи офіційний образ phpmyadmin/phpmyadmin, та надає графічний інструмент для управління базою даних через веб-інтерфейс. Конфігурація компонента відбувається за допомогою змінних оточення: змінна `PMA_HOST` вказує на ім'я хоста бази даних у внутрішній мережі Docker, а `MYSQL_ROOT_PASSWORD` встановлює пароль для зв'язку. Веб-інтерфейс phpMyAdmin буде доступний на порті 8081 машини-хоста.

Розгортання повної конфігурації оточення здійснюється за допомогою єдиної команди :

docker-compose up --build -d

Після введення цієї команди, усі три компоненти (сервіси) ініціюються одночасно, беручи до уваги встановлені між ними зв'язки. На рис. 3.32 показано, який вигляд мають контейнери платформи, коли розгортання системи за допомогою Docker Desktop завершилося.

Рисунок 3.32 – Стан контейнерів системи в середовищі Docker Desktop

Вебплатформа буде доступна через посилання http://localhost:8080, тоді як інтерфейс для керування базою даних відкривається на http://localhost:8081. Опція -d забезпечує запуск цих контейнерів у фоновому режимі, а встановлене у налаштуваннях кожного сервісу значення restart: always гарантує самостійний рестарт у разі виникнення помилок або після перезавантаження машини.

Використання архітектури на базі Docker надає платформі ряд відчутних плюсів. У першу чергу, середовище для розробки стає ідентичним на будь-якому робочому комп'ютері, що усуває розбіжності у налаштуваннях між різними розробниками . По-друге, процес виведення системи на робочий (продакшн) сервер спрощується до копіювання файлів проєкту та запуску тієї самої команди. По-третє, відокремленість сервісів підсилює захищеність: вихід із ладу одного контейнера не спричиняє автоматичного збою функціонування інших. До того

ж, можна окремо масштабувати певні частини системи, не припиняючи роботу всього програмного продукту.

3.4 Аналіз реалізованої системи

Після завершення розробки блог-платформи, ми здійснили ґрунтовне дослідження готового продукту. Метою цього було встановити, наскільки втілена система відповідає тим критеріям, які стояли на початку, а також зрозуміти її переваги та куди рухатися далі з покращеннями.

– Відповідність функціональних вимог. Усі вимоги функціонального характеру, окреслені на стадії проєктування, були впроваджені без жодних винятків. Програмний комплекс забезпечує повний життєвий цикл управління контентом, що публікується: від заведення, правки, зміни статусу між «опубліковано» та «чорновик», до остаточного видалення. Тематичне розбиття матеріалів реалізоване за допомогою таблиці topics, яка дає змогу об'єднувати публікації відповідно до визначених IT-сфер. Пошук по всьому тексту, доступний як зареєстрованим, так і незареєстрованим відвідувачам, працює шляхом зіставлення через пошук відповідностей у заголовках та основному змісті матеріалів.

Схема підтвердження особистості й розмежування прав доступу впроваджена для трьох рівнів взаємодії: неідентифікований гість, зареєстрований користувач та адміністратор. Забезпечення безпеки паролів гарантується їхнім хешуванням за допомогою функції password_hash() із застосуванням стандарту PASSWORD_DEFAULT. Механізм залишення коментарів підтримує структуру з двома рівнями вкладеності, надаючи адміністратору можливість модерації. Адміністративний інтерфейс охоплює чотири головні блоки для керування: власне публікаціями, категоріями, обліковими записами користувачів та коментарями.

– Технічний опис реалізації. З погляду технологій, архітектура системи базується на класичній трирівневій моделі "клієнт-сервер". Логіка, що працює на сервері, написана мовою PHP, відповідає за опрацювання HTTP-запитів та комунікацію з СУБД MySQL. Для доступу до даних використовується інтерфейс PDO, що гарантує безпеку через застосування параметризованих запитів, тим самим унеможливлюючи загрози SQL-ін'єкцій. Схема бази даних була розроблена згідно з принципами третьої нормальної форми, що сприяє зменшенню дублювання даних та забезпеченню надійної цілісності зв'язків між елементами системи.

Фронтенд розроблено із застосуванням фреймворку Bootstrap 5 та мови JavaScript, що забезпечує адаптивне відображення елементів інтерфейсу на будь-яких пристроях, незалежно від габаритів екрана — чи то смартфон, чи то великий монітор стаціонарного комп'ютера. Для розгортання системи впроваджено контейнеризацію через Docker, яка слугує запорукою стабільності середовища виконання, незалежно від специфіки конфігурації машини-хоста.

– Виявлені обмеження системи. Під час аналізу було виявлено низку завад у нинішньому втіленні, які, хоча й не суперечать поставленій меті, проте окреслюють потенційні напрямки для розвитку платформи у наступних ітераціях.

Насамперед, бракує визначеної ролі «автор», яка б дала змогу зареєстрованим членам самостійно формувати й виставляти матеріали без потреби у правах адміністратора. На сьогоднішній день можливість публікувати вміст покладена винятково на адміністратора, що відповідає чинній схемі управління, проте це може стати стримуючим фактором при розширенні спільноти.

По-друге, у системі немає вбудованого інструменту для попереднього ознайомлення (попереднього показу) статті до її фінального оприлюднення, що підвищує вірогідність наявності помилок оформлення у вже випущених текстах.

– Оцінка відповідності нефункціональним вимогам. Підтвердження гнучкості інтерфейсу досягнуто через коректне відображення на різноманітних пристроях. Головне меню навігації трансформується відповідно до розміру екрана: на вузьких дисплеях горизонтальне представлення змінюється на висувну бокову панель (offcanvas).

Захищеність системи гарантується завдяки застосуванню багаторівневої системи захисту: паролі зберігаються у захешованому вигляді, користувацькі дані проходять обробку за допомогою `htmlspecialchars()` для запобігання XSS-атак, а кожен доступ до закритих даних на сервері суворо контролюється щодо наявності відповідних дозволів.

Використання Docker-контейнерів гарантує стабільність середовища та легке перенесення проєкту: запуск усієї необхідної інфраструктури здійснюється однією командою `docker-compose up --build -d`, що значно полегшує перехід між середовищами розробки та використання.

Аналіз вже впровадженого функціоналу дає змогу окреслити потенційні шляхи для подальшого розширення проєкту. До найбільш значущих пріоритетів варто віднести: запровадження окремої ролі «автора» для децентралізованого наповнення системи матеріалами, а також розробку системи сповіщень користувачів про появу свіжих коментарів до їхніх публікацій.

Підсумовуючи, розроблена платформа повністю відповідає поставленим завданням і являє собою готовий до експлуатації інструмент для розміщення технічної інформації серед фахівців IT-сфери. Виявлені наразі обмеження мають характер запланованих архітектурних рішень й жодним чином не заважають нормальній роботі платформи в рамках визначених критеріїв.

3.5 Висновки до третього розділу

У межах третього розділу було завершено повний цикл роботи над платформою для ведення блогу: спершу спроєктували базу даних, яка складається з чотирьох взаємопов'язаних таблиць, дотримуючись принципів

третьої нормальної форми. Далі створили адмін-панель, у якій реалізовано функціонал для управління контентом, керування категоріями, обліковими записами користувачів та коментарями. Паралельно розроблено й публічний інтерфейс, який підтримує пошукові механізми, класифікацію матеріалів за категоріями та багаторівневу структуру коментарів. Гнучкість дизайну досягнута завдяки використанню фреймворку Bootstrap 5, а процес впровадження системи було організовано за допомогою трьох окремих Docker-контейнерів, що забезпечує єдність середовища виконання незалежно від особливостей апаратної платформи, на якій відбувається розгортання. Оцінка функціональності впровадженої платформи підтвердила, що вона задовольняє всі встановлені вимоги — як ті, що стосуються функціоналу, так і нефункціональні аспекти. Крім того, цей аналіз допоміг визначити пріоритетні вектори для наступного етапу розвитку платформи.

ВИСНОВКИ

У ході роботи було успішно досягнуто поставленої мети – створено веб-орієнтованої інформаційної системи управління контентом на базі блог-платформи, орієнтованої на ІТ-спільноту.

Для досягнення поставленої мети було виконано наступні сформовані завдання роботи:

- здійснено розбір предметної сфери, включно з оглядом наявних майданчиків для розміщення ІТ-матеріалів.
- сформульовано перелік функціональних потреб системи;
- обґрунтовано остаточний вибір технічного набору інструментів;
- розроблено логічну схему бази даних та чітко прописаних взаємозв'язків між ними;
- реалізовано адміністративний інтерфейс;
- розроблено публічну частину платформи;
- впроваджено гнучкий дизайн, що ґрунтується на Bootstrap 5, гарантуючи належний вигляд на будь-якому пристрої;
- Виконано розгортання системи із застосуванням технології Docker.

Створений майданчик для публікацій дозволяє розміщувати і упорядковувати матеріали технічного спрямування, надає можливість розмежування прав доступу для різних ролей користувачів та дозволяє обмінюватися думками щодо опублікованого. Платформа усуває недоліки існуючих рішень, поєднуючи тематичну структуризацію, зручне адміністрування та повний контроль над системою.

Веб-рішення реалізовано, використовуючи мову програмування PHP та систему керування базами даних MySQL, контейнеризоване за допомогою Docker та доступне для використання як інструмент поширення фахових знань серед технічної спільноти.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. What Is a Blog? How To Create a Blog for Your Business [Електронний ресурс]. – Режим доступу: https://www.shopify.com/ae/blog/what-is-a-blog – Дата доступу: 03.11.2025.
2. Why It’s Important to Create Relevant Blog Content for Your Business [Електронний ресурс]. – Режим доступу: https://www.linkedin.com/pulse/why-its-important-create-relevant-blog-content-your-marco-reuter – Дата доступу: 05.11.2025.
3. What is best platform for tech blogging and building a developer community? [Електронний ресурс]. – Режим доступу: https://dev.to/mwanjemike/what-is-best-platform-for-tech-blogging-and-building-a-developer-community-5ho2 – Дата доступу: 22.11.2025.
4. Як обрати ідеальну платформу для свого блогу? [Електронний ресурс]. – Режим доступу: https://tuthost.ua/uk/blog/yak-obrati-idealnu-platformu-dlya-svogo-blogu/ – Дата доступу: 24.11.2025.
5. What is Dev.to and Why Every Developer Should Use It [Електронний ресурс]. – Режим доступу: https://dev.to/megha_m/what-is-devto-and-why-every-developer-should-use-it-22b1 – Дата доступу: 25.11.2025.
6. Огляд CMS Wordpress: що це, плюси та мінуси, приклади сайтів на Вордпресс [Електронний ресурс]. – Режим доступу: https://interkassa.com/blog/oglyad-cms-wordpress-plyusi-minusi-prikladi – Дата доступу: 27.11.2025.
7. Ghost CMS – платформа для створення лендінгів та блогів [Електронний ресурс]. – Режим доступу: https://www.komarov.design/ghost-cms-platforma-dlya-stvorennya-lendingiv-ta-blogiv/ – Дата доступу: 28.11.2025.
8. 10 порад, як створити класний блог самостійно [Електронний ресурс]. – Режим доступу: https://infounion.com.ua/ua/blog/kak-sozdat-blog-samostojatelno.html – Дата доступу: 30.11.2025.
9. Wiegers K., Beatty J. Software Requirements –3rd ed.–Microsoft Press, 2013.–673 p.

10. Основні види контенту та як їх правильно оформлювати [Електронний ресурс]. – Режим доступу: https://seomarket.ua/blog/osnovni-vydy-kontentu-ta-iak-ikh-pravylno-oformliuvaty/ – Дата доступу: 30.12.2025.
11. Ключові слова для пошуку [Електронний ресурс]. – Режим доступу: https://weblana.com.ua/blog-ua/klyuchovi-slova-poshuku – Дата доступу: 01.12.2025.
12. Автентифікація (аутентифікація): що це, методи та особливості [Електронний ресурс]. – Режим доступу: https://www.zen.com/ua/blog/authentication-definition-methods-features/ – Дата доступу: 01.12.2025.
13. 7 способів керування коментарями на вашому сайті, які ви повинні знати [Електронний ресурс]. – Режим доступу: https://genius.space/lab/7-sposobov-uravleniya-kommenariyami-na-vashem-sate-kotoe-vi-obyazni-znat/ – Дата доступу: 02.12.2025.
14. Адмін-панелі для бізнесу: від простого до складного [Електронний ресурс]. – Режим доступу: https://the-complex.agency/blog/admin-paneli-dlya-biznesu-vid-prostogo-do-skladnogo – Дата доступу: 03.12.2025.
15. Bootstrap Documentation. Introduction [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/5.3/getting-started/introduction/ – Дата доступу: 04.12.2025.
16. MySQL Documentation [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/ – Дата доступу: 05.12.2025.
17. Docker Documentation [Електронний ресурс]. – Режим доступу: https://docs.docker.com/ – Дата доступу: 06.12.2025.
18. PHP Documentation [Електронний ресурс]. – Режим доступу: https://www.php.net/docs.php – Дата доступу: 07.12.2025.
19. Розробка вебсайтів Technologies PHP [Електронний ресурс]. – Режим доступу: https://brander.ua/technologies/php – Дата доступу: 08.12.2025.

20. Етапи створення веб-додатку. Основи PHP та MySQL [Електронний ресурс]. – Режим доступу: https://promoter.net.ua/articles/etapi-stvorennya-veb-dodatku-osnovi-php-ta-mysql.html – Дата доступу: 08.12.2025.
21. MacIntyre P., Eisemann E., Secured D. PHP: The Good Parts. – Sebastopol : O'Reilly Media, 2010. – 173 p.
22. Tatroe K., MacIntyre P. Programming PHP: Creating Dynamic Web Pages. – 4th ed. – Sebastopol : O'Reilly Media, 2020. – 897 p.
23. Що обрати: PHP або Python? [Електронний ресурс]. – Режим доступу: https://wezom.com.ua/ua/blog/chto-vybrat-php-ili-python – Дата доступу: 12.12.2025.
24. The Modern JavaScript Tutorial [Електронний ресурс]. – Режим доступу: https://javascript.info/ – Дата доступу: 13.12.2025.
25. JavaScript [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript – Дата доступу: 14.12.2025.
26. Flanagan D. JavaScript: The Definitive Guide: Master the World’s Most-Used Programming Language. – 7th ed. – Sebastopol : O’Reilly Media, 2020. – 706 p.
27. Чому JavaScript — перспективна мова програмування? [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/topic/35184/ – Дата доступу: 16.12.2025.
28. Чому Bootstrap такий популярний [Електронний ресурс]. – Режим доступу: https://foxminded.ua/bootstrap/ – Дата доступу: 18.12.2025.
29. PostgreSQL чи MySQL. Як обрати оптимальну базу даних для вашого проєкту [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/topic/51621/ – Дата доступу: 22.12.2025.
30. MySQL Notes for Professionals. – GoalKicker.com, 2018. – 199 p. [Електронний ресурс]. – Режим доступу: https://www.slideshare.net/slideshow/mysql-notes-for-professionals-231841782/231841782#12
31. Докер проти Кубернетеса: переваги та недоліки [Електронний ресурс]. – Режим доступу: https://blog.desdelinux.net/... – Дата доступу: 24.12.2025.

ДОДАТОК А

Інтерфейс користувацьких сторінок вебзастосунку

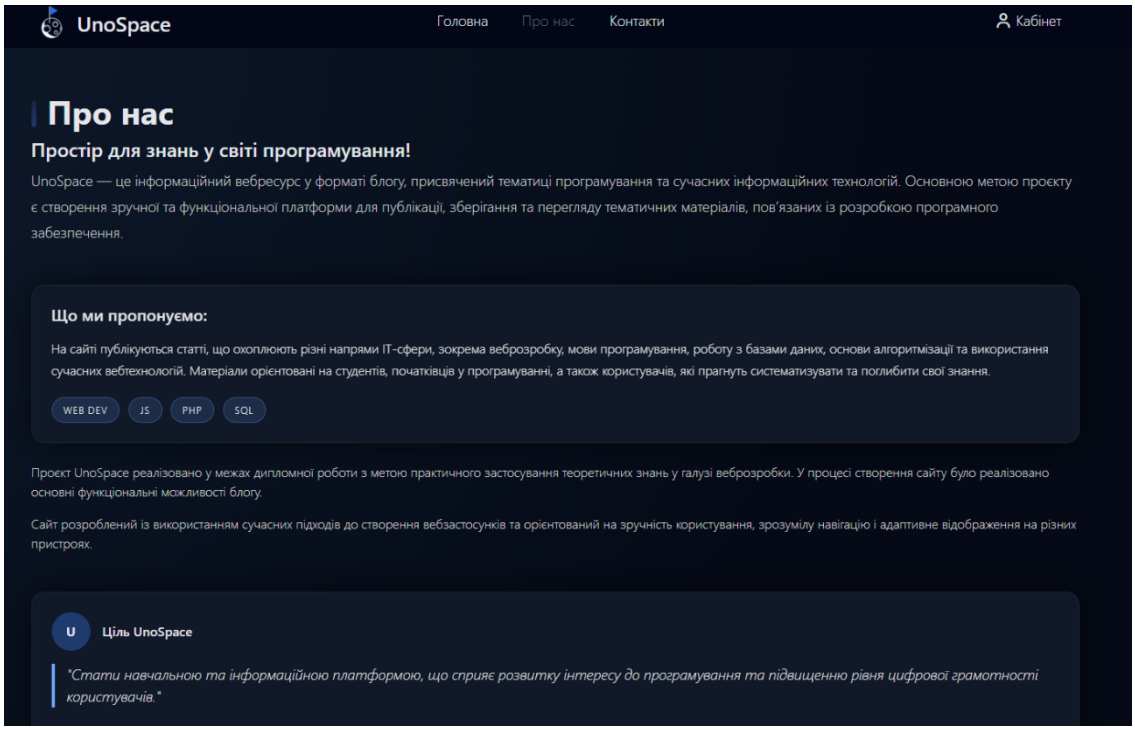

Рисунок А.1 – Сторінка «Про нас» блог-платформи

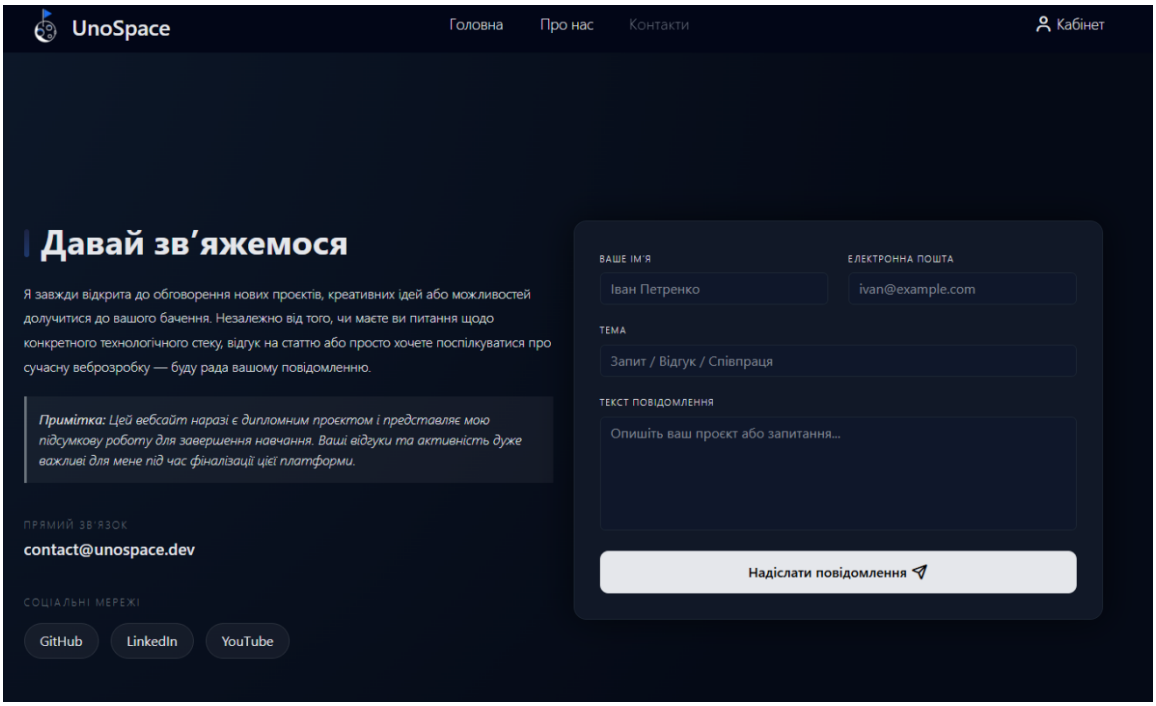

Рисунок А.2 – Сторінка «Контакти» блог-платформи

ДОДАТОК Б

Лістинг Б.1 – Реалізація реєстрації та авторизації користувачів

```
function setUserSession($user) {
    $_SESSION['id'] = $user['id'];
    $_SESSION['login'] = $user['username'];
    $_SESSION['email'] = $user['email'];
    $_SESSION['admin'] = $user['admin'];
    if($_SESSION['admin']){
        header('location: ' . BASE_URL . 'admin/users');
    }else{
        header('location: '. BASE_URL);
    }
}
// Код для форми реєстрації
if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['button-reg'])){
    $admin = 0;
    $login = trim($_POST['login']);
    $email = trim($_POST['email']);
    $passF = trim($_POST['pass-first']);
    $passS = trim($_POST['pass-second']);

    if($login === '' || $email === '' || $passF === '' ){
        array_push($errMsg,"Не всі поля заповнені!");
    }elseif(mb_strlen($login, 'UTF-8') < 2){
        array_push($errMsg, "Логін має містити щонайменше 2 символи.");

    }elseif($passF !== $passS){
        array_push($errMsg, "Паролі мають бути однаковими!");

    }else{
        $existence = selectOne('users', ['email' => $email ]);
        if( $existence && $existence['email'] === $email){
            array_push($errMsg,    "Користувач    із    такою    поштою    вже
зареєстрований!");
        }else{
            $pass = password_hash($passF,  PASSWORD_DEFAULT);
            $post = [
            'admin' =>$admin,
```

```
            'username' =>$login,
            'email' =>$email,
            'password' => $pass
            ];
            $id = insert('users', $post);
            $user = selectOne('users', ['id' => $id]);
            setUserSession($user);
        }
    }
}else{
    $login = '';
    $email = '';
}

//Код для форми авторизації
if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['button-log'])){
    $email = trim($_POST['email']);
    $pass = trim($_POST['password']);

    if($email === '' || $pass === '' ){
        array_push($errMsg,"Не всі поля заповнені!");
    }else{
        $existence = selectOne('users', ['email' => $email ]);
        if( $existence && password_verify($pass, $existence['password'])){
            setUserSession($existence);
        }else{
           array_push($errMsg,"Пошта або пароль введено неправильно!");
        }
    }
}else{
    $email = '';
}
```

Лістинг Б.2– Реалізація обробки постів

```
require_once SITE_ROOT . "/app/database/db.php";
if(!$_SESSION){
    header('location' . BASE_URL . 'log.php');
}
if (!defined('ROOT_PATH')) {
```

```
    require_once dirname(__DIR__, 2) . '/path.php';
}
$errMsg = [];
$id = '';
$title  = '';
$content = '';
$img = '';
$topic = '';
$topics = selectAll('topics');
$posts = selectAll('posts');
$postsAdm = selectAllFromPostsWithUsers('posts', 'users');

if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['add_post'])){
    $title = trim($_POST['title']);
    $content = trim($_POST['content']);
    $topic = trim($_POST['topic']);
    $publish = isset($_POST['publish']) ? 1 : 0;

    if(!empty($_FILES['img']['name'])){
        $imgName = time() . "_" . $_FILES['img']['name'];
        $fileTmpName = $_FILES['img']['tmp_name'];
        $fileType = $_FILES['img']['type'];
        $destination    =    ROOT_PATH    .    "/assets/images/posts/"    .
basename($imgName);

        if (strpos($fileType, 'image') === false) {
            array_push($errMsg, "Можна завантажувати лише зображення.");
        } else {
            // Перевірка розміру (пікселів)
            $imgInfo = getimagesize($fileTmpName);
            if ($imgInfo === false) {
                array_push($errMsg, "Файл не є зображенням.");
            } else {
                $width = $imgInfo[0];
                $height = $imgInfo[1];
                if ($width < 300 || $height < 300) {
                    array_push($errMsg,   "Зображення   занадто   маленьке
(мінімум 300x300 пікселів).");
                }
                if ($width > 2000 || $height > 2000) {
                    array_push($errMsg,   "Зображення   занадто   велике
(максимум 2000x2000 пікселів).");
```

```
                }
            }
        }

        if ($_FILES['img']['size'] > 2 * 1024 * 1024) {
            array_push($errMsg, "Файл занадто великий (максимум 2MB).");
        }

        if (empty($errMsg)) {
            $result = move_uploaded_file($fileTmpName, $destination);
            if ($result) {
                $_POST['img'] = $imgName;
            } else {
                array_push($errMsg, "Помилка завантаження зображення на
сервер.");
            }
        }

    } else {
        array_push($errMsg, "Помилка: ви не вибрали зображення.");
    }

    if($title === '' || $content === '' || $topic === ''){
        array_push($errMsg, "Не всі поля заповнені!");
    }elseif(mb_strlen($title, 'UTF8') < 7){
        array_push($errMsg, "Назва поста має бути більше 7 символів.");
    }

    if (empty($errMsg)) {
        $post = [
            'id_user' => $_SESSION['id'],
            'title' => $title,
            'content' => $content,
            'img' => $_POST['img'],
            'status' => $publish,
            'id_topic' => $topic
        ];

        $post = insert('posts', $post);
        $post = selectOne('posts', ['id' => $id]);
        header('Location: ' . BASE_URL . 'admin/posts');
```

```
    }
}else{
    $id = '';
    $title ='';
    $content = '';
    $publish = '';
    $topic = '';
}

if($_SERVER['REQUEST_METHOD'] === 'GET' && isset($_GET['id'])){

    $post = selectOne('posts', ['id' => $_GET['id']]);

    $id = $post['id'];
    $title = $post['title'];
    $content = $post['content'];
    $topic = $post['id_topic'];
    $publish = $post['status'];

}

if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['edit_post'])){
    $id = $_POST['id'];
    $title = trim($_POST['title']);
    $content = trim($_POST['content']);
    $topic = trim($_POST['topic']);
    $publish = isset($_POST['publish']) ? 1 : 0;

    if(!empty($_FILES['img']['name'])){
        $imgName = time() . "_" . $_FILES['img']['name'];
        $fileTmpName = $_FILES['img']['tmp_name'];
        $fileType = $_FILES['img']['type'];
        $destination    =    ROOT_PATH    .    "/assets/images/posts/"    .
basename($imgName);

        if (strpos($fileType, 'image') === false) {
            array_push($errMsg, "Можна завантажувати лише зображення.");
        } else {
            $imgInfo = getimagesize($fileTmpName);
            if ($imgInfo === false) {
                array_push($errMsg, "Файл не є зображенням.");
            } else {
```

```
                $width = $imgInfo[0];
                $height = $imgInfo[1];
                if ($width < 300 || $height < 300) {
                    array_push($errMsg, "Зображення занадто маленьке
(мінімум 300x300 пікселів).");
                }
                if ($width > 2000 || $height > 2000) {
                    array_push($errMsg, "Зображення занадто велике
(максимум 2000x2000 пікселів).");
                }
            }
        }

        if ($_FILES['img']['size'] > 2 * 1024 * 1024) {
            array_push($errMsg, "Файл занадто великий (максимум 2MB).");
        }

        if (empty($errMsg)) {
            $result = move_uploaded_file($fileTmpName, $destination);
            if ($result) {
                $_POST['img'] = $imgName;
            } else {
                array_push($errMsg, "Помилка завантаження зображення на
сервер.");
            }
        }
    } else {

        $_POST['img'] = $post['img'];
    }

    if($title === '' || $content === '' || $topic === ''){
        array_push($errMsg, "Не всі поля заповнені!");
    }elseif(mb_strlen($title, 'UTF8') < 7){
        array_push($errMsg, "Назва поста має бути більше 7 символів.");
    }

    if (empty($errMsg)) {
        $post = [
            'id_user' => $_SESSION['id'],
            'title' => $title,
            'content' => $content,
```

```
            'img' => $_POST['img'],
            'status' => $publish,
            'id_topic' => $topic
        ];

        update('posts', $id,$post);
        header('location: ' . BASE_URL . 'admin/posts');
    }
}

if($_SERVER['REQUEST_METHOD'] === 'GET' && isset($_GET['pub_id'])){

    $id = $_GET['pub_id'];
    $publish = $_GET['publish'];
    update('posts', $id, ['status' => $publish]);
    header('location: ' . BASE_URL . 'admin/posts');
    exit();
}

if($_SERVER['REQUEST_METHOD'] === 'GET' && isset($_GET['delete_id'])){

    $id = $_GET['delete_id'];
    delete('posts', $id);
    header('location: ' . BASE_URL . 'admin/posts');
}
```