МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

__Литвиненко О.Є._____
"_____"_____2022 р.

ДИПЛОМНИЙ ПРОЄКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: **«Веб-додаток для розміщення та редагування статей»**

Виконавець: Бугай А.М.

Керівник: Сябрук І.М.

Нормоконтролер: Тупота Є.В.

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)

Освітньо-професійна програма «Системне програмування»
Форма навчання денна

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Литвиненко О.Є._____
«_____»_____2022 р.

ЗАВДАННЯ
на виконання дипломної роботи (проєкту)
Бугая Антона Миколайовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. **Тема дипломної роботи (проєкту)** «Веб-додаток для розміщення та редагування статей»
затверджена наказом ректора від «_15_» _лютого_2022_р. №_251/ст_.
2. **Термін виконання роботи (проєкту):** з__16.05.2022____ по ___19.06.2022_____
3. **Вихідні дані до роботи (проєкту):** _веб-додаток для розмішення та редагування статей
4.**Зміст пояснювальної записки:**
__1.Аналіз з інтернет-додатків
__2.Особливості розробки веб-додатків
__3.Створення веб-додатку

5. **Перелік обов'язкового графічного (ілюстративного) матеріалу:**
_1. Алгоритм створення статті
_2. Алгоритм додавання даних до бази
_3. Діаграма зв'язків між таблицями
_4. Діаграма класів веб-додатку
_5. Сторінки додатку
_6. Інструменти розробки

6. **Календарний план-графік**

№ п/п	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1	Аналіз поняття веб-додатку, принципів його роботи та видів	16.05.22	
2	Розробка першого розділу дипломного проекту	17.05.22	
3	Аналіз літературних джерел розробки веб-додатків	17.05.22-18.05.22	
4	Написання другого розділу дипломного проекту	17.05.22-19.05.22	
5	Вивчення технологій розробки веб-додатків	19.05.22-20.05.22	
6	Практика використання технологій розробки веб-додатків	21.05.22-23.05.22	
7	Розробка веб-додатку	24.05.22-28.05.22	
8	Написання третього розділу дипломного проекту	28.05.22-29.05.22	
9	Оформлення пояснювальної записки	30.05.22-08.06.22	

7. **Дата видачі завдання:** «_16__» _травня_2022_ р.

Керівник дипломної роботи (проєкту) _Сябрук І. М._ _____
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання __Бугай А.М_____ _____
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Веб-додаток для розміщення та редагування статей»: 54 сторінки, 19 рисунків, 5 таблиць, 13 літературних джерел, 1 додаток.

Ключові слова: ВЕБ-ДОДАТОК, ВЕБ, , БЛОГ, СТАТТЯ, *PHP*, *HTML*, *CSS*.

Об‘єкт дослідження: веб-додаток для розміщення та редагування статей.

Предмет дослідження: розробка веб-додатків.

Мета дослідження – розробка веб-додатку для редагування та розміщення статей, для використання його в бізнесі, підвищення кількості відвідувачів сайту та для того щоб виражати самого себе, ставати популярним в мережі інтернет.

Важливість даної розробки полягає у створенні зручного додатку для розміщення різного виду інформаційного текстового контенту, його коментування та розповсюдження, який буде доступний для всіх, у кого є з‘єднання з інтернетом, та на будь-якій операційній системі.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

HTML — стандартизована мова розмітки для створення веб-сайтів.

CSS – каскадні таблиці стилів, формальна мова для опису зовнішнього вигляду веб-сторінок.

JavaScript — кросплатформна об'єктно-орієнтована мова сценаріїв, яка використовується для взаємодії з веб-сторінками.

Bootstrap — безкоштовний набір інструментів для створення веб-додатків і веб-сайтів.

PHP — мова сценаріїв, яка використовується для створення динамічних веб-сторінок на стороні веб-сервера.

MySQL — система управління реляційною базою даних.

SQL – мова запитів до баз даних.

Open Server – програмне забезпечення, через яке запускається локальний веб-сервер.

Фреймворк — готова модель в IT, заготовка, шаблон програмної платформи, на основі якої можна писати власні сценарії.

ВСТУП

У зв‘язку з поширенням інтернет технологій у всьому світі, все більше людей починають користуватися інформаційними інтернет ресурсами, замість бібліотек, друкованих журналів та газет.

Веб-додатки для редагування та розміщення інформаційних статей, іншими словами веб блоги, за останні кілька років стали асоціюватися з відомими людьми, бізнесом та хобі. Такі додатки можуть використовуватися для досить різних цілей, починаючи зі стрічки новин до персонального блогу відомого письменника, який показує свої думки читачам.

Веб-блоги можуть бути як самостійним проектом, так і частиною більш великого проекту, націленого на велику кількість глядачів. Для бізнесу перш за все це засіб для залучення відвідувачів на ваш сайт. Це найважливіше та фундаментальне твердження, яке має розуміти кожен підприємець, який підключає свій бізнес до Інтернету. На тлі розвитку сучасного інтернет-бізнесу, з його швидкістю укладання угод та розповсюдження інформації, необхідно усвідомлювати, що прямий продаж по телефону та електронній пошті працює лише до певної міри, поки кількість клієнтів не перевищить таку межу, коли на обробку та відправку повідомлень буде витрачатися весь робочий час.

Блог – це відмінний спосіб привернути увагу аудиторії та збільшити кількість відвідувачів та об’єми продаж для вашого бізнесу. Не слід розглядати це як непотрібну розкіш. Навіть випадкові повідомлення у блогах можуть допомогти бізнесу.

Як веб-блог працює на вас? Багато повідомлень у блогах приваблюють трафік на ваш сайт, оскільки це покращує пошукову оптимізацію. Блог дозволяє вам ділитися інформацією, яка перетворює відвідувачів на клієнтів. Також інформація з блогу може поширюватись без вашої участі. Клієнт або відвідувач може легко зробити репост посту, що сподобався. Новинки та традиційні ЗМІ також покладаються на корпоративні блоги. Різновиди веб-блогів:

Існують наступні різновиди веб-блогів:

- Персональний блог - ідеальне місце для творчості та самовираження. Автор сам вирішує, як писати, про що і для кого. Наприклад про моду, кімнатні рослини, тварин, знаменитості, їжу, автомобілі чи мотоцикли.
- Професійний блог - зазвичай організовують професійні блогери — відомі громадськості люди, яких читають та до думки яких дослухаються. Вони пишуть про все, але в основному на теми, специфічні для клієнтів. Майже всі дії провідних професійних блогерів спрямовані на отримання грошового заробітку.
- Тематичний блог - сайт зазвичай присвячений одній темі. Автор – людина, яка спеціалізується на конкретній галузі.
- Спільний блог - для таких додатків матеріали створюються не одним автором, а всіма бажаючими. Люди можуть писати на будь-яку тему, головне, щоб читачі були зацікавлені та постійно поверталися на сайт-блог, тим самим збільшуючи його популярність та кількість відвідувачів. У такому блозі частіше розміщуються поточні статті, оскільки вони мають попит у певної цільової групи.
- Партнерський блог - принцип ведення аналогічний до співавторства книги. Партнерський блог добре просуває спільний бізнес. Підвищує інтерес читачів як до рекламованого продукту, так і особистості авторів блогу.
- Медіа-блог - тут публікуються не лише текстові та графічні матеріали, а й відео. Чим більше публікується відео, тим більший інтерес у публіки. Таким чином, користувач краще та легше сприймає інформацію, коли він бачить та чує її одночасно. Багато знаменитостей ведуть особисті блоги і викладають відео про все у світі. Таким чином, вони постійно викликають інтерес глядачів та підвищують власну популярність.

РОЗДІЛ 1
АНАЛІЗ ІНТЕРНЕТ-ДОДАТКІВ

1.1. Поняття веб-додатку

Веб-додаток – це будь-яка програма, яка виконує певну функцію, використовуючи веб-браузер. Програма може бути простою, як, наприклад, дошка оголошень або контактна форма на сайті, або складною як тестовий процесор чи багатофункціональний додаток, який має вигляд сайту, але функціоналом зрівнюється з великим додатком, який завантажується на комп‘ютер чи мобільний телефон.

Що таке клієнт? «Клієнт» використовується в середовищі клієнт-сервер для позначення програми, яка використовується для запуску додатку. Клієнт-сервер – це мережа, в якій кілька комп‘ютерів обмінюються інформацією, яка потім записується в базу даних. «Клієнт» - це програма, яка використовується для введення інформації та відправки її на сервер для подальшої обробки, а «сервер» - це програма, яка обробляє отриману інформацію, записує її до бази даних, зчитує інформацію з бази даних та відправляє її на клієнт, для виведення користувачеві.

Веб-додаток звільняє розробника від відповідальності за розробку для певного типу комп‘ютера та операційної системи, тому їх може використовувати кожен, хто має доступ до інтернету. Оскільки клієнт запускається в веб-браузері, користувач може використовувати операційні системи такі як *Windows* або *MacOS*. Виконується в різних браузерах як на комп‘ютерах так і на мобільних пристроях.

Веб-додатки зазичай використовують комбінацію сценаріїв на сервері (*ASP.NET, PHP* та ін.) та сценаріїв на клієнтській частині (*HTML, CSS, JavaScript* та ін.).

При цьому у користувача немає доступу до серверної частини, тільки до клієнтської, цим обумовлюється безпека опрацьовуваних даних.

Основні переваги веб-додатків:

- Веб-додатки можуть використовуватися в будь якій операційній системі, оскільки вони підтримують сучасні браузери.
- Через те, що у веб-додатку використовується один і той самий код, порівняно з desktop програмами, додатки легше підтримувати.
- Додаток простіше програмувати, адже цей процес не включає в себе багато роботи з елементами ПК (ядро, процесор, відеокарта).
- Ні відміну від мобільних додатків, для веб-додатків не потрібне схвалення жодних платформ, щоб випустити програму.
- Це більше економічний варіант для будь-якого підприємства, оскільки вони не вимагають підписки чи покупки ліцензій, а можуть використовуватися як SaaS-сервіс, що значно дешевше. Але є виключення для деяких сервісів, за які треба платити.

Веб-додаток та веб-сайт здається поняття ідентичні, але у них є досить суттєві відмінності, які представлені в таблиці 1.1.

Таблиця 1.1.

Відмінності веб-додатків та веб-сайтів

Параметр	Веб-додаток	Веб-сайт
Основне призначення	Створюється для того, щоб взаємодіяти з користувачем	Має на увазі лише наявність статей, без будь якої можливості впливу на читача
Взаємодія з користувачем	Користувач може проводити маніпуляції	Користувач може читати інформаційний

	з даними, але з обмеженим доступом	контент, але ніяк не може його змінювати
Аутентифікація	Більшість додатків вимагають аутентифікації, щоб користувач міг скористатися програмою	Для сайтів не потрібна обов‘язкова аутентифікація. Сайт може лише мати пароль, який відсилає адміністратор ресурсу.
Завдання та складність	Веб-додаток має багато функцій та покриває безліч проблем	Веб-сайт відображає статичну сторінку, на якій розміщена інформація
Модифікація проекту	Щоб внести будь які зміни в проект треба проводити ревю коду та після вписувати те, що ви хочете змінити	Досить просто вносити зміни лише зробивши кілька змін в HTNL-коді сторінки

1.2. Як працює веб-додаток

Користувач створює запит, який повинен буте відправитися на веб-сервер за допомогою протоколів, зазвичай це *http* або *https*. Після того, як сервер отримав запит, він проводить його обробку та звертається до бази даних. Фінальним етапом сервер відправляє оброблену інформацію, або зчитану з бази даних таким самим запитом на *front*-частину (клієнт), де вона знову обробляється для коректного відображення користувачі.

Існують наступні види веб-додатків:

1) *SPA (Single Page Application)* – односторінкова інтерактивна програма. Користувач перемикається між вкладками, при цьому залишаючись на одній

сторінці, при цьому завантажуються лише необхідні елементи, а не вся сторінка, що грає на користь швидкості.

Прикладом *SPA* є електронна пошта *Gmail*.

2) *MPA (Multi Page Application)* – традиційні багатосторінкові програми. Користувач взаємодіє з додатком, завантажуються нові сторінки. Обмін даних відбувається повільніше. Це зазвичай великі веб-додатки.

   Прикладом *MPA* є інтернет-магазини.

3) *PWA* – прогресивна програма, близька за своїми можливостями, функціями та якістю користувацького досвіду до нативних комп'ютерних програм та мобільних додатків. Чіткого розмежування між не-*PWA* та *PWA* немає, але можна виділити низку характеристик.

   Зокрема *PWA* має містити проксі-шар *(Service Worker)* та *Web App* маніфест. По суті, браузер виступає віртуальною машиною для запуску аеб-додатків, подібно до того, як операційна система запускає програми, а *Android* – *apk*-файли.

Крім технічної класифікації існує також класифікація на основі їх призначень:

1) *E-commerce* системи. Дані системи створюються для того, щоб клієнти могли замовляти і продавати товари без сторонніх осіб. Найбільш яскравими видами *E-commerce* платформ є: маркетплейси, онлайн-каталоги, інтерне магазини.
2) *CRM*-системи. Ці системи розробляються для автоматизації відділу продаж та всіх заявок, що надходять у фірму. За рахунок *CRM*-системи можна: відслідковувати всі продажі компанії, призначати зустрічі, бачити історію взаємодій всіх клієнтів.
3) *ERP*-системи. Це веб-системи, які включають не тільки автоматизацію відділу продаж, аде й усіх ресурсів та підрозділів компанії. Завдяки *ERP*-системі можна бачити ефективність кожного підрозділу та ставити відповідні завдання.
4) Корпоративні портали. Це програми, які виступають в ролі спеціального модуля для холдингу. За рахунок цього додатку можуть вирішуватись такі

проблеми, як: швидке інформування всіх співробітників компанії, корпоративне навчання, контроль співробітників (*hr*-модуль).

Розробка веб-додатків поділяється на такі етапи:

- Постановка задачі;
- Складання технічного завдання;
- Прототипування;
- Створення дизайну;
- Програмування;
- Тестування;

1.3. Порівняння Веб-додатків та комп‘ютерних програм

Встановлення, оновлення.

Веб-додаток не вимагає встановлення, всі оновлення виконуються на сервері і миттєво розгортаються для користувачів - досить оновити сторінку або вийти з системи та знову увійти. Але іноді потрібно встановити додаткові бібліотеки або використовувати безпечні мережеві протоколи, щоб змусити його працювати.

Комп‘ютерні програми необхідно встановити на комп'ютер або мобільний пристрій та оновлювати кожного разу, коли виходить нова версія. Незважаючи на те, що процес здебільшого автоматизований, він все ж таки забирає у користувачів час і ресурси пристрою. Крім того, потрібно відстежувати версії на кожному комп'ютері, смартфоні та планшеті.

Публікація / розгортання.

Веб-додаток публікується на локальному або віддаленому сервері, і саме там відбувається процес оновлення. При цьому обов‘язково потрібен сервер. Адже, крім *Front-end* (клієнт), з яким користувачі працюють через браузер, потрібно розмістити *Back-end* (сервер).

Комп‘ютерна програма має бути встановлена вручну на кожному пристрої. У компанії з великою кількістю робочих місць це може забрати багато часу.

Перевага полягає в тому, що не потрібно вибирати сервер або шукати ресурси для публікації, якщо це не клієнт-серверне рішення.

Надійність.

Робота веб-додатка залежить не тільки від того, наскільки правильно він створений та характеристик пристрою, але й від швидкості інтернет-з'єднання та продуктивності віддаленого сервера.

Комп‘ютерна програма працює в автономному режимі, тому найважливіші аспекти це якість коду та стабільність обладнання, на якому цей код працює. Однак якщо зв'язок із сервером необхідний, то виникають ті ж проблеми, що і з веб-додатком.

Доступність.

Веб-додаток доступний з будь-якої точки світу і з будь-якого пристрою, а файли користувача завжди під рукою. Але лише за наявності підключення до інтернету або реалізована можливість роботи в автономному режимі та завантаження і вивантаження даних.

Комп‘ютерна програма доступна завжди, але тільки з того пристрою, на якому вона встановлена. Для роботи з різними пристроями потрібно встановити її на кожен із них, а також з'ясувати, де зберігаються файли, щоб завжди був доступ до них.

Кросплатформенність.

Буде це настільний комп'ютер, ноутбук, планшет або смартфон, веб-додаток однаково добре працює на будь-якому пристрої, оскільки він практично не залежить від обладнання або операційної системи. Головне - правильний браузер. Як правило, для більшості веб-додатків підходять браузери *Google Chrome, Mozilla Firefox, Apple Safari* або *Microsoft Edge/Internet Explorer*.

Комп‘ютерна програма залежить від операційної системи, процесора, відеокарти та інших параметрів. Доводиться враховувати нюанси кожного середовища, писати сценарії з урахуванням можливих варіантів, наймати окремих розробників або навіть цілі команди для версій для різних операційних систем.

Функціональність, швидкодія.

Веб-додаток повністю залежить від браузера та його технологій. Існує ряд обмежень, таких як доступ до обладнання вашого пристрою. Ці та деякі інші обмеження на даний момент неможливо оминути. Але ряд проблем можна вирішити за принципом «Що не можна переписати, можна надбудовувати чи розширювати». Редактори документів, зображень, звуку, відео, *3D* графіки; системи управління проектами; файлове сховище; Конструктори без коду – успішно працюють у браузерах. Інструменти швидкої інтеграції сервісів та інтерфейсних бібліотек ще більше розширюють існуючі можливості.

Комп‘ютерна програма дозволяє реалізувати буквально будь-яку функцію - у цьому вона явно перевершує Інтернет. У будь-якому разі повноцінного онлайн-аналогу *Photoshop* або *Sony Vegas* ще нема. Системні утиліти, безумовно, є сферою розробки настільних комп'ютерів. Як і з програмами, які повинні довго працювати у фоні, наприклад, чати, працювати з ними через браузер просто незручно. Також таке програмне забезпечення частіше використовується для конкретних проектів з нестандартними інтерфейсами або функціями. Тому веб-розробка поки що не загрожує десктопним програмістам — ці технології розвиватимуться паралельно, але під різні завдання.

Щодо швидкості роботи все не так однозначно, як може здатися. Хоча браузерний клієнт постійно спілкується із сервером, його продуктивність багато в чому залежить від того, наскільки правильно він спроектований, «чистоти» коду, апаратних можливостей та стабільності каналу зв'язку. Відмінності у продуктивності, які виявляються під час тестування, часто залишаються непоміченими для користувачів.

Безпека.

Веб-додаток, розроблений з використанням сучасних протоколів та інструментів безпеки, може повністю забезпечити безпеку даних. Однак на деякі моменти розробники не мають жодного впливу: браузер, віддалений сервер, канал зв'язку – вони можуть підвищити рівень безпеки за допомогою додаткових засобів перевірки, але й знизити його за рахунок вразливостей. Явний плюс для користувачів: таке програмне забезпечення легше контролювати. Обмеження

середовища знижують можливість прихованого доступу до файлів або запуску процесів.

Комп‘ютерна програма налаштовується гнучкіше, а значить, всі потенційні вразливості теоретично можна передбачити при розробці. Насправді це малоймовірно. Однак зробити її повністю безпечною все ж таки можливо. Але лише в тому випадку, якщо пристрій, на якому вона встановлена, не підключатиметься навіть до захищеної локальної мережі. В іншому випадку ризик залишається.

Важко (якщо не неможливо) однозначно сказати, що безпечніше. На це впливає багато факторів, насамперед людський. Але мета всіх заходів безпеки якраз і полягає у захисті від людського фактору в різних його проявах.

Але довіра до десктопного софту вища. Деякі організації принципово не згодні з роботою в браузерах, багато користувачів досі відносяться до них із підозрою. Проте ситуація змінюється — із розвитком технологій зростає лояльність людей.

1.4. Висновки до розділу

Веб-додаток – це будь-яка програма, яка використовує веб-браузер. Програма може бути простою, як, наприклад, дошка оголошень або контактна форма на сайті, або складною як тестовий процесор чи багатофункціональний додаток, який має вигляд сайту, але функціоналом зрівнюється з великим додатком, який завантажується на комп‘ютер чи мобільний телефон.

Можливості браузерної розробки величезні, і їх потенціал далеко не вичерпаний. Технології розвиваються, IT-ринок зростає, пропонуючи все нові й нові програми - роблять вибір на користь Інтернету просто тому, що він зручніший. Коли ми говоримо про рішення для корпоративних замовників, без браузерних програм не обійтися. Вони гнучкі, універсальні, не вимагають попередньої підготовки середовища та дозволяють економити фінанси компанії, апаратні ресурси та час співробітників.

РОЗДІЛ 2
ОСОБЛИВОСТІ РОЗРОБКИ ВЕБ-ДОДАТКІВ

2.1. Технології проектування веб-додатків

Сьогодні веб-додатки отримали великий розвиток у різних сферах життя суспільства. Робота веб-додатка здійснюється через технологію клієнт-сервер, в якій клієнт це браузер, а сервер - веб-сервер. За початку наведемо загальну схему веб-застосунків, показану нижче на рис. 2.1.

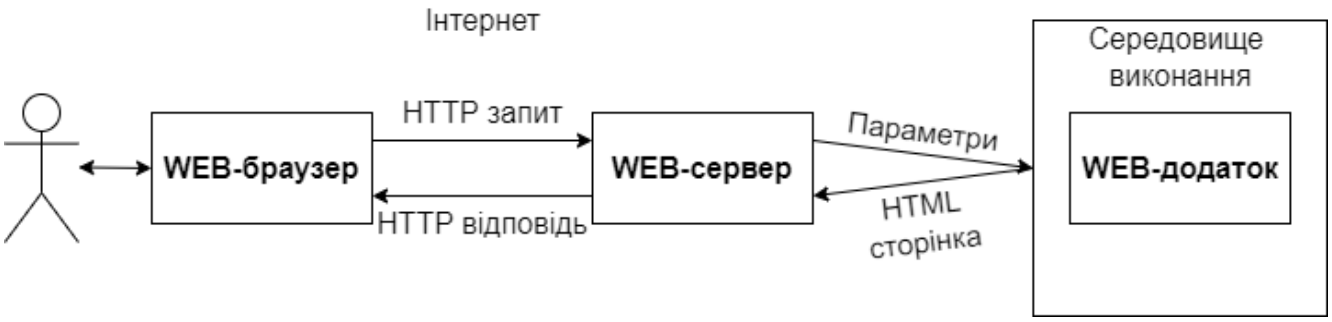

Рис. 2.1 Принцип роботи веб-додатку

На рис. 2.1 показано, що клієнт, який звертається до веб-браузера, відправляє *HTTP*-запит на певну *URL*-адресу, яка вказує на динамічний ресурс, а саме на сам веб-додаток. Далі сервер формує на основі веб-додатку *HTML*-сторінку, яка за допомогою браузера відображається клієнту. З опису схеми можна зробити висновок, що основна робота веб-програми виконується на стороні сервера.

На даний момент існує багато технологій, що реалізують логіку веб-додатків на стороні сервера.

Першою широко використовуваною технологією була *CGI (Common Gateway Interface)*, яка особливо підходить для створення динамічних веб-сторінок та використовується для зв'язку між клієнтом (веб-браузером) та веб-сервером.

Ця технологія являє собою набір правил, що дозволяють програмі працювати на різних серверних операційних системах.

Відповідно до технології *CGI, HTTP*-запит, що містить посилання на динамічну сторінку до веб-сервера, створює новий процес та запускає потрібну прикладну програму. Технологія *CGI* дозволяє використовувати будь-яку мову програмування, здатну працювати з пристроями вводу/виводу. Крім того, при розробці веб-застосунку можна використовувати сценарії *CGI*, такі як *Python, Perl, Tcl* тощо. Коли програма *CGI* містить сценарії, її виконання викликає обробник сценаріїв (інтерпретатор сценаріїв), в *HTTP*-запит якого передаються дані та ім'я файлу, який містить запитуваний сценарій. Після запуску цього скрипта програма повертає згенеровану *HTML* сторінку клієнту.

Незважаючи на те, що технологія *CGI* дозволяє динамічно генерувати інформацію в інтернеті, вона має істотні недоліки. Один з основних недоліків – продуктивність. Причина такої низької продуктивності криється в обробці *HTTP*-запиту: на кожну обробку такого запиту веб-сервер генерує новий процес, який завершується лише після завершення програми, а якщо процесів багато, вони починають конкурувати за ресурси оперативної пам'яті.

Наступною технологією, що набула досить широкого поширення, була *Java Servlets*, або просто сервлети. Ця технологія вирішує проблему продуктивності, виконуючи всі запити в одному процесі та розподіляючи їх по потоках усередині процесів. Це означає, що код сервлета буте безпечним. Перевага використання сервлетів також полягає в їхній незалежності від платформи, оскільки вони працюють на віртуальній машині *Java*. *Java Servlets* володіють широким набором функцій, що можна досягти завдяки великій кількості бібліотек.

Java Servlet Pages (JSP) є розширенням технології *Java Servlet*, раніше розробленої *Sun Microsystems*, що означає, що їх архітектури взаємопов'язані. Ця

технологія забезпечує швидку та спрощену розробку веб-додатків з використанням шаблонного підходу. Шаблони сторінок *JSP* дуже схожі на шаблони сторінок *HTML* та шаблони *ASP* та *PHP*. Відмінність цієї технології від аналогічних технологій полягає в тому, що при доступі до сторінки код у тегах не інтерпретується, а попередньо компілюється в *Java Servlets*. Ця процедура виконується при першому запуску контейнера сторінки або сервлета, тому що описаний вище процес займає багато часу. *JSP* гармонійно поєднує у собі шаблонну реалізацію сторінок та всі переваги платформи *Java*.

Новою технологією розробки веб-застосунків є технологія *.NET*, розроблена Microsoft. Платформа *.NET* значно спростила процес розробки додатків та підвищила надійність коду. Стали доступні функції автоматичного управління часом життя об'єктів, обробки винятків та налагодження, з'явилися бібліотеки та нейтральні мови програмування. Набір стандартних базових класів надає розробнику доступ до служб платформи за допомогою будь-якої *.NET*-сумісної мови програмування. Загальномовне середовище виконання разом з базовими класами утворює ядро платформи. (нове покоління *ASP*, що дозволяє використовувати будь-яку мову, сумісну з *.NET*), а також *Windows Forms* та *Web Forms* (класи, що реалізують локальні та веб-програми). Вихідний код компілюється за такою схемою: генерується код проміжної мови *(Microsoft Intermediate Language)*. На відміну від старої версії, де компілятор генерував власний код, цей тип компіляції дозволяє запускати скомпілований файл на будь-якій платформі процесу. Нові функції *ASP.NET* відповідають останнім вимогам. Ось деякі з них:

- широкий вибір бібліотек;
- незалежність від мови платформи;
- нові засоби обробки помилок;

2.2. Клієнтська частина програми

Front-end, тобто клієнт, це інтерактивна частина програми, яка запускається у веб-браузері на комп'ютері, смартфоні або планшеті.

Клієнтська частина реалізує інтерфейс веб-додатка і завантажується у вигляді динамічних веб-сторінок.

Розглянемо основні компоненти стека інтерфейсів.

HTML — це стандартна мова розмітки, яка використовується для створення веб-проектів. З його елементами можна відображати будівельні блоки сторінок, а також представляти форматований текст, зображення, таблиці, форми введення даних тощо. Документ *HTML* містить додаткову інформацію про стиль або макет на мові *CSS* та про поведінку на мові *JavaScript*.

Каскадні таблиці стилів *(CSS)* - це мова розмітки, що визначає зовнішній вигляд та розташування елементів *HTML*. *HTML* визначає структуру, а *CSS* – стиль елементів. Шрифти, кольори, стилі, розміщення окремих елементів та відображення сторінок на різних кінцевих пристроях задаються за допомогою *CSS*.

JavaScript – це мова програмування, яка допомагає реалізувати складну поведінку веб-сторінки. У більшості випадків *JavaScript* використовується для створення привабливих інтерактивних елементів для веб-сторінок, які покращують взаємодію користувача. За допомогою *JavaScript* ви можете створювати меню, анімацію, відеоплеєри, інтерактивні карти та навіть прості браузерні ігри.

Для більш ефективної розробки *JavaScript* розробники використовують фреймворки, які є важливою частиною стека технологій. Фреймворк — це робоче середовище, тип каркасу, що допомагає ефективніше створювати та підтримувати програмні продукти.

2.3. Популярні фреймворки та бібліотеки *JavaScript*

Бібліотек та фреймворків для мови *JavaScript* на сьогоднішній день є дуже багато для виконання найрізноманітніших завдань, починаючи з розробки веб-

додатків, закінчуючи прикладними програмами для комп‘ютерів та мобільних пристроїв.

Розглянемо найпопулярніші фреймворки та бібліотеки для створення клієнтської частини веб-додатків:

- *React* — це бібліотека з відкритим вихідним кодом для створення інтерфейсів користувача. За допомогою *React* розробники створюють веб-програми, які змінюють контент без перезавантаження сторінки. Це дозволяє програмам швидко реагувати на дії користувача, наприклад, коли вони заповнюють форми, застосовують фільтри, додають товари в кошик і таке інше. Однією з головних рис *React* є його універсальність. Цю бібліотеку можна використовувати на серверних та мобільних платформах, тут роль відвграє *React Native*. Ще однією важливою можливістю бібліотеки є декларативність. У *React* розробник описує, як компоненти інтерфейсу користувача виглядають у різних станах. Декларативний підхід скорочує код та робить його зрозумілим. *React* базується на компонентах. Кожен компонент повертає частину інтерфейсу користувача зі своїм власним станом. Комбінуючи компоненти, програміст створює завершений інтерфейс веб-програми.
- *Angular* – це фреймворк від *Google*. Насамперед він націлений на розробку *SPA*-рішень. Цей інструмент розробки найбільш відомий тим, що пропонує розробникам найкраще середовище для об'єднання *JavaScript* з *HTML* та *CSS*. До веб-програм на *Angular.js* відносяться *Google* і *Youtube*.
- *Vue* - це прогресивний фреймворк для створення інтерфейсів користувача. На відміну від монолітних фреймворків, *Vue* піддається поетапній реалізації. Його можна легко інтегрувати в інші бібліотеки та існуючі проекти. *Vue* також добре підходить для створення складних односторінкових програм при поєднанні з сучасними інструментами та додатковими бібліотеками.

2.4. Серверна частина веб-додатку

Для програмування серверної частини може використовуватися мова PHP, але не обов'язково вона.

Для розробки веб-застосунків можна використовувати практично будь-які сучасні мови програмування:

- *PHP*
- *Perl*
- *Ruby*
- *Java*
- платформа *.NET* (мови *VB.NET, C#* та інші підтримувані *.NET*)
- *C/C++*

Незалежно від мови, на якій написана серверна частина веб-програми, способи обробки запитів та взаємодії з користувачем залишаються такими ж.

Під серверною частиною розуміється набір програмно-апаратних засобів, з допомогою яких реалізується логіка роботи веб-застосунку. Це відбувається поза браузером та комп'ютером користувача. *Back-end* включає адміністративну панель, управління даними і логіку їх передачі у *Front-end*-запитах.

Завдання проектування серверної частини полягає в тому, щоб гарантувати, що відповідь сервера дійде до клієнта і розроблені блоки працюють належним чином. А також створити для замовника зручне та безпечне середовище для наповнення та оновлення контенту на сайті.

PHP - одна з найбільш широко використовуваних мов веб-розробки. Майже завжди користувачі приходять на веб-сайт за інформацією, яка постійно змінюється, і потрібно переглядати поточний статус. Ця мова програмування приймає запит від веб-сервера, запускає скрипт і повертає результат у вигляді готового HTML-коду. Сервер надсилає цей результат у браузер користувачеві, який, у свою чергу, відображає його.

Для створення веб-застосунків на PHP є багато фреймворків, найпопулярніші серед них є Laravel, Yii2 та Symfony. Їх функції: підтримують

функціональне, інтеграційне та модульне тестування, полегшують масштабування додатків, слідують передовим методам розробки та пропонують широкий спектр шаблонів проектування. Все це допомагає підтримувати чисту, мінімальну та ефективну базу коду, яку легко модифікувати.

Java — мова програмування загального призначення, яка зазвичай використовується як для веб-розробки, так і для розробки під *Android*. До переваг веб-додатків *Java* відносяться кросплатформенність, багатофункціональність, надійність та гнучкість.

Для розробки веб-програм на *Java* найчастіше використовується фреймворк *Spring*. Він пропонує комплексну модель розробки та налаштування для сучасних бізнес-додатків від проектів електронної комерції до великих порталів, від освітніх платформ до державних ресурсів. Ключовим елементом *Spring* є підтримка інфраструктури прикладного рівня, що дозволяє розробникам зосередитись на бізнес-логіці без непотрібного налаштування, залежного від середовища виконання. Також використовується Hibernate із фреймворків Java, що полегшує розробку програми для взаємодії з базою даних.

Node.js - платформа, яка запускає код *JavaScript* поза браузером. *Node.js* дозволяє розробникам використовувати *JavaScript* для отримання інструментів командного рядка. На стороні сервера його можна використовувати для запуску сценаріїв для обробки контенту динамічної веб-сторінки до того, як він буде доступний у веб-браузері користувача.

2.5. Бази даних у розробці веб-додатків

Мови програмування, які традиційно використовуються для веб-розробки (*Perl, PHP, ASP* та інші), дозволяють реалізувати практично будь-яке завдання. Але з їхньою допомогою складно обробляти великі обсяги даних, до того ж складних за структурою. Розробка таких програм потребує все роботи програмістів, обсяг програмного коду та кількість помилок зростають у геометричній прогресії, а надійність програмного забезпечення знижується.

У такій ситуації на допомогу програмісту приходять бази даних. Згідно з класичним визначенням, база даних - це впорядкований набір інформації, що зберігається в наборах, кожен із яких містить записи однорідного типу. Системи управління базами даних (СУБД) пропонують програмісту найпотужніші інструменти для створення, оновлення та обробки великих обсягів інформації зі складною структурою.

У класичній теорії існує три типи, три структури баз даних: ієрархічна, мережева та реляційна. Нині переважають реляційні бази даних.

Ієрархічна структура представляє сукупність елементів, пов'язаних між собою за певними правилами, схема ієрархічної моделі бази даних показана на рис. 2.2.

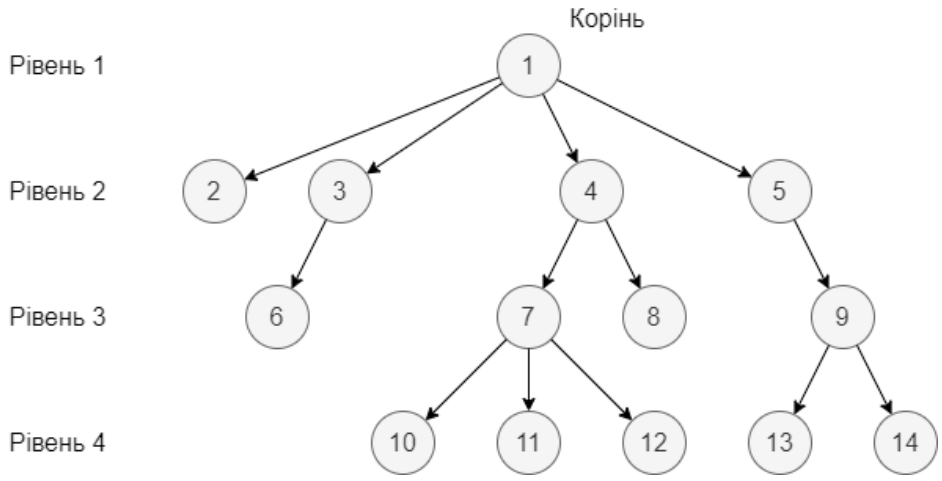

Рис.2.2 Ієрархічна структура баз даних

Об'єкти, пов'язані ієрархічними відносинами, утворюють орієнтований граф. До основних понять ієрархічної структури відносяться рівень, елемент (вузол), зв'язок. Вузол – це набір атрибутів даних, що описують об'єкт. В ієрархічній деревоподібній діаграмі вузли представлені вершинами графа. Кожен вузол нижчого рівня з'єднаний лише з одним вузлом вищого рівня. Ієрархічне дерево має лише один вузол, не підпорядкований жодному іншому вузлу і що знаходиться на верхньому (першому) рівні. Залежні вузли перебувають на другому, третьому рівні тощо. Кількість дерев у базі даних визначаються кількістю основних записів. Для кожного запису бази даних існує лише один

(ієрархічний) шлях від кореневого запису. До основних недоліків ієрархічних моделей відносяться:

- неефективна реалізація відносин типу N:N;
- повільний доступ до сегментів даних нижніх ієрархічних рівнів;
- чітка спрямованість на певні типи запитів;

Мережева модель бази даних - це набір об'єктів різного рівня, де кожен об'єкт може бути пов'язаний з іншими. схема мережевої моделі баз даних показана на рис. 2.3.

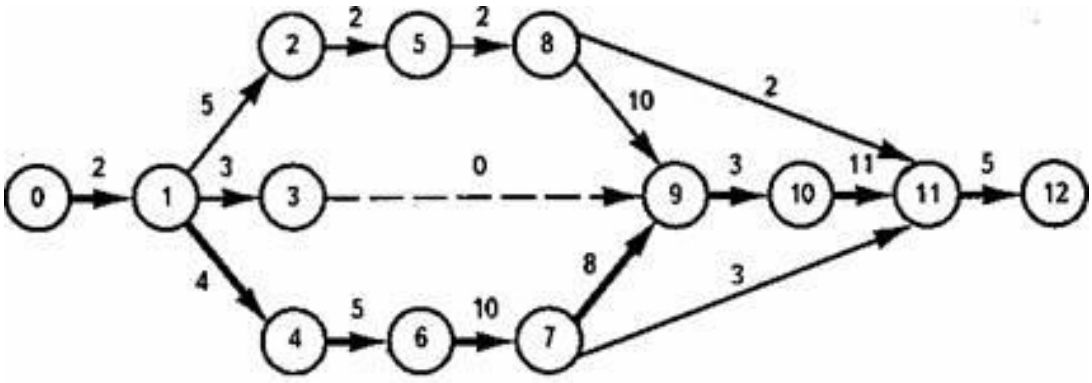

Рис.2.4 Схема мережевої моделі баз даних

Мережева модель даних аналогічна до ієрархічної. Вона має самі основні компоненти (вузол, рівень, з'єднання), але характер їх взаємозв'язку принципово інший. У мережній моделі приймається вільний зв'язок між елементами різного рівня.

Вершина, вузол — це базове поняття: точка, де ребра й дуги можуть сходитися і виходити. Мережа - це спрямована множина кінцевих графів з початковою вершиною (джерело) і кінцевою (стік). Таким чином, мережева модель являє собою граф виду «мережа». Рівень - це уявна горизонтальна лінія або площина, яка є обмеженням висоти. Цей термін часто використовується як міра величини, розвитку, значущості або ж ступінь якості.

Реляційна модель баз даних відрізняється простотою структури даних, зручним табличним представленням, можливістю використання формального алгебраїчного реляційного апарату та реляційних обчислень для обробки даних.

Реляційна модель вимагає організації даних як двомірних таблиць. Кожна реляційна таблиця є двовимірним масивом і має наступні властивості:

- кожна клітинка таблиці є елементом даних;
- всі стовпці таблиці однорідні, тобто всі елементи стовпця мають однаковий тип даних (число, символ тощо) і однакову довжину;
- кожен стовпець має унікальне ім'я;
- у таблиці немає однакових рядків;
- Порядок рядків і стовпців, що передаються, може бути довільним.

Кожен рядок таблиці є записом, що відповідає описуваному об'єкту. Кожен стовпець є атрибутом об'єкта, що є його властивістю. Клітинки таблиці містять дані; Кожна клітинка представляє значення атрибута об'єкта, що відповідає цьому рядку. Кожне значення має інформацію про об'єкт.

Лідером серед баз даних, які сьогодні використовуються для розробки веб-додатків, безсумнівно, є *MySQL*. Основна перевага *MySQL* – це її простота. Як наслідок, найвища швидкість при виконанні *SQL*-запитів та необхідність явного програмування основних правил збереження цілісності та несуперечності даних на рівні сервера додатків.

Інші бази даних, які використовуються для веб-розробки, включають *Oracle* та *PostgreSQL*. *PostgreSQL* - це безкоштовне ядро бази даних з відкритим вихідним кодом, призначене в першу чергу для роботи в *UNIX*-подібних системах. Але *Oracle* явний лідер на ринку високопродуктивних комерційних корпоративних баз даних.

2.6. Висновки до розділу

На сьогоднішній день веб-додатки стали дуже популярними, та швидко розвиваються. Основний принци роботи веб-додатку полягає в тому, що людина взаємодіє з клієнтською частиною програми через свій браузер. Звідти відправляє запит за допомогою *HTTP*-протоколу на серверну частину, куди користувач доступу не має. На серверній частині спрацьовує сценарій, обробляє інформацію,

за потреби звертається до бази даних та в кінці повертає відповідь також протоколом *HTTP*, або захищеним протоколом *HTTPS*.

Клієнтська частина, або *Front-end* створюється за допомогою стандартизованої мови розмітки HTML, мови каскадних таблиць стилів та мови програмування JavaScript задля створення інтерактивності сторінки. До розробки можуть залучатися різні *Front-end* бібліотеки та фреймворки на мові *JavaScript* для кращої взаємодії з користувачем та інтерактивності.

Серверна частина створюється за допомогою однієї або кількох мов програмування. Завдання Back-end частини полягає в обробці даних та посиланню відповіді клієнту для подальшого відображення. Серверна частина має змогу працювати з базами даних для обробки великих об‘ємів інформації.

Бази даних існують кількох типів: ієрархічна, мережева, реляційна. На сьогоднішній день найпопулярнішими є реляційні бази даних. Для роботи з базами даних існують системи управління базами даних (СУБД).

РОЗДІЛ 3
СТВОРЕННЯ ВЕБ-ДОДАТКУ

3.1. Структура веб-додатку

Під час розробки додатку для розміщення та редагування статей було використано наступні технології:

- Веб-браузер *Google Chrome*;
- Інтегроване середовище розробки (*IDE*) *PhpStorm*;
- Мова гіпертекстової розмітки *HTML*;
- Мова каскадних таблиць стилів *CSS*;
- Бібліотека шаблонів *Bootstrap*;
- Мова програмування *PHP*;
- Шаблон проектування *MVC* (*Model View Controller*);
- Шаблон проектування *ActiveRecord*;
- Шаблон проектування *ORM* (*Object-Relational Mapping*);
- Шаблон проектування *Singletone*;
- Система управління базами даних *MySQL*;
- Інтерфейс для зручної роботи з базами даних *phpMyAdmin*;
- Локальний сервер *Open Server*;
- Сервер *Apache*;

Сценарії сторінок клієнтської частини знаходяться в файлах:

- Папка *articles*:
  - *add.php* – сторінка створення статті;
  - *deleted.php* – сторінка інформації про видалення статті;

- *edit.php* – сторінка редагування статті;
- *view.php* – сторінка, на якій виводиться стаття та коментарі до неї;

- Папка *comments*:
  - *edit.php* – сторінка редагування статей;
- Папка *errors*:
  - *401.php* – помилка авторизації;
  - *403.php* – помилка доступу;
  - *404.php* – помилка 404, сторінка не знайдена;
  - *405.php* – помилка 405, користувач не знайдений;
  - *406.php* – помилка 406, невірний код активації;
  - *500.php* – помилка сервера;
- Папка *mail*:
  - *userActivation.php* – шаблон підтвердження користувача;
- Папка *main*:
  - *main.php* – сторінка, на якій виводиться список всіх статей;
- Папка *users*:
  - *cabinet.php* – кабінет користувача;
  - *login.php* – сторінка логіну;
  - *signUp.php* – сторінка реєстрації;
  - *signUpSuccessful.php* – сторінка інформацій про успішну реєстрацію;
- Папка *admin*:
  - *articles.php* – інформація про всі статті;
  - *comments.php* – інформація про коментарі;
  - *users.php* – інформація про користувачів;
  - *view.php* – сторінка адміністративної панелі;

«Шапка» (верхня частина сайту та навігаційна панель) додатку та «підвал» (нижня частина сайту) знаходяться в файлах:

- *header.php*

- *footer.php*

3.2. Розробка клієнтської частини додатку

Для розробки клієнта веб-додатку використовуємо бібліотеку шаблонів *Bootstrap*, яка в собі містить готові компоненти на мові *HTML*, стилізовані за допомогою *CSS* та *JavaScript* з використанням бібліотеки *jQuery*.

Голова та підвал сайту у нас будуть незмінними. Отже для зручності розробки розіб‘ємо сторінку на три частини: хедер (голова сайту), основний контент та футер (підвал сайту). Ці три частини розділимо на окремі файли, серед яких *header.php* і *footer.php*. Основний контент для кожної сторінки буде різний, і змінюватиметься в залежності від вибраних параметрів, отже кожна сторінка це окремий файл, в який імпортуємо голову та підвал. Сторінки будуються таким чином:

```
<?php include __DIR__ . ‘/../header.php’?> //Імпортуємо голову сайту
<p>Привіт, <?=$name?>!</p> //Головний контент
<?php include __DIR__ . ‘/../footer.php’?> //Імпортуємо підвал сайту
```

В прикладі можна побачити таку конструкцію *<?=$name?>*, яка означає, що на це місце буде підставлено значення змінної *$name*, це особливості мови *PHP*, конструкції якої можна вбудовувати одразу в *HTML*, і навпаки, код *HTML* можна вбудовувати в сценарій *PHP*.

Перехід між сторінками в додатку відбувається за допомогою механізму роутингу. URI шлях будується таким чином: *http://<назва проекту>/<шлях>?<GET параметри>*

Приклад шляху до сторінки статті буде виглядати наступним чином:

http://blog/articles/4

В цьому прикладі видно, що сторінка буде виводити статтю, ідентифікатор якої 4.

Всі шляхи прописуються в файлі *routes.php*.

Для того, щоб взаємодіяти з додатком, наприклад, залишати коментарі до статей, а не тільки їх читати, користувачу слід увійти в свій обліковий запис, або, якщо його нема, зареєструватися. Реєстрація відбувається на сторінці *register.php*.

Форма має такий вигляд:

```
<form action=”/users/register” method=”POST”>
    <div>
        <label for=”email”>
        <input type=”email” name=”email” id=”email” placeholder=”Email…”>
    </div>
    <div>
        <label for=”nickname”>
        <input type=”text” name=” nickname” id=” nickname” placeholder=” Nickname …”>
    </div>
    <div>
        <label for=”password”>
        <input type=”password” name=” password” id=” password” placeholder=” Password …”>
    </div>
    <input type=”submit” value=”Реєстрація” >
</form>
```

Така форма, при натисканні відправляє дані методом POST на адресу */users/register*. Після цього на сервері дані обробляються та повертається відповідь у вигляді HTML-сторінки з повідомленням про реєстрацію, або помилку з текстом, що саме сталося.

Дані відправляються з форми методом *POST*, в якому дані зберігаються у вигляді ключ-значення. Після цього, для їх обробки в мові *PHP* існують глобальні змінні, якими можна користуватися з будь-якої точки програми. Для обробки

даних, які відправлялися методом *POST* є відповідна глобальна змінна *$_POST*, яка являє собою асоціативний масив.

Після того, як користувач увійшов у свій обліковий запис, він має можливість залишати коментарі до існуючих статей, видаляти та редагувати коментарі, які залишив цей користувач. Створювати статті має можливість тільки користувач з правами доступу «адміністратор». Також адміністратор має можливість редагувати і видаляти статті, та всі коментарі до статей.

Для створення зовнішнього вигляду використовуємо сітку *Bootstrap*, яка будується з рядків та дванадцяти стовпчиків. Формується сітка таким чином, що в контейнері *div* з класом *.row* розміщуються контейнер з класами *col-* з числами від 1 до 12.

Приклад двох контейнерів, які по горизонталі займають відповідно 3 та 9 стовпців на великих екранах та 6 і 6 на середніх:

```
<div class=”container”>
    <div class=”row”>
        <div class=”col-lg-3 col-md-6”>
        <div class=”col-lg-9 col-md-6”>
    </div>
</div>
```

Для адаптації сторінок використовуються суфікси *xs, sm, md, lg, xl, xxl*. Щоб адаптувати сторінку, до класів *col* додають один із суфіксів, наприклад: *col-md-3, col-sm-9*. Зміни розмірів сітки показано в таблиці 3.1.

Таблиця 3.1

Зміна розмірів сітки *Bootstrap*

Позначення	Розміри сторінки	Назва класу
1	2	3
xs	<576 пікселів	*col-*
sm	≥576 пікселів	*col-sm-*
md	≥768 пікселів	*col-md-*

lg	≥992 пікселів	*col-lg-*
xl	≥1200 пікселів	*col-xl-*
xxl	≥1400 пікселів	*col-xxl-*

3.3.Налаштування локального серверу

Open Server Panel – це безкоштовне портативне програмне середовище, спеціально розроблене для веб-розробників з урахуванням їх рекомендацій та запитів.

Цей програмний пакет містить ретельно підібраний набір серверного програмного забезпечення та практичну і продуману утиліту управління з потужними можливостями налаштування для всіх доступних компонентів. *OSPanel* широко використовується для розробки, налагодження та тестування веб-проектів, а також розгортання веб-сервісів у локальних мережах.

Для того, щоб користуватися сервером, спершу його треба завантажити, для цього можна використати офіційний сайт *https://ospanel.io/*. Після завантаження та встановлення програми, на панелі задач Windows з‘явиться червоний прапорець. Це свідчить про те, що сервер встановлений, але не запущений. Спершу слід його налаштувати. Обираємо модулі, які будемо використовувати, вони зображені на рис. 3.1.

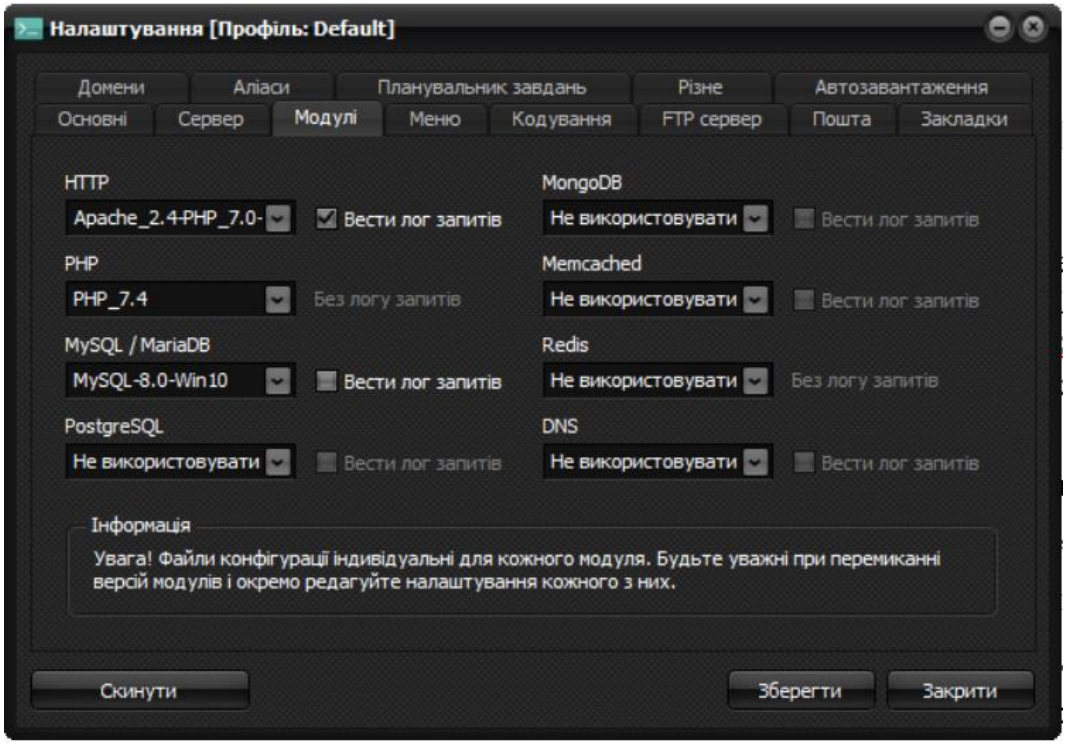

Рис.3.1 Підключення модулів до локального серверу.

Для свого проекту я підключив наступні модулі:

- *Apache 2.4* – локальний сервер;
- *PHP 7.4* – підтримувана мова програмування;
- *MySQL 8.0* – сервер СУБД;

Всі інші налаштування залишаються за замовчуванням. Для коректної роботи додатку слід налаштувати локальний сервер відносно проекту. Для цього створюється в папці проекту спеціальний файл із зарезервованою назвою *.htaccess* в якому пишуться спеціальні директиви. Приклад директив:

RewriteEngine on

RewriteCond %{SCRIPT_FILENAME} !-d

RewriteCond %{SCRIPT_FILENAME} !-f

RewriteRule ^(.)$ ./index.php?route=$1 [QSA,L]*

Директорія додатку повинна знаходитися в кореневий папці *Open Server* в директорії *domains*, щоб відкрити проект в браузері треба в першу чергу запустити сервер, щоб червоний прапорець змінився на зелений та в пошуковій стрічці написати назву проекту і додати порт :80, це порт за замовчуванням, тому

деякі браузери дозволяють його відкидати. Приклади *url*-адреси для відкриття проекту:

- *http://blog:80/*
- *http://blog/*

3.4. Створення серверної частини додатку

За серверну частину веб-додатку відповідає мова програмування скриптова мова програмування *PHP* версії 7.4, яка вільно розповсюджується інтернетом та може працювати на будь-якому веб-сервері. Для створення архітектури додатку за основу був взятий шаблон проектування *MVC (Model-View-Controller). MVC* — це об‘єктно орієнтований шаблон проектування веб-застосунків, який містить кілька менших компонентів. У *MVC* модель даних програми, інтерфейс користувача і логіка взаємодії з користувачем розділені на три окремих компоненти, тому зміни в одному з цих компонентів практично не впливають на інші.

Основна мета використання *MVC* - відокремити дані та бізнес-логіку від візуалізації. Такий поділ збільшує можливість повторного використання програмного коду: наприклад, додавання представлення даних існуючого маршруту не тільки в вигляді *HTML*, а й у форматі *JSON*, *XML*, *PDF*, *XLSX* стає дуже простим і не потребує будь-яких змін вихідного маршруту. рівня бізнес-логіки Це також спрощує обслуговування коду: наприклад, зміни зовнішнього вигляду не впливають на бізнес-логіку, а зміни бізнес-логіки не впливають на візуалізацію.

Концепція *MVC* розділяє лані, представлення та обробку дій користувача на компоненти:

- Модель *(Model)* - надає об'єктну модель конкретної предметної області, містить дані та методи роботи з цими даними, відповідає на запити контролера, повертаючи дані та/або змінюючи свій стан. При цьому модель не містить інформації про способи візуалізації даних

або формати їх подання, а також безпосередньо не взаємодіє з користувачем. Моделі додатку, які містяться в директорії *models*:

- *Article.php;*
- *Comment.php;*
- *User.php;*

- Представлення *(View)* - відповідає за відображення інформації (візуалізацію). Одні й самі дані можуть бути представлені по-різному й у різних форматах. Наприклад, набір об'єктів, що використовують різні уявлення, може бути представлений на рівні інтерфейсу користувача як таблиця або список; На рівні *API* можна експортувати дані в *JSON*, *XML* або *XSLX*. Модель *view* знаходиться в файлі *View.php*.
- Контролер *(Controller)* — забезпечує зв'язок між користувачем та системою, використовує модель та представлення для реалізації необхідної реакції на дії користувача. Як правило, на рівні контролера здійснюється фільтрація отриманих даних та авторизація - перевіряються права користувача на виконання дій або отримання інформації. Список контролерів в папці *controllers*:
  - *AbstractController.php*;
  - *AdminController.php*;
  - *ArticlesController.php*;
  - *CommentsController.php*;
  - *MainController.php*;
  - *UsersController.php*;

Користувач взаємодіє з додатком через контролери. За кожну сутність відповідає окремий контролер, наприклад: за статті відповідає контролер *ArticlesContoller.php*, за користувачів *UsersController.php*, за коментарі – *CommentsController.php*.

Як і з контролерами, моделі також відповідають кожній окремій сутності та таблиці в базі даних, з якою ця модель працює. Зберігають дані таблиць в

об‘єктах класу, де кожна властивість класу відповідає одному стовпцю таблиці, а кожен об‘єкт відповідає за один запис в таблиці. За це відповідає концепція *ORM* або *Object-Relational Mapping* (об'єктно-реляційне відображення) – технологія програмування, яка пов'язує бази даних з концепціями об'єктно-орієнтованих мов програмування. Інакше кажучи, це принцип роботи з БД в об‘єктно орієнтованому коді. Тобто структура та дані в БД відображаються в об'єктах у сценарії. І коли ми працюємо з цими об'єктами, ми змінюємо базу даних і навпаки. Основні принципи:

- Таблицям відповідатимуть окремі класи. Наприклад, таблиці *articles* буде відповідати клас *Article*;
- У класах буде описано властивості об'єктів. Кожна властивість відповідатиме полю в таблиці. Наприклад, буде властивість ->*authorId*, вона буде відповідати стовпцю *author_id* у таблиці *articles*;
- Кожен об'єкт відповідає одному запису у таблиці бази даних. Тобто об'єкт класу *Article* буде відповідати одному рядку таблиці *articles*.

Моделі працюють з даними, тому логіку звернення до баз даних також слід описувати в моделях, а в контролерах користуватись спеціальними методами об‘єктів, щоб чітко відділити дані від основної логіки програми. Тому в веб-додатку реалізована саме така схема роботи з даними. Приклад методу контролера, який дістає статтю за ідентифікатором, коментарі до цієї статті та передає всі дані представленню для виведення:

```
Public function view($articleId):void {
    $article = Article::getById($articleId); //отримуємо об‘єкт моделі article, тобто запис бази даних
    if($article === null){ //перевіряємо, чи існує стаття
        throw new NotFoundException(); // Якщо її не існує «кидаємо» помилку
    }
    $comments = Comment::getCommentByArticleId($articleId); //отримуємо масив об’єктів коментарів
```

$author = $article->getAuthor(); //отримуємо об'єкт автора

$this->view->renderHtml('articles/view.php', ['article' => $article, 'author' => $author, 'comments' => $comments]); //передаємо дані представленню

}

Представлення має єдину функцію, виводити в браузер сформовану сторінку, назва якої передається, та з параметрами, які потрібно на цій сторінці відобразити.

3.5. Проектування баз даних

Для проектування бази даних веб-додатку використовуємо інтерфейс *phpMyAdmin*, який є модулем *Open Server*. Це веб-додаток з відкритим кодом, написаний мовою PHP і є веб-інтерфейсом для адміністрування СУБД MySQL. *PhpMyAdmin* дозволяє через браузер і не тільки здійснювати адміністрування сервера *MySQL*, запускати команди *SQL* та переглядати вміст таблиць та баз даних. Програма користується великою популярністю у веб-розробників, тому що дозволяє управляти СУБД *MySQL* без безпосереднього введення *SQL* команд. Інтерфейс *phpMyAdmin* показано на рис.3.2.

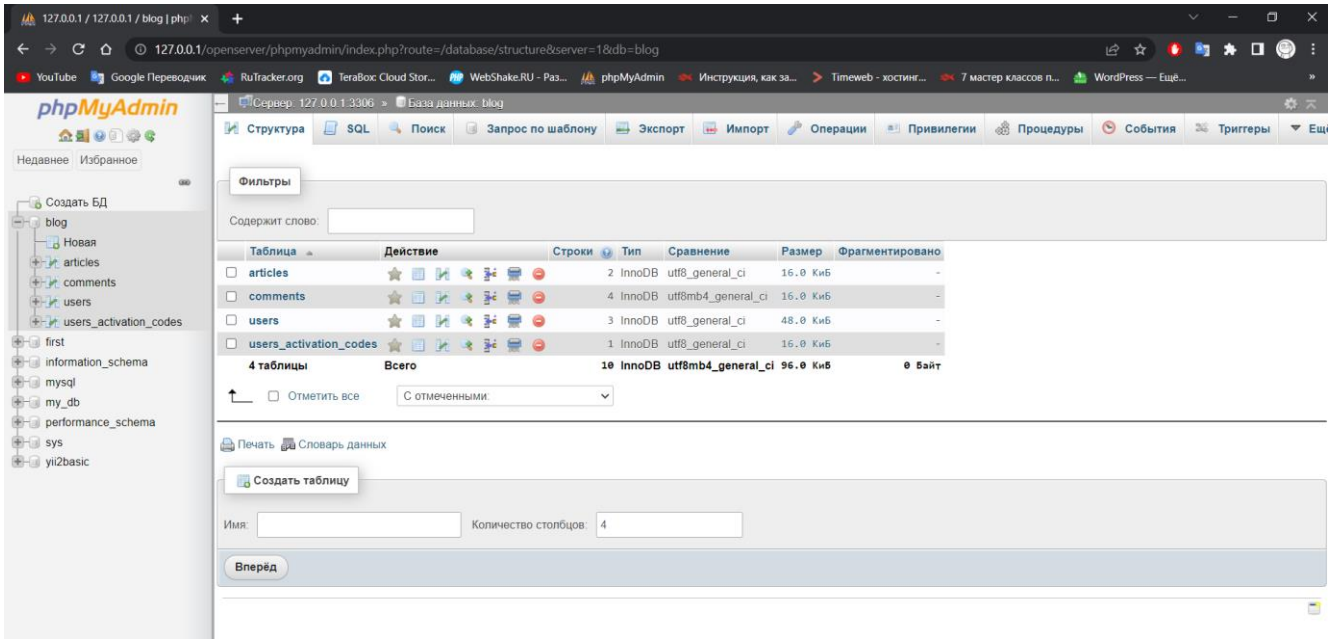

Рис.3.2 Інтерфейс *phpMyAdmin*

Під час розробки веб-додатку необхідно створити кілька таблиць бази даних, які представлені в таблиці 3.2

Таблиця.3.2

Опис таблиць бази даних

Назва	Опис	Особливості
1	2	3
articles	Таблиця статей	Тут розміщуються статті, які можуть додавати тільки користувачі з правами «адміністратор».
comments	Таблиця коментарів	Коментарі можуть додавати всі зареєстровані користувачі. Записи прив‘язуються до користувачів з таблиці *users.*
users	Користувачі	Під час створення користувача, його статус активації дорівнює 0, це означає що користувачу потрібно підтвердити пошту, інакше він не зможе увійти в свій обліковий запис. Всі паролі користувачі хешуються.
users_activation_codes	Ключі активації користувачів	Під час створення нового користувача до нього

		прив‘язується хеш-ключ, за допомогою якого він підтверджує свою пошту. Після підтвердження ключ, прив‘язаний до користувача, видаляється з таблиці.

В проектуванні баз даних є таке поняття, як зв‘язок між таблицями (*relation*). Існує три типи зв‘язків: один до одного (1:1), один до багатьох (1:Б) та багато до багатьох (Б:Б).Принцип зв‘язку 1:1 полягає в тому, що одному запису в одній таблиці може відповідати тільки один запис з іншої. Прикладом такого зв‘язку є таблиця *users_activation_codes*, де до одного нового користувача прив‘язується один ключ підтвердження. Зв‘язок 1:Б відрізняється від 1:1 тим, що одному запису в одній таблиці може відповідати багато записів в іншій, наприклад в таблиці comments розміщуються коментарі, кожен запис в якій, прив‘язаний до користувача в таблиці users, але в одного користувача може бути багато створених коментарів. Б:Б зв‘язок полягає в тому, що багато записів з одної таблиці вказують на багато записів з іншої. Щоб реалізувати такий зв‘язок слід створити додаткову таблицю в базі даних та вносити туди дані, які зв‘язані між собою. Його використовують зазвичай коли проектують інтернет-магазини для того, щоб присвоювати товарам різні категорії.

Опис зв‘язків бази даних створюваного веб-додатку показано і таблиці 3.3.

Таблиця 3.3

Опис зв‘язків між таблицями

Таблиця	Зв‘язок	Таблиця
1	2	3
users	1:Б	*articles*
users	1:Б	*comments*
users	1:1	*users_activation_codes*

articles	1:Б	*comments*

Опис атрибутів таблиць показано в таблиці 3.4.

Таблиця 3.4

Опис атрибутів таблиць

Таблиця	Атрибут	Тип даних	Опис	Значення за замовчуванням
1	2	3	4	5
articles	*id*	*int*	Ідентифікатор статті	*auto_increment*
	author_id	*int*	Ідентифікатор автора	Немає
	name	*varchar(255)*	Назва	Немає
	text	*text*	Текст статті	Немає
	created_at	*Datetime*	Дата створення	*current_timespamp*
comments	*id*	*int*	Ідентифікатор коментаря	*auto_increment*
	user_id	*int*	Ідентифікатор автора	немає
	article_id	*int*	Ідентифікатор статті	немає
	text	*text*	Текст	немає
	date	*datetime*	Дата створення	*current_timespamp*
users	*id*	*int*	Ідентифікатор	*auto_increment*
	nickname	*varchar(128)*	Псевдонім	немає
	email	*varchar(255)*	Електронна пошта	немає
	is_confirmed	*vinyint(1)*	Статус підтвердження	0
	role	*enum (‘admin’, ’user’)*	Права	немає
	password_ hash	*varchar(255)*	Хеш-пароль	немає
	auth_token	*varchar(255)*	Токен авторизації	немає
	created_at	*datetime*	Дата створення	*current_timespamp*
users_ activation_ codes	*id*	*int*	Ідентифікатор	*auto_increment*
	user_id	*int*	Ідентифікатор користувача	немає
	code	*varchar(255)*	Код	немає

Діаграма зв‘язків між таблицями показана на рис. 3.3.

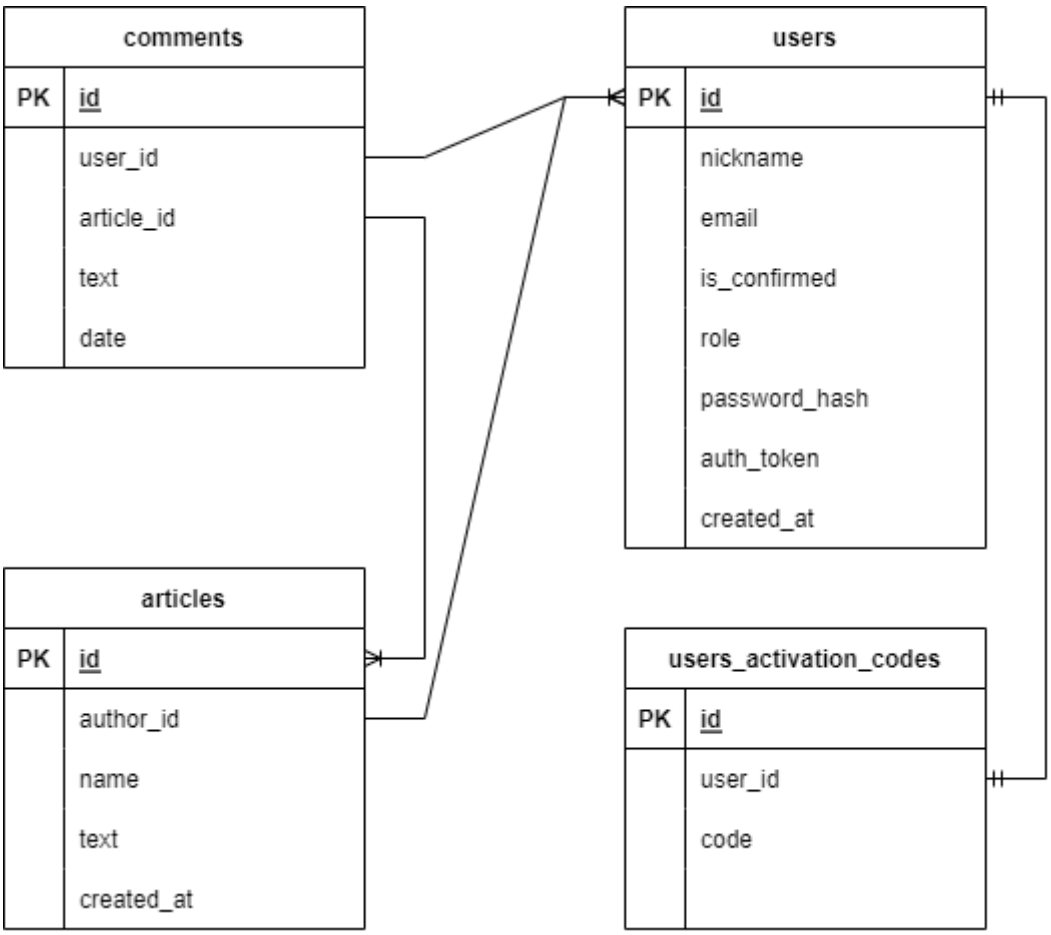

Рис.3.3 Діаграма зв‘язків між таблицями

Для підключення веб-додатку до бази даних використовується об‘єктно орієнтований інструмент мови *PHP* під назвою *PDO*. Реалізація з‘єднання та відправи запитів реалізовано в файлі *Db.php*. Налаштування додатку для підключення до баз даних – у файлі *settings.php*:

```
<?php
return [
    'db' => [
        'host' => 'localhost',
        'dbname' => 'blog',
        'user' => 'root',
        'password' => ''
    ]
];
```

Реалізація таблиць бази даних мовою *SQL*:

```
CREATE TABLE `articles` (
`id` int NOT NULL,
`author_id` int NOT NULL,
`name` varchar(255) NOT NULL,
`text` text NOT NULL,
`created_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
CREATE TABLE `articles` (
`id` int NOT NULL,
`author_id` int NOT NULL,
`name` varchar(255) NOT NULL,
`text` text NOT NULL,
`created_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
CREATE TABLE `users` (
`id` int NOT NULL,
`nickname` varchar(128) NOT NULL,
`email` varchar(255) NOT NULL,
`is_confirmed` tinyint(1) NOT NULL DEFAULT '1',
`role` enum('admin','user') NOT NULL,
`password_hash` varchar(255) NOT NULL,
`auth_token` varchar(255) NOT NULL,
`created_at` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
CREATE TABLE `users_activation_codes` (
`id` int NOT NULL,
`user_id` int NOT NULL,
`code` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
```

3.6. Робота з веб-додатком

Для запуску веб-додатку в пошуковій стрічці запускаємо сценарій *http://blog/*. Після запуску та завантаження, користувач попадає на головну сторінку, яка складається з шапки сайту, меню, яке можна приховати, та основної частини. На головній сторінці тут виводиться список всіх статей (рис. 3.4).

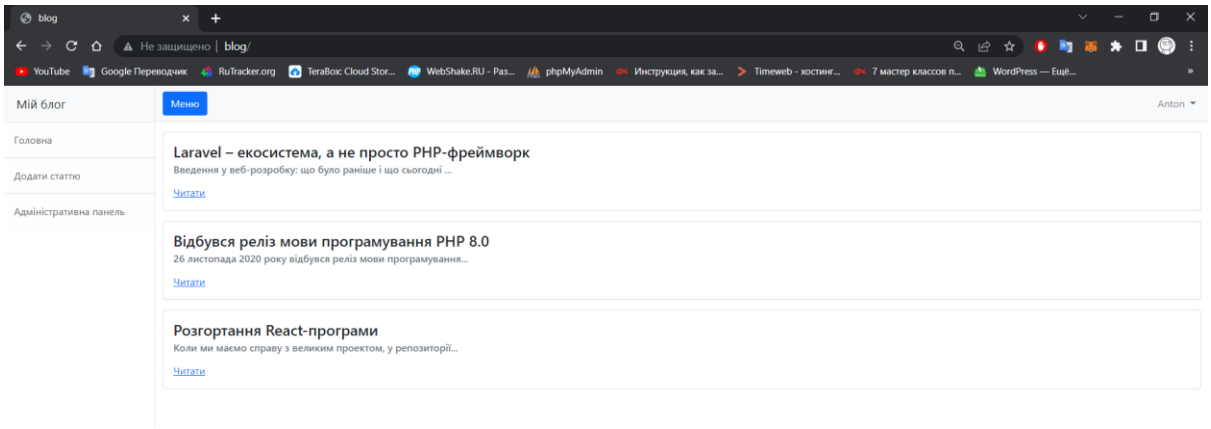

Рис.3.4 Головна сторінка

Тут читач бачить список всіх статей, він має змогу, не входячи свій обліковий запис, читати статті та переглядати коментарі до них (рис.3.5).

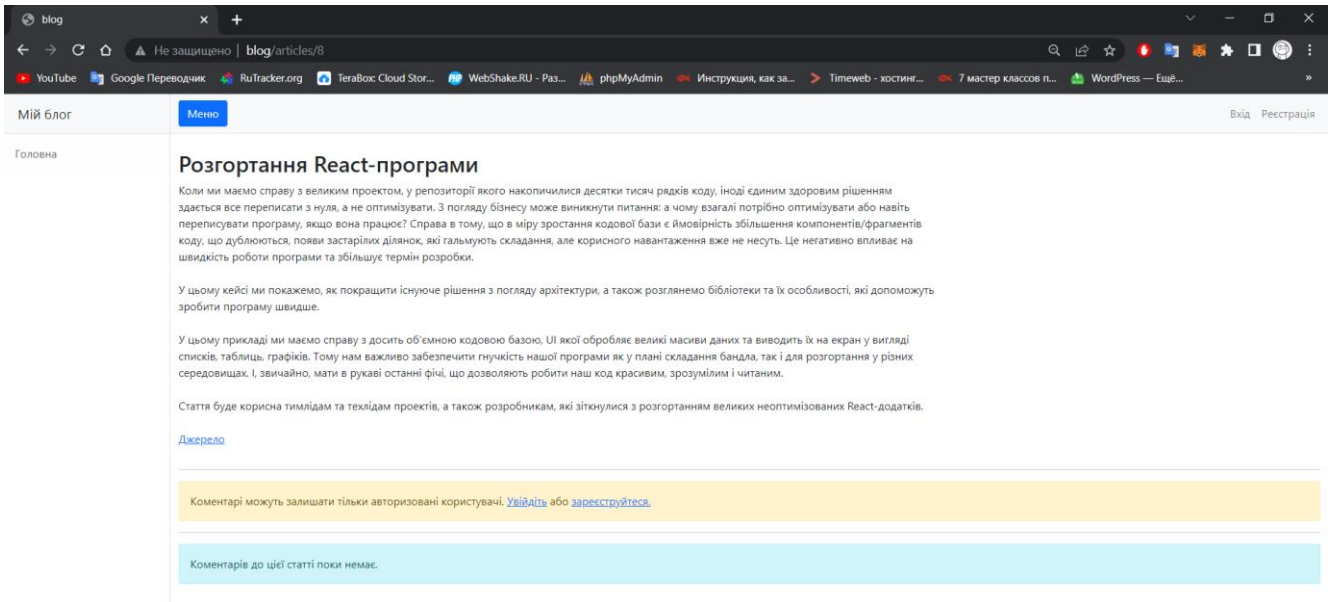

Рис.3.5 Сторінка із статтею

Вкінці кожної статті є розділ з коментарями. Якщо коментарів до цієї статті ще нема, користувачу показується відповідне повідомлення. Також коментарі можуть залишати тільки авторизовані користувачі (рис.3.6).

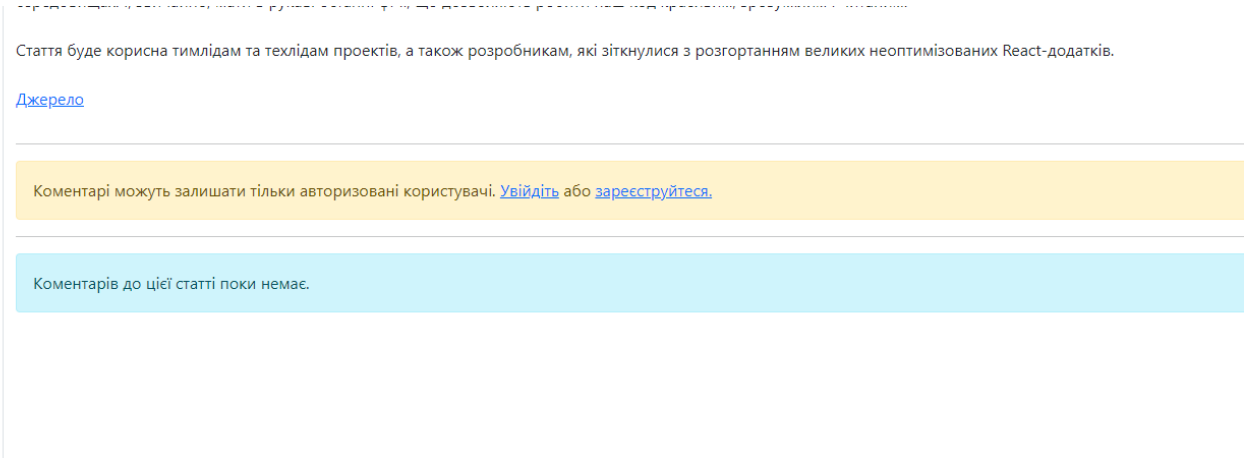

Рис. 3.6 Розділ з коментарями

Якщо користувач ввійшов у свій обліковий запис, в нього з‘явиться форма відправки тексту, де він може залишити свій коментар та додати його до всіх інших, натиснувши відповідну кнопку «Додати» (рис.3.7).

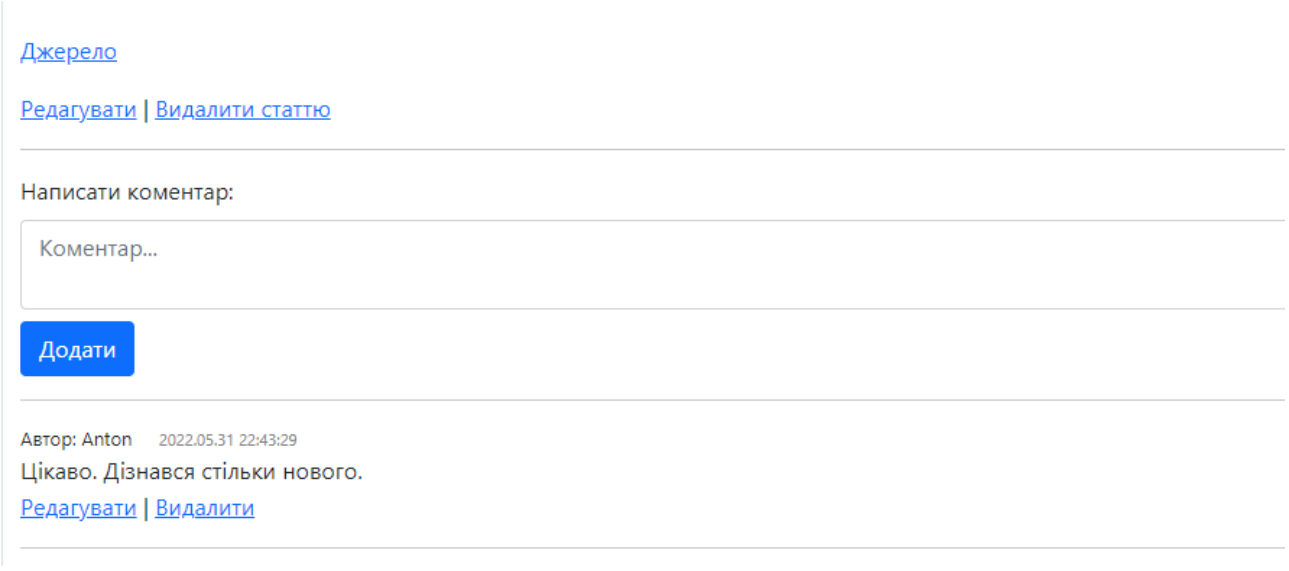

Рис. 3.7 Відкритий розділ з коментарями

Для того, щоб увійти в свій обліковий запис, користувач натискає відповідну кнопку в шапці сайту «Увійти» та потрапляє на сторінку логіну, де він вводить свої дані в поля вводу та відправляє запит на сервер (рис.3.8).

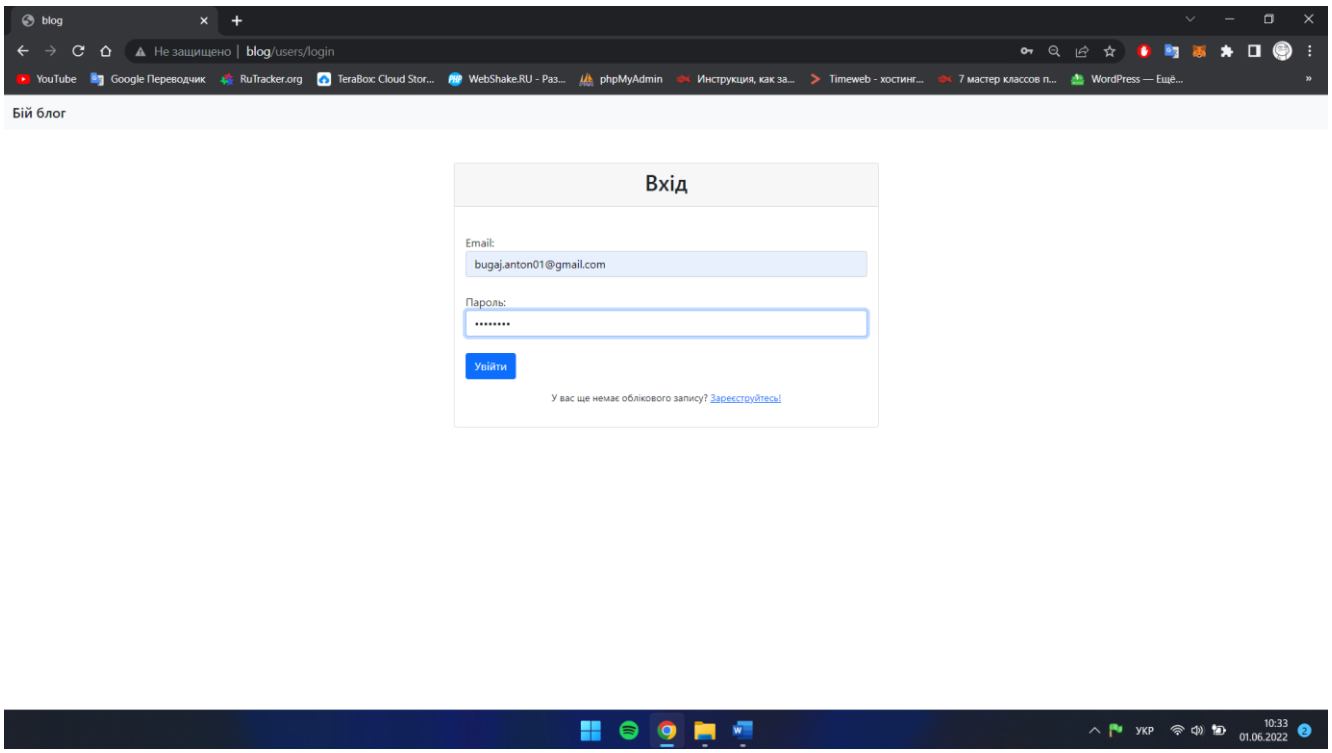

Рис. 3.8 Сторінка входу в обліковий запис користувача

Або якщо в нього ще нема створеного облікового запису, він може натиснути на кнопку «реєстрація» в шапці сайту чи на посилання на сторінці входу. Відкриється сторінка реєстрації (рис.3.9).

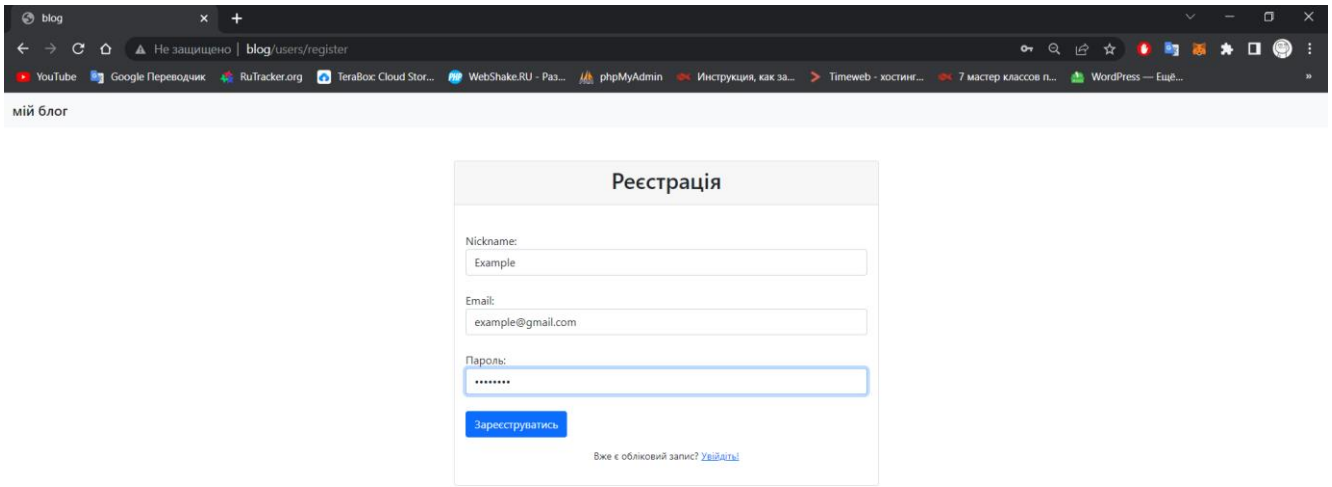

Рис. 3.9 Сторінка реєстрації користувача

Після успішної реєстрації виводиться відповідне повідомлення та користувач має підтвердити свою пошту, просто перейти по посиланню, яке

прийде йому не електронну скриньку інакше він не зможе увійти в свій обліковий запис.

Користувач з правами доступу «юзер» може залишати коментарі до статей та переглядати свій обліковий запис в особистому кабінеті користувача, в якому виводиться вся інформація про нього, та коментарі, які він залишив. (рис.3.10)

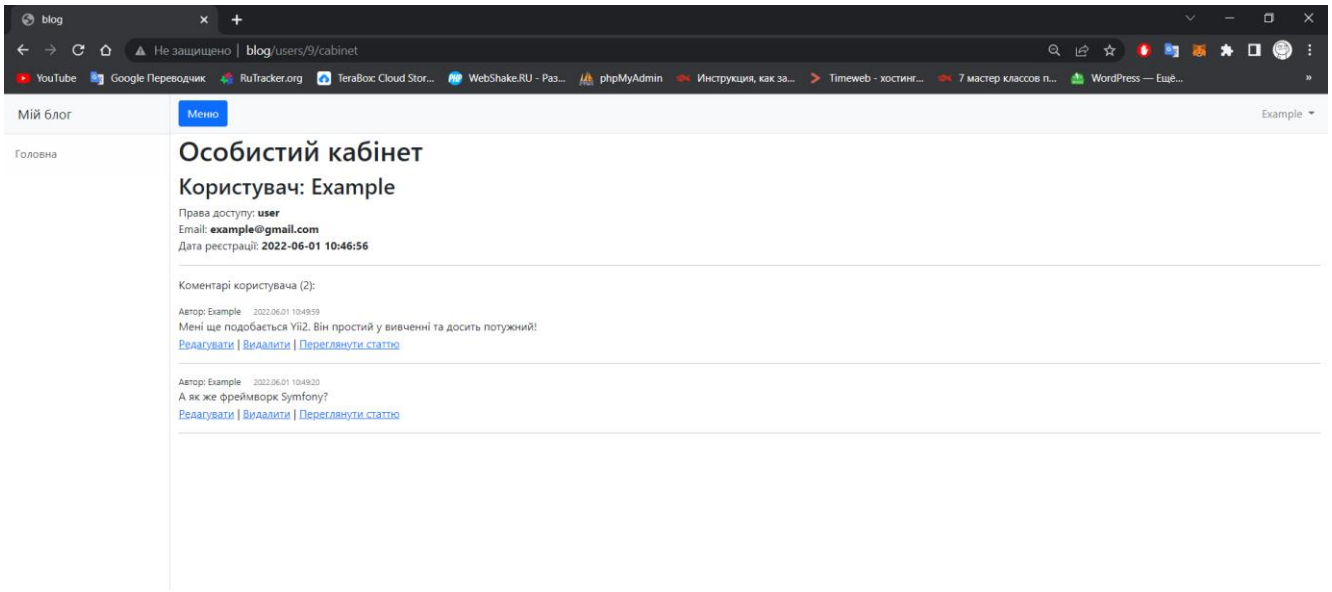

Рис.3.10 Особистий кабінет користувача

Для входу в особистий кабінет треба натиснути на своє ім‘я в правому верхньому кутку сайту, з‘явиться випадаюче меню в якому розміщено дві кнопки: «Особистий кабінет» та «Вихід». Натиснувши на «Вихід», користувач виходить із облікового запису та повертається на головну сторінку.

З точки зору адміністратора можливостей контролю більше. Адміністратор може створювати, видаляти та редагувати статті. Редагувати та видаляти будь-які коментарі, переглядати особисті кабінети користувачів та користуватися адміністративною панеллю.

Додавання нової статті відбувається таким чином: адміністратор відкриває спеціальну сторінку (рис.3.11), вводить назву статті, її текст та відправляє на обробку кнопкою «Створити».

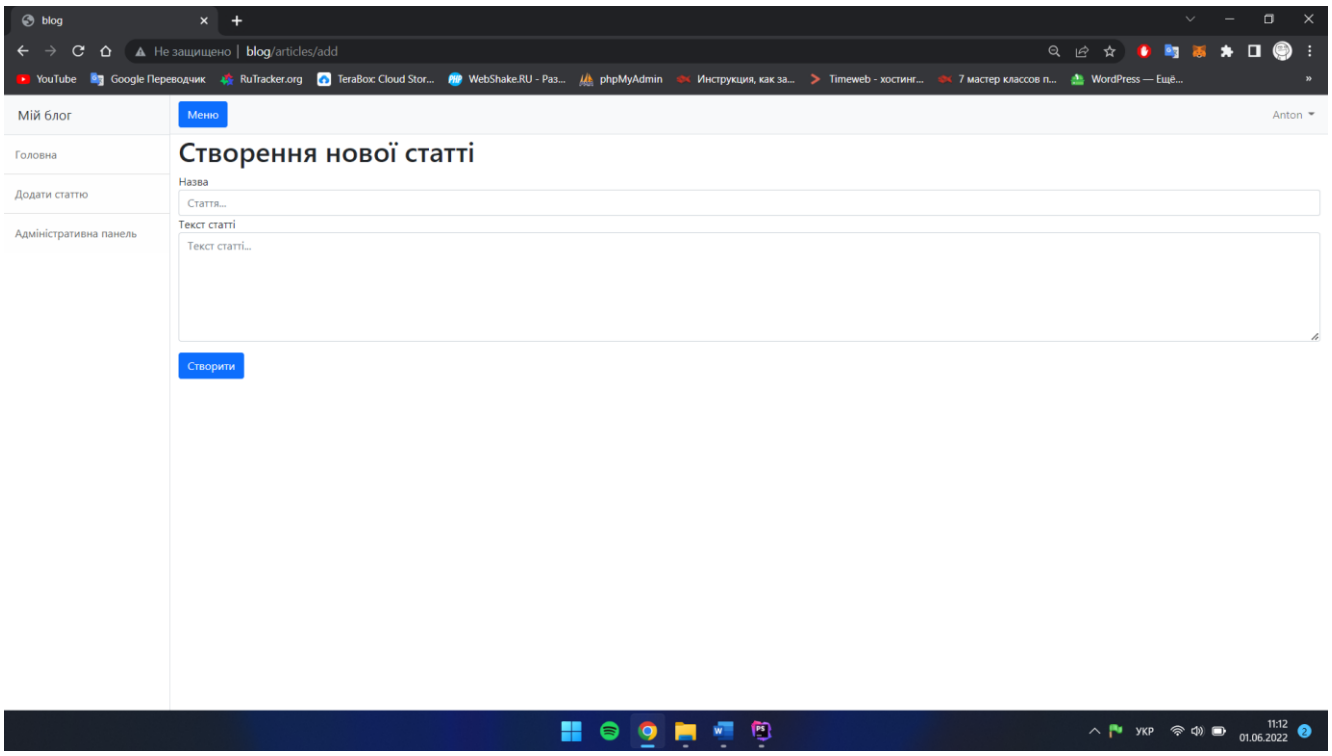

Рис.3.11 Сторінка створення статті

Для редагування статті, адміністратор повинен перейти за відповідним посилання на сторінці потрібної статті. Він може знайти її на головній сторінці, або в адміністративній панелі (рис.3.12).

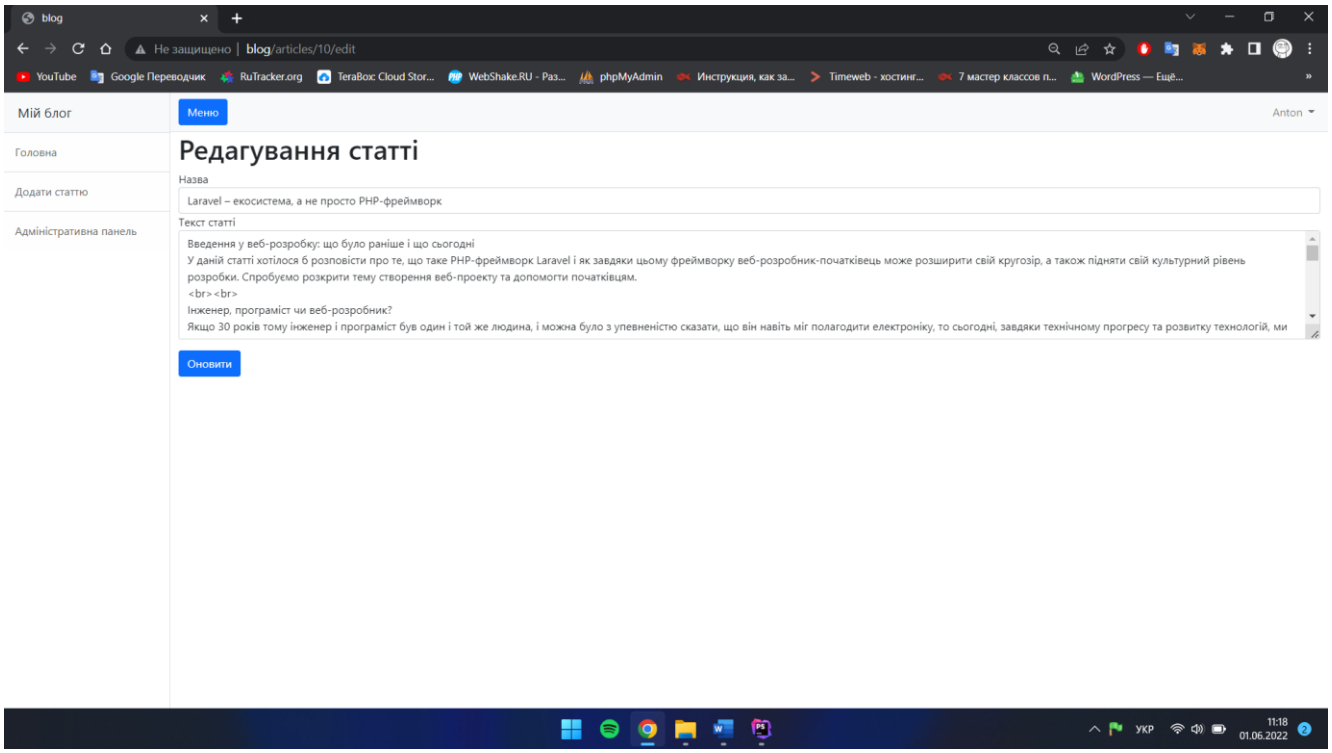

Рис.3.11 Сторінка редагування статті

Таким самим чином працює редагування коментарів. Адміністративна панель відповідає за інформацію про все: коментарі, статті та користувачів. Вся інформація розбита по категоріям (рис.3.12).

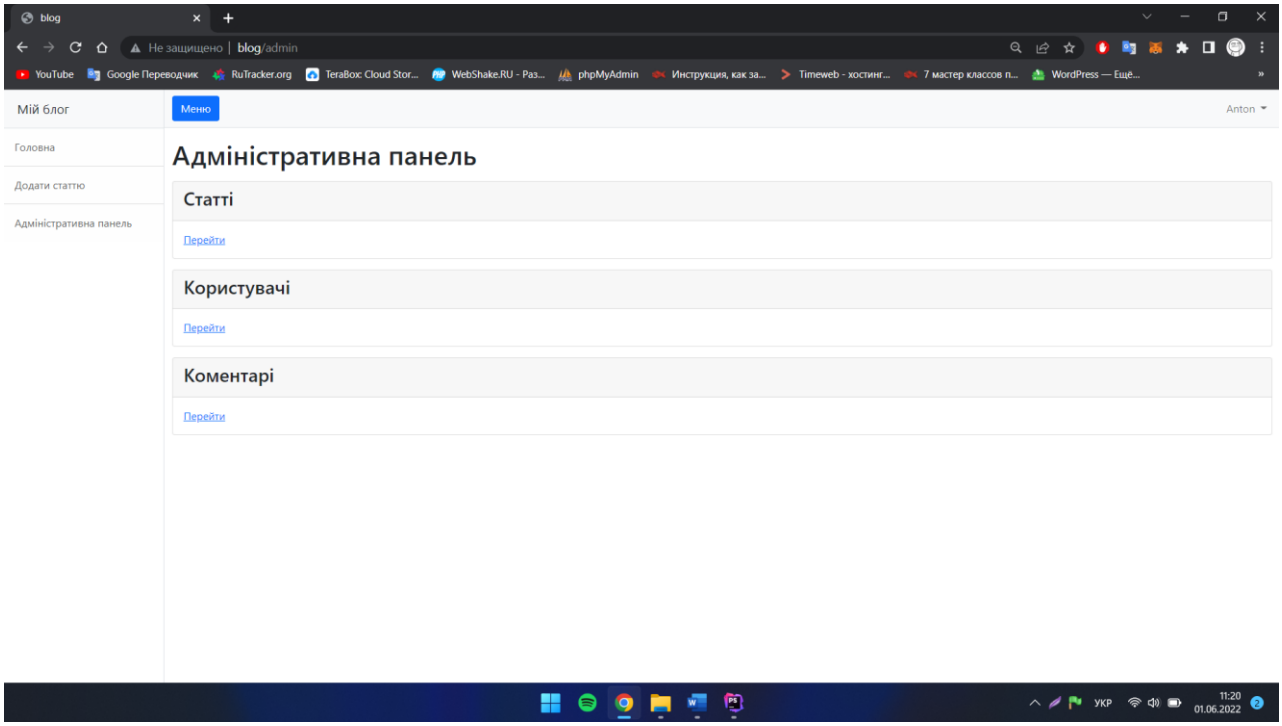

Рис. 3.12 Адміністративна панель додатку

Розділ «статті» адміністративної панелі зображений на рис. 3.13.

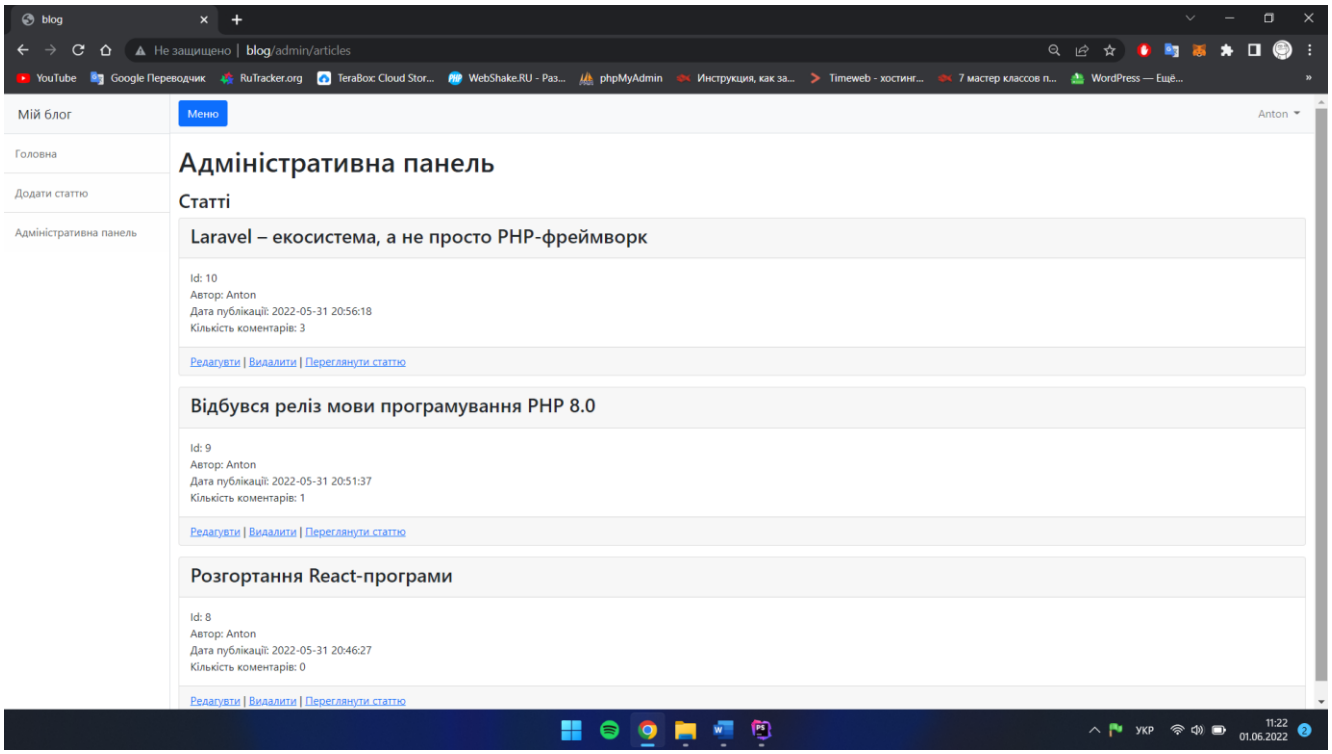

Рис. 3.13 Розділ «статті» адміністративної панелі

Розділ «Користувачі» адміністративної панелі зображено на рис.3.14.

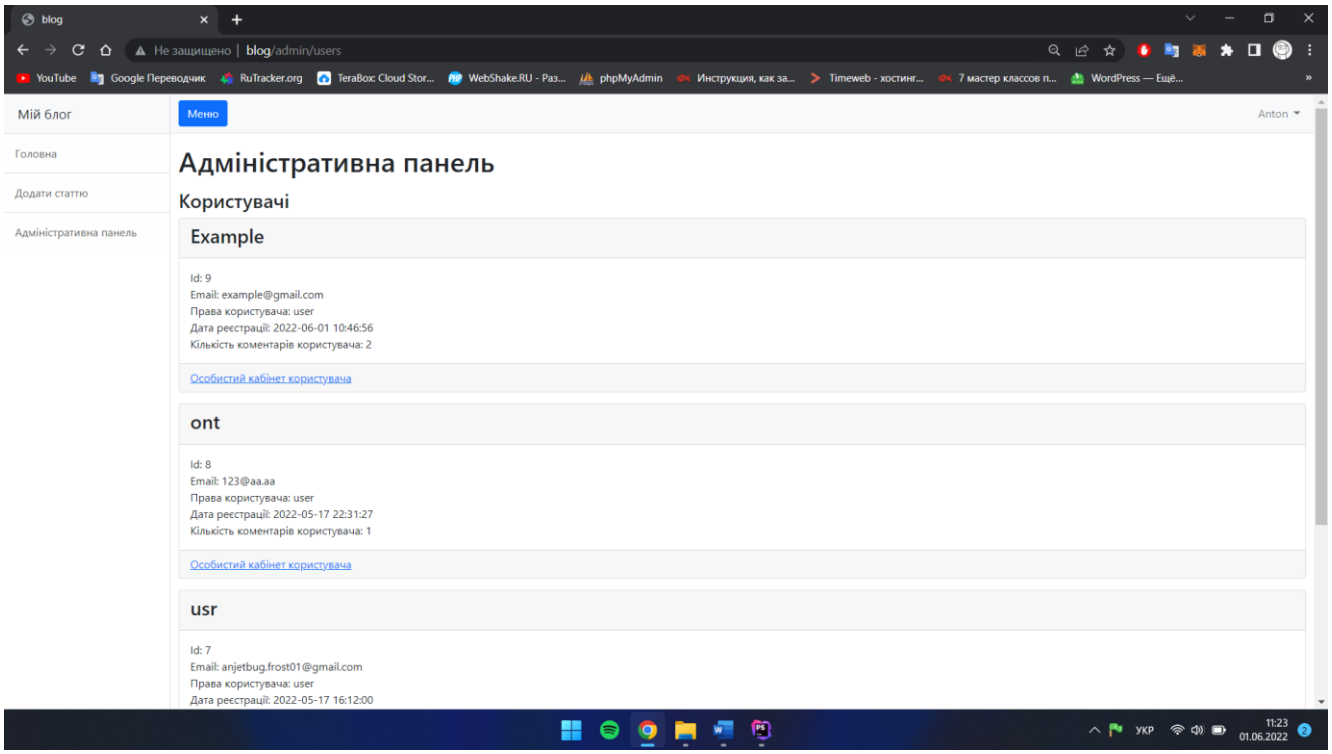

Рис. 3.14 Розділ «Користувачі» адміністративної панелі

Розділ «Коментарі» адміністративної панелі зображено на рис.3.15.

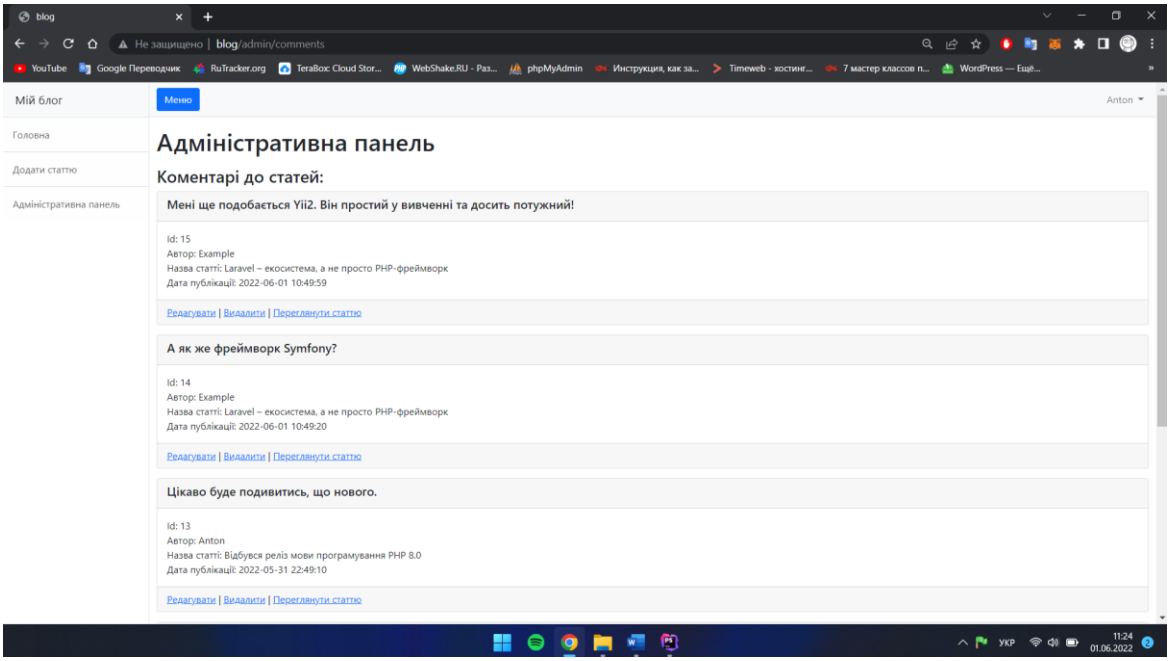

Рис.3.15 Розділ «Коментарі» адміністративної панелі

3.7. Висновки до розділу

В цьому розділі було показано розробку веб-додатку для розміщення та редагування статей. Веб-додаток створений за допомогою таких веб технологій:

- *HTML;*
- *CSS;*
- *PHP;*
- *Open Server;*
- *PhpMyAdmin;*
- *SQL;*
- *MySQL;*

Для створення та адаптації клієнтської частини під різні пристрої була використана бібліотека шаблонів *Bootstrap*, яка в свою чергу використовує *HTML, CSS, JavaScript*. Для відправки запитів були створені спеціальні форми, вміст яких потім оброблявся та заносився в базу даних.

База даних спроектована за допомогою СУБД *MySQL* та мови запитів *SQL*. Для зручного їх адміністрування використано інструмент *phpMyAdmin*. Весь додаток працює на сервері *Apache* версії 2.4, база даних працює на окремому сервері *MySQL*. Всі сервери є модулями до *Open Server* та працюють паралельно.

ВИСНОВКИ

Під час виконання дипломного проекту було розроблено веб-додаток для розміщення та редагування статей. Для створення додатку були проведені наступні дії:

- Аналіз веб-додатків, їх принципів роботи та видів;
- Аналіз веб-технологій для створення додатків;
- Вивчення нових технологій для створення веб-додатків;
- Практика в створенні простих додатків на основі вивчених технологій:

При написанні роботи було визначено, що веб-додаток являє собою програму, яка запускається в браузері, незалежно від операційної системи користувача. Працює така програма на сервері та складається з двох частин – клієнтської та серверної. Клієнтська частина працює напряму з користувачем через браузер, має свій інтерфейс та відправляє запит з даними на серверну частину для їх подальшої обробки. Серверна частина – це частина програми, яка розміщується та працює на сервері, має змогу працювати з базами даних. Основною задачею серверної частини є обробка даних та відправка їх до клієнтської частини для подальшого відображення користувачеві.

Для написання клієнтської частини використовуються такі технології:

- *HTML;*
- *CSS;*
- *JavaScript;*

Також різні бібліотеки шаблонів, наприклад *Bootstrap*.

Серверна частина може використовувати багато технології, таких як: *JAVA, PHP, .NET, NodeJS*. Веб-додаток для розміщення та редагування статей написаний на скриптовій мові програмування *PHP* та для роботи з базами даних використовує СУБД *MySQL*.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тузовський А.Ф. Проектування інтернет додатків. – М.: Вид-во ТПУ, 2010. – 200 с.
2. ДСТУ ГОСТ 3008-95 «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.»
3. vc.ru [Електронний ресурс]. – Режим доступу: https://vc.ru/services/297762-desktopnoe-ili-veb-prilozhenie-plyusy-i-minusy
4. uk.go-travels.com [Електронний ресурс]. – Режим доступу: https://uk.go-travels.com/73084-what-is-a-web-application-3486637-6454152
5. webcase.com.ua [Електронний ресурс]. – Режим доступу: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/#f2
6. dic.academic.ru [Електронний ресурс]. – Режим доступу: https://dic.academic.ru/dic.nsf/ruwiki/1705625
7. habr.com [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/137388/
8. stud.com.ua [Електронний ресурс]. – Режим доступу: https://stud.com.ua/97678/informatika/proektuvannya_dodatkiv
9. www.znannya.org [Електронний ресурс]. – Режим доступу: http://www.znannya.org/?view=WebDev
10. dzudzylo.com [Електронний ресурс]. – Режим доступу: https://dzudzylo.com/javascript/shho-take-ajax-ta-yak-vona-pratsyuye.html
11. deltahost.ua [Електронний ресурс]. – Режим доступу: https://deltahost.ua/tipy-setevyx-protokolov-i-ix-naznachenie-http-ip-ssh-ftp-pop3-mac.html
12. www.ukraine.com.ua [Електронний ресурс]. – Режим доступу: https://www.ukraine.com.ua/blog/seo-optimization/chto-takoe-https-i-zachem-on-nuzhen-kazhdomu-sajtu.html
13. www.azoft.ru [Електронний ресурс]. – Режим доступу:

https://www.azoft.ru/blog/web-development-stack/