МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії

"До захисту допущена"

Зав. кафедри комп'ютерних наук та прикладної математики

_____

“____” _____20___року

КВАЛІФІКАЦІЙНА РОБОТА

«Проєктування та розробка клієнтської та серверної частин веб-додатку “Блог” з використанням React, Redux Toolkit та NestJS»

Виконала: **Карпова Алевтина Максимівна** _____

група ІПЗ-41 (підпис)

Керівник: **Зубик Ярослав Ярославович** _____

(підпис)

Рівне 2024

Національний університет водного господарства та природокористування

ННІ інститут **автоматики, кібернетики та обчислювальної техніки**

Кафедра **комп'ютерних наук та прикладної математики**

Освітньо-кваліфікаційний рівень **бакалавр**

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____

_____

"____" _____20___року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

_____Карповій Алевтині_Максимівній_____

1. Тема роботи: "Проєктування та розробка клієнтської та серверної частин веб-додатку "Блог" з використанням React, Redux Toolkit та NestJS"

керівник роботи: Зубик Ярослав Ярославович

затверджені наказом вищого навчального закладу від «___» _____20__ року №_____

2. Строки подання студентом роботи "17" червня 2024 р.

3. Вихідні дані до роботи: Розроблений веб-додаток з використанням React, Redux Toolkit та NestJS

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно опрацювати)

1. Огляд та аналіз існуючих аналогів веб-додатків типу "Блог".
2. Огляд серверної та клієнтської частини веб-додатку.
3. Розробка веб-додатку з використанням React, Redux Toolkit NestJS.

5. Консультанти по роботі із зазначенням розділів, що їх стосуються.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Розділ 1	Ст.викл. Зубик Я. Я.		
Розділ 2	Ст.викл. Зубик Я. Я.		
Розділ 3	Ст.викл. Зубик Я. Я.		

6. Дата видачі завдання 1.11.23 р.____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітки
1.	Аналіз предметної області	02.11.2023р. - 30.01.2024р.	Виконано
2.	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	02.02.2024р. - 08.03.2024р.	Виконано
3.	Аналіз сучасних технологій для розробки серверної частини	08.03.2024р.	Виконано
4.	Аналіз сучасних технологій для розробки користувацької частини	02.03.2024р. - 08.04.2024р.	Виконано
5.	Програмна реалізація серверної частини застосунку	08.04.2024р. - 14.05.2024р.	Виконано
6.	Представлення отриманих результатів науковому керівнику та узгодження плану подальшого дослідження	14.03.2024р. - 04.05.2024р.	Виконано
7.	Реалізація користувацького інтерфейсу	04.05.2024р. - 26.05.2024р.	Виконано
8.	Оформлення звіту	12.05.2024р. - 30.05.2024р.	Виконано

Студент _____ Карпова А.М.
(підпис)

Керівник роботи _____ Зубик Я.Я.
(підпис)

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота: 59 с., 28 малюнків, 10 джерел.

Метою кваліфікаційної роботи є дослідження сучасних технологій розробки веб-додатків виду блог, аналіз переваг та недоліків впроваджених в кваліфікаційній роботі технологій.

Об'єктом дослідження сучасні веб-додатки "Блог", їх застосування та практичне значення.

Предметом дослідження сучасні технології React, Redux Toolkit та NestJS.

Методи дослідження – аналіз літератури та сайтів, пошуковий та порівняльний.

Ключові слова: HTML**,** CSS, ORM, JWT, JS, СУБД, BJSON, DOM, JSX.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HTML - Мова розмітки гіпертексту.

CSS – Cascade Style Sheets.

JS – JavaScript.

ORM - Об'єктно-реляційне відображення.

JWT - JSON Web Token.

СУБД – Система управління базами даних.

BJSON - Binary JSON.

DOM - Об'єктна модель документа.

JSX - JavaScript XML.

ВСТУП

У 20-му столітті винахід інтернету відкрив нові можливості для комунікації, що стало важливим фактором у сучасному цифровому світі. Веб-блоги, безсумнівно, відіграють ключову роль у цьому процесі, ставши невід'ємною складовою онлайн-діяльності. Завдяки широкому доступу до Інтернету, блоги стали важливим джерелом інформації для багатьох людей.

Сучасні блоги є одними з найпопулярніших веб-застосунків, які надають платформу для публікації різноманітних матеріалів - від освітнього до розважального контенту. Вони охоплюють різноманітні теми, починаючи від особистих думок та досвіду до професійних статей. Блоги стали не тільки засобом вираження індивідуальності, але й засобом обміну корисної інформації.

У сучасному інформаційному просторі важко уявити життя без блогів, які стали важливим елементом культури в Інтернеті. Вони створюють можливість не лише ділитися думками та досвідом, але й отримувати корисну інформацію від інших користувачів.

Крім того, блоги є ефективним інструментом для підтримки взаємодії з аудиторією, створення лояльності та підвищення свідомості про бренд. Вони дозволяють підтримувати діалог з читачами через коментарі, взаємодію в соціальних мережах та інші форми зв'язку. Загалом, блоги не лише розширюють доступ до інформації та сприяють культурній різноманітності, але й сприяють розвитку спільнот та підтримці взаємодії між користувачами.

Постановка завдання

Метою кваліфікаційної роботи є розробка веб-додатку для ведення авторського блогу з акцентом на інтуїтивно зрозумілий інтерфейс використовуючи React, Redux Toolkit та NestJS.

Реалізація поставленої мети представляє собою необхідність вирішення наступних завдань:

1. Провести аналіз предметної області та визначити вимоги до веб-додатку.
2. Дослідити технології для розробки клієнтської та серверної частин веб-додатку.
3. Розробити архітектуру клієнтської та серверної частин додатку, визначити основні компоненти та їх взаємодію для забезпечення ефективної роботи додатку.
4. Розробка клієнтської та серверної частин веб-додатку з використанням обраних технологій.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття та значення блогу

Блог є відносно новим різновидом веб-ресурсу інформаційного представлення, що характеризується регулярною публікацією записів, які зазвичай відображаються в зворотному хронологічному порядку. Тобто, найновіші записи знаходяться на початку сторінки. Блоги можуть охоплювати широкий спектр тем, включаючи особисті щоденники, професійні статті, новини, аналітичні огляди, технічні інструкції та багато інших.

З розвитком інформаційних технологій та інтернету, блоги стали важливим інструментом для особистої та професійної комунікації. Вони дозволяють авторам висловлювати свої думки, а також взаємодіяти з аудиторією через коментарі та обговорення. Це створює унікальну можливість для встановлення прямого діалогу між автором та читачами, що сприяє формуванню спільнот за інтересами та зміцненню взаємодії між користувачами.

Блоги також мають велике значення для поширення інформації та знань. Вони слугують платформами для обговорення актуальних тем, обміну досвідом та експертними думками. Завдяки своїй доступності та простоті використання, блоги досить швидко здобули популярність і продовжують залишатися затребуваними й до сьогодні.

1.2 Класифікація блогів

Класифікація блогів може бути досить різноманітною і залежати від різних критеріїв, таких як: зміст, цільова аудиторія, мета, формат та стиль. За змістом розрізняють такі типи блогів як: Особисті, що використовуються для ведення щоденників, подорожей та обміну особистими інтересами; Тематичні блоги фокусуються на конкретних темах, таких як технології, мода, здоров'я та фінанси, забезпечуючи спеціалізовану інформацію для зацікавлених читачів.

Корпоративні блоги допомагають компаніям ділитися новинами, анонсами продуктів та історіями успіху, зміцнюючи відносини з клієнтами; Новинні блоги дозволяють оперативно поширювати новини, аналітичні статті та інтерв'ю, допомагаючи медіа-компаніям висвітлювати актуальні події; Освітні блоги використовуються для створення навчальних матеріалів, посібників та відеоуроків, надаючи платформу для вчителів і тренерів для обміну знаннями з учнями.

За критерієм цільової аудиторії визначають блоги, що орієнтовані на широкий загал, спеціалістів у певних галузях та корпоративні блоги. Типи блогів за метою можуть бути інформаційні, розважальні та маркетингові. Критерій формату визначає текстові блоги, фотоблоги, відеоблоги та подкасти. За стилем блоги поділяються на формальні та неформальні блоги. Також специфічні блоги, такі як: мікроблоги, колективні блоги та гостьові блоги.

Серед найбільш актуальних типів виділяються тематичні блоги та особисті, з яких, власне, почалася історія блогінгу. Сучасні особисті блоги часто мають формат «лайфстайл» і містять розповіді з власного життя авторів.

1.3 Аналіз програмних продуктів-аналогів сервісів для блогінгу

Аналіз програмних продуктів-аналогів є важливою складовою процесу обґрунтування вибору платформи для ведення власних блогів. На сьогоднішній день існує широкий асортимент програмних рішень, які надають можливості для створення та управління веб-блогами. Серед популярних програмних продуктів, що сприяють розвитку цієї сфери, були обрані такі як WordPress, Blogger та Tumblr. Кожна з цих платформ має свої унікальні особливості, переваги та недоліки, які необхідно врахувати при виборі оптимального рішення для створення власного блогу.

Ghost – це платформа для ведення блогів із відкритим вихідним кодом призначена для спрощення процесу публікації в Інтернеті для окремих блогерів та для онлайн-видань.

До переваг можна віднести:

1. Мінімалістичний інтерфейс, що спрощує процес написання та публікації контенту.
2. Вбудовані інструменти для підписок і оплати контенту, що дозволяє авторам отримувати дохід від свого блогу.
3. Відкрите API та підтримка розширень дозволяють розробникам налаштовувати функціонал платформи під свої потреби.

До недоліків можна віднести:

1. Вимагає певного рівня технічних знань для самостійного хостингу і налаштування.

Blogger – це безкоштовна платформа для ведення блогів, створена компанією Google. Вона дозволяє користувачам легко створювати, редагувати та публікувати блоги без необхідності технічних знань. Blogger пропонує інтуїтивно зрозумілий інтерфейс, інтеграцію з іншими сервісами Google, такими як Google Analytics і AdSense, а також безкоштовний хостинг на серверах Google.

Платформа включає в себе наступні переваги:

1. Простота використання, що робить платформу більш доступною.
2. Безкоштовний хостинг, що усуває потребу у виборі хостинг-провайдера.
3. Тісна інтеграція з сервісами Google, що забезпечує зручне управління та моніторинг.

Серед недоліків цієї платформи можна виділити:

1. Жорстка цензура контенту, що може обмежувати свободу вираження.

2. Залежність від Google, що може створювати ризики у разі змін політики компанії.
3. Обмежений контроль над власним блогом, оскільки користувачі не мають повного доступу до усіх налаштувань та можливостей платформи.

Tumblr – це соціальна медіа-платформа та мікроблогінговий сервіс, який дозволяє користувачам публікувати короткі текстові записи, зображення, відео, посилання та аудіофайли. Заснована в 2007 році та набула популярності через свій простий у використанні інтерфейс, можливості спілкування з іншими користувачами через коментарі, репости та підписки, а також за своєю орієнтацією на молоду аудиторію та різноманітністю культурного контенту.

Основні переваги містять:

1. Соціальні функції, що дозволяють користувачам легко взаємодіяти з іншими та розповсюджувати контент.
2. Простоту публікації, яка сприяє швидкому та легкому розміщенню різних типів вмісту.
3. Інтуїтивно зрозумілий інтерфейс, який спрощує навігацію та використання платформи.

Основні недоліки містять:

1. Обмежену можливість розміщення великих текстових матеріалів, оскільки платформа спрямована на короткі, миттєві публікації.
2. Залежність від платформи, що може обмежувати контроль користувачів над їхнім вмістом та даними.

2. ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ

2.1 Аналіз сучасних технологій серверної частини веб-додатку

Сучасна розробка веб-додатків вимагає ретельного вибору технологій для забезпечення високої продуктивності, надійності, масштабованості та безпеки системи. Серверна частина веб-додатку відіграє ключову роль у обробці даних, взаємодії з базами даних, реалізації бізнес-логіки та забезпеченні користувацького досвіду. Обґрунтований вибір найбільш відповідних технологій, дозволять ефективно реалізувати цілі проєкту та забезпечити його успішну роботу у довгостроковій перспективі. На сьогодні доступна велика кількість технологій, які пропонують різні підходи до розробки серверних додатків, враховуючи потреби конкретного проєкту та вимоги до його функціональності і безпеки.

2.1.1 Аналіз мов програмування

JavaScript – динамічна, об'єктно-орієнтована прототипна мова, яка використовується для розробки як на серверній частині веб-додатку так і на клієнтській. Найчастіше для серверної розробки JavaScript використовується в поєднанні з фреймворками Express, Next.js та Meteor.js.

Загальні характеристики JavaScript включають:

1. Динамічна Типізація: відсутність потреби в явному визначенні типу змінної перед її використанням дозволяє розробникам працювати більш гнучко та швидко адаптувати код до змінних умов.
2. Асинхронна обробка: можливість виконувати операції в фоновому режимі, такі як завантаження даних або робота з файлами, забезпечує швидкість роботи програм та підвищує ефективність веб-сторінок.

3. Незалежність: JavaScript є платформо-незалежною мовою програмування, що дозволяє запускати один і той же код на різних браузерах і операційних системах.
4. Продуктивність: можливість розподілу завдань між сервером і клієнтом дозволяє оптимізувати роботу обох компонентів системи і підвищує їх ефективність.
5. Використання Інтерпретатора: інтерпретація JavaScript дозволяє розробникам швидко внести зміни в код під час розробки і вирішувати проблеми без перекомпіляції.

Переваги:

- Широка популярність серед розробників.
- Велика спільнота та розширена екосистема з багатьма бібліотеками та фреймворками.
- Розширена функціональність.
- Асинхронність.

Недоліки:

- Слабка типізація.
- Можлива несумісність зі старими версіями браузерів.

Java - об'єктно-орієнтована мова програмування, призначена для створення надійних, безпечних та масштабованих додатків. Широко використовується у веб-розробці, розробці мобільних додатків та корпоративних системах.

Основні характеристики Java:

1. Об'єктно-орієнтоване програмування (ООП): Java підтримує принципи ООП, такі як наслідування, поліморфізм, інкапсуляція та абстракція, що сприяє модульності та повторному використанню коду.
2. Платформонезалежність: однією з ключових особливостей Java є принцип "Write Once, Run Anywhere". Код написаний на Java, компілюється в байт-код, який може виконуватись на будь-якій платформі, що підтримує JVM.
3. Безпека: Java має вбудовані механізми безпеки, що дозволяють створювати надійні і захищені програми, що ключає в себе управління пам'яттю, перевірку типів та забезпечення безпеки виконання.
4. Розподілена система: підтримка розподілених обчислень, що дозволяють створювати розподілені додатки. Включає в себе підтримку мережевих протоколів та технологій, таких як RMI (Remote Method Invocation) та CORBA (Common Object Request Broker Architecture).
5. Мультипоточність: надає вбудовану підтримку багатопочності, що дозволяє створювати програми, які можуть виконувати кілька завдань одночасно.

   Переваги:

   - Висока продуктивність.
   - Високий рівень безпеки.
   - Масштабованість.
   - Широка екосистема.
   - Мультипоточність.

   Недоліки:

   - Високі вимоги до пам'яті і ресурсів.

- Обмежений контроль над збором сміття.

Python – високорівнева, інтерпретована та об'єктно-орієнтована мова програмування із суворою динамічною типізацією. Активно використовується в різних сферах програмування завдяки своїй гнучкості та потужним можливостям.

Загальні характеристики:

1. Простота і читабельність: зрозумілий і лаконічний синтаксис робить код легким для читання і написання, що знижує бар'єр для входу новачків у програмування.
2. Високий рівень абстракції: надає можливість зосередитися на вирішенні проблеми, а не на деталях реалізації.
3. Підтримка різних парадигм програмування: підтримка об'єктно-орієнтованого, процедурного та функціонального програмування, що робить його універсальним інструментом для різних задач.

Переваги:

- Велика стандартна бібліотека.
- Широкий спектр застосувань.
- Активна і велика спільнота.

Недоліки:

- Низька продуктивність.
- Обмежена підтримка мобільних платформ.
- Динамічна типізація.

PHP (Hypertext Preprocessor) — це скриптова мова програмування, була спеціально розроблена для створення динамічних веб-сайтів.

Загальні характеристики:

1. Гнучкість: інтеграція з HTML дозволяє вбудовувати PHP-код безпосередньо в сторінки веб-сайтів. Аналогічно має сумісність з різними базами даних, пристроями та серверами.
2. Простота: легкий і зрозумілий синтаксис.
3. Масштабованість: підтримка розробки великих і складних веб-проєктів завдяки великій кількості фреймворків і бібліотек.
4. Швидкість: споживання малої кількості пам'яті під час виконання коду дозволяє скоротити час завантаження, внаслідок чого покращує продуктивність програми.

Переваги:

- Широкий вибір фреймворків і бібліотек.
- Підтримка великої кількості веб-хостинг-провайдерів.
- Широке використання і підтримка.
- Ефективність для створення динамічних веб-сайтів.

Недоліки:

- Проблеми з безпекою.
- Повільна швидкість виконання.

2.1.2 Аналіз фреймворків для розробки серверної частини веб-додатку

Spring Framework - фреймворк для розробки програмного забезпечення на мові Java, що надає комплексний набір інструментів та бібліотек для створення різноманітних застосунків, включаючи веб-додатки, мікросервіси, корпоративні застосунки та інші програмні рішення.

Основні характеристики:

1. Інверсія керування (IoC): модуль Spring, який дозволяє створювати слабко пов'язані системи. Замість того, щоб об'єкти самостійно створювали свої залежності, вони отримують їх від контейнера IoC, що сприяє зниженню зв'язності компонентів та полегшує тестування і підтримку коду. IoC реалізується за допомогою конфігураційних файлів XML або анотацій.
2. Аспектно-орієнтоване програмування (Aspect-Oriented Programming, AOP): відокремлення різних аспектів функціональності програми, такі як логування, безпека, управління транзакціями, без зміни бізнес-логіки, що сприяє кращій модульності та дозволяє уникнути дублювання коду.
3. Підтримка транзакцій: забезпечення потужних засобів для управління транзакціями як програмно, так і декларативно. Програмне управління транзакціями дозволяє явно вказувати межі транзакцій у коді, тоді як декларативне управління використовує анотації або конфігурації XML для визначення транзакційних політик, що спрощує роботу з транзакціями, забезпечуючи надійність операцій.
4. Широкі можливості інтеграції: Spring також підтримує інтеграцію з різноманітними технологіями та іншими фреймворками, такими як Hibernate для роботи з базами даних, Spring MVC для реалізації веб-додатків, та іншими.

Django - фреймворк на мові програмування Python, призначений для швидкої розробки веб-додатків, який дозволяє полегшити процес розробки складних, динамічних веб-додатків і зменшити кількість необхідного коду. Окрім того, забезпечує багато вбудованих компонентів та функцій, що значно спрощують процес створення веб-додатків.

Основні характеристики:

1. Архітектура: використання архітектурного шаблону, аналогічного до MVC, відомого як "Model-View-Template" (MVT). Цей підхід забезпечує чітке розділення логіки даних, бізнес-логіки та представлення, що сприяє покращенню структури коду та його підтримуваності.
2. Безпека: Django забезпечує багато вбудованих механізмів безпеки, таких як захист від SQL-ін'єкцій, міжсайтових скриптових атак (XSS), міжсайтових запитів підробки (CSRF) та інших поширених вразливостей, а також підтримує безпечну автентифікацію та авторизацію користувачів.
3. Масштабованість та гнучкість: Django є ефективним інструментом для розробки як невеликих проєктів, так і масштабних веб-додатків з великою кількістю користувачів, завдяки своїй високій масштабованості та гнучкості. Він забезпечує підтримку різноманітних баз даних та легко інтегрується з іншими технологіями, що робить його універсальним рішенням для різних типів проєктів.
4. Автоматичне адміністрування: Django має вбудовану адміністративну панель, яка автоматично генерується на основі моделей даних, що дозволяє розробникам швидко створювати інтерфейси для управління контентом без додаткового кодування.

Laravel – популярний веб-фреймворк на мові програмування PHP, створений з метою спрощення розробки веб-додатків шляхом надання багатого набору інструментів. Laravel дотримується принципу "convention over configuration" (конвенція замість конфігурації) і забезпечує розробникам зручне середовище для швидкої розробки, тестування та розгортання додатків.

Фреймворк підтримує такі концепції, як модульність, інверсія управління, впровадження залежностей та багато інших.

Основні характеристики:

1. Архітектура MVC (Model-View-Controller): Laravel дотримується архітектурного шаблону MVC, що забезпечує чітке розділення логіки додатку, представлення та управління даними. Це сприяє підвищенню організованості коду та його легкій підтримці.
2. Впровадження залежностей (Dependency Injection): Laravel використовує принцип впровадження залежностей для управління об'єктами та їх залежностями. Це дозволяє зменшити зв'язаність компонентів та полегшує тестування і модифікацію коду.
3. Міграції баз даних та ORM Eloquent: Laravel надає зручний інструмент для управління міграціями баз даних, що дозволяє легко створювати, модифікувати та синхронізувати схеми баз даних. ORM Eloquent забезпечує простий та інтуїтивний спосіб роботи з базами даних через об'єктно-реляційне відображення.
4. Підтримка RESTful API: Laravel підтримує створення RESTful API, що дозволяє легко будувати сервіси для взаємодії з клієнтськими додатками та іншими сервісами.
5. Кешування та управління сесіями: Laravel надає широкий набір інструментів для кешування даних та управління сесіями користувачів, що дозволяє покращити продуктивність та масштабованість додатків.

NestJS є сучасним фреймворком для створення масштабованих та ефективних серверних додатків на платформі Node.js. Він базується на принципах об'єктно-орієнтованого та функціонального програмування та надає готову архітектуру додатків.

Основні характеристики:

1. Архітектурний шаблон і модульність: NestJS надає підтримку архітектурного шаблону MVC (Model-View-Controller), що дозволяє чітко розділити логіку додатку. Також підтримує інші архітектурні шаблони, такі як модульне програмування, що полегшує розширення та підтримку коду.
2. Модульність та розширюваність: NestJS дозволяє створювати додатки з модульною структурою, що полегшує впровадження змін і додавання нових функціональностей без значних змін у вже існуючому коді.
3. Висока продуктивність: NestJS надає підтримку асинхронності, що дозволяє ефективно обробляти велику кількість запитів та подій. Це особливо важливо для побудови масштабованих та високопродуктивних додатків.
4. Підтримка TypeScript: TypeScript є основною мовою розробки в NestJS, що дозволяє використовувати переваги сильної типізації, інтерфейсів та інших сучасних можливостей мови JavaScript.
5. Тестування: Фреймворк забезпечує вбудовану підтримку для автоматизованого тестування, включаючи юніт-тести та інтеграційні тести.

У рамках проведеного аналізу були розглянуті різні мови програмування та фреймворки. З урахуванням специфічних вимог типу проєкту, кожен з цих інструментів має свої переваги та особливості. На основі детального огляду кожної з технологій, зважаючи на наші потреби в продуктивності, швидкодії, масштабованості та зручності у розробці, оптимальним вибором був визначений фреймворк NestJS.

2.2 Аналіз сучасних СУБД

СУБД як комплекс програмних засобів, що дозволяють створювати та маніпулювати даними у БД є невід'ємним компонентом у розробці веб-додатків, оскільки основна роль СУБД полягає в забезпеченні структурованого зберігання і доступу до даних, необхідних для операцій та функціонування додатку. Основна мета СУБД полягає у забезпеченні надійності, безпеки та оптимізації обробки інформації, що є важливим аспектом для успішної роботи веб-додатків.

Системи управління базами даних використовують різні моделі для організації та зберігання даних. Кожна модель має свої унікальні особливості і відповідає різним вимогам та типам додатків. Основні моделі СУБД включають: реляційну, ієрархічну, мережеву, об'єктно-орієнтовану та документно-орієнтовану моделі.

Реляційна модель є найбільш широко використовуваною серед комерційних СУБД у сучасних інформаційних системах. Вона базується на таблицях, що складаються з рядків і стовпців, де кожен рядок представляє окремий запис, а кожний стовпець - атрибут цього запису, що дозволяє чітко визначити структуру і тип даних, які зберігаються в базі.

Основна перевага реляційних баз даних полягає в їхній здатності забезпечувати високу цілісність і достовірність даних завдяки системі обмежень і правил. Ця система регулює допустимі значення та зв'язки між таблицями, що виконується за допомогою концепцій нормалізації. Нормалізація дозволяє уникнути дублювання даних і зменшує ризик виникнення непередбачених ситуацій при внесенні змін до структури бази даних.

Реляційні бази даних використовують структурований запит до даних через мову SQL, що дозволяє виконувати широкий спектр операцій, таких як вставка, оновлення, видалення і вибірка даних. Це забезпечує ефективний і

стандартизований спосіб взаємодії з базою даних, що полегшує розробку додатків і інтеграцію з іншими системами.

Хоча реляційні бази даних мають добре визначену структуру і типові операції з даними, що робить їх ідеальними для обробки структурованої інформації. Проте, вони можуть бути менш ефективними при роботі з великими обсягами даних, які не мають чітко визначеної структури, або при виконанні складних запитів, що потребують обробки великої кількості з'єднань між таблицями.

Таким чином, реляційні бази даних залишаються основою для багатьох додатків завдяки своїй надійності, гнучкості і потужності для обробки структурованих даних. Їх вибір є оптимальним для систем, де важлива чітка структура даних, висока цілісність і стандартизовані методи взаємодії з базою даних.

Серед реляційних СУБД можна виділити наступні популярні системи:

MongoDB - документно-орієнтована NoSQL база даних, розроблена для зберігання та управління даними у вигляді документів у форматі BSON. Ця база даних набула популярності завдяки своїй гнучкій схемі даних, масштабованості та високій швидкодії.

Основні характеристики:

1. Гнучкість схеми: MongoDB не вимагає строгих схем даних, що дозволяє змінювати структуру документів в процесі розвитку додатка без переробки всіх існуючих даних.
2. Масштабованість: MongoDB підтримує горизонтальне масштабування шляхом розподілення даних на кластери. Це дозволяє обробляти великі обсяги даних та запитів, забезпечуючи високу доступність системи.

3. Індексація і швидкодія: MongoDB підтримує індексацію для швидкого доступу до даних та оптимізації запитів. Вона також використовує кешування в пам'яті для підвищення продуктивності.

Переваги:

- Гнучкість структури даних .
- Висока швидкодія.
- Висока доступність і надійність.

Недоліки:

- Специфічність документно-орієнтованої моделі.
- Вимоги до ресурсів.

PostgreSQL – це потужна, об'єктно-реляційна система управління базами даних з відкритим вихідним кодом, яка підтримує більшість стандартів SQL і надає розширені функціональні можливості для розробки складних додатків.

Основні характеристики:

1. Підтримка транзакцій: Забезпечення повної підтримки транзакцій, включаючи ACID (Atomicity, Consistency, Isolation, Durability), що гарантує надійність та цілісність даних.
2. Безпека: Вбудовані механізми контролю доступу, аутентифікації та шифрування даних забезпечують високий рівень безпеки.
3. Інтеграція з іншими технологіями: легко інтегрується з багатьма іншими технологіями та інструментами, включаючи різні мови програмування, фреймворки та системи управління.

4. Розширюваність: Надає можливість користувачам створювати власні типи даних, функції та оператори, а також використовувати сторонні модулі та розширення для розширення функціональних можливостей.
5. Підтримка JSON: Вбудована підтримка JSON дозволяє зберігати та маніпулювати напівструктурованими даними.
6. Висока продуктивність: Завдяки оптимізованому ядру та підтримці різноманітних індексів (B-tree, GiST, GIN, BRIN) PostgreSQL забезпечує високу продуктивність навіть при великих обсягах даних.

Переваги:

- Надійність та стабільність.
- Гнучкість та масштабованість.
- Висока продуктивність.
- Безкоштовність і відкритий вихідний код.

Недоліки:

- Велике використання ресурсів.
- Складність налаштування.

Oracle Database – провідна комерційна реляційна система управління базами даних розроблена компанією Oracle Corporation. Відома своєю надійністю, продуктивністю, масштабованістю та набором великої кількості функцій для високих навантажень та великих обсягів даних.

Основні характеристики:

1. Масштабованість: система підтримує масштабованість вертикальну (шляхом збільшення ресурсів окремого сервера), та горизонтальну (шляхом розподілу навантаження між кількома серверами), що

дозволяє обробляти величезні обсяги даних і витримувати високу навантаженість.

2. Висока продуктивність: Oracle Database забезпечує високу продуктивність завдяки оптимізації запитів, використанню передових алгоритмів індексації, кешуванню даних та підтримці паралельного виконання запитів.
3. Безпека: Oracle надає розширені функції безпеки, включаючи шифрування даних, контроль доступу на рівні рядків, управління користувачами і ролями, що забезпечує захист даних від несанкціонованого доступу та втручання.
4. Розширюваність і інтеграція: підтримка безлічі типів даних та наявність потужних інструментів для інтеграції з іншими системами і додатками, дозволяє легко адаптувати базу даних до потреб конкретного проєкту.
5. Гнучкість у розгортанні: Oracle Database може бути розгорнута як на фізичних серверах, так і в хмарних середовищах, забезпечуючи гнучкість у виборі інфраструктури та можливість легкого масштабування.

Переваги:

- Висока продуктивність.
- Надійність та масштабованість.
- Великий спектр функцій для управління даними та бізнес-аналітики.
- Розширені функції безпеки

Недоліки:

- Високі вартість на ліцензії та обслуговування.
- Складність налаштування та адміністрування.

Проведений аналіз різних систем управління базами даних показав, що кожна з розглянутих СУБД має свої сильні сторони та особливості, які відповідають різним вимогам застосування. Враховуючи потреби для розробки нашого веб-додатку, який потребує високої надійності оптимальним вибором був визначений PostgreSQL. Система підтримує різноманітні типи даних, включаючи JSON дані, що робить її ідеальним вибором для веб-додатків, які працюють з різноманітними типами інформації та потребують високої стабільності та розширюваності. Також важливою перевагою є підтримка складних SQL запитів і розширених аналітичних функцій, що забезпечує високу продуктивність обробки даних.

PostgreSQL відповідає сучасним вимогам до надійної, продуктивної та гнучкої системи управління базами даних для веб-додатків. Розширені можливості роблять її ідеальним вибором для будь-якого проєкту, що потребує ефективного зберігання, обробки та доступу до даних у веб-середовищі.

2.3 Аналіз сучасних технологій клієнтської частини веб-додатку

Веб-інтерфейс є важливою складовою сучасних веб-додатків, забезпечуючи зручність та ефективність взаємодії користувачів з програмним забезпеченням. Вибір відповідних технологій для розробки веб-інтерфейсу має впливове значення не тільки на зовнішній вигляд додатку, але й його продуктивність, масштабованість та загальний користувацький досвід. Основними компонентами клієнтських технологій є HTML, CSS та JavaScript.

HTML - основна мова розмітки для створення веб-сторінок, що визначає їх структуру вмісту за допомогою елементів та тегів. HTML використовується браузерами для інтерпретації і відображення вмісту веб-сторінок для користувачів. Вона є основним стандартом для побудови веб-інтерфейсів та

забезпечує основний каркас для всіх інших технологій, які використовуються для розширення функціональності веб-додатків.

CSS - це мова опису стилів, яка використовується для задання вигляду HTML-документів і форматування веб-сторінок. Основним завданням CSS є розділення вмісту сторінки від його представлення, що дозволяє ефективно керувати виглядом кожного елемента на сторінці.

Окрім чистого CSS, також існує безліч бібліотек та фреймворків стилізації, які спрощують розробку веб-інтерфейсів та надають додаткові можливості для створення привабливого інтерфейсу. Деякі з них користуються значною популярністю, а саме такі як: Bootstrap, Tailwind, Foundation, Bulma, Materialize.

JavaScript - високорівнева інтерпретована мова програмування, яка використовується для створення динамічних веб-сторінок, є однією з основних технологій веб-розробки та грає важливу роль у забезпеченні взаємодії користувачів з веб-сторінками.

Окрім цих основних технологій, сучасний ринок пропонує широкий спектр інструментів та фреймворків для створення веб-інтерфейсів, які є важливими складовими сучасної розробки, надаючи різноманітні функції та інструменти для полегшення процесу розробки, забезпечення оптимальної продуктивності та підвищення користувацького досвіду. Кожен з них має свої унікальні переваги та особливості.

React - популярна бібліотека, що розроблена компанією Facebook для створення користувацьких інтерфейсів з використанням компонентного підходу. Окрім того бібліотека є одним з найпопулярніших виборів для створення односторінкових додатків (SPA) та додатків з великою кількістю інтерактивності. React дозволяє розробникам створювати великі веб-додатки, які

можуть змінювати дані без перезавантаження сторінки. Основною метою React є надання швидкості, простоти та масштабованості.

Основні характеристики React:

1. Компонентний підхід: React побудований на компонентній архітектурі. Компоненти є основними блоками React-додатків і дозволяють розробникам розділяти інтерфейс на незалежні, повторно використовувані частини. Кожен компонент можна розглядати як невеликий, самодостатній модуль, який містить свій власний логічний і візуальний зміст.
2. JSX (JavaScript XML): JSX - синтаксичне розширення для JavaScript, яке дозволяє розробникам писати HTML-подібний код всередині JavaScript. Це робить код більш читабельним і зрозумілим. JSX також полегшує виявлення помилок у коді завдяки його суворій синтаксичній перевірці.
3. Односпрямований потік даних: React використовує односпрямований потік даних, що спрощує розуміння і контроль стану додатка. Дані передаються вниз від батьківських компонентів до дочірніх, що робить систему більш передбачуваною та легкими для відстеження.
4. Віртуальний DOM: Однією з ключових інновацій React є використання віртуального DOM. Віртуальний DOM є легковаговою копією реального DOM, що дозволяє React ефективно визначати мінімальні зміни, необхідні для оновлення інтерфейсу користувача. Це значно покращує продуктивність додатків, зменшуючи кількість операцій над реальним DOM.
5. Серверний рендеринг: React підтримує серверний рендеринг (SSR), що дозволяє рендерити компоненти на сервері до їх передачі на клієнт. Це покращує продуктивність додатків і SEO, забезпечуючи швидке завантаження сторінок і кращу видимість для пошукових систем.

6. Управління станом: React надає можливість керування станом компонентів через об’єкт state. Стан компонента можна змінювати, що призводить до повторного рендерингу компонента. Для більш складного управління станом у великих додатках використовуються додаткові бібліотеки, такі як Redux та MobX.
7. Універсальність: React можна використовувати не тільки для веб-додатків. За допомогою React Native, розробники можуть створювати нативні мобільні додатки для iOS та Android, використовуючи той самий підхід.
8. Розширюваність та екосистема: React має розвинуту екосистему та багато інструментів, які спрощують розробку додатків. Надає такі інструменти, як React DevTools для налагодження, Create React App для швидкого створення нових проєктів, та Next.js для серверного рендерингу і генерації статичних сайтів.

Переваги:

- Висока продуктивність.
- Гнучкість і розширюваність.
- Легка інтеграція.
- Велика спільнота та підтримка.

Недоліки:

- Погана документація.
- Складність синтаксису JSX.
- Необхідність у додаткових бібліотеках.

Angular - фреймворк розроблений компанією Google, використовується для розробки односторінкових веб-додатків на основі TypeScript. Angular надає

широкий набір інструментів для створення складних інтерфейсів і співпрацює з різноманітними сервісами веб-розробки.

Основні характеристики:

1. Компонентна архітектура: фреймворк базується на компонентній архітектурі, де кожен елемент інтерфейсу є компонентом. Компоненти є самодостатніми блоками, які містять HTML-шаблони, CSS-стилі і TypeScript-код для логіки компонента.
2. Типізація: використання TypeScript - строго типізованої версії JavaScript дозволяє виявляти і виправляти помилки на етапі розробки, що значно покращує стабільність і підтримку коду, особливо в великих проєктах.
3. Шаблонізація і директиви: Angular використовує HTML з розширеними можливостями, такими як директиви і власні атрибути, що дозволяють легко маніпулювати DOM і створювати динамічні інтерфейси користувача.
4. Модульність і залежності: Angular підтримує модульну структуру, що дозволяє організувати додаток на незалежні модулі і забезпечує легку управління залежностями між компонентами.

Переваги:

- Широкий набір інструментів.
- Спільнота та підтримка.
- Інструменти для тестування.
- Масштабованість.

Недоліки:

- Відносно повільна продуктивність.

- Висока складність.

Vue - фреймворк з відкритим вихідним кодом для створення односторінкових додатків та користувацьких інтерфейсів, що використовує модель MVVM. Фреймворк відомий своєю простотою і легкістю у використанні, що робить його популярним та активно використовується для розробки сучасних веб-додатків.

Основні характеристики:

1. Гнучкість і поступовий підхід: фреймворк можна впроваджувати поступово в існуючі проєкти, що робить його ідеальним вибором для тих, хто шукає модульний підхід до розробки.
2. Компонентна архітектура: Vue базується на компонентній архітектурі, що дозволяє виокремлювати різні частини інтерфейсу у незалежні компоненти. Кожен компонент має свій власний стан, логіку та представлення, що спрощує розробку, тестування та підтримку коду.
3. Реактивність: Vue використовує власну реалізацію реактивного об'єкта, який автоматично відслідковує зміни в стані компонентів і оновлює відповідні частини DOM, що дозволяє підтримувати додаток у стані який завжди відповідає актуальним даним.
4. Простота використання: Vue має простий і зрозумілий синтаксис, що дозволяє легко визначати структуру інтерфейсу і його логіку.
5. Інтеграція з іншими технологіями: Інтеграція з іншими бібліотеками і інструментами, такими як Vue Router для маршрутизації і Vuex для управління станом, дозволяє створювати комплексні додатки, що відповідають потребам сучасних веб-додатків.

Переваги:

- Простота у використанні.

- Гнучка система.
- Висока продуктивність.
- Проста інтеграція .

Недоліки:

- Обмежена екосистема.
- Обмежена функціональність стандартних бібліотек.

Отже, під час проведення аналізу різних фреймворків для розробки клієнтської частини веб-додатків було розглянуто та визначено переваги та недоліки кожного з них. Кожен з цих інструментів має свої унікальні особливості, що роблять його відповідним для певних типів проєктів та різних вимог застосування. З урахуванням вимог нашого веб-додатку, таких як висока продуктивність, зручність у розробці, масштабованість та підтримка сучасних технологій, оптимальним вибором було визначено бібліотеку React, що відповідає нашим потребам і дозволить досягти необхідних результатів у реалізації проєкту.

React виділяється своєю високою продуктивністю і гнучкістю. Його компонентна архітектура забезпечує легку реюзабельність і підтримку коду, що особливо важливо для великих і складних додатків. Використання Virtual DOM дозволяє мінімізувати оновлення реального DOM, що значно покращує швидкодію додатків. React також має широку і активну спільноту, яка пропонує безліч сторонніх бібліотек і інструментів, що спрощують процес розробки і розширюють можливості фреймворку.

Така бібліотека як Redux Toolkit є ідеальним доповненням до React для управління складним станом додатка. Бібліотека пропонує зручний інтерфейс для організації коду Redux, що дозволяє зберігати стан додатка централізовано і забезпечує простий механізм для зміни стану через дії. Це особливо корисно для

додатків з великою кількістю взаємодій між компонентами або для управління даними, які потребують загального доступу.

Цей підхід дозволяє розробникам підтримувати чистоту коду, полегшує відлагодження та розширення додатків, особливо в умовах зростаючої складності. Разом React і Redux Toolkit забезпечують потужний інструментарій для створення сучасних веб-додатків, які відповідають вимогам ефективності, масштабованості та зручності в розробці.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Архітектура веб-додатку

Архітектура веб-додатків складається з декількох ключових компонентів, які забезпечують функціонування та взаємодію різних частин системи. Основні компоненти архітектури веб-додатку включають:

- **Веб-браузер:** Клієнтський компонент є основною складовою, що взаємодіє з користувачем, обробляє введення даних та керує відображенням інформації. Він контролює взаємодію користувача з додатком і, за необхідності, перевіряє введені дані.
- **Сервер:** Компонент серверної частини виконує бізнес-логіку та обробляє запити від користувачів, направляючи їх до відповідних компонентів. Він управляє всіма операціями програми та може обробляти запити від різних клієнтів.
- **База даних:** Забезпечує додаток необхідними даними. Виконує завдання, пов'язані з обробкою даних. У багаторівневих архітектурах сервери баз даних можуть також виконувати бізнес-логіку через збережені процедури.

Рис. 3.1 – Трирівнева архітектура веб-додатку.

Монолітна архітектура веб-додатку

Розробка веб-додатку була здійснена з використанням монолітної архітектури, яка включає два окремі моноліти: адміністративний і користувацький. Адміністративний моноліт відповідає за управління контентом, тоді як клієнтський моноліт забезпечує інтерфейс для кінцевих користувачів, надаючи їм доступ до основних функцій додатку.

Монолітна архітектура - це підхід до побудови програмного забезпечення, при якому всі компоненти програми об'єднані в одне єдине ціле. У такій архітектурі всі функції додатку, такі як користувацький інтерфейс, бізнес-логіка, доступ до даних та інші служби, інтегровані та розгортаються як одне єдине ціле.

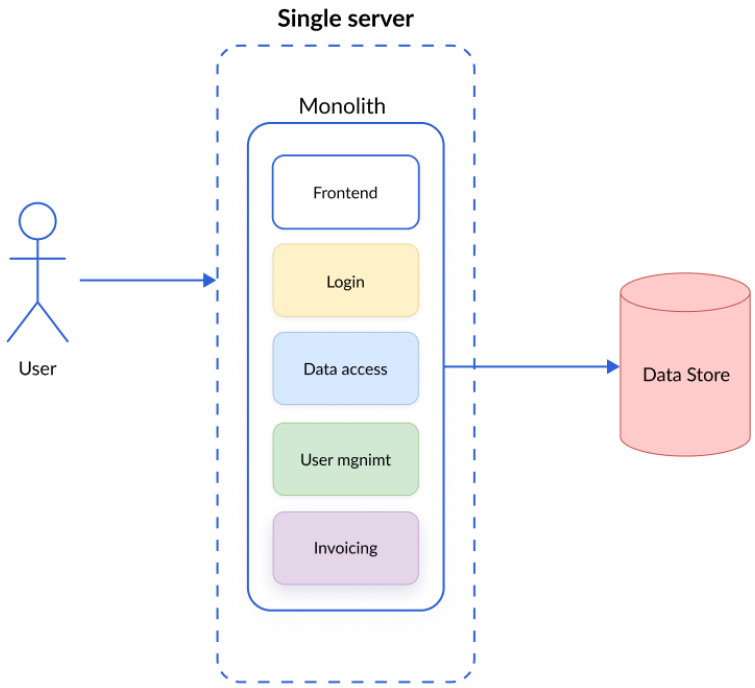

Рис. 3.2 – Монолітна архітектура веб-додатку

Обидва моноліти використовують архітектурний шаблон Flux для управління станом додатку, що забезпечує односторонній потік даних та складається з наступних основних компонентів:

- Dispatcher: Централізований компонент, що відповідає за передачу дій до обробників.

- Stores: Місця зберігання стану додатку, які оновлюються у відповідь на дії.
- Actions: Об'єкти, що інкапсулюють нові дані та команди, які надсилаються через Dispatcher.
- Views: Компоненти, що відображають дані з Stores та ініціюють Actions.

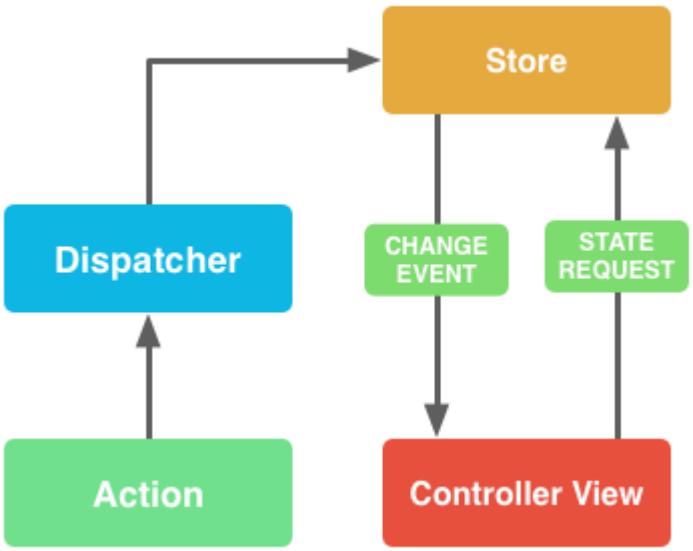

Рис. 3.3 – Архітектура Flux.

Використання шаблону Flux дозволяє чітко розділити логіку додатку, забезпечуючи кращу підтримуваність та передбачуваний і контрольований потік даних. Адміністративний та клієнтський моноліти взаємодіють через визначені API, що забезпечує чітке розділення відповідальностей та зменшує зв'язність між ними. Це дозволяє розгортати та оновлювати кожен моноліт незалежно один від одного.

3.2 Вимоги до веб-додатку

Визначення вимог є ключовим етапом у процесі розробки, який вимагає детального розгляду різноманітних аспектів таких як функціональні і нефункціональні характеристики та можливості системи. Цей процес необхідний для досягнення поставлених цілей проєкту і задоволення потреб користувачів. Визначення вимог включає в себе аналіз потреб і очікування користувачів, а також врахування технічних обмежень і вимог до продукту, що розробляється.

Функціональні вимоги веб-додатку включають:

1. Реєстрація та авторизація користувачів:

    - Нові користувачі повинні мати можливість створювати облікові через GitHub або GitLab .
    - Користувачі повинні мати можливість увійти в систему, використовуючи свої облікові дані.
    - Користувачі у разі потреби повинні мати можливість встановити нові облікові дані.
    - Система повинна включати різні рівні доступу таких як адміністратор та гість.

2. Управління контентом:

    - Адміністратор повинен мати змогу створювати нові пости, редагувати та видаляти їх.
    - Система повинна надавати інформацію про усі дані системи користувачам з роллю «Адміністратор».

3. Взаємодія з постами:

- Користувачі повинні мати змогу переглядати доступні пости без авторизації.

Функціональні вимоги до веб-додатку визначають ключові можливості, необхідні для забезпечення основних операцій користувачів. Виконання цих вимог гарантує, що користувачі зможуть ефективно взаємодіяти з додатком та використовувати всі його функціональні можливості.

Нефункціональні вимоги веб-додатку включають:

1. Продуктивність:

    - Система повинна підтримувати швидкий час відгуку на запити користувачів.
    - Час відображення результатів повинен бути мінімальним.

2. Безпека:

    - Система повинна забезпечувати захист даних користувачів від несанкціонованого доступу.
    - Механізми аутентифікації та авторизації повинні бути імплементовані згідно з сучасними стандартами безпеки.

3. Інтерфейс користувача:

    - Інтерфейс користувача повинен бути зрозумілим та зручним у використанні.
    - Інтерфейс повинен коректно відображатися на різних пристроях і розширеннях екранів, забезпечуючи однакову функціональність та зручність у користуванні.

Нефункціональні вимоги гарантують, що додаток буде надійним, продуктивним, безпечним та доступним для користувачів різних пристроїв. Дотримання цих вимог є фундаментальною основою для створення якісного і зручного веб-додатку, який відповідає очікуванням користувачів та бізнес-цілям проєкту.

3.3 Програмна реалізація веб-додатку “Блог”

Для реалізації серверної частини веб-додатку було використано платформу Node.js та фреймворк NestJS. Node.js забезпечує ефективне, асинхронне виконання коду на стороні сервера, що дозволяє створювати високопродуктивні та масштабовані веб-додатки.

На етапі розробки веб-додатку першим кроком було створення файлу package.json, який є центральною точкою конфігурації проєкту. Цей файл визначає всі необхідні залежності, скрипти для автоматизації завдань та інші параметри, необхідні для повноцінного функціонування додатку.

Рис. 3.4 Підключені бібліотеки проєкту

NestJS - це модульний фреймворк для створення ефективних і масштабованих серверних застосунків на Node.js.

Prisma – бібліотека, яка дозволяє взаємодіяти з базами даних через програмні об'єкти.

Argon2 - бібліотека для хешування паролів.

Class-transformer – бібліотека для простого конвертування об'єктів з одного формату в інший, таких як JSON в класи і навпаки.

Class-validator - бібліотека для валідації даних.

Rxjs – бібліотека, яка використовується для роботи з асинхронними подіями.

Passport - бібліотека для аутентифікації користувачів.

Passport-jwt - модуль який дозволяє використовувати JWT для аутентифікації користувачів.

JWT - це спосіб передачі інформації між двома сторонами у вигляді JSON об'єкта, що складається з трьох частин: заголовка, який містить метадані про токен, тіла що містить корисну інформацію та підпису, який забезпечує його перевірку та відновлення. JWT зазвичай використовується для аутентифікації і авторизації користувачів у веб-додатках.

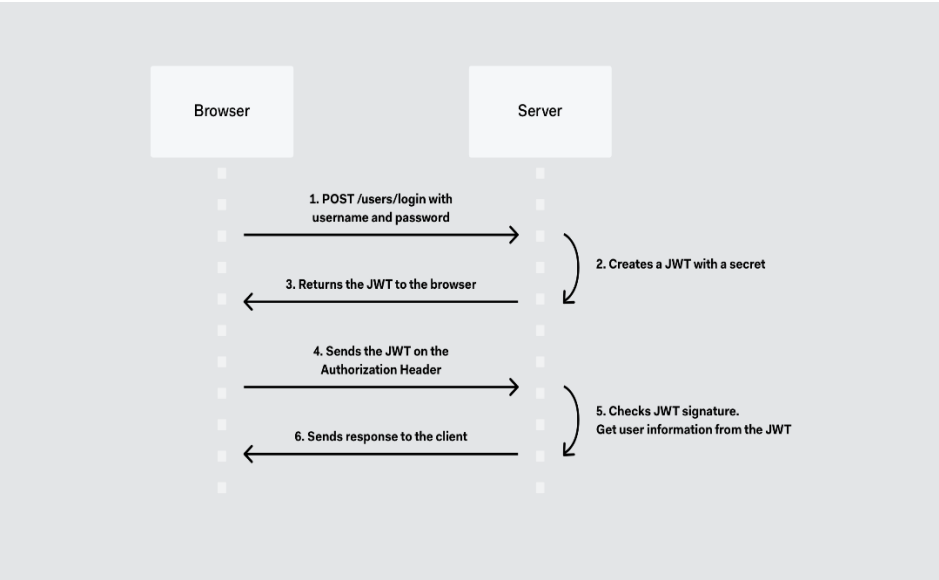

Рис. 3.5 - Аутентифікація за допомогою токену

Принцип роботи JWT:

Після того, як користувач аутентифікується на сервері, сервер генерує JWT, який містить інформацію про користувача та деякі метадані. Токен підписується за допомогою секретного ключа або пари ключів, для забезпечення цілісності даних під час їх передачі. Після підписання JWT передається користувачу, який надалі буде використовувати його для подальшої аутентифікації або доступу до ресурсів.

Коли користувач надсилає запит з JWT на сервер, сервер перевіряє підпис токена, щоб переконатися, що дані не були змінені під час передачі. Після перевірки сервер розшифровує токен і використовує інформацію, що міститься в ньому, для визначення ідентифікації користувача або встановлення його прав доступу.

JWT дозволяє забезпечити безпечний обмін даними між клієнтом і сервером, зменшуючи потребу у зберіганні стану сеансу на сервері і покращуючи масштабованість системи.

Для забезпечення безпеки у веб-додатку використовуються гуарди та стратегії.

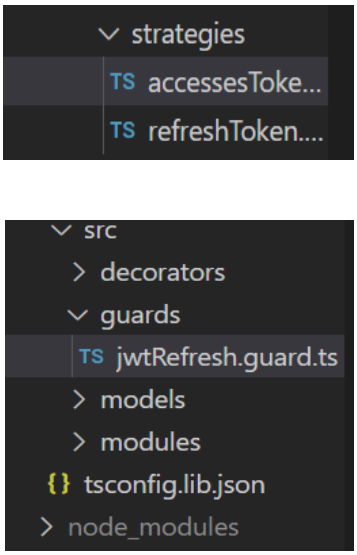

Рис. 3.6 – Файлова структура системи захисту

У папці «strategies» класи «AccessTokenStrategy» та «RefreshTokenStrategy» виконують функції аутентифікації користувачів, забезпечуючи безпеку та контроль доступу до різних ресурсів.

Guard – механізм, що дозволяє налаштовувати і контролювати доступ до різних частин веб-додатка. Guard перевіряє доступ за допомогою JWT-токена, дозволяючи доступ лише тим користувачам, які мають дійсний токен оновлення.

apps > admin > src > modules > auth > strategies > TS accessesToken.strategy.ts > ...

```ts
import { Injectable } from '@nestjs/common'
import { ConfigService } from '@nestjs/config'
import { PassportStrategy } from '@nestjs/passport'
import { ExtractJwt, Strategy } from 'passport-jwt'

type JwtPayload = {
  sub: string
  email: string
}

@Injectable()
export class AccessTokenStrategy extends PassportStrategy(Strategy, 'jwt') {
  constructor(config: ConfigService) {
    super({
      jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
      secretOrKey: config.get<string>('ADMIN_JWT_SECRET_AT'),
    })
  }

  validate(payload: JwtPayload) {
    return payload
  }
}
```

Рис. 3.7 – Аутентифікація користувача за допомогою JWT токена доступу.

```ts
import { Injectable } from '@nestjs/common'
import { PassportStrategy } from '@nestjs/passport'
import { ConfigService } from '@nestjs/config'
import { ExtractJwt, Strategy } from 'passport-jwt'
import { Request } from 'express'

@Injectable()
export class RefreshTokenStrategy extends PassportStrategy(
  Strategy,
  'jwt-refresh',
) {
  constructor(config: ConfigService) {
    super({
      jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
      secretOrKey: config.get<string>('ADMIN_JWT_SECRET_RT'),
      passReqToCallback: true,
    })
  }

  validate(req: Request, payload: any) {
    const refreshToken = req.get('authorization').replace('Bearer', '').trim()
    return {
      ...payload,
      refreshToken,
    }
  }
}
```

Рис. 3.8 – Аутентифікація користувача за допомогою JWT токена оновлення.

Для повноцінного функціонування серверної частини створюємо та підключаємось до бази даних через файл «.env», який знаходиться в кореневій папці проєкту.

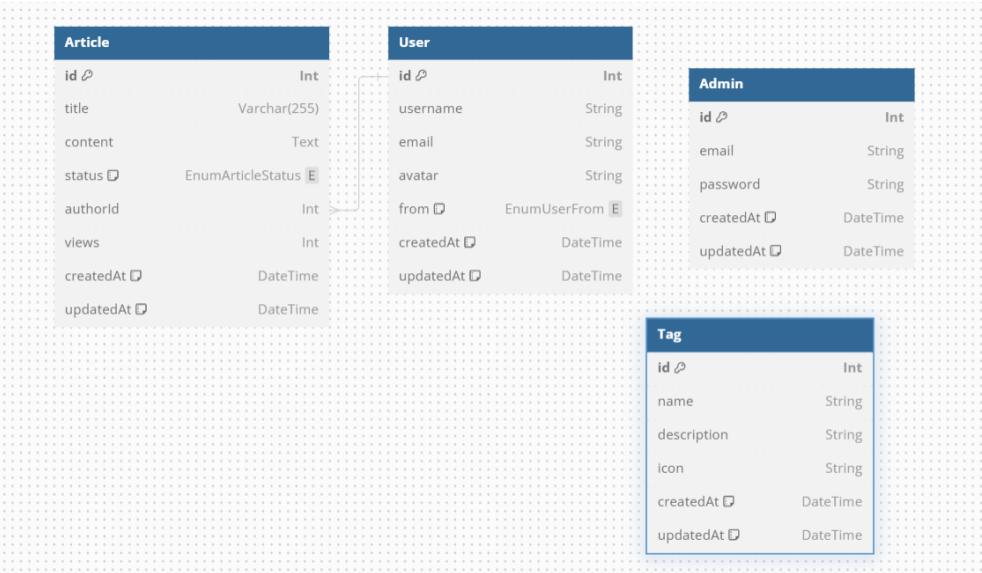

Рис. 3.9 – Структура бази даних

Наступним кроком було створення через Prisma ORM самих моделей, що відображають структуру даних у веб-додатку. Моделі використовуються для взаємодії з базою даних та визначають основні поля і відносини між сутностями.

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model User {
  id       Int        @id @default(autoincrement())
  username String
  email    String     @unique
  avatar   String
  from     E_UserFrom

  Article             Article[]
  ArticleComments     ArticleComment[]
  ArticleCommentVotes ArticleCommentVotes[]
  ArticleReactions    ArticleReaction[]     @relation("ArticleReactionOnUser")

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
}

enum E_UserFrom {
  GITHUB
  GITLAB
}
```

Рис. 3.10 – Створення моделі користувача.

```
model Admin {
  id       Int    @id @default(autoincrement())
  email    String @unique
  password String

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
}
```

Рис. 3.11 – Створення моделі адміністратора.

```
model Article {
  id       Int             @id @default(autoincrement())
  title    String          @db.VarChar(255)
  content  String
  status   E_ArticleStatus
  author   User            @relation(fields: [authorId], references: [id])
  authorId Int
  views    Int

  tags Tag[] @relation("TagsOnArticles")

  ArticleComment  ArticleComment[]
  ArticleReaction ArticleReaction[]

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
}

enum E_ArticleStatus {
  DRAFT
  CHECKED
  PUBLISHED
}
```

Рис. 3.12 – Створення моделі статті.

Після чого генеруємо міграцію для оновлення структури бази даних. Міграція в базі даних - це процес зміни структури або версії бази даних без втрати даних, що вже зберігаються в ній. Основна мета міграції полягає в забезпеченні того, щоб будь-які зміни в базі даних були виконані безпечно і ефективно.

```sql
-- CreateEnum
CREATE TYPE "UserFrom" AS ENUM ('GITHUB', 'GITLAB');

-- CreateTable
CREATE TABLE "Article" (
    "id" SERIAL NOT NULL,
    "title" VARCHAR(255) NOT NULL,
    "createdAt" TIMESTAMP(6) NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "content" TEXT NOT NULL,

    CONSTRAINT "Article_pkey" PRIMARY KEY ("id")
);

-- CreateTable
CREATE TABLE "User" (
    "id" SERIAL NOT NULL,
    "createdAt" TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "username" TEXT NOT NULL,
    "email" TEXT NOT NULL,
    "avatart" TEXT NOT NULL,
    "from" "UserFrom" NOT NULL,

    CONSTRAINT "User_pkey" PRIMARY KEY ("id")
);
```

Рис. 3.13 - Створення міграції.

Далі наповнюємо нашу базу даних початковими даними, використовуючи Prisma Client для створення нових записів у відповідних таблицях.

```ts
import { E_UserFrom, PrismaClient } from '@prisma/client'

const prisma = new PrismaClient()

export const usersSeed = async () => {
  return await prisma.user.create({
    data: {
      avatar: 'https://avatars.githubusercontent.com/u/25635916?v=4',
      email: 'user@mail.com',
      from: E_UserFrom.GITHUB,
      username: 'Yaroslav',
    },
  })
}
```

Рис. 3.14 - Наповнення бази даних.

Інтерфейс програми, побудований на базі React з інтеграцією Redux, що забезпечує ефективний механізм керування станом додатку та зручне управління

взаємодією між компонентами. Redux дозволяє організувати стан додатку, що полегшує відлагодження, розширення та підтримку коду, зокрема шляхом централізації управління станом та застосування однозначного потоку даних.

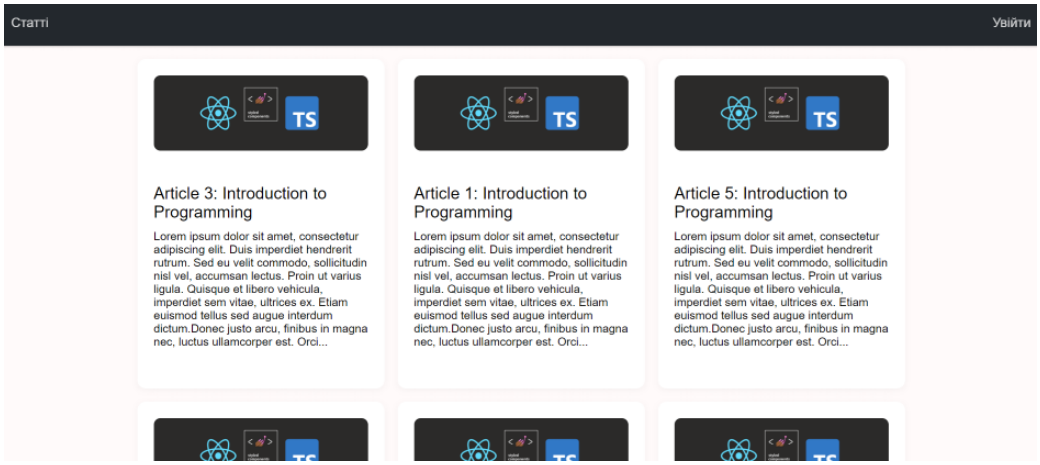

Рис. 3.15 – Вигляд головної сторінки веб-додатку.

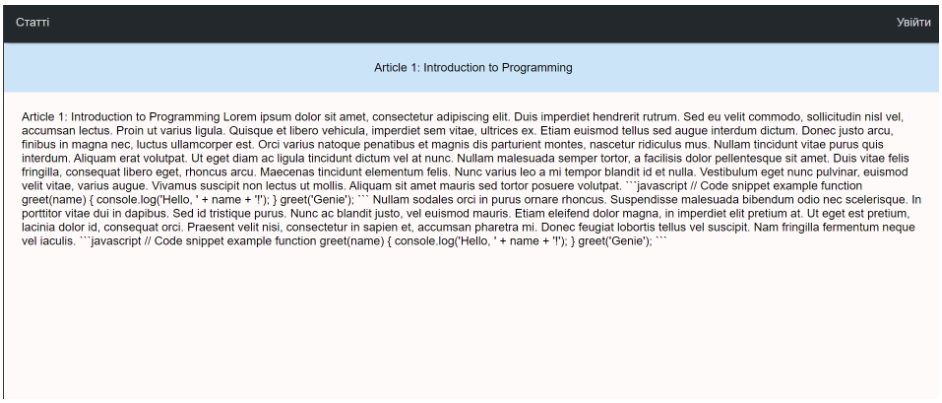

Рис. 3.16 – Вигляд сторінки обраної статті.

Користувачу надається можливість авторизуватися за допомогою облікових записів GitHub та GitLab. Цей процес забезпечує зручний та безпечний спосіб входу до системи, використовуючи наявні облікові дані. Після авторизації користувач перенаправляється на відповідні форми входу GitHub або GitLab, де він повинен підтвердити свій доступ. Після успішної аутентифікації, користувач повертається до додатку, що дозволяє зберігати безперервність роботи та захищати конфіденційність даних.

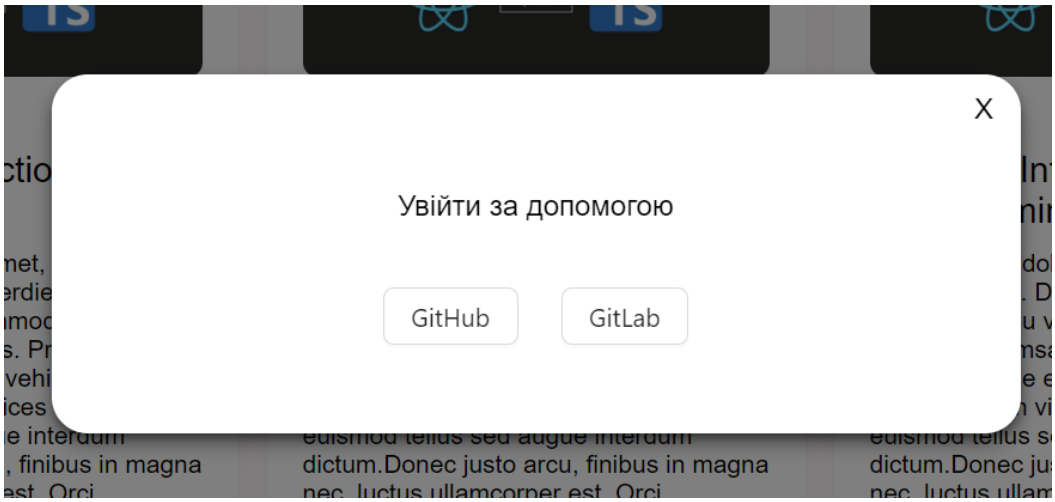

Рис. 3.16 – Вигляд форми авторизації користувача.

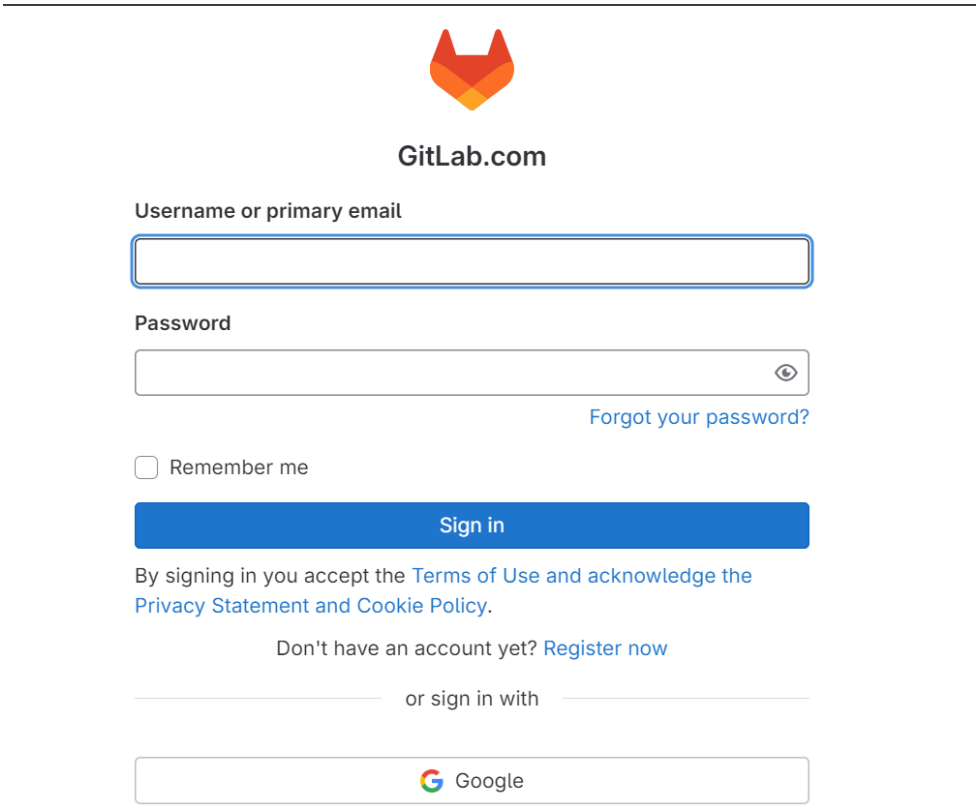

Рис. 3.17 – Перенаправлення на GitLab

```tsx
export const AuthModal = () => {
  const closeLastModal = useModalManagerStore((state : I_ModalManagerStore) => state.closeLastModal)

  const handleCloseModal = () => {
    closeLastModal()
  }

  const handleGitHubAuth = async () => {
    location.replace(import.meta.env.VITE_GITHUB_AUTH_URL)
  }

  const handleGitLabAuth = async () => {
    location.replace(import.meta.env.VITE_GITLAB_AUTH_URL)
  }


  return (
    <S.BaseModal>
      <S.CloseButton onClick={handleCloseModal}>X</S.CloseButton>
      <S.ContentModal>
          <div>Увійти за допомогою</div>
        <S.ButtonstModal>
      <Button  onClick={handleGitHubAuth}>GitHub</Button>
      <Button onClick={handleGitLabAuth}>GitLab</Button>
      </S.ButtonstModal>
</S.ContentModal>
    </S.BaseModal>
```

Рис. 3.18 – Код форми авторизації

Для створення статей автору надається доступ до адміністративної панелі. У цій панелі автор має можливість легко керувати контентом, створювати нові матеріали, редагувати існуючі та публікувати їх на платформі. Адміністративна панель спроектована таким чином, щоб забезпечити інтуїтивно зрозумілий інтерфейс та інструменти для роботи з контентом.

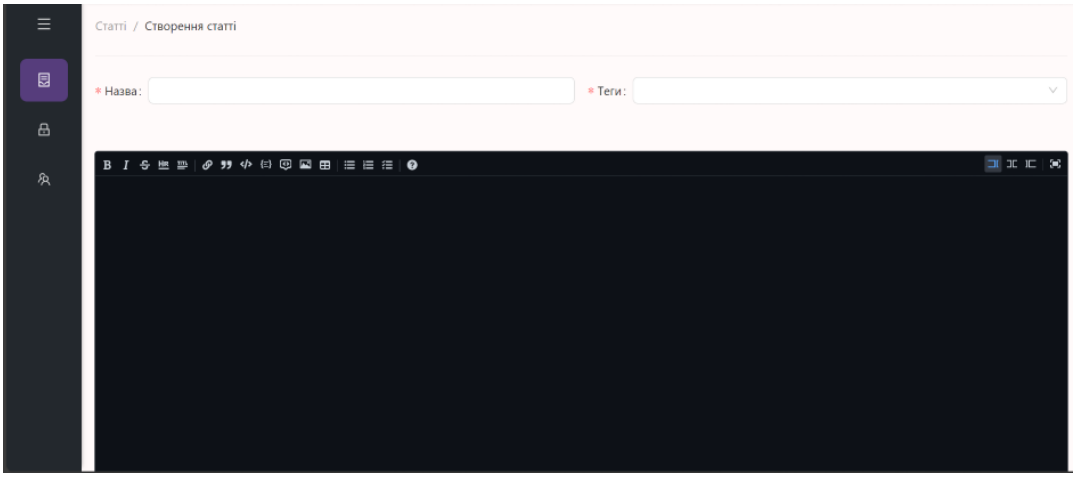

Рис. 3.19 – Створення статті.

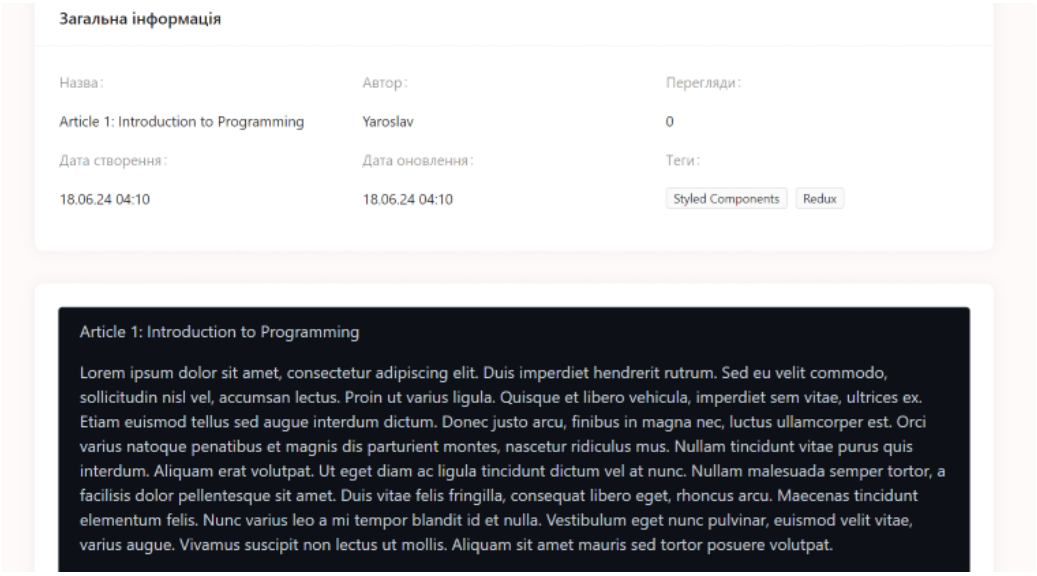

Рис. 3.20 – Перегляд загальної інформації.

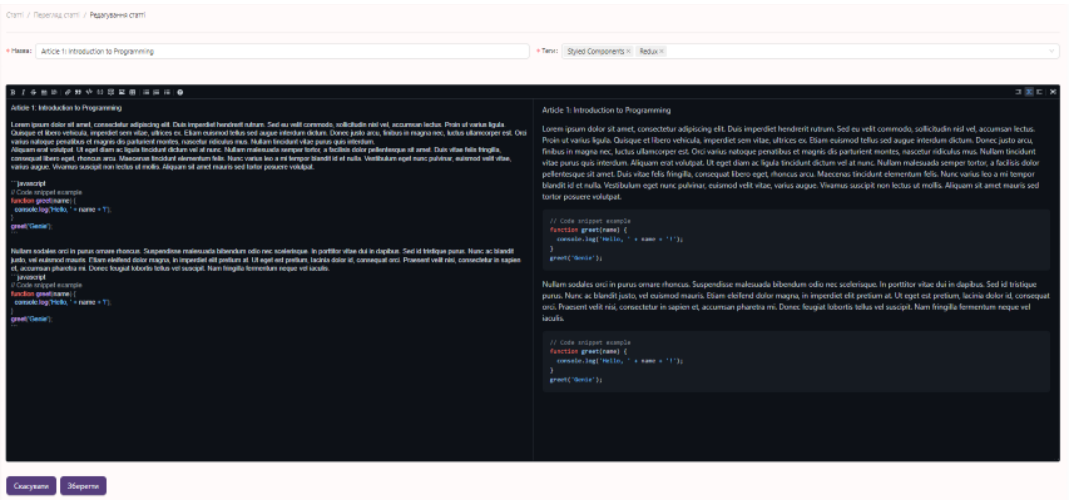

Рис. 3.21 – Редагування статті.

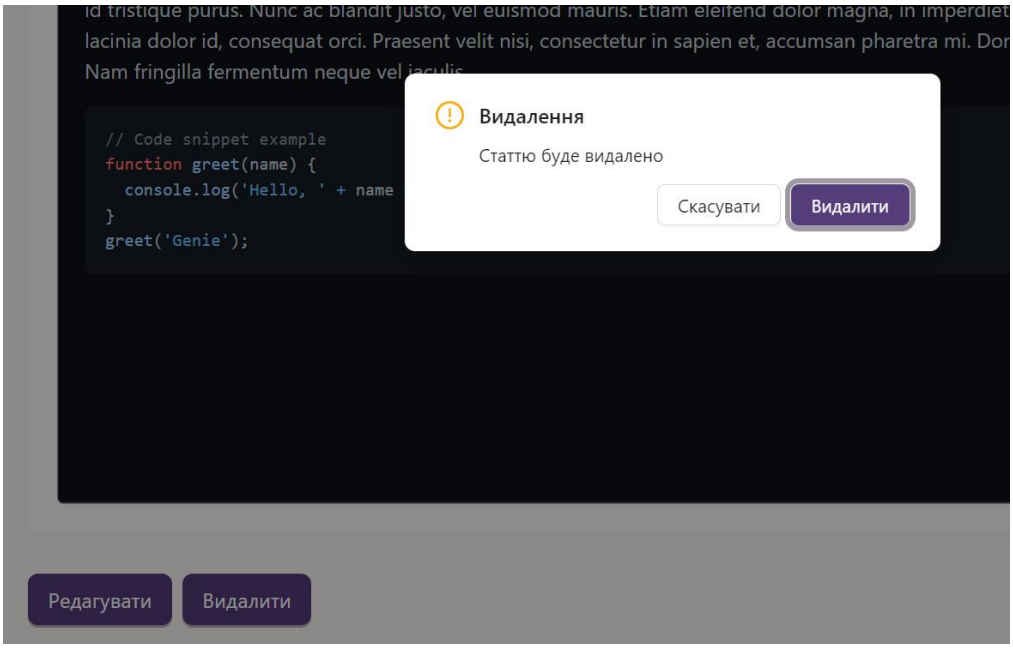

Рис. 3.22 – Видалення статті.

Адміністративна панель також дозволяє переглядати існуючі облікові записи адміністраторів та створювати нові. Це забезпечує гнучке управління ролями та правами доступу, дозволяючи призначати нових.

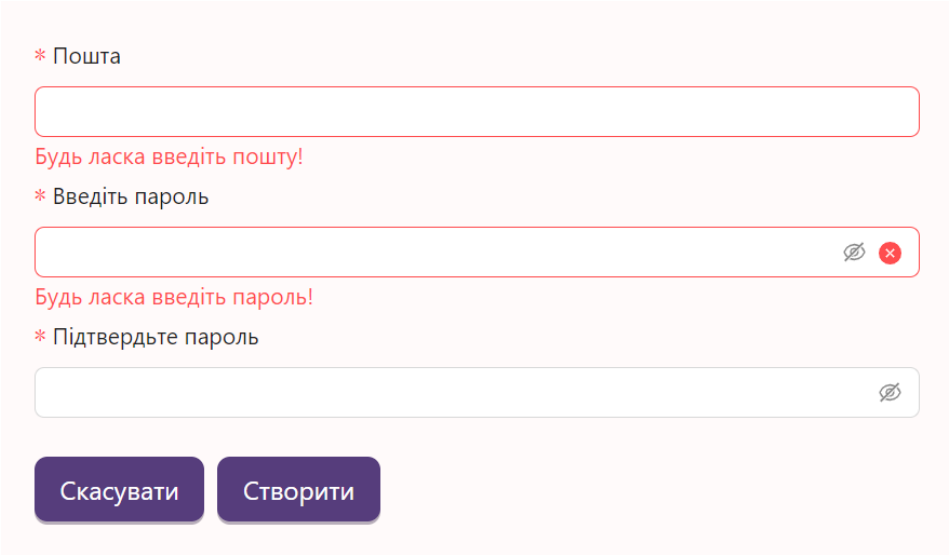

Рис. 3.23 – Форма створення адміністратора.

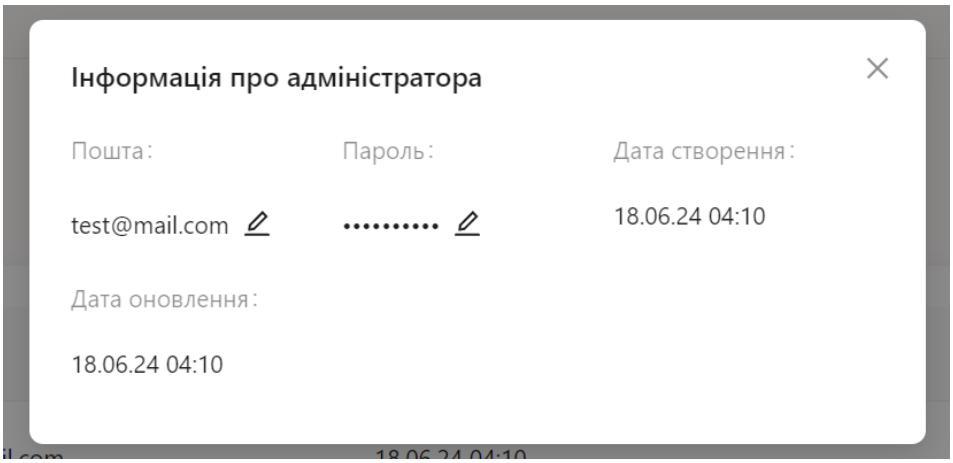

Рис. 3.24 - Перегляд інформації про адміністратора.

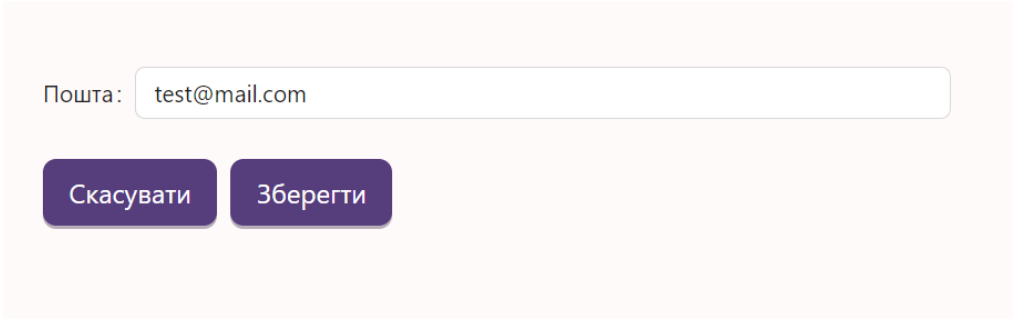

Рис. 3.25 - Редагування пошти адміністратора.

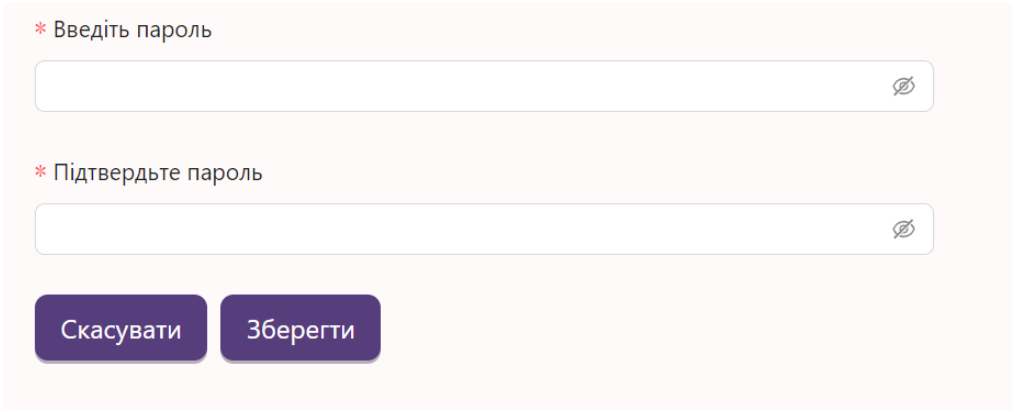

Рис. 3.26 – Зміна паролю адміністратора.

Також на платформі доступна можливість перегляду детальної інформації про користувачів. Це включає основні дані про кожного користувача, такі як ім'я, електронну адресу, дату реєстрації та інші профільні дані.

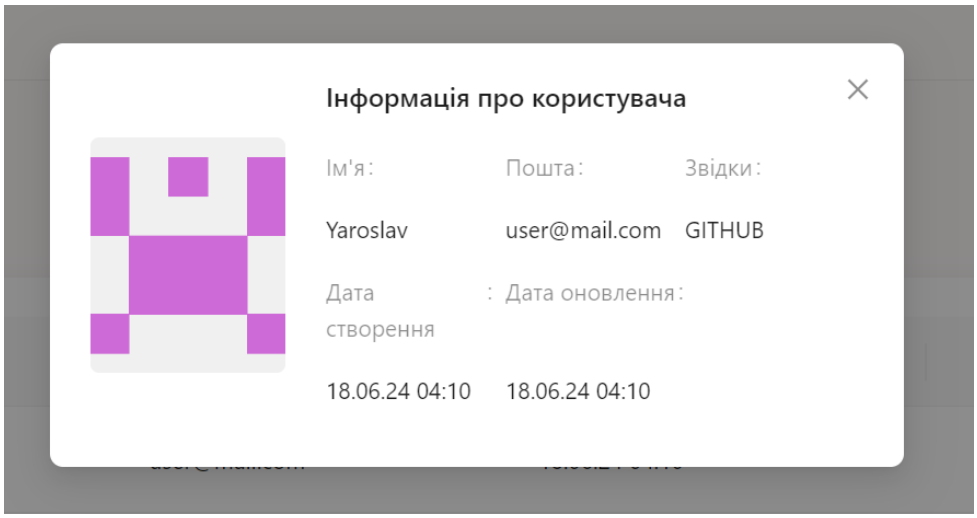

Рис. 3.27 – Перегляд інформації про користувача.

На платформі доступна можливість переглядати різні списки у вигляді таблиць, включаючи усі статті, адміністраторів та користувачів. Кожна таблиця має функції фільтрування, що дозволяють швидко знаходити потрібні дані та впорядковувати їх за різними критеріями. Таким чином, адміністраторам забезпечено зручний інструмент для управління та аналізу інформації, спрощуючи моніторингу на платформі.

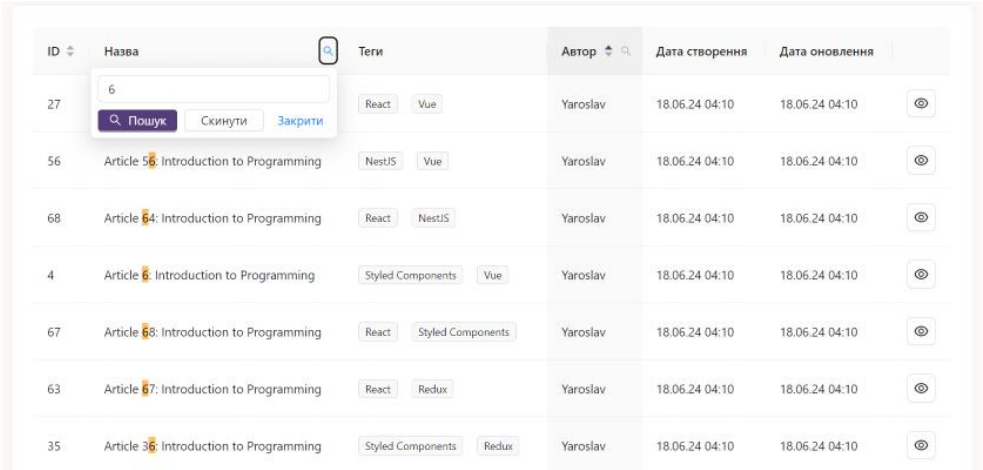

Рис. 3.28 –Таблиця з статтями

ВИСНОВКИ

У цій дипломній роботі було досліджено та реалізовано процес розробки веб-додатку типу блог, який включає у себе ключові компоненти і функціональні можливості для зручного управління контентом. В роботі були розглянуті сучасні технології розробки, зокрема використання React для створення інтерфейсу користувача та Redux Toolkit для управління станом додатку.

Основні аспекти роботи включали проектування та реалізацію бази даних для зберігання статей та інших важливих даних, розробку адміністративної панелі для управління контентом, а також інтеграцію з механізмами зовнішньої аутентифікації користувачів.

Розроблений веб-додаток “Блог” є ефективним інструментом для створення та управління контентом, який може бути використаний як в особистих проєктах, так і в комерційних застосуваннях. Цей проєкт може служити основою для подальшого розвитку та вдосконалення, зокрема додавання нових функцій та оптимізації існуючих. В цілому, цей веб-додаток може бути цікавим для користувачів, які зацікавлені у створенні особистих авторських блогів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Трофименко О. Г. Веб-дизайн та HTML-програмування / О. Г. Трофименко. – Одеса: Фенікс, 2018. – 194 с
2. Mead, A. (2018). The Complete Node.js Developer Course (3rd ed.).Udemy
3. React [Електронний ресурс] – Режим доступу до ресурсу:https://react.dev/learn.
4. JavaScript [Електронний ресурс] – Режим доступу до ресурсу:https://developer.mozilla.org/en-US/docs/Web/JavaScript.
5. Alex Banks and Eve Porcello. Learning React: Functional Web Development with React and Redux 1st edition. O Reilly, 2017. 350p
6. Документація до Prisma ORM: веб сайт. URL: https://www.prisma.io/docs.(дата звернення: 06.05.2023).
7. Документація Nest.js[Електронний ресурс] : https://docs.nestjs.com/
8. Документація Redux[Електронний ресурс] : https://redux.js.org/
9. Документація Antd[Електронний ресурс]: https://ant.design/
10. Хавербеке Марейн. Виразний JavaScript. 2 видання, 2015. – 745 с.