Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента *Грудєва Антона Михайловича*
(ПІБ)

академічної групи *122М-20-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *«122 Комп'ютерні науки»*
(назва освітньої програми)

на тему: *Спеціалізована інформаційна система для забезпечення форуму IT-фахівців*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	*Проф. Мороз Б.І.*			
економічний	*Доц. Касьяненко Л.В.*			

Рецензент				

Нормоконтролер	*Доц. Реута О.В.*			

Дніпро
2022

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 20 22 року

ЗАВДАННЯ
на виконанння кваліфікаційної роботи

спеціальності
122 Комп'ютерні науки
(код і назва спеціальності)

студенту *122М-20-2* *Грудєву Антону Михайловичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Спеціалізована інформаційна система для забезпечення форуму IT-фахівців*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 10.12.2021 р. № 1036–с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процеси проєктування, розробки і використання вебсайтів та вебзастосунків на базі технологій стеку MERN та програмної платформи Node.js.

Предмет досліджень – моделі, методи та інформаційні технології серверної та клієнтської складових для забезпечення якості вебресурсів при створенні інформаційних систем – вебсайтів.

Мета НДР – підвищення ефективності роботи вебсайту форуму шляхом підбору сукупності технологій, які забезпечили б гнучкість системи з погляду внесення нового функціоналу та легкість підтримки в умовах модифікування вимог до неї в цілому та її окремих характеристик.

Вихідні дані для проведення роботи – теоретичні та експериментальні

дослідження для вирішення задач проєктування інформаційної системи вебсайту та підходи до вибору технологій реалізації.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень визначається тим, що розроблено і обґрунтовано модель реалізації вебсайту форуму для IT-фахівців, що дозволило ефективно виконувати модифікацію, тобто внесення змін до системи, на основі моделювання різних рівнів абстракції процесів та змінено один з елементів MERN стеку технологій на реляційну БД для полегшення масштабування та оптимізації в першу чергу.

Практична цінність результатів полягає у тому, що запропоновані в роботі моделі і методи, дозволяють: накопичувати та використовувати знання експерта для вирішення задач проєктування ІС для вебсередовища; а також удосконалено MERN стек технологій, що полегшує масштабування ІС.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити безпосереднє використання технологій серверної та клієнтскької сторін розробки. В результаті роботи повинен бути розроблений програмний комплекс для забезпечення форуму IT-фахівців.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	12.09.2021-30.09.2021
Збір, дослідження та систематизація інформації щодо проєктування та реалізації інформаційних систем для веб-середовища	01.10.2021-31.10.2021
Розробка автоматизованої системи для вирішення задачі забезпечення форуму IT-фахівців і модифікація стеку MERN – підключенням MySQL СУБД	01.11.2021-16.12.2021

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи очікується позитивним завдяки тому, що дану систему можна інтегрувати в будь-яку

установу, таким чином зменшити затрати на забезпечення підписок на схожі сервіси від комерційних установ.

Соціальний ефект від реалізації результатів роботи очікується позитивним завдяки можливості накопичувати та використовувати знання експерта для вирішення задач проектування ІС для вебсередовища.

Завдання видав _____ *Мороз Б.І.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Грудєв А.М.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2021 р.*

Термін подання кваліфікаційної роботи до ЕК *19.01.2022*

РЕФЕРАТ

Пояснювальна записка: 88 с., 45 рис., 3 дод., 25 джерела.

Об'єкт дослідження: процеси проєктування, розробки і використання вебсайтів та вебзастосунків на базі технологій стеку MERN та програмної платформи Node.js.

Предмет дослідження: ефективне використання моделей, методів та інформаційних технології серверної та клієнтської складових для забезпечення якості вебресурсів при створенні інформаційних систем – вебсайтів.

Мета роботи: підвищення ефективності роботи вебсайту форуму шляхом підбору сукупності технологій, які забезпечили б гнучкість системи з погляду внесення нового функціоналу та легкість підтримки в умовах модифікування вимог до неї в цілому та її окремих характеристик.

Методи дослідження. Для вирішення поставлених задач використані методи: аналізу даних, теорії множин, об'єктно-орієнтованого та функціонального програмуваннь, проектування ІС, паттернів програмування.

Новизна отриманих результатів визначається тим, що розроблено і обґрунтовано модель реалізації вебсайту форуму для ІТ-фахівців, що дозволило ефективно виконувати модифікацію системи в подальшому, на основі моделювання різних рівнів абстракції процесів та змінено один з елементів MERN стеку технологій на реляційну БД для полегшення масштабування та оптимізації запитів.

Практична цінність результатів полягає у тому, що запропоновані в роботі моделі і методи, дозволяють: накопичувати та використовувати знання експерта для вирішення задач проєктування ІС для вебсередовища; а також удосконалено MERN стек технологій, що полегшує масштабування ІС.

Область застосування. Розроблена інформаційна система може застосовуватися для вирішення задач реалізації корпоративного форуму чи спеціалізованого сайту-форуму.

Значення роботи та висновки. Запропонована методика дозволяє підвищети ефективність роботи вебсайту форуму шляхом підбору сукупності технологій, які забезпечують гнучкість та динамічність модифікації системи в цілому, та її окремих характеристик.

Прогнози щодо розвитку досліджень. Удосконалення моделі вебсайту за рахунок врахування взаємного впливу складових елементів модифікованого стеку технологій.

У розділі «Економіка» проведені розрахунки трудомісткості й тривалості розробки програмного забезпечення, витрат на створення ПЗ, а також проведені маркетингові дослідження ринку збуту створеного програмного продукту.

Список ключових слів: веб-застосунок, комп’ютер, вебсайт, фреймворк, додаток, сервер, клієнт, http-запит, http-відповідь, база даних, інформаційна система, форум.

ABSTRACT

Explanatory note: 88 p., 45 figures, 3 adds, 25 sources.

Object of research: processes of designing, developing and using websites and web applications with the use of MERN stack technologies and Node.js software platform.

Subject of research: models, methods and information technologies of server and client sides for ensuring the quality of web resources in terms of creating information systems - websites.

Purpose of Master's thesis: to increase the efficiency of the forum website by selecting a set of technologies that would provide flexibility of the system in terms of introducing new functionality and ease of support within the context of modifying of modifying the system's requirements and its characteristics as well.

Research methods. Methods were used to solve the set tasks: data analysis, set theory, object-oriented and functional programming, information systems design, programming patterns.

Originality of research. The novelty of the obtained results is determined by the fact that a model of the forum website for IT specialists implementation was developed and substantiated, which allowed to add modifications easily, based on modeling different levels of process abstraction and one of the MERN elements was improved to enable easy scaling for highload system.

Practical value of the results is that the proposed models and methods allow: to accumulate and use expert knowledge to solve problems of designing informational systems for the web environment; and the MERN technology stack has been improved to facilitate system's scaling.

Scope. The developed information system can be used to solve problems of corporate forum or specialized site-forum.

The value of the work and conclusions. The improved technique allows to design information systems with a significant reduction in both material and time costs, which is confirmed by the developed software product in this project.

Forecasts for research development. Improving the website model by taking into account the mutual influence of the components of the modified technology stack.

In the Economics section calculations of the complexity and duration of software development, the cost of creating software, as well as marketing research of the market for the created software product are documented.

Keyword list: web application, computer, website, framework, application, server, client, http-request, http-response, database, information system, forum.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІС – інформаційна система;

БД – база даних;

ОС – операційна система;

ПК – персональний комп’ютер;

СУБД – система управління базою даних;

API - Application Programming Interface;

CSS – Cascading Style Sheets;

CMS – Content Management System;

HTML – HyperText Markup Language;

HTTP – Hyper Text Transfer Protocol;

MVT –Model-View-Template;

MVC – Model-View-Controller;

NPM – node package manager;

SPA – single page application;

UI – User Interface;

REST API – Representational State Transfer Application Programming Interface.

ЗМІСТ

РЕФЕРАТ 5
ABSTRACT 6
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 7
ВСТУП 10
РОЗДІЛ 1. АНАЛІЗ ТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ
1.1. Задача забезпечення якості вебсайту 14
1.2. Існуючі підходи до реалізації інформаційних систем вебресурсів 18
1.3. Порівняльний аналіз засобів досягнення гнучкості та масштабованості вебсайту 24
1.4. Висновки 28
РОЗДІЛ 2. ЗБІР, СИСТЕМАТИЗАЦІЯ Й ДОСЛІДЖЕННЯ ІНФОРМАЦІЇ ПРО СУЧАСНІ МОДЕЛІ ІНФОРМАЦІЙНИХ СИСТЕМ НА БАЗІ MERN СУКУПНОСТІ ТЕХНОЛОГІЙ
2.1 Основи побудови ІС вебсайту за мікосервісною архітектурою 30
2.2. MERN стек технологій 32
2.3. Використання Redux для управління даними клієнтської частини 39
2.4. Порівняльне тестування баз даних MySQL та MongoDB 41
2.4.1. Обгрунтування вибіру структури таблиць 41
2.4.2. Опис процесу тестування 42
2.5. Аналіз результатів тестування 44
2.5.1. Результати з індексацією 44
2.5.2 Результати без індексації 47
2.6. Висновки 50
РОЗДІЛ 3. СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ЗАБЕЗПЕЧЕННЯ ЕФЕКТИВНОСТІ РОБОТИ ФОРУМУ ІТ-ФАХІВЦІВ
3.1. Структура інформаційної системи форуму ІТ-фахівців 51
3.2. Структура та опис бази даних 53
3.3. Опис ІС форуму та алгоритмів її функціонування 55
3.4. Висновки 53

РОЗДІЛ 4. ЕКОНОМІЧНИЙ РОЗДІЛ

4.1. Визначення трудомісткості розробки програмного забезпечення..... 67
4.2. Розрахунок витрат на створення програми 70
4.3. Маркетингові дослідження ринку збуту розробленого програмного продукту 71
4.4. Оцінка економічної ефективності впровадження програмного забезпечення 72
4.5. Висновки 72
ВИСНОВКИ..... 73
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 75
Додаток А. Код програми..... 77
Додаток Б. Відгук керівника економічного розділу 87
Додаток В. Перелік файлів на диску 88

ВСТУП

Актуальність дослідження. За існуванням значної кількості ІС з різними властивостями і задачами, які вони вирішують, існує необхідність у зборі та аналізі інформації про наявні сучасні засоби розробки вебзастосунків. Знання про існуючі технології, фреймворки, бібліотеки та можливості їх взаємодії дає змогу правильно підібрати стек для створення конкретних проєктів, що позитивно впливає на побудову злагодженої архітектури ІС. Це є значною перевагою у подальшій роботі над проєктом, бо зменшує вартість розроблюваних систем, їх подальшу підтримку, а також значно зменшує ризик необхідності перебудови системи при масштабуванні.

Всі найпоширеніші операційні системи, браузери, на різних видах пристроїв підтримують скриптову інтерпретовану мову – Javascript. Згідно статистики JS використовується у 95% вебзастосунків. Відмінною властивістю є те, що такі системи можуть працювати без встановлення будь-яких виконуваних файлів на комп’ютер користувача. Тобто, якщо потрібно, щоб створеним застосунком можна було б користуватися без додаткових налаогоджень – то Javascript майже єдиний інструмент для реалізації. Javascript – найпопулярніша мова для розробки клієнтської частини веб ресурсів, бо лише її можуть інтерпритувати браузери без попередньої компіляції.

Створений для більш вузьких та тривіальних задач, Javascript, завдяки еволюції веб напряму в сторону інтерактивних вебзастосунків, набув розгалуджену інфраструктуру фреймворків, бібліотек, протоколів та методів. Також з’явилась платформа для виконання клієнтських і серверних застосунків - Node.js.

В основу даної платформи автор Райан Даль заклав подійно-орієнтований підхід. Зі збільшенням популярності серед інтернет користувачів соціальних мереж та інших інтерактивних вебсайтів – різко зросла популярність цієї технології, як платформи для створення застосунків, що

швидко реагують на дії користувачів: чатів, інструментів спільної роботи, новин, ігор, блогів, комерційних та інших ресурсів. Перелік інформаційних корпорацій, які використовують Node.js у розробці та підтримці своїх систем у мережі Інтернет: Reddit, Wikipedia, Yahoo, PayPal, NASA і т.д. "Навколо Node.js створена сучасна екосистема, і ми отримуємо від неї численні переваги", – висловився інженер eBay Сентіл Падманабхан. Інструменти розробки, бібліотеки тестування і автоматизації та багато інших корисних компонентів розповсюджуються через пакетний менеджер NPM, по цій причині базові знання роботи з Node.js є невід'ємними для будь-якого веброзробника.

У даній кваліфікаційній систематизується інформація з відкритих джерел про проєктування надійних та легкомодифікованих вебсервісів, фреймворки, програмну платформу Node.js та ефективні можливості їх взаємного використання. Практичними прикладами отриманих результатів виступають знання про наявні тенденції розвитку і набори інструментів розробки, а також створена програма – спеціалізована інформаційна система для забезпечення форуму програмістів «PROLOG».

Актуальність роботи та галузь застосування. Автоматизовані інформаційні системи для реалізації вебсайтів мають бути спроєктовані дуже чітко та з поглядом на майбутні можливі модифікації та оновлення. Однак існує багато підходів до реалізації проєктів такого типу – одним з найпоширеніших підходів є використання MERN стеку технологій: NoSQL СУБД MongoDB, Express.js - Node.js фреймворк для реалізації бекенд частини, бібліотека для розробки інтерфейсу користувача React.js; та сама платформа Node.js, яка дозволяє виконувати код, написаний на мові програмування JavaScript, в окремому середовищі, відмінному від браузера. Системи управління реляційними базами даних (RDBMS) можуть використовуватися для ефективного зберігання і оперуванням великим обсягом даних. Однак на продуктивність СУБД може негативно вплинути вимога повної узгодженості транзакцій, коли система гарантує набір властивостей, відомих як ACID

(атомарність, узгодженість, ізоляція і довговічність). На відміну від RDBMS, нереляційні сховища даних (NoSQL) часто призначені для забезпечення тільки так званої можливої узгодженості для подальшого підвищення масштабованості, але в цьому випадку з часом постане питання оптимізації роботи БД, і виникне багато додаткових проблемних моментів, бо на даний момент NoSQL БД не є легкими для оптимізації. Тому в даній роботі вирішено використати RDBMS СУБД MySQL, з налаштуванням під даний стек, що забезпечило б легку масштабованість та оптимізовану й швидку роботу системи.

Мета кваліфікаційної роботи – ефективне користування початківцями можливими інструментами і шляхами розробки вебзастосунків за допомогою інформування про новітні тенденції, фреймворки та методи їх використання й підхід до проєктування.

Завдання дослідження – для досягнення поставленої мети в роботі сформульовані і вирішені такі завдання:

1. Проаналізувати новітні тенденції розвитку розробки вебзастосунків та сайтів;

2. Зібрати, систематизувати та дослідити інформацію про фреймворки й програмну платформу Node.js та стек MERN;

3. Розробити спеціалізовану інформаційну систему для забезпечення форуму IT-спеціалістів «PROLOG» з використанням модифікованого стеку MERN для демонстрації можливостей правильно підібраного набору інструментів.

Об’єкт дослідження – процеси проєктування, розробки і використання вебсайтів та вебзастосунків на базі технологій стеку MERN та програмної платформи Node.js.

Предмет дослідження – ефективне використання моделей, методів та інформаційних технології серверної та клієнтської складових для забезпечення якості вебресурсів при створенні інформаційних систем – вебсайтів..

Методи дослідження – у процесі дослідження були використані теоретичні та емпіричні наукові методи: опису, аналізу, абстрагування, узагальнення на основі даних, класифікації і пояснення.

Наукова новизна отриманих результатів визначається тим, що розроблено і обґрунтовано модель реалізації вебсайту форуму для IT-фахівців, що дозволило ефективно виконувати модифікацію системи в подальшому, на основі моделювання різних рівнів абстракції процесів та змінено один з елементів MERN стеку технологій на реляційну БД для полегшення масштабування та оптимізації запитів.

Практична цінність – результати проведеного дослідження і розроблений застосунок можуть бути використані розробниками, аспірантами та студентами які цікавляться або спеціалізуються у сфері програмування та тестування клієнтської та/або серверної частин вебзастосунків, серверів чи вебсайтів з використанням мови програмування Javascript на платформі Node.js та MERN стеку технологій.

Особистий внесок автора:

1. Наукові результати роботи отримані автором самостійно;

2. Вибір методів досліджень і технологій реалізації;

3.Розробка теоретичної частини роботи, що збирає, систематизує і досліджує інформацію про існуючі тенденції розробки ПЗ для веб напряму;

4. Розробка веб застосунку форуму IT-фахівців «PROLOG» з використанням стеку технологій розробки MERN на основі програмної платформи Node.js;

5. Оцінка отриманих результатів.

Структура та обсяг роботи. Робота складається зі вступу, чотирьох розділів та висновків. Містить 88 сторінки, у тому числі 60 сторінок тексту основної частини з 45 рисунками, список використаних джерел із 25 пунктами на 2 сторінках, 3 додатки на 11 сторінках

РОЗДІЛ 1
АНАЛІЗ ТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Задача забезпечення якості вебсайту

Інформація, яка зберігається в мережі Інтернет, розташовується на комп'ютерах або так званих вебсерверах, на яких, в свою чергу, встановлено спеціальне програмне забезпечення, що дозволяє користувачам знаходити потрібну інформацію. Більша частина такої інформації впорядкована у вигляді вебсайтів, кожен з яких має своє ім'я або адресу в Інтернеті. Для того, щоб переглядати вебсайти на комп'ютері користувача має бути встановлено спеціальне програмне забезпечення – браузер, який буде виступати у ролі клієнта. Виходячи з того, яка адреса (ім'я) вебсайту задається для пошуку клієнтом, браузер відображає відповідну інформацію у своєму робочому вікні.

Вебсайти являють собою механізм спілкування між власниками сайту та його користувачами або - між самими користувачами. Власники сайтів зазвичай формулюють завдання вебсайту та визначають основні правила взаємодії в його межах, в той час як користувачі - це ті суб’єкти, які відвідують сайт та намагаються користуватися представленим на ньому вмістом або можливостями які він надає. Напрвлення зв'язку між власником сайту і його відвідувачем може змінюватися. Але в більшості випадків власники сайтів надають користувачам певну інформацію, що визначає односторонню взаємодію. В сьогоднішніх реаліях дані стають з кожним днем більш цінною категорією, тому цей принцип відходить на задній план і сайти збирають якомога більше інформації про своїх користувачів.

Вебсайти можна класифікувати за декількома категоріями:

1. інформаційні: надають інформацію по конкретній темі або про деяку організацію, тобто забезпечують презентаційну функцію. Такі сайти найпоширеніші в мережі Інтернет; в процесі свого існування вони

часто переймають деякі риси інших категорій сайтів – надають більш широкі можливості для взаємодії з користувачами;

2. операційні: Закривають певну потребу в реалізації виконання деякої операції або задачі. Під цю категорію відносяться сайти, зайняті в електронній комерції;
3. вебсайти спільнот: Представляють інформацію або функціонал, пов'язані із здійсненням певних операцій, але акцент робиться на взаємодії між користувачами. Такі сайти мають фокусуються на конкретній темі, деякій події, особі тощо. Їх мета забезпечити взаємодію між користувачами зі схожими інтересами;
4. розважальні: Створюються для проведення ігор або якоїсь цікавої взаємодії, для цього виду сайтів можуть бути використані елементи характерні для операційного та інформаційного типів;
5. інші сайти: Здебільшого ця категорія представлена художніми або експериментальними сайтами, особистими блогами, а також сайтами, які можуть не слідувати загальноприйнятим угодам в Інтернеті та не мати чітко визначеного економічного призначення.

Крім цього, можна виділити сайти організацій, які підтримують або в якомусь сенсі платять за існування свого сайту. В рамках цього типу класифікації можна виділити такі основні групи:

1. Корпоративні: створюються та підтримуються певною організацією або особою для отримання комерційної вигоди - безпосередньо за допомогою електронної комерції, або побічно через стимулювання придбання товарів або послуг поза мережею Інтернет;
2. Урядові: Вищим органом по відношенню до такого сайту в кінцевому підсумку є урядова організація. Головною метою таких сайтів є задоволення будь-якої суспільної або правової потреби;
3. Освітні: Такі сайти знаходяться під контролем якогось навчального закладу (можуть мати відношення до урядових органів); такі сайти

забезпечують освітні або дослідницькі задачі;

4. Філантропічні: Головна мета таких сайтів є просування цілей благодійної діяльності некомерційної організації або приватної особи;
5. Персональні: Такі сайти ведуться деякою особою або особами, як правило для вираження творчої енергії або з метою самовираження особистості.

Класифікація може виявитися складним завданням. Наприклад, освітні сайти насправді можуть потрапляти в категорію урядових. Деякі сайти з категорії персональних можуть, з високою ймовірність, належати до групи філантропічних або комерційних - в залежності від причини, з якої був створений сайт. В даній кваліфікаційній роботі розглядається створення операційного повнофункціонального вебсайту форуму для IT-фахівців «PROLOG».

Важливим аспектом в процесі розробки сайту є етап створення дизайну – це перше, на що звертає увагу користувач при заходженні на сайт за його інтернет-адресою. При успішній реалізації дизайну користувачу буде зручно використовувати сайт та час знаходження на ньому буде більший, аніж у випадку погано спланованого дизайну. Тому на етапі проєктування вебсайту слід приділити увагу на цільову аудиторію та її інтереси, потреби, вподобання тощо. Ці задачі розглядають такі дисципліни, як UX та UI. UX або User Experience, тобто досвід користувача – це дисципліна, яка вивчає досвід взаємодії користувача з сайтом, та реакцію й сприяняття, які виникають в результаті цієї взаємодії.

UI або User Interface, тобто інтерфейс користувача – це те, як виглядає інтерфейс та які фізичні характеристики він має. UI визначає кольорову палітру вебсайту, зручність розташування структурних елементів сайту на вебсторінці, на скільки вдало підібрані текстові шрифти та розміри текстів.

Будь-який вебсайт складається з пов'язаних між собою вебсторінок.

Кожна сторінка вебсайту включає в себе текстовий файл з розширенням html, в якому міститься текстова інформація та спеціальні команди на мові HTML, завдання яких визначити вид інформації для відображення в браузері. До складу вебсторінки не входить графічна, аудіо- та відео-інформація, - вона являє собою окремі файли, на які вказуються посилання.

Для відображення інформації про певну предметну галузь використовують бази даних. Предметна галузь - це частина реального світу, що розглядається в межах галузі проєктування, відображена у вигляді бази даних. Для опису предметної галузі у вигляді бази даних необхідно визначити яка саме інформація буде зберігатися і оброблятися.

Всі вебсайти прийнято розділяти на дві групи: статичні і динамічні.

Статичний сайт - це набір статичних HTML-сторінок, які пов'язані між собою посиланнями. Статичні HTML-сторінки створюються вручну, зберігаються і завантажуються на сервер, після чого при кожному зверненні до сайту представляються користувачеві в незмінному вигляді. Для того щоб внести зміни інформації на таких сторінках, необхідно вручну редагувати програмний код сторінки.

Статичні сайти мають як свої переваги, так і недоліки. До переваг статичних сайтів відносять такі:

- створюють мінімальне навантаження на сервер;
- швидко завантажуються;
- розробка не потребує значних коштів;
- простота перенесення на новий хостинг.

Серед недоліків статичних сайтів особливо виділяється складність оновлення сайту, внесення будь-яких змін. Управління сайтом неможливе без знань і умінь в галузі вебпрограмування - це може спричинити за собою додаткові витрати при необхідності додавання нових матеріалів на сайт, нових розділів або категорій тощо.

Динамічний сайт - це сайт, в якому передбачена можливість редагувати

вміст сторінок сайту, без звернення до процесу програмування. Відображення сторінок таких вебсайтів засноване на шаблонній структурі, в основі якої - динамічне інформаційне наповнення. У більшості випадків інформаційне наповнення сторінок забезпечується ресурами з бази даних. Під час запиту сторінки користувачем відбувається вибірка потрібної інформації з бази даних, яка вставляється в шаблон, утворюючи при цьому нову вебсторінку, і пересилається вебсервером в призначений для користувача вебклієнт (браузер), який і відображає її належним чином. Можливість вносити зміни в усі сторінки надається тільки певній групі користувачів, таким як, адміністратори або привілейовані користувачі. В окремих випадках вносити зміни в контент вебсторінки допускаються й анонімним користувачам (наприклад, на форумах - додавання коментарів).

Отже, можна виділити в першу чергу необхідніть відокремлення при проєктуванні типу інформаційної системи вебсайт – статичний чи динамічний характер вона має. З цього буде формуватися початкові вимоги до технологій та патернів проєктування системи в цілому.

1.2. Існуючі підходи до реалізації інформаційних систем вебресурсів

Веброзробка – один з найрозвиненіших напрямів програмування. Для даного типу розробки є величезний набір інструментів, але базовим інструментом є мова Javascript. Знання Javascript вважається ключовою навичкою в контексті веброзробки. Скрипти Javascript – це програми, які працюють з об'єктами HTML-документа, за допомогою сценаріїв Javascript можливо представити елементи HTML як об'єкти, при клієнтському програмуванні.

Вікно браузера відповідає об'єкту вікна, а документ HTML, завантажений у вікно, відповідає об'єкту документа. Об'єкт документа є частиною вікна. Елементи документа HTML відповідають об'єктам, які є

частиною документа. Весь набір має ієрархічну структуру, яка називається об'єктною моделлю документа, або DOM. Об'єкт являє собою контейнер для зберігання інформації. Він характеризується властивостями, методами і подіями, на які він може реагувати.

Мова програмування Javascript призначена для створення інтерактивних HTML-документів. Це мова для розробки вбудованих додатків, які працюють як на стороні клієнта, так і на стороні сервера. Клієнтські програми запускаються браузером, переглядають вебдокументи на машині користувача, серверні застосунки працюють на сервері. Обидва типи застосунків використовують загальний мовний компонент який називається ядром. Він включає визначення стандартних об'єктів і структур (змінних, функцій, основних об'єктів і інструментів), а також відповідні компоненти мовних надбудов, які містять всі типи додатків для визначення об'єктів.

Команди сценаріїв Javascript вбудовуються в HTML-сторінки і інтерпретуються в машинний код вбудованим у браузер інтерпретатором (наприклад, V8 у Chrome), що виконується компьютером при завантаженні сторінок чи певних діях з боку користувача або сервера.

Призначенням клієнтських скриптів Javascript є:

- Швидка валідація заповнених користувачем полів HTML-форм перед їх відправкою на сервер. Скрипти Javascript можуть обробляти дані, введені користувачами в поля форми, а також події, які відбуваються під час маніпуляції користувачем з мишею, копіювати інші HTML-сторінки у вікно браузера або скасовувати вже завантажені сторінки;
- Створення динамічних HTML-сторінок разом з каскадними таблицями стилів і об'єктною моделлю документа;
- Взаємодія з користувачем при вирішенні "локальних" завдань. Зокрема, скрипти Javascript широко використовуються для створення різних візуальних ефектів. Наприклад, зміна зовнішнього вигляду елементів управління, на які наведений курсор миші, анімація графічних зображень,

створення звукових ефектів і т.д. Механізм локальних файлів дозволяє сценаріям Javascript зберегти локальну інформацію, введену користувачем на комп'ютері.

Асинхронність – одна з особливостей мови Javascript. Необхідно зрозуміти плюси і мінуси цієї технології. У синхронних програмах, якщо є два рядки коду (Рядок 2 після Рядка 1), то Рядок 2 не може працювати до тих пір, поки Рядок 1 не завершить своє виконання. В асинхронних програмах, моливо мати два рядки коду (Рядок 2 після Рядка 1), де Рядок 1 містить команди, які будуть виконуватися у майбутньому, а Рядок 2 працює до завершення команди в Рядку 1. Принцип синхронного коду можна порівняти з роботою залізничної каси – ви знаходитесь в черзі людей, які чекають, щоб купити собі квитки на потяг. Ви стоїте в черзі і не можете купити квиток на потяг, поки ваша черга не настане. Так само і люди за вами не зможуть купити квитки, поки ви не закінчили. А принцип асинхронного коду можна порівняти з обслуговуванням у ресторані: люди можуть замовляти їжу у той же час як і ви замовляєте щось собі. Вам не потрібно чекати, поки люди до вас завершать своє замовлення, щоб зробити те, щоб вам потрібно, якщо ваше замовлення виконується швидше, і навпаки. Всі отримають свої замовлення по мірі їх виконання.

Якщо розглянути принципи потоковості – можна зробити висновок, що асинхронний – не означає одночасний чи багатопотоковий. Javascript має асинхронний характер, але він однопотоковий. Це наче в кафе з одним працівником, що робить все – готує і подає. Але, якщо цей працівник працює швидко і може переключатися між завданнями ефективно, то може здаватися, що у кафе працюють кілька робітників.

Щоб створити синхронну послідовність операцій – Javascript код потрібно структурувати по іншому і найпростіший спосіб це зробити – використати callback функції або функції зворотнього виклику. В Javascript вони використовуються дуже активно. Майже всі вебзастосунки використовують зворотній виклик для обробки подій, як то: таймери, події

DOM, а також для асинхронних запитів (AJAX) та ін. В розробці існує ряд завдань, де кожен крок залежить від результатів попереднього кроку. Не важко отримати таке використовуючи синхронні мови програмування. При спробі зробити це в асинхронному коді, дуже легко отримати так зване callback-пекло – розповсюджену проблему, коли кількість вкладених одна в одну функцій зворотнього виклику стає надмірно великою. Сервери на платформі Node.js і код з великою кількістю AJAX-запитів часто страждають від цієї проблеми. Такий код важко читати, а спроба реорганізувати його, коли потрібно внести зміни, ускладнює та сповільнює виконання роботи. Глибоко "заплутані" функції зворотнього виклику в коді – це знак, що коду необхідна реорганізація. Існує кілька різних стратегій для цього. Promises (укр. “обіцянка”) – популярний спосіб позбавлення від надмірної кількості зворотніх викликів. Ідея полягає в тому, що замість того, щоб використовувати функції, які приймають вхідні дані і зворотні виклики, створити функцію, що повертає об'єкт promise, тобто, об'єкт, що представляє значення, яке буде існувати в майбутньому.

Promises стали стандартом в ECMAScript 6. Варто відзначити відкритість мови Javascript. Вона має велику кількість бібліотек і фреймворків, що дозволяє легко вирішувати завдання будь-якого типу. Можна писати комп’ютерні і мобільні додатки з використанням платформи Electron та NativeScript або React Native відповідно.

Для початку роботи потрібно знати базові концепції Javascript та веброзробки і програмування в цілому:

- Об'єктно-орієнтована JS – конструктори, об’єкти і спадкування.
- Функціонал JS – функції вищого порядку, схема, рекурсія.
- Основи HTML, CSS, JQuery.
- Git – система контролю версій. Необхідний інструмент для розробників для управління файлами на серверах, співпраці з командою, поширення коду і уникнення розбіжностей при редагуванні.

Бази даних, схеми, моделі і ORM – одні з найважливіших елементів

веброзробки. Якщо в застосунку необхідно завантажувати або зберігати будь-які дані, які б не втрачалися при оновленні сторінки – необхідно використовувати базу даних. Потрібно розрізняти реляційні і нереляційні бази даних і розуміти типи з'єднань. Також необхідно знати SQL і познайомитись із різними системами управління базами даних. Уміння працювати з ORM теж важливе. Також корисно вивчати популярні бібліотеки (наприклад, Bootstrap) і препроцесори CSS, як Sass, що допоможе писати на CSS швидше.

Мова розмітки HTML використовується для створення представлень вебсторінок, використовуючи спеціальні теги, які являють собою струтурні елементи сторінки, подальша стилізація вебсторінки ведеться за допомогою CSS. Мова CSS визначає селектори тегів, які дозволяють стилізувати будь-який елемент вебсторінки, наприклад, розмір шрифтів, колір тексту, відступи зовнішні та внутрішні. Також, мова CSS дає функціонал для створення адаптивності вебсайту на різних пристроях з відмінним розміром екрану. Бібліотека React.js надає унікальний синтаксис JSX, який полегкшує розробку компонентів, об’єднуючи в одне ціле JavaScript та технології HTML і CSS.

HTML і CSS є основою будь-якого веброзробника. Іноді веброзробникам не обов'язково знати їх досконало, але вони повинні їх розуміти. Щоб спростити роботу з HTML, ми можемо обрати один з популярних шаблонів, наприклад pug, JQuery для маніпуляції з DOM. Створюючи зовнішній вигляд сторінки за допомогою HTML і CSS, можливе використання трансляторів подій і бібліотеку JQuery для управління DOM. Оскільки різні браузери трохи різняться за поведінкою Javascript і реалізацією об'єктної моделі документа, потрібні налаштування до кожного браузера, якщо наші вебзастосунки націлені на нього. Оскільки Javascript є нетипізованим інтерпретатором і може працювати в різних середовищах, кожна з яких має свої власні міркування сумісності. Тому програміст повинен бути дуже обережним, щоб переконатися, що його код працює належним чином в найрізноманітніших можливих конфігураціях. Інтерпретатор аналізує кожен блок скрипта окремо.

На відміну від більшості мов, Javascript не керується тільки концепціями введення (input) і виведення (output). Він спроєктований таким чином, щоб запускатися як мова сценаріїв, вбудованих в середовище виконання.

Найпопулярніше середовище виконання це браузер, проте інтерпретатор Javascript присутній в програмній платформі Node.js, яка дозволяє виконувати його поза межами браузера.

Як було викладено вище існують два типи інформаційних систем типу вебсайт: статичні та динамічні. На відміну від статичних, динамічні сайти більш гнучкі в управлінні. Такі сайти можна розробляти «з чистого аркуша», вручну створюючи всі необхідні програмні коди, скрипти і т.д. Однак набагато частіше для створення динамічних сайтів використовуються спеціальні системи управління контентом – CMS або фреймворки для розробки вебдодатків, наприклад Express.js для Node.js. CMS дозволяють використовувати вже готові програмні модулі і компоненти, без необхідності кожного разу створювати їх заново. На основі однієї CMS можна створити будь-яку кількість динамічних сайтів. У динамічних сайтах реалізовано поділ змісту і оформлення вебсторінок, що, в свою чергу, дозволяє оперативно змінювати інформацію на сайтах без необхідності змінювати програмні коди сторінок. Це є однією з найголовніших переваг динамічних сайтів. Крім перерахованих переваг, динамічні сайти мають і ряд недоліків. У порівнянні зі статичними сайтами, динамічні потребують більої кількості місця під збереження даних для своєї ефективної роботи та дають велике навантаження на сервер - отже, вони більш вимогливі до хостингу й ресурсів сервера.

Альтернативою використання CMS є розробка сайту за допомогою фреймовркв для реалізації серверної логіки та клієнтської частини. Фрейморк – це набір бібліотек, налаштування над конкретною мовою програмування, що забезпечує ефективне виконання певної задачі, у випадку з Node.js – розробка вебдодатків та вебсайтів.

1.3. Порівняльний аналіз засобів досягнення гнучкості та масштабованості вебсайту

На даний момент існує багато рішень підходів до проєктування ІС вебсайту, одними з найефективніших інструментів для створення вебсайтів є технології стеку MERN, який включає Node.js та Express.js Framework розроблений на мові JS. Він сумісний з клієнтськими бібліотеками та фреймворками для проєктування клієнтської частини (наприклад ReactJS), окрім цього надає можливість аідключення зовнішніх шаблонізаторів, які забезпечують динамічність вебсторінок. При такому підході до розробки є змога створювати високоефективні вебдодатки, односторінкові сайти, які будуть працювати дуже швидко, навіть за умови високого навантаження та значним об’ємом даних, з яким ведеться робота. Також оновлення таких вебдодатків й процес логування можна контролювати та налаштовувати під свої потреби більш повно, аніж у випадку CMS.

Express.js доволі популярний - використовується на багатьох сайтах, в тому числі таких, як Trello, NetFlix, Instagram і багатьох інших; є безкоштовним - розвивається як «open source» проєкт, його вихідний код відкритий, його можна знайти репозиторії на Github.

Даний фреймворк може бути використаний з рядом проєктних патернів. Наприклад, при монолітній архітектурі можна використати підхід Model-View-Template або скорочено MVT [10, 11], який за фактом є модифікацією паттерна MVC (Model-View-Controller).

Схематично архітектура MVT виглядає наступним чином (рис. 1.1):

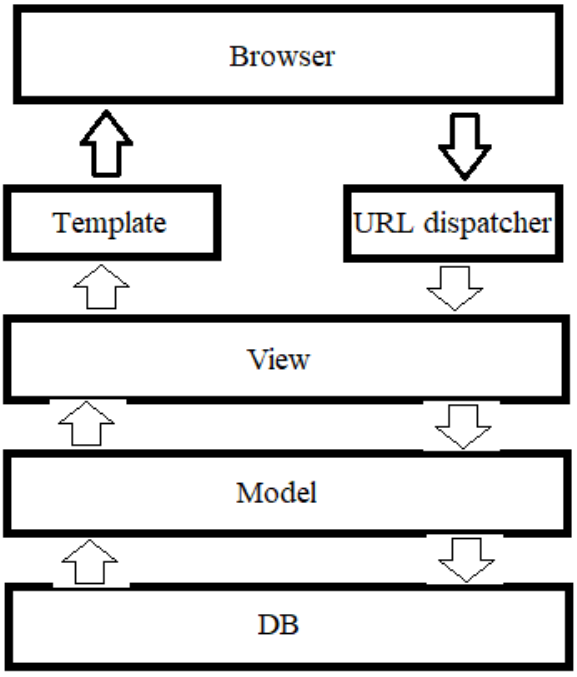

Рис.1.1. Схема архітектури MVT

Основні елементи патерну:

1. URL dispatcher (роутер): при отриманні запиту на підставі деякої URL - адреси визначає, який View повинен обробляти даний запит.

2. view (уявлення): отримує запит, обробляє його і відправляє користувачеві деяку відповідь. Якщо для обробки запиту необхідно звернення до моделі і бази даних, то View взаємодіє з ними. Для створення відповіді може застосовувати Template. В архітектурі MVC цей компонент є контролером.

3. model (модель): описує дані, які використовуються в додатку. Окремі класи, як правило, відповідають таблицям в базі даних.

4. template (шаблон): представляє логіку View у вигляді згенерованої розмітки HTML. У MVC цьому компоненту відповідає View. Функціональність даного компоненту можна розширити, створивши клієнтську логіку на JavaScript або на фреймворці чи бібліотеці для цієї мови.

Коли до вебдодатка приходить запит, то URL dispatcher визначає, з яким ресурсом зіставляється даний запит і передає цей запит обраному ресурсу.

Ресурс фактично являє функцію або View, який отримує запит і певним чином обробляє його. В процесі обробки View може звертатися до моделей та бази даних, отримувати з неї дані, або, навпаки, додавати в неї нові дані. Результат обробки запиту відправляється у відповідь, і користувач бачить його у вікні браузера. Як правило, результат обробки запиту представляє згенерований html-код, для генерації якого застосовуються шаблони (Template).

Значним недоліком такого підходу до розробки є збільшення кодової бази інформаціної системи та ускладнення її підтримки з часом. Такі монолітні системи складно оновлювати та переводити на нові версії зілежних бібліотек, платформ та інших технологій, які використовуєються проєктом. При потраплянні на розробку нового фахівця треба витрачати більше часу для його ознайомлення з процесами системи. У великому об’ємі коду легше створити помилки та складніше їх знайти. Недолік стає дуже явним з ростом функціональності системи – а це є невід’ємною частиною розвитку програмного застосунку. Також тести, що покривають кодову базу в більшості випадків будуть проходити значний проміжок часу, з цим можливо боротись, але все ж таки складність ІС буде лише зростати.

Ще одним підходом, який набрав популярність в теперішній час є створення не монолітного застосунку, який являє собою об’єднання бекенд та фронтенд логіки в одне ціле, а навпаки розгалудження глобального застосунку на так звані мікросервіси. Мікросервіси працюють через API, який надає сервер – по зверненню до адресів API клієнт може отримувати необхідні дані і на своїй стороні виконувати компонування інтерфейсу користувача на основі цих даних.

При такій архітектурі вся кодова база розгалужена по-перше за логікою на сервений та клієнтський застосунки. По-друге, за контекстом, яким оперує дана підсистема – мікросервіс. Наприклад, в ІС вебсайт форуму мікросервісами можуть бути – система авторизації, роботи з обговореннями, комунікації з користувачами і т.д. При такому підході дуже легко вносити зміни в ІС та модифікувати її під негайні потреби. Тобто легко притримуватись принципів

предметно-орієнтованого проєктування.

CMS краще застосовувати у наступних випадках:

1. шаблонне рішення, яке покривається можливостями CMS;
2. швидке, тимчасове або недовгострокового рішення;
3. для невеликих бюджетів на розробку вебдодатк;
4. у програмітса незначні знання у галузі розробки вебдодатків.

Фреймворк краще застосовувати у наступних випадках:

1. нетиповий проєкт;
2. проєкт, що активно змінюється або підлаштовується під тренди;
3. у програміста, який буде займатися розробкою досить досвіду, щоб створити якісний вебдодаток на обраному фрейморку для серверної логіки.

Для розробки мікросервісів ідеальним варіантом є MERN стек програмування, тому що наразі це є однією з найпопулярніших комбінацій для сучасних програмних комплексів, адже React.js дозволяє створити функціональну, підтримувану та розширювану FrontEnd частину системи дуже швидко; в той час, як Node.js дозволяє створити високоефективну та легку в обслуговуванні й розширенні серверну частину. Також важливим фактором такого вибору стала наявність великої JavaScript спільноти та кількості готових додатків, бібліотек, статей, доповідей, що допомагають створити надійну платформу будь-якої складності.

Отже, всі із зазначених підходів мають право на існування і вибір між ними слід робити, виходячи з можливостей на розробку конкретного вебдодатку та задач, функціональність для рішення котрих він має реалізовувати. Але перш за все виділяється підхід з використанням мікросервісної архітекрутри.

1.4. Висновки

Виникнення платформи Node.js якісним чином вплинуло на варітивність підходів до розробки інформаційних систем вебсайтів. Завдяці цій технології програміст має змогу розробляти обидві складові системи вебресурсу на одній мові Javascript, що дає змогу швидше «зануритись» у логіку аніж у конструкції різних мов. Архітектурний підхід для системи, яку буде легко підтримувати та модифікувати – мікросервіс. Такий підхід складвється з розробки двох застосунків – серверного та клієнтського. У бекенді виникають наступні задачі: створення API для фронтенду, розробка бізнес-логіки, інтеграція коду зі сторонніми API, робота з базою даних, розгортання продукту в мережі Інтернет чи іншому середовищі (англ. deployment), можлива необхідність написання автоматичних тестів, робота з інструментами AWS, Azure, Google тощо.

Об’єднання розробки серверної та клієнтської частин на основі мови програмування Javascript та використанням платформи Node.js, дозволяє скоротити витрати на вивчення мов програмування. Але постає проблема вибору технологій і фреймворків для застосування в межах платформи.

Екосистема фреймворків має широкий вибір на готові реалізації багатьох функціональних можливостей, які можуть знадобитись при розробці високоефективних вебресурсів. Розробникам при роботі над типовими завданнями не потрібно щось вигадувати, адже вони можуть скористатися вже створеною спільною реалізацією. Це не тільки скорочує витрати часу і грошей, але і дозволяє досягти більш високої стабільності рішення, адже компонент, що використовується і допрацьовується тисячами інших розробників зазвичай більш якісно реалізований і краще протестований на всіляких сценаріях, ніж рішення, яке може в адекватні терміни розробити один розробник або навіть невелика команда. Можлива відносно проста реалізація будь-якої бізнес логіки, а не тільки тої, що була закладена в систему спочатку. На сьогоднішній день, зазвичай, розробка без використання платформи ведеться в двох випадках –

проєкт простий і не вимагає подальшого розвитку або дуже навантажений і вимагає глибокої оптимізації (наприклад вебсервіси з десятками тисяч звернень в секунду). В інших випадках розробка на основі програмної платформи вважається швидшою і якіснішою. Проєкти на базі фреймворків легше масштабуються та модернізуються. З точки зору бізнесу – розробка за допомогою фреймворка часто дає економічно більш ефективний і більш якісний результат, ніж написання проєкту чистою мовою програмування без використання будь-яких платформ.

У ході виконання кваліфікаційної роботи потрібно відповісти на головне питання: чи є MySQL швидшим та краще оптимізованим, ніж MongoDB?

Для того, щоб відповісти на це питання, буде використаний кількісний підхід до дослідження. Кількісне дослідження використовує цифри та вимірювання, щоб зв’язати емпіричні спостереження з потенційною неупередженою відповіддю. Для збору цих цифр буде запропонована відтворювана методологія, щоб кожен бажаючий міг перевірити правильність роботи. Методологія буде реалізована у вигляді інструменту для тестування.

РОЗДІЛ 2
ЗБІР, СИСТЕМАТИЗАЦІЯ Й ДОСЛІДЖЕННЯ ІНФОРМАЦІЇ ПРО СУЧАСНІ МОДЕЛІ ІНФОРМАЦІЙНИХ СИСТЕМ НА БАЗІ MERN СУКУПНОСТІ ТЕХНОЛОГІЙ

2.1. Основи побудови ІС вебсайту за мікосервісною архітектурою

Стрімкий розвиток та розповсюдження мережевих хмарних сервісів на початок 2010-х років призвело до розчарування в класичному, так званому монолітному варіанті архітектури додатків. Через складність окремих модулів, що часто являють собою цілі програмні системи, а також через необхідність забезпечувати сумісність між ними за допомогою стандартних протоколів, внесення будь-яких змін та доповнень стало нетривіальним завданням, що забирає надто багато часу.

Як відповідь на цей виклик була запропонована архітектура мікросервісів як розподілена система найпростіших модулів, що легко замінюються, що виконують по можливості єдину елементарну функцію. При цьому мікросервісна система має симетричну, однорангову, а не ієрархічну організацію, що знімає необхідність складної організації взаємозв'язків. Сервіси зв'язуються між собою та клієнтами з використанням легких протоколів, наприклад, через HTTP або текстовими повідомленнями. В результаті створюється система, проста в розгортанні та модернізації з функціями автоматичної розробки та оновлення.

На 2021 рік мікросервісна архітектура вважається найбільш природним підходом для розробки хмарних додатків. Такі складаються з безлічі слабко пов'язаних і дрібних модулів, що розробляються незалежно, — сервісів. Для цих сервісів, як правило, характерні три властивості:

- мають власний стек, що включає базу та модель даних;
- вони взаємодіють один з одним за допомогою поєднання REST API,

потоків подій та брокера повідомлень;

- модулі застосунку підбираються з конкретних потреб бізнесу.

З точки зору бізнесу та суто організаційних завдань, переваги мікросервісів зводяться до трьох основних:

- легкість поновлення коду;
- різні команди можуть використовувати різні стеки для різних модулів;
- компоненти можуть масштабуватися незалежно один від одного, що знижує витрати та вартість масштабування всього додатку в цілому у випадках, якщо вузьким місцем виступає лише якась одна функція.

Щоб уточнити відмінності мікросервісів від інших архітектур, їх найчастіше порівнюють із монолітною архітектурою. Мікросервісний додаток складається з безлічі дрібних незалежних і слабко пов'язаних між собою сервісів, у той час як у моноліті всі його компоненти тісно взаємопов'язані та працюють як єдиний сервіс. Окрім іншого, це означає, що якщо якийсь один процес у додатку з монолітною архітектурою стає більш затребуваним, доводиться масштабувати весь додаток загалом. Збій у якомусь одному процесі може поставити під загрозу всю систему. Зрештою, така складність обмежує можливості модернізації та ускладнює впровадження нових ідей.

В інформаційній системі, яка розглядається в кваліфікаційній роботі, ідея мікросервісів реалізується шляхом розробки так званого мікрофронтенду – частина системи, яка буде відповідати саме за функціонал взаємодії з темами та питаннями, які створюються на форумі. В майбутньому система може бути вдосконалена завдяки додаванню нових функцій в існуючий мікрофронтенд, чи інтеграції з новими системами. Зі сторони бекенд логіки – розроблено API, для функціоналу мікрофронтенда по взаємодії з темами та питаннями форуму. Тобто, є два Node.js застосунки – один відповідає за серверну логіку, інший за клієнтську частину, але відносяться вони до єдиного мікрофронтенду. Таким чином майбутнє вдосконалення та мсаштабування системи буде проходити досить легко, завдяки розподіленню процесів. Наприклад, створений бекенд

застосунок можливо в майбутньому зробити глобальним для всіх мікрофронтендів системи, та інтегрувати додатковий застосунок, який використовував би GraphQL технологію для зручного опису структур даних, які повертає сервер та відходу від REST API методів.

2.2. MERN стек технологій

MERN означає MongoDB, Express, React, Node, після чотирьох ключових технологій, які складають стек:

- MongoDB – нереляційна база даних;
- Express(.js) – веб-фреймворк Node.js;
- React(.js) - клієнтська платформа JavaScript;
- Node(.js) - головний веб-сервер JavaScript.

Express і Node складають середній (додатний) рівень. Express.js — це серверhbq веб-фреймворк, а Node.js — популярна й потужна серверна платформа JavaScript. Незалежно від того, який варіант ви виберете, є ідеальним підходом до роботи з JavaScript та JSON до кінця.

Архітектура MERN дозволяє легко побудувати 3-рівневу архітектуру (фронтенд, бекенд, база даних) повністю за допомогою JavaScript і JSON.

Верхній рівень стеку MERN – це React.js, декларативна JavaScript бібліотека для створення динамічних програм на стороні клієнта в HTML. React дозволяє створювати складні інтерфейси за допомогою простих компонентів, підключати їх до даних на вашому серверному сервері та відображати їх як HTML.

Сильна сторона React полягає в тому, що івн дає змогу працювати з інтерфейсами, що керуються станом даних, з мінімальним кодом і мінімальними проблемами, і він має всі переваги, сучасного веб-фреймворку: чудова підтримка форм, обробки помилок, подій, списків тощо. .

Наступний рівень нижче — це серверний фреймворк Express.js, який працює всередині сервера Node.js. Express.js має потужні моделі для маршрутизації URL-адрес (узгодження вхідної URL-адреси з функцією сервера) та обробки запитів і відповідей HTTP.

Здійснюючи запити XML HTTP (XHR) або GET або POST із інтерфейсу React.js, ви можете підключитися до функцій Express.js, які забезпечують роботу програми. Ці функції, у свою чергу, використовують драйвери БД з Node.js, або через зворотні виклики для використання промысів з JS, для доступу та оновлення даних убазі даних.

Express.js - вільний фреймворк для вебдодатків на мові JavaScript, що використовує ряд шаблонів проєктування MVC, REST API [10, 11, 14]. Даний фреймворк написаний на мові програмування JavaScript, тому його структура відповідає особливостям мови. Він підтримує ряд патернів, наприклад, патерн який найчастіше використовується – REST API. Ця архітектура дозволяє розробнику працювати з візуальним представленням і бізнес-логікою додатка окремо.

Документація Express.js визначає модель (model) як «джерело інформації про дані, в яких містяться ключові поля і поведінка даних» [10, 11, 14]. Зазвичай одна модель вказує на одну таблицю в базі даних. Node.js підтримує бази даних PostgreSQL, MySQL, і багато інших. Моделі містять інформацію про дані. Ці дані представлені атрибутами або полями. Оскільки модель являє собою простий об’єкт, вона нічого не знає про інші рівні абстракції сервісу. Взаємодія між рівнями відбувається через API. Модель відповідає за бізнес-логіку, методи, властивості і інші елементи, пов'язані з маніпуляцією даними. Також моделі дозволяють розробникам створювати, читати, оновлювати та видаляти об'єкти в базі даних. Контролер вирішує три завдання: приймає HTTP-запити, реалізує бізнес-логіку, відправляє HTTP-відповідь на запити. Тобто контролер отримує дані від моделі і надає фронтенд частині доступ до цих даних або попередньо обробляє дані і потім надає до них

доступ [11, 14]. Для Express.js також є реалізований потужний шаблонізатор Pug і власна мова розмітки. Шаблони являють собою файли з HTML-кодом, за допомогою якого відображаються дані. Вміст файлів може бути статичним або динамічним. Шаблони не містять бізнес-логіки. Тому вони тільки відображають дані.

Якщо програма зберігає будь-які дані, то знадобиться база даних ось тут і підходить MongoDB: документи JSON, створені у інтерфейсі React.js, можна надіслати на сервер Express.js, де вони можуть бути оброблені та (за умови їх дійсності) збережені безпосередньо в MongoDB для подальшого пошуку. Але використання нереляційної бази даних для проекту, який буде активно масштабуватися та потребуватиме оптимізації запитів БД, буде раціональніше вкористовувати реляційну СУБД, наприклад MySQL.

React.js - це JavaScript-бібліотека від Facebook для зручної розробки інтерфейсів, тобто зовнішньої частини сайтів та програм, з якою взаємодіє користувач. Головна перевага React.js – компоненти та стани та віртуальний DOM – об’єкт який складається з дерева елементів, з яких будується відображення сторінки та інтерфейсу для зміни цього відображення. Компонент – це частина коду, який відповідає за зовнішній вигляд одного з елементів сайту або програми. При цьому такі компоненти можуть бути вкладеними один в інший. Стан — це вся інформація про елемент, яка потрібна дляйого відображення. Наприклад, стан об'єкта термометр може описуватися властивостями: максимальна та мінімальна температури, міра виміру, поточна темепература. Провівши дослідження серед інших бібліотек чи фреймворків, було прийнято рішення використовувати React.js, бо його легко інтегрувати, масштабувати та оптимізувати під різні запити. Також дана бібліотека на даний момент є найпопулярнішою – це в свою чергу дає перевагу при пошуку інформації, додаткових пакетів, чи інших готових рішень, також цей вибір є раціональним з точки зору пошуку програмістів, які будуть підтримувати систему – зменшення затрат, так як багато спеціалістів за даним напрямком.

На сьогоднішній день найбільш поширеним підходом для організації інформаційної складової вебсайтів є реляційні бази даних. БД мінімізують дублювання даних, вдаючись до дублювання тільки з метою прискорення доступу до даних або для забезпечення відновлення БД при її руйнуванні. Одна з найважливіших характеристик БД – це незалежність даних від особливостей прикладних програм, які їх використовують. Іншою важливою особливістю БД є можливість зміни фізичних особливостей зберігання даних без зміни їх логічної структури. Бази даних дозволяють зберігати і отримувати доступ до інформації. Використання БД на вебсайті дозволяє відстежувати дані, автоматично оновлювати сайт та розпізнавати користувачів. Інформація може оброблятися, зберігатися і вилучатись з БД. В БД можна зберігати добре структуровану інформацію, таку як список користувачів, список замовлень, прайс-листи, товари тощо. Однак цим вид інформації, яка зберігається не обмежується. В сучасних інформаційних системах в базах даних можна зберігати і тексти, і зображення, і навіть скрипти або коди програм.

Функціонування БД забезпечується сукупністю мовних та програмних засобів, які називаються системою управління базами даних (СУБД).

Основне завдання СУБД - надати користувачеві БД можливість працювати з нею, не зважаючи уваги на деталі рівня апаратного забезпечення. Іншими словами, СУБД дозволяє кінцевому користувачеві розглядати БД як об'єкт більш високого рівня в порівнянні з апаратним забезпеченням, а також надає в розпорядження набір операцій, виражений в термінах мови маніпулювання даними високого рівня - SQL.

Складні й керовані даними вебсайти використовують СУБД з декількох причин. По-перше, за допомогою SQL програміст має змогу перекласти більшість завдань зберігання та управління даними на СУБД. По-друге, СУБД краще людини справляються з управлінням великими обсягами даних, тобто зменшується вірогідність похибки та виникнення проблем в процесі роботи з даними. Також БД зберігають дані постійно, в той час, як змінні і їх значення в

скриптах на мові програмування Python зазвичай існують лише протягом запиту даної сторінки. Завдяки цій постійності СУБД реалізують ефективну роботу з диском та процес кешування даних.

Зберігання інформації в БД також дозволяє програмісту писати менший обсяг коду так, як завдання обробки даних передаються до СУБД і втой самий час розглядати абстрактно всю систему управління даними.

СУБД зазвичай дають наступні переваги:

1. висока цілісність даних;
2. ефективне забезпечення доступу до одних і тих самих даних в один час;
3. високий рівень захисту даних;
4. незалежність від файлових структур;
5. усунення надмірності інформації;
6. економія дискового простору завдяки об'єднанню таблиць без втрат.

MySQL являє собою одну з найбільш надійних, швидких, якісних і відомих з усіх існуючих сучасних систем управління базами даних. Основною причиною цього є її безкоштовне розповсюдження разом зі своїми вихідними кодами, інша причина - це те, що MySQL досить швидка СУБД. Головна особливість роботи СУБД MySQL полягає в використанні мови структурованих запитів - SQL в ролі управління базою даних, а саме: для створення або видалення таблиць в базі даних, здійснення вибірки з бази даних, для безпосереднього заповнення таблиць даними. MySQL має високу стійкість, високою швидкістю роботи, простотою в налаштуванні і використанні, вихідні коди сервера компілюються на безлічі платформ. Основні можливості MySQL:

1. надає можливість одночасної роботи з базою даних необмеженому числу користувачів;
2. кількість рядків у таблицях може досягати 50 млн;
3. висока швидкість виконання команд користувачів;

4. проста і ефективна система безпеки.

Під платформу Node.js існує багато прикладних пакетів для роботи з БД, наприклад, Sequilize надає API для роботи з базою даних, тому напряму використовувати SQL немає потреби, але така можливість є – в представленнях Node.js можливо писати запити на чистому SQL, використовуючи спеціальні конструкції, які надає юіюліотека. ORM – технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктнубазу даних». Маючи список таблиць в базі даних та об'єктів в програмі, ORM автоматично перетворить запити з вигляду коду на JavaScript до запитів SQL. В результаті запиту об'єкта необхідний SQL-запит буде сформований автоматично і виконаний, а результати «магічним» чином перетворені в об'єкти всередині програми. З погляду програміста система повинна виглядати як постійне сховищео б'єктів. Він може просто створювати об'єкти і працювати з ними як завжди, а вони автоматично зберігатимуться в реляційній базі даних. На практиці все не так просто й прозоро. Всі системи ORM зазвичай проявляють себе в тому або іншому вигляді, зменшуючи в деякому роді можливість ігнорування бази даних. Більш того, рівень транзакцій може бути повільним і неефективним (особливо в термінах згенерованого SQL). Все це може привести до того, що програми працюватимуть повільніше і використовуватимуть більше пам'яті, ніж програми, написані «вручну». Але ORM позбавляє програміста від написання великої кількості коду, часто одноманітного і схильного до помилок, тим самим значно підвищуючи швидкість розробки. Крім того, більшість сучасних реалізацій ORM дозволяє програмістові при необхідності жорстко задати код SQL-запитів, який буде виконаний при тих чи інших діях (збереження в базу даних, завантаження, пошук тощо) з постійним об'єктом.

При проведенні досліджень архітектурних підходів, було прийнято використання мікросервісів (рис 2.1) – підхід, коли потік даних може бути

асинхронним. У такому разі можна буде використовувати системи черг для комунікації, які дають широкі можливості управління трафіком.

Бекенди мікросервісів, у міру своїх можливостей, коли у них звільняється потік виконання, підхоплюють ці дані. Такий підхід дає незалежність розробки сервісів (мікросервіси можуть бути написані різними мовами). До того ж, кожен мікросервіс працює незалежно від інших частин системи.

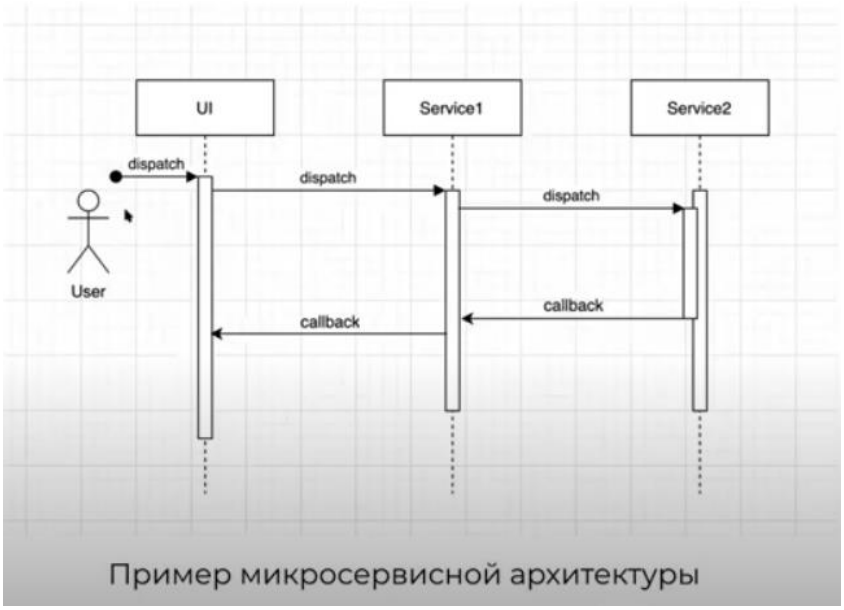

Рис.2.1. Схема архітектури MVT

Користувач робить запит. Проходить деякий час і UI робить запит на перший мікросервіс, де ці дані валідуються, та йдуть на другий мікросервіс. Там дані обробляються, потім відправляються назад. Весь цей час користувач чекає. Тобто це хороший кейс для мікросервісів, якщо під час обробки даних користувач може робити щось на UI, поки не отримає повідомлення про виконання запиту. Але зазвичай користувач нічого не робить на сторінці в очікуванні відгуку, тому за такої схеми він весь час обробки даних мікросервісами буде просто чекати.

В даній кваліфікаційні роботі після порівняння можливих підходів, було прийнято рішення розділення системи на дві окремі складові: фронтенд та

бекенд застосунки. Це дає змогу масштабувати та модифікувати систему легше та швидже, аніж при утримці всієї кодової бази в монолітному стані. Тобто є розподілення за процесами: робота з даними та відображення даних.

Node.js в кваліфікаійній роботі реалізує REST API сервіс для мікрофронтеду взаємодії коритстувача системи з темами, запитаннями форуму. Надається імплементація мікросервісної архітектури з точки погляду розробки ІС веб-сайт форуму. Для сховища даних обрана СУБД MySQL – після проведення порівняння з NoSQL СУБД MongoDB, прийнято таке рішення, бо подальше можливе масштабування та оптимізація буде легше для імплементації. Також реляційна БД швидше обробляє запити, налаштувавши індексацію даних, можна в декілька разів підвищити швидкість обробки даних.

2.3. Використання Redux для управління даними клєінтської частини

Механізм локального сховища даних компонента, що постачається разом із базовою бібліотекою (React), незручний тим, що таке сховище ізольоване. Наприклад, якщо потрібно, щоб різні незалежні компоненти реагували на будь-яку подію, доведеться або передавати локальний стан у вигляді пропсів дочірнім компонентам, або піднімати його до найближчого батьківського компонента. В обох випадках робити це не зручно. Код стає більш важким для читання, навантаженим, а компоненти, залежними від їх вкладеності. Redux знімає цю проблему, оскільки весь стан доступний усім компонентом без особливих труднощів.

По суті Redux – це об’єкт, який зберігає дані для всього застосунку та надає методи для можифікації цих даних, при будь-якій зміні стану застосунку – ця бібліотека точково оновить відображення компонентів. Redux наслідує архітектуру Flux, та надає декілька абстракцій: редьюсер – функція, яка приймає об’єкт екшену та необхідну частину стану застосунку; екшен – подія,

на яку необхідно виконати деяку зміну в стані застосунку; та сас стан застосунку. Даний підхід в процесі дослідження теоретичних матеріалів опинися найкращим, серед інших конкурентних технологій. Таким чином легко оперувати станом застосунку на клієнті.

Процес роботи Redux (рис. 2.2) складається з декількох частин: користувач здійснює якусь операцію в представленні, яка впливає на стан даних застосунку; ця операція створює так званий «екшен» - об’єкт події, на яку має зреагувати Redux «стор»; потім спрацьовує відповідний диспетчер екшенів і пеердає екшен до функцій-редьюсерів, які змінять або повернуть попередній стан «стору» залежно від характеристик екшену. Після цього цикл потоку даних закінчується і спеціальних механізм Redux – connect повідомляє React про потребу оновлення представлення.

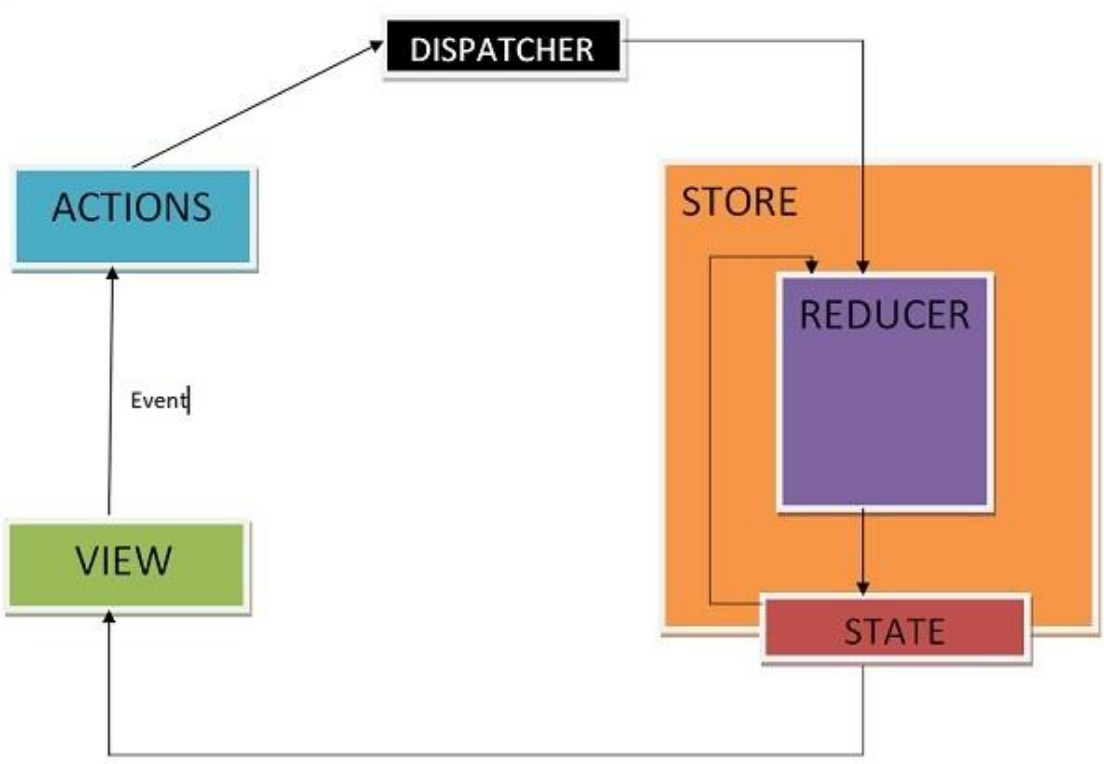

Рис.2.2. Схема архітектури Redux

2.4. Порівняльне тестування баз даних MySQL та MongoDB

2.4.1. Обгрунтування вибіру структури таблиць

Перша таблиця складатиметься з 30000 записів з однаковим текстом і випадковим зовнішнім ключем у діапазоні від 1 до 100000. Друга таблиця складатиметься з 100000 записів, поля mainKey та value кожного рядка будуть рівні між собою та приймати значення від 1 до 100000.

Текст у першій таблиці потрібен виключно для збільшення обсягу даних, що записуються. Атрибут mainKey другої таблиці має таку назву, щоб підкреслити, що вона не є первинним ключем. Не зробилен він первинним ключем з тих міркувань, що це уповільнить швидкість виконання операцій у MySQL через додаткову перевірку цілісності. Поля mainKey і value приймають однакові значення лише для спрощення сприйняття, концептуально це ні на що не впливає.

Слід зазначити, що таблиця 2 (рис 2.4) більша за таблицю 1 (рис. 2.3). По-перше, це має зробити пошук по mainKey трохи складніше. По-друге, внаслідок першого факту, це дозволить дещо ускладнити процедуру join-а двох таблиць.

```
{
    text: 'fifteen symbols',
    foreignKey: 'random(1,100000')
}
```

Рис.2.3. Схема таблиці № 1

```
{
    mainKey: 1...100000,
    value: 1...100000
}
```

Рис.2.4. Схема таблиці № 2

2.4.2. Опис процесу тестування

Для тестування ефективності двох БД проведено такі тести:

1. Вставка до 100000 рядків до таблицю № 1. Результати додавання рядків до таблиці № 2 не наведено через їхню аналогічність.

```
for i in range(100000):
    cursor.execute("insert into mainKey_value values(%d, %d)" % (i, i))
```

Рис.2.5. Код для MySQL

```
for i in range(100000):
    db.mainKey_value.insert({'mainKey': i, 'value': i})
```

Рис.2.6. Код для MongoDB

2. Поєднання двох таблиць. У MySQL використовувано LEFT JOIN, У MongoDB - $lookup. Тут варіюватимеься розмір таблиці № 1 від 10000 рядків до 30000.

```
1   for i in range(30000):
2       cursor.execute("insert into text_foreignKey values('fifteen symbols', %d)" % generatedKeys[i])
3       if not (i + 1) % 10000:
4           cursor.execute("select * from text_foreignKey left join mainKey_value on text_foreignKey.foreignKey =
mainKey_value.mainKey")
5           selectedList = list(cursor)
```

Рис.2.7. Код для MySQL

```
1   for i in range(30000):
2       db.text_foreignKey.insert({'text': 'fifteen symbols', 'foreignKey': generatedKeys[i]})
3       if not (i + 1) % 10000:
4           aggregated = db.text_foreignKey.aggregate([
5               {
6                   "$lookup":
7                       {
8                           "from": "mainKey_value",
9                           "localField": "foreignKey",
10                          "foreignField": "mainKey",
11                          "as": "foreignKeyDoc"
12                      }
13              }
14          ])
15          aggregated_list = list(aggregated)
```

Рис.2.8. Код для MongoDB

3. Пошук до 10000 рядків у таблиці № 2 на ключі mainKey.

```
for i in range(rowRange):
    cursor.execute("select * from mainKey_value where mainKey = %d" % generatedKeys[i])
    foundRows = list(cursor)
```

Рис.2.9. Код для MySQL

```
for i in range(docRange):
    foundKeys = list(db.mainKey_value.find({'mainKey': generatedKeys[i]}))
```

Рис.2.10. Код для MongoDB

4. Зміна до 10000 рядків у таблиці № 2 за ключом mainKey.

```
for i in range(rowRange):
    cursor.execute("update mainKey_value set value = %d where mainKey = %d;" % (random.randint(1, 100000), generatedKeys[i]))
```

Рис.2.11. Код для MySQL

```
1     for i in range(docRange):
2         db.mainKey_value.update_many({'mainKey':generatedKeys[i]}, {'$set':{'value':random.randint(1, 100000)}})
```

Рис.2.12. Код для MongoDB

5. Видалення до 10000 у таблиці № 2 за ключом mainKey.

```
for i in range(rowRange):
    cursor.execute("delete from mainKey_value where mainKey = %d" % generatedKeys[i])
```

Рис.2.13. Код для MySQL

```
for i in range(docRange):
    db.mainKey_value.delete_many({'mainKey':generatedKeys[i]})
```

Рис.2.14. Код для MongoDB

Для всіх тестів, крім першого, потрібно виконувати пошук атрибуту mainKey. Тому всі тести проведено двічі: з індексуванням mainKey і без.

2.5. Аналіз результатів тестування

2.5.1. Результати з індексацією:

1. Вставка до 100000 рядків до таблицю № 1:

MySQL приблизно на 63 відсотки швидкісніший.

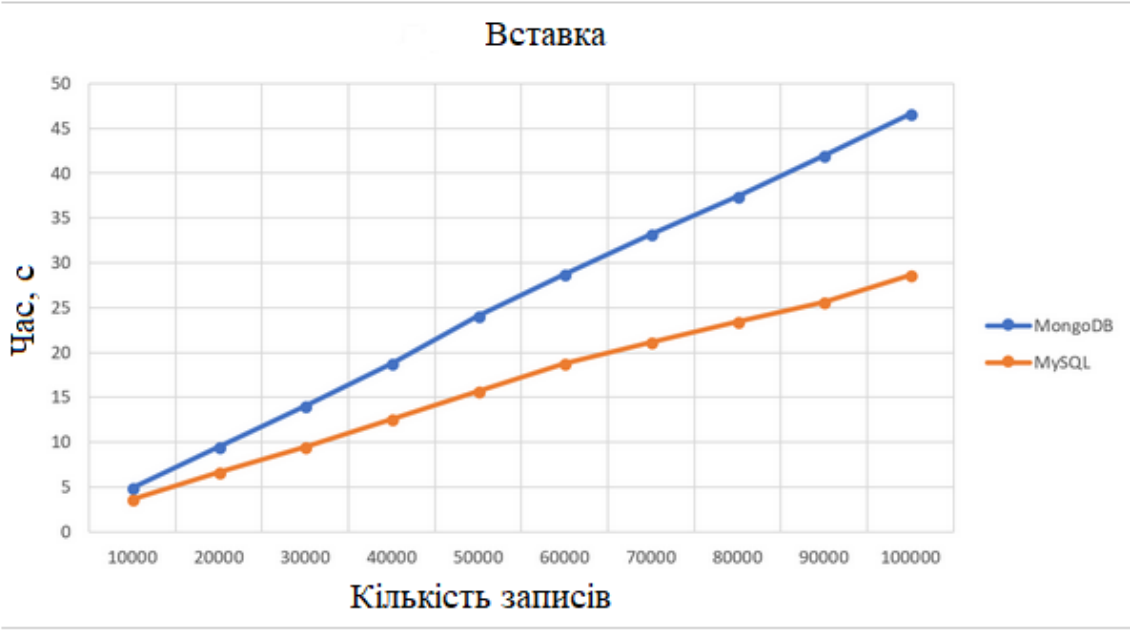

Рис.2.15. Результати тесту по вставкам

1. Поєднання двох таблиць:

MySQL в 12 разів кращу продуктивність показав.

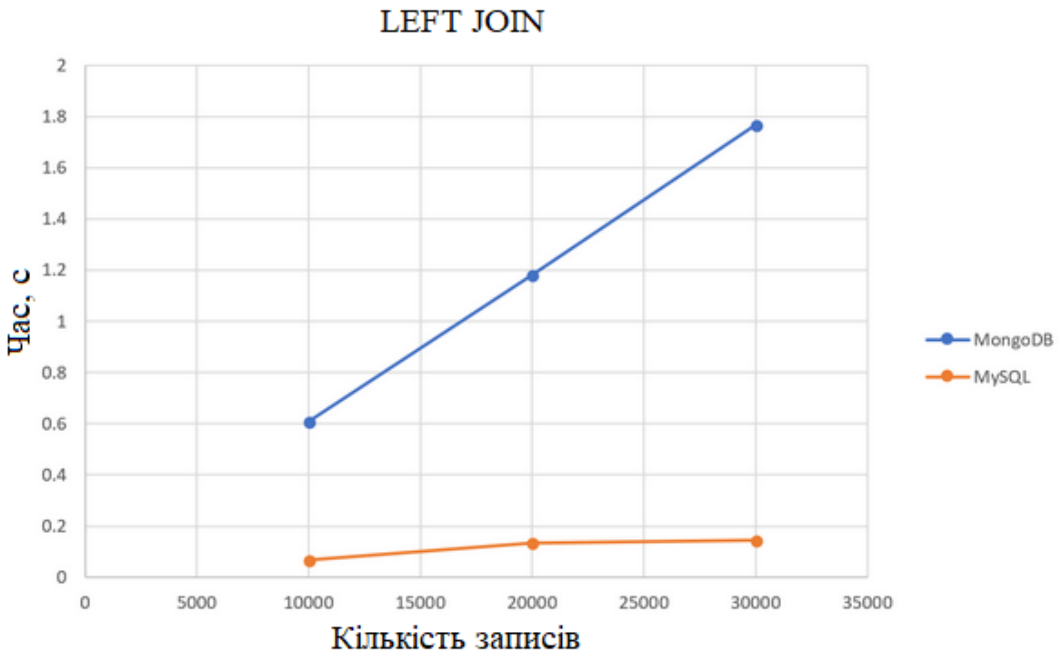

Рис.2.16. Результати тесту по LEFT JOIN

3. Пошук до 10000 рядків у таблиці № 2 на ключі mainKey:

MySQL приблизно на 95 відсотків швидкісніший.

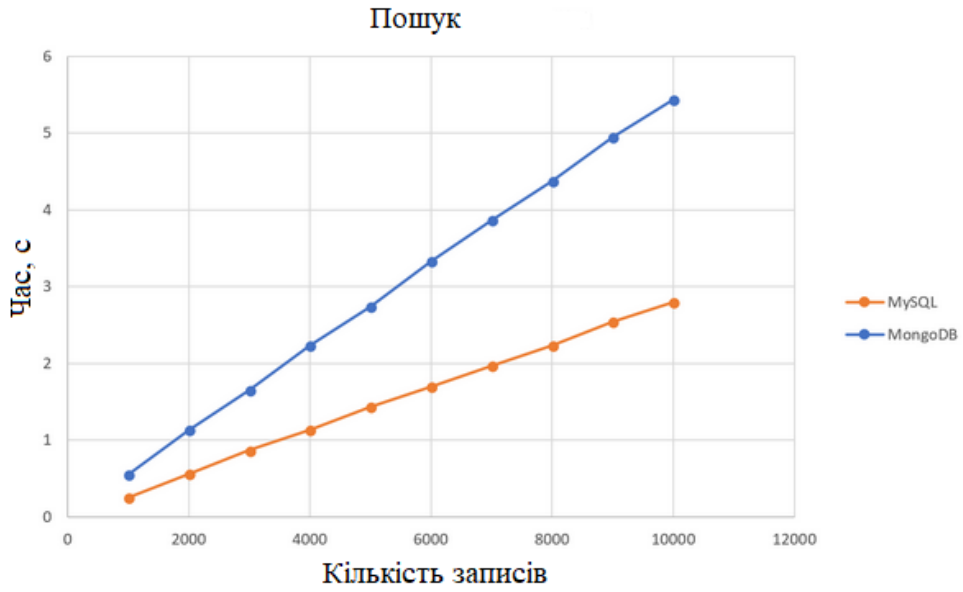

Рис.2.17. Результати тесту по пошуку

4. Зміна до 10000 рядків у таблиці № 2 за ключом mainKey: MySQL приблизно на 47 відсотки швидкісніший.

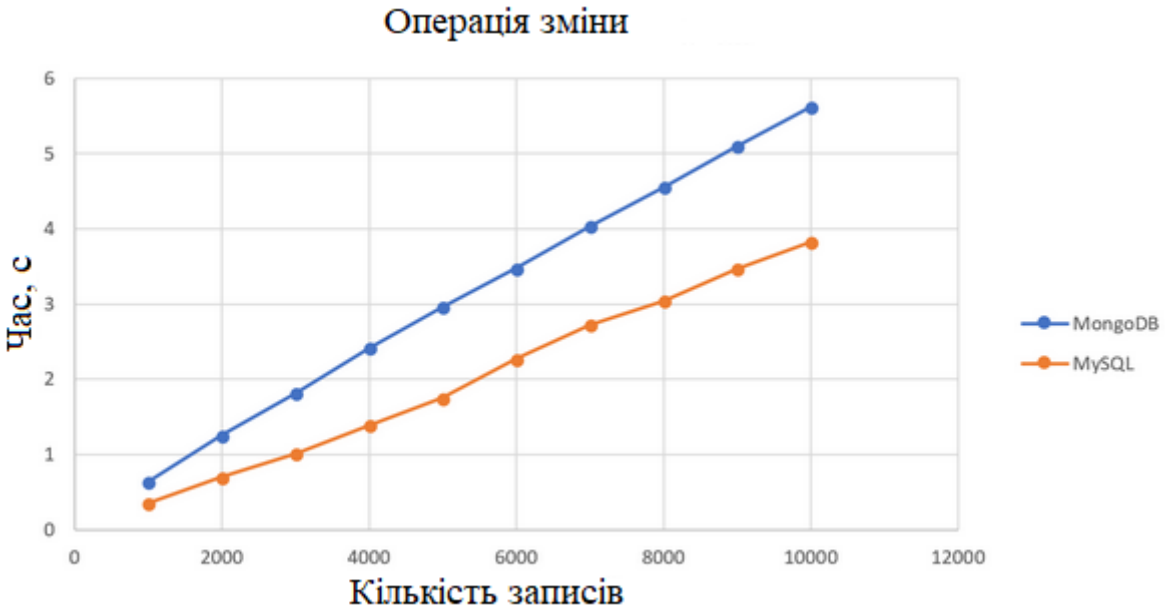

Рис.2.18. Результати тесту по зміні

5. Видалення до 10000 у таблиці № 2 за ключом mainKey: MySQL приблизно на 53 відсотки швидкісніший.

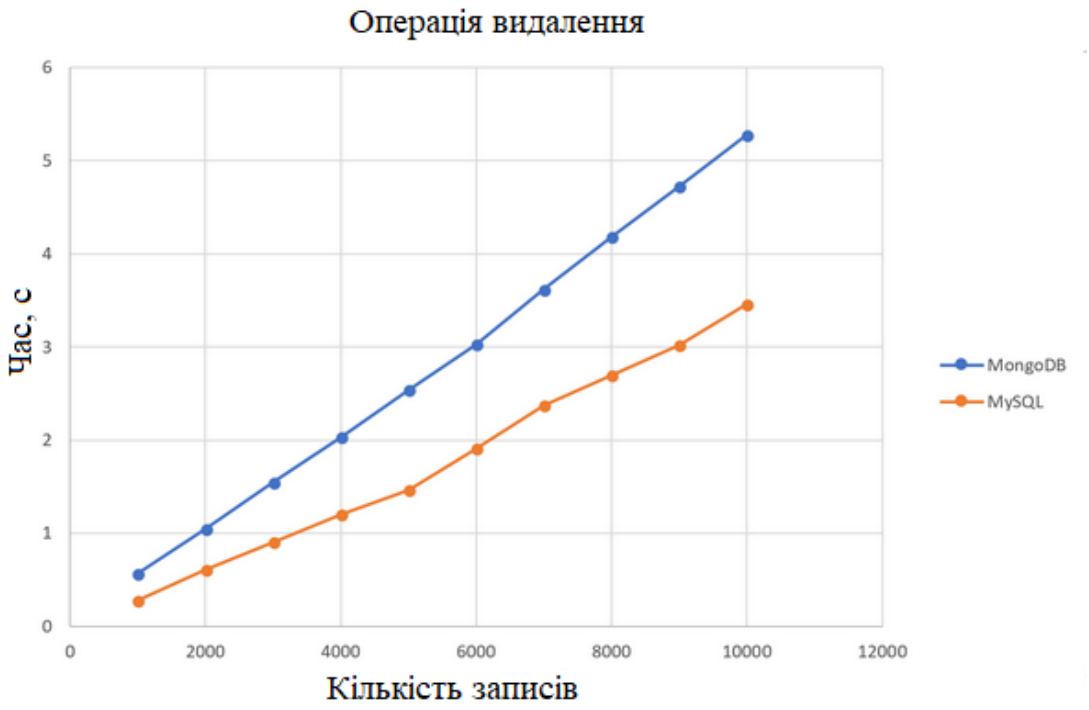

Рис.2.19. Результати тесту по видаленню

2.5.2. Результати без індексації:

1. Вставка до 100000 рядків до таблицю № 1:

MySQL приблизно на 75 відсотки швидкісніший.

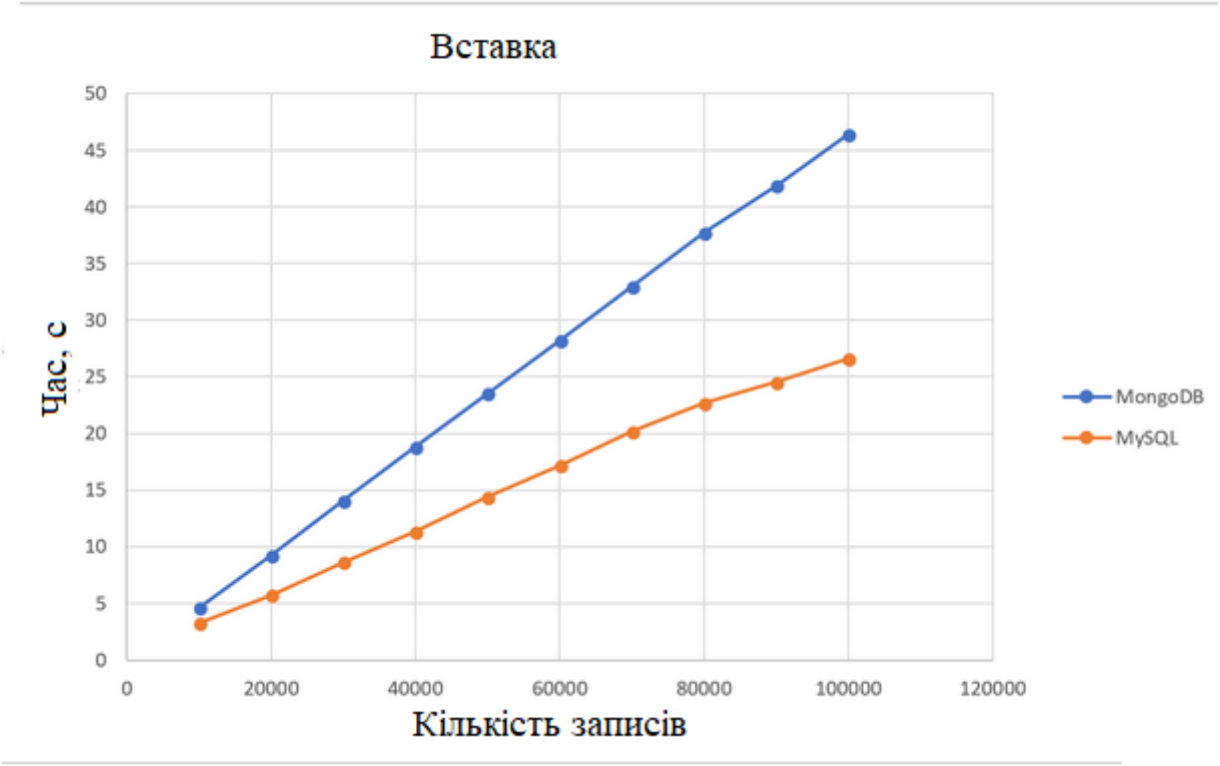

Рис.2.20. Результати тесту по вставкам

2. Поєднання двох таблиць:

У цьому тесті MySQL, як і раніше, має велику перевагу - приблизно у 4.24 рази.

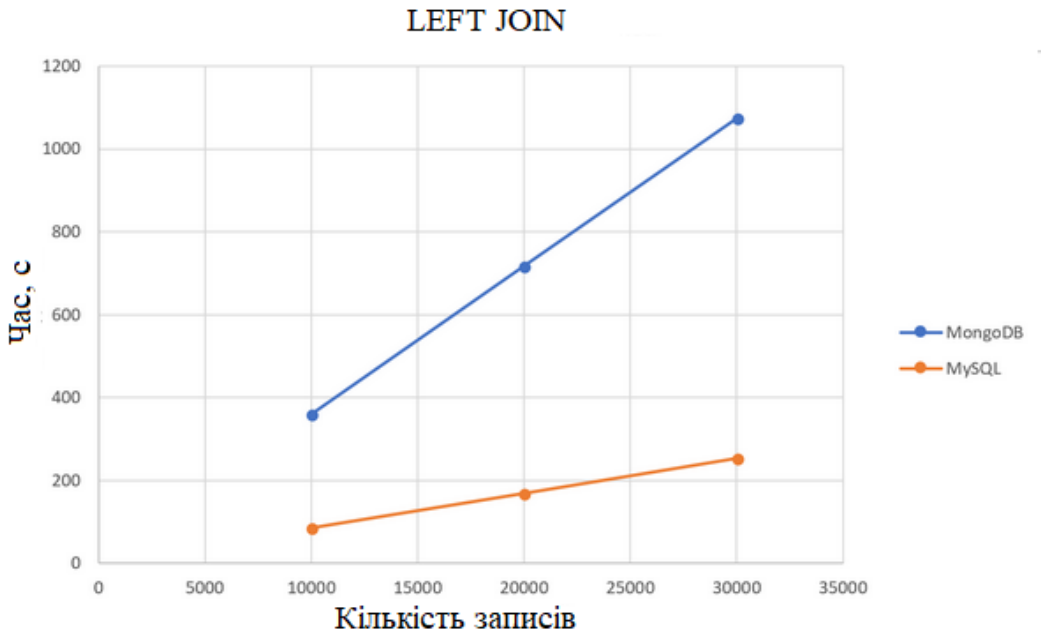

Рис.2.21. Результати тесту по LEFT JOIN

3. Пошук до 10000 рядків у таблиці № 2 на ключі mainKey:

Обидві СУБД показали ідентичні результати.

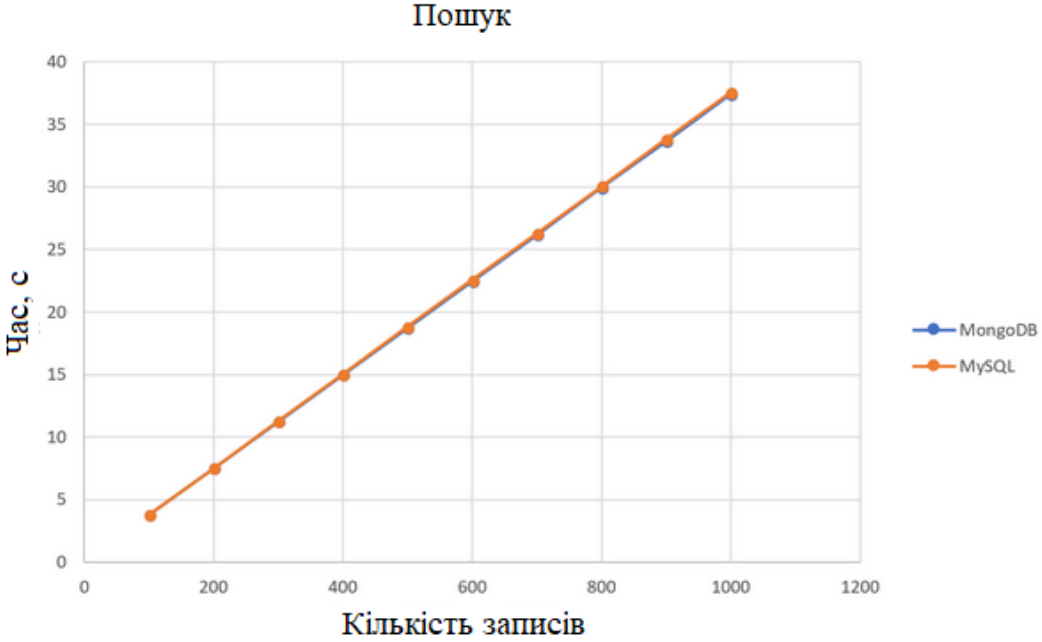

Рис.2.22. Результати тесту по пошуку

4. Зміна до 10000 рядків у таблиці № 2 за ключом mainKey: MongoDB в даному тесті обійшла MySQL на 21 відсоток.

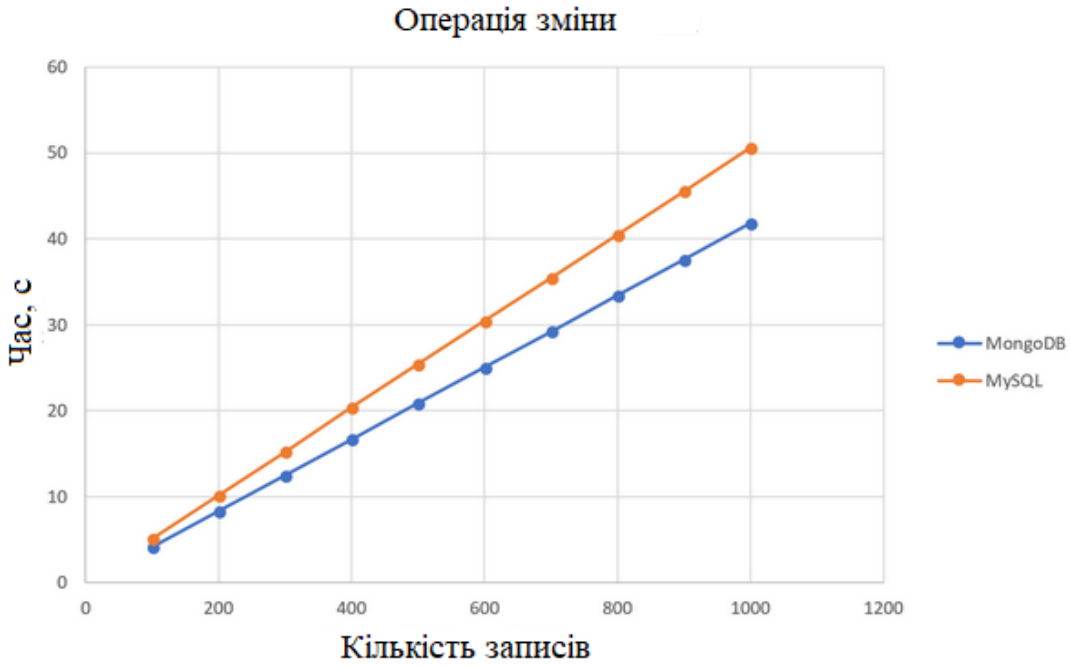

Рис.2.23. Результати тесту по зміні

5. Видалення до 10000 у таблиці № 2 за ключом mainKey: MongoDB в даному тесті обійшла MySQL на 24 відсотки.

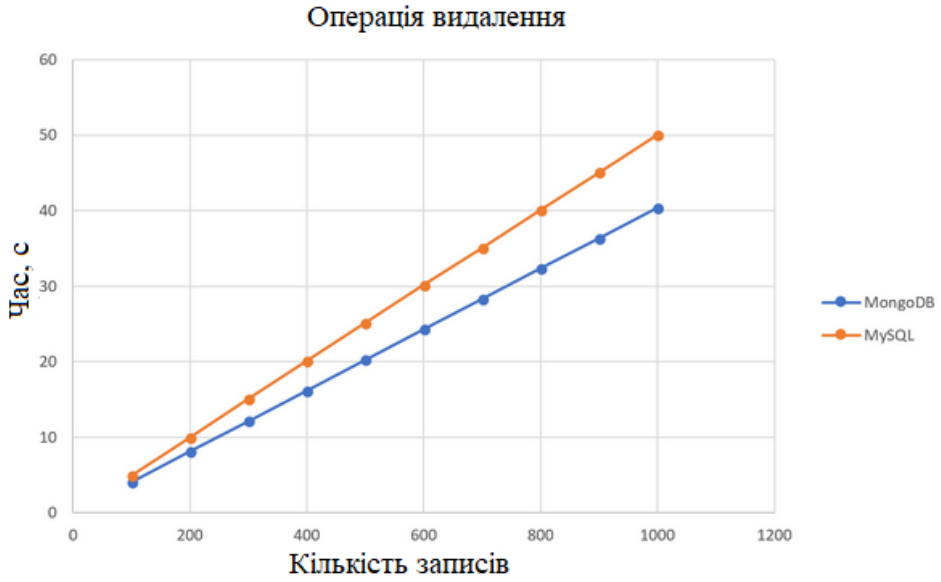

Рис.2.24. Результати тесту по видаленню

2.6. Висновки

В ході проведених досліджень та аналізу можливих підходів до проєктування автоматизованих інформаційних систем для забезпечення веб-сайтів, було обрано мікросервісний паттерн розробки з використанням MERN сукупності технологій. В якості СУБД для проєкту використано MySQL – реляційна БД, яка надає більшу швидкість операцій та краще підходить для оптимізації.

Отримані результати тестів цілком відповідають очікуванням. У тестах, де виконувались прості запити даних з однієї таблиці, маємо більш-менш схожі показники: MySQL виявилася швидше при роботі з таблицею, що має індекс, тоді як MongoDB вийшла переможцем, коли індекси не використовувалися. При цьому розрив між двома СУБД був не більше ніж удвічі, а не на порядок, як у випадку з LEFT JOIN. Звичайно, більший практичний інтерес має робота з проіндексованою таблицею.

Якщо ж розглядати роботу з даними, що мають деякі залежності один від одного, то тут переможець очевидний - MySQL.

Таким чином, якщо у даних присутні взаємозв'язки, то для досягнення максимальної швидкості роботи з ними набагато раціональніше використовувати MySQL (за умови, що дані досить нормалізовані). Навіть якщо взаємозв'язків не встановлено, але при цьому структура даних фіксована і заздалегідь відома, MySQL може виявитися найкращим вибором.

Після проведених досліджень було обрано мікросервісний підхід до проєктування системи та використання MERN сукупності технологій.

РОЗДІЛ 3
СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ЗАБЕЗПЕЧЕННЯ ЕФЕКТИВНОСТІ РОБОТИ ФОРУМУ ІТ-ФАХІВЦІВ

3.1. Структура інформаційної системи форуму ІТ-фахівців

Структура даної системи представляє собою вебсайт, в якому можна виділити дві основні частини: презентаційна (фронтенд) та серверна (бекенд).

З презентаційною частиною користувач взаємодіє безпосередньо, тут він має можливість зареєструватись у системі, отримати доступ до функціоналу створення, перегляду, коментарів та обговорень форуму.

Презентаційна частина взаємодіє з сервеною частиною комплексу для авторизації користувача, отримання його даних, отримання доступу до ресурсів, що знаходяться в системі, відправлення електронних. Взаємодія цих двух компонентів системи основана на HTTP-запитах та відповідях на них.

Розглянемо систему комунцкації презентаційної та серверної частини програмного комплексу. Ця система складається з двох об'єктів: вебклієнта та вебсервера. Між ними повинен існувати канал зв'язку. Веблієнт посилає запит на сервер, сервер відсилає відповідь.

Алгоритм взаємодії має таку структуру [10, 11]:

1. вебклієнт надсилає HTTP-запит певної вебсторінки. Наприклад, пошук оголошення на сайті, використовуючи HTML-форму на деякий URL;
2. вебсервер приймає запит на даний URL, та визначає яке представлення треба відобразити;
3. механізм Node.js починає синтаксичний аналіз сценарію представлення на виконання. React.js повертає відображення сторінки, якщо далі потрібні додаткові дані – запит проходить проксі та потрапляє на бекенд застосунок. Далі у сценарії серверного

застосунку присутні команда підключення до бази даних та контролер, який обробляє запит та відсилає відповідь у форматі JSON. Express.js відкриває з'єднання з сервером MySQL і відправляє необхідний запит;

4. сервер MySQL приймає запит до бази даних, обробляє його, а потім відправляє результати у відповідь;
5. механізм Express.js завершує виконання сценарію контролера, форматуючи результати запиту у вигляді JSON, після чого відправляє результати в до клієнта і вже клієнтський застосунок відображає інтерфейс з повним набором даних для конкретної сторінки.

Найчастіше програмне забезпечення - вебсервер Node.js і сервер баз даних знаходяться на одній машині. Поширеною практикою є перенесення сервера бази даних на окрему машину – зменшення навантаження на сервер, тобто другий варіант забезпечує більшу продуктивність.

Схема комунікації презентаційної та серверної частин (рис. 3.1):

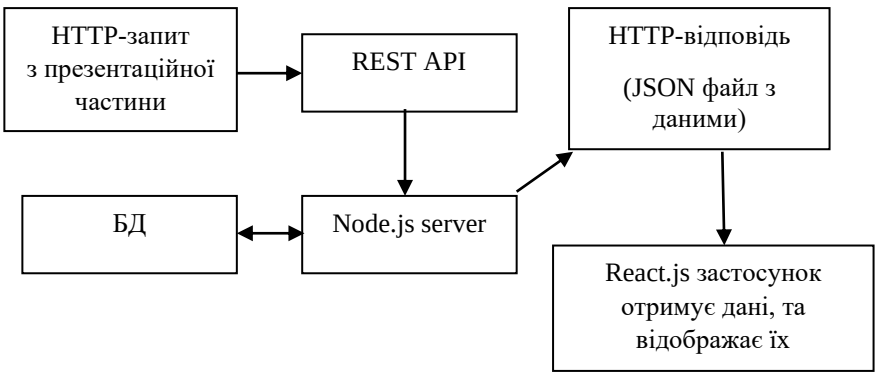

Рис. 3.1. Схема комунцкації презентаційної та серверної частин

Стосовно клієнтської частини можна виділити три основні складові: React, Redux, React-router. React – відповідає лише за оптимізоване оновлення UI при зміні вхідних даних в клієнтському застосунку, Redux виконує роль сховища даних на клієнті – та надає команди React про необхідність оновлення

інтерфейсу з новими даними. Тобто Redux виступає єдиним джерелом актуальних даних для React, на основі яких формується відображення. React-router виконує роль навігації по вебзастосунку без перезавантаження сторінки в браузері, тобто в режимі SPA.

3.2. Стурктура та опис бази даних

В якості БД для даної системи обрано – MySQL, яка являє собою одну з найбільш надійних, швидких, якісних і відомих з усіх існуючих сучасних систем управління базами даних. Основною причиною цього є її безкоштовне розповсюдження разом зі своїми вихідними кодами, інша причина - це те, що MySQL досить швидка СУБД. Головна особливість роботи СУБД MySQL полягає в використанні мови структурованих запитів - SQL в ролі управління базою даних, а саме: для створення або видалення таблиць в базі даних, здійснення вибірки з бази даних, для безпосереднього заповнення таблиць даними. MySQL має високу стійкість, високою швидкістю роботи, простотою в налаштуванні і використанні, вихідні коди сервера компілюються на безлічі платформ. Основні можливості MySQL:

1. надає можливість одночасної роботи з базою даних необмеженому числу користувачів;
2. кількість рядків у таблицях може досягати 50 млн;
3. висока швидкість виконання команд користувачів;
4. проста і ефективна система безпеки.

Node.js надає API для роботи з базою даних – за допомогою додаткового пакету Sequilize [12, 14], тому напряму використовувати SQL немає потреби, але така можливість є – можливо писати запити на чистому SQL, використовуючи спеціальні конструкції, які надає фреймворк. ORM – технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктнубазу даних».

Маючи список таблиць в базі даних та об'єктів в програмі, ORM автоматично перетворить запити з вигляду коду на JavaScipt до запитів SQL. В результаті запиту об'єкта необхідний SQL-запит буде сформований автоматично і виконаний, а результати «магічним» чином перетворені в об'єкти всередині програми. З погляду програміста система повинна виглядати як постійне сховищео б'єктів. Він може просто створювати об'єкти і працювати з ними як завжди, а вони автоматично зберігатимуться в реляційній базі даних. На практиці все не так просто й прозоро. Всі системи ORM зазвичай проявляють себе в тому або іншому вигляді, зменшуючи в деякому роді можливість ігнорування бази даних. Більш того, рівень транзакцій може бути повільним і неефективним (особливо в термінах згенерованого SQL). Все це може привести до того, що програми працюватимуть повільніше і використовуватимуть більше пам'яті, ніж програми, написані «вручну». Але ORM позбавляє програміста від написання великої кількості коду, часто одноманітного і схильного до помилок, тим самим значно підвищуючи швидкість розробки. Крім того, більшість сучасних реалізацій ORM дозволяє програмістові при необхідності жорстко задати код SQL-запитів, який буде виконаний при тих чи інших діях (збереження в базу даних, завантаження, пошук тощо) з постійним об'єктом.

Структура бази даних має наступний вигляд (рис. 3.2):

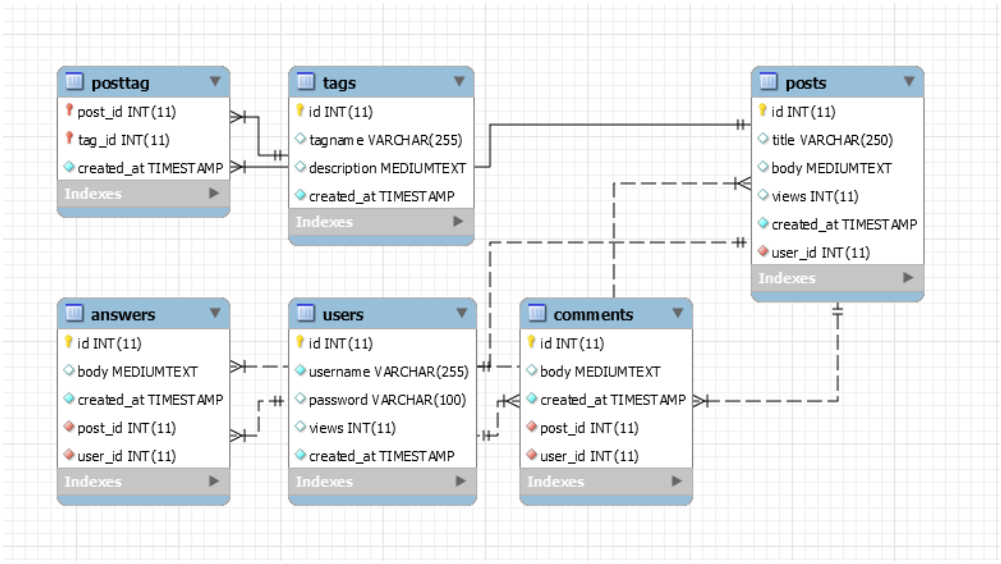

Рис. 3.2. Структура бази даних

3.3. Опис ІС форуму та алгоритмів її функціонування

Основним обробником даних системи є елементи серверного застосунку, які є складовою фреймворку Express.js. В якості вхідних даних в цьому комплексі використовуються текстові дані, які користувач вводить при створенні тегів, тем, обговорень.

Вихідними даними вебдодатку можна вважати множину збережених у системі тегів та обговорень, які доступні до перегляду користувачами.

Функціональне призначення системи полягає в наданні користувачу зручного та інтуїтивно-зрозумілого шляху створення тем для розгляду та відповіді на відкриті питання, які наявні в системі та відповідають характеристикам, зазначеним за допомогою елементів управління на сайті, конкретним користувачем.

Розроблений в ході кваліфікаційної роботи вебсайт форуму ІТ-фахівців «PROLOG» має відповідати наступним вимогам:

– зручний інтуїтивно зрозумілий інтерфейс для роботи з системою користувачами різних кваліфікацій;

– створення облікового запису та його подальше редагування;

– повинна бути реалізована мінімальна система автентифікації;

– створення тем для обговорення на форумі;

– коментування вже створених (існуючих) тем видалення;

– забезпечувати можливість роботи в браузерах на ПК, ноутбуках та мобільних пристроях різних характеристик та конфігурацій під управлінням сімейства ОС Windows, Mac OS, Linux, IOS, Android;

– не бути вимогливим до складу програмних та апаратних засобів;

– бути гнучким для модифікацій та оновлення;

– навігація по вебсайту, реалізована за допомогою текстових посилань на відповідні ресурси системи;

– наявність модулю процедури пошуку обговорень, збережених в системі.

Для розробки вебсайту застосувались такі мови програмування та технології:

– Sequilize ORM – API для роботи з базою даних;
– MySQL – обробка та зберігання даних;
– CSS – стилізація компонентів UI;
– HTML – створення розмітки UI;
– JavaScript – створення клієнтської логіки;
– React.js для створення компонентів інтерфейсу користувача;
– Redux для оперування даними застосунку на клієнтській стороні;
– Node.js у зв’язці з Express.js – створення бізнес логіки застосунку.

Інтерфейс користувача вебсайту складається з декількох основних модулів:

1. головна сторінка;
2. сторінка авторизації;
3. сторінка реєстрації;
4. сторінка з усіма обговореннями форуму;
5. сторінка тегів;
6. сторінка інформації про користувачів;

Після переходу за інтернет-адресою даного програмного комплексу, користувач побачить головну сторінку, з десятьма найновішими обговореннями, які були створені в системі (рис. 3.3):

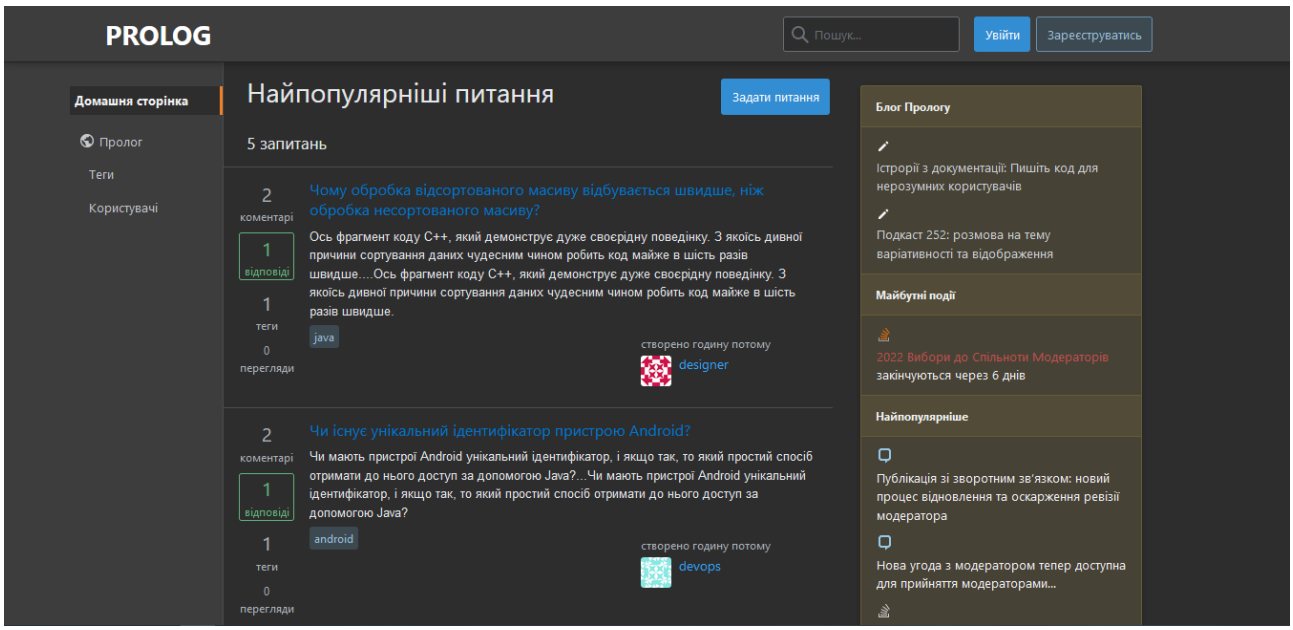

Рис. 3.3. Головна сторінка сайту

Якщо користувач вже має обліковий запис в системі, то він може авторизуватись за посиланням «Увійти», зазначивши у формі, яка з'явиться свій пароль та ім'я, вказане при реєстрації (рис. 3.4, рис 3.5):

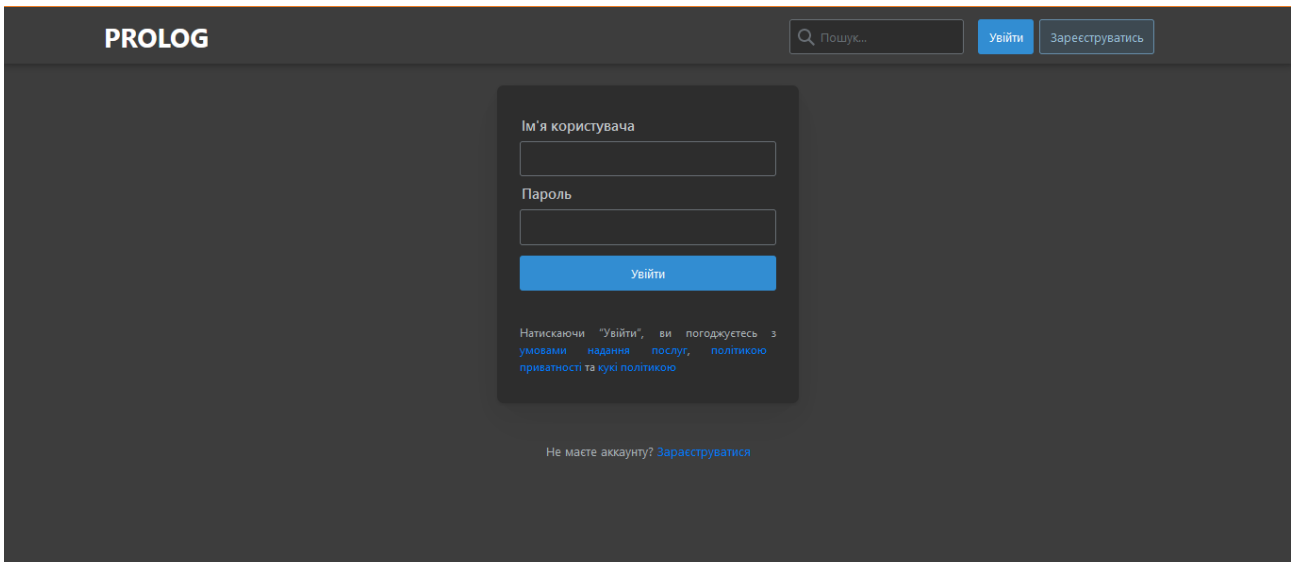

Рис 3.4. Сторінка авторизації (1)

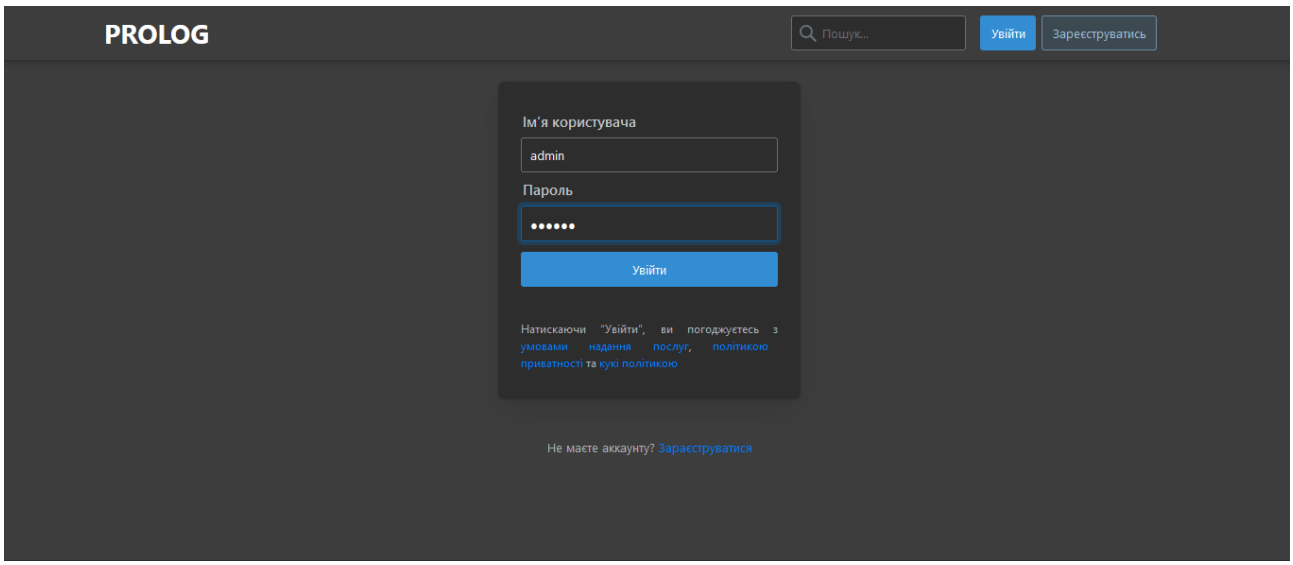

Рис 3.5. Сторінка авторизації (2)

Якщо користувач не має облікового запису в сисетмі, то за посиланням «Зареєструватись» йому відкриється сторінка для реєстрації (рис. 3.6), де потрібно буде ввести дані в форму:

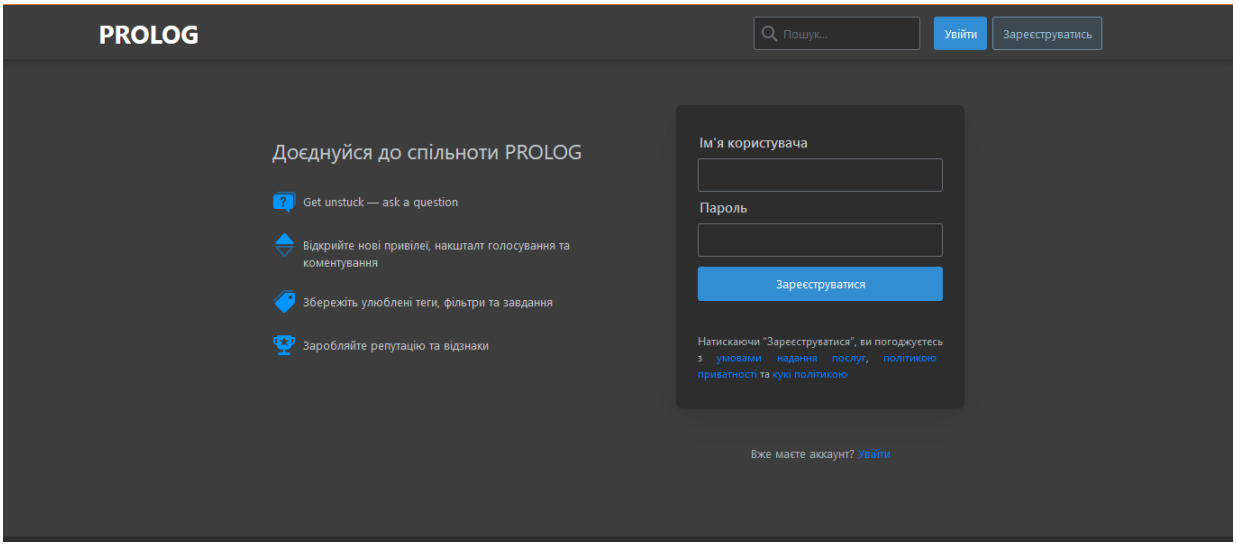

Рис. 3.6. Сторінка реєстрації в системі

Після того, як користувач введе правильно дані та натисне на кнопку «Зареєструватись» буде створено новий обліковий запис. Дана реалізація системи аутентифікації не є повною, в кваліфікаціцній роботі робився наголос на дослдіженні проєктування системи та використання реляційної бази даних. Тому в даному випадку система аутентифікації може бути розроблена таким

самим чином у якості мікрофронтенду та надавати цей функціонал іншим корпоративним застосункам.

Після процесу створення профілю, користувачу стає доступним меню системи (рис 3.7), яке дозволяє переходити між сторінками мікрофронтенду:

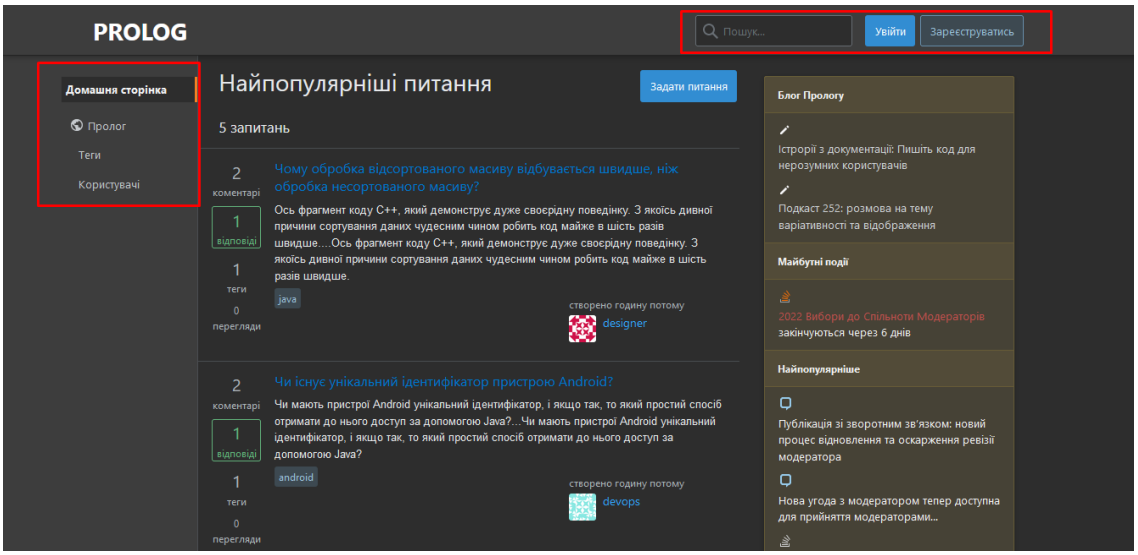

Рис. 3.7. Меню мікрофронтенду

Мікрофронтенд складається з декількох сторінок: сторінка для перегляду створених обговорень за посиланням «Пролог» (рис. 3.8) та сторінка для детального перегляду обговорення (рис. 3.9), сторінка для перегляду всіх наявних у системі тегів за посиланням «Теги» (рис. 3.10) та сторінка детальної інформації по тегу (рис. 3.11), сторінка з переліком всіх користувачів «Користувачі» (рис. 3.12) та сторінка з детальною інформацією по користувачу (рис. 3.13), сторінка 404 помилки (рис. 3.14):

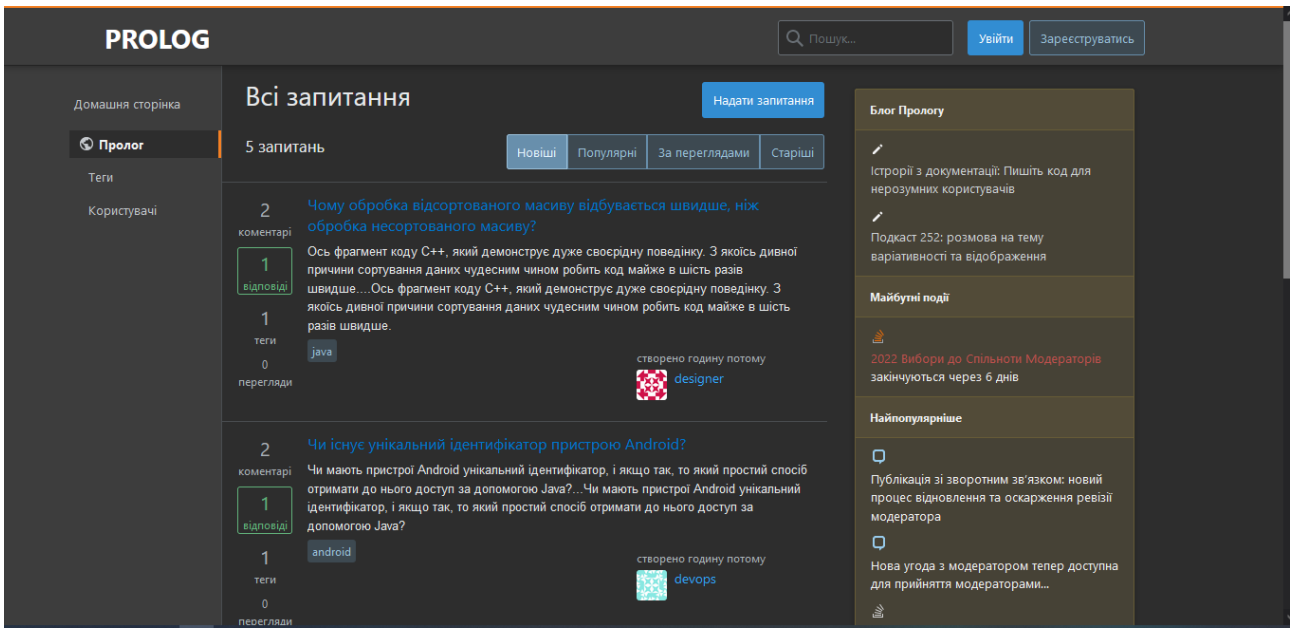

Рис. 3.8. Сторінка нових обговорень

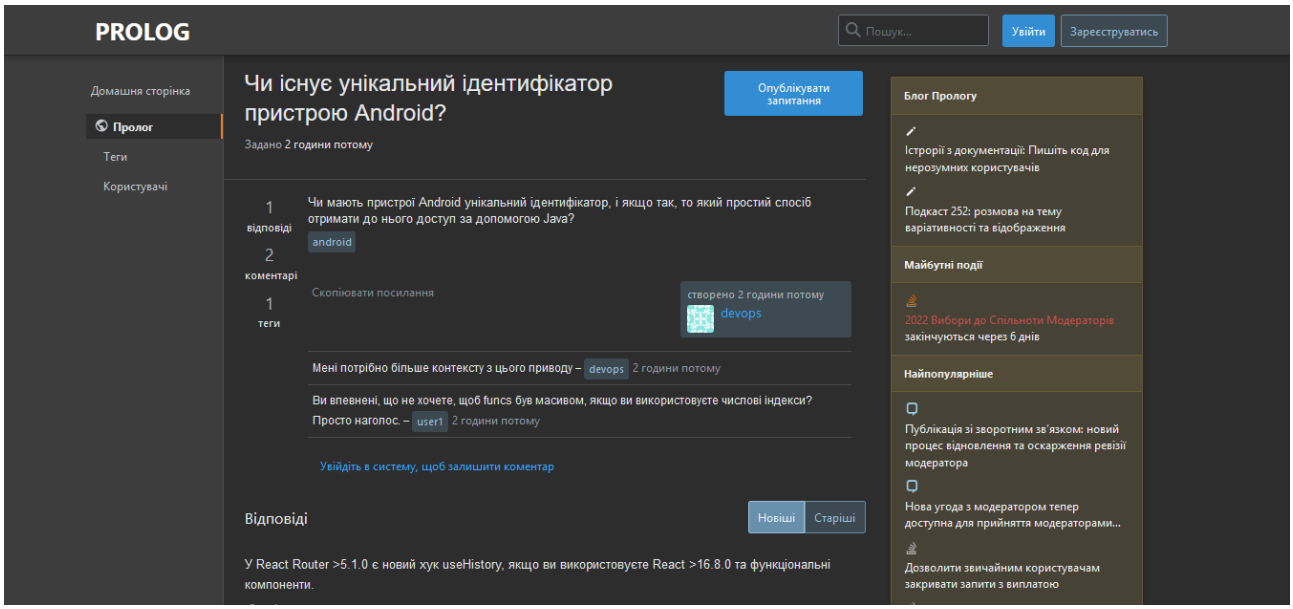

Рис. 3.9. Сторінка детального перегляду обговорення

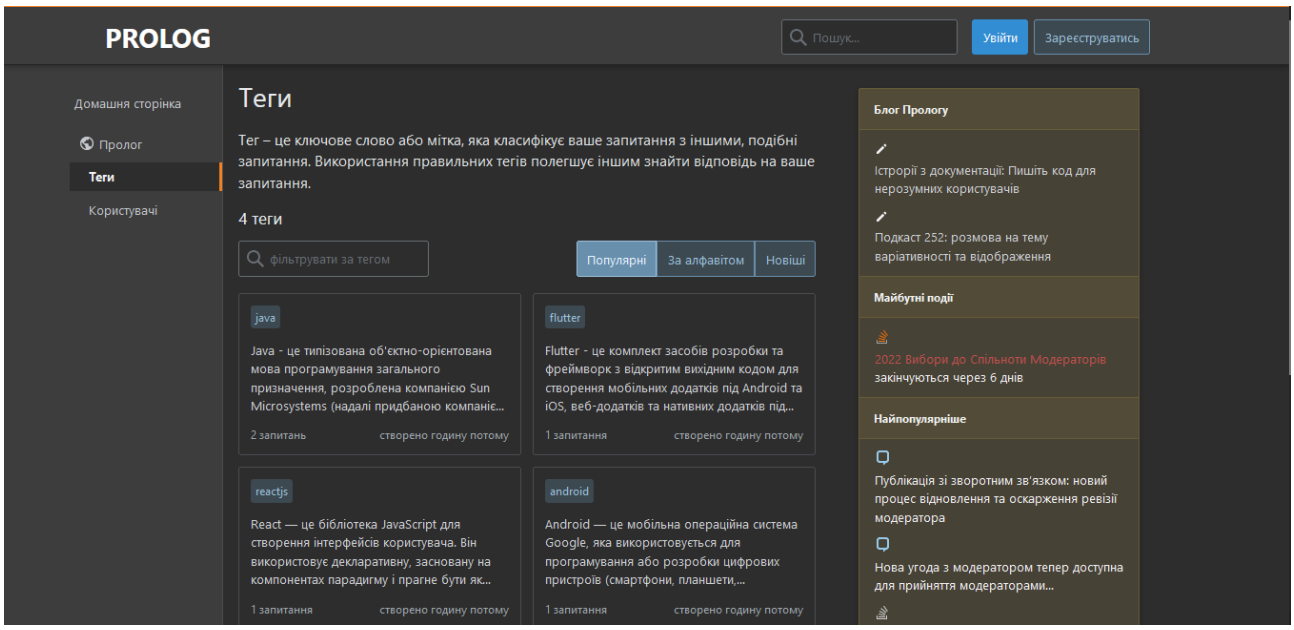

Рис. 3.10. Сторінка перегляду тегів

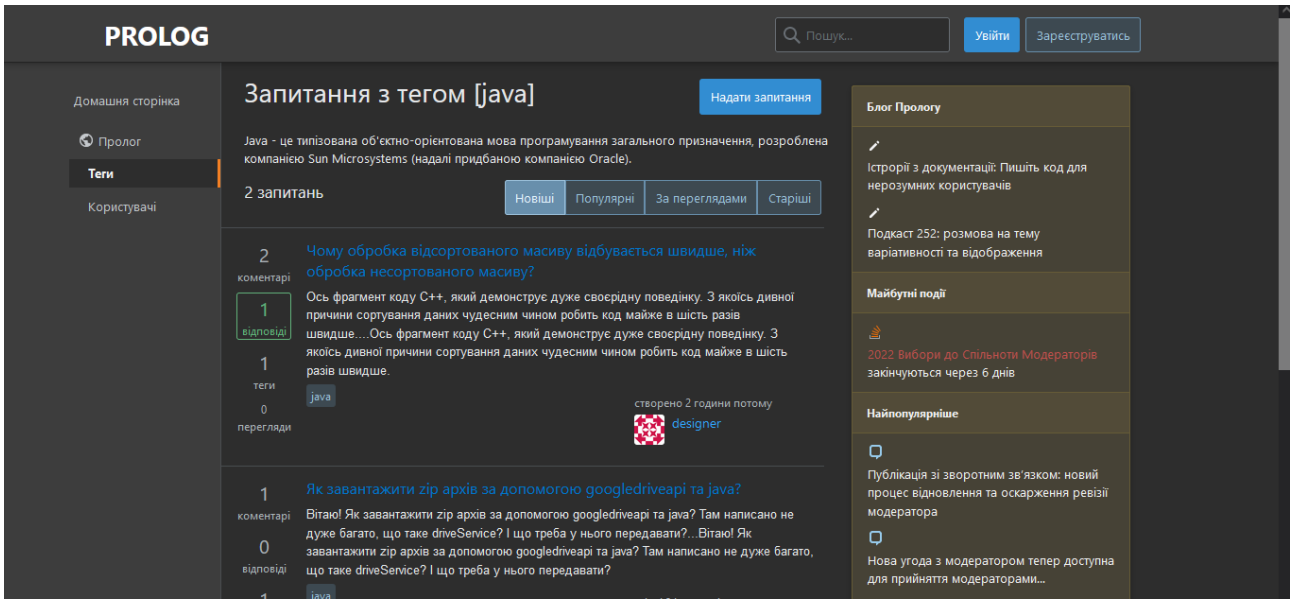

Рис. 3.11. Сторінка детального перегляду тегу

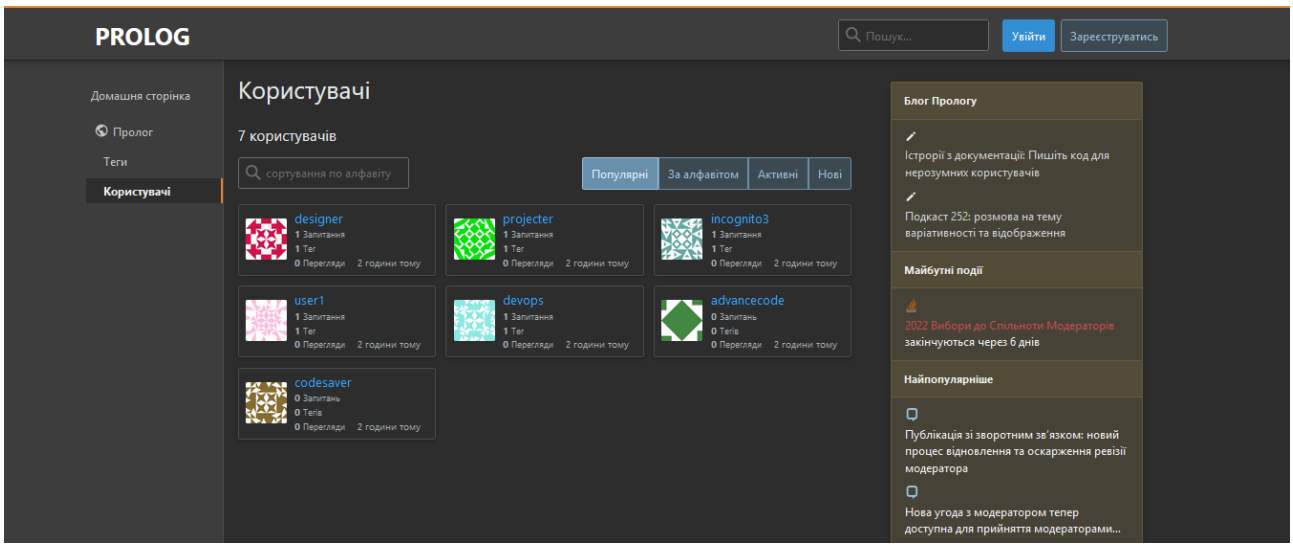

Рис. 3.12. Сторінка користувачів системи

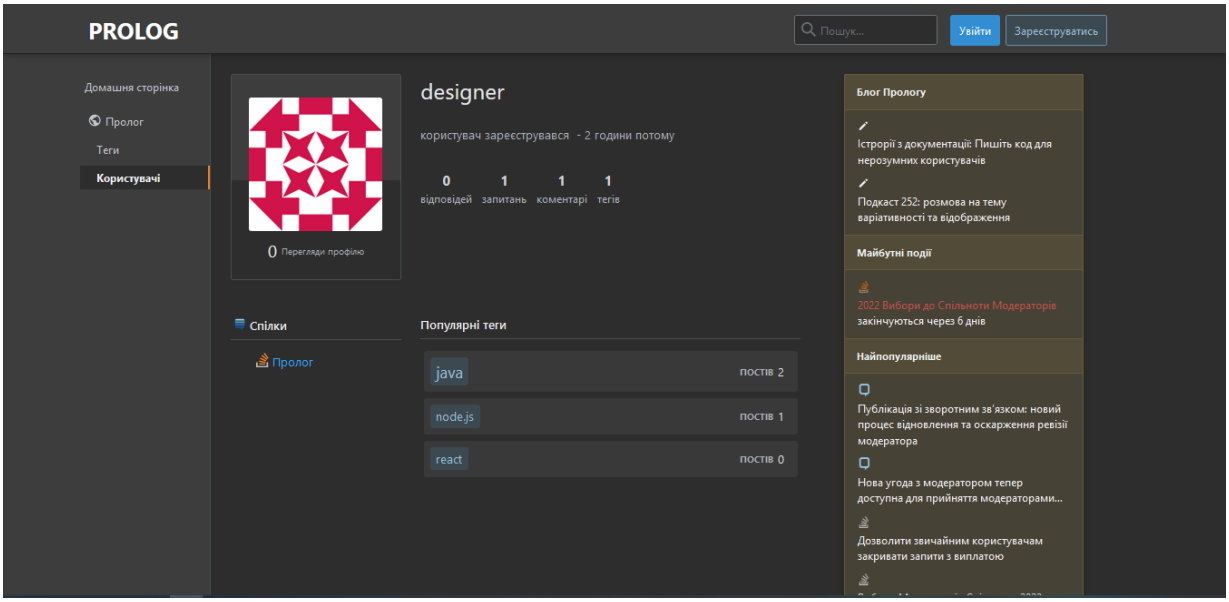

Рис. 3.13. Сторінка профілю користувача

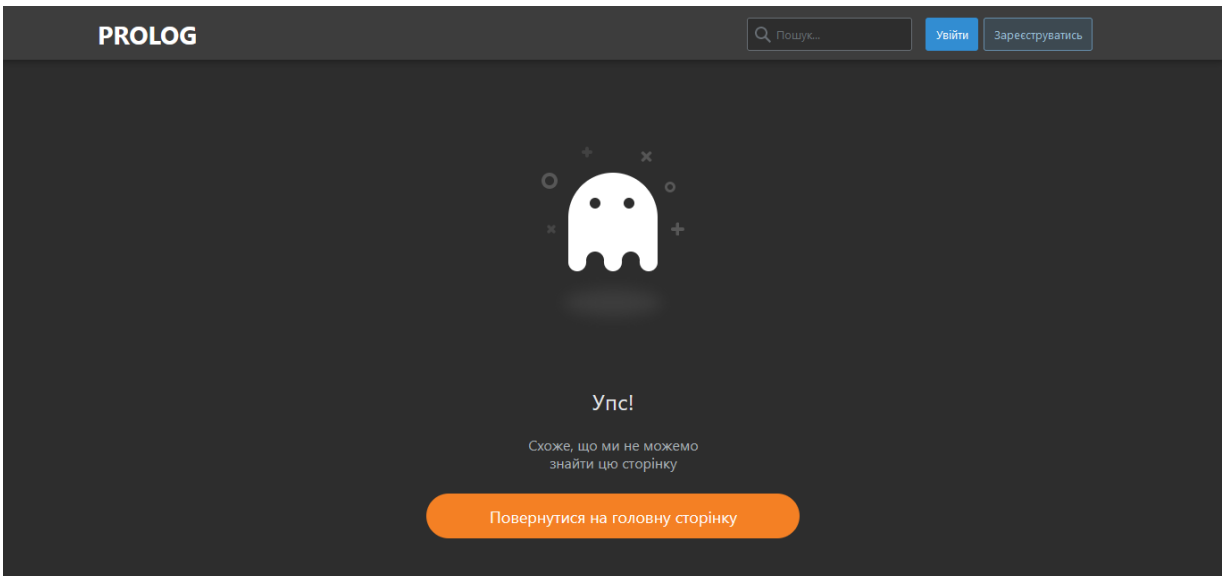

Рис. 3.14. Сторінка 404 помилки

У вікні мікрофронтенду користувач має змогу створити нове обговорення за посиланням «Задати запитання» (рис. 3.15):

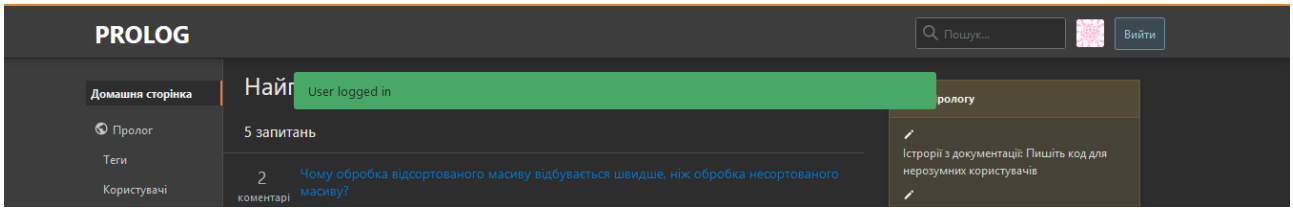

Рис. 3.15. Посилання з особистого профілю на створення оголошення

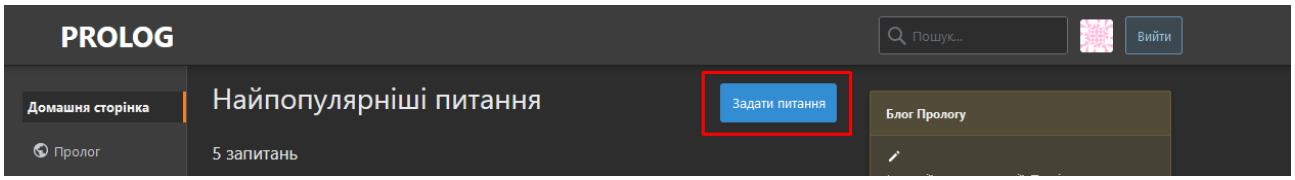

Рис. 3.16. Посилання на створення обговорення

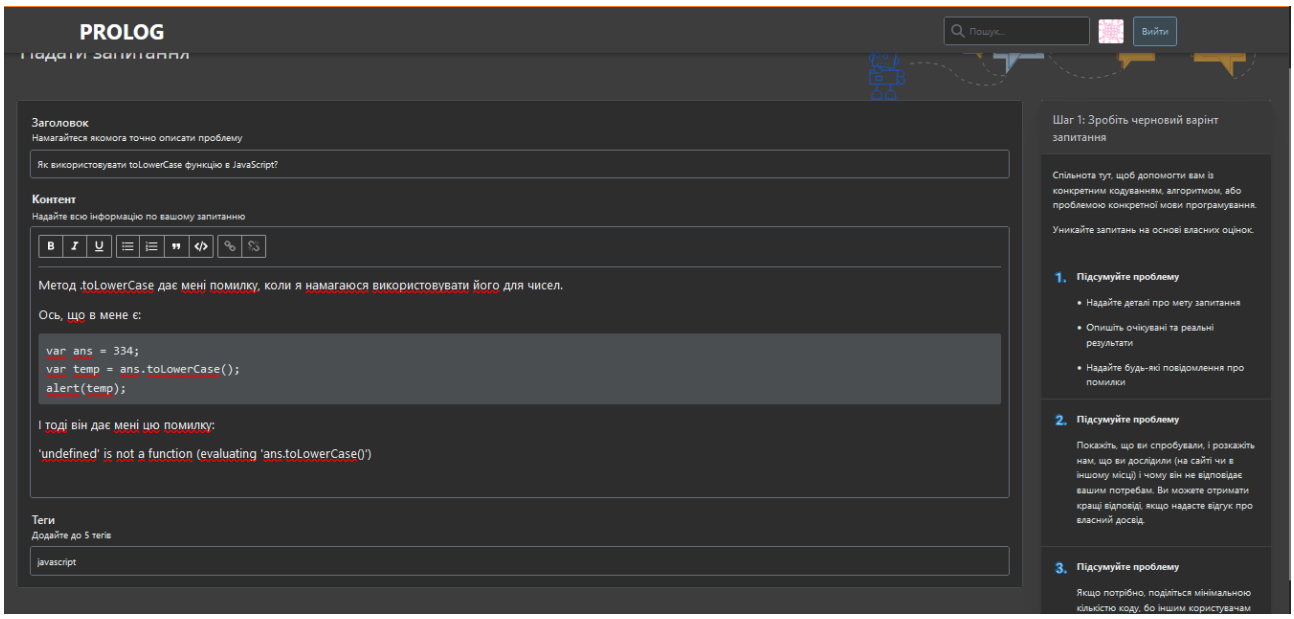

Рис. 3.17. Сторінка для створення оголошення

Після переходу за посиланням користувачу стане доступна форма для створення нового обговорення, що буде мати наступні поля (рис. 3.18):

a. теги, до яких відноситься обговорення – «Теги»
b. Заголовок для обговорення – «Заголовок»
c. Контент обговорення – «Контент»

Після вдалого створення обговорення користувача буде перенесено на сторінку детального опису:

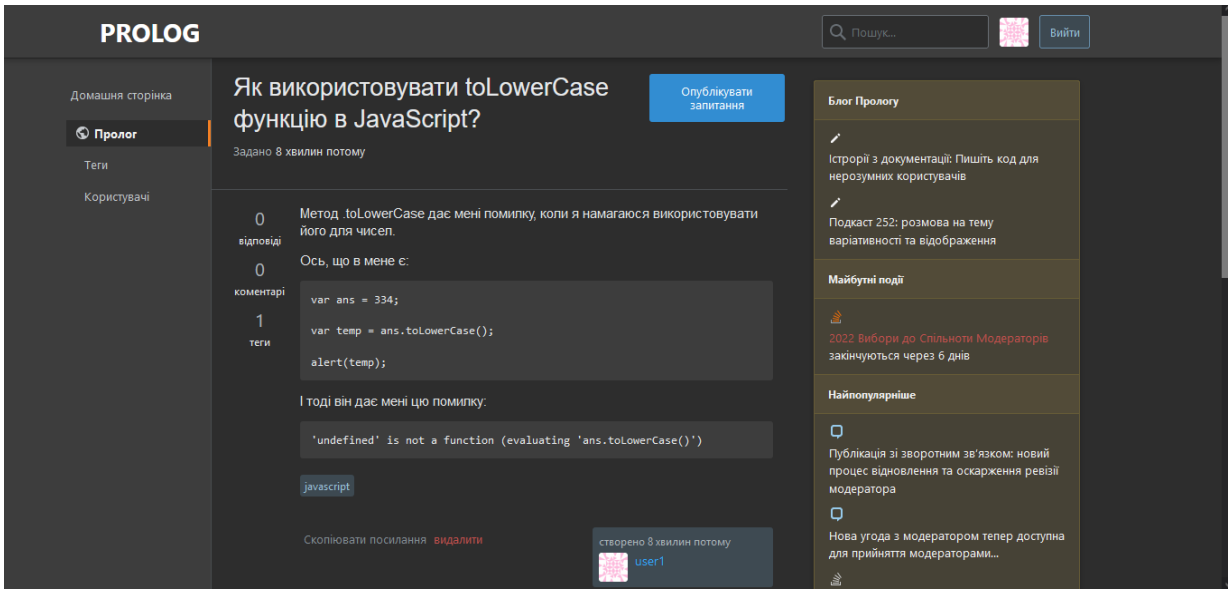

Рис. 3.18. Сторінка зі створеним обговоренням

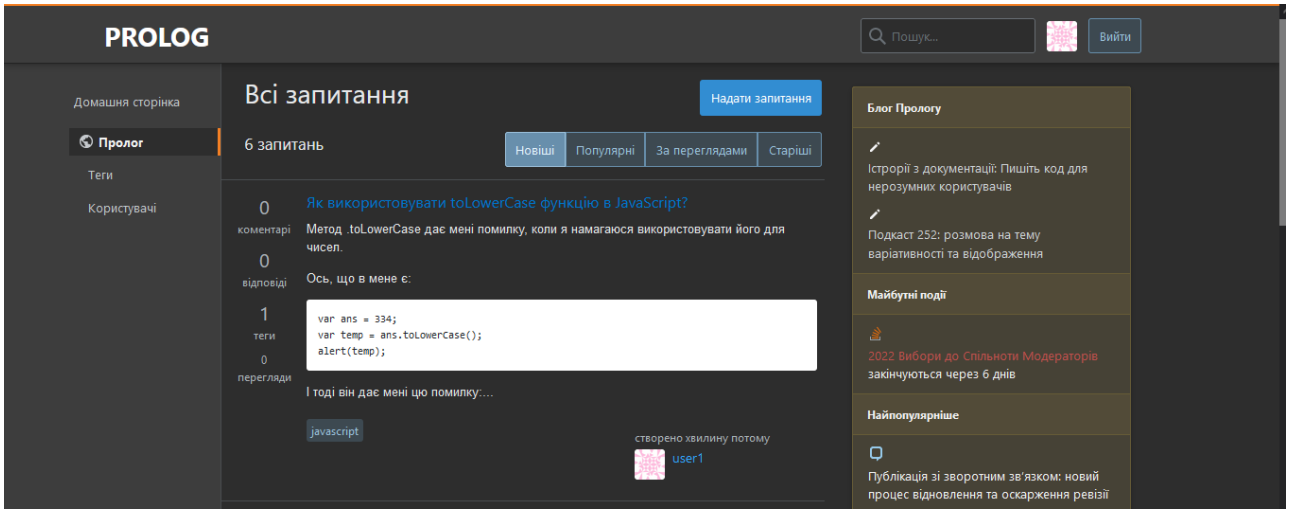

Рис. 3.19. Створене обговорення в рубриці нових обговорень

Рис. 3.20. Нотифікація про успішне видалення створеного обговорення

Після успішної процедури створення нового обговорення користувача буде перенаправлено сторінку з детальним описом цього обговорення, де буде доступна функція видалення. На сторінці з новими обговореннями у списку нове з'явиться зверху, а також буде показане повідомлення про успішне завершення операції створення обговорення в системі форуму (рис. 3.20).

3.4 Висновки

Розробка з використанням сукупності інструментів MERN має значні переваги:

- Дозволяє вести розробку не змінюючи основну мову – Javascript;
- Основний будівельний блок React – це компонент, що підтримує стан коспонентів UI, а також відтворюється оптимізовано самостійно. Розбиття програми на компоненти дозволяє розробникам зосередитися на бізнес логіці та

обґрунтуванні програми. Компоненти взаємодіють зі своїми підкомпонентами шляхом обміну інформацією про стан у вигляді властивостей лише для читання, а з батьківськими – зворотніми викликами;

● Оскільки React може працювати на сервері, його код можна використовувати як у браузерах, так і на серверах. Це означає, що є можливість створювати сторінки на сервері за потреби;

● Усі технології MERN є безкоштовними і мають відкритий сирцевий код. Крім того, MERN Stack підтримується величезною спільнотою розробників, в яких можна питатися поради;

● Стек MERN підтримує архітектуру MVC, що робить процес розробки вільним та легко інтегрується з REST API;

Результати проведених тестувань демонструють швидкість роботи сервера на основі Node.js з використанням БД MySQL. Використання шаблону проєктування ПЗ MVC дозволило створити злагоджену архітектуру вебсайту для забезпечення форуму IT-фахівців «PROLOG» і зберігає можливість масштабування, додавання нової бізнес логіки в найкоротші строки. Ретельне тестування застосунку показало добрі результати – він працює як слід і не має очевидних можливостей зламати логіку розробленої системи.

РОЗДІЛ 4
ЕКОНОМІЧНИЙ РОЗДІЛ

Початкові дані:

1. передбачуване число операторів – 1500;
2. коефіцієнт складності програми – 1,3;
3. коефіцієнт кореляції програми в ході її розробки – 0,07;
4. середня годинна заробітна плата програміста – 150 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,3;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,3;
7. вартість машино-години ЕОМ – 40 грн/год.

4.1. Визначення трудомісткості розробки програмного забезпечення

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \qquad (3.1)$$

де t_o– витрати праці на підготовку й опис поставленої задачі (приймається 160 людино-годин);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C(1 + p), \qquad (3.2)$$

де q – передбачуване число операторів;

C – коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1500 \cdot 1{,}3 \cdot (1 + 0{,}07) = 2086{,}5;$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85)K}, \text{ людино-годин}, \qquad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі, В=1.2 ... 1.5;

K – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 5 до 7 років він складає 1,3.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (В = 1,3). З урахуванням коефіцієнта кваліфікації k = 1,3, отримуємо витрати праці на вивчення опису завдання:

$$t_u = \frac{2086{,}5 \cdot 1{,}3}{75 \cdot 1{,}3} = 27{,}82 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25)K}; \qquad (3.4)$$

$$t_a = \frac{2086,5}{20 \cdot 1,3} = 80,3 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K}; \quad (3.5)$$

$$t_n = \frac{2086,5}{25 \cdot 1,3} = 64,2 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)K}; \quad (3.6)$$

$$t_{отл} = \frac{2086,5}{5 \cdot 1,3} = 321 \text{ людино-годин,}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1,5 \cdot t_{отл}; \quad (3.7)$$

$$t_{отл}^{к} = 1,5 \cdot 321 = 481,5 \text{ людино-годин}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}; \quad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$$t_{др} = \frac{Q}{(15...20)K}; \quad (3.9)$$

$$t_{др} = \frac{2086,5}{20 \cdot 1,3} = 80,25 \text{ людино-годин.}$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}; \qquad (3.10)$$

$$t_{до} = 0{,}75 \cdot 80{,}25 = 60{,}19 \text{ людино-годин.}$$

$$t_{д} = 80{,}25 + 60{,}19 = 140{,}44 \text{ людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

t = 160 + 27,82 + 80,3 + 64,2 + 321 + 140,44 = 793,76 людино-годин.

У результаті, в загальній складності необхідно 793,76 людино-годин для розробки даного програмного забезпечення.

4.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \qquad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \qquad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 793{,}76 \cdot 150 = 119064 \text{ грн.}$$

$З_{МВ}$– Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{М}, \text{ грн,} \qquad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$С_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 321 \cdot 40 = 12840 \text{ грн.}$$

$$K_{ПО} = 119064 + 12840 = 131904 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.} \qquad (3.14)$$

де B_k - число виконавців;

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні Fр=176 годин).

$$T = \frac{793{,}76}{1 \cdot 176} = 4{,}50 \text{ міс.}$$

4.3. Маркетингові дослідження ринку збуту розробленого програмного продукту

В наш час можна досить впевнено сказати, що такий показник як швидкість завантаження веб-сторінки дуже тісно пов’язаний з маркетинговим успіхом будь-якого продукту, адже він напряму впливає на конверсію (відношення загальної кількості користувачів до тієї кількості, що виконали бажану дію), кількість відмов та відвідуваність

Використання в проекті фреймворків та реляційної бази даних MySQL дозволяє зменшити час завантаження додатку, що позитивно впливає на статистику його відвідування користувачами, а також зменшити кількість переходів між сторінками за рахунок динамічного відображення контенту та механізмам повільного фонового завантаження (lazy loading), що дозволяють завантажувати лише ту частину додатку, на якій безпосередньо знаходиться користувач.

4.4. Оцінка економічної ефективності впровадження програмного забезпечення

Так як даний продукт не має економічних показників, ми не маємо можливості розрахувати його економічну ефективність впровадження, але ми отримаємо соціальний ефект від реалізації результатів роботи, який полягає у наступному:

- Ефективна архітектура, яка дозволяє легко масштабувати та інтегрувати інформаційну систему в інші;
- Швидша робота операцій з даними;
- Зменшення часу на вибір бази даних;
- Швидке відображення інтерфейсу на стороні клієнта.

4.5. Висновки

В ході виконання економічної частини було проаналізовано програмний комплекс інформаційної систенми форум IT-фахівців «PROLOG». Час розробки даного програмного забезпечення складає 793,76 людино-годин. Таким чином, очікувана тривалість розробки складе 4,5 місяця при 40 годинному робочому тижні (місячний фонд робочого часу 176 годин), а витрати на створення програмного забезпечення складатимуть 131904 грн.

ВИСНОВКИ

Метою даного проєкту є підвищення ефективності та зручності процесу оперування даними в інформаційних системах типу веб-сайт, шляхом використання реляційної бази даних MySQL та мікросервісного підходу в проєктуванні.

Для досягнення поставленої мети в роботі ставились та вирішувались наступні основні завдання:

- аналіз предметної галузі та постановка задачі;
- уточнення вимог до процесів системи форму «PROLOG»;
- розробка клієнтської частини вебсайту;
- проєктування бази даних для сисетми;
- оптимізація асинхронних процесів в системі;
- розробка алгоритму роботи серверної частини вебсайту;
- розробка алгоритму роботи клієнтської частини;
- прооєктування мікросервісу та мікрофронтенду.

У відповідності до проведеного аналізу, поставлені та вирішені основні функціональні задачі та вимоги до розроблюваного веб-сайту форуму:

- зручний інтуїтивно зрозумілий інтерфейс;
- робота в браузерах на ПК, ноутбуках та мобільних пристроях різних характеристик та конфігурацій під управлінням сімейства ОС Windows, Mac OS, Linux, IOS, Android;
- не бути вимогливою до складу програмних та апаратних засобів;

бути гнучким для модифікацій та оновлення.

У відповідності до проведеного аналізу поставлені основні функціональні задачі та вимоги до розроблюваної електронної дошки оголошень:

- зручний інтуїтивно зрозумілий інтерфейс;

– робота на смартфонах, ПК та ноутбуках різних характеристик та конфігурацій під управлінням сімейства ОС Windows, Mac OS, Linux, IOS, Android;

– не бути вимогливою до складу програмних та апаратних засобів;

– бути гнучкою для модифікацій та оновлення;

– шкидко виконувати операції з даними.

Ціллю кваліфікаційної роботи було оптимізація процесів роботи з даними в веб-застосунках на основі мікросервісної архітектури, а також забезпечення такого архітектурного підходу, який дозволив би легко масштабувати систему та модифікувати її. Ця мета була досягнута шляхом проведення тестів реляційної та об’єктної баз даних на однакових даних та врахочуючи налаштування індексації ключів таблиці.

Узагальнення отриманого практичного досвіду буде корисно для більш глибокого розуміння суті мікросервісної архітектури, стеку MERN та роботи з даними за допомогою MySQL СУБД, їх необхідності та можливостей, а також полегшить процес проєктування подібних систем. Проведене дослідження буде корично для розробників, які створюють веб-сервіси з великою кількість операцій або займаються процесом розгалудження монолітного застосунку на менші сервіси, які легше підтримувати.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бен Форта. Изучаем регулярные выражения. - М.: Диалектика, 2019. - 192 с.

2. Васильев А.Н. Python на примерах. Практический курс по программированию - М.: Наука и Техника, 2016. - 432 с.

3. Майк МакГрат. Программирование на Python для начинающих: [перевод с анг. М. А. Райтмана] - М.: Эксмо, 2015. - 192 с.

4. Марк Лутц. Python. Карманный справочник. - М.: Вильямс, 2014. - 320 с.

5. Методичні рекомендації до виконання кваліфікаційних робіт здобувачами другого (магістерського) рівня вищої освіти спеціальностей 121 «Інженерія програмного забезпечення» та 122 «Комп'ютерні науки» / Б.І. Мороз, О.В. Іванченко, О.В. Реута, О.С. Шевцова; М-во освіти і науки України, Нац. техн. ун-т "Дніпровська політехніка". – Дніпро : НТУ «ДП», 2021. – 57 с.

6. Хольгер Гаст. Объектно - ориентированное проектирование. Концепции и программный код. - М.: Вильямс, 2018. – 1040 с.

7. Початок роботи – React / URL: https://uk.reactjs.org/docs/getting-started.html#learn-react. дата звернення: 09.09.2021

8. React.js Быстрый старт / С. Стефанов. – Пітер: Вид-во "Питер", 2020. – 304 с.

9. XML — Вікіпедія / URL: https://ru.wikipedia.org/wiki/XML. дата звернення: 12.12.2021

10. Masse M. REST API Design Rulebook / M. Masse – O'Reilly Media, 2011. – 116 p.

11. Flanagan D. JavaScript: The Definitive Guide / D. Flanagan – O'Reilly Media, 2011. – 1096 p.

12. Simpson K. You Don't Know JS: Async & Performance / K. Simpson – O'Reilly Media, 2015. – 296 p.

13. Eric Matthes. Python flash Cards. - No Starch Press, 2019. - 101 p.

14. Cantelon M. Nodejs in Action / M. Cantelon – Manning Publications, 2013. – 416 p.

15. Lindley C. DOM Enlightenment: Exploring JavaScript and the Modern DOM / C. Lindley – O'Reilly Media, 2013. – 180 p.

16. JSON — Вікіпедія / URL: https://ru.wikipedia.org/wiki/JSON. дата звернення: 05.12.2021

17. Express - фреймворк веб-приложений Node.js / URL: https://expressjs.com/ru/. дата звернення: 15.11.2021

18. Chinnathambi K. Learning React: A Hands-On Guide to Building Web Applications Using React and Redux / K. Chinnathambi – Manning Publications, 2018. – 304 p.

19. Morgan J. Simplifying JavaScript: Writing Modern JavaScript with ES5, ES6, and Beyond / J. Morgan – Pragmatic Bookshelf, 2018. – 284 P.

20. React: The Virtual DOM / URL: https://www.codecademy.com/articles/react-virtual-dom. дата звернення: 10.10.2021

21. What is API Architecture? – More than you think. / URL: https://api-university.com/blog/what-is-api-architecture/. дата звернення: 13.09.2021

22. Hillar Gatson C. Django RESTful Web Services. - Packt Publishing, 2018. - 426 p.

23. Kronika Jake, Bendoraitis Aidas. Django 2 Web Development Cookbook. - M.: Packt Publishing, 2018. - 544 p.

24. Рефи та DOM – React / URL: https://uk.reactjs.org/docs/refs-and-the-dom.html. дата звернення: 20.01.2021

25. Про проект | Node.js / URL: https://nodejs.org/uk/about/. дата звернення: 01.01.2021.

ДОДАТОК А

КОД ПРОГРАМИ

databaseConfig.sql

```
CREATE DATABASE prolog;
USE prolog;
CREATE TABLE users(
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(100),
    views INT DEFAULT 0,
    created_at TIMESTAMP DEFAULT NOW()
);
CREATE TABLE posts(
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(250),
    body MEDIUMTEXT,
    views INT DEFAULT 0,
    created_at TIMESTAMP DEFAULT NOW(),
    user_id INT NOT NULL,
    FOREIGN KEY(user_id) REFERENCES users(id)
);
CREATE TABLE answers(
    id INT AUTO_INCREMENT PRIMARY KEY,
    body MEDIUMTEXT,
    created_at TIMESTAMP DEFAULT NOW(),
    post_id INT NOT NULL,
    FOREIGN KEY(post_id) REFERENCES posts(id),
    user_id INT NOT NULL,
    FOREIGN KEY(user_id) REFERENCES users(id)
);
CREATE TABLE comments(
    id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    body MEDIUMTEXT,
    created_at TIMESTAMP DEFAULT NOW(),
    post_id INT NOT NULL,
    FOREIGN KEY(post_id) REFERENCES posts(id),
    user_id INT NOT NULL,
    FOREIGN KEY(user_id) REFERENCES users(id)
 );
 CREATE TABLE tags(
   id INT AUTO_INCREMENT PRIMARY KEY,
   tagname VARCHAR(255) UNIQUE,
   description MEDIUMTEXT,
   created_at TIMESTAMP DEFAULT NOW()
);
CREATE TABLE posttag(
   post_id INT NOT NULL,
   FOREIGN KEY(post_id) REFERENCES posts(id),
   tag_id INT NOT NULL,
   FOREIGN KEY(tag_id) REFERENCES tags(id),
   created_at TIMESTAMP DEFAULT NOW(),
   PRIMARY KEY(post_id, tag_id)
);
```

mockData.sql

```
USE prolog;

INSERT INTO users(username, password)
VALUES
("user1", "qwerty1234"),
("devops", "qwerty1234"),
("designer", "qwerty1234"),
("projecter", "qwerty1234"),
("incognito3", "qwerty1234"),
("advancecode", "qwerty1234"),
("codesaver", "qwerty1234");

INSERT INTO posts(title, body, user_id) VALUES ("Як завантажити zip архів за допомогою googledriveapi та java?", "Вітаю! Як завантажити zip архів за допомогою googledriveapi та java? Там написано не дуже багато, що таке driveService? І що треба у нього передавати?", 1);
SET @v1 := (SELECT LAST_INSERT_ID());
INSERT IGNORE INTO tags(tagname, description) VALUES ("java", "Java - це типізована об'єктно-орієнтована мова програмування загального призначення, розроблена компанією Sun Microsystems (надалі придбаною компанією Oracle).");
SET @v2 := (SELECT id FROM tags WHERE tagname = "java");
INSERT INTO posttag(post_id, tag_id) VALUES(@v1, @v2);

INSERT INTO posts(title, body, user_id) VALUES ("Flutter - SEO оптимізація", "Є основний сайт, і є мобільний додаток, написаний на flutter. Є завдання розмістити web-версію flutter за адресою m.example.ru. Постало питання сео оптимізації. Чи є у когось реальний досвід застосування seo у flutter (ssr, пререндер, amp і тд)?
\n
Цікавить не код, а досвід застосування. Результати. Чи варто морочитися? Чи використовувати flutter web лише у сеонезависимых сайтах?", 4);
SET @v1 := (SELECT LAST_INSERT_ID());
INSERT IGNORE INTO tags(tagname, description) VALUES ("flutter", "Flutter - це комплект засобів розробки та фреймворк з відкритим вихідним кодом для створення мобільних додатків під Android та iOS, веб-додатків та нативних додатків під Windows, macOS та Linux з використанням мови програмування Dart, розроблений та розвивається корпорацією Google.");
SET @v2 := (SELECT id FROM tags WHERE tagname = "flutter");
INSERT INTO posttag(post_id, tag_id) VALUES(@v1, @v2);

INSERT INTO posts(title, body, user_id) VALUES ("Чому не видно програми React після збирання?", "Я вирішив зібрати свою програму React, ввів команду npm run build, але після запуску файлу index.js відкривається порожній документ.", 5);
SET @v1 := (SELECT LAST_INSERT_ID());
INSERT IGNORE INTO tags(tagname, description) VALUES ("reactjs", "React — це бібліотека JavaScript для створення інтерфейсів користувача. Він використовує декларативну, засновану на компонентах парадигму і прагне бути як ефективним, так і гнучким.");
SET @v2 := (SELECT id FROM tags WHERE tagname = "reactjs");
INSERT INTO posttag(post_id, tag_id) VALUES(@v1, @v2);

INSERT INTO posts(title, body, user_id) VALUES ("Чому обробка відсортованого масиву відбувається швидше, ніж обробка несортованого масиву?", "Ось фрагмент коду C++, який демонструє дуже
```

своєрідну поведінку. З якоїсь дивної причини сортування даних чудесним чином робить код майже в шість разів швидше.", 3);
SET @v1 := (SELECT LAST_INSERT_ID());
INSERT IGNORE INTO tags(tagname, description) VALUES ("java", "Java - це типізована об'єктно-орієнтована мова програмування загального призначення, розроблена компанією Sun Microsystems (надалі придбаною компанією Oracle).");
SET @v2 := (SELECT id FROM tags WHERE tagname = "java");
INSERT INTO posttag(post_id, tag_id) VALUES(@v1, @v2);

INSERT INTO posts(title, body, user_id) VALUES ("Чи існує унікальний ідентифікатор пристрою Android?", "Чи мають пристрої Android унікальний ідентифікатор, і якщо так, то який простий спосіб отримати до нього доступ за допомогою Java?", 2);
SET @v1 := (SELECT LAST_INSERT_ID());
INSERT IGNORE INTO tags(tagname, description) VALUES ("android", "Android — це мобільна операційна система Google, яка використовується для програмування або розробки цифрових пристроїв (смартфони, планшети, автомобілі, телевізори, Wear, Glass, IoT). Для тем, пов’язаних з Android, використовуйте специфічні для Android теги, такі як android-intent, android-activity, android-adapter тощо. Для інших питань, крім розробки чи програмування, але пов’язаних з платформою Android, скористайтеся цим посиланням: https:// android.stackexchange.com.");
SET @v2 := (SELECT id FROM tags WHERE tagname = "android");
INSERT INTO posttag(post_id, tag_id) VALUES(@v1, @v2);

INSERT INTO answers(body, user_id, post_id)
VALUES
("Просто видаліть розширений віджет, щоб уникнути заповнення доступного місця, і використовуйте батьківський контейнер із фіксованою висотою, такою ж, як значення itemExtent.", 1, 4),
("У React Router >5.1.0 є новий хук useHistory, якщо ви використовуєте React >16.8.0 та функціональні компоненти.", 2, 5),
("Хоча ви праві, що LocalDateTime і LocalDate не містять жодної інформації про часовий пояс, їхні методи тепер використовують часові пояси. Або той, який їм передано, або, якщо ви використовуєте варіант без аргументів, часовий пояс JVM за замовчуванням.", 1, 3);

INSERT INTO comments(body, user_id, post_id)
VALUES
("Мені потрібно більше контексту з цього приводу", 1, 4),
("Можу допомогти", 1, 4),
("Мені потрібно більше контексту з цього приводу", 2, 5),
("Ви впевнені, що не хочете, щоб funcs був масивом, якщо ви використовуєте числові індекси? Просто наголос.", 1, 5),
("Потрібна додаткова інформація", 3, 1);

AskForm.jsx

```
import React, {Fragment, useState, useRef} from 'react';
import {connect} from 'react-redux';
import PropTypes from 'prop-types';
import {addPost} from '../../../redux/posts/posts.actions';
import RichTextEditor from '../../../components/RichTextEditor/RichTextEditor.component';
```

```
import './AskForm.styles.scss';

const AskForm = ({addPost}) => {
  const [formData, setFormData] = useState({
    title: '',
    body: '',
    tagname: '',
  });

  const richTextEditorRef = useRef(null);

  const {title, body, tagname} = formData;

  const onChange = (e) =>
    setFormData({...formData, [e.target.name]: e.target.value});

  const onSubmit = async (e) => {
    e.preventDefault();
    addPost({title, body, tagname});
    setFormData({
      title: '',
      body: '',
      tagname: '',
    });
    richTextEditorRef.current.cleanEditorState();
  };

  const updateConvertedContent = (htmlConvertedContent) => {
    setFormData({...formData, body: htmlConvertedContent});
  };

  return (
    <Fragment>
      <form onSubmit={(e) => onSubmit(e)}>
        <div className='question-form p16 s-card'>
          <div className='question-layout'>
            <div className='title-grid'>
              <label className='form-label s-label'>
                Заголовок
                <p className='title-desc fw-normal fs-caption'>
                  Намагайтеся якомога точно описати проблему
                </p>
              </label>
              <input
                className='title-input s-input'
                type='text'
                name='title'
                value={title}
                onChange={(e) => onChange(e)}
                id='title'
                placeholder='наприклад: Як використовувати стрілкові функції в JS?'
```

```
            required
           />
          </div>
          <div className='body-grid'>
           <label className='form-label s-label fc-black-800'>
            Контент
            <p className='body-desc fw-normal fs-caption fc-black-800'>
             Надайте всю інформацію по вашому запитанню
            </p>
           </label>
           <div className='s-textarea rich-text-editor-container'>
            <RichTextEditor
             ref={richTextEditorRef}
             onChange={updateConvertedContent}
            />
           </div>
          </div>
          <div className='tag-grid'>
           <label className='form-label s-label'>
            Теги
            <p className='tag-desc fw-normal fs-caption'>
             Додайте до 5 тегів
            </p>
           </label>
           <input
            className='tag-input s-input'
            type='text'
            name='tagname'
            value={tagname}
            onChange={(e) => onChange(e)}
            id='tagname'
            placeholder='e.g. (python django array)'
            required
           />
          </div>
         </div>
        </div>
        <div className='post-button mt32'>
         <button
          className='s-btn s-btn__primary'
          id='submit-button'
          name='submit-button'
         >
          Опублікувати запитання
         </button>
        </div>
       </form>
      </Fragment>
    );
};
```

```
AskForm.propTypes = {
  addPost: PropTypes.func.isRequired,
};

export default connect(null, {addPost})(AskForm);
```

answers.js

```
const helperFunction = require('../helpers/helperFunction');

const create = (newAnswer, result) => {
  const query = `INSERT INTO answers(body,user_id,post_id) VALUES(?,?,?);`;

  pool.query(
    query,
    [newAnswer.body, newAnswer.user_id, newAnswer.post_id],
    (err, res) => {
      if (err) {
        console.log('error: ', err);
        result(
          helperFunction.responseHandler(
            false,
            err.statusCode,
            err.message,
            null,
          ),
          null,
        );
        return;
      }
      result(
        null,
        helperFunction.responseHandler(true, 200, 'Answer Added', res.insertId),
      );
    },
  );
};

const remove = (id, result) => {
```

```
  const query = ` DELETE FROM answers WHERE id = ?;`;

  pool.query(query, id, (err) => {
    if (err) {
      console.log('error: ', err);
      result(
        helperFunction.responseHandler(
          false,
          err.statusCode,
          err.message,
          null,
        ),
        null,
      );
      return;
    }
    result(
      null,
      helperFunction.responseHandler(true, 200, 'Answer Removed', null),
    );
  });
};

const retrieveAll = (postId, result) => {
  const query = ` SELECT
                answers.id, post_id, answers.user_id, username, answers.body, answers.created_at
                FROM answers
                JOIN posts ON posts.id = post_id
                JOIN users ON users.id = answers.user_id
                WHERE post_id = ?;`;

  pool.query(query, postId, (err, results) => {
```

```
    if (err || results.length === 0) {
      console.log('error: ', err);
      result(
        helperFunction.responseHandler(
          false,
          err ? err.statusCode : 404,
          err ? err.message : 'There are no answers',
          null,
        ),
        null,
      );
      return;
    }
    result(null, helperFunction.responseHandler(true, 200, 'Success', results));
  });
};

module.exports = {
  create,
  remove,
  retrieveAll,
};
```

Додаток Б

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій
Кафедра програмного забезпечення комп'ютерних систем

ВІДГУК

Керівника
економічної
частини Касьяненко Л.В., к.е.н., доцента каф. ПЕП та ПУ

(прізвище, ім'я, по батькові, вчене звання, посада, місце роботи)

На кваліфікаційну роботу

студента _____
(прізвище, ім'я, по батькові)

курсу II групи _____
спеціальності _____
на тему _____

_____

«___»_____2022 р. _____
(підпис)

Додаток В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Диплом Грудєв.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом Грудєв.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Грудєв.rar	Архів. Містить коди програми
Презентація	
Грудєв.ppt	Презентація кваліфікаційної роботи