

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра Інформатики

ЗВІТ
ПРО ПРАКТИКУ ЗА ТЕМОЮ КВАЛІФІКАЦІЙНОЇ РОБОТИ
в період з 8 квітня по 18 травня 2024 року

Тема індивідуального завдання:

Розробка вебсайту з використанням react та node.js

Звіт розробив
ст. гр. ІТІНФ-20-1
Самченко С. О.
Залікова книжка
№ 20.122.ІТІНФ.0121

Керівник практики
доц. Белова Н. В.
Робота захищена із оцінкою

«__» _____ 2024 року

Харків 2024

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 60 с., 2 табл., 20 рис., 4 дод., 31 джерело.

КЛІЄНТ-СЕРВЕР, ВЕБ ЗАСТОСУНОК, РОЗРОБКА ПОВНОЦІННОГО САЙТУ, РОЗРОБКА ВЕБ БЛОГУ, FULLSTACK, FRONTEND REACTJS, BACKEND NODEJS.

Об'єктом роботи є веб-застосунок, призначений для розміщення інформації яка цікавить певних читачів.

Метою роботи є повноцінна розробка(клієнтської та серверної частини) власного веб блогу, що дозволить людям публікувати інформацію.

Було проведено ретельне дослідження усіх можливих прикладів веб блогів для обміну інформації між їх користувачами , на підставі чого було виявлено їх основні переваги та недоліки. Була побудована інформаційна модель веб блогу для обміну інформації.

У результаті роботи здійснена програмна реалізація веб-застосунку для обміну інформації з великою аудиторією.

CLIENT-SERVER, WEB STACK, DEVELOPMENT OF A FULL SITE, DEVELOPMENT OF A WEBLOG, FULLSTACK, FRONTEND REACTJS, BACKEND NODEJS.

The object of the work is a web site used for posting information to attract previous readers.

The goal of this work is the complete development (client and server parts) of a powerful web blog to allow people to publish information.

A thorough investigation of all possible applications of web blogs for the exchange of information between their users was carried out, from which their main advantages and shortcomings were revealed. An information model was created to create a web blog for information exchange.

As a result of the work, a program implementation of a web site for exchanging information with a large audience was created.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів	4
Вступ.....	5
1 Огляд предметної області та постановка задачі	6
1.1 Огляд архітектури клієнт-сервер	6
1.2 Пояснення терміну веб-застосунок	8
1.2.1 Призначення веб-застосунків та принцип їх роботи.....	8
1.2.2 Чим відрізняється вебсайт від веб-застосунку	10
1.3 Огляд веб-застосунків, призначених для публікації інформації ...	11
1.3.1 Пояснення терміну веб блог	11
1.3.2 Приклади існуючих веб блогів.....	12
1.3.3 Основні характеристики веб блогу	15
1.4 Постановка задачі.....	Error! Bookmark not defined.
2 Моделювання веб-застосунку.....	Error! Bookmark not defined.
2.1 Огляд складових частин для розробки веб-застосунку	18
2.1.1 Моделювання клієнтської частини веб-блогу.....	Error! Bookmark not defined.
2.1.2 Моделювання серверної частини веб-блогу	24
2.2 Огляд та моделювання бази даних	Error! Bookmark not defined.
2.2.1 MobgoDB як система управління БД.....	Error! Bookmark not defined.
2.2.2 Моделювання БД веб-застосунку	26
2.3 Обґрунтування вибору середовища розробки	31
3 Програмна реалізація та демонстрація роботи веб-блогу.....	34
3.1 Програмна реалізація	Error! Bookmark not defined.
3.2 Інструкція користувача	Error! Bookmark not defined.
Висновки.....	Error! Bookmark not defined.
Перелік джерел посилання	Error! Bookmark not defined.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

JS – мова програмування JavaScript

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

СУБД – Система управління базою даних

SQL – Structured query language

БД – База даних

JSON – JavaScript Object Notation

API – Application Programming Interface

ВСТУП

React та Node.js – це два популярні інструменти для розробки веб-сайтів та веб-застосунків. Сьогодні мільйони розробників використовують їх для створення та підтримки своїх сайтів, інтернет магазинів, соціальних мереж, криптобірж та багато іншого.

В даній роботі ці 2 інструменти будуть використовуватися для створення повноцінного блогу та всіх його сучасних можливостей. Блог – це веб-сайт або онлайн-платформа, на якій автори (блогери) публікують свої записи або статті у хронологічному порядку. Ці записи, відомі як "пости", зазвичай відображаються у зворотному хронологічному порядку, з найновішими постами вгорі сторінки. Нині стало досить популярно ділитися своїми новинами у різних мережах. Даний проєкт розроблений приблизно за таким же принципом як і будь-яка сучасний веб блог, але тільки в менших масштабах.

Блоги можуть бути орієнтовані на різні теми та призначені для різних аудиторій. Наприклад, блоги можуть бути присвячені технічним новинам та оглядам, особистому життю, моді, кулінарії, подорожам, музиці, літературі та багато іншого. У блозі, розробленому в цьому проєкті, не буде конкретної тематики і люди зможуть ділитися з читачами будь-якою інформацією.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВА ЗАДАЧІ

1.1 Огляд архітектури клієнт-сервер

Архітектура клієнт-сервер є однією з найпоширеніших моделей веб-розробки, що використовується для побудови сучасних веб-застосунків. Вона полягає в тому, що клієнти (наприклад, веб-браузери або мобільні застосунки) взаємодіють з серверами (наприклад, веб-серверами) для отримання доступу до різних ресурсів та послуг.

На клієнтській стороні розташована вся інтерактивна частина застосунка, з якою користувач безпосередньо взаємодіє. Це включає в себе графічний інтерфейс користувача (UI), який відображається у веб-браузері або мобільному застосунку, логіку застосунка, що керує взаємодією з користувачем та обробляє введення, а також локальні дані, такі як кешовані дані, налаштування та інше. Клієнтська сторона також включає в себе бібліотеки та фреймворки, які використовуються для розробки інтерфейсу, такі як React, Angular або Vue.js.

У цій роботі як раз буде задіяний один із них (React). React – це JavaScript бібліотека, розроблена Facebook для створення інтерфейсів користувача. Вона дозволяє розробникам створювати динамічні та інтерактивні веб-застосунки, поділяючи інтерфейс користувача на компоненти. React використовує Virtual DOM для ефективного оновлення інтерфейсу користувача та забезпечення швидкого відтворення змін.

Серверна сторона відповідає за обробку запитів від клієнтів, виконання бізнес-логіки застосунка та надання необхідних даних та ресурсів. Вона містить в собі бізнес-логіку, яка керує основними функціями застосунка, зберігання даних у базі даних або інших сховищах, API для взаємодії з клієнтами, а також механізми аутентифікації та авторизації, які забезпечують безпеку та конфіденційність. Серверна частина може бути реалізована такими мовами програмування як наприклад: JavaScript(Node.js),

Python(Django), Java(Spring Boot), Ruby(Ruby on Rails), PHP(Laravel), Go(Go Kit), C#(ASP.NET Core), Kotlin(Ktor), Swift(Vapor).

У цій роботі за основу розробки серверної частини буде відповідати мова програмування JavaScript та спеціально створений для цих задач фреймворк Node.js. Node.js – це середовище виконання JavaScript, побудоване на двигуні V8 від Google Chrome. Вона дозволяє розробникам створювати серверні програми на JavaScript. Node.js має потужні інструменти для роботи з мережею, файловою системою та базами даних, що робить його ідеальним вибором для створення веб-серверів та API.

Отже, веб застосунки складаються з інтерфейсу виконаного у вигляді певної сторінки в інтернеті та набору механізмів, які необхідні для реалізації логіки програми. На рисунку 1.1 зображена схема взаємодії між цими компонентами.



Рисунок 1.1 – Схема взаємодії складових веб-застосунку

Основні протоколи, які використовуються для комунікації між клієнтською та серверною сторонами, це HTTP (Hypertext Transfer Protocol) та WebSocket. HTTP використовується для передачі запитів та відповідей між клієнтом та сервером, тоді як WebSocket дозволяє зберігати постійне з'єднання між ними для обміну даними в реальному часі, що дозволяє реалізувати функціонал чату, онлайн-ігор та інших реактивних застосунків.

Безпека є важливим аспектом архітектури клієнт-сервер. Шифрування використовується для захисту конфіденційності даних під час їх передачі через мережу, а також інші заходи безпеки, такі як обробка введення, захист від атак типу CSRF (міжсайтовий міждоменний скриптинг) та XSS (міжсайтовий скриптинг), а також управління доступом.

У цілому, архітектура клієнт-сервер надає гнучку та масштабовану модель для розробки веб-застосунків, яка дозволяє ефективно розділяти відповідальність між клієнтською та серверною сторонами застосунка.

1.2 Пояснення терміну веб-застосунок

1.2.1 Призначення веб-застосунків та принцип їх роботи

Веб-застосунок – це програмне забезпечення, яке працює у веб-браузері та надає користувачам можливість взаємодіяти з ним через Інтернет. Веб-застосунок не потрібно завантажувати на персональний комп'ютер, оскільки він доступний онлайн у режимі реального часу. Користувачі можуть отримати доступ до веб-застосунку через будь-який існуючий браузер. Браузер – це спеціальне програмне забезпечення, що використовується для перегляду веб-сторінок та контенту з цих сторінок у Інтернеті. Найвідомішими на сьогодні браузерами є Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Opera та інші.

Головна мета будь-якого веб-застосунку полягає в створенні зручного, ефективного та інтуїтивно зрозумілого інтерфейсу для користувачів. Вони можуть бути призначені для різноманітних цілей: інформаційні сторінки, калькулятори для розрахунків, інтернет-магазини, онлайн біржі, соціальні мережі, та інші.

Для повноцінного функціонування будь-кого веб-застосунку обов'язково потрібні: сторінка, яка відображає характер та можливості застосунку, сервер та база даних. Як було зазначено в пункті 1.1, на

клієнтській стороні скрипт відповідає за можливості застосунку. При запуску веб-застосунку у браузері завантажується скрипт, який відображає усі графічні елементи, текст та медіафайли з серверу, які повинні бути на сторінці. Коли користувач починає взаємодіяти з такими елементами як авторизація або завантаження файлів до застосунку ця інформація оброблюється у серверній частині і передається до необхідних баз даних.

База даних – це організована колекція даних, яка зберігається та управляється системою, що забезпечує доступ до даних у різний час та для різних користувачів. База даних може бути реляційною(MySQL, PostgreSQL, Oracle) або нереляційною(MongoDB, Cassandra, Redis). Реляційні бази даних організовані у вигляді таблиць з рядками та стовпцями, де дані взаємозв'язані за допомогою ключів. Нереляційні бази даних не використовують табличну структуру і можуть зберігати дані у вигляді документів, графів, або ключ-значення пар.

Веб-застосунки в основному мають короткі цикли розробки та невеликі розробницькі команди. Більшість веб-застосунків розробляються на JavaScript, HTML та CSS для клієнтської частини, та мовах, таких як Python, Java або Ruby для серверної частини. Адміністратори відповідають за управління контентом та співпрацюють з розробниками для визначення стратегії та шляхів розвитку веб-застосунку.

У зв'язку з передовими технологіями сьогодення, розробка різноманітних програм і інструментів для повноцінних веб-застосунків стала досить доступною. Проте, цей процес вимагає від учасників проєкту високого рівня кваліфікації, оскільки робота з багатофункціональними та складними завданнями потребує глибоких знань та досвіду. Таким чином, розробка великих веб-застосунків передбачає залучення висококваліфікованих фахівців з відповідним досвідом і навичками.

В наш час існує велика кількість веб-застосунків у різних сферах життя. Деякі з них включають соціальні мережі, наприклад, Facebook, Instagram, Twitter; електронні комерційні платформи, такі як Amazon, eBay,

Alibaba; різноманітні сервіси з відео та мультимедійним контентом, наприклад, YouTube, Netflix, Spotify; а також інструменти для співпраці та роботи, такі як Google Docs, Microsoft Office Online, Trello. Крім цього, існують веб-застосунки для навчання, ведення особистого бюджету, подорожей, здоров'я та фітнесу, новин та багато інших. Веб-застосунки стали невід'ємною частиною нашого повсякденного життя та діяльності.

1.2.2 Чим відрізняється вебсайт від веб-застосунку

Веб-сайт і веб-застосунок мають суттєві відмінності в своїй природі та призначенні.

Веб-сайт зазвичай є колекцією веб-сторінок, які можуть містити інформацію про певну компанію, організацію, продукт або послугу. Основна мета веб-сайту полягає в тому, щоб надати користувачам інформацію чи розважальний контент, а також можливість зв'язку з власником сайту через контактні форми чи інші засоби комунікації. Веб-сайти можуть бути статичними, коли їхній вміст не змінюється часто, або динамічними, коли вони використовують скрипти або бази даних для генерації контенту на льоту.

У той час як веб-застосунок – це програмне забезпечення, що працює у веб-браузері і надає інтерактивний інтерфейс для користувача. Веб-застосунки можуть мати різноманітні функції та можуть бути спрямовані на вирішення різних завдань, від обробки даних і взаємодії з користувачем до забезпечення конкретних послуг або функціональності. Наприклад, онлайн-магазин, система керування відносинами з клієнтами (CRM), соціальна мережа або веб-платформа для навчання - це приклади веб-застосунків.

Отже, основна відмінність між веб-сайтом і веб-застосунком полягає у їхньому призначенні та функціональності. Веб-сайт зазвичай надає

інформацію та контент, тоді як веб-застосунок надає інтерактивні можливості та функціональність для виконання конкретних завдань.

1.3 Огляд веб-застосунків, призначених для обміну інформації

1.3.1 Пояснення терміну веб блог

Веб-блог – це веб-застосунок, завдяки якому автор чи група авторів регулярно публікують записи стосовно свого життя або конкретної тематики, часто у хронологічному порядку. Ці записи можуть бути короткими думками, оглядами, рецензіями, новинами, особистими історіями чи будь-якою іншою інформацією, яка цікава аудиторії. Блоги можуть охоплювати різноманітні теми, такі як мода, подорожі, технології, кулінарія, мистецтво, політика, бізнес тощо.

Блоги надають можливість авторам взаємодіяти зі своєю аудиторією через коментарі та соціальні мережі, забезпечуючи двосторонню комунікацію. Крім того, вони можуть використовуватися для розвитку особистого бренду, просування бізнесу або надання експертної інформації у певній галузі.

Блоги можна класифікувати за різними критеріями, такими як тематика, формат подачі або цільова аудиторія. Наприклад, за тематикою можна виділити особисті блоги, де автор ділиться своїми думками, досвідом і щоденними подіями в стилі онлайн-щоденника, блоги про подорожі, які містять розповіді, поради та фотографії для планування подорожей, а також технічні блоги, присвячені оглядам гаджетів, навчальним матеріалам з програмування та новинам у сфері ІТ. Серед інших популярних тематик - кулінарні блоги з рецептами і порадами для приготування їжі, блоги про моду та красу з оглядами трендів у стилі та косметиці, політичні блоги з аналізом політичних процесів і державної політики, блоги про фінанси та бізнес, що пропонують поради з особистих фінансів, стратегії розвитку

бізнесу та інтерв'ю з підприємцями, а також блоги про мистецтво і дизайн, які висвітлюють тренди у мистецтві та поради для дизайнерів. Спортивні блоги зазвичай аналізують спортивні події, обговорюють матчі та дають поради з фітнесу і тренувань.

Формат подачі контенту у блогах також варіюється. Текстові блоги містять переважно текстові публікації з додаванням графічного контенту, фотоблоги зосереджені на фотографіях з мінімальним текстовим супроводом, а відеоблоги (влоги) надають інформацію у відеоформаті. Подкасти є аудіоблогами з розмовами або монологам, а мікроблоги пропонують короткі пости, на зразок тих, що публікуються у Twitter, де інформація подається лаконічно.

Крім того, блоги можуть бути орієнтовані на конкретну аудиторію. Блоги для професіоналів приваблюють людей певної професії і пропонують спеціалізований контент. Освітні блоги призначені для навчання та надання інструкцій, а розважальні створені для розваг, зосереджуючись на курйозах, жартах та новинах у сфері шоу-бізнесу.

1.3.2 Приклади існуючих веб блогів

Найпростішим та найпопулярнішим прикладом персонального блогу є досить відома площадка відеохостингу YouTube від компанії Google. Сьогодні кожна людина може створити свій власний блог(канал) на цій платформі та ділитися із аудиторією будь-якою інформацією(у рамках правил YouTube) шляхом публікації відео, звичайних текстових постів та проведенням прямих трансляцій(стріми).

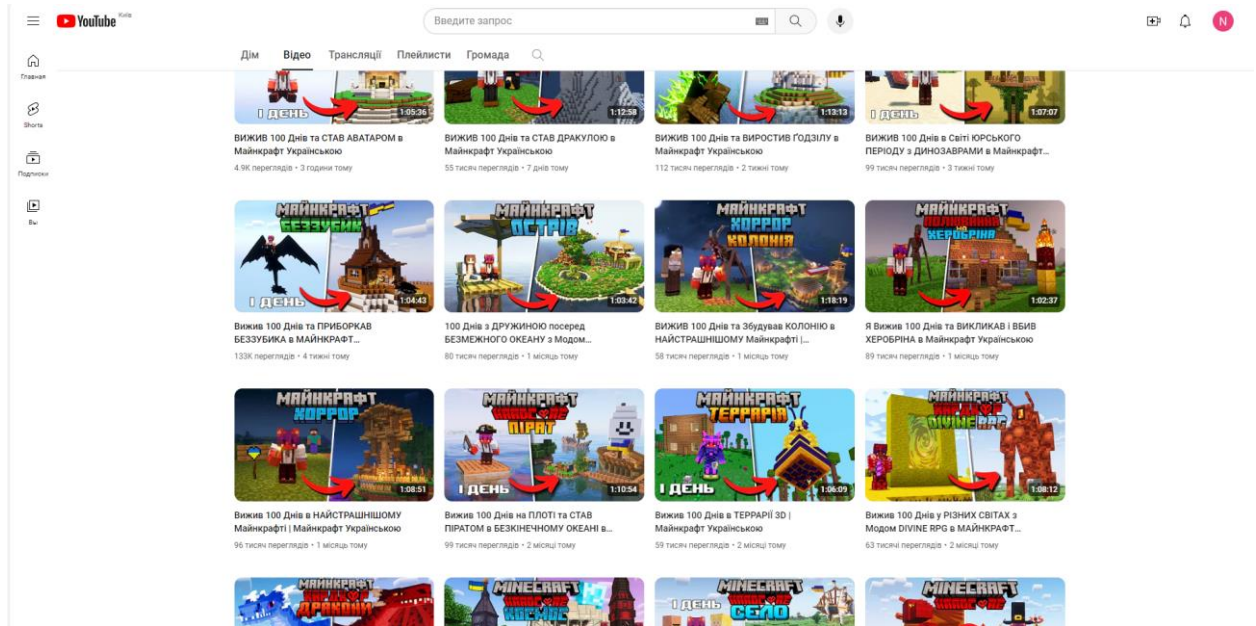


Рисунок 1.2 – Представлення вигляду примітивного каналу на платформі YouTube

Схожим на YouTube, але більш вузькоспрямованим на прямі ефіри, є стриминговий сервіс під назвою Twitch. Так само як і на YouTube, будь-яка людина може створити власний блог(канал) та почати публікувати інформацією для інших користувачів площадки. На цій платформі блогери в основному діляться своїм контентом та новинами з аудиторією через прямі трансляції. Такий формат блогу є головною задумкою та особливістю цього веб застосунку. І хоча існує багато подібних веб ресурсів(у тому ж числі і YouTube), переважна більшість блогерів та представників аудиторії, яка зацікавлена саме у такому форматі подачі та сприйняття інформації, віддають перевагу саме Twitch, через зручність платформи та велику кількість способів інтерактивної взаємодії з аудиторією.

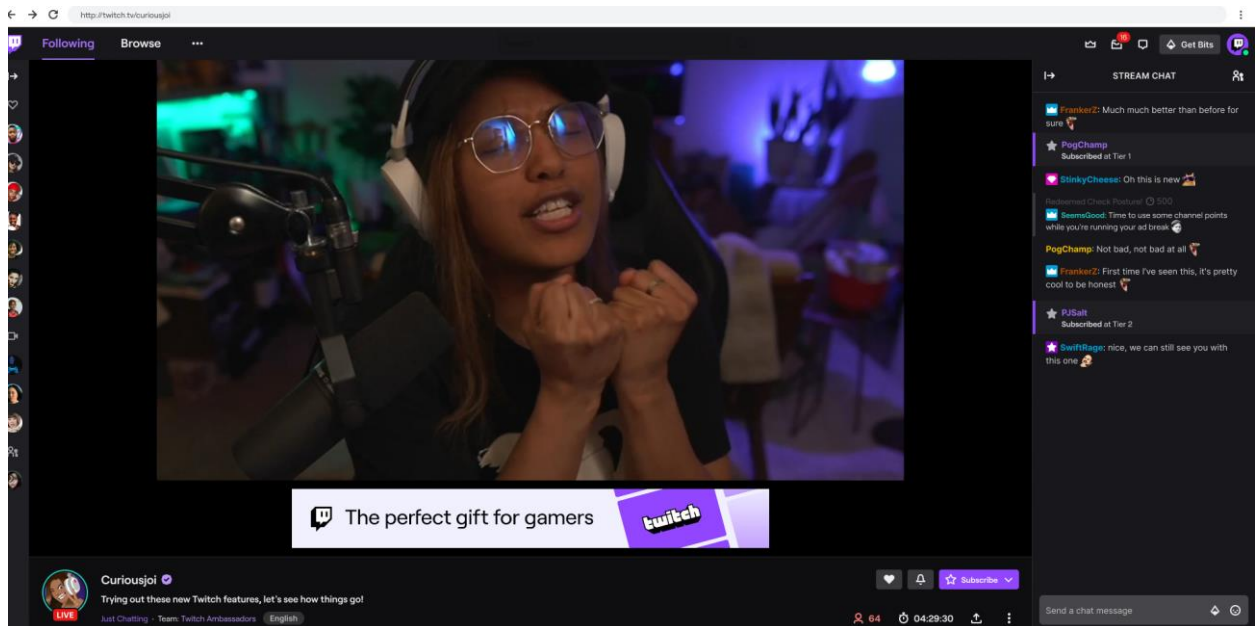


Рисунок 1.3 – Наглядний приклад того, як влаштований обмін контентом з аудиторією на платформі Twitch

Окрім YouTube та Twitch існує безліч відомих подібних веб застосунків, які надають своїм юзерам можливість ділитися корисною або розважальною інформацією з іншими. Така можливість може бути як основною ідеєю та функцією платформи(як наприклад YouTube, Twitch, TikTok, Instagram) так і додатковою можливістю соцмережі або месенджеру(як наприклад канал новин у Telegram або група, присвячена певній тематиці у Facebook).

Але не всіх людей влаштовують принципи, алгоритми, правила, функціонал або просто зовнішній вигляд популярних веб блогів. Через це деякі блогери створюють власні подібні веб застосунки, за своїми перевагами та смаками, на яких вони можуть розміщувати будь який контент у рамках закону з повним налаштуванням зовнішнього вигляду, алгоритмів публікації та обробки їх персонального веб блогу. На Рисунку 1.4 приведено приклад одного із таких веб застосунків.

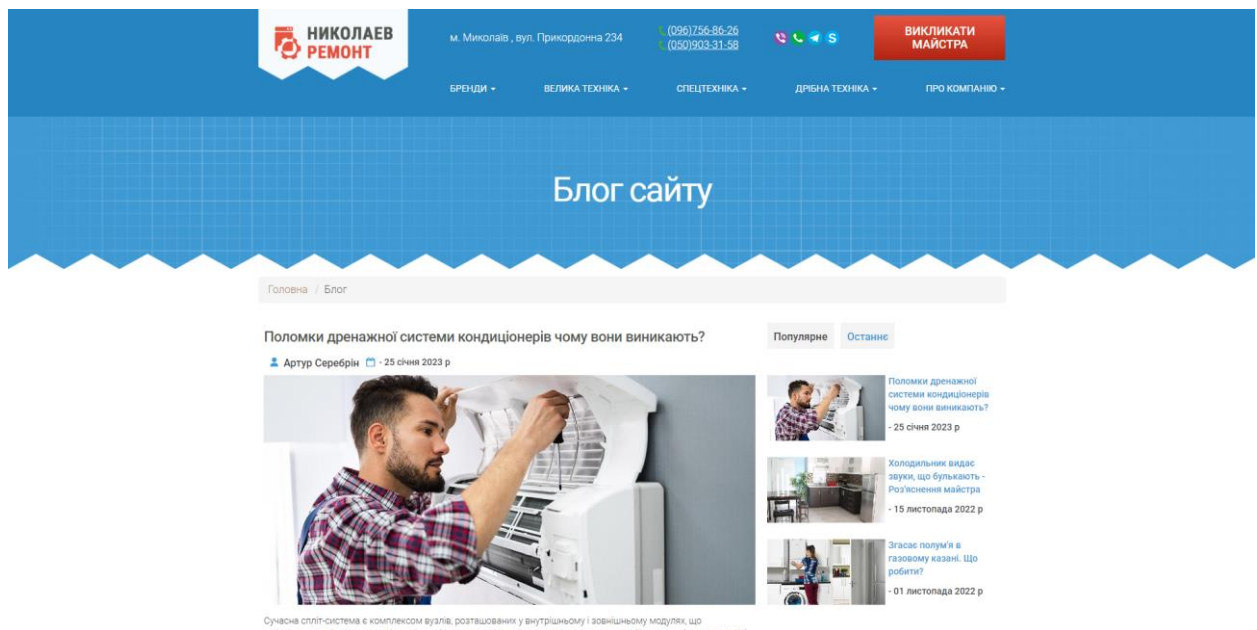


Рисунок 1.4 – Приклад власно-розробленого веб блогу

Цей приклад є досить детальною демонстрацією того, як автор може налаштувати кожний елемент свого веб застосунку. Тематикою даного блогу є ремонт несправної побутової техніки. Окрім того, що автор розповідає своїм читачам про свій досвід роботи, чому відбуваються випадки поломок, як можна самому вирішити подібну проблему та як такої проблеми запобігти – він також надає людям свої послуги у питанні відновлення зламаних пристроїв. Окрім цього помітно, що блог має унікальний дизайн, зручну навігацію, контактну інформацію та незвичайну сітку зі статтями, присвяченій головній тематиці блогу. У сукупності такі елементи сприяють гарному враженню, бажанню ознайомитися з інформацією та викликають довіру відвідувачів ресурсу, що безпосередньо є гарним результатом.

1.3.3 Основні характеристики веб блогу

Виходячи з даних, отриманих при огляді та аналізі існуючих веб блогів до їх основних характеристик можна віднести такі аспекти як:

- Тематика, яка цікавить певну групу людей;

- Має набір публікацій, які зазвичай включають статті, пости, рецензії, інструкції або інші типи контенту;
- Може бути ведений однією особою або групою авторів;
- Веб-блоги часто мають можливість коментування, що дозволяє читачам висловлювати свої думки та відгуки щодо публікацій;
- Блог можна вести як на готовому ресурсі(але погоджуватися із правилами, які встановлені власниками) так і розробити власний з нуля(де правила також встановлюються нами і регулюються тільки законом);
- Дизайн блогу може бути різноманітним і відображати його тематику, стиль та індивідуальність автора.

1.2 Постановка задачі

Таким чином, розробка власного веб блогу є актуальною задачею, тому Після проведення аналізу предметної області, необхідно поставити конкретні завдання для розробки веб-застосунку, який надає можливість ділитися з читачами певною інформацією.

Об'єктом роботи є веб-застосунок, призначений для розміщення інформації яка цікавить певних читачів.

Метою роботи є повноцінна розробка(клієнтської та серверної частини) власного веб блогу, що дозволить людям публікувати інформацію.

Для досягнення мети необхідно реалізувати наступні завдання:

- ознайомитися з існуючими на сьогодні веб блогами та з принципом їх роботи;
- розробити концепцію створення веб-застосунку;
- опанувати необхідні для розробки технології;
- створити логіку здійснення публікацій та реалізувати зворотній зв'язку;

- розробити клієнтську частину застосунку;
- створити серверну частину;
- реалізувати зв'язок застосунку із БД.

2 МОДЕЛЮВАННЯ ВЕБ-ЗАСТОСУНКУ

2.1 Огляд складових частин для розробки веб-застосунку

2.1.1 Моделювання клієнтської частини веб-блогу

Фронтенд, або клієнтська частина веб-застосунку, є тим, що користувач бачить та з чим він активно взаємодіє через веб-застосунок. Основні складові структури фронтенду включають в себе HTML, CSS та JavaScript (рисунок 2.1).

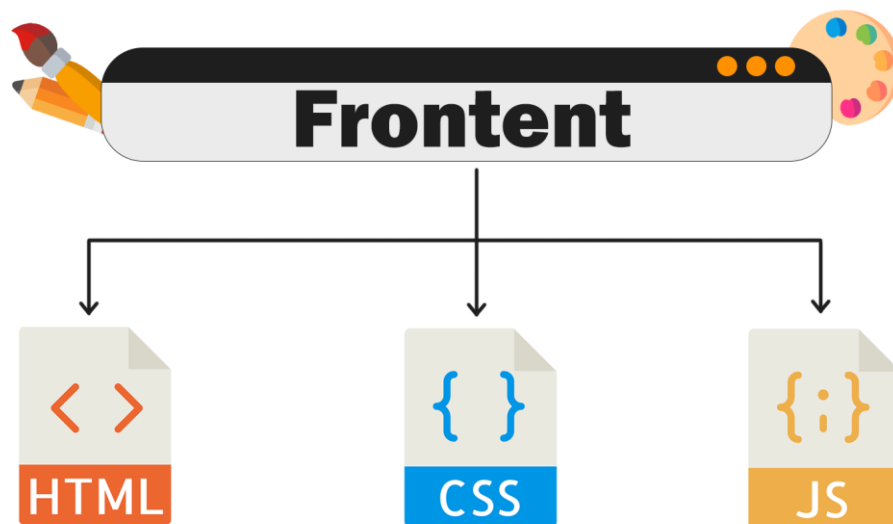


Рисунок 2.1 – Структурні складові фронтенд розробки

Проте, у сучасній веб-розробці, дуже рідко можна зустріти сайт, який був створений саме на чистих HTML, CSS та JavaScript. Сьогодні, верстальники віддають перевагу більш сучасним та зручним інструментам для цих задач, які називаються фреймворки та бібліотеки. Вони можуть відрізнятися синтаксисом, структурою, логікою, побудовою, але ціль у них одна – спростити роботу розробникам. Найвідомішими фреймворками та

бібліотеками у сфері фронтенд є React, Vue та Angular. У цій роботі буде використовуватися та описуватися React.

React – це потужна бібліотека JavaScript, яка дозволяє розробляти веб-застосунки з високою швидкістю та динамічними інтерфейсами. Основними концепціями React є компонентний підхід, віртуальний DOM та односторонній потік даних.

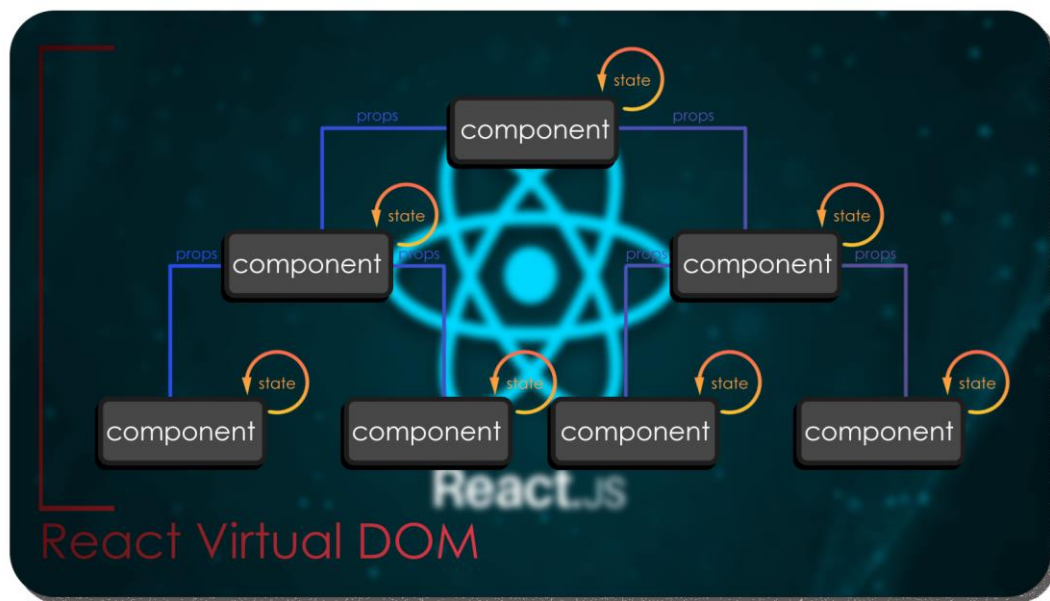


Рисунок 2.2 – Логіка React застосунку

У React, все обертається навколо компонентів. Компоненти – це незалежні, перевикористовувані блоки, які складаються з HTML-подібної розмітки та логіки JavaScript. Кожен компонент може приймати вхідні дані (props) і повертати елементи візуалізації, які React потім рендерить в браузер.

У відмінності від традиційного DOM-моделі, React використовує віртуальний DOM, який є копією реального DOM. При зміні стану компонентів, React вносить зміни в віртуальний DOM, а потім порівнює його з реальним DOM та виконує мінімальну кількість маніпуляцій для оновлення сторінки. Це робить процес рендерингу ефективнішим та швидшим.

Структура React додатку зазвичай складається з кореневого компонента, який рендериться в DOM, та його дочірніх компонентів. Компоненти можуть вкладатися один в одного, утворюючи дерево компонентів. Кожен компонент може мати власний стан та властивості, які визначають його поведінку та вигляд.

Поняття стану (state) в React використовується для зберігання даних, які можуть змінюватися в процесі роботи додатку. Стан може бути оновлений зовні (за допомогою подій) або внутрішньо (відповідно до логіки компоненту). Властивості (props) використовуються для передачі даних від батьківського компонента до дочірніх.

Крім того, React надає можливість використовувати хуки (hooks), які дозволяють використовувати стан та інші функціональності React у функціональних компонентах. Це робить React більш гнучким та потужним інструментом для розробки веб-застосунків.

У таблиці 2.1 приведено декілька вагомих критерій для порівняння React зі звичайною веб-розробкою.

Таблиця 2.1 – порівняння React з простою версткою

Критерій	React	Звичайна верстка
Продуктивність	Використання віртуального DOM	Відсутність віртуального DOM
Масштабованість	Легке масштабування	Складніше масштабування
Підтримка	Велика та активна спільнота	Менш активна спільнота
Функціональність	Компонентна архітектура, JSX, управління станом та пропсами	Відсутність таких можливостей
Швидкість	Швидше розробка	Повільніша розробка

розробки	завдяки компонентній архітектурі та інструментам для розробки	
Спільнота	Активна спільнота розробників, яка забезпечує підтримку та розвиток	Менш активна спільнота, менше ресурсів для підтримки
Навчання	Велика кількість документації, туторіалів та онлайн-курсів	Менш доступних ресурсів для навчання, менше інструкцій
Продуктивність	Швидкіше виконання завдань завдяки оптимізації та ефективному управлінню станом	Повільніше виконання завдань, менша оптимізація
Гнучкість	Більша гнучкість у побудові складних інтерфейсів та компонентів	Обмежена гнучкість у побудові складних інтерфейсів
Тестування	Зручніше тестування компонентів та стану за допомогою інструментів, таких як Jest або Enzyme	Обмежені можливості тестування, менше інструментів

Але за допомогою React створюється лише оболонку сайту, щоб стилізувати сторінку та примінити до неї власний дизайн використовується CSS. Проте для CSS також існують свої вдосконалення, наприклад SCSS, який також використовується у даній роботі.

SCSS – це препроцесор CSS, який розширює функціонал та можливості до звичайного CSS. Основна перевага SCSS полягає в його розширеній синтаксичній можливості, яка включає змінні, вкладені правила, міксини, спадковість та інші функції, що дозволяють писати CSS більш організовано і зручно.

Змінні дозволяють зберігати значення, такі як кольори чи розміри, в одному місці і використовувати їх потім в усій таблиці стилів, що робить код більш читабельним і легко змінюваним. Вкладені правила дозволяють вкладати одні стилі всередині інших, що полегшує організацію та структуру коду. Міксини дозволяють створювати набір стилів, які можна використовувати повторно для різних елементів.

Крім цього, SCSS підтримує оператори, функції, умовні конструкції та інші функціональні можливості, які розширюють можливості CSS і полегшують розробку та обслуговування стилів. Після написання коду на SCSS, його можна компілювати в звичайний CSS, який використовується для відображення на веб-сторінці. У таблиці 2.2 приведено декілька вагомих критеріїв для порівняння SCSS та CSS.

Таблиця 2.2 – порівняння SCSS та CSS

Критерій	CSS	SCSS
Синтаксис	Звичайний CSS	Розширений синтаксис SCSS
Змінні	Немає підтримки змінних	Підтримка змінних
Вкладеність	Обмежена можливість вкладеності	Безліч вкладених правил
Міксини	Відсутність можливості використання міксинів	Можливість використовувати міксини
Імпорт	Обмежена підтримка імпорту CSS файлів	Повна підтримка імпорту
Наслідування	Обмежена підтримка	Повна підтримка наслідування

	наслідування	
Підтримка	Підтримується всіма браузерами	Підтримується всіма браузерами
Розширення	Можливість використовувати розширені функції та селектори	Підтримка розширених функцій та селекторів

Також для стилізації використовуються шаблони з бібліотеки Material-UI. Material-UI (або MUI) – це бібліотека компонентів інтерфейсу користувача для React, яка базується на дизайні Google Material Design. Вона надає набір готових до використання компонентів і стилів, що допомагає розробникам швидко створювати стильні та сучасні веб-застосунки.

Material-UI пропонує широкий вибір різноманітних компонентів, таких як кнопки, форми, таблиці, модальні вікна, навігаційні панелі та багато інших. Всі вони мають чистий та консистентний дизайн, який дотримується принципів Material Design.

Однією з переваг Material-UI є його простота використання. Завдяки прописаній документації та демонстраційним прикладам, розробники можуть швидко оволодіти бібліотекою та почати створювати додатки. Крім того, Material-UI підтримує адаптивний дизайн, що дозволяє створювати додатки, які працюють на різних пристроях і розмірах екрану.

Бібліотека також надає можливість налаштування стилів компонентів за допомогою тем і стилів. Це дозволяє розробникам легко змінювати вигляд та поведінку компонентів, відповідно до потреб їх додатка.

Узагальнюючи, Material-UI є потужною та зручною бібліотекою компонентів для React, яка допомагає розробникам створювати красиві та функціональні веб-застосунки швидко та ефективно.

Таким чином, для розробки структури веб-застосунку буде використовуватися бібліотека React: вибір обумовлений її широким

функціональним спектром, що спрощує розробку та підтримку клієнтської частини програми. Для стилізації застосунку буде використовуватися SCSS, тому що цей препроцесор пришвидшує та значно спрощує процес стилізації вебсайту та бібліотека Material-UI, в якій є всі необхідні шаблони для даного проєкту.

2.1.2 Моделювання серверної частини веб-блогу

Бекенд – це та частина програмного забезпечення, яка забезпечує обробку даних та виконання логіки на сервері. В контексті веб-розробки, бекенд відповідає за всі серверні операції, що стосуються обробки запитів, зберігання та управління даними, а також взаємодію з іншими системами.

Основні функції бекенду включають управління базами даних, реалізацію бізнес-логіки додатку, обробку та перевірку даних, аутентифікацію та авторизацію користувачів, обробку запитів від клієнтів та відправлення їм відповідей. Крім того, бекенд може включати в себе такі компоненти, як кешування, розподілений обчислювальний потік, обробку файлів, роботу з мережевими службами та інші.

Технології, що використовуються для створення бекенду, можуть включати мови програмування, такі як JavaScript (за допомогою Node.js), Python, Ruby, Java, PHP, а також фреймворки та інструменти для розробки веб-застосунків, такі як Express.js для Node.js, Django для Python, Ruby on Rails для Ruby, Spring для Java та Laravel для PHP. Бекенд інтегрується з фронтом (клієнтською частиною) для створення повноцінного веб-застосунку, що надає користувачеві доступ до всіх необхідних функціональних можливостей.

В контексті розробки проєкту було обрано мову програмування, яка в змозі забезпечити реалізацію поставлених питань, а саме – Node.js.

Node.js є платформою, яка дозволяє виконувати JavaScript на серверному боці. Це означає, що розробники можуть створювати веб-

застосунки, які працюють на сервері, використовуючи JavaScript. Однією з ключових особливостей Node.js є можливість використання JavaScript для програмування як на клієнтській, так і на серверній стороні, що спрощує розробку та збільшує переносимість коду між фронтом і бекендом.

Node.js використовує асинхронний та подійно-орієнтований підхід до програмування. Це означає, що код Node.js виконується неблокуючим способом, що дозволяє обробляти багато запитів та подій одночасно. Це робить його особливо ефективним для створення високонавантажених веб-застосунків, таких як веб-сервери та API.

Однією з ключових переваг Node.js є його ефективність та масштабованість. Вбудована підтримка механізмів масштабування, таких як кластеризація та багатопоточність, дозволяє легко розгортати та масштабувати додатки для обробки великого обсягу запитів.

Крім того, Node.js має широкий вибір бібліотек та фреймворків, які полегшують розробку веб-застосунків. Наприклад, Express.js - це популярний фреймворк для створення веб-серверів та API з використанням Node.js.

Загалом, Node.js є потужною та універсальною платформою для створення різноманітних веб-застосунків, яка дозволяє розробникам швидко та ефективно створювати сучасні та масштабовані додатки за допомогою JavaScript.

2.2 Огляд та моделювання бази даних

2.2.1 MobgoDB як система управління БД

Для розробки бази даних веб-застосунку було обрано MongoDB – це документ-орієнтована база даних, яка використовує JSON-подібні документи для зберігання даних. Основна особливість MongoDB полягає в тому, що вона дозволяє зберігати дані у вигляді документів, які можуть бути складними структурами даних, а не просто таблицями зі стовпцями, як у традиційних реляційних базах даних.

У MongoDB дані зберігаються у вигляді колекцій, які містять документи. Кожен документ може мати різну структуру, але зазвичай вони містять ключі та значення у форматі JSON. Це дозволяє зберігати дані більш гнучко і при цьому забезпечує швидкий доступ до них.

Однією з переваг MongoDB є гнучкість схеми даних. У відміну від реляційних баз даних, де схема даних повинна бути заздалегідь визначена, в MongoDB можна зберігати документи з різними структурами у тій же колекції. Це робить її ідеальним вибором для проектів, де структура даних може змінюватися з часом.

Ще однією перевагою MongoDB є горизонтальне масштабування. MongoDB дозволяє розподіляти дані по кількох серверах (кластеризація), щоб забезпечити підтримку великих обсягів даних та високу доступність системи.

В цілому, MongoDB є потужною та гнучкою базою даних, яка підходить для різноманітних проектів, особливо тих, де важливо швидко реагувати на зміни в структурі даних та масштабувати систему з ростом обсягів інформації.

2.2.2 Моделювання БД веб-застосунку

У застосунку передбачається збереження нових користувачів для авторизації та звірка зареєстрованих для авторизації, тому виникає необхідність у відповідній для цього таблиці, яка називається «Users».

Для реєстрації та авторизації користувачу необхідно надати наступні данні:

- fullName: Це поле призначене для створення імені кожного користувача. Людина може внести як своє справжнє ім'я, так і використати

псевдонім, яким наіменовує себе у інтернеті. Дане поле є строковим(string) та має ключ «required: true», що робить поле обов’язковим для заповнення.

– email: Поле, яке відповідно використовується для вводу електронної адреси користувача. Також являє собою строковий тип даних та є обов’язковим для заповнення. Окрім цього має флаг «unique: true», що означає, що поле повинно бути унікальним, так як два та більше юзера не можуть мати однакову пошту.

– passwordHash: Поле, яке зберігає згенерований користувачем пароль від його аканту. Також являє собою строковий тип даних та є обов’язковим для заповнення.

– avatarUrl: Так як об’єктом даної роботи виступає веб-блог, у користувачів має бути можливість встановити зображення для свого профілю. Саме для реалізації цієї можливості існує дане поле. Незважаючи на своє призначення, воно є строковим, і зберігає наіменування файлу, розташованого на бекенді. Воно також є необов’язковим, так як не кожна людина може захотіти встановити собі аватар.

– id: Поле ідентифікатора є автоматичним, та створюється самою БД. Воно є унікальним, строковим та ключовим (PRIMARY KEY).

Таким чином, у таблиці «Users», надана користувачами інформація буде зберігатися у полях fullName, email, passwordHash, avatarUrl та id.

Скрипт відношення представлений у лістингу 2.1.

Лістинг 2.1 Реалізація таблиці «Users»:

```
const UserSchema = new mongoose.Schema(
  {
    fullName: {
      type: String,
      required: true,
    },
    email: {
      type: String,
```

```

    required: true,
    unique: true,
  },
  passwordHash: {
    type: String,
    required: true,
  },
  avatarUrl: String,
},
{
  timestamps: true,
},
);
export default mongoose.model('User', UserSchema);

```

Оскільки веб-блог передбачає перед собою мету ділитися якоюсь інформацією з аудиторією, виникає необхідність десь зберігати данні, стосовно кожного такого посту. Тому доцільно для реалізації такого функціоналу створити окрему таблицю «Posts», яка буде зберігати всю необхідну для цього інформацію. У ній будуть знаходитися наступні поля:

- title: Поле, яке зберігає в собі інформацію про заголовок публікації. Воно є строковим та обов’язковим.
- text: Поле, у якому знаходиться основний контент публікації – інформація. Воно є строковим, обов’язковим та унікальним, щоб юзери не цупили один у одного контент та не публікували одну й та саму інформацію на блозі.
- tags: Щоб користувачам блогу було легше та зручніше знаходити інформацію, в якій вони зацікавлені, функціонал веб-застосунку надає можливість авторам використовувати теги(ключові слова). Вони є не

обов'язковим атрибутом і представляють собою масив, перерахованих строкових значень.

- `viewsCount`: Для статистики та розуміння, являється опублікована інформація цікавою та корисною, кожен пост, має лічильник переглядів цілісного типу, значення якого по замовчуванню дорівнює нулю.

- `user`: Поле, яке є зовнішнім ключем (FOREIGN KEY), та зсилається на данні (`id`) автора публікації, які знаходяться у таблиці «Users». Звичайно, поле є обов'язковим.

- `imageUrl`: Необов'язкове поле, яке існує на випадок, якщо автор захоче виділитися гучним заголовком. По аналогії з полем «`avatarUrl`» із таблиці «Users» – також є строковим і зберігає в собі найменування картинки.

Слід зазначити, що дане відношення також має ключ «`timestamps: true`». У MongoDB, параметр «`timestamps: true`» в схемі документа вказує, що колекція, до якої відноситься документ, буде автоматично додавати два додаткові поля: «`createdAt`» і «`updatedAt`». Ці поля автоматично оновлюються MongoDB при створенні нового документа (додавання в колекцію) і при оновленні існуючого документа (зміни в колекції).

Поле «`createdAt`» містить дату та час створення документа, а «`updatedAt`» містить дату та час останнього оновлення документа. Ці поля можуть бути корисні для відстеження часу створення та оновлення даних в базі даних, і вони додаються автоматично без потреби вручну встановлювати значення для них.

Таким чином, у таблиці «Posts», інформація про публікації авторів буде зберігатися у полях `title`, `text`, `tags`, `viewsCount`, `user` та `imageUrl`.

Скрипт відношення представлений у лістингу 2.2.

Лістинг 2.2 Реалізація таблиці «Posts»:

```
const PostSchema = new mongoose.Schema(
  {
    title: {
```

```
    type: String,
    required: true,
  },
  text: {
    type: String,
    required: true,
    unique: true,
  },
  tags: {
    type: Array,
    default: [],
  },
  viewsCount: {
    type: Number,
    default: 0,
  },
  user: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true,
  },
  imageUrl: String,
},
{
  timestamps: true,
},
);
export default mongoose.model('Post', PostSchema);
```

Таким чином, на підставі створених відношень буде створена наступна БД веб-застосунку (рисунок 2.3).

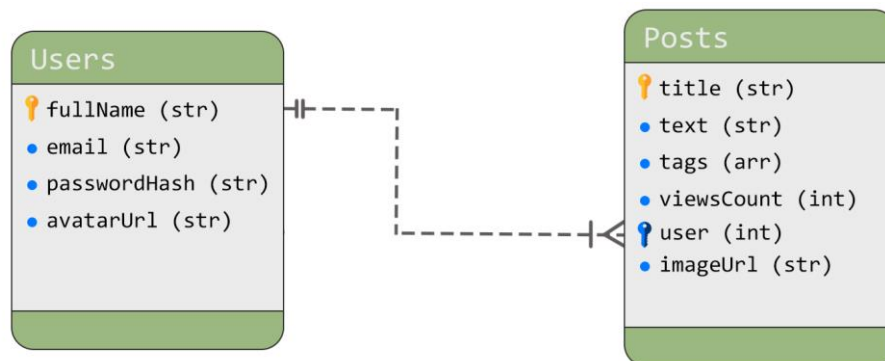


Рисунок 2.3 – Візуальне представлення моделі БД

2.3 Обґрунтування вибору середовища розробки

Для досягнення більшої ефективності та максимально комфортної роботи потрібно вибрати відповідне середовище, яке буде повністю влаштовувати програміста. Ідеальна IDE повинна мати зручний і інтуїтивно зрозумілий інтерфейс, що дозволяє розробникам ефективно працювати з кодом. Вона має включати широкий набір функцій, таких як автодоповнення коду, вбудовану документацію, інструменти для налагодження та відлагодження коду, а також можливості рефакторингу. Саме таким середовищем можна вважати Visual Studio Code (VS Code), якому віддають перевагу велика кількість розробників (особливо на ОС Windows).

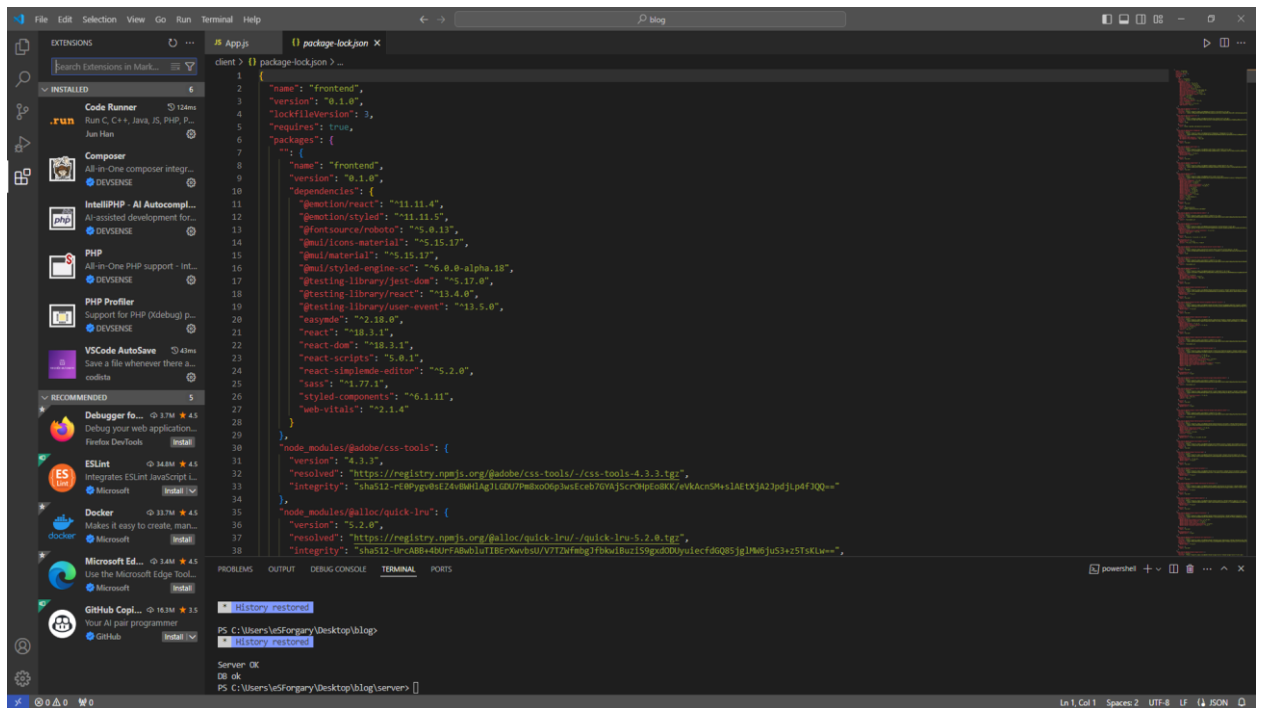


Рисунок 2.4 – Інтерфейс та можливості Visual Studio Code

VS Code – це потужний текстовий редактор, що розроблений компанією Microsoft. Він має широкий набір функцій і можливостей, які роблять його популярним серед розробників усіх рівнів.

VS Code є відкритим проектом і розповсюджується безкоштовно, що означає, що будь-який розробник може переглядати, редагувати та вносити вклад у його розвиток. Це створює сприятливе середовище для спільної роботи та співтворчості.

Редактор має велику кількість розширень та плагінів, які дозволяють розширити його функціональність та значно спростити процес розробки. Є можливість встановлювати плагіни для роботи з різними мовами програмування, інтеграції з різними сервісами та фреймворками, а також для вдосконалення робочого процесу.

VS Code надає зручні інструменти для швидкого переміщення між файлами, функціями та рядками коду. Це забезпечує швидкий доступ до необхідних ресурсів і сприяє підвищенню продуктивності розробника.

Редактор має функцію автодоповнення, яка пропонує можливі варіанти завершення коду на основі контексту та типу даних. Це значно полегшує

написання коду, запобігає допущенню помилок і прискорює процес розробки.

VS Code ідеально поєднується з різними системами контролю версій, такими як Git, що дозволяє зручно вести роботу зі сховищами коду, використовуючи всі основні функції контролю версій. Інтеграція з Git дозволяє легко відслідковувати зміни в коді, створювати гілки і злиття, а також виконувати інші операції керування версіями.

У VS Code вбудована можливість взаємодії з командним рядком (терміналом) безпосередньо в редакторі, що дозволяє виконувати команди, запускати скрипти та встановлювати залежності без переходу до зовнішнього терміналу. Це забезпечує зручний та безперервний робочий процес.

VS Code надає підтримку для багатьох мов програмування, включаючи JavaScript, TypeScript, Python, Java, C#, HTML/CSS, і багато інших. Кожна мова має свої вбудовані функції та розширення для поліпшення робочого процесу, що робить VS Code універсальним інструментом для розробки різноманітних проектів.

Редактор підтримує спільну роботу над проектами за допомогою вбудованих інструментів для обміну кодом та комунікації з іншими членами команди. Це дозволяє легко об'єднувати зусилля та спільно працювати над проектами в реальному часі, спрощуючи командну роботу і підвищуючи ефективність розробки.

Таким чином, враховуючи усі можливості редактора Visual Studio Code можна вважати його ідеальним вибором для реалізації поставлених задач у розробці клієнтської та серверної частини веб-застосунку.

JWT, MERN,

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ДЕМОНСТРАЦІЯ РОБОТИ ВЕБ-БЛОГУ

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований метод покращення зображень за допомогою

Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст».

Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст».

Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст» Текст стилем «Основний текст».

Результати роботи апробовано у вигляді 3 тез доповідей під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [42], XV Регіональної студентської науково-технічної конференції «НАУКА – ПЕРШІ КРОКИ» [43], XXVI Міжнародної науково-практичної конференції «Topical issues of practice and science» [44].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

Приклад, як оформити навчальний посібник:

1. Путятін, Є.П., Гороховатський, В.О., & Матат, О.О. (2006). Методи та алгоритми комп'ютерного зору: навч. посібник.
2. Творошенко, І.С. (2021). Технології прийняття рішень в інформаційних системах: навч. посібник. *Харків: ХНУРЕ*.
3. Гороховатський, В.О., & Творошенко, І.С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.
4. Кобилін, О.А., & Творошенко, І.С. (2021). Методи цифрової обробки зображень: навч. посібник. *Харків: ХНУРЕ*.
5. Гороховатський В.О., Творошенко І.С. (2022) Аналіз багатовимірних даних за описом у формі множини компонент: монографія. Харків: ХНУРЕ, 124 с.

Приклад, як оформити статтю у журналі:

6. Kobylin O., Gorokhovatskyi V., Tvoroshenko I., and Peredrii O. (2020) The application of non-parametric statistics methods in image classifiers based on structural description components, *Telecommunications and Radio Engineering*, 79(10), pp. 855-863.
7. Daradkeh, Y.I., Tvoroshenko, I., Gorokhovatskyi, V., Latiff, L.A., and Ahmad, N. (2021) Development of Effective Methods for Structural Image Recognition Using the Principles of Data Granulation and Apparatus of Fuzzy Logic, *IEEE Access*, 9, pp. 13417-13428.
8. Daradkeh, Y.I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., and Al-Dhaifallah, M. (2021) Methods of Classification of Images on the Basis of the Values of Statistical Distributions for the Composition of Structural Description Components, *IEEE Access*, 9, pp. 92964-92973.

9. Gorokhovatskyi, V.O., Tvoroshenko, I.S., and Peredrii O.O. (2020) Image classification method modification based on model of logic processing of bit description weights vector, *Telecommunications and Radio Engineering*, 79(1), pp. 59-69.
10. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Cluster representation of the structural description of images for effective classification, *Computers, Materials & Continua*, 73(3), pp. 6069-6084.
11. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, 6(3), С. 5-12.
12. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Al-Dhaifallah M. (2022) Classification of Images Based on a System of Hierarchical Features, *Computers, Materials & Continua*, 72(1), pp. 1785-1797.
13. Tvoroshenko I., and Gorokhovatskyi V. (2022) The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, *International Journal of Engineering and Information Systems*, 6(2), pp. 40-48.
14. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.
15. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), pp. 25-36.
16. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

17. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, *IEEE Access*, vol. 11, pp. 126938-126949.

18. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, *Indonesian Journal of Electrical Engineering and Computer Science*, 33(1), pp. 113-125.

19. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, *International Journal of Academic Engineering Research*, 7(9), pp. 64-72.

20. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, *International Journal of Academic and Applied Research*, 7(9), pp. 57-70.

21. Tvoroshenko I., Pomazan V., Gorokhovatskyi V., and Kobylin O. (2023) Application of video data classification models using convolutional neural networks, *International Journal of Academic and Applied Research*, 7(11), pp. 134-145.

Приклад, як оформити тези конференції:

22. Gorokhovatskyi, V., Gorokhovatskyi, O., Yevgenyi, P., & Olena, P. (2018). Quantization of the Space of Structural Image Features as a Way to.

23. Gorokhovatskyi, V., & Tvoroshenko, I. (2020). Image Classification Based on the Kohonen Network and the Data Space Modification.

24. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. *International scientific symposium «INTELLIGENT SOLUTIONS-S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium, September 28, 2023, Kyiv-Uzhorod, Ukraine*, pp. 25-27.


Приклад, як оформити електронне посилання:

25. The flow of improved BRISK algorithm. URL: https://www.researchgate.net/figure/the-flow-of-improved-BRISK-algorithm_fig1_328946366 (дата звернення 14.04.2024).

Приклад, як оформити посилання на свою попередню кваліфікаційну роботу:

26. Іванов І.І. Дослідження методів пошуку зображення у базах даних: кваліфікаційна робота першого (бакалаврського) рівня вищої освіти: 122 Комп'ютерні науки. Харків, 2022. 75 с.

Увага!!! Усі посилання на джерела повинні бути у форматі APA!!!

Щоб правильно оформити посилання, рекомендуємо скористатися <https://scholar.google.com.ua>. Можна знайти необхідну літературу за автором чи тематикою, вибрати процитовану літературу, а потім за допомогою  перейти до трьох видів форматів та скопіювати текст формату APA.

У цьому розділі має бути не менше 30 джерел інформації, $\frac{3}{4}$ із них – за останні 5 років, не менше $\frac{1}{2}$ публікацій керівника кваліфікаційної роботи або інших викладачів кафедри.

Забороняється цитування в тексті та внесення до бібліографічних списків тих джерел, які опубліковані російською мовою в будь-якій країні, а також джерел іншими мовами, якщо вони опубліковані на території росії та білорусі. Тобто в перелік джерел посилання НЕ вносимо джерела на російській мові (навіть ваших керівників кваліфікаційних робіт!!!), а також, якщо видавництво публікації розташоване на вказаних вище територіях.

ЗАСТОСУНОК А

ТЕСТОВІ ЗОБРАЖЕННЯ

A photograph of a handwritten word 'people,' in cursive script on a light-colored, textured background. The word is written in dark ink and is slightly tilted to the right.

Рисунок А.1 – Приклад рукописного тексту № 1

A photograph of a handwritten word 'meeting' in cursive script on a light-colored, textured background. The word is written in dark ink and is slightly tilted to the right.

Рисунок А.2 – Приклад рукописного тексту № 2

A photograph of a handwritten phrase 'as a percentage of social service' in cursive script on a light-colored, textured background. The phrase is written in dark ink and is slightly tilted to the right.

Рисунок А.3 – Приклад рукописного тексту № 3