МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:
«Web-блог з використанням стеку MEAN»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Студент гр. ІК.м-91 **Карепін В.О.**

Керівник роботи **Проценко О.Б.**

СУМИ 2020

_____

(назва вузу)

Факультет _____ Кафедра _____
Спеціальність _____

Затверджую:
зав.кафедрою _____
_____
“_____” _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

_____

(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) _____
_____

затверджую наказом по інституту від “____” _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____
3. Вхідні данні до проекту (роботи) _____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
_____
_____
_____
_____
_____

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) _____
_____
_____
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: сторінки, рисунків, джерел літератури, додатки.

Об'єкт дослідження – Web-блог з використанням стеку MEAN

Мета роботи – розробка та програмна реалізація веб-додатку.

Методи дослідження – розробка та програмна реалізація веб-додатку.

Результати − здійснено вибір методів вирішення поставленого завдання, розроблений веб-додаток з використанням стеку MEAN, який має можливість адаптуватися під будь-який гаджет чи комп’ютер, має серверну та клієнтську частину.

UI UX, MEAN, JAVASCRIPT, ANGULAR, EXPRESSJS

Зміст

ВСТУП

Розвиток інформаційних технологій виходить на новий рівень інформатизації та автоматизації, що вимагає створення і використання нових інформаційних технологій та систем. В Україні та світі переходять на нові технології управління та виробництва продукції, в тому числі інтелектуальні. Зрозуміло, що сучасний етап розвитку науки і технологій вимагає фахівців відповідного рівня, здатних проектувати, розробляти, інтегрувати і підтримувати інформаційні системи, засновані на новітніх технологіях і архітектурних платформах.

На сьогоднішній день в процесі розробки якісних сайтів величезну роль грає веб-дизайн та веб-розробка. Користувачі частіше за все звертаються до інтернет-блогів, інфопорталів, щоб отримати корисну інформацію. Кожного дня кількість інтернет-блогів зростає з великою кількістю. Кожен блог або інфопортал направлений на окрему тематику. Завдяки чому користувачі мають можливість отримувати відповіді на їх запитання. Більше того, користувачі можуть реєструватись у інфопорталах / блогах та ділитися своїм досвідом з іншими. Блог – нова альтернатива форумів, яка є зручнішою та сучаснішою.

Сьогодні розробка якісного блогу багато в чому залежить від рівня професіоналізму веб-розробника. Для коректної праці блогу, розробник повинен забезпечити не лише гарний вигляд сайту, але й технічну частину пов’язану з базою даних, серверною та фронт-енд частиною. По суті, веб-розробка включає в себе не лише написання коду, а й: створення структури сайту, визначення цільових користувачів, коректне з’єднання з базою даних та сервером для зручного використання порталу.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

Блог – це інформативний веб-сайт або онлайн-журнал, який показує найновіші повідомлення. Він також повинен регулярно оновлюватися і зазвичай ним керує окрема особа або невелика група людей. Погляди і думки в блозі найчастіше ставляться до однієї або декількох суміжних тем. Це все, що стосується визначення. Хоча в останні роки лінія, що розділяє блоги і сайти, стала розмитою.

Велику популярність блоги отримали з розвитком інтернет-технологій, коли у кожної людини з'явився вільний доступ до інтернету. Сьогодні це не просто можливість ділитися своїми думками з іншими людьми, це можливість заробляти на улюбленій справі і здобути більшу популярність.

У блогосфері нерідкі випадки, коли молодий хлопець чи дівчина стають мільйонерами. Монетизація блогів може приносити пристойний дохід. І все, що для цього потрібно – регулярно публікувати цікавий контент. Маючи велику кількість читачів, глядачів, блогер стає об'єктом уваги рекламодавців. Його аудиторія лояльна і довіряє йому, отже, вона добре конвертується.

Існують такі види ведення блогів:

- текстові блоги;
- мікроблоги (небагато тексту + картинка або відео);
- відеоблог;
- стрімінг;
- ведення сторінки в соціальних мережах;
- створення власного сайту.

1.1 Огляд відомих рішень

Веб-програмування – це область веб-розробки і тип дизайну, призначений для проектування системи веб-інтерфейсів, є дуже важливою частиною побудови

веб-сайту. Система проектування є одним із загальних правил. Це дозволяє командам розробляти, писати тексти та виробляти продукт у будь-якій формі. Зазвичай це набір стилів або компонентів, які можна описати, зібрати на декількох монтажних дошках та виконати основні завдання. Системи іноді доповнюють, наприклад, шаблони UX, брендинг Voice & Tone. Якщо ви маєте справу з великим продуктом, важливо подбати про Voice & Tone.

Tone of voice (ToV) – це тональність, якої притримується бренд у комунікаціях із своїм користувачем. Це внутрішні правила взаємодії компаній з аудиторією, які є єдиними для всіх каналів: веб-сайт, поштова розсилка, соціальні мережі та навіть дзвінки за телефоном. Складаються з вимог до стилю мови, форми та даних корпоративної інформації. Зазвичай описуються в Style Guide або брендбуці.

Веб-програмування – галузь веб-розробки і різновид дизайну, в завдання якої входить проектування-системи веб-інтерфейсів, що є дуже важливою частиною побудови сайту.

Зазвичай це набір стилів чи компонентів, які зібрані на кількох артбордах і виконують базові задачі. Іноді системи доповнюються, наприклад, UX-патернами, брендингом, Voice & Tone. Тобто, дизайн система є важливим стартом у будь-якому проекті.

Дизайн-система – це:

– зазвичай: візуальний стиль та UI-компоненти;

– іноді: UX-патерни, брендинг, Voice & Tone;

– рідко: доступність (accessibility), локалізація та візуалізація даних.

Вона розробляється за допомогою:

– зазвичай: HTML, CSS, документів і гайдлайнів;

– іноді: дизайн-асетів (Sketch-файлів, бібліотек тощо);

– рідко: JS-фреймворків (React, Angular тощо).

Усе це робиться для продуктових команд (дизайнерів, розробників, будь-кого ще), які і є користувачами дизайн-систем.

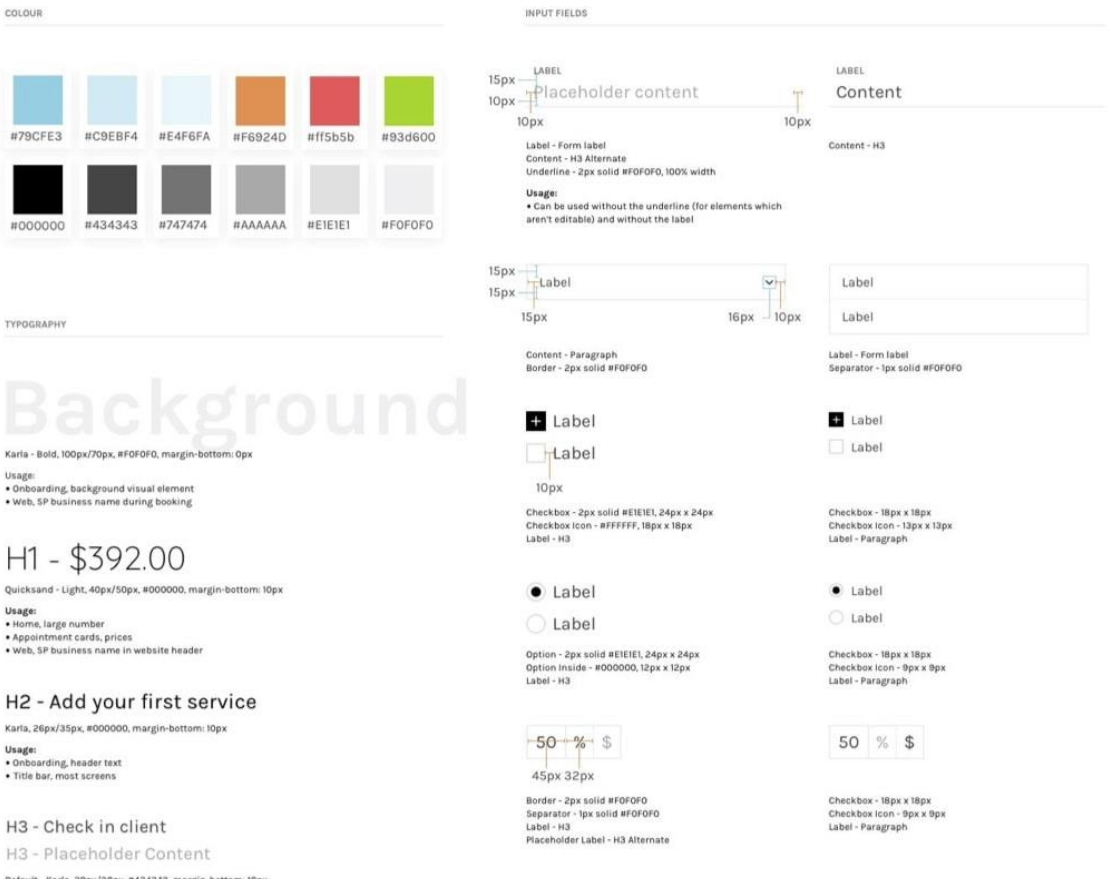

Рисунок 1.1 – Приклад дизайн-системи

На даний момент дане рішення вже реалізоване на декількох веб-ресурсах. Кожен із них має можливість надавати інформацію і є блогом, але без можливості реєстрації користувачів. Тобто, вони є звичайною платформою для публікації новин.

Проте, дані ресурси мають свої певні недоліки. Відокремити можна такі: незручний інтерфейс, забагато зайвої інформації, відсутня стилістика сайту, відсутність реєстрації.

Для прикладу і порівняння можна відокремити декілька ресурсів зі схожими можливостями. Щоб отримати перелік таких сайтів, достатньо звернутися до пошукової системи google.com і створити запит «блог». Перейшовши на один з таких сайтів можна одразу помітити згадані вище недоліки.

1: https://vas3k.ru

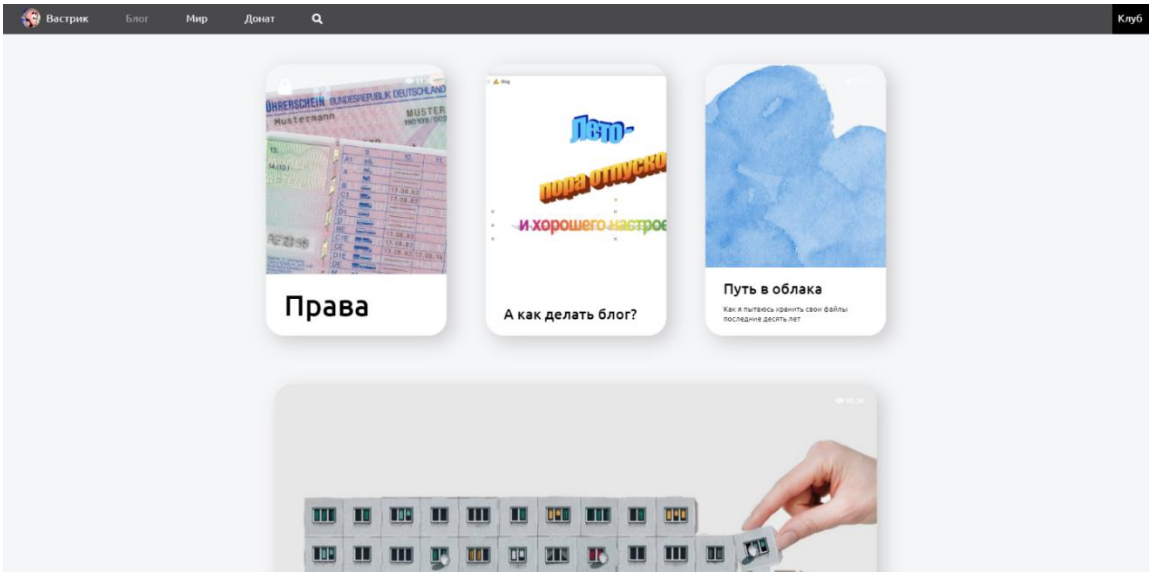

Рисунок 1.2 – Головна сторінка блогу https://vas3k.ru

Головний недолік цього ресурсу полягає в тому, що в нього досить не інтуїтивний інтерфейс. Користувачу, який вперше потрапить на даний сайт, буде потрібно витратити багато часу, щоб знайти цікавий йому «блок». З його вигляду не можливо нічого зрозуміти. Такий блог викликає лише одне бажання – покинути його.

Також недоліком потрібно відмітити вхід до адмін панелі, або це є реєстрація, не зрозуміло. Велике зображення та кнопка, яка тільки все псує.

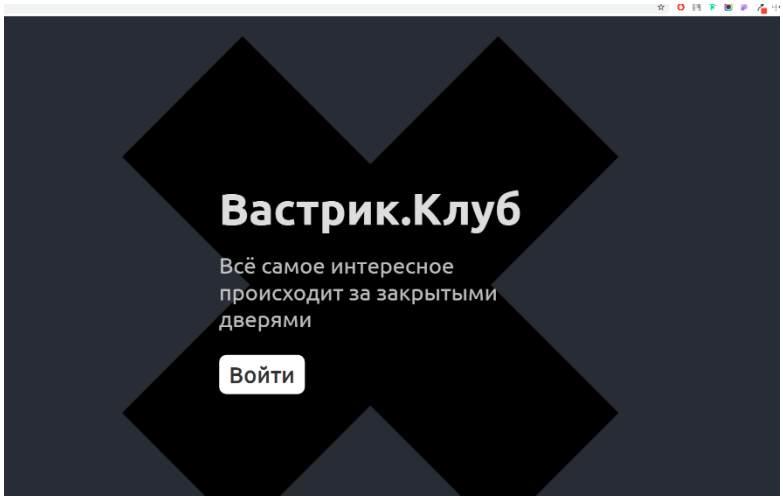

Рисунок 1.3 – Сторінка входу або реєстрації до блогу

2: http://maximilyahov.ru/

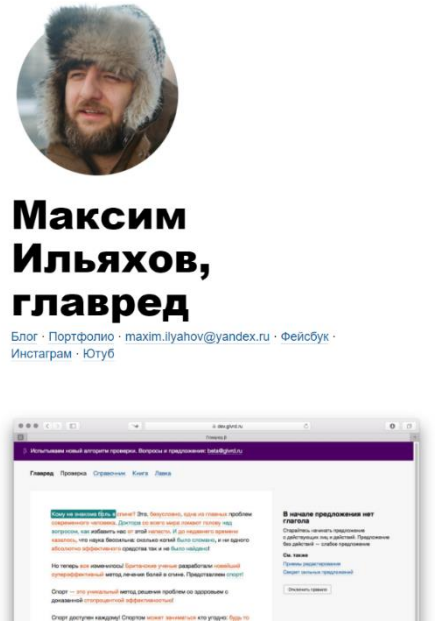

Рисунок 1.4 – Головна сторінка сайту http://maximilyahov.ru/

Даний ресурс також має недоліки, проте є більш коректним у порівнянні з попереднім сайтом. На сайті відсутні зайві елементи, проте він є занадто мінімалістичним, категорії підкріплюються лише зверху. Входу або реєстрації немає. Перехід до статті також не є коректним.

3: http://deathbypassivevoice.com/

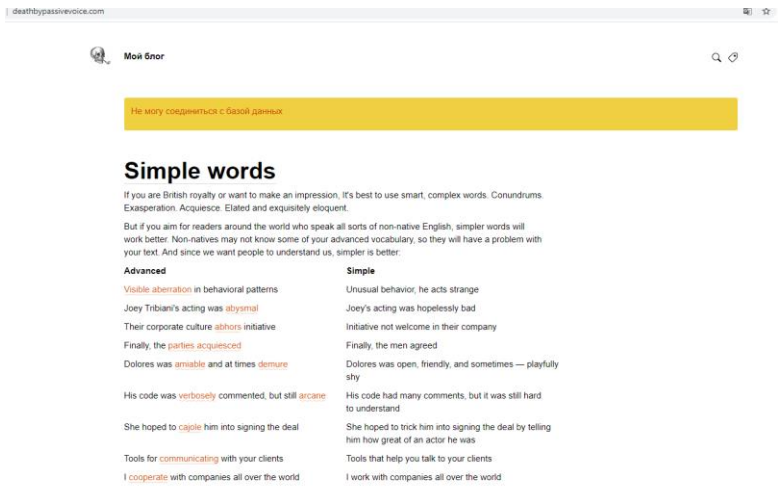

Рисунок 1.5 – Головна сторінка сайту http://deathbypassivevoice.com/

У блозі взагалі відсутнє з’єднання з базою даних, та блог не є функціональним.

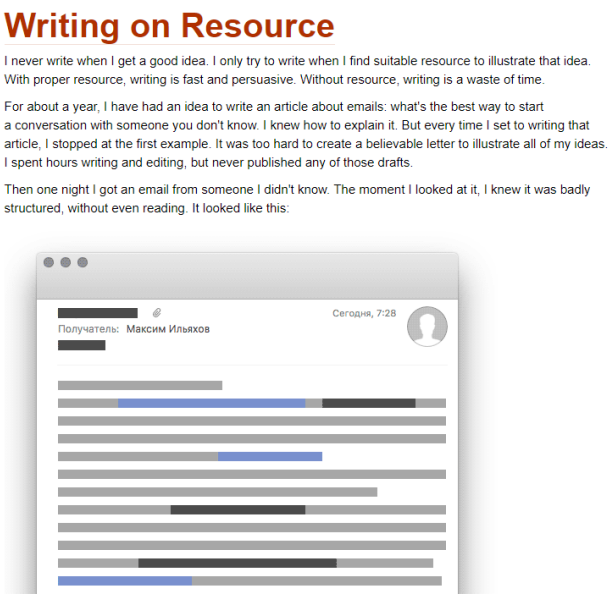

Рисунок 1.6 – Сторінка з статтею

Сторінка з лінком веде до головної сторінки, перейти на окрему сторінку з статтею неможливо, так як «якорі» встановлені на переадресацію до головної сторінки.

Клієнтські технології

Клієнтська частина

Технології створення веб-додатків можна розділити на два види: клієнтські, тобто використовуються веб-браузерами та веб-клієнтами, наприклад, додатками або клієнтами які користуються сайтами, додатками для миттєвого обміну повідомленнями (фейсбук, твіттер, інстаграм, тощо) і серверні, тобто використовуються на веб-серверах, користувачу не помітна робоча частина веб-серверу.

Клієнтські технології застосовуються головним чином для підвищення інтерактивності додатків, наприклад для перевірки коректності даних, що

вводяться без додаткового звернення до сервера, і для створення зручного для користувача інтерфейсу.

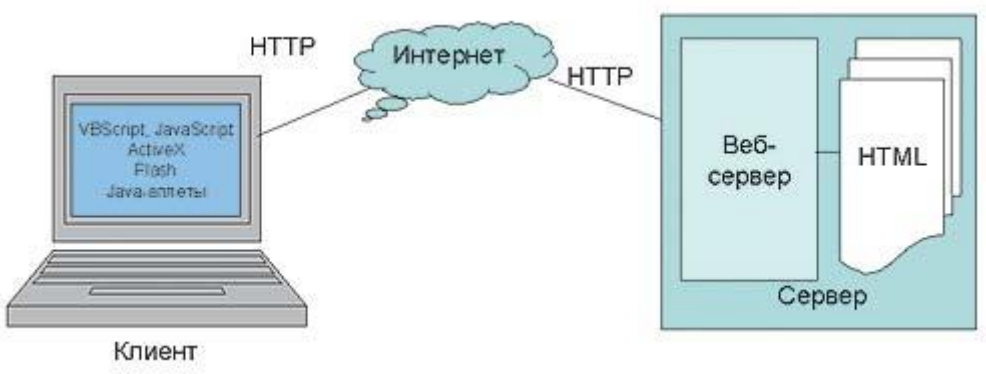

Рисунок 1.7 – Схема зображення клієнтської та серверної частини

Більшість сучасних веб-браузерів, створених для різних платформ і пристроїв, здатна інтерпретувати впроваджений HTML-сторінку код на скриптових мовах, таких як VBScript і JavaScript [8]. Типові приклади застосування запровадженого клієнтського коду - перевірка коректності введених користувачем даних без звернення до веб-сервера, створення деяких елементів дизайну - на зразок спливаючих кнопок і меню, а також управління іншими об'єктами, впровадженими в HTML-сторінку. Це є тим, що спочатку розробляється і відображається візуально в дизайн системі. Після чого, це все переводиться в код, а вже потім з’являється клієнтська частина.

HTML і CSS: це основні будівельні блоки будь-якого сайту. HTML диктує організацію і контент сайту. CSS містить код для кожного графічного елемента - від фонів, блоків, кнопок і шрифтів - який становить зовнішній вигляд веб-сайту. JavaScript - це клієнтська скриптова мова програмування. Найбільш широко використовуваний клієнтський скрипт - майже кожен інтерфейс сайту - це поєднання JavaScript, HTML і CSS [1]. На JavaScript є безліч фреймворків, які спрощують його і надають йому більшу гнучкість.

Серверна частина

Клієнт-серверну частину можна означити, як концепцію інформаційної мережі, в якій основна частина її ресурсів зосереджена в серверах. Така архітектура визначає такі типи компонентів:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

– рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

– прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації;

– рівень управління даними, який забезпечує зберігання даних та доступ до них.

Стек MEAN

До стеку MEAN входять три фреймворки JavaScript та база даних, орієнтована на документи NoSQL – база даних. MEAN складається з чотирьох компонентів: MongoDB, Express, Angular, NodeJs.

MongoDB - це безкоштовна база даних, орієнтована на документи, організована таким чином, щоб забезпечити як масштабованість, так і гнучкість у розробці. Дані в MongoDB не записуються в таблиці та стовпці, як у реляційній базі даних. Натомість, MongoDB зберігає подібні до JSON документи з динамічними схемами.

Документо-орієнтовні СУБД використовуються для збереження JSON-документів у колекціях і здійснення запитів за потрібними полями. Дану базу даних можна використовувати для створення аплікацій, в яких не будуть зберігатися занадто великі кількості зв'язків. Гарним прикладом таких аплікацій є двигун для блог-платформи або збереження каталогу продуктів.

Express.js – це фреймворк для сервера додатків Node.js призначений для створення односторінкових, багатосторінкових та гібридних веб-додатків. Це фактично стандартний серверний фреймворк для node.js.

AngularJS – це структурний фреймворк для динамічних веб-додатків. Він може використовувати HTML як мову шаблону та розширити синтаксис HTML, щоб чітко та стисло описати компоненти вашої програми. За допомогою Angular при написанні коду, можно значно скоротити час та код.

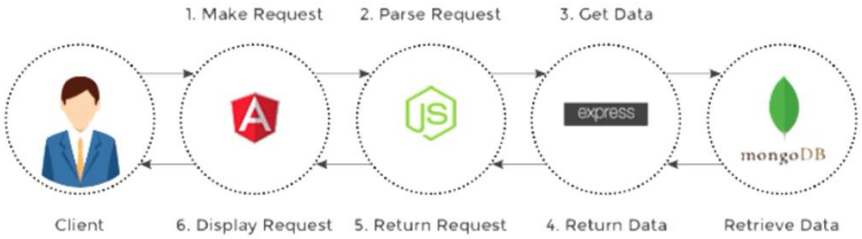

Рисунок 1.8 – Схема роботи стеку MEAN

Користувач спочатку взаємодіє через графічний інтерфейс, який реалізовано через AngularJs. Після взаємодії між графічним інтерфейсом, запит приймає NodeJs. Обробка самого запиту, яка дія сталася і що повинно виконуватися відбувається через ExpressJs. Якщо буде необхідно, йде підключення до бази даних MongoDB і отримання від неї будь-якої інформації. Після отримання даних, інформація повертається до ExpressJs, звідти передається до платформи NodeJs. Після цього циклу все повертається до користувача.

Постановка задачі

Дана робота спрямована на створення зрозумілого та лаконічного порталу, на якому можна легко знайти потрібну інформацію та з можливістю публікації своїх статтей.

Для реалізації поставленої задачі буде використовуватись стек MEAN (Mongo, Express, Angular, NodeJs) За допомогою цього стеку будет створено повноцінний веб-додаток.

Розробити веб-додаток, який матиме візуальну та серверну частину (Front-end та Back-end). Необхідно створити такі сторінки:

– головна сторінка – на якій будуть відображатись найостанніші додані статті;
– сторінка з статтею;
– форма реєстрації – реєстрування для користувачів;
– можливість авторизації користувачів;
– особистий кабінет, через який можна створювати власні статті.

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ

2.1 Вибір програмної реалізації

Для початку потрібно проаналізувати та створити план структури розробки блогу та макету реалізації. Для цього будуть використані знання HTML5, SCSS, Bootstrap, Angular, NodeJs, навички з Adobe Photoshop для створення макету та його верстки.

Для створення блогу було використане таке програмне забезпечення:

– Редактори для написання коду – Visual Studio Code;

– Графічні редактори - Adobe Photoshop та Figma;

– Написання сайту за допомогою HTML, SCSS, Angular, Bootstrap, NodeJs;

– Браузери: Google Chrome, Mozzila Firefox, Opera;

Для досягнення поставленої цілі необхідно створити зрозумілий та лаконічний дизайн, де користувач матиме можливість зручно орієнтуватися на сайті, та отримати швидку відповідь на необхідні запитання, або додавати свої статті до блогу.

Інтерфейс програми

Стартове вікно програми для написання веб-блогу за основу було взято програмне забезпечення Visual Studio Code , який зображено на Рисунку 2.1

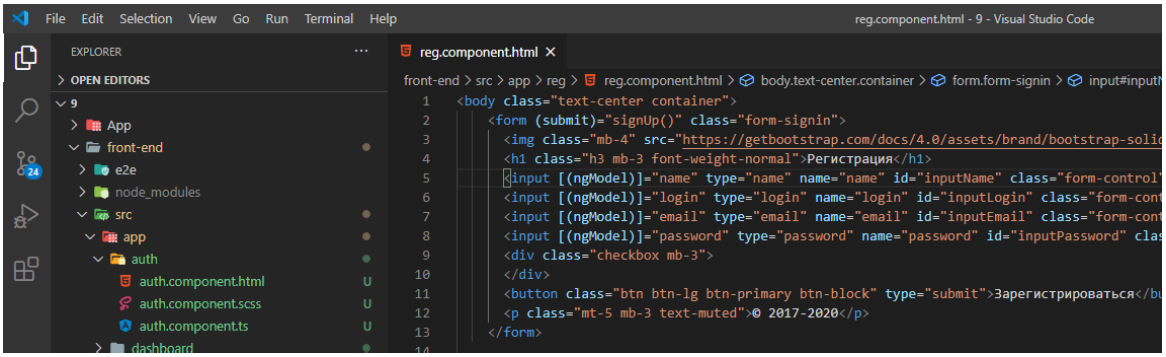

Рисунок 2.1 – Вигляд вікна Visual Studio Code

Плагіни

Перевагою є те, що Visual Studio Code дозволяє встановлювати багато плагінів, які значно покращують процес розробки та допомагають заощадити багато часу.

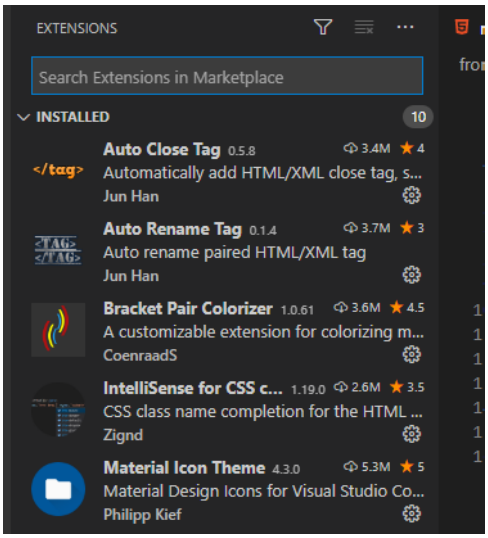

Рисунок 2.2 – Вигляд плагінів в Visual Studio Code

2. 2 Вибір і обґрунтування середовища розробки

Існує багато різновидів середовища розробки для написання коду, з них найпоширеніші це – SublimeText, Notepad++, Visual Studio Code, Brackets тощо. Програміст повинен самостійно обрати середовище, з яким йому буде зручно працювати. Всі вони мають одну ціль – інтерпретувати код, проте, необхідно обирати з можливості та зручності того чи іншого програмного забезпечення.

Середовище Visual Studio Code

Visual Studio Code – засіб для створення, редагуваннята зневадження сучасних веб-застосунків і програм для хмарних систем. Visual Studio Code знаходиться в безкоштовному доступі в інтернеті та підтримується різними версіями для платформ Windows, Linux і OS X.

Редактор містить вбудовані інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і

контекстних підказок. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціонується як легковаге рішення, що дозволяє обійтися без повного інтегрованого середовища розробки.

Підтримує велику кількість мов та має додаткові плагіни для покращення роботи JavaScript, C++, C#, TypeScript, jade, PHP, Python, XML, Batch, F#, DockerFile, Coffeecript, Java, JSON, HTML, CSS, LESS і SASS, Haxe.

Локальний веб-сервер

Node.js дає можливість писати сервери за допомогою JavaScript з неймовірною продуктивністю. Node.js - середовище виконання, яке використовує той самий рушій V8 Javascript, який ви можете знайти в браузері Google Chrome.

NPM - це менеджер пакетів, який ви можете залучити до свого проекту. Наприклад, у проекті використовується Bootstrap, не потрібно вручну копіювати всі необхідні файли з Bootstrap і зберігати цю бібліотеку у сховищі GIT, просто вкажіть, що вам потрібно Bootstrap, у файлі конфігурації проекту, і менеджер пакунків npm встановить Bootstrap у копії вашого проекту, і не потрібно зберігати всі ці бібліотеки у сховищі GIT вашого проекту. [11]

За допомогою npm Manager можна додавати величезну кількість бібліотек та інструментів до свого проекту. Одним із головних модулей npm є:

Package.json – в ньому вказані базові дані: назва проекту, автор версія. Після ініціалізаціїї якого, всі встановленні пакети через npm будуть записуватись в package.json

Після інсталяції бібліотеки express

> npm install express

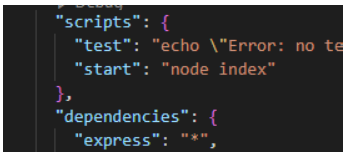

Рисунок 2.3 – Встановлена бібліотека express в package.json

2. 3 Вибір СУБД

MySQL

MySQL – це перевірена технологія протягом часу. MySQL використовується великими компаніями більш, ніж 18 років. Так як він використовує стандарт SQL, є можливість досить простої міграції на інші SQL-бази даних. Є можливість транзакцій. Підтримуються складні запити, включаючи аналітику.

SQL - проста мова програмування, яка має невелику кількість команд і для вивчення цієї мови поріг входження невеликий. Structured Query Language - мова структурованих запитів, яка була розроблена для роботи з БД, а саме, щоб отримувати/додавати/змінювати дані, мати можливість опрацьовувати великі масиви інформації та швидко отримувати структуровану та згруповану інформацію. SQL має свої команди (оператори), за допомогою яких віддаються вказівки для вибірки даних.

ID	Month	Product	City	Quantity	Amount
2	April	Bikes	Montreal	12	$4 500,00
3	April	Bikes	Montreal	56	$21 000,00
4	April	Bikes	San Francisco	854	$320 250,00
5	April	Bikes	New York	25	$9 375,00
6	April	Skates	Montreal	56	$5 544,00
7	April	Skates	Toronto	854	$84 546,00
8	April	Skates	San Francisco	25	$2 475,00
9	April	Skates	New York	663	$65 637,00
10	April	Skis Long	Montreal	854	$209 230,00
11	April	Skis Long	Toronto	25	$6 125,00
12	April	Skis Long	San Francisco	663	$162 435,00
13	April	Skis Long	New York	21	$5 145,00
14	April	Skis Short	Montreal	21	$4 389,00

Рисунок 2.4 – Приклад використання мови SQL

MongoDB

З точки зору MongoDB, тут перевага в тому, що за основу взятий гнучкий JSON-формат документів. Для розробників не потрібно приділяти багато уваги з додаванням колонок в SQL-базах даних. Не потрібно вивчати мову SQL. Прості запити рідше створюють проблеми. Якщо подивитися на проблеми продуктивності, в основному вони виникають, коли люди пишуть складні запити з JOIN в купу таблиць і GROUP BY. У MongoDB вбудована досить проста масштабованість з використанням технології шардинга. Якщо виникають складні запити, ми їх зазвичай вирішуємо на стороні додатку. [7]

Одним з популярних стандартів обміну та зберігання даних є JSON (JavaScript Object Notation). JSON ефективно описує складні структурні дані. Спосіб зберігання даних MongoDB у цьому відношенні схожий на JSON, хоча формально JSON не використовується. Для зберігання в MongoDB використовується формат BSON.

BSON дозволяє швидше працювати з даними: швидкий пошук та обробка. MongoDB – гарний вибір при створенні програми, концепція якої включає роботу з документами. Наприклад, до цього типу додатків можна віднести платформу для ведення блогу, де кожен автор може мати кілька блогів, і кожен із них містить багато коментарів. База даних, що використовується для такої програми, повинна легко розширюватися, і MongoDB для цього ідеально підходить. Як зазначалося вище, орієнтовані на документи СУБД використовуються для зберігання документів JSON у "колекціях" та для виконання запитів у необхідних полях. Можна використовувати цю базу даних для створення додатків, які не містять занадто багато посилань. Гарним прикладом такої програми є інструмент для платформи ведення блогу або для зберігання каталогів товарів.

Вибір фреймворка Angular

Фреймворк JS (або на стороні клієнта) - це технологія, яка надає нам інструменти для створення веб-програми, але також визначає дизайн програми та

організацію коду. Бібліотека спеціалізується на конкретних задачах. Наприклад, jQuery спеціалізується на роботах з DOM, фонова потужність занадто мала. Ангуляр призначений для того, щоб побудувати архітектуру всього додатка.

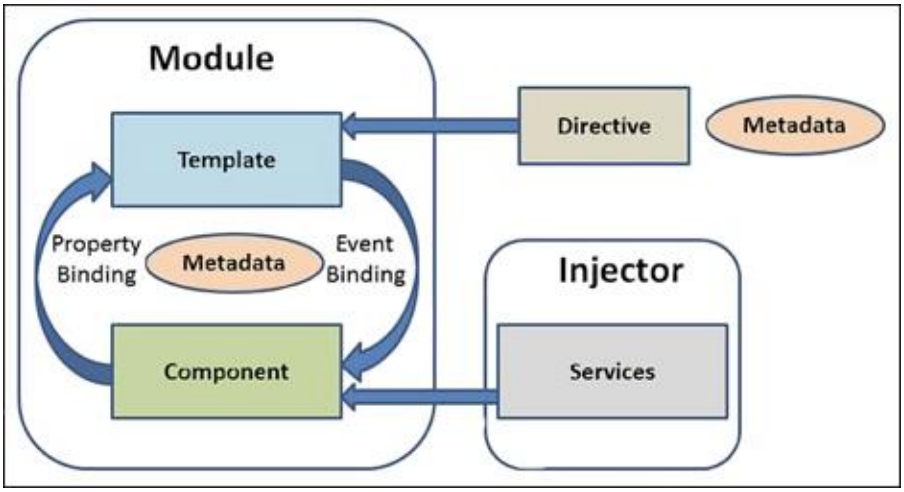

Рисунок 2.5 – Загальний огляд архітектури Angular

Angular застосунки пишуться на TypeScript, а не на чистому JavaScript.

Архітектура Angular складається з:

– Module;
– Component;
– Template;
– Service;
– Router;
– Pipe;
– Directives.

Модулі (Module) – структурні одиниці застосунку, які інкапсулюють певну логіку. Це структури, які зберігають певні компоненти, директиви та сервіси, об’єднані певною логікою. Наприклад: перегляд списку надісланих листів.

Компоненти (Component) – зберігає дані та логіку відображення цих даних у шаблоні. Шаблон тісно пов’язаний з компонентом. Дані з компонента можна з легкістю відображати у шаблоні, використовуючи спеціальний синтаксис.

Шаблон (Template) – фрагмент html-коду з додаванням синтаксису. Він дозволяє впроваджувати в шаблон дані з компонента без використання innerHTML та подібних методів.

Сервіс (Service) в Angular [9] являє собою typescript класи, які виконують задачі, пов’язані з отриманням, зберіганням та обробкою даних. Наприклад, логування, перетворення даних для подальшої передачі у компонент, звернення до backend та ін. На відміну від компонентів та директив сервіси не працюють з представленнями (шаблонами) напряму.

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Інформаційна модель

Головна сторінка

Створений сайт має такі компоненти як: кнопки, додаткові сторінки з дописами, адмін панель, реєстрацію та авторизацію. Вигляд сторінки з компонентами під час розробки зображено на **рисунку 3.1**.

Схема головної сторінки сайту-каталогу для інструментів.

Рисунок 3.1 – Блок-схема головної сторінки сайту

Аналітика та тестування цільової аудиторії

Цільова аудиторія - це група людей, які задовольняють свої потреби за допомогою сервісу, веб-додатку. Додаток може вирішити їхні проблеми та відповісти на всі запитання. Кожен власник компанії, займаючись виробництвом або продажем якогось продукту, в загальних рисах уявляє для кого він це робить.

Однак загальних характеристик недостатньо, якщо фірма хоче рости і розвиватися, і не загубитися серед конкурентів. Рекомендуємо провести якісний аналіз цільової аудиторії, що дозволить грамотно планувати продаж і проводити рекламні компанії.

Для того, щоб мати загальну картину щодо цільової аудиторії та їх потреб, було проведенно тестування серед користувачів. Для тестування, загалом було опитано 60 респондентів.

Метод анкетування

Для аналізу використовувався метод анкетування. Технологія анкетування – це метод оцінки, який заключається в тому, що цей спосіб дозволяє зібрати великий об’єм інформації для якого не потрібно мати коштів, підготовлених працівників та лабораторні обладнання. Анкети як правило, використовуються на початку дослідження, а також для збору зворотнього зв’язку про поточний проект. Користувачі погано заповнюють великі анкети, як наслідок: в таких випадках часто дають найменш ціннішу інформацію. Тому, головне використовувати найточніші питання. Двозначні питання які не відповідають задачі, потрібно позбавитись. Кожне питання повинно бути присвячене лише одній темі і питання не повинно маніпулювати користувачем анкетування, підштовхуючи його відповісти певним чином.

Для того щоб упевнитися в тому, що анкетування дає якісну і надійну інформацію, можна провести опитування повторно і попросити респондента заповнити ту же анкету вдруге. Після того як анкетування завершено, рекомендується провести пілотне дослідження, під час якого якість анкетування буде проведена на меншому числі опитаних.

Для дослідження було створено гугл-форму, яка була розміщена у різних чатах та за допомогою «сарафаного радіо»

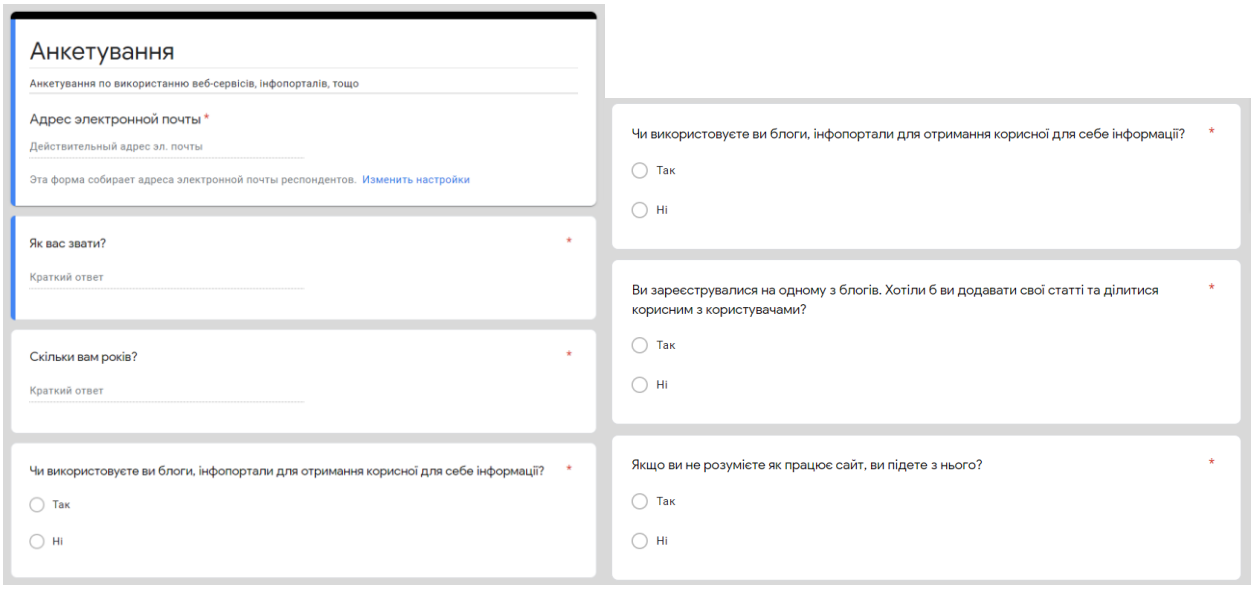

Рисунок 3.2 – Питання які входять до анкетування

Результат, який було отримано за 12 годин анкетування:

Респондентів: 60. З них жінок: 35, чоловіків: 25

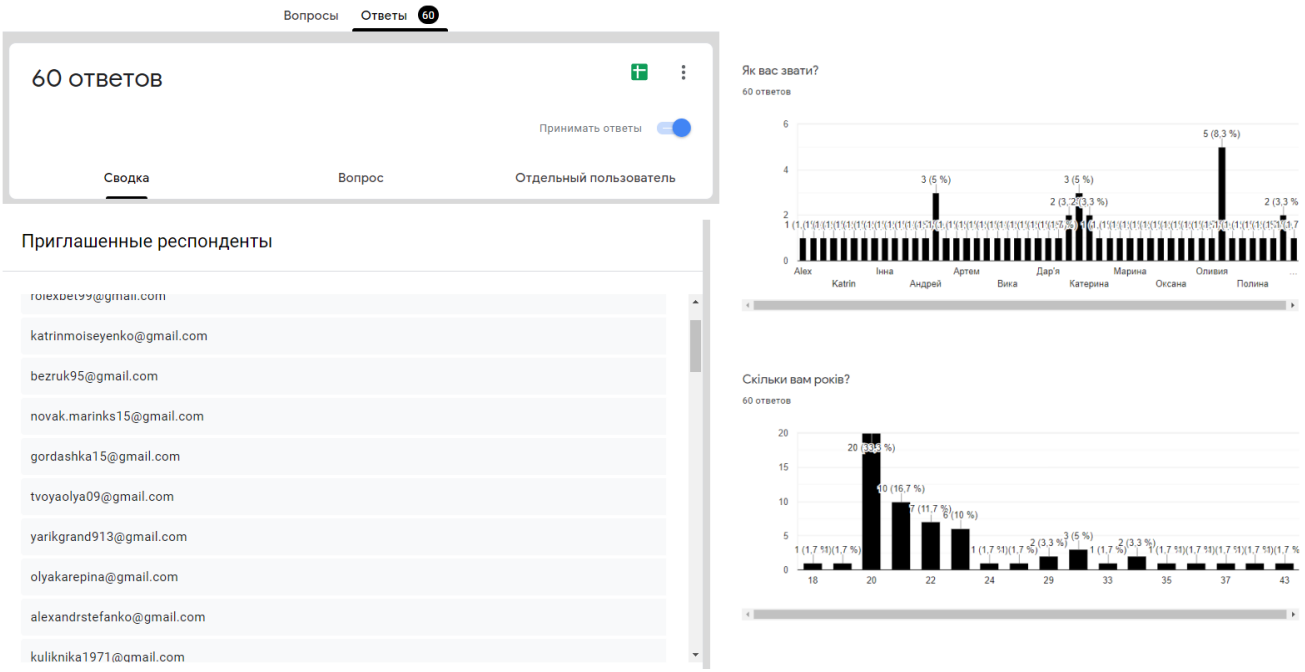

Рисунок 3.3 – Результати тестування

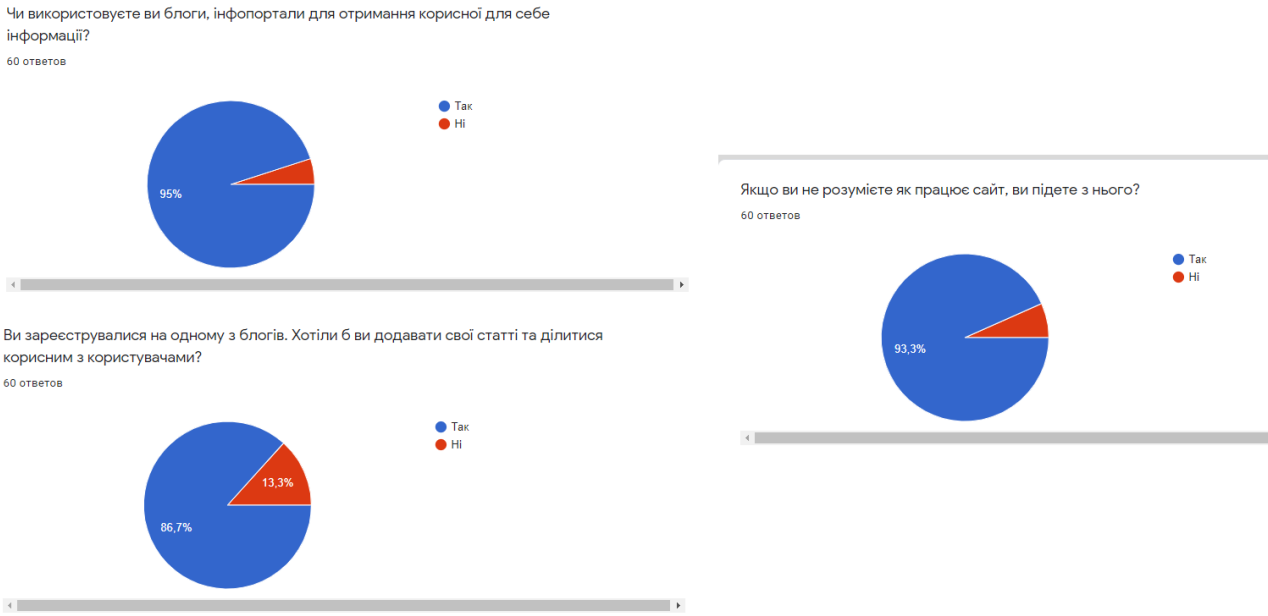

Рисунок 3.4 – Результати тестування

Серед усіх респондентів середній показник віку 20-30 років, де 93,3% відсотка відмовляються продовжувати використовувати сайт, якщо він не є інтуітивно зрозумілим. 86,7% мають потребу в додаванні статтей до блогів або інфопорталів. Та 95% із 100% користуються інтернет-порталами. З усього вище наведеного можна зробитии висновки щодо портрету цільової аудиторії та їх потреби. Розробка веб-додатку буде створюватись, схиляючись до цього аналізу.

Структура сайту

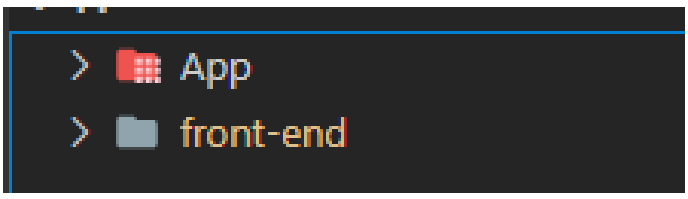

Рисунок 3.5 – Структура проекту

Для зручності розробки і написанню коду. Проект був поділений на дві частини: фронт-ент (клієнтська частина) та бек енд (серверна частина).

На етапі реалізації сайту вже зроблено:

– Створений макет сайту;

– Головна сторінка сайту;

– Категорії статтей;

– Сторінка зі статтею;

– База даних;

– Адмін панель.

3.2 Програмна реалізація

В першу чергу був створений файл package.json через пакетний менеджер npm.

Написання серверної частини для блога

package.json - це стартовий файл з описом проекту, який включає в себе, версію, опис, головний файл, усі підключенні скрипти та бібліоткеки завдяки яким і буде працювати сайт.

Вміст package.json:

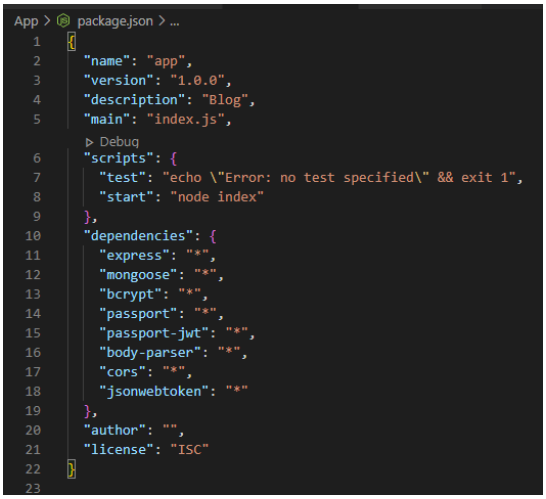

Рисунок 3.6 – Підключені бібліотеки проекту

Express – швидкий, гнучкий, мінімалістичний веб-фреймворк для додатків Node.js , який встановлюється завдякки npm:

Bycrypt – для отримання хеш-паролів, забезпечує безпеку користувачам та шифрує пароль.

Mongoose – це бібліотека JavaScript, що дозволяє вам визначати схеми зі строго-типізованими даними.

Passport – бібліотека для роботи з реєстрацією та авторизацією.

Passport-jwt – необхідна для роботи з реєстрацією та авторизацією користувача.

Body-parser – дозволяє парсити url запити, які передаються разом з пост-даними. Дозволяє відслідковувати дані, які передаються через форму входу.

Cors – механізм, який використовує додаткові HTTP-заголовки, щоб дати можливість агенту користувача отримувати дозволи на доступ до обраних ресурсів з сервера на джерелі (домені). [10]

Jsonwebtoken – це стандарт токена доступу на основі JSON. Використовується для верифікації тверджень.

Веб-токен

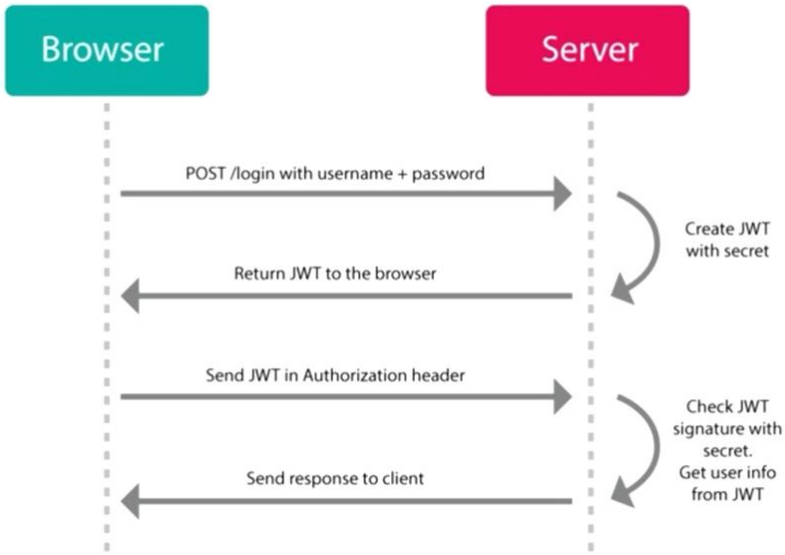

Рисунок 3.7 – Отримання веб токену [10]

Отримання веб токену тільки в тому випадку , якщо реєстрація була проведена успішно. Якщо користувач успішно авторизувався на сайті, то сервер створює токен, який підписується секретним ключем. Далі токен передається

клієнту, який в подальшому використовує даний токен для відправки запитів на сервер і отримує необхідні дані від сервера. Таким чином за допомогою токена, користувач підтверджує свою авторизацію на сайті.

У головному файлі index.js

Рисунок 3.8 – Підключення бібліотек у файлі index.js

Модель користувача

Створення моделі користувача, яка обробляє вхідні дані. На сайті реалізовано можливість реєстрації, авторизації та вихід з особистого кабінету. При реєстрації користувача ми отримуємо всі дані, які будуть ним введено, після чого, отриманні дані надсилаються до бази даних MongoDb. MongoDb – база даних яка зберігає всі елементи в форматі об’єкту. Для додання нового елементу, необхідно спочатку додати в базу даних новий об’єкт.

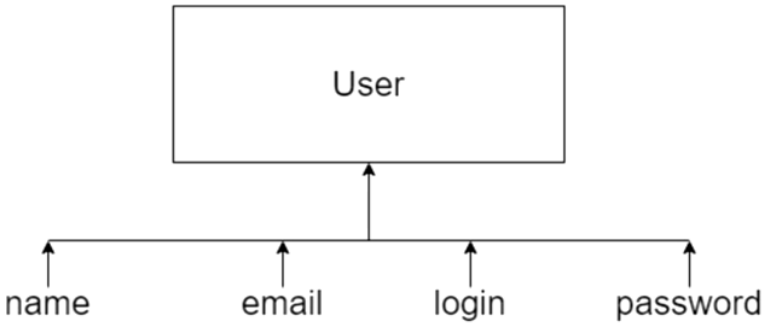

Рисунок 3.9 – Модель користувача

На рисунку 3.9 зображена модель користувача, яка схематично показує, які дані необхідно отримувати при реєстрації від користувача (name, email, login, password)

У файлі user.js створюється схема (шаблон), завдяки якої при зверненні до бази даних mangoose через функцію Schema кожен раз будемо отримувати ті дані, які будуть запитуватись у користувача.

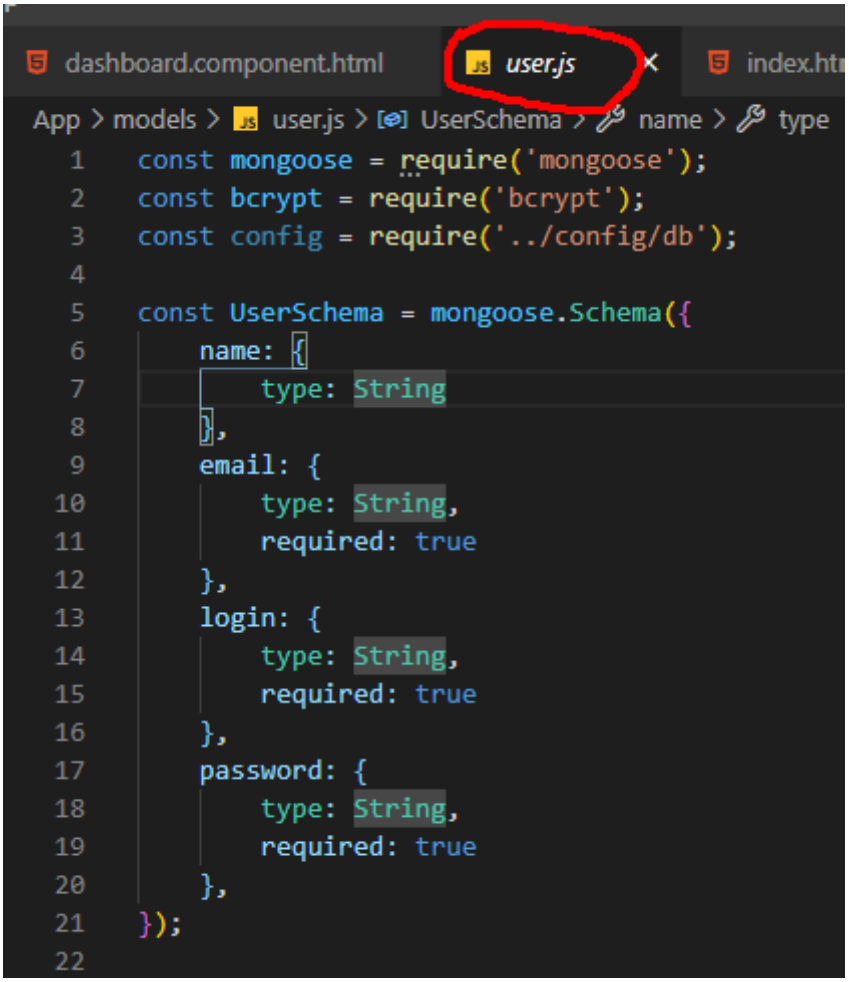

Рисунок 3.10 – Створення схеми моделі користувача

Авторизація користувача

Passport - це проміжне програмне забезпечення для автентифікації для Node.js. Надзвичайно гнучкий та модульний. Авторизація є повністю безпечною, користувач не зможе авторизуватися в аккаунт, якого не існує та зламати систему. [14]

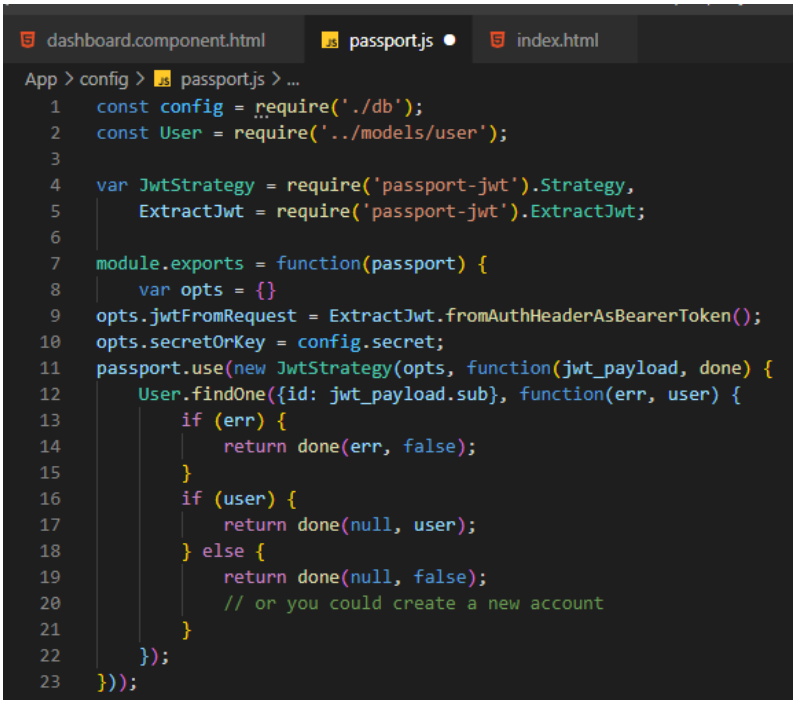

Рисунок 3.11 – Експорт бібліотеки Passport

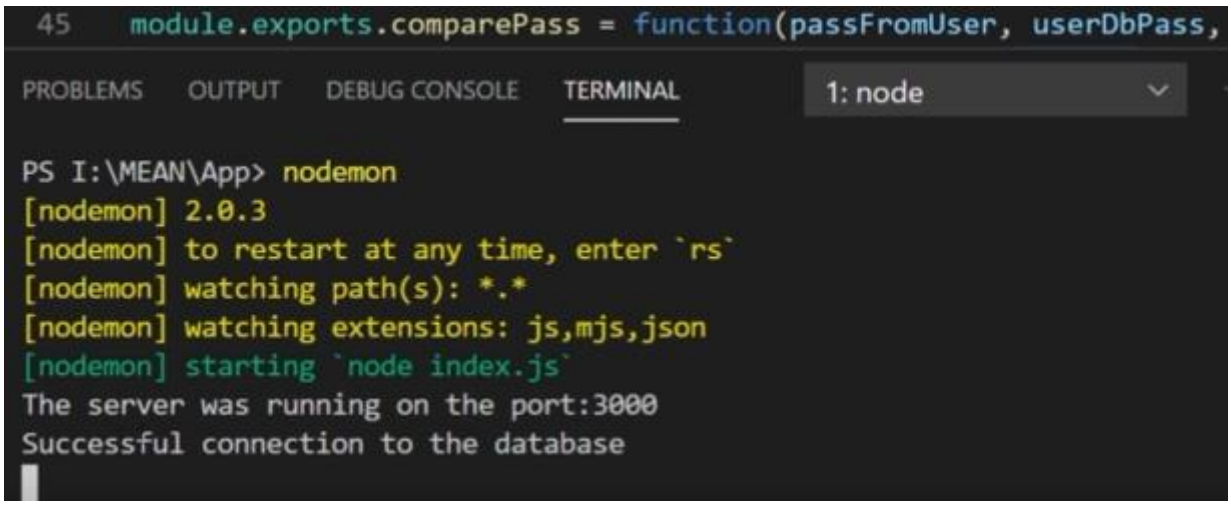

Рисунок 3.12 – Успішний запуск серверу

Написання клієнтської частини для блога

Вся клієнтська частина написана за допомогою бібліотеки Angular. Для створення необхідно встановити Angular CLI. Angular CLI – інтерфейс командної строки для розробки на Angular. [11]

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS I:\MEAN> npm install -g @angular/cli
```

Рисунок 3.13 – Встановлення Angular

Створення шапки сайту за допомогою компонентів Bootstrap4. Та задання роутингів. Синтаксис роутингів в Angular шлях до сторінок вказується за допомогою [routrLink]

```
dashboard.component.html | passport.js | header.component.html | index.html
front-end > src > app > header > header.component.html > div.d-flex.flex-column.flex-md-row.align-items-center.p-3.px-md-4.mb-3.bg-white.border-bo
1  <div class="d-flex flex-column flex-md-row align-items-center p-3 px-md-4 mb-3 bg-white border-bottom box-shadow"
2    <h5 class="my-0 mr-md-auto font-weight-normal">Инсайдер</h5>
3    <nav class="my-2 my-md-0 mr-md-3">
4      <a class="p-2 text-dark" [routerLink]="['/']">Главная</a>
5      <a *ngIf="authService.isAuthenticated()" class="p-2 text-dark" [routerLink]="['/dashboard']">Личный кабинет
6    </nav>
7    <a *ngIf="!authService.isAuthenticated()" class="btn btn-outline-primary mr-3" [routerLink]="['/auth']">Вход<
8    <a *ngIf="!authService.isAuthenticated()" class="btn btn-outline-primary" [routerLink]="['/reg']">Регистрация
9    <a *ngIf="authService.isAuthenticated()" class="btn btn-outline-warning" (click)="logoutUser()">Выход</a>
10 </div>
```

Рисунок 3.14 – Хедер блогу

Валідація при заповнені інпутів. Для реалізації повідомлень щодо некоректного вводу було встановленно додатковий модуль FlashMessages

```
> npm i angular2-flash-messages
```

```
ashboard.component.html | passport.js | app.module.ts | index.html
t-end > src > app > app.module.ts > ...
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { FlashMessagesModule } from 'angular2-flash-messages';

import { AppRoutingModule } from './app-routing.module';
```

Рисунок 3.15 – Імпорт модуля повідомлень

Для реєстрації модуля та коректної праці, необхідно його зареєструвати в масиві Imports.

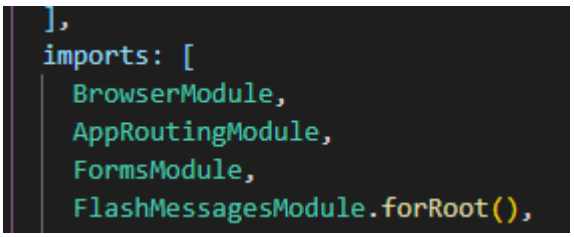

Рисунок 3.16 – Реєстрація модуля в масиві Imports

Для того, щоб повідомлення спливало понад формою реєстрації або авторизації, його необходно помістити до файлу app.components.html

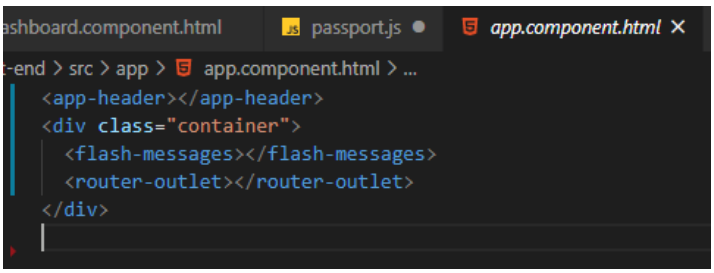

Рисунок 3.17 – Додання до файлу app.components.html

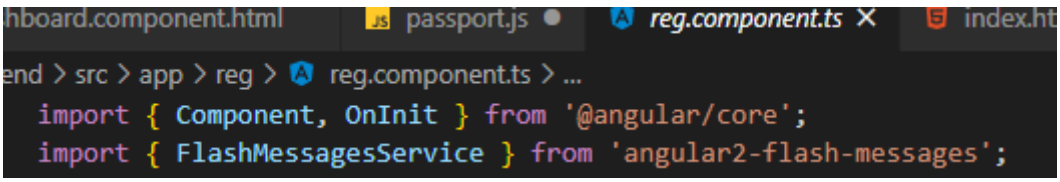

Рисунок 3.18 – Додання до файлу reg.components.ts

Додаємо в конструктор private _flashMessagesService: FlashMessagesService

Для відображення повідомленя передаємо два параметра.

```
ngOnInit() {
    // 1st parameter is a flash message text – текст повідомлення.
```

// 2nd parameter is optional. You can pass object with options. – параметр який є опціональним, який можна передавати або не передавати.

this._flashMessagesService.show('We are in about component!', { cssClass: 'alert-success', timeout: 1000 });

}

Валідація користувача та перевірка на введення даних

```
signUp() {
  const user = {
    name: this.name,
    login: this.login,
    email: this.email,
    password: this.password,
  }
  if(!user.name) {
    this._flashMessagesService.show('Введіть ваше ім'я',
    { cssClass: 'alert-danger', timeout: 3000 });
    return false
  }
  else if(!user.login) {
    this._flashMessagesService.show('Введіть ваш логін',
    { cssClass: 'alert-danger', timeout: 3000 });
    return false
  }
  else if(!user.email) {
    this._flashMessagesService.show('Введіть ваш емейл',
    { cssClass: 'alert-danger', timeout: 3000 });
    return false
  }
  else if(!user.password) {
    this._flashMessagesService.show('Введіть ваш пароль',
    { cssClass: 'alert-danger', timeout: 3000 });
    return false
  }
```

Рисунок 3.19 – Валідація даних

У масиві user отримуємо такі поля як: ім’я, логін, емейл та пароль.

If(!user.name) звертання до об’єкту user після чого до поля name. Якщо у об’єкта поле name являється пустим, виконується наступний код.

this._flashMessagesService.show('Введіть ваше ім’я', - перший параметр текст повідомлення.

{ cssClass: 'alert-danger', timeout: 3000 }); - другой параметр, передаємо об’єкт.

return false – виходимо з методу та не реєструємо користувача

this._flashMessagesService – звертання до сервісу, який був записаний у конструкторі.

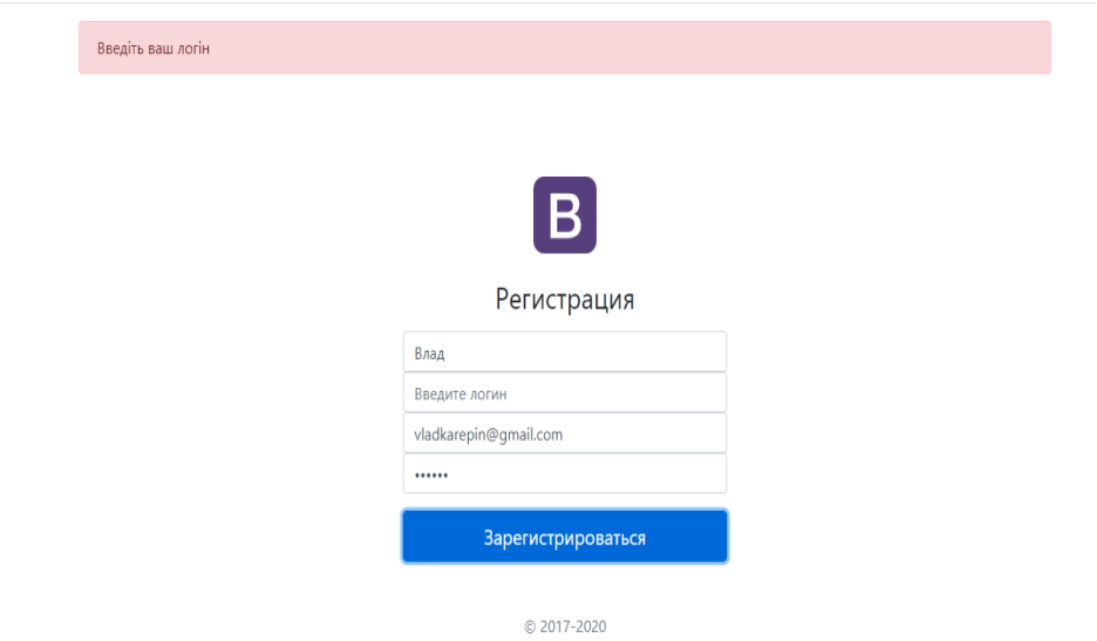

Рисунок 3.20 – Перевірка на введення даних

Сторінка авторизації

Для створення сторінки авторизації необхідно продублювати сторінку реєстрації, та до файлу auth.components.ts імпортувати необхідні сервіси в конструктор:

```
private _flashMessagesService: FlashMessagesService,
private authService: AuthService,
private router: Router
```

Метод SignIn() відповідає за аторизації користувача

```
constructor(
  private _flashMessagesService: FlashMessagesService,
  private authService: AuthService,
  private router: Router
) { }

ngOnInit(): void {
}

signIn() {
  const user = {
    login: this.login,
    password: this.password,
  }
  if(!user.login) {
    this._flashMessagesService.show('Введіть ваш логін',
    { cssClass: 'alert-danger', timeout: 3000 });
    return false
  }
  else if(!user.password) {
    this._flashMessagesService.show('Введіть ваш пароль',
    { cssClass: 'alert-danger', timeout: 3000 });
    return false
  }

  this.authService.authUser(user).subscribe( data => {
    if (!data.success) {
      this._flashMessagesService.show(data.msg,
      { cssClass: 'alert-danger', timeout: 3000 });
```

Рисунок 3.21 – Метод SignIn()

Якщо юзер не був зареєстрований на сайті, визивається метод authUser(user) – метод який буде перенаправляти користувача до реєстрації.

```
registerUser(user) {
  let headers = new Headers()
  headers.append('Content-Type', 'application/json')
  return this.http.post('http://localhost:3000/account/reg', user,
  { headers: headers}).pipe(map(res => res.json()))
}

authUser(user) {
  let headers = new Headers()
  headers.append('Content-Type', 'application/json')
  return this.http.post('http://localhost:3000/account/auth', user,
  { headers: headers}).pipe(map(res => res.json()))
}
```

Рисунок 3.22 – Метод authUser(user)

```
router.post('/auth', (req, res) => {
    const login = req.body.login;
    const password = req.body.password;

    User.getUserByLogin(login, (err, user) => {
        if(err) throw err;
        if(!user) {
            return res.json({success: false, msg: "Нажаль, цей юзер не зареєстрований на сайті :("})
        };
```

Рисунок 3.23 – Відправка пост запроса

У файлі account.js який лежить у папці серверу який відповідає за роутинг та обробку лінків. [13]

router.post('/auth', (req, res) => { - на сторінці авторизації ми приймаємо пост запрос

const login = req.body.login;

const password = req.body.password;

User.getUserByLogin(login, (err, user) => { - після чого визивається функція getUserByLogin

if(err) throw err;

if(!user) {

return res.json({success: false, msg: "Нажаль, цей юзер не зареєстрований на сайті :("}) };

На сторінці авторизації ми приймаємо пост запит, після чого визивається функція getUserByLogin. Дана функція повертає користувача з бази даних, по логіну, який ввів користувач. Якщо користувача з таким логіном не було знайдено у базі даних, функція повертає відповідь у вигляді json об’єкту, який складається з двох елементів: success в першому елементі передає false – як наслідок неуспішна авторизація. У другому елементі передає msg з повідомленням "На жаль, цей юзер не зареєстрований на сайті :("

```
User.comparePass(password, user.password, (err, isMatch) => {
    if(err) throw err;
    if(isMatch) {
        const token = jwt.sign(user.toJSON(), config.secret, {
            expiresIn: 3600 * 24
        });

        res.json({
            success: true,
            token: 'JWT' + token,
            user: {
                id: user._id,
                name: user.name,
                login: user.login,
                email: user.email,
            }
        })
    } else {
        return res.json({success: false, msg: "Паролі не збігаються :("})
    }
})
```

Рисунок 3.24 – Функція comparePass

Якщо користувач був авторизований, визивається функція comparePass - порівнює пароль, який ввів користувач з паролем, який лежить в базі даних MongoDB. Якщо паролі збігаються, створюється токен, після якого відправляється відповідь у вигляді json об'єкту. З такими елементами:

```
res.json({
    success: true,
    token: 'JWT' + token,
    user: {
        id: user._id,
        name: user.name,
        login: user.login,
        email: user.email,                }
```

success: true – успішна вторизація

token: 'JWT' + token – ключ авторизації

user: { } – передає самого користувача

Якщо паролі не збігаються, повертаємо за допомогою json об’єкту відповідь, який складається з двох елементів: success в першому елементі передає false – як наслідок неуспішна авторизація. У другому елементі передає msg з повідомленням "Паролі не збігаються :("

Головна сторінка

На головній сторінці знаходяться нові додані статті та сортування за категоріями.

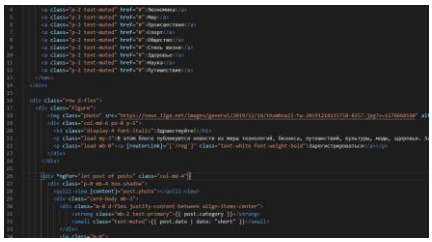

Рисунок 3.25 – Код головної сторінки

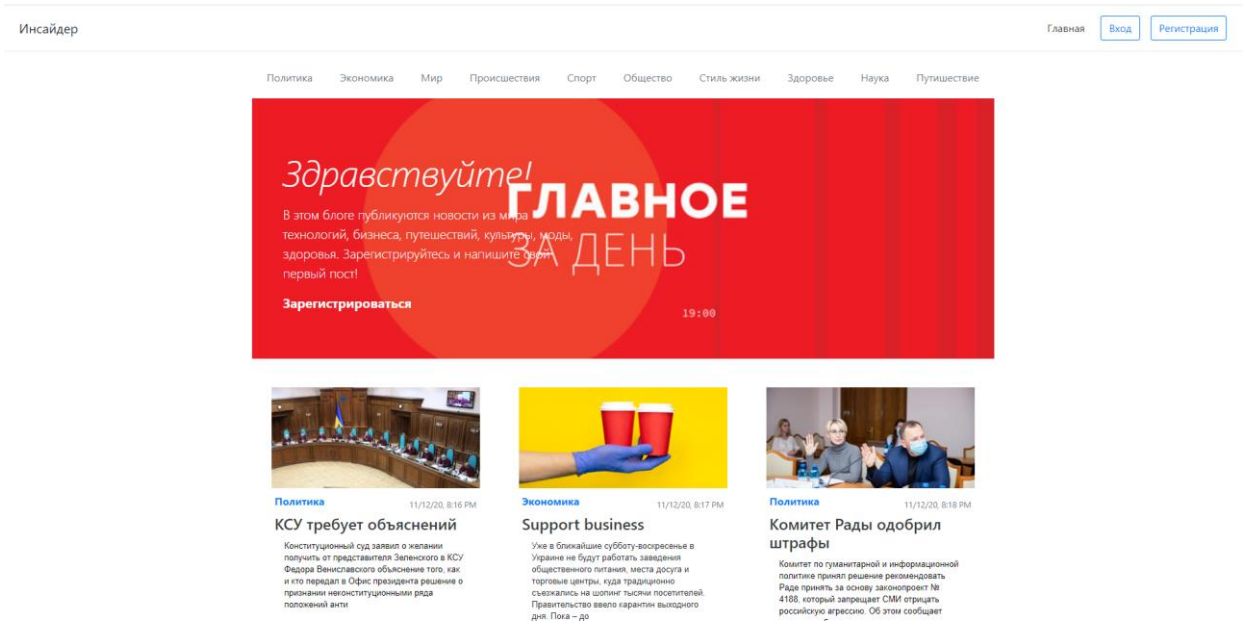

Рисунок 3.26 – Загальний вигляд головної сторінки

Форма створення статтей

Форма створення статтей знаходиться у особистому кабінеті. Тільки авторизований користувач має можливість додавати свої статті.

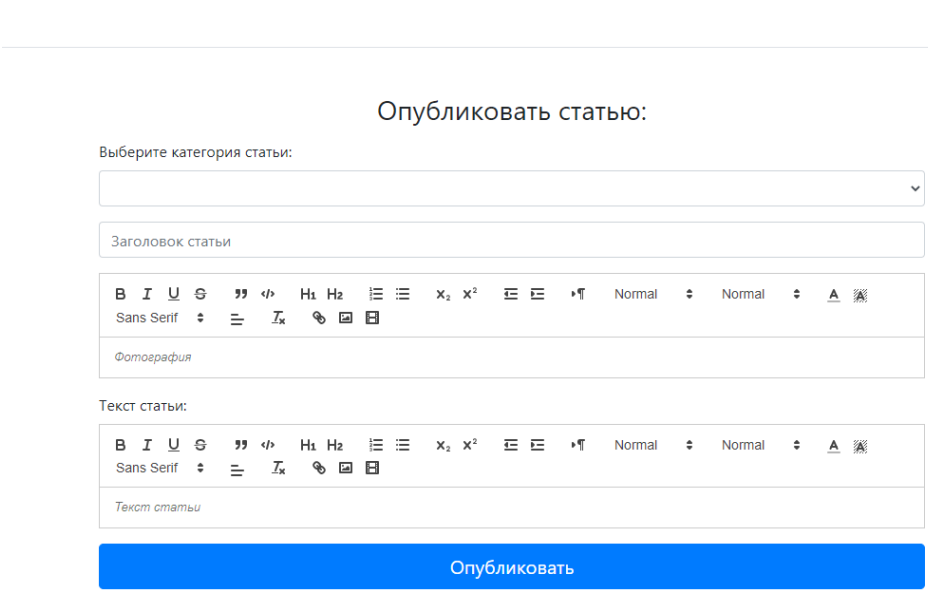

Рисунок 3.27 – Вигляд особистого кабінету

```
) { }

ngOnInit(): void {
}

createPost() {
  const post = {
    category: this.category,
    title: this.title,
    photo: this.photo,
    text: this.text,
    author: JSON.parse(localStorage.getItem("Користувач")).login,
    date: new Date()
  }
  if(!post.category) {
    this._flashMessagesService.show('Виберіть категорію',
    { cssClass: 'alert-danger', timeout: 3000 });
    return false
  }
  else if(!post.title) {
    this._flashMessagesService.show('Введіть заголовок',
    { cssClass: 'alert-danger', timeout: 3000 });
    return false
```

Рисунок 3.28 – Метод createPost()

Метод createPost() відповідає за створення статтей на сайті. Об’єкт включає в себе всі змінні які були додані в файл dashboard.components.ts [15]

createPost() {

```
const post = {
  category: this.category,
  title: this.title,
  photo: this.photo,
  text: this.text,
  author: JSON.parse(localStorage.getItem("Користувач")).login,
  date: new Date()
}
```

Для відображення автора статті, звертаємось до localStorage. Метод getItem() дозволяє отримувати елемент з ключем "user". – є рядком. За допомогою JSON.parse рядок перетворюється в об’єкт, після чого у об’єкта дістаємо поле login (автора статті). date: new Date() – дата створення статті.

```
}
if(!post.category) {
  this._flashMessagesService.show('Виберіть категорію',
  { cssClass: 'alert-danger', timeout: 3000 });
  return false
}
else if(!post.title) {
  this._flashMessagesService.show('Введіть заголовок',
  { cssClass: 'alert-danger', timeout: 3000 });
  return false
}
else if(!post.photo) {
  this._flashMessagesService.show('Додайте фото',
  { cssClass: 'alert-danger', timeout: 3000 });
  return false
}
```

Рисунок 3.29 – Перевірка на валідність заповнення контенту

Сторінка статті

Для створення сторінки статті, було створено новий компонент за допомогою термінала та команди Angular ng g c post -skipTests

```
Try the new cross-platform PowerShell https://ak

PS I:\MEAN> cd front-end
PS I:\MEAN\front-end> ng g c post --skipTests
```

Рисунок 3.30 – Створення компоненту статті

dashboard.component.ts | app-routing.module.ts

front-end > src > app > app-routing.module.ts > ...

```
1  import { NgModule } from '@angular/core';
2  import { Routes, RouterModule } from '@angular/router';
3  import { HomeComponent } from './home/home.component';
4  import { RegComponent } from './reg/reg.component';
5  import { AuthComponent } from './auth/auth.component';
6  import { DashboardComponent } from './dashboard/dashboard.component';
7  import { AuthGuard } from './auth.guard'
8  import { PostComponent } from './post/post.component';
9
10 const routes: Routes = [
11   {path: '', component: HomeComponent},
12   {path: 'reg', component: RegComponent},
13   {path: 'auth', component: AuthComponent},
14   {path: 'post/:id', component: PostComponent},
15   {path: 'dashboard', component: DashboardComponent, canActivate: [Aut
```

Рисунок 3.31 – Додання сторінки до роутингу

У головному файлі app-routing.module.ts прописано роутінг до нового компоненту post (сторінку статті).

{path: 'post/:id', component: PostComponent},

При переході за шляхом 'post/:id – id поста забираємо з бази даних MongoDB завантажується необхідний пост. [12]

У HTML файлі home.component.html формуються всі пости.

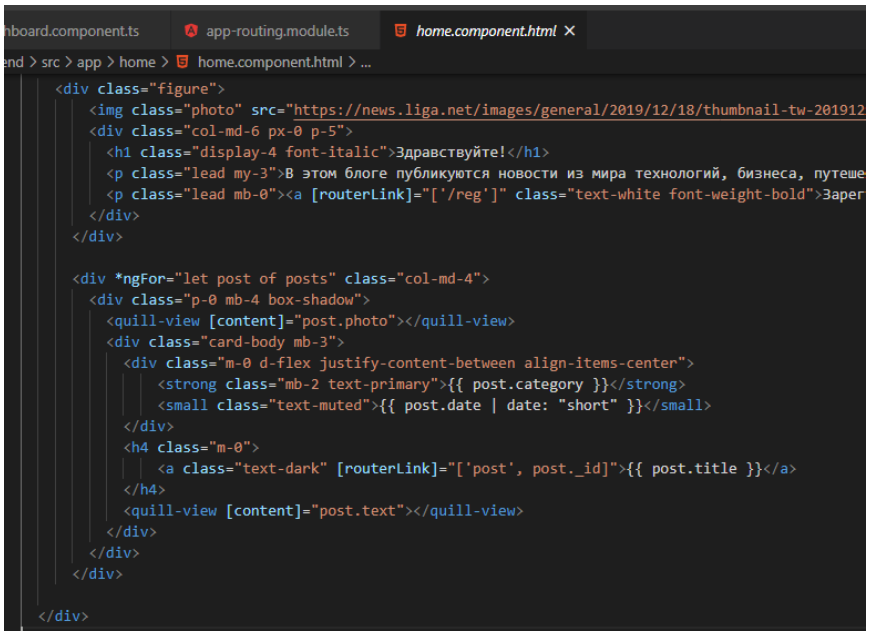

Рисунок 3.32 – Формування статті у HTML файлі

Завдяки цій команді:

{{ post.title }}

Ми отримуємо шлях post, та поле id. В базі даних MongoDB – вони храняться у форматі: post._id.

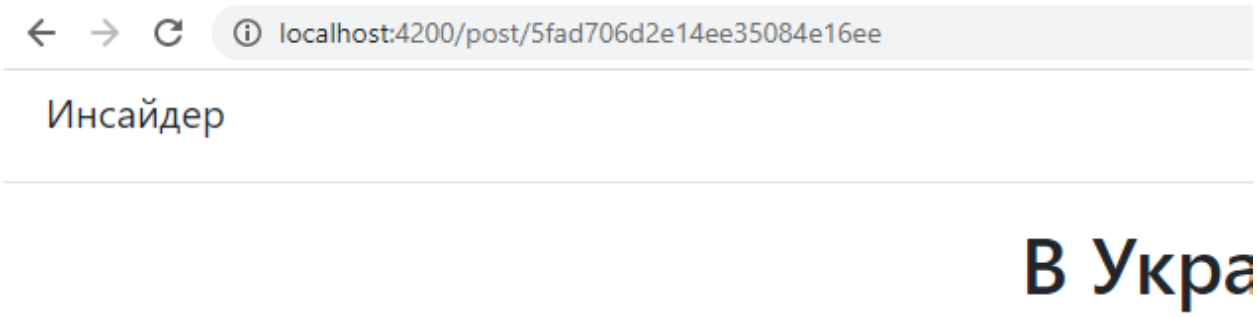

Рисунок 3.33 – Унікальний id статті

В Украине ввели карантин выходного дня

Удалить статью

Общество

Author: @andrei, Date: 11/12/20, 8:27 PM

Премьер Денис Шмыгаль отметил, что ограничения предварительно вводятся на три уик-энда подряд – с 14 ноября и до 30 ноября. Согласно проекту постановления о карантине выходного дня, с 00:00 субботы и до 00:00 понедельника будут запрещены:

1. Прием посетителей заведениями общепита (рестораны, кафе, бары и тд);
2. Прием посетителей торгово-развлекательными центрами и другими развлекательными учреждениями;
3. Работа субъектов хозяйственной деятельности, которые работают в сфере торговли и занимаются бытовым обслуживанием населения, **кроме:**

– торговли продуктами в магазинах, 60% торговой площади которых предназначены для торговли продуктами питания, топливом, лекарственными средствами и изделиями медицинского предназначения, ветеринарными препаратами, кормами;
- предоставления финансовых услуг, деятельности финансовых учреждений, деятельности по инкассации и перевозке валютных ценностей; работы операторов почтовой связи, а также медицинской и ветеринарной практики, деятельность автозаправочных комплексов (без зон питания), деятельности по ремонту и техобслуживанию транспорта, деятельности по техобслуживанию регистраторов расчетных операций, деятельность по ремонту компьютеров, бытовых изделий, предметов личного употребления;
– торговой деятельности и предоставлении услуг общепита по адресной доставке.
4. Деятельность учреждений культуры и проведение массовых культурных мероприятий. При этом разрешается производство аудиовизуальных произведений на ограниченной съемочной площадке.

Также в Украине отменили адаптивный карантин: теперь по всей стране будут действовать правила оранжевой зоны с некоторыми усилениями.

Глава Минздрава Максим Степанов объяснил, что главная цель ужесточения ограничений – разорвать цепь передачи болезни и избежать коллапса медицинской системы. Он также говорил, что лучшим с эпидемиологической точки зрения решением было бы ввести локдаун на три недели, но это может быть пагубно для экономики.

• Идею карантина выходного дня критиковал министр культуры Александр Ткаченко. Он говорил, что это может

Рисунок 3.34 –Вигляд сторінки з статтею

ВИСНОВКИ

Отже, інформаційні технології відіграють важливу роль в житті кожного з нас. Вони використовуються постійно і важко уявити хоча б один день без них. Це проявляється як у повсякденному житті звичайної людини, так і в робочих моментах веб-розробника. Інформація заполонила весь простір. Кожного дня ми стикаємося з величезним її потоком. Найчастіше джерелом є Інтернет-ресурси. Для знаходження найрізноманітнішої інформації існує колосальна кількість інтернет-блогів та інфопорталів. Попит на їх розробку зростає кожного дня , так як збільшується кількість інформації і людей, які хочуть її донести – блогерів. Для цього існують веб-розробники та веб-дизайнери, мета яких не просто створити сайт, але й якісно його продумати, щоб забезпечити зрозуміле та легке використання.

На сьогоднішній день є багато комплексів серверного програмного забезпечення, що використовуються для веб-розробки. Однією з сучасних тенденцій веб-побудови є використання найрізноманітніших технологій для створення вдосконалених веб-додатків. Але, як зазначалося вище в роботі, є деякі недоліки, які свідчать про некоординовані та невмілі засоби розробки, які шкодять як розробнику, так і клієнту. Для створення якісних програм світового класу потрібні такі складові як системи баз даних, інтерфейсні фреймворки, бібліотеки та сервери. Вже створено багато технологій для розробки, які допомагають задовольнити потребу в цифровій трансформації. Стек MEAN – одна з таких технологій, яка вважається однією з найкращих.

Дана робота присвячена розробці веб-додатку з використанням стеку MEAN. За допомогою цієї технології може бути вирішено ряд проблем, що пов’язані з використанням блогу з метою пошуку інформації. Зараз існує велика кількість найрізноманітніших ресурсів, де здавалося б, можна знайти відповіді на всі запитання, але є безліч перешкод. Найбільш розповсюдженими є застаріла інформація, незрозумілий інтерфейс та функціонал.

В даній роботі є рішення цих проблем, а саме створення блогу за допомогою стеку MEAN. Завдяки великій кількості переваг цієї технології веб-розробник має змогу створити дійсно потужний, зручний блог, де користувач знайде всю необхідну інформацію, не витративши часу на розуміння того, як влаштований цей додаток.

В подальшому, можна покращити роботу блогу, додавши більше функціоналу у вигляді фільтрів, пошук за категоріями, додаток користувачів в друзі і рейтинг на статті.

СПИСОК ЛІТЕРАТУРИ

1. Кроудер Д. Создание веб-сайта для чайников. – Харків: Ранок, 2016. - 336 с.

2. Петюшкин А. HTML. Экспресс-курс. – Київ:Фенікс-паблішинг, 2016. - 192 с.

3. Гультяев А. К. Уроки Web-мастера. Технология. Дизайн. Инструменты. – Харків:ПЕТ, 2015. - 448 с.

4. Джереми К. HTML5 для веб-дизайнеров. – Київ:Віват, 2015. - 410 с.

5. Дуванов А.А. Web-конструирование. DHTML. – Одесса:Освіта України, 2014. - 555 с.

6. Лоусон, Б. Изучаем HTML5, Ангуляр. Библиотека специалиста. Шарп. – Харків:Ранок, 2016. - 304 с.

7. Прохоренок Н. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера. – Київ:Основи, 2018. – 912 с.

8. Яков Ф. Моисеев А., Angular и TypeScript. Сайтостроение для профессионалов, – Одесса: Ранок, 2018. – 464 с.

9. Фримен А. Angular для профессионалов, – Київ: Віват, 2018 – 409 с.

10. Фрісбі М. Angular 2 Cookbook: Discover over 70 recipes that provide the solutions you need to know to face every challenge in Angular 2, – Харків: Фенікс-пабліш, 2017: – 388 с.

11. Інтернет-статья https://telegraf.design/proyektuvannya-dyzajn-systemy/

12. Інтернет-ресурс habr https://habr.com/ru/company/piter/blog/279237/

13. Інтернет-ресурс https://code.tutsplus.com/ru/tutorials/introduction-to-the-mean-stack

14. Інтернет-ресурс https://otakoyi.ua/osobennosti-sozdaniya-web-s-pomoshchyu-mean-stek

15. Інтернет ресурс https://metanit.com/web/nodejs/1.1.php

ДОДАТОК

```
<div class="nav-scroller py-1 mb-2">
    <nav class="nav d-flex justify-content-between">
      <a class="p-2 text-muted" href="#">Политика</a>
      <a class="p-2 text-muted" href="#">Экономика</a>
      <a class="p-2 text-muted" href="#">Мир</a>
      <a class="p-2 text-muted" href="#">Происшествия</a>
      <a class="p-2 text-muted" href="#">Спорт</a>
      <a class="p-2 text-muted" href="#">Общество</a>
      <a class="p-2 text-muted" href="#">Стиль жизни</a>
      <a class="p-2 text-muted" href="#">Здоровье</a>
      <a class="p-2 text-muted" href="#">Наука</a>
      <a class="p-2 text-muted" href="#">Путишествие</a>
    </nav>
  </div>
  <div class="row d-flex">
    <div class="figure">
        <img class="photo" src="https://news.liga.net/images/general/2019/12/18/thu
mbnail-tw-20191218115758-8257.jpg?v=1576668160" alt="">
        <div class="col-md-6 px-0 p-5">
          <h1 class="display-4 font-italic">Здравствуйте!</h1>
          <p class="lead my-
3">В этом блоге публикуются новости из мира технологий, бизнеса, путешествий, культ
уры, моды, здоровья. Зарегистрируйтесь и напишите свой первый пост!</p>
          <p class="lead mb-0"><a [routerLink]="['/reg']" class="text-white font-
weight-bold">Зарегистрироваться</a></p>
        </div>
      </div>
      <div *ngFor="let post of posts" class="col-md-4">
        <div class="p-0 mb-4 box-shadow">
          <quill-view [content]="post.photo"></quill-view>
          <div class="card-body mb-3">
            <div class="m-0 d-flex justify-content-between align-items-center">
                <strong class="mb-2 text-primary">{{ post.category }}</strong>
                <small class="text-muted">{{ post.date | date: "short" }}</small>
            </div>
            <h4 class="m-0">
                <a class="text-
dark" [routerLink]="['post', post._id]">{{ post.title }}</a>
            </h4>
            <quill-view [content]="post.text"></quill-view>
          </div>
        </div>
      </div>
   </div>
```

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { RegComponent } from './reg/reg.component';
import { AuthComponent } from './auth/auth.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { AuthGuard } from './auth.guard'
import { PostComponent } from './post/post.component';

const routes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'reg', component: RegComponent},
  {path: 'auth', component: AuthComponent},
  {path: 'post/:id', component: PostComponent},
  {path: 'dashboard', component: DashboardComponent, canActivate: [AuthGuard]},
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

<body class="container">
    <form (submit)="createPost()" class="form-signin">
      <h1 class="h3 mb-3 text-center font-weight-normal">Опубликовать статью:</h1>
      <label>Выберите категория статьи:</label>
      <select [(ngModel)]="category" class="form-control mb-
3" name="category" id="category">
          <option value="Политика">Политика</option>
          <option value="Экономика">Экономика</option>
          <option value="Мир">Мир</option>
          <option value="Происшествия">Происшествия</option>
          <option value="Спорт">Спорт</option>
          <option value="Общество">Общество</option>
          <option value="Стиль жизни">Стиль жизни</option>
          <option value="Здоровье">Здоровье</option>
          <option value="Наука">Наука</option>
          <option value="Путишествие">Путишествие</option>
      </select>
      <input [(ngModel)]="title" type="title" name="title" id="inputTitle" class="f
orm-control mb-3" placeholder="Заголовок статьи">
      <quill-
editor [(ngModel)]="photo" name="photo" placeholder="Фотография"></quill-editor>
      <label class="mt-3">Текст статьи:</label>
      <quill-
editor [(ngModel)]="text" name="text" placeholder="Текст статьи"></quill-editor>
      <div class="checkbox mb-3">
```

```
      </div>
      <button class="btn btn-lg btn-primary btn-block mb-
5" type="submit">Опубликовать</button>
    </form>
  </body>

const express = require('express');
const router = express.Router();
const User = require('../models/user');
const Post = require('../models/post');
const passport = require('passport');
const jwt = require('jsonwebtoken');
const config = require('../config/db');

router.post('/reg', (req, res) => {
    let newUser = new User({
        name: req.body.name,
        email: req.body.email,
        login: req.body.login,
        password: req.body.password,
    });

    User.addUser(newUser, (err, user) => {
        if(err) {
            res.json({success: false, msg: "User has not been added."})
        }
        else {
            res.json({success: true, msg: "User has been added."})
        }
    });
});

router.post('/auth', (req, res) => {
    const login = req.body.login;
    const password = req.body.password;

    User.getUserByLogin(login, (err, user) => {
        if(err) throw err;
        if(!user) {
            return res.json({success: false, msg: "Нажаль, цей юзер не зареєстрован
ий на сайті :("})
        };

        User.comparePass(password, user.password, (err, isMatch) => {
            if(err) throw err;
            if(isMatch) {
                const token = jwt.sign(user.toJSON(), config.secret, {
                    expiresIn: 3600 * 24
                });
```

```
                res.json({
                    success: true,
                    token: 'JWT' + token,
                    user: {
                        id: user._id,
                        name: user.name,
                        login: user.login,
                        email: user.email,
                    }
                })
            } else {
                return res.json({success: false, msg: "Паролі не збігаються :("})
            }
        })
    })
});

router.post('/dashboard', (req, res) => {
    let newPost = new Post({
        category: req.body.category,
        title: req.body.title,
        photo: req.body.photo,
        text: req.body.text,
        author: req.body.author,
        date: req.body.date,
    });

    Post.addPost(newPost, (err, user) => {
        if(err) {
            res.json({success: false, msg: "Post has not been added."})
        }
        else {
            res.json({success: true, msg: "Post has been added."})
        }
    });
});

module.exports = router;
```