Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Медіасистем та технологій
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження корисності фреймворків для Front-end розробки
(тема)

Виконав:
студент 2 курсу, групи ТЕМВм-21-1


Чахальян Д.Є.

Спеціальності 186 Видавництво та поліграфія
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма
Технології електронних мультимедійних видань

Керівник 
проф. Кулішова Н.Є.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри МСТ _____ Дейнеко Ж.В.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Медіасистем та технологій
Рівень вищої освіти другий (магістерський)
Спеціальність 186 Видавництво та поліграфія
(код і повна назва)
Тип програми Освітньо-професійна
Освітня програма Технології електронних мультимедійних видань
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
« 31 » жовтня 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Чахальяну Денису Євгеновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження корисності фреймворків для Front-end розробки

Затверджена наказом університету від 31.10.2022 № 1432 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 6 грудня 2022 р

3. Вихідні дані до роботи
Сайти-блоги; Мови розробки: HTML5, CSS3, SCSS, JavaScript, TypeScript;
Front-end фреймворки: Angular, Vue.js; Середовище розповсюдження: Інтернет.

4. Перелік питань, що потрібно опрацювати в роботі
Огляд літератури за темою дослідження і аналіз стану проблеми і постановка задачі дослідження; Важливість використання Front-end фреймворків; Відмінності процесу розробки; Експериментальна частина; Економічна частина; Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій
Огляд літератури за темою дослідження і аналіз стану проблеми і постановка задачі дослідження; Важливість використання Front-end фреймворків; Відмінності процесу розробки; Експериментальна частина; Економічна частина; Висновки.

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	проф. Кулішова Н.Є.		21.11.2022
Економічна частина	проф. Полозова Т.В.		19.11.2022

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за темою дослідження і аналіз стану проблеми і постановка задачі дослідження	01.11.2022	виконано
2	Важливість використання Front-end фреймворків	06.11.2022	виконано
3	Відмінності процесу розробки	11.11.2022	виконано
4	Експериментальна частина	17.11.2022	виконано
5	Економічна частина	23.11.2022	виконано
6	Оформлення пояснювальної записки	01.12.2022	виконано
7	Оформлення графічної частини	04.12.2022	виконано
8	Захист кваліфікаційної роботи	07.12.2022	виконано

Дата видачі завдання 31 жовтня 2022 р.

Студент _____ Чахальян Д.Є.
(підпис)

Керівник роботи _____ проф. Кулішова Н.Є.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 79 с., 67 рис., 11 табл., 26 джерел.

КЛЮЧОВІ СЛОВА. WEB-СТОРІНКА, HTML5, CSS3, SCSS, JAVASCRIPT, TYPESCRIPT, ANGULAR, VUE.JS, FRONT-END, БІБЛІОТЕКИ, БЕЗПЕКА, ОПТИМІЗАЦІЯ, МАСШТАБОВАНІСТЬ, РОЗМІР ЗБІРКИ, АРХІТЕКТУРА.

У даній кваліфікаційній роботі вирішені такі питання як: огляд літератури за темою дослідження і аналіз стану проблеми і постановка задачі дослідження; важливість використання Front-end фреймворків; відмінності процесу розробки; способи підключення Front-end фреймворків; проведення експериментальної частини; розробка сайтів використовуючи фреймворки для Front-end розробки Angular та Vue.js; вибір методу проведення експерименту та його проведення з використанням обраного методу; проведення економічного обґрунтування доцільності проведення науково-дослідної роботи.

ABSTRACT

The explanatory note to the qualification work: 79 p., 67 pic., 11 tab., 26 references.

KEYWORDS. WEB PAGE, HTML5, CSS3, SCSS, JAVASCRIPT, TYPESCRIPT, ANGULAR, VUE.JS, FRONT-END, LIBRARIES, SECURITY, OPTIMIZATION, SCALE, ASSEMBLY SIZE, ARCHITECTURE.

In this qualifying work, such issues are resolved as: a review of the literature on the research topic and an analysis of the state of the problem and the formulation of the research problem; the importance of using Front-end frameworks; development process differences; ways to connect Front-end frameworks; conducting the experimental part; website development using front-end development frameworks Angular and Vue.js; the choice of the method of conducting the experiment and its conduct using the selected method; conducting an economic justification for the feasibility of research work.

ЗМІСТ

С.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

HTML	HyperText Markup Language
CSS	Cascading Style Sheets
npm	Node Package Manager
JS	JavaScript
TS	TypeScript
ID	Identity Document
PHP	Personal Home Page
URL	Uniform Resource Locator
SPA	Single Page Application
DOM	Document Object Model
ПК	Персональний комп'ютер
ПЗ	Програмне забезпечення
HTTP	Hypertext Transfer Protocol
AOT	Ahead of Time
gzip	GNU’s Not UNIX Zip
API	Application Programming Interface
SCSS	Sassy CSS
XML	Extensible Markup Language

ВСТУП

Актуальність виконаної кваліфікаційної роботи полягає в тому, що зараз існує дуже багато фреймворків для Front-end розробки сайтів, та перед початком створення проекту актуальним є питання – який саме фреймворк використовувати для того, щоб отримати більш якісний продукт.

Створюючи цифровий продукт, потрібно заздалегідь думати про його розвиток та подальшу долю. Але якість розробки може суттєво відрізнятись, залежно від обраного фреймворку. Найпопулярнішими сьогодні є Vue чи Angular. Перший – простий у освоєнні та швидко розвивається, другий – самодостатній і дозволяє без проблем створювати справді унікальні та складні продукти. Тому, під час виконання кваліфікаційної роботи, буде більш досконало вивчено найпопулярніші фреймворки, такі, як Angular та Vue.js.

Об’єктом дослідження є процес розробки сайту. Предмет дослідження – фреймворки Angular та Vue.js для Front-end розробки, метод експертного оцінювання.

Метою кваліфікаційної роботи є підвищення показника продуктивності сайтів завдяки вибору найбільш ефективного Front-end фреймворку. В роботі проведено огляд літератури за темою дослідження, аналіз стану проблеми і поставлено задачу дослідження, досліджено важливість використання Front-end фреймворків та відмінності процесу розробки. Після чого виконано експериментальну частину, яка складається зі створення сайтів, вибору та обґрунтування методу проведення експерименту та самого проведення експерименту. Результати дослідження покажуть, який з Front-end фреймворків буде більш придатним для сучасних проектів.

1 ОГЛЯД ЛІТЕРАТУРИ ЗА ТЕМОЮ ДОСЛІДЖЕННЯ І АНАЛІЗ СТАНУ ПРОБЛЕМИ. ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Визначення фреймворку. Особливості

Фреймворк – це програмна платформа, що визначає структуру програмної системи; це програмне забезпечення, яке полегшує розробку та об'єднання різних компонентів великого програмного проекту [1].

Створення програмного забезпечення справді складний процес. Він включає безліч завдань, таких як проектування, кодування і тестування. Тільки при кодуванні програмістам доводиться піклуватися про наявність в сайті розділу оголошень, синтаксису, виразів, складання сміття, винятків та багато іншого.

Програмні фреймворки полегшують життя розробникам веб-сайтів та програм, дозволяючи їм контролювати процес розробки програмного забезпечення за допомогою єдиної платформи.

Переваги використання програмних фреймворків:

– економія часу;

– масштабований код;

– безпека.

1.2 Front-end фреймворки

Front-end, мовою веб-додатків, відноситься до області програми або веб-сайту, з якою відвідувачі взаємодіють безпосередньо. Інтерфейсні веб-платформи знижують складність роботи програмістам, яким доводиться вручну створювати код, щоб описувати поведінку та взаємодію між користувачами та сайтом. Ці фреймворки пропонують попередньо написані коди, які розробники можуть використовувати як готову базу [2].

Найбільш популярними фреймворками для проектів сьогодні є Angular та Vue.js. Кожен з них має свої особливості, переваги та недоліки, пристосовані під певні завдання. Тому розглянемо саме ці продукти [3].

1.3 Angular

Angular є одним із найстаріших фреймворків. Він з'явився у 2009 році та пройшов шлях перетворень. Сьогодні це багатофункціональний інструмент для створення веб-додатків та складних за своєю структурою та архітектурою продуктів.

Належить ця технологія компанії Google, має величезну підтримку. За рахунок цього Angular постійно розвивається, має велику активну спільноту та високий рівень бази знань. Він має відкритий вихідний код і працює з мовою програмування TypeScript.

Головна особливість цього фреймворку – класово-модульний підхід до розробки. Angular має велику кількість готових рішень, які дозволяють масштабувати проект надалі максимально швидко та безпечно. Адже строга структура будь-якого проекту зменшує ймовірність помилок під час роботи. Виявляється певна строгість та послідовність у розробці, що обмежує дії розробника. Тому інструмент не надто гнучкий. Але це позитивно впливає на покращення працездатності та простоту тестування.

Angular є самодостатнім рішенням і не потребує використання додаткових інструментів. Але поріг входу в розробку досить високий, фахівцю потрібно розуміти TypeScript, JavaScript, бібліотеку RxJS. Це такий потужний інструмент, призначений для створення великих продуктів для бізнесу.

Переваги:

– код виходить чистим, ймовірність помилок сильно знижується;

– можна повторно використати готові частини коду;

– регулярно виходять оновлення, тому високий рівень безпеки та якості веб-продуктів;

– окремі частини коду швидко тестуються;

– висока продуктивність, програма може бути великою, але при цьому працювати швидко;

– двостороннє зв'язування даних дозволяє миттєво відображати зміни на веб-сторінці в інтерфейсі користувача;

– велика кількість простих готових шаблонів на вирішення типових завдань;

– класово-модульний підхід дозволяє масштабувати готові продукти;

– має просту інтеграцію із сторонніми компонентами;

– здатний вирішити практично будь-яке завдання.

Переваг багато, але є недоліки. Фреймворк має жорстку структурованість, тому створювати «креативні» рішення у ньому важко. Він більше підходить для стандартних програм, від яких потрібно виконувати стандартні процеси. Крім того, сам фреймворк громіздкий. Навіть для створення простого сайту, для якого знадобиться 10% можливостей Angular, доведеться встановлювати весь пакет.

Ще один великий недолік – високий поріг входу. Навчальних матеріалів не так багато, база знань має прогалини, тому програмісти нерідко навчаються на «власних помилках» [4].

Сьогодні позиції цього середовища розробки значно знизилися. Багато розробників стали скаржитися на нього набагато частіше, а спільнота The State of Javascript опублікувала результати опитувань, згідно з якими Angular сьогодні вважають «вмираючим» фреймворком. Тим не менш, на фахівців, які знають цей продукт, все ще є великий попит [5].

Angular надає фреймворк для створення клієнтських додатків. Перш за все він націлений на розробку SPA-рішень, тобто односторінкових додатків. В цьому плані Angular є спадкоємцем іншого фреймворка AngularJS. У той же час Angular це не нова версія AngularJS, а принципово новий фреймворк [6].

Спочатку Angular створювався як друга версія AngularJS. Angular 2 був переписаний з нуля на TypeScript, має іншу архітектуру і не є зворотно сумісним з AngularJS, у зв'язку з чим для запобігання плутанини було вирішено розвивати його як окремий фреймворк, нумерація версій якого починається з другої [7].

1.4 Vue.js

Vue.js є JavaScript фреймворк, який з'явився в 2015 році. Його автор – Еван Ю, на той момент був співробітником компанії Google. Спочатку інструмент відносили до аматорських, оскільки тривалий час він підтримувався лише своїм творцем. Але поступово співтовариство Vue зростало і міцніло.

Сьогодні Vue.js є улюбленим інструментом для багатьох сучасних розробників. По-перше, можна використовувати TypeScript у розробці. По-друге, фреймворк легко освоюється та простий у використанні. Низький поріг входу дає можливість поринути у процес розробки та робити гнучкі інструменти для бізнесу. Регулярні оновлення дозволяють підвищити функціонал, крім того можна використовувати сторонні бібліотеки. Інструмент підходить для реалізації практично будь-якої ідеї. Продукти, виготовлені на цьому фреймворку, легко впроваджуються в бізнес, а сама технологія коштує недорого. Тепер, коли було досліджено загальне уявлення про дві технології, давайте проведемо порівняння Vue та Angular. Останні оновлення допомогли значно обійти Angular за багатьма показниками. Але об'єктивно цей фреймворк ще не досяг рівня Angular.

Переваги:

– є прогресивним продуктом, який дозволяє працювати з уже готовими додатками та поступово переводити їх на свою екосистему, не порушуючи працездатності;

– дуже мало важить, проекти швидко завантажуються, а пошукові роботи їх краще ранжують;

– велика бібліотека дозволяє швидко створювати бізнес-додатки;

– активна спільнота моментально відповідає на поставлені питання та допомагає розібратися із завданнями, а база знань формується з відповідей програмістів;

– використовувати Vue.js можна як в особистих, так і комерційних цілях, так як продукт має ліцензію MIT;

– можливість використання сторонніх бібліотек суттєво розширює інструменти для розробки, тому можна створювати проекти будь-якої складності та вирішувати будь-які завдання;

– простий в освоєнні, низький поріг входу, знаючи JavaScript, вивчити фреймворк можна за 2-4 дні, крім того, він працює з TypeScript;

– висока продуктивність за рахунок використання Virtual DOM.

Але фреймворк не здатний протистояти потужностям та високому рівню продуктивності, тому підходить для легковажних проектів. Тим не менш, більшість цифрових продуктів для малого та середнього бізнесу не потребують високої продуктивності. Тому їх можна сміливо створювати на Vue.js [1].

Зараз більшість азіатських компаній (наприклад, Xiaomi та Alibaba) перейшли на цей фреймворк. Опитування показали, що 90% тих, хто використав Vue, з радістю працюватимуть з ним у майбутньому; 70% тих, хто чув про Vue, зацікавлені у його вивченні [5].

Vue використовує синтаксис шаблонів на основі HTML, що дозволяє декларативно зв'язувати рендеринг DOM з основними екземплярами даних в Vue. Всі Vue шаблони валідні HTML, і можуть бути розпарсені браузерами та HTML парсерами. Всередині Vue компілює шаблони в рендерингові функції віртуального DOM. В поєднанні з реактивною системою, Vue здатний розумно обчислити кількість компонентів для ре-рендерингу та

застосувати мінімальну кількість маніпуляцій з DOM, коли стан застосунку зміниться.

Одна із найвиразніших особливостей Vue – це ненав'язлива реактивна система. Моделі це просто плоскі JavaScript об'єкти. Це робить керування станами дуже простим та інтуїтивним. Vue надає оптимізований ре-рендеринг з коробки без потреби робити що-небудь додатково. Кожен компонент слідкує за своїми реактивними залежностями під час рендерингу, тому система знає точно коли має відбуватись ре-рендеринг і які компоненти потрібно ре-рендерити.

1.5 Постановка задачі дослідження

В наш час дуже багато компаній використовують Front-end фреймворки для розробки своїх сайтів. Все це через те, що вони надають сайту необхідну безпеку та продуктивність, тому користувачу приємно знаходитись на ньому. Динаміку популярності фреймворків Angular та Vue.js можна побачити на рис. 1.1-1.2.

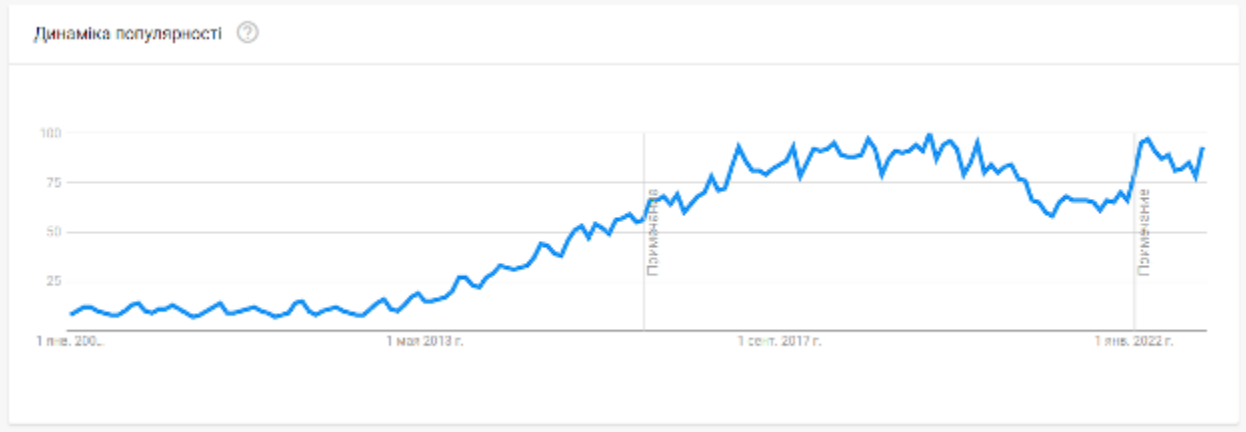


Рисунок 1.1 – Динаміка популярності Angular

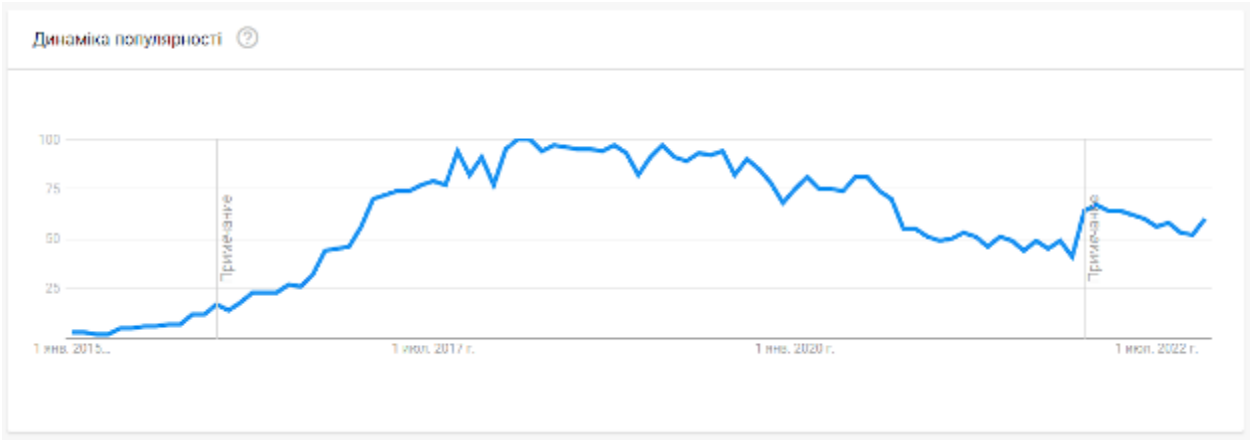


Рисунок 1.2 – Динаміка популярності Vue.js

Показником є те, що велика кількість саме відомих компаній також користується Front-end фреймворками. Наприклад, Angular використовується на сайтах Microsoft, Google та Mcdonalds (рис. 1.3-1.5), а Vue.js використовується на сайтах Nintendo, Grammarly та Laravel Spark (рис. 1.6-1.8).

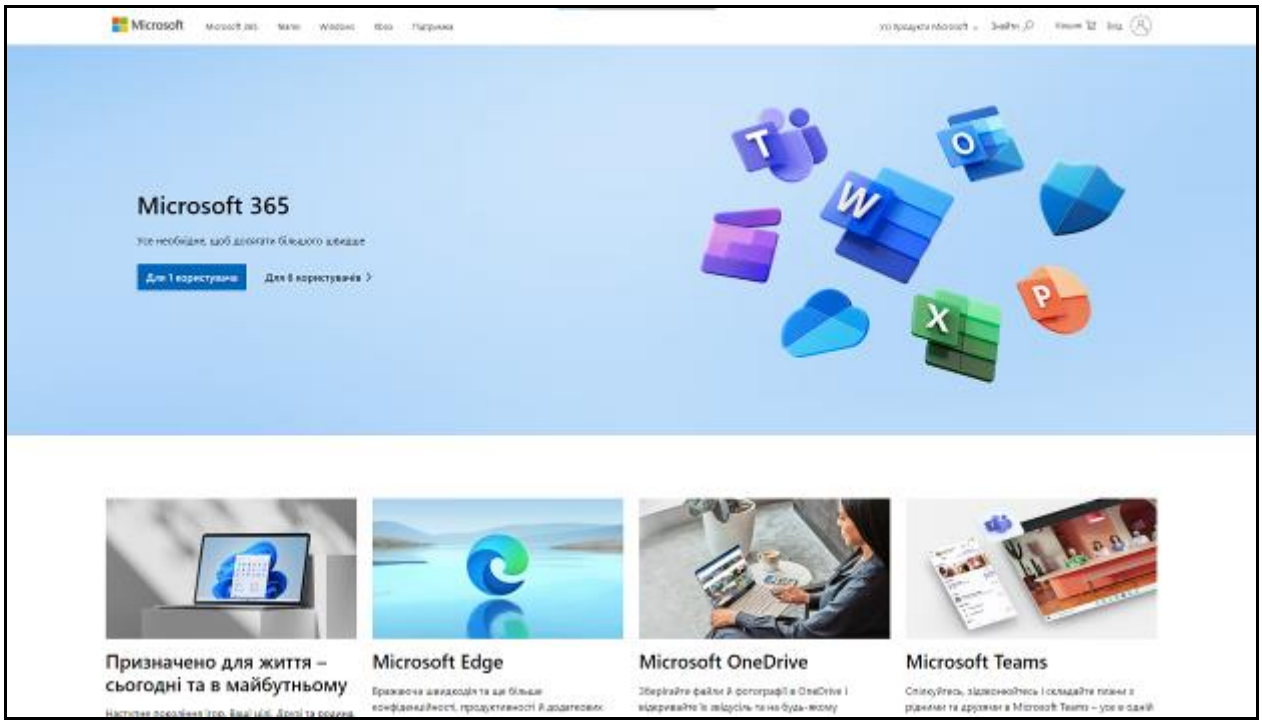


Рисунок 1.3 – Сайт Microsoft

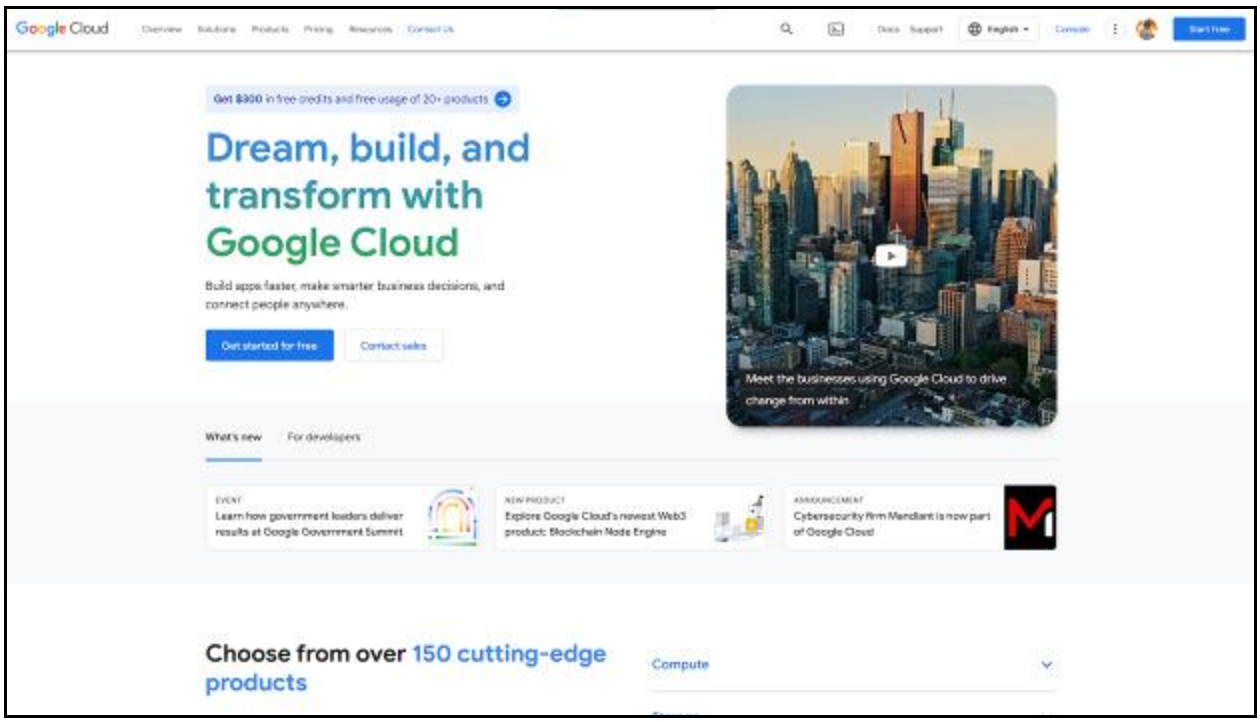


Рисунок 1.4 – Сайт Google

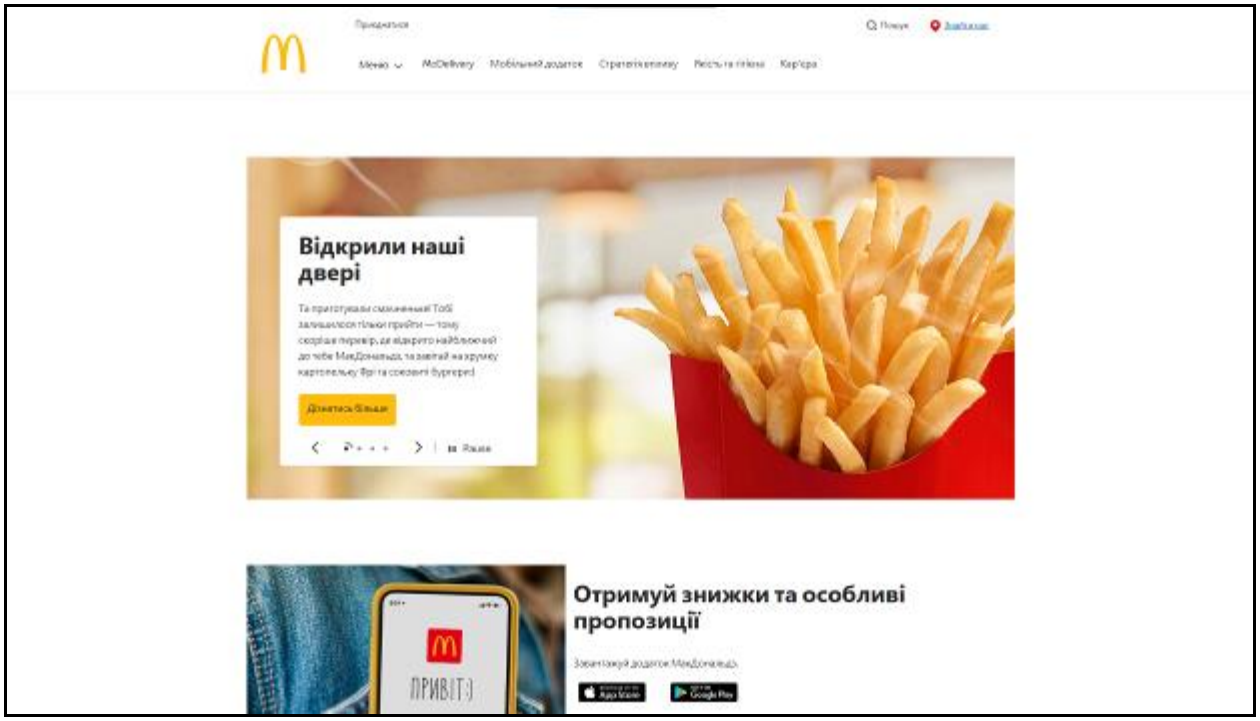


Рисунок 1.5 – Сайт Mcdonalds

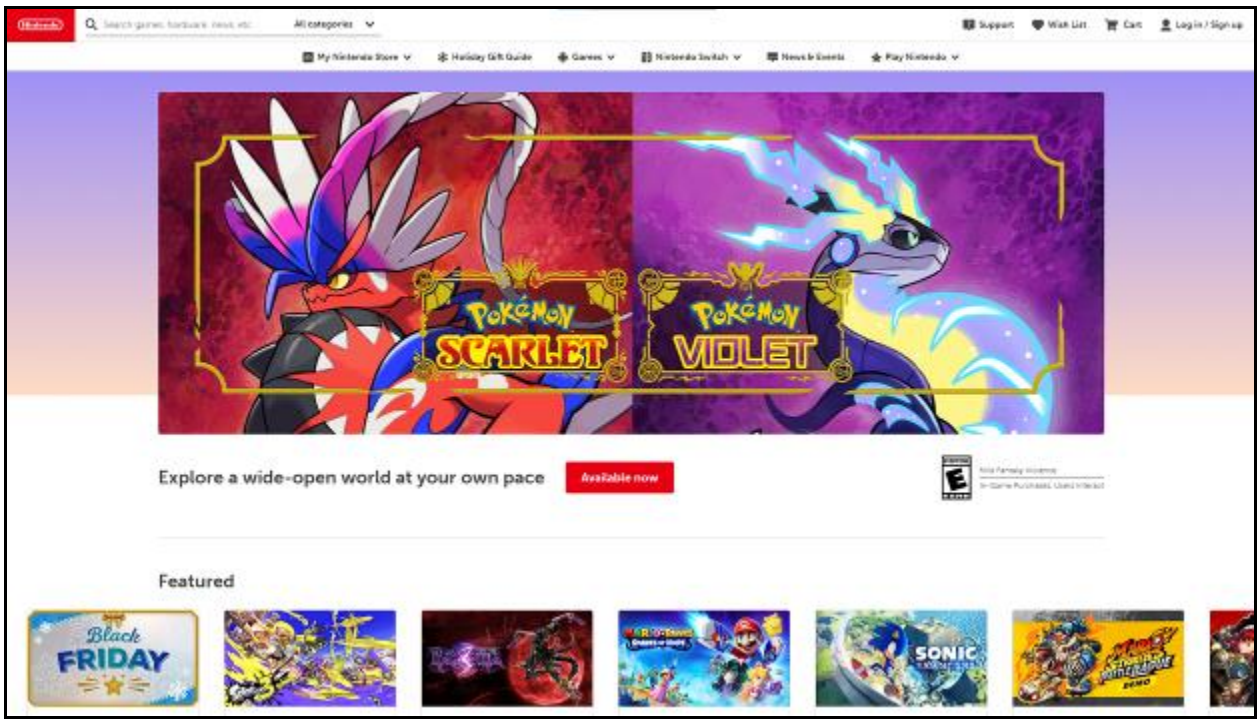


Рисунок 1.6 – Сайт Nintendo

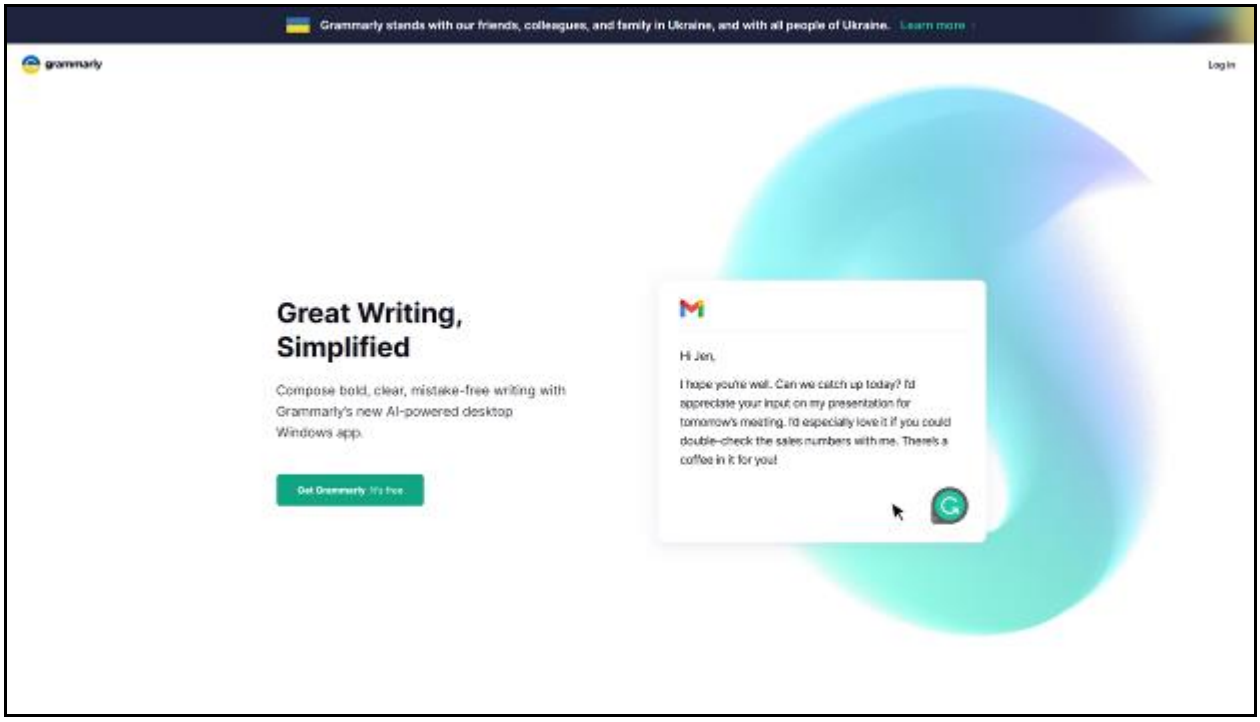


Рисунок 1.7 – Сайт Grammarly

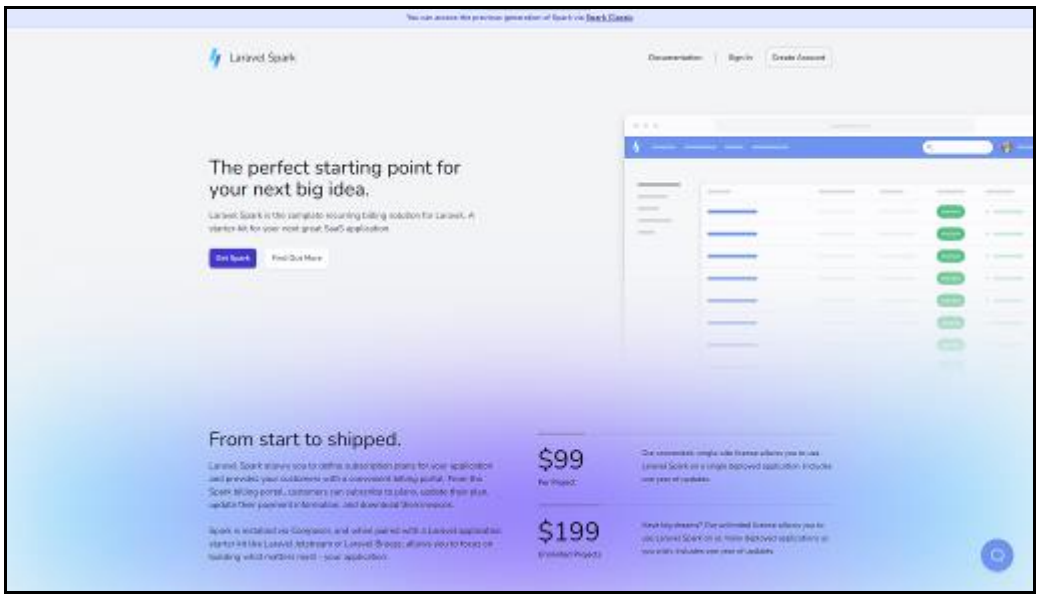


Рисунок 1.8 – Сайт Laravel Spark

Різниця між фреймворками неочевидна, але може призводити до різних тривалостей процесу проектування, отже, і до зміни його вартості, також до різних результатів щодо продуктивності створеного сайту. Тому виникає необхідність для кожного конкретного завдання розробки сайту вирішити, яким фреймворком буде краще користуватися, щоб скоротити тривалість процесу проектування, але забезпечити високу продуктивність сайту. Звідси випливає основне завдання дослідження.

Метою кваліфікаційної роботи є підвищення показника продуктивності сайтів завдяки вибору найбільш ефективного Front-end фреймворку.

Об’єктом дослідження є процес розробки сайту.

Предмет дослідження – фреймворки Angular та Vue.js для Front-end розробки, метод експертного оцінювання.

Для досягнення основної мети роботи потрібно розв’язати кілька задач:

– дослідити основні критерії, за якими співставляють властивості фреймворків, що використовуються для розробки сайтів;

– визначити, як використання тих чи інших фреймворків впливає на процес розробки сайту;

– розробити методику покращення процесу розробки сайтів;

– провести експериментальне дослідження розробленої методики.

2 ВАЖЛИВІСТЬ ВИКОРИСТАННЯ FRONT-END ФРЕЙМВОРКІВ

2.1 Недоліки типового процесу розробки

В наш час все більше виникають випадки, коли сайти піддаються зламу з боку стороннього софту, чи виникають проблеми з оптимізацією продуктивності через розташування на сайті великої кількості інформації, фото, відео, тощо. Якщо проект необхідно постійно доповнювати новою інформацією, постами, статтями, товарами чи іншим, через це виникають проблеми зі створенням великої кількості схожих сторінок, із чого випливає проблема з розміром збірки сайту та продуктивністю, тобто такі сайти не мають можливості ефективно масштабуватися. В таких випадках навіть код HTML, CSS та скрипти не стискаються, що також впливає на продуктивність.

2.2 Використання Front-end фреймворків

Саме через ці недоліки все більше сайтів переходять на використання Front-end фреймворків. Вони надають сайту необхідну безпеку, яку, в іншому випадку, необхідно було б шукати шляхи усунення слабких місць сайту, і не факт, що вони будуть працювати коректно. Також можна не врахувати деякі моменти, які у майбутньому можуть негативно вплинути на безпеку. У той час, Front-end фреймворки надають необхідніші, актуальні та ефективніші способи реалізації безпеки до проекту, які ще й постійно оновлюються та зазвичай підтримуються великими компаніями, чим забезпечують безпеку, навіть якщо з’явиться новий вид чи софт для зламу сайтів.

Front-end фреймворки дають можливість до правильного написання коду сайту враховуючи стандарти та принципи програмування, а також роблять сторінки більш продуктивними, через що, при переходах на різні сторінки сайту не буде відбуватися завантаження сторінки, такі сайти звуться

реактивними додатками. Тим самим фреймворки забезпечують можливість у майбутньому масштабувати проект надалі, доповнюючи новими сторінками чи новою інформацією, та не піклуючись про погану оптимізацію.

Розмір збірки також буде меншим, за інші проекти, оскільки, Front-end фреймворки, при будуванні проекту стискають весь присутній код, за допомогою існуючих алгоритмів, через що, у результаті виходить той самий сайт, але важити він буде набагато менше.

Все перераховане вище буде коректно працювати, якщо весь код буде правильно написаний та вистроєна правильна архітектура проекту. Через це, поріг входу до Front-end фреймворків набагато вищий. Для них необхідно знати не тільки основи HTML, CSS та JavaScript (у деяких випадках необхідно знати і TypeSctipt), але ще і розбиратися у синтаксисі обраного Front-end фреймворку, з чого випливає і вміння швидко знаходити необхідну інформацію у його документації.

Через це можна сказати, що головними показниками корисності Front-end фреймворків є саме безпека, оптимізація, масштабованість, розмір збірки, архітектура та складність написання коду з боку програміста.

2.3 Безпека

Під безпекою веб-сервісів, як правило, розуміють забезпечення схоронності даних і їх недоступність для сторонніх осіб, а також здатність додатків зберігати працездатність при кібератаці і не піддаватися зараженню вірусами.

Безпека сайтів залежить від якості їх програмного коду і від компетентності адміністратора веб-сервера. Тобто причиною погроз безпеки може бути як вразливість самого сайту перед кібератакою (наприклад, відсутність захисту від перебору паролів), так і помилки, допущені адміністратором веб-сервера (наприклад, несвоєчасне оновлення ПЗ). Не менш частою причиною зламів є незнання або недотримання

співробітниками банальних правил безпеки (прості паролі, введення даних на фішингових сайтах, зараження вірусами ПК адміністраторів) [8].

Обидва фреймворки мають високий рівень безпеки. Angular має більше довіри від користувачів, тому що він довше на ринку і має підтримку компанії Google. Vue спочатку створювався як авторський проект і довгий час підтримувався тільки однією людиною. Але зараз компанія розробників зросла і рівень безпеки відповідає всім вимогам.

2.4 Оптимізація

Вкрай важливо, щоб сайт був належним чином оптимізований для пошукових систем. В іншому випадку користувачі не зможуть знайти сайт. Крім того, щоб побороти конкуренцію в інтернеті, веб-сайт повинен бути зручним для користувача, простим і інтуїтивно зрозумілим.

Але досить часто в процесі створення веб-сайту власники бізнесу настільки зосереджені на тому, щоб правильно оформити дизайн і контент, що часто забувають про самий фундаментальний елемент, який забезпечує більший трафік веб-сайту – швидкість. Саме тут починається оптимізація продуктивності сайту.

В цілому, чим більше HTTP-запитів отримує ваш веб-сайт, тим повільніше він буде завантажуватися. Браузер обмежений відкриттям певної кількості одночасних підключень до одного хосту. Щоб уникнути цього, необхідно зменшити кількість окремих елементів сторінки за допомогою консолідації ресурсів, щоб менші файли, наприклад зображення, можна було об'єднувати в один файл. Це допомагає скоротити HTTP-запити, а також кількість циклів, необхідних для завантаження веб-сторінки.

Потрібно зробити так, щоб максимальний час очікування завантаження веб-сайту був менше 3 секунд. Якщо веб-сторінка працює повільніше, користувачі просто не хочуть чекати і клацають на інший рядок в пошуку. Ось чому дуже важливо вжити заходів для оптимізації зображень, знайдених

на сайті. У більшості випадків веб-сайт завантажується повільно, тому що завантаження зображень займає багато часу, що негативно впливає на призначений для користувача досвід.

Звичайно, висока якість зображень на сайті також має вирішальне значення, оскільки вона допомагає привернути увагу людей. Але достатньо стиснути або обрізати зображення, щоб вони не зайняли багато часу при завантаженні [9].

Швидкість завантаження сторінок сайту в браузері є одним з важливих факторів пошукового ранжування, а також має вплив на формування враження відвідувачів від роботи з сайтом. Якщо ця швидкість низька, то це може призвести до погіршення індексації сайту пошуковими системами і до зниження лояльності відвідувачів. Обидва фреймворки мають більш-менш однакову оптимізацію продуктивності проектів.

2.5 Масштабованість

Масштабованість – це спроможність системи, мережі або процесу справлятись зі зростаючим обсягом робіт (робочим навантаженням), або інакше кажучи, це – потенціал, достатній для того, щоб відреагувати на зростання інтенсивності запитів, розширення вимог і збільшення їх кількісних характеристик [10]. Обидва фреймворки здатні вносити зміни та масштабувати продукт. Причому у Angular за рахунок модульної архітектури можливостей набагато більше, а сам процес масштабування відбувається легше.

2.6 Розмір збірки

Щодо розміру збірки, то обидва фреймворки досить непогані. Але якщо вдатись до голих цифр, то схоже, що Vue 3 таки обганяє Angular. Остання версія Angular – з AOT-компіляцією і tree-shaking, змогла значно

зменшити розмір збірки. Однак повнофункціональний проект Vue з включеними Vuex та Vue Router (~30КБ gzip) все ж значно легший, ніж AOT скомпільований додаток, створений з Angular-cli (~65КБ gzip) [11].

2.7 Архітектура

Angular, Vue.js та інші подібні фреймворки ґрунтуються на компонентах. Тобто, додаток складається з безлічі компонентів, які можуть містити в собі і бізнес-логіку, уявлення та багато іншого. Таким чином, розробники в багатьох проектах пишуть всю логіку в компонентах. Тобто, якщо компонент описує якусь велику частину функціоналу з великою кількістю (можливо складної) логіки, то вся ця логіка залишається в компоненті. З'являються десятки методів та тисячі рядків коду. А якщо врахувати те, що, наприклад, у Vue.js є такі поняття як computed, watch, mounted, created, то логіку пишуть ще й у всі ці частини компонента. У результаті, щоб знайти якусь частину коду, необхідно буде витратити багато часу.

Приблизно в 2008 році, стосовно backend, була запропонована "шарова" архітектура. Основна ідея цієї архітектури полягає в тому, що весь код програми слід розбивати на певні шари, які виконують певну роботу та не дуже знають про інші шари. З подібним розбиттям програма стає набагато простішою у підтримуванні, писати тести, шукати відповідальні зони та взагалі читати код. Саме шарова архітектура використовується у Vue.js.

В той час, у Angular використовується модульна архітектура. У світі веб-розробки модульна архітектура Angular це щось особливе. Angular-модулі додають до системи додатковий рівень логічного угруповання, важливо, щоб їх структура якнайкраще відповідала вимогам сайту. Знання про те, як розділяти та поєднувати функціонал програми, користуючись якісно спроектованими модулями, це фундаментальна частина створення архітектури Angular-додатків.

Існують такі види Angular-модулів:

– Declarations/Widget Module. Модулі із оголошеннями різних сутностей. Як приклад подібних модулів можна навести набори компонентів інтерфейсу користувача, директив, пайпів;

– Services Module. Модулі Сервісів. Наприклад – HttpClientModule;

– Routing Module. Модулі маршрутизації;

– Domain Feature Module. Модулі, що реалізують ключові завдання програми;

– Core/Shared Module. Core-модуль це модуль для оголошення глобальних сервісів. Shared-модуль – це модуль, у якому оголошують компоненти спільного використання.

2.8 Складність написання коду з боку програміста

Складність написання коду з боку програміста напряму впливає на вартість розробки. На Angular-спеціалістів високий попит, їх послуги коштують дорожче, бо поріг входу досить високий. Поріг входу на Vue нижчий, для розробки потрібні менш досвідчені фахівці, тому вартість дешевше. За фактом, ціна продукту, зробленого на Vue, буде дешевшою. Тому написання коду з боку програміста буде більш складнішим у Angular проекті, ніж у Vue.

3 ВІДМІННОСТІ ПРОЦЕСУ РОЗРОБКИ

3.1 Етапи типового процесу розробки

Типовий процес розробки сайтів розподіляється на декілька етапів. Для початку проводиться проектування навігації сайту, щоб розуміти масштаб розроблюваного сайту. Наступним кроком створюються шаблони сторінок сайту, наприклад, шаблон для сторінок всіх користувачів чи шаблон для адміністративної панелі. Далі, використовуючи створені шаблони, створюються сторінки сайту, наповнюються контентом та реалізується базовий функціонал деяких елементів. Після чого реалізується сама навігація між сторінками, у типовому процесі розробки це зазвичай виконується завдяки додаванню атрибута href до тегу <a>. Далі додається функціонал взаємодії бази даних з сайтом, наприклад, вхід до системи, отримання контенту з бази даних, його додавання, видалення чи зміна. Та останнім кроком є тестування і усунення знайдених недоліків. Схему процесу показано на рис. 3.1.

Рисунок 3.1 – Схема типового процесу розробки сайтів

3.2 Спосіб підключення фреймворку Angular

Першим етапом розробки проекту за допомогою Front-end фреймворку Angular є саме його підключення. В офіційній документації є багато способів

це зробити, але буде розглянуто найпопулярніший та простіший – за допомогою npm, вбудованого у програмне забезпечення Node.js.

Node.js – програмна платформа, заснована на двигуні V8 (компілюючому JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованої мови на мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями вводу-виводу через свій API, написаний на C++, підключати інші зовнішні бібліотеки, написані різними мовами, забезпечуючи виклики до них із JavaScript-коду [12].

Раніше можна було запустити JavaScript тільки в браузері, але розробники розширили його, і тепер можна запускати JS на комп'ютері як окрему програму. Так виник Node.js. Тепер можна зробити набагато більше з JavaScript, ніж просто інтерактивні веб-сайти – тепер JavaScript має можливість робити те, що можуть робити інші скриптові мови програмування, такі як Python.

Npm – менеджер пакетів, що входить до складу Node.js. Установка існуючих пакетів здійснюється за допомогою команди у консолі: npm install <packagename> [12]. Усі доступні для встановлення пакети та їх короткий опис: npm search. Цією ж командою можна проводити вибірковий пошук пакетів.

Тобто для початку треба встановити актуальну версію Node.js, яку можна знайти на офіційному сайті (рис. 3.2).

Після цього необхідно відкрити консоль, шляхом натискання Win+R (відкриється утиліта «Виконати») та ввести туди назву «cmd», через що і відкриється консоль (рис. 3.3-3.4).

Тепер, розглянемо спосіб встановлення Angular до проекту. Перш за все, необхідно перейти до офіційної документації, та знайти розділ Setup, у якому вказана команда, яку необхідно вписати до консолі, щоб встановити цей фреймворк (рис. 3.5).

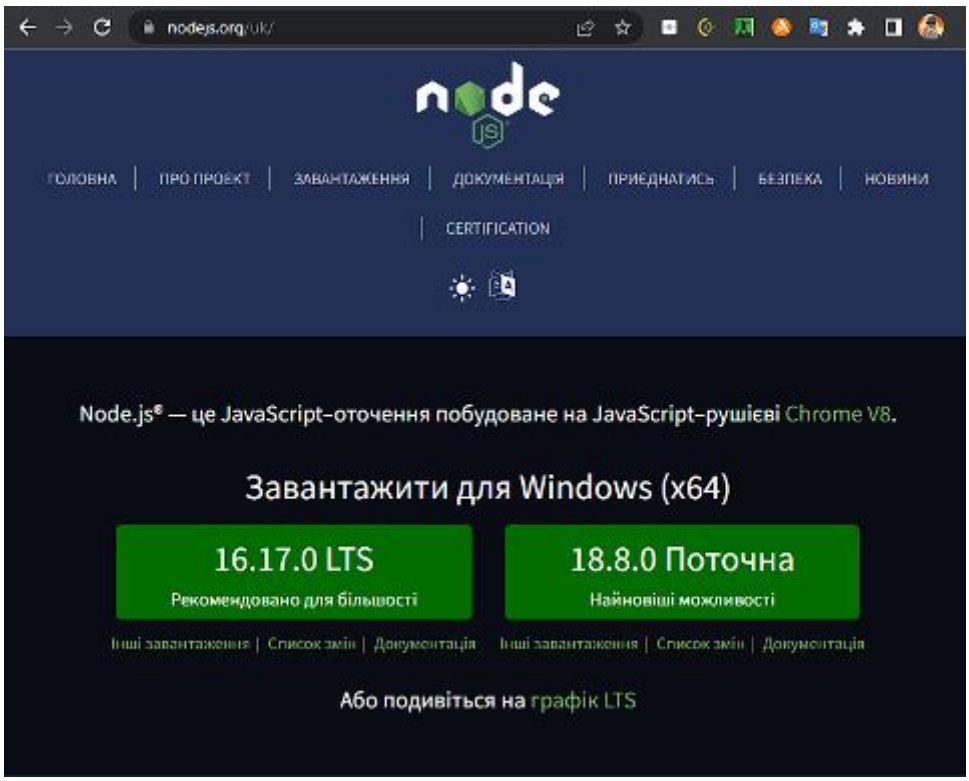


Рисунок 3.2 – Офіційна сторінка Node.js

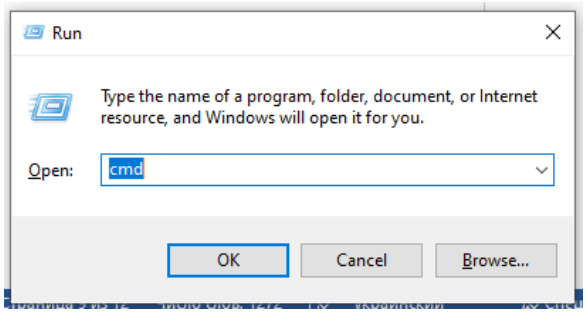


Рисунок 3.3 – Утиліта «Виконати»

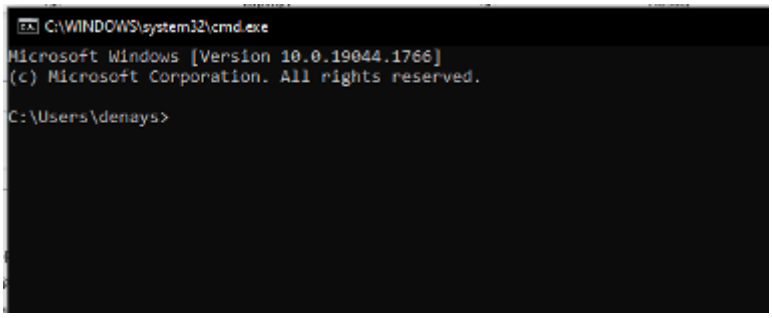


Рисунок 3.4 – Консоль

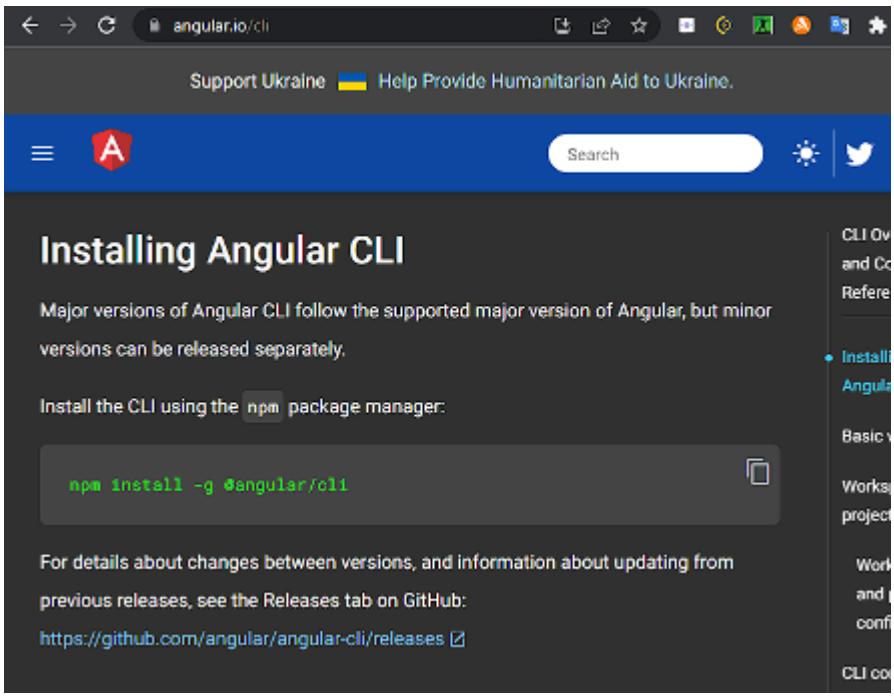


Рисунок 3.5 – Сторінка з інструкцією встановлення Angular

Як можна побачити, необхідна команда – `npm install -g @angular/cli` [13]. Але ще є невідомий флаг `-g`, він означає, що http-server має бути встановлений глобально та бути доступним для всіх додатків. Після написання цієї команди, та натискання Enter розпочнеться встановлення фреймворка (рис. 3.6).

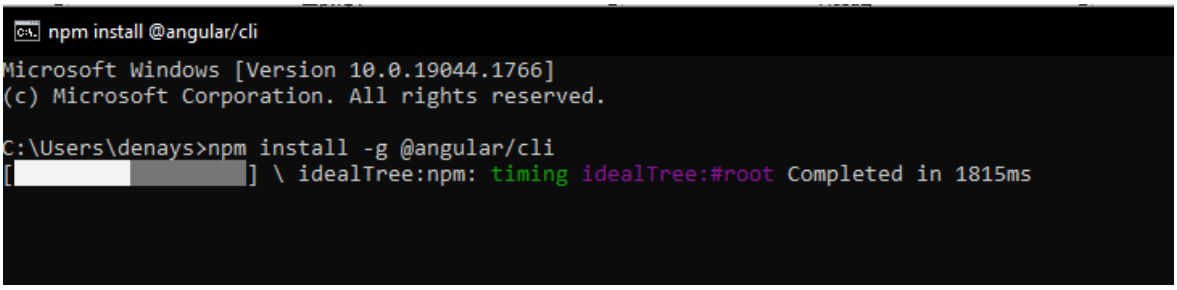


Рисунок 3.6 – Процес установки

Далі, необхідно у консолі обрати потрібну папку, до якої буде додано новий проект. Робиться це шляхом написання cd <шлях до папки>. Приклад можна побачити на рисунку 3.7. І натиснувши Enter, можна побачити що шлях зліва, змінився на вказаний.

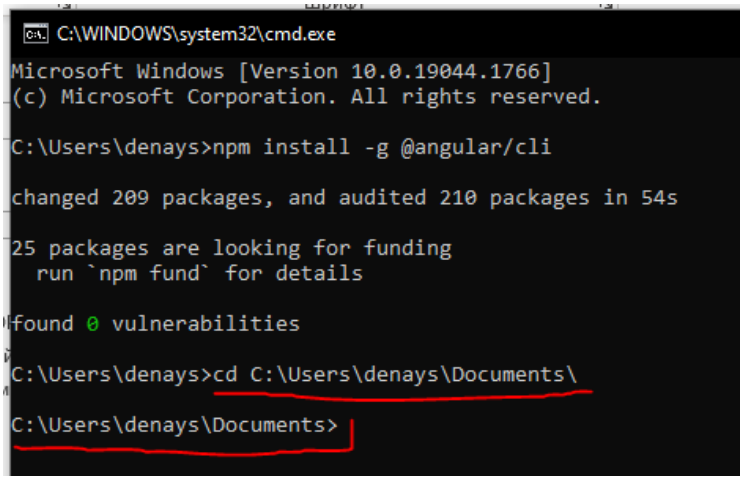


Рисунок 3.7 – Код обирання необхідної папки та результат

Тепер, для створення нового проекту у обраній папці, необхідно написати команду ng new <назва папки проекту> папка з цією назвою буде автоматично створена за заданим шляхом, та до неї буде додано всі необхідні файли для подальшого створення сайту. Під час встановлювання консоль буде задавати питання на використання тих чи інших додатків, наприклад маршрутизації, чи обирання того, яким способом буде проводитись стилізація сайту (рис. 3.8-3.9).

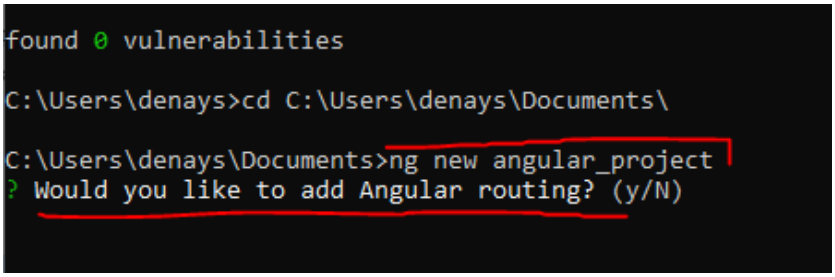


Рисунок 3.8 – Питання на використання маршрутизації

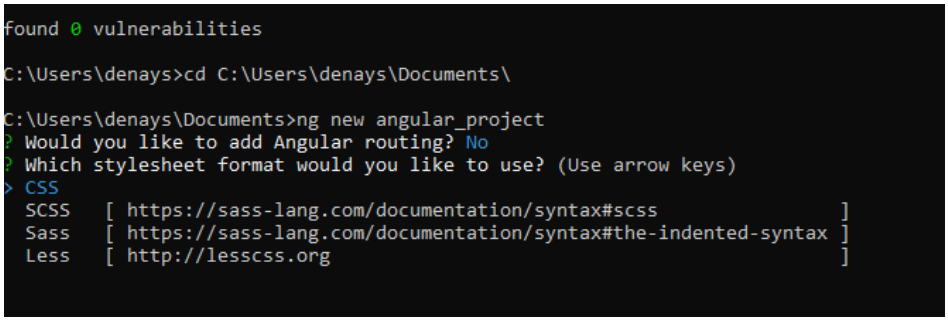


Рисунок 3.9 – Питання на використання якої стилізації

Далі буде проводитися встановлення необхідних пакетів, для коректної роботи цього фреймворка та створення відповідних, пов’язаних між собою, файлів у обраній директорії (рис. 3.10-3.11).

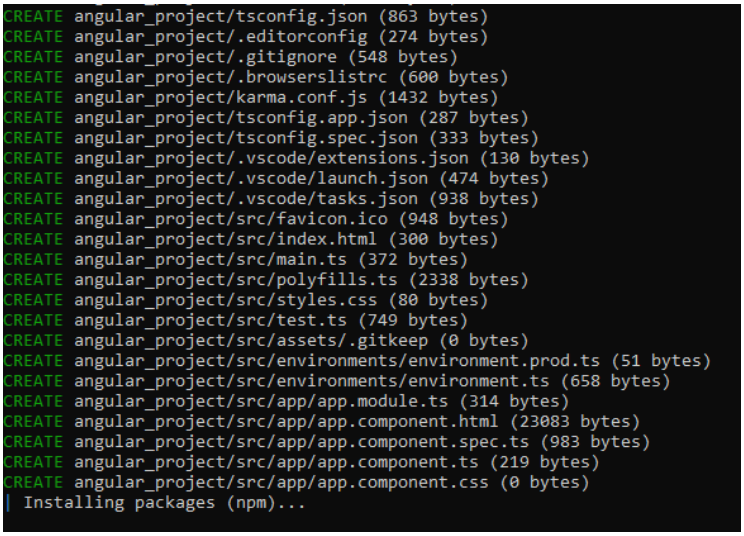


Рисунок 3.10 – Встановлення пакетів

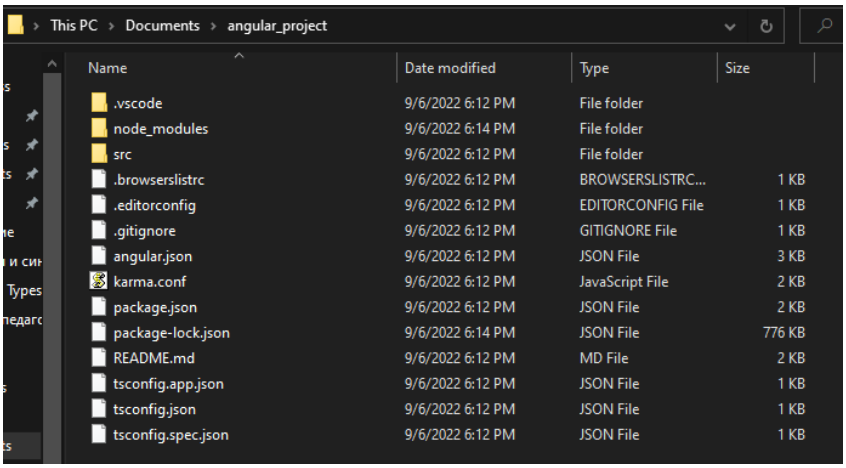


Рисунок 3.11 – Створена папка

Після чого потрібно перейти до створеної папки у консолі, щоб потім, завдяки ній, запустити локальний сервер. Перехід до папки виконується за допомогою попередньої команди cd <назва проекту>, а запуск локального сервера за допомогою npm start (рис. 3.12, 3.13). Після чого, за вказаною адресою буде розташовуватися локальний сервер [13] (рис. 3.14).

```
C:\Users\denays\Documents>cd angular_project

C:\Users\denays\Documents\angular_project>npm start
```

Рисунок 3.12 – Команди для запуска локального сервера

```
> angular-project@0.0.0 start
> ng serve

✓ Browser application bundle generation complete.

Initial Chunk Files   | Names         |  Raw Size
vendor.js             | vendor        |   1.78 MB |
polyfills.js          | polyfills     | 319.83 kB |
styles.css, styles.js | styles        | 211.86 kB |
main.js               | main          |  48.06 kB |
runtime.js            | runtime       |   6.53 kB |

                      | Initial Total |   2.35 MB

Build at: 2022-09-06T18:11:30.251Z - Hash: f78f96c3d9d9b0fa - Time: 33545ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **


✓ Compiled successfully.
✓ Browser application bundle generation complete.

5 unchanged chunks

Build at: 2022-09-06T18:11:30.868Z - Hash: f78f96c3d9d9b0fa - Time: 183ms

✓ Compiled successfully.
```

Рисунок 3.13 – Результат

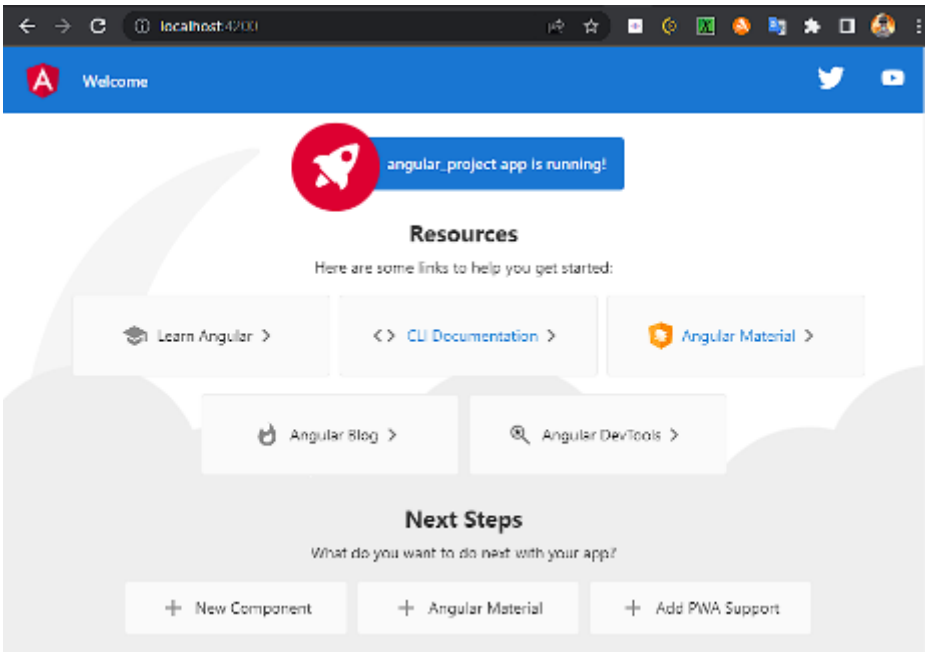


Рисунок 3.14 – Шаблон сторінки розташований на локальному сервері

3.3 Спосіб підключення фреймворку Vue.js

Для того, щоб підключити Vue.js до проекту, необхідно проробити всі шляхи, що були вказані у Angular підключенні, до моменту, коли відкриваємо консоль. У випадку з цим фреймворком, потрібно написати

команду `npm init vue@latest` та відповісти на всі питання що будуть з’являтися (рис. 3.15).

Після чого так само обрати шлях до необхідної папки та прописати команду `cd <your-project-name>`, `npm install` та `npm run dev` результати команд можна побачити на рис. 3.16.

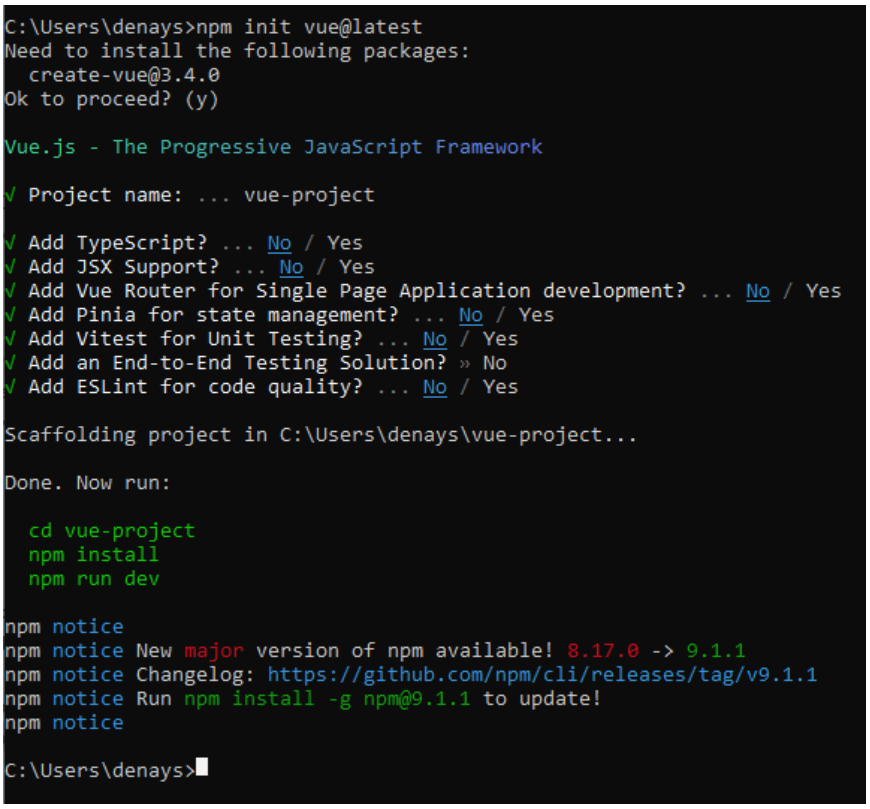


Рисунок 3.15 – Процес установки

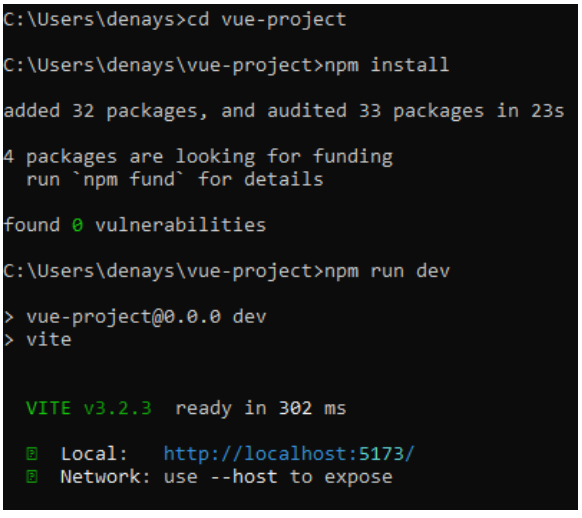


Рисунок 3.16 – Результати виконання команд

Через що, за заданим локальним посиланням можна відкрити шаблон сторінки Vue.js (рис. 3.17).

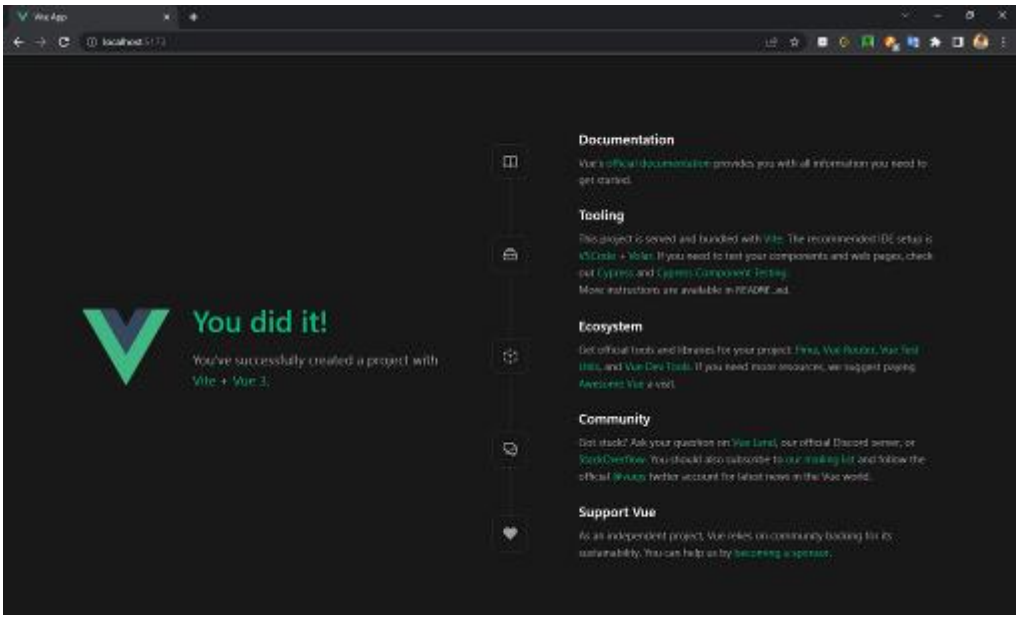


Рисунок 3.17 – Шаблон сторінки розташований на локальному сервері

Таким чином будо розглянуто способи встановлення Front-end фреймворків Angular та Vue.js до проекту. Після чого можна переходити до етапів, які були перераховані у підрозділі 3.1.

3.4 Відмінності процесу розробки Angular

У Angular, шаблони створені у другому етапі, зазвичай виносяться до одного файлу. Тобто, один шаблон – один файл, коли у типовому процесі розробки до кожної сторінки, де використовується один і той самий шаблон, додається однаковий код. Наприклад, теги <!doctype html>, <html lang="en">, <head>, <body>, <footer>, панель навігації чи різні елементи сайту, які використовуються на багатьох сторінках. При типовому процесі розробки вони розташовуються у кожному файлі сторінки чи у кожному файлі сторінки, де вони використовуються. При розробці з використанням Front-end фреймворків, такі елементи виносяться в окремі файли.

У Angular проекті, спочатку автоматично створюється файл index.html, при початковій ініціалізації проекту, до нього входять такі теги як <!doctype html>, <html lang="en">, <head>, <body> за бажанням можна додавати інші, але не рекомендується, бо це самий базовий шаблон (рис. 3.18).

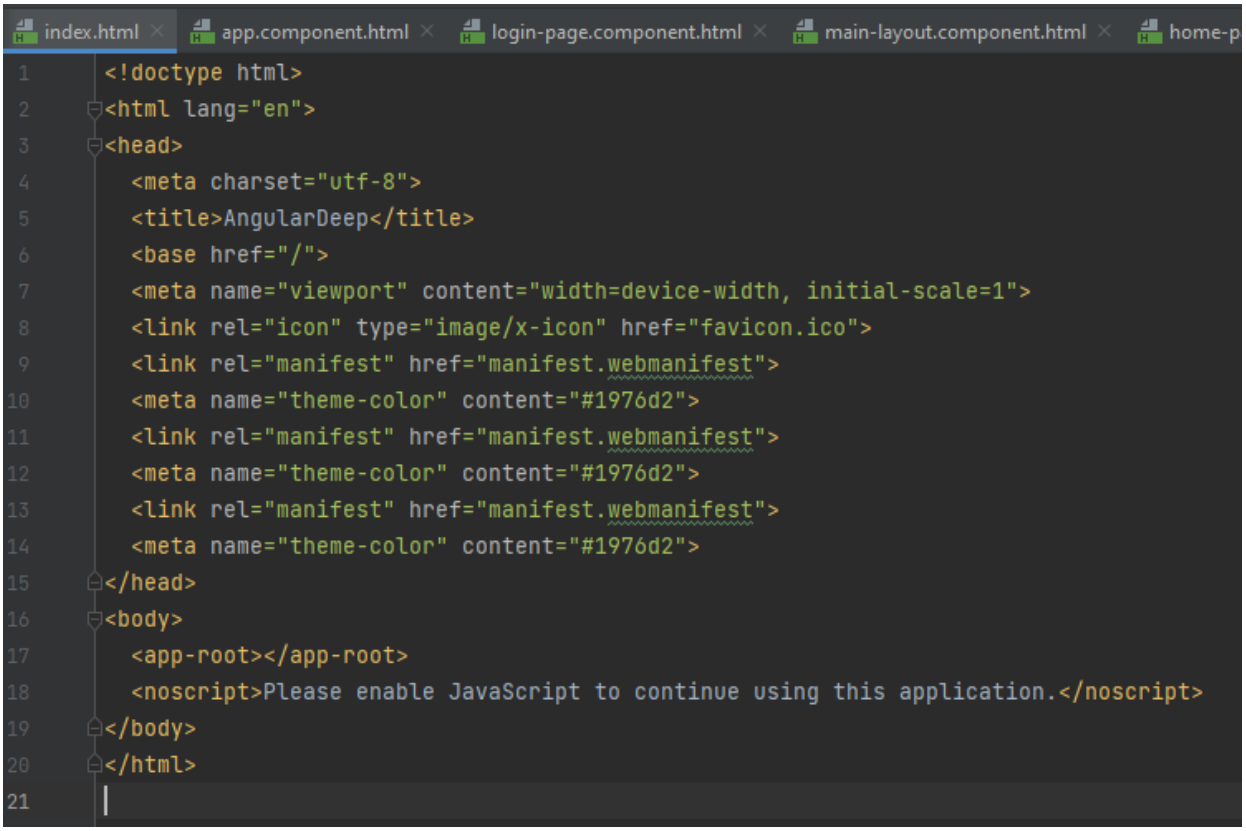


Рисунок 3.18 – Створений файл index.html

Можна ще побачити такі теги, як <app-root> та <noscript>. Перший – це Angular тег, який переадресовує на файл app.component.ts, який у свою чергу відповідає за навігацію сторінок (рис. 3.19). У ньому можна побачити selector – це зміна, яка відповідає тегу, який використовується. Наступні два – це пов’язані html та scss (це той самий css, але більш зручніший) файли, це базова структура компоненту у Angular. До html файлу входить один Angular тег (рис. 3.20), який саме і відповідає за навігацію, точніше він використовує інформацію з файлу, у якому написана логіка навігації (рис. 3.21). Це є так званий модуль навігації чи маршрутизації (Routing Module).

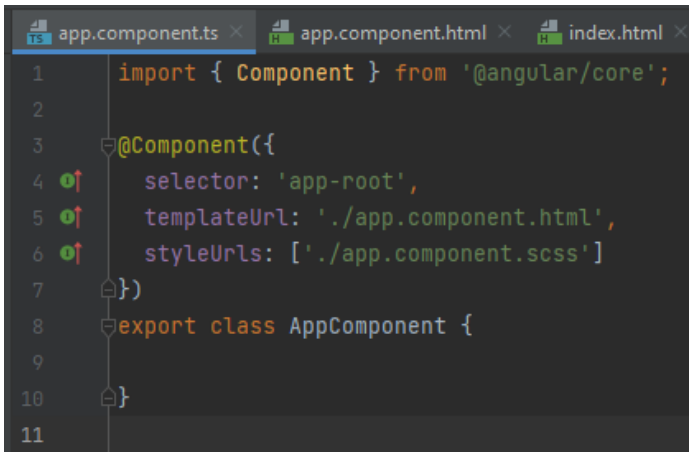


Рисунок 3.19 – Файл app.component.ts

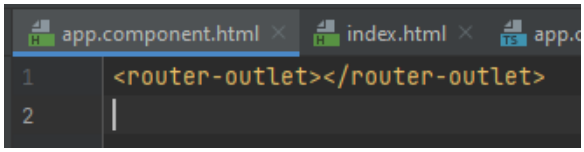


Рисунок 3.20 – Файл app.component.html

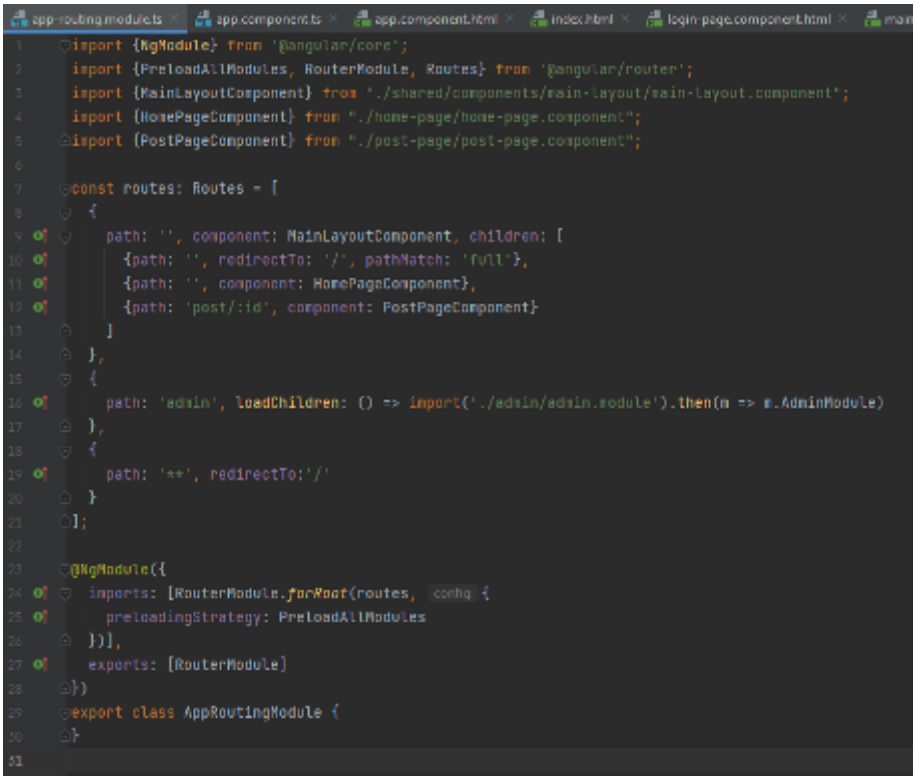


Рисунок 3.21 – Файл навігації app-routing.module.ts

Другий тег на рис. 3.18 <noscript> – це html тег, який визначає альтернативний контент, який відображатиметься у тому випадку, якщо

користувач відключив у браузері підтримку скриптів, або браузер не підтримує скрипти.

Далі, через модуль навігації відображаються інші шаблони, наприклад, на рис. 3.21 можна побачити що якщо path (тобто що буде написано у посиланні після початкового його значення) буде пустим, то необхідно використовувати компонент MainLayoutComponent (рис. 3.22), який в свою чергу також використовує тег <router-outlet>, який вже відповідає навігаційній інформації зі значення children, який передається третім параметром до об’єкту і є масивом. До масиву треба внести ті сторінки, які відповідатимуть головному шаблону, але відображатимуть іншу інформацію. Наприклад, компонент головної сторінки (рис. 3.23) чи компонент якогось поста в залежності від переданого id (рис. 3.24).

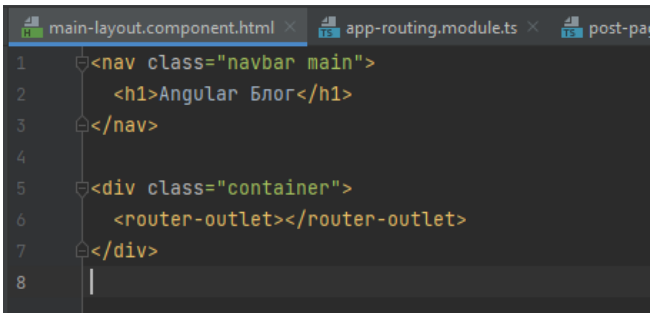


Рисунок 3.22 – Файл main-layout.component.html

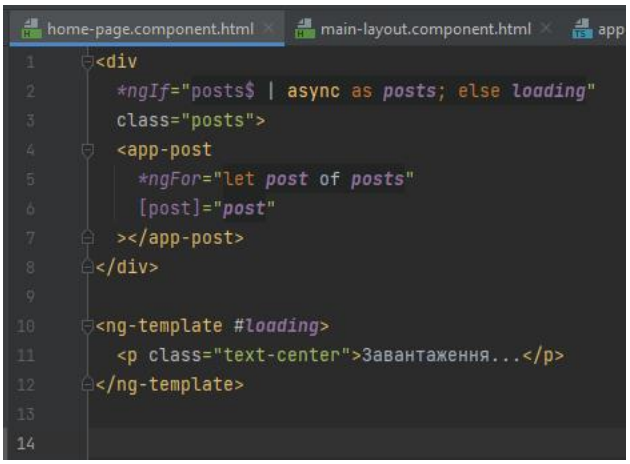


Рисунок 3.23 – Файл home-page.component.html

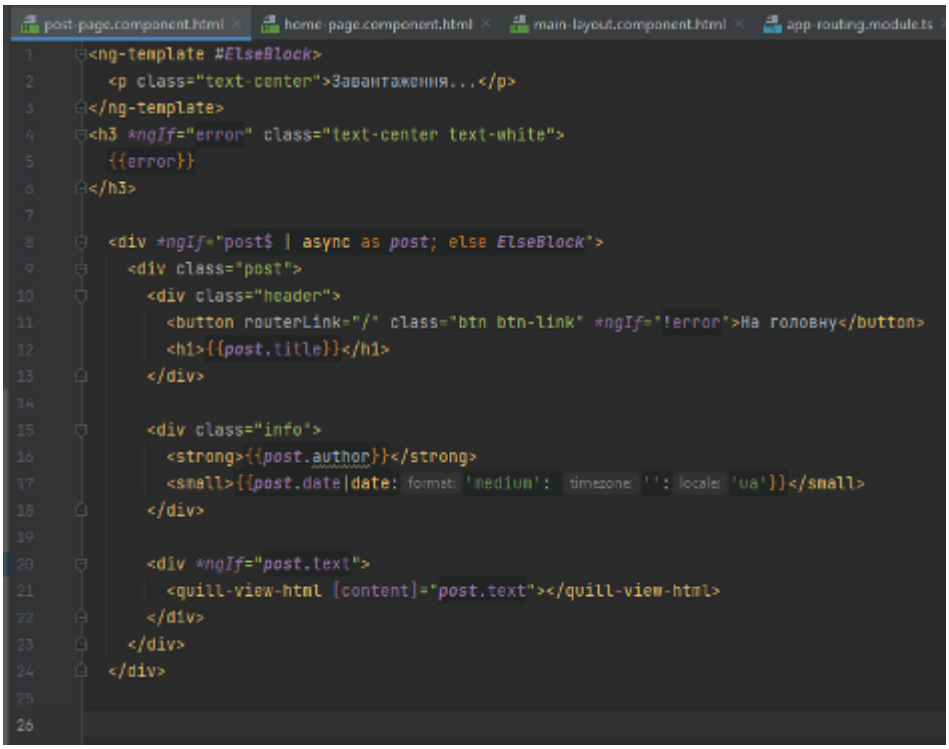


Рисунок 3.24 – Файл post-page.component.html

Кожен файл на рис. 3.22–3.24 – це компоненти, які так само, як і головний app.component пов’язані із відповідними .scss файлами у .ts файлі. У файлі .ts будь-якого компоненту можна писати відповідний функціонал на мові програмування TypeScript.

У випадку з path: 'admin' іде переадресація до відповідного модулю адміністративної панелі, де так само розписано навігацію, але відповідну до контенту цієї панелі. Останній path: '**' робить переадресацію до '/' (redirectTo:'/'), якщо відповідної сторінки не існує.

Такий файл як app-routing.module.ts є допоміжним, якщо багато коду займає написання логіки навігації, є доцільним винесення її до такого типу файлу. Тому в його кінці написано відповідні імпорти, які правильно беруть інформацію з написаних роутерів та потім експортуються. Теж саме можна і написати у файлі app.module.ts у експорті та імпорті, і все буде так само працювати. Приклад використання такої навігації можна побачити на прикладі admin.module.ts на рис. 3.25.

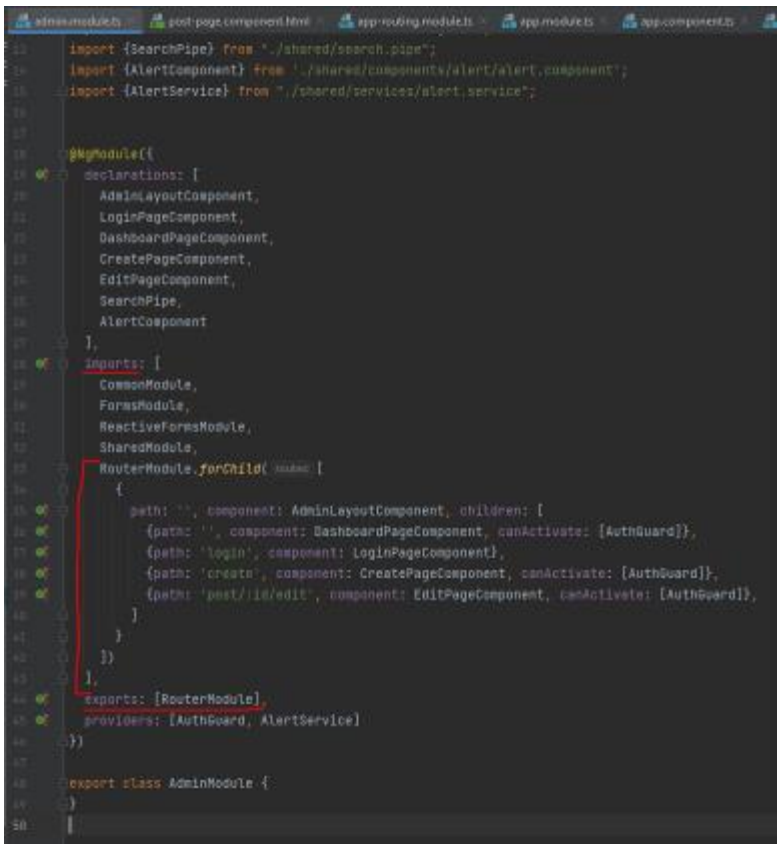


Рисунок 3.25 – Інший спосіб написання навігації

Значення canActivate: [AuthGuard] означає, що на цю сторінку можна потрапити, лише якщо людина увійде до системи, якщо ні, то пройде переадресація на сторінку входу до системи. Метод, який це робить, знаходиться у сервісі AuthGuard.

Реалізація переходів між сторінками відбувається за допомогою Angular атрибуту routerLink, який за необхідності можна забіндити. Бінд атрибутів у Angular (атрибут написаний у [art_name]) – це то саме, що і звичайний атрибут, але є можливість написання незвичних даних до них та навіть змінних, ці дані потім сам Angular оброблює і правильно використовує (рис. 3.26).

Таким чином виконується четвертий етап процесу розробки сайтів – реалізація навігації між сторінками, але з використанням Front-end фреймворків.

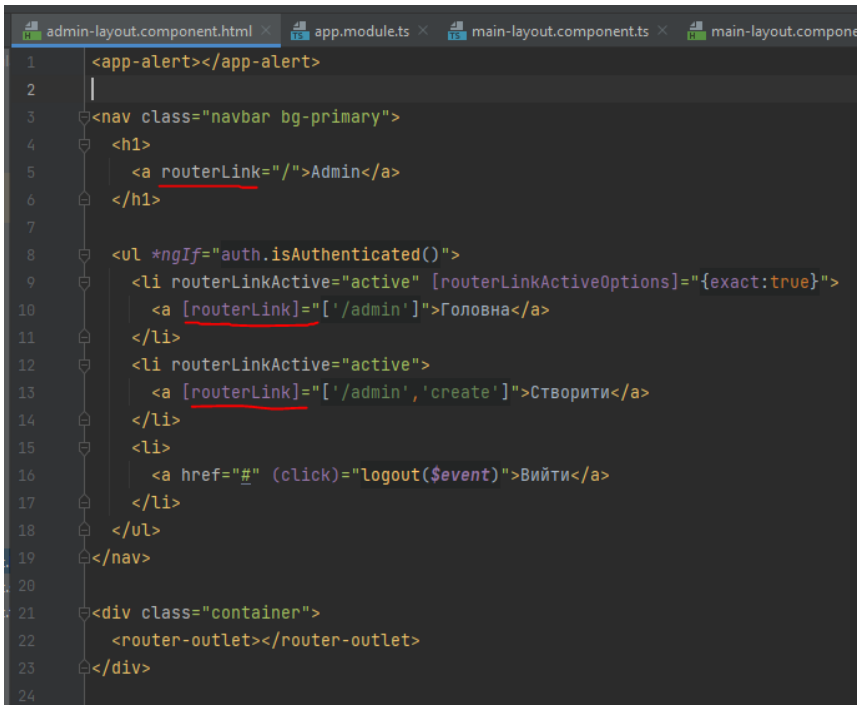


Рисунок 3.26 – Реалізація routerLink

Це виглядає набагато складніше, ніж просто використання тегу <a> та атрибута href, але тут немає нічого складного. В результаті такої реалізації навігації додаток стає реактивним, через що не відбувається завантаження у випадку переходу на іншу сторінку сайту.

У файлі, на рис. 3.25, також додаються всі існуючі компоненти та пайпи, які були створені та використовуються, у значенні declarations, який може бути чи звичайною змінною з одним значенням чи об’єктом з багатьма значеннями.

Також у Angular є свої модулі, які допомагають робити функціональну частину проекту, враховуючи її оптимізацію. Наприклад, FormsModule експортує необхідні провайдери та директиви для форм, керованих шаблонами, роблячи їх доступними для імпорту NgModules, які імпортують цей модуль. FormsModule використовує двостороннє зв’язування даних для оновлення моделі даних у компоненті під час внесення змін у шаблон і навпаки. Чи, наприклад, HttpClientModule, який налаштовує інжектор залежностей із HttpClient допоміжними службами для XSRF. Він служить для взаємодії з сервером та надсилання запитів за протоколом http. Цей клас

визначає ряд методів для відправки різноманітних запитів: GET, POST, PUT, DELETE. Цей клас побудований поверх стандартного об'єкта в JavaScript – XMLHttpRequest [14].

Angular надає змогу до використання сервісів, які доцільно створювати в окремій папці. Сервіси можуть бути будь-якими і зазвичай вони допомагають виносити функціонал для перевірки інформації отриманої чи написаної користувачем. Наприклад, можна створити сервіс для того, щоб прочитати відповідь з бази даних, та обробити її можливі варіанти. Для входу до системи можна обробити отримані запити, якщо такого користувача не існує, якщо пароль введено неправильно чи якщо все добре – виконати відповідні дії. Це може бути чи з’явлення повідомлення з відповідною інформацією про помилку, чи переадресація на наступну сторінку, якщо всі дані коректно введені.

Сервіси в Angular представляють досить широкий спектр класів, які виконують деякі специфічні завдання, наприклад логування, роботу з даними та інше.

На відміну від компонентів і директив, сервіси не працюють з уявленнями, тобто з розміткою html, не надають на неї прямого впливу. Вони виконують строго певне і досить вузьке завдання.

Стандартні завдання сервісів:

– надання даних додатку. Сервіс може сам зберігати дані в пам'яті або для отримання даних може звертатися до якогось джерела даних, наприклад, до сервера;

– сервіс може представляти канал взаємодії між окремими компонентами програми;

– сервіс може інкапсулювати бізнес-логіку, різноманітні обчислювальні завдання, завдання з логування, які краще виносити з компонентів. Таким чином, код компонентів буде зосереджений безпосередньо на роботі з уявленнями. Крім того, тим самим ми також можемо вирішити проблему

повторення коду, якщо нам потрібно виконати те саме завдання в різних компонентах і класах [15].

Angular директиви застосовуються для зміни зовнішнього вигляду чи поведінки DOM-элемента. Виділяють три типи директив:

– з власним шаблоном чи компонентом (компоненти є директивами);

– структурні, що змінюють структуру DOM-дерева;

– атрибути, що змінюють зовнішній вигляд або стандартну поведінку елемента DOM-дерева.

За замовчуванням Angular має ряд вбудованих директив. До вбудованих структурних директив відносяться: *ngIf, *ngFor, *ngSwitch та ін. Всі Angular директиви цієї групи починаються з символу * [16].

Angular пайп, або фільтр, потрібен для перетворення даних прямо в HTML-шаблоні. Наприклад, відображення дати та часу у бажаному форматі або завдання формату виведення числового значення. Angular має ряд вбудованих фільтрів, але також передбачена можливість створення власних [17]. Зазвичай пайпи пишуться після змінної через перпендикулярну пряму.

Наприклад, на рис. 3.27, 3.28, можна побачити типове використання директив та пайпів.

home-page.component.html | post-page.component.html

```
<div
  *ngIf="posts$ | async as posts; else loading"
  class="posts">
  <app-post
    *ngFor="let post of posts"
    [post]="post"
  ></app-post>
</div>

<ng-template #loading>
  <p class="text-center">Завантаження...</p>
</ng-template>
```

Рисунок 3.27 – Використання директив та пайпу

```
<div class="info">
  <strong>{{post.author}}</strong>
  <small>{{post.date|date: format: 'medium': timezone: '': locale: 'ua'}}</small>
</div>
```

Рисунок 3.28 – Використання пайпу

На рис. 3.27 використовується директива *ngIf, яка говорить про те, що цей тег буде відображено у випадку, якщо змінна posts$ пройдена через пайп async, який перетворює потік даних з отриманих з бази даних до об’єкту як шаблон об’єкта posts. У іншому випадку відображає завантаження. Після чого передає до компоненту post дані кожного посту об’єкта posts використовуючи пайп *ngFor. Існуючі директиви і пайпи у Angular-і також можна знайти у офіційній документації з відповідними прикладами.

Пов’язання з базою даних у Angular зазвичай відбувається за допомогою ServiceWorkerModule, у якого визивається register, до якого передається значення за замовчуванням script та opts з параметрами enabled та registrationStrategy. До enabled передається boolean, зазвичай він – false. А до другого – час який визначає, коли він буде зареєстрований у браузері (рис. 3.29).

```
@NgModule({
  declarations: [
    AppComponent,
    MainLayoutComponent,
    HomePageComponent,
    PostPageComponent,
    PostComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    SharedModule,
    ServiceWorkerModule.register( script: 'ngsw-worker.js', opts: {
      enabled: environment.production,
      registrationStrategy: 'registerWhenStable:30000'
    })
  ],
  providers: [INTERCEPTOR_PROVIDER],
  bootstrap: [AppComponent]
})
export class AppModule {
}
```

Рисунок 3.29 – Приклад використання ServiceWorkerModule

У прикладі змінна enabled приймає значення окремо винесеного об’єкта, у якого вже знаходиться і булеве значення, і ключ бази даних та її посилання (рис. 3.30). Після чого, наприклад, в окремому сервісі можна звертатися до необхідних значень та брати відповідні елементи з бази даних (рис. 3.31).

Рисунок 3.30 – Допоміжний об’єкт

Рисунок 3.31 – Приклад використання об’єкта у сервісі

Таким чином виконується третій та п’ятий етапи процесу розробки сайту використовуючи Front-end фреймворк Angular, а саме – використання шаблонів для створення подібних сторінок, наповнення контентом, функціоналом та додавання функціоналу взаємодії з базою даних.

3.5 Відмінності процесу розробки Vue.js

У випадку з Vue.js відбувається майже те саме, що і у випадку з Angular-ом, але є багато відмінностей. У нього створюється майже такий самий базовий шаблон сторінки index.html (рис. 3.32), але, як можна побачити – у <body> використовується звичайний тег який має id= "app", а у <script> йде підключення до головного JavaScript файлу.

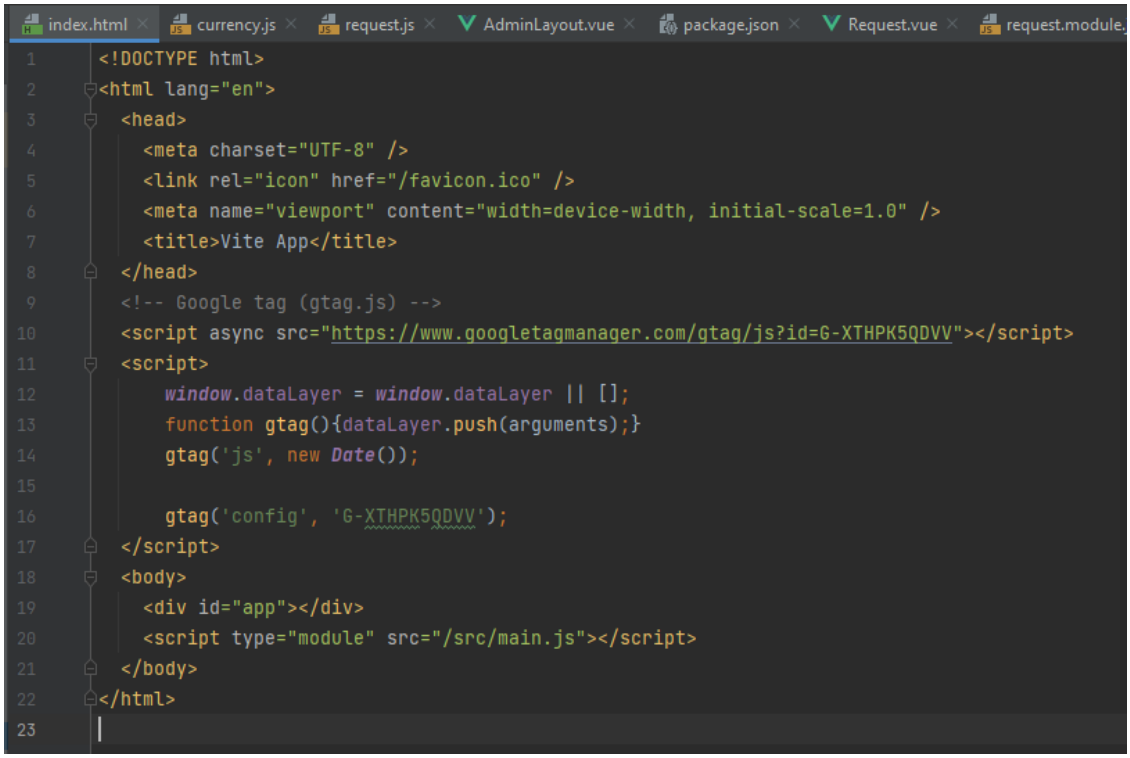


Рисунок 3.32 - Створений файл index.html

Vue.js розуміє, що додаток потрібно розташовувати саме у цьому . У файлі main.js створюється сам додаток та пов’язується з відповідним йому файлом. Також у ньому можна додавати файли, які відповідають роутерам, сховищу даних чи можуть бути підключені стороні елементи доданих у додаток бібліотек, тим самим пов’язуючи з додатком. Через це стає можливим їх використання (рис. 3.33).

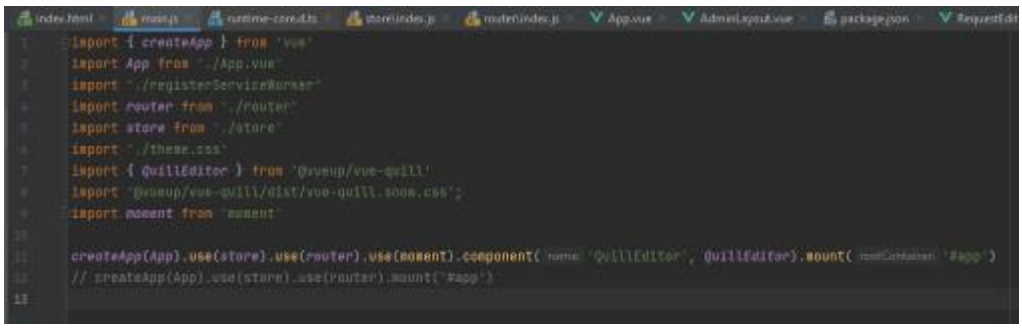


Рисунок 3.33 – Приклад файлу main.js

До createApp передається компонент, який необхідно створити та до якого необхідно додати теги, в яких буде відображатися необхідний компонент сторінки. Для кожної програми потрібен «кореневий компонент», який може містити інші компоненти як дочірні. Нижче, у тезі <script> необхідно вказати існуючі компоненти, та у хуці setup() використати функцію useRouter, яка імпортується з бібліотеки Vue.js. та повернути об’єкт зі значенням layout, у якому визвати computed, який приймає у значення функцію () => route.meta.layout (рис. 3.34). Це робиться для того, щоб додати до маршрутів довільну інформацію, як від назви переходів, хто може отримати доступ до маршруту, тощо.

Хоча файл є .vue, але насправді це той самий html. Файл *.vue – це власний формат Vue, який використовує HTML-подібний синтаксис для опису компонента. Кожен *.vue файл складається з трьох головних секцій: <template>, <script> і <style>, а також опціональних блоків користувача [18]. Зазвичай у тезі <script> пишуть функціональну частину відповідного компонента.

Вище скрипту можна побачити тег <template> у якому є тег, який змінюється в залежності від відкритої сторінки. Тобто, якщо буде відкрита будь-яка головна сторінка, то буде використовуватися тег <home-layout />, а використання директиви v-if каже про те, що це повинно спрацьовувати тільки тоді, якщо існує layout.

Рисунок 3.34 – Приклад App.vue файлу

У свою чергу кожен компонент також може у собі мати інші компоненти, які відносяться до обраної сторінки сайту. Наприклад, до AdminLayout можна прив’язати такі компоненти, як TheNavbar, який відображатиме навігацію по сторінкам чи AppMessage, який буде виводити повідомлення при виконанні якоїсь дії чи помилки (рис. 3.35).

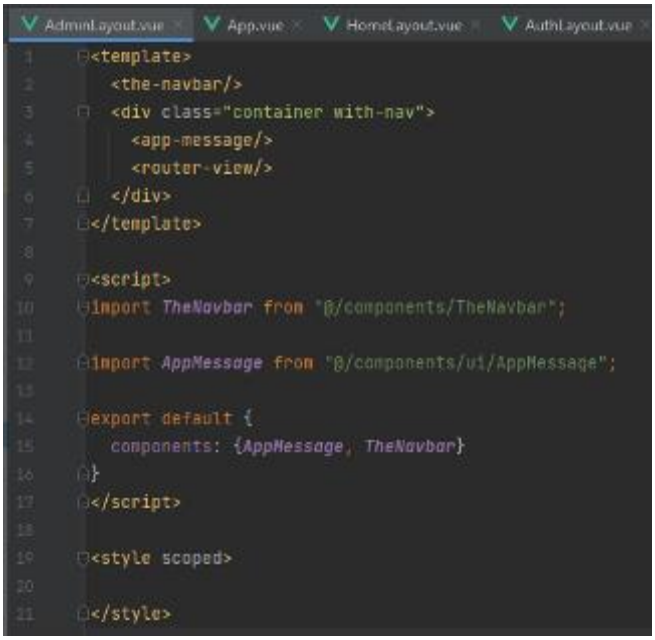


Рисунок 3.35 – Компонент AdminLayout

У цьому варіанті, у тезі <template>, в будь-якому випадку буде відображатися панель навігації – тег <the-navbar/> та повідомлення – тег <app-message/>, але у самому файлі AppMessage у повідомлення значення відображення – false. Таким чином можна задати майбутнє місце розташування якогось компоненту.

Тег <router-view/> відображає відповідний компонент в залежності від посилання сторінки, таким самим чином, як і <router-outlet> у Angular. Залежності роутерів зазвичай написані у файлі у окремій папці (рис. 3.36).

Рисунок 3.36 – Файл навігації

Як можна побачити, у цьому варіанті routers – це масив об’єктів, де кожен об’єкт – це один роутер. У ньому вказується path – частина посилання після головної адреси, name – ім’я роутера (може бути будь-яким), component – це пов'язаний компонент відносно посилання, який необхідно відображати. Далі йде meta – це необов’язкове значення, але інколи є дуже зручним, бо в нього можна записувати інформацію, яку необхідно передати в залежності

від відкритої сторінки. Наприклад, layout використовується у файлі App.vue на рис. 3.34, чи auth використовується у файлі навігації та переадресовує не зареєстрованого користувача на сторінку входу до системи, якщо він намагався зайти до адміністративної панелі (рис. 3.37).

```
const router = createRouter( options: {
    history: createWebHistory(process.env.BASE_URL),
    routes,
    linkActiveClass: 'active',
    linkExactActiveClass: 'active'
})

router.beforeEach( guard: (to : RouteLocationNormalized , from : RouteLocationNormalized , next : NavigationGuardNext ) => {
    const requireAuth = to.meta.auth

    if (requireAuth && store.getters['auth/isAuthenticated']) {
        next()
    } else if (requireAuth && !store.getters['auth/isAuthenticated']) {
        next('admin/auth?message=auth')
    } else {
        next()
    }
})

export default router
```

Рисунок 3.37 – Перевірка входу до системи

У Vue.js немає модулів, на відміну від Angular, тому деякі деталі необхідно робити вручну, наприклад при роботі з формами чи взаємодію з сервером.

Також у Vue.js немає пайпів, тому працювати з інформацією стає складніше, але це дає вільний простір для створення власного функціоналу. Наприклад, на рис. 3.38 показано, що для дати використовується окрема функція, яка працює з іншою бібліотекою та форматує дату.

```
momentDate: date => {
  return moment(new Date(date)).locale("uk").format('D MMM. YYYY p., HH:mm:ss')
}
```

Рисунок 3.38 – Функція форматування дати

Але в нього є вбудовані директиви, так само, як і в Angular. До вбудованих структурних директив відносяться: v-text, v-if, v-else та ін. Всі Vue.js директиви цієї групи починаються із "v-" (рис. 3.39). Усі існуючі директиви можна знайти у офіційній документації.

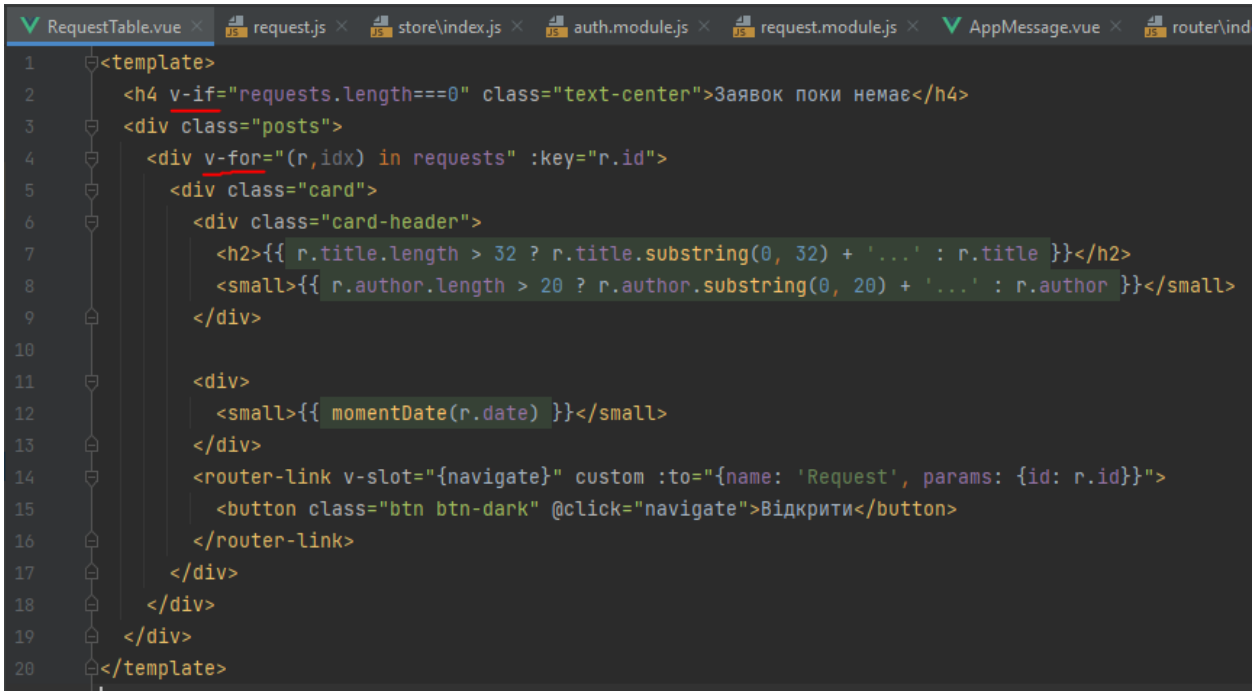


Рисунок 3.39 – Приклади використання директив

Підключення до бази даних відбувається за допомогою окремого HTTP клієнту axios. Axios – це клієнт HTTP на основі Promise для node.js та браузера. Він ізоморфний (може працювати у браузері та nodejs з тією ж базою кодів). На стороні сервера він використовує рідний http-модуль node.js, тоді як на клієнті (браузер) він використовує XMLHttpRequests [19].

У окремому файлі проекту створюється такий клієнт, до якого передається значення baseURL як посилання до бази даних та нижче проводиться обробка можливих помилок, наприклад, помилка 401 (рис. 3.40).

Якщо надійшло повідомлення про помилку 401 (Unauthorized), це означає, що хтось намагається отримати доступ до сторінки, на яку потрібно спочатку увійти, використовуючи дійсний ID користувача та пароль для перегляду [20].

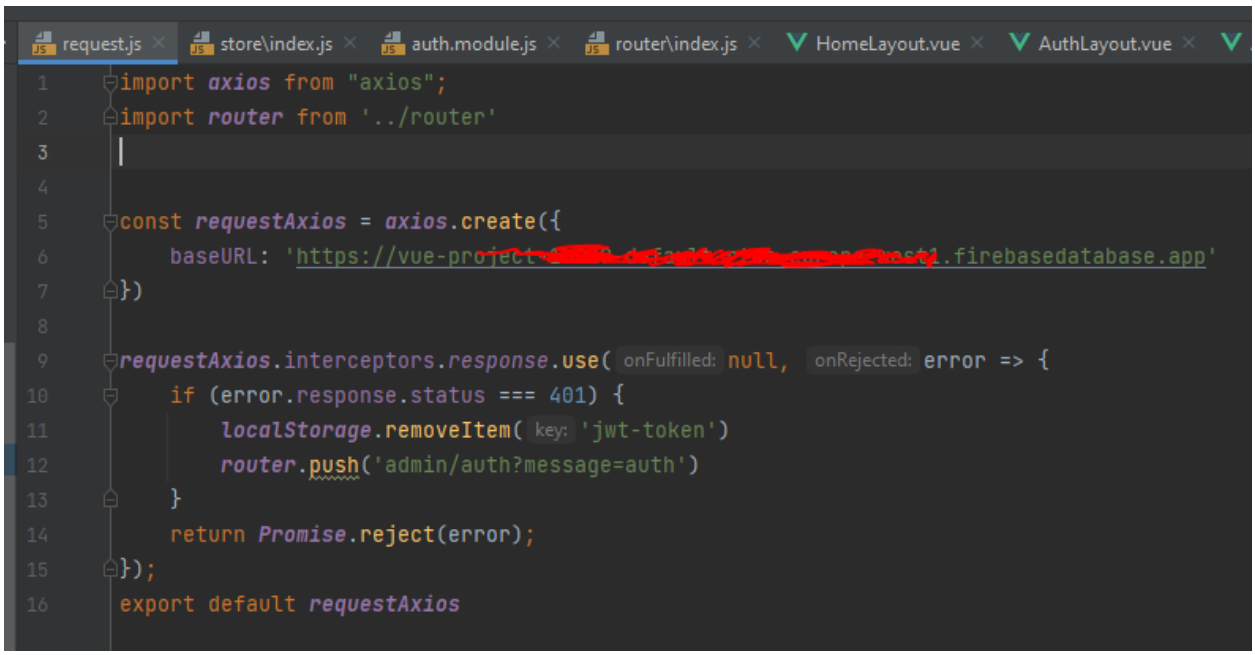


Рисунок 3.40 – Підключення до бази даних

Після цього є можливість до використання, у окремому файлі, стандартних запитів до бази даних, на рис. 3.41, 3.42 можна побачити запити POST та GET. За допомогою dispatch проводиться передача даних до функції setMessage, яка показує повідомлення про успішне виконання запиту чи помилку.

```
async create({commit, dispatch}, payload) {
    try {
        const token = store.getters['auth/token']
        const {data} = await axios.post( url: `/posts.json?auth=${token}`, payload)
        commit('addRequest', {
            ...payload,
            id: data.name
        })
        dispatch('setMessage', {
            value: 'Публікація була створена!',
            type: 'alert-success'
        }, {root: true})
    } catch (e) {
        dispatch('setMessage', {
            value: e.message,
            type: 'alert-danger'
        }, {root: true})
    }
},
```

Рисунок 3.41 – Запит POST

```
async load({commit, dispatch}) {
    try {
        const token = store.getters['auth/token']
        const {data} = await axios.get( url: `/posts.json?auth=${token}`)
        const requests = Object.keys(data).map(id => ({...data[id], id}))
            .sort( compareFn: (a : any & {id: any} ,b : any & {id: any} ) => new Date(b.date) - new Date(a.date))
        commit('setRequests', requests)
    } catch (e) {
        dispatch('setMessage', {
            value: e.message,
            type: 'alert-danger'
        }, {root: true})
    }
},
```

Рисунок 3.42 – Запит GET

Таким чином виконуються 2-5 етапи процесу розробки сайту використовуючи Front-end фреймворк Vue.js. Як можна побачити, логіка залишається майже такою самою, як і у Angular, але синтаксис сильно відрізняється. Також можна сказати, що кожен з них має свої унікальні особливості.

Через що, пропонується включити процедуру вибору відповідного Front-end фреймворку у процес розробки сайту. Тоді схема процесу виглядатиме так, як показано на рис. 3.43.

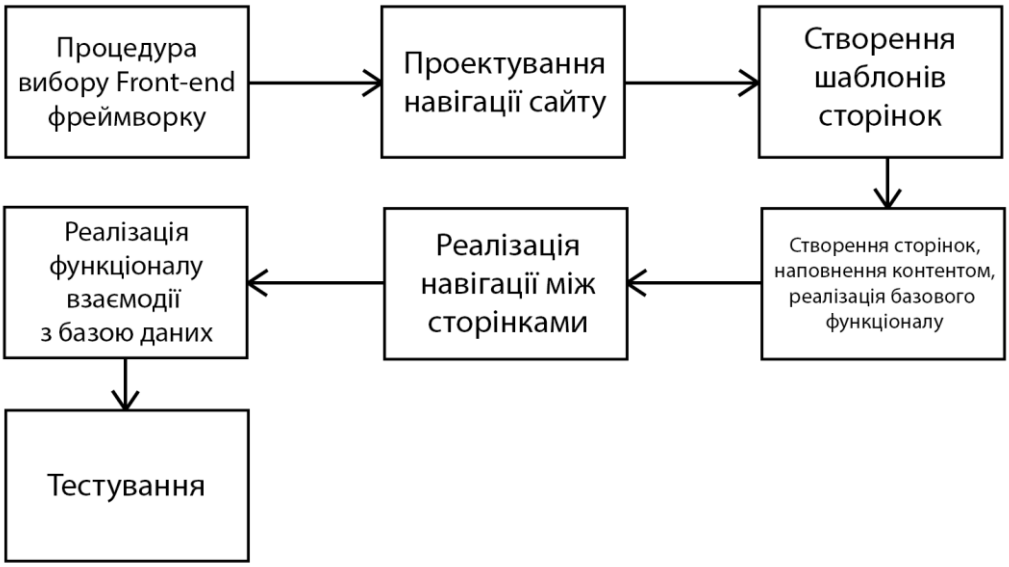


Рисунок 3.43 – Схема процесу розробки

4 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

4.1 Створення сайтів

В експериментальній частині роботи описується практична реалізація запропонованих математичних моделей, методів і алгоритмів. Для цього розробляють план експериментальних досліджень, описують процес виконання і приводять їх результати [21]. Тому, у моєму випадку, експерименту частину перш за все потрібно починати з створення сайтів.

Для створення Web-сайтів, в першу чергу слід визначитися з послідовністю задач. У моєму випадку, було обрано саме таку послідовність: визначення цілей і завдань проекту, планування архітектури проекту, вибір інструментальних засобів, проектування графічного інтерфейсу, розробка дизайн-макетів сторінок, верстання та програмування, наповнення контентом та тестування і запуск.

Оскільки мета цих сайтів полягає в їх функціональному порівнянні, тому і потрібно сконцентрувати увагу саме на функціональних можливостях Front-end фреймворків та реалізувати їх. Для того, щоб сайти працювали правильно, буде використовуватися додавання компонентів, модулів, роутерів та сервісів. Все це для того, щоб повністю використати можливості фреймворків та досягти максимально можливої оптимізації.

Для експериментальної розробки було обрано блог, на головній сторінці якого будуть відображатися всі існуючі пости, кожен з яких можна відкрити. Також існуватиме панель адміністрації, для входу до якої необхідно зайти до системи. Сама реєстрація буде відсутня, оскільки вона існуватиме лише для адміністраторів. На панелі адміністратора можна буде редагувати будь-який існуючий пост, видалити, чи створити новий. Також в ній буде присутня фільтрація постів за їх назвою.

Написання коду буде проведено у програмному забезпеченні PhpStorm. Ця програма є платною, але вона виправдовує витрачені на неї гроші, бо надає дуже зручний у використанні інтерфейс, має змогу швидко підключити Git (середовище для викладання у інтернет створених проектів), та додавати безліч бібліотек та фреймворків за допомогою вбудованої консолі. В ній є можливість написання основного коду, який пов'язується між собою, наприклад натиснувши на клас деякого елемента, з затриманою кнопкою Ctrl, програма покаже, де ще використовується цей клас (тобто, де написані його стилі чи де цей клас використовується у файлах JavaScript), чи якщо перейменувати деякий файл, перейменуються усі посилання на нього у всьому коді, заздалегідь спитавши у користувача про необхідність цього. Таким чином це забезпечення заощаджує велику кількість часу, тому через всі ці відмінності було обрано саме PhpStorm. PhpStorm являє собою інтелектуальний редактор для PHP, HTML і JavaScript з можливостями аналізу коду на льоту, запобігання помилок у сирцевому коді і автоматизованими засобами рефакторинга для PHP і JavaScript [22]. Також це програмне забезпечення для зручності можна налаштувати під необхідні Front-end фреймворки, за допомогою системи плагінів. Таким чином програма сама буде попереджувати про помилки з написанням коду обраного фреймворку, причому не тільки синтаксичні, але й про помилки, наприклад з неправильною передачею типу змінної до якогось методу.

Для встановлення допоміжних бібліотек, було обрано сервіс npm, який за допомогою команд може завантажити бібліотеку, як paket manager, до будь-якого проекту, після чого буде можливість до їх використання. Сам npm – це менеджер пакетів, що входить до складу Node.js. Установка його проводиться за допомогою команд, які вводяться в консоль [23].

Графічний інтерфейс буде максимально простим, оскільки головна мета сайтів – показати функціональну складову фреймворків, тому наголос буде ставитися саме на це. Тож, було створено саме такі дизайн-макети сторінок для головної сторінки, сторінки поста на головній сторінці та

сторінці адміністраторів та головної панелі адміністратора, які можна побачити на рис. 4.1-4.3.

Рисунок 4.1 – Головна сторінка

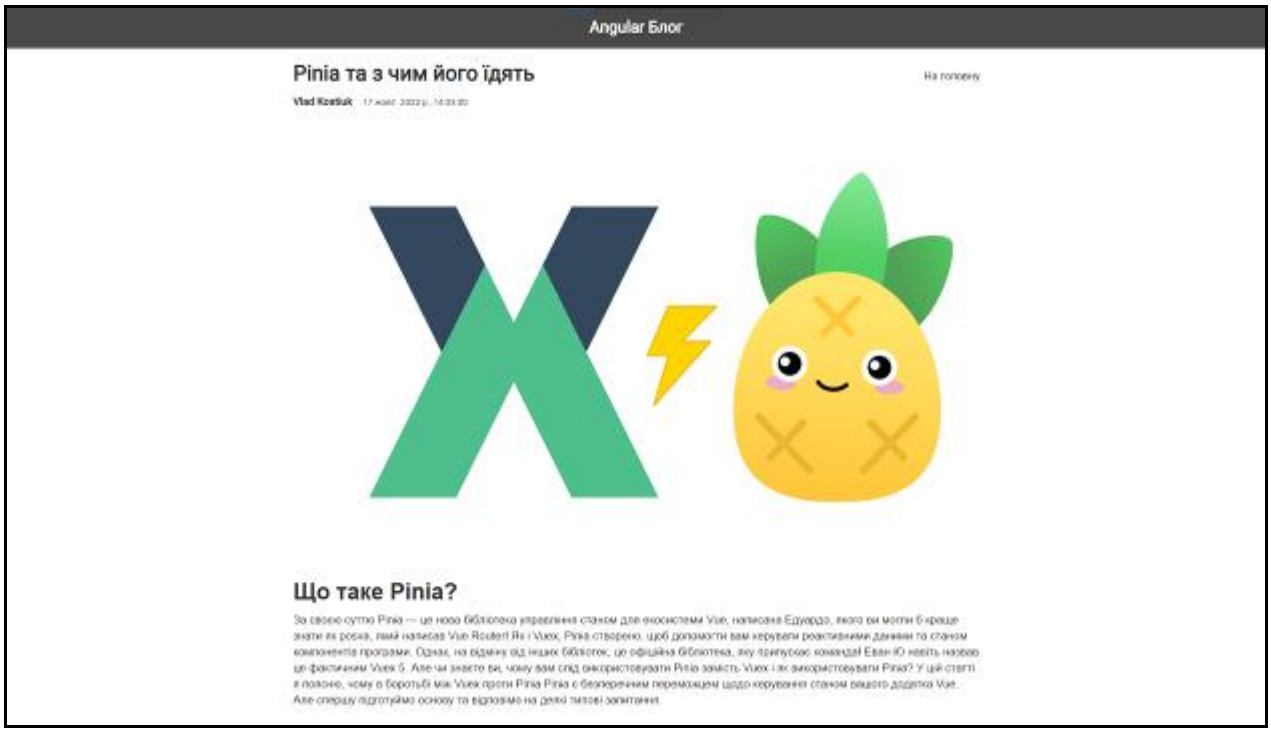


Рисунок 4.2 – Сторінка поста

Рисунок 4.3 – Панель адміністратора

Наступним етапом є верстання, програмування та наповнення контентом. Використовуючи актуальну інформацію з документації по кожному фреймворку, були створені компоненти, модулі (використовуються тільки в Angular), роутери та сервіси (були використані тільки в Angular). Приклади можна побачити на рисунках 4.4-4.12. Було правильно їх пов’язано, оскільки це має велике значення. Наприклад, у компонента головної сторінки є компонент списку постів, а сторінка одного поста є компонентом компонента головної сторінки. Також пов’язання модулів, роутерів та сервісів повинно відповідати лише тим компонентам, на яких використовуються, а змінні яких мають бути правильно захищені.

```
<ng-template #ElseBlock>
  <p class="text-center">Завантаження...</p>
</ng-template>
<h3 *ngIf="error" class="text-center text-white">
  {{error}}
</h3>

<div *ngIf="post$ | async as post; else ElseBlock">
  <div class="post">
    <div class="header">
      <button routerLink="/" class="btn btn-link" *ngIf="!error">На головну</button>
      <h1>{{post.title}}</h1>
    </div>

    <div class="info">
      <strong>{{post.author}}</strong>
      <small>{{post.date|date: format 'medium': timezone '': locale 'ua'}}</small>
    </div>

    <div *ngIf="post.text">
      <quill-view-html [content]="post.text"></quill-view-html>
    </div>
  </div>
</div>
```

Рисунок 4.4 – Компонент сторінки поста на Angular у файлі .html

Рисунок 4.5 – Компонент сторінки поста на Angular у файлі .ts

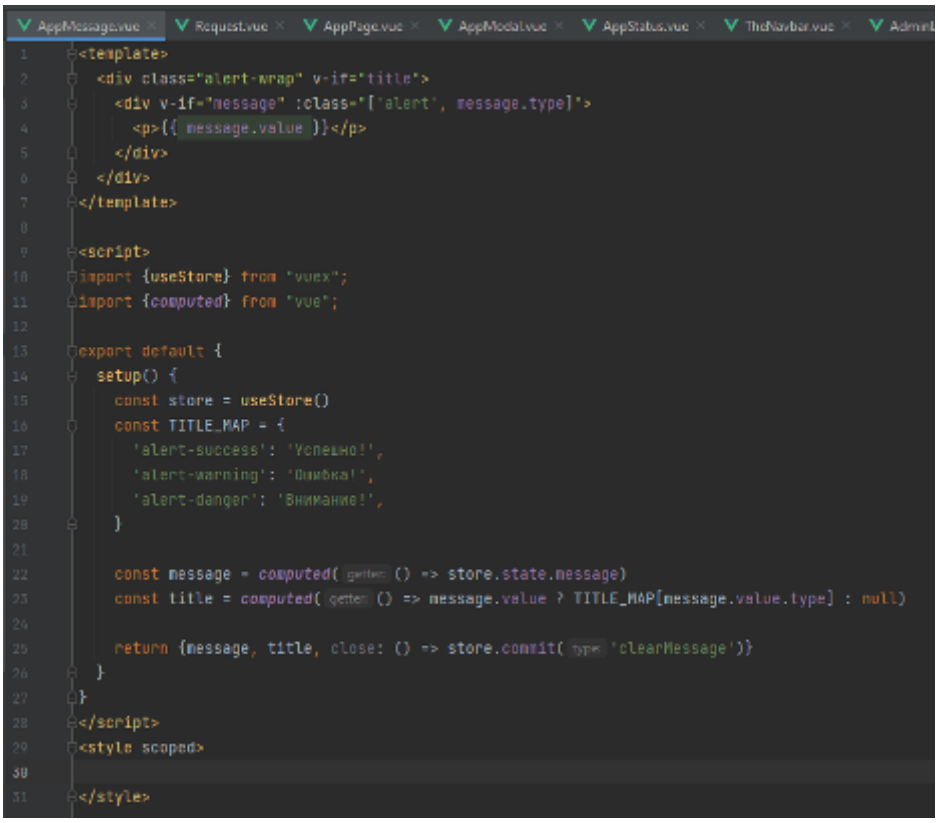


Рисунок 4.6 – Компонент повідомлення на Vue.js

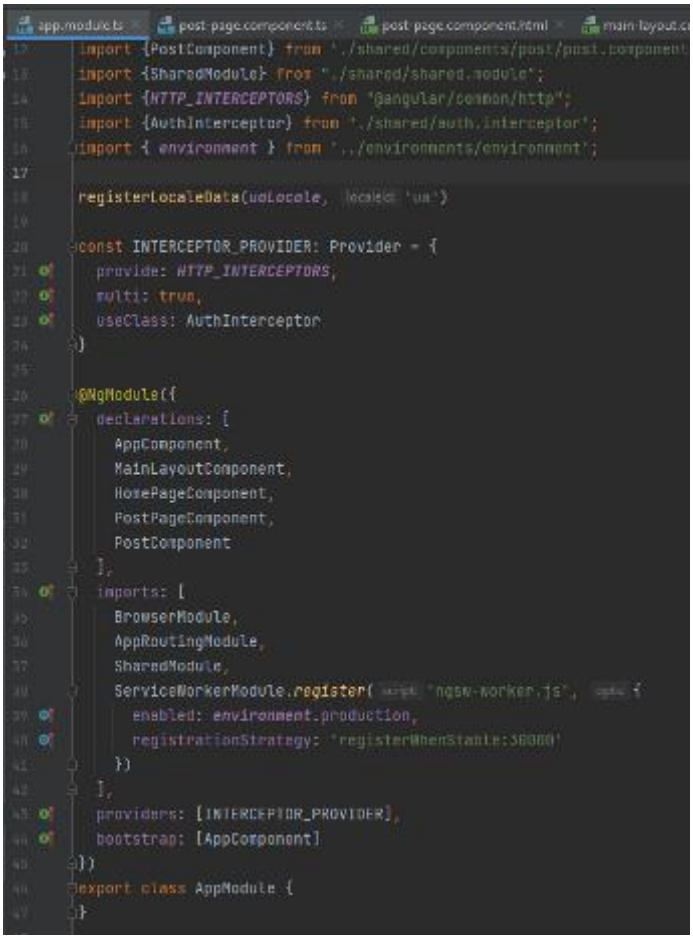


Рисунок 4.7 – Модуль на Angular

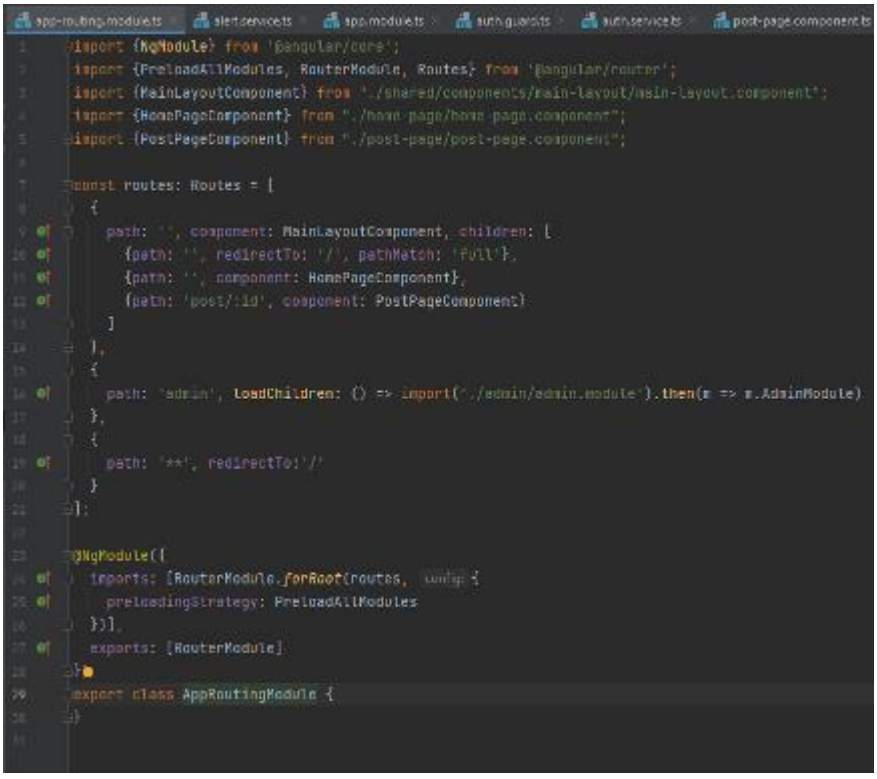
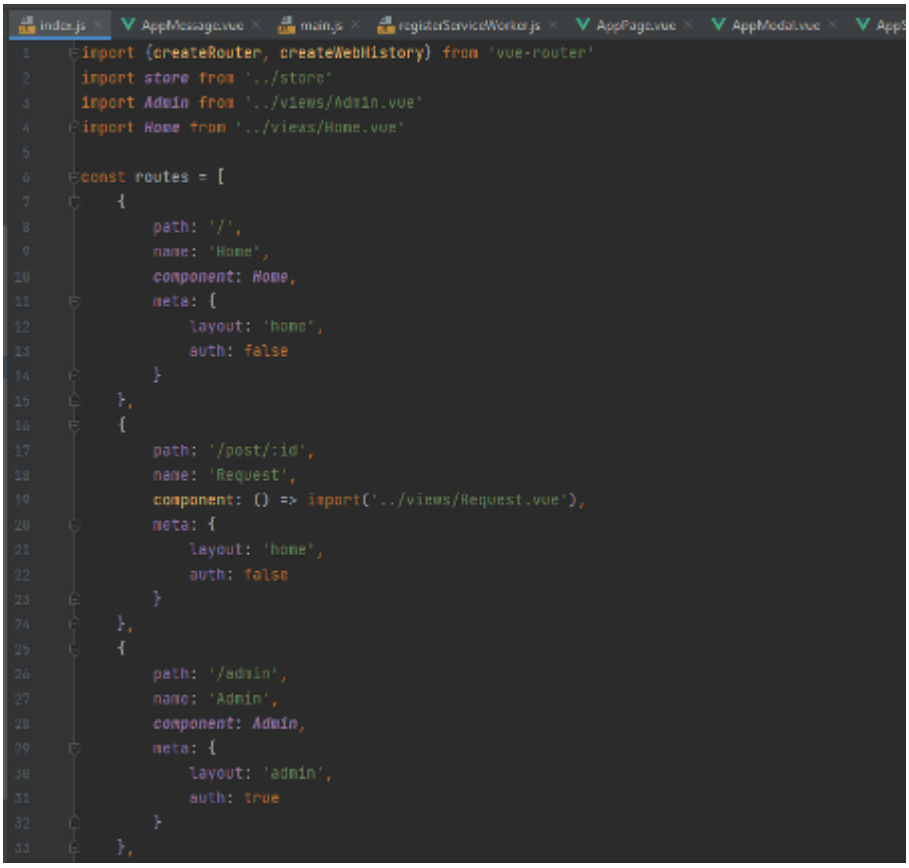


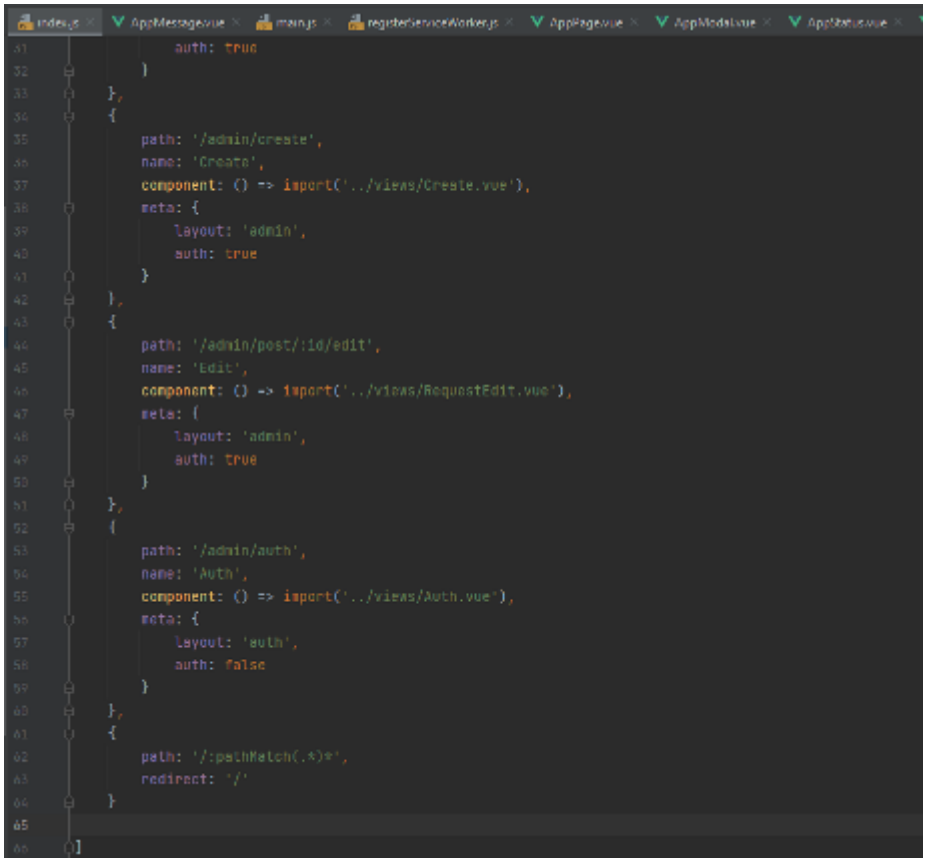
Рисунок 4.8 – Роутери на Angular



```
import {createRouter, createWebHistory} from 'vue-router'
import store from '../store'
import Admin from '../views/Admin.vue'
import Home from '../views/Home.vue'

const routes = [
    {
        path: '/',
        name: 'Home',
        component: Home,
        meta: {
            layout: 'home',
            auth: false
        }
    },
    {
        path: '/post/:id',
        name: 'Request',
        component: () => import('../views/Request.vue'),
        meta: {
            layout: 'home',
            auth: false
        }
    },
    {
        path: '/admin',
        name: 'Admin',
        component: Admin,
        meta: {
            layout: 'admin',
            auth: true
        }
    },
```

Рисунок 4.9 – Роутери на Vue.js



```
            auth: true
        }
    },
    {
        path: '/admin/create',
        name: 'Create',
        component: () => import('../views/Create.vue'),
        meta: {
            layout: 'admin',
            auth: true
        }
    },
    {
        path: '/admin/post/:id/edit',
        name: 'Edit',
        component: () => import('../views/RequestEdit.vue'),
        meta: {
            layout: 'admin',
            auth: true
        }
    },
    {
        path: '/admin/auth',
        name: 'Auth',
        component: () => import('../views/Auth.vue'),
        meta: {
            layout: 'auth',
            auth: false
        }
    },
    {
        path: '/:pathMatch(.*)*',
        redirect: '/'
    }

]
```

Рисунок 4.10 – Роутери на Vue.js, продовження

```
const router = createRouter( options: {
    history: createWebHistory(process.env.BASE_URL),
    routes,
    linkActiveClass: 'active',
    linkExactActiveClass: 'active'
})

router.beforeEach( guard: (to : RouteLocationNormalized , from : RouteLocationNormalized , next : NavigationGuardNext ) => {
    const requireAuth = to.meta.auth

    if (requireAuth && store.getters['auth/isAuthenticated']) {
        next()
    } else if (requireAuth && !store.getters['auth/isAuthenticated']) {
        next('admin/auth?message=auth')
    } else {
        next()
    }
})

export default router
```

Рисунок 4.11 – Роутери на Vue.js, продовження

```
alert.service.ts | app-routing.module.ts | app.module.ts | auth.guard.ts | auth.servi

import {Injectable} from "@angular/core";
import {Subject} from "rxjs";

export type AlertType = 'success' | 'warning' | 'danger' | 'info'

export interface Alert {
  type: AlertType
  text: string
}

@Injectable()
export class AlertService {
  public alert$ = new Subject<Alert>()

  success(text: string) {
    this.alert$.next( value: {type: 'success', text})
  }
  warning(text: string) {
    this.alert$.next( value: {type: 'warning', text})
  }
  danger(text: string) {
    this.alert$.next( value: {type: 'danger', text})
  }

  info(text: string) {
    this.alert$.next( value: {type: 'info', text})
  }
}
```

Рисунок 4.12 – Сервіс повідомлення на Angular

Контентом сайту було обрано інформацію про нові технології по фреймворкам Vue.js та Angular англійською та українською мовами. Хоча контент сайту не грає ніякої ролі у моїй роботі, але необхідно було його чимось наповнити, щоб було візуально видно, як він працює.

Тестування сайтів проводилось такими способами, як: тестування на кросбраузерність та тестування відображення на різних пристроях з різними розмірами екрану.

Спочатку сайти протестовано на кросбраузерність – це властивість сайту відображатися та працювати без жодних проблем у різних браузерах. Тестування проводилося у найпопулярніших браузерах: Google Chrome, Microsoft Edge, Opera, Mozilla Firefox. У всіх браузерах відображення та функціонал сайтів не загубився. Все гарно відображалося та працювало без будь-яких проблем.

Наступним етапом тестування було саме тестування на різних екранах та пристроях за допомогою «Переглянути код», функцією, яка є майже у всіх сучасних браузерах. При усіх розмірах екрану всі сторінки, та особливості сайтів добре працювали.

Запуск на хостингу було зроблено за допомогою сервісу Google Firebase, який надає змогу як і використовувати вбудовану базу даних, так і проводити запуск на безкоштовному домені.

4.2 Вибір методу

Експеримент служить для отримання нових наукових знань. Від звичайного пасивного спостереження експеримент відрізняється активним впливом дослідника на явище, що вивчається. Основна мета експерименту – перевірка теоретичних положень, підтвердження робочої гіпотези, а також ширше та глибше вивчення теми наукового дослідження.

Розрізняють експерименти природні та штучні. Перші експерименти притаманні соціальних явищ, наприклад, виробництва. Штучні експерименти

широко застосовуються насамперед у технічних науках. Експериментальні дослідження своєю чергою поділяються на лабораторні та виробничі [24].

Оскільки тема кваліфікаційної роботи – «Дослідження корисності фреймворків для Front-end розробки», тому методи оцінки повинні бути пов’язані з якістю досвіду користувача, сприйняття користувача та досвіду взаємодії. Для формального розрахунку буде обрано метод експертного оцінювання.

4.3 Обґрунтування вибору методу та аналіз прогнозованих результатів

Для оцінки якості досвіду користувача було обрано метод експертного оцінювання, оскільки цей метод допомагає у ході роботи з фокус-групою оцінити обрані експертами критерії, відповідні до функціональності сайтів. Цей метод краще інших підійде для отримання правильних результатів кваліфікаційної роботи. Тепер можна отримати результати за допомогою математичних розрахунків та на їх основі зробити висновки.

4.4 Проведення експерименту

Метод експертних оцінок – один з основних класів методів науково-технічного прогнозування, який ґрунтується на припущенні, що на основі думок експертів можна збудувати адекватну модель майбутнього розвитку об'єкта прогнозування [25].

Першим етапом цього методу є визначення головних критеріїв оцінювання сайтів, на основі опитувань експертів. Таким чином було обрано такі критерії, як: оптимізація (швидкість завантаження сторінок сайту), масштабованість, архітектура, складність написання коду з боку програміста, безпека та розмір збірки.

Далі необхідно провести оцінювання експертами обраних критеріїв, щоб знайти їх значимість, та думки експертів щодо узгодженості V (табл. 4.1).

Таблиця 4.1 – Оцінка критеріїв

	Експерт														
Критерій	1	2	3	4	5	6	7	8	9	10	Сума	Вага кри.	X	σ	V
Оптимізація	4	3	6	5	4	6	1	2	4	1	36	0.17	3.6	0.55	0.15
Масштабованість	2	4	3	4	5	1	4	1	1	3	28	0.13	2.8	0.44	0.16
Архітектура	6	5	5	6	6	4	6	3	6	6	53	0.25	5.3	0.32	0.06
Складність написання коду з боку програміста	1	2	1	3	1	2	2	5	2	5	24	0.11	2.4	0.45	0.19
Безпека	5	6	4	2	3	5	5	6	5	4	45	0.21	4.5	0.38	0.08
Розмір збірки	3	1	2	1	2	3	3	4	3	2	24	0.11	2.4	0.29	0.12

Чим вище значення ваги критерія, тим він більш важливіший за інші. Як можна побачити з результату, найважливішим критерієм на думку експертів є саме архітектура проекту, далі йде безпека, оптимізація, масштабованість та складність написання коду з боку програміста і розмір збірки.

За цими критеріями експерти оцінили кожний з двох розроблених сайтів. Відносні ваги сайтів, думки експертів щодо узгодженості V для кожного сайту наведені в табл. 4.2-4.7.

Таблиця 4.2 – Оцінка фреймворків за оптимізацією

	Експерт														
Проект	1	2	3	4	5	6	7	8	9	10	Сумма	Вага кри.	X	σ	V
Angular	2	2	1	2	1	2	2	2	2	1	17	0.57	1.7	0.14	0.09
Vue.js	1	1	2	1	2	1	1	1	1	2	13	0.43	1.3	0.14	0.11
	Оптимізація														

Таблиця 4.3 – Оцінка фреймворків за масштабованістю

	Експерт														
Проект	1	2	3	4	5	6	7	8	9	10	Сума	Вага кри.	X	σ	V
Angular	2	1	1	1	2	1	2	2	2	2	16	0.53	1.6	0.15	0.10
Vue.js	1	2	2	2	1	2	1	1	1	1	14	0.47	1.4	0.15	0.11
	Масштабованість														

Таблиця 4.4 – Оцінка фреймворків за архітектурою

	Експерт														
Проект	1	2	3	4	5	6	7	8	9	10	Сума	Вага кри.	X	σ	V
Angular	1	2	2	2	1	2	2	2	1	2	17	0.57	1.7	0.14	0.09
Vue.js	2	1	1	1	2	1	1	1	2	1	13	0.43	1.3	0.14	0.11
	Архітектура														

Таблиця 4.5 – Оцінка фреймворків за складністю написання коду з боку програміста

	Експерт														
Проект	1	2	3	4	5	6	7	8	9	10	Сума	Вага кри.	X	σ	V
Angular	1	1	1	1	2	2	1	2	1	2	14	0.47	1.4	0.15	0.11
Vue.js	2	2	2	2	1	1	2	1	2	1	16	0.53	1.6	0.15	0.10
	Складність написання коду з боку програміста														

Таблиця 4.6 – Оцінка фреймворків за безпекою

	Експерт														
Проект	1	2	3	4	5	6	7	8	9	10	Сума	Вага кри.	X	σ	V
Angular	2	2	2	2	1	1	2	2	2	2	18	0.60	1.8	0.13	0.07
Vue.js	1	1	1	1	2	2	1	1	1	1	12	0.40	1.2	0.13	0.11
	Безпека														

Таблиця 4.7 – Оцінка фреймворків за розміром збірки

	Експерт														
Проект	1	2	3	4	5	6	7	8	9	10	Сума	Вага кри.	X	σ	V
Angular	1	1	1	1	2	1	1	1	1	2	12	0.40	1.2	0.13	0.11
Vue.js	2	2	2	2	1	2	2	2	2	1	18	0.60	1.8	0.13	0.07
	Розмір збірки														

Для підтвердження результату оцінювання експертів за показником оптимізації було використано сервіс PageSpeed Insights компанії Google, який аналізує сайт по його посиланню і визначає якість продуктивності для мобільних пристроїв та комп’ютера (рис. 4.13-4.16).

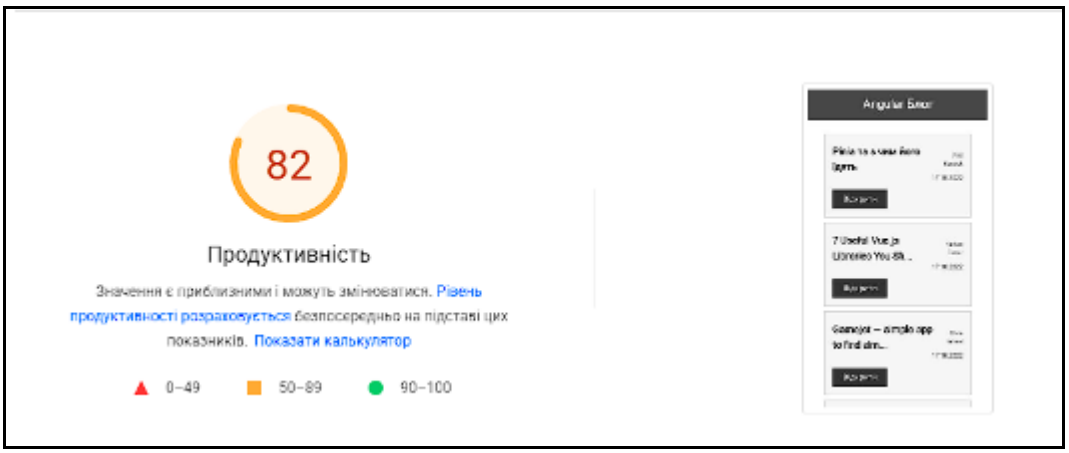


Рисунок 4.13 – Продуктивність Angular на мобільному пристрої

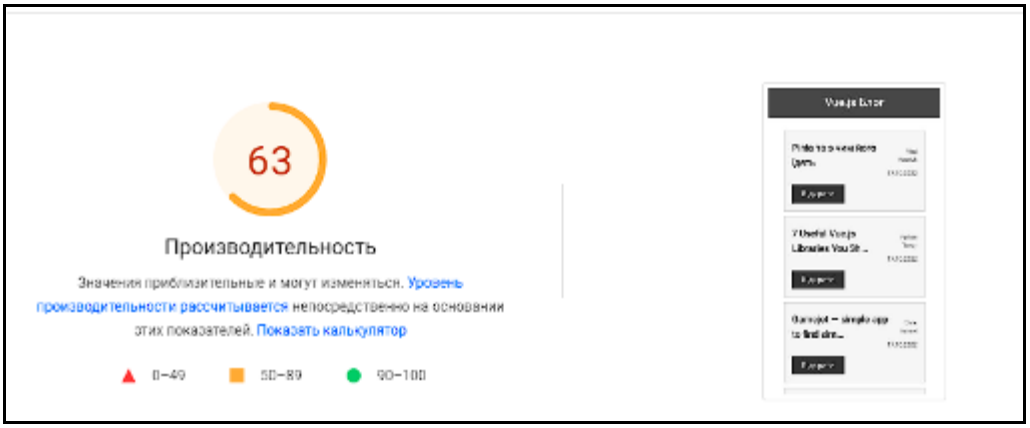


Рисунок 4.14 – Продуктивність Vue.js на мобільному пристрої

Рисунок 4.15 – Продуктивність Angular на комп’ютері

Рисунок 4.16 – Продуктивність Vue.js на комп’ютері

Результати сервісу PageSpeed Insights повністю підтверджують отримані оцінки експертів за даним показником.

За отриманим значенням ваги критерія можна сказати, що, на думку експертів, Angular краще Vue.js по критеріями оптимізації, масштабованості, архітектурі та безпеки, але все ж таки за складністю написання коду з боку програміста та розміру збірки Vue.js ліпший за Angular.

Після цього необхідно порівняти фреймворки за узагальненими рейтингами:

$$R_1 = 0{,}17 \times 0{,}57 + 0{,}13 \times 0{,}53 + 0{,}25 \times 0{,}57 + 0{,}11 \times 0{,}47 + 0{,}21 \times 0{,}60 + 0{,}11 \times 0{,}40 = 0{,}54;$$
$$R_2 = 0{,}17 \times 0{,}43 + 0{,}13 \times 0{,}47 + 0{,}25 \times 0{,}43 + 0{,}11 \times 0{,}53 + 0{,}21 \times 0{,}40 + 0{,}11 \times 0{,}60 = 0{,}46.$$

Очевидно, що Angular (R_1 = 0,54), за думкою експертів, буде кращім вибором до вашого проекту, ніж Vue.js (R_2 = 0,46), але при цьому Vue.js має свої позитивні якості, на відміну від Angular.

Цей висновок підтверджує об’єктивний показник продуктивності, визначений стороннім ресурсом PageSpeed Insights. Таким чином, додавання етапу вибору фреймворку та використання для цього експертного методу оцінювання дозволяє обрати найбільш корисний інструмент для створення заданого типу сайту, та забезпечити при цьому найвищу можливу якість.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Характеристика науково-дослідної роботи

Метою даного розділу є економічне обґрунтування витрат на проведення науково-дослідної роботи (НДР), в межах якої передбачається дослідження корисності фреймворків для Front-end розробки. Під час такого обґрунтування буде здійснено: розрахунок трудовитрат та заробітної плати працівникам, розрахунок одноразових витрат, оцінку результатів НДР.

Реалізація НДР передбачає такі етапи:

- аналіз предметної області;
- визначення алгоритму реалізації проекту;
- дослідження фреймворків для Front-end розробки;
- створення сайтів за допомогою Front-end фреймворків;
- вибір методів для проведення експерименту;
- проведення експерименту та аналіз отриманих результатів.

5.2 Етапи виконання НДР, їх трудомісткість та заробітна плата

Під час виконання науково-дослідної роботи було проведено аналіз літератури за темою дослідження та виявлено що, створюючи цифровий продукт, потрібно заздалегідь думати про його розвиток та подальшу долю. Але якість розробки може суттєво відрізнятись, залежно від обраного фреймворку. Найпопулярнішими сьогодні є Vue чи Angular. Тому було більш детальніше розглянуто важливість використання Front-end фреймворків та їх відмінності процесу розробки відносно типового. Після чого проведено експеримент, шляхом створення двох сайтів, використовуючи для першого фреймворк Angular, а для другого Vue та методом експертного оцінювання, за обраними критеріями, було отримано результати, за якими можна зробити

висновок, який з фреймворків є більш кращим вибором до вашого проекту. У результаті оцінено продуктивність та якість розроблювальних сайтів для подальшого використання.

Умовно науково-дослідну роботу (НДР) можна розділити на такі етапи: підготовчий, основний і заключний.

На стадії виконання підготовчого етапу здійснено підбір експертами критеріїв оцінки якості розроблених сайтів та проведено оцінку важливості кожного критерія.

На етапі виконання основної частини НДР були виконані такі роботи:

– розробка сайтів з використанням обраних Front-end фреймворків;

– оцінка якості розроблених прототипів;

– обробка та аналіз результатів експерименту;

У заключній частині здійснюється оцінка ефективності виконання НДР, складання звіту з НДР, захист звіту.

Найбільш складною та відповідальною частиною при плануванні НДР є розрахунок трудомісткості робіт, тому що трудові витрати часто становлять основну частину вартості науково-дослідних робіт і безпосередньо впливають на строки розробки.

Для виконання роботи було залучено 14 осіб, контролював процес керівник роботи, тобто робоча загальна чисельність на виконання НДР склала 15 осіб. До складу групи виконавців увійшли:

– керівник експертами – 1 особа, заробітна плата 14000 грн/міс.;

– учасники експертного оцінювання – 10 осіб, заробітна плата 9000 грн/міс.;

– Front-end розробник Angular – 1 особа, заробітна платня 25000 грн/міс.;

– Front-end розробник Vue.js – 1 особа, заробітна платня 19000 грн/міс.;

– спеціаліст у сфері обробки статистичних даних – 1 особа, заробітна плата 16000 грн/міс.;

– керівник роботи – 1 особа, заробітна плата 9000 грн/міс.

Проведемо розрахунок трудовитрат і заробітної плати виконавців робіт.

Середньоденна заробітна плата виконавця робіт (Зср.дн.) розраховується за формулою:

$$З_{ср.дн.} = \frac{З_{ср.міс.}}{n}, \quad (5.1)$$

де $З_{ср.міс.}$ – середньомісячна зарплата виконавця роботи;

n – число робочих днів у місяці, (n = 22).

Етапи виконання НДР, перелік і зміст робіт, трудомісткість їх виконання, заробітна плата виконавців робіт представлені в табл. 5.1.

Таблиця 5.1 – Розрахунок трудовитрат і заробітної плати виконавців робіт

Перелік робіт	Кіль-кість вико-навців	Посада виконавця	Трудомісткість робіт, люд.-днів	Середньоденна заробітна плата, грн.	Сума заробітної плати, грн.
1	2	3	4	5	6
1. Підготовчий етап					
1.1. Розробка та затвердження ТЗ	1	Керівник роботи	2	409,09	818,18
1.2 Підготовка довідкових матеріалів та даних для виконання НДР	1	Керівник роботи	2	409,09	818,18
2. Основний етап					
2.1 Постановка задачі	1	Керівник роботи	1	409,09	409,09
2.2 Обирання критеріїв оцінювання	1	Керівник експертами	1	636,36	636,36
2.2.1 Група експертів	10	Учасники експертного оцінювання	1	409,09	4090,9
2.3 Обробка результатів	1	Керівник експертами	1	636,36	636,36
2.4 Розробка сайту використовуючи Angular	1	Front-end розробник Angular	16	1136,36	18181,76
2.4 Розробка сайту використовуючи Vue.js	1	Front-end розробник Vue.js	16	863,64	13818,24
2.5 Проведення експертного оцінювання	1	Керівник експертами	1	636,36	636,36
2.5.1 Група експертів	10	Учасники експертного оцінювання	1	409,09	4090,9

Продовження таблиці 5.1

1	2	3	4	5	6
2.6 Обробка результатів експерименту	1	Спеціаліст в сфері обробки статистичних даних	3	727,27	2181,81
3. Заключний етап					
3.1 Технічне оформлення звіту про виконання НДР	1	Керівник роботи	2	409,09	818,18
Всього			47		47136,32

Таким чином, сума витрат на заробітну плату в межах виконання НДР складе 47136,32 грн.

5.3 Розрахунок одноразових витрат на розробку НДР

Калькуляція собівартості розраховується відповідно до існуючих нормативних актів України. До складу калькуляції входять такі статті витрат:

– матеріальні витрати;

– витрати на оплату праці;

– єдиний соціальний внесок;

– амортизація основних засобів (вартість машинного часу);

– витрати на спожиту електроенергію;

– інші витрати.

До інших витрат відносяться адміністративні витрати (водопостачання, водовідведення, опалення, освітлення) та вартість послуг зв’язку.

Матеріальні витрати визначаються витратами на матеріали, визначені їх потребою для виконання робіт, і цін, що діють на момент складання калькуляції. Матеріальні витрати розраховуються за такою формулою:

$$M = \sum_{j=1}^{n} Q_j \times Ц_j, \tag{5.2}$$

де Q_j – кількість використаних одиниць j-го виду матеріалів, $j = (1 \div n)$;

M – сумарні витрати на матеріали, в тому числі малоцінні предмети, що швидко зношуються (носії, папір, канцелярське приладдя тощо), або на літературу, яка необхідна для проведення роботи, тощо;

$Ц_j$ – ціна одиниці j-го виду матеріалів.

Розрахунок матеріальних витрат представлено в табл. 5.2.

Таблиця 5.2 – Розрахунок матеріальних витрат

Найменування	Од. вим.	Кількість, од.	Ціна, грн	Сума, грн.
Олівець механічний	уп.	1	100,00	100,00
Ручки	уп.	1	250,00	250,00
Папір	уп.	1	250,00	250,00
Степлер	шт.	2	30,00	60,00
Скріпки для степлеру	уп.	2	30,00	60,00
Калькулятор	шт.	4	37,00	148,00
Всього				868

Витрати на оплату праці розраховуються, виходячи з необхідного для виконання робіт складу й кількості працівників, а також із середньомісячної заробітної плати. Відповідно до проведених розрахунків витрати на оплату праці виконавців роботи дорівнюють 47136,32 грн.

Єдиний внесок на загальнодержавне соціальне страхування (ЄСВ) – консолідований страховий внесок, збір якого здійснюється в систему загально обов’язкового державного соціального страхування в обов’язковому порядку і на регулярній основі з метою забезпечення захисту у випадках, передбачених законодавством, прав застрахованих осіб і членів їх сімей на отримання страхових виплат (послуг) за діючими видами загальнообов’язкового державного соціального страхування.

Ставка єдиного соціального внеску складає 22 % від витрат на оплату праці, тобто розмір ЄСВ дорівнює 10369,99 грн.

Під час виконання НДР застосовувалось наступне обладнання: комп’ютер вартістю 10000 грн та принтер вартістю 3000 грн.

Вищенаведене устаткування є власністю організації виконавця, тому доцільно розрахувати суму амортизаційних відрахувань на період виконання НДР.

Амортизація основних засобів розраховується за формулою:

$$AB = \sum_{k=1}^{L} \frac{BO_k}{TE_k} \times T\,, \tag{5.3}$$

де AB – сума амортизаційних відрахувань, нарахованих під час проведення науково-дослідної роботи;

BO_k – вартість основних засобів k-го виду;

TE_k – термін експлуатації основних засобів k-го виду, днів;

T – термін науково-дослідницької роботи, днів;

L – кількість видів обладнання.

Підставивши відомі значення у (5.3), визначимо величину амортизаційних відрахувань:

$$AB = \frac{10000 \cdot 47}{730} + \frac{3000 \cdot 47}{730} = 643{,}84 + 193{,}15 = 836{,}99 \text{ (грн).}$$

Витрати на використану обладнанням електроенергію (B_e):

$$B_e = M \cdot t \cdot T_{кВт} \tag{5.4}$$

де M – потужність устаткування, тобто кількість енергії, споживаної за одиницю часу (кВт/година);

t – кількість годин використання устаткування за період проведення науково-дослідницької роботи;

$T_{кВт}$ – тариф, тобто вартість використання 1 кВт електроенергії.

Споживна потужність комп’ютера складає 0,5 кВт та принтера 0,8 кВт за годину. Тариф споживачів за першим класом напруги, тобто 35 кВт та більше), складає 1,7808 грн./кВт·годин (без ПДВ). Підставивши значення у формулу (5.4), визначимо величину витрат (B_e) на спожиту електроенергію:

$$B_е = 0{,}5 \cdot 376 \cdot 1{,}7808 + 0{,}8 \cdot 5 \cdot 1{,}7808 = 334{,}79 + 7{,}12 = 341{,}91 \text{ грн.}$$

До інших статей витрат відносяться такі:

– адміністративні витрати: (водопостачання, водовідведення, освітлення, опалення), які прийнято у розмірі 20% від витрат на оплату праці;

– вартість оплати послуг зв'язку.

Вартість оплати послуг зв'язку становитиме:

а) Інтернет – 150 грн. на місяць (безлімітний пакет); всього 450 грн. за 47 днів виконання НДР;

б) телефон – 60 грн. на місяць; усього 180 грн. за 47 днів.

За період виконання НДР витрати на відрядження, аутсорсинг, інформаційні послуги та маркетингові заходи не були потрібними.

Для виконання НДР використовувалося програмне забезпечення, наприклад, для розробки сайтів – PhpStorm, сервіс, який коштує $9,90 в місяць; усього $29,70 за 47 дні виконання НДР, тобто 1095,27 грн.

Результати розрахунку кошторису витрат, тобто одноразових витрат, на виконання НДР, наведені в табл. 5.3.

Таблиця 5.3 − Кошторис витрат на розробку НДР

№ з/п	Показник	Сума, грн.
1	Заробітна плата	47136,32
2	Єдиний соціальний внесок (22,0 % від п.1)	10369,99
3	Матеріальні витрати	868
4	Амортизація основних засобів	836,99
5	Витрати на спожиту електроенергію	341,91
6	Інші витрати, у тому числі:	–
6.1	Адміністративні витрати (20,0 % від п.1)	9427,26
6.2	Вартість послуг зв'язку	630,00
6.3	Вартість програмного забезпечення	1095,27
7	Усього витрати	70705,74

Таким чином, кошторис витрат на виконання даної НДР відбиває сумарні витрати за статтями і складає 70705,74 грн.

5.4 Оцінка результатів науково-дослідної роботи

Результат – це наслідок послідовності дій, виконаних під час НДР, виражений якісно або кількісно. В загальному випадку оцінка результатів НДР – це визначення ефективності отриманих рішень порівняно з сучасним науково-технічним рівнем.

Відповідно до теми даного дослідження у якості результату впровадження НДР визначено збільшення продуктивності сайтів в результаті використання фреймворків для Front-end розробки.

Результат від впровадження НДР визначається за формулою:

$$\Delta P_j = \left| Хб_j - Хн_j \right|, \tag{5.5}$$

де ΔP_j – покращення j-ої характеристики досліджуваного процесу за рахунок впровадження результатів НДР ($j=1,m$);

m – кількість досліджуваних характеристик;

$Хб_j$ – базове значення j-ої характеристики;

$Хн_j$ – нове значення j-ої характеристики після впровадження НДР.

У якості досліджуваної характеристики обрано продуктивність. Для отримання результатів продуктивності сайтів, було використано сервіс PageSpeed Insights компанії Google, який аналізує сайт по його посиланню і визначає якість продуктивності для мобільних пристроїв та комп’ютера. Отримані результати тестування наведені у таблиці 5.4.

Таблиця 5.4 – Показник продуктивності

Показник	Angular	Vue.js
Продуктивність (%)	82	63

Підставивши відповідні значення до формули (5.5), визначимо результат від впровадження НДР у чисельному вигляді:

$$\Delta P_1 = |82 - 63| = 19\%.$$

Сайт з використанням Front-end фреймворку Angular за результатами тестування виявився більш продуктивним, у відмінності від сайту створеному за допомогою Vue.js .

Далі проведено оцінку економічної ефективності отриманого результату виконаної науково-дослідної роботи.

5.5 Визначення економічної ефективності результатів НДР

Для визначення економічної ефективності результатів НДР необхідно порівняти витрати на розробку НДР з отриманими результатами.

Основним показником економічної ефективності науково-дослідної роботи є коефіцієнт «ефект-витрати», який розраховується за формулою:

$$K_{ев} = \frac{\Delta P_j}{B_p}, \tag{5.6}$$

де B_p – витрати (кошторисна вартість) на виконання НДР, грн;

$K_{ев}$ – коефіцієнт «ефект-витрати», який відбиває, наскільки кожна гривня витрат НДР змінює *j*-ту характеристику досліджуваного процесу.

Підставивши раніше визначені значення до (5.6), розрахуємо чисельне значення коефіцієнту «ефект-витрати»:

$$K_{ев} = \frac{19}{70705,74} \cdot 100\% = 0,0269 \text{ (\%)}.$$

У результаті проведених досліджень, можна зробити висновок про те, що кожна гривня витрат на розробку НДР забезпечує підвищення продуктивності розроблювального сайту на 0,0269 %. Дана науково-дослідна робота має позитивний показник економічної ефективності. Роботу у цілому можна вважати ефективною або такою, що має науковий і технічний рівень.

ВИСНОВКИ

Правильний вибір фреймворку для Front-end розробки сайтів є невід’ємною частиною для створення будь-якого великого проекту. В наш час існує дуже багато фреймворків для Front-end розробки сайтів, та перед початком створення проекту актуальним є питання який саме фреймворк використовувати для того, щоб отримати більш якісний продукт.

В даній кваліфікаційній роботі було виявлено, що, створюючи цифровий продукт, потрібно заздалегідь думати про його розвиток та подальшу долю. Було розглянуто важливість використання Front-end фреймворків та їх відмінності в процесі розробки сайту відносно типового.

Виходячи з результатів дослідження, можна зробити висновки, що врахування думки експертів під час вибору фреймворку для заданого проекту дозволяє забезпечити найвищу можливу продуктивність сайту. Також було виявлено, що для сучасних Front-end фреймворків, найголовнішими критеріями є саме: архітектура проекту, безпека та оптимізація (швидкість завантаження сторінок сайту), а такими критеріями, як: масштабованість, складність написання коду з боку програміста та розміром збірки можна знехтувати.

В роботі також було реалізовано експериментальну частину у вигляді розробки сайту за допомогою обраних фреймворків та порівняння якості розробок, а також виконано економічне обґрунтування доцільності проведення науково-дослідної роботи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Фреймворк. URL: https://bit.ly/3wXzLR2 (дата звернення: 11.11.2022).

2. 10 Найкращих фреймворків для фронтенду та бекенду. URL: https://bit.ly/3BfLtt1 (дата звернення: 15.11.2022).

3. Чахальян Д.Є., Кулішова Н.Є. Порівняльний аналіз фремворків для front-end розробки // Глобалізація наукових знань: міжнародна співпраця та інтеграція галузей наук: матеріали III Міжнародної студентської наукової конференції, м. Дніпро, 23 вересня, 2022рік. Вінниця: ГО «Європейська наукова платформа», 2022. С. 104-105.

4. Vue vs Angular: що буде краще для вашого інтерфейсу – Wezom. URL: https://bit.ly/3CLPLHU (дата звернення: 15.11.2022).

5. Найкращі Frontend фреймоворки та біблиіотеки JavaScript (JS) 2020 – Merehead. URL: https://bit.ly/3wZoe3E (дата звернення: 15.11.2022).

6. Angular – Вікі ЦДПУ. URL: https://bit.ly/3KRlTh9 (дата звернення: 15.11.2022).

7. Angular (фреймворк). URL: https://bit.ly/3D0q4oV (дата звернення: 15.11.2022).

8. Безпека сайтів і веб-додатків. URL: https://bit.ly/3tpMmu4 (дата звернення: 15.11.2022).

9. Популярні методи оптимізації продуктивності сайту. URL: https://bit.ly/3UJX8XM (дата звернення: 15.11.2022).

10. Масштабованість систем операцій на підприємствах. URL: https://bit.ly/3TC88pp (дата звернення: 15.11.2022).

11. Comparison with Other Frameworks – Vue.js. URL: https://bit.ly/3tlKD9d (дата звернення: 15.11.2022).

12. Node.js. URL: https://bit.ly/3Et7jL9 (дата звернення: 15.11.2022).

13. Angular – CLI Overview and Command Reference. URL: https://bit.ly/2WCe7g7 (дата звернення: 15.11.2022).

14. Angular | HttpClient и відправка замовлень. URL: https://bit.ly/3hBLYWP (дата звернення: 15.11.2022).

15. Angular | Сервіси и dependency injection. URL: https://bit.ly/3O2qn60 (дата звернення: 15.11.2022).

16. Директиви | Angular з прикладами коду. URL: https://bit.ly/3TwjLy4 (дата звернення: 15.11.2022).

17. Pipe | Angular з прикладами коду. URL: https://bit.ly/3TDn0U9 (дата звернення: 15.11.2022).

18. Однофайлові компоненти Vue | Vue Loader. URL: https://bit.ly/3GgJJCz (дата звернення: 15.11.2022).

19. Починаючи | Axios Docs. URL: https://bit.ly/3EujmaR (дата звернення: 15.11.2022).

20. Помилки на опублікованих сайтах: Помилка 401 (Unauthorized). URL: https://bit.ly/3EdJ0j8 (дата звернення: 15.11.2022).

21. Методичні вказівки з виконання кваліфікаційної роботи здобувачів вищої освіти на другому (магістерському) рівні для студентів усіх форм навчання спеціальності 186 «Видавництво та поліграфія» / Н.Є. Кулішова, В.П. Ткаченко, І.О. Мілютченко, Б.П. Косіковська. Харків: ХНУРЕ, 2020. 52 с.

22. PhpStorm. URL: https://bit.ly/3whnm7J (дата звернення: 15.11.2022).

23. npm. URL: https://bit.ly/3o7sgkO (дата звернення: 15.11.2022).

24. Класифікація, типи і завдання експерименту — методи теоретичних і експериментальних досліджень. URL: https://bit.ly/3Exibrk (дата звернення: 15.11.2022).

25. Метод експертних оцінок. URL: https://bit.ly/3s1pZL1 (дата звернення: 15.11.2022).

26. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлювання Нац. стандарт України [чинний від 07.01.2016]. Київ : ДП «УкрНДНЦ», 2016. 17 с.