ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Погребенник Юрій Григорович

УДК 004.05

Ефективність застосування різних платформ для створення FRONT END з типовими елементами інтерфейсу в різних браузерах

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації магістра

Нормоконтроль

_____к.т.н. Мануляк І. З.
(підпис, дата, розшифрування підпису)

Студент

_____ Погребенник Ю.Г.
(підпис, дата, розшифрування підпису)

Допускається до захисту
Завідувач кафедри

_____к.т.н. Пашкевич О.П.
(підпис, дата, розшифрування підпису)

Керівник роботи :

_____ к.т.н. Мануляк І.З.
(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2021

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «магістр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

« ___ » _____ 2021 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЕКТ) СТУДЕНТУ

Погребеннику Юрію Григоровичу

(прізвище, ім'я, по батькові)

1. Тема магістерської роботи

Ефективність застосування різних платформ для створення FRONT END з типовими елементами інтерфейсу в різних браузерах

керівник роботи:
Мануляк Ірина Зіновіївна, кандидат технічних наук

затверджена наказом вищого навчального закладу від « 30 » вересня 2021 року № 11/1 НВ

2. Термін подання студентом роботи 03.12.2021

3. Зміст магістерської роботи (перелік питань, які потрібно розробити)

1.Огляд веб-платформ та використаних технологій

2.Розробка структури проекту

3.Розробка програмного забезпечення та аналіз ефективності фронт-енд платформ

4. Дата видачі завдання 22.02.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи (проекту)	Термін виконання етапів роботи	Примітка
1.	Короткий опис використовуваних мов програмування та їх алгоритм роботи.		Виконано
2.	Опис роботи та відмінність браузерів в яких буде проводитись дослідження.		Виконано
4.	Дослідження та структуризація потрібних для аналізу даних.		Виконано
5.	Опис та аналітика даних (графіків та таблиць) зібраних за період створення та дослідження.		Виконано

Студент _____ _____

(підпис) (прізвище та ініціали)

Керівник роботи _____ _____

(підпис) (прізвище та ініціали)

Вихідні дані проекту:

JavaScript, ReactJs, VueJs, AngularJs, HTML5, CSS, Google Chrome, Safari, Firefox, графіки аналізу веб-браузерів, графіки популярності фронт-енд платформ опис та порівняння аналогічних досліджень.

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
1	Тема кваліфікаційної роботи	8	Вигляд веб-сайту
2	Актуальність теми	9	Результати порівняння створення проекту
3	Об'єкт та предмет дослідження	10	Графіки recources та DOMContentLoaded
4	Методи дослідження	11	Графік finish
5	Практична цінність, наукова новизна, апробація	12	Діаграма порівняння Summary
6	Аналогічні дослідження	13	Висновок
7	UML діаграми		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

HTML – HyperText Markup Language;

CSS – Cascading Style Sheets;

JS – JavaScript;

JSS – JavaScript Style Sheet;

NPM – Node Package Manager;

MVC – Model-View-Controller.

IDE – Integrated development environment;

CDN – Content Delivery Networ;

CLI – Command line interface;

DOM – Document Object Model;

LTS – Long-Term Support.

ВСТУП

Актуальність теми. Більше 70% людей світу користуються інтернетом, і хоба б раз користувалась веб-сайтом. І для більшості важливо щоб все що вони хочуть знайти завантажувалось як найшвидше і найефективніше. Тому важливо щоб розробники які розробляють веб-сайти використовували самий ефективний із можливих методів для розробки. Актуальність даної робота є в тому що в ній досліджується декілька фронт-енд платформ для створення веб-сайтів а саме ReactJS, Angular, VueJS, в декількох самих популярних браузерах Google Chrome, Mozila, Firefox. Тобто розробники будуть знати які технології краще використовувати для більшої ефективності їхніх веб-сайтів, що в свою чергу зробить користування веб-сайтами більш зручним.

Мета і завдання дослідження. Відповідно до теми в роботі поставлені такі завдання:

- з’ясування того які із існуючих веб-платформ найкращі, для подальшої роботи з ними;
- розроблення трьох веб-сайтів з типовими елементами інтерфейсу на кожній із запропонованих платформ;
- впевнетись що розроблені веб-додатки коректно відображаюьбся та працюють в різних браузерах;
- визначення необхідних досліджень для порівняння ефективності;
- структурування отриманих даних;
- з’ясування за допомогою отриманих даних та створених діаграм, яка із веб платформ є найефектинішою.

Об’єкт дослідження. є процеси компанування (верстки) веб-сайтів на основі типових елементів інтерфесу для різних браузерів.

Предметом дослідження. методи і засоби різних платформ створення front end веб-сайтів.

Методи дослідження. базуються на на теорії інформації, методах системного аналізу.

Наукова новизна одержаних результатів. В даній роботі вперше одержано фактичне порівняння ефективності фронт-енд платформ в різних браузерах, що суттєво відрізняється від існуючих досліджень, в яких порівняння проводиться абстрактно.

Практичне значення одержаних результатів. графічні та табличні залежності ефективності функціонування розглянутих front end платформ при реалізації веб-сайтів на основі типових елементів інтерфейсу, отримані результати рекомендовані для впровадження у компанії «ШАЗ АЛЬФА ЮЕЙ».

Апробація результатів дослідження. Матеріали кваліфікаційної роботи були представлені в IX Студентському науковому симпозіумі «Співдружність наук: архітектури, економіки, право», який відбувся 19-20 листопада 2021 року в університеті Короля Данила.

Структура. Загальний обсяг сторінок основної частини є 91 сторінка. Список використаних джерел містить 43 позицій.

РОЗДІЛ 1. ОГЛЯД ВЕБ-ПЛАТФОРМ ТА ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

1.1 Вимоги до дослідження ефективності різних платформ для створення фронт-енд з типовими елементами інтерфейсу в різних браузерів

Наукове дослідження – процес дослідження певного об'єкта (предмета або явища) за допомогою наукових методів, що має на меті встановлення закономірностей його виникнення, розвитку і перетворення в інтересах раціонального використання у практичній діяльності людей.

Frontend (фронт-енд) – це частина веб-сайтів, з якою користувач може контактувати і взаємодіяти напряму. У фронт-енд входить відображення функціональних елементів призначених для користувача інтерфейсу, що виконуються в браузері, а також опрацювання запитів користувачів до сервісу.

У свою чергу, web-додаток – додаток, в якому клієнтом виступає в основному браузер. Логіка web-додатку розподілена на дві логічні частини бек-енд і фронт-енд, зберігання даних здійснюється переважно на бек-енді, тобто на сервері, а у мережі відбувається обмін інформацією. Простіше кажучи, це те, що бачить користувач і які дії виконує кожен раз, коли підключається до мережі інтернет і відкриває будь-який браузер [1].

Останні роки фронт-енд розробка стала дуже популярною тому що все більше користувачів використовують веб-додатки, а деякі із веб-додатків не потребують бек-енд частини. Тому саме фронт-енд був обраний для порівняння ефектиності його веб-платформ.

З технологічної точки зору існує чотири тлумачення значення ефективність:

- відношення корисного ефекту (результату) до витрат на його одержання;
- властивість певного процесу, яка зумовлена його якістю та кількістю засобів, що беруть участь у процесі, а також конкретною ситуацією; Ефективність уможливлює виконання певної задачі; характеризується певним співвідношенням між отриманим сумарним ефектом та сумарними витратами на створення і використання засобів, що беруть участь у процесі, його організацію та здійснення;
- у системах обробки інформації – швидкість обробки одиниці інформації, питомі витрати на обробки одиниці інформації;
- (від лат. effectivus – діяльний, творчий) – відносний ефект, результативність процесу, операції, проекту, що визначається як відношення результату до витрат, які зумовили його одержання[2].

Одним із показником ефективності веб-платформи є швидкість загрузки сайту на якій вона написана. Core Web Vitals є чиннами, які Google використовує для оцінки якості ресурсу, їх було обновлено і почато використовувати в 2021р. Ці чинники враховують в тому числі час, що необхідний на очікування першого взаємодії з контентом. І чим більший цей час, тим менше шансів мати гарне ранжування веб-сайту в Google індексації. Це доводять дослідження, які проводились в інтернеті, на основі інформації від декількох тисяч користувачів, що більшість людей залишають сайт, якщо він завантажується довше ніж 3 секунд, і кожна наступна секунда очікування зменшує кількість користувачів приблизно на сім процентів.

Швидкість загрузки сайту впливає на таке:

- індексація сайту, можливо таке що пошукові роботи зовсім не зможуть проіндексувати сайт через те що він загружується занадто довго, тому що в роботів існує ліміт часу який вони будуть чекати загрузку веб-сайту. Це грозить тим що сайт не буде появлятись в пошуку Google або інших пошукових систем, що є великим мінусом для його власників;

- перебування користувачів на сайті тісно пов'язані зі швидкістю завантаження сторінок. Чим більше користувачів іде з сайту до його загрузки, тим нижча статистика сайту, як результат сайт буде з нижчим ранжуванням в пошукових системах;
- конверсія падає, якщо користувач чекає на загрузку сторінки кожний раз довше 5 секунд, це призводить до того що знижуються показники тривалості перебування користувачів на сайті та кількість внутрішніх переходів. Наприклад інтернет-магазин не зможе продавати свої товари ефективно, якщо на ньому просто не буде користувачів.

Елементи графічного інтерфейсу користувача – це ті елементи, якими користується графічні інтерфейси користувача (GUI), щоб запропонувати послідовний візуальна мова представляти інформацію, що зберігається в комп'ютерах. Вони полегшують людям, які володіють не значними навичками роботи з комп’ютером, працювати та використовувати комп’ютерне програмне забезпечення.

Типові елементи інтерфейсу

Стандартними елементами інтерфейсу, вважаються наступні елементи: список (list box), вкладка (tab), кнопка (button), радіокнопка (radio button), прапорець (check box), значок (іконка, icon), дерево – ієрархічний список (tree view), список, що розкривається (combo box, drop-down list), поліредагування (textbox, editfield), елемент для відображення табличних даних (grid view), меню (menu), яке має такі три види,головне меню вікна (main menu), контекстне меню (popup menu), спадаючеменю (pulldownmenu),вікно (window), яке має два підвиди: діалогове вікно (dialog box), модальне вікно (modal window), панель (panel), панель інструментів (toolbar), смуга прокрутки (scrollbar), повзунок (slider),рядок стану (status bar).

На рисунку 1.1 показані деякі із типових елементів інтерфейсу, які будуть викостовуватись в даному проекті, а саме елементи редагування, тобто ті в які користувач може вводити різні дані, з впевненістю можна сказати що ці елементи є на всіх веб-сайтах.

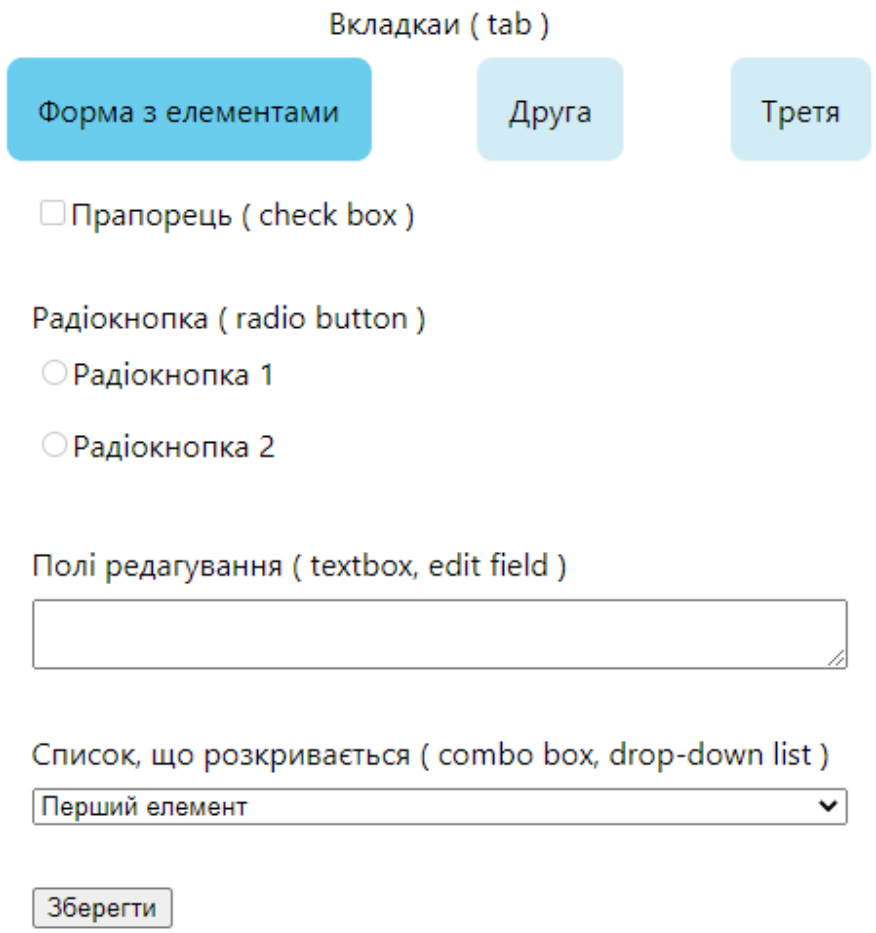

Рисунок 1.1 – Декілька із типових елементів інтерфейсу

Для проведення якісного дослідження необхідно мати великий набір навичок, ось перелік цих навичок:

- вміння проводити аналіз за допомогою існуючої в інтернеті інформації;
- знання мови програмування JavaScript;
- знання фреймворку React;
- знання фреймворку Angular;
- знання фреймфорку VueJs;
- вміння використовувати CSSта JSS;
- використання менеджера пакетів npm;
- знання методів перевірки ефективності веб-платформ;
- вміння користуватись браузерами для аналізу.

1.2 Аналіз існуючих веб-браузерів

Браузер – програмне забезпечення для комп'ютера або іншого електронного пристрою, що дає можливість користувачеві взаємодіяти з текстом, малюнками або іншою інформацією на гіпертекстовій вебсторінці, як правило, під'єднаного до Інтернету. Посилання, які ведуть на інші вебсторінки можуть міститись в текстах та малюнках, перенаправляючи на той же вебсайт або на інший. Вебпереглядач з допомогою гіперпосилань дозволяє користувачеві швидко та просто отримувати інформацію, розміщену на багатьох вебсторінках.

Вебпереглядач приєднується до сервера HTTP, отримує з нього документ і форматує його для представлення користувачеві або намагається викликати зовнішню програму, яка це зробить, залежно від формату документа. Формати документа, які вебпереглядач повинен представляти без допомоги зовнішніх програм, визначає World Wide Web Consortium (скорочено W3C). До них належать формати текстових документів HTML та XHTML, а також найпоширеніші формати растрової графіки GIF, JPEG та PNG (останній –розробка W3C) [3]. Також в наш час браузери підтримують PDF та MP3 без додаткових програм.

Адресування сторінок відбувається за допомогою URL (UniformResourceLocator, RFC 1738), який інтерпретується, як адреса, що починається з http: для протоколу HTTP, який вже рахується не достатньо захищеним, тому замість HTTP вже використовується HTTPS. А всі веб-сайти, які використовують HTTP блокуються новітніми браузерами. Багато навігаторів також підтримують інші типи URL та їх відповідні протоколи, як, наприклад, gopher: для Gopher (ієрархічний протокол гіперпосилань), ftp: для протоколу перенесення файлів FTP, rtsp: для протоколу потоків реального часу RTSP, та https: для HTTPS (HTTPSecure, що розширює HTTP за допомогою SecureSocketsLayerSSL або TransportLayerSecurityTLS).

Найпершим веб переглядачем був Mosaic, розроблений в Національному центрі застосування суперкомп'ютерів (NCSA) Іллінойського університету в Урбана-Шампейн.

Браузери використовують кешування щоб зробити взаємодію між веб-сайтом і користувачем швидшою. Всі дані браузера містяться в проміжному буфері, який ще називається кешування, він забезпечує швидкий доступ до інформації, ймовірність запиту якій понад усе. Витягти всю необхідну інформацію з кеша можна значно швидше, ніж з вихідного сховища, що є важливою перевагою. Але існує значний недолік – розмір. Основою структури є набори записів. Кожна з них асоціюється з певним елементом або блоком даних, які виступають копією того, що є в основній пам'яті.

Але для дослідження потрібно обрати браузери які є найбільш популярними на даний час, тому що ними користується велика кількість людей.

Для знаходження трьох найбільш популярних браузерів було використано веб-сайт [4] тому, що будь-яка стаття в інтернеті з дослідженнями популярності браузерів посилається на цей веб-сайт.

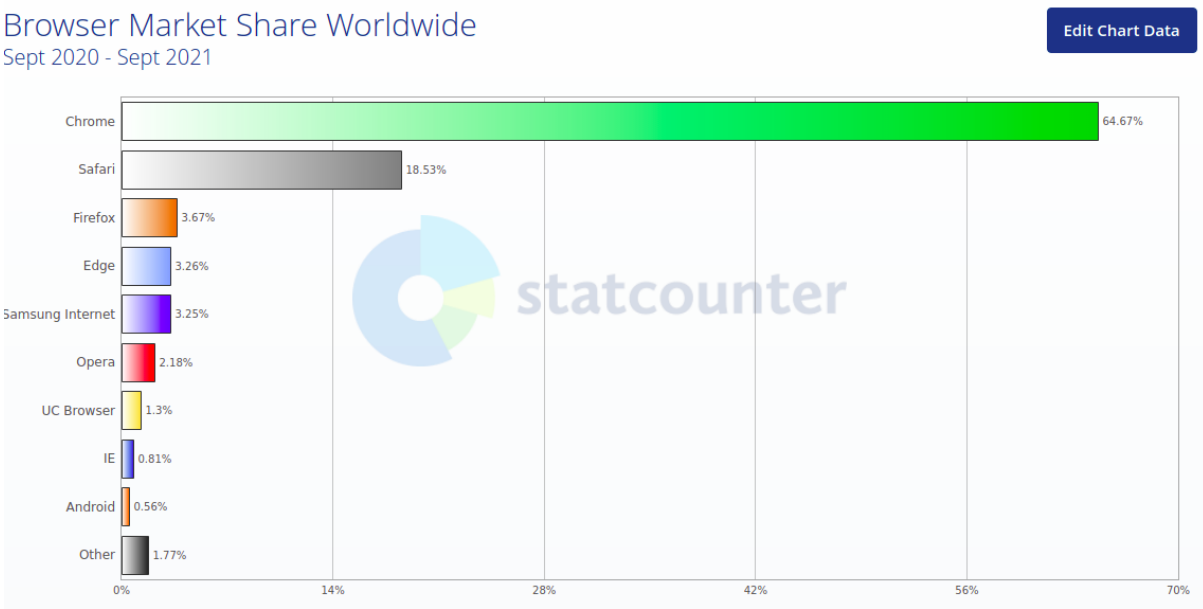

Рисунок 1.2 – Статистика популярності браузерів на період вересень 2021р.

Як можна побачити на рисунку 1.2 на вересень 2021 року[5] найпопулярнішими браузерами на цей час є Chrome, Safari та Firefox, саме тому ми будемо використовувати їх, щоб дане дослідження було корисно як можна більшому числу людей.

1.3 Аналіз веб-платформ

Веб платформа – це сукупність технологій що розроблені як відкриті стандарти Консорціумом Всесвітнього павутиння (W3C) та іншими органами стандартизації: WHATWG, UnicodeConsortium, IETF і EcmaInternational. Це гіперонім, що був введений W3C, і в 2011 році він був визначений CEOW3C Джеффом Яффе як «платформа для інновацій, консолідації та економічної ефективності». Створення стандартів з урахуванням TheevergreenWeb (де мають місце швидкість, автоматизоване оновлення ПЗ, співпраця вендорів, стандартизація і конкуренція) дозволило додати нові можливості при вирішенні ризиків щодо безпеки та конфіденційності [6].

Веб платформа включає в себе технології – комп'ютерні мови та API, які спочатку були створені для публікації веб сторінок.

В ході дослідження буде вибрано три найпопулярніші веб-платформи (мови програмування), для порівняння їхньої ефективності. Для знаходження трьох найпопулярніших веб платформ було використано веб-сайт який показує кількість пошукових запитів в пошуковій системі google а саме: https://trends.google.com.ua/trends. І хоча цей метод не покаже ідеально точний показник популярності, але це найбільш точні дані які можливо знайти в публічному доступі.

В ході проведення роботи досліджено багато статтей з інформацією про найпопулярніші фронт-енд мовами програмування. Було складено такий список найпопулярніших мов програмування React,VueJs, Angular, Ember, Backbone.js, JQuery. На рисунку (рис 1.3) показано порівняння цих всіх мов програмування.

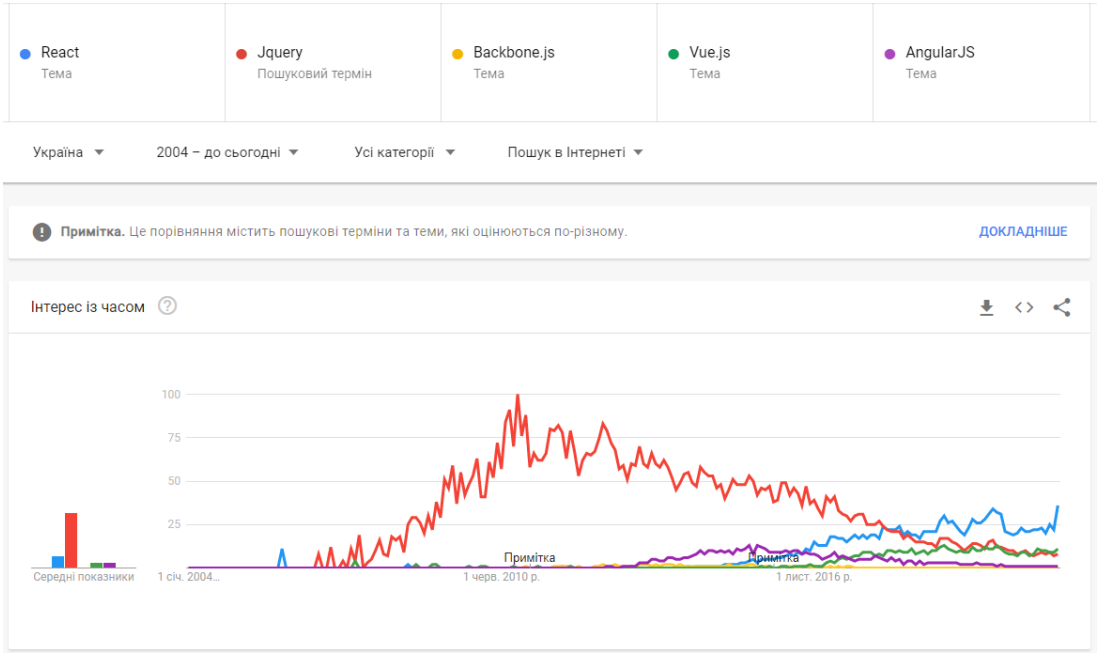

Рисунок 1.3 – Порівняння веб-платформ

Методом порівняння та виключенРисунокня не популярних платформ було виявлено три (рис 1.4) мови які підходять під потреби дослідження, це: React, Angular, Vue.js.

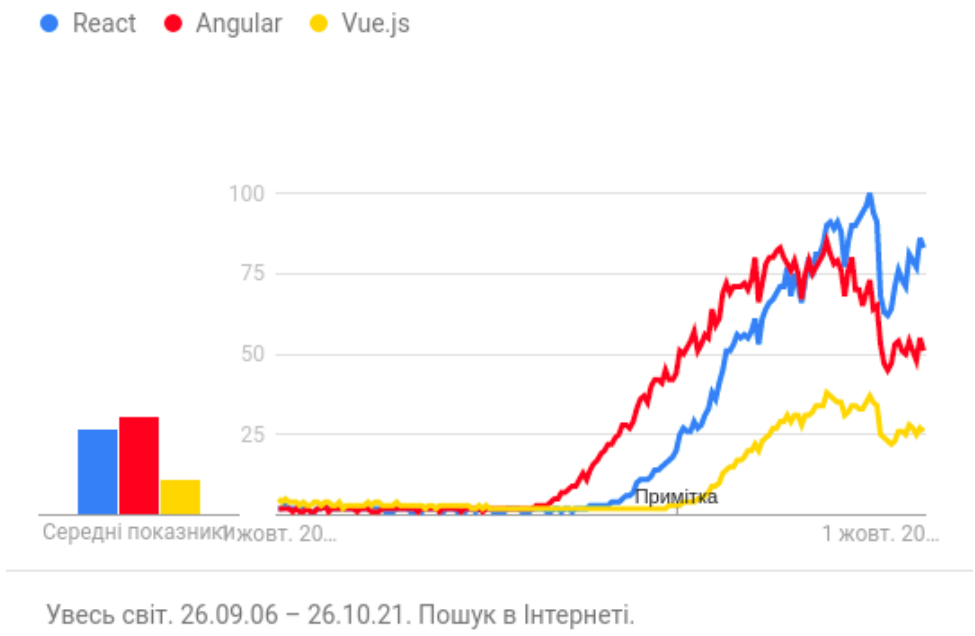

Рисунок 1.4 – Три відповідні нашим критеріям веб-платформи

В подальшому для дослідження буде використовуватись саме ці три веб-платформи, тому що чим більше людей використовують платформу для розробки тим більш стабільною вона являється, і тим більшому числу людей буде корисний даний проект.

1.4 Опис аналогічних досліджень

В підрозділі 1.2 були описані веб-браузери, а в підрозділі 1.3 було описані веб-платформи, які є найбльш популярними в даний час і аналіз яких принесе найбільшу користь для людей, які будуть використовувати дану роботу, але перед початком дослідження потрібно впевнитись, що такого ще немає на просторах інтернету.

Одним із аналогів можна вважати статтю на веб-сайті[7]. В статті описуються плюси та мінуси кожної із запропонованихплатформ, також уточнюється рік створення, поточна версія, ким створена та посилання на веб-сайт самої платформи (рис 1.5).

	Angular	React	Vue
Initial release	2010	2013	2014
Official site	angular.io	reactjs.org	vuejs.org
Current version	11	17.x	3.0.x
Used by	Google, Wix	Facebook, Uber	Alibaba, GitLab

Рисунок 1.5 – Інформація про веб-платформи

Також в статті описано ліцензії та популярність самих платформ, і інформацію про відкриті ваканції на різних сайтах (рис 1.6).

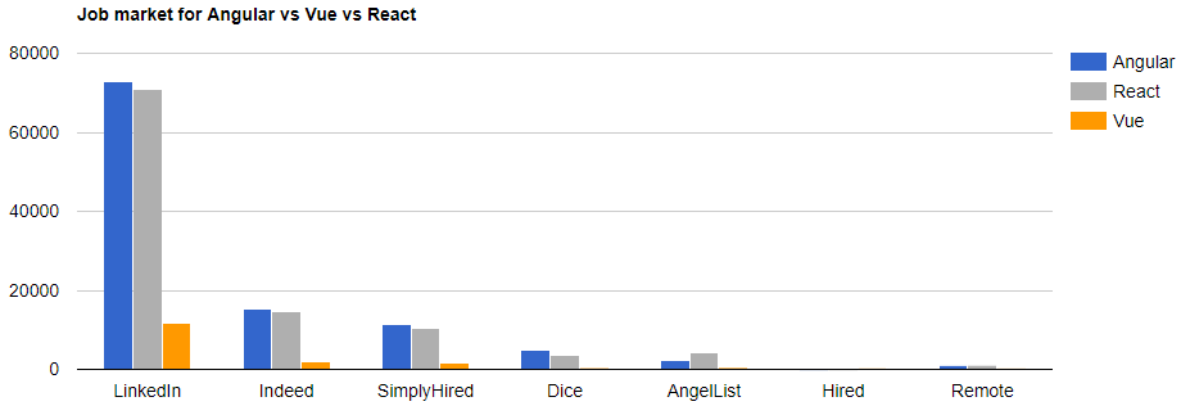

Рисунок 1.6 – Графік з відкритими ваканціями для веб-платформ

Але в статті немає ніякого фактичного порівняння ефективності цих платформ, немає інформації про швидкість завантаження веб-сайту написаного на одній із платформ, тому дане дослідження є більш делатьним та більш ціло-напрямлене на конкретну проблему ніж ця стаття.

Також є дві інші статті [8], [9], які можна віднести до аналогів дослідження.

В цих статтях також описуються веб-платформи, які були обрані для даного дослідження. Для обидвох статтей навіть було проведено опитування між користувачами, які працюють як фронт-енд розробники. Результатом став графік, який зображений на рисунку 3, на якому можна побачити, що Reactта VueJsє першими, а ось Angular вже відстає від них та знаходиться внизу. Але головним мінусом цих статей те що одна була написана в 2020 році, а інша в 2019 році, що насправді для веб-платформ вже може бути застарілою інформацією, так же само як і минула стаття. В даних статтях немає дослідження ефективності цих платформ в браузерах. Тому дослідження, яке було проведено є кращим за існуючий аналог.

На рисунку 1.7 зображений графік най популярніших веб-платформ, за версією цих статтей, за даними якої, трьома найпопулярнішими платформами є React, VueJs та Express. А ось Angular є дев’ятим, але всі платформи які стоять вище не відносяться до фронт-енд платформ, тому не приймають участь в даному дослідженні.

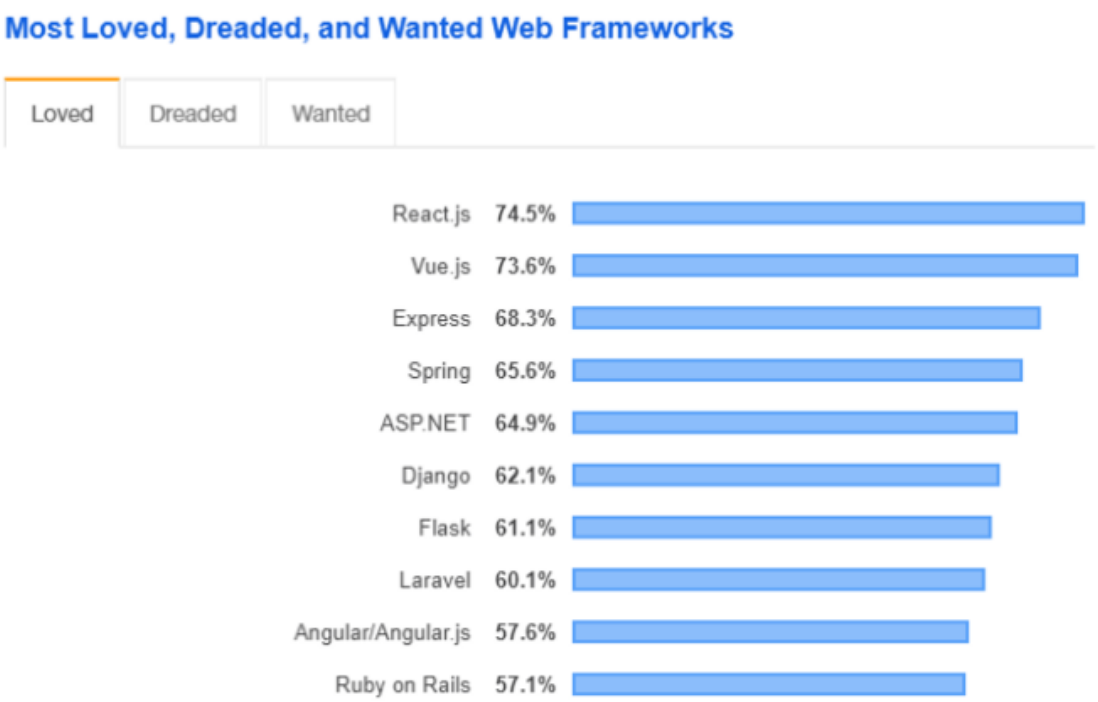

Рисунок 1.7 – Най більш відоміші веб-платформи

Можна знайти ще декілька статтей в який були проведені дослідження веб-платформ які були описані в розділі 1.3, але у всіх із них є свої мінуси, хтось не порівнює ефективність, а в більшості із них дослідження ведеться тільки двох веб-платформ. Тому даний проект створюється для того, щоб покрити мінуси інших статей, та дослідити ефективність усіх трьох веб-платформ і порівняти їх у браузерах.

1.5 Технології проекту

Виходячи з аналізу, який був проведений в пункті 1.3 для дослідження буде використано такі веб-платформи, React, Angular, Vue.js.

Всі вони мають в своїй основі мову програмування JavaScript та деякі спільні технології потрібні для коректної роботи веб-платформи.

Спільне в трьох вибраних веб-платформах:

- JavaScript;
- npm;

- Jss & SCSS.

JavaScript спочатку був створений для того, щоб «оживляти веб-сторінки».

Програми на цій мові називаються сценаріями. Вони можуть бути написані прямо в HTML-форматі веб-сторінки і запускатися автоматично при завантаженні сторінки.

Сценарії надаються і виконуються у вигляді звичайного тексту. Для їх запуску не потрібно спеціальної підготовки або компіляції.

Коли був створений JavaScript, він спочатку мав іншу назву: «LiveScript». Але Java в той час була дуже популярною, тому було вирішено, що позиціонування нової мови як «молодшого брата» Java допоможе.

Але в міру свого розвитку JavaScript став повністю незалежною мовою зі своєю власною специфікацією під назвою ECMAScript, і тепер він взагалі не має ніякого відношення до Java.

Сьогодні JavaScript може виконуватися не тільки в браузері, але і на сервері або фактично на будь-якому пристрої, на якому є спеціальна програма, звана движком JavaScript.

Браузер має вбудований движок, який іноді називають «віртуальною машиною JavaScript».

Мова JavaScript використовується для:

- написання сценаріїв веб-сторінок для надання їм інтерактивності;
- створення односторінкових веб-застосунків (React, AngularJS, Vue.js);
- програмування на стороні сервера (Node.js);
- стаціонарних застосунків (Electron, NW.js);
- мобільнихзастосунків (React Native, Cordova);
- сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);
- всередині PDF-документів тощо.

JavaScript має низку властивостей об'єктно-орієнтованої мови, але завдяки концепції прототипів підтримка об'єктів в ній відрізняється від традиційних мов ООП. Крім того, JavaScript має ряд властивостей, притаманних функціональним мовам, – функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання (closures) – що додає мові додаткову гнучкість.

JavaScript має С-подібний синтаксис, але в порівнянні з мовою С має такі корінні відмінності:

- об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів;
- функції як об'єкти першого классу;
- опрацювання винятків;
- автоматичне приведення типів;
- автоматичне збирання сміття;
- стрілочна функція;
- анонімні функції.

JavaScript містить декілька вбудованих об'єктів: Date, Global, Object, Function, Number, Array, String, Boolean, Math, Error, RegExp. Крім того, JavaScript містить набір вбудованих операторів, що управляють логікою виконання програм, та набір операцій, які, грубо кажучи, не обов'язково є функціями або методами. Синтаксис JavaScript в основному відповідає синтаксису мови Java (тобто, зрештою, успадкований від С), але спрощений порівняно з ним, щоб зробити мову сценаріїв легкою для вивчення. Так, приміром, декларація функції може стояти в тексті програми після неї, а декларація змінної не містить її типу, властивості також не мають типів.

Зараз все популярнішим стає нова версія JavaScript, яка має назву ECMAScript 6 (ES6). Одним із переваг ES6 є поява класів, які використовуються як логічна одиниця програми, оголошення та використання методів в клісі можна побачити на рисунку 1.8. а саме створення класу MyClass та оголошення в ньому двох змінних значення, яких дальше

використовується для обчислення. Також на рисунку є новий функціонал ES6,а саме const, який разом із let замінили собою привичне для JavaScript оголошення змінної var.

Саме тому що JavaScript є таким популярним він використовується для всіх платформ, які були обрані для дослідження.

```
class MyClass {
   constructor () {
   this.myValue1 = 1;
   this.myValue2 = 2;
   }
}

const mc = new MyClass();
mc.myValue1 = mc.myValue2 * 2;
```

Рисунок 1.8 – Оголошення і використання класу в JavaScript

npm-це дві речі: перш за все, це онлайн-сховище для публікації відкритих вихідних тексту Node.JS проектів; по-друге, це утиліта командного рядка для взаємодії з зазначеним репозиторієм, яка допомагає в установці пакетів, управлінні версіями і управлінні залежностями. Безліч Node.JS бібліотеки та Додатки публікуються в npm, і кожен день додається ще більше. Якщо є пакет, який розробник хоче встановити, його можна встановити за допомогою однієї команди командного рядка.

npm може управляти пакунками, які є локальними залежностями певного проекту, а також глобально інстальованими інструментами JavaScript. При використанні npm як менеджера залежності для локального проекту, можна встановити одною командою всі залежності проекту через файл package.json. У файлі package.json кожна залежність може визначати діапазон дійсних версій, використовуючи схему семантичної версії, що дозволяє розробникам автоматично оновлювати свої пакети, одночасно уникаючи небажаних змін [10].

На даних моментпрmє найбільш популярним та найбільш зрозумілим у використанні менеджером пакунків в світі. І більшість проектів, в яких

потрібні додаткові бібліотеки для JavaScript, використовують npm, так само як і для створення проекту на мові React-Native потрібний npm, який створить початкові налаштування додатку, щоб його можна було запустити. Саме тому в проекті був використаний npm.

JSS (бібліотека) – стилизація css на мові Javascript в декларативному стилі. JSS не компілюється в inline-style і може повторно перевикористовуватись, уникаючи конфліктів, за рахунок того що генерує унікальні імена класів. Має можливість працювати посередництвом server-side в nodejs і має розширення [.js]. В jss немає обмежень на установку стилів для псевдо-селекторів чи псевдо-елементів і подібного. Описується jss в стилі camelCase (рис.1.9). Одною із ключових можливостей jss – можливість архітектурного підходу в описі стилів [11].

```
const styles = {
    backgroundColor: "grey",
    border: 0,
    borderRadius: 3,
    boxShadow: "0 3px 5px 2px",
    color: "black",
    height: 48,
    padding: "0 30px",
    margin: 10
  };
```

Рисунок 1.9 – Приклад Jss константи в коді програми

Sass (англ. Syntactically Awesome Stylesheets) – скриптова метамова, яка інтерпретується в каскадні таблиці стилів (CSS). Sass призначений для підвищення рівня абстракції коду та спрощення файлів CSS. Спроектована Гемптоном Кетліном та розроблена Наталі Вейзенбаум.

Мова Sass має два синтаксиси:

- sass (оригінальний) – дуже чутливий до структури написання, тому що повністю відсутні фігурні дужки, кожний вкладений елемент реалізований

за допомогою відступів, а саме пробілів, а правила відокремлюються переведенням рядка;

- scss (новий) – подібний до CSS, вкладені елементи реалізовані за допомогою фігурних дужок.

Файли sass-синтаксису мають розширення .sass, scss-синтаксису – .scss.

Sass розширює CSS, надаючи кілька механізмів, доступних в більш традиційних мовах програмування, зокрема об'єктно-орієнтованих мовах, але недоступних для CSS. Інтерпретатор Sass транслює SassScript у блоки правил CSS. По суті, Sass – це синтаксичний цукор для CSS.

Sass та Scss дозволяє визначати змінні. Змінні починаються зі знака долара ($). Присвоєння значень змінних здійснюється за допомогою двокрапки (:).

SassScript підтримує чотири типи даних:

- логічний (булевий) тип;
- рядок (з лапками чи без);
- число;
- колір (ім'я або імена).

Змінна може бути аргументом чи результатом однієї чи кількох функцій. Під час трансляції значення змінних вставляються у вихідний (тобто результуючий) документ CSS [12].

На проекті ми використовуємо саме SCSS через його зрозумілішу структуру від SASS чи CSS, оскільки для даного проекту було використано менеджера пакетів npm то із налаштуванням компіляції SCSS в код CSS не виникає ніяких неочікуваних проблем, тому що npm дозволяє це зробити за допомогою додаткового пакету node-scss.

React-найпопулярніша інтерфейсна бібліотека JavaScript в області веб-розробки. Він використовується як великими, відомими компаніями, так і новими стартапами (Netflix, Airbnb, Instagram та New York Times, і це лише деякі з них).

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків [13].

Основною новинкою в React та причиною чому він став такий популярний це Віртуальний DOM. React підтримує віртуальний DOM, а не покладається виключно на DOM браузера. Це дозволяє бібліотеці визначити, які частини DOM змінилися, порівняно (diff) зі збереженою версією віртуального DOM, і таким чином визначити, як найефективніше оновити DOM браузера. Таким чином програміст працює зі сторінкою, але бібліотека самостійно вирішує які компоненти сторінки треба оновити.

Компоненти React зазвичай написані на JSX (рис. 1.10). Код написаний на JSX компілюється у виклики методів бібліотеки React. Розробники можуть так само писати на чистому JavaScript. JSX нагадує іншу мову, яку створили у компанії Фейсбук для розширення PHP, XHP.

```
class App extends React.Component {
  render() {
    const i = 1;
    return (
      <div>
        <h1>{ i === 1 ? 'true' : 'false' }</h1>
      </div>
    );
  }
}
```

Рисунок1.10 – Код написаний на JSX

Angular-це веб-платформа розробки, побудована на TypeScript. В якості платформи Angular включає в себе:

- платформа на основі компонентів для створення масштабованих веб-додатків;
- колекція добре інтегрованих бібліотек, які охоплюють широкий спектр функцій, включаючи маршрутизацію, управління формами, зв'язок клієнт-сервер і багато іншого;
- набір інструментів розробника, які допоможуть розробляти, створювати, тестувати і оновлювати ваш код.

Коли створюється додатки за допомогою Angular, використовується переваги платформи, яка може масштабуватися від проектів одного розробника до додатків корпоративного рівня. Angular розроблений для того, щоб максимально спростити оновлення, щоб ви могли скористатися останніми розробками з мінімальними зусиллями. Найкраще те, що екосистема Angular складається з різноманітної групи з більш ніж 1,7 мільйона розробників, авторів бібліотек і творців контенту.

Angular – написаний на TypeScriptfront-end фреймворк з відкритим кодом, який розробляється під керівництвом AngularTeam у компанії Google, а також спільнотою приватних розробників та корпорацій. Angular– це AngularJS, який був переосмислений та перероблений тією ж командою розробників. [14]

Такий підхід потенційно може сповільнити рендеринг, коли AngularJS перевіряє надто багато змінних в $scopes на кожній ітерації циклу.

Програмне маніпулювання DOM: Декларативно описуючи як повинен змінюватися UI в залежності від зміни стану програми, ви звільняєтесь від низькорівневих маніпуляцій з DOM. Більшості застосунків, написаних за допомогою Angular, ніколи не доведеться програмно маніпулювати DOM, хоча ви це можете це робити якщо захочете.

Починаючи з 9 версії Angular всі нові програми використовують компілятор Ivy. Тож команда Angular працюватиме над покращенням цього

компілятора, що в свою чергу має призвести до зменшення загального розміру пакунка. Очікується, що кожна наступна версія буде зворотно сумісною з попередньою. Google також обіцяє публікувати оновлення двічі на рік.

Vue – це прогресивна платформа для створення користувацьких інтерфейсів. Він розроблений з нуля і може легко стати бібліотекою та основою залежно від ваших цілей. Він складається з основної бібліотеки, яка фокусується на рівні представлення, і екосистеми допоміжних бібліотек.

Vue використовував кращі функції інших бібліотек, такі як двостороння прив'язка даних і директиви від Angular, реалізація віртуального DOM, синтаксис шаблонів від React. Крім того, налаштування досить просте.

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.

Vue сам по собі не включає роутингу, та є vue-router пакет, який вирішує це питання. Він підтримує зв'язування вкладенних шляхів з вкладеними компонентами і пропонує деталізований контроль над переходами. Vue дозволяє створення додатків за допомогою компонетів. Якщо додати vue-router до цього, все що потрібно зробити це зв'язати ваші компоненти з роутами і дозвольте vue-router вирішувати де їх рендерити.

1.6 Постановка задачі

Дослідження показують що кожна друга, а то і більше людей в світі користується інтернетом. Кожний користується інтернетом по своєму, хтось шукає інформацію потрібну для роботи чи навчання, хтось для соціальних

мереж, а хтось просто для того, щоб переглядати смішні картинки чи відео, але щоб вони не робили в інтернеті – вони використовують для цього браузери та веб-сайти.

Дане дослідження спрямоване на те, щоб проаналізувати існуючі веб-платворми та за допомогою проаналізованих даних визначити найефективнішу платформу, яка буде швидкою для розробників при написанні та для користувачів при користуванні

Для точності аналіз буде проводитись в трьох різних браузерах та в їх мобільних варіантах та на сайтах які проводять оцінку існуючих веб-сайтів.

Головна мета цього дослідження – це пошук найефективнішої веб-платформи, щоб програмісти, які планують розробляти веб-сайти могли вибрати її для своєї роботи.

Основні функціональні можливості проектів, які будуть використані для порівняння ефективності веб-платформ:

- перехід між сторінками за допомогою «Меню»;
- прокрутка зображень в повзунку «Слайдер»;
- зміна відображеної інформації за допомогою «Вкладок»;
- можливість використання полів редагування які відносяться до типових елементів інтерфейсу;
- відкриття модального вікна.

І хоча функціональність проекту може здатись простою на перший погляд, насправді всі веб-сайти складаються із цих елементів, но на кожному із сайтів ці елементи відображені по своєму та в різній кількості. Для дослідження нам буде вистачати фенкціоналу, який описаний в списку вище.

Детальний опис веб-додатку і етапівдослідження буде наведено в наступних розділах.

Завданнями кваліфікаційної роботи є:

- розробити три ідентичні веб-додатки, на різних веб-платформах для точності порівняння;
- використати типові елементи інтерфейсу;

- виставити розроблені додатки на хост-сервер, для можливості знайти їх онлайн інструментами провірки;
- проаналізувати дані отримані за допомогою інструментів;
- проаналізувати дані отримані в різних браузерах при дослідженні;
- за допомогою проаналізованих даних знайти найефективнішу веб-платформу.

Висновок до розділу 1

В розділі 1 було описану інформацію потрібну для коректного розуміння проекту та опис вимог до дослідження. Було проведено аналіз існуючих веб-браузерів та веб-платформ, описано їхні плюси та мінуси, які можна знайти в інтернеті про дані платформи та браузери. Проаналізовано причину по якій дана кваліфікаційна робота є актуальною та унікальною на просторах інтернету.

Описані технології, які будуть використовуватись при розробці проекту. Також була поставлена задача до кваліфікаційної роботи.

РОЗДІЛ 2. РОЗРОБКА СТРУКТУРИ ПРОЕКТУ

2.1 Unified Modeling Language діаграми

Для розробки проекту на даних платформах було створено UML діаграми, щоб зрозуміти структуру та всі можливі випадки використання проекту, та для можливості створення ідентичних проектів на всіх трьох платформах.

UML дозволяє розробникам програмного забезпечення досягти угоди в графічних позначеннях для представлення загальних понять, таких як клас, компонент, узагальнення (generalization), об'єднання (aggregation) і поведінка і більше сконцентруватися на проектуванні й архітектурі.

В UML використовується 6 видів структурних діаграм:

- класів;
- компонентів;
- розгортування;
- об’єктів;
- пакетів;

Та 3 діаграми поведінки:

- діяльності;
- станів;
- прецедентів.

Для даного проекту не можливо створити діаграму класів, тому що дані в ньому зберігаються як звичайний JavaScript об’єкт, та не має серверної частини, якій могли б бути потрібні структуровані дані.

Також для даного проекту не має сенсу створювати діаграми об’єктів та пакетів, по тій же причині що описана вище, всі дані зберігаються як звичайний JavaScript об'єкт, і не можуть бути належним чином описані за допомогою цих діаграм.

2.1.1 Розробка діаграми компонентів

На рисунку 2.1 зображено діаграму компонентів створену для даного проекту, але перед початком її аналізу потрібно детально розглянути що ж таке діаграма компонентів.

Діаграма компонентів – діаграма в UML, на ній відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент показує розбиття програмної системи на структурні компоненти та зв'язки (залежності) між компонентами. Як фізичний компонент можуть виступати файли, бібліотеки, модулі, виконувані файли, пакети і т. п. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються, разом використовуючи структурні зв'язки англ. assembly connector, щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

В більшості веб-сайтів використовується клієнт-сервер архітектура, але в нашому випадку для дослідження потрібна тільки клієнтська частина цієї архітектури.

Client – це клієнтська частина додатку, в нашому випадку це веб-дотаток, який створений за допомогою типових елементів інтерфейсу. Цю частину архітектури можливо описати такою діаграмою компонентів (рис 2.1).

В ході написання проекту створено таку діаграму компонентів, на якій можна побачити всі компоненти проекту, та зв’язки між ними.

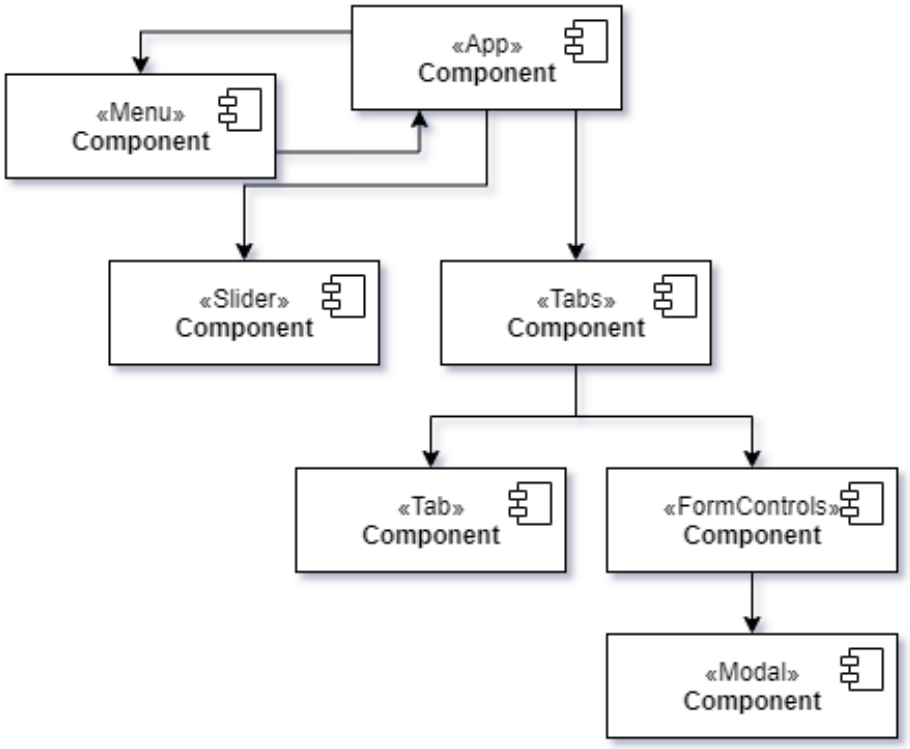

Рисунок 2.1 – Діаграма компонентів для даного проекту

Більш детальна інформація про зображену на рисунку 2.1 діаграму:

App component – є головною компонентою в системі і відповідає за відмалювання всіх інших компонентів, він зв’язаний з компонентом menuдвохстороннім зв’язком, причина в цьому те що в аррзнаходиться інформація про відкриту в данний час сторінку, а в menuзнаходиться контролер, який дозволяє змінювати цю сторінку.

В залежності від того який компонент був вибраний в Menu на сторінці користувач бачить або Slider або Tabs компоненти.

Slider – компонент із слайдером, складається із зображень та кнопок за допомогою яких можна прокручувати показані на екрані зображення.

Tabs – компонент, який позвоняє вибрати показану на екрані інформацію, чимось нагадує Menu компонент, але змінює тільки маленьку частину сторінки а не цілу сторінку.

В залужності від того яка таба вибрана в Tabs компоненті в під вкладці може бути відмальовано Tab або FormControls.

Tab – є під-табою із інформацією.

FormControl – є табою, в якій знаходиться форма з типічними елементами, які позволяють користувачу вводити дані, а саме: прапорець, радіо кнопка, полі редагування, список, що розкривається, кнопка.

Modal – є компонентом, який підмальовується на сторінці тільки в тому разі якщо користувач натисне кнопку «Зберегти» в компоненті FormControl, він показує дані користувачу, які були введені в форму на попередньому кроці.

В кожній із фронт-енд платформ є своє визначення компонента як такого, але всі вони є майже ідентичними.

React component – складова React програми. Ці компоненти дозволяють розділити інтерфейс користувача на незалежні частини, придатні до повторного використання, і сприймати їх як такі, що функціонують окремо один від одного. В розробці проекту було використано два типи react компонентів: це функціональні (рис. 2.2) та класові компоненти. Із появою хуків в React все більш популярними стаюсь функціональні компоненти повність замінюючи собою класові компоненти.

```
const ModalWindow = ({ isModalOpened, setIsModalOpened, data }) => (
  <div className={'modal-wrapper'} style={{ display: isModalOpened ? 'flex' : 'none' }}>
    <div className={'modal-block'}>
      <div className={'modal-header'}>{texts.modalLabel}</div>
      <div className={'modal-content'}>{JSON.stringify(data)}</div>
    </div>
    <div
      className={'backdrop'}
      onClick={() => {
        setIsModalOpened(false)
      }}
    />
  </div>
);
```

Рисунок 2.2 – Функціональний компонент проекту

2.1.2 Розробка діаграми розгортання

Діаграма розгортання була створена для того щоб зрозуміти як саме дані передаються в проекті, та для того щоб уніфікувати проекти на всіх веб-платформах.

Діаграма розгортання (англ. deploymentdiagram) –діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент[15].

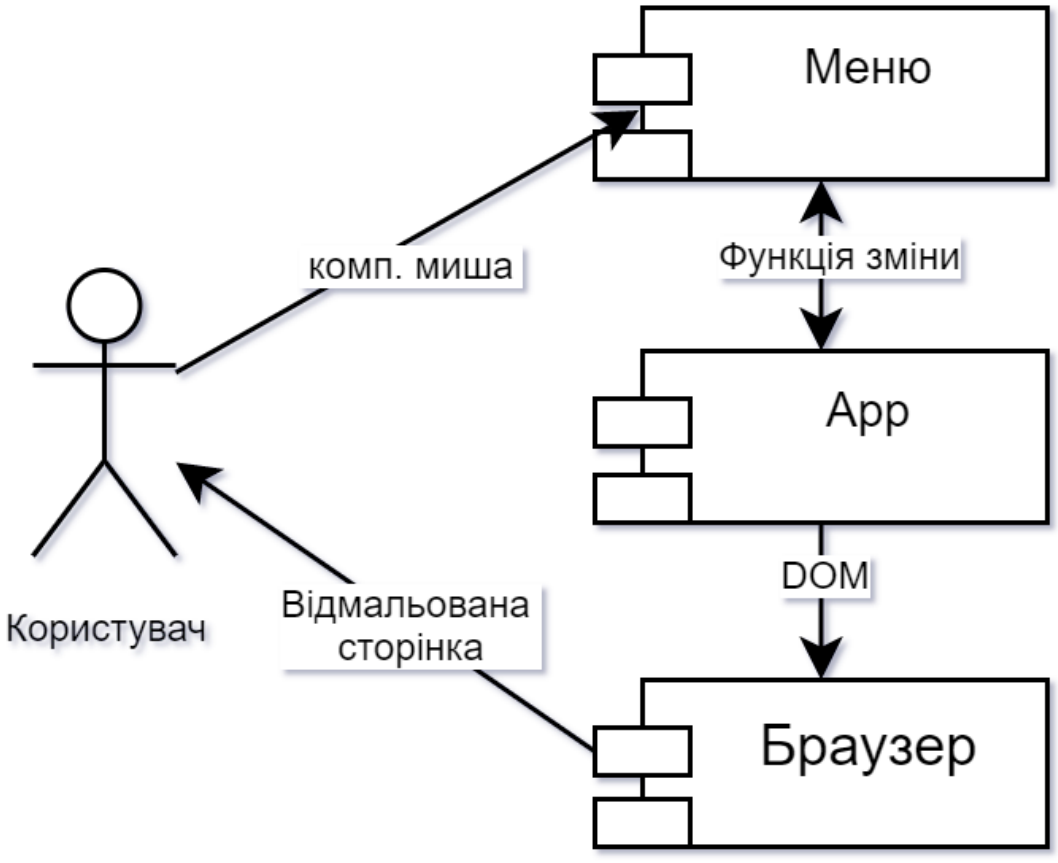

Рисунок 2.3 – Діаграма розгортання зміни сторінки

За допомогою діаграми розгортання можна описати обчислювальні вузли під час роботи програми, на рисунку 1 зображено якраз така діаграма, яка описує зміну відображеної сторінки.

Користувач за допомогою комп’ютерної мишки може вибрати із меню можливий варіант відображення сторінки. Після того як він вибрав і натиснув, дані про вибраний елемент записується в App компонент за допомогою функції зміни, ця функція є зв’язком між App та Меню, в залежності від даних відбувається обробка DOM обєктів, і в кожній веб-платформі це відбувається по своєму. Оброблений DOM відправляється в браузер, а браузер в свою чергу відмальовує по DOMу потрібну сторінку і показує її користувачеві.

І хоча для кожної веб-платформи ця діаграма є такою ж самою, обробка самої інформації в кожній різна, виходячи з цього час який тратиться на зміну сторінки може відрізнятись, якщо це проста сторінка то на не замітні мілісекунди, але чим складніші сторінки тим більша може бути різниця в часі відображення. Для даної роботи важливо який час витрачається при зміні сторінки тому що це також є однією із складових ефектиності веб-платформи

2.1.3 Розробка діаграми станів

На рисунку 2.4 зображено діаграму станів даного проекту, вона створена для того щоб побачити всю поведінку програми та всі можливі стани в яких вона знаходиться.

Діаграма станів – діаграма, що визначає зміну станів об'єкту у часі, одна з діаграм моделювання поведінки в UML. Подає об'єкт як автомат з теорії автоматів зі стандартизованими умовними позначеннями.

Елементами діаграми є:

- коло, що позначає початковий стан;
- коло з маленьким колом усередині, що позначає кінцевий стан (якщо є);

– округлений прямокутник, що позначає окремий стан. Верхівка прямокутника містить назву стану, в середині може бути горизонтальна лінія, під якою записуються активності, що відбуваються в даному стані;

– стрілка, що позначає перехід. Назва події (якщо є), що викликає перехід, відзначається над/під стрілкою. Вартовий вираз може бути доданий перед «/» і укладений у квадратні дужки (назва_події), він означає, що перехід відбувається лише за умови істинності виразу. Якщо при переході відбувається якась активність, то воно додається після «/» (назва події);

– товста горизонтальна лінія, яка є точкою об'єднання або розгалуження переходів[16].

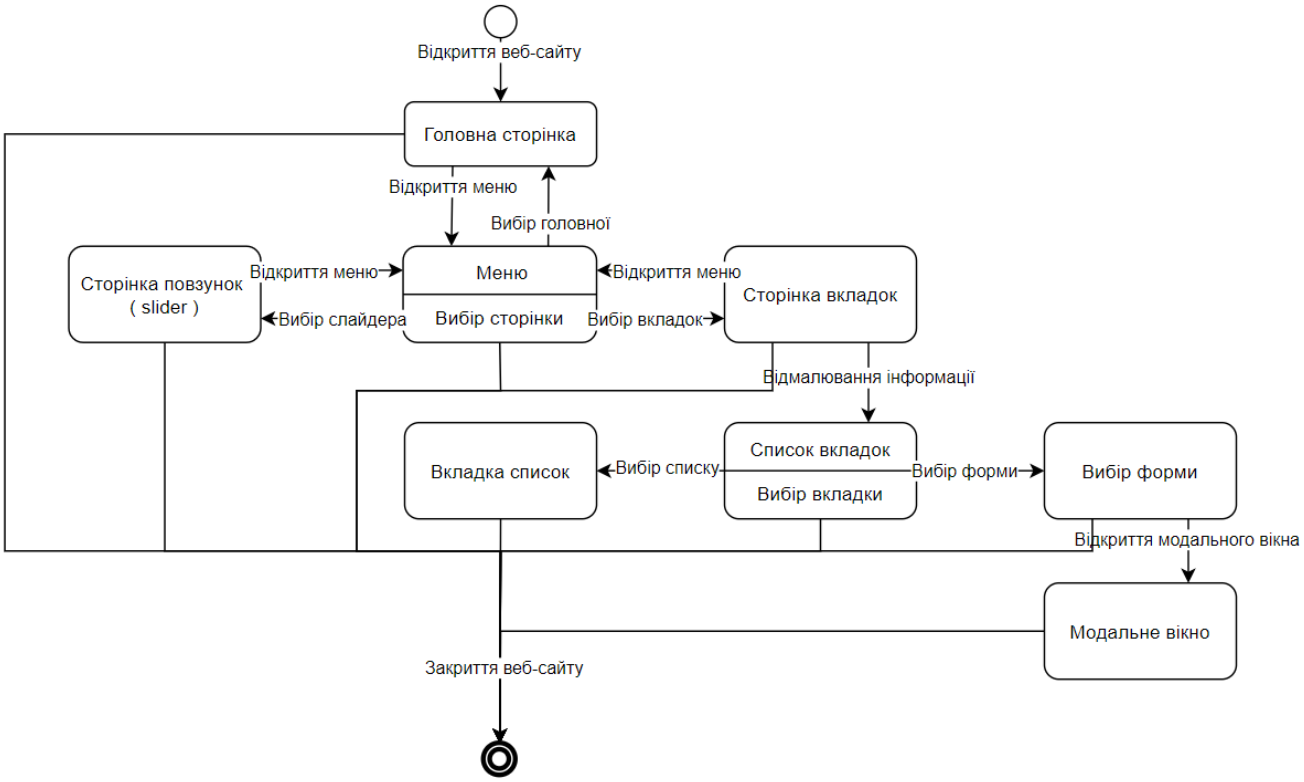

Рисунок 2.4 – Діаграма станів веб-сайту

На рисунку 2.4 зображено діаграму станів, яка була створена в ході виконання дослідження. Стартовою точною користування сайтом є саме його відкриття, після чого користувач потрапляє на «Головну» сторінку, на якій в нього є можливість відкрити меню із вибором сторінки. В меню є три

варіанти вибору це «головна», «повзунок» та «вкладки», переходячи на кожну із цих сторінок користувач все ще може відкрити меню, що явно вказано на діаграмі станів. Відкривши сторінку «вкладки» користувач бачить список із вкладками, де він може вибрати із двох існуючих вкладок «список» та «форма». У вкладці форма розташоване також модальне вікно, яке користувач може відкрити знаходячись тільки у цій вкладці. Функціонал проекту ніяк не зупиняє користувача від того щоб він міг покинути веб-сайт, тому покинути його він може в любий час в не залежності від того в якому стані знаходиться веб-сайт.

2.1.4 Розробка діаграми прецедентів

Діаграма прецедентів є важливою частиною розробки проекту, тому вона була створена та зображена на рисунку 2.5, де можна побачити всі сутності та всіх варіантів використання проекту. Для кращого розуміння що саме зображено на рисунку було додано інформацію про цю діаграму.

Діаграма прецедентів – це поведінковий тип діаграми UML, часто використовуваний для аналізу різних систем. Вони дозволяють візуалізувати різні типи ролей в системі і те, як ці ролі взаємодіють з системою.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (англ. use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором. При цьому

нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

- асоціації (англ. association relationship);
- включення (англ. include relationship);
- розширення (англ. extend relationship);
- узагальнення (англ. generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відносин включення, розширення й узагальнення.

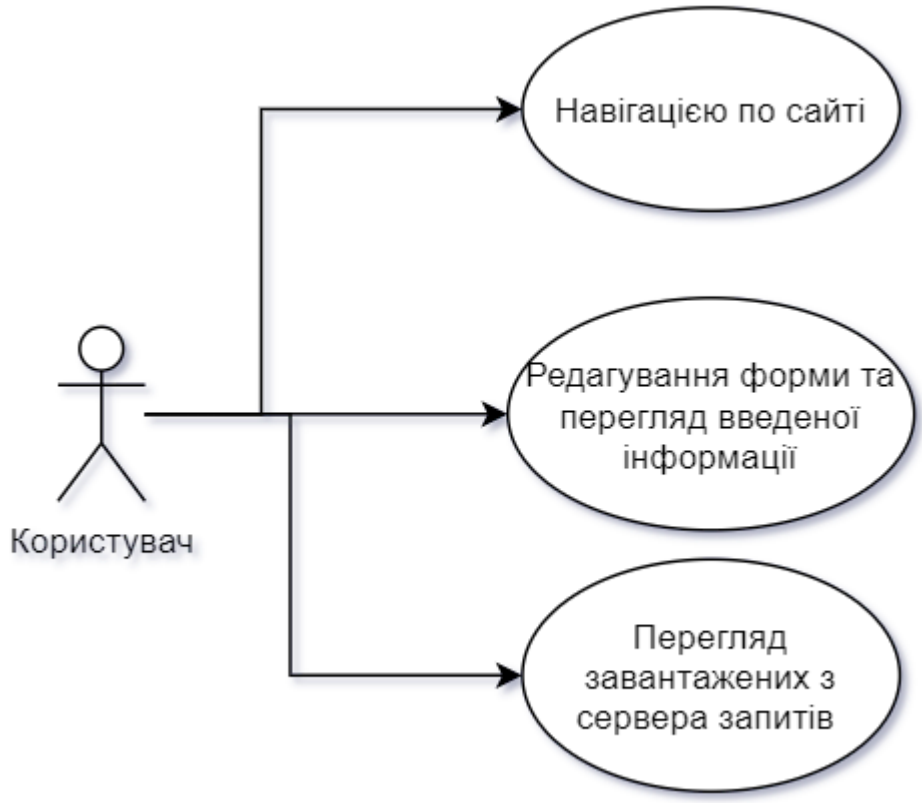

Рисунок 2.5 – Діаграма прецедентів

На рисунку 2.5 зображена діаграма прецедентів, яка показує відношення між акторами та функціями на веб-сайті. На ньому ми можемо бачити що існує тільки один актор «користувач», тобто всі люди які будуть

користуватись веб-сайтом мають доступ до одного і того ж функціоналу. Для аналізу самим головним функціоналом є «Перегляд завантажених з сервера запитів» тому, що за допомогою нього ми можемо бачити що саме потрібно для коректної роботи веб-платформи та час за який ці необхідні елементи завантажуються.

2.2 Model-View-Controller архітектура

Архітектура проекту дуже важлива при розробці проекту, тому було досліджено що на даний момент найбільш підходящою для проекту– є модель–вигляд–контролер.

Модель–вигляд–контролер (або Модель–представлення–контролер, англ. Model-view-controller, MVC) – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення (рис. 2.6).

Мета шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку, розділяючи його на три логічні частини проекту.

Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.

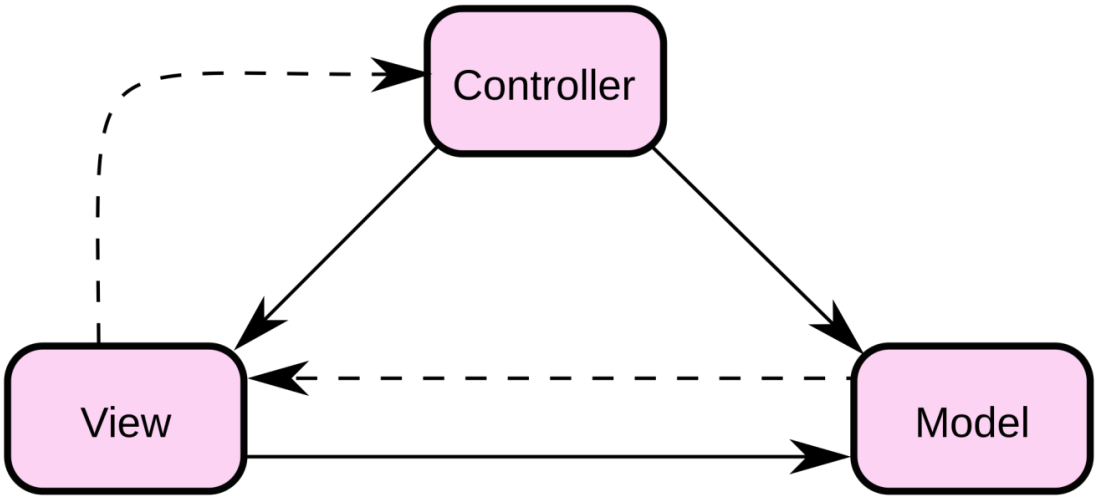

Рисунок 2.6 – Діаграма взаємодії між компонентами шаблону MVC

Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду, здебільшого контролер обробляє дані, у великих проектах він представлений як сервіси, які маніпулюють даними, та відправляє чи отримує їх з сервера, а в даній роботі вн реалізований для збереження даних про відкриту сторінку чи вкладку.

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через контролер внесе зміни до моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися [17].

В даному проекті частина model представлена як звичайні константи, та використовується для збереження всіх елементів меню (рис 2.7), та табів.

```
const menuItems = [
  { name: texts.sliderItem, value: 'slider', icon: home },
  { name: texts.tabsItem, value: 'tabs' }
]
```

Рисунок 2.7 – Вигляд model, список елементів меню

View модель виводить дані в браузер для перегляду його користувачум, прикладом є відмалювання одного із типових елементів інтерфейсу, модального вікна (рис 2.8).

```
const ModalWindow = ({ isModalOpened, setIsModalOpened, data }) => (
  <div className={'modal-wrapper'} style={{ display: isModalOpened ? 'flex' : 'none' }}>
    <div className={'modal-block'}>
      <div className={'modal-header'}>{texts.modalLabel}</div>
      <div className={'modal-content'}>{JSON.stringify(data)}</div>
    </div>
    <div
      className={'backdrop'}
      onClick={() => {
        setIsModalOpened(false)
      }}
    />
  </div>
);
```

Рисунок 2.8 – Вигляд «View» частини даного веб-сайту

Частиною Controller в даному додатку виступають контролери, за допомогою яких відбувається зміна інформації на сторінці(menu) та вкладці (tabs). Приклад цієї моделі можна побачити на рисунках 2.9 та 2.10.

```
const [ selectedMenu, setSelectedMenu ] = useState( initialState: '');
```

Рисунок 2.9 – Контролер збереження

На рисунку 2.9 представлений приклад контролеру збереження взятий із проекту, за допомогою, якого дані зберігаються в компоненті, а на наступному рисунку 2.10 зображено контролер зміни, який перезаписує збережені, попереднім контролером дані.

Приклади зображені на цих рисунках взяті із React проекту написаного для даного дослідження, в інших веб-платформах вигляд кожного із елементів архітектури може бути іншим, але загальний смисл завжди буде залишатись таким же самим.

```
<div
  className={`d-flex menu-ele
  onClick={() => {
    setSelectedMenu(value)
  }}
  key={value}
>
```

Рисунок 2.10 – Контролер зміни

2.3 Структура проекту

Оскільки в даній роботі повинно бути три ідентичні проекти на різних платформах, потрібно детально описати структуру проекту для точності при розробці. Існує багато можливих виріантів опису структури проекту, для даного дослідження було обрано діаграму, яка покаже всю навігацію в проекті.

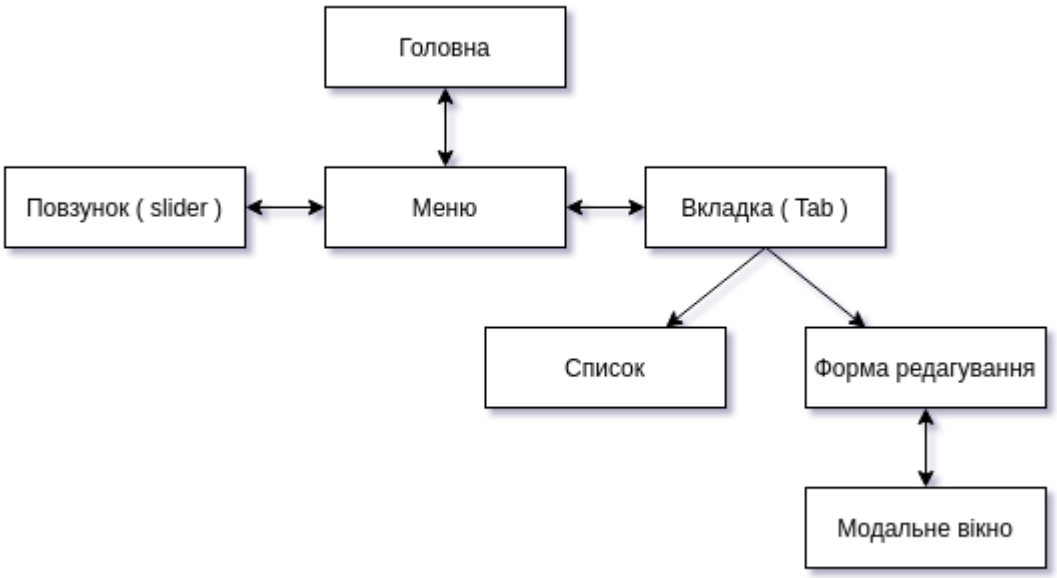

Рисунок 2.11– Навігація на веб-сайті

Навігація на проекті реалізована за допомогою інструментів та функцій, які надані веб-платформою по замовчуванню, це зроблено для того щоб не

порушувати точність дослідження додатковими бібліотеками, які для різної платформи можуть мати різну функціональність та швидкість завантаження.

На рисунку 2.11 зображено схему навігації по веб-сайті.

При відкритті веб-сайту користувач опиняється на головній сторінці, на якій буде розміщено всі типові елементи інтерфейсу, які потрібні для порівняння ефективності.

На всіх сторінках сайту є можливість відкрити меню в якому на даний момент знаходяться два варіанти вибору, а саме: Повзунок та Вкладка.

При переході на сторінку «Повзунок» буде відмальовано сторінку із слайдером в якому будуть зображення потрібні для порівняння швидкості завантаження сторінки.

На сторінці ж «Вкладка» знаходяться вкладки, які користувач може переключати. Вкладки також реалізовані без застосування будь-яких бібліотек, для точності аналізу. Є три під-таби, які користувач може вибрати натиснувши мишкою по назві таби. В одній із таб розташований типовий елемент список. В іншій табі розташована форма редагування, яка складається із типових елементів інтерфейсу для введення даних а саме:

- радіокнопка (radio button);
- прапорець (check box);
- список, що розкривається (combo box, drop-down list);
- поліредагування (textbox, edit field).

Користувач може редагувати всі поля, які знаходяться в цій вкладці, після редагування він може натиснути кнопку «Save» одразу пісня натискання якої відкриється модальне вікно із інформацією яку ввів користувач.

Всі елементи та переходи на сторінках є типовими елементами інтерфейсу і потрібні для коректного порівняння ефективності веб-платформ в браузерах.

Реалізація на різних веб-платформах є різною, але навігація та елементи на сторінках є такою ж самою для всіх.

2.4 Duck архітектура

В розділі 1.3 було проаналізовані, які платформи є найбільш популярні, особливості саме цих платформбудуть описані в даному розділі, тому що архітектура проектів все ціло залежить від того ні якій платформі написаний веб-сайт. У всіх платформах є свої спільні та відмінні особливості архітектури.

На проекті буде використовуватись duckarchitecture. Названа вона так через те, що чим більше папок та файлів в проекті, тим більше вони своєю структурою будуть нагадувати качиний клюв, це можна частково побачити на рисунку 2.12.

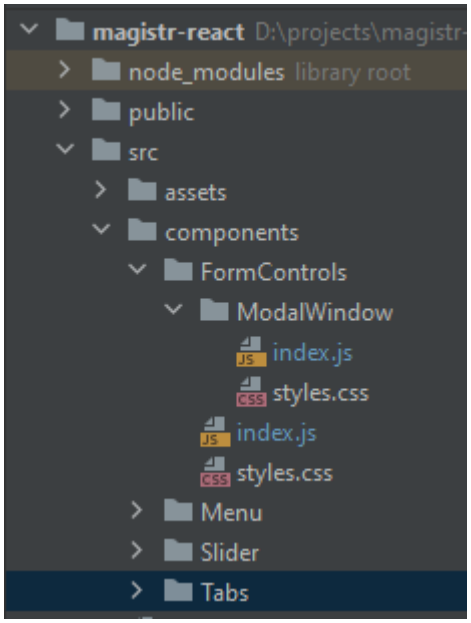

Рисунок 2.12 – Duck архітектура React проекту

Головною ідеєю цієї архітектури є те, що всі файли зв’язані з компонентом повинні знаходитись в одній папці з ним, чим спрощують пошук та швидкість доступу до необхідних файлів. Також на рисунку 2.12 можна побачити, що в проекті Reactзв’язані з компонентом файли –це файли стилю, але на інших платформах це можуть бути інші файли, наприклад в VueJs проекті буде тільки один файл, а вже Angularбуде багато файлів, які на пряму зв’язані з компонентом, і це можна побачити на рисунку 2.13.

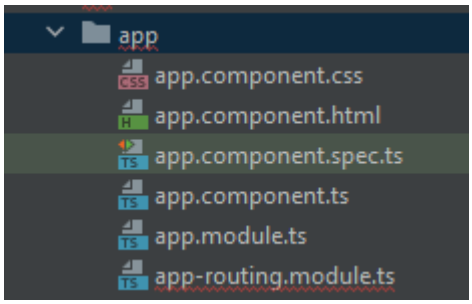

Рисунок 2.13 – Duck архітектура Angular проекту

І хоча ця архітектура має свої мінуси, так як дуже багато папок, які можна би було не створювати, вона найкраще підходить для нашого дослідження.

Висновок до розділу 2

Результатом розділу стало повністю описана структура проекту, описано плюси та мінуси всіх підходів до створення. Створено діаграми компонентів, розгортання, станів та прецедентів. Також описано структуру проекту, навігацію по проекті та з яких типових елементів складається веб-сайт.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА АНАЛІЗ ЕФЕКТИВНОСТІ ФРОНТ-ЕНД ПЛАТФОРМ

3.1 Порівняння часу та простоти створення проекту на різних платформах

Простота створення проекту на тій чи іншій платформі також впливає на загальну ефективність, тому що чим простіше створення тим швидше програміст почне роботу над проектом, не витрачаючи свій час на непотрібні початкові налаштування. В даному підрозділі буде описано етапи створення проекту на різних веб-платформах та буде проведено порівняння. Ефективність буде визначений за допомогою порівняння часу і сил які були витрачені на створення.

Аналіз проводитись по тих кроках які є відмінні для кожної із платформ, порівняно час, який потрібно для створення проекту та зручність для пошуку всіх необхідних установок платформи.

Є інструменти, які потрібно встановити для всіх трьох платформ перед початком створення проекту, ці інструменти потрібні для всіх фронт-енд платформ, це:

- node;
- npm або yarn;
- webstorm або інше інтегроване середовище розробки для роботи з проектом.

Установка node можлива двома способами.

Перший спосіб установки node–це встановлення додаткового інструмента для менеджменту версій nodeпід назвою nvm. Для його установки потрібно перейти на офіційний сайт (https://github.com/coreybutler/nvm-windows#node-version-manager-nvm-for-windows) та пройти інструкцію по установці. Даний спосіб є корисним тоді,

коли є більше ніж один довготривалий проект з різними версіями node. Оскільки проекти для дослідження є новими і для точності аналізу будуть використовувати одну версію для всіх платформ, цей варіант можна відкинути.

Другий спосіб установкиnode – це завантажити його з офіційного веб-сайту https://nodejs.org/en/, де можна вибрати, яку версію пакетів потрібно завантажити для встановлення (рис 3.1).

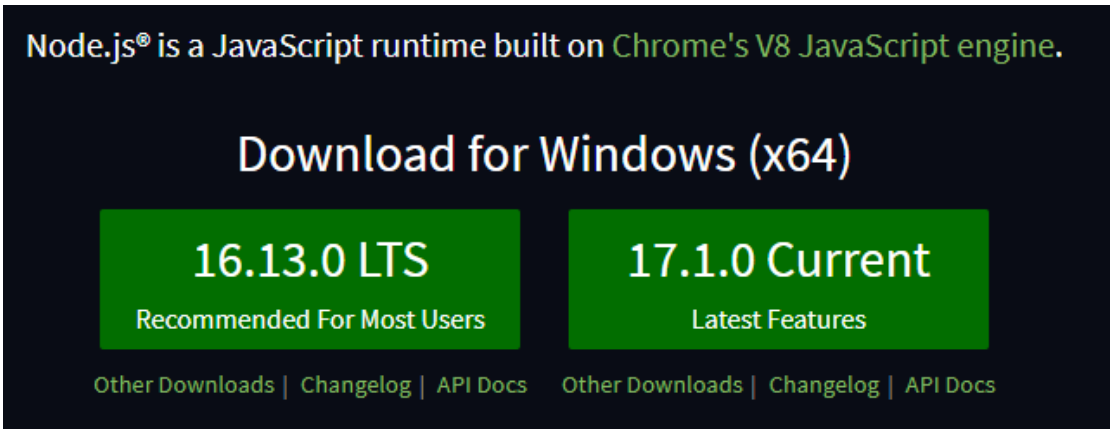

Рисунок 3.1 – Вибір версії node

Потрібно 16.13.0 LTS, необхідно завантажити та встановити застосунок по інструкції, що описана на сайті.

Також є можливість завантажити старіші версії node, таке буває потрібно у випадках, коли проект вже існує багато часу і встановлення нової версії може призвести до поломок в проекті, і не має можливості запуску проекту, але оскільки для дослідження будуть створені нові проекти,можна взяти останню LTS(Long-TermSupport), що означає що це остання стабільна версія.

Менеджер пакетів npmвстановлюється по замовчуванню з node, та версія менеджера на пряму залежить від версії встановленого node. Тому не потрібно ніяких додаткових маніпуляцій для установки.

Інтегроване середовище розробки ніяк не впливає на точність дослідження, але чим більш вміло розробник ним користується тим більша

буде швидкість написання коду для проекту. Зараз найпопулярнішим середовищем для фронт-енд розробки є WebStorm, тому і в даному проекті буде використовуватись він.

Для установки потрібно перейти на офіційний сайт програми https://www.jetbrains.com/help/webstorm/installation-guide.html#standalone і вибрати свою операційну систему та почати завантаження. Після завантаження потрібно пройти всі кроки установки. Ось так виглядає початковий екран середовища після першого запуску (рис 3.2):

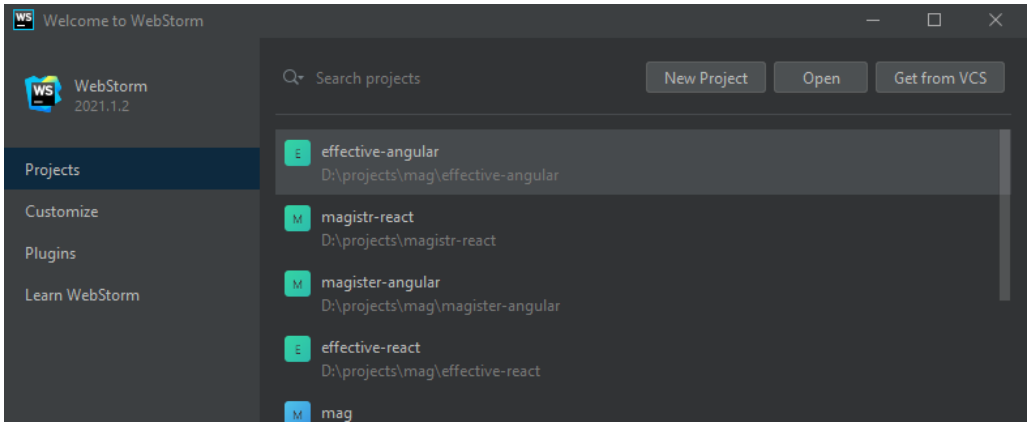

Рисунок 3.2 – Початковий екран програми WebStorm

На початковому екрані програми WebStorm потрібно буде вибрати проект, який буде створений в цьому розділі і над яким буде проводитись розробка.

Кожний розробник налаштовує IDE для себе, так щоб йому було як най зручніше користуватись ним та писати код, багато хто налаштовує його і під конкретну платформу, але для точності всі налаштування будуть по замовчуванню.

Створення React проекту

Для створення React проекту не потрібно ніяких додаткових установок, того що було встановлено nodeта npmвже достатньо для створення.

Спочатку потрібно відкрити папку де потрібно його створити після цього відкрити термінал і в ньому виконати:

```
npx create-react-app effective-react
```

Одразу як команда почне своє виконання буде створено папку під назвою effective-react, тако ю ж буде назва проекту в програмних файлах, та почнеться завантаження всіх необхідних даних для функціонування проекту. Як команда завершить своє виконання розробнику потрібно відкрити термінал в папці створеного проекту і ввести та виконати:

```
npm start
```

Після виконання цієї команди буде відкрито вікно браузера, із створеним сайтом, вигляд якого показаний на рисунку 3.3.

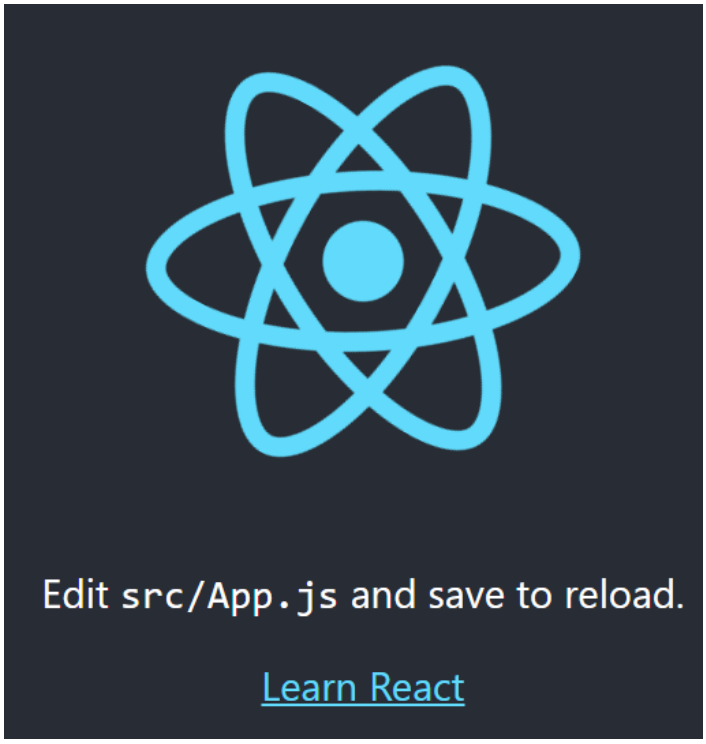

Рисунок 3.3 – Викляд Reactвеб-сайту після створення

І це все що потрібно для ініціалізації React проекту, розробник може починати розробку проекту.

React платформа використовується в таких відомих проетках, як:

- він був розроблений і використовувався для супроводу та кодування рекламних кампаній Facebook;
- twitter, популярна у всьому світі соціальна мережа;
- дуже популярний додаток для обміну повідомленнями з високим трафіком, Whatsapp;
- обмін фотографіями на відомій платформі соціальних мереж Instagram.

Створення Angular проекту

Для створення Angularпрокту потрібно встановити додатковий CLI, команду для встановлення якого можна знайти на офіційному сайті Angular. Команда для встановки CLI:

```
npm install -g @angular/cli
```

CLIAngularдозволяє користуватись консольними командами, які призначені для створення проекту, для створення компонентів, модулів, запуску проекту і т.д..

Після того як знайшли необхідну команду потрібно її ввести в терміналі, в не залежності від місця в якому він відкритий, встановлення займе декілька хвилин, після успішної установки буде повідомлення про це, потрібно перезапустити термінал для того щоб команди стали доступні в терміналі.

Перезапустивши термінал потрібно перейти в необхідну папку де планується створення проету, та виконати наступну команду:

```
ng new effective-angular
```

Перед початком сторення проекту в терміналі появиться декілька запитань про проект (рис 3.4), це потрібно для того щоб CLI розумів що саме потрібно створювати, бо налаштувань проекту може бути багато різних.

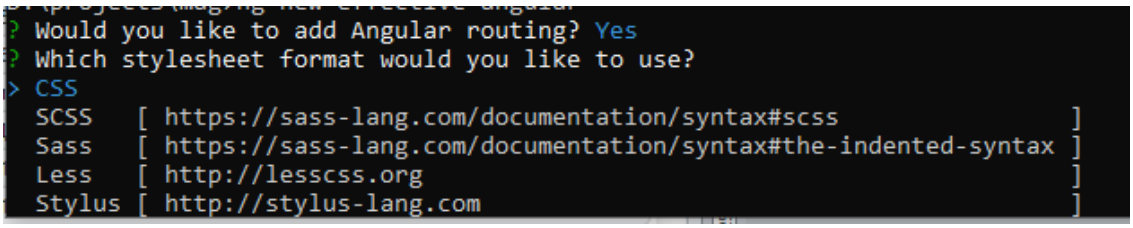

Рисунок 3.4 – Питання при створенні Angularпроекту

Одразу після відповіді на запитання запуститься створення проекту під назвою effective-angular та встановить всі необхідні залежності і бібліотеки для коректнох роботи на даній веб-платформі.

Для запуску створеного проекту потрібно перейти в папку створеного проекту в терміналі та ввести там команду:

```
ng serve
```

Команда запустить проект, та виведе в терміналі хост під яким проект запущений, потрібно буде перейти по даному посиланню щоб побачити початковий стан веб-сайту за замовчуванням хостом для щойно створеного проекту є «`http://localhost:4200/`», або є можливість добавити «`--open`» на кінці попередньої команди щоб браузер відкривався одразу ж після запуску проекту.

На рисунку 3.5 зображений вигляд веб-сайту після створення, тобто розробник може починати роботу над проектом.

Angular використовувався для створення таких проектів, як:

- розроблено компанією Google і використовується у всіх його додатках Google AdWords;
- The Guardian, провідна компанія у видавничому секторі;
- Weather.com, широко використовуваний веб-сайт для прогнозу погоди по всьому світу.

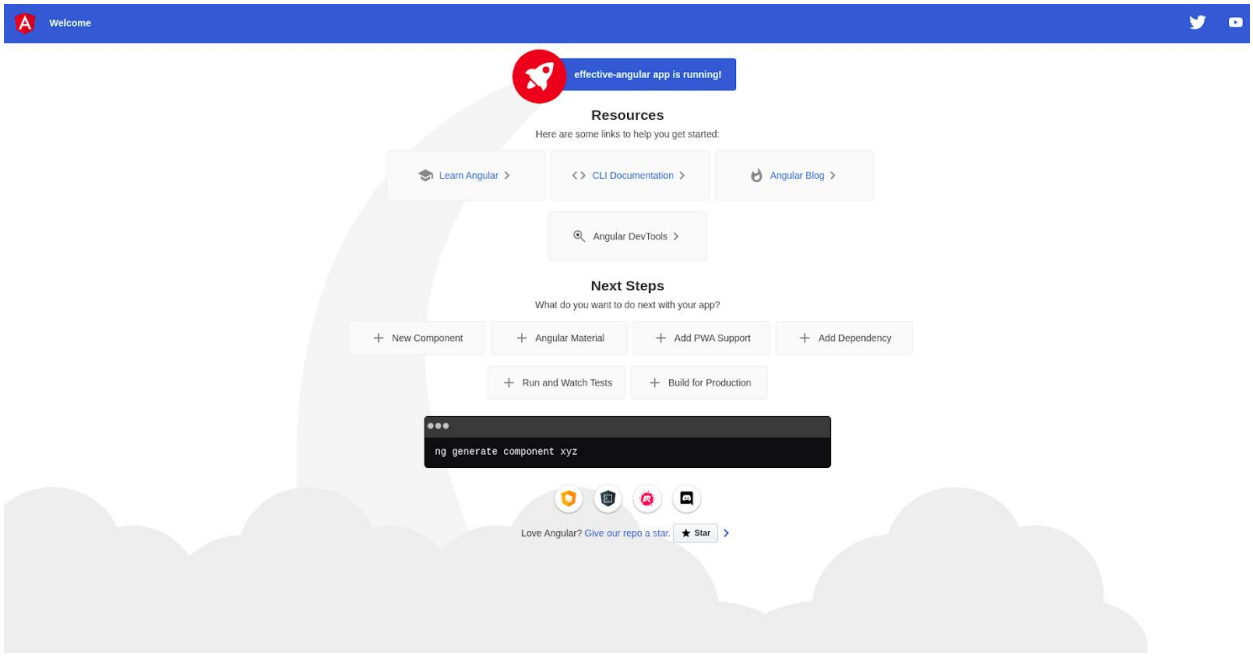

Рис 3.5 – Початковий вигляд веб-сайту на Angular

Створення VueJs проекту

На офіційному сайті веб-платформи vueJs розміщені три можливі варіанти використання vue на проекті, давайте розглянемо їх в тому ж порядку що і розміщений на сайті.

Першим є додавання за допомогою CDN, тобто вставити такий тег з посиланням в headабо в кінець htmlсторінки:

```
<script
src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></
script>
```

І при запуску проекту всі необхідні для роботи бібліотеки будуть загружені через посилання вказане в тегу вище.

Такий метод не рекомендується, тому що можливо таке що дане посилання може припинити своє існування, наприклад підтримку версії припинили або і взагалі видалили версію по якихось причинах. Також мінус такого встановлення те, що IDE не бачить коректно те, що на проекті встановлений vueJsта не буде допомогати в написанні коду для нього. І також мінусом є те, що цей варіант підходить для вже створених проектів.

Другим методом є додавання vueJs за допомогою npm.

Команда для встановлення за допомогою npm:

```
npm install vue
```

Командою додається всі необхідні файли та залежності для проекту, і можна починати роботу на vueJs.

В цілому таке встановлення є рекомендованим і є хорошою практикою, але нажаль це підходить для вже створених проектів, де вже налаштований npm, що не підходить для даного дослідження.

І третім методом створення проекту є встановлення CLI, нажаль офіційний сайт vueJsв описі цього методу перенаправляє користувачів на інший сайт https://cli.vuejs.org/ де є опис та кроки створення проекту. Саме третій метод є самим підходящим для дослідження, тому що це створення проекту з нуля, і всі файли в проекті будуть налаштовані на те що в ньому буде vue.

Для встановлення CLI потрібно виконати наступну команду в терміналі, в не залежності де він відкритий, для того щоб встановити глобально необхідні бібліотеки:

```
npm install -g @vue/cli @vue/cli-service-global
```

Після виконання для користувачів стануть доступні всі команди з CLI.

Потрібно перезапустити термінал та перейти в папку, де планується створення проекту та виконати команду:

```
vue create effective-vue
```

Почавши виконання команди, термінал почне запитувати, що саме потрібно встановити та з якими параметрами, в залежності від того яка версія

встановлюється питання можуть змінюватися, на момент проведення дослідження було тільки 1 питання, яке зображено на рис 3.6.

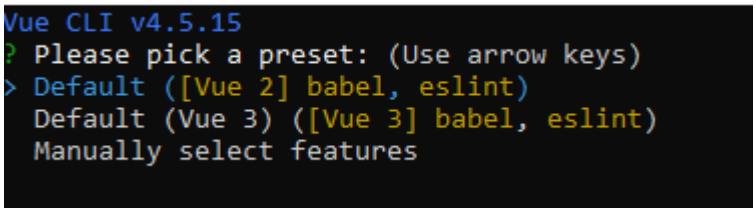

Рисунок 3.6– Запитання в терміналі при встановленні vueJs

Питання того яку версію vueJs потрібно встановити, тому що в деяких випадках можливе таке що необхідно старішу версію. Для даного дослідження потрібна сама остання версія, тому обрано Default (Vue 3). Після відповіді на запитання починається встановлення самого проекту. Закінчивши з установкою буде показане повідомлення, що все добре та команда для переходу в папку проекту і запуску, перейшовши в папку вводиться команду:

```
npm run serve
```

Проект запуститься за замовчуванням на хості http://localhost:8080/, потрібно буде перейти в браузер та відрити його щоб побачити початкову сторінку після створення проекту (рис 3.7). На відмінну від React кожний раз при запуску проекту розробник повинен сам переходити в браузер та відкривати необхідний веб-сайт, що можна приписати як невеликий мінус для даної платформи. На веб-сайті одразу після встановлення є список корисних посилань де можна навчатись написанню на цій платформі, та є корисними для розробників які тільки навчаються писати на цій фронт-енд платформі, що можна віднести до плюсів.

Це всі кроки які були зроблені для встановки проекту на VueJs.

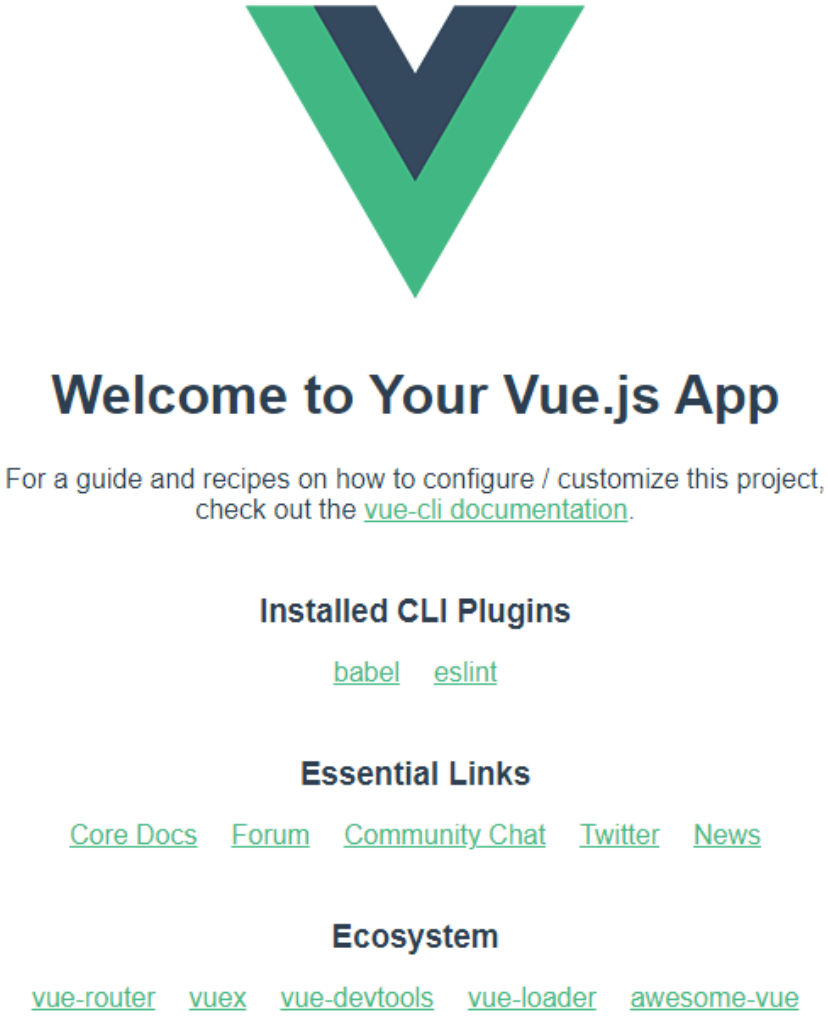

Рисунок 3.7 – Початкова сторінка після створення VueJs проекту

Vue також використовувався для створення проектів відомих на цілий світ, це:

- 9GAG, сайт соціальної мережі для обміну актуальним контентом;
- GitLab, менеджер репозиторію, який дозволяє команді розробляти або дублювати код.

Було описано всі необхідні кроки для створення проектів, від ідеї до початку розробки самого проекту після його установки.

Можна побачити що самою простою та найшвидшою установкою – установка React платформи, тому що для неї не потрібно ніяких додаткових установок окрім тих що є загальними для всіх трьох платформ, що зменшує час витрачений на пошуки необхідних команд та робить установку

простішою, також не великим плюсом є автоматичне відкриття браузеру після старту проекту.

Angulari VueJs потребують установки їх власних CLI, питання є в обидвох платформах, хоча від версії може відрізнятись кількість питань, тому на перше місце вони вже не претендують. Невеликою перевагою Angular–є те що вся необхідна інформація знаходиться на одному веб-сайті, в VueJs же є два веб сайти і при установці через CLI потбірно використовувати інформацію з обидвох, що для початківців в розробці може зайняти багато часу при першому налаштуванні. Також невеликим плюсом Angularнад VueJsє те що в інструкції по установці одразу описано як налаштувати автоматичне відкриття веб-сайту після запуску проекту, у VueJs, якщо і є така можливість, вона не описана при установці, а значить на її пошуки користувач потратить ще більше часу. Тому на другому місці є Angular,і хоча по простоті він такий же як і VueJs, часу затраченого на створення пішло набагато менше через правильне розташування інформації про нього.

Висновок порівняння

З порівняння можна зробити висновок, що самим простим та найшвидшим в створенні є React, потім більш складнішим є Angular, а через не правильне розташування інформації про платформу, що забирає більше часу на створення VueJsє найгіршим з трьох.

3.2 Аналіз зручності та простоти розробки проекту

На основі аналізу проведеного в розділі 1.3 для розробки було вибрано три веб-платформи, а саме React, Angular, VueJs. Для дослідження цих платформ потрібно створити та розробити веб-сайт на кожній із запропонованих платформ, вимогою є те що вони будуть ідентичні та без використання будь-яких додаткових бібліотек, щоб дослідження були якомога точними.

В розділі буде описано розробку на кожній із платформ та порівняно простоту використання та написання кожної із них.

Розробка React платформи

В офіційній бібліотеці React, не надається так багато можливостей, як наприклад в Angular, це дає свободу вибору інструментів програмування за вибором розробника. Іншими словами, Reactє більш гнучким при розробці проекту.

Сторонні рішення, такі як маршрутизація react, можуть бути інтегровані з цією платформою javaScript. Крім цього, можна використовувати MobX і Redux для підтримки завдань управління даними, а також React можна використовувати як додаткову бібліотеку на проекті, що буде доповнювати вже існуючий проект.

Якщо наймати кращих розробників програмного забезпечення з великим досвідом, то їм дуже сподобається така гнучкість. Крім того, буде створений дивовижний і динамічний додаток, розроблений для власного бізнесу.

Після створення проекту, яке описано в розділі 3.1. Проект складається із початкових файлів, які потрібні при наступній розробці. Початковий проект складається з таких файлів:

- public – папка, де знаходяться два логотипи даної платформи, та htmlфайл, який є тільки одним htmlфайлом на проекті, і відповідає за розміщенняReactкоду на сторінці;
- src – папка, в ній розміщені всі файли з React кодом, і весь код, який буде писатись знаходиметься в цій папці,після встановлення там знаходиться index.js – файл, в якому здійснюється ініціалізація React, тобто при кожному запуску обов'язково викликається цей файл (рис 3.8);
- setupTest.js – в ньому знаходяться тести, які провіряють чи проект взагалі запуститься, App.js – приклад сторінки написаної на React;

Рисунок 3.8 – index.js, ініціалізація React проекту

- README.md – в якому описується інформація про те як запустити проект та із посиланнями на корисні статті, які повинні прочитати всі хто хоче розвиватись як розробник на даній платформі;
- .gitignore – файлу, в якому описано які папки та файли не потрібно вивантажувати в git, цей файл є в більшості проектів даного часу, тому що ці проекти використовують системи контролю версій;
- package.json – в цьому файлі описані команди доступні в проекті та бібліотеки, їх версії, які використовуються в додатку, основна мета його це впевнитись в тому, що всі розробники на проекті використовують одні і тіж версії бібліотек та для їх зручного завантаження. Всі бібліотеки повинні бути описані в цьому файлі та завантажуватись командою:

```
npm install
```

Це всі файли які присутні після створення проекту, структура і розміщення їх є зрозумілим.

В ході розробки було створено ще декілька папок та файлів, по структурі, яка рекомендована при розробці на даній платформі. Було створено такі додаткові елементи:

- assets – папка, в якій знаходяться додаткові матеріали, шрифти, картинки, іконки. В цій папці було створено дві підпапки iconsta images, в яких відповідно знаходяться всі іконки та картинки, які є на проекті;
- components – папка, з React компонентами проекту, вона складається з чотирьох підпапок:
  - Menu – компонент, який відповідає за те щоб відобразити та обробити все що зв’язано з меню, яке знаходиться на кожній сторінці сайту;
  - Tabs – вкладки, є головним компонентом однієї із сторінок;
  - Slider – повзунок з зображеннями, є іншим компонентом сторінки, яка реалізована в проекті;
  - FormContorl – компонент з формою редагування, яка містить в собі найбільшу кількість типових елементів інтерфейсу:
  - ModalWindow – модальне вікно, яке знаходиться під-компонентом FormControl.
- constants.js – файл в якому знаходяться всі тексти, які присутні на веб-сторінці, їх зберігають в окремому файлі на той випадок якщо на сайті буде більше однієї мови, щоб для додавання наступних мов потрібно було тільки добавити файл з відповідною мовою.

Кожна із підпапок компонента складається із двох файлів а саме:

- index.js – в якому знаходиться код написаний для компонента та його логіка;
- styles.scss – де знаходяться стилі написані для даного компонента.

Для створення додаткових папок та файлів потрібно правою кнопкою в IDEнатиснути на папку в якій планується створення та вибрати потрібний тип (рис 3.9).

Структура за якою створено всі файли та папки проекту описана в розділах 2.3 та 2.4.

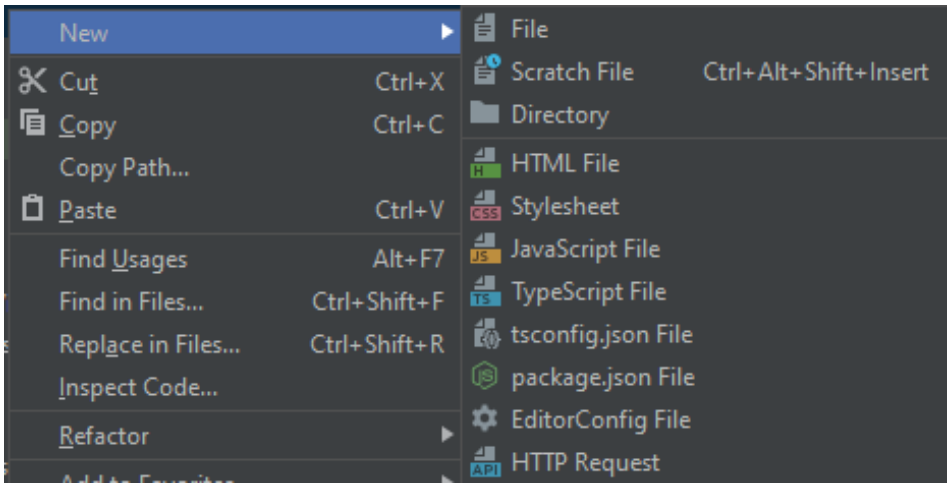

Рисунок 3.9 – Вибір створюваного елементу

Після опису всієї структури проекту та планування майбутніх кроків можна приступити до розробки самого проекту. Розробка була почата з компоненту Menu, в якому знаходиться меню та імплементація всіх можливих маніпуляцій над ним. В розробці було використано функціональні компонентиReact, дані в яких зберігаються за допомогою хуків (hooks). Хук в якому зберігається інформація про те яка сторінка відкрита в даний момент часу:

```
const [ selectedMenu, setSelectedMenu ] = useState('');
```

```
<div
  onClick={() => {
    setSelectedMenu(value)
  }}
>
```

Рисунок 3.10 – Перезапис вибраної сторінки за допомогою хука

І за допомогою нього ж можливий перезапис вибраної сторінки, це зображено на рисунку 3.10.

В меню було реалізовано тільки два можливі варіанти вибору сторінок, а саме: slider та tabs на яких відповідно знаходяться повзунок з зображенями та вкладки.

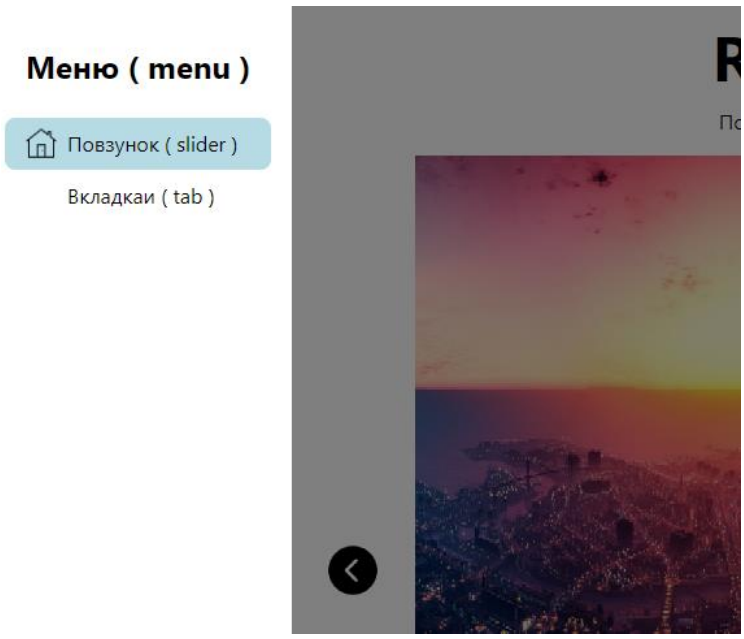

Рисунок 3.11 – Відкрите меню

На рисунку 3.11 зображено частковий вигляд веб-сторінки коли меню є відкритим, відкриття на закриття меню також реалізовано за допомогою хуків, так само як описано вище.

Сторінка Slider (Повзунок з зображеннями), зображена на рисунку 3.12, для проекту було вибрано декілька зображень, які мають великий розмір, а саме 1MB, 1.6MB, 3.4MB та 17.1MB. Такі розміри були обрані тому, що швидкість завантаження зображень також частково залежить від платформи на якій написаний веб-сайт, також швидкість завантаження зображень залежить від браузера в якому це буде перевірено. Швидкість завантаження може відрізнятись від першого разу коли користувач потрапив на сторінку і від наступних разів, тому що браузер робить кешування зображення.

Для порівняння буде братись середня швидкість завантаження зображення, за п’ять раз, для точності порівняння.

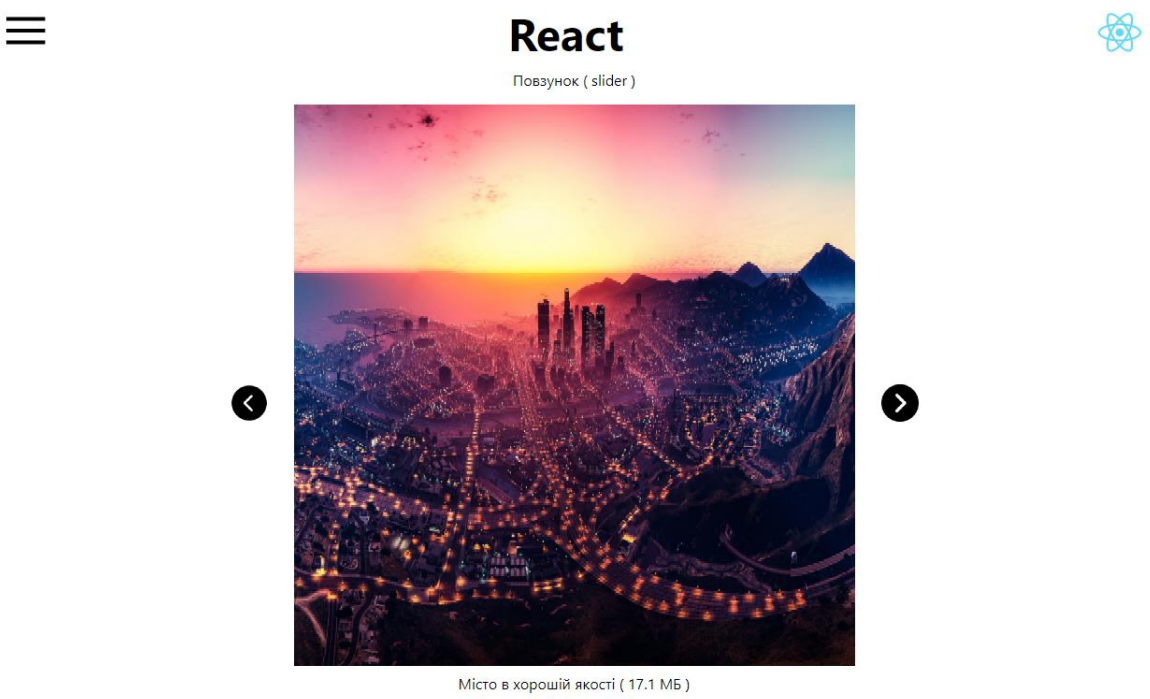

Рисунок 3.12 – Вигляд сторінки Slider

Сторінка Slider складається із двох кнопок які дозволяють переключати зображення на наступне та попереднє, цей функціонал реалізований за допомогою хуків та додаткових функції (рис 3.13), які вираховують яке саме зображення потрібно показати користувачу, та власне зображень.

```
const clickPrevious = () => {
    const previous = showedImage === 0 ? imageForTest.length - 1 : showedImage - 1;
    setShowedImage(previous);
};
```

Рисунок 3.13 – Функція зміни зображення на попереднє

Наступною сторінкою є Tabsтобто вкладки. Вкладки це типовий елемент інтерфейсу, який є напевно на 99% сайтів. Головна їх мета в тому щоб показувати інформацію в залежності від того яка вкладка вибрана в даний момент. На сайті створено три вкладки, кожна з яких показує свою інформацію. На рисунку 3.14 зображено зовнішній вигляд вкладок.

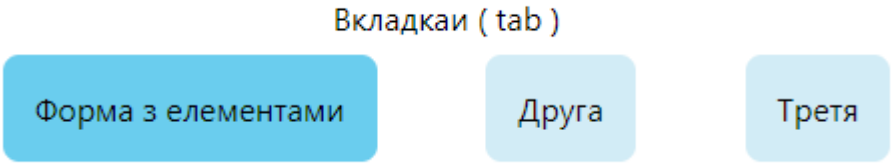

Рисунок 3.14– Зовнішній вигляд вкладок

Зміна самих вкладок оброблена за допомогою хуків, в яких зберігається інформація про відкриту вкладку, самі ж текстові назви вкладок беруться з constants.jsфайлу, який був описаний вище.

У вкладці форма з елементами знаходяться типові елементи редагування, такі як кнопка, радіокнопка, прапорець, значок, список, що розкривається, поле редагування (рис 3.15) та після кліку по кнопці «Зберегти» відкривається модальне вікно.

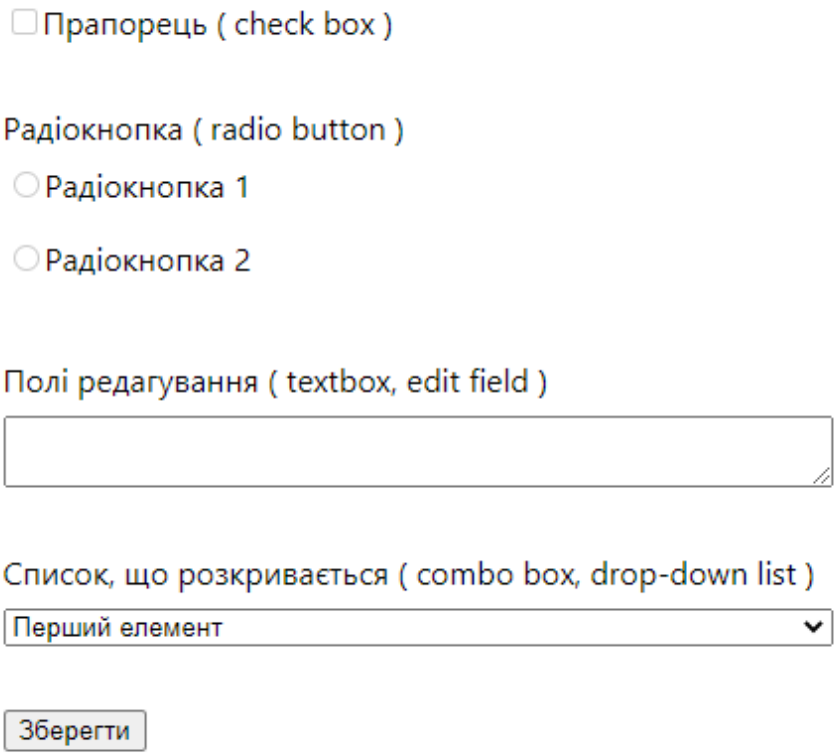

Рисунок 3.15 – Типові елементи редагування

Всі ці елементи редагування реалізовані за допомогою функцій, які надає Reactза замовчуванням, це зроблено для того щоб не добавляти бібліотеки, які можуть повпливати на результат досліджень. Та була написана додаткова функція, за допомогою якої дані передаються в хук для збереження:

```
const updateField = (name, value) => {
  setFormValues({ ...formValues, [name]: value });
};
```

Функція отримує з елемента редагування інформацію про його назву, та дані які записані в даний момент часу в цьому елементі.

Останнім типовим елементом інтерфейсу, який присутній на веб-сайті є модальне вікно, яке відкривається тоді коли користувач натисне на кнопку «Зберегти» в формі редагування, цю кнопку можливо побачити на рисунку 3.15, а на рисунку 3.16 зображений зовнішній вигляд модального вікна.

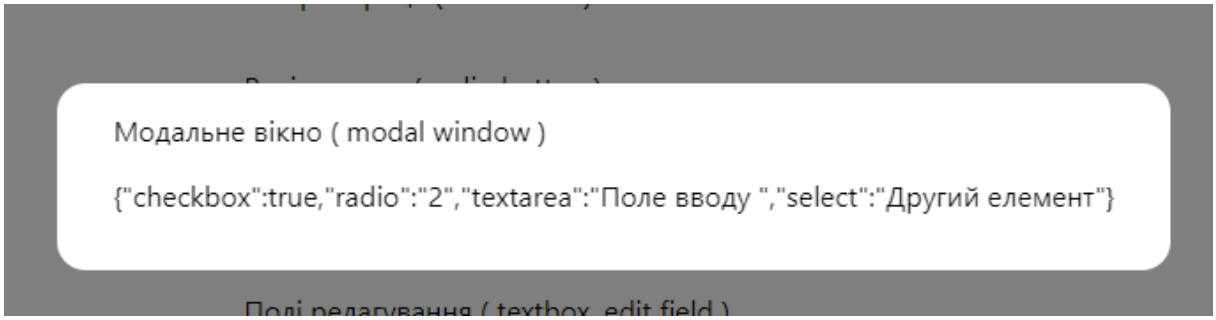

Рисунок 3.16 – Зовнішній вигляд модального вікна

В модальному вікні знаходиться інформація зі всіма назвами полів та всіма даними, які введені в ці поля. Закрити ж модальне вікно можна за допомогою натискання в любому місці сторінки лівою кнопкою мишки.

Весь код який пишеться для Reactкомпонента розміщений в одному файлі, та також можливі додаткові файли зі стилями, які знаходяться на тому ж архітектурному рівні.

Кожний компонент складається з таких частин як:

- імпорти використовуваних в компоненті бібліотек, компонентів, функції, констант, стилів;
- оголошення компоненти та його назва;
- список змінних переданих від батківського компонента;
- оголошення змінних використовуваних в компоненті;
- функції, які використовуються в компоненті та, або передаються в дочірній компонент;
- returnв якому знаходиться код написаний на JSX, який є аналогом htmlрозмітки;
- експортування компонента.

Вся інформація потрібна для реалізації та для навчання на цій веб-платформі знаходилась в одному місці, на яке можна перейти за цим посиланням: https://reactjs.org/docs/hello-world.html, не потрібно було проводити додаткових пошуків того як працює чи як реалізувати той чи інший елемент, в загальному архітектура є зрозуміла з першого погляду іншого розробника.

Розробка Angular платформи

Angular пропонуєвсе–відмаршрутизаціїдошаблонів. Розробникам не потрібно переходити на який-небудь інший інструмент для розробки додатків. В залежності від точки зору це можна вважати перевагою або недоліком.

Не потрібно вибирати бібліотеки маршрутизації. Можна просто почати процес кодування з усім, що передбачено в пакеті Angular.

Angularпроект складається з таких файлів та папок після закінчення встановлення:

- e2e – папка, в якій знаходяться всі тести, які написані для перевірки того чи запускається проект, і в майбутньому тести будуть знаходитись тут;

- tsconfig.---.json – налаштування потрібні для TypeScript, замість «---» може знаходитись app, base, spec, або ж воно може бути пустим, це все налаштування для різних ситуації;
- tslint.json – загальні правила написання коду описані в цьому файлі, це зроблено для того щоб всі розробники писали в одному стилі, і не тратили свій час в майбутньому на такі речі як розбирання не правильно побудованого коду;
- karma.conf.js – налаштування тестів;
- src – папка, в якій знаходиться весь код проекту, тут знаходяться такі файли та папки:
- test.ts – тест який провіряє чи все написано правильно;
- styles.css – стилі, якібудуть використані для всього сайту;
- polyfills.ts – налаштування імпортів, які використовуються для різних браузерів;
- main.ts – головний файл ініціалізації проекту, в залежності від варіанту збірки використовує налаштуваннядля розробки чи продакшину;
- index.html – головний html файл проекту, в ньому знаходиться ініціалізація Angularчастини проекту, та в ньому знаходиться теги head,body;
- favicon.ico – іконка з логотипом Angular проекту;
- assets – в якому нічого не знаходиться на момент після створення проекту;
- environments – папка, в якій знаходяться два файли з налаштуваннями, які використовуються в main.ts;
- app – знаходиться початковий код проекту, який написаний для прикладу того як працює код на Angular.

Спільними для Angular та React є такі файли: README.md, .gitignore, package.json. Про ці файли можна прочитати вище в описі Reactпроекту.

Фактично при розробці проекту було додано зображення та іконки в папку assets, які відповідно були збережені в папки images та icons.

Та додаткові файли компонентів, які створювались в ході розробки, та були збережені в app. Архітектуру цієї папки можна побачити на рисунку 3.17.

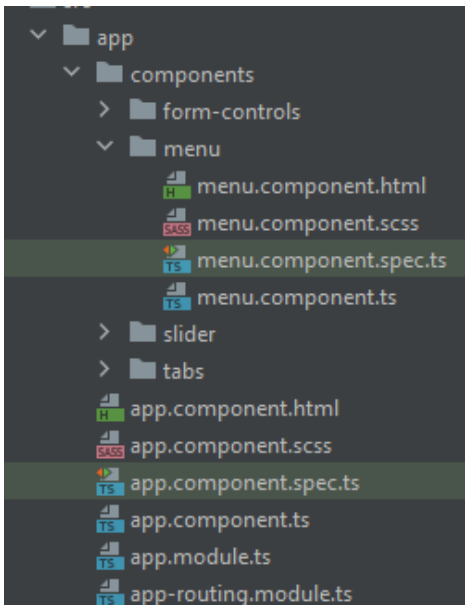

Рисунок 3.17 – Архітектура папки app

Як можна побачити структура папок є аналогічною до тої, що була в React проекті, це зроблено для точності аналізу ефективності, але самі файли відрізняються від таких як в React. Структура Angularкомпоненту розділена на такі частини:

- name.component.html – файл з htmlрозміткою;
- name.component.scss – в цьому файлі знаходяться всі стилі, які відносяться до компонента;
- name.component.spec.ts – початкові тести компонента, по замовчуванню перевіряють чи компонент написаний правильно та чи відобразиться на сторінці коректно;
- name.components.ts – тут знаходиться весь Angularкод.

В кожній папці nameвідрізняється та відповідає його назві, наприклад для компонентаmenu всі файли перелічені вище будуть починатись з menu, наприклад menu.component.html.

Проект, який створений на Angularє таким же самим як він і був на React, відрізняється тільки платформа, та функціонал, який надається цією платформою.

Оголошення компонента виконується в .tsфайлі на його вигляд можна глянути на рисунку 3.18.

```
@Component({
  selector: 'app-menu',
  templateUrl: './menu.component.html',
  styleUrls: ['./menu.component.scss']
})
export class MenuComponent implements OnInit {
```

Рисунок 3.18– Оголошення Angularкомпонента

Спочатку оголошується декоратор @Component, це означає що наступні рядки будуть оголошенням самого компонента, значення selector вказує на те, як буде викликатись цей компонент в середині іншого компонента. templateUrl вкахує на файл htmlрозмітки, який використовує компонент, stylesUrls це масив із можливих файлів стилю, в даному випадку це тільки один файл, який генерується автоматично, але можливо добавити більше файлів, та експортування компонента, для того щоб його можна було використовувати в інших компонентах.

```
<div class="content" role="main">
  <app-slider></app-slider>
  <app-tabs></app-tabs>

  <div class="footer"></div>
</div>
```

Рисунок 3.19 – Використання імпортованих компонентів

Всі компоненти які є в проекті імпортуються в одному app.module.ts файлі, це головний файл в якому також відбувається імпортування використовуваних бібліотек та загальних налаштувань. А на рис 3.19 зображене використання імпортованих бібліотек, а саме двох різних сторінок Slider та Tabs.

Переключення сторінки реалізовано за допомогою додаткових функцій, які описані в класовому компоненті, ці функції зображені на рисунку 3.20.

```
isMenuOpened = false;
selectedMenu = null;

setIsMenuOpened = (value: any) => {
  this.isMenuOpened = value;
}

setSelectedMenu = (value: any) => {
  this.selectedMenu = value;
}
```

Рисунок 3.20 – Функції компонента Menu

Розглянувши зображення детальніше, стає зрозуміло що isMenuOpenedє змінною, яка відповідає за те чи відкрите меню чи воно закрите. selectedMenuце параметр, який зберігає в собі інформацію про відкриту зараз сторінки, за замовчуванням він є null, що означає що ні одна сторінка не відкрита зараз. setIsMenuOpened це функція, за допомогою якої відбувається відкриття та закриття меню. setSelectedMenu функція, яка здійснює зміну сторінки, обновляючи необхідний параметр.

Дуже схоже реалізоване переключення вкладок на їх сторінці, також є параметр в який відбувається запис, та функція яка цей запис виконує.

Кожний Angular компонент складається з таких частин:

- імпортування необхідних для компоненту бібліотек, констант, функції і зображень;

- декоратор @Component, в якому описані файли зв’язані з даним компонентом;
- назва класового компоненту та його експортування;
- конструктор в якому описані параметри, які будуть використані в компоненті;
- ngOnInit – функція, яка викликається одразу після того як завантажується компонент;
- функції які можуть бути використані в компоненті, та написані розробником власноруч;
- експортування створених констант та функцій.

До мінусів цієї платформи можна віднести те, що кожний компонент має багато файлів, які можуть бути не потрібні, або через малий обсяг коду в цих файлів вони могли б бути і одним файлом.

Вся необхідна інформація для початківця в веб-розробці для цієї платформи знаходилась на одному веб-сайті, та не потрібно було шукати ніякої додаткової інформації для реалізації всіх необхідних типових елементів.

Розробка VueJs платформи

Платформа Vue так само проста, як і платформа React. Але екосистема Vue на своєму офіційному сайті може багато чого запропонувати своїм розробникам. Деякі з доповнень-це маршрутизатор Vue для маршрутизації, а для управління станом є Vuex.

VueJs також має візуалізацію на стороні сервера vue, щоб ініціювати розробку програми на стороні сервера. Отже, він не такий структурований, як Angular, але і не такий гнучкий, як React.

На рисунку 3.21 зображено архітектуру проекту одразу після створення VueJs проекту, яка дуже схожа до архітектури React.

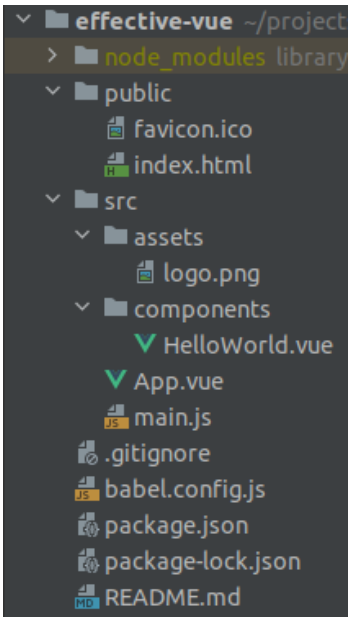

Рисунок 3.21 – Архітектура VueJs проекту

Більшість елементів архітектури є такими ж, як і в React та Angular проектах, відміняється хіба що:

- babel.config.js – налаштування для babel, що є збірником проекту;
- public – як і в React тут знаходяться публічні елементи, в даному випадку іконка з логотипом платформи для вкладки браузера, та html файл ініціалізації;
- src – папка, з всім кодом проекту, складається з таких елементів:
- assets – статичні елементи проекту, такі як зображення та іконки, в даний момент знаходиться тільки логотип платформи, для відображення на сторінці;
- components – папка в якій будуть знаходитись всі компоненти проекту, в даний момент тільки HelloWorld.vue, який створений для прикладу того як потрібно писати код на платформі;
- main.js – файл в якому відбувається ініціалізація та запуск всьо проекту;
- App.vue – головний та найвищий vue компонент, який запускається одразу після запуску проекту, і в ньому ж і реалізована логіка того які компоненти будуть показані наступні.

Структура ж готового проекту нагадує структуру, яка була реалізована при створенні React проекту, тобто були створені такі папки з компонентами: Menu, Slider, Tabs, FormControls, ModalWindow, які відповідають своїм аналогами в React проекті.

Кожний із vue компонентів складається із трьох частин, початок це <template>, в ній записується html розмітка, другою частиною є <script> в неї записується весь js код, який використовуєься в компоненті, <style> частина із стилями для компонента, ці стилі будуть використовуватись тільки для цього компонента і ніде більше.

```
isMenuOpened: false,
selectedMenu: '',
setIsMenuOpened: (value) => {
  this.isMenuOpened = value
},
setSelectedMenu: (value) => {
  this.selectedMenu = value;
}
```

Рисунок 3.22 – Функція переключення сторінки

Для реалізації переключення сторінки було використані функціонал, який надає платформа, на рисунку 3.22 можна побачити значення та функції, які використовуються для відкриття та закриття меню, а саме isMenuOpened: false це змінна яка зберігає дані про те в якому стані зараз меню, відкрите чи закрите, в свою чергу setIsMenuOpened відповідає за зміну цього значення в залежності від того яке value прийшло.

Вибрану в даний момент сторінку зберігає параметр selectedMenu: ‘’ у вигляді string змінної, а setSelectedMenu це функція, яка по функціоналу схожа із setIsMenuOpened, також зберігає вибрану сторінку.

Таким же чином було реалізовано відкриття та закриття модального вікна на сторінці із формою.

```
<div class="checkbox-wrap">
  <input
      type="checkbox"
      id="scales"
      name="scales"
      checked="formValues.checkbox"
      v-on:change="updateField( name: 'checkbox', !formValues.checkbox)"
  />
  <label for="scales">{{ checkboxLabel }}</label>
</div>
```

Рисунок 3.23 – Реалізація типового елементу інтерфейсу, прапорця

Типові елементи редагування реалізовані за допомогою html, який зображений на рисунку 3.23 та змінними обробки інформації, які знаходяться нижче.

```
formValues: {},
updateField: (name, value) => {
this.formValues[name] = value;
}
```

Розібравши код детальніше стає зрозуміло що formValues зберігає дані про всі редактовані поля які знаходиться в FormControl компоненті, updateField отрмує name що є назвою поля, та value що є значеннями поля, і за допомогою отриманих даних зберігає їх в створену вище змінну.

Кожний компонент складається з таких елементів:

- template– тег в якому знаходиться htmlрозмітка;
- script – частина в якій знаходиться код VueJskомпонента, складається із імпортів використовуваних бібліотек, констант які будуть використовуватись в компоненті, експортування об’єкта з ім’ям та datафункцією. В data функції оголошені змінні та функції даного компонента;
- style – тег в якому знаходиться всі стилі для даного компонента.

Структура папок та файлів дуже схожа з тією що була створена на React, і є простішою ніж Angular.

Як і в попередніх двох платформах вся необхідна інформація знаходиться на одному веб-сайті, і хоча вона була на різних сторінках це не викликало ніяких проблем при пошуку необхідної інформації, тому це не можна назвати мінусом.

Порівняння ефективності розробки

В цьому розділі було розроблено та описано етапи розробки проекту на трьох різних веб-платформах. Порівняти час розробки на кожній із платформ дуже важко, тому що все залежить від індивідуальних здібностей розробника, який проводить розробку даного проекту, отже час в цьому порівнянні враховуватись не буде.

Для кожної із платформ не викликало складнощів знайти інформацію, яка необхідна для реалізації задуманого проекту, і вся ця інформація доступна на офіційному веб-сайті платформи. Тому за цим критерієм не можливо визначити яка із платформ є найкращою.

В порівнянні структури найскладнішою можна виділити структуру Angular, тому що створювати компонент потрібно тільки за допомогою команд в консолі і в кожному компоненті знаходиться багато файлів, що в загальному розділяє функціонал по логічний частинах, а також ускладнює його. Імпорт компонента в Angular виконується в одному головному файлі, що водночас є зручно, однак якщо потрібно знайти, який саме використовується компонент в html розмітці – вже витрачається більше часу ніж в двох інших платформах. Саме тому Angular можна назвати самим складним та найменш ефективним в розробці.

React та VueJs дуже схожі в логіці створення компонента, для можливості використання потрібно створити файл та імпортувати його тільки в той компонент, де він буде використовуватись, всі частини компонента можуть бути в одному файлі компоненту. React дає змогу створити один додатковий файл для стилів, якщо це потрібно.В цілому це дозволяє розділити логіку компонента від статичних елементів, тому в цьому випадку це можна розглянути як плюс React платформи над VueJs.

Виходячи з порівнянь проведених вище можна зробити висновок що Reactта VueJsділять перше місце тому що немає великих перевад в одного над іншим. Angularж однозначно є гіршим в цьому плані, тому він на третьому місці.

3.3 Порівняння ефективності веб-платформ в різних браузерах

Від браузера яким користується користувач залежить те, як швидко і якісно завантажується сторінка, тому що кожний браузер написаний по своєму і навіть бувають ситуації, коли розробникам потрібно робити певні зміни в коді через те, що в якомусь із браузерів щось не працює. Тому перевірити ефективність платформ в різних браузерах є важливо, бо в одному можуть бути кращі результати ніж в іншому.

В розділі 1.3 б було проаналізовано те, які зараз браузери користуються популярністю в користувачів найбільше, це: GoogleChrome, Firefox та Safari, саме їх і буде використано для даного порівняння ефективності.

Розробники вважають за краще використовувати та вивчати React, а потім Vue, і в останнє – Angular.

Здатність користувачів писати код на певній мові програмування аналіз графіку зображено на рисунку 3.24. Коли справа доходить до Angular, ReactтаVue, то Angular відступає на крок у порівнянні з двома іншими.

Це пов'язано з тим, що Vue і React не вимагають найму веб-розробників, які володіютьtypeScript. Але серед цих двох Vue має перевагу, оскільки розробники вважають його більш «дружнім».

Згідно зі статистикою, наданою дослідженням Stateofjs, Vue набрав більше балів, ніж angular, коли розробників запитали: «використовував це тавикористовував би знову». Розробникам було надано безліч варіантів на вибір, які розглядалися в якості параметрів для порівняння.

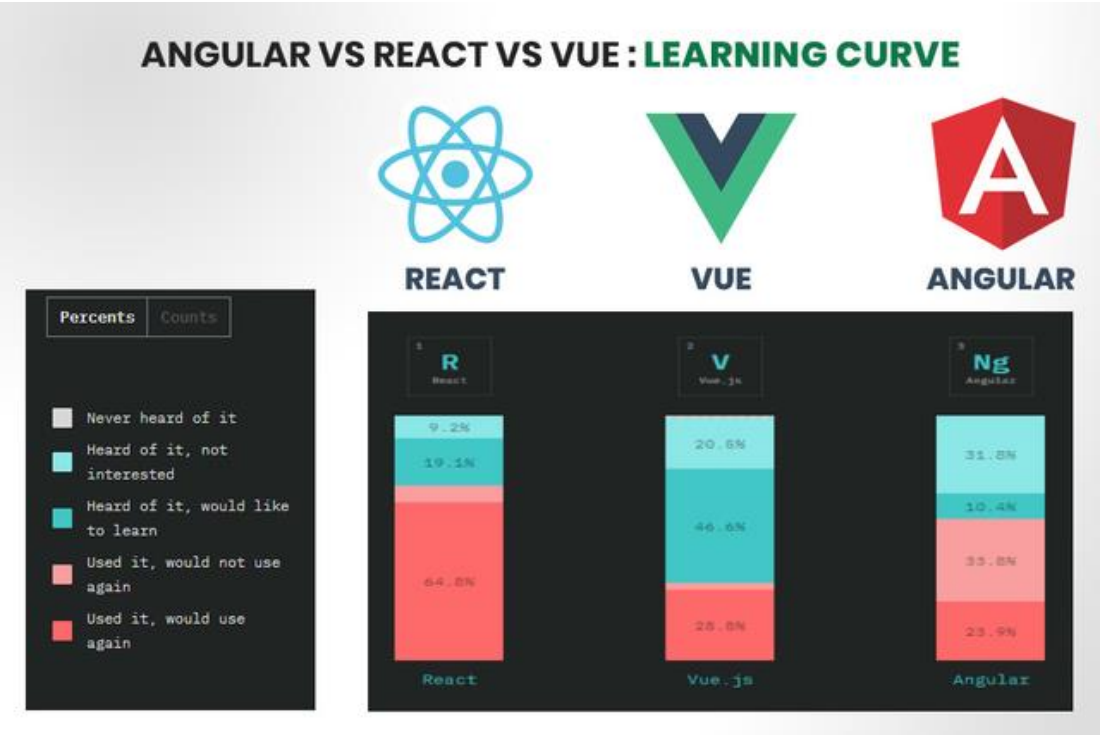

Рисунок 3.24 – Графік аналізу платформ

Як зображено на рисунку 3.24 можна побачити, що Vue отримав максимальну кількість голосів у варіанті «чув про це і хотів би навчитися». Це ясно показує яскраву криву навчання для цієї бібліотеки JS в майбутньому. Поряд з цим, графік являє собою кристалічне уявлення про зменшення популярності Angular.

А ось про React більшість розробників кажуть що працювали з ним та хотіли б працювати ще, що показує успішність платформи як такої.

Найбільш популярною думкою в інтернеті є те що Angular–це все, що потрібно, але він не дуже гнучкий, React – найбільш гнучкий, Vue – не надто гнучкий.

Одним із методів дослідження буде порівняння швидкості завантаження сторінки в цих браузерах. Швидкість завантаження можна побачити відкривши панель розробників на любій із сторінок любого веб-сайту, та перейшовши у вкладку Network. На рисунку 3.25 зображено приклад того як відображається швидкість завантаження сторінки в Chrome.

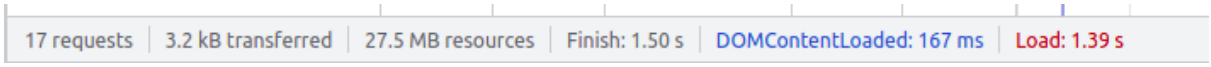

Рисунок 3.25 – Швидкість завантаження сторінки в браузері Chrome

Розберемо всі дані з рисунка 3.25 детальніше з ліва на право, перша інформація «requests», показує на те, скільки запитів між браузером та проектом було зроблено. Не можна сказати, що їх кількість впливає на швидкість завантаження сайту, тому що наприклад тільки одним запитом всі дані грузились би набагато довше, а якщо б запитів було б наприклад сто, то загрузка відбувалась також довше бо на обробку кожного запиту йшов час, тому цей параметр не буде приймати участь в аналізі ефективності.

Наступний параметр «transferred» показує те, скільки зжатих даних було передано. Цей параметр дуже не стабільний та навіть, в одному і тому ж браузері і на тій же платформі може показувати різні дані при кожному перезавантаженні сторінки. Так наприклад, перший раз він показав 3.2 kb, наступний 712.4 kb, а на третій раз 20.4 Mb. Тому цей параметр також не буде використовуватись для аналізу.

Параметр «resources» є важливим для порівняння, цей параметр показує те, скільки даних потрібно завантажити для відображення веб-сайту. Чим менше тут число тим краще, тому що зрозуміло, що 2 Mb завантажуться набагато швидше ніж 27.5 MB.

Таблиця 3.1 – Значення параметра resources

	React	Angular	VueJs
Google Chrome	27.5 Mb	28.4 Mb	27.9 Mb
Firefox	22.06 Mb	27.07 Mb	23.10 Mb
Safari	23.4 Mb	27.9 Mb	25.2 Mb

Більш детально розібравши дані показані на таблиці можна побачити, що від браузера в якому проводились заміри також залежить те, скільки даних

завантажується при кожному обновленні сторінки. І Google Chrome завантажує найбільше даних для всіх трьох платформ, що є мінусом для нього, Safary завантажує менше даних ніж Chrome, але Firefox завантажує найменше, тобто можна зробити висновок, що в Firefox завантаження сторінок буде найшвидшим. Дивлячись же на платформи зрозуміло що найменше даних завантажується для React, потім всього лиш на 1 Mb більше завантажується для VueJs, а ось для Angular завантажується ще на 4Mb більше ніж для vueJs.

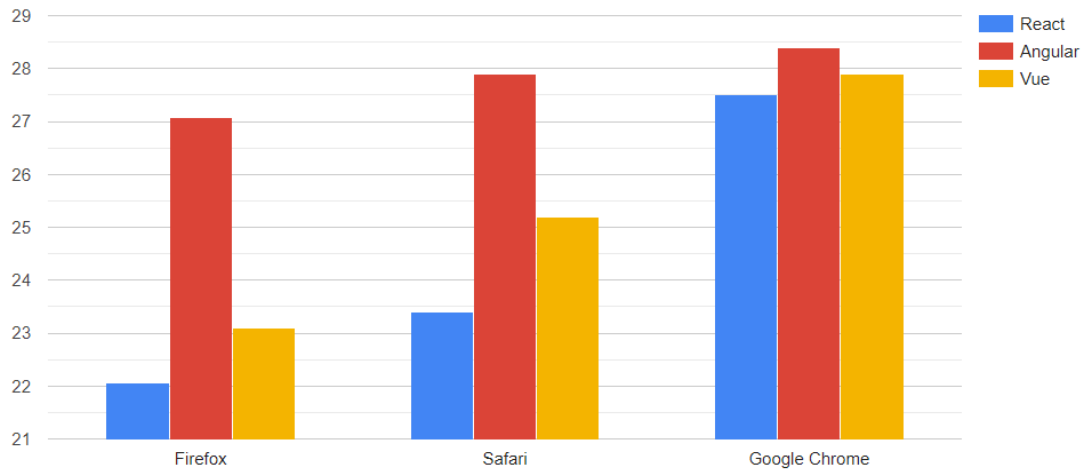

Рисунок 3.26 – Графік параметру resources

На рисунку 3.26 представлений графік, із даними параметра resources, на якому чітко видно що React у всіх браузерах є найменш об'ємною платформою. Висновок такий що найменший розмір завантажуваних файлів є в веб-платформи React. Наступний по розміру завантажуваних файлів є vueJs, і найгіршим по цьому параметрі є Angular. Також можна побачити що найкращим браузером по цьому параметрі є Firefox, який на диво на третьому місці по популярності серед користувачів.

Параметр «Finish» показує час від заходу на сторінку завантаження готової сторінки. Чим менший час тим краще для веб-сайту, тому що по

статистиці якщо цей час більше 4 секунд то більше половини користувачів покидають веб-сайт.

Нижче наведено таблицю з даними від платформ та браузерів в яких проводилось дослідження.

Таблиця 3.2 – Значення параметру finish

	React	Angular	VueJs
Google Chrome	1.50 s	1.58 s	1.55 s
Firefox	279 ms	1.09 s	574 ms
Safari	742 ms	1.24 s	755 ms

Проаналізувавши інформацію, яку отримано в ході дослідження можна побачити, що веб-сайт довше за все завантажується в GoogleChrome браузері, хоча цей браузер є самим популярним. Safari завантажує сайти швидше, але найкраще сайт завантажує Firefox.

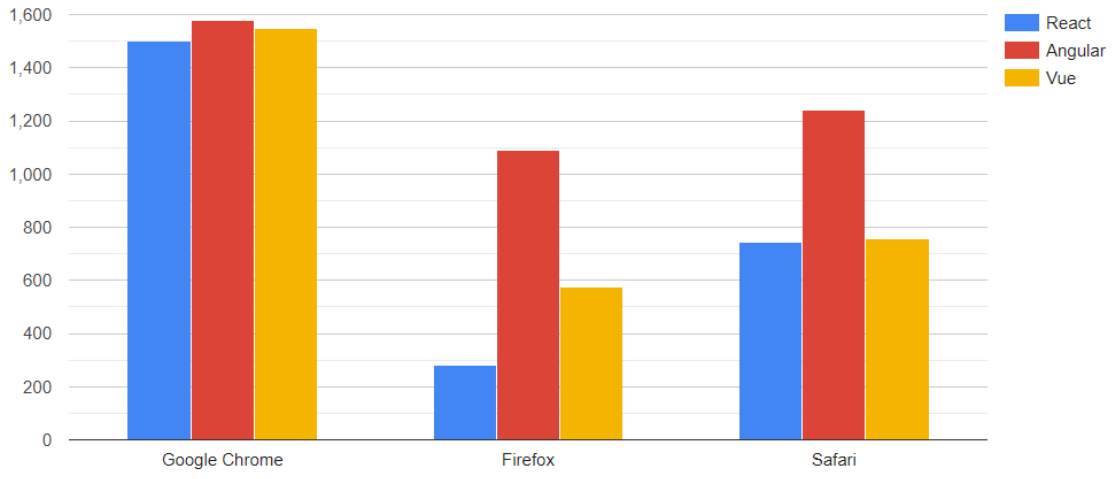

Рисунок 3.27 – Графік параметру finish

На рисунку 3.27 предсатвлено графік в якому відображаться дані параментру finish, проаналізувавши отримаємо висновок що найкращий

показник між веб-платформами має React, який завантажується найшвидше між всіма запропонованими веб-платформами.

VueJs є другим по швидкості завантаження сторінки, в той час як Angular є третім, при тому що в Firefox його швидкість більш завантаження повільніша більше ніж в 3 рази в порівнянні з React.

Параметр «DOMContentLoaded» показує час від заходу на сторінку до моменту, коли весь html показаний для користувача, при тому що картинки, стилі та шрифти можуть бути ще не завантажені, це дуже важливий показник, тому чим цей час менше тим краще. Під час аналізу буде створено ще одну таблицю з даними цього параметру.

Таблиця 3.3 – Значення параметру DOMContentLoaded

	React	Angular	VueJs
Google Chrome	167 ms	300 ms	184 ms
Firefox	93 ms	28 ms	220 ms
Safari	84 ms	115 ms	194 ms

По таблиці зображеній вище можна побачити що час загрузки по параметру «DOMContentLoaded» вже не є таким однозначним, і в кожному із браузерів цей найшвидший час різний для всіх платформ. Angular в браузері Firefox є найшвидшим, але водночас в Google Chrome це найдовша загрузка зі всіх платформ.

Тому якщо брати Google Chrome то най швидша платформа є React, для Firefox це є Angular, який завантажується за феноменальні 28 мілісекунди, потімна 65 мілісекунд більше потрібно для React платформи і більше ніж 100 секунддля VueJs. В Safari ж час завантаження знову ж таки найшвидший для React платформи.

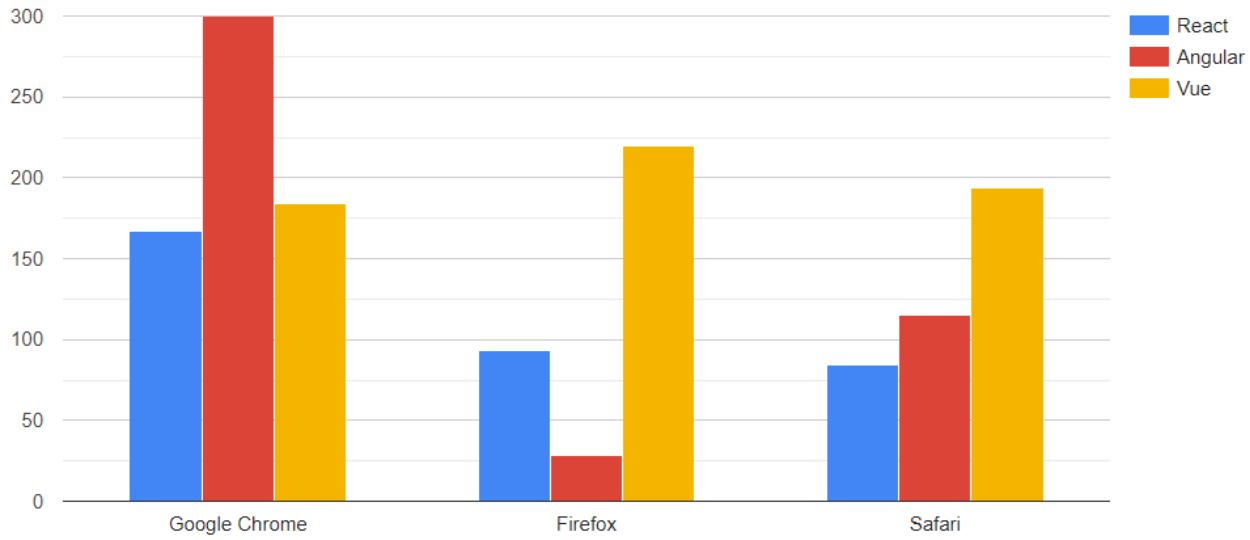

Рисунок 3.28 – Графік параметру DOMContentLoaded

Проаналізувавши графік зображений на рисунку 3.28 бачимо, що React найшвидший в двох із трьох браузерах, тому він на першому місці в цьому порівнянні, а Angular кращий тільки в Firefox браузері, тому він на другому, VueJS хоча не можна сказати, що завантаження є дуже довгим, але все ж таки він на третьому місці.

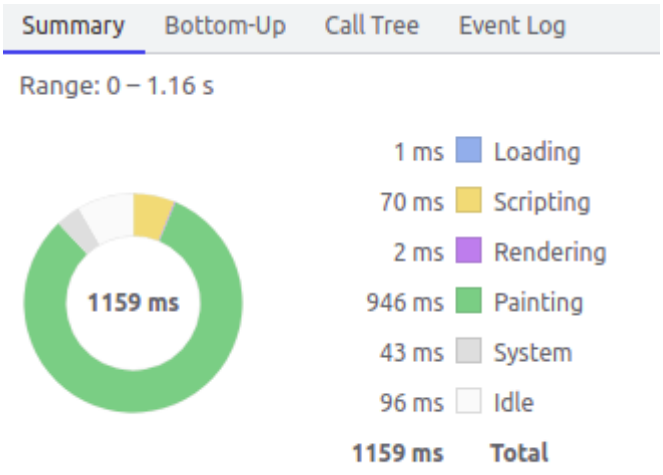

Рисунок 3.29 – React performance summary

Іншим методом, щоб побачити час завантаження сторінки – відкрити панель розробників на веб-сайті та перейти у вкладку Performance, де буде

описано час завантаження в деталях і можна буде бачити, які із частин проекту завантажуються найшвидше, а які потрібно оптимізовувати. Ця вкладка складається із багатьох корисних інформаціїйних вікон, те що потрібне зображене на рисунку 3.29, на ньому зображена інформація з підвкладки Summary. Нажаль ця вкладка доступна тільки в Google Chrome, але вона в покаже те що в інших браузерах побачити не можливо. Це інформація в деталях описує те, як швидко завантажуються ті чи інші частини сайту.

Кожний параметр відповідає за свою частину завантаження, та є важливий, ось опис те що за що вони відповідають:

Loading показує час який браузер просто чекає на якісь дії, в більшості випадків цей параметр є завжди маленьким по часі.

Scripting це час який витрачається на виконання коду, який написав девелопер, або коду самої платформи.

Rendering час та відображення всього згенерованого вигляду.

Painting це відмалювання та правильна побудова веб-сайту.

System це час потрібний для браузера, для того щоб він зрозумів що потрібно робити.

Idle це фактична бездія браузера, по різних причинах, але вона також є частиною першого завантаження сторінки.

Total це загальний витрачений час на завантаження сайту, тобто всі показники просумовані.

Для дослідження потрібно відкрити кожний розроблених проектів, та отримати інформацію з цієї вкладки, для кожної із платформ буде свої дані, тому що вони оптимізовані по різному. Кожне обновлення цієї вкладки може показати різний час завантаження, для точності було взяте середнє число за 15 перезавантажень, але даже з урахуванням того що час завантаження може бути різним кожне перезавантаження не міняло загальний час більше ніж на 100 мілісекунд, що є цілком точним результатом, оскільки при аналізі немає спірних моментів, яка із платформ швидша.

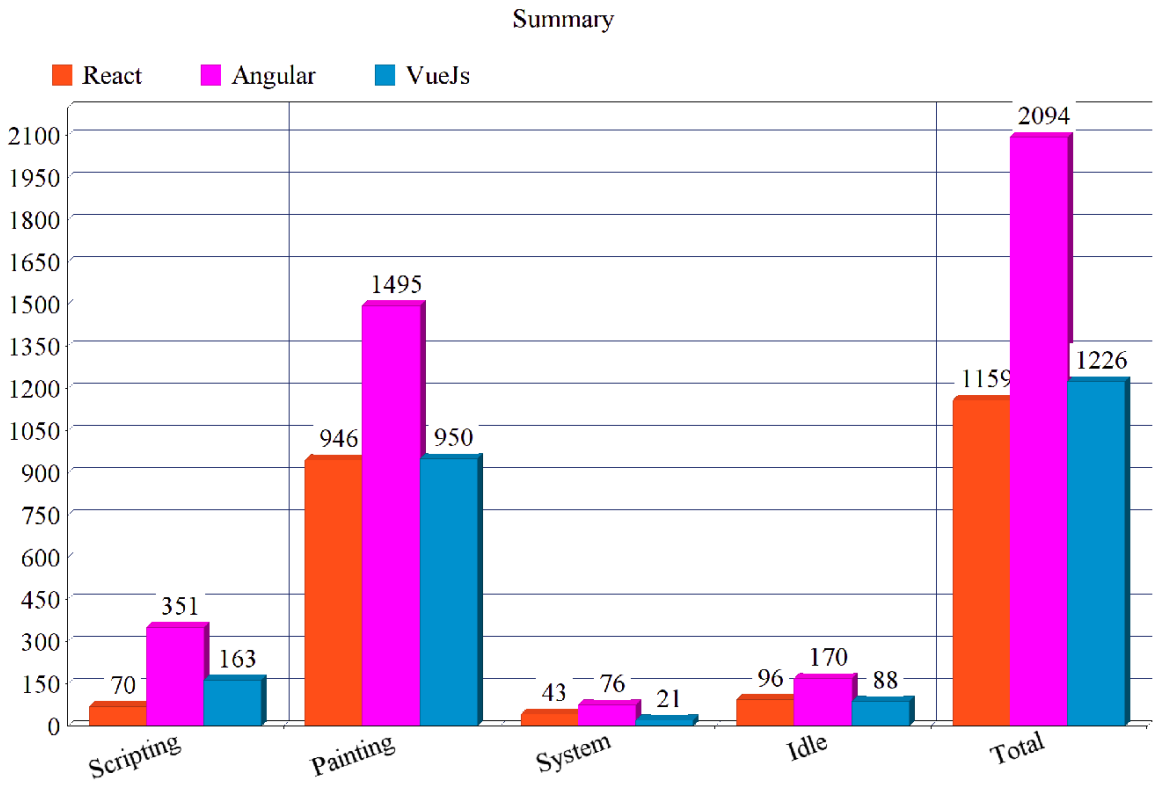

Рисунок 3.30 – Стовпчата діаграма порівняння Summary по платформах

На рисунку 3.31 зображена стовпчата діаграма даних по всіх платформах даного дослідження, які взяти із підвкладки Summary. На діаграмі не вказані Loading та Rendering, тому що ці завантаження займають менше 3 мілісекунд у всіх трьох випадках. На діаграмі чітко зображено всі часи завантаження сторінки і чим менший цей час, тим краще. Angular є найдовше завантажуваною платформою, вона завантажується довше за VueJs на 868 мілісекунд, а і на 935 мілісекунди довше ніж React, що є дуже погано оскільки, це всього лиш налаштований проект без додаткових бібліотек та з типовими елементами інтерфейсу. React та VueJs в свою чергу змагаються за перше місце і мають свої переваги в кожному із показників, і React в загальному швидший всього лиш на 67 мілісекунди.

Виходячи з попереднього аналізу веб-платформ в різних браузерах, можна зробити висновок що в даний час React є найефективнішою веб-платформою. Angular є найгіршою веб-платформою по ефективності, напевно саме тому ця платформа втрачає свою популярність в розробників, що можна

побачити в розділі 1.3 де була проаналізована популярність платформ. VueJs є другою по ефективності платформою, не набагато відстаючи від React, але ця платформа є відносно новою і не такою ж оптимізованою як React. Цілком можливо що в майбутньому VueJs буде най ефективнішою платформою, але це залежить тільки від розробників даних платформ.

Дане дослідження не є ідеально точним та може містити деякі відхилення, через багато факторів, таких як досвід розробника на різних платформах (мовах, фреймворках), ефективність платформи в залежності від комп'ютера на якому проводилось дослідження, швидкості інтернету в момент отримання даних, та багато іншого. Хоч дослідження може містити відхилення, результат аналізу все рівно буде залишатись в такому ж діапазоні точності.

Висновок до розділу 3

Результати дослідження проведених в розділі 3 можна розділити на три частини, виходячи з аналізу кожного із підрозділів.

Було проведено порівняння часу та простота створення проекту на кожній із платформ. За цим параметром було визначено, що перше місце займає React, друге –Angular, і третє–VueJs.

При аналізі простоти та зручності розробки проекту було отримано результат, що React та VueJs займають перше місце, в той час як Angular є найгіршим в цьому плані.

А при аналізі ефективності кожної із платформ було виявлено, що найшвидшою на даний момент є React, не набагато відстаючи по ефективності є VueJs, а Angular є найгіршим із запропонованих платформ.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було проаналізовано ефективність застосування різних платформ для створення FRONT END з типовими елементами інтерфейсу в різних браузерах.

В ході роботи над кваліфікаційною роботою було чітко поставлені вимоги до розроблюваних проектів, та до методів аналізу ефективності.

Наступним кроком стало вибір необхідних для аналізу браузерів, які були обрані за свою популярність в світі серед людей, і цими браузерами стали GoogleChrome, Firefoxта Safari.

Також проаналізовано існуючи веб–платформи та вибрані найпопулярніші на даний момент серед розробників, а саме: React, Angular та VueJs.

Було розглянуто аналоги досліджень проведені для даних платформ та застосовано методи порівняння аналогів з аналізом кваліфікаційної роботи. За допомогою цього методу описано плюси та мінуси існуючих досліджень та проаналізовано цінність аналізу виконаного в цій кваліфікаційній роботі.

Також описано плюси та мінуси обраних веб–платформ, та необхідні для їх розробки інструменти.

Розроблено структуру проекту та створено блок–схеми і діаграми, які в деталях описують обрану структуру. Для опису структури описано такі діаграми:

- діаграма компонентів;
- діаграма розгортання;
- діаграма станів;
- діаграма прецедентів.

Також описана архітектура та структура проектів для всіх трьох платформ, і обрана така, що найкраще підходить для всіх.

В результаті дослідження було проаналізовано такі частини життєвого циклу проекту:

- створення проекту – в якому було проаналізовано час та простоту створення і виявлено те, що Reactє найпростіший в цьому плані, Angularне набагато важчий, а от VueJsє найважчим із трьох;
- розробка проекту – в якому було проаналізовано зручність та простота розробки, де немає однозначно найкращої платформи між React та VueJs, але найгіршою є Angular;

Та головне дослідження даної роботи, це порівняння ефективності веб-платформи в різних браузерах. Результатом цього дослідження є висновок, який проведено з даних, які було взято під час аналізу ефективності в браузерах.

Тому можна зробити такий висновок, що найефективнішою веб-платформою в даний час є React, на другому місці є VueJs, і на третьому – Angular.

І хоча дане дослідження може містити відхилення, але порядок найефективніших платформ буде точно таким же самим, в не залежності від відхилень.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Розробка з боку фронт-енд URL: https://dan-it.com.ua/uk/blog/rozrobka-z-boku-front-end-shho-ce-take-i-chim-vidriznjaietsja-vid-back-end/ (дата звернення 10.03.2021)

2. Ефективність URL: https://uk.wikipedia.org/wiki/Ефективність (дата звернення 11.03.2021)

3. Браузер URL: https://uk.wikipedia.org/wiki/Браузер (дата звернення 13.03.2021)

4. Статистика браузерів URL: https://gs.statcounter.com (дата звернення 16.03.2021)

5. Статистика браузерів за 2021р. URL: https://gs.statcounter.com/browser-market-share#monthly-202009-202109-bar (дата звернення 18.03.2021)

6. Веб-платформа URL: https://uk.wikipedia.org/wiki/Веб_платформа (дата звернення 20.03.2021)

7. Angular vs Vue vs React URL: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/ (дата звернення 1.04.2021)

8. Angular vs Vue vs React best choice URL: https://medium.com/hackernoon/angular-vs-react-vs-vue-which-is-the-best-choice-for-2019-16ce0deb3847 (дата звернення 5.04.2021)

9. Angular vs Vue vs React best choice URL: https://hackernoon.com/angular-vs-react-vs-vue-which-is-the-best-choice-for-2019-16ce0deb3847 (дата звернення 7.04.2021)

10. Npm URL: https://uk.wikipedia.org/wiki/Npm (дата звернення 7.04.2021)

11. JSS URL: https://cssinjs.org/jss-api/ (дата звернення 8.04.2021)

12. SASS URL:https://uk.wikipedia.org/wiki/Sass (дата звернення 9.04.2021)

13. React URL: https://uk.wikipedia.org/wiki/React (дата звернення 11.04.2021)

14. Angular URL: https://uk.wikipedia.org/wiki/Angular_(фреймворк) (дата звернення 14.04.2021)

15. Діаграма розгортання URL: https://uk.wikipedia.org/wiki/Діаграма_розгортання (дата звернення 15.04.2021)

16. Діаграма станів URL: https://uk.wikipedia.org/wiki/Діаграма_станів_(UML) (дата звернення 16.04.2021)

17. Клієнт-серверна архітектура URL: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура (дата звернення 22.04.2021)

18. Уніфікована мова програмування URL: https://bestreferat.ru/referat-142716.html (дата звернення 23.04.2021)

19. Діаграма класів URL: https://www.ua5.org/oop/392-diagrami-klasiv.html (дата звернення 25.04.2021)

20. Модель Вид Контроллер URL: https://uk.wikipedia.org/wiki/Модель-вид-контролер (дата звернення 2.05.2021)

21. Дэвид Сойер Макфарланд. JavaScript и jQuery. Исчерпывающее руководство. Пер. с англ. М.: Издательский дом «Вильямс», 2017. 450 с.

22. Мэтью Макдональд, Веб-разработка. 2-е изд.: Пер. с англ. М.: Издательский дом "Вильямс", 2015. 640 с.

23. Дэвид Флэнаган, JavaScript – Подробное руководство 6-ое издание Пер. с англ. М.: Издательский дом «Вильямс», 2015. 300 с.

24. Bonnie Eisenman Learning React Native 2nd Edition, 2018. 420 с.

25. Алекс Бэнкс, Ева Порселло, Reactи Redux функциональная веб-разработка. Видавник: «Питер» 2018. 336 с.

26. Величко С.В., Межов Е.В. Современные СУБД для создания единой информационной среды в больших информационных системах. // Вестник воронежского гос. техн. университета. 2003. № 3. С.68-73.

27. Карпова Т.С. Базы данных. Модели, разработка, реализация. Издательский дом «Питер», 2001. 303 с.

28. HTTP-cookie. Матеріал з Вікіпедії вільної енциклопедії. . URL: https://uk.wikipedia.org/wiki/cookie (дата звернення: 15.03.2019).

29. JavaScript. Матеріал з habr. URL: https://habr.com/ru/company/ruvds/blog/429552/ (дата звернення: 17.03.2019).

30. Семчук А.Р., Юрченко І.В. Економічна інформатика. Навчальний посібник.– Чернівці: МВІЦ "Місто", 2008. 426 с.

31. HTML [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/HTML. (дата звернення 5.06.2021)

32. HTML [Електронний ресурс]. – Режим доступу: https://developer.org/uk/docs/Web/HTML. (дата звернення 7.06.2021)

33. Основы Sass [Електронний ресурс]. – Режим доступу: https://sass-scss.ru/guide/. (дата звернення 7.06.2021)

34. Развёрнутое руководство по Sass/SCSS [Електронний ресурс]. – Режим доступу: https://tproger.ru/translations/complete-sass-guide/. (дата звернення 7.06.2021)

35. CSS [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/CSS. (дата звернення 7.06.2021)

36. Справочник CSS [Електронний ресурс]. – Режим доступу: htmlbook.ru/css. (дата звернення 7.06.2021)

37. Основи CSS [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/uk/docs/Learn/Getting_started_with_the_web/CSS_basics. (дата звернення 7.06.2021)

38. Методология БЭМ [Електронний ресурс]. – Режим доступу: https://ru.bem.info/methodology/.(дата звернення 8.06.2021)

39. JavaScript [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/JavaScript. (дата звернення 8.06.2021)

40. Современный учебник JavaScript [Електронний ресурс]. – Режим доступу: https://learn.javascript.ru/.(дата звернення 8.06.2021)

41. React [Електронний ресурс]. – Режим доступу: https://uk.reactjs.org/.(дата звернення 10.06.2021)

42. Вступ до JSX [Електронний ресурс]. – Режим доступу: https://uk.reactjs.org/docs/introducing-jsx.html. (дата звернення 10.06.2021)

43. JSX подробности [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/3192