МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ

ІМ. Ю.М. ПОТЕБНІ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему **Порівняльний аналіз Web-фреймворків**

Виконав: студент 2 курсу, групи 8.1210-2іпз
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
Гулак Є.І.
(ініціали та прізвище)
Керівник доцент, канд. техн. наук Міхайлуца О.М.
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент Директор ТОВ «Дісітел» П.О.Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ

ІМ. Ю.М. ПОТЕБНІ

Кафедра _____ програмного забезпечення автоматизованих систем
Рівень вищої освіти _____другий (магістерський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Інженерія програмного забезпечення
(код та назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри_В.Г. Вербицький
"_01_" __вересня_____2021_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Гулаку Євгенію Ігоровичу
(прізвище, ім'я, по батькові)

1.Тема роботи Порівняльний аналіз Web-фреймворків

керівник роботи __Міхайлуца О.М., доцент, канд. технічних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від _"30"_____червня 2021 року № 974-с

2. Строк подання студентом кваліфікаційної роботи___30.11.2020_____
3. Вихідні дані магістерської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження та аналіз актуальних фреймворків;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- дослідження поставленої проблеми та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
___слайдів презентації

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата
		Завдання прийняв

7. Дата видачі завдання 1.09.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	02.09-10.09.21	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	11.09-12.09.2021	виконано
3	Аналіз існуючих методів рішення	13.09-17.09.20	виконано
4	Дослідження та аналіз фреймворків	18.09-24.09.20	виконано
5	Узгодження подальших дій з науковим керівником	25.09-26.09.20	виконано
6	Аналіз теоретичних відомостей	27.08-15.09.20	виконано
7	Проектування застосунку на двох фреймворках	15.09-23.09.20	виконано
8	Узгодження інтерфейсу та архітектури з науковим керівником	23.09-24.09.20	виконано
9	Реалізація функціоналу та інтерфейсу	25.09-10.10.20	виконано
10	Представлення отриманих результатів науковому керівнику і узгодження плану подальшого дослідження	11.10-13.10.20	виконано
11	Тестування та опис застосунку	14.10-20.10.20	виконано
12	Проведення порівняльного аналізу розроблених застосунку	21.10-22.11.21	виконано
13	Оформлення звіту	23.11-30.11.21	виконано

Студент _____ Гулак Є.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Міхайлуца О.М.
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ Скрипник І.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок: 87

Рисунків: 51

Джерел: 26

Гулак Є.І. Порівняльний аналіз фреймворків та Anqular та React.

Кваліфікаційна робота для здобуття ступеня вищої освіти магістра за спеціальністю 121 — Інженерія програмного забезпечення, науковий керівник Міхайлуца О.М. Інженерний навчально-науковий інститут Запорізького національного університету.

Завданням науково - дослідницької роботи є дослідження можливостей фреймворків, та порівняння їх характеристик в певних умовах. Для досягнення мети потрібно створити програмний застосунок на обох фреймворках та виконати їх порівняння.

Мета роботи — порівняння та аналіз можливостей фреймворків «Anqular» та «React».

Методи досліджень — порівняння та аналіз.

Результатом роботи є комп'ютерний SPA додаток «Оренда автомобілів»

Галузь застосування охоплює різні категорії людей та установ що в майбутньому зацікавлені в перспективі розвитку класичних програмних застосунків.

Ключові слова: *web-фреймворки, фреймворки, angular, react, spa, react virtual dom, mvvm-модель, mvc-модель, css, html.*

SUMMARY

Pages: 87

Figures: 51

Sources: 26

Hulak E.I. Comparative analysis of frameworks and Anqular and React.

Qualification work for obtaining a higher education degree of a master in specialty 121 — Software Engineering, scientific adviser O. M. Mikhailutsa. Engineering educational and scientific institute of ZNU, 2021.

The task of research work is to study the possibilities of frameworks, and compare their characteristics in certain conditions. To achieve this goal, you need to create a software application on both frameworks and compare them.

The purpose of the work is to compare and analyze the capabilities of the Anqular and React frameworks.

Research methods — comparison and analysis.

The result is a computer SPA application "Car Rental"

The field of application covers various categories of people and institutions that are interested in the future development of classic software applications.

Key words: *web-frameworks, frameworks, angular, react, spa, react virtual dom, mvvm-model, mvc-model, css, html.*

ЗМІСТ

ВСТУП

Актуальність теми

Розвиток Інтернету повністю пов’язаний з системою проектування сайтів. Масова поява сайтів спричинила проблему їх якості. Власники веб-додатків зацікавлені в поліпшенні і підтримці своїх сайтів. Будучи веб-розробником, потрібно розвиватися з тенденціями в технологіях.

Малий бізнес, банки і великі промислові підприємства залежать від веб-додатків. Користувач в пошуках продукту або послуги, зазвичай відвідує веб-сайт компанії. Завдяки розвитку веб-технологій державні послуги підвищилися як з точки зору доступності, так і з точки зору якості.

Для ефективного вирішення завдань, пов'язаних з розробкою веб-додатків, широке застосування отримали веб-фреймворки. Дана технологія дозволяє:

використовувати готові універсальні компоненти, скорочує дублювання коду, спрощує налагодження і підвищує надійність готового програмного продукту.

Як відомо, існує безліч фреймворків, включаючи Ember, Node, Polymer, Meteor і багато іншого, але за даними аналітиків Google Trends, Angular і React є найпопулярнішими веб-технологіями в 2020 і 2021 роках.

Не варто переоцінювати важливість вибору конкретного інструменту - детальне вивчення одного полегшить розуміння інших. Так чи інакше, вивчаються не тільки особливості, характерні для одного фреймворка, але і загальні концепції, застосовні до веб-розробки в цілому: проектування компонентів, розуміння потоку даних, управління власністю, шаблонами.

Вибір між двома інтерфейсними платформами, такими як Angular або React для проектів, є предметом суперечок. Головна причина порівняння полягає в тому, аби мінімізувати плутанину. Однак не можливо визначити, що є найкращим, але можливо проаналізувати, що кращим в конкретних ситуаціях.

Мета і завдання дослідження

Метою дослідження є порівняння двох фронтенд-фреймворків Angular та React на прикладі готового веб-додатку, а також визначення найбільш підходящого інструменту для певної сфери веб технологій.

Об’єкт дослідження

Об’єктами дослідження є фронтенд-фреймворки Angular та React.

Предмет дослідження

Веб-сайт з оренди автомобілів.

Методи дослідження

Для порівняння представлених фреймворків використовуються такі критерії для аналізу та порівняння:

- Дослідження об’єктів та предметів.
- Порівняльний аналіз програмних продуктів.
- Порівняння, встановлення подібностей, виявлення відмінностей та встановлення загальностей.
- Аналіз отриманих результатів досліджень.

Наукова новизна одержаних результатів

На основі створеного за стосунку оцінено мову, зручність і продуктивність, а також прийняття рішення. Обидві технології мають свій набір переваг і подібностей, і це дійсно зводиться до того, який функціонал буде пропонувати додаток.

Практичне значення одержаних результатів

З огляду на розміри кожного фреймворка і бібліотеки, React підходять для легких додатків, в той час як Angular добре підходить для складних і важких додатків. Через велику кількість налаштованих інструментів з коробки розробка веб-додатків швидше і простіше в Angular ніж в React,. Однак, архітектура React простіша в масштабуванні, ніж в Angular.

Angular — це повний набір інструментів, який пропонує всі інструменти від розробки до тестування при створенні веб-додатків, в той час як React — це гнучка бібліотека, яка вимагає підтримки інших бібліотек для розробки.

Апробація одержаних результатів

Результати дослідження були представлені на науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених Запорізь-кого

національного університету «Молода наука-2021» [1], а також на на-уково-технічній конференції студентів, аспірантів, магістрантів і викладачів Інженерного навчально-наукового інституту Запорізького національного університету [2].

Глосарій

Веб-сайт — це набір пов'язаних веб-сторінок з інформацією про вашу компанію, продукт, послугу. Вони зазвичай пояснюють деталі про ваш бізнес, а точніше пояснюють те, що робиття, а також продукти та послуги, які пропонуються потенційним клієнтам. Основна функція веб-сайту полягає в тому, щоб описати бізнес зацікавленим користувачам і в багатьох випадках запропонувати продукт або послугу.

Інтерфейс — це сукупність засобів, методів та правил, призначених для взаємодії елементів системи (або цілих систем) між собою.

Цільова сторінка — це єдина, окрема веб-сторінка, призначена для заохочення користувачів до своєї пропозиції. Це може бути що завгодно від електронних книг або купонів на знижки до безкоштовних пробних версій або демонстраційних версій.

Фреймворк — програмне забезпечення, що полегшує розробку та об'єднання різних компонентів великого програмного проекту.

Бібліотека — це сукупність функцій і функціональних можливостей, які можна використовувати для досягнення певної мети.

Document Object Model (DOM) — об'єктна модель даних веб-застосунку.

HTML(HyperTextMarkupLanguage)- стандартизована мова розмітки документів для перегляду веб-сторінок у браузері.

CSS (Cascading Style Sheets) — **мова, яка** використовується для форматування зовнішнього вигляду різних структурних елементів або стилізації чи дизайну.

JavaScript — мова програмування, яка використовується для опису функціональності та обробки всіх динамічних елементів на веб-сторінці.

API (Application Programming Interface) — це набір функцій, який дозволяє програмам отримувати доступ до даних і взаємодіяти з зовнішніми компонентами програмного забезпечення, операційними системами або мікросервісами.

Model-view-controller (MVC) —— поділу даних програми, і логіки, що управляє, на три окремих компоненти: модель, уявлення і контролер — таким чином, що модифікація кожного компонента може здійснюватися незалежно. Модель надає дані та реагує на команди контролера, змінюючи свій стан.

Архітектура — це базова організація системи, втілена в її компонентах, їхніх відносинах між собою і з оточенням, а також принципи, що визначають проектування та розвиток системи програми, видимі ззовні властивості цих елементів і зв'язки між ними.

Мова — це штучна мова, створена для передачі команд машинам, зокрема комп'ютерам. Мови програмування використовуються для створення програм, які контролюють поведінку машин, та для запису алгоритмів.

Прив'язка даних — процес приєднання даних в одному об'єкті до іншого об'єкту. Він забезпечує зручний шлях передачі даних між різними рівнями застосунку. Прив'язка даних вимагає властивості джерела, властивості адресату та події запуску, яка вказує, коли копіювати дані з джерела прив’язки до адресату прив’язки. Об'єкт здійснює запущену подію, коли змінюється властивість джерела.

РОЗДІЛ 1 ОГЛЯД І АНАЛІЗ СУЧАСНИХ ФРЕЙМВОРКІВ

1.1 Сучасні види фреймворків

Фреймворк — програмна структура, будь то фронтенд, або бекенд, включає стандартизований, попередньо написаний код, що робить розробку певних функціональних можливостей простішою і швидшою. Фреймворк вказує, як потрібно кодувати і якою повинна бути система архітектури, тобто розташування файлів та взаємодія частин коду між собою. Отже, це є багаторівнева структура, яка вказує, як програми можуть або повинні бути побудовані, і як їх компоненти будуть взаємозв'язані. Фреймворки можуть включати в себе програми підтримки, компілятори, бібліотеки коду, набори інструментальних засобів і інтерфейси прикладного програмування (API), які об'єднувати всі різні компоненти, що забезпечують розробку проекту або системи. Проектувальники фреймворків мають на меті полегшити розробку програмного забезпечення, дозволяючи розробникам програм присвятити свій час розробці програмного забезпечення, а не працювати з більш стандартними інструментами низького рівня робочої системи, тим самим скорочуючи загальний час розробки [3].

На сьогоднішній день існують такі види фреймворків:

- Фреймворки додатків.
- Фреймворки концептуальної моделі.

Фреймворк додатків (Application framework) — це бібліотека програмного забезпечення, яка забезпечує фундаментальну структуру для підтримки, розробки додатків для певного середовища. Фреймворк додатків виступає як опора для створення програми. Мета розробки фреймворків додатків полягає в тому, щоб зменшити загальні проблеми, з якими стикаються розробники під час розробки додатків. Це досягається за допомогою використання коду, який можна застосовувати в різних модулях програми. Фреймворки додатків використовуються не тільки при розробці графічного інтерфейсу користувача (GUI), а й в інших областях, таких як веб-додатки. Фреймворки додатків діють як інструменти для забезпечення структури та шаблонів для побудови програми.

Фреймворк концептуальної моделі — це абстрактне поняття даної структури для визначення способів вирішення конкретної проблеми [4].

1.2 Архітектура шаблону MVC (Model-View-Controller)

Модель-Вид-Контролер (Model-View-Controller) — це шаблон проектування, в якому модель даних програми, інтерфейс користувача і керуюча логіка розділені на три окремих компоненти так, що модифікація одного з компонентів надає мінімальний вплив на інші.

Архітектурні компоненти шаблону MVC призначені для обробки різних аспектів застосування в розробці програмного забезпечення. Шаблон дизайну MVC служить для відділення презентаційного шару від бізнес-логіки.MVC популярний в розробці додатків і веб-сайтів, і це один з найбільш широко використовуваних моделей розробки програмного забезпечення для розробки додатків і веб-сайтів. MVC – шаблон проектування розділяє проблеми в одному з трьох шарів (Рис.1).

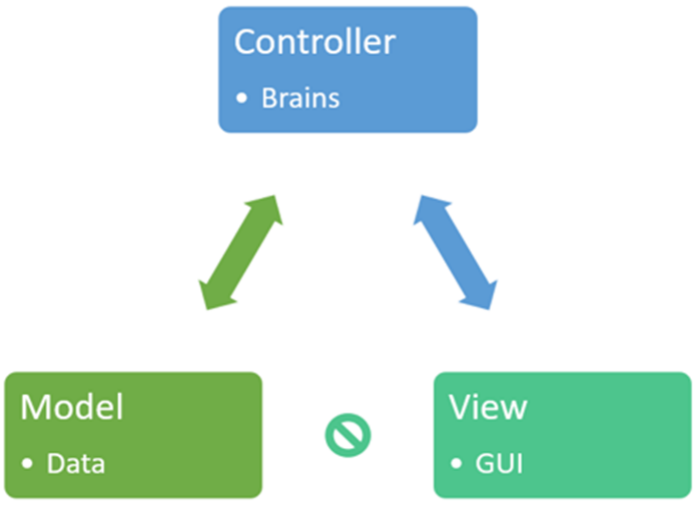

Рис.*1 Шаблон проектування MVC*

Модель: зберігає та керує даними.

Вид: графічний інтерфейс користувача. Вид є візуальним представленням або поданням даних, таких як діаграма, таблиця, форма. Вид містить усі функціональні можливості, які безпосередньо взаємодіють з користувачем - наприклад, натискання кнопки або подія введення тексту чи символів.

Контролер — це мізки програми. Контролер підключає модель і вид. Контролер перетворює вхідні дані з подання на вимоги до отримання або оновлення даних у моделі. Контролер отримує вхід з поля зору, використовує логіку для перекладу вводу на вимогу до моделі, модель захоплює дані, контролер передає дані з моделі назад у вигляд, щоб користувач побачив на дисплеї.

MVC популярний у веб-додатках, одна з причин полягає в тому, що обов'язки розподіляються між клієнтом і сервером (Рис.2).

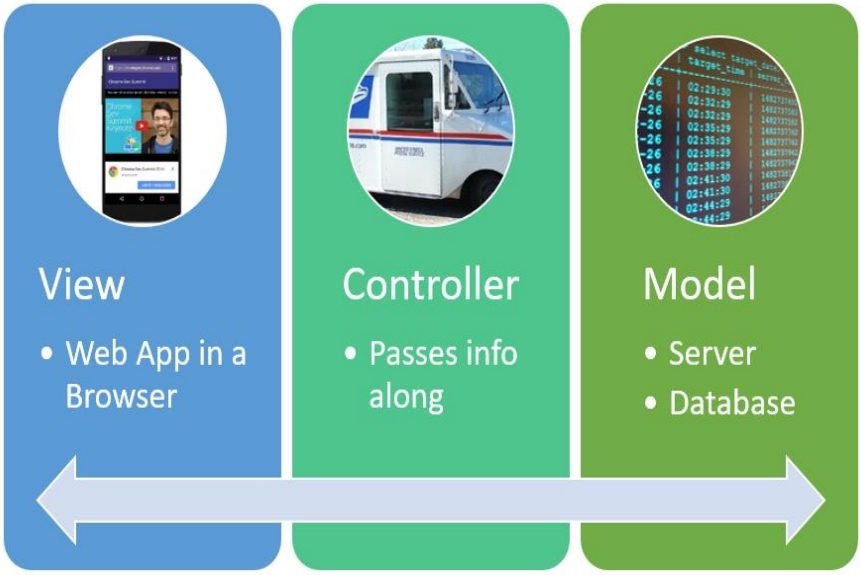

Рис. *2 Веб-додаток MVC*

Переваги MVC

- Використовується для графічних інтерфейсів користувача.

- Популярний у веб-додатках.
- MVC корисний шаблон проектування при плануванні розробки.
- Розділення проблем на шари (тобто цей код ділиться на основі функції для моделі, виду або контролера).
- Добре працює з Ruby on Rails.
- Вільно з'єднаний.
- Видалення непотрібних залежностей.
- Повторне використання без модифікації.
- MVC робить модельні класи багаторазовими у використанні без модифікації.
- Повторне використання коду.
- Код масштабується.
- Висока згуртованість або компактність.
- Підтримка кількох подань.
- Кожна частина може бути протестована.

Недоліки MVC:

- Необхідність використання великої кількості ресурсів.
- Ускладнений механізм розподілу програми на модулі.
- Ускладнено процес розширення функціоналу.

Конструкція MVC дозволяє розділити проблеми — розділяючи логіку між трьома шарами, так що кожен шар може діяти самостійно. MVC слабо з'єднаний, це означає, що кожен шар, а саме: модель, вид і контролер, діють незалежно один від одного.

Розробники можуть модифікувати одну з частин, а інші дві частини повинні продовжувати працювати і не вимагати модифікацій. При розробці програмного забезпечення MVC логіка в кожному з трьох шарів є незалежною. Все в поданні діє незалежно від моделі і контролера, подання не буде мати ніякої бізнес логіки, залежної від моделі.

Створення незалежних моделей і переглядів робить організацію коду простою і легкою для розуміння і полегшує обслуговування. Програмісти можуть виправити помилку в поданні, не змінюючи код моделі [5].

1.3 Архітектура шаблону MVVM (Model-View-ViewModel)

MVVM — це уточнення дизайну MVC, за допомогою якого ViewModel в MVVM полегшує розділення розробки графічного інтерфейсу користувача (UI), тобто презентаційного шару. ViewModel (VM) відповідає за перетворення об'єктів даних з моделі таким чином, щоб об'єкти легше керувались та представлялись. У цьому відношенні ViewModel є більш моделлю, ніж видом, і обробляє більшість, якщо не всю бізнес-логіку відображення моделі. MVVM — це варіація моделі презентації Мартіна Фаулера в тому, що модель презентації абстрагує або відокремлює модель таким чином, що вона не залежить від конкретної платформи інтерфейсу користувача. Отже, ні вид, ні шар моделі не знають один про одного. Приклад роботи шаблону зображено на рисунку 3.

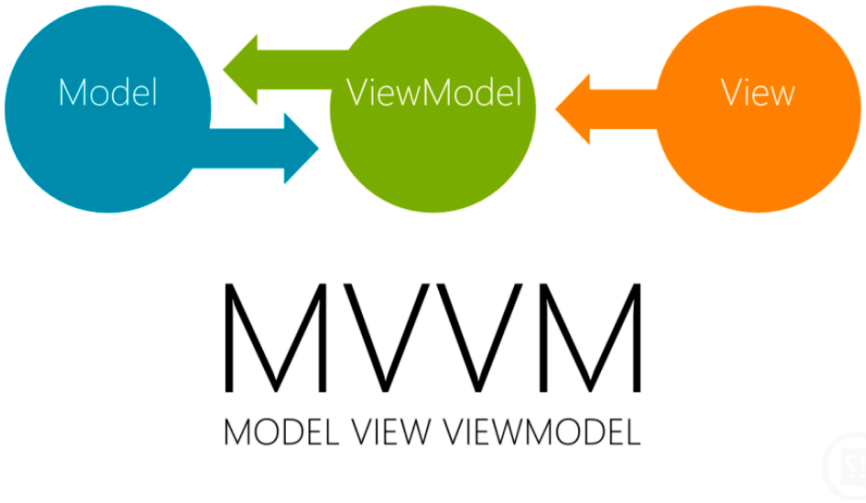

Рис.*3 Шаблон проектування MVVM*

Плюси :

- Компоненти слабо зв’язані.
- Databinding зменшує кількість коду.
- До однієї ViewModel можна прив’язати декілька View.

Минуси:

- Показ Toast та діалогів, які займають багато місця на екрані.
- Показ Анімації чи будь яких даних з затримкою.
- Необхідність обробки команд в View.

1.4 Аналіз front-end фреймворків

Розробка front-end — це стиль комп’ютерного програмування, який зосереджується на кодуванні та створенні елементів і функцій веб-сайту, які будуть бачити користувачі. Мова йде про те, щоб візуальні аспекти веб-сайту були функціональними. Робота розробника полягає в тому, щоб кодувати та втілювати в життя візуальні елементи веб-сайту. Розробники більше зосереджені на тому, що бачить користувач, коли відвідує веб-сайт або додаток. Крім того, слід переконатися, що з сайтом легко взаємодіяти, а також безперебійно працювати. Ці розробники беруть візуальний дизайн від UX і UI дизайнерів і оживляють веб-сайт, переконавшись, що він добре функціонує для користувача. Одним із багатьох способів використання навичок інтерфейсу є створення статичного веб-сайту, який є веб-сайтом із фіксованим вмістом та доставляється у браузер користувача точно так, як зберігається. Він можете зіткнутися зі статичним веб-сайтом**,** якщо натрапити на просту цільову сторінку або веб-сайт малого бізнесу, який не дозволяє користувачам виконувати будь-які інтерактивні завдання.

Розробники інтерфейсу створюють такі елементи, як:

- Кнопки.
- Макети.

- Навігація.
- Зображення.
- Графіка.
- Анімації.
- Організація контенту.

Для створення адаптивного дизайну використовуються CSS фреймворки. Вданий час існують десятки таких фреймвоків, кожний з яких має свої переваги й недоліки, у деяких з них гарний функціонал , але вони мають велику вагу. Розглянемо найпопулярніші з них [6].

1.4.1 Огляд сучасних front-end фреймворків

React – відкрита бібліотека JavaScript для створення односторінкових веб додатків, суть якої полягає в роботі з Shadow Dom та швидким оновленням DOM дерев на стороні клієнта(SPA) (Рис.4).

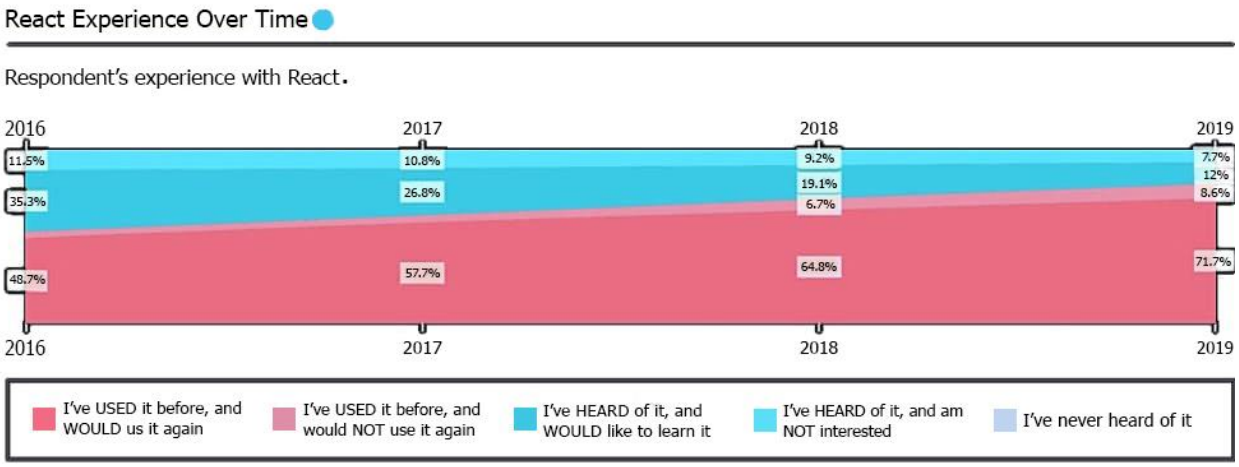

Рис.*4 Графік завантаження React*

Цей інтерфейсний фреймворк працює швидше за інших. Завдяки цьому програмісти можуть заощадити багато часу та стати більш ефективними.

React плюси:

- Хороша продуктивність, оскільки він базується на віртуальній моделі DOM.
- Його компоненти можна повторно використовувати у різних частинах програми.
- Велика спільнота постійно розвивається та робить фреймворк стабільним.
- Він універсальний і може використовуватись з будь-якими фреймворками.
- Односторонній потік даних.

Мінуси React:

- Використання складного JSX змушує розробників та дизайнерів скаржитися .
- Його динаміка може іноді впливати на пошукову оптимізацію Google (SEO).
- Підтримує лише результати зовнішнього інтерфейсу.

Angular — це фреймворк для розробки односторінкових додатків від Google (Рис. 5).

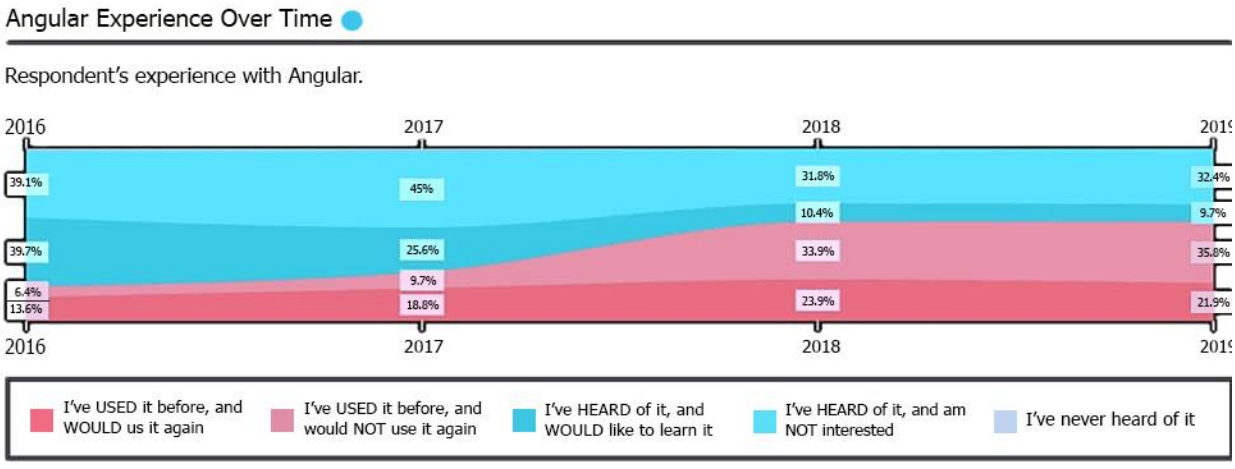

Рис.*5 Графік завантаження Angular*

Така структура використовує двосторонню прив'язку даних. При цьому забезпечується динамічна синхронізація даних між рівнем представлення та рівнем моделі даних в архітектурі MVW. Angular побудований виключно на TypeScript, що забезпечує плавну та ефективну роботу. Іншими словами, TypeScript – це основна мова Angular, це кросплатформний фреймворк. Використовуючи Angular, програмісти можуть створювати веб-сайти, веб-програми, мобільні програми та настільні програми.

Angular плюси:

- Автоматично вловлює зміни на рівні моделі, самостійно змінюючи код HTML.
- Прості рішення для тестування на основі Angular підтримують інтеграцію модульних тестів.
- Впровадження залежностей забезпечує просте перенесення та взаємодію між усіма компонентами складної системи.
- Angular відмінно працює із зовнішніми бібліотеками, такими як jQuery, UnderscoreJS або Ionic framework.
- Вбудований зв'язок REST через $http та $resource.

Angular мінуси:

- Важко вивчити складні процеси. Це не найпростіший фреймворк для новачків, які не мають досвіду роботи з JavaScript та TypeScript.
- Під час відкриття програми всі скрипти завантажуються одночасно. Це впливає на ефективність та швидкість роботи програмістів.
- Для повної індексації сайту потрібно використовувати зовнішні інструменти, які повертають правильний HTML-код [7].

Vue.js — фрейморк для створення веб-інтерфейсів який має двосторонню прив'язку даних (Сервер-клієнт), і візуалізацію на стороні сервера. Він є кращим вибором для швидкої розробки крос-платформних додатків.

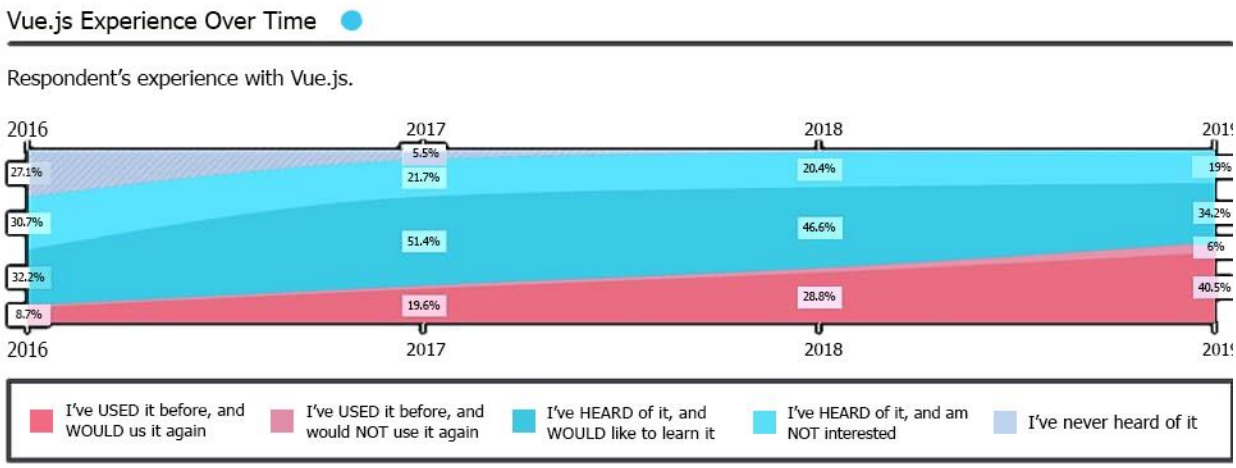

Рис.*6 Графік завантаження Vue.js*

Даний фреймворк працює як універсальний JavaScript-фреймворк, поєднуючи в собі інструменти з Angular, React та Ember. Він легший за своїх попередників. Така структура працює з віртуальною DOM та використовує двосторонню прив'язку. Все це робить Vue.js складним та сучасним рішенням для розробки складних веб-додатків.Vue.js заснований на відкритому вихідному коді.

Плюси Vue.js:

- Серверний рендеринг на основі того, що надає Angular2 та React.
- Швидкість та ефективність як для односторінкових, так і для багатосторінкових проектів.
- Інтуїтивно зрозумілі функції та простий синтаксис роблять фреймворк простим у освоєнні для новачків.
- Підтримує еластичне та гнучке програмування на основі зрозумілого та читаного коду.

- Підтримує використання машинопису.

Мінуси Vue.js:

- Невелика підтримка. Vue.js. є менш популярним для просунутих програмістів, які звикли до фреймворків JavaScript.
- Гнучкість. Це як перевага так і мінус Vue.js. Вибір багатьох компонентів може викликати проблеми. Особливо під час реалізації масштабного проекту, над яким працює багато програмістів.
- Мовний бар'єр. Багато нових механізмів і плагінів, що використовуються в Vue.js, написані китайською мовою.

Backbone — JavaScript фреймворк, який допоможе в створенні односторінкових додатків, вирішуючи деякі основні проблеми, що виникають при їх розробці (Рис.7). Backbone є легким в освоєнні інтерфейсним середовищем, яке дозволяє програмістам розробляти клієнтські веб-додатки та мобільні додатки. Він приймає імперативну парадигму програмування під час роботи з DOM. Отже, Backbone складається з команд, які повинні виконувати програми. Backbone.js сумісний із REST API, що робить його бездоганним для синхронізації між бек та фронт фреймворками.

Рис.*7 Графік завантаженняBackbone*

Backbone.js плюси:

- Забезпечує швидку та плавну роботу.
- Заснований на бібліотеці з відкритим вихідним кодом, має понад 100 користувацьких робочих розширень.
- Його моделі та колекції сумісні з архітектурою RESTful. Є можливість зручно отримати дані з сервера.
- Дуже чуйний. Усі зміни коду можна відразу побачити у додатку.
- Backbone.js не має готових структур. У ньому є лише кілька дуже простих інструментів для розробки макету програми.

Backbone.js мінуси:

- Щоб писати складніші програми, необхідно завантажити додаткові плагіни та розширення.
- Невеликий розмір. Водночас це перевага. Проте, якщо використовувати Backbone.js у повній мірі, необхідно додавати у проект фреймворки Underscore.js та jQuery.
- Backbone.js не має готових структур. У ньому є лише кілька дуже простих інструментів для розробки макету програми.

Ember — JavaScript фреймворк, який спрощує створення масштабованих односторінкових додатків. Представляє з себе каркас, що реалізує MVC шаблон (Рис.8).

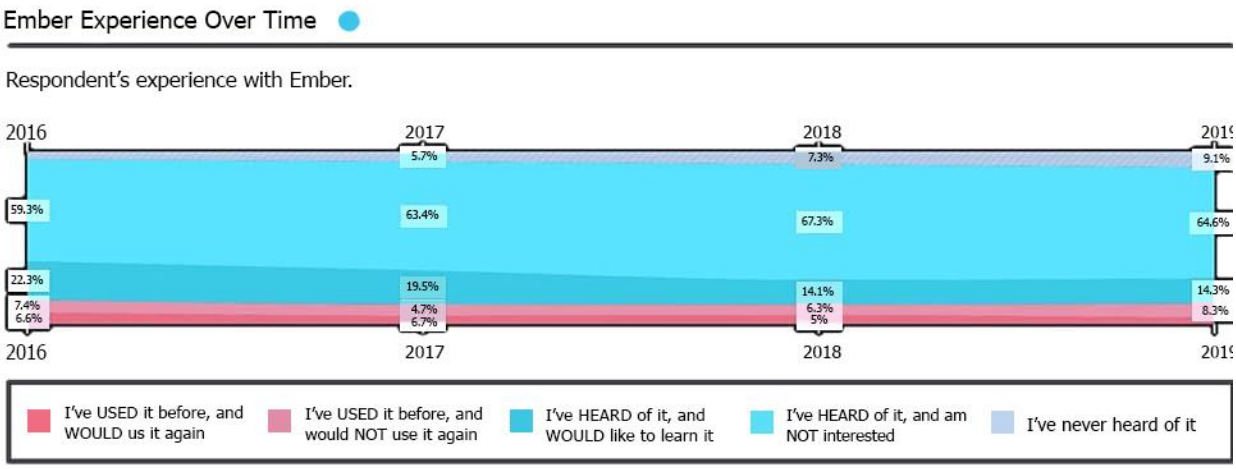

Рис.*8 Графік завантаження Ember*

Двостороння прив'язка даних, як у Angular.js. Ця функція синхронізує і модель, і вигляд. Наявність Fastboot.js дозволяє прискорити рендеринг всіх серверних DOM та підвищити продуктивність складних інтерфейсів. Ember.js є одним із основних компонентів складного стеку зовнішнього інтерфейсу, підготовленого командою Ember. Ці рішення також включають Ember CLI, Ember Data, Ember Inspector, Fastboot та Liquid Fire.

Плюси Ember.js:

- Він використовує прості для розуміння шаблони, що дозволяють легко і гнучко розробляти інтерфейс користувача.
- Є можливість додавати маршрути вкладення поверх подання шаблону.
- Просте налагодження завдяки рішенням Ember Inspector.
- Користувачі отримують хорошу документацію з фреймворку JS.
- Усі шаблони можна попередньо скомпілювати на сервері.

Мінуси Ember.js:

- Вивчення фреймворку вважається важким. Ember.js рекомендується просунутим користувачам, які добре знаються на інтерфейних фреймворках JS.

- Фреймворк не використовує прості моделі JavaScript, що робить веб-застосунки більш складними, ніж у випадку з Angular.js [8].

1.4.2 Переваги та недоліки використання front-end фреймворків

Фреймворки JavaScript постійно розвиваються. Вони використовуються для веб-додатків, і для мобільних. Всі вони надають різні бібліотеки, функції та розширення. Деякі підходять для односторінкових веб-додатків, інші краще працюють із більшими проектами. З розглянутого вище можна виділити наступні переваги та недоліки фреймворків:

Переваги:

- зручність написання коду;
- більш висока швидкість роботи;
- зниження витрат розробки продукту;
- оптимізація тимчасових витрат;
- написання максимально чистого коду без необхідності проводити рефакторинг.

Недоліки:

- слабкий захист від зовнішніх загроз через відкритий код більшості фреймворків;
- складність навчання, адже кожен фреймворк – результат роботи розробника, не завжди можна відразу зрозуміти, що і як працює у певному фреймворку;
- необхідно заново освоювати середовище розробки при переході з одного фреймворку на інший.

1.5 Аналіз back-end фреймворків

Бекенд (back-end) — це програмно-апаратна частина сервісу, набір засобів, за допомогою яких здійснюється реалізація логіки веб-сайту. Логіка сайту полягає у трьох кроках:

- Відправка інформації від користувача.
- Обробка на сервері.
- Отримання інформації та форматування коду в вид, що читається.

Backend-розробник використовує будь-які інструменти, які є доступними на сервері. На практиці програмісти застосовують будь-яку з універсальних мов: PHP, Ruby, Python, Java. Крім того, при backend-розробці використовується такі СУБД, як MySQL, PostgreSQL, SQLite, MongoDB та ін.

Backend-розробник виконує:

- Бізнес-логіки та back-end системи для підтримки продукту.
- Створення функціональних API.
- Проектування та розробка єдиної бібліотеки компонентів для дизайнерів та розробників.
- Підтримка та розвиток інфраструктурного інструменту на основі SourceJS.

1.5.1 Огляд сучасних back-end фреймворків

Django — це провідне серверне середовище з відкритим вихідним кодом, що базується на мові програмування Python. Він слідує шаблону контролера представлення моделі (MVC). Django підходить для розробки складних та багатофункціональних веб-сайтів на основі баз даних.

Ця внутрішня структура забезпечує оптимальну підключеність, скорочену кількість коду, більшу можливість повторного використання та більш швидку

розробку. Він використовує Python для всіх операцій у Django та надає додатковий інтерфейс адміністратора, який допомагає створювати, читати, оновлювати та видаляти операції. Django використовується багатьма відомими веб-сайтами, такими як Disqus, Mozilla та The Washington Times.

Переваги Django:

- Швидка структура.
- Багатофункціональний.
- Оптимальна безпека.
- Висока масштабованість.
- Універсальний фреймворк.

Недоліки Django:

- Використання шаблону маршрутизації із зазначенням URL.
- Django надто монолітний.
- Все базується на ORM Django.
- Компоненти розгортаються спільно.
- Необхідне вміння володіти всією системою для роботи.

Особливості Django:

- Середовище з відкритим вихідним кодом, яке використовується для веб-застосунків на основі Python.
- Система іменування має власну систему створення для всіх інструментів і функцій. Має просту у використанні панель адміністратора в порівнянні з іншими.
- Деякі ключові функції Django включають простий синтаксис, базову архітектуру MVC, ORM (Object Relational Mapper), підтримку проміжного програмного забезпечення та бібліотеки HTTP. Django має власний веб-сервер, фреймворк для модульного тестування Python та компоненти, які необхідні для вирішення деяких випадків.

Laravel — це веб-фреймворк PHP з відкритим вихідним кодом для розробки веб-додатків на основі Symfony, які випливають з архітектури модель – вид – контролер (MVC). Він пропонує модульну пакувальну систему із спеціальним менеджером залежностей. Laravel надає своїм користувачам кілька способів доступу до реляційних баз даних, а також утиліт для обслуговування та розгортання програм. Laravel має ліцензію MIT і вихідний код розміщено на GitHub.

Переваги Laravel:

- Аутентифікація.
- Простий.
- Бекенди кешування.
- Журнали.
- Тестування.

Недоліки Laravel:

- Відсутність вбудованої підтримки.
- Проблема з оновленнями.
- Важка документація.
- Не працює для спільного хостингу.
- Порушено зворотну сумісність між версіями.

Особливості Laravel:

- Механізм шаблонів: фреймворк Laravel має легкі вбудовані шаблони, які можна використовувати для створення макетів та заповнення вмісту. Він також надає віджети з кодом JS та CSS. Шаблони Laravel призначені для розробки як простих, так і складних макетів із розділами.

- Підтримка архітектури MVC: Laravel пропонує підтримку шаблонів архітектури MVC для ефективного поділу рівнів уявлення та бізнес-логіки.

Laravel MVC пропонує безліч функцій, сприяє підвищенню продуктивності та покращує як масштабованість, так і безпеку.

- Eloquent Object Relational Mapping: користувачі Laravel можуть використовувати Eloquent Object Relational Mapping (ORM), включаючи просту реалізацію PHP Active Record. ORM дозволяє розробникам додатків створювати запити до бази даних із використанням синтаксису PHP без написання коду SQL. ORM порівняно швидше за інші PHP-фреймворки.
- Безпека: фреймворк Laravel забезпечує надійну безпеку веб-додатків з використанням хешованих паролів. В результаті паролі не зберігаються у базах даних у текстовому форматі. Алгоритм хешування Bcrypt використовується Laravel для створення зашифрованих паролів.

Ruby on Rails є платформою серверних веб-додатків на основі Ruby з ліцензією MIT. Rails — це середовище MVC, яке пропонує структури бази даних за замовченням, веб-сторінки та веб-служби. Ruby on Rails просуває використання веб-стандартів, таких як XML або JSON, для передачі даних та CSS, JavaScript та HTML для взаємодії. Rails надає пріоритет використанню шаблонів програмної інженерії, таких як шаблон активного запису, угода конфігурації (CoC) і не повторюватися (DRY).

Переваги Ruby on Rails:

- Економія часу.
- Послідовність.
- Рентабельність.
- Якісна розробка.
- Масштабованість.

Недоліки Ruby on Rails:

- Мова створювалася для Linux, тому при написанні проектів для Windows можуть виникнути деякі проблеми.

- Низька швидкість робот.
- Проекти на ньому досить громіздкі.
- Складний debug коду.
- Погана російськомовна документація.

Особливості Ruby on Rails:

- Архітектура MVC – Ruby on Rails заснована на MVC, є однією з найбільш широко використовуваних архітектур веб-додатків у всьому світі. Ця архітектура поділяє коди за їхніми функціями, тобто рівнем даних, рівнем представлення та підтримкою рівня ресурсів.
- Active Record — Ruby on Rails використовує бібліотеку, відому як активний запис, що дозволяє легко виконувати проектування запитів до бази даних. Запити написані мовою програмування Ruby перетворюються на запити SQL, які отримують вихідні дані та повертають об'єкти.
- Угода замість конфігурації — файли конфігурації не використовуються в Ruby on Rails, оскільки він забезпечує динамічні розширення, відображення та угоди під час виконання. Програмні системи, такі як структури веб-додатків Java, використовують кілька конфігураційних файлів, кожен з яких має кілька налаштувань.
- Тестувати просто — Ruby on Rails пропонує RSpec, просте у використанні налаштування модульного тестування. Користувачі можуть легко протестувати функції за допомогою окремих викликів і переконатися, що програма проходить відповідне тестування.

Express.js є платформою веб-додатків Node.js і програмним забезпеченням з відкритим вихідним кодом, доступним за ліцензією MIT. Він використовується для створення API-інтерфейсів та веб-додатків і вважається стандартним серверним середовищем Node.js. Express – це серверний компонент стека MEAN разом з інтерфейсною структурою AngularJS та базами даних.

Переваги Express JS:

- Простота навчання.
- Використовує функції Full-stack JS.
- Забезпечує високу продуктивність.

Недоліки Express JS:

- Великий обсяг ручної роботи.
- Використовується застарілий підхід callbacks функцій.
- Js не пропонує рішень у сфері безпеки.
- Структура Express.js дещо розпливчаста.

Особливості Express.js:

- Швидке програмування на стороні сервера — Express.js — це фреймворк Node.js, що пропонує багато з найбільш широко використовуваних функцій Node.js. Ці функції використовуються у різних частинах програми. Розробники Express JS можуть миттєво включити кілька рядків коду замість того, щоб писати великі обсяги коду. Розробка веб-застосунків за допомогою Express виконується швидше, ніж розробка з використанням тільки Node.js.

• Маршрутизація. Маршрутизація — це функція, яка дозволяє веб-програмам зберігати стан веб-сторінок через URL-адресу. URL-адреси можуть бути передані іншим, і користувачі можуть відвідувати URL-адреси, щоб перейти на сторінку сховища. Node.js пропонує фундаментальний механізм маршрутизації в порівнянні з Express.js, який надає складніший механізм маршрутизації, здатний обробляти динамічні URL-адреси.

- Налагодження. Помилки часто зустрічаються під час розробки проектів і можуть викликати повномасштабні збої в роботі додатків. Розробники повинні негайно виявляти джерела помилок та виправляти помилки. Express.js пропонує зручну систему налагодження, що дозволяє розробникам легко визначити причини помилок програми.

CakePHP — широко використовувана веб-платформа з відкритим кодом. Він написаний на PHP і слідує архітектурі MVC. Він був доступний за ліцензією MIT та заснований на концепціях Ruby on Rails. CakePHP використовує популярні концепції проектування та розробки програмного забезпечення, включаючи активний запис, фронт-контролер, модель-вид-контролер, угода про конфігурацію та відображення даних.

Переваги CakePHP:

- ORM або об'єктно-реляційне зіставлення.
- Розширюваність.
- Функціональність.

Недоліки CakePHP:

- Низька продуктивність.
- Слабка документація.
- Нестійкість до CSRF-атак.
- Немає російськомовної спільноти.
- Строгі угоди з іменування.
- Низька швидкість розвитку.

Особливості CakePHP:

- Взаємодія з базою даних із вбудованим CRUD.
- Будівельні риштування для додатків.
- Гнучке ліцензування.
- Архітектура MVC.
- Сумісність із кількома версіями PHP.
- Генерація коду.
- Вбудована перевірка.
- AJAX, HTML форми, JavaScript View Helpers.

- Менеджер запитів, що містить маршрути та налаштовані URL-адреси.
- Гнучке створення шаблонів за допомогою синтаксису PHP із допоміжними функціями.
- Гнучкість ACL.
- Гнучкість кешування.
- Локалізація.
- Компоненти для обробки файлів cookie, сеансу, електронної пошти та запитів.
- Працює в каталогах веб-сайтів, не потребуючи особливого налаштування Apache [9-10].

1.5.2 Переваги та недоліки використання back-end фреймворків

Під час вибору фреймворку можуть виникнути певні труднощі, пов'язані з визначенням завдань, які він може виконувати, та його призначення. Не кожен фреймворк має якісну інтеграцію і хорошу документацію. Якщо для створення сайту потрібно знайти зручний і простий в освоєнні фреймворк, то необхідно ретельно підійти до питання його вибору.

Загальні переваги використання back-end фреймворків:

- Гнучкість розробки та розвитку проекту.
- Ефективне використання ресурсів сервера.
- Відкритий код.
- Постійний розвиток і вдосконалення фреймворка.
- Світова популярність, велика кількість розробників.
- Фреймворк дозволяє сконцентруватися на вирішенні архітектурних завдань, а не базових, як при розробці без його застосування.
- Дозволяє вузько вирішувати поставлену задачу.

Одна з головних переваг фреймворка – зручна розробка нестандартних проектів. Жоден великий нестандартний проект не розроблений на готовій CMS – вони для цього не призначені. Усі оригінальні проекти розробляють на фреймворках. Web-проект, розроблений за допомогою фреймворку, розвивається динамічно. При зміні вимог змінюється і сайт, достатньо лише замінити окремий блок (модуль), створити новий розділ або внести новизну в дизайні.

Недоліки застосування back-end фреймворків:

- 1 файл = 1 клас.
- Багатоне використовуваного коду.
- Складність в навчанні.
- Готових модулів і компонентів немає, всі додатковімодулі необхідно замовляти у розробників [11].

1.6 Варіанти взаємодії frontend та backend

Розглянемо кілька прикладів того, як взаємодіють між собою ці дві сфери розробки,а саме взаємозв'язок цих сфер. Розділимо принципи роботи на чотири популярні варіанти:

- Перший тип має на увазі відправку HTTP-запиту на сервер, де вона шукається, вбудовується в шаблон і повертається користувачеві в читаному вигляді HTML-сторінки.
- Другий — це інструментарій під назвою AJAX. У такому випадку запит надсилається за допомогою JavaScript, який підключено до веб-браузера. Відповідь повертається до XML або JSON, а читанням цих форматів відбувається за допомогою JS.

- Третій — це односторінкові сайти, які завантажують дані без попереднього оновлення сторінки. Здійснюється така операція за допомогою AJAX або фреймворків Angular та Ember;
- Четвертий — це бібліотеки Ember або React, що підключаються. Вони призначені для використання програми одночасно на сервері та в клієнтській частині. Дві сфери, що розглядаються, зв'язуються через AJAX і HTML-код з обробкою на сервері [12].

1.7 Актуальність порівняння фреймворків Angular та React

Перш ніж вирішити, який фреймворк використати, доцільно подивитися на його популярність. Чим більша і активніша спільнота, тим швидше вона знайде вирішення будь-якої несподіваної проблеми.

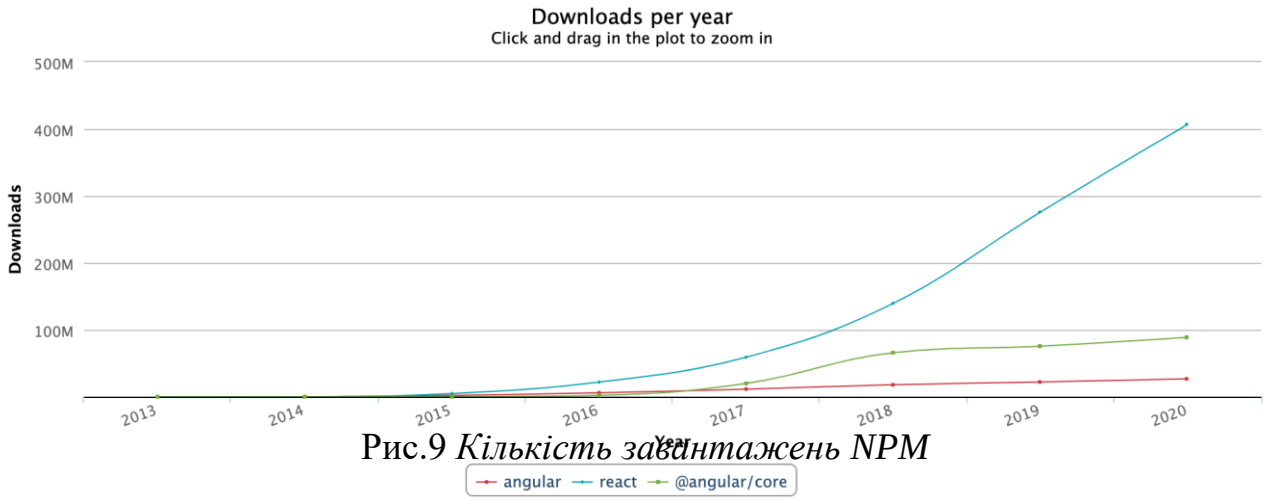

Рис.9 *Кількість завантажень NPM*

На Github, React має 164 тис. зірок, у Angular майже 130 тис. зірок. Це показує, що обидва фреймворка мають дуже велику популярність та актуальність серед користувачів, оскільки вони залишаються на вершині найпопулярніших ресурсів протягом довгого часу. Цей ріст у вигляді кількості завантажень Node

Package Manager (**npm**) відображено на рисунку 9. На діаграмі видно, що **React має більше завантажень npm, ніж Angular** з 2015 року. Кількість продовжує зростати, на відміну від Angular, який має тенденцію до стагнації.

Згідно з опитуванням Stackoverflow 2020 (Рис.10) , можна помітити, що 35,9% респондентів віддають перевагу React, а 25,1% використовують замість них Angular. Крім того, це опитування також ілюструє, що, незважаючи на те, що React є одним з найпопулярніших веб-фреймворків, Angular також вважається найстарішим.

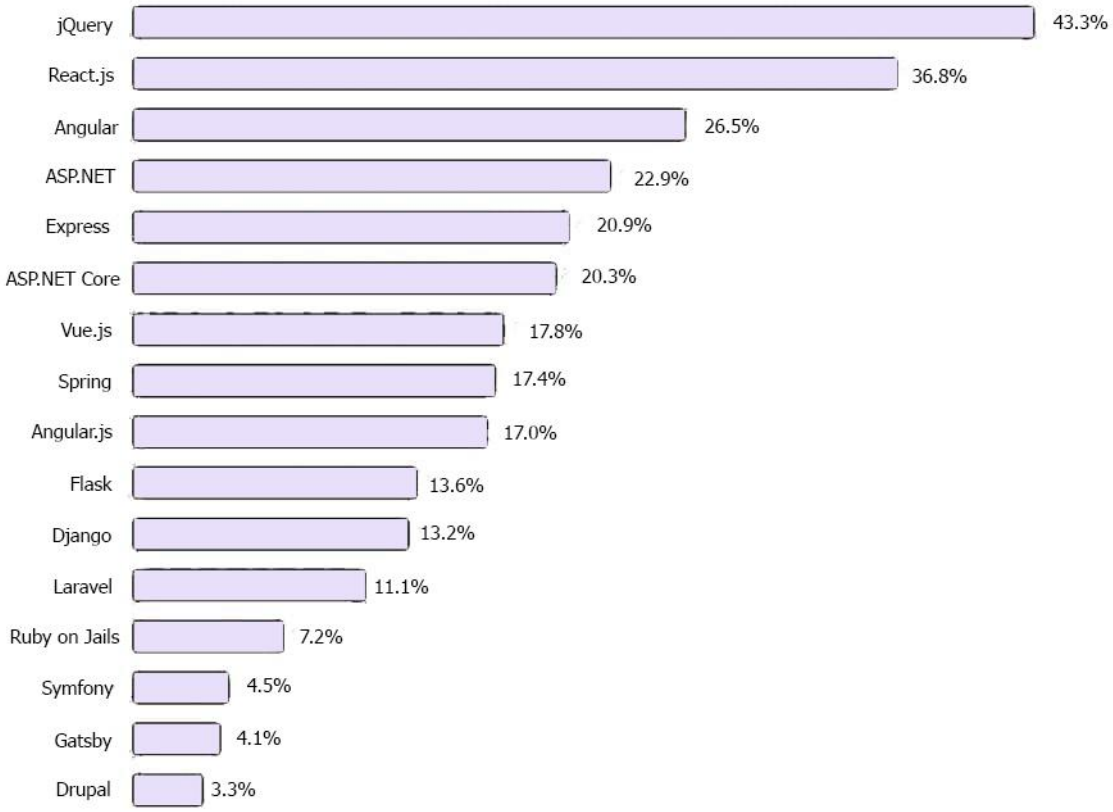

Рис.*10 Опитування Stackoverflow 2020*

Серед проектів, які найбільш підходять під методологію та функціонал Angular виділяють:

- *Корпоративні веб-програми.* TypeScript має всі функції, необхідні для розробки великомасштабних проектів. TypeScript оснащений функціями автозаповнення, розширеного рефакторингу та навігації. Завдяки архітектурі інструменту можна легко повторно використовувати і підтримувати код.
- *Програми з динамічним контентом*. Оскільки основною метою Angular було створення односторінкових веб-застосунків, він має широкий набір інструментів для розробки SPA, для яких контент повинен динамічно змінюватися в залежності від поведінки та уподобань користувача. Використання залежностей гарантує, що у разі зміни одного компонента інші пов'язані з ним компоненти будуть змінені автоматично.
- *Прогресивні веб-застосунки (PWA)*. PWA були розроблені Google в 2015 році. Команда Angular працює над спрощенням процесу створення PWA. Angular 5 має вбудовану підтримку PWA. Нові команди інтерфейсу командного рядка, що поставляються з Angular 6, дозволяють розробникам легко перетворювати свої веб-програми на прогресивні веб-додатки.

Численні переваги Angular для розробників – основна причина, через яку багато популярних компаній по всьому світу використовують цей інструмент для своїх проектів.

Переваги React

- React є декларативним, що допомагає створювати інтерактивні інтерфейси користувача. React використовується для розробки простих уявлень для кожного стану у програмі, а також ефективно оновить і відобразить потрібні компоненти, у разі змінення даних. Декларативні уявлення роблять код більш передбачуваним і легшим для налагодження.
- ReactDOM — це двигун, який робить його не тільки простим для написання та візуалізації, але й потужним. За його допомогою є можливість

створювати HTML дуже просто та швидко, передавати дані з JS в HTML і відображати весь HTML як елемент.

- Компоненти є будівельними блоками сторінок реакції. Сторінка може мати кілька компонентів. Кожен компонент може керувати своїм власним станом і представляти частину сторінки і може мати власне джерело даних. Можна легко передавати розширені дані через свою програму і зберігати стан поза межами DOM.
- React може відображати на сервері за допомогою Node. Node (раніше Node.js) став популярним методом написання програмування на стороні служби на основі JavaScript.
- React Native швидко розвивається і стає основною мовою програмування для написання нативних мобільних додатків. Розробники React можуть використовувати свої наявні навички для створення мобільних додатків за допомогою React Native.

До відомих проектів на React, відносять*:*

- *Facebook*. У ньому React використовується частково, але і у версії як для персонального комп'ютера, так і для мобільного додатку.
- *Instagram*. У такому популярному додатку React відводиться величезна роль. Починаючи з можливості визначення геопозиції та закінчуючи точністю функціоналу пошуку.
- *Netflix*. Найактивніше задіюється на платформі Gibbon. Основною функцією стає можливість настроїти параметри для телевізорів із низькою продуктивністю. Бібліотека допомагає прискорити завантаження та підвищити продуктивність.
- *Yahoo! Mail*. Завдяки Facebook ці сервіси стали впорядкованими в плані архітектури, а саме: легке налагодження, низький рівень входження, незалежно від компонентів, що розгортаються. React підходить завдяки таким

властивостям, як: односторонній потік даних, можливість використовувати віртуальний DOM, активна спільнота.

- WhatsApp Фахівці сервісу вирішили використовувати React для створення інтерфейсів користувача [13].

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ЗАСОБІВ ТА МЕТОДІВ ДЛЯ ПОРІВНЯННЯ ПРЕДСТАВЛЕННИХ ФРЕЙМВОРКІВ

2.1 Технології розробки інтерфейсу користувача

Інтерфейс користувача розвивався з появою інтерфейсу командного рядка, який вперше з'явився як майже порожній екран з рядком для введення користувача. Користувачі покладалися на клавіатуру та набір команд для навігації по обміну інформацією з комп'ютером.

Цей інтерфейс командного рядка привів до інтерфейсу, в якому переважали меню (списки варіантів, написаних текстом). Потім з'явився графічний інтерфейс користувача, створений в дослідницькому центрі Xerox в Пало-Альто (PARC), прийнятий і вдосконалений Apple і ефективно стандартизований Microsoft у своїх операційних системах Windows.

Інтерфейс користувача (UI) — це точка взаємодії людини комп'ютера та зв'язку в пристрої. Це може включати екрани дисплея , клавіатуру , мишу та зовнішній вигляд робочого столу . Це також спосіб, за допомогою якого користувач взаємодіє з програмою або веб-сайтом .

Зростаюча залежність багатьох компаній від веб-додатків і мобільних додатків привела до того, що багато компаній надають перевагу інтерфейсу користувача, намагаючись покращити загальний досвід користувача [14].

Виділяють наступні типи інтерфейсів користувача:

- графічний інтерфейс користувача (GUI);
- інтерфейс командного рядка (CLI);
- керований меню інтерфейс користувача, сенсорний інтерфейс користувача;
- SILK-інтерфейс;

- інтерфейс користувача на основі форм користувача природною мовою.

Зробимо короткий опис вище вказаних видів інтерфейсу.

Графічний інтерфейс користувача (GUI) — це тип інтерфейсу користувача, за допомогою якого користувачі взаємодіють з електронними пристроями, використовуючи візуальні індикаторні уявлення.

SILK-інтерфейс (від speech — мова, image — образ, language — мова, knowledge — знання) – це голосовий інтерфейс, тобто управління голосом. Він з'явився після винаходу звукових карт. Комп'ютер виділяє з промови команди та інтерпретує у двійковий код. Відбувається виконання команди і користувач бачить на екрані результат роботи.

Інтерфейс командного рядка (CLI) — це текстовий інтерфейс користувача (UI), який використовується для перегляду та керування файлами комп'ютера. Інтерфейси командного рядка також називають інтерфейсами командного рядка, консольними та символьними.

2.2 Ключові критерії порівняння фреймворків

Для дослідження представлених фреймворків React та Angular доцільно переглянути та подивитися на кожну існуючу характеристику, щоб переконатися в їх перевагах або недоліках та заздалегідь оцінити можливі ризики.

Визначення основних характеристик можна зробити різними способами, а саме: дослідження об'єктів та предметів, аналіз для порівняння програмних продуктів та порівняння, виявлення відмінностей та встановлення загальностей, а також аналіз отриманих результатів досліджень.

Для аналізу розглядаються наступні показники:

- **Прив'язка даних.**

- Мова.
- Використання DOM.
- Архітектура.
- Стан.
- Легкість навчання.

2.2.1 Форма прив’язки даних

Прив'язка даних — це механізм, який з'єднує модель даних з інтерфейсом користувача (UI). Для прив’язки даних можна використовувати будь-який код, в межах цієї роботи зв’язування даних відноситься до бібліотеки або фреймворку, що надає послуги зв’язування даних через API. Це не вимагає від розробника вручну отримувати значення моделі, а також виконувати якісь маніпуляції з DOM або іншими шаблонами рядків, щоб заповнити інтерфейс користувача даними моделі. Послуги прив’язки даних користуються великою популярністю у розробників, оскільки вони значно спрощують процес оновлення інтерфейсу користувача та зменшують кількість коду у додатках.

Модель даних, інтерфейс користувача та механізм прив’язки можуть мати різні форми, але в кінцевому підсумку мета полягає в тому, щоб з'єднати інтерфейс користувача з якоюсь моделлю. Існує три основні форми прив’язки даних: одноразова, одностороння та двостороння. Вибір одного або іншого пов’язаний з багатьма факторами [15].

Одноразове зв'язування даних відбувається один раз між моделлю та інтерфейсом користувача. Як правило, коли спочатку створюється інтерфейс користувача, значення модельних даних у цей конкретний момент часу використовуються для заповнення новоствореного інтерфейсу. Прив’язка відбувається один раз, і немає автоматизованого механізму оновлення інтерфейсу користувача, коли відбуваються майбутні зміни моделі. Зазвичай, одноразові

бібліотеки прив’язки, що використовуються, включають шаблони підкреслення та шаблони керування (Рис.11). З точки зору продуктивності, разова прив’язка – це найвища продуктивність, тому що це відбувається один раз і тому вона не вимагає жодних механізмів виявлення змін для виявлення змін в базовій моделі або в інтерфейсі користувача. Механізми виявлення змін вносять потенційні проблеми з продуктивністю у роботі веб-сторінки, оскільки має бути виконано блокування коду, щоб визначити, чи потрібно розповсюджувати зміни моделі до інтерфейсу користувача.

Рис.*11 Одноразове зв'язування даних*

Одноразове прив'язування даних є практичним, коли дані ніколи або дуже рідко змінюються. У разі, якщо дані змінюються регулярно або дуже часто, одноразова прив’язка даних стає перешкодою для простих та ефективних оновлень інтерфейсу користувача. Для вирішення проблеми оновлення інтерфейсу користувача для регулярної зміни даних моделі використовується одностороннє (або двостороннє) прив’язування даних.

Одностороннє зв'язування даних створює постійний зв'язок між моделлю та інтерфейсом користувача (Рис.12). Зміни моделі відображаються в інтерфейсі користувача через певний процес, керований бібліотекою або фреймворком. Одностороннє зв'язування даних поширює лише зміни від моделі до інтерфейсу користувача, а не навпаки.

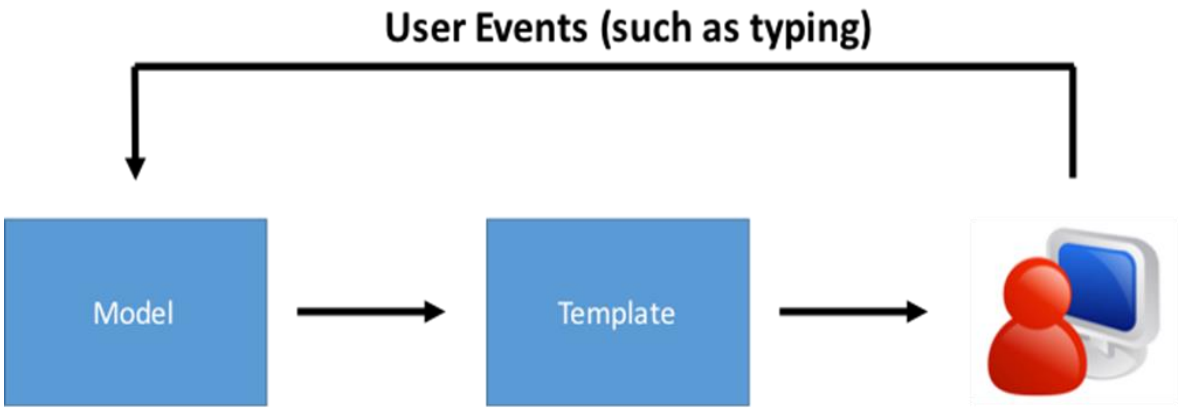

Рис.*12 Одностороннє зв'язування даних*

Центральне місце в реалізації одностороннього (і двостороннього) прив'язування даних займає автоматизований механізм визначення того, чи відбулися зміни в моделі. Цей процес називається виявленням змін.

Виявлення змін. З точки зору одностороннього зв’язування даних, виявлення змін включає два аспекти: знати, коли модель змінилася, і знати, що змінилося в інтерфейсі користувача в результаті змін моделі.

Для виявлення змін моделі доступні три основні шаблони: вручну, підписка видавця та асинхронні обгортки. Виявлення змін вручну може відбуватися у двох формах – явному та неявному. Явне виявлення змін вручну – це коли розробник повинен чітко повідомити бібліотеці або фреймворку, що відбулася зміна (або потенційно відбулася), і, відповідно, оновити інтерфейс користувача новими даними моделі. Після того, як було виявлено зміну моделі, потрібно оновити інтерфейс користувача і відбудеться другий етап виявлення змін. У більшості одноразових прив'язок даних і навіть у багатьох односторонніх системах прив'язки даних DOM оновлюється неефективно, щоб відобразити зміни моделі. Неефективність виявляється як руйнування гілки дерева DOM, що оновлюється, і повне відтворення та повторне її відображення. Як правило, це не найефективніший спосіб оновлення інтерфейсу користувача, але це найпростіший спосіб для розробника кодувати процес оновлення інтерфейсу

користувача. Однак, ця ефективність не завжди поширюється на всю бібліотеку або фреймворк, а тим більше на користувацький код інтерфейсу, наданий розробниками.

Двостороннє зв'язування даних. Одноразові та односторонні прив'язки даних перетікають від моделі до інтерфейсу користувача. Двосторонній включає односторонній, але він також полегшує потік даних від інтерфейсу користувача до моделі (Рис.13).

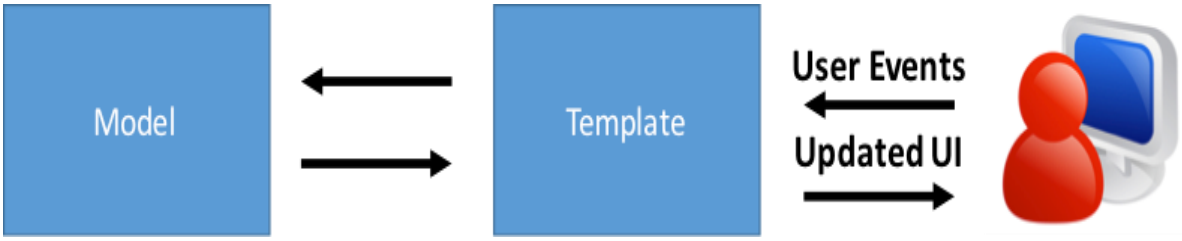

Рис.*13 Двостороннє зв'язування даних*

Незалежно від того, чи бібліотека чи фреймворк явно підтримують двосторонню прив'язку даних, єдиний спосіб отримати нові дані з інтерфейсу користувача — це події інтерфейсу, такі як клавіша введення, натискання кнопки чи клік. Оновлення моделі з інтерфейсу користувача здійснюється за допомогою одного з двох шляхів: елемента інтерфейсу до події інтерфейсу для моделі або події інтерфейсу для моделювання до елемента інтерфейсу користувача. Як правило, перший описує двостороннє зв'язування даних, тоді як другий є формою одностороннього зв'язування даних. Різниця полягає в тому, як оновлюється елемент інтерфейсу. Коли елемент інтерфейсу оновлюється за допомогою моделі та дії введення користувача, він має два джерела даних. Коли елемент інтерфейсу завжди оновлюється лише з моделі, він має одне джерело даних. Єдине джерело даних – результати подій інтерфейсу, викликаних взаємодією користувача, які спочатку оновлюють модель, а потім модель оновлює елемент інтерфейсу. Сам елемент не оновлюється безпосередньо ініційованою подією.

Отже, прив'язка даних — важливий інструмент, який використовується розробниками для заповнення інтерфейсів користувача даними моделі. Існує багато способів зв’язування даних, кожен зі своїми перевагами та вартістю. Важливі фактори включають продуктивність, потік даних та складність коду. Існує багато фреймворків та бібліотек, які пропонують різні підходи до прив’язки даних, включаючи Handlebars, KnockoutJS, Angular 1 & 2 та React.

Незалежно від використовуваного методу, ключовим є розуміння концепцій та ідей, що лежать в основі зв’язування даних, і того, як обрана бібліотека.

2.2.2 Javascript та Typescript як основа фреймворків

Javascript — це мова сценаріїв, яка допомагає створювати інтерактивні веб-сторінки. Мова дотримується правил програмування на стороні клієнта, тому працює у веб-браузері користувача без потреби в будь-яких ресурсах веб-сервера. Також можна використовувати Javascript з іншими технологіями, такими як REST API, XML тощо. Ідея розробки цього сценарію полягає в тому, щоб зробити його додатковою мовою сценаріїв, як VisualBasic для C++ у мовних сімействах Microsoft. Однак JavaScript не призначена для великих складних програм. Вона була розроблена для програм із кількома сотнями рядків коду.

Особливості JavaScript:

- Мова сценаріїв JavaScript є об’єктно-центрованою мовою, як VisualBasic, і має концепції, побудовані навколо концепції об’єктів. Об’єктно-центровані мови широко використовуються для таких функцій, як поліморфізм, завдяки якому об’єкт може бути прийнятий у кількох формах.
- Технологія Clientedge — веб-браузер, який використовується користувачем. Дані сервера завантажуються клієнтом, а потім користувач

отримує доступ до них після їх візуалізації. Користувач може використовувати клієнт або браузер для перегляду веб-сайтів і взаємодії з ними.

- Перевірка введених даних користувача, також відома як перевірка форми, дозволяє користувачам ініціювати взаємодію з клієнтом за допомогою заповнення форми веб-сторінки. Деталі форми повинні бути підтверджені клієнтом, щоб підтвердити інформацію, надану користувачем.

Переваги JavaScript.

- Завантаження сервера. Java Script є клієнтською мовою, яка може динамічно знижувати вимоги сервера. Деякі програми також можуть не потребувати сервера для функціонування.
- Розширені інтерфейси. JavaScript можна використовувати для розробки функцій, таких як функції перетягування для повзунків. Це може радикально покращити користувальницький інтерфейс та досвід роботи на веб-сайті.
- Розширена функціональність. Розробники JavaScript можуть розширити функціональність веб-сторінки, створюючи фрагменти JavaScript для кількох сторонніх доповнень.

Недоліки JavaScript.

- Безпека на стороні клієнта. Код JavaScript виконується на комп'ютері користувача, що часто призводить до проблем безпеки через зловмисні дії. Саме тому багато розробників відключають JavaScript.
- Підтримка браузера. JavaScript можна інтерпретувати по-різному залежно від того, який браузер інтерпретує його. У результаті створення міжбраузерного коду це може бути проблемою для деяких розробників.

TypeScript — це сучасна мова розробки Javascript. Це статично скомпільована мова для написання зрозумілого та простого коду Javascript. Його

можна запустити на Nodejs або будь-якому браузері, який підтримує ECMAScript 3 або новіші версії. TypeScript надає необов'язкові статичні типи, класи та інтерфейс. Для великого проекту JavaScript застосування Typescript може дати більш надійне програмне забезпечення, яке легко розгортається за допомогою звичайної програми JavaScript.

Особливості TypeScript.

- TypeScript — дуже схожий на JavaScript і часто вважається таким самим. Він використовує деякі компоненти програм, запозичені з JavaScript. Щоб добре використовувати TypeScript, розробники повинні знати роботу JavaScript. Це тому, що код TypeScript перед виконанням перекладається в код JavaScript.
- Підтримка бібліотеки JS. TypeScript пропонує підтримку бібліотек JavaScript, а скомпільований код можна взяти з будь-якої частини коду JS. Код JavaScript, згенерований з TypeScript, використовує всі поточні інструменти, фреймворки та бібліотеки JS.
- Перейменування файлів. Користувачі вважають перейменування файлів JavaScript у файли TypeScript досить зручним. Будь-який файл .js можна легко перейменувати у файл .ts і зібрати з різними файлами TypeScript.

Переваги TypeScript.

- Строга введення. У TypeScript елементи, визначені певним чином, залишаються такими, якими вони були встановлені.
- Структурний тип типізації. Доступність структурної типізації є важливою для користувачів, які мають намір повністю визначити використовувані структури. TypeScript дозволяє розробникам легше покладатися на призначені структури.
- Анотації типу. Використання анотацій типу TypeScript дозволяє користувачам явно вказувати тип, який вони мають намір використовувати.

Недоліки TypeScript.

- Вимагає високого рівня зусиль для реалізації коду.
- Складно пропонує підтримку теоретичних занять.
- Потребує документ визначення для використання сторонньої бібліотеки, і цей документ не завжди може бути доступним.
- Розробники повинні пам’ятати про природу визначення типу в TypeScript.
- Перед запуском програми необхідно конвертувати TypeScript у JavaScript [16].

2.2.3 Об'єктна модель даних веб-застосунку

Зазвичай використовують DOM або створюють віртуальний DOM.

Віртуальний DOM — це спрощена версія DOM. За допомогою віртуального DOM можливо змінити будь-який елемент дуже швидко і без необхідності заново відтворювати весь DOM. Це кардинально змінює продуктивність з хорошої на відмінну.

Маніпулювання DOM — це сучасна інтерактивна мережа. Вона також набагато повільніше, ніж більшість операцій JavaScript. Ця повільність погіршується тим фактом, що більшість фреймворків JavaScript оновлюють DOM набагато частіше чим потрібно.

У разі роботи зі списком, який містить десять елементів, більшість фреймворків JavaScript відновляють весь список. Це в десять разів більше роботи, ніж потрібно. Змінюєтьсяя лише один елемент, але останні дев’ять перебудовуються так, як вони були раніше. Відновлення списку не становить великої роботи для веб-браузера, але сучасні веб-сайти можуть використовувати величезну кількість маніпуляцій з DOM. Неефективне оновлення є серйозною проблемою, для вирішення якої користувачі використовують віртуальним DOM.

2.2.4 Архітектура

Шаблон розробки програмного забезпечення MVC надає основний приклад в JavaScript, HTML та CSS. З часом розвиваються похідні від MVC: MVP і MVVM. Сучасна Front-End розробка знаходиться під значним впливом MVC, але деякі моменти не змінилися, наприклад, розділення проблем (SoC).

MVVM — це уточнення дизайну MVC, за допомогою якого ViewModel в MVVM полегшує розділення розробки графічного інтерфейсу користувача (UI), тобто презентаційного шару. ViewModel (VM) відповідає за викриття (перетворення) об'єктів даних з моделі таким чином, щоб об'єкти легше керувались та представлялись. У цьому відношенні ViewModel є більш моделлю, ніж поданням, і обробляє більшість, якщо не всю бізнес-логіку відображення подання. MVVM — це варіація моделі презентації Мартіна Фаулера в тому, що модель презентації абстрагує або виокремлює подання таким чином, що воно не залежить від конкретної платформи інтерфейсу користувача. Отже, ні шар, що презентує, ні шар подання не знають один про одного. MVVM був винайдений архітекторами Microsoft з метою спрощення програмування UIs на основі подій. Шаблон був включений в Фонд презентацій Windows (WPF) (графічна система Microsoft.NET) і Silverlight (впровадження інтернет-додатків WPF). Model-view-viewmodel називають сполучною моделлю, щоб відобразити той факт, що подання обробляє поведінку, події та інформацію про зв'язування даних.

2.2.5 Інструменти управління станом

Управління станом — це просто спосіб стимулювати спілкування або обмін даними між компонентами. Він створює конкретну структуру даних, яка представляє стан додатка, котрий можливо читати та писати. Починаючи з React 16.8, кожен компонент React, функціональний чи класовий, може мати стан. У найпростішому визначенні стан або State – це об’єкт JavaScript, який представляє частину компонента, яка може змінюватися на основі результату дії користувача.

Можна також сказати, що стани – це просто пам’ять компонента. Коли користувач виконує дію у типовій програмі React, у стані компонента відбуваються зміни. Хоча це не погано, але це швидко стає проблемою, якщо додаток починає масштабуватися. Отже складність такого додатка надзвичайно погіршує відстеження всіх залежностей.

Припустимо, що ми створюємо додаток електронної комерції. У такому додатку майже кожен елемент може бути компонентом, наприклад, кошик для покупок, кнопки, сеанс перегляду кошика, оплата, рядок введення, тощо. У такому додатку лише одна дія користувача, що додає до кошика, може вплинути на багато інших компонентів, таких як:

- зміна стану самого компонента візка чи кошика;
- додавання кошика до історії придбань користувача;
- оформлення чи придбання товару.

Наведені вище сценарії показують важливість стану у типовому додатку React. Для управління станом у цій програмі можливо використовувати будь-яку бібліотеку на свій вибір, вони все одно виконуватимуть одну й ту саму роботу незалежно від варіації бібліотеки. Зазвичай стан потрібно буде підняти до найближчого батьківського компонента, а потім до тих пір, поки він не потрапить до спільного батьківського для обох компонентів, яким потрібен стан, а потім передати його. Цей процес може бути надзвичайним і викликає складність утримання стану. Часто це може вимагати передачі даних компонентам, яким вони навіть не потрібні.

Зі збільшенням розміру програми управління станом стає слабо керованим. Тому виникає потреба в інструменті управління станом, як Redux або Recoil, використання яких полегшує підтримку цих станів. Розглянемо бібліотеки управління станом такі як Redux, Hooks та Recoil та їх унікальність [17].

2.2.6 Аналіз бібліотеки управління станом Redux

Redux – це майже перша бібліотека управління станом на основі React. Бібліотека станового управління Redux була створена для вирішення проблеми в розглянутому додатку електронної комерції. Він надає об’єкт JavaScript під назвою store, який після налаштування включає всі стани у програмі та оновлює їх у разі потреби. Redux можна використовувати як автономне управління станом з будь-якого фреймворку.

Redux є однією з найпопулярніших бібліотек управління станом React. Redux дозволяє керувати станом додатка в одному місці та зберігати зміни у додатку більш передбачуваними шляхом, тобто таким що відстежується. Це полегшує з’ясування змін у додатку. Однак, усі ці переваги мають певні обмеження та компроміси. Часто розробники вважають, що використання Redux додає деякий шаблонний код, роблячи дрібниці здавалося б переважними, проте це залежить виключно від архітектурного рішення програми.

Один із найпростіших способів дізнатися, коли потрібно використовувати Redux, це коли локальне управління станом починає виглядати не керовано. У міру зростання програми обмін станами між компонентами стає складним. Використання Redux з React полегшує відстеження, та те яка дія викликає будь-які зміни, а отже, спрощує додаток та полегшує його обслуговування.

РОЗДІЛ 3. ПРОЕКТ ПРОГРАМНОЇ СИСТЕМИ «ОРЕНДА АВТОМОБІЛІВ В ЗАПОРІЖЖІ»

Для виконання порівняльного аналізу розроблено односторінковий додаток (SPA) з аренди автомобілів на двох фреймворках React та Angular.

3.1 Single Page Application (SPA)

Звичайний сайт складається з безлічі HTML-сторінок. Натискаючи на посилання, браузер завантажує нові сторінки за цими посиланнями, з'являється відчуття руху від однієї сторінки до іншої. Сторінки можуть лежати як файли на якомусь сервері або генеруватися під запит якоюсь серверною програмою. Але, умовно кажучи, кожен екран сайту – це окрема технічна сутність, окремий документ.

SPA працює так: коли користувач відкриває сторінку, браузер завантажує одразу весь код програми, але показує лише конкретний модуль, тобто частину сайту, яка потрібна користувачу. Коли користувач переходить в іншу частину програми, браузер бере вже завантажені дані та показує йому. Якщо потрібно, динамічно підвантажує потрібний контент з сервера без оновлення сторінки. З одного боку, такі програми працюють швидко та менше навантажують сервер, з іншого боку, вони потребують більшого завантаження на старті [18].

Можливості SPA.

Односторінкові програми найчастіше використовують у сервісах, де користувач проводить на одній сторінці багато часу або робить з нею якісь дії, наприклад:

- переглядає пошту та зазначає листи як спам;
- пише пости та коментує чужі;

- дивиться серіали;
- вибирає потрібний товар чи послугу.

Веб-версії Gmail, Facebook Netflix, AirBnB та Pinterest — односторінкові програми. Технологія настільки поширена, що її використовують навіть для веб-сайтів компаній, наприклад, сторінка Digital Agency London. SPA може обмінюватися даними з сервером без перезавантаження сторінки за допомогою ajax-запитів. Завдяки цьому наповнення сторінки може змінюватись динамічно. Наприклад, раніше у соціальній мережі потрібно було перезавантажити сторінку, щоб перевірити, чи немає нових повідомлень, тепер вони з'являються автоматично.

Переваги SPA.

- SPA швидкі. Перехід між модулями в додатку відбувається швидше: потрібні ресурси вже завантажені, необхідно просто підставити дані, які користувач запитав. Часто при цьому сервер повертає не важкий HTML, а легкий JSON або XML.
- Використання JSON спрощує розробку програми для різних платформ. Якщо для веб-версії розробити звичайний сайт, який приймає від сервера HTML, то для мобільного додатка доведеться писати доопрацювання, оскільки HTML не підійде. JSON робить відповідь сервера універсальною.
- SPA гнучкі. Якщо користувач весь час працює з однією сторінкою, простіше робити цікаві переходи та анімацію елементів. Можна працювати зі станом кнопок, вкладок та перемикачів. Таким чином, інтерфейс SPA може бути схожий швидше на повноцінний додаток, а не на простий сайт.
- SPA працюють скрізь. Все, що потрібно для SPA, це підтримка JavaScript. Такі сайти добре працюють і на робочому столі, і в Інтернеті, навіть можуть частково замінити повноцінні мобільні програми.

Недоліки SPA.

- Проблеми із SEO. За замовчуванням у додатків напружені відносини з пошуковими машинами, а саме вони натреновані індексувати окремі сторінки, кожна з яких має заголовок, опис та інші метатеги.
- Проблема із деякими системами аналітики. Стандартний тег Google Аналітики добре підходить для звичайних сайтів, оскільки його код виконується під час кожного завантаження нової сторінки. Однак, під час роботи з односторінковим додатком такий код буде виконано лише один раз. Тобто, щоб коректно збирати аналітику, доведеться самостійно налаштувати відстеження потрібних подій.
- Залежність від Інтернету. Для запуску веб-програми потрібний зв'язок із сервером, так що в більшості випадків без інтернету не обійтися, як і зі звичайними сайтами. Цим SPA програють звичайним програмам. Виянятком є той факт, що якщо під час першого завантаження браузер отримує всі дані і більше нічого не потрібно підвантажувати, то можна працювати і без інтернету.
- SPA не для новачків. Написати таку програму на простому HTML і CSS не вдасться, потрібно знати JS. Часто для цього використовують фреймворки, такі як React, Angular, Vue, Ember та інші. У будь-якому випадку для проекту потрібні компетентніші розробники.

3.2 Проект з оренди автомобілів

3.2.1 Актуальність оренди автомобілів

Останнім часом в Україні великий попит має така послуга, як оренда автомобіля. Мати права водія, стати водієм і їздити на власному автомобілі – мрія багатьох людей сьогодні. Але не завжди здійснення цієї мрії можливе: багато хто з певних причин сісти за кермо автомобіля не може. Є і люди, які не мають власного автомобіля, однак, мають права. Випадки коли немає коштів на

власний автомобіль, настає гостра необхідність кудись мобільно дістатися вирішує саме оренда авто. В даний час це як ніколи актуально. Також причиною звернення до компанії такого типу є поломка машини в людини, яка може обходитися без власного автомобіля.

3.2.2 Візуальна частина проекту

В даній частині знаходиться контактна інформація щодо оренди авто з графіком роботи, адресою та мобільними телефонами (Рис.14).

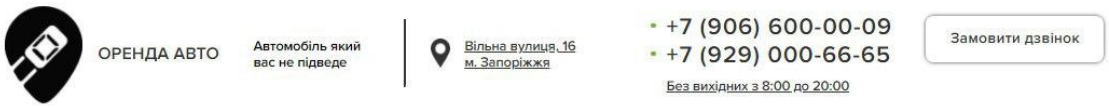

Рис.*14 Контактна інформація*

В головному меню користувачу надано інформацію та характеристики авто, які є в наявності (Рис.15).

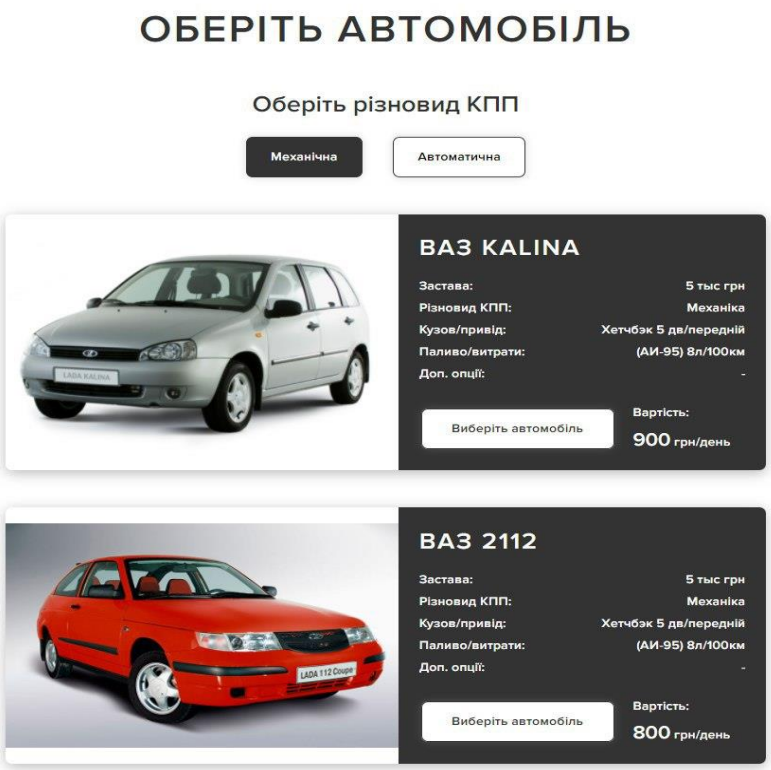

Рис.*15 Меню вибору автомобіля*

Також на сайті зазначено умови, виконавши які, клієнт може орендувати авто (Рис.16).

Рис.16 Умови

Користувач має можливість отримати зворотній зв'язок з менеджером та отримати консультацію і відповіді на питання, що його турбують (Рис.17).

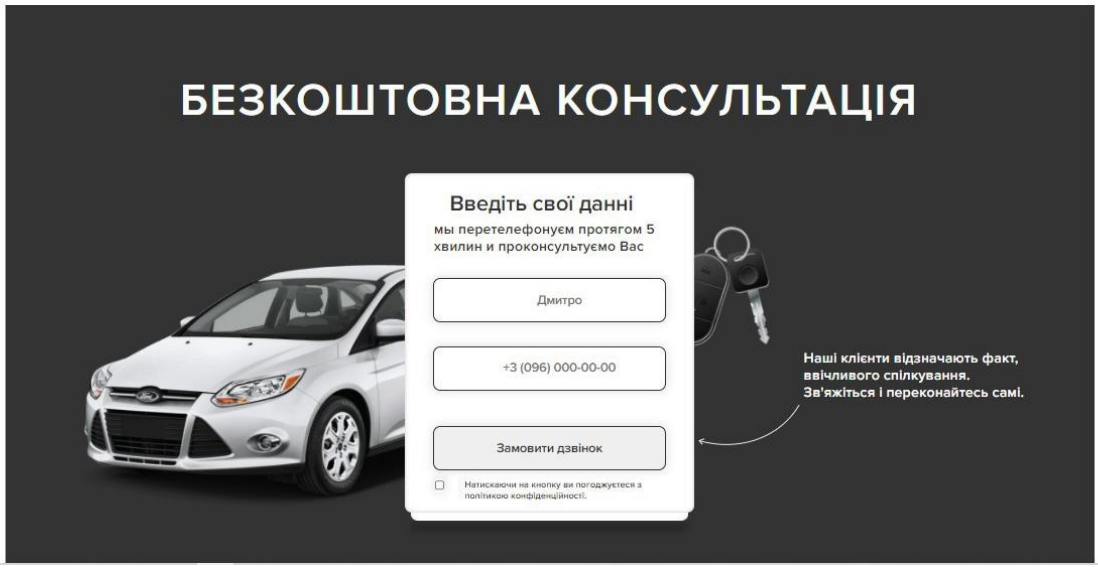

Рис.*17 Зворотній зв'язок*

На наведених рисунках 18, 19 та 20 представлена додаткова інформація для клієнта, а саме: основні переваги сервісу та порівняння з ринком, відгуки клієнтів, найбільш часті запитання, на які можна отримати відповідь на сайті просто натиснувши на нього.

Рис.*18 Основні переваги сервісу*

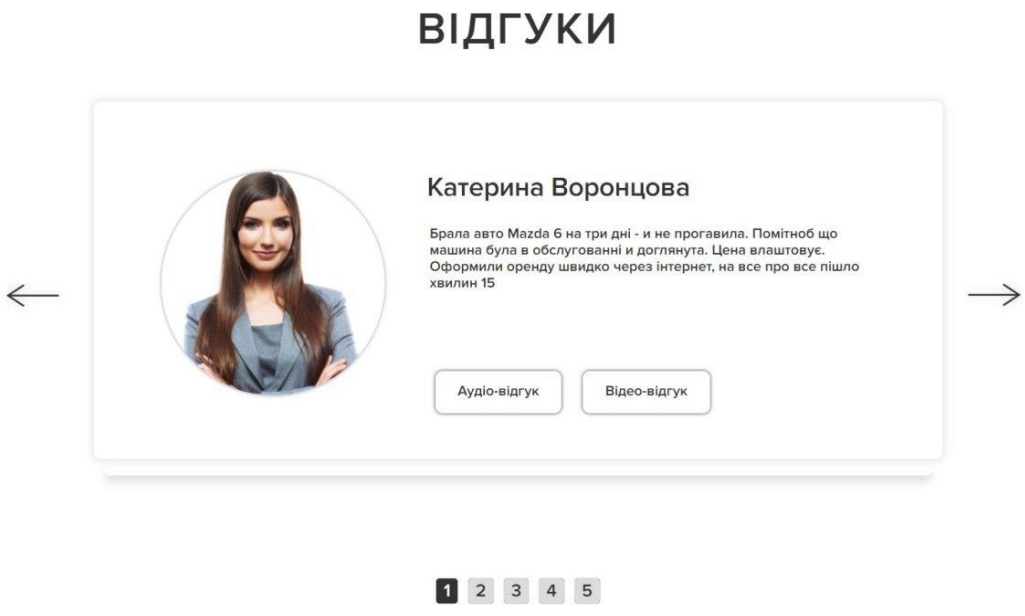

Рис.*19 Відгуки клієнтів*

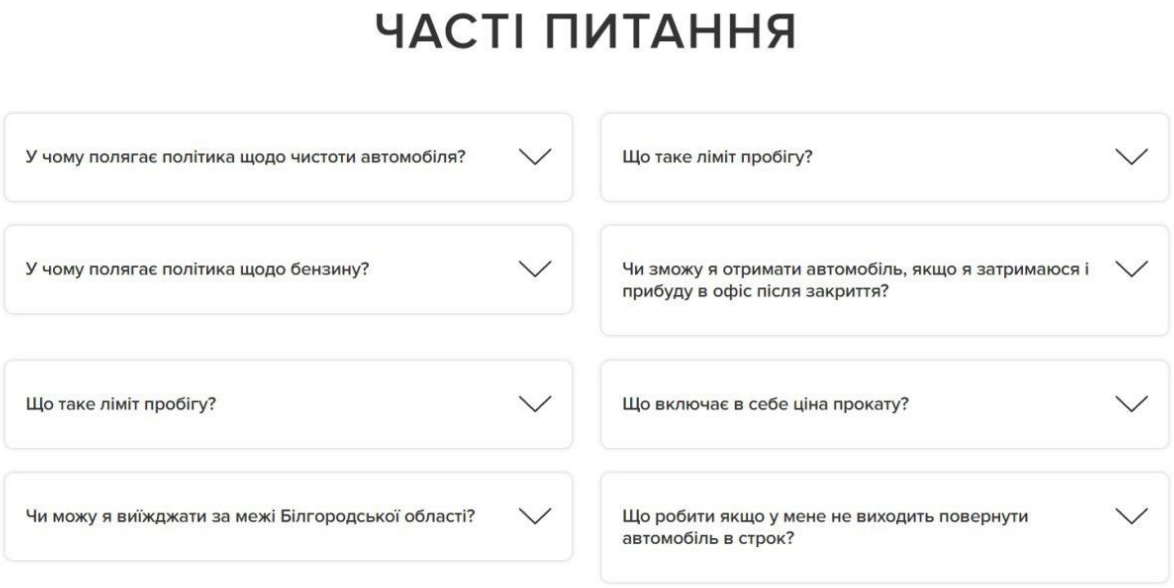

Рис.*20 Відповіді на питання*

3.2.3 Технічна частина проекту

3.2.3.1 Реалізація на Angular

Angular представляє фреймворк від компанії Google для створення клієнтських програм. Насамперед він націлений розробку SPA (Single Page Application), тобто односторінкових додатків. У цьому плані Angular є спадкоємцем іншого фреймворку AngularJS. У той же час Angular – це не нова версія AngularJS, а принципово новий фреймворк. Angular надає таку функціональність як двостороннє зв'язування, що дозволяє динамічно змінювати дані в одному місці інтерфейсу при зміні даних моделі в іншому, шаблони, маршрутизація і так далі. Однією з ключових особливостей Angular є те, що він використовує мову програмування TypeScript. Але це не означає що можливо писати лише мовою TypeScript. За бажанням можемо писати програми

на Angular за допомогою таких мов як Dart або JavaScript. Однак TypeScript є основною мовою для Angular [19].

Проект оренди авто, який реалізовано за допомогою фреймворка Angular, розбито на тринадцять компонентів для зручної навігації по проекту та більшого розуміння структури та архітектури. На рисунку 21 зображено UML-діаграму проекту.

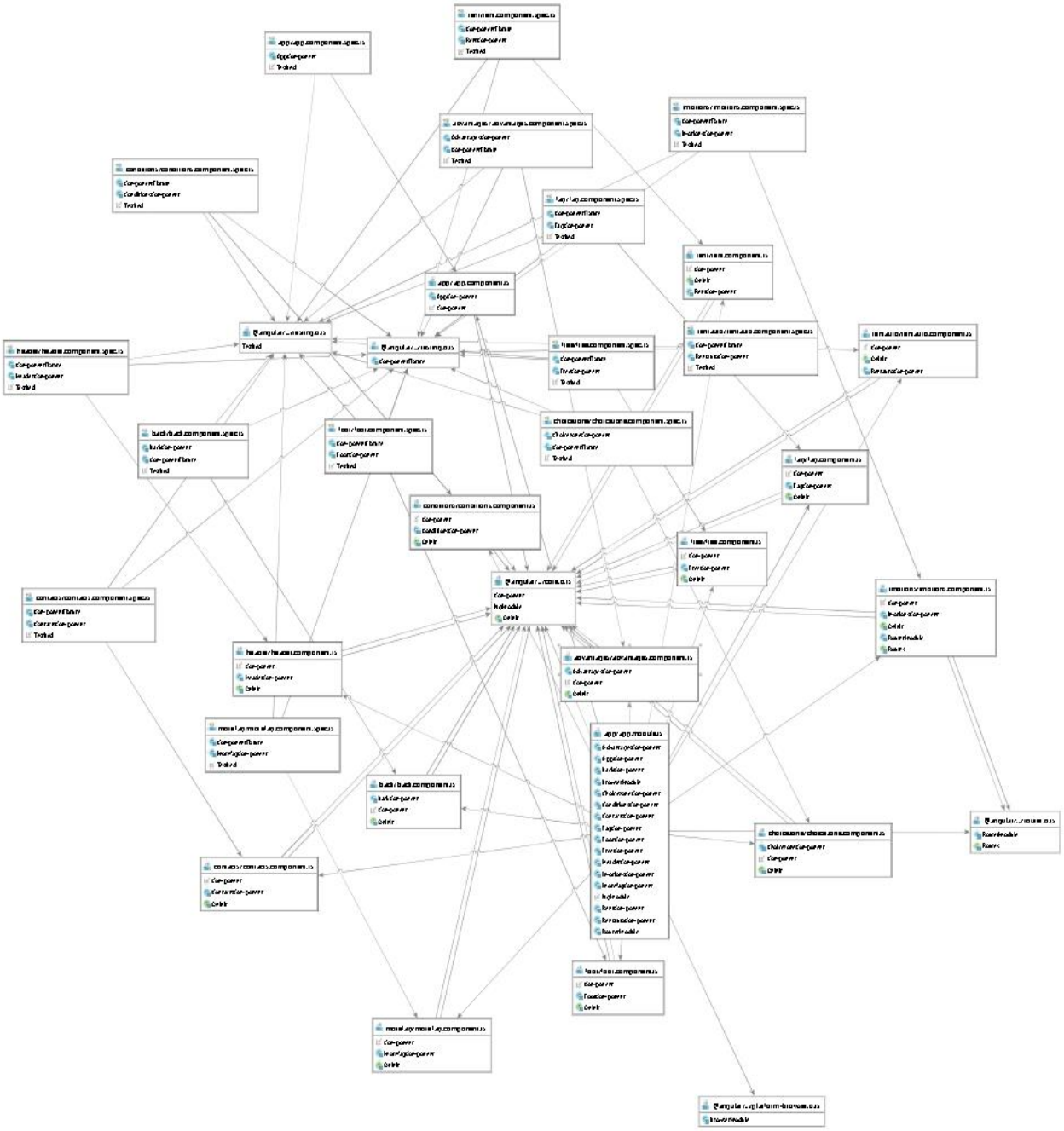

Рис.*21 UML-діаграма всіх компонентів проекту (Angular)*

Компоненти:

Header — заголовочний компонент сайту, який містить коротку інформацію про компанію (рис.22). Структуру компоненту наведено у лістингу 1.

Рис.*22 UML-діаграма компоненту Header*

Лістинг *1 Компонент Header*

```
import { Component, OnInit } from '@angular/core';
@Component({
selector: 'app-advantages',
templateUrl: './advantages.component.html',
styleUrls: ['./advantages.component.css']
})
export class AdvantagesComponent implements OnInit {
constructor() { }
ngOnInit(): void {
```

Rent – рекламний елемент сайту з навігаційною кнопкою (Рис.23). Структуру компоненту наведено у лістингу 2.

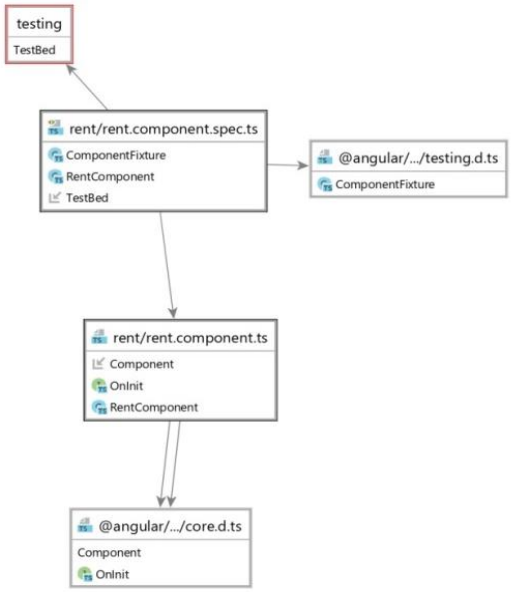

Рис.*23 UML-діаграма компоненту Rent*

Лістинг *2 Компонент Rent*

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-rent',
  templateUrl: './rent.component.html',
  styleUrls: ['./rent.component.css']
})
export class RentComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {
  }
}
```

Back — інформація про поточні досягнення компанії (Рис.24). Структуру компоненту наведено у лістингу 3.

Лістинг *3 Компонент Back*

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-back',
  templateUrl: './back.component.html',
```

```
  styleUrls: ['./back.component.css']
})
export class BackComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {
  }
}
```

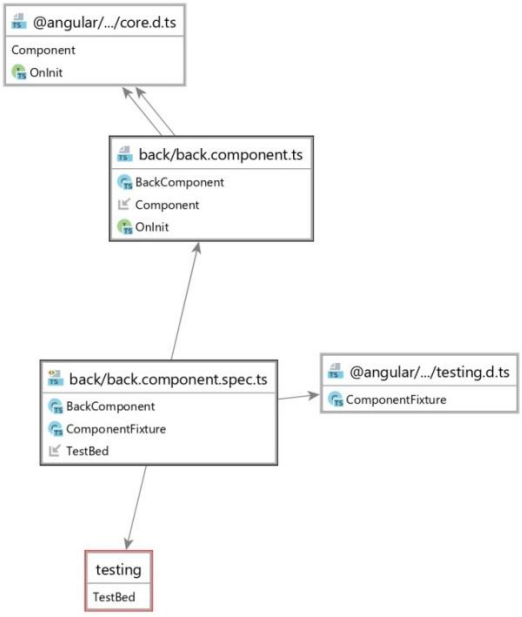

Рис.24 UML-діаграма компоненту Back

ChoiceZone — секція вибору автомобіля (Рис.25). Структуру компоненту наведено у лістингу 4.

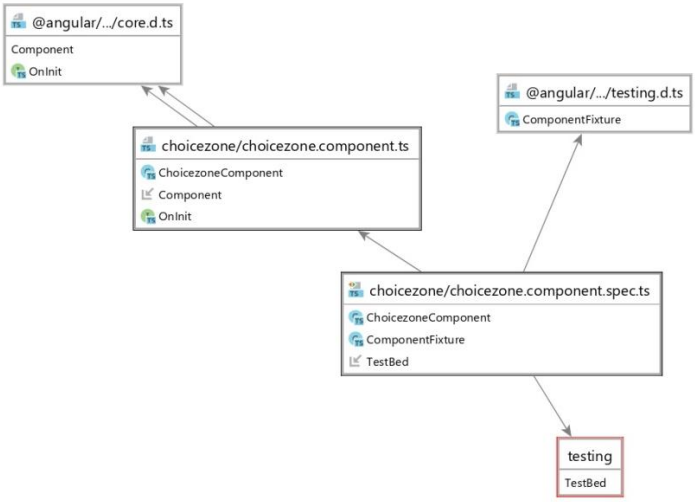

Рис.*25 UML-діаграма компоненту ChoiceZone*

Лістинг *4 Компонент Choicezone*

```
import {Component, OnInit} from '@angular/core';
@Component({
  selector: 'app-choicezone',
  templateUrl: './choicezone.component.html',
  styleUrls: ['./choicezone.component.css']
})
export class ChoicezoneComponent implements OnInit {
  AutoActive = '';
  autoDetailAct: string;
  TechActive = '';
  techDetailAct = 'HideTechDetail';
  showAuto(SelectorAuto): void {
    this.AutoActive = SelectorAuto;
    this.TechActive = 'Choice NotActive ';
    this.autoDetailAct = 'ShowTechDetail';
    this.techDetailAct = 'HideTechDetail';
  }
  showTech(SelectorAuto): void {
    this.TechActive = SelectorAuto;
    this.AutoActive = 'Choice NotActive ';
    this.techDetailAct = 'ShowAutoDetail';
    this.autoDetailAct = 'HideAutoDetail';
  }
  constructor() {
  }
  ngOnInit(): void {
  }
}
```

Conditions — умови оренди автомобіля (Рис.26). Структуру компоненту наведено у лістингу 5.

Лістинг *5 Компонент Conditions*

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-conditions',
  templateUrl: './conditions.component.html',
```

```
  styleUrls: ['./conditions.component.css']
})
export class ConditionsComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {
  }
}
```

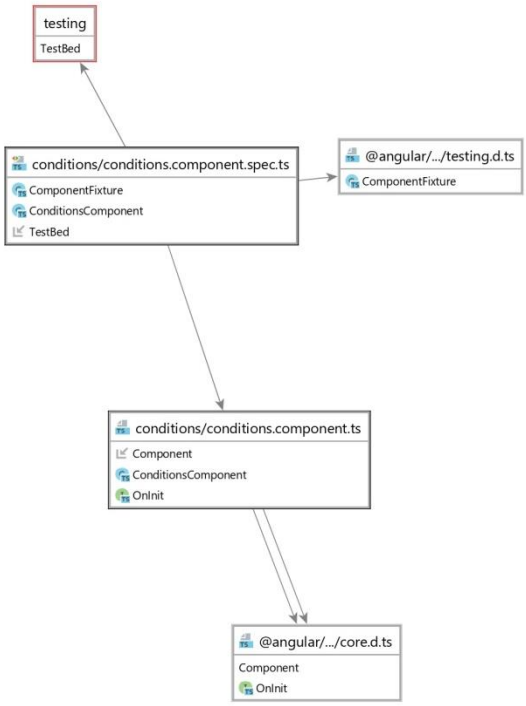

Рис.*26 UML-діаграма компоненту Conditions*

Free — безкоштовна консультація (Рис.27). Структуру компоненту наведено у лістингу 6.

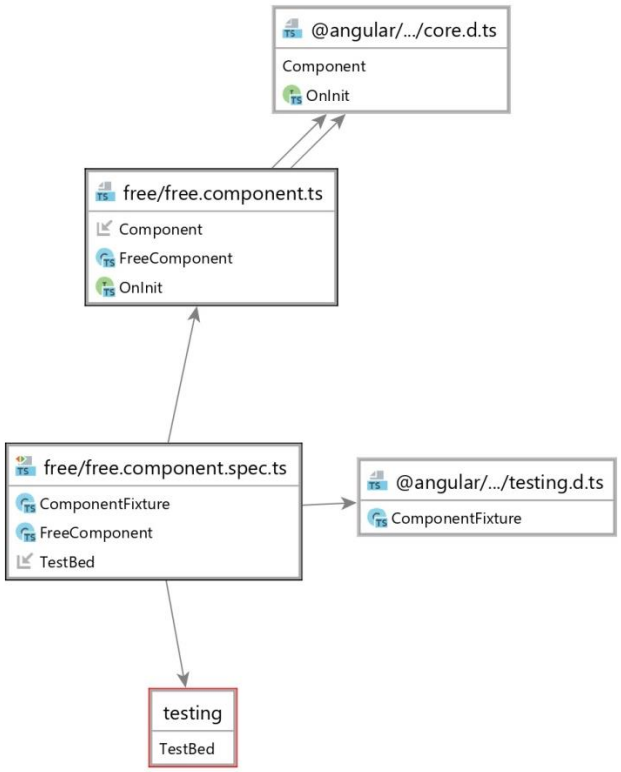

Рис.27 UML-діаграма компоненту Free

Лістинг *6 Компонент Free*

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-free',
  templateUrl: './free.component.html',
  styleUrls: ['./free.component.css']
})
export class FreeComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {
  }
}
```

Advantages — переваги перед типовими представниками (Рис.28). Структуру компоненту наведено у лістингу 7.

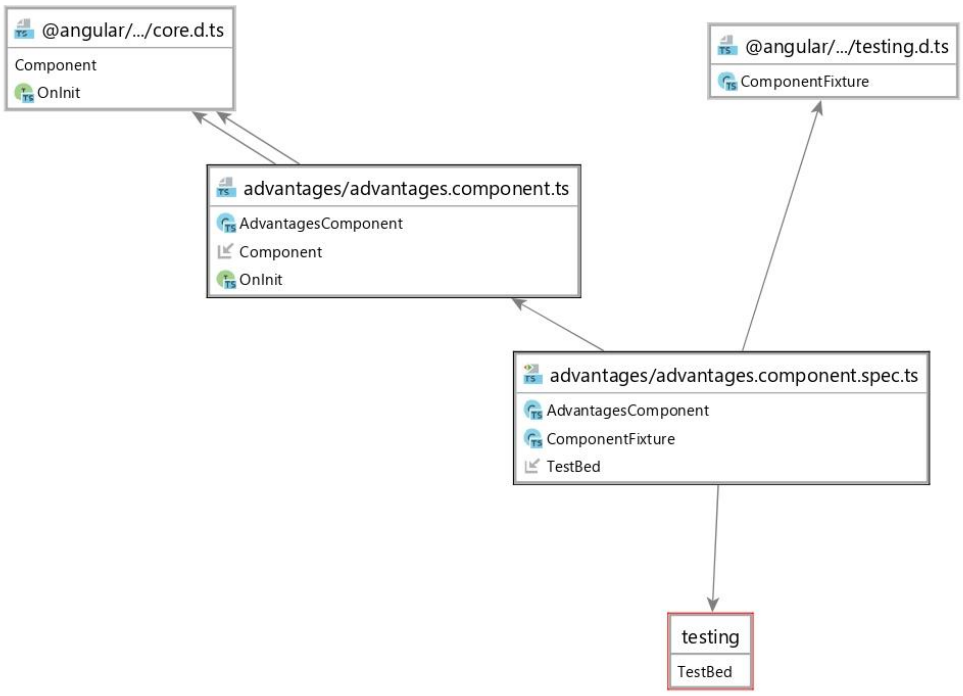

Рис.*28 UML-діаграма компоненту Advantages*

Лістинг *7 Компонент Advantages*

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-advantages',
  templateUrl: './advantages.component.html',
  styleUrls: ['./advantages.component.css']
})
export class AdvantagesComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {
  }
}
```

Emotions — відгуки реальних користувачів (Рис.29). Структуру компоненту наведено у лістингу 8.

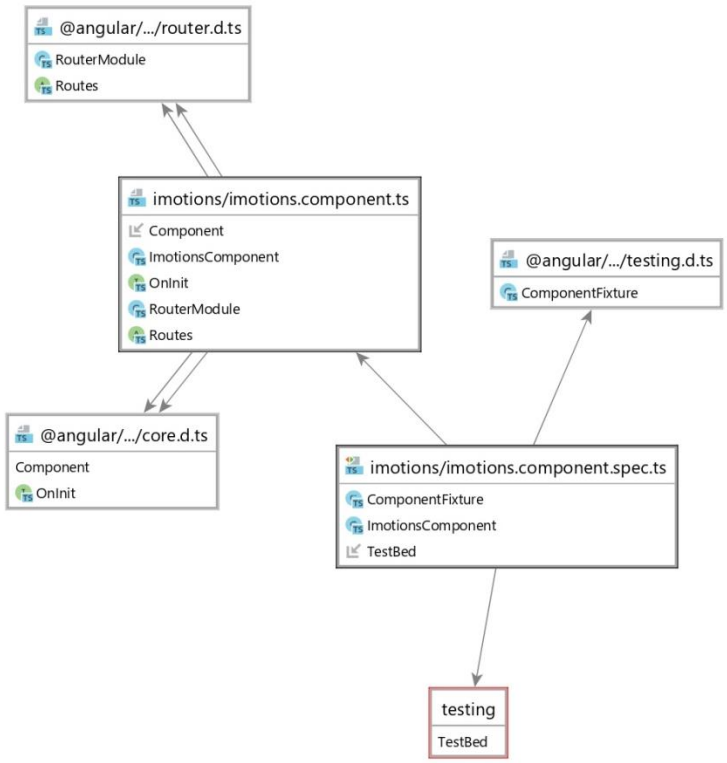

Рис.*29 UML-діаграма компоненту Emotions*

Лістинг *8 Компонент Emotions*

```
import {Component, OnInit} from '@angular/core';
import {RouterModule, Routes} from '@angular/router';
@Component({
  selector: 'app-imotions',
  templateUrl: './imotions.component.html',
  styleUrls: ['./imotions.component.css']
})
export class ImotionsComponent implements OnInit {
  counter = 0;
  count = 0;
  IsIt: any = 0;
  allTags: NodeListOf<any>;
  ActTag;
  AllSlides: NodeListOf<any>;
  AllSlideCount = 0;
  ActiveSlide;
  constructor() {
  }
  startIt(ChangeIt, ChangeItTwo): void {
```

```
    if (!this.IsIt) {
      this.allTags = document.querySelectorAll(ChangeIt);
      this.count = this.allTags.length - 1;
      this.ActTag = this.allTags[0];
      this.AllSlides = document.querySelectorAll(ChangeItTwo);
      this.ActiveSlide = this.AllSlides[0];
      this.AllSlideCount = this.AllSlides.length - 1;
      this.IsIt = 1;
    }
  }
  moveNew(): void {
    this.allTags[this.counter].setAttribute('class',
'active');
    this.AllSlides[this.counter].setAttribute('class', 'slider
show');
    this.ActTag = this.allTags[this.counter];
    this.ActiveSlide = this.AllSlides[this.counter];
  }
  back(ChangeIt, NotAct, ChangeItTwo): void {
    this.startIt(ChangeIt, ChangeItTwo);
    this.ActTag.setAttribute('class', NotAct);
    this.ActiveSlide.setAttribute('class', 'slider');
    this.counter > 0 ? this.counter-- : this.counter =
this.count;
    this.moveNew();
  }
  forward(ChangeIt, NotAct, ChangeItTwo): void {
    this.startIt(ChangeIt, ChangeItTwo);
    this.ActTag.setAttribute('class', NotAct);
    this.ActiveSlide.setAttribute('class', 'slider');
    this.counter < this.count ? this.counter++ : this.counter
= 0;
    this.moveNew();
  }
  indicateChange(ChangeIt, NotAct, ChangeItTwo, numberSlide):
void {
    this.startIt(ChangeIt, ChangeItTwo);
    this.ActTag.setAttribute('class', NotAct);
    this.ActiveSlide.setAttribute('class', 'slider');
    this.counter = numberSlide.textContent - 1;
    this.moveNew();
  }
  ngOnInit(): void {
  }
}
```

RentAuto — короткий опис того, як проходить процес оредни авто (Рис.30). Структуру компоненту наведено у лістингу 9.

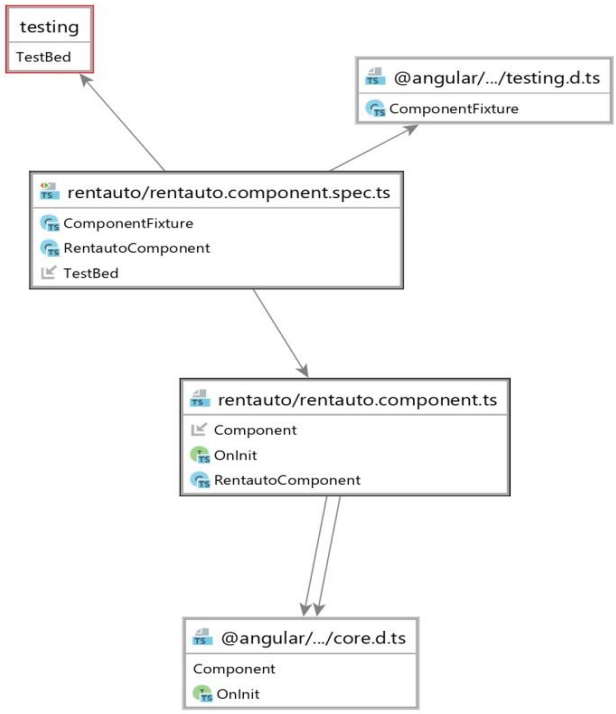

Рис.*30 UML-діаграма компоненту RentAuto*

Лістинг *9 Компонент RentAuto*

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-rentauto',
  templateUrl: './rentauto.component.html',
  styleUrls: ['./rentauto.component.css']
})
export class RentautoComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {
  }
}
```

Faq — важливі питання та відповіді на них (Рис.31). Структуру компоненту наведено у лістингу 10.

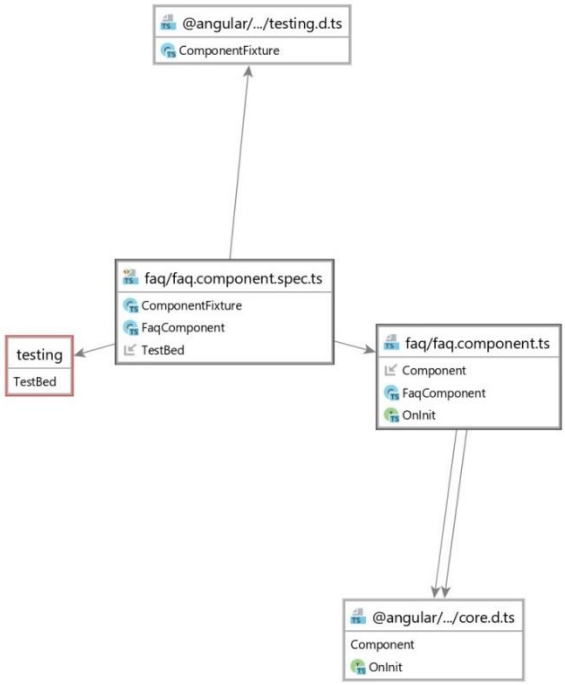

Рис.*31 UML-діаграма компоненту Faq*

Лістинг *10 Компонент Faq*

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-faq',
  templateUrl: './faq.component.html',
  styleUrls: ['./faq.component.css']
})
export class FaqComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {
  }
}
```

Contact — форма зворотного зв'язку (Рис.32). Структуру компоненту наведено у лістингу 11.

Лістинг *11 Компонент* Contacts

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-contacts',
  templateUrl: './contacts.component.html',
  styleUrls: ['./contacts.component.css']
})
export class ContactsComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {
  }
}
```

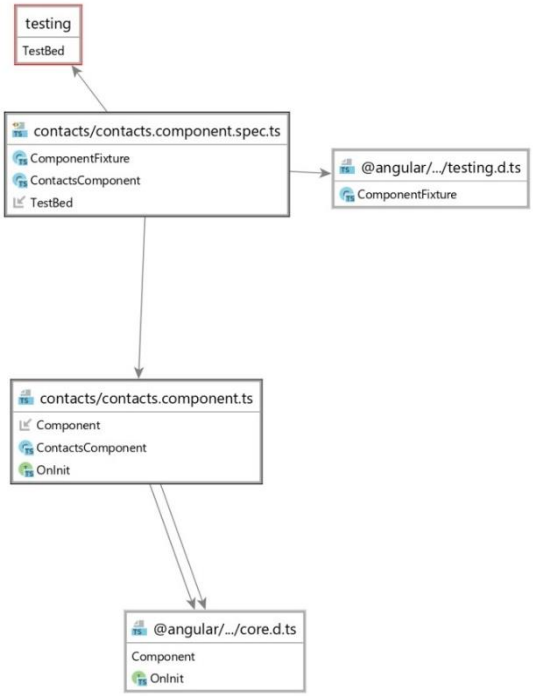

Рис.*32 UML-діаграма компоненту Contact*

Foot — контактна інформація з компанією та посилання на соціальні мережі (Рис.33). Структуру компоненту наведено у лістингу 12.

Лістинг *12 Компонент Foot*

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-foot',
  templateUrl: './foot.component.html',
  styleUrls: ['./foot.component.css']
})
export class FootComponent implements OnInit {
  constructor() { }
```

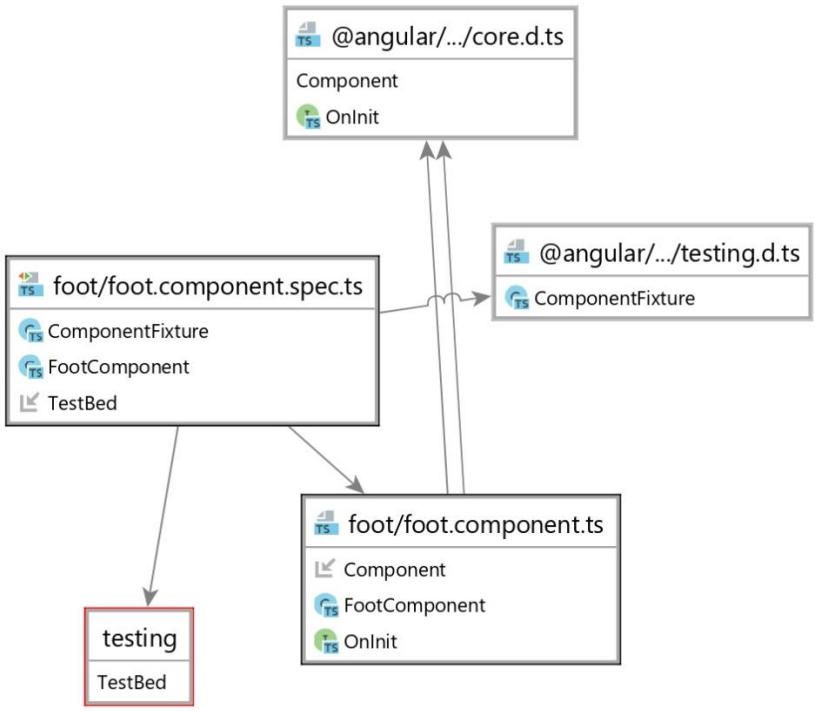

Рис.*33 UML-діаграма компоненту Foot*

3.2.3.2 Реалізація на React

React — це бібліотека JavaScript, яка використовується для створення інтерфейсу користувача. React було створено компанією Facebook, а перший реліз бібліотеки відбувся у березні 2013 року. Поточною версією на даний момент є версія React v17.0. Спочатку React призначався для Інтернету, для створення сайтів, але потім з'явилася платформа React Native, яка вже призначалася для мобільних пристроїв. React представляє ідеальний інструмент для створення масштабованих веб-додатків (в даному випадку йдеться про

фронтенд), особливо в тих ситуаціях, коли програма представляє SPA (односторінковий додаток). React відносно простий у освоєнні, має зрозумілий та лаконічний синтаксис.

Проект оренди авто який реалізовано за допомогою фреймворка React розбито також на тринадцять компонентів як і реалізанця на Angular. Для зручності порівняння всі назви та властивості компонентів ідентичні. На рисунку 34 зображено UML-діаграму реалізації на React. UML-діаграми компонентів представлено на рисунках 35-46 [20].

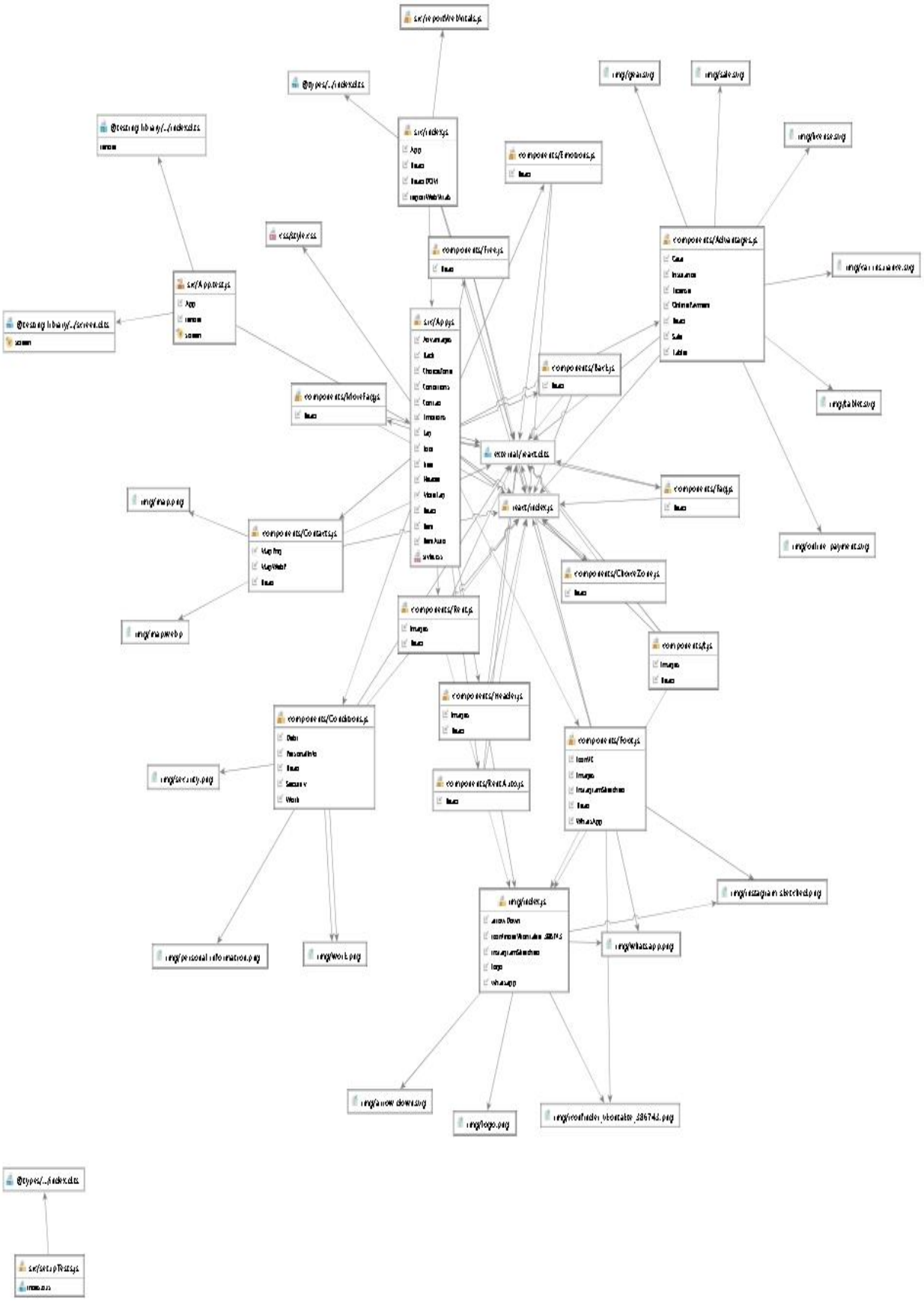

Рис.34 *UML-діаграма всіх компонентів проекту(React)*

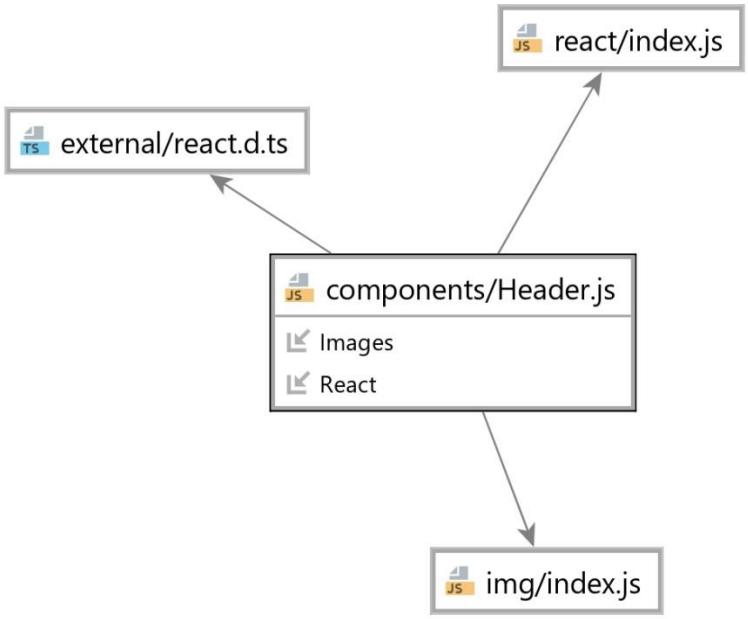

Рис.35 *UML-діаграма компоненту Header*

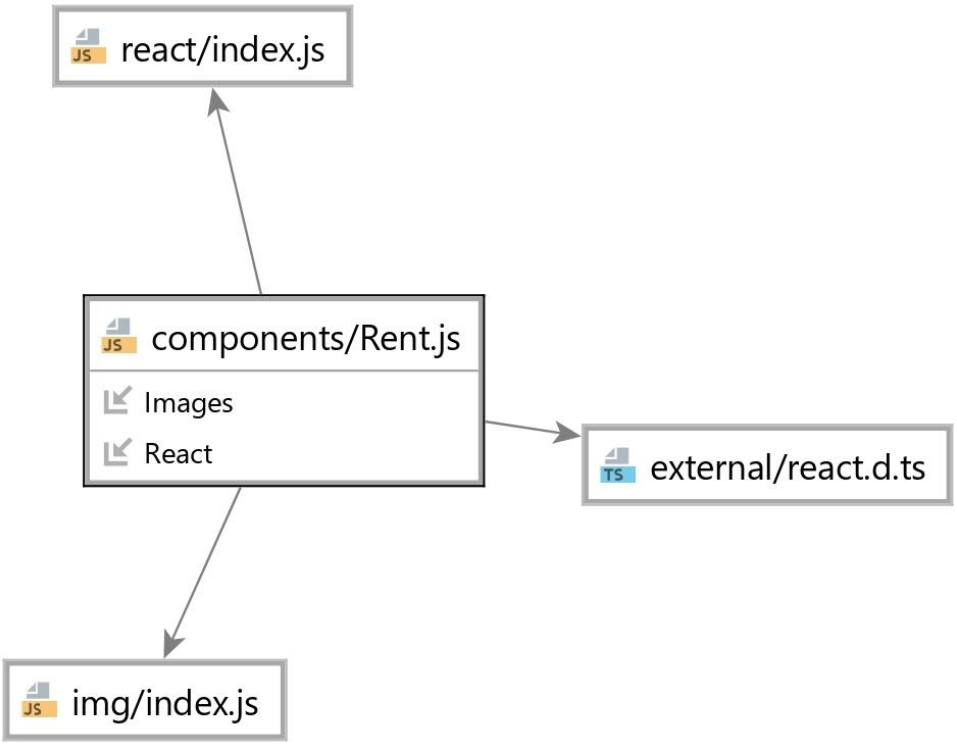

Рис.36 *UML-діаграма компоненту Rent*

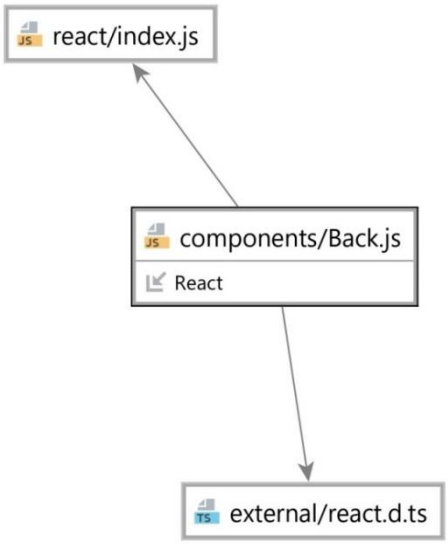

Рис.*37 UML-діаграма компоненту Back*

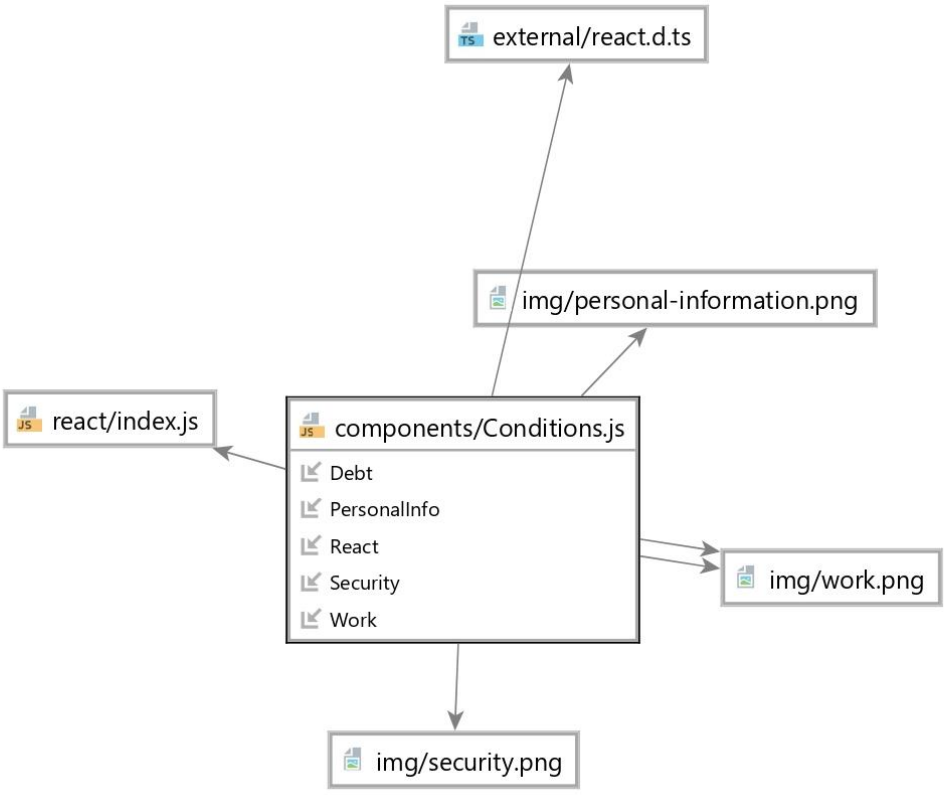

Рис.*38 UML-діаграма компоненту Conditions*

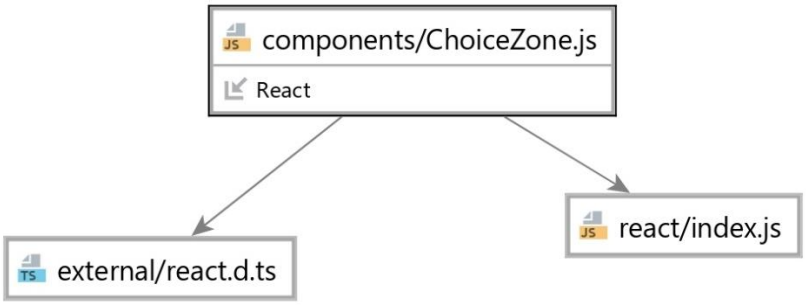

Рис.*39 UML-діаграма компоненту ChoiceZone*

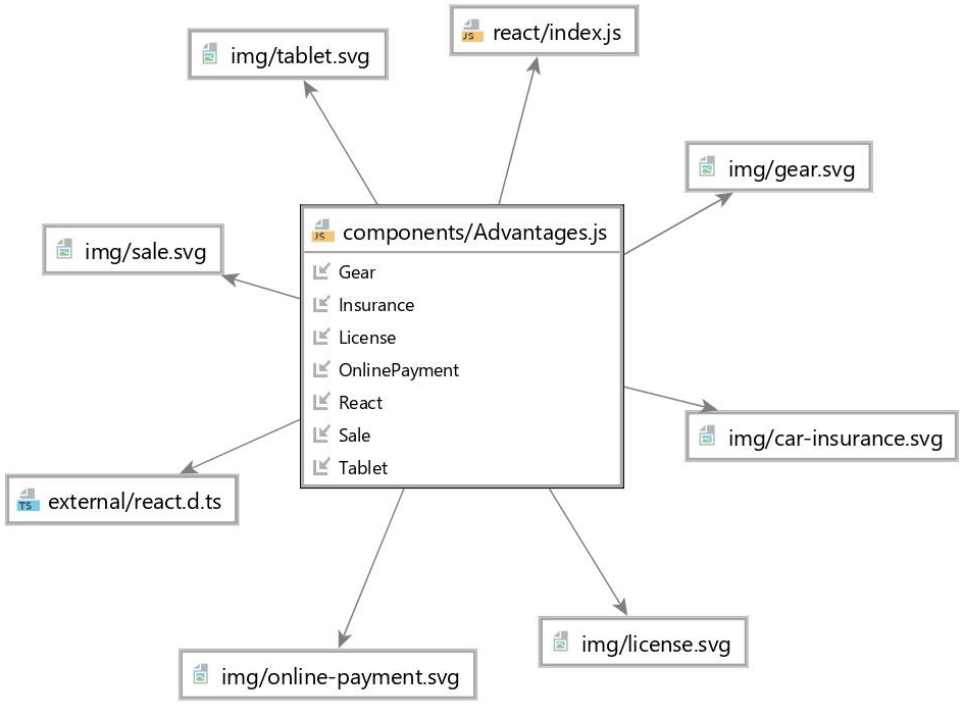

Рис.*40 UML-діаграма компоненту Advantages*

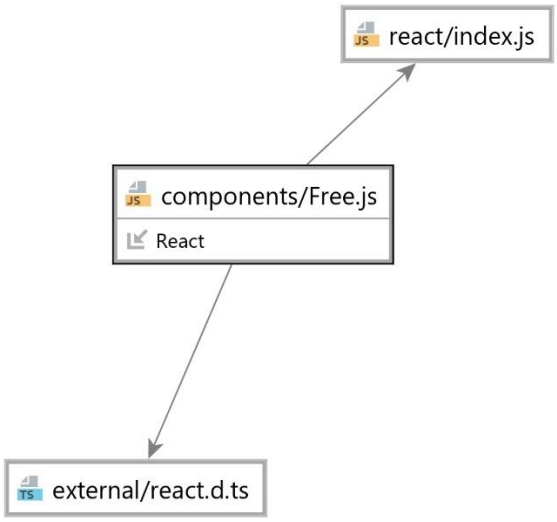

Рис.*41 UML-діаграма компоненту Free*

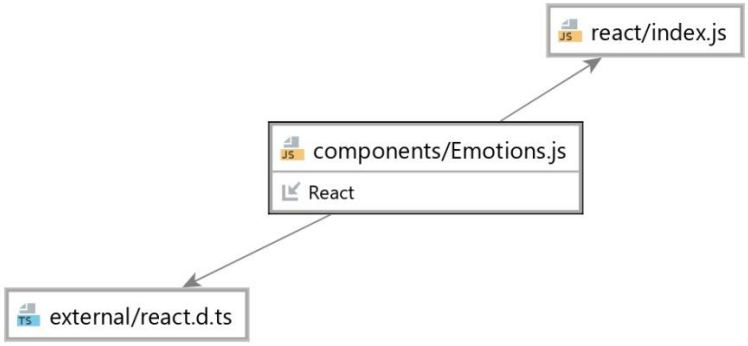

Рис.*42 UML-діаграма компоненту Emotions*

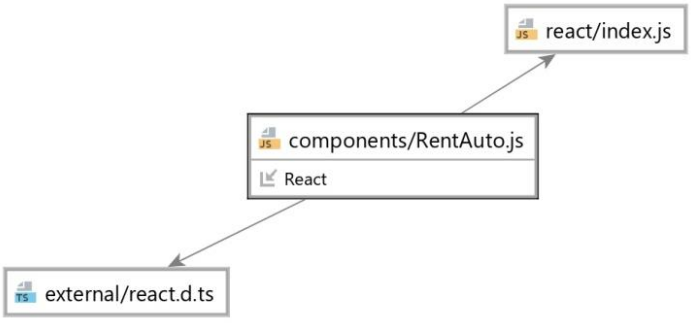

Рис.*43 UML-діаграма компоненту RentAuto*

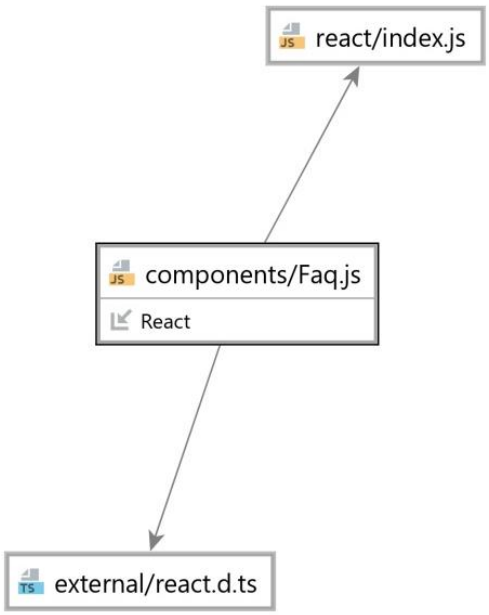

Рис.*44 UML-діаграма компоненту Faq*

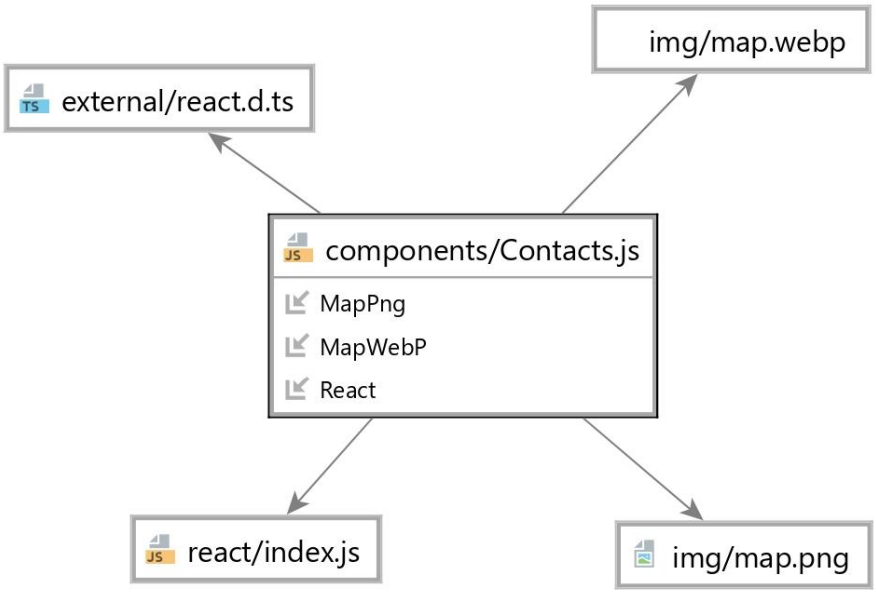

Рис.*45 UML-діаграма компоненту Contact*

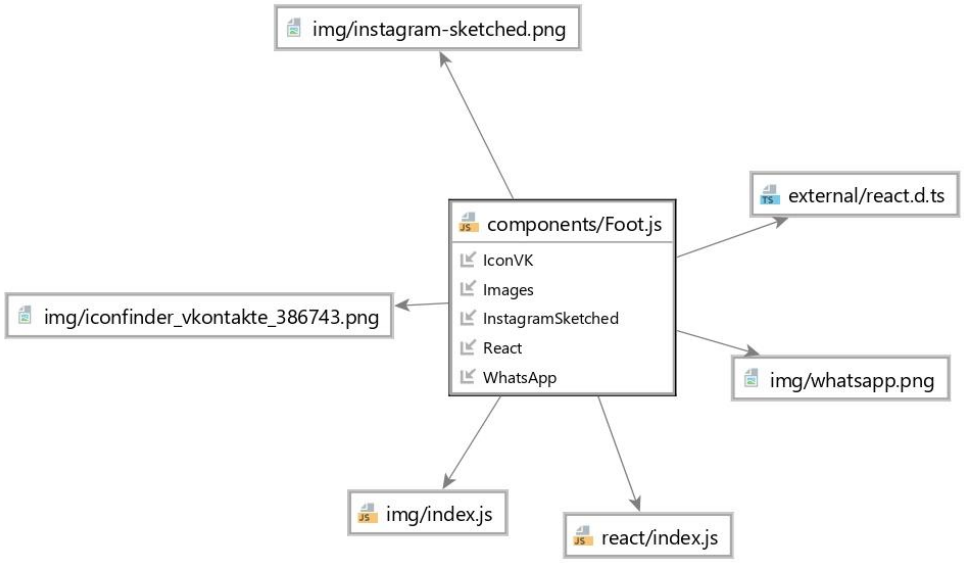

Рис.*46 UML-діаграма компоненту Foot*

3.3 Засоби реалізації

Для реалізації програмного продукту було використано наступні засоби:

- середовище розробки PHPShtorm 2021;
- платформа NodeJS;
- фреймворки Angular, React.

Середовище розробки PHPShtorm 2021

PhpStorm — це середовище розробки. IDE забезпечує підтримку PHP 5.3/5.4/5.5/5.6/7.0/7.1/7.2, миттєво запобігає помилкам, надає точне автодоповнення та безпечні рефакторинги, а також можливість редагування коду на HTML, CSS та JavaScript. PhpStorm забезпечує налагодження коду JavaScript і надає широкий діапазон можливостей: знаходження точки зупинки в HTML і JavaScript, налаштування параметрів точки зупинки, тестування синтаксису коду в режимі реального часу [21].

Платформа NodeJS

Node.js — це середовище виконання коду JavaScript, яке побудоване на основі JavaScript Chrome V8, яке дозволяє транслювати виклики мовою JavaScript в машинний код. Це спільна платформа, на якій запускається код Angular та React. Node.js насамперед призначений для написання серверних програм на мові JavaScript. Хоча також існують проекти написання десктопних додатків і навіть написання коду для мікроконтролерів. Але перш за все Node.js — це платформа для створення веб-додатків [22].

РОЗДІЛ 4. ПОРІВНЯЛЬНИЙ АНАЛІЗ ФРЕЙМВОРКІВ НА ОСНОВІ РОЗРОБЛЕНОГО ПРОЕКТУ

4.1 Перевірка швидкості завантаження сторінки

Щоб перевірити швидкість завантаження сторінки було використано ChromeDevTools.

Chrome DevTools — це набір інструментів для веб-розробників, вбудованих безпосередньо у браузер Google Chrome.

Швидкість завантаження сторінки– один із факторів, що враховують пошукові системи для ранжування сайту.

За допомогою ChromeDevTools було перевірено такі параметри як завантаження, сценарії, візуалізація, прорисовка, завантаження системи (Рис 46,47). Сходячи з цих параметрів швидкість загрузки сторінки на Angular склала – 1491 ms, а на React 624 ms. React загрузився швидше ніж Angular на 867ms. Це все завдяки Virtual DOM. Додавання та видалення вузлів DOM займає менше ніж встановлення властивості для об’єкта JavaScript яке використовується в Angular.

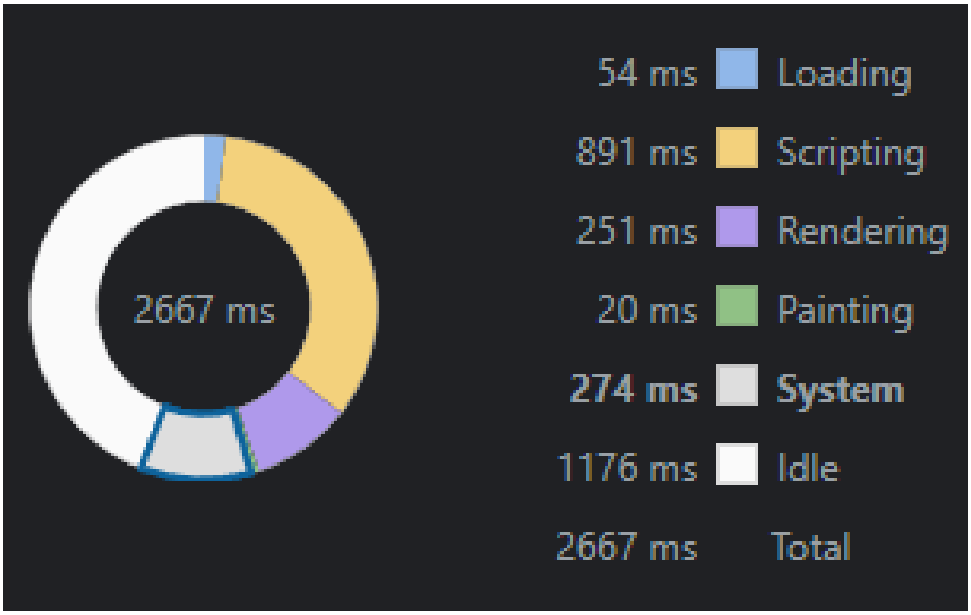

Рис. *46 Швидкіть завантаження сторінки на Angular*

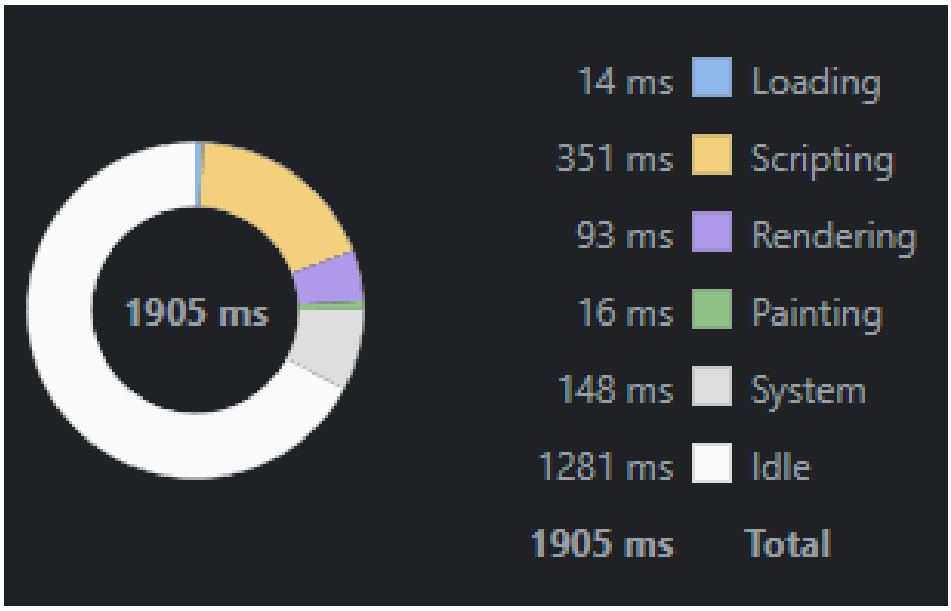

Рис. *47 Швидкість завантаження сторінки на React*

4.2 Порівняння ваги проекту

Для отримання точних орієнтирів, які можна було б використати для порівняння, зроблено наступне:

- Створено кожну програму та запущено через ng build-prod для Angular та npm run build React.
- Створено стислі .gzip версії кожного сформованого файлу JavaScript шляхом запуску gzip <file-name>.js - keep.
- Запущено http-server-gzip у відповідних створених папках збірки кожного додатка, щоб був локальний HTTP-сервер, який можливо було б використовувати для тестування кожної програми.
- Відкрито індексну сторінку кожного додатка в новому, анонімному вікні Chrome.

Збірка Angular закінчилася трьома файлами JavaScript, які були об'єднані 93,0 КБ після стиснення gzip (Рис.48).

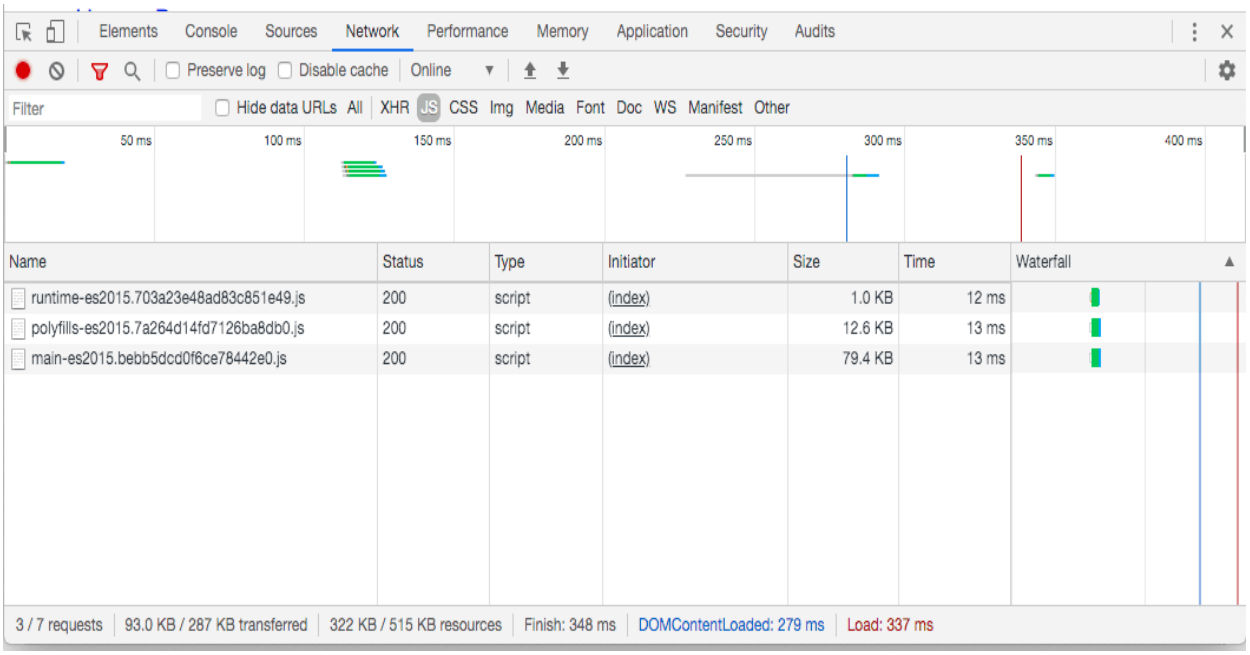

Рис.*48 Збірка Angular*

Збірка React закінчилася двома файлами JavaScript, які були об'єднані, 4 КБ після стиснення gzip (Рис.49).

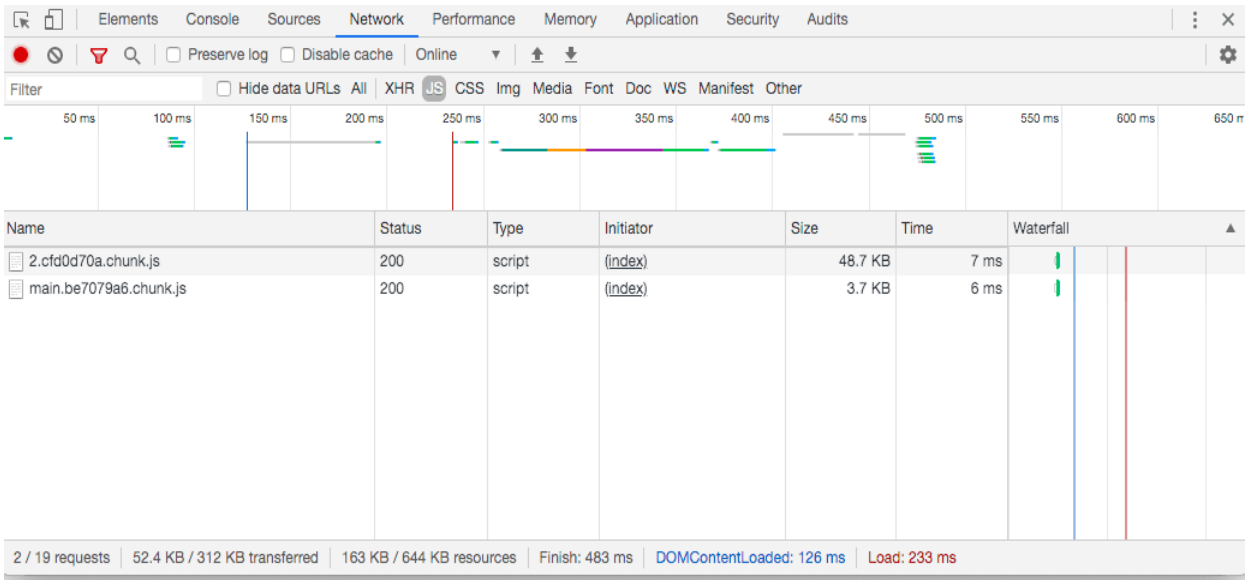

Рис.*49 Збірка React*

Загалом різниця між вагою збірок файлів Angular та React склала 41 КБ. React зменшив навантаження JavaScript з 93 КБ до 52 КБ, це незначна різниця, але досить помітна при розробці великих застосунків.

4.3 Порівняння кількості компонентів

Кількість компонентів при розробці одного і того ж за стосунку виявилась ідентичною по тринадцяті компонентах в кожній реалізації (Рис.50,51). Для простоти порівняння будо додатково винесено асети та стилі.

```
<Header/>
<Rent/>
<Back/>
<ChoiceZone/>
<Conditions/>
<Free/>
<Advantages/>
<Emotions/>
<RentAuto/>
<Faq/>
<MoreFaq/>
<Contact/>
<Foot/>
```

Рис.*50 Компоненти Angular*

```
<app-header class="top-info"></app-header>
<app-rent></app-rent>
<app-back></app-back>
<app-choicezone [ngClass]="{'first': true, 'second': true, 'third': false}"></app-choicezone>
<app-conditions></app-conditions>
<app-free></app-free>
<app-advantages class="advantage"></app-advantages>
<app-imotions></app-imotions>
<app-rentauto></app-rentauto>
<app-faq></app-faq>
<app-morefaq></app-morefaq>
<app-contacts></app-contacts>
<app-foot></app-foot>
```

Рис. *51 Компоненти React*

4.4 Порівняння кількості коду

Реалізація застосунку на Angular мала 755 рядків коду, а на React 806 рядків. Різниця склала 51 рядок в користь Angular. Тобто однаковий застосунок швидше буде написати за допомогою Angular.

4.5 Порівняння структури

React має просту структуру проекту за замовчуванням в порівнянні з Angular. Маленьке дерево файлів спрощує навігацію програмою під час кодування, але справжньою перевагою є архітектура створеного додатка,оскільки компоненти React значно легші для створення та використання. Простота React не стосується лише його файлової структури та моделі компонентів. В Angular не є зручним набір залежностей, які фреймворк встановлює під час запуску програми. Наприклад, у великих програмах Angular я виявив, що відстеження моїх власних залежностей та залежностей devDependencies стає проблематичним, оскільки вони можуть загубитися серед усіх залежностей Angular. Структура React полегшує роботу розробника тим, що просте дерево файлів, модель компонентів та список залежностей. Хоча React візуально робить ці процеси простими, з програмної сторони відбувається набагато більше [23].

4.5 Порівняння способів обробки CSS

В Angular CSS працює в одному напрямку. Коли визначається компонент, можна надати styleUrls властивість метаданих і передати цій властивості масив URL-адрес таблиці стилів. В Angular CSS, який пишеться у цих файлах, охоплюється таким компонентом. Вважається, що цей спосіб обробки CSS досить зручний та ефективний. Поміщаючи всі правила для всього додатка та спільні

імена класів у .css файл на рівні додатка, а потім розміщуючи свій стиль для окремих компонентів у файлах із областю застосування досить легко та зручно.

Використання CSS у React дозволяє писати серію імен класів, які за замовчуванням націлені на компонент. Це дає обмеження використання лише назв класів у файлах CSS, і тому потрібно вручну застосувати ці назви класів за допомогою JSX і це може стати незручно та мати велику кількість помилок у реальних програмах, де потрібно керувати багатьма назвами класів. Іншим популярним варіантом стилізації на рівні компонентів у React є використання техніки під назвою CSS-in-JS, це дозволяє безпосередньо застосовувати правила CSS у JavaScript. Така методологія має певні проблеми з продуктивністю, оскільки стилізовані компоненти застосовують CSS під час виконання, а не під час збірки.

Виходячи з цих плюсів та мінусів, можна зробити висновок, що Angular має перевагу над React. Angular має один розумний та ефективний спосіб обробки CSS [24-26].

4.6 Головні відмінності та переваги фреймворків

Кожен із розглянутих фреймворків є досить потужним та має свої особливості. Аналізуючи проведені порівняння можна виділити такі переваги та недоліки Angular та React:

Переваги Angular:

- Велика кількість функцій.
- Функції взаємозалежні.
- Можливість безпосередньо отримувати інформацію.
- Можливість працювати окремо в одному розділі програми, використовуючи наявні дані.

- Мінімальний ризик помилок.

Недоліки Angular:

- В основі складна мова програмування.
- Помилки під час міграції між версіями.

Переваги React:

- В основі лежить проста мова програмування.
- Велика гнучкість програми.
- Використання DOM.
- Програма витримує великі навантаження.
- React і SEO. Покращена взаємодія користувачів з розробленим ресурсом.
- Забезпечує незмінність батьківських даних.
- Відкрита бібліотеку даних.
- Невелика вага бази даних.
- Забезпечує просту міграцію між версіями.

Недоліки React:

- Неупорядкованість документації.
- Занадто великий вибір інструментів.
- Для освоєння потрібно витратити тривалий час.

ВИСНОВКИ

1. Проведено дослідження та аналіз сучасних видів як фронтенд так і бекенд фреймворків. Проаналізовано їх архітектури, виділено переваги використання та їх недоліки. Розглянуто варіанти взаємодії між собою.

2. Досліджено актуальність порівняння фреймворків Angular та React, та методів їх порівняння. Виділено ключові критеріїв для порівняння.

3. Досліджено основи фреймворків, виділено їх ключові переваги та недоліки.

4. На основі проведених досліджень та аналізу реалізовано односторінковий застосунок на фронтенд фреймворках Angular та React.

5. Досліджено компоненти та методи реалізації та їх опис.

6. Проведено порівняльний аналізу застосунків на основі таких критеріїв як швидкість завантаження сторінки, вага,кількості компонентів,коду, структура.

Результати роботи можуть бути використані веб-розробниками при проектуванні і розробці сайтів. Це дозволять їм більш точно обрати фреймворк для своїх майбутніх проектів, а також допоможе новачкам орієнтуватися у виборі фреймворка для створення WEB-додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гулак Є. Порівняльний аналіз Web-фреймворків. *Молода наука-2021* : зб. наук. праць студентів, аспірантів і молодих вчених. Запоріжжя : ЗНУ, 2021. Т.5. С. 85–86.

2. Гулак Є., Міхайлуца О.М. Порівняльний аналіз 3 JAVASCRIPT фреймворків для розробки веб-застосунку. Матеріали І Всеукраїнської науково-практичної конференції здобувачів вищої освіти, аспірантів та молодих вчених «Актуальні питання сталого науково-технічного та соціально-економічного розвитку регіонів України» Запоріжжя: ЗНУ, 2021. С. 324-325.

3. What is a Web Framework? URL: https://jeffknupp.com/blog/2014/03/03/what-is-a-web-framework/ (дата звернення: 07.03.2021).

4. Digital, Social and Mobile in 2015 report indicates URL : http://wearesocial.com/sg/special-reports/digital-socialmobile-2015 (дата звернення: 12.03.2021).

5. Результат тестування шести популярних фреймворків на продуктивність URL: http://www.alrond.com/ru/2007/jan/25/rezultaty-testirovanija-6-frameworks/ (дата звернення: 25.03.2021).

6. Використання PHP фреймворків в розробці сайту URL: http://ukrbukva.net/page,5,39718-Ispol-zovanie-PHP-freiymvorkov-v-razrabotke-saiyta.html.

7. Full-Stack React, TypeScript, and Node: Build cloud-ready web applications using React 17 with Hooks and GraphQL. Adam Boduch. Dec 18, 2020. 236 с.

8. Angular 2 Development with TypeScript. Fain Y., Moiseev A. Manning Publications. 2016. 28 с.

9. Angular 2 URL: https://angular.io (дата звернення: 08.04.2021).

10. Learning Angular: A no-nonsense beginner's guide to building web applications with Angular 10 and TypeScript, 3rd Edition. Muhammad Ahsan Ayaz. Sep 7, 2020. 94 с.

11. Angular Development with TypeScript. Aristeidis Bampakos. Dec 17, 2018. 15 с.

12. Beginning Angular with Typescript (updated to Angular 9). Doguhan Uluca. Apr 6, 2020. 33 с.

13. Angular: Up and Running: Learning Angular, Step by Step. Greg Lim. Jun 22, 2018. 32 с.

14. ASP.NET Core 5 and Angular: Full-stack web development with .NET 5 and Angular 11, 4th Edition. Valerio De Sanctis. Jan 29, 2021. 231 с.

15. React 17 Design Patterns and Best Practices: Design, build, and deploy production-ready web applications using industry-standard practices, 3rd Edition. Carlos Santana Roldán. May 17, 2021. 154с.

16. React и Redux. Функціональна веб-розробка Бэнкс Алекс 2018. 450 с.

17. Learning React: Modern Patterns for Developing React Apps. Alex Banks. Jul 7, 2020. 91 с.

18. Full-Stack React, TypeScript, and Node: Build cloud-ready web applications using React 17 with Hooks and GraphQL. Adam Boduch. Dec 18, 2020. 272 с.

19. HTML5 и CSS3. Веб-розробка за стандартами нового покоління Брайан П. Хоган. 2013. 128 с.

20. CSS: The Definitive Guide: Visual Presentation for the Web. Eric A. Meyer. Nov 21, 2017. 86 с.

21. Responsive Web Design with HTML5 and CSS: Develop future-proof responsive websites using the latest HTML5 and CSS techniques, 3rd Edition. Estelle Weyl. Apr 30, 2020. 231 c.

22. Web Design with HTML, CSS, JavaScript and jQuery Set. Ben Frain. Jul 8, 2014. 71 c.

23. CSS in Depth. Jon Duckett. Apr 7, 2018. 22 c.

24. Modern CSS: Master the Key Concepts of CSS for Modern Web Development. Joe Attardi. Oct 7, 2020. 33 c.

25. CSS Pocket Reference: Visual Presentation for the Web. May 3, 2018. 37 c.

26. HTML and CSS: Design and Build Websites. Lea Verou. Nov 8, 2011. 63 c.

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Гулак Євгеній Ігорович, студент _2_ курсу, форми навчання денної, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти se-216@stu.zsea.edu.ua , — підтверджую, що написана мною кваліфікаційна робота на тему «Порівняльний аналіз Web-фреймворків» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений;

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 30.11.2021 Підпис_____ ПІБ (студент) Гулак Євгеній Ігорович

Дата 30.11.2021 Підпис_____ ПІБ (науковий керівник) Олена Миколаївна Міхайлуца