Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ ТА ПОРІВНЯННЯ ФРЕЙМВОРКІВ ДЛЯ РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКІВ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-21-1
Свічкарьов О.В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Творошенко І.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
Освітня програма Інформатика
(повна назва освітньої програми)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Свічкарьову Олександру Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та порівняння фреймворків для розроблення вебзастосунків

затверджена наказом по університету від 9 листопада 2022 року № 1469Ст

2. Термін подання студентом роботи до екзаменаційної комісії 21 листопада 2022 р.

3. Вихідні дані до роботи мова програмування JavaScript та TypeScript, актуальні версії фреймворків React, Vue, Angular, допоміжні бібліотеки для програмування.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд сучасних фреймворків.
2. Аналіз особливостей фреймворків.
3. Побудова вебзастосунку за допомогою актуальних фреймворків.
4. Порівняння ефективності та популярності вибраних фреймворків.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми вибору фреймворку для побудови вебзастосунків, постановка задачі, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	09.11.2022	
2	Аналіз завдання, підбір літератури	10.11.22-12.11.22	
3	Аналіз літератури з досліджуваної проблеми	12.11.22-13.11.22	
4	Аналіз сучасних систем розпізнавання реквізитів банківських карток	13.11.22	
5	Аналіз методів розпізнавання банківських карток	14.11.22	
6	Програмна реалізація	15.11.22-18.11.22	
7	Оформлення пояснювальної записки	18.11.22-19.11.22	
8	Перевірка на плагіат	24.11.2022	
9	Рецензування	25.11.2022	
10	Підготовка презентації та доповіді	27.11.2022	
11	Занесення роботи в електронний архів	29.11.2022	
12	Попередній захист кваліфікаційної роботи	30.11.2022	

Дата видачі завдання 9 листопада 2022 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Творошенко І.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 60 с., 3 табл., 30 рис., 48 джерел.

АНАЛІЗ, ФРЕЙМВОРК, ВЕБЗАСТОСУНОК, ВЕБСАЙТ, FRONTEND, ANGULAR, REACT, VUE, JAVASCRIPT, ПРОГРАМА, БІБЛІОТЕКА, КОД, РОЗРОБКА, ОПТИМІЗАЦІЯ.

Об’єктом дослідження є процес тестування вебзастосунків на трьох різних фреймворках, а саме React, Vue та Angular, що застосовуються на базі мов програмування JavaScript і TypeScript.

Метою дослідження є порівняння вибраних фреймворків задля визначення їх популярності у світі, ефективності та масштабності під час тестування однакових класів задач.

У результаті роботи здійснена програмна реалізація трьох вебзастосунків за допомогою вибраних фреймворків, що виконують важкі операції задля визначення швидкості кожного фреймворку.

ANALYSIS, FRAMEWORK, WEBAPP, WEBSITE, FRONTEND, ANGULAR, REACT, VUE, JAVASCRIPT, PROGRAM, LIBRARY, CODE, DEVELOPMENT, OPTIMIZATION.

The object of the research is the process of testing web applications on three different frameworks, namely React, Vue and Angular, which are used on the basis of the JavaScript and TypeScript programming languages.

The purpose of the research is to compare the selected frameworks to determine their popularity in the world, efficiency and scalability when testing the same classes of tasks.

As a result of the work, the software implementation of three web applications was carried out using selected frameworks that perform heavy operations to determine the speed of each framework.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПК – персональний комп’ютер

SPA – Single Page Application (односторінковий вебзастосунок)

HTML – HyperText Markup Language (мова розмітки гіпертексту)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

PWA – Progressive Web App (прогресивний вебзастосунок)

DOM – Document Object Model (об’єктна модель документа)

VDOM – Virtual Document Object Model (віртуальна об’єктна модель документа)

MVVM – Model-View-ViewModel (модель уявлення уявлення-модель)

SFC – Sequential Function Chart (послідовні функціональні схеми)

MERN – Mongo, Express, React, Node

MVC – Model-View-Controller (модель уявлення-контролер)

UI – User Interface (інтерфейс користувача)

RIA – Rich Internet Application (насичений інтернет-додаток)

XML – Extensible Markup Language (розширювана мова розмітки)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту)

URL – Uniform Resource Locator (уніфікований покажчик ресурсу)

SFC – Single File Components (однофайлові компоненти)

PWA – Progressive Web Applications (прогресивні вебзастосунки)

ВСТУП

Побудова вебсайту розділяється за трьома основними напрямками: серверна частина, зовнішній інтерфейс та база даних.

Зовнішній інтерфейс – це набір засобів, за допомогою яких користувач взаємодіє з вебсайтом або будь-якою іншою програмою через браузер. Зовнішні інтерфейси набули широкого поширення завдяки зростанню популярності Всесвітньої павутини і, як наслідок, повсюдному поширенню веббраузерів.

Кількість сайтів перевищує мільярд вебсторінок і головним завданням зовнішнього інтерфейсу є швидкість загрузки сторінки та інтуїтивно зрозумілий інтерфейс. Задля цього треба ретельно підійти до питання вибору професійного застосунку для створення вебзастосунку, тобто фреймворку.

Актуальність дослідження полягає у щоденному розвитку вебтехнологій, цифровій трансформації сучасного світу та побудові якісного продукту, для мільйонів користувачів.

1 АНАЛІЗ ІСНУЮЧИХ ФРЕЙМВОРКІВ ДЛЯ РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКІВ

1.1 Роль фреймворків під час розроблення вебзастосунків

Швидкість, з якою поширюється інформація сьогодні, є вражаючою. Інтернет став каталізатором до поширення інформації. В Інтернеті знаходяться мільйони вебсайтів різного спрямування. Розвиток Інтернету нерозривно пов’язано з проєктуванням сайтів. Масова поява сайтів спровокувала проблему їх якості. Популярність створення вебресурсів сприяла розробці різних систем і програм, які спрощують процес написання сайту. Також вони допомагають підвищити ефективність роботи, а також дозволяють розробнику сфокусуватися над основною логікою програми.

Головна мета сайту – допомогти замовнику з найменшими витратами отримати найбільшу віддачу від бізнесу. Для досягнення цієї мети розробники спільно із замовником покращує бізнес-процеси у компанії клієнта. Тому лицева сторона сайту – не тільки привабливі картинки, а і гіпотеза про те, як має працювати вся система.

Використання фреймворків стає все більше популярним, адже розробка за допомогою фреймворку зменшує навантаження на процес створення вебзастосунків, це досягається тим, що фреймворк позбавляє від проблеми використання повторюваного коду. Застосування фреймворків робить процес написання програми більш легким і функціональним.

Без використання фреймворків стає набагато складніше створювати вебзастосунки, супроводжувати і модернізувати їх. Проаналізувавши інформацію з мережі Інтернет, можна побачити, що існують сотні фреймворків для створення вебзастосунків. Тому досить складно зробити вибір фреймворку, так як кожен з них має велику кількість привабливих функцій та доповнень.

Неправильний вибір фреймворку може стати основною причиною невдачі проєкту.

Найпопулярнішими застосунками для зовнішнього інтерфейсу є AngularJS від компанії Google, React.js від Facebook та Vue.js від колишніх розробників React.

Робота над зовнішнім інтерфейсом сайту починається вже в той момент, коли дизайнер тільки-но приступив до роботи над проєктом, і не закінчується до кінця життєвого циклу сайту. Дизайнер радиться з верстальником, які візуальні рішення прийняти, щоб реалізувати проєкт у рамках бюджету та часу, а головне, коректно втілити у життя за допомогою сучасних технологій верстки.

Після обговорення структури, дизайн можна спростити або навпаки ускладнити, якщо стандарти і технології це дозволяють. Спілкування команди дизайнерів із верстальниками не припиняється протягом усієї роботи над сайтом.

Коли дизайн закінчено, frontend-розробнику передаються готові макети, набір елементів та можливості користувача на майбутньому вебзастосунку. У них міститься інформація про зміст елементів, їх взаємодію один з одним, загальні принципи побудови системи тощо. Робота над лицевою стороною передбачає, що сайт розробляється програмістами на основі наданої дизайнерами документації. На практиці це означає, що від дизайнера не потрібно намалювати абсолютно всі макети – достатньо задати принципи їхньої побудови.

Проєктування зовнішнього інтерфейсу складається з таких кроків:

– проаналізувати макети та супровідну документацію;

– виділити загальні елементи (якщо дизайнер не зробив UI-застосунки, що трапляється тільки в мікропроєктах і дуже рідко);

– розділити макети на блоки (так простіше тримати всю систему у голові);

– виділити складні елементи. Найчастіше навколо них будується вся структура сайту та взаємодія інших елементів один з одним;

– продумати структуру папок та файлів: вони повинні мати осмислені назви та відображати структуру системи;

– визначити програмну взаємодію елементів один з одним;

– обміркувати систему назв html-класів. Наразі популярна модифікована БЕМ-методологія. Вона допомагає робити класи на сторінці унікальними і дозволяє одним коротким поглядом на назву елемента зрозуміти, що за елемент і яке його призначення. Завдяки особливостям БЕМ можна уникнути проблем із структурою складних проєктів. Для зручної роботи з DOM у скриптах, варто задуматися про резервування ключових слів: new, actvie, closed, action та використовувати у зв'язці з іншими класами. Це зробить код більш читаним і дозволить простіше маніпулювати елементами в скриптах;

– спроєктувати систему взаємодії JavaScript-класів. У Ruby on Rails за замовчуванням вбудований coffeescript, який допомагає організувати взаємодію та набагато спрощує синтаксис. Стандарт ECMAScript 6 частина нововведень бере з coffeescript. Це означає, що coffeescript впливає на подальший розвиток стандартів і на нього варто звернути увагу іншим розробникам;

– передбачити можливі зміни. Проєкт – живий організм, який постійно змінюється. Тому в програмному коді передбачаються можливості його швидкої та безболісної зміни або навіть видалення;

– виявити слабкі місця – замислитись, що може піти не так. Це позбавить проблем надалі, зробить систему більш стійкою до зовнішніх впливів.

Після того, як зрозуміло, що згорнути і запрограмувати, починаємо основну роботу.

Інформаційна технологія JavaScript є невід'ємною частиною сьогодення Інтернету, адже використовується на 95% усіх вебсайтів та визначає потребу сучасного життя [1–5]. Користувачі пишуть статті, керують своїм бюджетом, транслюють музику, дивляться фільми та миттєво спілкуються з іншими на великій відстані за допомогою текстового, аудіо- чи відеочату [6–11].

Комп’ютерна мережа дозволяє робити те, що раніше було можливим лише у програмах, встановлених на власних комп’ютерах. Інтерактивні вебсайти часто називають вебзастосунками [12, 13].

Поява сучасних фреймворків JavaScript значно полегшила створення динамічних інтерактивних програм [14–20]. Фреймворк – це бібліотека, яка пропонує альтернативи щодо створення програмного забезпечення. Зазначені способи забезпечують:

– передбачуваність та однорідність програми;

– передбачуваність, що дозволяє масштабувати програмне забезпечення до величезних розмірів і залишатися придатним для обслуговування;

– передбачуваність і ремонтопридатність, що є важливими для справності та довговічності програмного забезпечення.

Кожен фреймворк базується на компонентах і дозволяє швидко створювати функції інтерфейсу користувача, їх можна використовувати для створення зовнішніх застосунків, однак, вони мають різну структуру та архітектуру, тому має сенс порівняти їх і зрозуміти їхні відмінності.

1.2 Класифікація існуючих фреймворків для розроблення вебзастосунків

Існує три основні фреймворки для створення вебзастосунків, про які знає кожен frontend-розробник (рис. 1.1).

– React;

– Vue.js;

– Angular.

React – JavaScript-бібліотека з відкритим вихідним кодом для створення інтерфейсів користувача, вона дозволяє створювати компоненти інтерфейсу для мобільного і ПК-обладнання. Її рекомендують використовувати для розроблення SPA та корпоративних платформ.

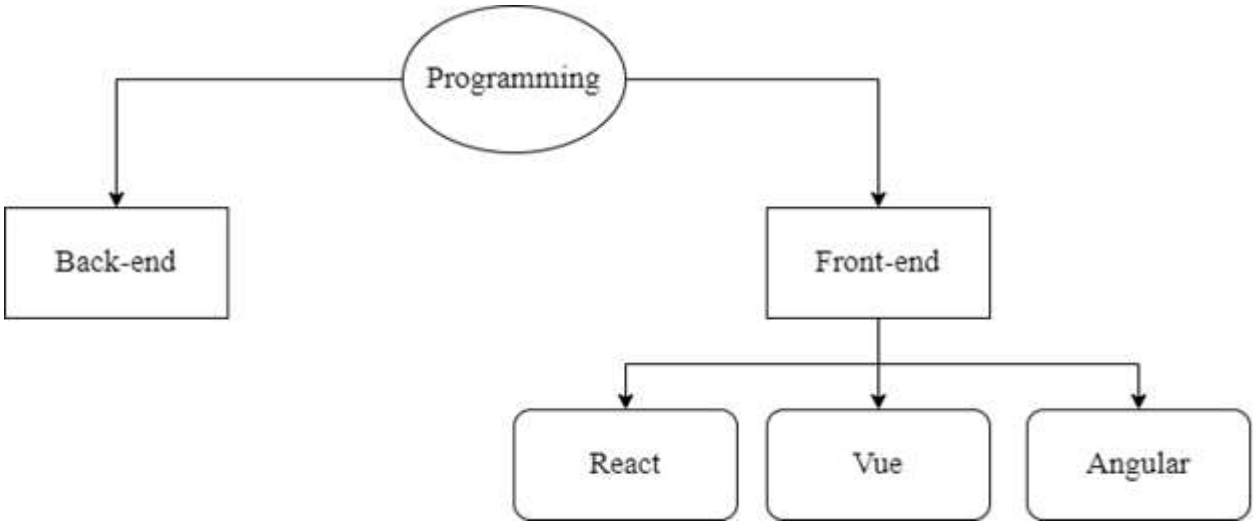

Рисунок 1.1 – UML-діаграма ієрархії фреймворків

GitHub React – це другий за популярністю фреймворк, його використовують Facebook, Instagram, WhatsApp.

Існує декілька переваг даного фреймворку:

– легке поєднання JavaScript та HTML;

– проста розробка динамічних вебзастосунків;

– просте налагодження;

– підтримка спільноти.

Vue.js – прогресивний фреймворк, який можна інтегрувати з уже готовими проєктами та бібліотеками JS. У 2020 році Vue став найкращим фреймворком на GitHub, обійшовши Angular та React. За останні роки інтерес до даного фреймворку зріс на 18%–20%, його використовують Stack Overflow, GitLab, Adobe.

Існує декілька переваг даного фреймворку:

– високий ступінь налаштовування;

– легкий у навчанні;

– підтримка CSS переходів та анімації;

– гнучкість та модульність.

Angular вебфреймворк дозволяє JavaScript інтегруватися з HTML та CSS.

На його основі збудовано понад 400 тисяч сайтів по всьому світу: вебзастосунки для ПК та мобільних пристроїв, а також такі, що підходить для корпоративного програмного забезпечення. Він використовується Google, Microsoft та YouTube.

Існує декілька переваг даного фреймворку:

– допомагає створювати прогресивні програми (PWA);

– зручно маніпулювати DOM-елементами;

– висока швидкість та продуктивність;

– зростаюча крива навчання;

– вбудований механізм застосування залежностей;

– підтримка Google;

– потужна екосистема.

1.3 Аналіз сучасних фреймворків для розроблення вебзастосунків

1.3.1 Архітектура сучасних фреймворків

React – це бібліотека інтерфейсу користувача, Angular – повноцінний інтерфейсний фреймворк, а Vue.js – прогресивний фреймворк. Кожен фреймворк базується на компонентах і дозволяє швидко створювати функціональні можливості інтерфейсу користувача. Однак, усі вони мають різну структуру та архітектуру, тому спочатку проаналізуємо їхні архітектурні відмінності, щоб зрозуміти філософію, що стоїть за ними.

React не забезпечує дотримання конкретної структури проєкту, можна почати використовувати React лише з кількох рядків коду.

React можна використовувати як бібліотеку інтерфейсу користувача для візуалізації елементів без застосування конкретної структури проєкту, тому це не суто фреймворк. Елементи React – це найменші будівельні блоки програм React. Вони потужніші за елементи DOM, оскільки React DOM ефективно оновлює їх щоразу, коли щось змінюється.

Компоненти – це більші будівельні блоки, які визначають незалежні та багаторазово використані частини. Вони беруть вхідні дані, які називаються props, і створюють елементи, які потім відображаються користувачеві.

React базується на JavaScript, але здебільшого поєднується з JSX (JavaScript XML), розширенням синтаксису, яке дозволяє створювати елементи, що містять HTML і JavaScript одночасно. Усе, що створено за допомогою JSX, також можна створити за допомогою React JavaScript API, але більшість розробників віддають перевагу JSX, оскільки він більш інтуїтивно зрозумілий.

Базова бібліотека **Vue.js** фокусується лише на шарі View. Його називають прогресивним фреймворком, оскільки можна розширити його функціональність за допомогою офіційних і сторонніх пакетів, таких як Vue Router або Vuex, і перетворити його на справжній фреймворк. Хоча Vue не пов’язаний строго з шаблоном Model-View-ViewModel (MVVM), його дизайн частково є ним. За допомогою Vue програмування відбувається здебільшого на рівні ViewModel, щоб переконатися, що дані програми обробляються таким чином, щоб фреймворк відтворював фактичне подання.

Синтаксис шаблону Vue дозволяє створювати компоненти View і поєднує знайомий HTML зі спеціальними командами та функціями. Цей синтаксис шаблонів є кращим, хоча також підтримуються необроблені JavaScript і JSX.

Компоненти у Vue невеликі, автономні та придатні для повторного використання в усій програмі. Однофайлові компоненти (SFC) із розширенням .vue містять HTML, CSS і JavaScript, тому весь відповідний код міститься в одному файлі. SFC – це рекомендований спосіб організації коду в проєктах Vue.js, особливо у великих.

AngularJS, оригінальний фреймворк, є фреймворком MVC (Model-View-Controller). Але в Angular 2 немає суворого зв’язку з шаблонами MV*, оскільки він також базується на компонентах. Проєкти в Angular структуровані за модулями, компонентами та службами. Кожен застосунок Angular має принаймні один кореневий компонент і один кореневий модуль.

Кожен компонент в Angular містить шаблон, клас, який визначає логіку програми, і метадані (декоратори).

Метадані для компонента повідомляють Angular, де знайти будівельні блоки, необхідні для створення та представлення свого перегляду.

Шаблони Angular написані в HTML, але можуть включати, серед іншого, синтаксис шаблону Angular зі спеціальними командами для виведення реактивних даних і візуалізації кількох елементів. Сервіси в Angular використовують компоненти для делегування завдань бізнес-логіки, таких як пошук даних або перевірка введених даних. Вони є важливою частиною програм Angular. Хоча Angular не вимагає їх використання, рекомендується структурувати програми, як набір окремих служб, що можна повторно використовувати.

Angular побудовано на TypeScript, тому його використання рекомендовано для максимально зручної роботи, але звичайний JavaScript також підтримується.

1.3.2 Екосистема сучасних фреймворків

Багато зовнішніх застосунків покладаються на глобальне керування станом для зберігання інформації, наприклад, про те, хто увійшов у систему та інші налаштування користувача.

Найпопулярнішим проєктом керування станом JavaScript є Redux.

Більшість розробників використовують офіційні прив’язки React для Redux, які підтримує команда Redux. Завдяки популярності React, знайти вхідні компоненти та готові до використання елементи надзвичайно легко. Усі вони доступні лише за допомогою пошуку в Google або GitHub.

Екосистема React також включає React Native, який дозволяє створювати нативні програми для iOS та Android з єдиної кодової бази, написаної в React. Тому React також може бути чудовим вибором для створення мобільних застосунків із використанням вебтехнологій.

React є частиною стеку MERN, який включає MongoDB, ExpressJS, React і NodeJS. Чудова особливість цієї комбінації полягає в тому, що вся програма працює на одній мові – JavaScript.

Хоча Redux можна використовувати у Vue, офіційних прив'язок немає. Vuex є офіційною бібліотекою керування станом, створеною спеціально для програм Vue. Застосунок налагоджують за допомогою інструментів розробника Vue. На початку Vue було важче знайти готові до використання компоненти. Оскільки спільнота зросла, існує широкий вибір компонентів введення та розширених елементів, які можна використовувати для прискорення розвитку.

Для розробки мобільних застосунків є проєкт під назвою Weex. Weex розроблений і використовується Alibaba, але він не такий зрілий і потужний, як React Native. Більше того, оскільки проєкт розробляється та використовується у Китаї, важче знайти документацію та усунути несправності.

Vue добре інтегрується з Laravel, тому їх часто використовують разом. Laravel пропонує повний каркас JavaScript і CSS, який дозволяє використовувати Vue у нових проєктах.

Щодо екосистеми Angular, то можна використовувати проєкт NgRx для керування станом. Він був прототипом Redux, але створений спеціально для Angular. Як і у Vue та React, є багато готових до використання компонентів, які можна імпортувати у свої проєкти. У проєкті Angular Material є багато офіційних компонентів, наприклад, проєкт Google, який пропонує компоненти Material Design для програм Angular.

Можна створювати кросплатформні мобільні програми в Angular за допомогою NativeScript. Він також підтримує Vue, але підтримка Angular є більш зрілою. Angular є частиною відомого стеку MEAN, який поєднує Angular з MongoDB, ExpressJS і NodeJS. Як і стек MERN, він повністю покладається на JavaScript як для інтерфейсу, так і для серверу.

Вибираючи фреймворк або бібліотеку, також треба думати про продуктивність. У багатьох випадках не доведеться турбуватися про продуктивність, особливо, якщо у розробці знаходиться невеликий проєкт.

Однак, у міру збільшення масштабів і складності проєкту продуктивність може стати проблемою. Важливо зауважити, що якість розробки та дотримання найкращих практик є важливішими, ніж вибір фреймворку.

1.4 Аналіз літературних джерел щодо апробації результатів застосування фреймворків під час розроблення вебзастосунків

Стаття [21] представляє реальний приклад, де компанія humansoft переробила одне зі своїх основних рішень, використовуючи одну з найновіших і використовуваних вебтехнологій React.js. React.js – це фреймворк, спеціально розроблений для роботи з рівнем візуалізації вебзастосунків та їх використання в бізнес-контексті, а саме в сфері управління людськими ресурсами, на яку спрямовано рішення. Завдяки цьому процесу було досягнуто справжнього рівня задоволеності клієнтів, оскільки вони підкреслили більшу гнучкість і простоту використання продукту, що дозволило оптимізувати та збільшити можливості процедур управління людськими ресурсами.

Стаття [22] описує геніальну ідею React, який робить користувальницький інтерфейс надійним і знімає величезну вагу з програмістів, щоб вони могли зосередитися на більших обмеженнях і бізнес-міркуваннях. React.js дає змогу розробникам програмного забезпечення створювати величезну вебпрограму, яка може використовувати дані та змінюватися через деякий час без перезавантаження сторінки.

У документі [23] описано деякі практичні способи подолання проблем з фреймворком React.js, які часто виникають, зокрема повторне відображення компонентів, затримка програми через виконання фонових обчислень, затримка через обробку великих наборів даних за один відрізок тощо. Також у статті описаний ефективний за часом алгоритм пошуку, який можна використовувати для пошуку об’єктів у великому наборі даних.

У документі [24] описана реалізація фреймворку React.js з бібліотекою Redux, сервісом Firebase, безсерверний сервіс, який пропонує багато функцій для веброзробки в зручний спосіб, та фреймворком Bootstrap 4 для пришвидшення верстки сторінок.

У статті [25] розроблено систему навчання в коледжі на основі різноманітних інформаційних Інтернет-технологій. Зокрема, інтерфейс розроблено прогресивною платформою Vue.js; загальну структуру системи створено на основі архітектури браузер-сервер (B/S); функції системи реалізовано за допомогою HTML, Node.js і технології баз даних; сумісність між мобільним терміналом і настільним комп'ютером реалізована за допомогою Bootstrap. Тестування системи показало, що система навчання коледжу на основі Vue.js працювала стабільно, досягла поставлених цілей і задовольняла попит користувачів.

У статті [26] показано, чому фреймворки потрібні для розробки та як компанії можуть використовувати їх у своїх інтересах. Також розглянуті базові концепції фреймворку, що надає можливість вивчати в школах для легшого розуміння концепцій розвитку.

Документ [27] складається з досліджень, які доводять, як застосування одного з найпопулярніших фреймворків JavaScript, Vue.js, може створити позитивний ефект для процесу розробки на основі певних особливостей фреймворку. Крім того, документ виступає як інструкція зі створення загального проєкту Vue.js, а також надає авторські стандарти кодування та найкращі практики, які дуже корисні для нових розробників.

У статті [28] порівнюється продуктивность популярних фреймворків JavaScript Angular і Vue.js у контексті розробки ігор. Критеріями порівняння наступні: час обміну даними з сервером і відтворення різних компонентів програми, споживання пам'яті під час оновлення поточної інформації про гру та повернення користувача до поточної гри, рівень завантаження браузера та розмір кінцевих файлів програми. Результати тестування показують, що фреймворк Vue.js ефективніший.

У документі [29] наводяться результати опитування про Angular.js, включаючи відповіді 460 розробників: визначення найбільш цінних функцій Angular.js (користувальницькі компоненти інтерфейсу, впровадження залежностей і двостороннє зв’язування даних) і найбільш проблематичні аспекти фреймворку (наприклад, продуктивність і реалізація директив).

У документі [30] представлено та оцінено дві популярні системи тестування вебавтоматизації: Protractor і Karma. Як додаток до опису тестових фреймворків у цьому документі представлені результати виконання тестів на оптимізованому та неоптимізованому вебзастосунку, що використовується в реальних системах.

У статті [31] порівнюється чотири фреймворки JavaScript: AngularJS, Ember, Knockout, Backbone. Усі чотири фреймворки базуються на MVC або подібній архітектурі. Крім того, у документі представлені переваги та недоліки кожного фреймворку, вплив на швидкість застосунків, способи тестування таких застосунків та способи покращення безпеки коду.

У документі [32] проводиться ретельна оцінка якості та продуктивності найпопулярніших фреймворків JavaScript, беручи до уваги добре встановлені фактори якості програмного забезпечення та тести продуктивності. Основним результатом є висвітлювання плюсів та мінусів фреймворків JavaScript у різних сферах інтересу та освітлювання проблемних моментів коду, які, ймовірно, потребують покращення в наступних версіях.

1.5 Постановка задачі дослідження

Актуальність даного дослідження полягає у важливості вибору фреймворку для розроблення вебзастосунку з метою підвищення продуктивності і швидкості продукту та відповідність до очікувань користувачів.

Об’єктом дослідження є процес тестування вебзастосунків на трьох різних фреймворках, а саме React, Vue та Angular, що застосовуються на базі мов програмування JavaScript і TypeScript.

Метою дослідження є порівняння вибраних фреймворків задля визначення їх популярності у світі, ефективності та масштабності під час тестування однакових класів задач.

Враховуючи мету роботи, необхідно вирішити такі завдання:

– проаналізувати сучасні тенденції розробки вебзастосунків;

– виявити особливості фреймворків, що розглядаються;

– проаналізувати тенденції у виборі фреймворку;

– змоделювати вебзастосунок;

– програмно реалізувати вебзастосунок для трьох фреймворків;

– здійснити порівняльний аналіз продуктивності та швидкості реалізованих вебсторінок;

– визначити перспективи подальшої роботи.

2 ДОСЛІДЖЕННЯ ВИБРАНИХ ФРЕЙМВОРКІВ ДЛЯ РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКІВ

2.1 Фреймворк React

React – популярна бібліотека мови програмування JavaScript, яка використовується для створення вебінтерфейсів користувача. Це значно полегшує створення інтерфейсів, розділяючи кожну сторінку на невеликі фрагменти, які називаються компонентами.

Приклад поділу сторінки на компоненти (рис. 2.1) [33].

Рисунок 2.1 – Поділ головної сторінки Google на компоненти

Ключовою особливістю React є склад компонентів. Компоненти, написані різними людьми, повинні добре працювати разом. Важливо, щоб розробник міг додати функціональність до компонента, не викликаючи змін у всьому коді.

Наприклад, повинна бути можливість ввести деякий внутрішній стан у компонент без зміни компонентів, які від нього залежать. Повинна бути можливість додавати код ініціалізації та знищення до будь-якого компонента за потреби. Все це реалізовано в React.

Компоненти часто називають просто функціями. Компоненти в React описують поведінку, яку можна створити. Ця ідея стосується візуалізації, життєвого циклу та стану. Деякі сторонні бібліотеки, такі як Relay, додають додаткові функції до компонентів, наприклад, описують залежності даних.

React прагматичний. Це пов’язано з потребами продуктів, про які йдеться у Facebook. Незважаючи на вплив деяких менш популярних парадигм, таких як функціональне програмування, проєкт прагне залишатися доступним для широкого кола розробників з різним досвідом і рівнем кваліфікації.

У React потік даних є односпрямованим. Це означає, що дані течуть, як водоспад, зверху вниз, від батьківського до дочірнього через незмінні об’єкти, доступні лише для читання, які називаються props.

Props – це лише аргумент функції, з яким можна внутрішньо працювати, але не змінювати. Для передачі даних між сусідніми компонентами необхідно використовувати посередника – їхнього батька. Спочатку потрібно передати дані від дочірнього до батьківського, як аргумент зворотного виклику. Потім призначити ці дані батьківському стану та передати їх через властивості іншому компоненту. Механізм локального зберігання компонентів, який постачається з базовою бібліотекою (React), незручний, оскільки таке сховище є ізольованим.

Наприклад, якщо треба, щоб різні незалежні компоненти реагували на певну подію, потрібно або передати локальний стан, як атрибут дочірнім компонентам, або підвищити його до найближчого батьківського компонента.

У будь-якому випадку це незручно. Код стає складнішим, його важче читати, а компоненти стають залежними від їх вкладеності. Redux усуває цю проблему, оскільки всі стани доступні для всього компонента без особливих проблем. Redux – це універсальний інструмент розробки, який можна використовувати в поєднанні з різними бібліотеками та фреймворками.

Віртуальний DOM (VDOM) – це концепція програмування, в якій ідеальне або «віртуальне» представлення інтерфейсу користувача зберігається в пам’яті та синхронізується з «реальним» DOM за допомогою бібліотеки, такої як ReactDOM. Цей процес називається звіркою.

Це те, що робить React API декларативним – треба вказати, у якому стані має бути UI, і React переконається, що DOM відповідає цьому стану. Абстрагує маніпуляції атрибутами, обробку подій і ручне оновлення DOM, які в іншому випадку довелося б використовувати в розробці застосунків.

Оскільки «віртуальний DOM» – це шаблон, а не конкретна технологія, цей термін іноді використовується для позначення різних речей. У світі React «віртуальний DOM», зазвичай, асоціюється з елементами React, оскільки вони є об'єктами, які представляють інтерфейс користувача. Однак, React також використовує внутрішні об'єкти, які називаються потоками, для зберігання додаткової інформації про дерево компонентів. Їх також можна вважати частиною реалізації «віртуального DOM» React (рис. 2.2) [34].

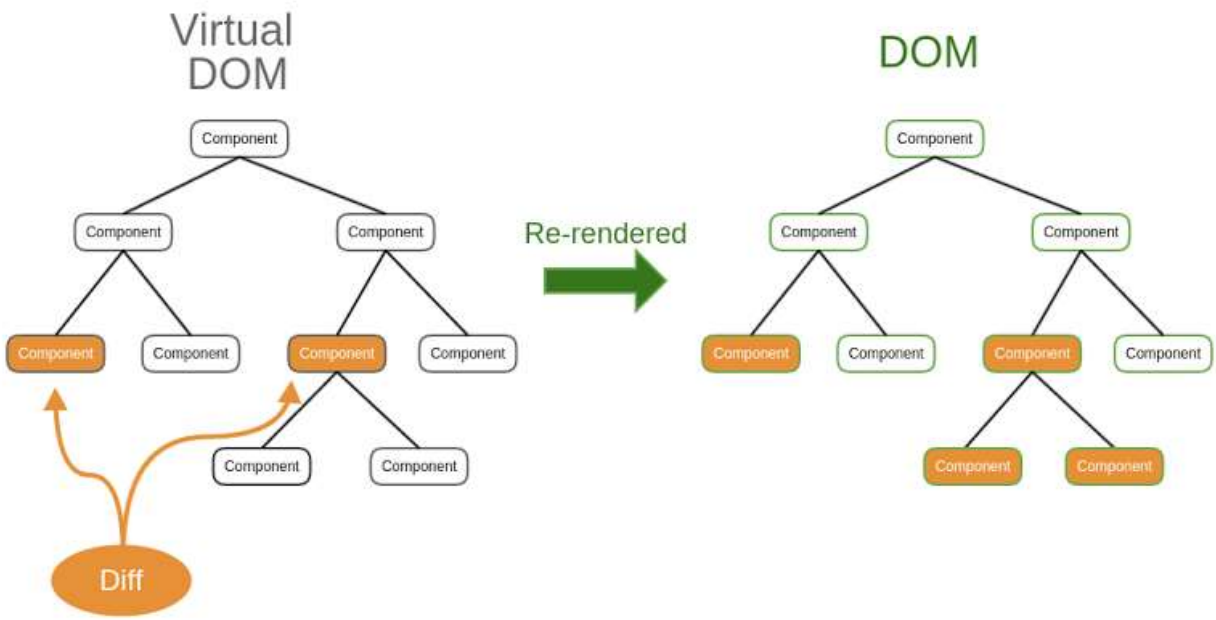

Рисунок 2.2 – Візуалізація роботи Virtual DOM

JSX – це абстракція, яка дозволяє використовувати синтаксис HTML у коді JavaScript і дозволяє створювати компоненти React, які виглядають як стандартні теги HTML.

JSX – це мова шаблонів для елементів React, тому вона служить основою для будь-якої розмітки, яку React відображає у програмі.

Оскільки JSX дозволяє використовувати JavaScript у своїх тегах, можна скористатися перевагами функцій і методів JavaScript, включаючи зіставлення масивів і скорочену обчислення умовних конструкцій.

Під час роботи компонент проходить ряд фаз життєвого циклу. У кожній фазі викликається певна функція, у якій можна визначити довільні дії:

– constructor(properties): конструктор, у якому відбувається ініціалізація компонента;

– static getDerivedStateFromProps (props, state): викликається безпосередньо перед відтворенням компонента. Цей метод не має доступу до поточного об’єкта-компонента (тобто доступ до об’єкта-компонента через this) і повинен повернути об’єкт, щоб оновити об’єкт стану, або null, якщо немає чого оновлювати;

– render(): візуалізація компонента;

– componentDidMount(): викликається після відтворення компонента;

– componentWillUnmount(): викликається перед видаленням з DOM.

Бібліотека React.js доступна в компактній версії Minimal React. Налаштування цього маленького пакету не потребує ні часу, ні зусиль. Він оснащений зручною опцією поділу коду, яка допомагає скоротити час завантаження вебсторінки, оскільки ця функція запобігає одночасному відображенню всіх компонентів.

React.js містить низку компонентів, інтеграцій з іншими платформами, що дозволяє розробникам створювати складні вебзастосунки для будь-яких цілей. Це пояснює величезну популярність бібліотеки React.js.

Найбільшою перевагою використання React.js є те, що нові версії цієї бібліотеки повністю сумісні з попередніми випусками. Старі бібліотеки не стануть марними після випуску оновлень.

Розробкою динамічних вебзастосунків займаються багато компаній. Пакет React.js знайде застосування в будь-якій з них, оскільки значно скорочує термін роботи над проєктами, покращує функціональність веброзробок та знижує обсяги написання коду вручну.

Віртуальний DOM робить розробку простіше, швидше, зрозуміліше ніж будь-яка інша технологія. Крім того, використання цієї технології допомагає уникнути повного перезавантаження даних із сервера при внесенні змін до окремого компонента.

Концепція компонентів, що повторно використовуються в React.js, передбачає багаторазове використання одних і тих же компонентів у різних розділах програми, що значно скорочує терміни роботи над проєктом.

Вивчення React.js не має труднощів, оскільки це середовище розробки використовує HTML, JavaScript та зручні доповнення. Вбудовані інструменти та фреймворки допомагають розробнику зрозуміти логіку React.js.

Пакет React Native можна встановити для створення мобільних програм, проте ця бібліотека може бути частиною розробки сайту. React Native дозволяє керувати мінімумом програмного коду. React Native дозволяє обходитися мінімум програмного коду.

На основі бібліотеки React.js можна створювати найрізноманітніші типи мобільних застосунків і сторінок. Ось декілька цікавих ідей:

– соціальні мережі (React.js підходить для створення соціальних платформ: Twitter, Facebook, Instagram використовують багато функцій цієї бібліотеки. React.js полегшує розробку функцій для розміщення текстового та медіаконтенту. Створення новинних каналів і трансляцій, впровадження системи коментарів, лайків і аутентифікації користувачів);

– електронний магазин (гігант онлайн-роздрібної торгівлі Amazon використовує React.js. Бібліотека дозволяє розробляти надійні та багатофункціональні системи управління продажами електронної комерції, які підходять для сайтів з високим відвідуванням. Функції React.js використовуються в популярному мобільному застосунку для доставки їжі UberEats);

– розваги (розважальні платформи, які використовують React.js, включають Netflix, YouTube, TikTok і Spotify. Функціонал цієї бібліотеки дозволяє створювати інтерактивні розважальні вебсервіси та мобільні застосунки з можливостями соціальних мереж – коментарі, лайки, підписки, рейтинги);

– месенджери (окрім Whatsapp, функції бібліотеки React.js також використовуються у Viber та Facebook Messenger. На основі бібліотеки React Native можна створити систему підтримки користувачів у реальному часі – на сайті компанії чи в мобільному застосунку);

– планувальники завдань (бібліотека React.js чудово підходить для розробки застосунків, пов'язаних з управлінням проєктами та особистим часом, веденням нотаток, розподілом завдань і співпрацею. Todoist, Notion, Things, серед інших, засновані на React.js).

React.js гнучка, універсальна технологія, яка б ідея не була, кваліфікований розробник зможе її реалізувати. Створення вебсервісу та застосунку на основі React.js сприятиме зростанню компанії, оскільки дозволить бізнесу використовувати всі останні тренди.

2.2 Фреймворк Vue

Vue – це прогресивний фреймворк для побудови інтерфейсів користувача. Він розроблений з нуля для поступового впровадження та може легко масштабуватися між бібліотекою та фреймворком залежно від різних випадків використання. Він складається з доступної основної бібліотеки, яка зосереджена лише на рівні перегляду, та екосистеми допоміжних бібліотек, які допомагають справлятися зі складністю великих односторінкових програм [35].

Vue чудово підходить для початківців розробників. Vue є одним із найпопулярніших фреймворків JavaScript, призначених для реалізації користувацьких інтерфейсів у застосунках і на вебзастосунків. Він легко інтегрується в будь-яку веброзробку як основний інтерфейсний інструмент, що є головною відмінною рисою цього фреймворку. Vue відкриває широкі можливості у світі вебдизайну для початківців програмістів, оскільки має досить широкий набір функцій. У той же час цей фреймворк не надто складний для вивчення в порівнянні з його «конкурентами»: React або Angular.

Vue позиціонує себе готовим «фреймворком» для розробки інтерфейсів на мові програмування JavaScript. Всередині нього вже зібрані всі необхідні бібліотеки, які полегшують і прискорюють розробку. Головні особливості фреймворку:

– фреймворк Vue базується на шаблоні MVVM, який є похідним від класичного шаблону Model-View-Controller. Його поява сприяла розділенню frontend розробки графічного інтерфейсу користувача та внутрішньої бізнес-логіки, що істотно підвищує ефективність створення вебзастосунків. В основі MVVM лежить рівень ViewModel, дещо схожий на конвертер значень, відповідальний за зміну об'єктів даних у моделі;

– легкий у використанні та нескладно інтегрований. Це один із найбільш зручних для початківців інструментів із низьким часом навчання порівняно з іншими фреймворками. Він також буде інтуїтивно зрозумілим для розробників, які раніше працювали з jQuery та Angular. Його також легко інтегрувати з іншими бібліотеками або використовувати як окремий проєкт;

– Vue.js невеликий і швидкий з точки зору продуктивності, що робить його одним із основних інструментів для розробки мобільних застосунків;

– реактивне зв'язування даних. Це надає перевагу кожного разу, коли дані змінюються. При зміні будь-якого компоненту, елементу чи змінної – DOM автоматично оновлюється, тобто сторінка редагується автоматично;

– віртуальний DOM. По суті, він служить інтерфейсом між сценаріями та тегами, полегшуючи надсилання та отримання даних із HTML. При зміні елементів CSS в деяких випадках необхідно перебудувати все дерево. Vue.js створює копію у віртуальній DOM, а не в DOM браузера. Усуваючи необхідність дублювати інформацію, загальна продуктивність підвищиться.

Vue – це фреймворк, який дозволяє створювати вебзастосунки на основі компонентної архітектури. Це означає, що при створені програми з автономних компонентів їх можна багаторазово використовувати. В ідеалі жоден із компонентів не повинен залежати один від одного. Це чудовий спосіб створювати великі програми, які зручно підтримувати. У Vue кожна частина програми є компонентом (рис. 2.3).

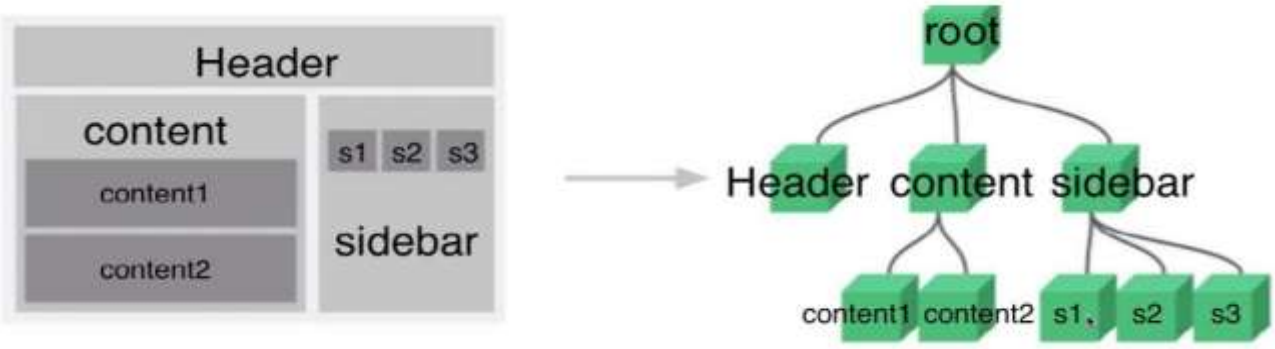

Рисунок 2.3 – Концепція компонентів у фреймворку Vue

На рисунку 2.3 зліва знаходиться типовий вебсайт, поділений на розділи. Кожен розділ є окремим компонентом. Головним контейнером програми є кореневий компонент (root), який огортає всі інші компоненти. Кореневий компонент має три дочірніх елементи: заголовок (верхній), основний вміст (ліворуч) і бічну панель (праворуч), які в свою чергу мають свої дочірні елементи.

Праворуч відображена модель дерева компонентів, яке представляє зв’язки «батьки-нащадки» компонентів усередині програми. Кожен додаток Vue можна представити у вигляді такого дерева, що спрощує розуміння побудови вебзастосунків.

Зазвичай, незалежно від того, який фреймворк використовується, кожен компонент має свою розмітку (HTML), стилі (CSS) та функціонал (JavaScript). У таких фреймворках, як Angular або React, кожна або деякі з цих частин зазвичай знаходяться в окремих файлах. У Vue є дуже чітка концепція так званих однофайлових компонентів (Single File Components). Ця особливість дозволяє помістити весь код, який пов’язаний з певним компонентом, в один файл із розширенням .vue.

Усі компоненти формують односторонню зв’язку між дочірньою та батьківською властивостями (props): коли батьківська властивість оновлюється, вона переходить до дочірньої, але не навпаки. Це запобігає дочірнім компонентам від випадкової зміни стану батьківського, що може ускладнити розуміння потоку даних програми.

Vue – має такі частини, що відповідають за оновлення стану:

– state (стан) – джерело правди, що керує вебзастосунком;

– view (вид) – декларативне відображення стану;

– actions (дії) – можливі способи зміни стану у відповідь на введення користувача з view.

Представлення концепції одностороннього потоку даних у фреймворку Vue (рис. 2.4) [36].

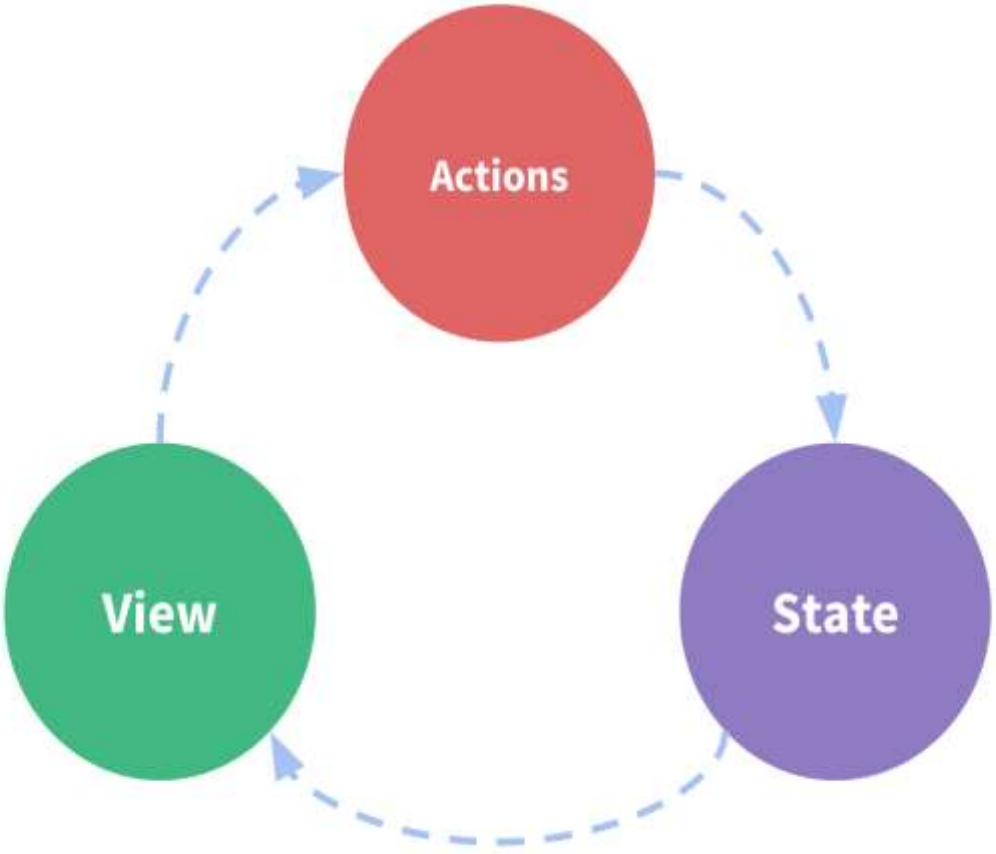

Рисунок 2.4 – Контроль над станом у Vue

Складність великих застосунків часто зростає через розподіл частин стану між багатьма компонентами та зв’язками між ними. Щоб вирішити цю проблему, Vue пропонує Vuex: власну бібліотеку управління станом (рис. 2.5). Vuex навіть інтегрується з інструментами розробника, надаючи доступ до функціональності програми з веббраузеру [37].

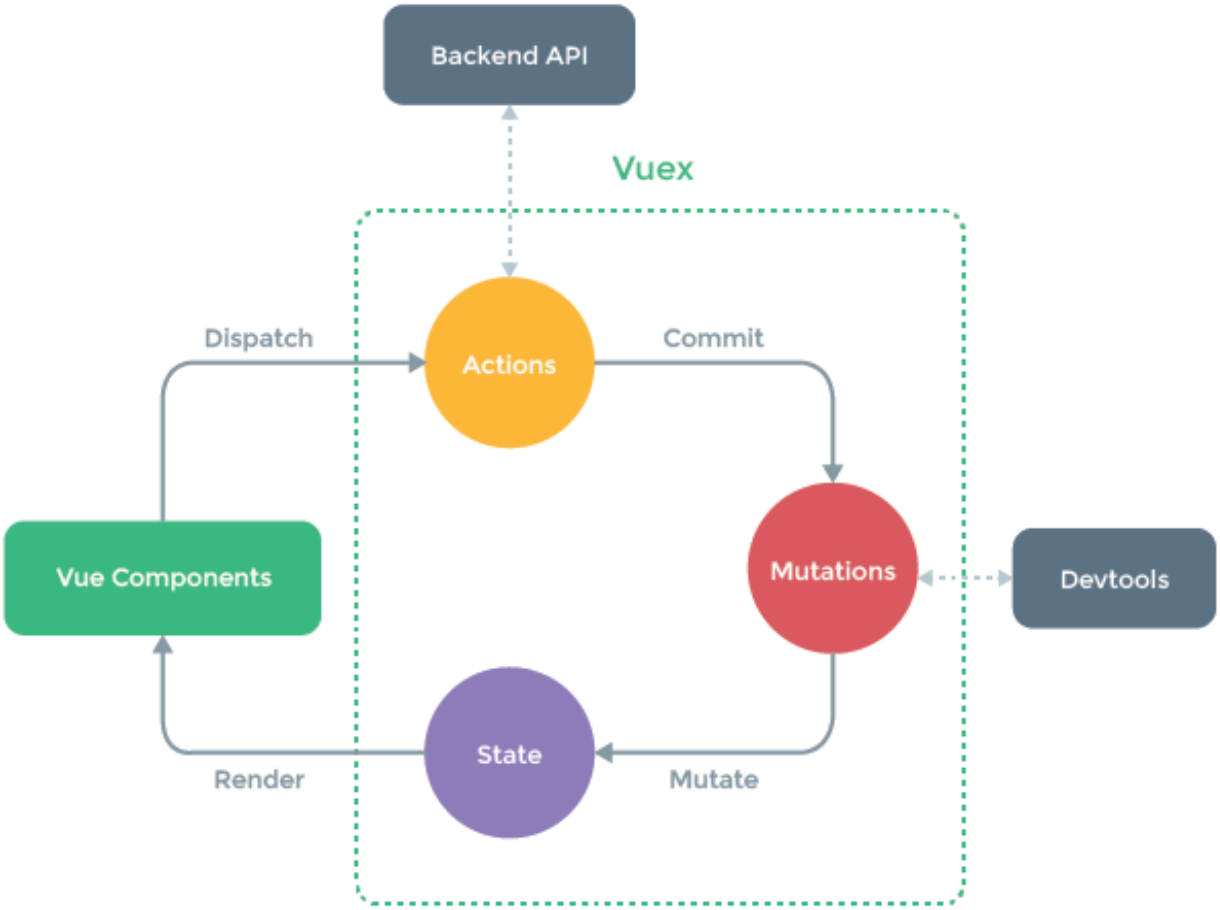

Рисунок 2.5 – Контроль над станом у Vue за допомогою Vuex бібліотеки

2.3 Фреймворк Angular

AngularJS – це фреймворк для динамічних вебзастосунків. Це дозволяє використовувати HTML як мову шаблонів і розширювати синтаксис HTML, щоб чітко та лаконічно виражати компоненти програми.

Зв'язування даних Angular і впровадження залежностей усуває більшість коду, який тепер потрібно писати. І все це відбувається в браузері, що робить його ідеальним партнером для будь-якої серверної технології.

Головні переваги фреймворку AngularJS:

– AngularJS – це потужне середовище розробки на основі JavaScript для створення Rich Internet Application (RIA);

– AngularJS дає розробникам можливість писати клієнтську програму (за допомогою JavaScript) у чистому вигляді MVC;

– програма, написана на AngularJS, є кросбраузерною. AngularJS автоматично аналізує код JavaScript, який підходить для кожного браузера;

– AngularJS є відкритим кодом, повністю безкоштовним і використовується тисячами розробників у всьому світі. Він ліцензований за ліцензією Apache версії 2.0.

Загалом, AngularJS – це платформа для створення великомасштабних, високопродуктивних, простих у обслуговуванні вебзастосунків.

Нижче наведено найважливіші ключові функції AngularJS:

– зв’язування даних – це автоматична синхронізація даних між компонентами моделі та представлення;

– область дії – це об’єкти, що належать моделі. Вони діють, як посередник між контролером і представленням;

– контролер – це функції JavaScript, що прив’язані до певної області;

– служби – AngularJS поставляється з кількома вбудованими службами, такими як $https: для створення запитів XML/HTTP. Це окремі об’єкти, які створюються лише один раз у програмі;

– фільтри – вони вибирають підмножину елементів із масиву та повертають новий масив;

– директиви – це теги для елементів DOM (таких як елементи, атрибути, CSS тощо). Їх можна використовувати для створення власних тегів HTML, які слугуватимуть новими користувацькими контент-модулями. AngularJS має вбудовані директиви (ngBind, ngModel тощо);

– шаблони представляють собою відтворені види з інформацією про драйвер і модель. Це може бути один файл (наприклад, index.html) або кілька переглядів на одній сторінці за допомогою «розділу»;

– маршрутизація – це концепція перемикання переглядів;

– MVC – це шаблон проєктування для поділу програми на різні частини (які називаються Model, View і Controller), кожна зі своїми обов’язками. AngularJS не реалізує MVC у традиційному розумінні, а радше щось ближче до MVVM;

– прямі посилання – прямі посилання дозволяють закодувати стан програми в URL-адресу, щоб її можна було зберегти як закладку. Потім програму можна відновити з URL-адреси до того самого стану;

– ін’єкція залежностей – AngularJS має вбудовану підсистему ін’єкції залежностей, яка допомагає розробникам, спрощуючи розробку, розуміння та тестування програми.

Наступна діаграма зображує важливі частини AngularJS (рис. 2.6) [38].

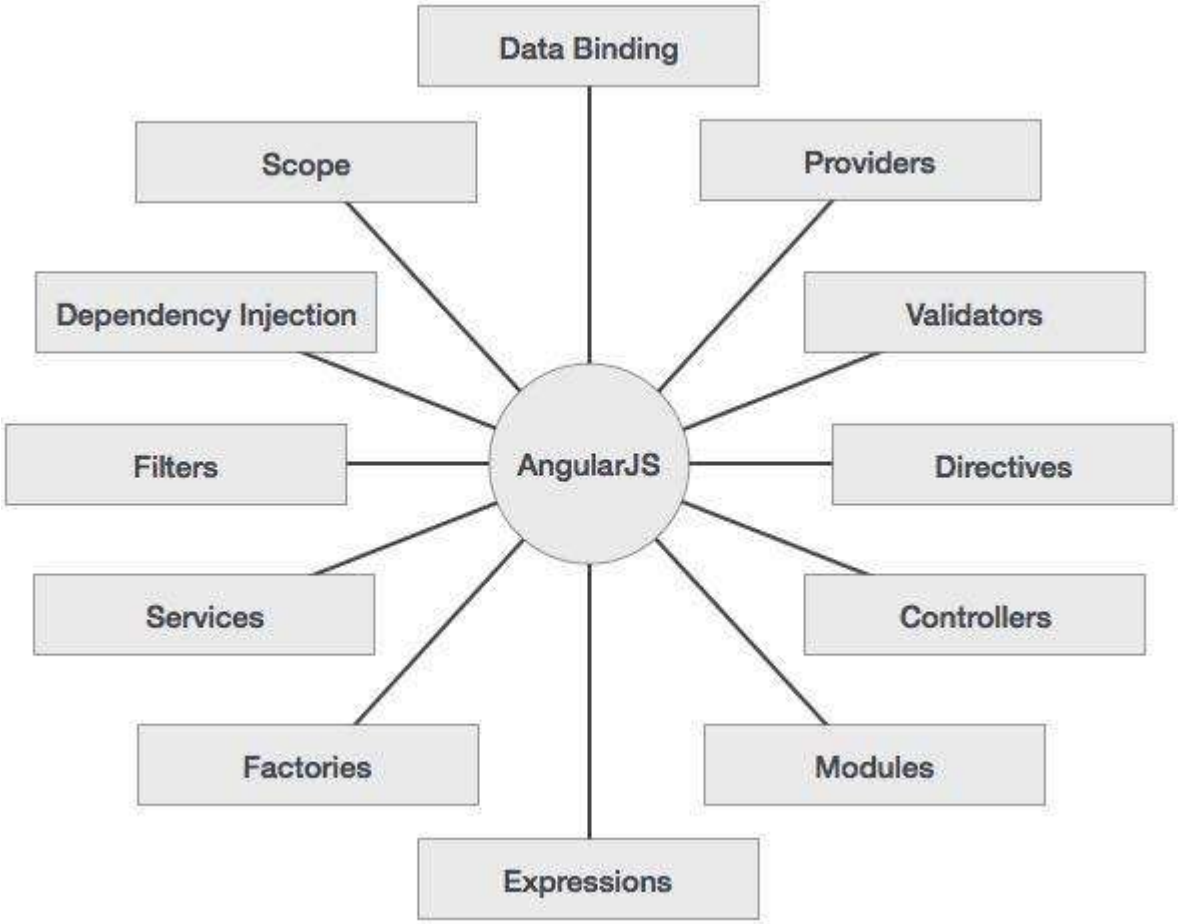

Рисунок 2.6 – Основні концепції AngularJS

Компоненти – це будівельні блоки, з яких складається програма (рис. 2.7). Компонент містить клас TypeScript із декоратором @Component(), шаблон HTML і стилі.

Декоратор @Component() визначає таку інформацію, специфічну для Angular:

– селектор CSS, який визначає, як компонент використовується в шаблоні. Елементи HTML у вашому шаблоні, які відповідають цьому селектору, стають екземплярами компонента;

– HTML-шаблон, який інструктує Angular, як відобразити компонент;

– додатковий набір стилів CSS, які визначають зовнішній вигляд елементів шаблону HTML Нижче наведено мінімальний компонент Angular.

```
import { Component } from '@angular/core';

@Component({
  // tslint:disable-next-line:component-selector
  selector: 'hello-world',
  template: `
    <h2>Hello World</h2>
    <p>This is my first component!</p>
  `
})
export class HelloWorldComponent {
  // The code in this class drives the component's behavior.
}
```

Рисунок 2.7 – Мінімальний компонент Angular

Для використання цього компонента в інших компонентах програми, треба написати наступне (рис. 2.8).

```
<hello-world></hello-world>
```

Рисунок 2.8 – Метод використання компоненту

Компонентна модель Angular пропонує потужну інкапсуляцію та інтуїтивно зрозумілу структуру програми.

Компоненти також роблять програму легшою для модульного тестування та можуть покращити загальну читабельність коду.

Angular CLI – це найшвидший, простий і рекомендований спосіб розробки вебзастосунків Angular. Angular CLI дозволяє спростити написання коду та робить деякі завдання безпроблемними, наприклад:

– команда ng build компілює вебзастосунок Angular у вихідний каталог;

– команда ng serve створює та обслуговує програму, для зміни у файлі;

– команда ng generate створює або змінює файли на основі схеми;

– команда ng test виконує модульні тести на заданому проєкті;

– команда ng e2e створює та обслуговує вебзастосунок Angular, а потім запускає наскрізні тести.

Численні переваги Angular дійсно стають очевидними, коли програма збільшується, і треба додати такі функції, як навігація сайтом або введення для користувачів. За допомогою фреймворку Angular, можна включити одну з багатьох власних бібліотек, які надає Angular. Деякі з доступних бібліотек:

– Angular Router – це розширена клієнтська навігація та маршрутизація на основі компонентів Angular. Підтримує відкладене завантаження, вкладені маршрути, спеціальне зіставлення шляху тощо;

– Angular Forms – це уніфікована система створення та перевірки форм;

– Angular HttpClient – це надійний HTTP-клієнт, який може забезпечити більш розширений зв’язок клієнт-сервер;

– Angular Animations – потужна система керування анімаціями на основі стану програми;

– Angular PWA – інструменти створення прогресивних вебзастосунків.

Ці бібліотеки розширюють можливості програми, дозволяючи більше зосередитися на функціях, які роблять програму унікальною.

Бібліотеки створені для бездоганної інтеграції та оновлення одночасно з фреймворком Angular. Ці бібліотеки потрібні лише тоді, коли вони можуть допомогти додати функції до своїх програм або вирішити конкретну проблему.

3 ПОРІВНЯННЯ ВИБРАНИХ ФРЕЙМВОРКІВ ДЛЯ РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКІВ ПЕВНОЇ ПРЕДМЕТНОЇ ОБЛАСТІ

3.1 Вибір інструментальних засобів для реалізації вибраних фреймворків

Для реалізації вебзастосунків було вибрано середовище розроблення IDE WebStorm. WebStorm – це інтегроване середовище для розробки на JavaScript та пов’язаних з ним технологіях. Його головною перевагою є зручний і розумний редактор JavaScript, HTML і CSS, який підтримує також інші мови, наприклад TypeScript, CoffeeScript, Dart, Sass і потрібні фреймворки Angular, React і Vue.

WebStorm робить розробку проєкту простою та зручною, забезпечуючи підсвічування та автодоповнення коду, його аналіз по ходу редагування, швидку навігацію та рефакторинг. Він має потужні інструменти налагодження та інтеграції із системами керування версіями (Git, GitHub), розуміє структуру проєкту та код, відстежує помилки за допомогою систем ESLint, JSLint, TSLint та пропонує їх рішення. Вбудовані в IDE інструменти для тестування та роботи з проєктом допомагають у розробці та роблять її зручнішою та продуктивнішою.

Були обрані мови програмування JavaScript та TypeScript, для можливості застосування необхідних фреймворків на цих мовах програмування. Фреймворки надають можливість створити асинхронну функцію, що зможе рахувати в мілісекундах виконання коду, що дозволяє не використовувати стороні застосунки для тестування швидкості вебзастосунків.

Швидкість застосунків було перевірено та виміряно в браузері Google Chrome, який працює локально в операційній системі Ubuntu 22.04, встановленій на ноутбуці HP ProBook 450 G5. Ноутбук має процесор i5 1,6 ГГц і 16 ГБ оперативної пам’яті (рис. 3.1).

Рисунок 3.1 – Технічні властивості комп’ютера для тесту програм

3.2 Етапи програмної реалізації вибраних фреймворків для розроблення вебзастосунків

Програми тестувалися локально на ноутбуці, щоб уникнути будь-яких аномалій, які могли бути спричинені швидкістю Інтернету. Загалом було проведено три перевірки. Кожен тест вимірював час завантаження домашньої сторінки та час, необхідний для виконання функцій, викликаних кнопками. Кнопки «Створити 1000» і «Створити 10 000» було натиснуто двічі, щоб побачити, чи є якісь відмінності під час повторного запуску тієї самої функції. Перед натисканням кнопки «Створити 10 000» сторінка була очищена, натиснувши кнопку «Очистити», щоб на порожній сторінці було створено нові 10 000 рядків. Швидкість вимірювалася в мілісекундах.

Для збору даних про продуктивність фреймворку, в кожен із них вбудовано невелику тестову програму для перевірки швидкості завантаження. Усі тестові програми мають однакову бізнес-логіку та візуальний дизайн.

Тестові програми мають таблицю даних і меню з п’ятьма функціями:

– створити таблицю з 1000 рядків;

– створити таблицю з 10 000 рядків;

– додати 1000 рядків до таблиці;

– додати 10 000 рядків до таблиці;

– видалити всі рядки.

Рядки, які додані до таблиці, складаються з двох стовпців: 1 містить номер рядка, а інший складається з рядка з 3 випадкових слів (рис. 3.2).

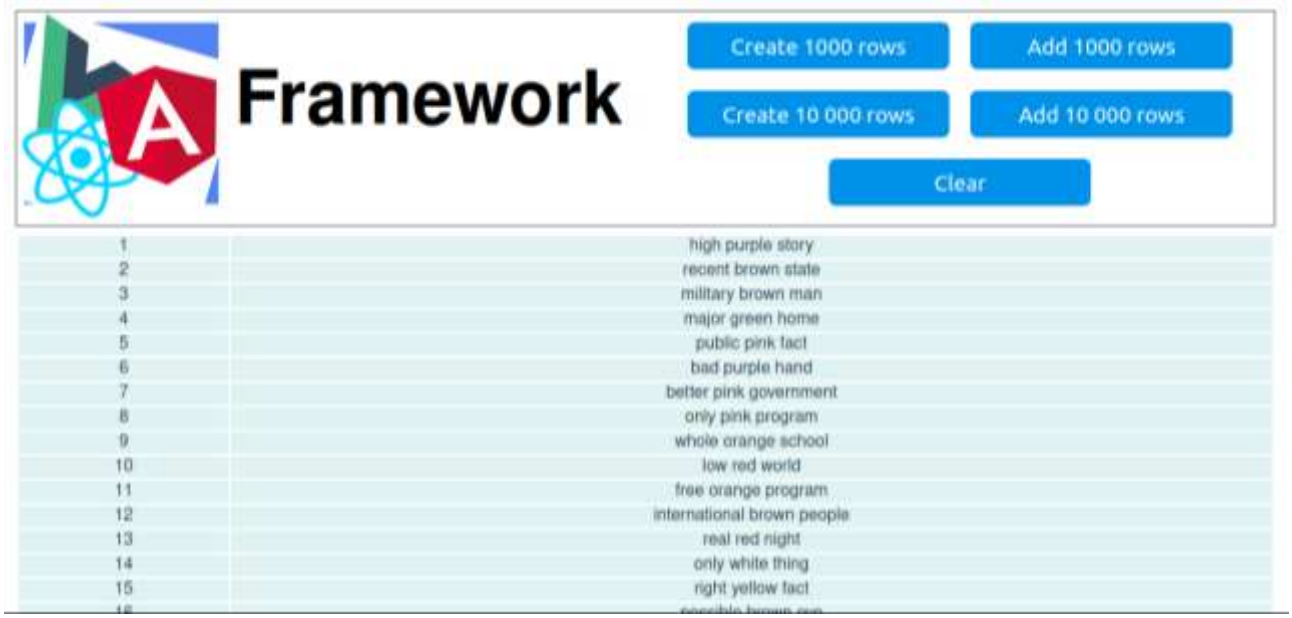

Рисунок 3.2 – Дизайн і структура програми

Програми поділяються на три окремі компоненти, причому App є батьківським компонентом, який має двох дочірніх компонентів (рис. 3.3).

Усі три програми використовують однаковий файл CSS для стилізації та однакову функцію JavaScript для генерації даних. Дані для рядків таблиці створюються функцією buildData в окремому файлі та імпортуються в проєкти як залежність (рис. 3.4).

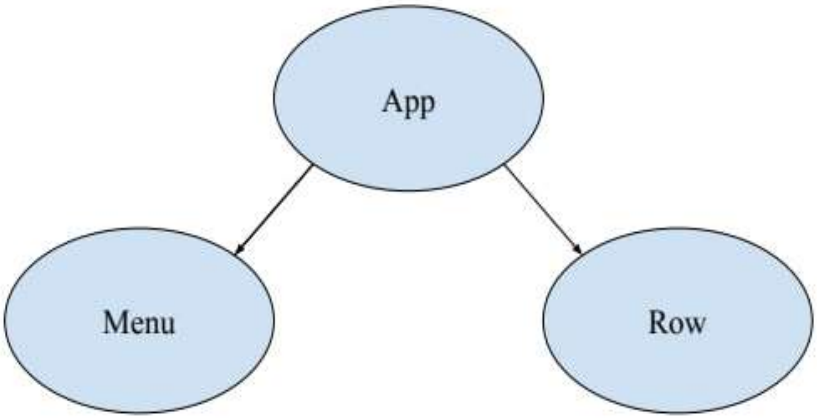

Рисунок 3.3 – Дерево компонентів програми

```
function random(max) {
    return Math.round( * Math.random() * 1000) % max;
}

const A = [
  "bad", "best", "better", "big", "certain", "clear", "different", "early", "easy", "economic", "federal",
  "free", "full", "good", "great", "hard", "high", "human", "important", "international", "large", "late",
  "little", "local", "long", "low", "major", "military", "national", "new", "old", "only", "other", "political",
  "possible", "public", "real", "recent", "right", "small", "social", "special",
  "strong", "sure", "true", "white", "whole", "young", "crazy", "helpful", "mushy"
];
const B = [
  "red", "yellow", "blue", "green", "pink", "brown", "purple", "brown", "white", "black", "orange"
];
const C = [
  "area", "book", "business", "case", "child", "company", "country", "day", "eye", "fact", "family", "government",
  "group", "hand", "home", "job", "life", "lot", "man", "money", "month", "mother", "Mr", "night", "number",
  "part", "people", "place", "point", "problem", "program", "question", "right", "room", "school", "state", "story",
  "student", "study", "system", "thing", "time", "water", "way", "week", "woman", "word", "work", "world", "year"
];


const buildData = function (count) {
  let nextId = 1;
  const data = new Array(count);
  for (let i = 0; i < count; i++) {
    data[i] = {
      id: nextId++,
      label: `${A[random(A.length)]} ${B[random(B.length)]} ${C[random(C.length)]}`,
    };
  }
  return data;
}

export default buildData;
```

Рисунок 3.4 – Функція генерації слів buildData

Функція buildData приймає довжину рядка як параметр під назвою count і створює масив із такою довжиною. Потім викликається цикл for для передачі об’єкта для кожної позиції в масиві.

Кожен об’єкт складається з двох ключів: id і label. Id це ціле число, яке збільшується з кожною ітерацією циклу for. Label – це рядок із трьох слів, створений шляхом виклику випадкового числа та використання його як індексу для вибору слів із масивів з іменами A, B та C.

Робоча збірка програми React становить 585 КБ. Розробка вебзастосунку на фреймворку React починається з батьківського компонента App. Файл App.js починається з імпорту залежностей, таких як клас компонентів із бібліотеки React, файл CSS і дочірні компоненти.

На рисунку 3.5 показано імпорт компонентів програми.

```
import React, { Component } from 'react';
import './App.css';
import Menu from './components/Menu';
import Row from './components/Row';
import buildData from './dummyData';
```

Рисунок 3.5 – Імпорт необхідних залежностей

Після імпорту компонент App оголошується як клас JavaScript і розширює бібліотеку React. Компоненти React можна розділити на дві частини: логіку компонента та вигляд компонента. App є компонентом із збереженням стану та починається з оголошення стану, за яким слідують функції, які складають логіку компонента (рис. 3.6).

Render – це функція візуалізації, яка повертає відображення компонентів програми. Перегляд компонентів програми складається з div, що складається з компонента меню та таблиці HTML. HTML-таблиця містить компонент Row, який відображається в ній на основі кількості, зазначеної в стані програми. Стан програми можна змінити за допомогою функцій, оголошених у першій половині компонента App. Ці функції передаються як реквізити компоненту Menu. Хоча дані передаються лише зверху вниз і не можуть бути передані від дочірніх до батьківських компонентів, все одно можна оновити стан компонента App з компонента Menu.

```
class App extends Component {

  state = {
    data:[],
  }

  create = (amount) => {
    this.setState( state: { data: buildData(amount) });
  }

  add = (amount) => {
    this.setState( state: { data: this.state.data.concat(buildData(amount)) });
  }

  remove = () => {
    this.setState( state: { data: [] });
  }
```

Рисунок 3.6 – Функції компонента App

Лише виклик функції передається від батьківського до дочірнього компонента. Функції викликаються в компоненті Menu, але оголошуються в компоненті App і використовуватимуть компонент App як середовище виконання, змінюючи стан компонентів App (рис. 3.7).

Компонент Menu є компонентом без стану, але він оголошений як клас, щоб проілюструвати можливості. Завдяки ньому вебзастосунок відображує кнопки та логотип програми. Спочатку йде іморт необхідних залежностей, а саме файлу зі стилізацією та логотипом вебзастосунку. Потім викликається функція Render задля відображення меню вебзастосунку (рис. 3.8).

Компонент Row – це функціональний компонент без стану, який приймає властивості як вхідні дані та повертає представлення. Немає потреби в компоненті класу, якщо компонент не має стану. Єдине завдання цього компоненту це відображення одного рядка з номером та трьома випадковими словами (рис. 3.9).

```
render() {
  return (
    <div className="App">
      <Menu
        create={this.create}
        add={this.add}
        remove={this.remove}
      />

      <table className="data-table"><tbody>
        {this.state.data.map((item, i :number ) => (
          <Row key={i} item={item} ></Row>
        ))}
      </tbody></table>

    </div>
  );
 }
}
```

Рисунок 3.7 – Функція відображення компоненту

```
import React, { Component } from 'react';
import '../App.css';
import logo from '../logo.png'

class Menu extends Component{
  render(){
    const { create, add,  remove } = this.props;
    return(
      <div className="menu-container" >

        <img src={logo} alt={'logo'} />
        <div className="framework" >
          <h1> Framework </h1>
        </div>

        <div className="buttons-container">
          <button id="create1000" className="Btn" onClick={() => create(1000) } >Create 1000 rows</button>
          <button id="add1000" className="Btn" onClick={() => {add(1000)}} >Add 1000 rows</button>
          <button id="create1000" className="Btn" onClick={() => {create(10000)}} >Create 10 000 rows</button>
          <button id="add10000" className="Btn" onClick={() => {add(10000)}} >Add 10 000 rows</button>
          <button id="delete" className="Btn" onClick={remove} >Clear</button>
        </div>
      </div>
    );
  }
}
export default Menu;
```

Рисунок 3.8 – Компонент Menu

```
import React, { Component } from 'react';

class Row extends Component {

  render() {
    let { item } = this.props;
    return (
      <tr className="data-row ">
        <td className="small-col" >{item.id}</td>
        <td className="big-col" >{item.label}</td>
      </tr>);
  }
}
export default Row;
```

Рисунок 3.9 – Компонент Row

Програма Vue складається також з трьох компонентів. App є основним і кореневим компонентом, що має два дочірніх компоненти: Menu та Row. Виробнича збірка застосунку Vue складає 610 КБ.

Компонент App починається з частини шаблону перегляду, як показано на рисунку 3.10. Шаблон складається з App div, що містить компонент Menu та таблицю, яка отримує компоненти Row, які відображаються в ньому на основі властивості даних компонента App.

```
<template>
  <div id="app">
    <Menu v-bind="{createRows, addRows, removeRows}" />
    <table class="data-table">
      <tbody>
        <Row v-for="item in data"
        :rowId="item.id"
        :rowLabel="item.label"
        class="data-row" />
      </tbody>
    </table>
  </div>
</template>
```

Рисунок 3.10 – Відображення вебзастосунку у компоненті App

Друга половина компонента App складається з тегу сценарію, що складається з JavaScript, який утворює екземпляр Vue і його властивості. Він починається з імпорту дочірніх компонентів і функції buildData. Потім компонент App експортується як об’єкт, що складається з функції даних, об’єкта компонентів і об’єкта методів (рис. 3.11).

```
<script>
import Menu from './components/Menu.vue'
import Row from './components/Row.vue'
import buildData from './dummyData'

export default {
  name: 'app',
  data() {
    return {
      numberOfRows: 0,
      data: [],
    }
  },
  components: {
    Menu,
    Row
  },
  methods: {
    addRows( amount) {
      this.numberOfRows = this.numberOfRows + amount;
      let data = this.data;
      this.data = data.concat(buildData(amount))
    },
    createRows( amount) {
      this.numberOfRows = amount;
      this.data = buildData(this.numberOfRows);
    },
    removeRows() {
      this.numberOfRows = 0;
      this.data = [];
    },
  },
}
</script>
```

Рисунок 3.11 – Імпорти на функції компоненту App

Подібно до компонента App, компонент Menu складається з двох частин: частини шаблону, що складається з HTML, і тегу сценарію, що складається з JavaScript (рис. 3.12).

```
<template>
  <div>
    <div id="menu" class="menu-container" >

      <img alt="Logo" src="../assets/logo.png">
      <div class="framework" >
        <h1> Framework </h1>
      </div>

      <div class="buttons-container">
        <button class="Btn" @click="createRows(1000)"> Create 1000 rows </button>
        <button class="Btn" @click="addRows(1000)"> Add 1000 rows </button>
        <button class="Btn" @click="createRows(10000)"> Create 10 000 rows </button>
        <button class="Btn" @click="addRows(10000)"> Add 10 000 rows </button>
        <button class="Btn" @click="removeRows()"> Clear </button>
      </div>
    </div>
  </div>
</template>

<script>
  export default  {
    props: ['createRows', 'addRows', 'removeRows'],
  }
</script>
```

Рисунок 3.12 – Компонент Menu

Компонент Row не має жодної логіки програми, тому він написаний як функціональний компонент, який лише відображає строку. Це робить код набагато коротшим і чіткішим, як це можна побачити на рисунку 3.13.

Робоча збірка програми Angular становить 15,7 МБ. Компоненти програми зберігаються в кількох файлах, маючи окремий файл для шаблону HTML, стилю CSS і для JavaScript.

```
<template functional>
  <tr>
    <td class="small-col" slot="rowId" >{{props.rowId}}</td>
    <td class="big-col" slot="rowLabel" >{{props.rowLabel}}</td>
  </tr>
</template>
```

Рисунок 3.13 – Компонент Row

На рисунку 3.14 показано вміст файлу TypeScript, який визначає компонент App.

```
import { Component } from '@angular/core';
import buildData from '../dummyData';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  numberOfRows = 0;
  data: object[];
  add =  0;
  create =  0;

  onAdd($event: number) {
    this.numberOfRows = this.numberOfRows + $event;
    if (this.data === undefined) {
      this.data = buildData($event);
    } else {
      const data = this.data;
      this.data = data.concat(buildData($event));
    }
  }

  onCreate($event: number) {
    this.numberOfRows = $event;
    this.data = buildData(this.numberOfRows);
  }

  onRemove() {
    this.numberOfRows = 0;
    this.data = [];
  }
}
```

Рисунок 3.14 – Декларація компоненту App

Файл починається з імпорту необхідних залежностей, таких як клас Component з ядра Angular і функція для створення тестових даних з dummyData. Далі встановлюються такі властивості компонента, як ідентифікатор компонента та розташування файлів підтримки. Потім компонент App експортується з декількома властивостями даних, які служать станом компонентів, і з кількома методами, які можна викликати для оновлення стану.

На рисунку 3.15 показано шаблон або подання компонентів програми. Шаблон складається з div з ідентифікатором програми, який служить контейнером для двох дочірніх елементів. Перший дочірній елемент – це компонент з назвою app-menu, і він отримує кілька функцій як властивість. Другим елементом є HTML-таблиця, у яку зациклюються компоненти рядка програми за допомогою директиви Angular ngFor.

```
<div id="app">
  <app-menu
    (addEvent)="onAdd($event)"
    (createEvent)="onCreate($event)"
    (removeEvent)="onRemove($event)"
  ></app-menu>

  <table class="data-table">
    <tbody>
      <app-row
        *ngFor="let row of data"
        [id]="row.id"
        [label]="row.label"
      ></app-row>
    </tbody>
  </table>
</div>
```

Рисунок 3.15 – Шаблон компоненту App

На рисунку 3.16 показано логіку компонента меню. Компонент меню починається з необхідних імпортів, таких як клас компонента, вихід і EventEmiter з ядра Angular.

```
import { Component, Output, EventEmitter} from '@angular/core';

@Component({
  selector: 'app-menu',
  templateUrl: './menu.component.html',
  styleUrls: ['./menu.component.css']
})

export class MenuComponent {
  @Output() addEvent = new EventEmitter<number>();
  @Output() createEvent = new EventEmitter<number>();
  @Output() removeEvent = new EventEmitter<number>();

  addRows(val: number) {
    this.addEvent.emit(val);
  }

  createRows(val: number) {
    this.createEvent.emit(val);
  }

  removeRows() {
    this.removeEvent.emit( value: 0);
  }
}
```

Рисунок 3.16 – Логіка компоненту Menu

Після імпорту встановлюються параметри компонентів: шаблон перегляду та його стиль. Далі йде оголошення класу MenuComponent, яке починається зі списку методів, які повертає компонент. Файл закінчується визначенням функцій, які потрібно повернути.

Шаблон компонента Menu складається з кількох елементів дизайну, таких як заголовок, логотип і контейнер для кнопок меню (рис. 3.17). Кожній кнопці призначено метод, визначений користувачем у файлі Menu typescript.

Логічна частина компонента Row починається з деяких імпортів залежностей і продовжується налаштуваннями компонентів, де селектор компонентів оголошується з підтримкою розташування файлів.

```
<div id="menu" class="menu-container" >

  <img alt="Logo" src="../../assets/logo.png">
  <div class="framework" >
    <h1> Framework </h1>
  </div>

  <div class="buttons-container">
    <button class="Btn" (click)="createRows( val: 1000)"> Create 1000 rows </button>
    <button class="Btn" (click)="addRows( val: 1000)"> Add 1000 rows </button>
    <button class="Btn" (click)="createRows( val: 10000)"> Create 10 000 rows </button>
    <button class="Btn" (click)="addRows( val: 10000)"> Add 10 000 rows </button>
    <button class="Btn" (click)="removeRows()"> Clear </button>
  </div>
</div>
```

Рисунок 3.17 – Шаблон компонента Menu

Компонент Row завершується експортом класу RowComponent із двома властивостями вхідних даних: id і label (рис. 3.18).

```
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-row',
  templateUrl: './row.component.html',
  styleUrls: ['./row.component.css']
})

export class RowComponent {
  @Input() id: number;
  @Input() label: string;

}
```

Рисунок 3.18 – Логіка компоненту Row

Шаблон компонентів рядка складається лише з одного елемента рядка таблиці, що складається з 2 стовпців. Стовпцям таблиці призначені класи, а вміст передається в ній виразом Angular за допомогою подвійних фігурних дужок (рис. 3.19).

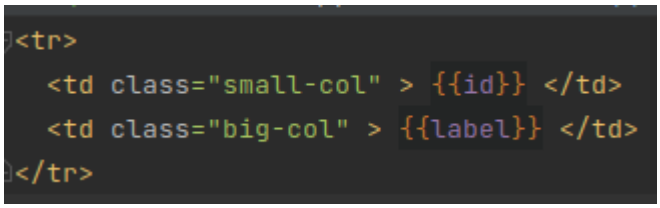

Рисунок 3.19 – Шаблон компонента Row

3.3 Тестування та порівняльний аналіз досліджених фреймворків для розроблення вебзастосунків

Порівнювати програмні інструменти, особливо фреймворки, складно, оскільки вони мають широкий спектр варіантів використання та доменів, одні кращі в одному, а інші – в інших.

У цьому розділі Angular, React і Vue порівнюються за трьома категоріями: популярність серед користувачів, продуктивність фреймворку та швидкість завантаження програми.

3.3.1 Порівняльний аналіз популярності фреймворків

Щоб порівняти популярність, було зібрано та проаналізовано дані з провідних галузевих платформ, таких як GitHub, NPM і Stack Overflow.

GitHub це платформа розробки програмного забезпечення, яку використовують 94 мільйонів розробників для розміщення та керування проєктами з відкритим кодом і корпоративними проєктами на 2022 рік [39].

Код хостингу, написаний багатьма різними розробниками для багатьох різних цілей, робить GitHub унікальним і надійним джерелом для спроб отримати розуміння ринку розробки програмного забезпечення та його тенденцій. Інформація, щодо використання фреймворків Angular, React і Vue на GitHub (табл. 3.1) [40–42].

Таблиця 3.1 – Статистика фреймворків репозиторіїв GitHub

	React	Vue	Angular
GitHub Stars	197,503	200,600	84,745
GitHub Forks	40,968	33,035	22,494
GitHub Issues	845	341	1110

Користувачі GitHub можуть відмічати зірочкою цікаві сховища, що полегшує їх пошук пізніше та сповіщає GitHub ті теми, про які користувач хоче знати.

Сховища GitHub сортируються в пошуку за кількістю зірок (GitHub stars).

Користувачі можуть створювати особисті копії репозиторіїв інших користувачів і вільно вносити в них зміни, не впливаючи на головний проєкт. Копії можна оновити, надіславши запит на злиття до джерела, що робить головний проєкт кращим для всіх.

Кількість копій (GitHub forks), які виходять із сховища, є хорошим показником того, скільки розробників або команд намагаються покращити проєкт на додаток до оригінальних авторів.

GitHub має вбудований засіб відстеження помилок під назвою GitHub Issues. Засіб дозволяє легко відстежувати завдання та ділитися ними з командою розробників.

Кількість проблем може дати певне уявлення про масштаби та складність проєкту, а також про підтримку його спільнотою.

Менеджер пакетів Node є найбільшим реєстром програмного забезпечення у світі. Це дозволяє розробникам з відкритим кодом ділитися та позичати пакети JavaScript. Його також використовують багато приватних організацій для управління внутрішніми проєктами.

Графік завантажень пакетів Angular, React і Vue за останній рік за допомогою NPM (рис. 3.20) [43].

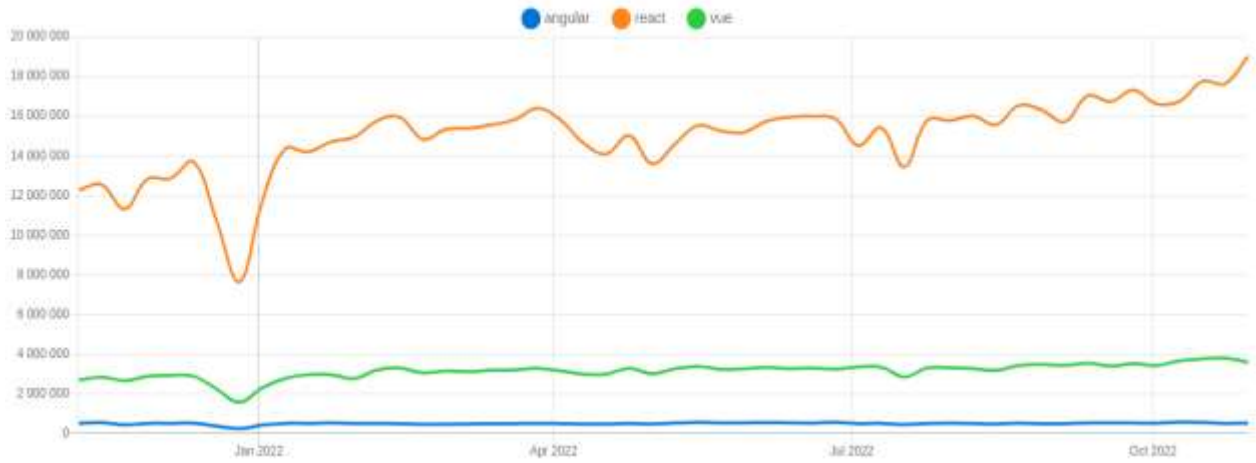

Рисунок 3.20 – Результат порівняння фреймворків
за кількістю завантажень за 2022 рік

На рисунку 3.21 показано відсоток загальної кількості запитань про Angular, React або Vue протягом всього часу у StackOverflow [44].

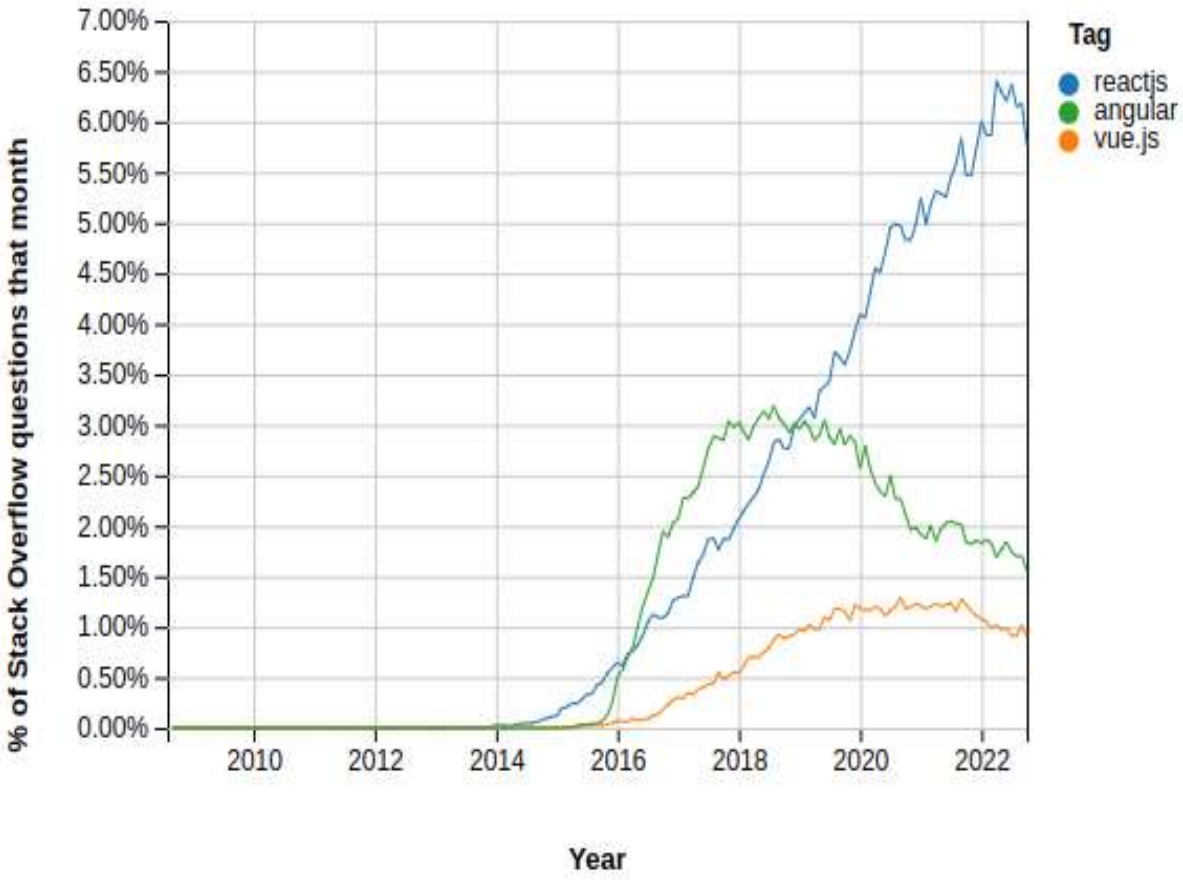

Рисунок 3.21 – Відсоток запитань про Angular, React і Vue у StackOverflow

StackOverflow – це онлайн-платформа, на якій користувачі можуть ставити запитання щодо програмування та відповідати на них.

Відповідно до їх домашньої сторінки [45], понад 560 мільйонів відвідують StackOverflow щомісяця. Ця величезна база розробників і ентузіастів технологій, які обговорюють різноманітні мови програмування, робить її ідеальним місцем для розуміння того, що використовує та про що говорить світова спільнота розробників.

3.3.2 Тестування продуктивності та швидкості програми

Усі фреймворки мають різну побудову й можуть відрізнятися для різних типів і розмірів програм. Тим не менш, невеликі застосунки, створені для цього дослідження, повинні надавати певні вказівки на продуктивність даного фреймворку.

Як видно з прикладів коду, архітектура програми досить схожа для всіх трьох фреймворків, а кількість коду, написаного користувачем, є однаковою. Це може змінитися, якщо масштаб і обсяг застосунків зростатимуть, але для цих типів невеликих застосунків суттєвої різниці немає.

Однак, існує велика різниця в розмірах робочих версій застосунків, як видно з таблиці 3.2. У той час як робочі версії React і Vue були досить схожими за розміром, остаточний додаток Angular був набагато більшим за інші. Розповсюдження Angular становило 15,7 мегабіт, що становить 15 700 кілобайт, що робить його в 25 разів більшим за Vue і приблизно в 27 разів більшим за робочу версію React.

Таблиця 3.2 – Розмір вебзастосунку на різних фреймворках

React	585 KB
Vue	610 KB
Angular	15.7 MB

Це тому, що Angular створено для набагато більших і складніших програм і постачається з більшою кодовою базою. Збірки робочих програм можна координувати між фреймворками під час створення більших програм. Але про це слід пам’ятати, створюючи невеликі програми в регіонах і доменах, де швидкість Інтернету та пропускна здатність мають значення.

Для кожної програми, побудованої у фреймворках, було проведено три тести продуктивності. Середні значення трьох результатів перевірки для кожного фреймворку об’єднані в таблицю 3.3.

У крайньому лівому стовпчику вказана функція, що вимірювалась. Результати відображаються у стовпцях праворуч у мілісекундах. Стовпці позначені кольором: зелений колір позначає найшвидшу швидкість роботи функції, жовтий – другу за швидкістю, а червоний – найповільнішу. Останні два рядки використовуються для аналізу.

Таблиця 3.3 – Швидкість завантаження фреймворків

Функція	**React**	**Vue**	**Angular**
Загрузка сторінки	260	239	397
Створення 1000 рядків	414	232	381
Відновлення 1000 рядків	190	117	326
Додавання 1000 рядків	368	185	254
Створення 10000 рядків	2806	1437	6261
Відновлення 10000 рядків	1173	546	7312
Додавання 10000 рядків	3854	1434	7620
Видалити усі	602	497	2096
Загалом	9667	4687	24647
Множник	2,06	1	5,26

У рядку під назвою «Загалом» обчислюється загальний час, використаний для виконання всіх функцій на фреймворк. Останній рядок, який називається «множник», використовується для ілюстрації загальної швидкості бігу відносно найшвидшого фреймворку (Vue).

Множник бере загальний час, витрачений Vue, і ділить на нього загальний час інших фреймворків, ілюструючи, у скільки разів певний фреймворк повільніший за Vue.

З огляду на те, що Angular має в десятки разів більший розмір файлів, можна з упевненістю припустити, що він буде найповільнішим із трьох, але, дивлячись на таблицю 3.3, можна побачити, що це не завжди так. Angular був найповільнішим під час завантаження сторінки, але різниця у швидкості завантаження була не такою великою, як різниця в розмірі програми. Angular швидше у створенні та додаванні 1000 рядків, ніж React. Хоча Vue мав трохи більший об’єм файлової системи, ніж React, він переміг Angular і React у всіх тестах, у п’ять разів швидше, ніж Angular, і вдвічі швидше, ніж React. React був приблизно в два з половиною рази швидшим, ніж Angular.

3.4 Перспективи подальшої роботи

Перспектива розвитку може бути пов’язана із масштабованістю фреймворків, додання нових бібліотек для облегшення написання функцій та методів та переймання кращих практик з інших фреймворків.

Для відображення повного результату потрібні додаткові дослідження для порівняння фреймворків у створенні великих, складніших багатосторінкових застосунків [46, 47].

Розроблення вебзастосунків йде дуже швидкими темпами і нехтування у виборі фреймворку відповідно до завдання на етапі розробки може призвести затримки у реалізації тієї або іншої частини майбутнього вебзастосунку, адже у кожного фреймворку є свої особливості, які треба знати перед використанням. На етапі використання вебзастосунку вибір непридатного фреймворку може спричинити сповільнений темп загрузки сторінки чи функцій і в подальшому втрати користувачів. Тобто подальше дослідження з розвитку даної теми завжди є та буде актуальною.

ВИСНОВКИ

У даній кваліфікаційній роботі проведено аналіз на порівняння сучасних фреймворків для побудови вебзастосунків. Варто сказати про такий важливий момент, як контекст використання тієї чи іншої технології. Потрібно розуміти, що на даний момент серед розглянутих вище рішень немає універсального, що володіє перевагою в будь-якій ситуації.

Кожен із розглянутих фреймворків має і переваги, і недоліки, немає явного лідера. Вибір того чи іншого інструменту багато в чому залежить від контексту використання. Це означає, що розробник, при виборі фреймворку, повинен якомога детальніше уявляти, що він хоче реалізувати. Чи буде це масштабуватися, підтримуватися тривалий час, який функціонал має мати, коли має бути реалізовано.

Звичайно, особливості та перспективи вебфреймворків відіграють велику роль, але не завжди основоположну.

Практична значущість дослідження полягає у:

– побудові моделі створення вебзастосунку;

– опису вибору фреймворку для конкретних концепцій вебзастосунку;

– опису найкращих практик фреймворків у розробці різноманітних частин вебсторінки.

Результати дослідження апробовано у вигляді тез доповідей під час XXXVII Міжнародної науково-практичної конференції «Сучасні шляхи вирішення новітніх проблем науки» у м. Варна, Болгарія [48].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Al-Dhaifallah M. (2022) Classification of Images Based on a System of Hierarchical Features, *Computers, Materials & Continua*, 72(1), pp. 1785–1797.

2. Tvoroshenko I., and Gorokhovatskyi V. (2022) The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, *International Journal of Engineering and Information Systems*, 6(2), pp. 40–48.

3. Tvoroshenko I., and Zarivchatskyi R. (2020) Analysis of existing methods for searching object in the video stream, *Abstracts of VI International Scientific and Practical Conference «About the problems of science and practice, tasks and ways to solve them» (October 26-30, 2020). Milan, Italy*, pp. 500–505.

4. Творошенко І.С. (2018) Особливості застосування сучасних принципів штучного інтелекту до розробки ефективних механізмів моделювання складних систем, *Science and Technology of the Present Time: Priority Development Directions of Ukraine and Poland*, pp. 118–121.

5. Tvoroshenko I.S. (2004) Structure and functions of intelligent decision-making tools in complex systems, *Artificial Intelligence*, № 4, С. 462–470.

6. Кучеренко Є.І., Творошенко І.С. (2011) Оперативне оцінювання простору станів складних розподілених об’єктів з використанням нечіткої інтервальної логіки, *Штучний інтелект*, № 3, С. 382–387.

7. Кучеренко Е.И., Творошенко И.С. (2003) Процессы принятия решений в сложных системах на основе нечетких интервальных представлений, *Вісник Національного технічного університету «ХПІ». Тематичний випуск: Системний аналіз, управління та інформаційні технології, Х.: НТУ «ХПІ»*, 1(7), С. 79–86.

8. Кучеренко Е.И., Корниловский А.В., Творошенко И.С. (2010) О методах настройки функций принадлежности в нечетких системах, *Системы управления, навигации и связи*, Т. 1, № 13, С. 94–98.

9. Gorokhovatskyi V., Putyatin Y., Gorokhovatskyi O., and Peredrii O. (2018) Quantization of the Space of Structural Image Features as a Way to Increase Recognition Performance, *The Second IEEE International Conference on DataStream Mining & Processing 21-25 August 2018. Lviv, Ukraine*, pp. 464–467.

10. Tvoroshenko I., and Tkachenko D. (2020) Mechanisms of image classification based on descriptors of local features, *Abstracts of IV International Scientific and Practical Conference «Integration of scientific bases into practice» (October 12-16, 2020). Stockholm, Sweden*, pp. 443–448.

11. Гороховатский В.А. (2003) Распознавание изображений в условиях неполной информации. Харків: ХНУРЭ, 112 с.

12. M. Ayaz Ahmad, Irina Tvoroshenko, Jalal Hasan Baker, Liubov Kochura, and Vyacheslav Lyashenko (2020) Interactive Geoinformation Three-Dimensional Model of a Landscape Park Using Geoinformatics Tools, *International Journal on Advanced Science, Engineering and Information Technology*, 10(5), pp. 2005–2013.

13. Творошенко І.С., Табашник В.А. (2018) Розробка просторової моделі геоінформаційної підтримки людей з обмеженими можливостями, що пересуваються на інвалідних колясках, у місті Харків, *Збірник наукових праць Харківського національного університету Повітряних Сил*, 1(55), С. 122–128.

14. Tvoroshenko I., and Dziubenko M. (2020) Modern methods of analysis of the movement scheme using video detection of vehicles, *Abstracts of V International Scientific and Practical Conference «Study of modern problems of civilization» (October 19-23, 2020). Oslo, Norway*, pp. 422–428.

15. Кучеренко Є.І., Творошенко І.С., Анопрієнко Т.В. (2016) Моделювання та оцінювання станів складних об’єктів із застосуванням формальної логіки, *Системи обробки інформації*, № 2, С. 76–82.

16. Tvoroshenko I. (2019) Development of models of spatial analysis of status of interactive processes of complex systems.

17. Gorokhovatsky V. (2014) Structural Analysis and Intellectual Data Processing in Computer Vision. SMIT: Kharkiv, Ukraine, 316 p.

18. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Cluster representation of the structural description of images for effective classification, *Computers, Materials & Continua*, 73(3), pp. 6069–6084.

19. Гороховатський В.О., Творошенко І.С. (2022) Аналіз багатовимірних даних за описом у формі множини компонент: монографія. Харків: ХНУРЕ.

20. Кучеренко Е.И., Творошенко И.С. (2010) Прикладные аспекты моделирования нечетких процессов в сложных системах, *Збірник наукових праць Харківського університету Повітряних сил*, 1(123), С. 127–131.

21. Mayelson de Sousa, Alexandrino Gonçalves (2020) A React.js case study, Institute of Electrical and Electronics Engineers, 38 p.

22. Prateek Rawat, Archana N. Mahajan (2020) ReactJS: A Modern Web Development Framework, *International Journal of Innovative Science and Research Technology.*

23. Arshad Javeed (2019) Performance Optimization Techniques for ReactJS, Institute of Electrical and Electronics Engineers, 164 p.

24. Quang Luong (2019) Web application development with reactjs framework, School of Technology, 61 p.

25. Song Junhui, Zhang Min, Xie Hua (2019) Design and Implementation of a Vue.js-Based College Teaching System, 102 p.

26. Pšenák Peter, Tibenský Matúš (2020) The usage of Vue JS framework for web application creation, *Mesterséges Intelligencia,* 2 (2), pp. 61-72.

27. Tran Nguyen (2020) Applying Vue.js framework in developing web applications, 38 p.

28. Roman Baida, Maksym Andriienko, Małgorzata Plechawska-Wójcik (2020) Perfomance analysis of frameworks Angular and Vue.js, 6 p.

29. Miguel Ramos, Marco Tulio Valente, Ricardo Terra, Gustavo Santos (2016) AngularJS in the wild: a survey with 460 developers, *PLATEAU 2016: Proceedings of the 7th International Workshop on Evaluation and Usability of Programming Languages and Tools*, pp. 9-16.

30. Nina Fat, Marijana Vujovic, Istvan Papp, Sebastian Novak (2016) Comparison of AngularJS framework testing tools, *International Journal of Innovative Science and Research Technology.*

31. Sanja Delcev, Drazen Draskovic (2018) Modern JavaScript frameworks: A Survey Study, *International Journal of Innovative Science and Research Technology.*

32. Andreas Gizas, Sotiris Christodoulou, Theodore Papatheodorou (2012) Comparative evaluation of javascript frameworks, *Companion: Proceedings of the 21st International Conference on World Wide Web*, pp 513-514.

33. Everything You Should Know About React: The Basics You Need to Start Building. URL: https://www.freecodecamp.org/news/everything-you-need-to-know-about-react-eaedf53238c4/ (дата звернення 05.11.2022).

34. Need for Virtual Dom URL: https://jayeshinde.medium.com/need-for-virtual-dom-19ebf6843d95 (дата звернення 05.11.2022).

35. Vue.js Introduction. URL: https://github.com/vuejs/vue (дата звернення 11.11.2022).

36. Why would you need Vuex? URL: https://blog.logrocket.com/do-you-really-need-vuex/ (дата звернення 11.11.2022).

37. Unidirectional data flow with Vuex. URL: https://medium.com/@softwarecf/uni-directional-data-flow-with-vuex-4f31d7ac8c83 (дата звернення 11.11.2022).

38. AngularJS - Quick Guide. What is AngularJS? URL: https://www.tutorialspoint.com/angularjs/angularjs_quick_guide.htm (дата звернення 05.11.2022).

39. State of open source software. URL: https://octoverse.github.com/#sustainable-communities (дата звернення 10.11.2022).

40. Github reposirory of Vue. URL: https://github.com/vuejs/vue (дата звернення 10.11.2022).

41. Github reposirory of React. URL: https://github.com/facebook/react (дата звернення 10.11.2022).

42. Github reposirory of Angular. URL: https://github.com/angular/angular (дата звернення 10.11.2022).

43. NPM Trends, Angular vs React vs Vue. URL: https://npmtrends.com/angular-vs-react-vs-vue (дата звернення 01.11.2022).

44. StackOverflow Update: 560M Pageviews A Month, 25 Servers, And It's All About Performance. URL: http://highscalability.com/blog/2014/7/21/stackoverflow-update-560m-pageviews-a-month-25-servers-and-i.html (дата звернення 10.11.2022).

45. StackOverflow Trends URL: https://insights.stackoverflow.com/trends?tags=reactjs%2Cvue.js%2Cangular (дата звернення 10.11.2022).

46. Гороховатський В., Творошенко І., Сидоренко Д. (2021) Класифікація зображень із використанням кластерного подання. Міжнародний науковий симпозіум «Інтелектуальні рішення-С». Обчислювальний інтелект (результати, проблеми, перспективи). Теорія прийняття рішень: праці міжн. наук. симпозіуму (Вересень 29, 2021). Київ – Ужгород, С. 44-45.

47. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень. *Сучасні інформаційні системи*, 6 (3), С. 5–12.

48. Свічкарьов О. (2022) Аналіз сучасних фреймворків для розроблення вебзастосунків, *Abstracts of XXXVII International Scientific and Practical Conference «Modern ways of solving the latest problems in science» (September 20 – 23, 2022). Varna, Bulgaria*, pp. 477-480.